

2011-01-01

Algorithms for Training Large-Scale Linear Programming Support Vector Regression and Classification

Pablo Rivas Perea

University of Texas at El Paso, privas@miners.utep.edu

Follow this and additional works at: https://digitalcommons.utep.edu/open_etd



Part of the [Computer Engineering Commons](#), [Computer Sciences Commons](#), and the [Electrical and Electronics Commons](#)

Recommended Citation

Rivas Perea, Pablo, "Algorithms for Training Large-Scale Linear Programming Support Vector Regression and Classification" (2011). *Open Access Theses & Dissertations*. 2568.
https://digitalcommons.utep.edu/open_etd/2568

This is brought to you for free and open access by DigitalCommons@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of DigitalCommons@UTEP. For more information, please contact lweber@utep.edu.

ALGORITHMS FOR TRAINING LARGE-SCALE LINEAR PROGRAMMING
SUPPORT VECTOR REGRESSION AND CLASSIFICATION

PABLO RIVAS PEREA

Department of Electrical and Computer Engineering

APPROVED:

Jose Gerardo Rosiles, Ph.D., Chair

Sergio D. Cabrera, Ph.D.

Wei Qian, Ph.D.

Miguel Arguez, Ph.D.

Patricia Witherspoon, Ph.D.
Dean of the Graduate School

©Copyright

by

Pablo Rivas Perea

2011

to my beautiful wife

NANCY

with tons of love

ALGORITHMS FOR TRAINING LARGE-SCALE LINEAR PROGRAMMING
SUPPORT VECTOR REGRESSION AND CLASSIFICATION

by
PABLO RIVAS PEREA

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2011

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Dr. Gerardo Rosiles of the Electrical and Computer Engineering Department at The University of Texas at El Paso, for his advice, encouragement, enduring patience, and constant support. He always answered my emails, even when it was 3:00 AM; he has so much energy, (*where does he get it?!*). He was always, *always* giving me his time, in spite of anything else that was going on. He modestly co-supported my conference expenses and always tried to get me everything I needed to complete this dissertation. His protective attitude towards me, his apprentice, kept me from losing focus on my dissertation. He never allowed me to panic at any time. I wish all students the honor of working with Dr. Rosiles.

I also wish to thank the other members of my committee, Dr. Sergio D. Cabrera and Dr. Wei Qian from the Electrical and Computer Engineering Department at The University of Texas at El Paso. Their suggestions, comments, and additional guidance were invaluable to the completion of this work. Particularly, Dr. Cabrera co-sponsored some of conferences at which I presented some of my research papers. Nonetheless, Dr. Qian was always willing to collaborate and support my research proposals and research papers. Both have been extremely helpful in many different ways so that I had everything I needed to complete this work.

Dr. Miguel Arguez, from the Department of Mathematical Sciences at The University of Texas at El Paso, was the one who brought the light into the darkness I was walking through. His courses in Numerical Optimization and Interior Point Methods for Linear Programming opened my eyes in a tremendous way. This was very helpful in understanding the magnitude of the contributions of this dissertation.

Thanks to Dr. James C. Tilton, from the Computational and Information Sciences and Technology Office (CISTO) at the NASA Goddard Space Flight Center (GSFC), Excellence in Information Science and Technology Award winner. His direction during my internship at NASA really helped me to forge new skills. I am grateful for his friendship, hospitality, patience, and understanding. It has been the best time of my life.

Dr. Elaine Fredericksen from the English Department at The University of Texas at El Paso, acted many times as my editor, even though it was not part of her job description. But more importantly, she led me to understand so many things about the English language, in such a way that now I feel very confident about writing almost anything in English. I also want to acknowledge the collaboration of my proof-reading group: Mary Reyna and Letty Moreno-Brown. And, of course, the helpful discussions of my content group: Juan L. Rios and Mazin M. Al-Zoubi.

Additionally, I want to thank The University of Texas at El Paso Electrical and Computer Engineering Department professors and staff for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as an electrical and computer engineer. This includes (but certainly is not limited to) the following individuals:

Sukie Quezada-Vazquez and Linda Romero

They helped me big time when I was hopelessly lost and new in town. Their help and availability made the transition from México to the US, much better than I expected. Both know everything about everything in the ECE department; many thanks to both. Sukie, Linda, your efficiency and efficacy is a great inspiration for me.

Lorenzo Solis

He was always there, showing up at the office with his big smile. This man was an inspiration for me because of his responsibility and passion for work. Besides, he was the only one who showed an interest in me for who I am as a human being, and not just to ask me about MATLAB, Linux, DSP, or Programming. We always had very nice conversations: short, funny, encouraging, motivating, and unconditional. Mr. Solis is the best janitor at our university, in my humble opinion.

Nito Gumataotao

Thanks to Nito, “the computer guy” of the Unix Lab, for his patience. I am sorry that I had to be remotely logged-in at 15 computers at the same time in order to finish my experiments on time. Also, my apologies for taking over the Parallel Computing licenses of the ECE department. Anyway, I acknowledge his total support on every little technical detail I needed help with. I recall one time I asked Nito if he could install \LaTeX , Kile, change my computer’s hostname, and upgrade the gcc compiler, and by the time I finished talking, he said, “Done.” In summary, thanks to Nito for his extraterrestrial skills in Unix security and administration.

Thanks to my friend Dr. Juan de Dios Cota (a.k.a. Juanito) for his patience while sharing an office with me. I hope the best for him and his family. I also want to acknowledge Carlos Ramirez’ kindness since he provided the naive interior point solver code that was partially used in this dissertation.

I want to thank my dear wife for putting up with me during the development of this work with continuing, loving support, and without complaint. I do not have the words to express all my feelings here, only that I love you, Nancy!

Many thanks to my parents and my in-laws for their prayers. Their support was felt every day during these long years. I am deeply grateful for that.

Finally, I want to thank Charles and Barbara Horak for always being there when I needed the most. Without their support, I would not have been able to finish my doctoral studies at UTEP, literally speaking. They “adopted” me and my wife for the time we were at El Paso city. They were the closest thing to a family we had here in El Paso. May the Lord bless and keep them.

This work was supported in part by the National Council for Science and Technology (CONACyT), Mexico, under grant 193324 / 303732, by the Texas Instruments Foundation Endowed Scholarship for graduate students, by the IEEE Computational Intelligence travel scholarship, and by the SEP-DGRI complementary scholarship. The partial support of the Electrical and Computer Engineering Department via teaching assistantships is also acknowledged.

NOTE: The first draft of this dissertation was submitted to my Supervising Committee on March, 2011.

Abstract

The main contribution of this dissertation is the development of a method to train a Support Vector Regression (SVR) model for the large-scale case where the number of training samples supersedes the computational resources. The proposed scheme consists of posing the SVR problem entirely as a Linear Programming (LP) problem and on the development of a sequential optimization method based on variables decomposition, constraints decomposition, and the use of primal-dual interior point methods. Experimental results demonstrate that the proposed approach has comparable performance with other SV-based classifiers. Particularly, experiments demonstrate that as the problem size increases, the sparser the solution becomes, and more computational efficiency can be gained in comparison with other methods. To reduce the LP-SVR training time, a method is developed that takes advantage of the fact that the support vectors (SVs) are likely to lie on the convex hull of each class. The algorithm uses the Mahalanobis distance from the class sample mean in order to rank each sample in the training set; then the samples with the largest distances are used as part of the initial working set. Experimental results show a reduction in the total training time as well as a significant decrease in the total iterations percentage. Results also suggest that using the speedup strategy, the SVs are found earlier in the learning process. Also, this research introduces a method to find the set of LP-SVR hyper-parameters; experimental results show that the algorithm provides hyper-parameters that minimize an estimate of the true test generalization error.

Finally, the SVR scheme shows state-of-the-art performance in various applications such as power load prediction forecasting, texture-based image segmentation, and classification of remotely sensed imagery. This demonstrates that the proposed learning scheme and the LP-SVR model are robust and efficient when compared with other methodologies for large-scale problems.

Table of Contents

	Page
Acknowledgements	v
Abstract	x
Table of Contents	xi
List of Tables	xvii
List of Figures	xix
List of Symbols	xxviii
List of Acronyms	xxxii
Chapter	
1 General Introduction	1
1.1 Statistical Pattern Recognition	1
1.2 Support Vector Learning and Kernel Functions	3
1.3 Large-Scale Support Vector Regression Training	5
1.4 Scope of This Dissertation	8
1.4.1 Problems to be Addressed	9
1.4.2 Research Questions	10
1.4.3 Dissertation Statement	10
1.5 Summary of Contributions	10
1.6 Overview of Remaining Chapters	11
2 A Review of Support Vector Regression	13
2.1 Introduction	13
2.2 Support Vector Regression Fundamental Principle	15
2.3 Support Vector Regression Learning Methods	21
2.4 Linear Programming Support Vector Regression	23
2.4.1 SVR Learning	24

2.4.2	ν -LPR	25
2.4.3	LP-SVR with Modified LPC	27
2.4.4	Wavelet Kernel-based LP-SVR	28
2.4.5	ℓ_1 -norm LP-SVR	29
2.5	Current Trends and Open Problems	30
3	Large-Scale Linear Programming Support Vector Regression	32
3.1	Introduction	32
3.2	Large-Scale LP-SVR Formulation	36
3.2.1	Primal, Dual, and KKT Conditions	36
3.2.2	Optimality and Sparseness	39
3.3	LP-SVR Variable Decomposition	42
3.4	LP-SVR Constraints Decomposition	49
3.5	Convergence and Optimality Conditions	53
3.6	Interior Points Convergence and Optimality	56
3.7	Experimental Evaluation of LS LP-SVR	58
3.7.1	Large-Scale Learning	61
3.7.2	Solution Sparseness	66
3.7.3	Computational Concerns	68
3.8	Conclusion	69
3.8.1	Large-Scale Learning	69
3.8.2	Model Performance and Sparseness	70
4	LP-SVR Training Speedup and Hyper-Parameters Estimation	71
4.1	Introduction	71
4.2	Within-Class Distances for Learning Speed Up	72
4.2.1	Within-Class Mahalanobis distance and Class-Convex Hull	75
4.3	Large-Scale LP-SVR Hyper-Parameter Estimation	77
4.4	Model Parameters Selection Criteria: Error Functions	80
4.4.1	Error Functions for Multi-Class and Two-Class Problems	81

4.4.2	Regression Problems	83
4.5	Adaptation of Newton Method	84
4.5.1	Newton Method Adaptation	88
4.5.2	Algorithm Discussion	89
4.5.3	Stopping Criteria	91
4.6	Experimental Results	93
4.6.1	Learning Speedup	93
4.6.2	Model Selection	95
4.7	Computational Concerns	100
4.7.1	Speedup by Sample Selection	100
4.7.2	Model Selection	102
4.8	Conclusion	103
4.8.1	Speedup by Sample Selection	103
4.8.2	Model Selection	104
5	Power Load Prediction	105
5.1	Introduction	105
5.2	Dataset	106
5.3	Training the Regression Models	107
5.3.1	Feed-Forward Neural Network	107
5.3.2	Bagged Regression Trees	108
5.3.3	Large-Scale Support Vector Regression	109
5.3.4	Linear Programming Support Vector Regression	109
5.4	Experimental Results	109
5.4.1	Experiment Design and Procedure	109
5.4.2	Quantitative Results	111
5.5	Conclusion	116
6	Texture Classification with LP-SVR and Biorthogonal Directional Filter Banks	121

6.1	Introduction to Texture Analysis	121
6.2	The Bamberger Directional Filter Bank	123
6.3	Multiresolution Directional Pyramids	127
6.3.1	Simple Lowpass-Highpass decomposition	128
6.3.2	Directional Pyramids	128
6.4	Feature Extraction	133
6.5	Classification Stage	136
6.6	Framework for Texture Segmentation with the DFB	136
6.6.1	Proposed LP-SVR Architectures	143
6.7	Evaluation of the Undecimated DP and LP-SVR for Texture Segmen- tation	146
6.8	Conclusions on Texture Segmentation	156
7	LP-SVR-Based Dust Storm Detection	157
7.1	Introduction	158
7.2	Feature Extraction	159
7.3	Maximum <i>a Posteriori</i> Detection	162
7.4	Probabilistic Neural Network Detection	164
7.5	Feed-Forward Back-Propagation Neural Network Detection	166
7.6	Linear Programming Support Vector Regression Detection	168
7.7	Results	168
7.7.1	Comparison between Methods	169
7.7.2	Dust over Land	171
7.7.3	Dust over Ocean	172
7.7.4	Extension to Segmentation and Classification	173
7.8	Conclusion	174
8	Conclusions	181
8.1	Contributions	181
8.1.1	Large Scale LP-SVR	181

8.1.2	LP-SVR Training Speedup and Hyper-Parameters Estimation	182
8.1.3	Applications	183
8.2	Future Work	186
8.2.1	Theoretical	186
8.2.2	Experimental	188
8.3	Publications	189
8.3.1	Theoretical Development	189
8.3.2	Applications	190
	References	191
Appendix		
A	Background in Support Vector Machines	218
A.1	Support Vector Machines Fundamental Principle	218
A.1.1	Soft-Margin Support Vector Machines	221
A.1.2	Kernel Expansion and Kernel Functions	223
A.1.3	Toy Example: XOR Problem	226
A.2	Summary of Support Vector Machines Learning Methods	230
A.2.1	Osuna's Decomposition Algorithm	230
A.2.2	Joachims' SVM ^{Light}	234
A.2.3	Platt's Sequential Minimal Optimization	237
A.2.4	Rifkin's SVMFu	239
A.2.5	Nystrom Method for Kernel Approximation	241
A.2.6	Hush's QP SVM Dual to Primal Mapping	242
A.2.7	Sra's LP to QP SVM Mapping	244
A.3	Linear Programming Support Vector Machines	245
B	Algorithms and Theorems Proofs	248
B.1	Algorithms	248
B.2	Theorems and Proofs	249
B.2.1	Theorem on the ESV Bound	249

B.2.2	Theorem on The Rank Bound	252
B.2.3	KKT Conditions of Constraints Decomposition Method	253
C	Interior Point Methods, Datasets, Performance Metrics, and Additional Tables	255
C.1	Primal Dual Interior Point Methods for Linear Programming	255
C.2	Data Sets Used in this Dissertation	258
C.3	Performance Metrics	271
C.3.1	Regression	271
C.3.2	Classification	272
C.4	Additional Tables from Chapter 3	278
D	Background in the Newton Method and in Kernel Functions	287
D.1	Newton Method for Error Function Minimization	287
D.1.1	Newton Method	288
D.1.2	Globalization Strategy	289
D.2	Background in Kernel Functions	291
D.2.1	Mercer's Theorem	291
D.2.2	Reproducing Kernel Hilbert Space	292
D.2.3	Representer Theorem	294
E	Remote Sensing	295
E.1	Non-Linear Enhancement for MODIS True Color Images	295
E.2	List of MODIS Level 1B Data Granules	296
	Curriculum Vitae	297

List of Tables

3.1	Summary of the Dimensions and Properties of the Datasets.	59
3.2	Accuracy	64
3.3	Balanced Error Rate	65
3.4	Mean Absolute Error.	65
3.5	Support Vectors	67
3.6	Sparse Support Vectors	68
4.1	Total Training Time without speedup	96
4.2	Total Training Time with Speedup.	96
4.3	Iterations reduction percent after speedup.	97
4.4	Summary of Behavior.	101
4.5	Summary of Testing Set Generalization Error Metrics and Statistical Properties of Residual Errors.	102
5.1	Variables Used for Prediction	106
5.2	Electricity Load Prediction Errors	111
6.1	Comparison of Segmentation Errors. ULap-UDFB system implemented with $J = 4$ and $L = 12$	148
7.1	Classifiers Performance. The processing time shown here is in milli- seconds.	170
A.1	Summary of XOR problem characterization.	227
C.1	Summary of the Dimensions and Properties of the Datasets.	267
C.2	Illustration of TP, FP, TN, and FN using a confusion matrix.	272

C.3	Illustration of TP, FP, TN, and FN for class 0, using a multi-class confusion matrix.	273
C.4	Illustration of TP, FP, TN, and FN for class 2, using a multi-class confusion matrix.	273
C.5	Classification targets and predicted values. Borrowed from [60]. . . .	276
C.6	Summary of Performance Metrics and Their Interpretation.	277
C.7	False Positives in Proportion to the Dataset Size.	278
C.8	False Negatives in Proportion to the Dataset Size.	279
C.9	True Positives Rate	279
C.10	False Positives Rate	280
C.11	Specificity	280
C.12	Positive Predictive Value	281
C.13	Negative Predictive Value	281
C.14	False Discovery Rate	282
C.15	Mathews Correlation Coefficient	282
C.16	F_1 -Score	283
C.17	Estimate of Scaled Error Rate	283
C.18	Area Under the Receiver Operating Characteristic Curve	284
C.19	Area Under the Receiver Operating Characteristic Curve Convex Hull	284
C.20	Root Mean Squared Error and Normalized Root Mean Squared Error.	285
C.21	Sum of Squared Error and Statistical Metrics	285
C.22	Exact Support Vectors	286
C.23	Saturated Support Vectors	286

List of Figures

1.1	Overview of a general pattern recognition process.	2
1.2	Different separating hyperplanes. The hyperplane denoted as $\mathbf{w}_B^T \mathbf{x}_i + b = 0$ correctly separates the two classes; however, $\mathbf{w}_C^T \mathbf{x}_i + b = 0$ maximizes the separation margin between classes. In contrast, $\mathbf{w}_A^T \mathbf{x}_i + b = 0$ is a poor separating hyperplane.	3
1.3	The hyperplane is defined by $\mathbf{w}^T \mathbf{x}_i + b = 0$, and the support vectors are those that satisfy the condition $\mathbf{w}^T \mathbf{x}_i + b = d_i$	4
1.4	The data points (input space) are projected to a higher dimension (feature space) using a kernel function.	4
2.1	Illustration of the ϵ -insensitive loss function.	16
2.2	Linear regression: illustration of an ϵ -insensitive tube, fitted to the data points shown as circles.	17
2.3	Illustration of an SVM for non-linear regression, showing the regression curve together with the ϵ -insensitive tube.	17
2.4	Depending on the problem size, one may find three different strategies for SVR training.	25
3.1	Variable decomposition strategy from Torii, <i>et al.</i> [187], in which a subset of the variables α_i is considered for the solution of the LP-SVR problem, for all $i \in \mathcal{B}$. The shaded area corresponds to an arbitrary selection of $ \mathcal{B} $ indices.	34

3.2	LP-SVR constraints decomposition strategy. This figure shows an example of the decomposition of the LP constraints. Here the coefficients matrix \mathbf{A} and vector \mathbf{b} are divided in τ blocks. This figure shows the linear programming chunking (LPC) approach by Bradley's, <i>et al.</i> [22,23].	35
3.3	LP-SVR variable decomposition strategy. This illustrates a decomposition of the LP variables vector \mathbf{z} and coefficients matrix \mathbf{A} in which a subset of the variables α_i is considered for the solution of the LP-SVR problem, for all $i \in \mathcal{B}$. The shaded area corresponds to an arbitrary selection of $ \mathcal{B} $ indices that produce a reduced LP: $\mathbf{z}_{\mathcal{B}}$ and $\mathbf{A}_{\mathcal{B}}$	44
3.4	Proposed LP-SVR constraints decomposition strategy. The proposed decomposition strategy exploits the LP-SVR structure to further reduce the problem size. Note that the reduced problem size is inversely proportional to τ . The notation \mathbf{A}_{ij} refers to an element of the matrix \mathbf{A} at the i -row and j -th column.	50
3.5	Behavior of the KKT conditions as the number of iterations progress. The primal, dual, and complementarity condition must converge to zero. The results shown represent the average value over several experiments on arbitrarily three-class non-separable classification problems.	58
3.6	Training Time as a Function of the Problem Size.	62
3.7	Total Number of Iterations as a Function of the Problem Size.	63
3.8	Objective Function as a Function of Iterations Number.	64
3.9	Support Vectors as a Function of Iterations. SPV SSV ESV SV.	66

4.1	Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and $k = 8$ maximum Mahalanobis distance samples. Note that both SVs and Convex Hull match the $k = 8$ maximum Mahalanobis distance samples.	73
4.2	Mahalanobis distance-ranking of class indices using feature vectors in either the input space or the kernel-induced feature space.	76
4.3	Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and k maximum Mahalanobis distance samples. Note that both SVs and Convex Hull match the k maximum Mahalanobis distance samples.	78
4.4	Response of the root mean squared error as a function of $\theta = [C, \sigma]$. Note, how the error surface is non-smooth and has many local minima.	80
4.5	Unit step function approximation. A function $\Psi\{\cdot\}$ was used for convenience in easing computations.	82
4.6	Comparison between a logarithmic grid refinement technique (a)-(c) [6,7] and the proposed adaptation of the newton method.	85
4.6	Continued... comparison between a logarithmic grid refinement technique (a)-(c) [6,7] and the proposed adaptation of the newton method.	86
4.6	Continued... comparison between a logarithmic grid refinement technique (a)-(c) [6,7] and the proposed adaptation of the newton method.	87
4.7	Sample Selection. Support Vectors as a Function of Iterations. SSV, ESV, SPV, and SV.	94
4.8	Support Vectors as a Function of Iterations. SPV SSV ESV SV. . . .	95
4.9	Total Model Selection Time and Number of Iterations as a Function of the Problem Size.	98
4.10	BER and AUC as a Function of Iterations Number.	99

4.11	RMSE, and MAE as a Function of Iterations Number.	100
5.1	Framework to build regression models for power load prediction. Blocks on the left indicate the input variables <i>i.e.</i> , attributes used to build the regression models.	108
5.2	Two-day window of true data compared with predicted for the four different methods (top). Error residuals for the four methods (bottom).	112
5.3	Christmas two-day window of true data compared with predicted for the four different methods (top). Error residuals for the four methods (bottom). Note the high prediction error between 14:00-21:00 Hrs. . .	113
5.4	Error distribution for the LS SVM, BRT, FFNN, and LP-SVR regression methods. The methods with smallest variances are FFNN and LP-SVR.	114
5.5	Absolute error distribution of the LS SVM, BRT, FFNN, and LP-SVR regression methods. The vertical lines indicate the mean absolute error for each of the four methods as reported in Table 5.2.	115
5.6	Average error by hour of day. Note the error proportional difference in early morning hours and afternoon hours.	116
5.7	Average error by day of week. Note the error proportional difference in working and non-working days.	117
5.8	Hourly breakdown of the LP-SVR mean absolute prediction error. Note that the early morning hours have smaller variability.	118
5.9	Daily breakdown of the LP-SVR mean absolute prediction error. Note that Mondays and Fridays have the largest average error.	119
5.10	Monthly breakdown of the LP-SVR mean absolute prediction error. Note that the month of December exhibits the largest average error, and the largest variability.	120
6.1	Classical segmentation system based on multichannel filtering. . . .	123

6.2	Frequency band partitions obtained using Bamberger DFB.	124
6.3	Eight-band DFB implementation with a tree structure with FFBs and backsampling matrices.	125
6.4	Block diagram for the FFB showing the ideal support for the 2-D filters.	126
6.5	Ladder structure implementation of a 2-D two-channel biorthogonal filter bank.	127
6.6	Magnitude Response of the synthesis filters of an eight band biorthog- onal ladder DFB.	129
6.7	Lowpass-highpass analysis structure. The DFB or the UDFB can be used to do a directional decomposition.	130
6.8	Frequency plane partitioning of the 2-D separable DWT.	130
6.9	Frequency plane partitioning of the proposed directional pyramids. . .	131
6.10	Invertible Undecimated Laplacian-UDFB pyramid structure with no decimation of the resolution components.	132
6.11	Test Image for UDBFs.	133
6.12	Laplacian-UDFB pyramid example with $J = 3$ and $N = 4$	134
6.13	Texture mixture number 1: “Nat-5c.” It contains five different texture classes. Mixture taken from [153].	137
6.14	Texture mixture number 2: “Nat-5v,” with five classes [153].	138
6.15	Texture mixture number 3: “Nat-5v2,” with five classes [153].	138
6.16	Texture mixture number 4: “Nat-5v3,” with five classes [153].	138
6.17	Texture mixture number 5: “Nat-5m,” with five classes [153].	139
6.18	Texture mixture number 6: “Nat-16c.” It contains 16 different texture classes. Mixture taken from [153].	139
6.19	Texture mixture number 7: “Nat-16v.” It contains 16 different texture classes. Mixture taken from [153].	140
6.20	Texture mixture number 8: “Nat-10.” It contains 16 different texture classes. Mixtures taken from Randen and Husøy [153].	140

6.21	Texture mixture number 9: “Nat-10v.” It contains 10 different texture classes. Mixtures taken from Randen and Husøy [153].	140
6.22	Texture mixture number 10: “d4d84.” It contains two different texture classes. Mixture taken from [153].	141
6.23	Texture mixture number 11: “d12d17.” It contains two different texture classes. Mixture taken from [153].	141
6.24	Texture mixture number 12: “d5d92.” It contains two different texture classes. Mixture taken from [153].	141
6.25	Basic frequency domain supports for DFB truncated subbands. Ω_1 and Ω_2 correspond to the eight-band DFB case.	142
6.26	LP-SVR architecture that uses all the directional subbands as inputs to produce an output.	144
6.27	LP-SVR architecture that uses a cascade approach receiving as input the current stage UDFBs and feeds the output to the next LP-SVR along with the next stage UDFBs.	145
6.28	LP-SVR architecture that uses an ensemble approach which receives the three UDFB stages at individual LP-SVR classifiers producing outputs connected to a final classifier that gives the segmentation result.	146
6.29	Texture segmentation maps result using ULAP-UDFB and LVQ [165].	152
6.30	Texture segmentation maps result using ULAP-UDFB and LP-SVR with a single architecture.	153
6.31	Texture segmentation maps result using ULAP-UDFB and LP-SVR with a cascaded architecture.	154
6.32	Texture segmentation maps result using ULAP-UDFB and LP-SVR with an ensemble architecture.	155
7.1	True color RGB composite of the southwestern US. Date 04/15/2001. Time 18:05 UTC. Satellite: Terra. Instrument: MODIS.	160

7.2	The hybrid architecture of the Probabilistic Neural Network. Note the probabilistic nature embedded in a neural architecture.	164
7.3	Dust storm over southwestern US. In (a) is shown the true color image and a region of interest in (j). In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 04/10/2001. Time 18:05 UTC. Satellite: Terra. Instrument: MODIS.	176
7.4	Dust storm over the middle east. In (a) is shown the true color image and a region of interest in (j). In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 06/05/2009. Time 07:50 UTC. Satellite: Terra. Instrument: MODIS.	177
7.5	Dust storm over Argentina. In (a) is shown the true color image, and (j) is the region of interest. In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 01/24/2010. Time 14:40 UTC. Satellite: Terra. Instrument: MODIS.	178

7.6	Dust storm over Australia. In (a) is shown the true color image, and (j) is the region of interest. In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR and LP-SVR trained with the complete training set respectively. Date 09/26/2009. Time 00:35 UTC. Satellite: Aqua. Instrument: MODIS.	179
7.7	Dust storm over southwestern US and northwestern Mexico. In (a) is shown the true color image. In (b) the region of the dust storm. In (c) the dust storm probability $p(d_1 \mathbf{x}) \equiv \Omega_1(\mathbf{x})$ using PNN. In (d) the classification/segmentation regions found using $\tau = 0.5$, $\tau = 0.8$, and $\tau = 0.9$. Date 04/06/2001. Time 18:30 UTC. Satellite: Terra. Instrument: MODIS.	180
A.1	Linearly non-separable data points, due to outliers in the data. Here the slack variables $\{\xi_i\}_{i=1}^N$ help in solving the problem in the presence of outliers that otherwise produce an unfeasibility.	222
A.2	Three different views of an arbitrary two-class classification problem.	226
A.3	With a suitable kernel function, the linear-inseparable data (in input space) can be transferred to linear separable in a higher dimensional space (in feature space).	227
C.1	Ripley dataset. Two classes non-separable.	258
C.2	Wine dataset. Two classes non-separable.	259
C.3	ADA dataset. Two classes non-separable.	260
C.4	GINA dataset. Two classes non-separable.	261
C.5	HIVA dataset. Two classes non-separable.	262
C.6	NOVA dataset. Two classes non-separable.	263
C.7	SYLVA dataset. Two classes non-separable.	264

C.8 Iris dataset. Two classes non-separable.	265
C.9 Spiral dataset. Two classes separable but with a highly non-linear decision function.	266
C.10 Synthetic S dataset. Three classes separable.	266
C.11 Synthetic NS dataset. Three classes non-separable.	267
C.12 Sinc function datasets. True function to approximate and the data provided.	268
C.13 MODIS dataset. Two classes non-separable.	269
C.14 Power Load dataset. Regression problem.	269
C.15 Power Load dataset. Sample statistical distribution across dimensions.	270
C.16 An example of the ROC graph, and the corresponding AUC for the example shown in Table	275

List of Symbols

\mathbb{P}	The set of prime numbers
\mathbb{N}	The set of natural numbers
\mathbb{Z}	The set of integers
\mathbb{I}	The set of irrational numbers
\mathbb{Q}	The set of rational numbers
\mathbb{R}	The set of real numbers
\mathbb{C}	The set of complex numbers
\mathbb{R}_+	The set of positive real numbers
γ, γ^*	Lagrange multipliers
α, α^*	Lagrange multipliers
ξ, ξ^*	Lagrange multipliers
\mathbf{w}	Row vector of weights
C	Constant defined by the SVM designer
d_i	Desired output for the i -th sample
q	Number of classes
b	Constant defined for bias
\mathbf{x}_i	The i -th training sample, or i -th realization
N	Number of realizations of \mathbf{x}
i, j	Integers used to refer to samples of N
$Q(\alpha)$	Utility function of the SVM dual
p	Degree of the polynomial kernel
κ_1, κ_2	Parameters of the sigmoidal kernel
$k(\cdot, \cdot)$	Kernel function
ℓ	Real number referring to the type of norm
$\ \cdot\ _1$	Norm 1

$\ \cdot\ _2$	Norm 2
$ \cdot $	Cardinality of, or, the size of
$L_\epsilon(d_i, y)$	The ϵ -insensitive function
ϵ	Error constant provided by SVR's designer
y_i	The estimator/classifier output of i -th sample
β	Row vector of weights in the kernel-induced space
m_1	Size of the kernel-induced space $k(\mathbf{x}_i, \mathbf{x}_j)$
m_0	Size of the input space \mathbf{x}_i
\mathbf{z}	Positive unknowns row vector of LP-SVR primal
\mathbf{c}	Column vector of \mathbf{z} 's coefficients in the obj. func.
\mathbf{A}	Matrix of coefficients for \mathbf{z}
\mathbf{b}	Column vector in the equality constraints
\mathbf{u}	Slack variable in LP-SVR inequality
n	Number of unknowns of \mathbf{z}
\mathbf{p}	Steepest descent direction
$\boldsymbol{\lambda}$	Vector of dual variables of LP-SVR
m	Number of constraints of $\boldsymbol{\lambda}$
\mathbf{s}	Row vector of dual slacks of LP-SVR
max	Maximize
min	Minimize
s.t.	Subject to
$(\cdot)^*$	Some variables with symbol $(*)$ are solutions
$(\cdot)^\star$	Some variables with symbol (\star) are solutions
\mathcal{O}	Denotes algorithms complexity
\mathcal{W}	Denotes a working set
\mathcal{F}	Denotes a fixed set
\mathcal{B}	Denotes the inices of a working set
\mathcal{M}	Denotes the inices of a fixed set

τ	Constant for number of blocks in LPC
\mathcal{X}	Space of the random variable \mathbf{x}
\mathcal{H}	Hilbert space induced by some kernel function
\mathcal{V}_S	Set of Saturated Support Vectors
\mathcal{V}_E	Set of Exact Support Vectors
\mathcal{V}_N	Set of Non-Support Vectors
\mathcal{V}_α	Set of Sparse Vectors
\mathcal{S}	Set of Support Vectors
\mathcal{A}	Set of non-zero coefficients
\mathcal{T}	Training set
\mathcal{D}	Testing set
\mathcal{V}	Validation set
\mathcal{R}	Reduced set of indices
\mathbf{K}	Kernel matrix
B_{\max}	Maximum size of the working set
B_0	Initial size of the working set
t	Number of iterations
t_0	First iteration
$(a \bmod b)$	The modulo operation between a and b
\mathbf{F}	A multidimensional function
\mathbf{J}	A Jacobian matrix
\mathbf{I}	Identity matrix
$\mathbf{0}$	A vector of zeros: $[0, \dots, 0]$
$\mathbf{1}$	A vector of ones: $[1, \dots, 1]$
\mathcal{F}	The feasible set of solutions in IPM
$\Delta[\cdot]$	The step direction in Newton methods
\mathbf{r}	Vector of residuals
$\boldsymbol{\theta}$	Vector of model hyper-parameters

$p(a b)$	Conditional probability of a given b
$\boldsymbol{\mu}$	Vector of expected values
$\boldsymbol{\Sigma}$	Covariance matrix
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian distribution
$\phi(\cdot)$	Kernel basis function
$\mathcal{E}[\cdot]$	Expected value
σ	RBF kernel parameter
$\boldsymbol{\varepsilon}$	Vector of termination measures

List of Acronyms

ADA Dataset: Adult database

AOD Aerosol optical density

AUC Area under the ROC curve

BER Balanced error rate

BRT Bagged Regression Tree

CCH Class convex hull

CMA Covariance matrix adaptation

DCT Discrete cosine transform

DFB Directional filter bank

DNA Deoxyribonucleic acid

DPM Daily peak MAPE

DWT Discrete wavelet transform

DP Directional pyramids

EA Evolutionary algorithms

ES Evolution strategies

CPU Central processing unit

ESER Estimate of scaled error rate

ESVs Exact Support Vectors

FFB Fan filter bank

FFNN Feed-forward neural network

FN False negative

FP False positive

Gb Giga bytes

GINA Dataset: Handwriting digits database

HIVA Dataset: HIV AIDS infections database

iid Independent identically distributed

IncSVM Incremental SVM

IPM Interior point method

KKT Karush-Kuhn-Tucker

KLT Karhunen-Loève transform

LP Linear programming

LP-SVR Linear programming support vector regression

LPC Linear Programming Chunking

LS LP-SVR Large-Scale LP-SVR

LS SVM Large-Scale SVM

LSSVM Least-Squares SVM

LVQ Learning vector quantization

MAE Mean absolute error

MAP Maximum empha posteriori

MAPE Mean absolute percent error

Mb Mega bytes

MD Mahalanobis distance

MODIS Moderate resolution imaging spectroradiometer

MR Multi-resolution

NE Normalized Error

NEPOOL New England Power Pool

NN Neural networks

NOVA Dataset: Text categorization database

NRMSE Normalized RMSE

NS Non-separable

NSVs Non-Support Vectors

PCA Principal component analysis

PDF Probability density function

PMF Probability mass function

PNN Probabilistic neural network

QP Quadratic programming

RBF Radial basis function

RGB Red green blue

RMSE Root mean squared error

ROC Receiver operating characteristic

SMO Sequential minimal optimization

SPVs Sparse Vectors

SSVs Saturated Support Vectors

SVs Support vectors

SVMs Support vector machines

SVR Support vector regression

SYLVA Dataset: Forest cover database

TDI Thermal dust index

TN True negative

TP True positive

UCI University of California Irvine

UDFB Undecimated DFB

UTC Coordinated Universal Time

WGN White Gaussian noise

Chapter 1

General Introduction

This dissertation focuses on some open issues in Support Vector Regression (SVR). The objective is to contribute and advance the machine learning area within the so called “statistical pattern recognition” community with original ideas to solve common problems. Throughout this document, the reader is driven to an understanding of SVR, the research questions addressed, and their significance. In the following paragraphs a general overview of SVR is given.

1.1 Statistical Pattern Recognition

From the typical pattern recognition process depicted in Figure 1.1, one can see that the range of possible applications is very broad. Statistical pattern recognition methods have focused mostly on the last two blocks: prediction and model selection.

Statistical pattern recognition methods based on optimization methods have shifted the paradigm in the machine learning area. The reason is that most learning algorithms proposed in the past 20 years focused on heuristics or on analogies with natural learning systems, such as evolutionary algorithms (EA) or neural networks (NN). EAs and NNs were mostly the result of biological findings and experimental tuning of parameters. However, the underlying reasons for their performance was not fully understood. Most of the work was based on designing heuristics to avoid local minima trapping in the training (design) process. In contrast, recent statistical pattern recognition algorithms overcome many of these disadvantages by using known theories in the mathematical sciences, *e.g.*, Mercer kernels [129], Hilbert spaces [41],

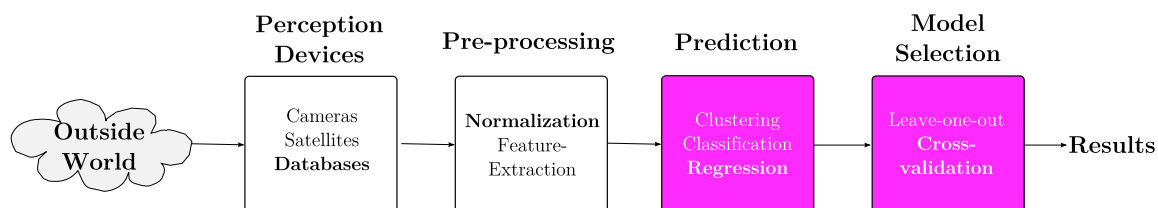


Figure 1.1: Overview of a general pattern recognition process.

and Kernel expansion [171].

In the last decade, a theoretical basis for learning machines has been developed to explicitly maximize their performance [43].

The combination of learning machines with kernel functions has led to novel pattern recognition algorithms characterized by their formulation as convex optimization problems. Convex optimization problems have the advantage of being free from local minima. In particular, it is now possible to model non-linear pattern recognition systems under the convenience of a linear representation using kernel functions [171].

Support vector machines (SVMs) are arguably the best known example among kernel learning algorithms. Since their introduction in 1992 [20], SVMs have been studied, generalized, and applied to a number of problems.

Currently, SVMs hold records in performance benchmarks for handwritten digit recognition [109], text categorization [5, 110, 115, 140, 166], information retrieval [19, 58, 61, 182, 204], and time-series prediction [42, 169, 172, 196, 220] and are commonly used in the analysis of DNA micro-array data [34, 70, 145, 146, 192]. Due to its novelty and potential, there are many areas of improvement, particularly in the numerical methods for problem posing [168] (*e.g.*, vectorial calculus and algebra) and optimization [38, 214] (*e.g.*, quadratic and linear programming). The following section explains the general concept of support vector learning and the clever idea of using kernel functions for non-linear problems.

1.2 Support Vector Learning and Kernel Functions

SVMs are a set of two-class supervised learning methods that are used for classification [197, 198]. The main idea of an SVM is to construct a hyperplane as the decision surface, based on data points called support vectors: this the reason for the name. SVMs aim to find the separation margin that maximizes its distance to positive and negative examples (shown as black and white dots in Figure 1.2 respectively). Figure 1.2 shows three different separating hyperplanes (or lines for this case), $\mathbf{w}_A^T \mathbf{x}_i + b = 0$ does not accurately separates the two groups of examples, in contrast $\mathbf{w}_B^T \mathbf{x}_i + b = 0$ and $\mathbf{w}_C^T \mathbf{x}_i + b = 0$ do. However, the plane $\mathbf{w}_C^T \mathbf{x}_i + b = 0$ clearly maximizes the margin between the two groups of examples, which in turn maximizes *generalization*, *i.e.* the ability of the algorithm to learn the true model from the training set.

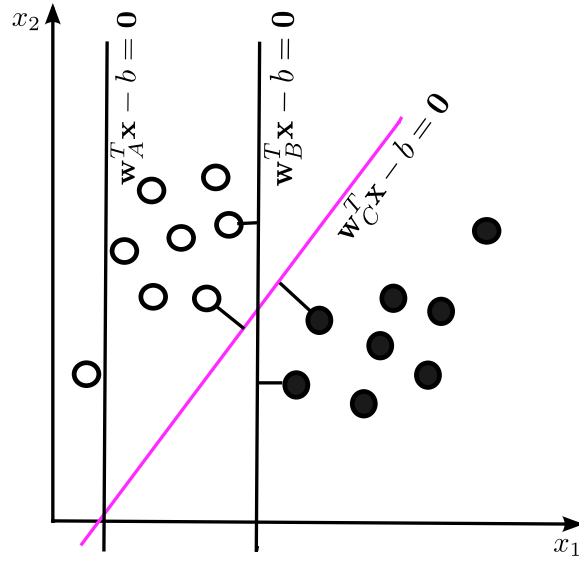


Figure 1.2: Different separating hyperplanes. The hyperplane denoted as $\mathbf{w}_B^T \mathbf{x}_i + b = 0$ correctly separates the two classes; however, $\mathbf{w}_C^T \mathbf{x}_i + b = 0$ maximizes the separation margin between classes. In contrast, $\mathbf{w}_A^T \mathbf{x}_i + b = 0$ is a poor separating hyperplane.

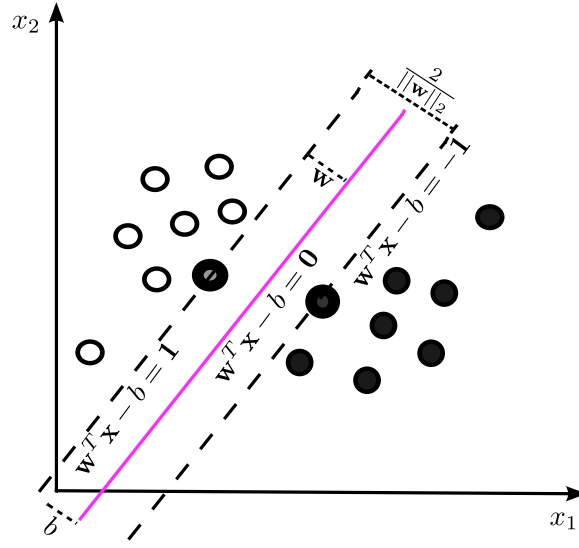


Figure 1.3: The hyperplane is defined by $\mathbf{w}^T \mathbf{x}_i + b = 0$, and the support vectors are those that satisfy the condition $\mathbf{w}^T \mathbf{x}_i + b = d_i$.

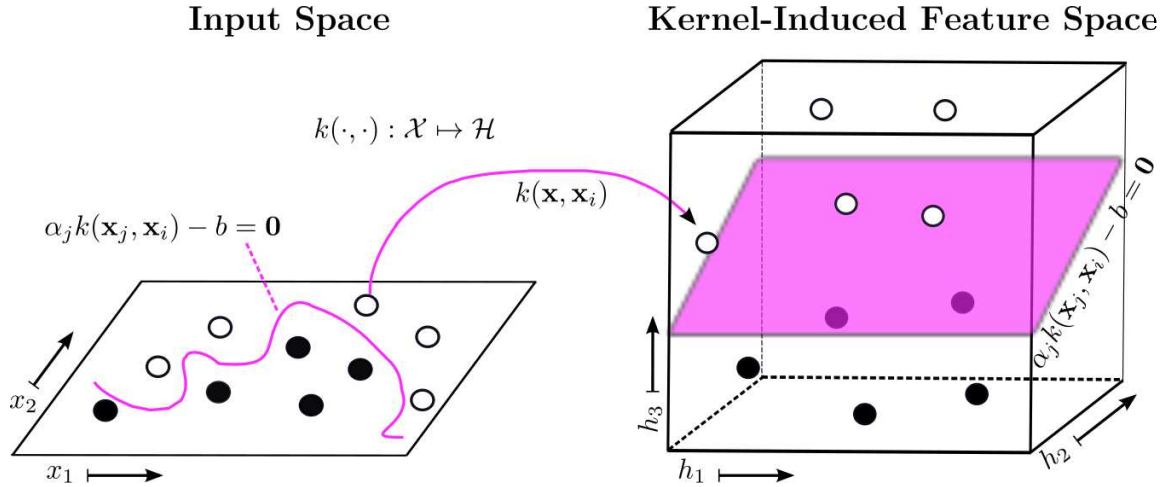


Figure 1.4: The data points (input space) are projected to a higher dimension (feature space) using a kernel function.

Each instance in the training set contains one “target output” (class labels) d , and an input feature vector \mathbf{x} . The goal of SVM is to produce a model that predicts target values of unlabeled data instances, given only their features.

Formally, the SVM principle for the linear case (Figure 1.2 - Figure 1.3) is to find the \mathbf{w} 's and b that form the hyperplane denoted as $\mathbf{w}^T \mathbf{x}_i + b = 0$ between any given data-points such that the data-points are separated with maximum margin. For the non-linear case, the aim is to find the same hyperplane that separates the two groups in a higher dimension space induced by a particular kernel function $k(\cdot, \cdot)$ that maps \mathbf{x} into $k(\mathbf{x}_i, \mathbf{x})$ as illustrated in Figure 1.4). The figure portraits the case of two groups of data that are non-linearly separable in the input space, which becomes linearly separable in the kernel-induced feature space. Therefore, both SVM and kernel functions together represented a tremendous breakthrough for supervised learning classifiers since the appearance of neural networks appearance.

In 1995, Vapnik, *et al.* [197, 198] and later Smola, *et al.* [54, 178] explored and developed the SVM approach for regression problems. This approach is commonly known as Support Vector Regression (SVR), which increased the application range of SVMs since SVRs can also perform multi-class pattern recognition [86]. This type of machine is typically formulated using quadratic optimization under the umbrella of convex optimization. In return, the solution is always global and is easy to compute for very small problems.

1.3 Large-Scale Support Vector Regression Training

The original training of an SVM [197, 198] was only valid for small data sets and was designed to find the solution of a QP problem using a constrained conjugate gradient algorithm. The size of the problem is directly proportional to the number of training vectors, making its solution computationally intractable even for moderate size training sets.

Later Vapnik, *et al.* [197, 198] improved this method by considering only those

variables associated to support vectors lying on the boundaries, whose gradient take them out of the feasible region. This modification allowed computational tractability of a limited number of large-scale problems (less than 5,000 samples).

In 1997, Osuna, *et al.* [139] proposed a decomposition method that achieves optimality by solving a sequence of smaller sub-problems from randomly selected samples out of a training dataset (see Appendix A.2.1). The algorithm selects the support vector coefficients that are active on either side of the optimal hyperplane (in a two class problem), then proceeds iteratively until the completion of the problem size. This algorithm performs satisfactory in applications of up to 110,000 data points.

Later in 1999, Joachims introduced the concept of “shrinking” under the SVM training context and named it “SVM^{light}” [98]. The idea is to get rid of points that have less probability of becoming support vectors, thereby saving time in the optimization problem solution (see Appendix A.2.2). Joachims’ method demonstrated to be faster than Osuna’s method. The author reported results for as many as 49,749 data points.

Also in 1999, Platt extended Osuna’s work with an algorithm called “sequential minimal optimization” (SMO) [148]. This is a famous algorithm implemented in several commercial and open-source software applications. The key idea behind Platt’s method is to decompose the large QP problem into a series of very small QP sub-problems. These sub-problems are as small as they can be solved without a QP solver, thus much faster (see Appendix A.2.3). Platt reported as many as 60,000 data points.

Collobert, *et al.* [39] reported in 2001 an adaptation of Joachims’ method for SVR problems named “SVM_{Torch}” (see Section 2.3). The authors show the implementation of reduction algorithms and give mathematical proof of convergence. The authors also reported using as many as 60,000 data points.

Rifkin presented the “SVM_{Fu}” algorithm in 2002 [155]. This work reviews and

synthesizes Osuna’s, Platt’s and Joachims’ work (see Appendix A.2.4). The key idea is to divide the large problem into sub-problems such that their Hessian matrices fit within memory limitations. In QP, Hessian matrices are useful since they describe the second partial derivatives of an n -dimensional function. Rifkin reports as many as 60,000 data points.

Mangasarian, *et al.* in 2002 [127], explored the very first linear programming (LP) approach to kernel-based regression. The authors proposed a linear programming chunking (LPC) algorithm with guaranteed termination (see Algorithm B.2). The method demonstrated efficiency, but slow convergence solving linear programs with commercial optimization software. The authors report solving linear programs with as many as 16,000 data points in a total of 19 days.

In 2005, Drineas, *et al.* [53] developed an algorithm to approximate a low rank kernel matrix in the form of (A.21). The computational time is reduced (see Appendix A.2.5). The authors report an interesting derivation of their ideas based on the Nystron method.

Continuing with QP-based methods, in 2006, Hush, *et al.* [90] proposed an algorithm that produces approximate solutions without compromising accuracy in a QP SVM problem. The authors propose a first stage where the algorithm provides an approximate solution for the QP dual, and a second stage that maps the QP-dual problem to the QP-primal problem, based on the duality theorem. The authors report over a 100,000 samples data-set.

In 2006, Sra [180] proposed an algorithm that explicitly takes advantage of the LP formulation for SVMs to produce an efficient training method for large-scale problems. The method converts the LP problem into a QP problem, applies Bregman [27] and Hildreth [81] decomposition, and then the QP problem is solved (See Appendix A.2.7). The author report solving a problem with as many as 20,000,000 samples.

For some time, no new LP approaches for SVR were developed until 2009, when Lu, *et al.* [123] reported an SVR method based on a novel LP formulation. The

authors present an alternative of the typical kernels and use wavelet-based kernels instead. The paper shows an application to nonlinear dynamical systems identification, but the authors do not elaborate on the data-set size.

In 2010, by the time this dissertation was already in progress, Torii, *et al.* [187] published work on a novel LP formulation along with three decomposition methods for training an LP-SVM approach. The authors report as many as 60,000 data points.

The development found up to date does not report any advances of sub-problem-based SVR approaches entirely motivated and aimed for large-scale LP-SVR formulations since Mangasarian, *et al.* [127]. Even the idea from Sra [180] that seems to work for up to 20,000,000 samples, fails to take advantage of the efficiency of LP solvers and still works for only a limited number of samples.

In spite of all these advances in SVMs and SVR, training schemes for large-scale applications are still required. It is of high importance to address the problem of large-scale training as technology advances and large amounts of data is being stored for further analysis. For example, consider a typical case in engineering when one has a five-minute remotely sensed multispectral multidimensional image of 36 bands of 2030×1053 pixels. For one single five-minute measurement, we have a sample space of 2,137,590 pixels at 36 variables occupying 587Mb of memory (in double precision format). Now consider having at least 50 observations; this accounts for 106,879,500 pixels and about 28Gb of memory. This represents a dramatic quantity that clearly illustrates the complexity of the problem addressed in this dissertation.

1.4 Scope of This Dissertation

The statistical learning theory developed by Vapnik, *et al.* [197,198] led to the SVMs, derived from the concept of *structural risk minimization* instead of *empirical risk minimization*. The training of an SVM is equivalent to solving a Quadratic Program

(QP) with linear constraints [139]. The number of data points is equivalent to the number of variables for the QP problem. The problem arises as the training data set size grows to thousands, as is typical of real-life applications.

A number of training methods have been proposed [53,90,98,139,148,155]. Many of these algorithms assume that the following conditions are satisfied: (i) the number relevant data points required to build the pattern recognition model (*i.e.* the number of support vectors) is manageable; and (ii) the total number of support vectors is smaller than the number of training data points. However, these conditions do not hold in most engineering applications for a large number of reasons.

This research proposes a training algorithm that is not based on the previous assumptions and that is more efficient. This is achieved by taking advantage of the efficiency of Linear Programming (LP) methods and by posing the problem as an LP problem rather than a QP problem.

1.4.1 Problems to be Addressed

In this dissertation an improved SVR formulation is developed which can be implemented directly with most primal-dual numerical solvers for linear programming (LP). It has been demonstrated that LP is more computationally efficient, faster, and more robust than QP [62,78]. Moreover, recent works [22,123,187] have demonstrated efficient LP formulations for SVMs.

Since in SVMs and SVR the complexity of the pattern recognition problem is proportional to the number and space-dimension of the training samples, this research presents a more efficient and faster training algorithm for SVR able to operate under these conditions. More objectively, this document presents a method for solving the optimization problem in the presence of large training sets.

1.4.2 Research Questions

This dissertation addressed the following research questions:

- Can any real-valued, large-scale training data set in the order of millions of data points be divided in smaller datasets, such that the support vectors of the large-scale problem are found from these small datasets by using a sequential linear programming support vector regression approach?
- Is it possible to use statistical properties of the large-scale data to reduce the sample size without losing the support vectors of the large-scale problem and then let the sequential approach find the solution of LP sub-problems?

1.4.3 Dissertation Statement

Training a Linear Programming Support Vector Machine for Regression (LP-SVR) in the presence of a large number of training samples is very difficult, and computationally expensive with state of the art methods. This research demonstrates that a real-valued large-scale training data set can be decomposed in smaller size data sets, such that the support vectors of the large-scale problem are found sequentially from these small blocks to find the optimal solution to the large-scale LP-SVR problem.

1.5 Summary of Contributions

The development of a training methodology specialized for large-scale data-sets based on linear programming methods for support vector regression problems contributes to the areas of statistical pattern recognition, machine learning, and its derived areas. This research provides a method for training LP-SVR machines specialized for large-scale problems. The key problem formulation for the LP primal optimization problem guarantees convex solutions, reliable models, and efficient computations.

The proposed research performs more efficiently than the SVMTorch method proposed by Collobert, *et al.* [39] and Torii, *et al.* [187]. The latter, to the best of our knowledge, is the only similar idea reported to date. In contrast to Collobert *et al.* this research proposes the usage of pure LP methods. And in contrast to Torii *et al.* this research uses a different LP-based SVR formulation.

The second key contribution is that this dissertation provides numerical evidence of the relationship between the statistical sense of distance and the number of support vectors. Thus, statistical distance-based sample ranking is utilized to accelerate the learning process.

A third contribution is the development of a model selection algorithm for LP-SVR. This approach uses an inexact quasi-Newton approach to find the best set of hyper-parameters.

Since the SVM has quite a large number of applications, these can also be extended to the SVR solution. This document shows applications to i) remote sensing dust storm detection, ii) power load prediction, iii) image texture segmentation, and iv) sonar mine detection.

1.6 Overview of Remaining Chapters

Chapter 2, shows the development of the fundamental concepts behind support vector regression. The chapter also presents a comprehensive literature review on training methods, specifically for large datasets. Then the chapter reviews the linear programming formulations for support vector regression and their training methods. Finally, a brief introduction to primal-dual interior point methods is given as background material; as well as information about the datasets and performance metrics used in this dissertation.

Chapter 3 reveals the proposed large-scale linear programming support vector regression formulation, as well as the proposed learning method. Chapter 3 outlines

the complete learning algorithm and Chapter 4 describes a method to find the hyper-parameters of the proposed LP-SVR formulation. Finally, the chapter explains a method that accelerates convergence of the learning process. The method uses the statistical concept of distance to pre-rank the data set. Experimental results are shown to demonstrate the claims of fast rate of convergence and efficiency.

Applications of the methods explained in Chapter 3 and Chapter 4 are discussed in Chapters 5 through 7.

Chapter 5 addresses an important issue regarding traditional power systems: load forecasting. In this chapter, a new electricity price and load prediction method is introduced. The proposed large-scale LP-SVR approach is applied to the real-life problem in the smart grid. The experimental results show significant improvement in short-term power load forecasting when compared with binary trees and neural network regressors.

Chapter 6 considers the application of a set of directional filter banks (DFBs) to the problem of texture classification. The DFB is used to provide a compact and efficient representation in which fast classification can be performed using the proposed LP-SVR approach. The resulting method is shown to yield higher performance than feature-based techniques reported previously.

Chapter 7 analyzes statistical and neural approaches to alleviate the lack of specialized remote sensing-based dust aerosol detection methods. Chapter 7 highlights the effectiveness of the proposed LP-SVR model in understanding dust storm phenomena.

Finally, Chapter 8 concludes this dissertation with a summary of the findings and relevance of all the analysis mentioned above.

Chapter 2

A Review of Support Vector Regression

This chapter summarizes the theory of support vector regression (SVR). After a brief introduction, notation, and some definitions, the fundamental principle of SVRs and its training is addressed. This leads to a better understanding of the Linear Programming (LP) formulation of SVRs.

If the reader is not familiar with the fundamental principle of Support Vector Machines (SVMs) or SVMs learning methods, it is suggested to consult Appendix A.

2.1 Introduction

In supervised learning methods, one is provided with a dataset that comprises a number of samples from an M -dimensional data vector $\mathbf{x} \in \mathbb{R}^M$, and a desired output class value $d \in \mathbb{Z}$. In the case of regression, the desired output value is $d \in \mathbb{R}$. This dataset is typically divided in three datasets: “training set,” “validation set,” and “testing set”. In this dissertation, the training set is used to train the pattern recognition models, and the validation set is used to adjust parameters during training. Later, the testing set is used to determine the true performance of the trained model. The testing set is not to be used at any stage of the training process.

Let $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ define the training set, where N is the number of samples available for training. Let $\mathcal{V} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_v}$ define the validation set, where N_v is the

number of samples available for validation. Let $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_t}$ denote the testing set, where N_t is the number of samples available for testing. This dissertation focuses on the case when N is very large, with $N \mid N_t \geq N_v$.

In this dissertation the ℓ_1 -norm of the vector \mathbf{x} is denoted as follows:

$$\|\mathbf{x}\|_1 = \sum_{i=1}^M |x_i|, \quad (2.1)$$

and the ℓ_2 -norm (a.k.a. as Euclidean norm) is defined as follows:

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^M x_i^2 \right)^{\frac{1}{2}} \equiv (\mathbf{x}^T \mathbf{x})^{\frac{1}{2}}, \quad (2.2)$$

where \mathbf{x}^T indicates the transpose of the vector \mathbf{x} ; and the operation $\mathbf{x}^T \mathbf{y}$ may be regarded as the “inner product” between vectors \mathbf{x} and \mathbf{y} .

The notation \mathbf{x}_i refers to the i -th vector \mathbf{x} for all $i = \{1, 2, \dots, N\}$; while the notation x_j refers to the j -th component of the vector \mathbf{x} for all $j = \{1, 2, \dots, M\}$. Matrices will appear in capital bold letters. For instance, a matrix \mathbf{X} with i rows and j columns, has elements that will be regarded as \mathbf{X}_{ij} . Matrices or vectors can also be represented with bold Greek symbols as appropriate. Non-bold Greek symbols and italic font will be used for variables; *e.g.*, $\alpha, \beta, \gamma, x, y, z, \Phi, \Sigma, \Omega, P, S, O$. The vector $\mathbf{1}$ is equivalent to a vector of ones with the appropriate size, $\mathbf{1} = [1, 1, \dots, 1]^T$. Similarly, the vector $\mathbf{0}$ is denoted as $\mathbf{0} = [0, 0, \dots, 0]^T$.

The sets (*i.e.*, sets of data, sets of variables, sets of sets, and sets of indices) will be regarded with calligraphic font, *e.g.*, $\mathcal{A}, \mathcal{B}, \mathcal{C}$. The sets of numbers will be regarded with double-barred font, *e.g.*, $\mathbb{D}, \mathbb{E}, \mathbb{F}$.

For probability, the notation of Duda, *et al.* [77] will be followed. Let x be a discrete random variable. Then, x can take on any value in the set $\mathcal{X} = \{v_1, v_2, \dots, v_m\}$. The probability of x being equal to some value v_i will be denoted as $P(v_i) \equiv \Pr[x = v_i]$. Since x is discrete, its probability mass function (PMF) will be regarded as $P(x)$;

if x is continuous, its probability density function (PDF) has the notation $p(x)$. The expected value of x will be denoted as $\mathcal{E}[x] \equiv \mu$; while the variance is denoted as $\text{Var}[x] \equiv \sigma^2 = \mathcal{E}[(x - \mu)^2]$.

With this notation in mind, let us start with the fundamentals of support vector regression.

2.2 Support Vector Regression Fundamental Principle

To begin with, let us consider the linear regression case where the dependency of a scalar observable d on a regressor \mathbf{x} is denoted as follows:

$$d = \mathbf{w}^T \mathbf{x} + b, \quad (2.3)$$

where the parameter vector \mathbf{w} and the bias b are the unknowns. The problem is to estimate \mathbf{w} and b given the training samples $\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N$ where the vector elements \mathbf{x}_i are assumed to be statistically independent and identically distributed (iid). The problem formulated by Vapnik is aimed to minimize, on the variables \mathbf{w} and b , the *structural risk functional*

$$R = \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N |d_i - y_i|_\epsilon \quad (2.4)$$

where the variable y_i is the estimator output produced in response to the input \mathbf{x}_i , that is $f(\mathbf{x}_i) \equiv \mathbf{w}^T \mathbf{x}_i + b = y_i$; the function $|\cdot|_\epsilon$ describes the ϵ -insensitive loss function:

$$L_\epsilon(d, f(\mathbf{x})) \equiv |d - f(\mathbf{x})|_\epsilon = \begin{cases} |d - f(\mathbf{x})| - \epsilon & \text{for } |d - f(\mathbf{x})| \geq \epsilon \\ 0 & \text{otherwise,} \end{cases} \quad (2.5)$$

where ϵ is a prescribed parameter (see Figure 2.1).

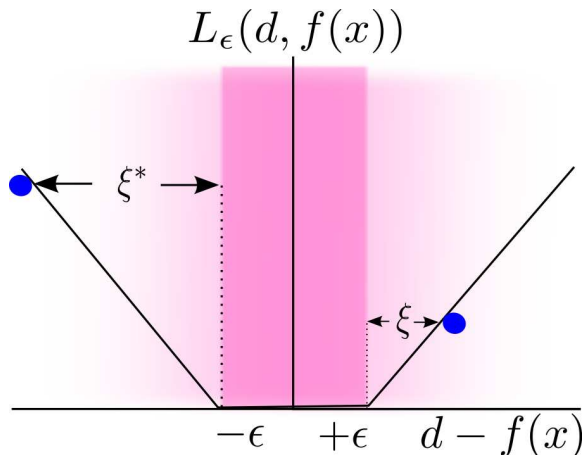


Figure 2.1: Illustration of the ϵ -insensitive loss function.

The structural functional (2.4) can be expressed as an optimization problem in its primal form as follows [78]:

$$\begin{aligned} \min_{\mathbf{w}, b, L_\epsilon} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N |d_i - y_i|_\epsilon \\ \text{s.t.} \quad & \begin{cases} d_i - y_i \leq \epsilon + \xi_i \\ y_i - d_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned}$$

where the summation in the cost function accounts for the ϵ -insensitive training error, which forms a tube where the solution is allowed to be defined without penalization, as shown in Figure 2.2 for the linear case, and in Figure 2.3 for the non-linear regression case. The constant $C > 0$ describes the trade off between the training error and the penalizing term $\|\mathbf{w}\|_2^2$. The term $\|\mathbf{w}\|_2^2$ is penalized to enforce a sparse solution on \mathbf{w} . The variables ξ_i and ξ_i^* are two sets of nonnegative slack variables that describe the ϵ -insensitive loss function.

The objective function in the primal can be rewritten in terms of the slack vari-

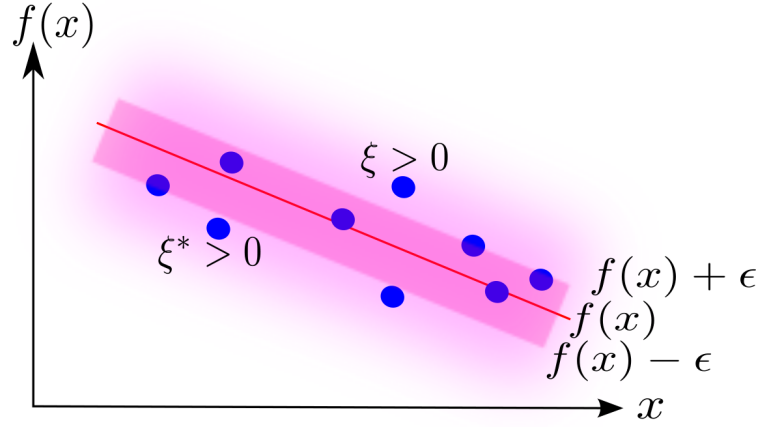


Figure 2.2: Linear regression: illustration of an ϵ -insensitive tube, fitted to the data points shown as circles.

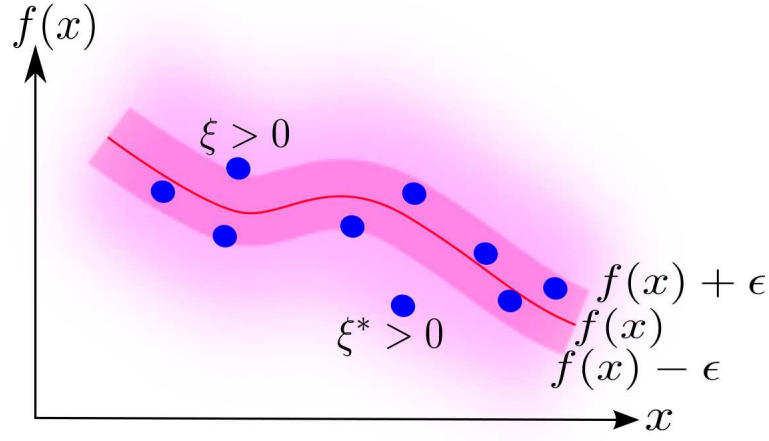


Figure 2.3: Illustration of an SVM for non-linear regression, showing the regression curve together with the ϵ -insensitive tube.

ables ξ and ξ^* , by observing the restrictions of the primal and the definition of the ϵ -insensitive function, and thus defining $\xi = d_i - y_i - \epsilon$ and $\xi^* = y_i - d_i - \epsilon$. Then

one obtains another common version of the primal problem as follows:

$$\begin{aligned}
& \min_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*} \quad \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \\
& \text{s.t.} \quad \begin{cases} d_i - \mathbf{w}^T \mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^T \mathbf{x}_i + b - d_i \leq \epsilon + \xi_i^* \\ \boldsymbol{\xi}, \boldsymbol{\xi}^* \geq \mathbf{0} \end{cases} \\
& \text{for} \quad i = 1, 2, \dots, N.
\end{aligned} \tag{2.6}$$

Next, we define the dual problem using the Lagrange multipliers method, where the following Lagrangian function is defined:

$$\begin{aligned}
L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*) &= \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N (\gamma_i \xi_i + \gamma_i^* \xi_i^*) \\
&\quad - \sum_{i=1}^N \alpha_i (\mathbf{w}^T \mathbf{x}_i + b - d_i + \epsilon + \xi_i) \\
&\quad - \sum_{i=1}^N \alpha_i^* (d_i - \mathbf{w}^T \mathbf{x}_i - b + \epsilon + \xi_i^*)
\end{aligned} \tag{2.7}$$

where $\gamma_i, \gamma_i^*, \alpha_i$, and α_i^* denote the Lagrange multipliers associated with the objective function and constraints respectively. The associated stationary points are defined by the following partial derivatives:

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^N (\alpha_i - \alpha_i^*) \mathbf{x}_i = \mathbf{0}, \tag{2.8a}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial b} = \sum_{i=1}^N (\alpha_i^* - \alpha_i) = 0, \tag{2.8b}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \boldsymbol{\xi}} = C\mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\gamma} = \mathbf{0}, \tag{2.8c}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \boldsymbol{\xi}^*} = C\mathbf{1} - \boldsymbol{\alpha}^* - \boldsymbol{\gamma}^* = \mathbf{0}. \tag{2.8d}$$

Then, the dual problem using the method of Lagrange multipliers is stated as follows:

$$\begin{aligned} & \max_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*} L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*) \\ & \text{s.t.} \quad \begin{cases} \frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \mathbf{w}} = \mathbf{0} \\ \frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial b} = 0 \\ \frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \boldsymbol{\xi}} = \mathbf{0} \\ \frac{\partial L(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*)}{\partial \boldsymbol{\xi}^*} = \mathbf{0} \\ \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^* \geq \mathbf{0} \end{cases} \end{aligned} \quad (2.9)$$

and the following problem is the expanded version of problem (2.9) above:

$$\begin{aligned} & \max_{\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^*} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*) - \sum_{i=1}^N (\gamma_i \xi_i + \gamma_i^* \xi_i^*) \\ & \text{s.t.} \quad \begin{cases} - \sum_{i=1}^N \alpha_i (\mathbf{w}^T \mathbf{x}_i + b - d_i + \epsilon + \xi_i) \\ - \sum_{i=1}^N \alpha_i^* (d_i - \mathbf{w}^T \mathbf{x}_i - b + \epsilon + \xi_i^*) \\ \mathbf{w} - \sum_{i=1}^N (\alpha_i - \alpha_i^*) \mathbf{x}_i = \mathbf{0} \\ \sum_{i=1}^N (\alpha_i^* - \alpha_i) = 0 \\ C\mathbf{1} - \boldsymbol{\alpha} - \boldsymbol{\gamma} = \mathbf{0} \\ C\mathbf{1} - \boldsymbol{\alpha}^* - \boldsymbol{\gamma}^* = \mathbf{0} \\ \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\gamma}, \boldsymbol{\gamma}^* \geq \mathbf{0} \end{cases} \\ & \text{for} \quad i = 1, 2, \dots, N. \end{aligned} \quad (2.10)$$

However, it is possible to remove three constraints by noticing from (2.8a) that one can solve for \mathbf{w} as follows

$$\mathbf{w} = \sum_{i=1}^N (\alpha_i - \alpha_i^*) \mathbf{x}_i \quad (2.11a)$$

and also one can solve for both $\boldsymbol{\gamma}$, and $\boldsymbol{\gamma}^*$ from (2.8c) and (2.8d) as

$$\boldsymbol{\gamma} = C\mathbf{1} - \boldsymbol{\alpha} \quad (2.11b)$$

$$\boldsymbol{\gamma}^* = C\mathbf{1} - \boldsymbol{\alpha}^* \quad (2.11c)$$

which also yields the boundary condition $\mathbf{0} \leq \boldsymbol{\alpha}, \boldsymbol{\alpha}^* \leq C\mathbf{1}$. If we substitute the equalities (2.11a)-(2.11c) into the objective function, and perform some analytic operations, we arrive to the following well known reduced dual problem [78]:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) \mathbf{x}_i^T \mathbf{x}_j \\ & -\epsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) + \sum_{i=1}^N d_i (\alpha_i - \alpha_i^*) \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N (\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \leq C \\ \alpha_i, \alpha_i^* \geq 0 \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned} \quad (2.12)$$

Problems (2.6) and (2.12) solve the linear regression problems, and for the non-linear regression case one just introduce a kernel function formulation. For the primal case, the sole modification is on the restrictions which are redefined as follows

$$\mathbf{w}^T k(\mathbf{x}_i, \cdot) + b - d_i \leq \epsilon + \xi_i \quad (2.13a)$$

$$d_i - \mathbf{w}^T k(\mathbf{x}_i, \cdot) - b \leq \epsilon + \xi_i^* \quad (2.13b)$$

for $i, j = 1, 2, \dots, N$. For the dual problem the objective function is redefined as

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \quad & -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(\mathbf{x}_i, \mathbf{x}_j) \\ & -\epsilon \sum_{i=1}^N (\alpha_i + \alpha_i^*) + \sum_{i=1}^N d_i (\alpha_i - \alpha_i^*). \end{aligned} \quad (2.14)$$

This formulation is used to implement learning methods discussed next.

2.3 Support Vector Regression Learning Methods

Support Vector Machines for Regression (SVRs) share the same advantages and disadvantages as SVMs. That means, SVMs training methods can be extended to SVRs whenever they are not directly dependent on the particularities of the SVM problem. Since most SVM training methods depend on the particular mathematical formulation of SVM, very few of them can be extended to SVR.

Collobert *et. al.* reported in 2001 an adaptation of Joachims' SVM method for SVR problems [39]. The authors start with the primal problem in (2.6), and then they reformulate the dual (2.12) with a vector-matrix notation in order to have the following Quadratic Programming (QP) minimization problem:

$$\begin{aligned}
 \min_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} \quad & Q(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_i^*) = \frac{1}{2} (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T \mathbf{K} (\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) \\
 & - (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T \mathbf{d} + \epsilon (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^T \mathbf{1} \\
 \text{s.t.} \quad & \begin{cases} (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*)^T \mathbf{1} = 0 \\ \alpha_i, \alpha_i^* \leq C \\ -\alpha_i, -\alpha_i^* \leq 0 \end{cases} \\
 \text{for} \quad & i = 1, 2, \dots, N
 \end{aligned} \tag{2.15}$$

where \mathbf{K} is the kernel matrix described in (A.21).

Next, the authors perform the same decomposition proposed by Osuna (see Section A.2.1) defining the working set \mathcal{B} and the fixed set \mathcal{M} , where the size of the working set is $|\mathcal{B}|$. Also, the authors extend Joachims' idea of the steepest descent direction \mathbf{p} to select the working set at each iteration of the SVR dual problem (see

Section A.2.2). That is, problem described in (A.30) is reformulated as follows:

$$\begin{aligned} \min \quad & \nabla Q(\boldsymbol{\alpha}_i, \boldsymbol{\alpha}_i^*)^T \mathbf{p} \\ \text{s.t.} \quad & \left\{ \begin{array}{ll} \sum_{i=1}^N p_i - p_i^* & = 0 \\ p_i & \geq 0 \quad \text{for } i : \alpha_i = 0 \\ p_i^* & \geq 0 \quad \text{for } i : \alpha_i^* = 0 \\ p_i & \leq 0 \quad \text{for } i : \alpha_i = C \\ p_i^* & \leq 0 \quad \text{for } i : \alpha_i^* = C \\ \mathbf{p} & \leq 1 \\ \mathbf{p} & \geq -1 \end{array} \right. \end{aligned} \quad (2.16)$$

where $\mathbf{p} = (p_1, \dots, p_N, p_1^*, \dots, p_N^*)$, and the size of the working set is given the by $|\mathcal{B}| \equiv |p_i|$ for all $p_i \neq 0$. The complete process is shown in Algorithm B.1.

Finally, the authors give proof of convergence following Platt's SMO idea selecting a working set size of 2 samples. The problems with this implementation are inherited from Joachims' and augmented due to the complexity of the SVR definition. Particularly, the issues with this algorithm are the inherent QP solution process and the additional steepest descent process. Furthermore, they introduced the concept of shrinking, that is, to permanently remove from the working set the bounded support vectors after a number of iterations. Although the heuristic of shrinking dramatically speeds up the training process, convergence is not guaranteed anymore. However, Collobert's *et. al.* shrinking method is still the most popular large-scale SVR training strategy.

2.4 Linear Programming Support Vector Regression

Linear Programming (LP) problems can be stated in the *standard* form

$$\begin{aligned} \max_{\mathbf{z} \in \mathbb{R}^n} \quad & \mathbf{c}^T \mathbf{z} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A} \mathbf{z} \leq \mathbf{b} \\ \mathbf{z} \geq \mathbf{0} \end{cases} \end{aligned} \quad (2.17)$$

where $\mathbf{z} \in \mathbb{R}^n$ is a vector containing the unknowns, $\mathbf{c} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ are vectors of known parameters, and $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix of known coefficients associated to \mathbf{z} through a linear relationship. However, there is also the *canonical* form

$$\begin{aligned} \min_{\mathbf{z} \in \mathbb{R}^n} \quad & \mathbf{c}^T \mathbf{z} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A} \mathbf{z} = \mathbf{b} \\ \mathbf{z} \geq \mathbf{0}. \end{cases} \end{aligned} \quad (2.18)$$

In fact, there is currently no agreement as to which of the previous problems is the canonical or standard form. In this dissertation, problem (2.18) will be regarded as the LP primal in its *canonical* form.

By introducing the Lagrange multipliers $(\boldsymbol{\lambda}, \mathbf{s})$ into the primal problem above, one obtains the following dual

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \mathbf{b}^T \boldsymbol{\lambda} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{s} \geq \mathbf{0}. \end{cases} \end{aligned} \quad (2.19)$$

where $\boldsymbol{\lambda}$ is a vector of dual variables defined over \mathbb{R}^m , and \mathbf{s} is a slack vector variable in \mathbb{R}^n .

The solution to the primal problem is denoted as \mathbf{z}^* , and the solution to the dual problem is denoted as $(\boldsymbol{\lambda}^*, \mathbf{s}^*)$. The duality theorem states that $\mathbf{c}^T \mathbf{z}^* = \mathbf{b}^T \boldsymbol{\lambda}^*$, which means that the solution \mathbf{z}^* also solves the dual, and the solution of the dual $(\boldsymbol{\lambda}^*, \mathbf{s}^*)$ also solves the primal [203].

The idea of Linear Programming Support Vector Regression (LP-SVR) is to pose the SVR problem (2.6) as a linear program either in its canonical (2.18) or standard (2.17) forms.

2.4.1 SVR Learning

Let n and m be the number of variables and constraints respectively in an SVR-based optimization problem. To solve such optimization problems, there are three known strategies. First, if the problem can fit within system memory, the strategy is to directly use interior point methods which have memory consumption of $\mathcal{O}(n^2)$. Second, if the problem is medium size, active-set methods are proven to be more appropriate [201]. They need $\mathcal{O}(n_{SV_s}^2)$ memory, where n_{SV_s} is characterized by the number of support vectors of the problem. That is, active-set methods deal with the complete set of support vectors only. Third, for very large datasets, one currently uses working set methods (*i.e.*, decomposition methods). Among these are Collobert's [39] or Mangasarian's [22]. These methods have memory consumption of $\mathcal{O}(n)$.

The training strategies are summarized in Figure 2.4. Particularly, this dissertation will introduce a working set algorithm for LP-SVR learning later in Chapter 3. But before that, let us review some LP-SVR formulations that will get the reader in the appropriate context of this dissertation.

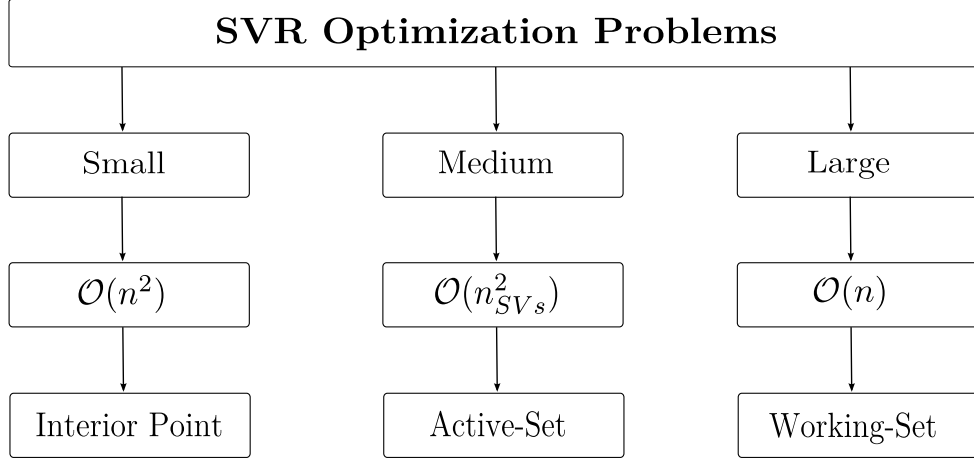


Figure 2.4: Depending on the problem size, one may find three different strategies for SVR training.

2.4.2 ν -LPR

Smola, *et al.* in 1999 [177], first introduced a linear programming approach for SVR. Their formulation uses the following ν -SVR model:

$$\begin{aligned}
 & \min_{\alpha^+, \alpha^-, b, \xi, \xi^*, \epsilon} \quad \sum_i^N \frac{1}{N} (\alpha_i^+ + \alpha_i^-) + \frac{C}{N} (\xi_i + \xi_i^*) + C\nu\epsilon & (2.20) \\
 & \text{s.t.} \quad \begin{cases} \sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) + b - d_i \leq \epsilon + \xi_i \\ d_i - \sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) - b \leq \epsilon + \xi_i^* \\ \alpha_i^-, \alpha_i^+, \xi_i, \xi_i^*, \epsilon \geq 0 \end{cases} \\
 & \text{for} \quad i = 1, 2, \dots, N.
 \end{aligned}$$

where $\nu \in [0, 1]$ can be used to control the number of errors in ϵ . In this way, the value of ϵ is automatically estimated. The ν -SVR comes from a special type of SVM called ν -SVM [201]. The authors named their formulation as “ ν -LPR,” as in “nu-linear programming regression.” Smola, *et al.* poses problem (2.20) as a linear

program in the following form:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{c}^T \mathbf{z} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{r} + \mathbf{b} - \mathbf{u} = \mathbf{0} \\ \mathbf{z}, \mathbf{u} \geq \mathbf{0}. \end{cases}, \end{aligned} \quad (2.21)$$

where $\mathbf{z} \in \mathbb{R}^{4N+1}$, $\mathbf{r} \in \mathbb{R}$, $\mathbf{u} \in \mathbb{R}^{2N}$, and the matrices $\mathbf{A} \in \mathbb{R}^{2N \times 4N+1}$, $\mathbf{B}, \mathbf{b} \in \mathbb{R}^{2N}$, $\mathbf{c} \in \mathbb{R}^{4N+1}$ are denoted as follows:

$$\mathbf{c} = \begin{pmatrix} 1 & 1 & C1 & C1 & C\nu N \end{pmatrix}, \quad (2.22a)$$

$$\mathbf{z} = \begin{pmatrix} \boldsymbol{\alpha}^+ & \boldsymbol{\alpha}^- & \boldsymbol{\xi}^+ & \boldsymbol{\xi}^- & \epsilon \end{pmatrix}, \quad (2.22b)$$

$$\mathbf{A} = \begin{pmatrix} -\mathbf{K} & \mathbf{K} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{K} & -\mathbf{K} & \mathbf{0} & \mathbf{1} & \mathbf{1} \end{pmatrix}, \quad (2.22c)$$

$$\mathbf{B} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} \mathbf{d} \\ -\mathbf{d} \end{pmatrix}. \quad (2.22d)$$

The dual problem is derived in order to use a primal-dual interior point solver with a predictor-corrector strategy. In doing this, they also derive the corresponding Karush-Kuhn-Tucker (KKT) conditions.

Smola, *et al.* [177] suggest that the parameter ν is proportional to the number of support vectors. Although the main contribution is to show the efficiency of an SVR using LP, a critique is that the authors eliminated one parameter by adding another, *i.e.*, ν instead of ϵ . Moreover, the problems with LP-SVR models is that the program dimension is very large, *i.e.*, $4N + 2$. As a consequence, the authors report

performing regression with training sets consisting of only 50 and 80 data points.

2.4.3 LP-SVR with Modified LPC

Mangasarian, *et al.* [127] in 2002 pioneered the linear programming formulations for large-scale SVRs. The authors analyzed the ν -LPR [177] method and provided means for training with large-scale datasets. The SVR formulation they use is the following

$$\begin{aligned}
& \min_{\alpha^+, \alpha^-, b, t, \epsilon} \quad \sum_i^N \frac{1}{N} (\alpha_i^+ + \alpha_i^-) + \frac{C}{N} t_i + C(1 - \mu)\epsilon \quad (2.23) \\
& \text{s.t.} \quad \begin{cases} \sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) + b - d_i \leq \epsilon + t_i \\ d_i - \sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) - b \leq \epsilon + t_i \\ \alpha_i^-, \alpha_i^+, t_i, \epsilon \geq 0 \end{cases} \\
& \text{for} \quad i = 1, 2, \dots, N.
\end{aligned}$$

where $\mu \in [0, 1]$ is a positive parameter that drives the tolerance error ϵ : if $\mu \rightarrow 1$ the errors are highly penalized.

If one take a closer look to problems (2.20) and (2.23), one notices that $t_i = (\xi_i + \xi_i^*)$. In fact, the authors show that problems (2.20) and (2.23) are equivalent.

The significance of this formulation is the use of a modified version of the Linear Programming Chunking (LPC) algorithm [22]. The LPC algorithm was originally utilized to solve the problem with a subset of the constraints. However, Mangasarian, *et al.* also proposed a modified version that constrains the variables as well. This modification promotes a smaller version of the problem, hence faster computations in terms of memory allocation.

The original LPC algorithm is listed in Algorithm B.2 in Appendix B. The algorithm assumes that every sub-problem has a vertex solution. LPC also considers that the matrix-vector block $\begin{pmatrix} \mathbf{A}^0 & \mathbf{b}^0 \end{pmatrix}$ is empty and that the set of active constraints at iteration j is $\begin{pmatrix} \bar{\mathbf{A}}^j & \bar{\mathbf{b}}^j \end{pmatrix}$. Bradley and Mangasarian [22] give proof of

finite termination.

2.4.4 Wavelet Kernel-based LP-SVR

In 2009, Lu, *et al.* [123], reported an SVR method based entirely in the LP primal in its standard form. The authors present an alternative to the typical kernels and use wavelet-based kernels instead. The paper shows an application to nonlinear dynamical systems identification [123]. The authors start by changing the ℓ_2^2 -norm by the ℓ_1 -norm, and also use the notion of the kernel functions for nonlinear problems; and they reformulate problem (2.6) as follows:

$$\begin{aligned} \min \quad & \frac{1}{2} \|\boldsymbol{\alpha}\|_1 + C \sum_{i=1}^N (\xi_i + \xi_i^*) \\ \text{s.t.} \quad & \begin{cases} d_i - \sum_{j=1}^N \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \leq \epsilon + \xi_i \\ \sum_{j=1}^N \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) - d_i \leq \epsilon + \xi_i^* \\ \boldsymbol{\xi}, \boldsymbol{\xi}^* \geq \mathbf{0} \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N, \end{aligned} \tag{2.24}$$

where $\boldsymbol{\alpha}$ comes from the Representer Theorem notation [170]. Then, the authors arrive at the following LP problem in its primal standard form:

$$\begin{aligned} \min_{\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\xi}} \quad & \begin{pmatrix} \mathbf{1} & \mathbf{1} & 2\mathbf{C} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \\ \boldsymbol{\xi} \end{pmatrix} \\ \text{s.t.} \quad & \left\{ \begin{pmatrix} \mathbf{K} & -\mathbf{K} & -\mathbf{I} \\ -\mathbf{K} & \mathbf{K} & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha}^+ \\ \boldsymbol{\alpha}^- \\ \boldsymbol{\xi} \end{pmatrix} \leq \begin{pmatrix} \mathbf{d} + \epsilon \mathbf{1} \\ \epsilon \mathbf{1} - \mathbf{d} \end{pmatrix} \right. \end{aligned} \tag{2.25}$$

where $\alpha_i = \alpha_i^+ - \alpha_i^-$; that is, the decomposition of the positive α_i^+ and the negative α_i^- part of α_i . Then, the authors show that this formulation performs around 25 times faster than the QP-SVR. No details were given with respect to the solver

used, and no proof of convergence was addressed. The authors also show that in measuring the root mean squared error (RMSE), the QP-SVR performs better than their LP-SVR approach. However, they also report that this formulation requires fewer support vectors. Hence, a sparser solution is obtained, having the advantage of a reduced number of support vectors, which also represents computationally efficient solutions.

2.4.5 ℓ_1 -norm LP-SVR

In 2010, Zhang, *et al.* [214], performed a study in regard to the sparseness of ℓ_1 -norm-based SVMs and SVRs. The authors reported the following formulation of an ℓ_1 -norm SVR

$$\begin{aligned}
& \max_{\boldsymbol{\alpha}^\pm, b^\pm, \boldsymbol{\xi}^{(*)}, \mathbf{u}^\pm} && -\sum_i^N \frac{1}{N} (\alpha_i^+ + \alpha_i^-) - \delta (b^+ + b^-) - C \sum_i^N (\xi_i + \xi_i^*) \\
& \text{s.t.} && \begin{cases} -\sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) - (b^+ - b^-) + d_i + u_i = \epsilon + \xi_i \\ -d_i + \sum_j^N (\alpha_j^+ - \alpha_j^-) k(\mathbf{x}_j, \mathbf{x}_i) + (b^+ - b^-) + u_i = \epsilon + \xi_i^* \\ \alpha_i^-, \alpha_i^+, \xi_i, \xi_i^*, \epsilon, b^+, b^-, u_i^+, u_i^- \geq 0 \end{cases} \\
& \text{for} && i = 1, 2, \dots, N.
\end{aligned} \tag{2.26}$$

where $\delta \in \mathbb{R}$ (typically small: $\delta = 1 \times 10^{-11}$) is used to avoid possible infinite optimal solutions, and u_i is a slack variable. Then, defining

$$\mathbf{A} = \begin{pmatrix} -\mathbf{K} & \mathbf{K} & -\mathbf{1} & \mathbf{1} & -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{K} & -\mathbf{K} & \mathbf{1} & -\mathbf{1} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} \mathbf{1}\epsilon - \mathbf{d} \\ \mathbf{1}\epsilon - \mathbf{d} \end{pmatrix}, \tag{2.27a}$$

$$\mathbf{z} = \begin{pmatrix} \boldsymbol{\alpha}^+ & \boldsymbol{\alpha}^- & b^+ & b^- & \boldsymbol{\xi} & \boldsymbol{\xi}^* & \mathbf{u}^+ & \mathbf{u}^- \end{pmatrix}^T, \tag{2.27b}$$

$$\mathbf{c} = \begin{pmatrix} -\mathbf{1} & -\mathbf{1} & -\delta & -\delta & -C\mathbf{1} & -C\mathbf{1} & \mathbf{0} & \mathbf{0} \end{pmatrix}^T \tag{2.27c}$$

the authors proceeded to solve the following linear programming variation

$$\begin{aligned} \max_{\mathbf{z}} \quad & \mathbf{c}^T \mathbf{z} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A} \mathbf{z} = \mathbf{b} \\ \mathbf{z} \geq \mathbf{0}. \end{cases} \end{aligned} \tag{2.28}$$

The evident problem with this formulation is the size of the problem: $6N + 2$ variables and $2N$ constraints. Therefore, the author only reports experiments for small size problems; *e.g.*, 351, 297, and 345 data points, using an RBF kernel. The poor formulation makes the size problem more evident since the authors used the simplex method. As a consequence very large training times will be experienced.

One obvious critique is the inclusion of δ , making the set of hyper-parameters $[\epsilon, C, \sigma, \delta]$ to be chosen more heuristically.

2.5 Current Trends and Open Problems

The Support Vector learning field is very active (see [3, 18, 25, 33, 37, 38, 69, 73, 85, 117, 118, 126, 134, 142, 173, 174, 185, 202, 207, 212]). As a consequence, the treatment of large-scale SVM is of interest to researchers in the field (see [30, 31, 50, 54, 55, 68, 79, 87, 96, 99, 116, 122, 135, 139, 141, 151, 170, 181, 199, 205, 208–210, 213, 215–217, 222]). However, a number of open issues that have to be addressed still exist.

Recently, SVR algorithmic development seems to be at a more stable stage; in spite of this, one issue is to find whether linear programming SVR approaches will lead to more satisfactory results [178, 214].

The problem of empirical tuning [171, 178] of SVR hyper-parameters (*i.e.*, C, σ), has to be devised to make SVRs less dependent on the skill of the experimenter [16, 178].

Optimization techniques developed within the context of SVRs are also required in order to improve treatment of large datasets [178]. This may be done in combina-

tion with reduced set methods for speeding up the training phase for large datasets. This topic is of huge importance as machine learning applications demand algorithms that are capable of dealing with datasets that are at least larger than 1 million samples [178].

Other problems out of the scope of this dissertation are still considered open. These include the following: more data dependent generalization bounds, efficient training algorithms, and automatic kernel selection procedures.

Chapter 3

Large-Scale Linear Programming

Support Vector Regression

In this chapter we develop a method to train a pattern recognition model in a large-scale setting. This chapter presents the main theoretical development of this research, consisting of four major points: (i) a Support Vector Regression (SVR) problem posed entirely on Linear Programming (LP) techniques, which is modeled on a primal and dual fashion followed by the definition of optimality conditions; (ii) a sequential optimization method based on variables decomposition, constraints decomposition, and primal-dual interior point methods for solving large-scale regression/classification problems; (iii) the convergence and optimality conditions of the sequential optimization; and (iv) experimental results of the proposed sequential optimization method over typical small-, medium-, and large-scale problems.

3.1 Introduction

In the following sections, we will explain Algorithm 3.1 which proposes a large-scale model that involves three major parts that increases computational tractability of large-scale problems.

First, the large-scale Linear Programming Support Vector Regression (LP-SVR) problem is reduced by variable decomposition, which exploits LP-SVR structure to produce a lower-dimensional representation of the decomposed LP sub-problem. The finite termination of the decomposition strategy is guaranteed by an adaptation of

Algorithm 3.1 Training Algorithm that uses a “Variables and Constraints Decomposition” Strategy to for a Large Scale Training Set.

Require: B_0 , initial working set size.

Require: $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$, a training set with N samples.

Require: τ , number of blocks for block decomposition.

```

1:  $\mathcal{B} \leftarrow$  randomly selected  $B_0$  indices as the initial working set.
2:  $\mathcal{M} \leftarrow$  those indices not in  $\mathcal{B}$ , that denotes the initial fixed set.
3: Begin ▷ Variables Decomposition.
4:    $\mathcal{W} \leftarrow \{\mathbf{x}_i, d_i\}_{i \in \mathcal{B}}$ 
5:   Fix  $\alpha_j = 0$  for all  $j \in \mathcal{M}$ . ▷ Variables in problem (3.2) can be ignored.
6:   Begin ▷ Constraints Decomposition.
7:     Pose the problem as an LP problem by defining  $\mathbf{A}, \mathbf{b}, \mathbf{c}$  with (3.3a)-(3.3d).
8:      $\mathbf{A}_{\mathcal{B}}, \mathbf{b}_{\mathcal{B}}, \mathbf{c}_{\mathcal{B}} \leftarrow \mathbf{A}, \mathbf{b}, \mathbf{c}$ 
9:      $\begin{pmatrix} \mathbf{A}_{\mathcal{R}}^1 & \mathbf{b}_{\mathcal{R}}^1 \\ \mathbf{A}_{\mathcal{R}}^2 & \mathbf{b}_{\mathcal{R}}^2 \\ \vdots & \vdots \\ \mathbf{A}_{\mathcal{R}}^\tau & \mathbf{b}_{\mathcal{R}}^\tau \end{pmatrix} \leftarrow \text{BLOCKPARTITION}(\tau, \mathbf{A}_{\mathcal{B}}, \mathbf{b}_{\mathcal{B}})$  ▷ Partition into  $\tau$  blocks.
10:     $t = 0$  ▷ Iterations counter.
11:    repeat
12:       $t = t + 1$ 
13:       $\mathbf{z}_{\mathcal{R}}^{(t)} \leftarrow \text{IPMSOLVELP}(\mathbf{c}_{\mathcal{R}}, \mathbf{A}_{\mathcal{R}}, \mathbf{b}_{\mathcal{R}})$  ▷ Solves (3.19).
14:       $z_{i, \mathcal{B}}^{(t)} = \begin{cases} z_{j, \mathcal{R}}^{(t)} & \text{if } j = i, \text{ for all } j \in \mathcal{R} \\ 0 & \text{otherwise,} \end{cases}$ 
15:      until  $\left( \mathbf{c}_{\mathcal{B}}^T \mathbf{z}_{\mathcal{B}}^{(t)} = \mathbf{c}_{\mathcal{B}}^T \mathbf{z}_{\mathcal{B}}^{(t+4)} \right)$  ▷ Stops if no change during four iterations.
16:       $\mathbf{z}_{\mathcal{B}} \leftarrow \mathbf{z}_{\mathcal{B}}^{(t)}$  ▷ Problem is solved for  $\mathcal{W}$ .
17:    End ▷ End Constraint Decomposition.
18:    for all  $j \in \mathcal{M}$  do ▷ To verify if problem is solved for  $\mathcal{M}$ .
19:      Reconstruct  $u_j, \xi_j$  using (3.12)-(3.13) and verify the primal LP.
20:      Fix  $\lambda_j = 0$ , reconstruct  $s_j$  using (3.14), and verify the dual LP.
21:       $\tilde{\mathcal{B}} \leftarrow \text{VERIFYCOMPLEMENTARITY}(z_j, s_j, B_0)$ 
22:    end for
23:    if  $z_j s_j \neq 0$  then ▷ Problem (3.2) is not solved
24:       $\mathcal{B} \leftarrow \text{CREATENEWWORKINGSET}(\mathcal{B}, B_0, \tilde{\mathcal{B}})$ 
25:    else
26:      Stop Training
27:    end if
28: End ▷ End Variable Decomposition.
29:  $(\alpha^+ \ \alpha^- \ b^+ \ b^- \ \xi \ \mathbf{u})^T \leftarrow \mathbf{z}$ 
Ensure:  $(\alpha^+, \alpha^-, b^+, b^-)$ , the SVR solution.

```

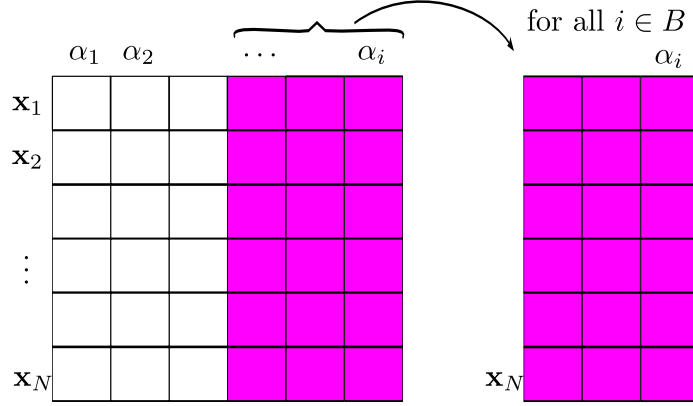


Figure 3.1: Variable decomposition strategy from Torii, *et al.* [187], in which a subset of the variables α_i is considered for the solution of the LP-SVR problem, for all $i \in \mathcal{B}$. The shaded area corresponds to an arbitrary selection of $|\mathcal{B}|$ indices.

Torii, *et al.* [187] infinite-loop prevention algorithm. This strategy is depicted in Figure 3.1.

Second, we will design a constraint decomposition strategy that generates a number of smaller sub-problems by again exploiting LP-SVR structure. The resulting LP problems are solved sequentially in a monotonically non-increasing objective function fashion. Convergence is guaranteed by adapting Bradley's, *et al.* [22, 23] theorems. Bradley's, *et al.* algorithm is known as Linear Programming Chunking (LPC), and it is depicted in Figure 3.2. As shown in the figure, the algorithm consist on dividing an LP into blocks of smaller LPs without taking any advantage of the SVR structure.

Third, the two-way-reduced LP-SVR sub-problems (*i.e.*, variables and constraint reduction) are solved using interior point methods, which have a very fast rate of convergence. This is the key that balances the total computational time of performing two decompositions and solving several LPs. Using any other approach (*i.e.*, simplex methods) would dramatically increase computational efforts. We use the method by Argaez *et al.* [9].

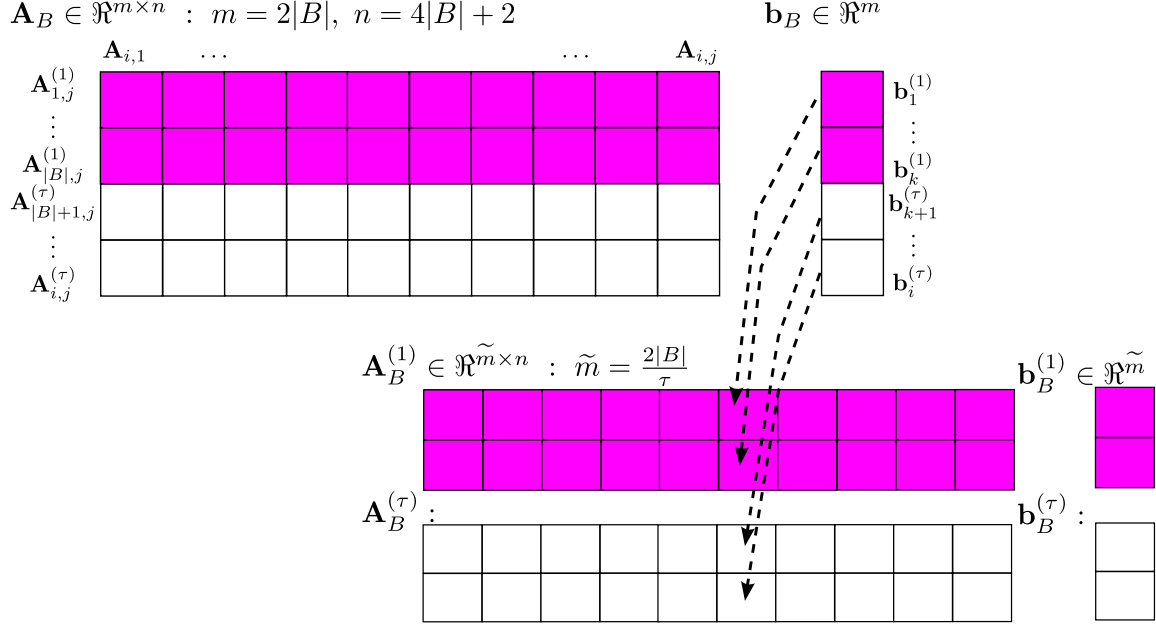


Figure 3.2: LP-SVR constraints decomposition strategy. This figure shows an example of the decomposition of the LP constraints. Here the coefficients matrix \mathbf{A} and vector \mathbf{b} are divided in τ blocks. This figure shows the linear programming chunking (LPC) approach by Bradley's, *et al.* [22,23].

In this work, we train large-scale datasets and at the same time produce sparser solutions in terms of the number of support vectors. This results in a more efficient model in terms of future kernel evaluations which can be made faster.

We will show that the proposed model is sparser than the regular SVR and other SV-based classifiers. Although this was studied by Zhang, *et al.* [214], we will demonstrate that as the problem size increases, the more sparser the solution becomes. The proposed approach is comparable with other formulations in terms of performance, which is desirable.

3.2 Large-Scale LP-SVR Formulation

Let us assume we have training samples $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^M$ is a regressor and d is the desired output. Then, one can define a non-linear SVR prediction function:

$$d_j \equiv f(\mathbf{x}_j) = \sum_{i=1}^N (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_i, \mathbf{x}_j) + (b^+ - b^-), \quad (3.1)$$

where $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \in \mathbb{R}_+^N$; $b^+, b^- \in \mathbb{R}_+$; $k(\cdot, \cdot)$ is a valid kernel function [41, 129] in the form of (A.18)-(A.20); $\alpha^+ = \max\{\alpha, 0\}$, $\alpha^- = \max\{-\alpha, 0\}$; $b^+ = \max\{b, 0\}$, $b^- = \max\{-b, 0\}$; $\boldsymbol{\alpha} \in \mathbb{R}^N$; and $b \in \mathbb{R}$. Kernel functions map the input feature vectors to the kernel-induced feature space denoted \mathcal{H} since these kernel functions follow the properties of Hilbert spaces [129]. The kernel-induced feature space for non-linear SVR can be defined as $\mathcal{H} = \left\{ f(\mathbf{x}_j) : f(\mathbf{x}_j) = \sum_{i=1}^N (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_i, \mathbf{x}_j) + (b^+ - b^-) \right\}$, for all $\mathbf{x}_j \in \mathbb{R}^M$ and $j = \{1, 2, \dots, N\}$.

The objective is to find the set of parameters $\boldsymbol{\alpha}$ and b . One can find these parameters via constrained optimization. Section 2.4 demonstrates that the primal SVR problem (2.6) eases the formulation and training. Therefore, the proposed model will be derived from problem (2.6).

3.2.1 Primal, Dual, and KKT Conditions

First, let us assume the mapping $k(\mathbf{x}_i, \mathbf{x}_j) : \mathcal{X}^{(N \times M) \times (M \times N)} \mapsto \mathcal{H}^{N \times N}$. Then, assume that the slack variables ξ_i, ξ_i^* can be expressed as simply $2\xi_i$ (see Section 2.4, $\xi_i \xi_i^* = 0$). Then, let us introduce a slack variable \mathbf{u} to get rid of the inequalities in the original SVR formulation (2.6). As a consequence of these assumptions, the

following optimization problem is proposed:

$$\begin{aligned}
& \min_{\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, b^+, b^-, \boldsymbol{\xi}, \mathbf{u}} && \sum_{i=1}^N (\alpha_i^+ + \alpha_i^- + 2C\xi_i) && (3.2) \\
& \text{s.t.} && \left\{ \begin{array}{l} -\sum_{i=1}^N (\alpha_i^+ - \alpha_i^-)k(\mathbf{x}_j, \mathbf{x}_i) \dots \\ \quad -b^+ + b^- - \xi_j + u_j = \epsilon - d_j \\ \sum_{i=1}^N (\alpha_i^+ - \alpha_i^-)k(\mathbf{x}_j, \mathbf{x}_i) \dots \\ \quad +b^+ - b^- - \xi_j + u_j = \epsilon + d_j \\ \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geq 0 \end{array} \right. \\
& \text{for} && j = 1, 2, \dots, N.
\end{aligned}$$

Problem (3.2) can be posed as the linear programming problem in (2.18). To do so, one can define the following equalities:

$$\mathbf{A} = \begin{pmatrix} -\mathbf{K} & \mathbf{K} & -1 & 1 & -\mathbf{I} & \mathbf{I} \\ \mathbf{K} & -\mathbf{K} & 1 & -1 & -\mathbf{I} & \mathbf{I} \end{pmatrix}, \quad (3.3a)$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{1}\epsilon - \mathbf{d} \\ \mathbf{1}\epsilon + \mathbf{d} \end{pmatrix}, \quad (3.3b)$$

$$\mathbf{z} = \left(\boldsymbol{\alpha}^+ \quad \boldsymbol{\alpha}^- \quad b^+ \quad b^- \quad \boldsymbol{\xi} \quad \mathbf{u} \right)^T, \quad (3.3c)$$

$$\mathbf{c} = \left(\mathbf{1} \quad \mathbf{1} \quad 0 \quad 0 \quad 2\mathbf{C} \quad \mathbf{0} \right)^T, \quad (3.3d)$$

where $\mathbf{A} \in \mathbb{R}^{(2N) \times (4N+2)}$, $\mathbf{b} \in \mathbb{R}^{2N}$, $\mathbf{z}, \mathbf{c} \in \mathbb{R}^{4N+2}$. If we use the above equalities, then problems (2.18) and (3.2) are identical, and we can claim that the problem has been posed as an LP problem.

We claim that problem (3.2) is an original formulation for LP-SVR. In comparison with the ν -LPR formulation by Smola, *et al.* [177] problem (3.2) (i) uses the canonical formulation, (ii) computes b , and \mathbf{u} implicitly, (iii) does not compute ϵ implicitly, (iv) does not require the parameter ν , (v) promotes efficiency in the sense of using only one ξ , and (vi) is a lower dimensional problem.

In comparison with Mangasarian, *et al.* [127], problem (3.2) (i) uses the canonical formulation, (ii) computes b implicitly, (iii) does not compute ϵ implicitly, and (iv) does not require the parameter μ . By (iii) and (iv) we provide the experimenter with more control of the sparseness of the solution [214]. In this case sparseness means fewer number of support vectors.

Similarly, Problem (3.2) in comparison to Lu, *et al.* [123] our LP-SVR formulation (3.2) (i) uses the canonical formulation and (ii) computes b implicitly. By (ii) the linear program (LP) size is reduced by a factor of $N^2 + N$.

In comparison with the ℓ_1 -norm LP-SVR formulation by Zhang, *et al.* [214] problem (3.2) does not require parameter δ and is more efficient in several ways: (i) uses only one ξ , (ii) avoids penalization of b , (iii) reduces computational efforts by forcing positivity in \mathbf{u} which reduces the LP problem size by $2N^2 + 2N$, and (iv) is a smaller problem.

Using equalities (3.3a)-(3.3d), we can obtain the dual problem of (3.2) as follows:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \mathbf{b}^T \boldsymbol{\lambda} \\ \text{s.t.} \quad & \begin{cases} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \\ \mathbf{s} \geq \mathbf{0}, \end{cases} \end{aligned} \tag{3.4}$$

which is equivalent to (2.19), where $\boldsymbol{\lambda}$ is a vector of dual variables defined over \mathbb{R}^{2N} , and \mathbf{s} is a slack vector variable in \mathbb{R}^{4N+2} .

Similarly, for the primal (3.2) and dual (3.4), the KKT conditions are defined as

follows:

$$\mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \quad (3.5a)$$

$$\mathbf{A}\mathbf{z} = \mathbf{b}, \quad (3.5b)$$

$$z_i s_i = 0, \quad (3.5c)$$

$$(\mathbf{z}, \mathbf{s}) \geq \mathbf{0}, \quad (3.5d)$$

$$\text{for } i = 1, 2, \dots, n,$$

where the equality $z_i s_i$ implies that one of both variables must be zero. This equality will be referred to as the *complementarity condition*. Note that the KKT conditions depend on the variables $(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$, and if the set of solutions $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ satisfy all the conditions, the problem is said to be solved. The set $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ is known as a *primal-dual solution*.

3.2.2 Optimality and Sparseness

Let \mathbf{z}^* be the solution to the primal problem (3.2), and let $(\boldsymbol{\lambda}^*, \mathbf{s}^*)$ be the solution to the dual problem (3.4). The proposed LP-SVR exhibits two important properties. First, that it has a *global solution*. That is, if \mathbf{z}^* is a minimum for problem (3.2), then \mathbf{z}^* is a global minimum since problem (3.2) is a convex problem (*i.e.*, a linear programming problem) [45, 46, 51, 62].

Second, its *optimality conditions* are well defined. That is, for problem (3.2), the KKT conditions (3.5a)-(3.5d) are necessary and sufficient for optimality since $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ is a solution to the primal (3.2) and dual (3.4), then it follows that the KKT conditions (3.5a)-(3.5d) are necessary and sufficient for optimality [45, 46, 62, 133].

One concern of the work presented here is to demonstrate that the solution of the proposed LP-SVR is better than that of SVRs in the sense of solution sparseness.

Sparseness in a solution is desired because any SV-based model relies on actual feature vectors \mathbf{x}_i to define the optimal set of model parameters (α_i, b) for all $i : \alpha_i \neq 0$. Especially since the feature vectors \mathbf{x}_i are required for kernel distances as shown in (3.1).

Zhang, *et al.* [214] performed a comprehensive study in regard to SVM sparseness. Zhang, *et al.* explains that sparseness of a learning machine depends on the problem and the precision of the solution. Then, the authors prove (see [214] Theorems 1 and 2) that, for their proposed LP-SVR, the solution is always sparser than regular SVRs. In fact, Zhang's theorems also hold for our formulation. To demonstrate this, the following definitions are given.

Definition 3.1 (Support Vectors). *Let $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ be a training set; let \mathbf{z} be a solution to problem (3.2); and let (3.1) be the regression function for problem (3.2). Then,*

1. $\mathcal{V}_S = \{\mathbf{x}_i : d_i - \epsilon < f(\mathbf{x}_i) < d_i + \epsilon\}$ defines the set of Saturated Support Vectors (SSVs).
2. $\mathcal{V}_E = \{\mathbf{x}_i : f(\mathbf{x}_i) = d_i + \epsilon, \text{ or } f(\mathbf{x}_i) = d_i - \epsilon\}$ defines the set of Exact Support Vectors (ESVs).
3. $\mathcal{V}_N = \{\mathbf{x}_i : f(\mathbf{x}_i) < d_i + \epsilon, \text{ or } f(\mathbf{x}_i) > d_i - \epsilon\}$ defines the set of Non-Support Vectors (NSVs).
4. $\mathcal{V}_\alpha = \{\mathbf{x}_i : \alpha_i \neq 0\}$ defines the set of Sparse Vectors (SPVs).
5. $N = |\mathcal{V}_S| + |\mathcal{V}_E| + |\mathcal{V}_N|$.
6. $\mathcal{S} = \mathcal{V}_S \cup \mathcal{V}_E$, means that the union of the SSVs and the ESVs is the set of Support Vectors (SVs).
7. $\mathcal{A} = \{\alpha_i : \alpha_i \neq 0\}$ denotes the set of Non-zero Coefficients of the decision function (3.1) and of problem (3.2).

Then, since Definition 3.1 is equivalent to Definition 2 in [214] by Zhang, *et al.*, then we can say that given an optimal solution \mathbf{z}^* to (3.2), the number of nonzero α_i coefficients of (3.2) has the following upper bound:

$$|\mathcal{A}| \leq |\mathcal{V}_E|, \quad (3.6)$$

for all $i : \alpha_i \neq 0$. This property states that ESVs characterize the sparseness of problem (3.2) just as SVs characterize the sparseness of any SVR formulation, such as (2.6), (2.12), or (2.15). The above property points out that the proposed LP-SVR problem (3.2) possess better sparseness than that of standard SVRs, since there are always several SSVs in standard SVRs, especially for practical noisy datasets used in recognition or regression problems [214].

Similarly, from Definition 2 in [214] by Zhang, *et al.*, we have that the number of nonzero coefficients of (3.2) has the following upper bound:

$$|\mathcal{A}| \leq \text{rank}(\mathbf{K}), \quad (3.7)$$

and the column vectors $k(\mathbf{x}_j, \mathbf{x}_1), k(\mathbf{x}_j, \mathbf{x}_2), \dots, k(\mathbf{x}_j, \mathbf{x}_i)$, are linearly independent for all $j \in \mathcal{A}$. This means that the LP-SVR regression function (3.1) can be exactly reproduced using only those samples that are SVs, without affecting performance. This property also indicates that vectors $k(\mathbf{x}_j, \mathbf{x}_1), k(\mathbf{x}_j, \mathbf{x}_2), \dots, k(\mathbf{x}_j, \mathbf{x}_i)$, for all $j \in \mathcal{A}$, in the decision function $f(\mathbf{x}) = \sum_{i \in \mathcal{A}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_i, \mathbf{x}) + (b^+ - b^-)$, are linearly independent. This means one cannot further reduce the number of basis functions in the regression function, and also indicates that the proposed LP-SVR (3.2) may lead to a sparser model representation.

Up to now, the properties of the proposed model have been explained. However, it is important to remark that problem (3.2) was designed to maximize computational efficiency without sacrificing accuracy. This has been achieved by not introducing unnecessary parameters into the problem and by posing a problem that minimizes

the number of SVs without affecting performance. In spite of this, the training phase (*i.e.*, learning process) still may be computationally expensive for applications with N larger than a few thousands. In the following section, a learning process for the case when N is very large is introduced.

3.3 LP-SVR Variable Decomposition

Let us consider the proposed Linear Program (3.2). In order to deal with large N problems, an idea is to divide the training set $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ in two subsets: a *working set* $\mathcal{W} = \{\mathbf{x}_i, d_i\}_{i \in \mathcal{B}}$ and a *fixed set* $\mathcal{F} = \{\mathbf{x}_i, d_i\}_{i \in \mathcal{M}}$; where we identify the members of each set with two index sets \mathcal{B} and \mathcal{M} respectively. These two sets are disjoint: $\mathcal{B} \cap \mathcal{M} = \emptyset$; and their union contains all the training set indices: $\mathcal{B} \cup \mathcal{M} = \{1, 2, \dots, N\}$. The initial size of \mathcal{B} is given by a parameter B_0 , ($2 \leq B_0 \leq N$), chosen heuristically. Note that for classification problems with q number of classes, B_0 is bounded as $q \leq B_0 \leq N$.

Under these definitions, a method is proposed, in which problem (3.2) is solved using only a subset of the variables $\mathbf{z} \geq 0$ from (3.3c), and a subset of the constraints $\mathbf{A}\mathbf{z} = \mathbf{b}$ from (3.3a and 3.3b). This approach is known as variables decomposition and constraints decomposition [187].

The variable decomposition strategy presented in this dissertation is an adaptation of the work by Torii, *et al.* [187], in which the author decomposes the variables of a linear program (LP). However, instead of simply decomposing variables of the LP, we exploit the properties of the LP-SVR to make the LP smaller. The constraints decomposition strategy is an adaptation of the linear programming chunking (LPC) algorithm introduced by Bradley, *et al.* [22, 23] but modified for LP-SVR efficiency.

Section 3.4 presents the strategy for solving an LP-SVR using a subset of the constraints $\mathbf{A}\mathbf{z} = \mathbf{b}$ from (3.3a and 3.3b). But first, let us address the problem of solving an LP-SVR using only a subset of the variables $\mathbf{z} \geq 0$ from (3.3c), which is

Algorithm 3.2 Variable Decomposition Strategy for Large-Scale LP-SVR Training. Modification of Torii, *et al.* [187] Algorithm.

- 1: Set \mathcal{B} with the first B_0 indices from \mathcal{T} .
 - 2: For all indices in \mathcal{B} **Solve LP** sub-problem.
 - 3: Verify if the current solution satisfies the KKT conditions for the indices in \mathcal{M} .
If so, then **Stop**.
 - 4: If $|\mathcal{B}| + 1 + \lceil \log |\mathcal{B}| \rceil \leq B_{\max}$, move the **worst** $1 + \lceil \log |\mathcal{B}| \rceil$ violating indices from \mathcal{M} into \mathcal{B} . Go to Step 2. Else, **Stop**.
 - 5: Those indices that have been at least l times in and out of the working set \mathcal{B} are moved permanently into \mathcal{B} .
-

summarized in Algorithm 3.2. Each step is explained in the following paragraphs.

In **Step 1**, a subset of the variables is chosen according to the indices in \mathcal{B} as illustrated in Figure 3.1 which follow the ideas of widely accepted QP-SVM methods presented in Chapter 2. Then we proceed to fix $\alpha_{\mathcal{M}} = \alpha_i^+ - \alpha_i^- = 0$, for all $i \in \mathcal{M}$. The problem is to find the variables $\alpha_{\mathcal{B}} = \alpha_i^+ - \alpha_i^-$, for all $i \in \mathcal{B}$; and then, since every fixed variable is associated with two constraints, these can be ignored under the assumption they are not support vectors, which results in the following sub-problem:

$$\begin{aligned}
 & \min_{\alpha_{\mathcal{B}}^+, \alpha_{\mathcal{B}}^-, b^+, b^-, \xi_{\mathcal{B}}, \mathbf{u}_{\mathcal{B}}} \quad \sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^- + 2C\xi_i) \tag{3.8} \\
 & \text{s.t.} \quad \left\{ \begin{array}{l} -\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) \\ \quad -b^+ + b^- - \xi_j + u_j = \epsilon - d_j \\ \sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) \\ \quad +b^+ - b^- - \xi_j + u_j = \epsilon + d_j \\ \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geq 0 \\ \text{for all } j \in \mathcal{B}. \end{array} \right.
 \end{aligned}$$

Let $\mathbf{z}_{\mathcal{B}}^* = (\alpha_{\mathcal{B}}^{+*}, \alpha_{\mathcal{B}}^{-*}, b^{+*}, b^{-*}, \xi_{\mathcal{B}}^*, \mathbf{u}_{\mathcal{B}}^*)$, denote the solution to linear programming problem (3.8). Problem (3.8) is a size-reduced version of problem (3.2), for which a comparison between LP problems is illustrated in Figure 3.3. The figure shows problem (3.2) in the left, and the reduced problem (3.8) is shown on the right.

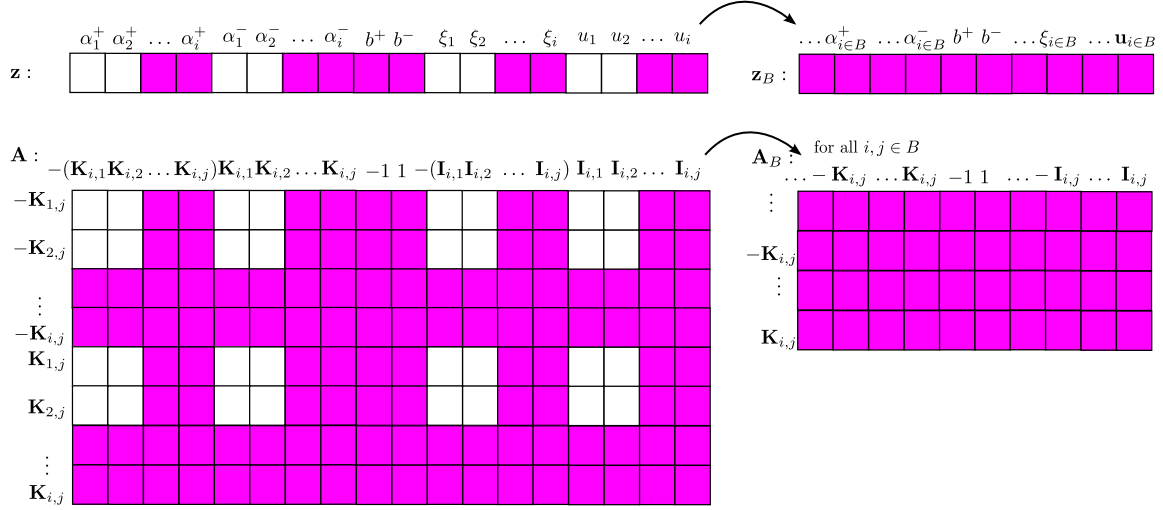


Figure 3.3: LP-SVR variable decomposition strategy. This illustrates a decomposition of the LP variables vector \mathbf{z} and coefficients matrix \mathbf{A} in which a subset of the variables α_i is considered for the solution of the LP-SVR problem, for all $i \in \mathcal{B}$. The shaded area corresponds to an arbitrary selection of $|\mathcal{B}|$ indices that produce a reduced LP: \mathbf{z}_B and \mathbf{A}_B .

Figure 3.3 shows the LP-SVR structure being exploited to reduce problem (3.2).

To derive the optimality conditions at a later stage, we obtain the following dual:

$$\begin{aligned}
 \min_{\lambda, \mathbf{s}} \quad & \sum_{i \in \mathcal{B}} \lambda_i (\epsilon - d_i) + \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} (\epsilon + d_i) \\
 \text{s.t.} \quad & \left\{ \begin{aligned}
 \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) - \sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) + s_j &= 1_j \\
 \sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) - \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) \\
 &\quad + s_{j+|\mathcal{B}|} = -1_j \\
 \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} - \lambda_i + s_{2|\mathcal{B}|+1} &= 1 \\
 \sum_{i \in \mathcal{B}} \lambda_i - \lambda_{i+|\mathcal{B}|} + s_{2|\mathcal{B}|+2} &= -1 \\
 -\lambda_i - \lambda_{i+|\mathcal{B}|} + s_{j+2|\mathcal{B}|+2} &= 2C_i \\
 \lambda_i + \lambda_{i+|\mathcal{B}|} + s_{j+3|\mathcal{B}|+2} &= 1_i \\
 \mathbf{s} &\geq \mathbf{0}
 \end{aligned} \right. \\
 & \text{for all } j \in \mathcal{B}
 \end{aligned} \tag{3.9}$$

and the KKT conditions (3.5a)-(3.5d) are rewritten as follows:

$$\sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) - \sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) + s_j = 1_j \quad (3.10a)$$

$$\sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) - \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) + s_{j+|\mathcal{B}|} = -1_j \quad (3.10b)$$

$$\sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} - \lambda_i + s_{2|\mathcal{B}|+1} = 1 \quad (3.10c)$$

$$\sum_{i \in \mathcal{B}} \lambda_i - \lambda_{i+|\mathcal{B}|} + s_{2|\mathcal{B}|+2} = -1 \quad (3.10d)$$

$$-\lambda_j - \lambda_{j+|\mathcal{B}|} + s_{j+2|\mathcal{B}|+2} = 2C_j \quad (3.10e)$$

$$\lambda_j + \lambda_{j+|\mathcal{B}|} + s_{j+3|\mathcal{B}|+2} = 1_j \quad (3.10f)$$

$$-\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) - b^+ + b^- - \xi_j + u_j = \epsilon - d_j \quad (3.10g)$$

$$\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) + b^+ - b^- - \xi_j + u_j = \epsilon + d_j \quad (3.10h)$$

$$s_j \alpha_j^+ + s_{j+|\mathcal{B}|} \alpha_j^- + s_{2|\mathcal{B}|+1} b^+ + s_{2|\mathcal{B}|+2} b^- \\ + s_{j+2|\mathcal{B}|+2} \xi_j + s_{j+3|\mathcal{B}|+2} u_j = 0 \quad (3.10i)$$

$$s_i, \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geq 0 \quad (3.10j)$$

for all $i, j \in \mathcal{B}$.

The primal sub-problem (3.8) has $4|\mathcal{B}| + 2$ variables and $2|\mathcal{B}|$ constraints; the dual sub-problem (3.9) has $2|\mathcal{B}|$ variables and $4|\mathcal{B}| + 2$ constraints; and the sub-problem KKT conditions (3.10a)-(3.10j) are necessary and sufficient for optimality as explained before.

In **Step 2**, we solve the sub-problem (3.8), (3.9), (3.10a)-(3.10j), *e.g.*, using LP interior point methods (IPM). In **Step 3**, we determine if problem (3.2),(3.4),(3.5a)-(3.5d) has been solved successfully. To do this, one needs to check the KKT conditions of the original problem. Since variables $\alpha_i^+, \alpha_i^- = 0$, for all $i \in \mathcal{M}$, and since we have a sub-problem solution, then, the values for the primal variables ξ_i and u_i

for $i \in \mathcal{M}$ can be estimated according to the following cases:

Case 1: When the following inequalities holds true for $j \in \mathcal{M}$:

$$-\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) - b^+ + b^- - \epsilon + d_j \geq 0 \quad (3.11a)$$

$$\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) + b^+ - b^- - \epsilon - d_j \geq 0, \quad (3.11b)$$

then, the values for the j -th index can be computed from (3.5b) as follows:

$$u_j = 2 \left(\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) + b^+ - b^- - d_j \right), \quad (3.12a)$$

$$\xi_j = 0. \quad (3.12b)$$

where \mathbf{u} is the vector of slacks that preserves the equality in the constraints. This means that if (3.11) are satisfied as equalities, the solution given by the sub-problem make the fixed set samples lie over the ϵ -tube. However, if (3.11) are not satisfied as equalities this means that the fixed set samples are outside the ϵ -tube region. The next case is the converse, which is desired: all the fixed set samples are within the ϵ -tube.

Case 2: When the inequalities (3.11) hold false, then the values for the j -th index are computed as follows:

$$u_j = 0, \quad (3.13a)$$

$$\xi_j = -2 \left(\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_j, \mathbf{x}_i) + b^+ - b^- - d_j \right). \quad (3.13b)$$

Next, the values for the dual variable are fixed $\lambda_i = 0$ for all $i \in \mathcal{M}$. Then, the values for the dual slack s_i for $i = 1, 2, \dots, 4|\mathcal{M}| + 2$ are estimated from (3.5a) as follows:

$$s_j = 1_j - \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) + \sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) \quad (3.14a)$$

$$s_{j+|\mathcal{B}|} = -1_j - \sum_{i \in \mathcal{B}} \lambda_i k(\mathbf{x}_j, \mathbf{x}_i) + \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} k(\mathbf{x}_j, \mathbf{x}_i) \quad (3.14b)$$

$$s_{2|\mathcal{B}|+1} = 1 - \sum_{i \in \mathcal{B}} \lambda_{i+|\mathcal{B}|} + \lambda_i \quad (3.14c)$$

$$s_{2|\mathcal{B}|+2} = -1 - \sum_{i \in \mathcal{B}} \lambda_i + \lambda_{i+|\mathcal{B}|} \quad (3.14d)$$

$$s_{j+2|\mathcal{B}|+2} = 2C_j + \lambda_j + \lambda_{j+|\mathcal{B}|} \quad (3.14e)$$

$$s_{j+3|\mathcal{B}|+2} = 1_j - \lambda_j - \lambda_{j+|\mathcal{B}|} \quad (3.14f)$$

for all $j \in \mathcal{M}$.

Note that KKT primal and dual conditions (3.5a)-(3.5d) have been already satisfied by (3.12), (3.13), and (3.14); however, the complementarity conditions have not. Then, we verify if the following conditions hold:

$$z_i s_i = 0 \quad (3.15a)$$

$$\mathbf{z}, \mathbf{s} \geq \mathbf{0} \quad (3.15b)$$

for all $i = 1, 2, \dots, 4|\mathcal{M}| + 2$. If there were no violations to (3.15), the global problem is said to be solved, and the method has converged for the set of parameters given. The global LP-SVR support vectors \mathbf{x}_i are those whose $(\alpha_i^+ - \alpha_i^-) \neq 0$ for all $i \in \mathcal{B}$.

In **Step 4**, we verify if there were any violations to (3.15); in such case, a new

working set is created. To do this, we look for inactive constraint indices (*i.e.*, those $(\alpha_i^+ - \alpha_i^-) = 0$ for all $i \in \mathcal{B}$) and move them into \mathcal{M} and then, we replace those indices with the indices that *most* violate the complementarity conditions (3.15) from \mathcal{M} into \mathcal{B} . By “*most* violation” we mean that the complementarity condition (3.15) has been sorted and we chose the largest violations first.

For practical purposes, a record is kept indicating which indices have been moved from \mathcal{M} into \mathcal{B} . In the case that all the constraints in \mathcal{B} are active (*i.e.*, all are support vectors), then, the size of \mathcal{B} is incremented by a scaling exponent as follows:

$$|\mathcal{B}|^{(t+1)} = \begin{cases} |\mathcal{B}|^{(t)} + 1 + \lceil \log |\mathcal{B}|^{(t)} \rceil, & \text{if } |\mathcal{B}|^{(t)} + 1 + \lceil \log |\mathcal{B}|^{(t)} \rceil \leq B_{\max} \\ B_{\max}, & \text{otherwise} \end{cases} \quad (3.16)$$

where B_{\max} is the maximum working set size allowed by the researcher, which is bounded to $B_0 < B_{\max} \leq |\mathcal{M}|$, and t is the current iteration at the variable decomposition strategy. Equation (3.16) is proposed here to smooth the increments in the working set size.

Steps 1-4 complete one iteration. These steps should be repeated until convergence. However, if after l iterations, the method has not converged, a check is performed to see if there are any indices that have been moving from \mathcal{M} into \mathcal{B} for at least l times. The constant l is an arbitrary parameter given by the researcher: typically $l = 10$.

Finally, in **Step 5**, we check if the condition $t > l$ holds true, if it does, the indices will be added to \mathcal{B} permanently, thus, preventing infinite loops.

The fact that our algorithm stops when the KKT conditions are satisfied guarantees the convergence to an optimal solution. Furthermore, our algorithm avoids a possible infinite loop by limiting indices from going in and out of the set \mathcal{B} for an undefined number of iterations. This guarantees that the algorithm will converge in a finite number of iterations. Of course, the solution will be sub-optimal if the algorithm stops when the maximum number of iterations t^{\max} is reached, or if the

Algorithm 3.3 Constraints Decomposition Strategy for Large-Scale LP-SVR Training: a modification of the algorithm introduced by Bradley and Mangasarian [22].

Require: LP-SVR with subset of variables: $\mathbf{c}_{\mathcal{B}}, \mathbf{b}_{\mathcal{B}}, \mathbf{A}_{\mathcal{B}}$.

Require: Parameters: t_{\max} , and τ .

- 1: Partition $(\mathbf{A}_{\mathcal{B}} \mid \mathbf{b}_{\mathcal{B}})$ into τ blocks with (3.21), where $1 < \tau < |\mathcal{B}|$. Then obtain constraint-sub-blocks $(\mathbf{A}_{\mathcal{R}} \mid \mathbf{b}_{\mathcal{R}})$.
- 2: For all indices in \mathcal{R} , **Solve LP** sub-sub-problem (3.22).
- 3: If $t \leq t_{\max}$, then go to Step 2 with a new block.
- 4: With (3.23) obtain $\mathbf{c}^T \mathbf{z}^{(t)}$ for \mathcal{B} .
- 5: If $\mathbf{c}^T \mathbf{z}^{(t)} = \mathbf{c}^T \mathbf{z}^{(t+t_{\max})}$, then **Stop**. Else, go to Step 2 with a new block.

Ensure: $\mathbf{z}^{(t)}$.

maximum working set size B_{\max} has been reached without a solution.

The next section explains the method for chunking the constraints.

3.4 LP-SVR Constraints Decomposition

Although the variable decomposition approach reduces the complexity of the linear program (LP) solution, the problem is likely to be increasing in size until it reaches the maximum working set size B_{\max} . As the LP size increases at each iterate the problem will become much slower proportionally to the current working set size $|\mathcal{B}|$. To overcome this difficulty, we present a modification of the constraint decomposition algorithm originally introduced by Bradley and Mangasarian [22]. In this algorithm which an LP-SVR sub-problem is solved using a subset of the constraints. This can be achieved by dividing the LP-sub-problem into blocks of smaller size and then we modify it such that the LP-SVR properties can be exploited one more time to further reduce the sub-problem.

The process of decomposing the constraints is illustrated in Figure 3.4 and summarized in Algorithm 3.3. The steps of Algorithm 3.3 will be explained in the following paragraphs, after we propose some definitions and equivalences.

We can pose problem (3.8) as an LP problem in the form of (2.18), by defining

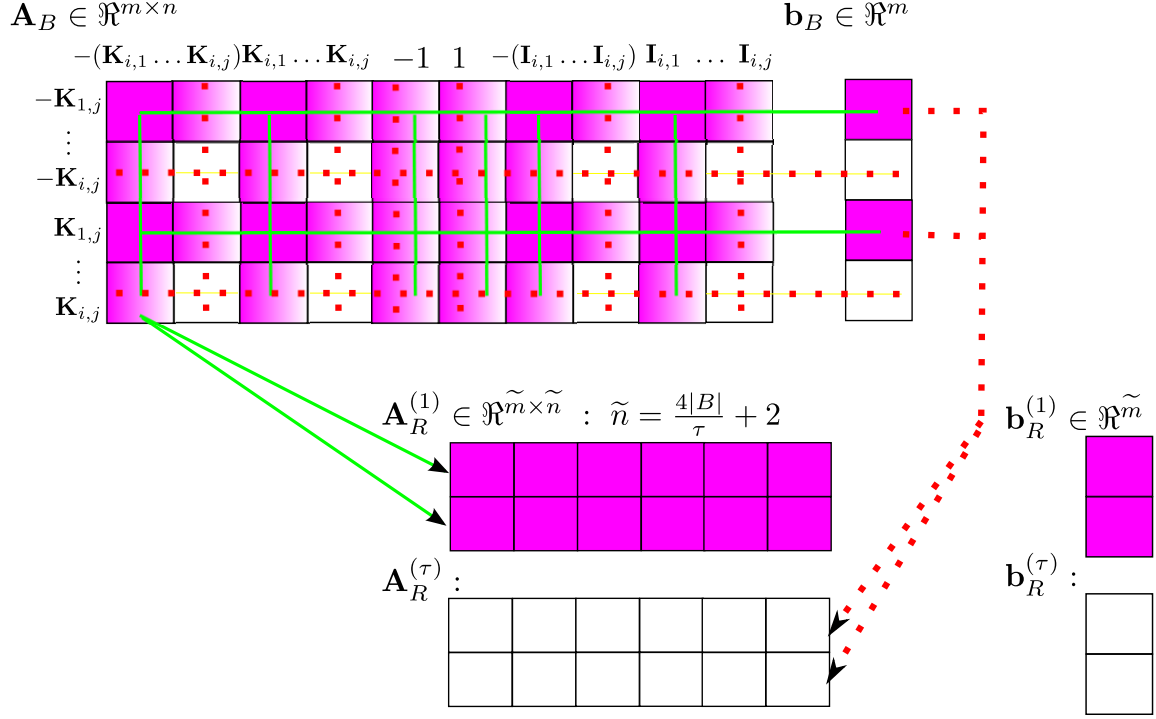


Figure 3.4: Proposed LP-SVR constraints decomposition strategy. The proposed decomposition strategy exploits the LP-SVR structure to further reduce the problem size. Note that the reduced problem size is inversely proportional to τ . The notation \mathbf{A}_{ij} refers to an element of the matrix \mathbf{A} at the i -row and j -th column.

the following equalities:

$$\mathbf{A}_B = \begin{pmatrix} -\mathbf{K}_B & \mathbf{K}_B & -1 & 1 & -\mathbf{I}_B & \mathbf{I}_B \\ \mathbf{K}_B & -\mathbf{K}_B & 1 & -1 & -\mathbf{I}_B & \mathbf{I}_B \end{pmatrix}, \quad (3.17a)$$

$$\mathbf{b}_B = \begin{pmatrix} \mathbf{1}_B \epsilon - \mathbf{d}_B \\ \mathbf{1}_B \epsilon + \mathbf{d}_B \end{pmatrix}, \quad (3.17b)$$

$$\mathbf{z}_B = \left(\boldsymbol{\alpha}^+_{\mathcal{B}} \quad \boldsymbol{\alpha}^-_{\mathcal{B}} \quad b^+ \quad b^- \quad \boldsymbol{\xi}_B \quad \mathbf{u}_B \right)^T, \quad (3.17c)$$

$$\mathbf{c}_B = \begin{pmatrix} \mathbf{1}_B & \mathbf{1}_B & \mathbf{0} & \mathbf{0} & 2\mathbf{C}_B & \mathbf{0}_B \end{pmatrix}^T, \quad (3.17d)$$

where $\mathbf{A}_B \in \mathbb{R}^{(2|B|) \times (4|B|+2)}$, $\mathbf{b}_B \in \mathbb{R}^{2|B|}$, $\mathbf{z}_B, \mathbf{c}_B \in \mathbb{R}^{4|B|+2}$. In this manner, problems (2.18) and (3.8) are identical, that is, we posed (3.8) as an LP problem.

Let \mathbf{z}_B be the unknown in the linear program given by $\min_{\mathbf{z}_B} \{\mathbf{c}_B^T \mathbf{z}_B\}$ subject to $\{\mathbf{A}_B \mathbf{z}_B = \mathbf{b}_B, \mathbf{z}_B \geq \mathbf{0}\}$; which is equivalent to (3.8). Then, allow the augmented matrix $\left(\mathbf{A}_B \mid \mathbf{b}_B \right)$ to be decomposed into the following τ blocks:

$$\left(\mathbf{A}_B \mid \mathbf{b}_B \right) = \begin{pmatrix} \mathbf{A}_B^{(1)} & \mathbf{b}_B^{(1)} \\ \mathbf{A}_B^{(2)} & \mathbf{b}_B^{(2)} \\ \vdots & \vdots \\ \mathbf{A}_B^{(\tau)} & \mathbf{b}_B^{(\tau)} \end{pmatrix}, \quad (3.18)$$

for all $1 < \tau < |B|$.

Let $t = 1, 2, \dots$, denote the current iteration and let $\mathbf{z}_B^{(t)}$ be the solution to the following linear program:

$$\mathbf{z}_B^{(t)} = \arg \min_{\mathbf{z}_B} \left\{ \mathbf{c}_B^T \mathbf{z}_B \mid \begin{array}{ll} \mathbf{A}_B^{(t \bmod \tau)} \mathbf{z}_B - \mathbf{b}_B^{(t \bmod \tau)} & = e_p \\ \bar{\mathbf{A}}_B^{(t \bmod \tau)-1} \mathbf{z}_B - \bar{\mathbf{b}}_B^{(t \bmod \tau)-1} & = \bar{e}_p \end{array} \right\} \quad (3.19)$$

where e_p , and \bar{e}_p are errors for the LP primal (3.19), $\left(\mathbf{A}_B^{(0)} \mid \mathbf{b}_B^{(0)} \right) = \emptyset$, and $\left(\mathbf{A}_B^{(t)} \mid \mathbf{b}_B^{(t)} \right)$ is the set of active constraints (*i.e.*, those equalities in (3.19) with $e_p, \bar{e}_p = 0$ for $\mathbf{z}_B^{(t)}$) with positive Lagrange multipliers. When $\mathbf{c}_B^T \mathbf{z}_B^{(t)}$ is equal to $\mathbf{c}_B^T \mathbf{z}_B^{(t+t_{\max})}$, then stop. Typically, the integer t_{\max} , controlling the stopping criteria, is set to $t_{\max} = 4$ or any small integer strictly positive [22].

As an example, let us consider $t = 1, \tau = 4$; at this point, $(1 \bmod 4) = 1$ and $(1 \bmod 4) - 1 = 0$, therefore, the LP in (3.19) only requires to satisfy the first part and not the second since it is empty by definition. In the next iteration $t = 2, \tau = 4$, $(2 \bmod 4) = 2$ and $(1 \bmod 4) - 1 = 1$, therefore, the LP in (3.19) requires both the

first and the second part to be satisfied; however, recall that in the previous iterations only active constraints were preserved for that block. Bradley and Mangasarian [22] demonstrated that this problem converges iteratively to the solution $\mathbf{z}_{\mathcal{B}}$ (See Theorem 3.2 in [22]).

However, for the proposed LP-SVR it is obvious that if the decomposition τ is carefully chosen, variables can be further reduced without affecting the solution at all. The key is for constraints associated with the same variable to be within the same block; when they are, all other variables can be omitted. Consider the constraints (3.17a) and consider the block-decomposition choosing the first $\lceil \frac{2N}{\tau} \rceil$ rows; then, the variables in the first $\lceil \frac{2N}{\tau} \rceil$ rows may have constraints related to the same variable in a different block. Therefore, it makes more sense (to our problem) to choose the blocks (3.18) in such a way that the properties of the proposed LP-SVR are exploited.

Clearly, if one uses (3.18) to select the elements of the τ -th block, the block itself can be further reduced by taking into account that variables associated with constraints that do not appear in the τ -th block do not play any role in the problem solution. To do so, define \mathcal{R} as the set of indices of variables associated with the τ -th block, where $\mathcal{R} \not\subseteq \mathcal{B}$ and $|\mathcal{R}| < |\mathcal{B}|$. Then, redefine (3.18) as follows:

$$\left(\mathbf{A}_{\mathcal{B}} \mid \mathbf{b}_{\mathcal{B}} \right) \equiv \left(\begin{array}{c|c} \mathbf{A}_{\mathcal{R}}^{(1)} & \mathbf{b}_{\mathcal{R}}^{(1)} \\ \hline \mathbf{A}_{\mathcal{R}}^{(2)} & \mathbf{b}_{\mathcal{R}}^{(2)} \\ \hline \vdots & \vdots \\ \hline \mathbf{A}_{\mathcal{R}}^{(\tau)} & \mathbf{b}_{\mathcal{R}}^{(\tau)} \end{array} \right), \quad (3.20)$$

for all $1 < \tau < |\mathcal{B}|$, where

$$\left(\mathbf{A}_{\mathcal{R}}^{(\tau)} \mid \mathbf{b}_{\mathcal{R}}^{(\tau)} \right) = \left(\begin{array}{ccccc|c} -\mathbf{K}_{i,j,\mathcal{B}} & \mathbf{K}_{i,j,\mathcal{B}} & -1 & 1 & -\mathbf{I}_{i,j,\mathcal{B}} & \mathbf{I}_{i,j,\mathcal{B}} & \mathbf{1}_{i,\mathcal{B}\epsilon} - \mathbf{d}_{i,\mathcal{B}} \\ \mathbf{K}_{i,j,\mathcal{B}} & -\mathbf{K}_{i,j,\mathcal{B}} & 1 & -1 & -\mathbf{I}_{i,j,\mathcal{B}} & \mathbf{I}_{i,j,\mathcal{B}} & \mathbf{1}_{i,\mathcal{B}\epsilon} + \mathbf{d}_{i,\mathcal{B}} \end{array} \right), \quad (3.21)$$

for all $i, j \in \mathcal{R}$, where $\mathbf{A}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{(2|\mathcal{R}|) \times (4|\mathcal{R}|+2)}$, $\mathbf{b}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{2|\mathcal{R}|}$, and consequently $\mathbf{z}_{\mathcal{R}}^{(\tau)}, \mathbf{c}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{4|\mathcal{R}|+2}$. This can greatly reduce the size of the problem since $|\mathcal{R}| < |\mathcal{B}|$.

Next, the LP solution needs to be redefined. At iteration $t = 1, 2, \dots$, let $\mathbf{z}_{\mathcal{R}}^{(t)}$ be the solution to the following linear program:

$$\mathbf{z}_{\mathcal{R}}^{(t)} = \arg \min_{\mathbf{z}_{\mathcal{R}}} \left\{ \mathbf{c}_{\mathcal{R}}^T \mathbf{z}_{\mathcal{R}} \mid \begin{array}{l} \mathbf{A}_{\mathcal{R}}^{(t \bmod \tau)} \mathbf{z}_{\mathcal{R}} - \mathbf{b}_{\mathcal{R}}^{(t \bmod \tau)} = e_p \\ \bar{\mathbf{A}}_{\mathcal{R}}^{(t \bmod \tau)-1} \mathbf{z}_{\mathcal{R}} - \bar{\mathbf{b}}_{\mathcal{R}}^{(t \bmod \tau)-1} = \bar{e}_p \end{array} \right\} \quad (3.22)$$

where e_p, \bar{e}_p are errors for the LP primal (3.22), $\left(\mathbf{A}_{\mathcal{R}}^{(0)} \mid \mathbf{b}_{\mathcal{R}}^{(0)} \right) = \emptyset$, and $\left(\mathbf{A}_{\mathcal{R}}^{(t)} \mid \mathbf{b}_{\mathcal{R}}^{(t)} \right)$ is the set of active constraints (*i.e.*, those equalities in (3.22) with $e_p, \bar{e}_p = 0$ for $\mathbf{z}_{\mathcal{R}}^{(t)}$) with positive Lagrange multipliers. Then, for the linear program (3.19), the solution $\mathbf{z}_{\mathcal{B}}^{(t)}$ at iteration t is given by

$$z_{i,\mathcal{B}}^{(t)} = \begin{cases} z_{j,\mathcal{R}}^{(t)} & \text{if } j = i, \text{ for all } j \in \mathcal{R} \\ 0 & \text{otherwise,} \end{cases} \quad (3.23)$$

for all $i \in \mathcal{B}$.

One of the main advantages is that the problem size can be very small, especially if the number of blocks is large. However, the size may increase as iterations progress. In the worst case scenario the size will be the same as in the traditional linear programming chunking (LPC) algorithm in which no SVR properties are exploited.

3.5 Convergence and Optimality Conditions

Algorithm 3.4 proposes the complete final algorithm that includes the combination of both variable and constraint decomposition.

As Algorithm 3.4 shows, the process of solving problem (3.2) involves the iterative sequential application of Algorithm 3.3, and Algorithm 3.2, until the KKT conditions (3.5a)-(3.5d) or a stopping criteria are satisfied. Clearly, if the KKT conditions are

Algorithm 3.4 Large-Scale LP-SVR Sequential Optimization

Require: LP-SVR problem data: \mathcal{T} .

Require: Parameters: $\epsilon, \sigma, C, B_0, B_{\max}, t_{\max}$, and τ .

1: For the initial and maximum working set size B_0 and B_{\max} , given a training set \mathcal{T} , execute **Algorithm 3.2** to obtain a working set \mathcal{B} and a fixed set \mathcal{M} .

- Initialize $t = 0$.
- For each LP problem given by the working set \mathcal{B} , divide in τ blocks executing **Algorithm 3.3**, $1 < \tau < |\mathcal{B}|$. **Solve LPs** $\mathbf{z}_{\mathcal{R}}$.
- If solution $\mathbf{z}_{\mathcal{B}}^{(t)}$ is steady for t_{\max} iterations, then go to Step 2. Else, solve with the next block.

2: If solution $\mathbf{z}^{(t)}$ satisfies KKT conditions (3.5a)-(3.5d) at iteration t , then **Stop**. Else, repeat Step 1 and update the working set \mathcal{B} and the fixed set \mathcal{M} .

Ensure: $(\mathbf{x}_i, \alpha_i, b)$ for all i such that $\alpha_i \neq 0$.

satisfied, the solution is guaranteed to be the optimal solution. Therefore, as long as Algorithms 3.3 and 3.2 *terminate* at an optimal solution, the complete process (Algorithm 3.4) also *converges to an optimal solution*.

To explain convergence and optimality of the decomposition algorithms, first, we discuss how Algorithm 3.3 terminates in a finite number of iterations; second, it follows to discuss how Algorithm 3.2 terminates in a finite number of iterations; and third, we discuss global convergence.

Finite Number of Iterations for Constraints Decomposition Algorithm

First, assume that the linear program $\min_{\mathbf{z}} \{\mathbf{c}^T \mathbf{z}\}$ subject to $\{\mathbf{A}\mathbf{z} = \mathbf{b}, \mathbf{z} \geq \mathbf{0}\}$ has a solution and all the sub-problems given by (3.19) and (3.22) also have solutions. Second, assume they satisfy the KKT conditions. Then, under these assumptions, Theorem 3.2 and Lemma 3.3 in [22] (see also Theorem 4 and Lemma 5 in [23]) hold true. This means that the sequence $\mathbf{z}^{(t)}$ generated by Algorithm 3.3 has the following properties:

1. The sequence $\mathbf{c}^T \mathbf{z}^{(t)}$ of the objective function values is non-increasing and is bounded by the global minimum of $\min_{\mathbf{z}} \{\mathbf{c}^T \mathbf{z}\}$ subject to $\{\mathbf{A}\mathbf{z} = \mathbf{b}, \mathbf{z} \geq \mathbf{0}\}$.

2. The sequence $\mathbf{c}^T \mathbf{z}^{(t)}$ of the objective function values becomes constant: $\mathbf{c}^T \mathbf{z}^{(t)} = \mathbf{c}^T \mathbf{z}^{t+1}$ for all $t \geq j$ for some $j \geq 1$.
3. For $t \geq j$, the active constraints in (3.19) at $\mathbf{z}^{(t)}$ with positive multipliers remain active for iteration $t + 1$.
4. For all $t \geq t_{\max}$, for some $t_{\max} \geq j$, $\mathbf{z}^{(t)}$ is a solution of the linear program $\min_{\mathbf{z}} \{\mathbf{c}^T \mathbf{z}\}$ subject to $\{\mathbf{A}\mathbf{z} = \mathbf{b}, \mathbf{z} \geq \mathbf{0}\}$, provided all active constraints at $\mathbf{z}^{(t)}$ have positive multipliers remain for $t \geq j$.

Finite Number of Iterations for Variables Decomposition Algorithm

Let $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ define a training with N samples. Then the number of variables and constraints in problem (3.2) is $4N + 2$ and $2N$ respectively. Now, following Torii, *et al.* algorithm in [187], we can define $\mathcal{B}^{(t)}$ as the working set at iteration t . Then, we let the following sequence

$$\mathcal{B}^{(t)} = \mathcal{B}^{(t+k+1)}, \quad \mathcal{B}^{(t+1)} = \mathcal{B}^{(t+k+2)}, \quad \dots, \quad \mathcal{B}^{(t+k)} = \mathcal{B}^{(t+2k+1)},$$

denote an infinite loop, since the same working set it is being repeated every k iterations. Now, suppose we use Algorithm 3.2, Step 5, to permanently add to the working set those constraints entering and leaving the working set for a number of iterations. Then, if for any reason, *e.g.*, the properties of the dataset samples, an infinite loop exists, it would be prevented at Step 5, since the constraints entering and exiting the working set by at least l times are added permanently to the working set, which implies that the infinite loop is broken. That is, the infinite loop will not occur at current $(t, t + k)$ or further iterations $(t + 1, t + k + 1)$. Moreover, since the number of both constraints and variables is finite, the number of infinite loops is also finite. Therefore, infinite loops are handled in finite steps. Hence, Algorithm 3.2 terminates in a finite number of iterations.

3.6 Interior Points Convergence and Optimality

Part of the computational robustness of the proposed decomposition methods rely on the usage of interior point methods (IPM) for linear programming (see Appendix C for a brief introduction to IPM, or the text [203] for a comprehensive review). Here we want to discuss shortly how the linear programs are being solved at the end.

First, let us consider the KKT conditions (3.5a)-(3.5d) established for our problem (3.2). Let us recall that the problem (2.18) is equivalent to (3.2) and that the KKT conditions (3.5a)-(3.5d) are also equivalent to (3.5a)-(3.5d). IPM considers the KKT conditions as the following function

$$\mathbf{F}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ \mathbf{A} \mathbf{z} - \mathbf{b} \\ \mathbf{X} \mathbf{S} \mathbf{1} \end{pmatrix} = \mathbf{0}, \quad (3.24a)$$

$$\mathbf{z}, \mathbf{s} \geq \mathbf{0} \quad (3.24b)$$

where $\mathbf{X} = \text{diag}(z_1, z_2, \dots, z_n)$, and $\mathbf{S} = \text{diag}(s_1, s_2, \dots, s_n)$. The IPM generates a set of solutions $\mathcal{F}^{(t)} = (\mathbf{z}^{(t)}, \boldsymbol{\lambda}^{(t)}, \mathbf{s}^{(t)})$ at each iteration t . The key idea is to find solutions $(\mathbf{z}^{(t)}, \boldsymbol{\lambda}^{(t)}, \mathbf{s}^{(t)})$ that satisfy $\mathbf{F}(\mathbf{z}^{(t)}, \boldsymbol{\lambda}^{(t)}, \mathbf{s}^{(t)}) = \mathbf{0}$ and more importantly $\mathbf{z}^{(t)}, \mathbf{s}^{(t)}$ being strictly positive, except at the solution where \mathbf{z} or \mathbf{s} may be equal to zero.

IPM surrounds the current point in a linear model in order to obtain the step direction $(\Delta \mathbf{z}, \Delta \boldsymbol{\lambda}, \Delta \mathbf{s})$ as follows:

$$\mathbf{J}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) \begin{pmatrix} \Delta \mathbf{z} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = -\mathbf{F}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}), \quad (3.25)$$

where $\mathbf{J}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$ is the Jacobian of $\mathbf{F}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$. Then the step direction (using a

predictor-corrector strategy) becomes

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{X} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{z} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{XS}\mathbf{1} - \Delta \mathbf{X}^{\text{aff}} \Delta \mathbf{S}^{\text{aff}} \mathbf{1} + \sigma \mu \mathbf{1} \end{pmatrix}, \quad (3.26)$$

where $\mathbf{r}_c = \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c}$ and $\mathbf{r}_b = \mathbf{A}\mathbf{z} - \mathbf{b}$ are residuals, $\Delta \mathbf{X}^{\text{aff}}, \Delta \mathbf{S}^{\text{aff}}$ are the affine-scaling direction, μ is the duality gap, and σ is an adaptive line-search parameter depending on μ . The new iterate is therefore

$$(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) + \alpha(\Delta \mathbf{z}, \Delta \boldsymbol{\lambda}, \Delta \mathbf{s}), \quad (3.27)$$

where $\alpha \in (0, 1]$ is appropriately chosen in order to maintain (\mathbf{z}, \mathbf{s}) strictly positive.

On The Convergence of IPM for LP

Let $\mathcal{F}^{(t)} = (\mathbf{z}^{(t)}, \boldsymbol{\lambda}^{(t)}, \mathbf{s}^{(t)})$ be the set of feasible solutions generated by Algorithm B.3 at iteration t . Under this definition, Zhang, Tapia, *et al.* [219], as well as [9, 95, 183, 218], demonstrate that IPM for LP exhibits the following properties: (i) $\mathcal{F}^{(t)}$ converges to \mathcal{F}^* ; (ii) the duality gap converges to zero $\mathbf{z}^{(t)T} \boldsymbol{\lambda}^{(t)} \rightarrow 0$ with **q -quadratic** behavior if all solutions are non-degenerate; and (iii) the duality gap converges to zero $\mathbf{z}^{(t)T} \boldsymbol{\lambda}^{(t)} \rightarrow 0$ with **q -superlinear** behavior if there is any degenerate solution.

The above properties demonstrate that LP-IPM is *q -quadratically* convergent to a feasible solution, *i.e.*, it is equivalent to the Newton method. Even in case of degeneracy, IPM is *q -superlinearly* convergent. In contrast, the simplex method is of exponential complexity. In spite of this, the simplex is typically used in most decomposition strategies for large-scale SVM.

Figure 3.5 shows the behavior of Primal, Dual, and Complementarity Condition using IPM for an arbitrarily three-class non-separable classification problem. The solution can easily be found in very few iterations.

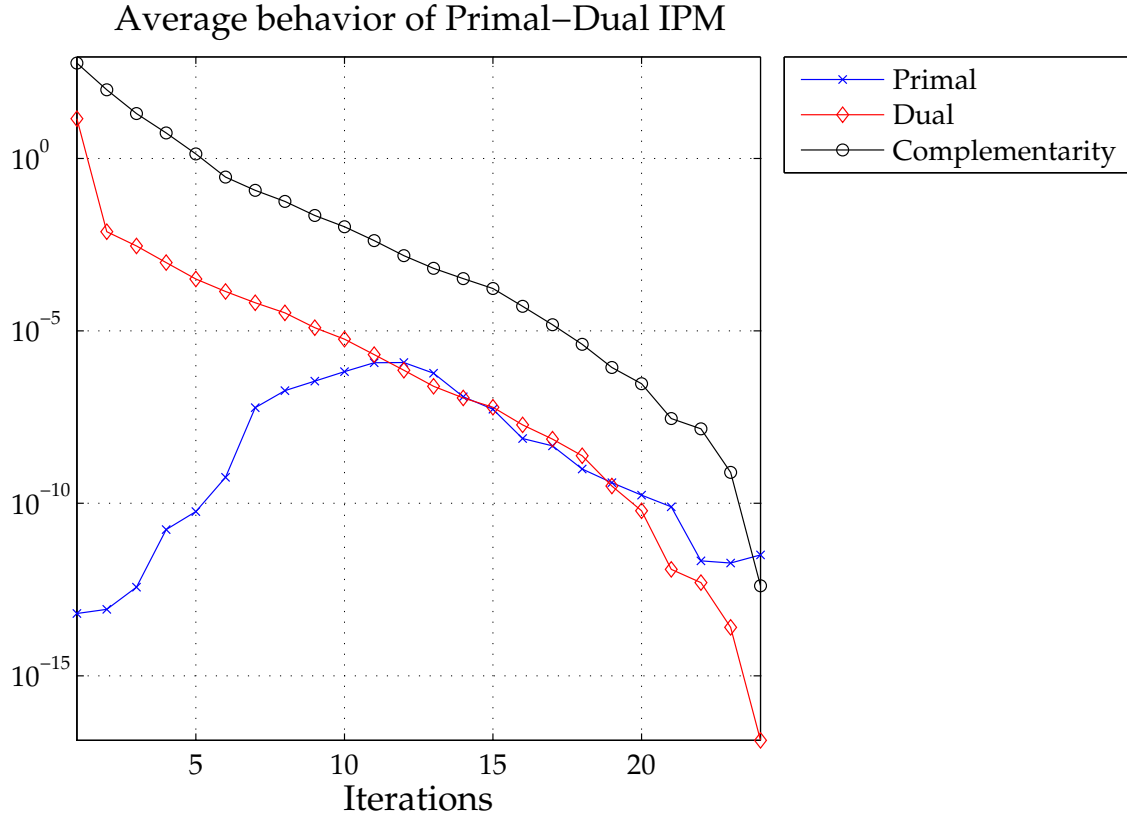


Figure 3.5: Behavior of the KKT conditions as the number of iterations progress. The primal, dual, and complementarity condition must converge to zero. The results shown represent the average value over several experiments on arbitrarily three-class non-separable classification problems.

3.7 Experimental Evaluation of LS LP-SVR

To show the effectiveness and efficiency of the proposed algorithms, simulations were performed over different datasets. The summary of the properties of these datasets are shown in Table 3.1. Note that the simulations include classification in two and multiple classes, as well as regression problems.

The well-known *Ripley* dataset problem [138,156] consists of two classes where the data for each class have been generated by a mixture of two Gaussian distributions.

The *Wine* dataset [64,107] contains results of wine chemical analysis within the

Table 3.1: Summary of the Dimensions and Properties of the Datasets.

Dataset	Classes	Features M	Training N	Testing N^*	Reference
Ripley	2	2	250	1,000	[138]
Wine	2	13	110	20	[107]
ADA	2	48	4,147	415	[98, 148]
GINA	2	970	3,153	315	[39, 148]
HIVA	2	1,617	3,845	384	[26]
NOVA	2	16,969	1,754	175	[26]
SYLVA	2	216	13,086	1,308	[26, 39]
Iris	3	4	130	20	[128]
Spiral	2	2	200	101	[206]
Synthetic S	3	2	3 million	3 million	—
Synthetic NS	3	2	3 million	3 million	—
$f(x) = \text{sinc}(x)$	\mathbb{R}	1	200	200	[144]
$f(x) = \text{sinc}(x) \times \pi$	\mathbb{R}	1	1 million	1 million	—

Italy region but was derived from three different vines. The analysis consists of 13 attributes of two different groups of wine. This dataset is part of the UCI machine-learning repository [65].

ADA is a marketing-related dataset [108]. The goal of ADA is to discover high revenue people from census data. This is a two-class classification problem. The raw data from the census bureau is known as the Adult database [98, 148] in the UCI machine-learning repository [65]. The 48 features include age, workclass, education, education, marital status, occupation, native country, etc.

GINA is a digit recognition-related dataset that is commonly known as the MNIST database of handwritten digits [112]. GINA aims to provide features for handwritten digit recognition [39, 148]. The problem consists of separating two-digit odd numbers from two-digit even numbers. Only the unit digit is informative for that task; therefore, at least one half of the features are distractors. Additionally, the pixels that are almost always blank were removed and the pixel order was randomized to hide the feature identity. This is a two class classification problem with

non-sparse continuous input variables, in which each class is composed of several clusters. It is a problem with heterogeneous classes.

HIVA is a dataset related to HIV infections. The goal of *HIVA* is to provide features for prediction of active compounds within an AIDS HIV infection. The dataset represents a two-class classification problem (active vs. inactive) consisting of 2000 sparse binary input variables. The variables represent properties of the molecule inferred from its structure. The problem is to relate structure to activity *i.e.*, a quantitative structure-activity relationship (QSAR) problem, to screen new compounds before actually testing them, *i.e.*, a high-throughput screening (HTS) problem. The data is made available by the National Cancer Institute (NCI), and has been used in [26].

NOVA is a text classification dataset. The data of *NOVA* come from the UCI repository [65] which is also known as Twenty-Newsgroup dataset [97]. Each text to classify corresponds to email text. The features consist of a sparse binary representation of a vocabulary of approximately 17,000 words.

SYLVA is an ecology-related dataset that is part of the UCI repository [65] under the name of Covertype Data Set [17,39]. The *SYLVA* dataset aims to provide features for forest cover type classification. The data is obtained from 30×30 meter cells by the US Forest Service (USFS) Region 2 Resource Information System (RIS). It represents a two-class classification problem, that classifies Ponderosa pine against everything else. The features consists of 216 input variables. Each pattern is composed of four records: two records matching the target and two records chosen at random. Thus one half of the features are distractors.

The *Iris* dataset is perhaps the best known database to be found in the pattern recognition literature and is also part of the UCI dataset [65]. The dataset contains three classes of 50 instances each, where each class refers to a type of Iris plant. One of the classes is linearly separable from the rest, which are not linearly separable [63,77].

The *Spiral* dataset is a synthetic dataset that consist of a two class problem with

an extremely non-linear decision surface, and is typically used to test the ability of classifiers in finding such difficult decision functions [206].

Similarly, the remaining datasets are synthetic. The *Synthetic S* is a non-linearly separable three-class problem whose classes are normally distributed. The *Synthetic NS* is identical, however, the classes are non-separable.

The two last examples are for regression purposes. The datasets objective is to fit the “sinc” function, which is a typical function to approximate [144]. The $f(x) = \text{sinc}(x)$ consists of unevenly sampled points from the sinc function. Similarly, $f(x) = \text{sinc}(x) \times \pi$ consists of unevenly spaced points from the sinc function that are affected by multiplicative white Gaussian noise (WGN); this makes it a very difficult function to fit.

The results of training and testing with the above datasets will be explained in the following paragraphs. The kernel choice in this dissertation is only the radial basis function (RBF) kernel. This choice is justified since RBF kernels are the most discriminant kernels [15, 73] and since RBF kernels produce very large datasets, fitting perfectly with what we intend to demonstrate in this research.

This section is divided in two different analysis: The first part analyzes the performance metrics of the proposed model against state of the art models using benchmarking datasets; while the second part analyzes the sparseness of the proposed model against state of the art models using benchmarking datasets.

These experiments show results using a testing set $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N^*}$, where N^* is the number of samples available for testing. The testing set \mathcal{D} has never been shown to the LP-SVR model before.

3.7.1 Large-Scale Learning

Figure 3.6 shows how the total training time (in minutes) varies as the problem size increases. The figure shows that the proposed approach is slower at smaller problems, and seems to be comparable to medium size problems; however, it is faster at larger

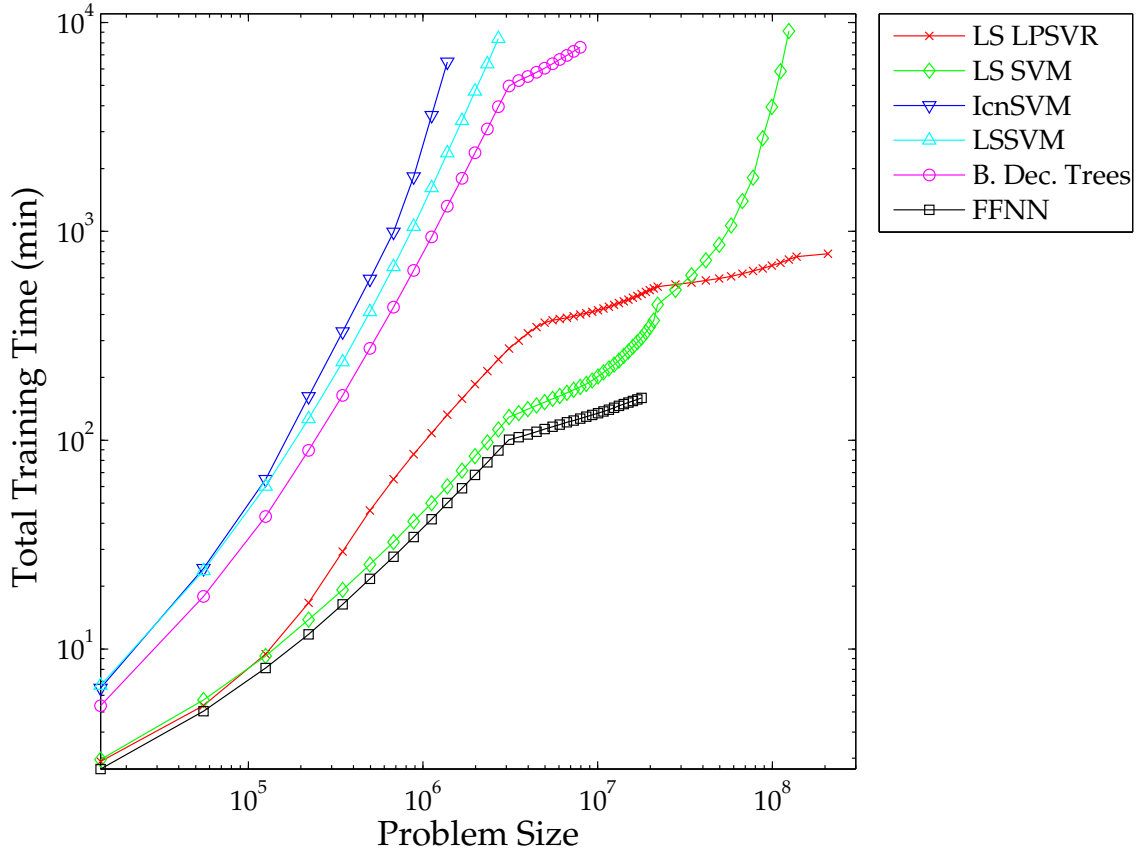


Figure 3.6: Training Time as a Function of the Problem Size.

problems.

Figure 3.7 shows the total number of iterations in relationship with the problem size. It is important to observe that at larger scale problems, the proposed model has a very consistent number of iterations. The reason is, that the majority of the iterations are made within the constraints decomposition strategy. Although the number of iterations seems to be constant, the effect of larger constraint decompositions is clearly expressed as an increase in total CPU time, as shown in Figure 3.6.

Figure 3.8 shows the monotonic non-increasing behavior of the objective function as iterations progress. As explained before, the proposed sequential optimization has the property of a never increasing objective function. Clearly, most small size and small number of classes are solved in very few iterates. As problem size and number

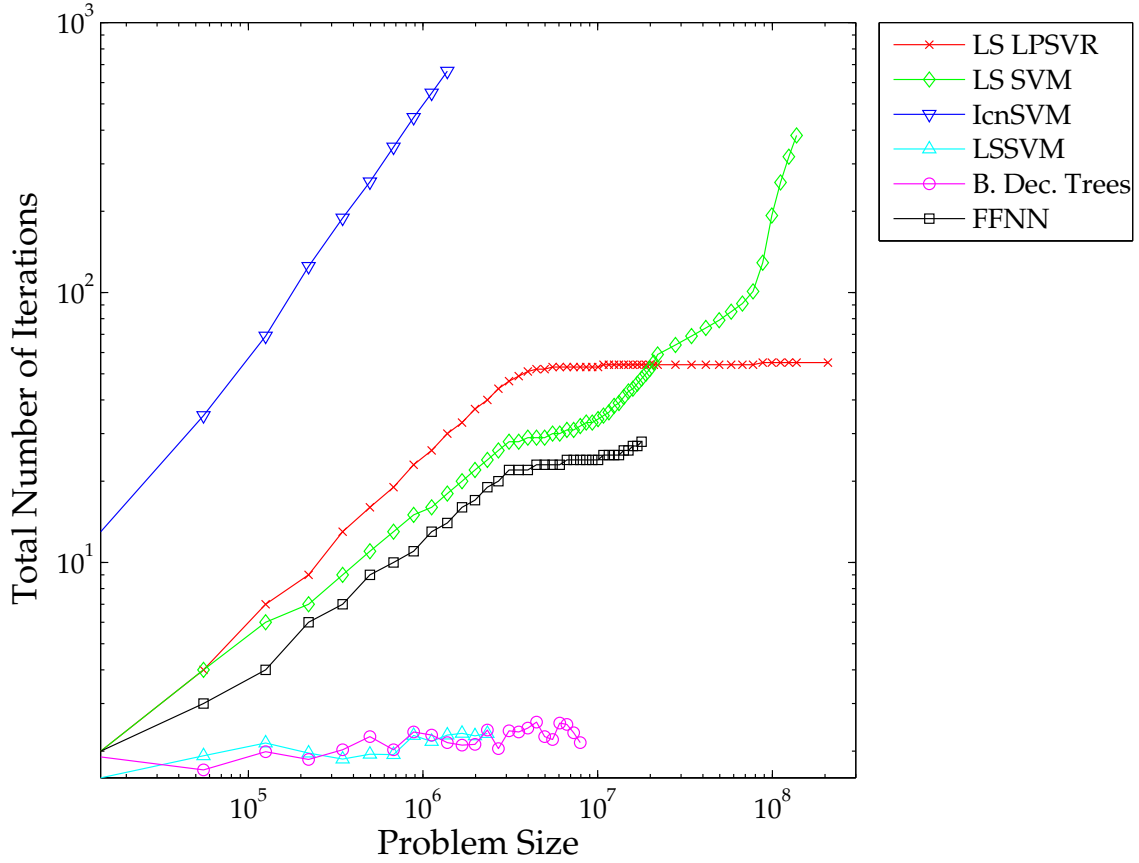


Figure 3.7: Total Number of Iterations as a Function of the Problem Size.

of classes increase the model takes more iterations to solve the problem. However, the objective function is always monotonically decreasing.

Table 3.2 shows a comparison of the accuracy. It can be observed that the proposed approach has comparable results to other SV-based and neural approaches. As problem size increases, a slight increase in accuracy can be observed. In average, the proposed approach possesses a slightly higher accuracy.

In the other hand, Table 3.3 shows a comparison of the balanced error rate. Whereas the overall accuracy can be biased if classes are unbalanced, the BER metric offers a “fair” estimate of error. It can be seen that the proposed LP-SVR learning model does not affect dramatically the performance of the classifier. On the contrary, we can observe that the error has no significant difference as compared to others in

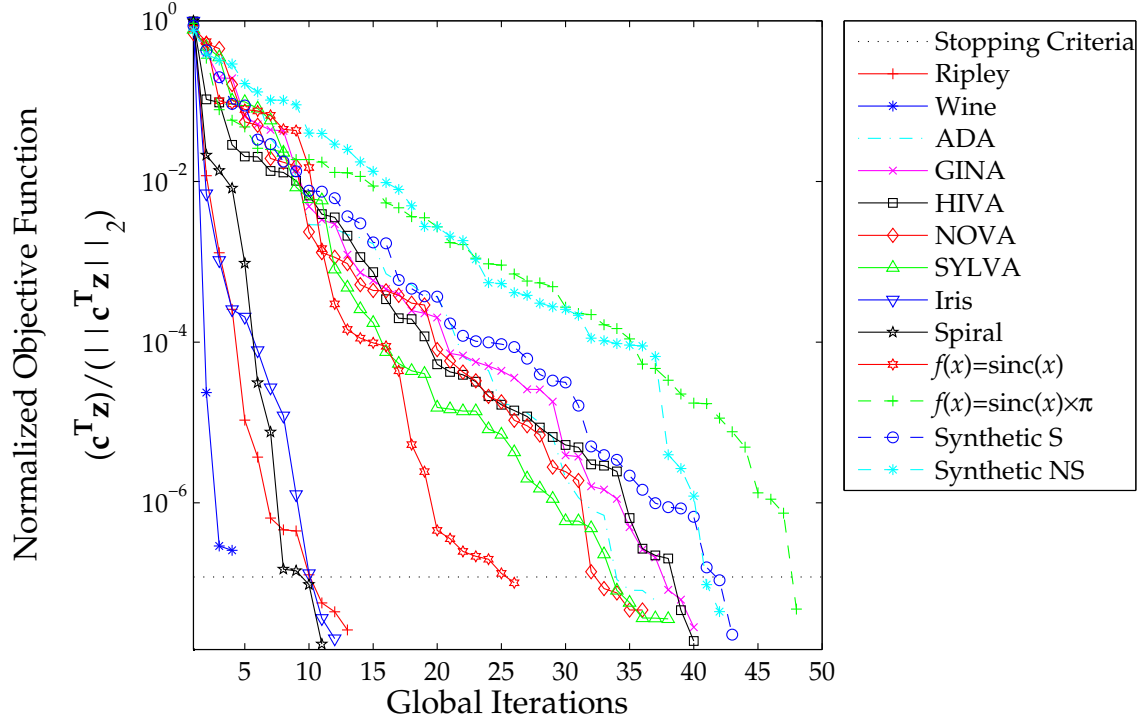


Figure 3.8: Objective Function as a Function of Iterations Number.

Table 3.2: Accuracy

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.916	0.915	0.91	0.906	0.908
Wine	1	1	1	1	0.99
ADA	0.841	0.851	—	0.843	0.839
GINA	0.997	0.997	—	0.99	0.997
HIVA	0.87	0.87	—	—	—
NOVA	1	1	1	—	—
SYLVA	0.998	0.999	—	—	0.996
Iris	1	1	0.998	0.998	0.998
Spiral	1	1	1	1	0.99
Synthetic S	0.989	0.99	—	—	—
Synthetic NS	0.984	0.985	—	—	—
Avg.	0.963	0.964	—	—	—

Table 3.3: Balanced Error Rate

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.084	0.085	0.09	0.094	0.092
Wine	0	0	0	0	0.008
ADA	0.135	0.148	—	0.15	0.15
GINA	0.003	0.003	—	0.009	0.003
HIVA	0.171	0.171	—	—	—
NOVA	0	0	0	—	—
SYLVA	0.001	0.006	—	—	0.014
Iris	0	0	0.001	0.001	0.001
Spiral	0	0	0	0	0.01
Synthetic S	0.011	0.01	—	—	—
Synthetic NS	0.016	0.015	—	—	—
Avg.	0.038	0.039	—	—	—

the average case.

For regression cases, Table 3.4 is presented, which shows a comparison of the mean absolute error among neural, binary, and SV-based learning methods. The

Table 3.4: Mean Absolute Error.

	Classifiers			
Dataset	LS SVR	LS LPSVR	B. Dec. Trees	FFNN [5, 20, 2]
Mean Absolute Error				
$f(x) = \text{sinc}(x)$	0.000938	0.000938	0.000948	0.000808
$f(x) = \text{sinc}(x) \times \pi$	0.093108	0.091457	0.093721	0.092401

table elucidates the idea that the proposed scheme produces results comparable to other SV-based approaches. The proposed model shows smaller errors even when compared with the neural network-based approach at a large-scale.

The proposed LP-SVR model with the large-scale training strategy was also evaluated using other performance metrics. The results lead to the same conclusions as those tables shown in this section. However, those results with other performance

metrics are included in Section C.4 as a reference.

3.7.2 Solution Sparseness

Figure 3.9 shows an analysis of the support vectors across iterations number. The

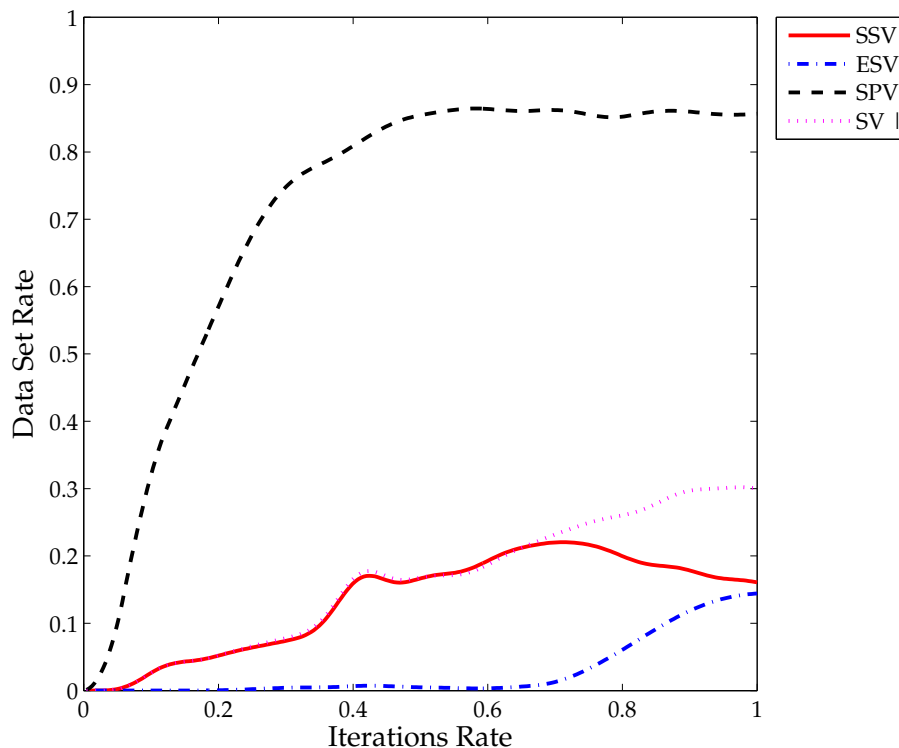


Figure 3.9: Support Vectors as a Function of Iterations. SPV SSV ESV SV.

figure shows the number of: support vectors (SVs), sparse support vectors (SPVs), saturated support vectors (SSV), and exact support vectors (ESV). The horizontal axis indicates that the total number of iterations of each case was expressed as a percentile in order to average all the cases; similarly, the vertical axis shows the data set as a percentile for averaging purposes. The figure can be interpreted as the proportion of the total dataset that is changing in proportion to the total number of iterations. It can be seen that the SPVs are being progressively found as iteration progress, and at final iterations the ESVs increase while the SSVs decrease.

Table 3.5 shows a comparison of the number of support vectors. It can be con-

Table 3.5: Support Vectors

Dataset	SV-Based Classifiers			
	LS SVM	LS LPSVR	IncSVM	LSSVM
Ripley	0.31	0.14	0.31	1.00
Wine	0.29	0.34	0.29	1.00
ADA	0.36	0.18	—	1.00
GINA	0.14	0.14	—	1.00
HIVA	0.34	0.34	—	—
NOVA	0.14	0.14	0.14	—
SYLVA	0.13	0.06	—	—
Iris	0.26	0.24	0.26	1.00
Spiral	0.36	0.36	0.36	1.00
$f(x) = \text{sinc}(x)$	0.34	0.34	0.33	1.00
$f(x) = \text{sinc}(x) \times \pi$	0.03613	0.00733	—	—
Synthetic S	0.00004	0.00001	—	—
Synthetic NS	0.00018	0.00004	—	—

cluded from this table, that the proportion of the support vectors is less at larger problems, as expected. However, in smaller problems the results are similar. In the case of the Least-Squares SVM (LSSVM) [26, 199] the total number of support vectors is equivalent to the total dataset, *i.e.*, all samples are support vectors. Similarly, the results shown in Table 3.6 compare the number of sparse support vectors, $\{\mathbf{x}_i : d_i - \epsilon < f(\mathbf{x}_i) < d_i + \epsilon\}$. In this case, we observe that as problem size increases, we have much less proportion of SSVs, which is desired. At smaller problems, the number of support vectors might be, sometimes, equivalent to the total training set, depending on the separability of the data and the number of samples.

The appendix contains two more tables: Table C.22 that shows a comparison of the number of exact support vectors and Table C.23 that compares the number of saturated support vectors. The information in this tables is consistent with the tables shown here.

Table 3.6: Sparse Support Vectors

Dataset	SV-Based Classifiers			
	LS SVM	LS LPSVR	IncSVM	LSSVM
Ripley	0.84	0.37	0.84	1.00
Wine	0.77	0.78	0.81	1.00
ADA	0.99	0.50	—	1.00
GINA	0.38	0.38	—	1.00
HIVA	0.93	0.93	—	—
NOVA	0.39	0.39	0.40	—
SYLVA	0.35	0.15	—	—
Iris	0.65	0.68	0.65	1.00
Spiral	0.98	0.98	0.97	1.00
$f(x) = \text{sinc}(x)$	0.92	0.91	0.91	1.00
$f(x) = \text{sinc}(x) \times \pi$	0.09906	0.02011	—	—
Synthetic S	0.000099	0.000020	—	—
Synthetic NS	0.000497	0.000109	—	—

3.7.3 Computational Concerns

The experiments that support the theoretical development presented in this dissertation arose some important implementation considerations.

The first consideration is in regard to global convergence of the proposed algorithms. In order to obtain a global solution, it is required that interior point methods converge. Then, the constraints decomposition algorithm must also converge to the optimal solution. Finally, the variable decomposition has to terminate in finite time. The conditions that guarantee this, is that every sub-problem leads to a feasible solution. In classification problems, the keys for a feasible solution are first to avoid under-determined coefficient matrices \mathbf{A} within any LP sub-problem, and second to make sure that there exists at least one feature vector per class represented in any LP-sub-problem [200]. In the case of regression, one needs to have at least three feature vectors representing of the pattern to be fit; otherwise, the problem becomes trivial, *i.e.*, a non-linear regression model trying to solve a simple N -dimensional line equation.

In the experiments of this dissertation, the maximum number of samples we worked with was $N = 3,000,000$ which leads to a problem size of $(2N) \times (4N + 2) = 72,000,012,000,000$ points. Computational limitations constrained the author of this dissertation in performing more comparisons over larger datasets. However, it is evident that the proposed scheme will still be able to handle larger datasets, *i.e.*, larger problems are still computationally tractable, unlike other large-scale approaches, such as the Incremental SVM (IncSVM) [209], the linear programming chunking [22, 23], or the quadratic programming chunking (LS SVM) [39].

In this dissertation we only covered the radial basis function (RBF) kernel. There are two main reasons for this: (i) RBF kernels are unarguably the most discriminant kernels for problems for whose one has very little or no-prior information [15, 73]; (ii) RBF kernels produce very large datasets, thus, they fit perfectly within the scope of this dissertation.

One final consideration we want to leave as future work is the usage of different loss functions apart from the ϵ -insensitive loss function. This function was adopted initially for computational cost reasons. However, other functions (*e.g.*, Huber loss function [88, 132] or quadratic loss function [83]) may improve the performance of the regression model.

3.8 Conclusion

3.8.1 Large-Scale Learning

The proposed large-scale model involved three major parts that increases computational tractability of large-scale problems.

First, the large-scale LP-SVR problem is reduced by variable decomposition a.k.a. column/variable chunking which exploits LP-SVR structure to produce a lower-dimensional representation of the decomposed LP sub-problem. The proof of

finite termination of the decomposition strategy is guaranteed by an infinite-loop prevention algorithm and by previous work from Torii, *et al.* in [187].

Second, we take a constraint decomposition strategy a.k.a. row/constraint chunking that generates a number of smaller sub-problems by exploiting again LP-SVR structure. The resulting LP problems are solved sequentially in a monotonically non-increasing objective function fashion. Proof of convergence is given applying Bradley’s [23] theorems under the correct assumptions.

Third, the two-way-reduced LP-SVR sub-problems are solved using interior point methods, which have a very high rate of convergence. This is the key that balances the total computational time of performing two decompositions and solving several LPs. Using other approaches (*e.g.* the simplex method) would dramatically increase computational efforts.

3.8.2 Model Performance and Sparseness

Finally, from the experiments results shown in this chapter we can conclude that the proposed approach is comparable with other formulations in terms of performance, which is desirable. However, in this work we are not looking for better performances, instead, we want to train large-scale datasets and at the same time produce sparser solutions in terms of the number of support vectors. As is known, if the solution is sparser, the model is more efficient in testing phase. That is, a sparse solution implies fewer number of support vectors and hence, future kernel evaluations can be made faster. The point here is that experimental results shown that the model is sparser than the regular SVR and other SV-based classifiers. Although this was studied by Zhang, *et al.* [214], we have experimentally demonstrated that as the problem size increases, the sparser the solution becomes.

The next chapter discusses the possibility of accelerating training time by pre-selecting the feature vectors that may have a higher chance of becoming support vectors, and also proposes a method for selecting model parameters.

Chapter 4

LP-SVR Training Speedup and Hyper-Parameters Estimation

4.1 Introduction

This chapter presents a learning speedup method based on the relationship between the support vectors and the within-class Mahalanobis distances among the training set. We explain how statistical properties of the data can be used to pre-rank the training set. Then we explain the relationship among the pre-ranked training set indices, convex hull indices, and the support vector indices. We will also explain how this method has better efficiency than those approaches based on the convex hull, especially, at large-scale problems.

This chapter also presents a method for estimating the Linear Programming Support Vector Regression (LP-SVR) model *hyper-parameters*, $\boldsymbol{\theta} = [C, \sigma]$. We start by defining error functions for classification and regression. Then we adapt the Newton method for function minimization and use it to find a $\boldsymbol{\theta}$ that minimizes a vector of error functions. We show how the proposed model provides a better estimate of the hyper-parameters than those approaches based on brute force search.

At the end of the chapter we conclude by explaining the findings of the experimental results over both the speedup alternative and the hyper-parameter estimation approach. Background material on the Newton method has been prepared for the reader unfamiliar on this technique. Such material is located in Appendix D.

4.2 Within-Class Distances for Learning Speed Up

Although the proposed algorithm for large-scale training of an LP-SVR has been mathematically proven to converge, the complete process might take a large amount of time to be resolved as N increases. This motivates, the exploration of alternatives for speeding up the learning process. Here, it is proposed to use an distance measure to rank the training set for speeding up the learning process.

It is known by the community [12,13,98,104,120,138,197,221] that in classification tasks, those data points closer to the decision boundaries (*i.e.*, convex hull of the data class) are more likely to be support vectors, as illustrated in Figure 1.2 and 4.1 with bold circles. Figure 4.1 depicts the relationship between convex hull and maximum Mahalanobis distance (MD) for an arbitrary two class problem. The figure shows the convex hull of each class, the support vectors (SVs) that define the separating hyperplane, and the eight samples having the maximum MD from their class center. Note that both SVs and convex hull match the eight samples with maximum MD.

Vapnik [197, 198] improved the speed of his learning method by considering only those variables on the boundaries of the feasible region instead of considering all the data, which allowed computational tractability of some problems. Then Joachims [98] defined a heuristic approach to identify variables at boundaries based on Lagrange multiplier estimates. Later, Bennett, *et al.* [12,13] posed the problem of finding the optimal separating hyperplane using the distance between class convex hulls. A similar concept was followed by Keerthi, *et al.* [104] in 2002, by Osuna, *et al.* [138] in 2003, and by Zhenbing, *et al.* [120] in 2010.

In this chapter we introduce a related approach which uses a probabilistic argument. In the following discussion we assume that our training data $\mathcal{T}_\phi = \{\mathbf{x}_i, d_i\}_{i=1}^N$ has been taken to the kernel-induced feature space.

It is well known that SVR and support vector machines (SVM) formulations do not make assumptions about the probability distribution of the data. Nonetheless,

Convex Hull, SVs, and Max Mahalanobis Distance

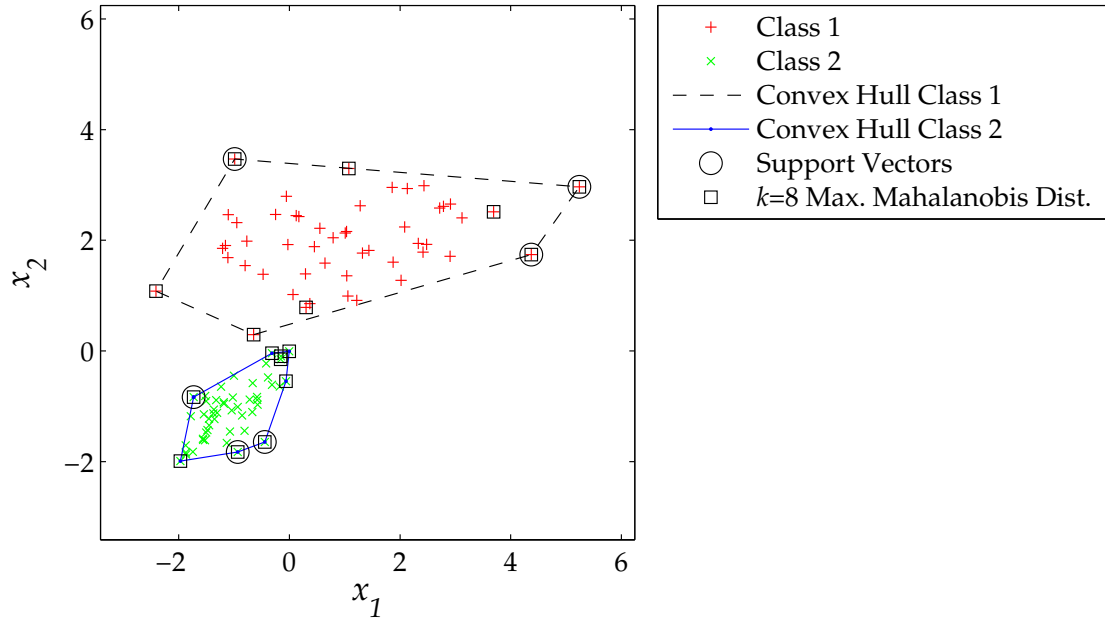


Figure 4.1: Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and $k = 8$ maximum Mahalanobis distance samples. Note that both SVs and Convex Hull match the $k = 8$ maximum Mahalanobis distance samples.

each class ω_j , should have a conditional class distribution $p(\mathbf{x}|\omega_j)$, where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^M$ is defined as an M -dimensional random variable which could be estimated if enough data points were available. Estimating a multidimensional PDF is difficult but we could make some basic assumptions. First, we could assume that the data has a uni-modal distribution which implies that data-samples would cluster around the class mean and the further a point is from its mean, the lower its probability and could be expected to be located on the convex hull of the data sample we are analyzing. A more strict assumption would be to consider that the $p(\mathbf{x}|\omega_j)$ are multivariate Gaussian distributed. Under this assumption each $p(\mathbf{x}|\omega_j)$ could be modeled using only the sample mean $\boldsymbol{\mu}_{\mathbf{x}|\omega_j}$ and a covariance matrix $\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j}$, that

is $p(\mathbf{x}|\omega_j) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$. It is also well known that we can use the squared Mahalanobis distance (MD)

$$D(\mathbf{x}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})^T \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j}), \quad (4.1)$$

as a measure of the distance of a data point with respect to its mean.

Based on these assumptions we propose a method for finding the SV candidates by computing the $D(\mathbf{x}_i)$ for all $i = \{1, 2, \dots, N\}$. Once all training vectors are sorted by their MD to their respective mean, and saved into the sets \mathcal{Z}_j for the j -th class, then we can form the initial working set \mathcal{B} of size B_{ini} using the procedure described in Algorithm 4.1 (explained in the next section). We traverse elements of $\mathcal{Z}_{j,i}$ into to \mathcal{B} until B_{ini} elements are added.

Algorithm 4.1 Mahalanobis Distance-Based Working-Set Selection for Large-Scale LP-SVR Training Speedup

Require: A training set $\mathcal{T}_\phi = \{\mathbf{x}_i, d_i\}_{i=1}^N$.

Require: A desired number of samples per class v .

```

1: for  $j = 1$  to  $|\mathcal{D}|$  do
2:   Estimate parameters  $(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$ . ▷ Sample mean and variance.
3:   for  $i = 1$  to  $N$  do
4:     Compute Mahalanobis distance  $D(\mathbf{x}_i)_j$  with (4.1).
5:   end for
6:   Obtain indices  $\mathcal{Z}_j$  corresponding to the sorted  $D_j$ . ▷ Descending order.
7:   for  $i = 1$  to  $v$  do
8:      $\mathcal{Z}_{j,i} = \mathcal{Z}(i)_j$ . ▷ In this case  $B_{\text{ini}} = k \equiv v \times |\mathcal{D}|$ .
9:   end for
10: end for
```

Ensure: Initial working set indices $\mathcal{B} \leftarrow \mathcal{Z}_{j,i}$.

Ensure: Initial fixed set indices $\mathcal{M} \leftarrow \{1, 2, \dots, N\} \setminus \mathcal{Z}_{j,i}$.

In this manner, the SVR could be trained faster if the first working-set \mathcal{B} contains those k samples, thereby, speeding up the training process. A similar approach to ours is given by Zhou, *et al.* [221] in 2010; but again the authors approach is still based on class and subclass convex hulls, which makes it computationally expensive.

To explain the proposed approach, consider the following definitions: Let $\mathcal{D} = \{\omega_1, \omega_2, \dots, \omega_j\}$ be the set of classes where j is the total number of classes. Let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_j\}$ denote a set of indices, where \mathcal{C}_j contains the indices of all those samples associated with the j -th class, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$ for all $i \neq j$, and $\mathcal{C} \equiv \{1, 2, \dots, N\}$.

4.2.1 Within-Class Mahalanobis distance and Class-Convex Hull

To explain the ideas behind the procedure shown in Algorithm 4.1, we will be considering the case of all the samples \mathbf{x}_i belonging to the j -th class, that is, all $i \in \mathcal{C}_j$. The same principles will apply to all classes.

One of the first steps is to estimate the parameters $(\boldsymbol{\mu}_{\mathbf{x}|\omega_j}, \boldsymbol{\Sigma}_{\mathbf{x}|\omega_j})$, *i.e.*, from observed events. Then the within-class MD from the i -th feature vector \mathbf{x}_i to the center of the j -th class $\boldsymbol{\mu}_{\mathbf{x}|\omega_j}$ is defined as $D(\mathbf{x}_i)$ from (4.1). Next, we define \mathcal{Z}_j as the set of indices corresponding to the ordered Mahalanobis distance samples of the j -th class computed with (4.1). The indices in \mathcal{Z}_j correspond to ordered values in descending form, as shown in Figure 4.2.

As mentioned before, we argument that the MD $D(\mathbf{x}_i)$ is related to the support vectors and the class convex hull (CCH), which is generally defined as follows:

$$\Theta(\omega_j) = \left\{ \sum_{i \in \mathcal{C}_j} \beta_i \mathbf{x}_i : i \in \mathcal{C}_j, \beta_i \in \mathbb{R}, \beta_i \geq 0, \sum_{i \in \mathcal{C}_j} \beta_i = 1 \right\}, \quad (4.2)$$

where a number of $|\mathcal{C}_j|$ points in the form of $\sum_{i \in \mathcal{C}_j} \beta_i \mathbf{x}_i$ are the boundaries of the j -th class sample cloud. Then we can define the sets of indices corresponding to the convex hull of the j -th as $\mathcal{S} = \Theta(\omega_j)$. The algorithm that obtains the convex hull has complexity of $\mathcal{O}(N^{\frac{M}{2}})$, where M is the dimensionality of the feature vector. The complexity of the method proposed here mas a complexity of $\mathcal{O}(L)$, where $L = \max [N \log N, \binom{M}{2}]$. This demonstrates that our model has lower complexity

Feature Space	Squared Mahalanobis Distance Estimation	Distance-Ranked Class Indices	
$\phi(\mathbf{x}_1)$	1.1273	2	} Z_1
$\phi(\mathbf{x}_2)$	7.4530	1	
\vdots	\vdots	\vdots	
$\phi(\mathbf{x}_{i \in C_1})$	0.8690	$i \in C_1$	
\vdots	\vdots	\vdots	\vdots
$\phi(\mathbf{x}_1)$	0.4723	$i \in C_j$	} Z_j
$\phi(\mathbf{x}_2)$	9.5481	1	
\vdots	\vdots	\vdots	
$\phi(\mathbf{x}_{i \in C_j})$	2.4671	2	

Figure 4.2: Mahalanobis distance-ranking of class indices using feature vectors in either the input space or the kernel-induced feature space.

than those based on convex hulls. Now, we define a relationship between \mathcal{Z} , \mathcal{S} , and SV in Proposition 4.1.

Proposition 4.1 (SVs and Within-Class Distances). *Assume classes in \mathcal{D} are linearly separable. Let $\mathcal{Z}_v = \{\mathcal{Z}(1), \mathcal{Z}(2), \dots, \mathcal{Z}(v)\}$ denote the v maximum Mahalanobis distance indices. Similarly, let $\mathcal{Z}_{j,i} = \{\mathcal{Z}_{1,v}, \mathcal{Z}_{2,v}, \dots, \mathcal{Z}_{j,v}\}$ be the set of v maximum Mahalanobis distance indices of all classes. Then*

1. *the maximum Mahalanobis distance samples indices contain the convex hull indices: $\mathcal{Z}_v \in \mathcal{S}$,*
2. *the maximum Mahalanobis distance samples indices contain the support vector indices: $\mathcal{Z}_{j,i} \in SV$,*

where v is an integer stating how many samples per class should be considered.

Proposition 4.1 states that the first v ranked MD indices \mathcal{Z}_v contain the class convex hull indices \mathcal{S} and, thus, contain the support vector indices. The integer v is bounded, $|\mathcal{D}| \leq v \leq |\mathcal{S}|$, and SV is as in Definition 3.1. Therefore, if the initial

working-set is fixed to the indices in $\mathcal{Z}_{j,i}$, the training process will converge faster. This is mainly because if the support vectors are found at the very first iterations, the problem will be solved faster. Since $\mathcal{Z}_{j,i}$ is more likely (based on Mahalanobis distance information) to contain support vector indices, one can conclude that the training will be faster. We have found that a good value for v is the quotient between the initial working set size B_{ini} and the total number of classes: $v = \left\lceil \frac{B_{\text{ini}}}{|\mathcal{D}|} \right\rceil$. This value v is used as input in Algorithm 4.1.

As an example, let us consider a case of a random variable $\mathbf{x} \in \mathbb{R}^2$. Then draw 50 samples that follows a multivariate normal distribution with parameters $\boldsymbol{\mu} = \begin{bmatrix} 1 & 2 \end{bmatrix}^T$ and $\boldsymbol{\Sigma} = \begin{bmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$ and assign these to class ω_1 ; draw 50 more from a multivariate copula [47] distribution with parameter $\rho = 0.8$, and assign these to class ω_2 . The problem is shown in Figure 4.3 with the resulting convex hull. The associated support vectors are also shown in Figure 4.3. Let us remark that the Mahalanobis distance is associated with the spreadness of the class sample cloud, such that the most uncertain samples have the highest Mahalanobis distance, as shown in Figure 4.3. It is also important to remark that the highest Mahalanobis distance correspond to the lowest probability samples, which is also correlated to the support vectors as mentioned before.

The speedup quantification and other numerical testing of Algorithm 4.1 will be given later in this chapter along with other experiments. The next section addresses another important problem in SV-based learning machines: Model Selection.

4.3 Large-Scale LP-SVR Hyper-Parameter Estimation

This section addresses the hyper-parameters estimation problem for the proposed LP-SVR. The hyper-parameters, in the case of LP-SVR, are $\boldsymbol{\theta} = [C, \sigma]$. For other

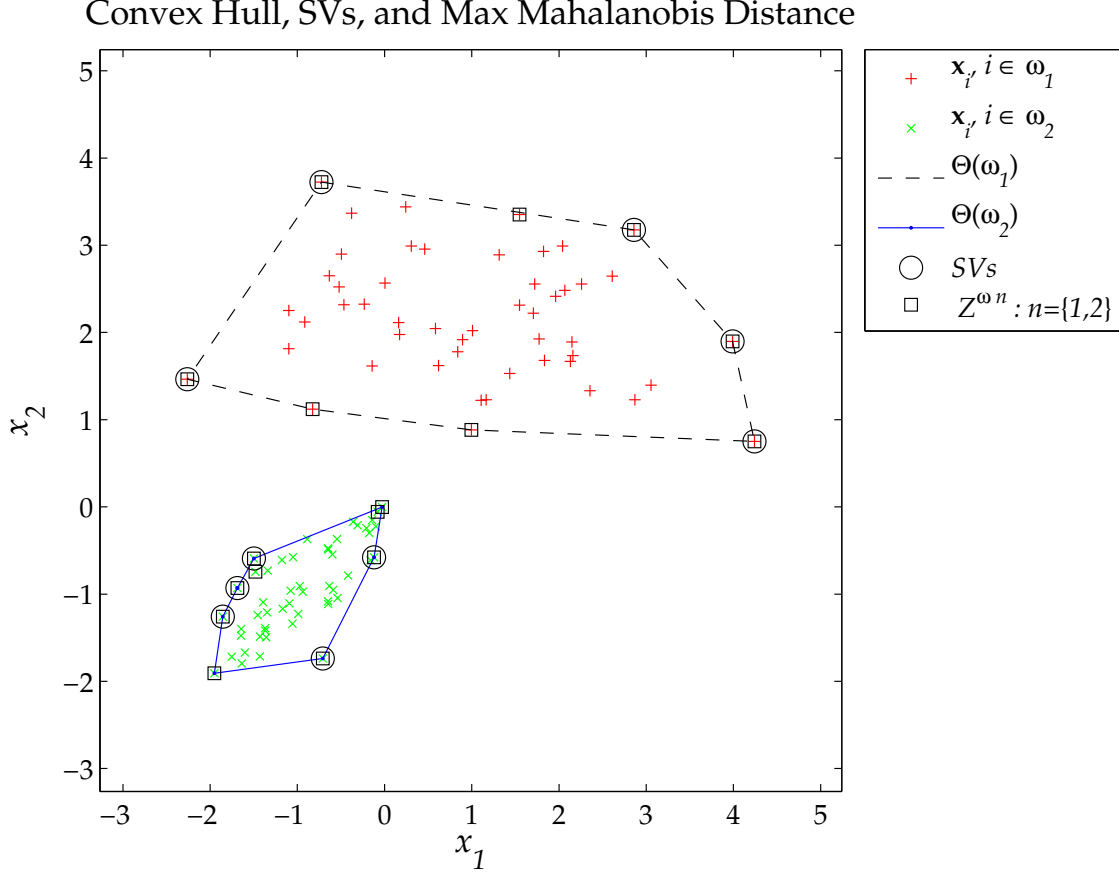


Figure 4.3: Relationship between convex hull and maximum Mahalanobis distance for the two class problem. Here it is shown the original separable two class problem, class convex hull, support vectors, and k maximum Mahalanobis distance samples. Note that both SVs and Convex Hull match the k maximum Mahalanobis distance samples.

models, such as the ν -SVM, the hyper parameters are $\boldsymbol{\theta} = [\nu, \sigma, \delta]$.

This is actually one of the general open problems in SV learning [178]. Broadly speaking, one tries to find those hyper-parameters $\boldsymbol{\theta}$ minimizing the generalization error of an SV-based learning machine. In this regard, Anguita, *et al.* [6], comments that “the estimation of the generalization error of a learning machine is still the holy grail of the research community.” The significance of this problem is that, if we can find a good generalization error estimate, then we can use a heuristic or mathematical

technique to find the hyper-parameters θ via minimization of the generalization error estimate.

Current efforts involve techniques of \mathcal{K} -fold cross validation [56], leave-one-out cross validation [6], bootstrapping [89], maximal discrepancy [7], and compression bound [6, 201]. However, most algorithms are problem dependent [26]. This statement is confirmed by Anguita, *et al.* [6]. The authors performed a comprehensive study on the above techniques and they ranked such techniques according to their ability to estimate the true test generalization error. Anguita, *et al.* [6], concluded that most of the methods they evaluated either underestimate or overestimate the true generalization error. Also, their research suggests that the \mathcal{K} -fold cross validation technique is one of the less risky techniques for estimating the true generalization error.

In this research we use the \mathcal{K} -fold cross validation technique to estimate the true test generalization error. Along with this technique, we define error functions as measures of the training generalization error for both classification and regression problems. We propose to minimize the estimated true generalization error by adapting the Newton method with line-search.

From the optimization point of view, the solution to the problem is non-trivial. To illustrate this, Figure 4.4 shows the output root mean squared error of an LP-SVR as a function of its hyper-parameters $[C, \sigma] = \theta$. Note how the error surface is non-smooth and has many local minima; therefore, it is non-convex. Our aim here is to adapt Newton method to provide an acceptable solution to the problem of finding the hyper-parameters.

We begin our discussion by defining the error functions that will measure the training generalization error.

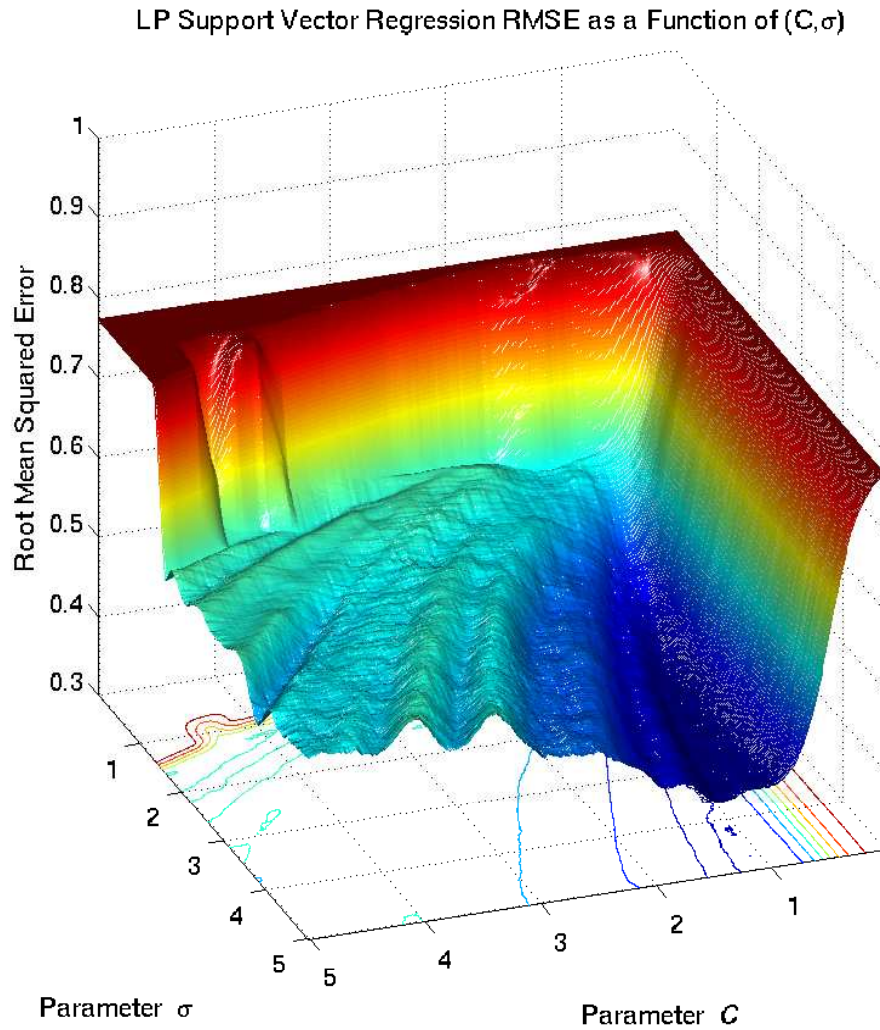


Figure 4.4: Response of the root mean squared error as a function of $\theta = [C, \sigma]$. Note, how the error surface is non-smooth and has many local minima.

4.4 Model Parameters Selection Criteria: Error Functions

As mentioned before, we want to minimize the true test generalization error. The kinds of generalization error measurements are, in fact, problem-type-dependent. In the following subsection, error metrics particular to classification and regression

problems for the LP-SVR model in (3.2) are chosen. To particularize, the objective is to estimate the model vector of parameters $\boldsymbol{\theta} = [C, \sigma]$.

4.4.1 Error Functions for Multi-Class and Two-Class Problems

The error functions we want to use for multi-class problems are two: a modified estimate of scaled error rate (ESER) and the balanced error rate (BER). The ESER metric is given by

$$f_1(\boldsymbol{\theta}) = \sum_{i=1}^N \zeta \Psi\{(y_i - d_i) - 0.5\} \quad (4.3)$$

where y is the actual output of the classifier LP-SVR when the input vector \mathbf{x} is presented at its input; ζ is a scaling factor used only to match the ESER to a desired range of values; and the $\Psi\{\cdot\}$ function is given by

$$\Psi\{r\} = \frac{1}{1 + e^{-\gamma r}} \quad (4.4)$$

which is an approximation to the well known ideal unit step function. The quality of the approximation is given by the parameter γ as illustrated in Figure 4.5. In all of our experiments, ζ is fixed to $\frac{1}{N}$. If $\zeta = \frac{1}{N}$, then the f_1 has values only within the interval $f_1 \in [0, 1]$, which is desired.

In some special cases, the ESER may become biased towards false positive counts, especially if we have a large number of unbalanced class examples. Therefore, we use the BER which is defined as follows:

$$f_2(\boldsymbol{\theta}) = \frac{1}{2} \left(\frac{FP}{TN + FP} + \frac{FN}{FN + TP} \right) \quad (4.5)$$

where TP stands for “True Positive,” FP “False Positive,” TN “True Negative,”

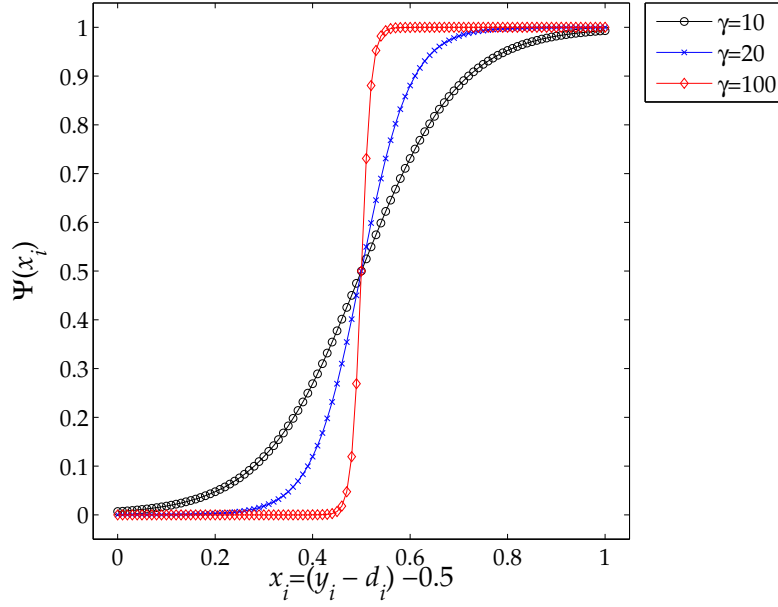


Figure 4.5: Unit step function approximation. A function $\Psi\{\cdot\}$ was used for convenience in easing computations.

and FN “False Negative.”

Clearly, the BER meets the classical misclassification rate if there are equal number of positive and negative examples; that is the case in which $TN + FP = FN + TP$ [26].

In a two class approach, it is more convenient to use the area under the receiver operating characteristic (ROC) curve, as well as the BER metric. It is well known that maximizing the area under the ROC curve (AUC) leads to better classifiers, and therefore, it is desirable to find ways to maximize the AUC during the training step in supervised classifiers. The AUC is estimated by means of adding successive areas of trapezoids. For a complete treatment of the ROC concept and AUC algorithm, see Appendix C or reference [60].

Let us define the function f_1 for the two class approach as follows:

$$f_1(\boldsymbol{\theta}) = 1 - AUC(\cdot)_{\boldsymbol{\theta}}, \quad (4.6)$$

where the $AUC(\cdot)$ is computed using Algorithms 1, 2, and 3 from [60]. Let us recall that ideally it is desired that $f_1(\boldsymbol{\theta}) = 0$, which evidently in (4.6) means a maximization of the AUC. The function f_2 for the two-class approach is the same BER as in (4.5).

4.4.2 Regression Problems

In regression one want to use a different measure of error. The error functions we want to use for regression are two: Sum of Square Error (SSE), and STATistical metrics (STAT). The SSE metric is given by

$$f_1(\boldsymbol{\theta}) = \sum_{i=1}^N (y_i - d_i)^2 \quad (4.7)$$

where y is the actual output of the classifier LP-SVR when the input vector \mathbf{x} is presented at its input.

The second metric is based on the statistical properties of the residual error given by the difference $y_i - d_i$. From estimation theory it is known that if we have the residual error expected value equal to zero, and a unit variance, we have achieved the least-squares solution to the regression problem, either linear or non-linear. Furthermore, it is understood that as the mean and variance of the residual error approach zero, the regression problem is better solved. Let us denote the expected value of the residual error as

$$\mu = \mathcal{E}[y_i - d_i] = \frac{1}{N} \sum_{i=1}^N y_i - d_i, \quad (4.8)$$

and the variance of the residual error as follows

$$\sigma^2 = \mathcal{E}[y_i - d_i - \mu]^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - d_i - \mu)^2, \quad (4.9)$$

from where it is desired that $\mu, \sigma^2 \rightarrow 0$. Hence, the second error metric is defined as:

$$f_2(\boldsymbol{\theta}) = \sigma^2 + \sqrt{\mu^2} \quad (4.10)$$

where the term $\sqrt{\mu^2}$ has the meaning of the absolute value of the mean, since $|\mu| = \sqrt{\mu^2}$ is easier to handle in optimization problems.

4.5 Adaptation of Newton Method

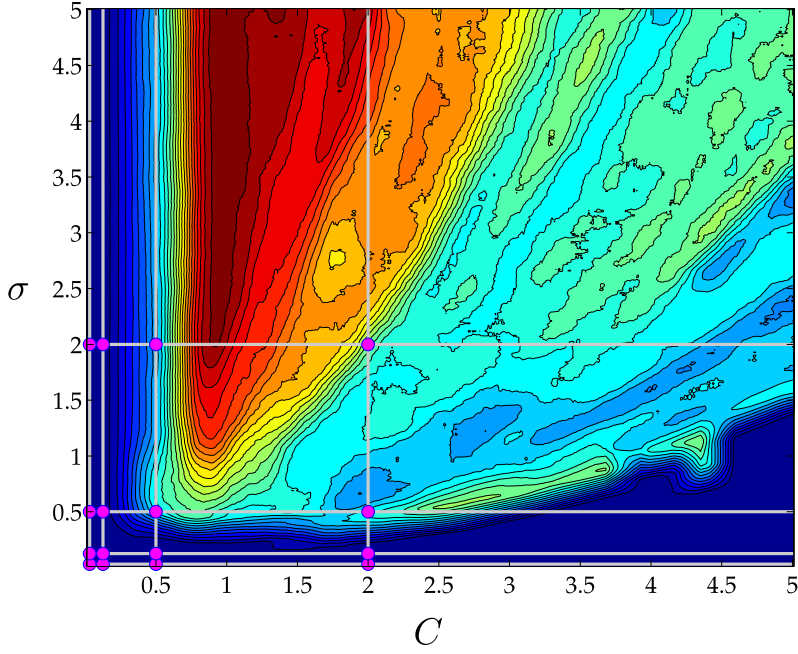
Different approaches to find the optimal set of parameters by minimizing some estimate of the true test generalization error [7, 56, 196]. Cawley [26], for example, introduced a method that uses the PRESS statistic as an error function and the simplex method as solver within the Least Squares SVM (LSSVM) context. However, Cawley method is not appropriate for large-scale applications since the LSSVM works well only for small problems.

The most popular strategy to address the problem of finding the hyper-parameters is the usage of a logarithmic grid [6, 7, 56]. This technique consists of varying C and σ in a logarithmic fashion. The intervals commonly used are the following:

$$C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}, \quad (4.11)$$

$$\sigma \in \{2^{-2}, 2^{-1}, \dots, 2^6, 2^7\}. \quad (4.12)$$

This is illustrated in Figure 4.6a. The figure shows a partial evaluation of the logarithmic grid and the points of evaluation are shown in circles. This technique requires 110 evaluations, *i.e.*, number of trainings. Then the point $\boldsymbol{\theta}$ with the lowest



(a) Error surface as function of hyper-parameters and logarithmic grid evaluation.

Figure 4.6: Comparison between a logarithmic grid refinement technique (a)-(c) [6, 7] and the proposed adaptation of the newton method.

error is chosen to be refined around its center. This process is repeated as needed. Figure 4.6b-c depicts this behavior only for two levels of refinement. One can observe how the second level of refinement leaves the method at a close-to-optimal value. Note also that for illustration purposes, the refinement grids in Figure 4.6a-c are partially shown, there are many more actually.

In contrast, the proposed adaptation of the Newton method provides a better solution in only two iterations, as shown in Figure 4.6d-e. Let us now explain how Newton method can be adapted to find the LP-SVR hyper-parameters. In our discussion we use Newton method combined with the metrics on Section 4.4, to find the model parameters of the proposed LP-SVR formulation (3.2). If the reader is not familiar with the Newton method, it is recommended for the reader to revise the development presented in Appendix D, particularly, in Section D.1.

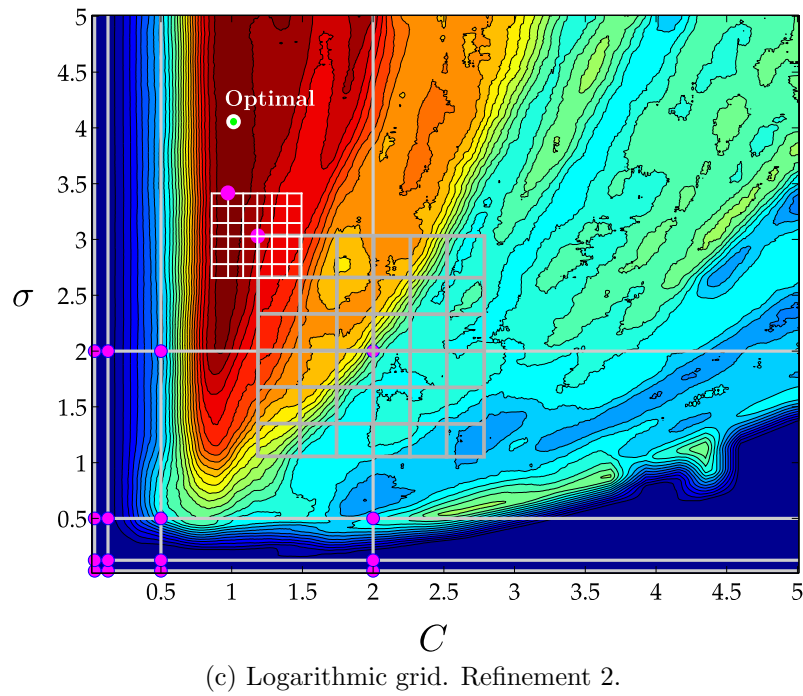
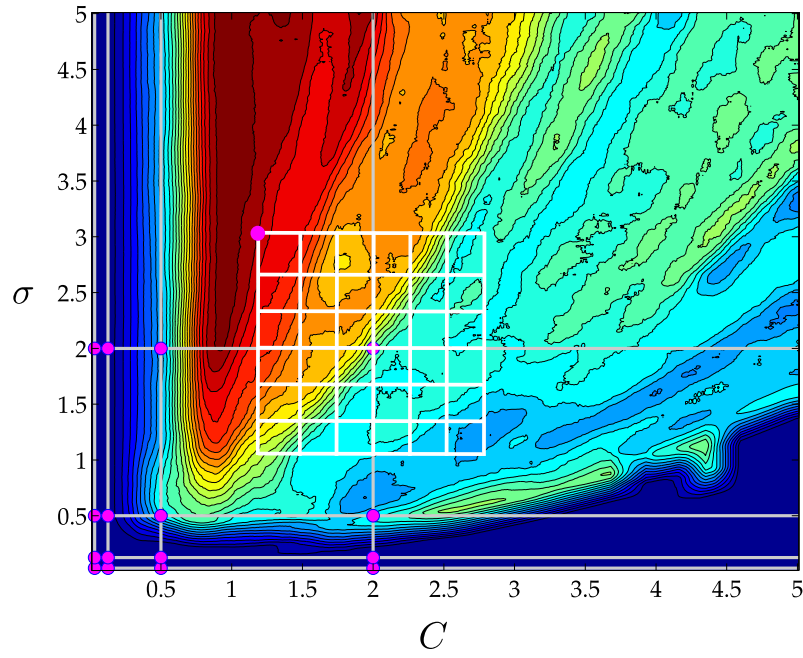


Figure 4.6: Continued... comparison between a logarithmic grid refinement technique (a)-(c) [6,7] and the proposed adaptation of the newton method.

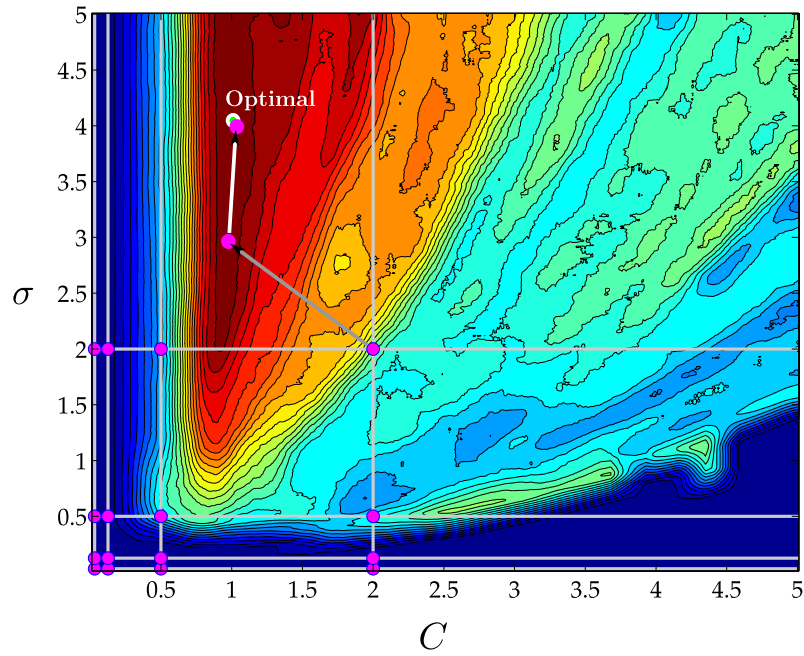
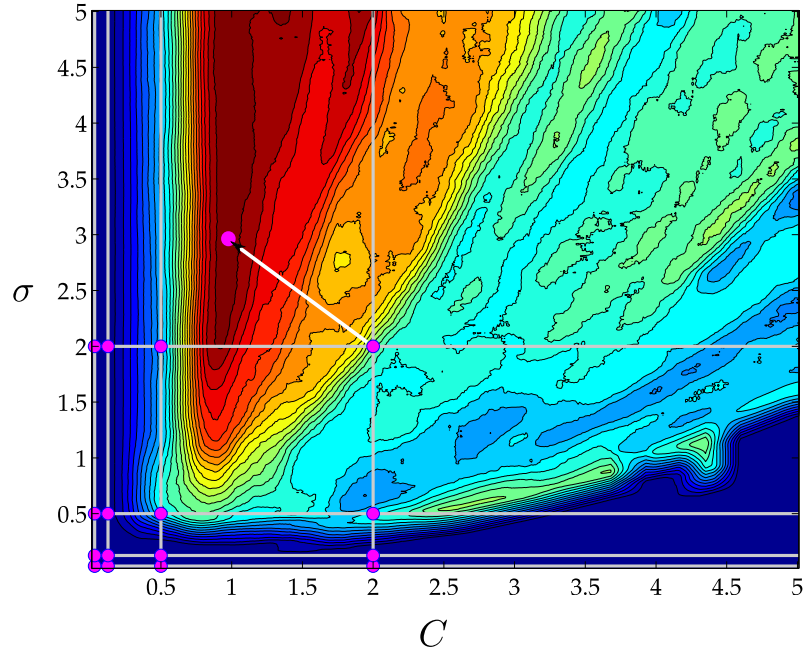


Figure 4.6: Continued... comparison between a logarithmic grid refinement technique (a)-(c) [6,7] and the proposed adaptation of the newton method.

4.5.1 Newton Method Adaptation

Newton method for function minimization can be adapted for the cases presented in Section 4.4. We start by defining a matrix of functions \mathbf{F} as follows:

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{pmatrix} f_1(\boldsymbol{\theta}) \\ f_2(\boldsymbol{\theta}) \end{pmatrix}_{2 \times 2} \quad (4.13)$$

where clearly $\mathbf{F} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ and $\boldsymbol{\theta} : \mathbb{R}^2 \mapsto \mathbb{R}$. The typical challenge is to compute the Jacobian matrix $\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta})$, since not all the error functions are differentiable, *i.e.*, (4.5) or (4.6). Then the classical approach is to estimate $\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta})$ via finite difference approximation, or secant approximation. For convenience, is used a finite difference approximation. In this case, $\tilde{\mathbf{J}}_{\mathbf{F}}(\boldsymbol{\theta})$ corresponds to a finite difference derivate approximation which solves (D.2) using (D.3) where

$$\frac{\partial f_n}{\partial \theta_1} \cong \frac{f_n(\theta_1 + h, \theta_2) - f_n(\theta_1, \theta_2)}{h} \quad (4.14)$$

$$\frac{\partial f_n}{\partial \theta_2} \cong \frac{f_n(\theta_1, \theta_2 + h) - f_n(\theta_1, \theta_2)}{h} \quad (4.15)$$

allowing h to be sufficiently small, as appropriate.

Next, we proceed to use a popular and fast way of estimating the true test error, given a vector of hyper parameters $\boldsymbol{\theta}$ and a training set \mathcal{T} : the \mathcal{K} -fold cross validation technique [16,77,78]. The idea is to partition the training set into a number of smaller sets, then train the LPSVR model with the smaller dataset and the remaining data is used as a validation set. The process is repeated switching partitions, and the performance is averaged. The quality of the true test error approximation increases along with the number of partitions.

In this research it has been defined the following rule for finding a “good” number

of partitions in \mathcal{K} :

$$|\mathcal{K}| = \max \left\{ 10, \left\lceil \frac{N}{B_{\max}} \right\rceil \right\}, \quad (4.16)$$

where B_{\max} represents the maximum working-set size; the function $\lceil \cdot \rceil$ represents a round up operation; and $|\mathcal{K}|$ is the number of partitions in \mathcal{K} . In the case of a small-scale implementation $|\mathcal{K}|$ will typically be the classic 10-fold cross validation approach.

The partitions in the set \mathcal{K} is denoted as

$$\mathcal{K} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_k\} \quad (4.17)$$

where $k = |\mathcal{K}|$, and \mathcal{P}_k denotes the k -th partition and contains the indices of those training data points in $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$. Therefore, we say that $\mathcal{K} \equiv \{1, 2, \dots, N\}$.

Formally, the training set \mathcal{T} is partitioned in $|\mathcal{K}|$ groups of data (ideally of equal size), then train the classifier with $|\mathcal{K}| - 1$ and use the remaining data as validation set. The process is repeated for all the partitions \mathcal{P}_k and the error is averaged as follows

$$\tilde{\mathbf{F}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{K}|} \sum_{k=1}^{|\mathcal{K}|} \mathbf{F}(\boldsymbol{\theta}_{\mathcal{P}_k}) \quad (4.18)$$

where $\mathbf{F}(\boldsymbol{\theta}_{\mathcal{P}_k})$ is the error obtained for the k -th partition; $\tilde{\mathbf{F}}(\boldsymbol{\theta})$ is an estimate of the true test error; $\mathcal{T}_{\mathcal{P}_k} = \{\mathbf{x}_i, d_i\}_{i \in \mathcal{P}_k}$ and $\mathcal{T}_{\mathcal{K}} = \{\mathbf{x}_i, d_i\}_{i \in \mathcal{K}}$.

4.5.2 Algorithm Discussion

The final algorithm (shown as Algorithm 4.2) requires the cross validation indices, \mathcal{K} , and also the training set \mathcal{T} from which the LP-SVR parameters producing the minimum error $\boldsymbol{\theta}^*$ will be estimated as $\tilde{\boldsymbol{\theta}}^t$.

Algorithm 4.2 Quasi-Newton method adaptation to find parameters $\boldsymbol{\theta}^*$ for an LP-SVR model.

Require: Cross validation indices \mathcal{K} .

Require: Training set \mathcal{T} .

1: Initial point $\boldsymbol{\theta}_{\mathcal{K}}^0$. ▷ With logarithmic grid.

2: **for** $t = 0, 1, 2, \dots$, until convergence **do**

3: $\mu_{\lambda} \leftarrow \min \text{eig} \left(\tilde{\mathbf{J}}_{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) \right)$

 Solve for $\Delta \boldsymbol{\theta}_{\mathcal{K}}^0$ depending on the condition:

4: **if** $\mu_{\lambda} > 0$ **then**

$$\tilde{\mathbf{J}}_{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) \Delta \boldsymbol{\theta}_{\mathcal{K}}^t = -\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) \quad (4.19)$$

5: **else**

$$\left(\tilde{\mathbf{J}}_{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) + (\mu_{\lambda} + \delta) \mathbf{I} \right) \Delta \boldsymbol{\theta}_{\mathcal{K}}^t = -\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) \quad (4.20)$$

6: **end if**

7: Sufficient decrease:

▷ Armijo's condition

 Find β_1 that satisfies:

$$\|\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t + \beta_1 \Delta \boldsymbol{\theta}_{\mathcal{K}}^t)\|_2 \leq \|\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t)\|_2 + \sqrt{1 - 2\beta_1\beta_2}$$

8: Update:

$$\boldsymbol{\theta}_{\mathcal{K}}^{t+1} = \boldsymbol{\theta}_{\mathcal{K}}^t + \beta_1 \Delta \boldsymbol{\theta}_{\mathcal{K}}^t$$

9: **end for**

Ensure: Model parameters estimate $\tilde{\boldsymbol{\theta}}^t = \boldsymbol{\theta}_{\mathcal{K}}^t$.

Then the algorithm proceeds using the approximation to the true Jacobian (4.13)-(4.15). However, note that every single function evaluation of (4.13) requires cross validation, as explained before. As a consequence, the Jacobian implies four function calls. The remaining steps are the linear system solution, the Armijo's condition, and the update.

The linear system in (4.19) requires special attention: because if we use a large number of error functions to minimize, then the linear systems becomes computationally expensive to solve. If this is the case, one possible approach is to use any well known direct approach such as the *LU*-factorization [136] or an indirect approach such as the classic conjugate gradient algorithm by Hestennes 1956 [80, 136]. However, in our case, since we use only two error functions, the linear system solution represents no significant computational expense.

The other special consideration with the linear system is when the Jacobian matrix is non invertible (*i.e.*, singular or degenerate). There is an easy way to verify if the Jacobian is singular: Look for the minimum eigenvalue and if it is less than or equal to zero, then the Jacobian is singular. This idea leads to a trick that consists of shifting the eigenvalues of the Jacobian so that it becomes non-singular for computational purposes, as denoted in (4.20).

In Algorithm 4.2, μ_λ is the minimum eigenvalue of the Jacobian, $\delta > 0$ is a constant sufficiently small that cannot be interpreted as zero, and \mathbf{I} is the identity matrix of identical size to the Jacobian. In (4.20) we typically chose $\delta = 1 \times 10^{-8}$.

4.5.3 Stopping Criteria

The stopping criteria used in this algorithm includes three conditions. First, a condition that monitors if the problem has reached an exact solution to the problem,

which is expressed as follows:

$$\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t) = \boldsymbol{\varepsilon} \quad (4.21)$$

where $\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2]^T$. Ideally, $\varepsilon_1 = 0$, and $\varepsilon_2 = 0$.

Second, the ℓ_2 -norm of the objective function is monitored, which measures the distance to zero (or to a certain threshold) from an approximate solution at iteration t . This can be expressed as follows:

$$\|\tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t)\|_2 \leq \varepsilon_3 \quad (4.22)$$

where ε_3 is some threshold, ideally $\varepsilon_3 = 0$.

Third, we set a condition that measures the change between solutions at each iterate, as follows:

$$\|\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t\|_2 \leq \varepsilon_4 \quad (4.23)$$

where ε_4 is typically set to a very small value.

Condition (4.23) states an early stopping criteria if the algorithm has no variability in terms of the updates at each iteration. However, it may happen that although the algorithm is updating the solution $\boldsymbol{\theta}^t$ at each iterate, no “significant” progress is made towards a solution. In such case, another classical early stopping criteria is used: maximum iterations. The criteria is simply

$$t \leq \varepsilon_5 \quad (4.24)$$

where ε_5 is the maximum number of iterations permitted. Experimental results for this method are shown further in this chapter.

4.6 Experimental Results

To show the effectiveness and efficiency of the proposed algorithms, simulations were performed over different datasets. The summary of the properties of these datasets are shown in Table C.1. Note that the simulations include classification in two and multiple classes, as well as regression problems. The results will be explained in the following paragraphs.

This section is divided in two different analysis. The first part analyzes the speedup resulted of using the proposed sample selection algorithm using state of the art models and using benchmarking datasets. The second part analyzes the model selection algorithm using the proposed algorithm and state of the art models using benchmarking datasets. In this experiments, $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N^*}$ is defined as the testing set, where N^* is the number of samples available for testing.

4.6.1 Learning Speedup

Figure 4.7 depicts the behavior of the support vectors across iterations using the speedup strategy. The figure shows the number of support vectors, sparse support vectors, saturated support vectors, and exact support vectors. Note how the support vectors are found early in the learning process. If we compare Figure 4.7 to the analysis presented in Figure 4.8 (which is repeated from Figure 3.9 by convenience), we notice that most of the support vectors are found in earlier iterates, which is the goal of the strategy. In Figure 3.9 it seems that in the average case, most of the sparse support vectors (SPV) are found in around 40% of the total iterations. In the other hand, Figure 4.7 tells that in around 20% of the total number of iterates.

Table 4.1 shows the total training time of the experiments described in Section 3.7.1 without speedup. Compare against Table 4.2 that shows the total training time after using the speedup strategy. From these two tables we can observe that if strategy is used the total training time decreases, particularly as the problem size

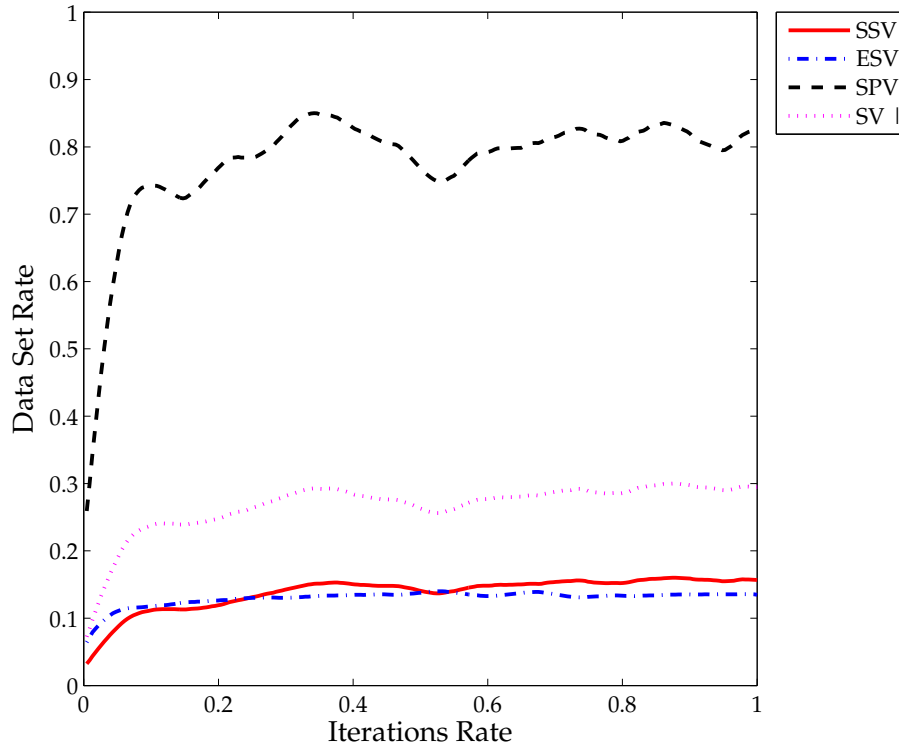


Figure 4.7: Sample Selection. Support Vectors as a Function of Iterations. SSV, ESV, SPV, and SV.

increases, as it was expected.

Besides a time reduction analysis, it is also important to observe if the total number of iterations is reduced if the speedup strategy is applied. Table 4.3 shows the total reduction of iterations, in percent, using the speedup strategy. Clearly, if the problem size is large, the reduction in number of iterations is also larger. A great benefit can be obtained of the the speedup strategy, especially, if the class cloud is as close as possible to a multivariate Gaussian distribution within the kernel-induced feature space.

Another interesting thing to notice is that the percentage of iterations reduction seem to be superior in proportion to the learning time after speedup. This follows from noticing that, even if the support vectors are found at early iterations, still, the

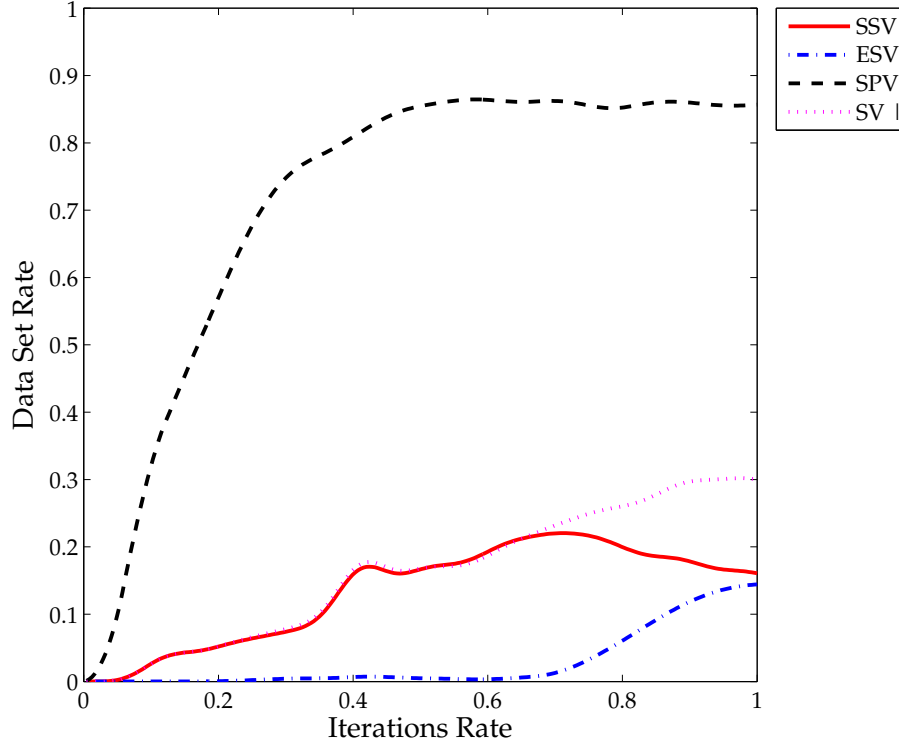


Figure 4.8: Support Vectors as a Function of Iterations. SPV SSV ESV SV.

learning process has to perform several time-consuming decompositions and comparisons that might increment in size, specially, if the number of support vectors is large. However, since support vectors are found early at the learning process, there is no need to solve some of the subsequent sub-problems since the support vectors found will satisfy the KKT conditions of many of the sub-problems.

4.6.2 Model Selection

Now, to illustrate the behavior of the proposed hyper-parameter selection approach, we present the results of experimental cases. Figure 4.9 shows the time as a function of the problem size as well as the total number of iterations as a function of problem size. It can be seen that the total number of iterations is not affected by the problem size. In spite of this, each iteration takes more time to be completed as problem size

Table 4.1: Total Training Time without speedup

Dataset	Classifiers					
	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN [1, 20, 1]
Ripley	9	16	8	4	—	4
Wine	6	4	6	3	—	5
ADA	75	174	—	4412	—	63
GINA	50	116	—	1403	—	48
HIVA	73	161	—	—	—	—
NOVA	25	47	258	—	—	—
SYLVA	247	495	—	—	—	190
Iris	6	4	3	3	—	3
Spiral	10	10	14	8	—	10
$f(x) = \text{sinc}(x)$	28	41	—	—	181	26
$f(x) = \text{sinc}(x) \times \pi$	9376	806	—	—	6933	602
Synthetic S	9349	794	—	—	—	—
Synthetic NS	9180	817	—	—	—	—
Avg.	2187	286	—	—	—	—

Table 4.2: Total Training Time with Speedup.

Dataset	Classifiers					
	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN [1, 20, 1]
Ripley	8	9	7	4	—	4
Wine	1	2	4	3	—	5
ADA	71	74	—	3533	—	63
GINA	38	63	—	1191	—	48
HIVA	57	115	—	—	—	—
NOVA	17	19	234	—	—	—
SYLVA	246	230	—	—	—	190
Iris	2	2	2	2	—	3
Spiral	5	5	5	5	—	10
$f(x) = \text{sinc}(x)$	9	8	—	—	15	26
$f(x) = \text{sinc}(x) \times \pi$	5672	764	—	—	5597	602
Synthetic S	7038	467	—	—	—	—
Synthetic NS	8608	672	—	—	—	—
Avg.	1674	187	—	—	—	—

Table 4.3: Iterations reduction percent after speedup.

Dataset	Classifiers					
	LS SVM	LP-SVR	IncSVM	LSSVM	D. Reg. Trees	FFNN [1, 20, 1]
Ripley	1	2	22	52	—	0
Wine	1	4	0	1	—	0
ADA	24	9	—	10	—	0
GINA	12	10	—	11	—	0
HIVA	39	9	—	—	—	—
NOVA	7	1	4	—	—	—
SYLVA	30	1	—	—	—	0
Iris	1	1	8	5	—	0
Spiral	15	12	4	7	—	0
$f(x) = \text{sinc}(x)$	5	13	—	—	1	0
$f(x) = \text{sinc}(x) \times \pi$	59	57	—	—	43	1
Synthetic S	23	51	—	—	—	—
Synthetic NS	27	37	—	—	—	—
Avg.	18.7	15.7	—	—	—	—

increases, which was expected.

Figure 4.10 analyzes the balanced error rate and the area under the ROC curve at each iterate; Figure 4.11 shows the root mean squared error and mean absolute value as the algorithm progresses iteratively. It can be seen that error functions *i.e.*, BER, RMSE, and MAE, decrease dramatically in the first iterations, and the quality measure AUC increases also at the early iterations. The behavior can be classified between quadratic and linear, as was expected. Note however, that this behavior is directly dependent of a “good” initial point given by the logarithmic grid technique. Finally, let us point out that the results shown in Figure 4.10 and Figure 4.11 is the result of averaging individual behaviors across all experiments.

Let us consider the results shown in Table 4.4. The second column shows the total number of iterations; in average we observe that the iterations are around eight, which is one of the most important properties of the method. Column three and four of Table 4.4 show the hyper-parameters found; while in the fifth column one see the

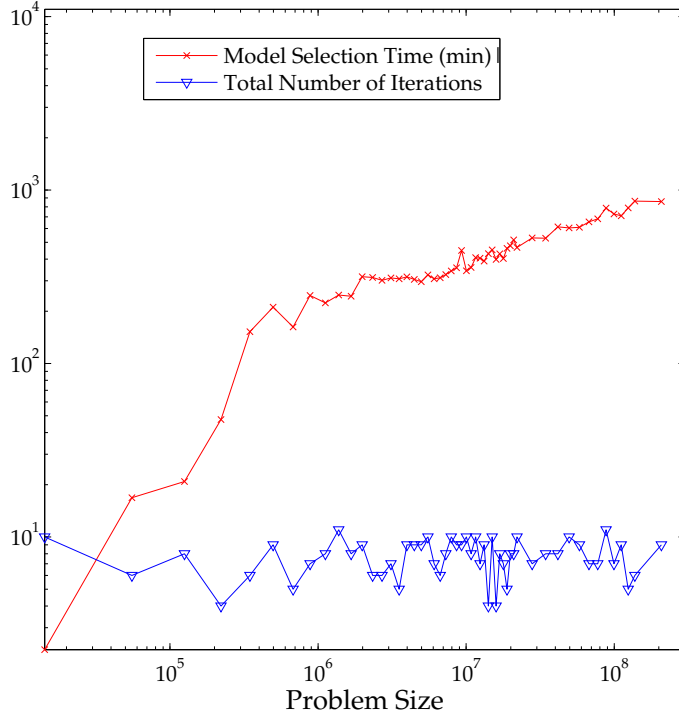


Figure 4.9: Total Model Selection Time and Number of Iterations as a Function of the Problem Size.

ℓ_2 -norm of the algorithm at the last iteration. Note how variable is this value depending on the dataset. Finally, the sixth column shows the criteria that made the algorithm stop; it is clear that the most common is the criteria ε_4 described in (4.23). This latter statement means that the algorithm stopped because no progress was being made towards the solution. Note also that, in average, the total number of iterations is expected to be around 8, as confirmed also in Figure 4.9.

The simulation results in Table 4.5 show $f_n(\tilde{\boldsymbol{\theta}}^*)$ which represents the result of the n -th function (or error criteria), evaluated at the approximated solution $\tilde{\boldsymbol{\theta}}^*$ using only the testing set \mathcal{D} . These results are shown in columns two through six. In column number two is shown the modified estimate of scaled error rate (4.3), which was used with parameters $\zeta = \frac{1}{N}$ and $\gamma = 100$. These parameter ζ was chosen by convenience in order to have an error within the interval $[0, 1]$. The third column

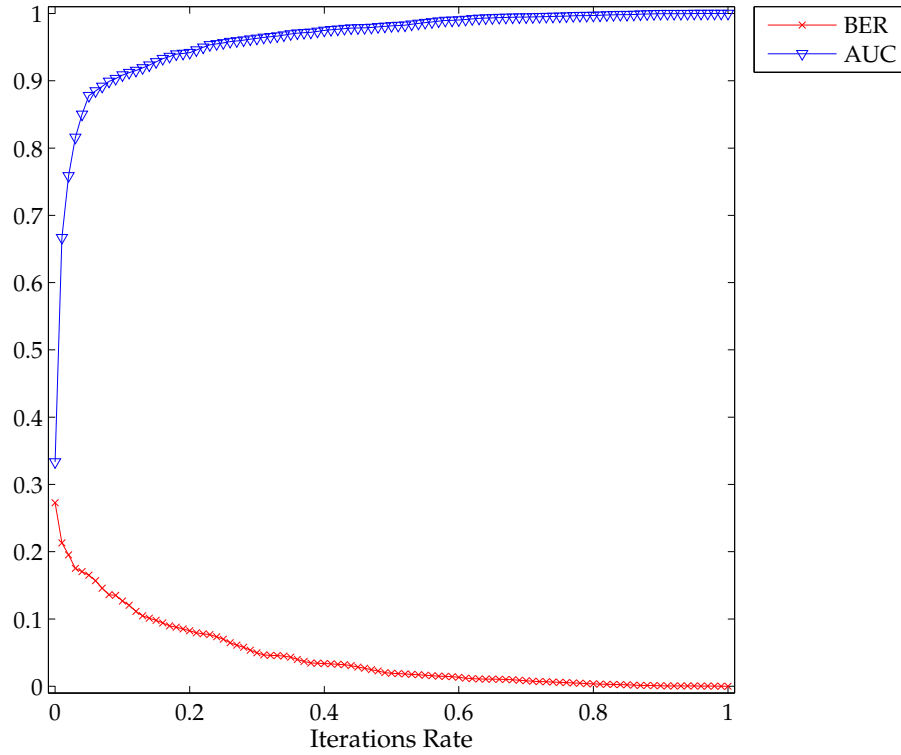


Figure 4.10: BER and AUC as a Function of Iterations Number.

displays results for when the balanced error rate (4.5) was utilized. The area under the ROC curve (4.6) shown in the fourth column also produces a result within the same interval as the BER. In contrast, regression error functions shown in the fifth and sixth column have a wide interval, but is always positive. That is, sum of squared error (4.7) and the statistical properties (4.10) fall into the interval $[0, \mathbb{R}_+]$. Note that classification error functions in average are zero for practical purposes, which is desirable.

Moreover, in Table 4.5 columns six through seven we show statistical properties of the residuals given by $(y - d)_{\mathcal{D}}$. This residual is acquired by showing the testing set \mathcal{D} to the LP-SVR model with hyper-parameters $\tilde{\theta}^*$ and measuring the output y . Ideally, we want the average of the residuals to be zero, as well as their standard deviation. This desired property is achieved during our simulations.

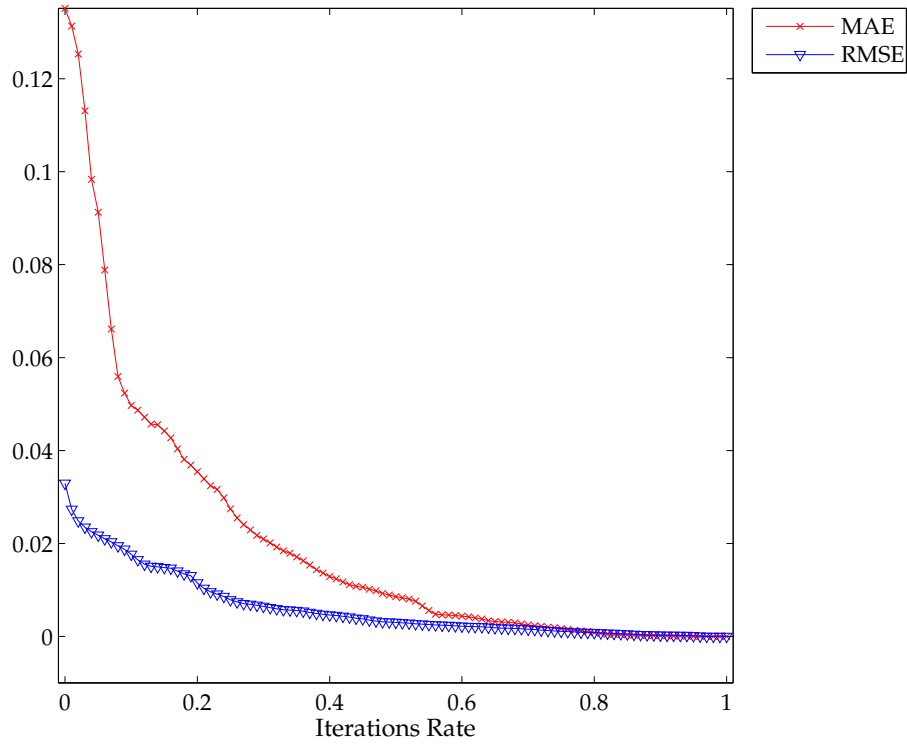


Figure 4.11: RMSE, and MAE as a Function of Iterations Number.

In Table 4.5 the average of the residuals $\mu \equiv \mathcal{E}[(y-d)_{\mathcal{D}}]$ and the residuals variance is given by $\sigma^2 \equiv \mathcal{E}[(y-d)_{\mathcal{D}} - \mu]^2$. These parameters are approximated from the traditional sample mean and unbiased sample variance estimators.

4.7 Computational Concerns

4.7.1 Speedup by Sample Selection

The primary concern of the speedup method is the computation of the covariance matrix $\Sigma_{\mathbf{x}|\omega_j}$. In our implementation, the covariance matrix was estimated with the

Table 4.4: Summary of Behavior.

	$\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4 = 1.1921 \times 10^{-7}$, and $\varepsilon_5 = 100$				
Dataset	t	C^t	σ^t	$\ \tilde{\mathbf{F}}(\boldsymbol{\theta}_{\mathcal{K}}^t)\ _2$	Stop Crit.
Ripley	12	5.047	0.2501	0.0021	ε_4
Wine	1	0.031	0.2500	0.0000	$\varepsilon_1, \varepsilon_2$
ADA	7	0.460	117.82	0.9813	ε_4
GINA	8	0.125	157.49	0.1658	ε_4
HIVA	11	0.500	8.0730	0.0954	ε_4
NOVA	15	2.004	4.7045	0.0313	ε_4
SYLVA	7	1.125	4096.1	0.1512	ε_4
Iris	6	11.46	1.7726	0.0019	ε_4
Spiral	5	8.287	0.2515	0.0001	$\varepsilon_1, \varepsilon_2$
Synthetic S	9	100.9	3.0403	0.0511	ε_4
Synthetic NS	11	1.034	3.1623	0.1943	ε_4
$f(x) = \text{sinc}(x)$	6	959.1	0.9978	0.0011	ε_3
$f(x) = \text{sinc}(x) \times \pi$	8	0.087	3.9101	0.1123	ε_4
Avg.	8.2	—	—	0.1502	ε_4

sample covariance matrix

$$\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j} = \frac{1}{|\mathcal{C}_j| - 1} \sum_{i \in \mathcal{C}_j} (\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})(\mathbf{x}_i - \boldsymbol{\mu}_{\mathbf{x}|\omega_j})^T \quad (4.25)$$

where $\mathbf{x}_i \in \mathbb{R}^M$ is some random vector with $|\mathcal{C}_j|$ realizations, and $\boldsymbol{\mu}_{\mathbf{x}|\omega_j} \equiv \mathcal{E}[\mathbf{x}_i]$ for all $i \in \mathcal{C}_j$. Clearly, $\boldsymbol{\Sigma}_{\mathbf{x}|\omega_j} \in \mathbb{R}^{M \times M}$, thus, problems with a very large number of variables *e.g.*, the NOVA dataset, cannot be resolved under current computational constraints. Therefore, some sort of feature reduction must be implemented to obtain the covariance matrix. In the case of the NOVA dataset, a large number of features are redundant or add no discriminant information and were eliminated without loss of generality.

Moreover, for the speedup process only, the kernel choice was also limited. The rule for selecting the kernel type is the following: If $N \geq 1000$ a polynomial kernel with degree $p = 1000$ is used, otherwise an RBF kernel is used. Since the purpose of

Table 4.5: Summary of Testing Set Generalization Error Metrics and Statistical Properties of Residual Errors.

Dataset	$f_n(\tilde{\theta}^*)$					$(y - d)_{\mathcal{D}}$	
	ESER	BER	1-AUC	SSE	STAT	μ	σ
Ripley	—	0.0852	0.0261	—	—	−0.0660	0.4023
Wine	—	0.0000	0.0000	—	—	0.0008	0.0007
ADA	—	0.1484	0.0669	—	—	0.0012	0.2001
GINA	—	0.0026	0.0000	—	—	0.0069	0.0397
HIVA	—	0.1710	0.0457	—	—	0.0270	0.1609
NOVA	—	0.0000	0.0000	—	—	0.0003	0.0189
SYLVA	—	0.0058	0.0000	—	—	−0.0016	0.1980
Iris	0.0009	0.0001	—	—	—	−0.0001	0.0022
Spiral	—	0.0000	0.0000	—	—	0.0001	0.0008
Synthetic S	219.20	0.0104	—	—	—	−0.0549	0.3958
Synthetic NS	328.79	0.0153	—	—	—	0.1997	0.5944
$f(x) = \text{sinc}(x)$	—	—	—	0.0003	0.0012	0.0001	0.0008
$f(x) = \text{sinc}(x) \times \pi$	—	—	—	13138	0.1147	0.4379	0.7646
Avg.	—	—	—	—	—	0.0525	0.2137

this dissertation is to deal with large-scale datasets most of the experiments used a polynomial kernel. The polynomial kernel is our second choice since it is known to be the second best after RBF kernels [130]. The degree of the polynomial kernel is directly related to the amount of data that can be efficiently handled for covariance matrix estimation purposes.

4.7.2 Model Selection

Although statistical properties of residuals based on the testing set demonstrate that the approach has an acceptable behavior, the reader must be aware that this approach has some characteristic properties that may lead to unexpected results. First, the algorithm works with an approximation to first order information, that in the worst case may also be singular. Second, the algorithm is not convergent to a global minimum; however a “good” initial point can be obtained with the logarithmic grid

technique. Third, the globalization strategy may become computationally expensive if the first order information leads far away from the solution. A good way to reduce the computational expense in finding a β_1 that produces a sufficient decrease at each iterate can be found in text books [51, 136].

Further research must be conducted in the three aspects mentioned above. In addition, different or more error functions may also be studied as well as the case when more LP-SVR parameters are being estimated, such as ϵ . Moreover, since the concepts discussed in this research also apply (with little or no modification) to other support vector (SV)-based learning machines, it will remain as future work to design and perform experiments over different SVR flavors.

4.8 Conclusion

4.8.1 Speedup by Sample Selection

Within the context of kernel-induced feature space one can assume the data is (or is close to be) linearly separable and then compute the distances from each point to the center of the class cloud. This is done using the Mahalanobis distance. Since the support vectors (SVs) most likely lie on the class cloud boundaries or within the class convex hull, we can use the Mahalanobis distance to rank the training set, such that, the samples with the largest distances are used first as part of the working set.

Experimental results suggest a reduction in the total training time, and a more dramatic decrease in the total iterations percentage. Results also suggest that, using the speedup strategy the support, the SVs are found early in the learning process.

Furthermore, the speedup strategy was tested with other methods with similar results, suggesting that the proposed approach is not particular to LP-SVR but rather useful for other SV-based methods.

4.8.2 Model Selection

An algorithm for LP-SVR model selection has also been discussed in this chapter. We discussed a quasi-Newton method with line-search that is applied to a function minimization problem. The method's Jacobian is computed via finite difference. We have explored the case of two-class, multi-class, and regression problems.

The proposed approach uses a \mathcal{K} -fold cross validation technique as a true test generalization error estimator. Also, particular error functions were defined for each type of problem: two, multi-class, and regression. The combination of a proper selection of good error functions, stable true generalization error estimator, and powerful optimization technique, resulted in a robust algorithm.

Simulation results suggest that the algorithm performs an estimation of hyper-parameters that produce a good minimization of the true test generalization error. The proposed model selection strategy was tested only with LP-SVR; however, there is no evidence that the strategy is particular to LP-SVR but rather is a generalized approach to classification methods. The experimentation of estimating other classifiers parameters remains as future work.

Chapter 5

Power Load Prediction

This chapter presents an important application to short term electricity load prediction with the proposed large-scale linear programming support vector regression (LP-SVR) model. The LP-SVR is compared with other two non-linear regression models: Feed Forward Neural Networks (FFNN) and Bagged Regression Trees (BRT). The three models are trained to predict hourly day-ahead loads given temperature predictions, holiday information and historical loads. The models are trained on hourly data from the New England Power Pool (NEPOOL) region (courtesy ISO New England) from 2004 to 2007 and tested on out-of-sample data from 2008. The models are shown to produce highly accurate day-ahead predictions with average errors around 1 – 2%. Note that in this chapter the term “load” refers to the electric power consumed by the public electric network circuit.

5.1 Introduction

Accurate load predictions are critical for short term operations and long term utilities planning. The load prediction impacts a number of decisions (*e.g.*, which generators to commit for a given period of time) and broadly affects wholesale electricity market prices [193]. Load prediction algorithms also feature prominently in reduced-form hybrid models for electricity price, which are some of the most accurate models for simulating markets and modeling energy derivatives [211].

Traditionally, utilities and marketers have used commercial software packages for performing load predictions. The main disadvantage of these is that they offer no

transparency into how the load predict is calculated. They also ignore important information, *e.g.*, regional loads and weather patterns. Therefore, they do not produce an accurate prediction.

This study considers several variables to build a prediction model and compares results among the Linear Programming Support Vector Regression (LP-SVR), Feed Forward Neural Network (FFNN), and Bagged Regression Trees (BRT). It is shown that the proposed LP-SVR model provides better forecasts than FFNN and BRT approaches.

5.2 Dataset

The dataset used for this electricity load prediction problem includes historical hourly temperatures and system loads from the New England Pool region. The original dataset was obtained from the New England ISO. At the time of writing this dissertation, the direct link to Zonal load data was the one shown in [91]. Table 5.1 shows the variables included for predicting the electricity load. These variables are called

Table 5.1: Variables Used for Prediction

Number	Description	Domain
1	Dry bulb temperature	\mathbb{R}
2	Dew point temperature	\mathbb{R}
3	Hour of day	\mathbb{Z}_+
4	Day of the week	\mathbb{Z}_+
5	Holiday/weekend flag	$\{0, 1\}$
6	Previous 24-hr average load	\mathbb{R}_+
7	24-hr lagged load	\mathbb{R}_+
8	168-hr (previous week) lagged load	\mathbb{R}_+

features. The features to consider are the bulb and dew temperature, the hour of the given day, the day of the week, and whether it is a holiday or weekend. Also, the features include the average load of the previous 24 hours, the lagged load of the previous 24 hours, and the previous week lagged load.

The training set $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{tr}}$ consists of input feature vectors $\mathbf{x}_i \in \mathbb{R}^8$ (consistent with the variables listed in Table 5.1) and targets d_i corresponding to the measured electricity load. The total number of training samples is $N_{tr} = 35064$. A testing set $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{te}}$ was also used, consisting of $N_{te} = 8784$ samples.

Each feature vector \mathbf{x}_i corresponds to one hour reading, *i.e.*, one complete day would be equivalent to 24 sequential feature vectors $\{\mathbf{x}_{i+1}, \mathbf{x}_{i+2}, \dots, \mathbf{x}_{i+24}\}$. Consequently, the training set consists of 1461 days, or four years of data. The testing set consists of one leap year of data or 366 days.

5.3 Training the Regression Models

The regression models will be constructed using the training set \mathcal{T} . The training procedure involves a training set partition into a new training set and a validation set \mathcal{V} , which is used to auto-adjust model parameters during the learning process. Once the model is trained and internally validated a testing phase follows in order to estimate the true performance errors of the models with unseen data. The complete regression modeling framework is shown in Figure 5.1. The actual regression models used in this study are briefly introduced in the following sections.

5.3.1 Feed-Forward Neural Network

The first regression model used was based on neural networks. In fact, this study uses the Feed-Forward Neural Network architecture introduced in Section 7.5 with very few modifications: the output is a single neuron, the activation function of the output is linear, the network has 20 neurons in the hidden layer, and the network uses the mean of absolute error (MAE) metric as the error function to minimize during training.

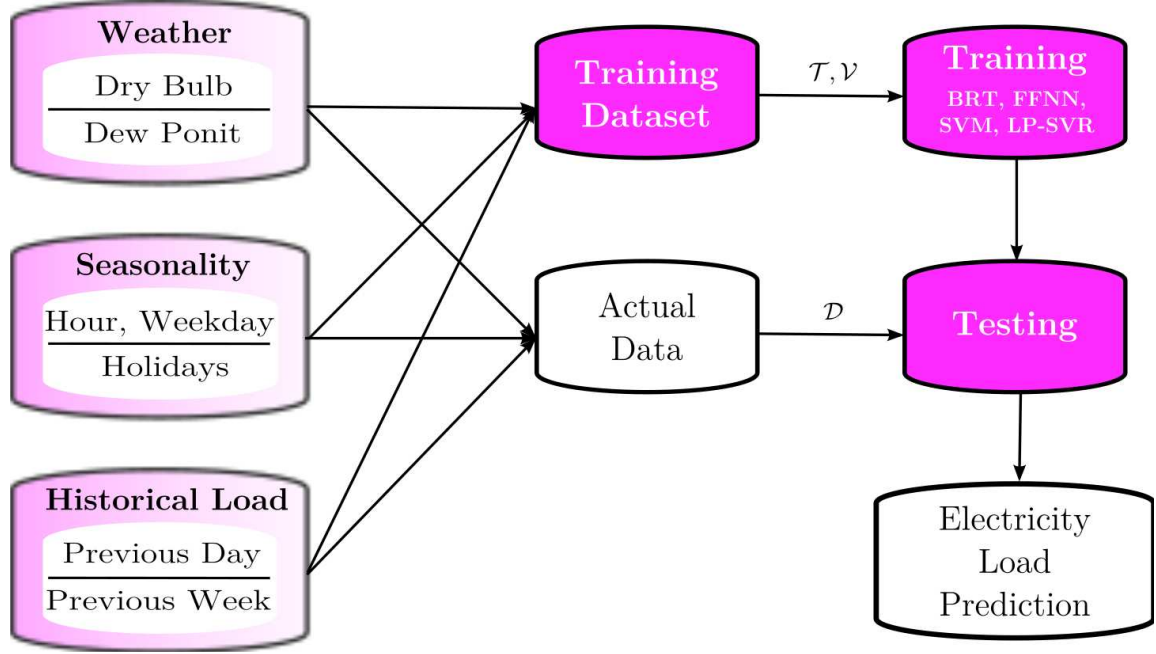


Figure 5.1: Framework to build regression models for power load prediction. Blocks on the left indicate the input variables *i.e.*, attributes used to build the regression models.

5.3.2 Bagged Regression Trees

Bagging stands for “bootstrap aggregation,” which is a type of ensemble learning [14]. The algorithm in general works as follows. To *bag* a regression tree on a training set $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{tr}}$, the algorithm generates several bootstrap clones of the training set and grows regression trees on these clones. These clones are obtained by randomly selecting N_{tr} samples out of N_{tr} with replacement. Then, the predicted response of a trained ensemble corresponds to the average predictions of individual trees [44].

The process of drawing N_{tr} out of N_{tr} samples with replacement omits an average of 37% samples for each regression tree. These are called “out-of-bag” observations. These out-of-bag observations are used as a validation set \mathcal{V} to estimate the predictive power. The average out-of-bag error is computed by averaging the out-of-bag predicted responses versus the true responses for all samples used for training. This

average out-of-bag error is an unbiased estimator of the true ensemble error and can be used to auto-adapt the learning process [14].

5.3.3 Large-Scale Support Vector Regression

Included in this study is the large-scale support vector regression (LS SVR) training strategy by Collobert, *et al.* [39], considered the most popular LS-SVR training strategy (see Section 2.3).

Collobert, *et al.* algorithm is an adaptation of Joachims' SVM method for SVR problems [39]. Also, the algorithm performs the same decomposition proposed by Osuna (see Section A.2.1). In summary, this method uses a decomposition algorithm, a chunking approach, and a shrinking strategy.

5.3.4 Linear Programming Support Vector Regression

Finally, this study also includes the LP-SVR formulation (3.2). Algorithm 3.4 was used to train the LP-SVR for the complete training set. The LP-SVR parameters used are $\sigma = 0.125$, $C = 0.5$, and $\epsilon = 0.1$; these have been found with Algorithm 4.2 from Section 4.5.2.

5.4 Experimental Results

5.4.1 Experiment Design and Procedure

The experiments consisted of training the four methods with a training dataset $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{tr}}$ as explained in Section 5.2. Then the following six error metrics were analyzed using the testing set $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{te}}$: Mean Absolute Percent Error (MAPE), Mean Absolute Error (MAE), Daily Peak MAPE (DPM), Normalized Error (NE), Root Mean Squared Error (RMSE), and Normalized Root Mean Squared Error (NRMSE).

The mean absolute percent error can be computed with the following equation:

$$\text{MAPE} = \frac{1}{N_{te}} \sum_{i=1}^{N_{te}} \left(\frac{|y_i - d_i|}{y_i} \times 100 \right), \quad (5.1)$$

where y_i is the i -th observed regression model output corresponding to the i -th input vector \mathbf{x}_i . The Mean Absolute Error is estimated with (C.6). The Daily Peak MAPE consists on analyzing the MAPE in a daily fashion. That is, within the testing set $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{te}}$ choose segments corresponding to a complete day:

$$\{\mathbf{x}_{i+1}, d_{i+1}\}, \{\mathbf{x}_{i+2}, d_{i+2}\}, \dots, \{\mathbf{x}_{i+24}, d_{i+24}\}$$

and observe the predicted daily output $\{y_{i+1}, y_{i+2}, \dots, y_{i+24}\}$ then estimate the peak MAPE of that day. Formally, the DPM can be defined as follows. Let a denote the number of days available in the testing set. Let \mathcal{J} be the set of sample indices corresponding to the different number of days:

$$\mathcal{J} = \{\mathcal{J}_1, \mathcal{J}_2, \dots, \mathcal{J}_a\}$$

where \mathcal{J}_a denotes the set of indices corresponding to samples of a -th day. Then the Daily Peak MAPE is obtained as follows:

$$\text{DPM} = \frac{1}{a} \sum_{k=1}^a \left(\arg \max_{i \in \mathcal{J}_k} \left[\frac{|y_{i \in \mathcal{J}_k} - d_{i \in \mathcal{J}_k}|}{y_{i \in \mathcal{J}_k}} \right] \times 100 \right). \quad (5.2)$$

The following equation is used to compute Normalized Error:

$$\text{NE} = \frac{\|y_i - d_i\|_2}{\|y_i\|_2}, \quad (5.3)$$

while the Root Mean Squared Error and Normalized Root Mean Squared Error are computed with (C.7) and (C.8) respectively.

5.4.2 Quantitative Results

Table 5.2 shows quantitative prediction errors using the metrics explained above: MAPE, MAE, DPM, NE, RMSE, and NRMSE.

Table 5.2: Electricity Load Prediction Errors

Error Measure	Units	BRT	FFNN	LS SVM	LP-SVR
MAPE	%	2.18	1.61	3.52	1.58
MAE	MWh	330.08	243.18	491.38	238.69
DPM	%	2.21	1.63	2.62	1.58
NE	—	0.030	0.022	0.040	0.021
RMSE	—	459.115	335.048	608.583	326.468
NRMSE	—	0.162	0.118	0.215	0.115

According to results in Table 5.2, the proposed LP-SVR model performs with lower error than BRT, FFNN, and LS SVM. This result is consistent for all the metrics. However, very small differences can be observed between the performance of FFNN and LP-SVR. This can be confirmed by observation in Figure 5.2 and Figure 5.3.

Figure 5.2 (top) shows a two-day window of true load compared with the predicted load for the four different methods, and also (bottom) shows the error residuals for the four methods. As expected, the results of FFNN and LP-SVR exhibit very little difference. In general most methods predict the true model to a relative low error. Figure 5.3 shows a particular two-day window for Christmas Eve. As for many holidays, Christmas Eve is very difficult to predict due to the high variability in electricity consumption. The figure demonstrates a considerable high prediction error between 14:00-21:00 Hrs on 12/24/2008.

Figure 5.4 shows the error distribution for the different methods. It can be concluded that FFNN and LP-SVR have smaller error variances. Similarly, Figure 5.5 illustrates the absolute error distribution, including the mean absolute error for each of the four methods. It can be seen that both FFNN and LP-SVR have almost the same MAEs.

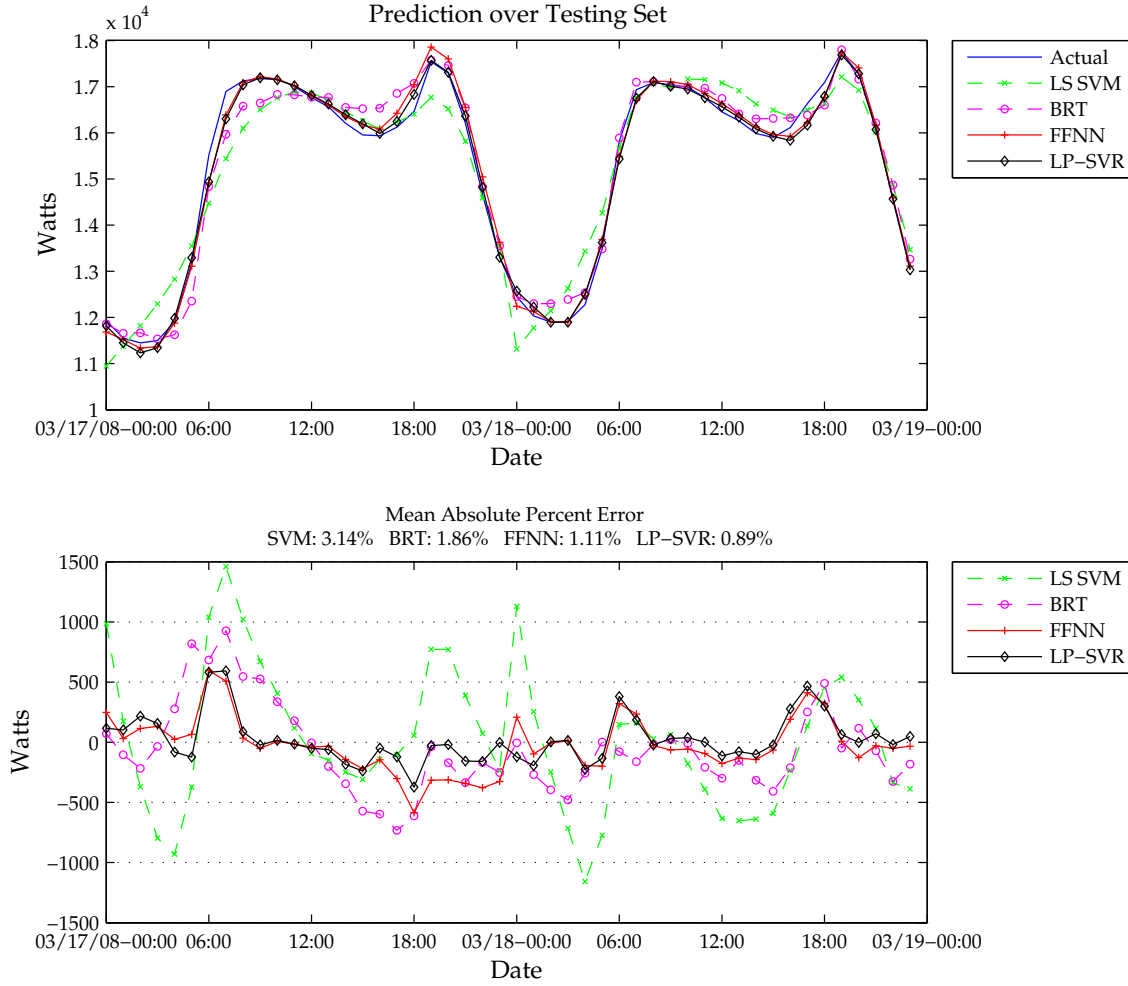


Figure 5.2: Two-day window of true data compared with predicted for the four different methods (top). Error residuals for the four methods (bottom).

An interesting analysis is the average error visualization by hour of day, shown in Figure 5.6. It can be seen that early morning hours (00:00-05:00) are the most “easy” to predict, *i.e.*, can be predicted with very small error. In contrast, the late morning trough afternoon hours (06:00-22:00) are predicted with larger errors.

Figure 5.7 illustrates the average error by day of the week. Clearly, the days that produce higher errors are those associated with Mondays through Fridays, that represent the work week. It is important to notice the error scale between Figure

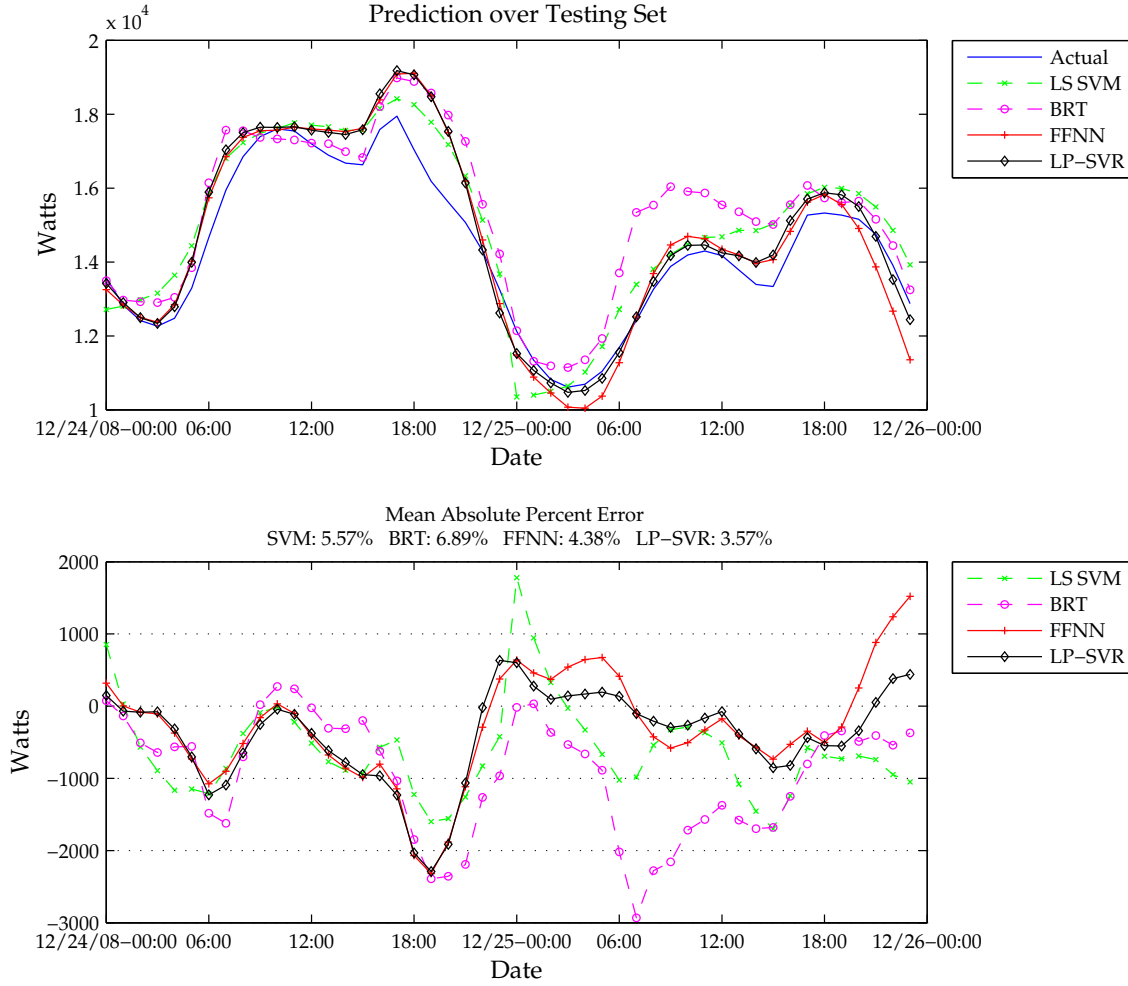


Figure 5.3: Christmas two-day window of true data compared with predicted for the four different methods (top). Error residuals for the four methods (bottom). Note the high prediction error between 14:00-21:00 Hrs.

5.6 and 5.7. In Figure 5.6 the largest error is below 1.8×10^4 , while in Figure 5.7 the largest error is below 1.6×10^4 . This implies that errors are expected to be greater if the prediction is based on hourly data. From this one can conclude that the prediction is more independent of the day of the week, and more dependent on the hour of the day.

The final analysis is in regard to the statistical properties of the errors of the

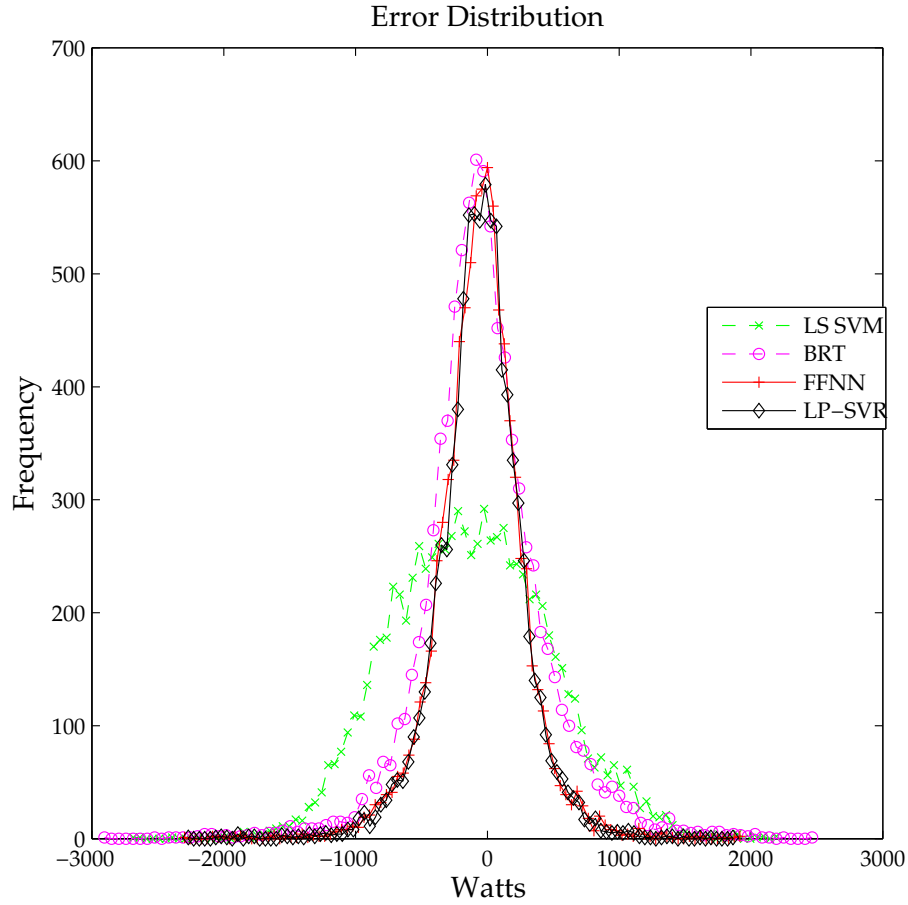


Figure 5.4: Error distribution for the LS SVM, BRT, FFNN, and LP-SVR regression methods. The methods with smallest variances are FFNN and LP-SVR.

proposed LP-SVR model. Figures 5.8 through 5.10 show statistical plots known as “box plots.” These plots provide the following information: on each box, the central mark is the median, the edges of the box are the 25-th and 75-th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers (+) are plotted individually. In terms of error measures, it is desired that the box plots have a very small box close to zero on the error axis, the median should be close to zero, the extrema points should be close to the box, and of course no outliers are desired.

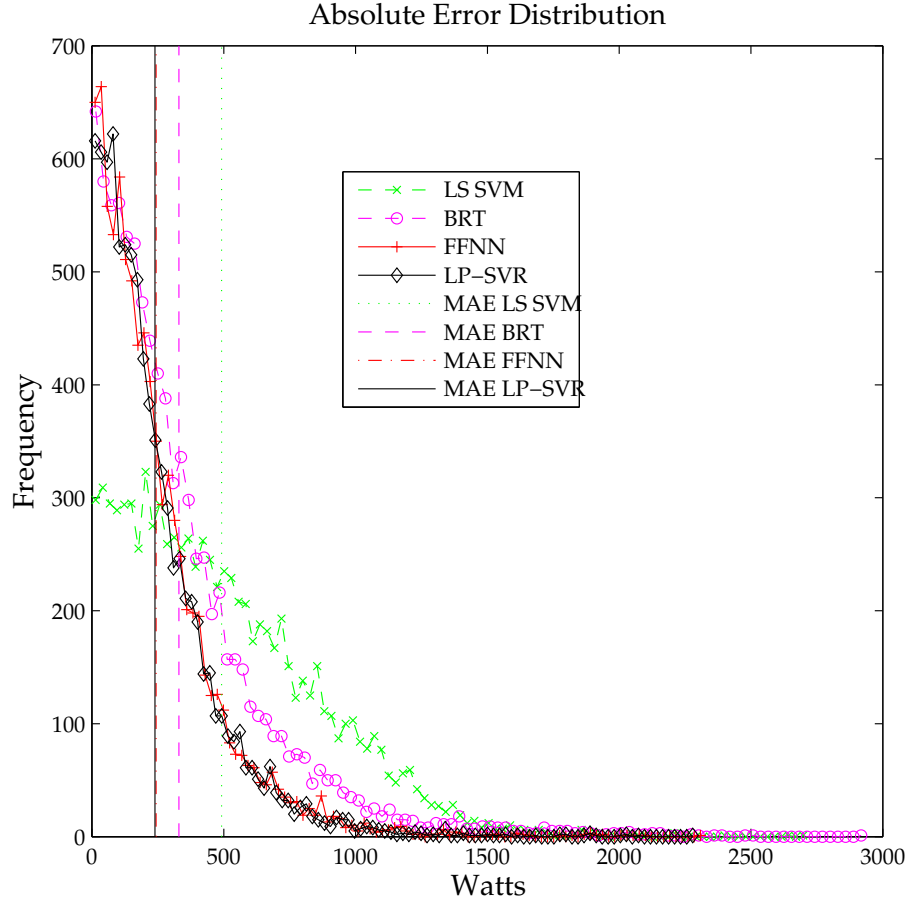


Figure 5.5: Absolute error distribution of the LS SVM, BRT, FFNN, and LP-SVR regression methods. The vertical lines indicate the mean absolute error for each of the four methods as reported in Table 5.2.

An hourly breakdown of the LP-SVR mean absolute prediction error is shown in Figure 5.8. It can be noticed that the early morning hours have smaller variability. Then a daily breakdown of the LP-SVR mean absolute prediction error appears in Figure 5.9, from which one can see that Mondays and Fridays have the largest average errors and that Fridays have many outliers. Finally, a monthly breakdown is shown in Figure 5.10. This figure clearly shows that the months of November and December exhibit the largest average errors, have the largest variabilities, and show many outliers.

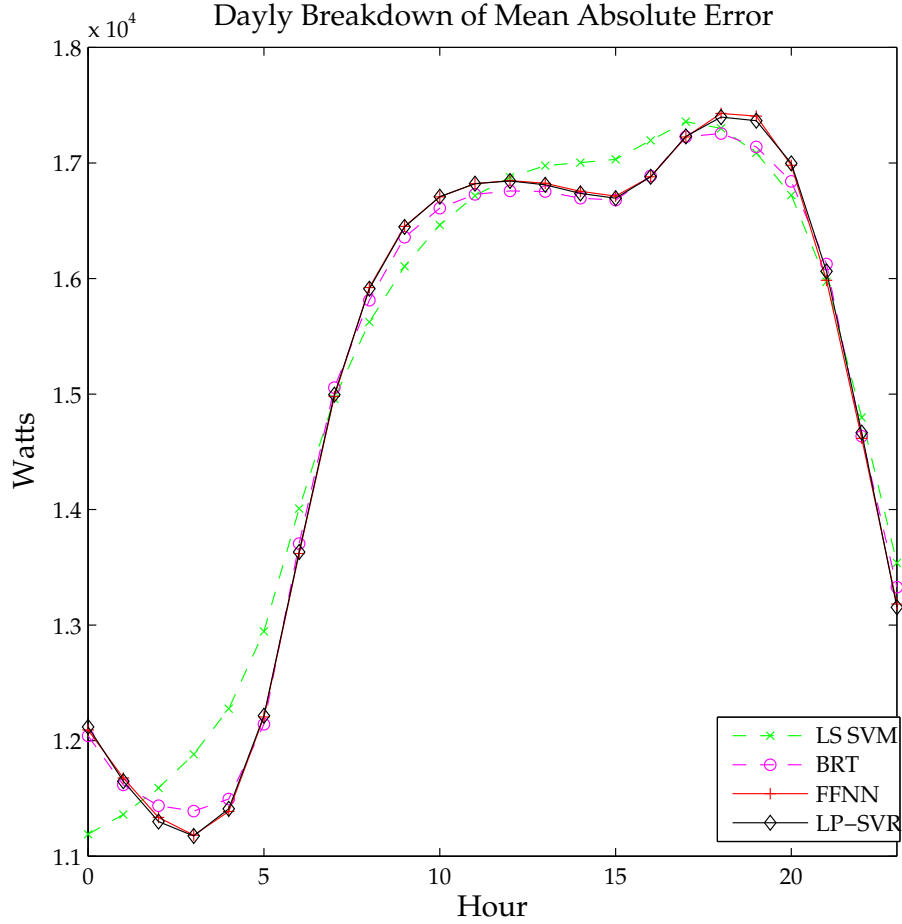


Figure 5.6: Average error by hour of day. Note the error proportional difference in early morning hours and afternoon hours.

5.5 Conclusion

This chapter presents an application of the proposed LP-SVR model to electricity load prediction. A number of eight different variables are utilized to construct regression models. The study includes a comparison of the LP-SVR model against other state of the art methods, such as FFNN, BRT and LS SVM.

Experimental results indicate that the proposed LP-SVR method gives the smallest error when compared against the other approaches. The LP-SVR shows a mean absolute percent error of 1.58% while the FFNN approach has a 1.61%. Similarly,

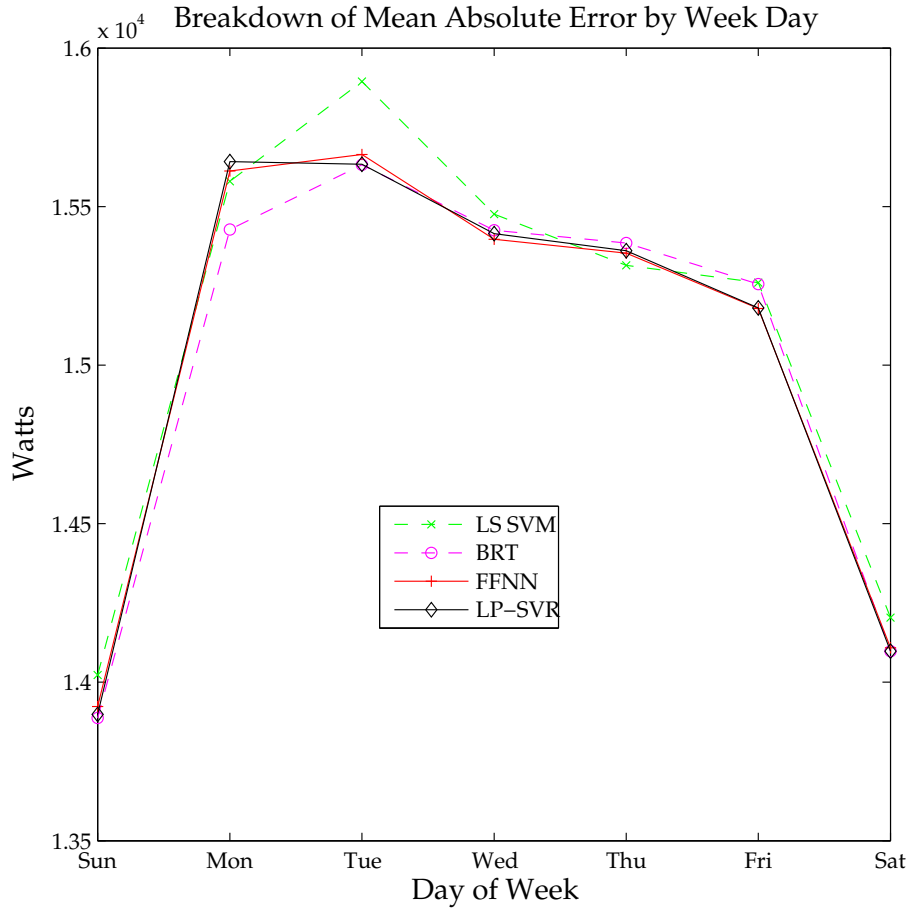


Figure 5.7: Average error by day of week. Note the error proportional difference in working and non-working days.

the FFNN method shows a 330MWh (Megawatts-hour) mean absolute error, whereas the LP-SVR approach gives a 238MWh mean absolute error. This is a significant difference in terms of the extra power that would need to be produced if FFNN was used.

The proposed LP-SVR model can be utilized for predicting power loads to a very low error, and it is comparable to FFNN and over-performs other state of the art methods such as: Bagged Regression Trees, and Large-Scale SVRs.

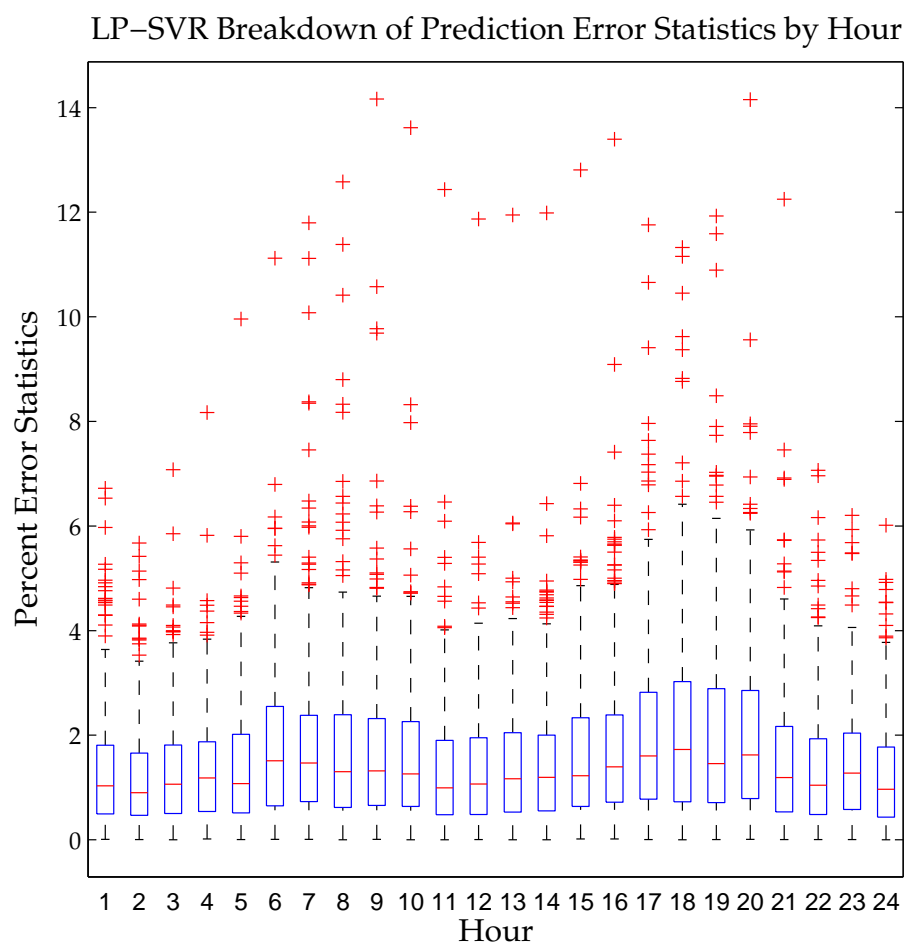


Figure 5.8: Hourly breakdown of the LP-SVR mean absolute prediction error. Note that the early morning hours have smaller variability.

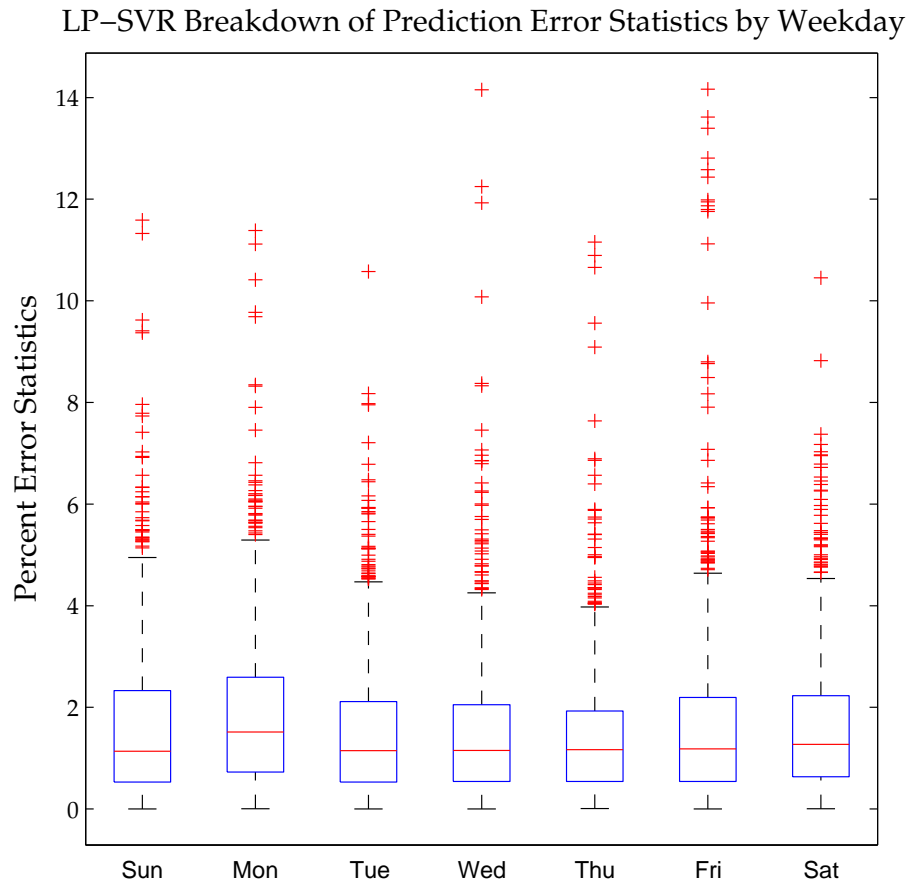


Figure 5.9: Daily breakdown of the LP-SVR mean absolute prediction error. Note that Mondays and Fridays have the largest average error.

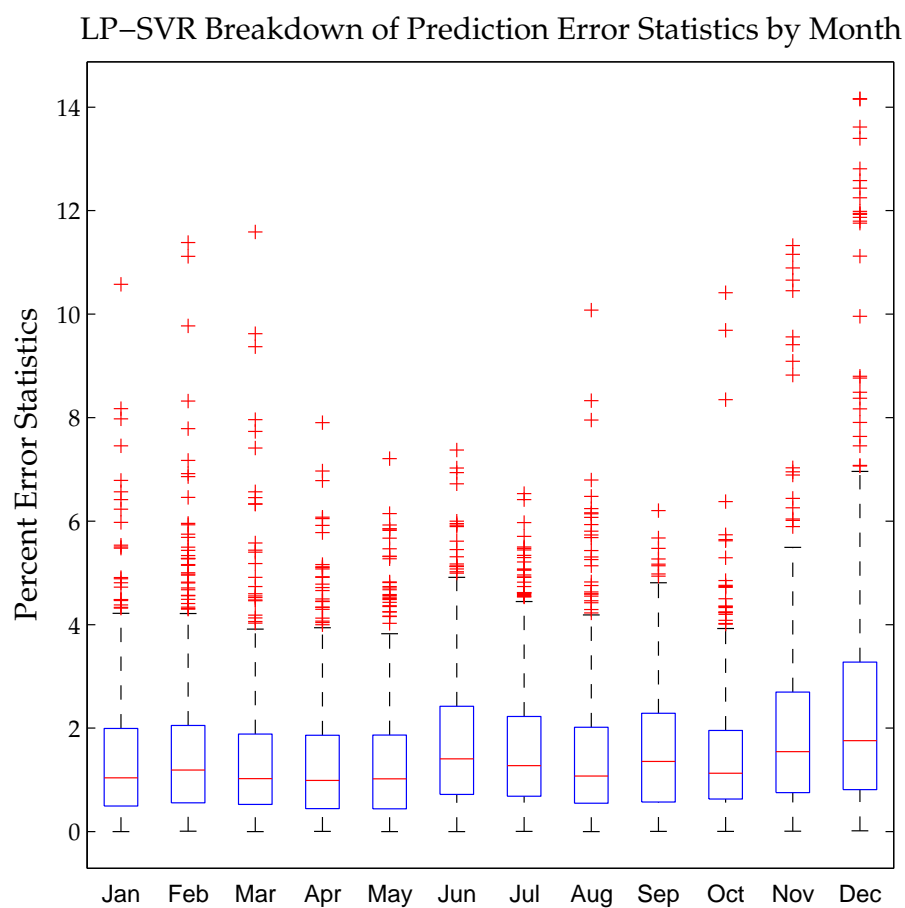


Figure 5.10: Monthly breakdown of the LP-SVR mean absolute prediction error. Note that the month of December exhibits the largest average error, and the largest variability.

Chapter 6

Texture Classification with LP-SVR and Biorthogonal Directional Filter Banks

The main contribution of this chapter is the use of the Linear Programming Support Vector Regression (LP-SVR) approach for texture segmentation using directional image pyramids to compute input features. We will demonstrate that the proposed classifier results are either superior or very competitive with respect to previously reported work. Furthermore, we also introduce the usage of LP-SVR in a cascaded and ensemble architectures to improve the performance of the single-instance LP-SVR.

6.1 Introduction to Texture Analysis

In image analysis, texture is well recognized as an important attribute for the recognition and interpretation of visual information. However, there is no uniformly accepted definition of texture. Generally speaking texture is related to the consistency of graininess or patterning over a surface. The visual variations between textures can go from random/stochastic to highly structured, with many textures containing both elements at different degrees.

Image segmentation is one of the early mechanisms from our visual system. From an image processing and computer vision perspective, the final outcome of segmen-

tation is the partitioning of an image into a coherent map of labels. The criteria for coherence can be tonal homogeneity, color and/or texture. Although our vision system is able to perform segmentation effortlessly, computational segmentation has remained an open area of research for a few decades.

Texture segmentation aims at partitioning an image in regions that display similar textural properties. The goal is to define and extract discriminative features which can be used to assign a texture class to each pixel on the image. Related problems are the detection of boundaries among texture regions and determining the number of textures composing an image.

Multichannel texture segmentation has been extensively studied in the literature. The idea is to apply some type of 2-D filter bank to an image in order to separate detail at different scales and orientation.

Intuition suggests that the textures composing the image will respond differently to each filter. For instance, a structural texture will have most of its energy concentrated at certain channels, while stochastic textures will have their energy more evenly distributed across bands. Measuring the local interaction of the subband coefficients allows us to characterize the overall properties of the texture.

Multichannel texture segmentation was pioneered by Laws in [111] using 25 simple 2-D separable kernels. Subsequently, a lot of attention was placed on Gabor functions because their spatial frequency response resembled the receptive fields of neurons on the visual cortex from mammals. The work of Jain and Farrokhnia [93] explored this relationship in detail from an engineering perspective. A drawback with Gabor functions is that the implementation is computationally expensive because the 2-D *discretized* filters are large and non-separable. To deal with complexity, other image decompositions have been used as the front end of the segmentation system. For instance, work with the DCT [153], wavelet transforms [119, 149], wavelet frames [189], and complex wavelet transforms [106] has been reported with different degrees of success.

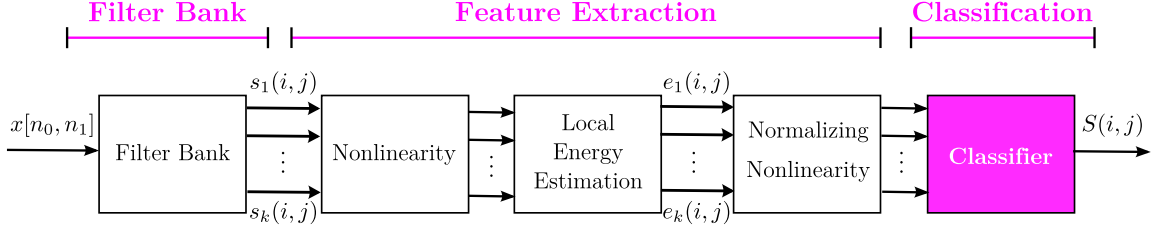


Figure 6.1: Classical segmentation system based on multichannel filtering.

A texture segmentation system based on multichannel filtering consist of three stages as depicted in Figure 6.1. The contribution of this chapter is to introduce the LP-SVR at the fifth stage which uses directional pyramids based on the Bamberger Directional Filter Bank.

6.2 The Bamberger Directional Filter Bank

Directional filter banks (DFBs) were originally conceptualized by Bamberger and Smith [11]. DFBs represented a relevant contribution within the image processing community. The DFB concept was further improved and expanded by the work from Park, *et al.* [143] and Rosiles and Smith [163–165] among others. Here we present the version of the DFB developed by Rosiles and Smith [163] which was used as the front end of a texture segmentation system reported in [164].

The DFB is a two-dimensional (2-D) filter bank that separates image detail (*i.e.*, edges and textures) according to their orientation. The DFB implements wedge-shaped filters that partition the 2-D frequency plane as depicted in Figure 6.2. These filters are difficult to design and computationally expensive to implement if one expects good filter properties. Bamberger found an efficient implementation through the wise use of 2-D multi-rate system properties. Figure 6.3 presents the implementation of the DFB as a tree structure, in which each stage is composed on a fan filter bank (FFB). The structure of the FFB and the support of the fan filters is also presented in Figure 6.4. As shown by Rosiles and Smith [161, 163, 165], the FFB

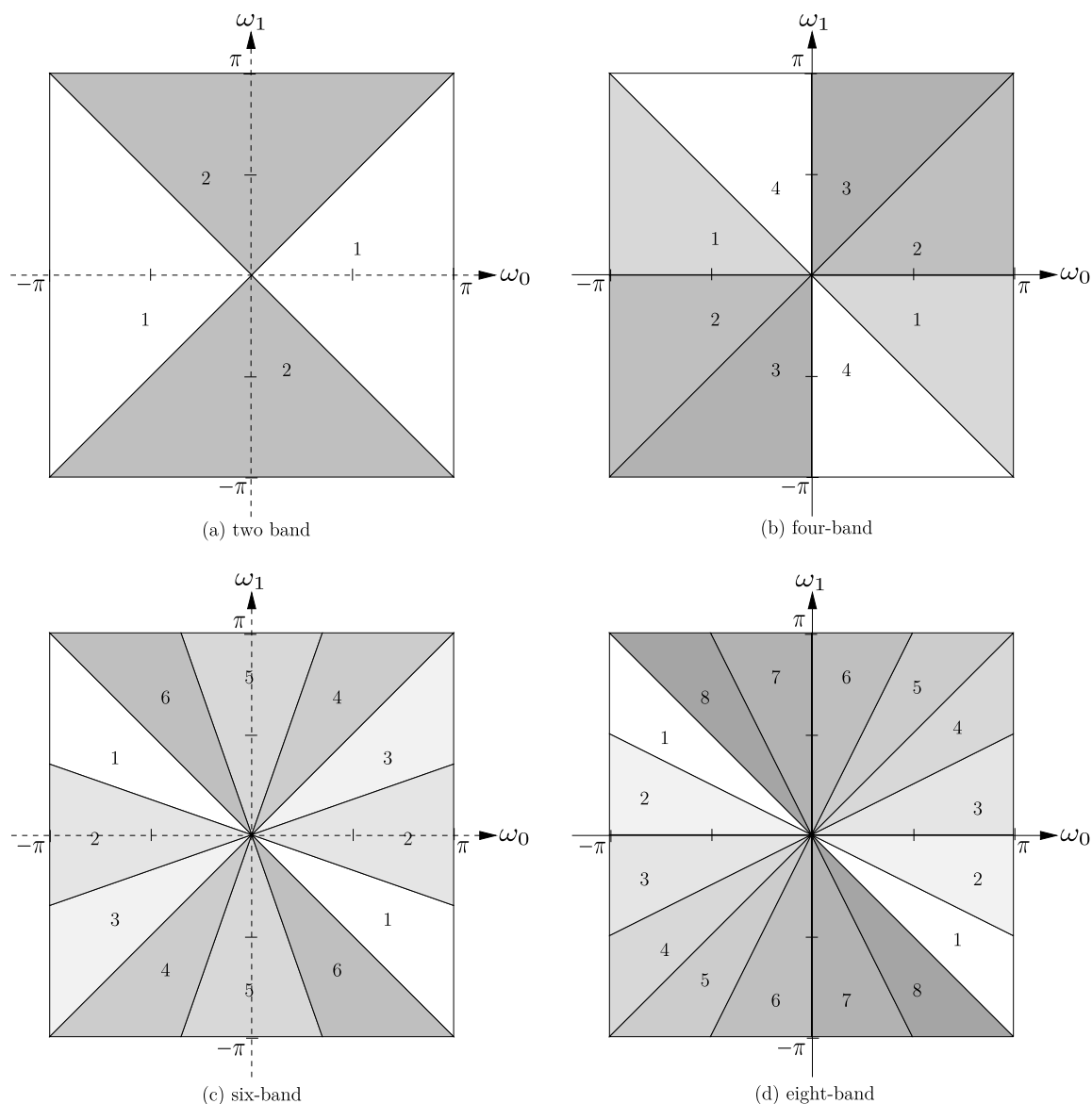


Figure 6.2: Frequency band partitions obtained using Bamberger DFB.

can be efficiently implemented using the ladder structures [8] (a.k.a. lifting structures) depicted in Figure 6.5. It is relevant to note that the filtering operations are implemented in row-column order using a one-dimensional (1-D) filter $\beta(z)$. Ladder structures represented a breakthrough for DFB theory. First, the 2-D filters are easy to implement using 1-D filters designed by the Parks-McClellan algorithm or other well known design methods. These filters are length L polynomials with an all-pass

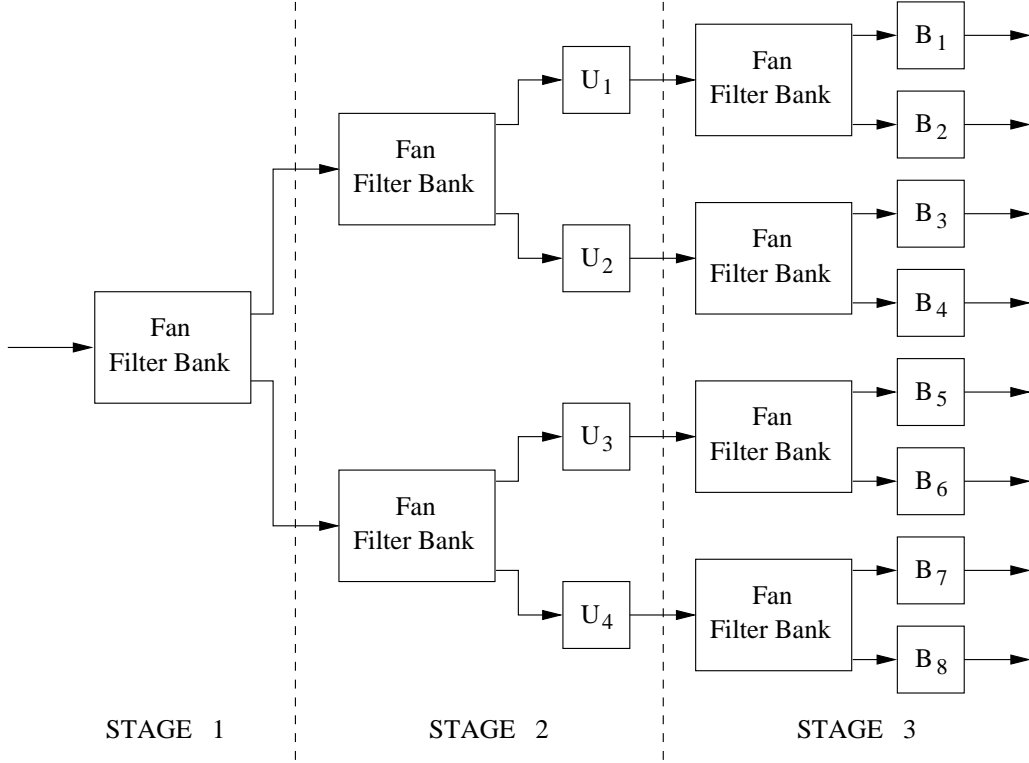


Figure 6.3: Eight-band DFB implementation with a tree structure with FFBs and backsampling matrices.

behavior. Moreover, any biorthogonal wavelet can be generalized to a 2-D structure for directional filtering purposes. Finally, the use of ladder structures, warranties perfect reconstruction. For more details on the formulation of this DFB the reader can consult [161, 165].

A key component of the DFB is the use of resampling matrices \mathbf{M} , \mathbf{U}_i , and \mathbf{B}_i in Figures 6.5 and 6.3. The \mathbf{M} matrices are 2×2 quincunx matrices that can upsample or downsample an image while inflicting a 2-D rotation by 45 degrees. The \mathbf{U}_i and \mathbf{B}_i are unimodular matrices have the effect of changing the sampling lattice without reducing the data rate. For a detailed explanation on 2-D sampling theory we refer the reader to [191]. Hence, by combining these resampling operations, the data is rotated and skewed in order to pose it in the correct orientation such that it can be

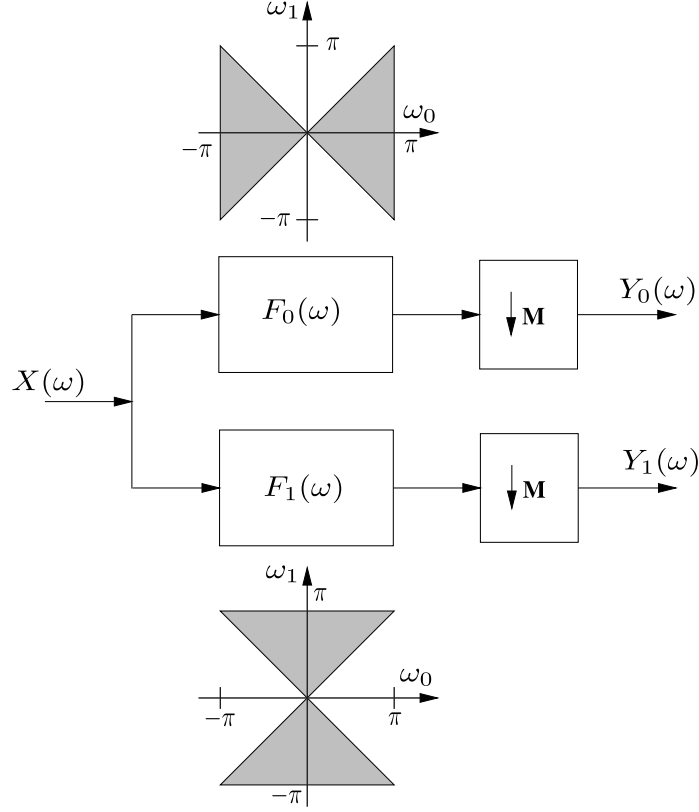


Figure 6.4: Block diagram for the FFB showing the ideal support for the 2-D filters.

filtered with the typical row-column processing shown in the ladder structure. The final unimodular matrices, \mathbf{B}_i , are used to re-align the sampling lattice to the correct row-column orientation (*i.e.*, they undo all the rotation and skews performed by the DFB).

Finally, another relevant contribution of the work in [163] is the derivation of the undecimated DFB (UDFB). It was shown that using a careful choice of resampling operations, the DFB downsampling operations could be removed to produce a UDFB where each output subband had the same dimensions as the input while retaining the same implementation efficiency. Hence for an N DFB, there is an expansion on the data by a factor of N . As opposed to the maximally-decimated DFB, the UDFB retains the shift invariance property which is important for pattern recognition

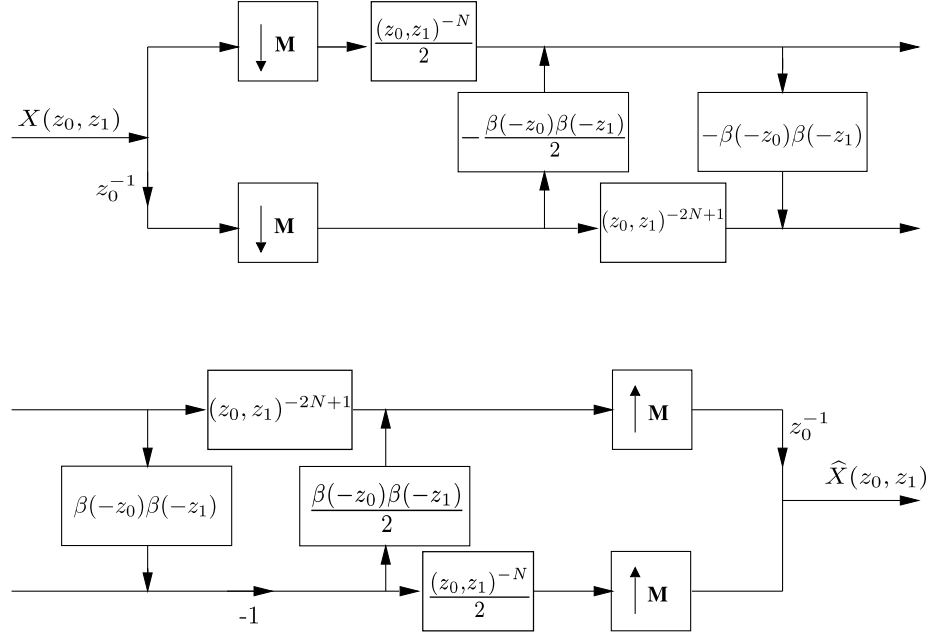


Figure 6.5: Ladder structure implementation of a 2-D two-channel biorthogonal filter bank.

problems, and simplifies the establishment of spatial relationships between pixels and coefficient and across different bands. This was difficult to achieve on the original DFB which produced subbands with non-uniform dimensions.

6.3 Multiresolution Directional Pyramids

The energy of most natural images is concentrated at low frequencies. If we decomposed a natural image directly with a DFB or a UDFB, the non-ideal wedge-shaped passbands of the filters would introduce low frequency energy in the subbands. Since directional information is related to mid and high frequency information, such as edges and textures, lowpass information can hinder our ability to capture and analyze these features. Moreover, the DFB is not capable of distinguishing between structures and detail at different scales. Hence, this limits the DFB ability to perform multiscale/multiresolution analysis.

To deal with these issues, structures that combine the DFB and UDFB with image pyramids have been proposed by Rosiles [165]. This concept is briefly introduced next.

6.3.1 Simple Lowpass-Highpass decomposition

Even if we wanted lowpass energy on the directional subbands, the non-ideal nature of the filters would preclude us to do so. This non-ideal nature becomes evident at the origin. Figure 6.6 shows how some of the channels fall short at reaching the origin and the vertex is blunt. Other magnitude responses straddle the origin and the passbands are not sharp around the origin. Consequently, uneven DC information distribution may be observed across bands. Since much of the directional information of interest resides on the mid to high frequencies, DC energy absence is usually desired.

Figure 6.7 shows an approach to deal with this issue: a “lowpass-highpass” decomposition structure. The image is first filtered with a lowpass filter $L_{\omega_c}(z_0, z_1)$ with cutoff frequency ω_c . To ameliorate computational complexity, $L_{\omega_c}(z_0, z_1)$ can be computed with a row-column separable 2-D filter. A simple difference with the original image produces the high frequency component used to perform the directional analysis. We note that in Figure 6.7, the residual is processed by the DFB, however we can also use the UDFB. If the subbands require no further processing, the decomposition is invertible.

6.3.2 Directional Pyramids

The 2-D discrete wavelet transform (DWT), the steerable pyramid [175], the complex-valued wavelet transform [106], 2-D Gabor representations [21, 93], and other image decomposition approaches have a multi-resolution (MR) or a pyramidal component. This component could be an implicit part of the decomposition (*e.g.*, the DWT), or could be implemented as a separate structure (*e.g.*, the steerable pyramid [175]). In

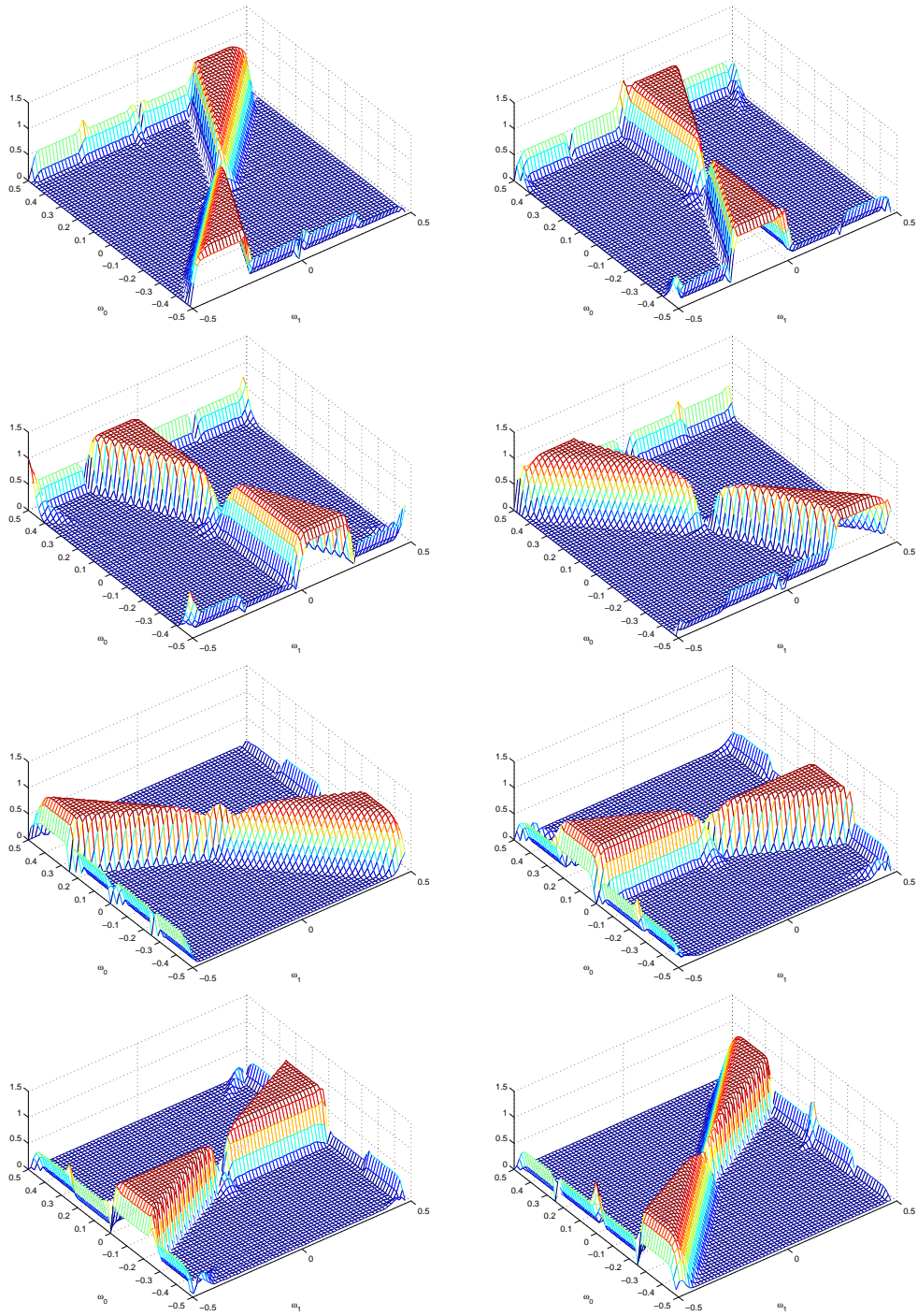


Figure 6.6: Magnitude Response of the synthesis filters of an eight band biorthogonal ladder DFB.

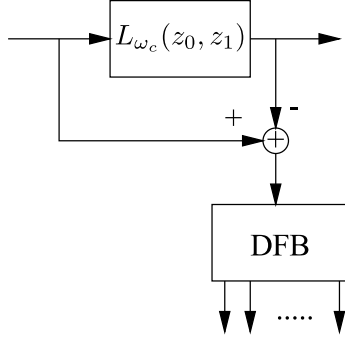


Figure 6.7: Lowpass-highpass analysis structure. The DFB or the UDFB can be used to do a directional decomposition.

the latter case, we say that the decomposition is polar-separable, implying that a radial frequency decomposition (*i.e.*, a pyramid) is performed independently of its directional decomposition. A comparison of the 2-D spectral partitioning between the DWT and a polar-separable structure can be made by observing Figure 6.8 and Figure 6.9 respectively.

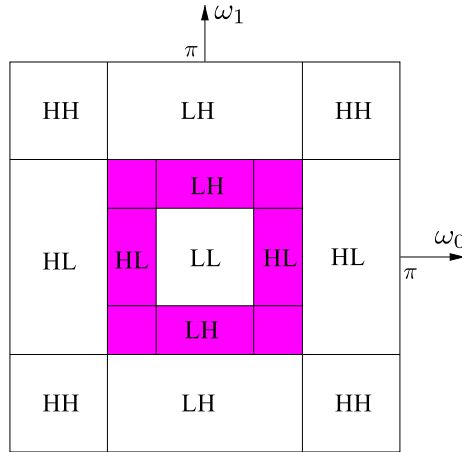


Figure 6.8: Frequency plane partitioning of the 2-D separable DWT.

For image analysis and processing, and particularly for image segmentation and classification, scale information has been used along with directional information to distinguish objects or features of different sizes. These polar decompositions were inspired by strong experimental evidence [48] pointing out that a similar kind of

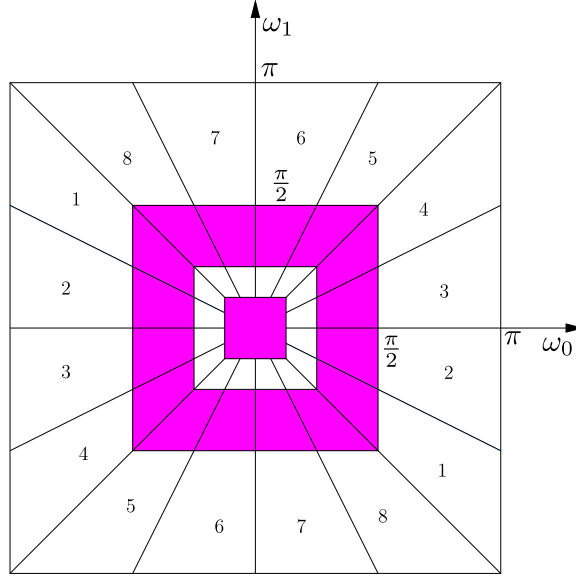


Figure 6.9: Frequency plane partitioning of the proposed directional pyramids.

processing takes place within the human visual system.

The Bamberger DFB is purely directional. Thus, we have the advantage to select the radial decomposition of our choice. Next, we introduce UDFB-based pyramidal structures. We will refer to these structures as Directional Pyramids (DPs) or Bamberger Pyramids [11]. We start by taking the output of a Laplacian or Laplacian-like pyramid [28] structure and further decompose the detail subbands with the UDFB producing large amounts of data. For the case where all radial bands are decomposed into an N -band UDFB, the data increase is given by a factor of $\frac{4N}{3}$. We refer to this structure as the Laplacian-UDFB pyramid.

A possible drawback of using a Laplacian-UDFB pyramid is that each detail level D_j of the pyramid is one quarter the size of the previous level D_{j-1} . This may produce accuracy problems and errors in the calculation of statistics.

To alleviate this drawback, we use structures in which the radial components are not downsampled. These modification to the original systems is shown in Figure 6.10. Similar to the undecimated DWT, we use one single lowpass prototype $L_{\frac{\pi}{2}}(z_0, z_1)$ which is modified for each resolution level as $L_{\frac{\pi}{2}}(z_0^{2^j}, z_1^{2^j})$ where $j = 0, 1, \dots, J - 1$.

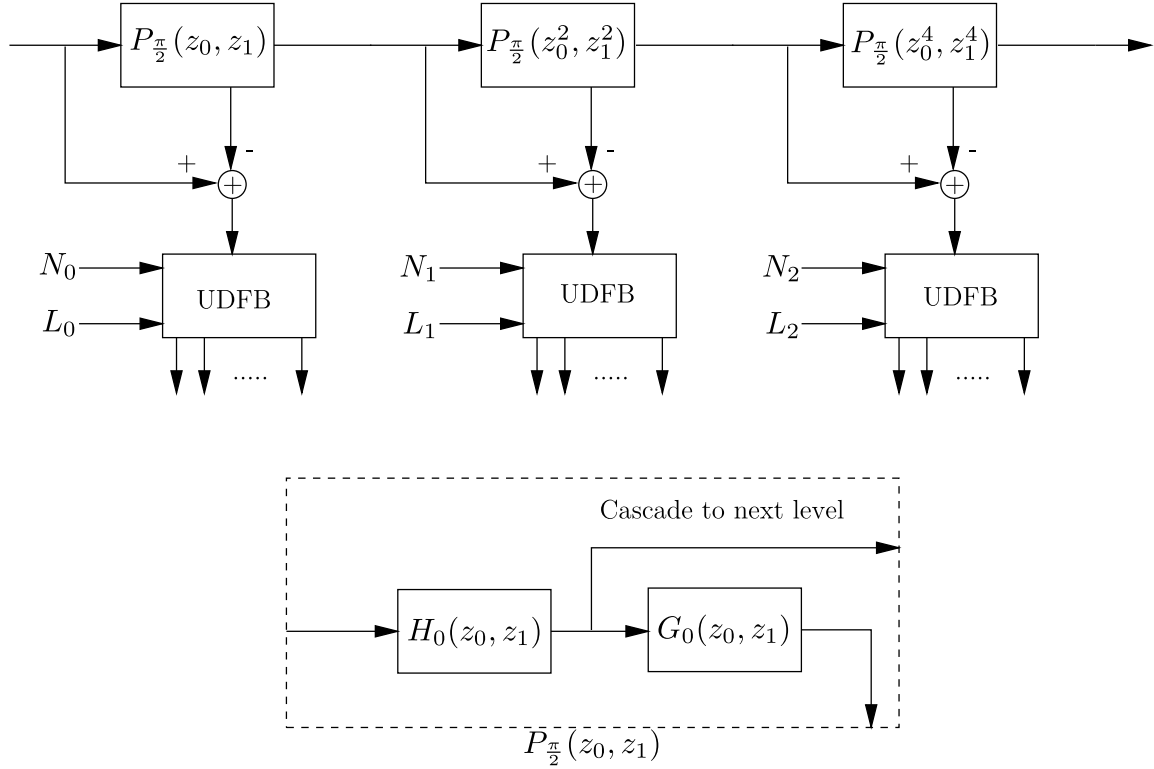


Figure 6.10: Invertible Undecimated Laplacian-UDFB pyramid structure with no decimation of the resolution components.

In the spatial domain this modification corresponds to inserting zeros in the impulse response of $L_{\frac{\pi}{2}}(z_0, z_1)$. When combined this with the UDFB, the data increases by a factor of $N(J - 1) + 1$. We identify this DP as the ULap-UDFB.

If we use the same number of directional bands for all J pyramid levels, all the subbands corresponding to the same orientation will have the same size. This is a nice feature which allows us to establish a one-to-one correspondence between subband coefficients at different resolutions, eliminating parent-child ambiguities for inter-subband processing.

A comparison of the frequency plane partitioning obtained with the DP structures and the traditional separable 2-D DWT can be made by observing Figure 6.8 and 6.9. We see that the 2-D DWT has limited angular sensitivity (diagonal, horizontal and

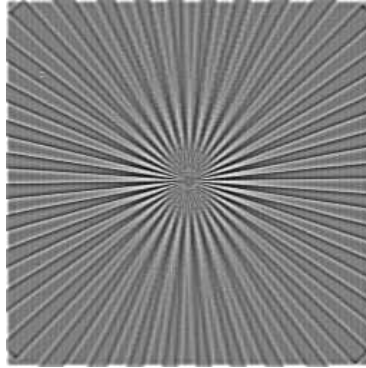


Figure 6.11: Test Image for UDBFs.

vertical directions), while the DP structures can be adapted to have 2^n directional subbands. However, it is important to note that the DWT is critically sampled, while DPs are moderately to substantially overcomplete, depending on which DP structure we select.

As an example, let us consider the test image shown in Figure 6.11 and its Laplacian-UDFB pyramid with decimation in Figure 6.12. We let $J = 3$ and $N = 4$ (note that the coarsest level consists only of a lowpass subband).

6.4 Feature Extraction

Features derived from the DFB provide excellent discriminability [165]. In this section, we perform texture segmentation using a well known multichannel system, and compare it against other segmentation schemes. Multichannel approaches aim to mimic our visual system to understand a scene according to its textural characteristics. In signal processing terms, the image received by the eye is decomposed into a representation whose partitioning of the 2-D frequency plane is similar to what is shown in Figure 6.9.

The feature extraction stage consist of a set of pre-conditioning operations whose objective is to reduce the variability within pixels of the same texture on channel

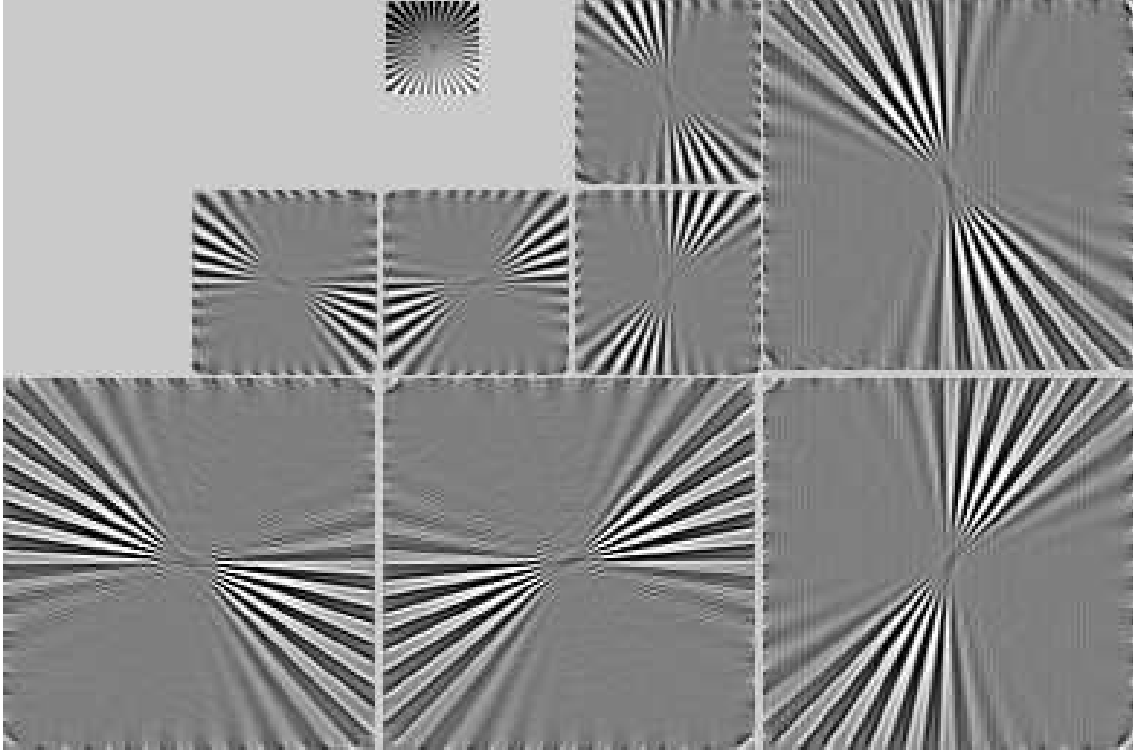


Figure 6.12: Laplacian-UDFB pyramid example with $J = 3$ and $N = 4$.

$s_k[i, j]$ such that texture regions approach uniformly shaded blobs easy to recognize.

The feature extraction stage consists of the second, third, and fourth operations depicted in Figure 6.1. First, each channel is passed through a non-linearity in order to rectify the oscillatory nature of the channels. As discussed in [93], the non-linearity acts as a “blob detector” which identifies primitive shapes and structures. The absolute value $f(x) = |x|$, and the squaring function $f(x) = x^2$, have been popular non-linearity choices too [153, 190]. In order to obtain some information about the interactions among these blobs, a measure of local energy is used. This operation consists of doing spatial smoothing with 2-D filters $g_k[i, j]$. The local energy map for the k -th channel is given by the convolution

$$m_k[i, j] = g_k[i, j] * f\{s_k[i, j]\} \quad (6.1)$$

Intuitively, averaging over a region with statistically similar primitives would produce a slow varying response that would be different from other textures within the image. The size and response of the $g_k(i, j)$ functions should be carefully selected. There are different opposing constraints that should be kept in mind. First, we want the filter dimensions to be as large as possible to have a good statistical representation of the primitives. Second, very large filters will preclude us having good localization around the region boundaries, where we will effectively be averaging blobs from different textures. Hence, filter support should be as small as possible.

Gaussian windows have shown to be a good compromise among these two requirements. They are known to provide optimal time-frequency localization. The basic 1-D Gaussian response is given by

$$g[n] = \frac{1}{\sqrt{2\pi}\sigma_s} e^{-\frac{1}{2}\frac{n^2}{\sigma_s^2}}, \quad (6.2)$$

where the σ_s parameter is specified as a function of the subband frequency content. As defined in [93], this parameter is dependent on the average channel frequency u_0 (*i.e.* the centroid), and it is given by

$$\sigma_s = \frac{1}{2\sqrt{2}u_0}. \quad (6.3)$$

The size of the window is set to $2\sigma_s$. It is a straightforward step to derive the 2-D filters $g_k[i, j]$.

The third state is used as a limiter to control the dynamic range of the energy maps. Typically the normalization non-linearity is also used, and is given by

$$f_2(x) = \log(x). \quad (6.4)$$

6.5 Classification Stage

As the last step of this stage, the energy maps obtained from each channel are combined to form feature vectors. In segmentation, each pixel on the original image must be assigned to some class. For a filter bank with N channels, each image pixel will count with an N -dimensional feature vector $\mathbf{f}_{i,j}$ given by

$$\mathbf{f}_{i,j} = [m_1(i,j) \ m_2(i,j) \ \dots \ m_N(i,j)]^T, \quad (6.5)$$

where $m_k(i,j)$ corresponds to the 2-D energy map of the k^{th} channel. The $\{i,j\}$ indices refer to a spatial location in both the original image and the energy map.

The feature vectors $\mathbf{f}_{i,j}$ are passed through a classifier to be assigned to one of the C texture classes in the image. In this dissertation, we assume that C is known beforehand. The final output of the classifier will be a classification map $S(i,j)$ which assigns each pixel in the original image to one of the textures classes or labels.

We plan to use then the Linear Programming Support Vector Regression (LP-SVR) approach introduced in this dissertation. The LP-SVR will be implemented in three different architectures: single LP-SVR, a cascaded LP-SVRs, and an ensemble of LP-SVRs. Assuming that the ground truth segmentation maps are available, we use classification error as a measure of accuracy.

6.6 Framework for Texture Segmentation with the DFB

In [162] Rosiles, *et al.* introduced a segmentation system based on the DFB. The results showed that the DFB is well suited for segmentation. In this case a the non-linearity from Equation (6.4), and the fuzzy C-means clustering algorithm for classification were used. Although this system performs well, in order to compare

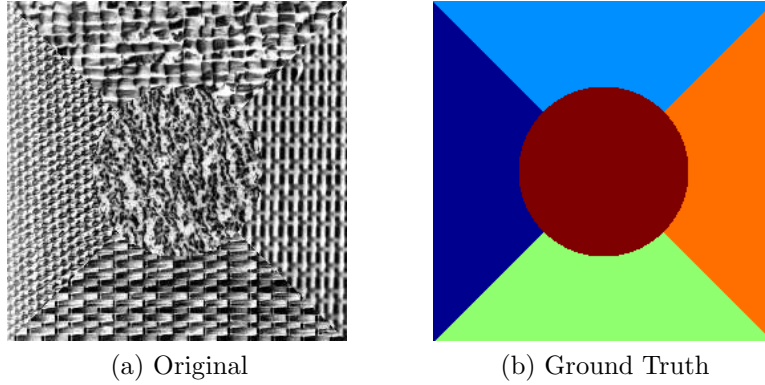


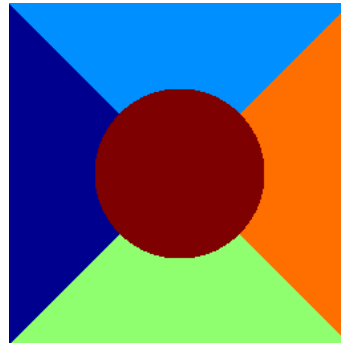
Figure 6.13: Texture mixture number 1: “Nat-5c.” It contains five different texture classes. Mixture taken from [153].

against other systems we would have to build a similar system for other multichannel representations. However, it turns out this type of comparison has already been reported by Randen and Husøy in [153]. They developed a framework which allows testing of many multichannel representations while keeping the other elements of the system in Figure 6.1 the same. Using this dataset, De Rivaz has reported results for the DT-CWT in [49] and more recently Marial, *et al.* have reported sparse dictionary-based schemes [125] using this dataset. For this reason, it makes sense that we adopt this framework for comparison purposes.

The image data set for this framework consists of the 12 texture mixtures shown in Figure 6.13a and 6.13b through Figure 6.24a and 6.24b. The mixtures exhibit different degrees of difficulty in terms of number of textures and boundaries. In some cases, textures are difficult to discern by a human observer. Training data corresponding to each texture mixture and “ground truth” maps are also made available. Texture data had their histograms equalized, so that textures could not be discriminated based on their first-order local statistics. Finally, the feature vectors were scaled by the normalization factors which give unit variance to the features of the mixture in Figure 6.13a.

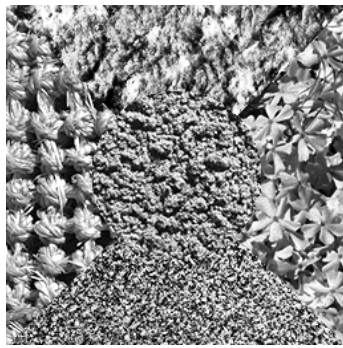


(a) Original

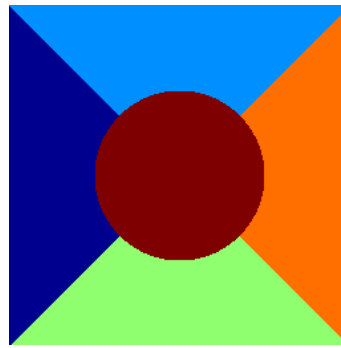


(b) Ground Truth

Figure 6.14: Texture mixture number 2: “Nat-5v,” with five classes [153].

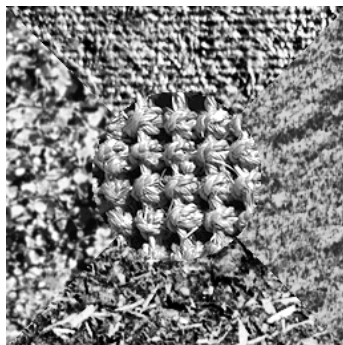


(a) Original

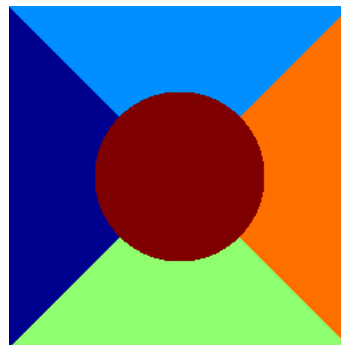


(b) Ground Truth

Figure 6.15: Texture mixture number 3: “Nat-5v2,” with five classes [153].

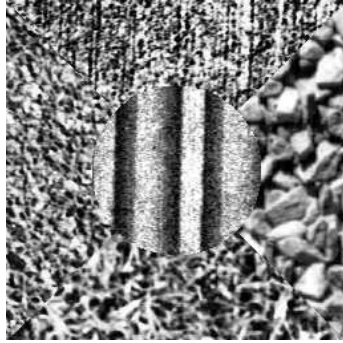


(a) Original

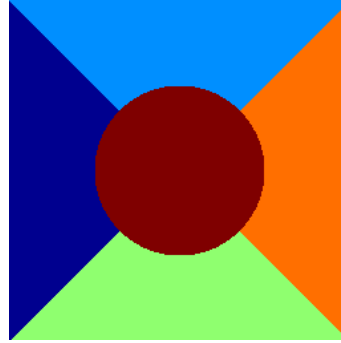


(b) Ground Truth

Figure 6.16: Texture mixture number 4: “Nat-5v3,” with five classes [153].

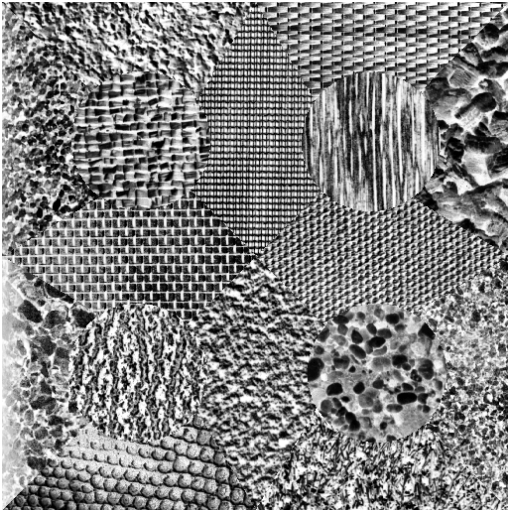


(a) Original

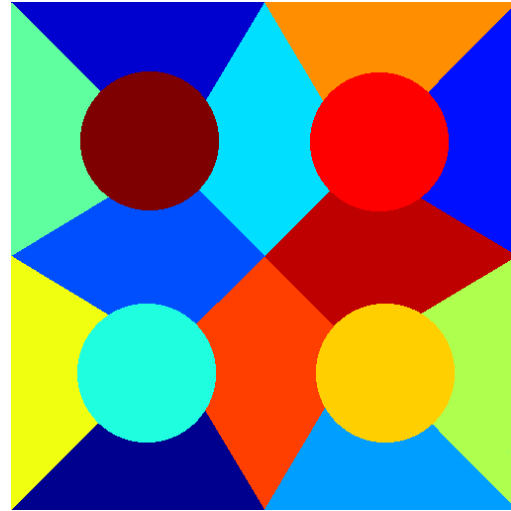


(b) Ground Truth

Figure 6.17: Texture mixture number 5: “Nat-5m,” with five classes [153].

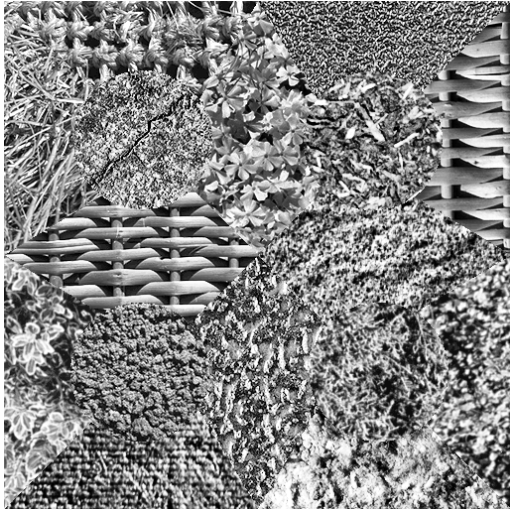


(a) Original

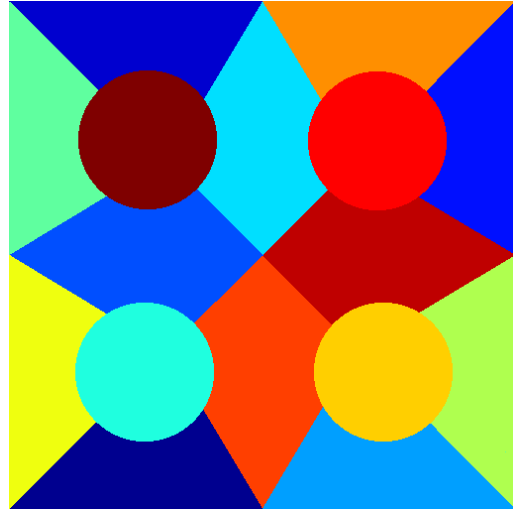


(b) Ground Truth

Figure 6.18: Texture mixture number 6: “Nat-16c.” It contains 16 different texture classes. Mixture taken from [153].

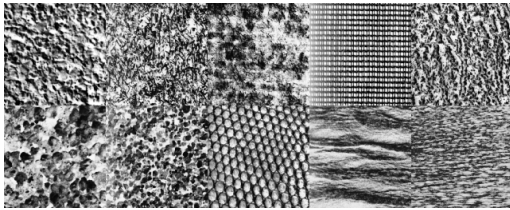


(a) Original

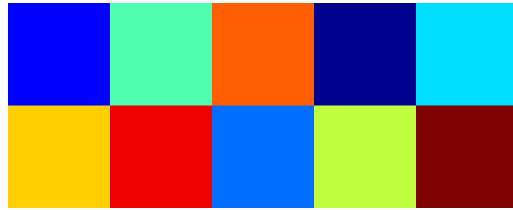


(b) Ground Truth

Figure 6.19: Texture mixture number 7: “Nat-16v.” It contains 16 different texture classes. Mixture taken from [153].

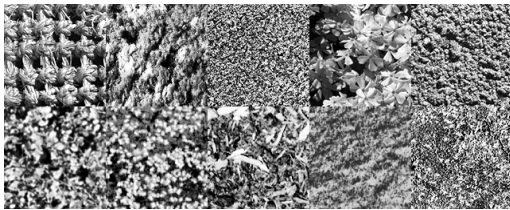


(a) Original

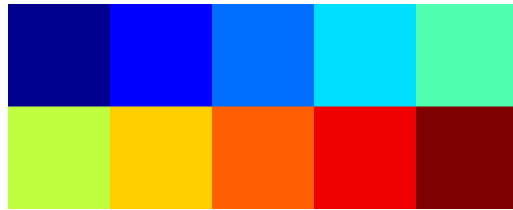


(b) Ground Truth

Figure 6.20: Texture mixture number 8: “Nat-10.” It contains 16 different texture classes. Mixtures taken from Randen and Husøy [153].

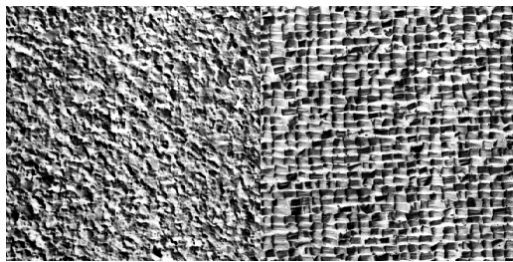


(a) Original

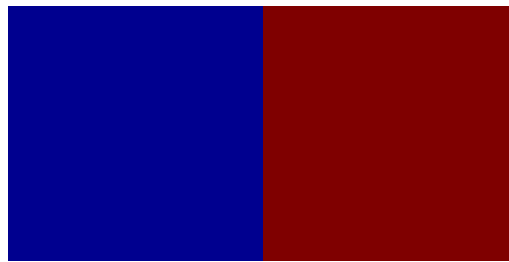


(b) Ground Truth

Figure 6.21: Texture mixture number 9: “Nat-10v.” It contains 10 different texture classes. Mixtures taken from Randen and Husøy [153].

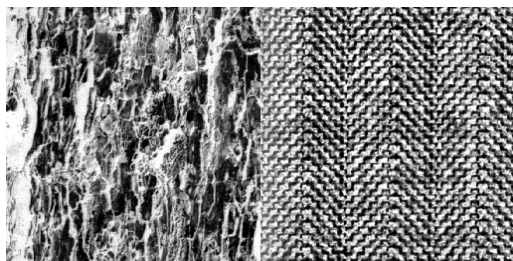


(a) Original

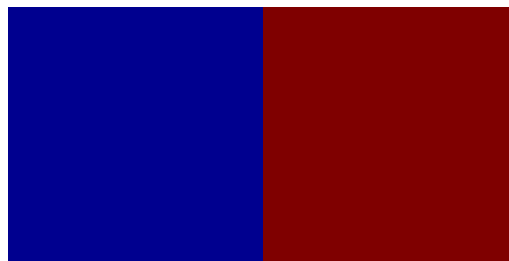


(b) Ground Truth

Figure 6.22: Texture mixture number 10: “d4d84.” It contains two different texture classes. Mixture taken from [153].

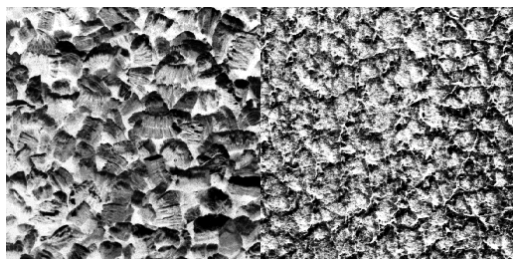


(a) Original

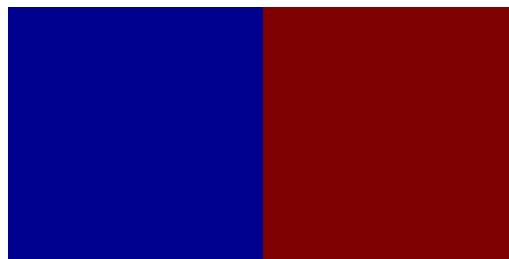


(b) Ground Truth

Figure 6.23: Texture mixture number 11: “d12d17.” It contains two different texture classes. Mixture taken from [153].



(a) Original



(b) Ground Truth

Figure 6.24: Texture mixture number 12: “d5d92.” It contains two different texture classes. Mixture taken from [153].

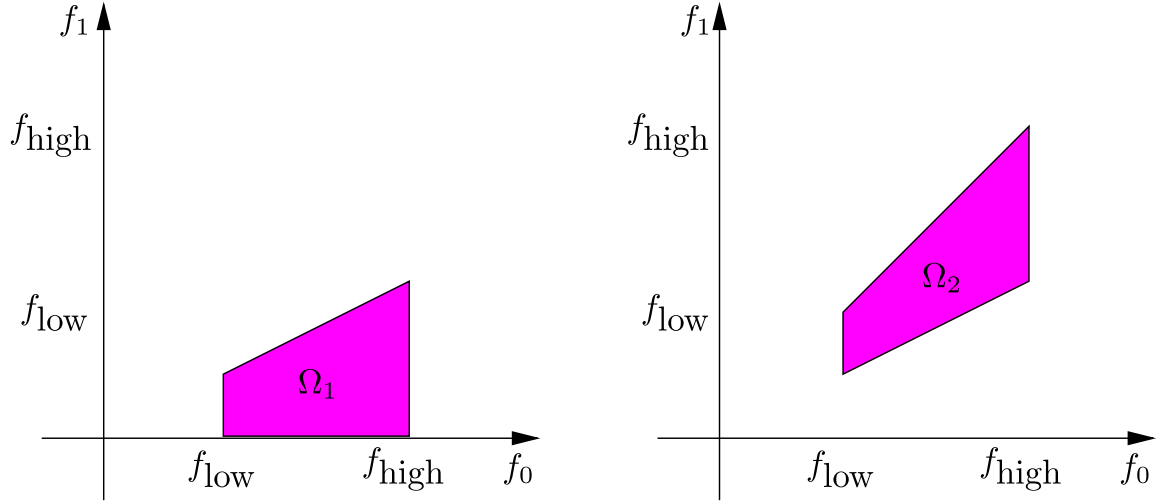


Figure 6.25: Basic frequency domain supports for DFB truncated subbands. Ω_1 and Ω_2 correspond to the eight-band DFB case.

Besides the multichannel representation on the feature extraction stage, the rectifying non-linearity is given by $f_1(x) = |x|^2$, and the normalization non-linearity is also used, and is given by (6.4). This combination of non-linearities was found to give the best segmentation results in a work by Unser and Eden [190].

For the Gaussian kernel $g[i, j]$ it is necessary to estimate the filter parameter σ_s in Equation (6.2). For Gabor functions σ_s is proportional to u_0 , the center frequency. In the DFB case, u_0 can be viewed as the “average” frequency of a subband. The directional subbands have a truncated wedge-shaped support as shown in Figure 6.25 for the eight-band case. Hence, the centroids of the regions represent the average coordinates (\bar{f}_0, \bar{f}_1) over a 2-D region. A detailed derivation for \bar{f}_0 and \bar{f}_1 is discussed in [164, 165] for these regions. Additionally, they found out that the value for σ_s was obtained on a separable fashion:

$$\sigma_s = \sqrt{\sigma_{s,0}^2 + \sigma_{s,1}^2}, \quad (6.6)$$

where

$$\sigma_{s,0} = \frac{1}{2\sqrt{2} f_0}, \quad (6.7)$$

and

$$\sigma_{s,1} = \frac{1}{2\sqrt{2} f_1}. \quad (6.8)$$

6.6.1 Proposed LP-SVR Architectures

The purpose of presenting different architectures for LP-SVR is two-folded: (i) improve the performance of a single unit SVR and (ii) introduce a novel multiresolution architecture specialized in texture segmentation. This subsection discusses three different architectures for the LP-SVR model proposed in this dissertation.

These architectures are part of the contribution of this dissertation. Their significance is related to the problem type and related to the flexibility of the model. In the next section we address the results of using these architectures.

Single Unit LP-SVR

The first architecture uses all the directional subbands as inputs to produce an output, as shown in Figure 6.26. This is the typical LP-SVR model used for classification. The understanding of the concept is fairly simple, since it is a single unit LP-SVR. This is exactly the model discussed in Chapter 3. Using this architecture, the training set, comprising the three levels of the DP, is received in full at the LP-SVR inputs and the large-scale training method is performed. Let y denote the output of the single unit LP-SVR. Then, from Figure 6.1, we have that $y \equiv S$.

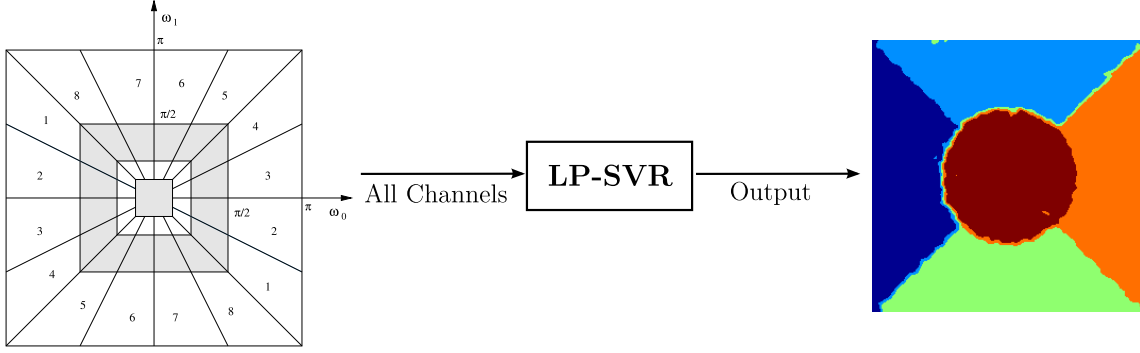


Figure 6.26: LP-SVR architecture that uses all the directional subbands as inputs to produce an output.

Cascaded or Multiresolution LP-SVR

The second architecture, illustrated in Figure 6.27, uses a cascade approach of the LP-SVR model. In the first stage, it receives as input the first level of the DP. In the second stage, another LP-SVR model receives as input the second level of the DP along with the first stage LP-SVR's output. Finally, the third stage LP-SVR classifier receives the third DP level along with the second stage LP-SVR output to produce a final segmentation result. This is effectively a multiresolution architecture, where higher resolution results are used to guide the segmentation at lower resolutions in the next stage.

Formally, define $y^{(j)}$ as the output of an LP-SVR at the j -th stage. Then, in terms of training sets, we can define the inputs at each stage as follows:

Stage 1: $\mathcal{T}_1 = \{\mathbf{x}_i, d_i\}_{i=1}^N$, for all \mathbf{x} in the first DP

Stage 2: $\mathcal{T}_2 = \{[\mathbf{x}_i \ y_i^{(1)}], d_i\}_{i=1}^N$, for all \mathbf{x} in the second DP

Stage 3: $\mathcal{T}_3 = \{[\mathbf{x}_i \ y_i^{(2)}], d_i\}_{i=1}^N$, for all \mathbf{x} in the third DP,

where $\mathbf{x} \in \mathbb{R}^8$. Finally the output is given by $y^{(3)}$, which, from Figure 6.1, we have that $y^{(3)} \equiv S$.

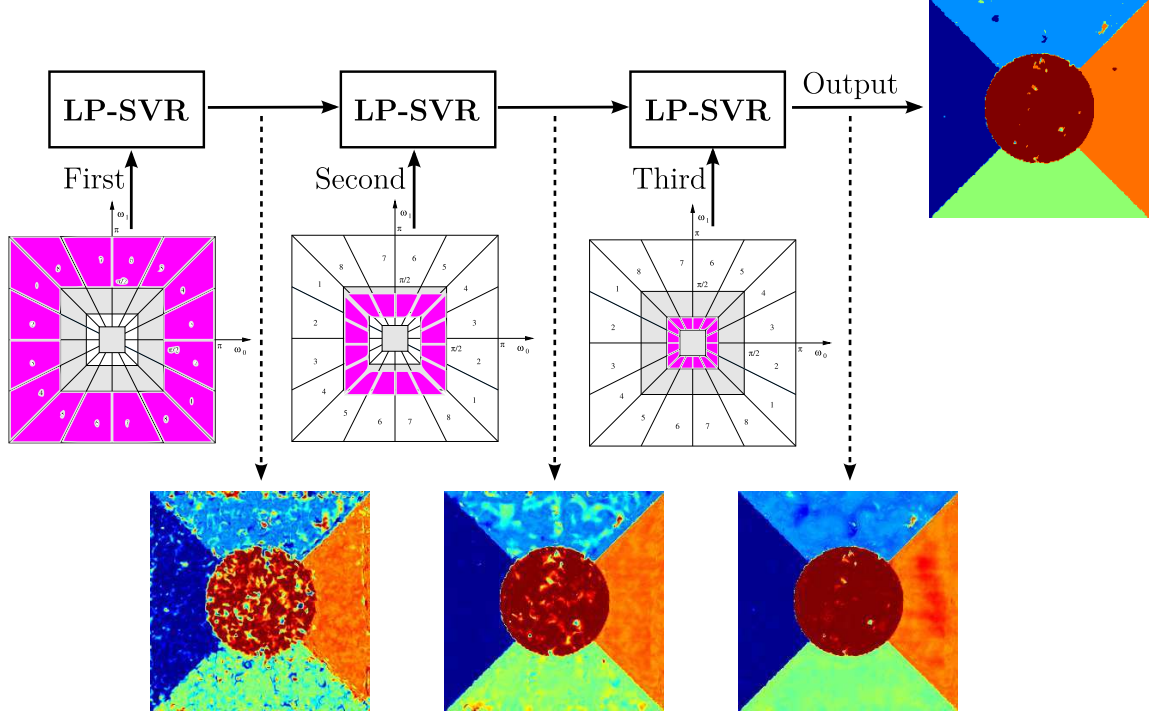


Figure 6.27: LP-SVR architecture that uses a cascade approach receiving as input the current stage UDFBs and feeds the output to the next LP-SVR along with the next stage UDFBs.

Ensemble of LP-SVR

The third architecture, depicted in Figure 6.28, is an LP-SVRs ensemble that receives the three DP levels of resolution at individual LP-SVR classifiers. Each LP-SVR produce an output $y^{(1)}$, $y^{(2)}$, and $y^{(3)}$. These outputs form a new training set $\mathcal{T} = \{[y_i^{(1)} \ y_i^{(2)} \ y_i^{(3)}], d_i\}_{i=1}^N$. Then, a final LP-SVR classifier is trained with \mathcal{T} , producing a segmentation result as illustrated in Figure 6.28. Let $y^{(f)}$ denote the output of the final LP-SVR, then we have that $y^{(f)} \equiv S$.

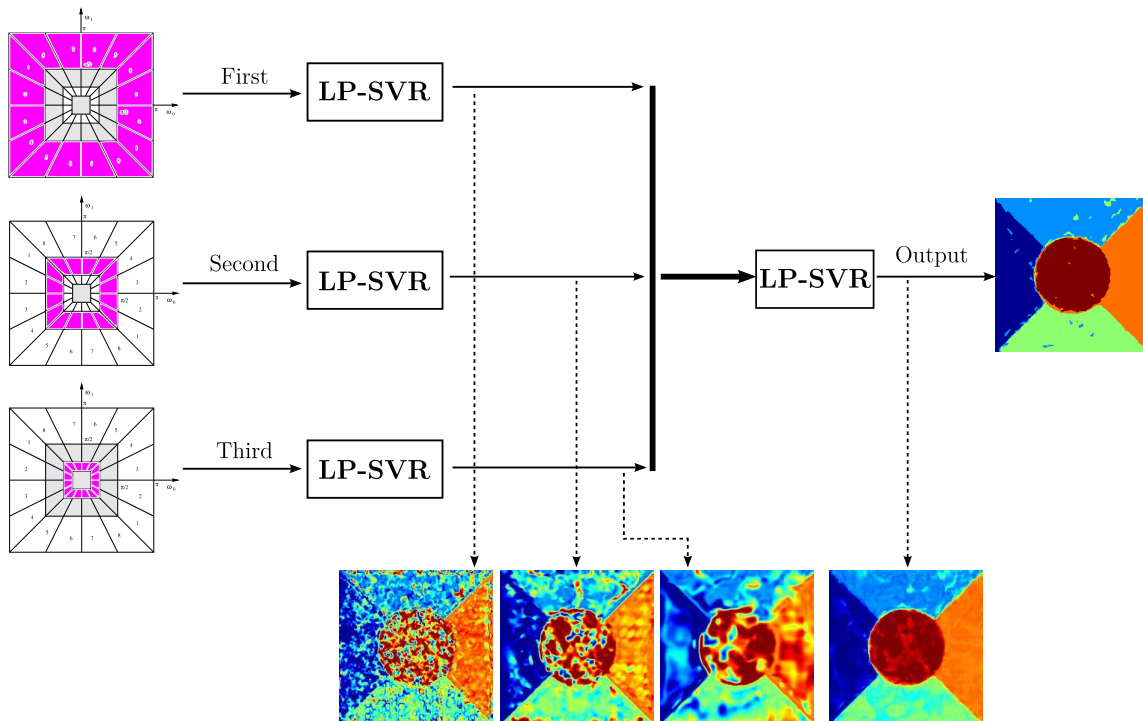


Figure 6.28: LP-SVR architecture that uses an ensemble approach which receives the three UDFB stages at individual LP-SVR classifiers producing outputs connected to a final classifier that gives the segmentation result.

6.7 Evaluation of the Undecimated DP and LP-SVR for Texture Segmentation

We evaluate the segmentation algorithm using the ULap-UDFB pyramid where the pyramidal and directional components are undecimated. Hence full-rate subbands are available for all channels. For our experiments, we determined that four pyramid levels ($J = 4$) provided the best performance-complexity tradeoff. We should note that the lowpass band was not used for feature extraction.

Texture segmentation is a clear example of a large-scale classification problem for SVRs. The different texture mixtures are represented by feature vectors $\mathbf{f} \in \mathbb{R}^{24}$, this size comes from the three pyramid levels and $N = 8$. To address the size of

the problem, let us recall that the LP-SVR has LP problems with $2|\mathcal{T}|$ constraints and $4|\mathcal{T}| + 2$ variables, where $|\mathcal{T}|$ denotes the size of the training set. Also let us recall that in our case we want to perform classification of a minimum of two classes and a maximum of 16 classes, each corresponding to different textures. This results in training sets varying in number of samples from 131,072 to 1,045,576, which are derived from the training images size as $2(256 \times 256)$ and $16(256 \times 256)$. These number of samples result in SVR linear programming (LP) problems with 137,439,477,760 to 8,796,097,216,512 points, *i.e.*, LP problems that have 524,290 variables and 262,144 constraints, up to LP problems with 4,194,306 variables and 2,097,152 constraints. Therefore, this problems is a excellent representative of a large-scale classification problem in image processing and analysis.

From the extensive evaluation in [165], we have chosen a set of UDFB parameters that provide good performance. Eight directional subbands ($N = 8$) were used for all directional pyramid resolutions. The wedge-shaped filters for the UDFB were implemented with a three-step ladder structure (see Figure 6.10) with 1-D prototypes $\beta(z)$ designed as length-12 Parks-McClellan filters.

In Table 6.1 we present the classification error results for a full-rate DP system with $J = 4$ and $L = 12$. Table 6.1 shows experimental results of work reported over the texture mixtures presented in the preceding figures; the first column identifies a mixture according to the number provided in Figures 6.13 to 6.24. The values in the table represent the segmentation error (in percentage) given by the proportion on misclassified pixels on the ground truth label maps also presented in the figures. The work we include is organized as follows: column two shows the work by Randen, *et al.* [153], column three shows the work by Topi, *et al.* [186], column four shows the work by Skretting, *et al.*, [176], column five shows the work by Di Lillo, *et al.* [52], column six shows the work by Mairal, *et al.* [125]. Then, on the seventh column appears the work by Rosiles [165] using the ULaup-UDFB pyramid with learning vector quantization (LVQ) classifier. On the remaining columns, we show the results

Table 6.1: Comparison of Segmentation Errors. ULap-UDFB system implemented with $J = 4$ and $L = 12$.

#	Previous Work Reported In					[165]	[165] with LP-SVR		
	[153]	[186]	[176]	[52]	[125]	$N = 8$ LVQ 800	$N = 6$ Sing.	Casc.	Ense.
1	7.2	6.7	5.5	3.37	1.61	4.67	3.41	1.26	3.97
2	18.9	14.3	7.3	16.05	16.42	19.48	3.48	2.71	4.11
3	20.6	10.2	13.2	13.03	4.15	12.37	3.36	7.50	6.01
4	16.8	9.1	5.6	6.62	3.67	17.01	4.04	5.72	4.74
5	17.2	8.0	10.5	8.15	4.58	14.18	4.02	2.10	3.40
6	34.7	15.3	17.1	18.66	9.04	31.12	7.22	41.05 \diamond	42.19 \diamond
7	41.7	20.7	17.2	21.67	8.80	48.02	10.90	40.86 \diamond	43.96 \diamond
8	32.3	18.1	18.9	21.96	2.24*	20.60	7.24	6.53	8.00
9	27.8	21.4	21.4	9.61	2.04*	37.88	7.50	6.97	12.33
10	0.7	0.4	-	0.36	0.17*	0.58	0.18	0.97	0.60
11	0.2	0.8	-	1.33	0.60*	1.57	0.12	0.03	0.01
12	2.5	5.3	-	1.14	0.78*	4.82	0.84	0.82	1.49
μ	18.4	10.9	-	10.16	4.50	17.69	4.36	9.71	10.90

of using the LP-SVR classifier combined with the DP for different architectures. The single unit LP-SVR architecture is shown in the eighth column, the multiresolution architecture is shown in the ninth column, and an LP-SVR ensemble architecture is shown in the tenth column.

The method with the lowest classification error is shown in bold font. In the average case, the single unit architecture reports the lowest classification error. While the cascaded approach can be ranked in the third place. Note that the errors reported with the symbol \diamond are those that are related to the issue of having more classes than features.

In the case of Mairal, *et al.* [125], the results for textures 8 and 9 show very low segmentation error. However, we believe these low results are an artifact resulting from their scheme *and* the way the textures are arranged over a uniform 2×5 grid. In [119], the texture mixtures are divided in 12×12 patches which are classified as a single pixel; in essence, the 12×12 block receives the same class label. This pro-

cess results on uneven texture boundaries across textures that need to be smoothed through an ad-hoc post-filtering step. The exact effect of the post-filtering step is difficult to quantify as far as how it benefits the overall segmentation result. However, it is clear that the configuration of textures 8 and 9 is favored by this approach since we can establish a well-defined relationship between the dimensions of the square texture regions (128×128) and the selected texture patch size. This generates a bias toward lower errors around texture boundaries. Moreover this scheme is highly dependent on the texture patch size. As discussed by Mairal, their algorithm performed poorly with texture mixture 2 as a result of having textures whose characteristics (*i.e.*, structure, orientation, randomness) could not be captured by the 12×12 patch size. He mentions the need for a multiresolution scheme in this case, but this idea has not been reported at the moment of this writing. Hence, although the approach in [119] is promising, it remains to be seen if it can be generalized to reduce its dependency on heuristics like patch size and post-filtering that are quite difficult to quantify in terms of visual content of images.

Hence, from this perspective, we believe that the LP-SVR scheme developed here represents a well founded classification scheme for texture segmentation with the some of the best results reported to date.

We also show the classification maps and the error maps for some of the test mixtures in Figures 6.29 through 6.32. The work reported by Rosiles [165] using an LVQ approach to classify the data is depicted in Figure 6.29. From the date [165] was published to date, there have been advances in large-scale machine learning, such as the one presented in this dissertation: the Linear Programming Support Vector Regression (LP-SVR). The remaining figures explain the different choices of LP-SVRs architecture.

Figure 6.30 shows the results of using a simple LP-SVR trained with the full set of features using the large-scale learning approach presented in Chapter 3. As can be observed from the figure, there are some classification errors in the texture

neighborhood borders while the inner regions show good classification.

The result of using a cascaded LP-SVR architecture is shown in Figure 6.31. This architecture takes advantage of the multiresolution properties of the problem and features. In this case, each LP-SVR received eight features corresponding to a single resolution level of the directional pyramid. This architecture results in a better classification of textures around boundaries as a result of separating the classification of texture features across resolutions. The first (finest resolution) takes care of the texture boundaries, while the other two refine the internal regions of the texture. This is clearly seen on Figure 6.31c. This is a very interesting result since we can see the texture boundaries are clearly and well delineated, while the inside areas have a high rate of misclassification. Moreover, the degradation of performance with respect to the single-SVR architecture is more than evident and we could conclude that the cascaded SVR method fails. However, we provide some explanations for this behavior at the end of this section.

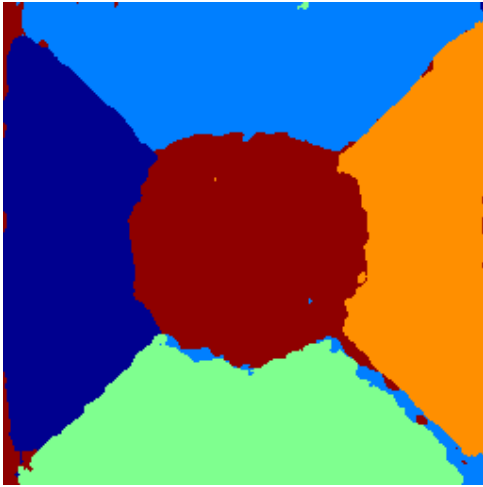
Figure 6.32 depicts the segmentation results for three mixtures using the ensemble architecture. In this case we see that the visual results are somewhere in between the single SVR and the cascade SVR architectures with respect to the classification around boundaries as inside each region. Here we see a similar failure for the sixteen texture mixtures where the region boundaries look well defined while the inner regions are highly misclassified.

Further inspection of Figure 6.31c and 6.32c provides some insight into the origin of the bad performance. For each class, the classification errors are “bounded” in the sense that for class k the errors correspond to class $k - 1$ and/or $k + 1$. This is clearly a by-product of LP-SVR which is actually predicting a value between 1 and 16 and the value is miss-predicted by ± 1 . In other words, the function $f(\mathbf{x})$ has a high degree of oscillation around the true value. This behavior within the real-valued output of function is very common and is typically attributed to the following reasons:

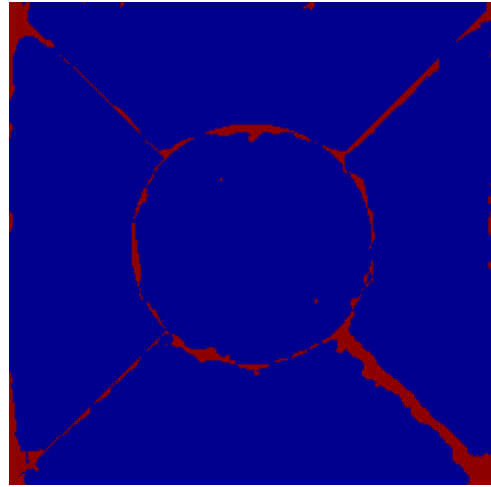
- The ϵ -tube defined on the optimization problem is too wide.
- The regressor does not have enough predictive power, or in terms of classification, enough discriminative power. In other words, the feature vector cannot be projected to a kernel space where the feature vectors are linearly separable.

In the first case, we could make it tighter. However, this could reduce the generalization capacity of the LP-SVR. The second factor makes sense intuitively. There seems to be a relationship between the feature vector dimension and the number of classes. For the single-SVR architecture, we see an excellent overall performance using a 24 dimensional feature space. For the other two cases where each SVR uses an eight-dimensional feature vector, we see the same excellent numerical performance (or even slightly better) for the two texture and five texture cases, and ten texture cases, with some detriment on the visual results for the ten texture case. However, for the sixteen texture mixtures the cascade- and ensemble-SVR architecture broke down. Hence this supports our conjecture on the relationship between feature vector dimension and number of classes. However, we can not claim that as the number of texture classes becomes very large, then the dimension of feature vector should grow equally. This should be the subject of extensive research and directs us towards the holy grail question in texture analysis: Is there a theory of textons [100, 101] that will allow us to efficiently represent textures with a well defined model?

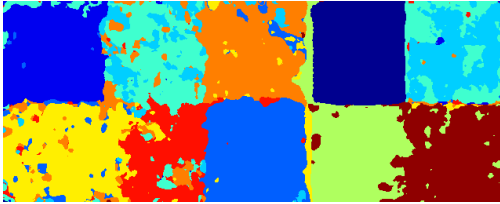
From a mathematical perspective, this indicates that the resulting regression function $f(\mathbf{x})$ as given in Equation (3.1) has a limited reconstruction ability as there is not enough “signal support” to provide good approximations. This is reflected by the oscillatory (and bounded) behavior of the segmentation maps. Hence, we leave as this as an open issue, subject to future work which should involve a detailed study on ways the discriminative capacity of SVRs over a large set of texture classes.



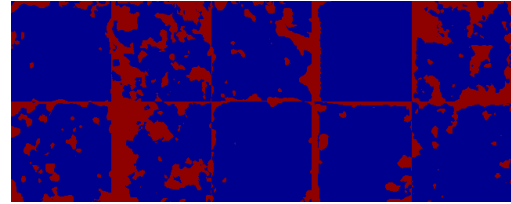
(a) Segmentation Map of "Nat-5c"



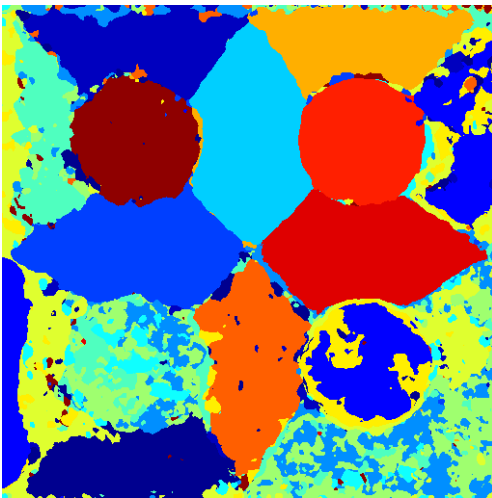
(b) Error Map of "Nat-5c"



(c) Segmentation Map of "Nat-10"



(d) Error Map of "Nat-10"

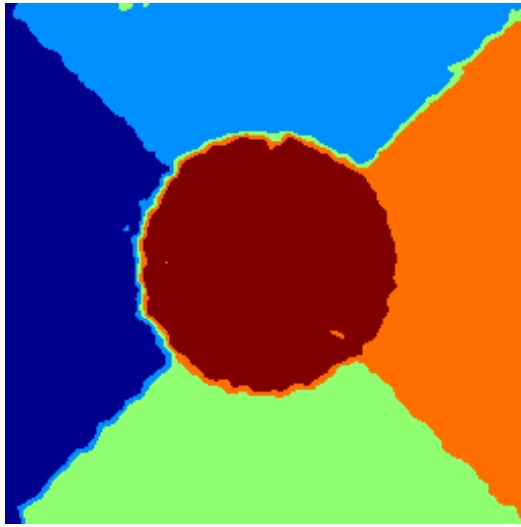


(e) Segmentation Map of "Nat-16c"

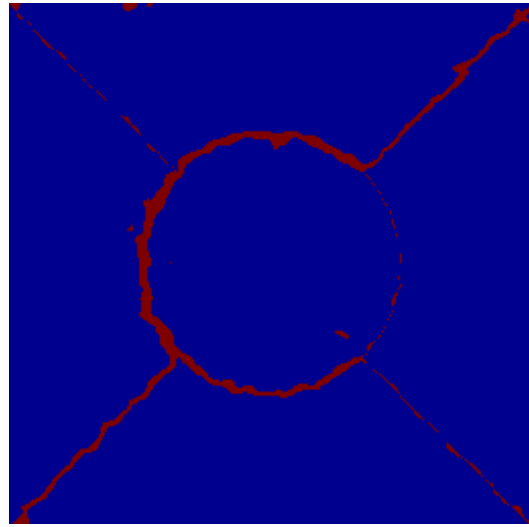


(f) Error Map of "Nat-16c"

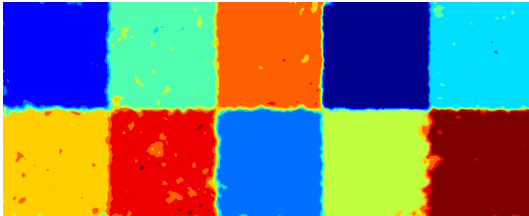
Figure 6.29: Texture segmentation maps result using ULAP-UDFB and LVQ [165].



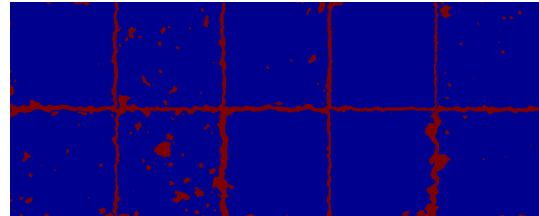
(a) Segmentation Map of "Nat-5c"



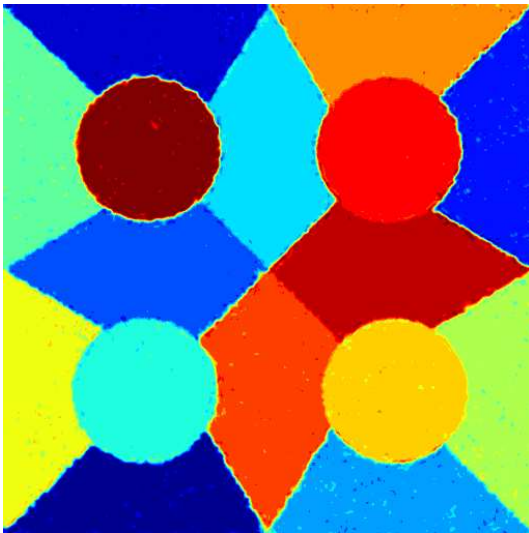
(b) Error Map of "Nat-5c"



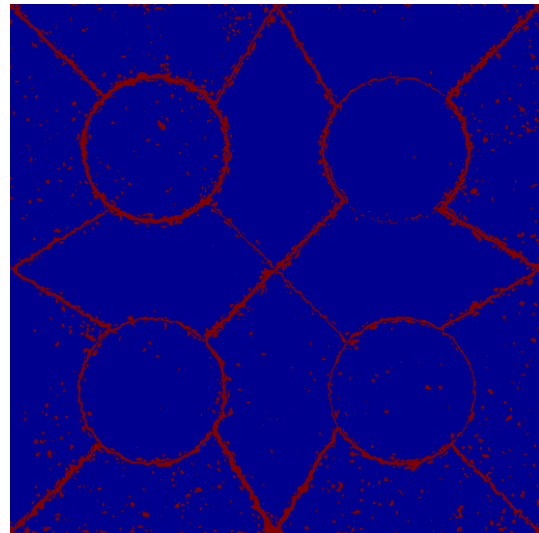
(c) Segmentation Map of "Nat-10"



(d) Error Map of "Nat-10"

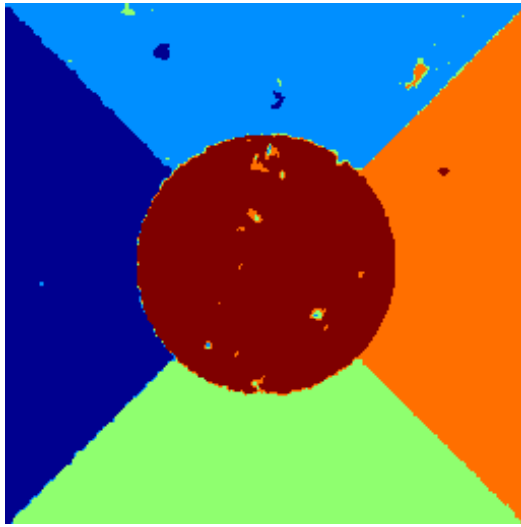


(e) Segmentation Map of "Nat-16c"

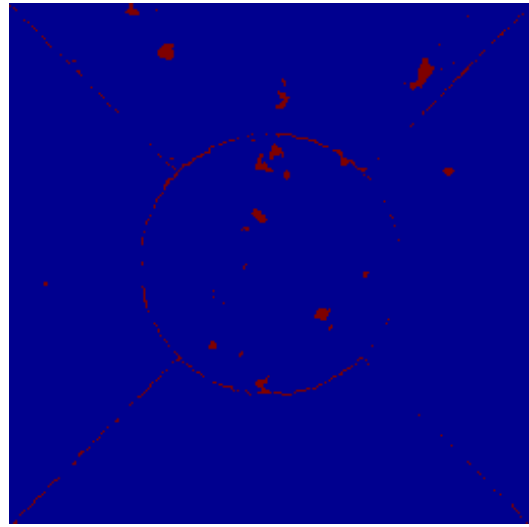


(f) Error Map of "Nat-16c"

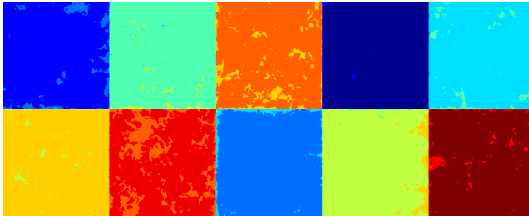
Figure 6.30: Texture segmentation maps result using ULAP-UDFB and LP-SVR with a single architecture.



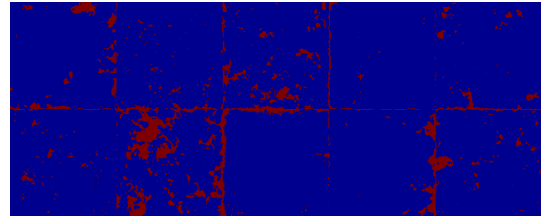
(a) Segmentation Map of "Nat-5c"



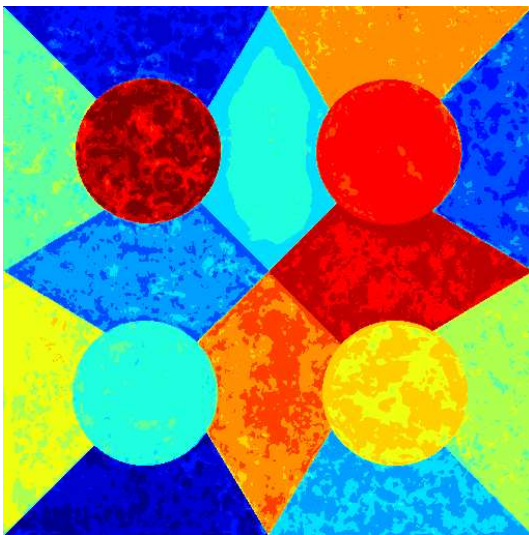
(b) Error Map of "Nat-5c"



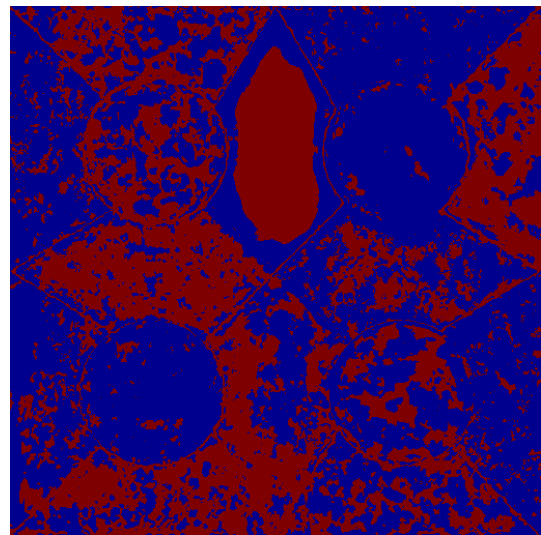
(c) Segmentation Map of "Nat-10"



(d) Error Map of "Nat-10"

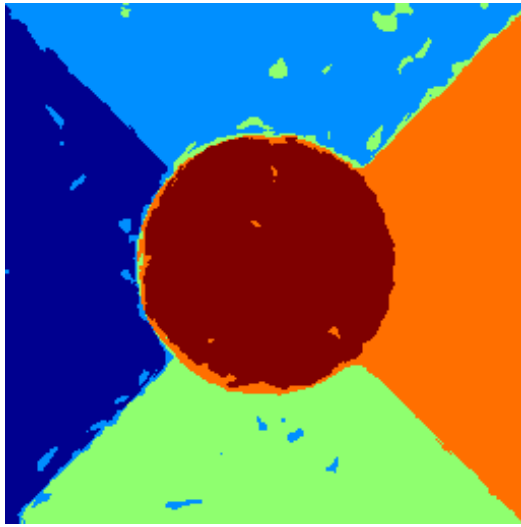


(e) Segmentation Map of "Nat-16c"

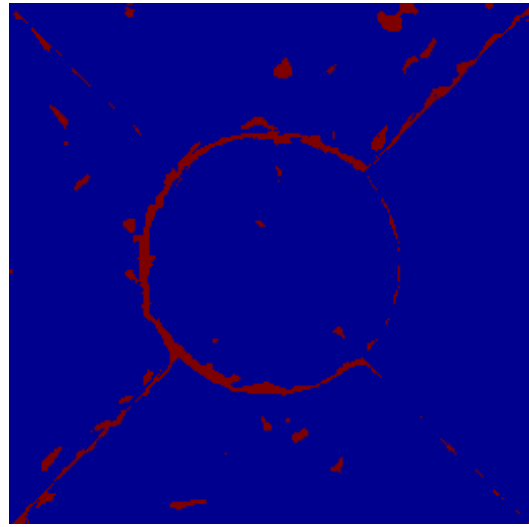


(f) Error Map of "Nat-16c"

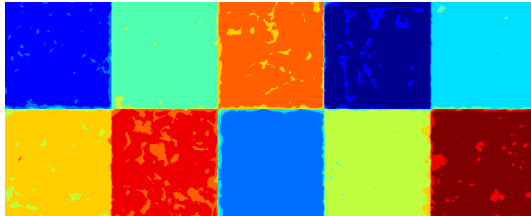
Figure 6.31: Texture segmentation maps result using ULAP-UDFB and LP-SVR with a cascaded architecture.



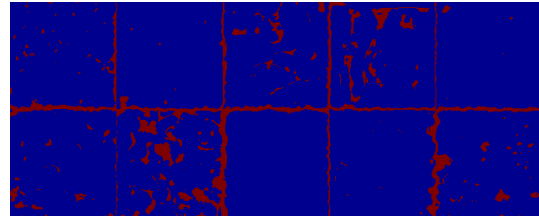
(a) Segmentation Map of "Nat-5c"



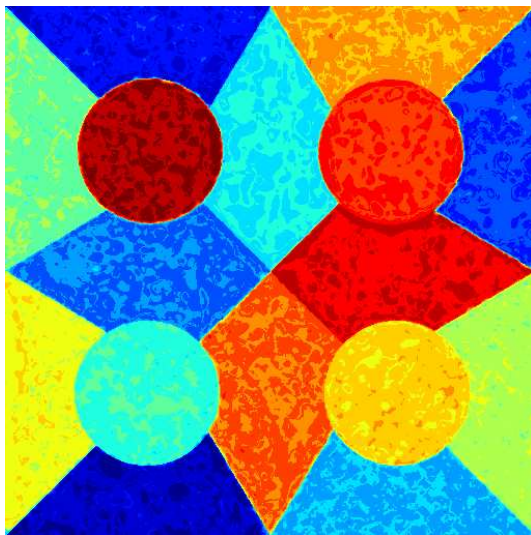
(b) Error Map of "Nat-5c"



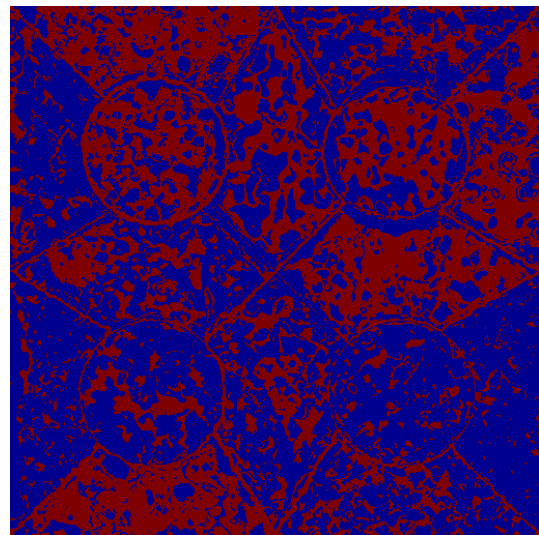
(c) Segmentation Map of "Nat-10"



(d) Error Map of "Nat-10"



(e) Segmentation Map of "Nat-16c"



(f) Error Map of "Nat-16c"

Figure 6.32: Texture segmentation maps result using ULAP-UDFB and LP-SVR with an ensemble architecture.

6.8 Conclusions on Texture Segmentation

In this chapter we showed that the large-scale (LS) linear programming support vector regression (LP-SVR) approach, is a powerful classification scheme for supervised texture segmentation. Experimental results suggests that combining the proposed LS LP-SVR with wavelet-based features provided the best results reported (as far as we know) to date on a well known dataset. We have explored different classifier configurations or architectures that take advantage of the multiresolution structure of the features. Such architectures demonstrate the flexibility of the proposed LS LP-SVR training scheme. These architectures include an ensemble-based configuration and a multiresolution or cascaded configuration. One limitation that we encountered was for those cases where the number of textures (*i.e.*, classes) on a mixture exceeded the number of features used in the SV regressor. This remains an open issue that will be explored in subsequent work. To close this chapter we emphasize that the results presented should be considered state-of-the-art in the texture segmentation area and we expect a lot of attention for our approach by the community.

Chapter 7

LP-SVR-Based Dust Storm Detection

Recent studies have found correlation between lung disease and dust storms. This prompted researchers to conduct dust air-borne suspended particle (aerosol) analysis. However, there is paucity of formal pattern recognition models in dust storm detection. This research analyzes the proposed Linear Programming Support Vector Regression (LP-SVR) approach as compares it with popular statistical and neural approaches. This study includes the Maximum *a Posteriori* Classifier (MAP), the hybrid Probabilistic Neural Network (PNN), and the Feed-Forward Neural Network (FFNN). The features utilized are multispectral thermal emissive bands from the Moderate Resolution Imaging Spectroradiometer (MODIS). We utilized four near infrared bands: B20 ($3.66 - 3.84\mu m$), B29 ($8.40 - 8.70\mu m$), B31 ($10.78 - 11.28\mu m$), and B32 ($11.77 - 12.27\mu m$). Numerical performance evaluation shows that the LP-SVR and the hybrid approach (PNN) perform better than the classic MAP and FFNN. Visually, the four methods accurately detect dust storms. The models demonstrated a strong ability to find non-trivial relationships within the spectral bands. The proposed methods demonstrated to be soil-independent and surface-invariant detection methods. This research highlights the effectiveness of LP-SVR, MAP, PNN, and FFNN in understanding dust storm phenomena.

7.1 Introduction

Dust storms are an inadvertent cause of several physical, environmental, and economic hazards. Recent studies show a direct correlation between exposure to high-levels of air-borne particle concentrations (aerosols) and the increase in mortality rate from cardiovascular, respiratory illness (*e.g.*, increase of asthma cases), and lung cancer [160]. Therefore, this situation represents a major concern for several health and safety agencies [66, 121]. From a scientific perspective, understanding dust storm genesis, formation, propagation and composition is important to reduce their impact or predict their effect.

Advances in remote sensing like multispectral instruments allow imaging of atmospheric and earth materials based on their spectral signature over the optical range. In particular, dust aerosols propagated through the atmosphere in the form of dust storms can be detected through current remote sensing instruments. Some of the most relevant systems are based in the Moderate Resolution Spectroradiometer (MODIS) Aerosol Optical Density (AOD) product [114], which is provided by the NASA Terra satellite. However, AOD products require a considerable amount of processing that introduces a significant delay (*i.e.*, two days after a satellite pass) before it can provide useful information on aerosol events. Other approaches are based on the so-called “band-math” [1, 32, 160] where simple operations between bands are used to provide a visual (and subjective) display of the presence of dust storms.

Other studies focus on dust storm transport and observe its trajectory [57, 94, 113, 124, 150, 159]. However, these findings can be further improved to be more accurate in analyzing and detecting dust [2, 82, 92, 102, 105, 131, 167, 188].

Furthermore, given the large amounts of data produced by the MODIS instrument, it is also desirable to have automated systems that assist scientists in finding or classifying different earth phenomena. For example, Aksoy, *et al.* [4] developed a visual grammar scheme that integrates low-level features to provide a high level

spatial scene description on land cover and land usage. As far as the authors know, similar automated schemes for dust detection based on statistical pattern recognition techniques have not been reported.

This chapter presents a feature (*i.e.* band) selection procedure based on spectral signatures known to express dust aerosols. These features are analyzed using four methods for the detection of dust storms from multispectral imagery using statistical, neural, and optimal classifiers. Based on reported data, a feature set that allows high performance, accuracy, and real-time detection of the dust aerosol is presented. The proposed feature set is extracted from MODIS spectral bands and tested with a maximum *a posteriori* (MAP) classifier, a probabilistic neural network (PNN), a feedforward neural network (FFNN) and a support vector regression (SVR) classifier. It will be shown that the PNN and LP-SVR approaches provide better detection and representation of dust storm events.

This chapter is organized as follows: Section 7.2 discusses the feature extraction methods. Sections 7.3, 7.4, 7.5, and 7.6 explain the proposed detection models for classification; Section 7.7 addresses a numerical comparison of results and discusses visual results over land and ocean; and Section 7.8 presents conclusions of this research.

7.2 Feature Extraction

The MODIS instrument is carried on board NASA's Terra and Aqua satellites. MODIS provides Earth's information in 36 spectral bands. It facilitates atmosphere, ocean, and land analysis. The data available in MODIS Level 1B make possible dust storm analysis. Dust storm visual assessment can be achieved using MODIS bands $B1$, $B3$, and $B4$ since they match the human visual spectrum. We produced RGB true color images by mapping $R = B1$, $G = B4$, and $B = B3$. The true color images are enhanced using a non-linear function explained in Appendix A. As an example,

Figure 7.1 shows a true color image of the south-western US area.

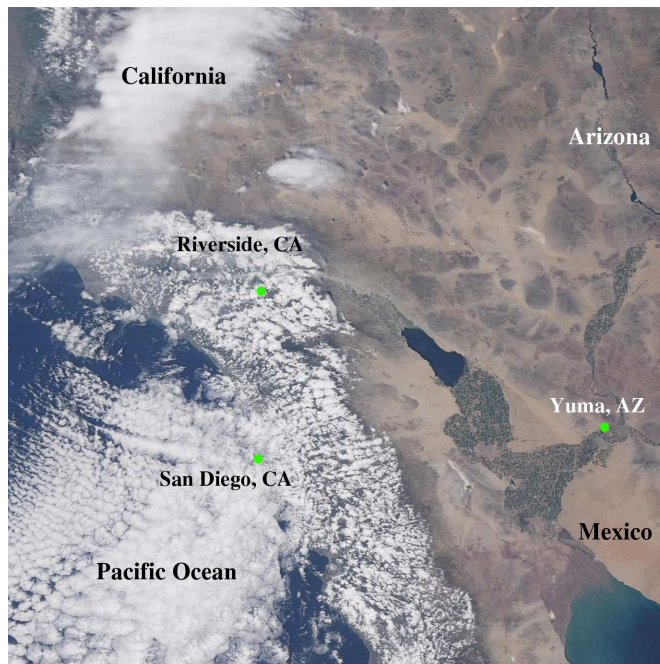


Figure 7.1: True color RGB composite of the southwestern US. Date 04/15/2001. Time 18:05 UTC. Satellite: Terra. Instrument: MODIS.

In 1997, Ackerman, *et al.* [184] demonstrated that using bands $B32$ and $B31$ improves dust storm visualization. Later, Hao, *et al.* [76] established that a linear combination of bands $B20$, $B29$, $B31$, and $B32$ can be utilized for dust storm visualization. Based on these findings, we designed a classification scheme using the following MODIS thermal emissive bands: $B20$ ($3.66 - 3.84\mu\text{m}$), $B29$ ($8.40 - 8.70\mu\text{m}$), $B31$ ($10.78 - 11.28\mu\text{m}$), and $B32$ ($11.77 - 12.27\mu\text{m}$).

MODIS level 1B contains 16-bit scaled thermal emissive bands that need to be recovered to their original scale in $\text{W}/\text{m}^2/\mu\text{m}/\text{sr}$, *i.e.*, Watts / squared meters / micrometers / Schrödingers. The recovery process is given by

$$I = \kappa(\omega - \eta), \quad (7.1)$$

where $I \in \mathbb{R}$ denotes the recovered radiance, κ is the radiance scale factor, η is the radiance offset, and ω is the scaled (*i.e.* raw) data. Our feature vector $\mathbf{x} \in \mathbb{R}^4$ consists of the following recovered radiances:

$$\mathbf{x}_i = [I_i^{B20}, I_i^{B29}, I_i^{B31}, I_i^{B32}] , \quad (7.2)$$

where \mathbf{x} is a set of features associated with multispectral data, and the superscript of I refers to an element of the specified spectral band. The feature vector \mathbf{x} will be associated with a class d . The variable d can take on either one of the following two values $d \in \{0, 1\}$, where 0 is background, and 1 is dust. Formally, a dataset is defined by the pair of a feature vector and a desired class value: $\{\mathbf{x}, d\}$.

For the reported experiments, 31 different events were used, corresponding to the south-western US and north-western Mexico area as reported in [137]. Each of the 31 events produced 2.7 million feature vectors associated to either dust or non-dust data. Typically, the number of feature vectors associated with dust is lower than with non-dust.

In pattern recognition methods, the data is generally separated into three sets: *training*, used to build the model; *testing*, used to adjust the model if needed and also to determine the numerical performance; and *validation*, used to determine the generalization capabilities of the model.

Some powerful pattern recognition approaches are constrained in the number of samples that can be used for training due to the computational complexity of the model. This chapter reports the Probabilistic Neural Network (PNN) approach as an example of a highly constrained method. The PNN requires as many neurons as the number of samples provided times the number of classes. Therefore, this chapter aims to use as few samples as possible for constructing the models, which also contributes to a less expensive hardware implementation.

In this research, the training set selection is based on Mather's criteria [102] and

the Karhunen Loeve Transformation (KLT). Mather's criteria establishes that the training samples number must be at least 30 times the number of bands ($|\mathbf{x}|$) times the number of classes, *i.e.*, in this study we would have to use at least $30 \times 4 \times 2 = 240$ training samples.

Given the data availability, twice as many training samples as Mather's criteria establishes are used: 480 training samples, 240 for dust class, and 240 for non-dust class. The selection of the training set was performed with KLT, also known as Principal Component Analysis (PCA). Instead of using KLT to reduce the number of bands used, KLT is used to identify the 240 most discriminant training samples per class; then these 480 samples become the training set for modeling the dust storm detection problem.

The choice of reducing the training set was driven by the fact that we want neural models with low complexity. Furthermore, the number of 240 samples per class matches a confidence level of 99.9% according to a well established sample number selection criteria described in [40]. All the remaining feature vectors were used as a testing set. The validation set consisted of eight worldwide events.

All the data samples were downloaded using NASA's WIST on-line tool [72]. The complete dataset provides approximately 85 million feature vectors. The complete list of data granules appears in [157].

7.3 Maximum *a Posteriori* Detection

Let \mathbf{x} from (7.2) be a random variable with probability density function (PDF) $p(\mathbf{x})$, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^4$. Let $d \in \mathcal{C} \subset \mathbb{R}$ be a random variable associated with the class values. Then let $P(d_i)$ be the prior probability mass function (PMF) of the i -th class. Let the conditional PDF $p(\mathbf{x}|d_i)$ be denoted as the likelihood of d_i with respect to the feature vector \mathbf{x} . Let $p(d_i|\mathbf{x})$ be the *a posteriori* (posterior) PDF. The Maximum *a Posteriori* Classifier (MAP) can be derived from Bayes theorem. It is an accepted

method in remotely sensed data classification and analysis [4]. Therefore, here we use the MAP for dust storm detection.

To obtain the maximum *a posteriori* probability of class d_i given that the feature vector \mathbf{x} has been observed, we start by assuming that $p(\mathbf{x})$ is uniform, and then we define the following discriminant function [77]:

$$\psi_i(\mathbf{x}) = p(\mathbf{x}|d_i) P(d_i), \quad (7.3)$$

that allows us to define the following decision rule: Assign feature vector \mathbf{x} to class d_i with probability $p(d_i|\mathbf{x}) \equiv \psi_i(\mathbf{x})$ if

$$\psi_i(\mathbf{x}) > \psi_j(\mathbf{x}) \quad \text{for all } j \neq i. \quad (7.4)$$

Now, since in [158] has been demonstrated that the likelihood PDF $p(\mathbf{x}|d_i)$ follows a multivariate Gaussian distribution, one can use the following Gaussian discriminant function [154]:

$$\psi_i(\mathbf{x}) = \frac{1}{(2\pi)^2 \det(\mathbf{\Sigma}_i)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \mathbf{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)} P(d_i) \quad (7.5)$$

where $(\cdot)^T$ denotes the transpose operation, $\mathbf{\Sigma}_i$ the likelihood covariance matrix for the i -th class, $\boldsymbol{\mu}_i$ denotes the likelihood mean vector for the i -th class, and $\det(\cdot)$ is the determinant function.

The next step is to estimate the following set of parameters: the mean vector $\hat{\boldsymbol{\mu}}_i$, the covariance matrix $\hat{\mathbf{\Sigma}}_i$, and the prior PMF $\hat{P}(d_i)$. In this research the sample mean and the sample covariance matrix were estimated from the reduced training set as described in Section 7.2; however, the prior class PMF was estimated from the complete dataset. Let us remark that the covariance matrix estimation will not be ill-posed due to the data sufficiency.

7.4 Probabilistic Neural Network Detection

Specht's Probabilistic Neural Network (PNN) is a semi-supervised neural network [179] used widely in pattern recognition applications. The PNN is inspired by Bayesian classification and does not require training. Although a PNN is robust for classification, it has large complexity $\mathcal{O}((N + 1)d)$; where N is the number of training samples and d is the dimension of the feature vectors.

The PNN estimates the true PDF of the feature vector assuming Gaussian distributions. The PNN has a four-layered architecture, as shown in Figure 7.2.

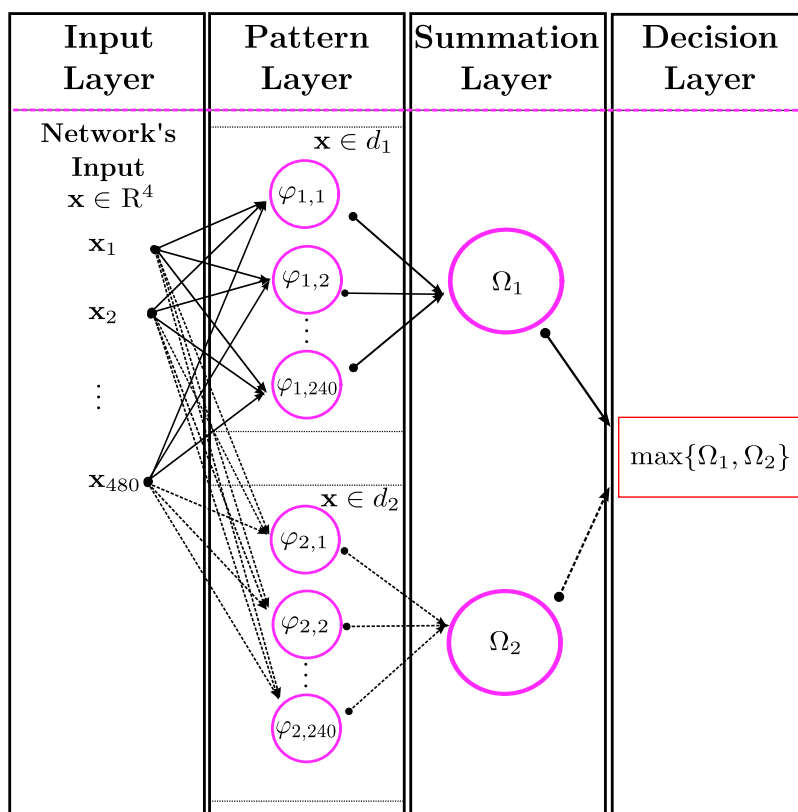


Figure 7.2: The hybrid architecture of the Probabilistic Neural Network. Note the probabilistic nature embedded in a neural architecture.

The input layer receives the features $\mathbf{x} \in \mathbb{R}^4$. The pattern layer contains radial basis functions $\varphi(\cdot)$ in each node. The number of nodes commensurates the N_i

number of samples for the i -th class. These nodes are called pattern units and are fully connected to the input nodes. The pattern layer output is defined as

$$\varphi_{ij}(\mathbf{x}) = \frac{1}{(2\pi)^2 \sigma^4} e^{-\frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x}_{ij})^T(\mathbf{x}-\mathbf{x}_{ij})}. \quad (7.6)$$

The summation layer contains summation units to complete the probability estimation. The number of summation units equals to the number of classes. The i -th summation unit receives input only from those pattern units belonging to the i -th class. This layer estimates the maximum likelihood of \mathbf{x} being classified as d . This is done by averaging and summarizing neuron's output belonging to the same class:

$$\Omega_i(\mathbf{x}) = \frac{1}{(2\pi)^2 \sigma^4} \frac{1}{N_i} \sum_{j=1}^{N_i} e^{-\frac{1}{2\sigma^2}(\mathbf{x}-\mathbf{x}_{ij})^T(\mathbf{x}-\mathbf{x}_{ij})} \quad (7.7a)$$

$$= \frac{1}{N_i} \sum_{j=1}^{N_i} \varphi_{ij}(\mathbf{x}). \quad (7.7b)$$

The last layer is called the “decision layer.” It classifies the feature vector \mathbf{x} according to the following Bayesian decision rule: Assign feature vector \mathbf{x} to class d_i with probability $p(d_i|\mathbf{x}) \equiv \Omega_i(\mathbf{x})$ if

$$\Omega_i(\mathbf{x}) > \Omega_j(\mathbf{x}) \quad \text{for all } j \neq i. \quad (7.8)$$

Thus, the maximum of the summation nodes output characterize the PNN general output. The function $\Omega_j(\cdot)$ gives the probability of the j -th class. This allows us to generate probabilistic visualizations of dust storms.

The parameter σ is estimated with Ramakrishnan, *et al.* method [152]. The authors normalize the feature vectors \mathbf{x}_i training the PNN using the following equation:

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \quad (7.9)$$

where x_{ij} is the j -th component of i -th feature vector \mathbf{x} , μ_j is the sample mean of the j -th component, and σ_j is the sample standard deviation of j -th component. The method developed is shown in Algorithm 7.1. This completes the PNN modeling

Algorithm 7.1 Ramakrishnan, *et al.* method [152] for estimating PNN parameter σ .

- 1: For each class, compute the variances of each component of the normalized training set (7.9).
 - 2: Find the difference between the two smallest variances.
 - 3: Set the value of σ equal to the difference obtained in Step 2.
-

since it needs no training phase.

7.5 Feed-Forward Back-Propagation Neural Network Detection

Multilayered feed-forward Neural Networks (FFNN) are of particular interest in pattern recognition and classification applications because they can approximate any square-integrable function to any desired degree of accuracy provided a training set [67, 84]. Therefore, we have designed a FFNN to model a dust storm by approximating the true posterior probability density function $p(d_i|\mathbf{x})$. A simple FFNN contains an input layer and an output layer, separated by l layers (the set of l layers is known as *hidden layer*) or neuron units. Given an input sample clamped to the input layer, the neuron units of the network compute their parameters according to the activity of previous layers. This research considers the particular neural topology where the input layer is fully connected to the first hidden layer, which is fully connected to the next layer until the output layer.

Given an input feature vector $\mathbf{x} \in \mathbb{R}^4$, the value of the j -th unit in the i -th layer is denoted $h_j^i(\mathbf{x})$, with $i = 0$ referring to the input layer, $i = l + 1$ referring to the output layer. We refer to the size of a layer as $|\mathbf{h}^i(\mathbf{x})|$. The default activation level

is determined by the internal bias b_j^i of that unit. The set of weights W_{jk}^i between $h_k^{i-1}(\mathbf{x})$ in layer $i - 1$ and unit $h_j^{i-1}(\mathbf{x})$ in layer i determines the activation of unit $h_j^i(\mathbf{x})$ as follows:

$$h_j^i(\mathbf{x}) = \Phi_{\text{sig}}(a_j^i(\mathbf{x})), \quad (7.10)$$

where $a_j^i(\mathbf{x}) = \sum_k W_{jk}^i h_k^{i-1}(\mathbf{x}) + b_j^i$, for all $i \in \{1, \dots, l\}$, with $h^0(\mathbf{x}) = \mathbf{x}$, and $\Phi_{\text{sig}}(\cdot) = \text{sigm}(\cdot)$ is the sigmoid activation function $\text{sigm}(a) = \frac{1}{1+e^{-a}}$. Given the last hidden layer, the output layer is computed similarly by

$$\mathbf{o}(\mathbf{x}) = \mathbf{h}^{l+1}(\mathbf{x}), \quad (7.11a)$$

$$\mathbf{h}^{l+1}(\mathbf{x}) = \Phi_{\text{sof}}(\mathbf{a}^{l+1}(\mathbf{x})), \quad (7.11b)$$

where $\mathbf{a}^{l+1}(\mathbf{x}) = \mathbf{W}^{l+1}\mathbf{h}^l(\mathbf{x}) + \mathbf{b}^{l+1}$, and the activation function $\Phi_{\text{sof}}(\cdot)$ is of the softmax which is known to better define class probabilities (see texts such as [16,24] for a detailed development). Thus, when an input sample \mathbf{x} is presented to the network, the application of (7.10) at each layer will generate a pattern of activity in the different layers of the neural network and produce an output with (7.11). The decision rule is to classify \mathbf{x} as d_i if the output neuron associated to the i -th class is greater than that associated with the j -th class. The posterior probability is then $p(d_i|\mathbf{x}) \equiv \Phi_{\text{sof}}^i(\mathbf{a}^{l+1}(\mathbf{x}))$.

The FFNN requires a training phase to build the model (\mathbf{W}, \mathbf{b}) . In this training phase, we used the ‘‘Levenberg-Marquardt’’ algorithm along with with a back-propagation strategy to update the weights \mathbf{W} and biases \mathbf{b} . As a learning function, we used the well established method of gradient descent with momentum weight and bias. The FFNN training phase ends when any of the following conditions holds:

- a number of 100 epochs (*i.e.* training iterations) is reached,
- the actual mean squared error (MSE) is 1×10^{-6} ,

- the gradient step size is less than or equal to 1×10^{-10} .

A well established technique for preventing over-fitting in the training was also implemented. This technique consists of partitioning the training set into two sets, training (80%) and validation (20%), such that when the MSE has not been decreased in the past five iterations using the internal validation set, the training phase stops and rolls back to the model (\mathbf{W}, \mathbf{b}) associated with the previous minimum MSE.

7.6 Linear Programming Support Vector Regression Detection

This study also includes the LP-SVR formulation (3.2), which was trained with the reduced training dataset using interior point methods and also with the complete training dataset for comparison. Algorithm 3.4 was used to train the LP-SVR for the complete training set. The LP-SVR parameters used are $\sigma = 0.125$, $C = 0.5$, and $\epsilon = 0.1$; these have been found with Algorithm 4.2.

The LP-SVR approach outputs an approximation to the true dust class posterior probability $p(d = 1|\mathbf{x})$, and the background posterior probability is assumed to be $p(d = 0|\mathbf{x}) = 1 - p(d = 1|\mathbf{x})$ by symmetry [77].

7.7 Results

The models presented in Sections 7.3 - 7.6 imply the process explained in the following three steps:

First, multispectral data from MODIS Level 1B thermal emissive bands B20, B29, B31 and B32 is recovered using (7.1).

Second, the recovered data is used to form i feature vectors \mathbf{x}_i with (7.2) and to define both a reduced training set $\mathcal{T}_r = \{\mathbf{x}_i, d_i\}_{i=1}^{N_r}$ as well as a larger training set

$\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{tr}}$, where $N_r = 480$ and $N_{tr} = 40$ million. MAP, PNN, FFNN, and LP-SVR are then trained with \mathcal{T}_r , and also the LP-SVR is trained with \mathcal{T} for comparison purposes. The testing set $\mathcal{D} = \{\mathbf{x}_i, d_i\}_{i=1}^{N_{te}}$ consists of $N_{te} = 45+$ million samples. The total dataset size N consisted of $N = 85+$ million samples.

Third, now that the classifiers are trained with \mathcal{T} or \mathcal{T}_r , dust detection is performed with \mathcal{D} and obtain a result using (7.4) for MAP, (7.8) for PNN, (7.11) for FFNN, and (3.2) trained with both the reduced and complete training sets. In this section we address the results of comparing the proposed dust storm detection models.

7.7.1 Comparison between Methods

We selected four performance metrics to evaluate the three proposed classification methods. These metrics are processing time per feature vector, precision, accuracy, and the area under the receiver operating characteristics (ROC) curve (AUC). The AUC is a widely used metric because of its superiority in reflecting the true performance of a classification system [60]. The precision and accuracy metrics can be computed with (C.13e) and (C.13c) respectively.

Sometimes these metrics may become biased towards false positive counts, and, considering that \mathcal{D} has more examples of the non-dust class, we use the balanced error rate (BER) defined in (4.5). Clearly, the BER meets the classical misclassification rate if there are equal number of samples per class, *i.e.*, classes are balanced (see [26]). The performance metrics were computed using only the validation set \mathcal{D} . The numerical results are shown in Table 7.1.

The interpretation of results in Table 7.1 are as follows. In terms of performance metrics, the PNN approach classifies better than the FFNN approach among the neural classifiers. Among the statistical classifiers, the LP-SVR approach classifiers better than the MAP classifier. Among all, the LP-SVR and the PNN approaches are the best. The superior AUC and smaller BER of LP-SVR may be attributed to

Table 7.1: Classifiers Performance. The processing time shown here is in milliseconds.

	Precision	Accuracy	AUC	BER	P. Time
MAP	0.5255	0.6779	0.4884	0.2259	0.0141
PNN	0.8080	0.8816	0.7035	0.0536	0.2393
FFNN	0.7664	0.8412	0.6293	0.0729	0.0459
LP-SVR	0.7907	0.8678	0.7117	0.0502	0.0809
LP-SVR \mathcal{T}	0.8295	0.9104	0.7349	0.0318	0.0974

the fact that it is solving an optimal classification problem; PNN is also shows high precision and accuracy because it is very close to a Mixture of Gaussian classification problem formulation, which both are well known to perform very well in two-class classification problems.

The results of Table 7.1 also suggest that LP-SVR was unbiased and that MAP was biased towards one class as its AUC and BER show. However, the MAP method proved to be faster than the other approaches as expected. Alternatively, the FFNN and LP-SVR show a good balance of speed, small error, good accuracy, and precision.

The processing time is an important measure when modeling “real-time” processing systems. In this case “real-time” means that a feature vector must be processed at least as fast as the data is being captured by the MODIS instrument. In the case of the MODIS instrument, a complete scan (*i.e.* 10×1053 pixels at 36 bands) is produced every 6.25 seconds. In this case, it produces almost one feature vector in 0.5 milliseconds. Thus, a real-time system must perform a classification in less than or equal to this time. Table 7.1 show the processing time per scan in milliseconds. Therefore, since the MAP and LP-SVR approaches take much less time to classify a feature vector, and since the PNN and FFNN approaches take less than 0.25 milliseconds to produce the classification result, both groups (*i.e.*, neural and statistical classifiers) can be considered suitable for real time detections at 1km resolution. In

contrast, the MODIS AOT product takes two days to be produced and released at a 10km resolution.

Thus far, it has been demonstrated numerically that the models studied are capable of assessing dust aerosol detection. However, we also want to perform a visual assessment of the results provided by the classifiers under study. For a visual assessment, we separate the results into two categories: dust over land, and dust over ocean. Since the methods were modeled using only dust over land data, the validation using events with dust over ocean will demonstrate the generalization capabilities of the methods.

7.7.2 Dust over Land

A major difficulty in detecting dust over land is the variability of the dust physical properties depending on the region. This soil variability can adversely impact the performance of the models. The reader must recall that the models were constructed using data from south-western US events. Therefore, here we demonstrate that the models possesses the ability to detect dust independently of the soil type.

Let us consider a case of dust storm detection within the south-western US region, *e.g.*, the dust storm of April 10, 2001, covering Chihuahua, Texas, and New Mexico. The true color image is shown in Figure 7.3 (a) and a region of interest in Figure 7.3 (j). Figure 7.3 (b) shows the result reported by the MODIS Aerosol Optical Density (AOD) level 2 product, where higher intensities are associated with more dense aerosols. Figure 7.3 (c) shows the result using Ackerman’s method [1,184]. Figure 7.3 (d) depicts the result obtained with Hao’s method [76], where brighter intensities are associated with a higher Thermal Dust Index (TDI). The dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively is depicted in Figure 7.3 (e)-(i) respectively. Clearly, PNN, FFNN, and SVR approaches perform well from the visual assessment perspective, while the MAP method shows conservative results. Ackerman’s method and Hao’s TDI are

well established visualization methods, they are not classification methods, and here we use them to ease the visual location of the dust.

Now, let us consider a case of a dust storm in a totally different region and soil type. We consider the dust storm from June 05, 2009, covering a very large area in the middle east. The true color image is shown in Figure 7.4 (a) and a region of interest in Figure 7.4 (j). Figure 7.4 (b) shows the result reported by the MODIS AOD product. Figure 7.4 (c) shows the result using Ackerman's method. Figure 7.4 (d) depicts the TDI obtained with Hao's method. The dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively is presented in Figure 7.4 (e)-(i) respectively. Clearly, the models have the capacity to detect the different types of dust without changing any parameters and without re-training any model.

7.7.3 Dust over Ocean

So far, the proposed methods performed well in training and validation cases for different types of dust. This suggests that the physical properties exhibited by the dust over ocean and over land are similar in the near infrared spectral range. These findings confirm Ackerman's, *et al.* [1, 184] and also Hao's, *et al.* [76] conclusions about dust aerosols over land and ocean.

The dust storm in Argentina on January 24, 2010, shown in Figure 7.5 (a), originated over land and extended for miles over the ocean. Figure 7.5 (b) shows the result reported by the MODIS AOD product. Figure 7.5 (c) shows the result using Ackerman's method. Figure 7.5 (d) depicts the TDI obtained with Hao's method. The dust storm's probability for MAP, PNN, FFNN, LP-SVR, LP-SVR trained with the complete training set is shown in Figure 7.5 (e)-(i) respectively. The region of interest is depicted in Figure 7.5 (j).

Similarly, we present the detection of a dust storm in Australia on September 26, 2009, shown in Figure 7.6 (a). The benchmark results of MODIS AOD, Ackerman's

method, and Hao's TDI are shown in Figure 7.6 (b)-(d) respectively. The proposed methods detected the dust storm over land and ocean as shown in Figure 7.6 (e)-(i). Furthermore, the methods can accurately discriminate similar signatures like smoke and clouds. It is evident that the methods provide a strong ability to infer non-trivial multispectral data relationships. Particularly the PNN, due to its hybrid neuro-statistical nature, performed better than MAP.

7.7.4 Extension to Segmentation and Classification

The presented models aimed to approximate the true class probabilities $p(d_i|\mathbf{x})$ to perform a Bayesian-like decision in the dust storm detection problem. This has an important advantage: the models can easily be translated to a classification or segmentation problem. Then, the classification problem is simply posed as the following decision rule [77]: Feature vector \mathbf{x} is classified as d_1 (dust) if

$$(\lambda(\alpha_0|d_1) - \lambda(\alpha_1|d_1)) p(d_1|\mathbf{x}) > (\lambda(\alpha_1|d_0) - \lambda(\alpha_0|d_0)) p(d_0|\mathbf{x}) \quad (7.12)$$

otherwise the feature vector is classified as d_0 (background). In (7.12) α_0 is the action of classifying \mathbf{x} as background, α_1 is the action of classifying \mathbf{x} as dust, and $\lambda(\alpha_i|d_j)$ denotes the loss of deciding α_i when the true class is d_j . Since the LP-SVR outputs only $p(d_1|\mathbf{x})$, then we assume $p(d_0|\mathbf{x}) = 1 - p(d_1|\mathbf{x})$ by symmetry [16, 77].

To illustrate the classification or segmentation of dust regions we define the following loss function:

$$\lambda(\alpha_i|d_j) = \begin{cases} 0 & i = j \\ 2\tau & i \neq j, \end{cases} \quad (7.13)$$

where τ controls the weight of misclassification. Clearly, if $\tau = 0.5$, then (7.13) becomes the traditional *symmetrical* or *zero-one* loss function [77].

Since the models described in this paper aim to approximate true posterior class

membership probabilities $p(d_i|\mathbf{x})$, the minimum error choice corresponds to a value of $\tau = 0.5$ [77]. However, the value of τ may be application dependent. For instance, let us consider military applications in which one might want to penalize false positives, in this case it would be appropriate to use some $\tau > 0.8$ or even $\tau > 0.9$.

As an example of segmentation let us consider the case of the dust storm of April 6, 2001, whose true color image is shown in Figure 7.7 (a) and a region of interest in (b). The probability of dust as approximated by the PNN approach is shown in Figure 7.7 (c). The segmentation, based on different weights (*i.e.* $\tau = \{0.5, 0.8, 0.9\}$), is shown in Figure 7.7 (d).

7.8 Conclusion

We compared four methods for dust storm detection: the Maximum *a Posteriori* Classifier (MAP), the Probabilistic Neural Network (PNN), the Feed-Forward Neural Network (FFNN), and the Linear Programming Support Vector Regression (LP-SVR).

The comparison among classifiers was made using different performance metrics, after the classifiers were modeled using a reduced training dataset. This comparison also included a large-scale approach to LP-SVR, that allowed us to use the complete (non-reduced) training dataset. Numerical results showed that the reduced training dataset contains significant data for modeling and performing dust detection based on very few feature vectors. More explicitly, we extracted features from MODIS thermal emissive spectral bands: B20, B29, B31, and B32. Sample feature vectors were reduced following a concept introduced by Mather. The reduction was performed with KLT to preserve the most discriminant samples. The model's parameters were estimated from these reduced sample feature vectors, aiming to approximate true posterior class membership probabilities.

Among the neural-network-based classifiers, the hybrid approach, PNN, per-

formed better than the FFNN after a numerical evaluation of different performance metrics. The LP-SVR approach demonstrated to be better than the MAP. Visually, the four methods performed an accurate detection.

MAP, PNN, FFNN, and LP-SVR were modeled using known cases from the southwestern US and north-western Mexico over land observations. Nevertheless, the methods provided accurate detection when tested over land at different geographical regions with different soil types. Furthermore, all methods were able to classify dust storms over the ocean. Numerical and visual results suggest that the MAP, PNN, FFNN, and LP-SVR classifiers are soil-independent; this in turn suggests that dust aerosols physical properties over land and ocean are very similar when analyzed within the near-infrared spectral range.

Furthermore, the four methods currently output 1 km spatial resolution results, which improves traditional Aerosol Optical Density (AOD)-based methods at 10 km spatial resolution.

In general, the four methods can be effectively utilized in the analysis of stratospheric dust, thereby helping researchers in the understanding of dust aerosol activity and transport.

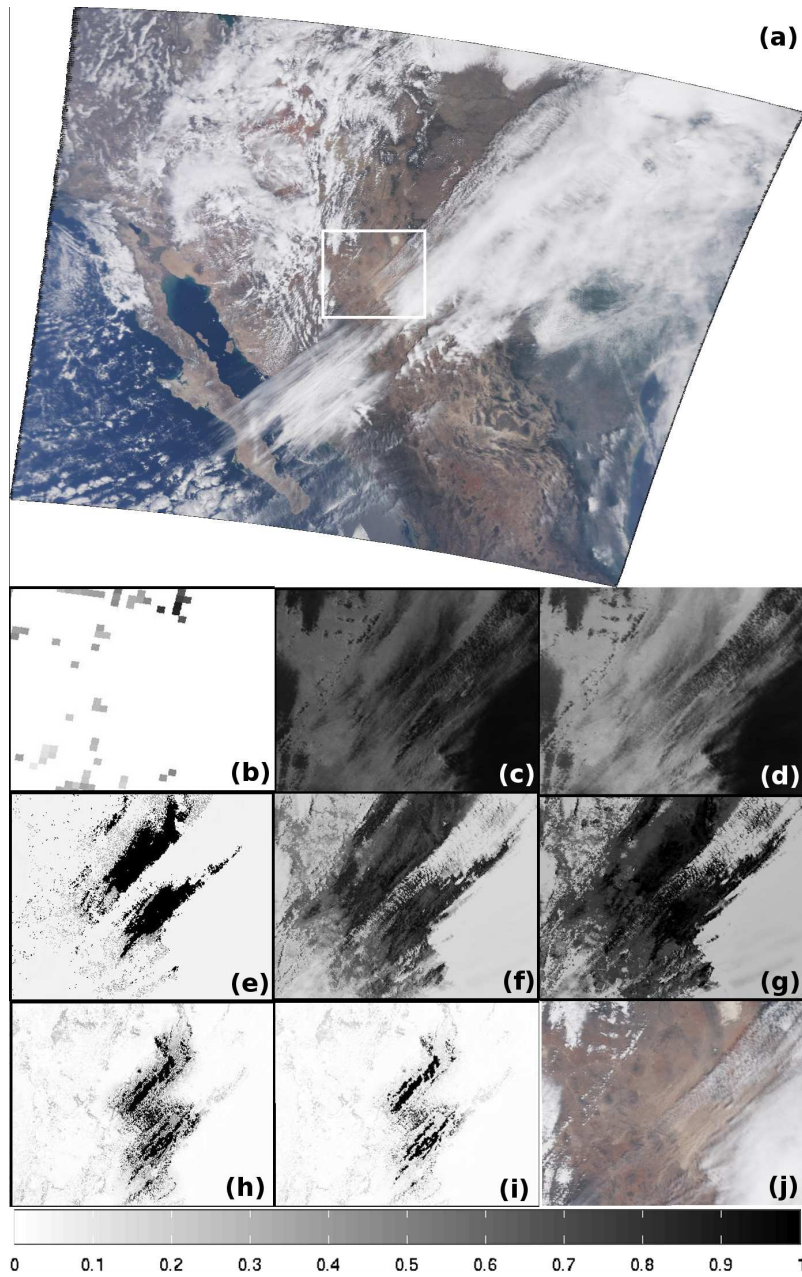


Figure 7.3: Dust storm over southwestern US. In (a) is shown the true color image and a region of interest in (j). In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 04/10/2001. Time 18:05 UTC. Satellite: Terra. Instrument: MODIS.

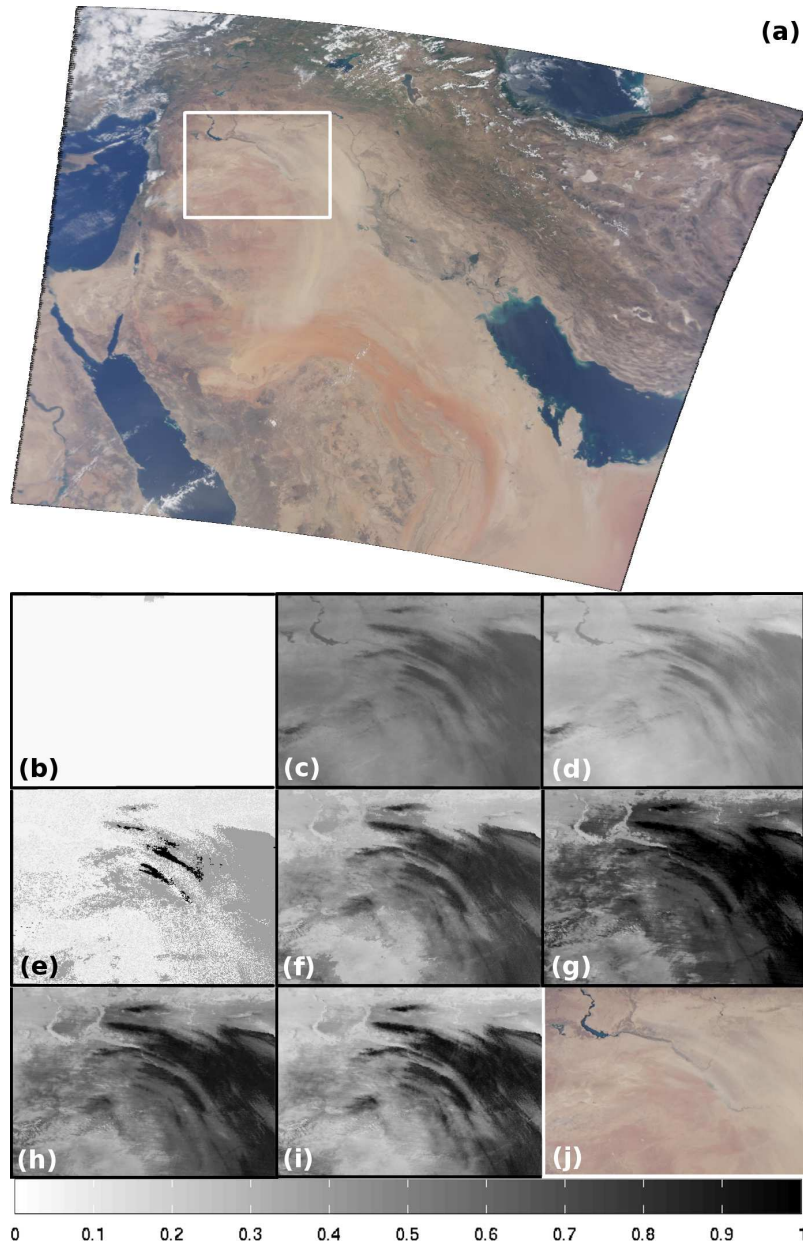


Figure 7.4: Dust storm over the middle east. In (a) is shown the true color image and a region of interest in (j). In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 06/05/2009. Time 07:50 UTC. Satellite: Terra. Instrument: MODIS.

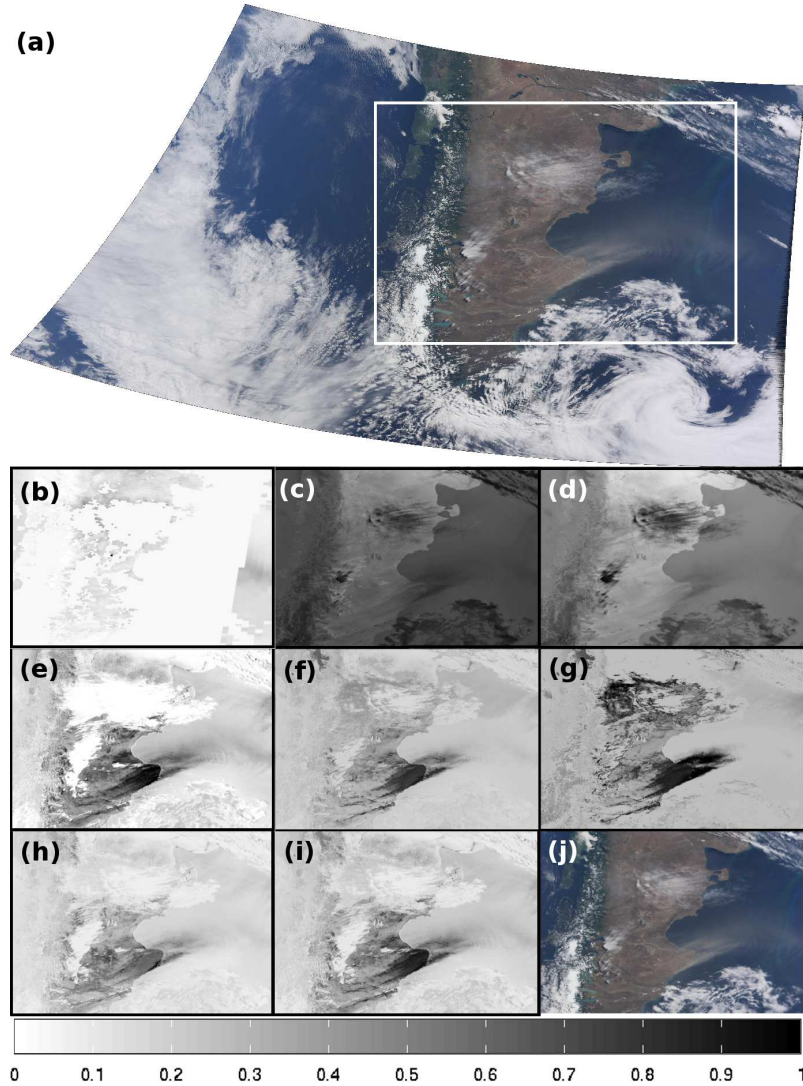


Figure 7.5: Dust storm over Argentina. In (a) is shown the true color image, and (j) is the region of interest. In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR, and LP-SVR trained with the complete training set respectively. Date 01/24/2010. Time 14:40 UTC. Satellite: Terra. Instrument: MODIS.

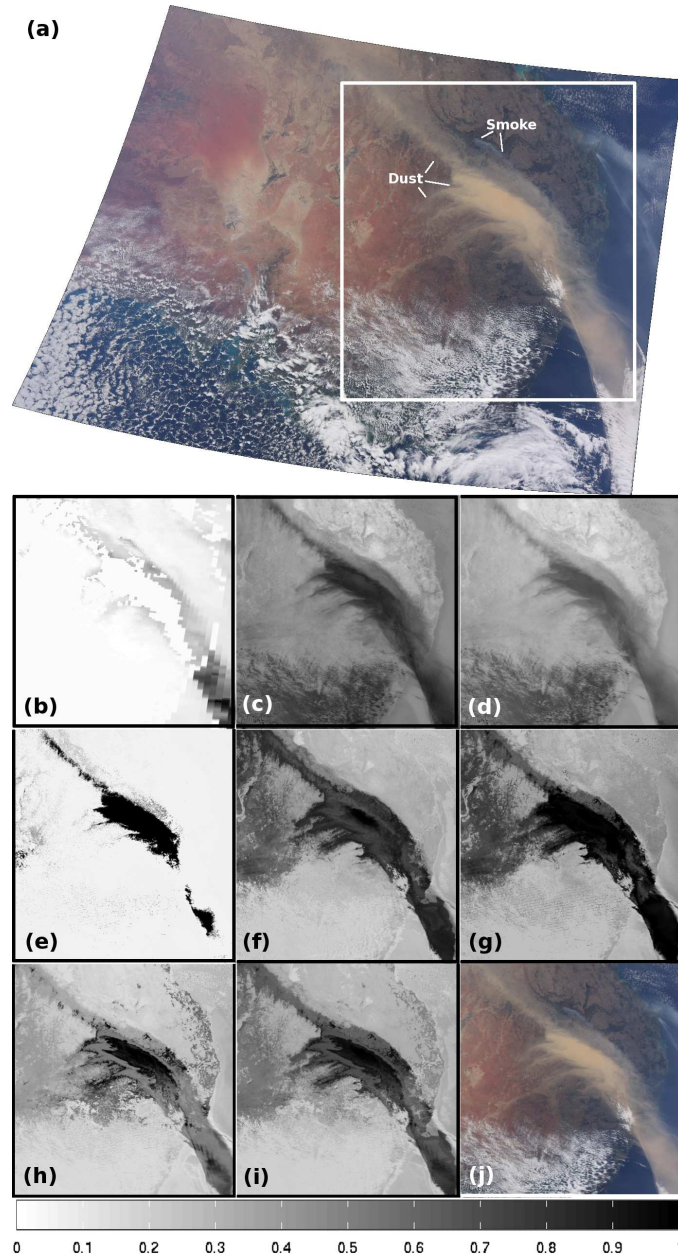


Figure 7.6: Dust storm over Australia. In (a) is shown the true color image, and (j) is the region of interest. In (b) the MODIS AOD. In (c) the result using Ackerman's method. In (d) the TDI according to Hao's method. In (e)-(i) the dust storm probability using MAP, PNN, FFNN, LP-SVR and LP-SVR trained with the complete training set respectively. Date 09/26/2009. Time 00:35 UTC. Satellite: Aqua. Instrument: MODIS.

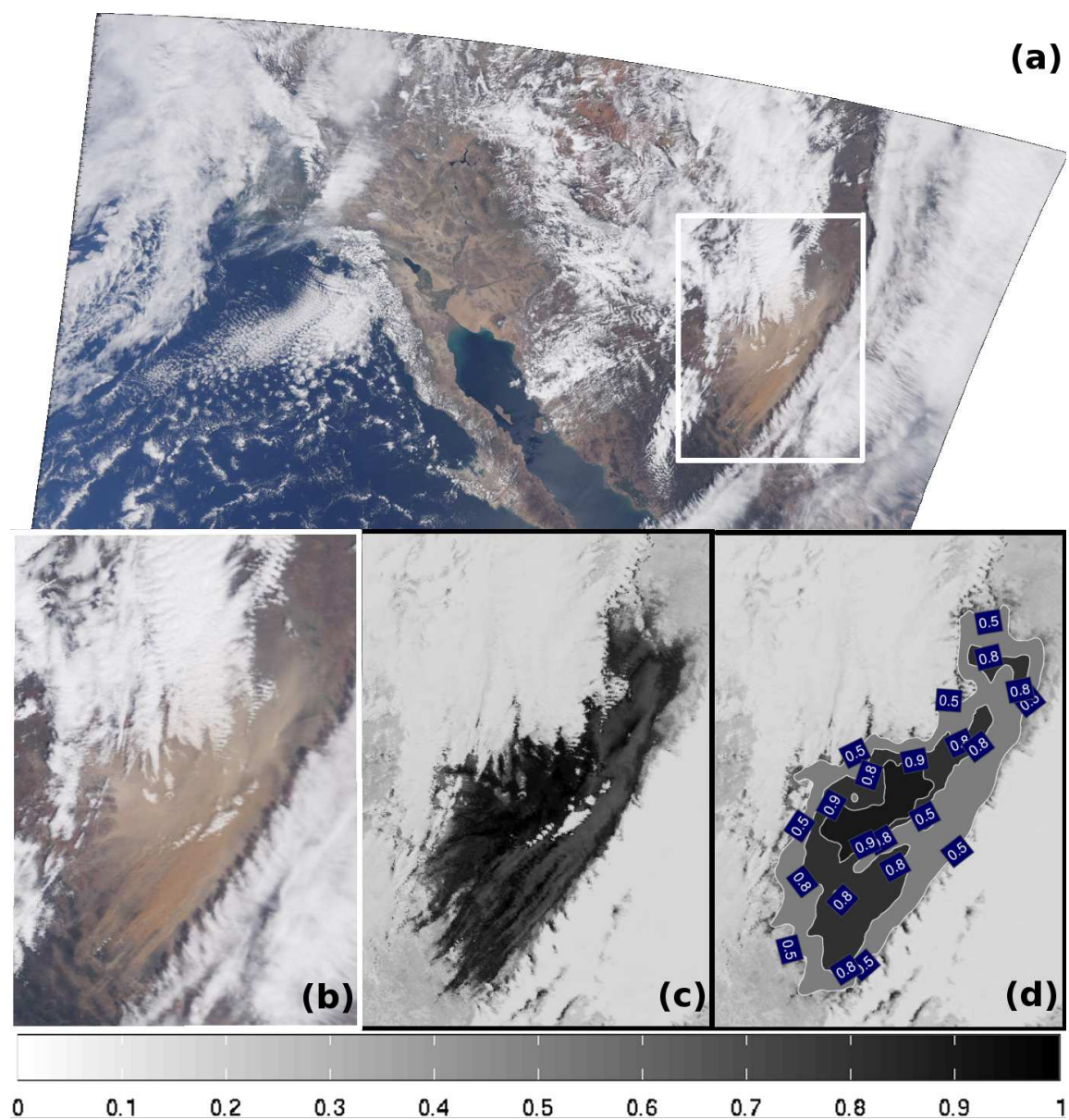


Figure 7.7: Dust storm over southwestern US and northwestern Mexico. In (a) is shown the true color image. In (b) the region of the dust storm. In (c) the dust storm probability $p(d_1|\mathbf{x}) \equiv \Omega_1(\mathbf{x})$ using PNN. In (d) the classification/segmentation regions found using $\tau = 0.5$, $\tau = 0.8$, and $\tau = 0.9$. Date 04/06/2001. Time 18:30 UTC. Satellite: Terra. Instrument: MODIS.

Chapter 8

Conclusions

The proposed methods deal with important issues in SVM and SVR theory. These methods allow the training over large data sets on a computationally tractable fashion.

Although conclusions have been given at each chapter, in this final chapter we summarize and discuss the conclusions in a chapter-wise fashion. Each section will provide a summary of conclusions related to that particular chapter topic. Then we will discuss future work and conclude the chapter with a list of publications.

8.1 Contributions

8.1.1 Large Scale LP-SVR

The major contribution of this research is an algorithm to train large-scale LP-SVRs. This contribution can be explained in three parts. First, we posed a Support Vector Regression (SVR) problem as an efficient Linear Programming (LP) problem. Such model was developed in a primal-dual fashion in such a way that the KKT conditions are necessary and sufficient for optimality.

Second, we introduced a large-scale training method based on a variables decomposition algorithm, a constraints decomposition algorithm, and primal-dual interior point methods. The proposed method finds a solution that satisfies optimality conditions in finite time while converging to a global minimum. Each of the variable-constraint-reduced LP-SVR sub-problems are solved using interior point methods

(IPM), which are known to have a very high rate of convergence. IPM is the key that balances the total computational expense of performing two decompositions and solving several LPs. Using other approaches (*e.g.* the simplex method) would dramatically increase computational efforts.

Third, experimental results demonstrate that the proposed approach is comparable with other formulations in terms of performance, which is desirable. Furthermore, experimental results shown that the model is sparser than the regular SVR and other SV-based classifiers. Particularly, experiments demonstrate that as the problem size increases, the sparser the solution becomes, and more computational efficiency can be gained in comparison with other methods.

Finally, let us remark that the experiments performed over the large-scale LP-SVR approach comprised state-of-the art benchmarking datasets, mostly from the UCI machine-learning repository [65]. Such datasets include the following: the *Ripley* dataset [138, 156], the *Wine* dataset [64, 107], the Adult database [98, 148], the MNIST database of handwritten digits [39, 112, 148], the *HIVA* dataset related to HIV infections, the Twenty-Newsgroup dataset [97], the Covertype dataset [17, 39], the *Iris* dataset [63, 77], the *Spiral* dataset [206], and the “sinc” function approximation problem [144]. In each dataset, the LP-SVR demonstrated competitive results when compared to other classifiers.

8.1.2 LP-SVR Training Speedup and Hyper-Parameters Estimation

Following the major contribution of a large-scale training method for an LP-SVR, we explored two other problems that are considered open problems within machine learning: (i) how to accelerate or improve the training time and (ii) how to estimate the hyper-parameters of SV-based learning machines. We have addressed problem (i) using statistical tools and problem (ii) well known optimization techniques.

We developed a method to reduce the LP-SVR training time, based on the fact that the support vectors (SVs) are likely to lie on the convex hull of each class. The algorithm we designed uses the Mahalanobis distance from the class sample mean, in order to rank each sample in the training set. Then, the samples with the largest distances are used as part of the initial working set. Experimental results shown a reduction in the total training time, as well as a significant decrease in the total iterations percentage. Results also suggest that using the speedup strategy the SVs are found earlier in the learning process. Moreover, the speedup strategy is not particular to LP-SVR and can be extended to other SV-based learning methods.

Next, we presented a method that finds the set of LP-SVR hyper-parameters. This scheme uses a line-search-based quasi-Newton method for function minimization. The proposed approach uses a \mathcal{K} -fold cross validation technique as a true test generalization error estimator. We particularly proposed the usage of different error functions for each type of problem: two-class, multi-class, and regression. The combination of a proper selection of good error functions, stable true generalization error estimator, and powerful optimization technique, resulted in a robust algorithm. Experimental results show that the algorithm performs an estimation of hyper-parameters that minimize the true test generalization error. Finally, let us remark that this approach can be easily adapted to other classification methods that require an estimation of hyper-parameters. However, the algorithm, as we developed, will work only for those classification methods that require an exact number of two hyper-parameters.

8.1.3 Applications

To show the robustness of the proposed large-scale LP-SVR learning method, this dissertation discusses different applications representative of large-scale real-life problems in the Smart Grid community, image processing area, and earth sciences. The following paragraphs summarize such contributions.

Power Load Prediction

Accurate forecasting of electric power consumption by the national electric power grid is critical for short term operations and long term utilities planning. The power load prediction impacts a number of decisions (*e.g.*, which generators to commit for a given period of time) and broadly affects wholesale electricity market prices.

Within the realm of short term power load prediction, we used the large-scale linear programming support vector regression (LP-SVR) model. The LP-SVR is compared with other two non-linear regression models: Feed Forward Neural Networks (FFNN) and Bagged Regression Trees (BRT). The three models are trained to predict hourly day-ahead loads given temperature predictions, holiday information and historical loads. Experimental results indicate that the proposed LP-SVR method gives the smallest error when compared against the other approaches. The LP-SVR shows a mean absolute percent error of 1.58% while the FFNN approach has a 1.61%. Similarly, the FFNN method shows a 330MWh (Megawatts-hour) mean absolute error, whereas the LP-SVR approach gives a 238MWh mean absolute error. This is a significant difference in terms of the extra power that would need to be produced if FFNN was used.

We conclude that LP-SVR model can be utilized for predicting power loads to a very low error, and it is comparable to FFNN and over-performs other state of the art methods such as: Bagged Regression Trees, and Large-Scale SVRs.

Texture Classification

We also showed that the large-scale (LS) linear programming support vector regression (LP-SVR) approach, is a powerful classification scheme for supervised texture segmentation. Researchers have explored this area with good results; however, no scheme based on LP-SVR has been reported as far as we know. For feature extraction we used a well known local energy scheme in combination with a redundant

directional pyramid image decomposition. This type of filter bank was proposed by Rosiles and Smith for texture segmentation [163]. Experimental results demonstrate that combining the proposed LS LP-SVR with wavelet-based features provide better results than those works reported to date.

A remarkable contribution is that we introduced different classifier configurations or architectures that took advantage of the multiresolution structure of the wavelet-based texture features. Such architectures demonstrate the flexibility of the proposed LS LP-SVR training scheme. These architectures include an ensemble-based configuration and a multiresolution or cascaded configuration.

Typically SVRs are used for regression problems; however, it is well known that can be used for classification as well. Thus, we claim that such architectures for LP-SVRs represent a unique contribution in the sense that no architectures like these have been reported for LP-SVRs in a multi-class classification problem.

Dust Storm Detection

Advances in remote sensing like multispectral instruments allow imaging of atmospheric and earth materials based on their spectral signature over the optical range. Methods to analyze dust aerosols have been proposed; however, there is paucity on specialized classification-based approaches for dust storm detection.

We compared four methods for dust storm detection: the Maximum *a Posteriori* Classifier (MAP), the Probabilistic Neural Network (PNN), the Feed-Forward Neural Network (FFNN), and the Linear Programming Support Vector Regression (LP-SVR). We extracted features from NASA MODIS thermal emissive spectral bands: B20, B29, B31, and B32. Sample feature vectors were reduced in size while preserving class manifolds. The model's parameters were estimated from these reduced sample feature vectors, aiming to approximate true posterior class membership probabilities.

Numerical results demonstrate that the reduced training dataset contains significant data for modeling and performing dust detection based on very few feature

vectors. The LP-SVR approach demonstrated to be better than the MAP classifier.

LP-SVR was modeled using known cases from the south-western US and north-western Mexico over land observations. Nevertheless, the method provided accurate detection when tested over land at different geographical regions with different soil types. Furthermore, LP-SVR was able to classify dust storms over the ocean. Numerical and visual results suggest that the LP-SVR classifier is soil-independent; this in turn suggests that dust aerosols physical properties over land and ocean are very similar when analyzed within the near-infrared spectral range.

Furthermore, the LP-SVR method currently output 1 km spatial resolution results, which improves traditional Aerosol Optical Density (AOD)-based methods at 10 km spatial resolution. In general, the LP-SVR can be effectively utilized in the analysis of stratospheric dust, thereby helping researchers in the understanding of dust aerosol activity and transport.

8.2 Future Work

8.2.1 Theoretical

LP-SVR Model

As a future work on the LP-SVR model, we want to use different loss functions apart from the ϵ -insensitive loss function used throughout in this document. Particularly, we want to use the Huber loss function [29, 88, 132]. The reason is that the Huber loss function behaves linearly at higher errors and quadratically at smaller errors:

$$L_H(d, f(\mathbf{x})) = \begin{cases} |d - f(\mathbf{x})| - \epsilon & \text{for } |d - f(\mathbf{x})| \geq \epsilon \\ \frac{(d - f(\mathbf{x}))^2}{4\epsilon} & \text{otherwise,} \end{cases} \quad (8.1)$$

where ϵ is a prescribed parameter.

We want to explore as well the quadratic loss function [83]:

$$L_Q(d, f(\mathbf{x})) = (d - f(\mathbf{x}))^2, \quad (8.2)$$

and the Laplacian loss function [36]:

$$L_P(d, f(\mathbf{x})) = |d - f(\mathbf{x})|. \quad (8.3)$$

The theoretical development consist on having to restate the optimization model and reformulate from scratch the SVR and pose it as an LP program if possible. Then, derive the KKT conditions and see if they can be reduced or simplified for computational efficiency. Also, the learning algorithm would have to be re-formulated since the LP program suffered a changed in the variables. This change aims to take advantage of the new LP-SVR structure.

Another research study would involve the possibility of introducing ϵ , C , or σ into the optimization problem without increasing the complexity of the SV-based regression machine. More specifically, we should avoid the insertion of hyper-parameters as part of the optimization problem if at the same time we introduce new variables or new degrees of freedom.

Finally, another future work consists of seeking novel LP formulations related to SV-based models that may have lower complexity. For instance, by performing research about novel LP problems that regularize an SV-based model or relax some of the current LP variables. However, this would require a large amount of time, possibly worthy of a totally new dissertation. Mainly because every algorithm discussed in this dissertation would change if the LP model is different from the one we are considering here.

LP-SVR Hyper-Parameters Estimation

An interesting point of improvement, in the proposed algorithm for hyper-parameter estimation, is the estimation of a “good” initial point. We should attempt to use more well-established techniques that cover a broad area of the error surface. Especially, a biologically-inspired heuristics such as Evolution Strategies (ES).

The ES of the type $(1, \lambda)$ -ES [10] and the CMA-ES [75] seem to be the better options for this case. This choice is made because both algorithms have a fast rate of convergence and low computational complexity.

Following ES algorithms we could find a good initial point by stopping ES algorithms at the slightest sign of a steady error variation. In this manner, our Newton-based algorithm can remain exactly the same.

Another chance of improvement would be to use different error functions to minimize. Particularly, we are interested on finding a pair of functions that produce a result in the same domain and interval, and such that each function covers a broader, yet different, kind of error surfaces. In an ideal world, a good pair of functions would provide a smooth error surface; however, current research achievements show no evidence of such functions. We believe such a pair of functions can be derived in the near future.

8.2.2 Experimental

In regard to the large-scale LP-SVR approach, we would leave as future work the inclusion of more comparisons to other related large-scale SVR approaches. This can be achieved either by obtaining compatible freely-distributed source codes, or, by translating our approach to other languages compatible with the freely-distributed source codes.

We leave as future work the theoretical and experimental extension of the learning-speedup to other SV-based learning methods, such as: ν -SVM, LP-SVM, and others.

The hyper-parameters can clearly be extended to other learning methods; however, we leave as future work the exploration of other classifiers that have hyper-parameters to compute, such as ν -SVM, LP-SVM, neural networks, fuzzy inference systems, k-means, and others.

In regard to the texture classification problem, we want to explore the relationship between multispectral features and the number of classes in an LP-SVR. If there is such a relationship, then we want to find a lower bound to the number of features required as a function of the number of classes.

8.3 Publications

This research has produced many sources of publications given the large number of possible applications as well as the theoretical development of the large-scale LP-SVR method itself. In the following paragraphs we provide the list of publications produced.

8.3.1 Theoretical Development

- In regard to the decomposition method for large-scale LP-SVR and its convergence. The paper is entitled “SEQUENTIAL OPTIMIZATION FOR LARGE SCALE SUPPORT VECTOR MACHINES FOR REGRESSION,” targeted for the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Paper ready for submission.
- The theoretical development of the speedup strategy and the parameter estimation algorithm produced the following paper: “TRAINING SPEEDUP AND HYPER-PARAMETER ESTIMATION FOR SV-BASED LEARNING MACHINES,” targeted for the *Journal of Neurocomputing*. Paper ready for submission.

8.3.2 Applications

In regard to applications of this work. We have the following papers:

- “STATISTICAL AND NEURAL PATTERN RECOGNITION METHODS FOR SOIL-INDEPENDENT DUST AEROSOL DETECTION,” for the *IEEE Transactions on Geoscience and Remote Sensing*. Paper ready for submission.
- “TEXTURE SEGMENTATION USING WAVELET-BASED FEATURES AND LINEAR PROGRAMMING SUPPORT VECTOR REGRESSION,” for the *EURASIP Journal on Image and Video Processing*, Paper ready for submission.
- “SHORT TERM ELECTRIC POWER CONSUMPTION FORECASTING USING LINEAR PROGRAMMING SUPPORT VECTOR REGRESSION,” for the *1st Southwest Energy Science and Engineering Symposium 2011*. Paper submitted.
- “LARGE-SCALE SONAR TARGET DETECTION WITH ℓ_1 -NORM SV REGRESSION BASED ON UNFEASIBLE INTERIOR POINT METHODS,” for the *2011 ITEA Live-Virtual-Constructive Conference*. Paper published.

References

- [1] S.A. Ackerman. Remote sensing aerosols using satellite infrared observations. *Journal of geophysical research*, 102(D14):17069, 1997.
- [2] A. Agarwal, H.M. El-Askary, T. El-Ghazawi, M. Kafatos, and J. Le-Moigne. Hierarchical PCA Techniques for Fusing Spatial and Spectral Observations With Application to MISR and Monitoring Dust Storms. *IEEE Geoscience and Remote Sensing Letters*, 4(4):678–682, 2007.
- [3] F.B. Akoa. Combining dc algorithms (dcas) and decomposition techniques for the training of nonpositive semidefinite kernels. *Neural Networks, IEEE Transactions on*, 19(11):1854–1872, nov. 2008.
- [4] S. Aksoy, K. Koperski, C. Tusk, G. Marchisio, and J.C. Tilton. Learning Bayesian classifiers for scene classification with a visual grammar. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):581–589, 2005.
- [5] H. Al-Mubaid and S.A. Umair. A new text categorization technique using distributional clustering and learning logic. *Knowledge and Data Engineering, IEEE Transactions on*, 18(9):1156–1165, sept. 2006.
- [6] D. Anguita, A. Boni, S. Ridella, F. Riveccio, and D. Sterpi. Theoretical and practical model selection methods for support vector classifiers. *Support vector machines: theory and applications*, pages 159–179, 2005.
- [7] D. Anguita, S. Ridella, F. Riveccio, and R. Zunino. Hyperparameter design criteria for support vector classifiers. *Neurocomputing*, 55(1-2):109–134, 2003.
- [8] R. Ansari, C.W. Kim, and M. Dedovic. Structure and design of two-channel filter banks derived from a triplet of halfband filters. *Circuits and Systems*

- II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(12):1487–1496, 1999.
- [9] Miguel Argáez and Leticia Velázquez. A new infeasible interior-point algorithm for linear programming. In *Proceedings of the 2003 conference on Diversity in computing*, TAPIA '03, pages 12–14, New York, NY, USA, 2003. ACM.
 - [10] A. Auger. Convergence results for the $(1, [\lambda])$ -SA-ES using the theory of irreducible Markov chains. *Theoretical Computer Science*, 334(1-3):35–69, 2005.
 - [11] R.H. Bamberg and M.J.T. Smith. A filter bank for the directional decomposition of images: Theory and design. *Signal Processing, IEEE Transactions on*, 40(4):882–893, 1992.
 - [12] K.P. Bennett and E.J. Bredensteiner. Duality and geometry in SVM classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000), June 29-July 2, 2000, Stanford University*, pages 57–64. Citeseer, 2000.
 - [13] K.P. Bennett and E.J. Bredensteiner. Geometry in learning. *Geometry at work: a collection of papers showing applications of geometry*, page 132, 2000.
 - [14] R.A. Berk. Bagging. *Statistical Learning from a Regression Perspective*, pages 1–24, 2008.
 - [15] E. Bermani, A. Boni, A. Kerhet, and A. Massa. Kernels evaluation of SVM-based estimators for inverse scattering problems. 2004.
 - [16] C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.

- [17] J.A. Blackard and D.J. Dean. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture*, 24(3):131–151, 1999.
- [18] Liefeng Bo, Licheng Jiao, and Ling Wang. Working set selection using functional gain for ls-svm. *Neural Networks, IEEE Transactions on*, 18(5):1541–1544, sept. 2007.
- [19] A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4):712–727, april 2008.
- [20] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, San Mateo, CA, 1992. ACM, Morgan Kaufman.
- [21] A.C. Bovik, M. Clark, and W.S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 55–73, 1990.
- [22] PS Bradley and OL Mangasarian. Massive data discrimination via linear support vector machines. *Optimization methods and software*, 13(1):1–10, 2000.
- [23] PS Bradley, OL Mangasarian, and DR Musicant. Optimization methods in massive data sets. In *Handbook of massive data sets*, pages 439–471. Kluwer Academic Publishers, 2002.
- [24] M.J. Canty. *Image analysis, classification and change detection in remote sensing: with algorithms for ENVI/IDL*. CRC, 2007.

- [25] L.J. Cao, S.S. Keerthi, Chong-Jin Ong, J.Q. Zhang, U. Periyathamby, Xiu Ju Fu, and H.P. Lee. Parallel sequential minimal optimization for the training of support vector machines. *Neural Networks, IEEE Transactions on*, 17(4):1039 – 1049, july 2006.
- [26] G.C. Cawley. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In *IEEE International Conference on Neural Networks, 2006. IJCNN '06.*, 0 2006.
- [27] Y. Censor and S.A. Zenios. *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, USA, 1997.
- [28] T. Chang and C.C.J. Kuo. Texture analysis and classification with tree-structured wavelet transform. *Image Processing, IEEE Transactions on*, 2(4):429–441, 1993.
- [29] O. Chapelle and S.S. Keerthi. Efficient algorithms for ranking with SVMs. *Information retrieval*, 13(3):201–215, 2010.
- [30] Guangxi Chen, Yan Cheng, and Jian Xu. Cluster reduction support vector machine for large-scale data set classification. In *Computational Intelligence and Industrial Application, 2008. PACIIA '08. Pacific-Asia Workshop on*, volume 1, pages 8 –12, dec. 2008.
- [31] Guangxi Chen, Jian Xu, and Xiaolin Xiang. Neighborhood preprocessing svm for large-scale data sets classification. In *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, volume 2, pages 245 –249, oct. 2008.
- [32] J.P. Chen, M.L. Lin, and H.H. Lee. Mineral dust emission estimated from satellite data. In *Proceedings of SPIE, the International Society for Optical*

- Engineering*, page 62990I. Society of Photo-Optical Instrumentation Engineers, 2006.
- [33] Pai-Hsuen Chen, Rong-En Fan, and Chih-Jen Lin. A study on smo-type decomposition methods for support vector machines. *Neural Networks, IEEE Transactions on*, 17(4):893 – 908, july 2006.
 - [34] Zhenyu Chen, Jianping Li, and Liwei Wei. Maker gene identification: a multiple kernel support vector machine approach. In *Bioinformatics and Biomedical Engineering, 2007. ICBBE 2007. The 1st International Conference on*, pages 276 –279, july 2007.
 - [35] V. Cherkassky, V.S. Cherkassky, and F. Mulier. *Learning from data: Concepts, theory, and methods*. Wiley-IEEE Press, 2007.
 - [36] W. Chu, S.S. Keerthi, and C.J. Ong. Bayesian support vector regression using a unified loss function. *Neural Networks, IEEE Transactions on*, 15(1):29–44, 2004.
 - [37] Wei Chu, Chong Jin Ong, and S.S. Keerthi. An improved conjugate gradient scheme to the solution of least squares svm. *Neural Networks, IEEE Transactions on*, 16(2):498 –501, march 2005.
 - [38] R. Collobert and S. Bengio. A gentle hessian for efficient gradient descent. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 5, pages V – 517–20 vol.5, may 2004.
 - [39] Ronan Collobert and Samy Bengio. Svmtorch: support vector machines for large-scale regression problems. *J. Mach. Learn. Res.*, 1:143–160, 2001.
 - [40] R.G. Congalton and K. Green. *Assessing the accuracy of remotely sensed data: principles and practices*. CRC, 2008.

- [41] R. Courant and D. Hilbert. *Methods of mathematical physics*. Interscience New York, 1966.
- [42] D. Coyle. Neural network based auto association and time-series prediction for biosignal processing in brain-computer interfaces. *Computational Intelligence Magazine, IEEE*, 4(4):47–59, november 2009.
- [43] N. Cristianini and B. Scholkopf. Support vector machines and kernel methods: the new generation of learning machines. *Ai Magazine*, 23(3):31, 2002.
- [44] A. Cutler, D.R. Cutler, and J.R. Stevens. Tree-based methods. *High-Dimensional Data Analysis in Cancer Research*, pages 1–19, 2009.
- [45] G.B. Dantzig. *Linear programming and extensions*. Princeton Univ Pr, 1998.
- [46] G.B. Dantzig and M.N. Thapa. *Linear programming: introduction*. Springer Verlag, 1997.
- [47] W.F. Darsow, B. Nguyen, and E.T. Olsen. Copulas and Markov processes. *Illinois Journal of Mathematics*, 36(4):600–642, 1992.
- [48] J.G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Optical Society of America, Journal, A: Optics and Image Science*, 2:1160–1169, 1985.
- [49] P. de Rivaz. *Complex wavelet based image analysis and synthesis*. PhD thesis, Ph. D. thesis, University of Cambridge, Cambridge, UK, 2000.
- [50] R. Debnath and H. Takahashi. Svm training: Second-order cone programming versus quadratic programming. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 1162 – 1168, july 2006.
- [51] J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Society for Industrial Mathematics, 1996.

- [52] A. Di Lillo, G. Motta, and J.A. Storer. Texture classification based on discriminative features extracted in the frequency domain. In *Image Processing, 2007. ICIIP 2007. IEEE International Conference on*, volume 2, pages II–53. IEEE, 2007.
- [53] Petros Drineas and Michael W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.
- [54] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik. Support vector regression machines. pages 155–161, 1996.
- [55] Hongle Du, Shaohua Teng, Xiufen Fu, Wei Zhang, and Yuanfang Pu. A cooperative intrusion detection system based on improved parallel svm. In *Pervasive Computing (JCPC), 2009 Joint Conferences on*, pages 515 –518, dec. 2009.
- [56] K. Duan, S.S. Keerthi, and A.N. Poo. Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing*, 51:41–59, 2003.
- [57] H. El-Askary, M. Kafatos, X. Liu, and T. El-Ghazawi. Introducing new approaches for dust storms detection using remote sensing technology. In *Proceedings of 2003 IEEE International Geoscience and Remote Sensing Symposium, 2003. IGARSS’03.*, volume 4, pages 2439–2441. IEEE, 2004.
- [58] S. Essid, G. Richard, and B. David. Musical instrument recognition by pairwise classification strategies. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(4):1401 –1412, july 2006.
- [59] D. W. Fanning. Bright modis images., Mar 2010.
- [60] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 31, 2004.

- [61] M. Ferecatu and N. Boujemaa. Interactive remote-sensing image retrieval using active relevance feedback. *Geoscience and Remote Sensing, IEEE Transactions on*, 45(4):818–826, april 2007.
- [62] M. C. Ferris, O. L. Mangasarian, and S. J. Wright. *Linear programming with MATLAB*. Society for Industrial and Applied Mathematics, 2007.
- [63] R.A. Fisher et al. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7:179–188, 1936.
- [64] M. Forina, R. Leardi, C. Armanino, and S. Lanteri. PARVUS: an extendable package of programs for data exploration, classification and correlation. *Institute of Pharmaceutical and Food Analysis Technologies, Genoa, Italy*, 1998.
- [65] A. Frank and A. Asuncion. UCI machine learning repository, 2010.
- [66] W.B. Gail. Remote sensing in the coming decade: the vision and the reality. *J Applied Remote Sensing*, 1:1–19, 2007.
- [67] A.R. Gallant and H. White. There exists a neural network that does not make avoidable mistakes. In *IEEE International Conference on Neural Networks, 1988.*, pages 657–664. IEEE, 2002.
- [68] Zhong Gao, Guanming Lu, and Daquan Gu. A novel hybrid system for large-scale chinese text classification problem. In *Frontier of Computer Science and Technology, 2008. FCST '08. Japan-China Joint Workshop on*, pages 121–124, dec. 2008.
- [69] I. Gokcen, D. Joachim, and J.R. Deller. Comparing optimal bounding ellipsoid and support vector machine active learning. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 172 – 175 Vol.1, aug. 2004.

- [70] Yun-Chao Gong and Chuan-Liang Chen. Semi-supervised method for gene expression data classification with gaussian fields and harmonic functions. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, dec. 2008.
- [71] R.P. Gorman and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural networks*, 1(1):75–89, 1988.
- [72] NASA GSFC. Warehouse inventory search tool (wist)., Mar 2010.
- [73] I. Hadzic and V. Kecman. Support vector machines trained by linear programming: theory and application in image compression and data classification. In *Neural Network Applications in Electrical Engineering, 2000. NEUREL 2000. Proceedings of the 5th Seminar on*, pages 18–23, 2000.
- [74] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: An update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
- [75] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 282–291. Springer, 2004.
- [76] X. Hao and J.J. Qu. Saharan dust storm detection using MODIS thermal infrared bands. *Journal of Applied Remote Sensing*, 1:013510, 2007.
- [77] P.E. Hart, R.O. Duda, and D.G. Stork. *Pattern classification*. Wiley, 2001.
- [78] S. S. Haykin. *Neural networks and learning machines*. Prentice Hall, 2009.
- [79] T. Hazan, A. Man, and A. Shashua. A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, june 2008.

- [80] M.R. Hestenes. Pseudoinversus and conjugate gradients. *Communications of the ACM*, 18(1):40–43, 1975.
- [81] C. Hildreth. A quadratic programming procedure. *Naval Research Logistics Quarterly*, 4(1):79–85, 1957.
- [82] D.W. Hillger and G.P. Ellrod. Detection of Important Atmospheric and Surface Features by Employing Principal Component Image Transformation of GOES Imagery. *Journal of Applied Meteorology*, 42:611–629, 2003.
- [83] D.H. Hong and C. Hwang. Interval regression analysis using quadratic loss support vector machine. *Fuzzy Systems, IEEE Transactions on*, 13(2):229–237, 2005.
- [84] LF Hoogerheide, JF Kaashoek, and HK Dijk. Neural network approximations to posterior densities: an analytical approach. *Econometric Institute Report*, 2010.
- [85] W.J. Hu and Q. Song. An accelerated decomposition algorithm for robust support vector machines. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 51(5):234 – 240, may 2004.
- [86] B. Huang, Z. Cai, Q. Gu, and C. Chen. Using Support Vector Regression for Classification. *Advanced Data Mining and Applications*, pages 581–588, 2008.
- [87] Fu Jie Huang and Y. LeCun. Large-scale learning with svm and convolutional for generic object categorization. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 284 – 291, june 2006.
- [88] P.J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

- [89] Z. Hui-ren and P. ZHENG. Method for Selecting Parameters of Least Squares Support Vector Machines Based on GA and Bootstrap [J]. *Journal of System Simulation*, 12, 2008.
- [90] D. Hush, P. Kelly, C. Scovel, and I. Steinwart. Qp algorithms with guaranteed accuracy and run time for support vector machines. *The Journal of Machine Learning Research*, 7:769, 2006.
- [91] New England ISO. Nepol zonal load data., Mar 2011.
- [92] P. H. Swain J. L. Kast and B. J. Davis. *ECHO User's Multiband Satellite Photography Using Digital Tech Guide*. LARS Publication, 083077, Purdue University.
- [93] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern recognition*, 24(12):1167–1186, 1991.
- [94] S. Janugani, V. Jayaram, SD Cabrera, JG Rosiles, TE Gill, and N.R. Rivera. *Directional analysis and filtering for dust storm detection in NOAA-AVHRR imagery*. University of Texas at El Paso, 2009.
- [95] J. Ji, F. Potra, RA Tapia, Y. Zhang, TX. Dept of Computational Rice University Houston, and Applied Mathematics. An interior-point method with polynomial complexity and superlinear convergence for linear complementarity problems. 1991.
- [96] Wenhan Jiang, Xiaofei Zhou, Hongchuan Hou, and Xinggang Lin. A new sampling-based svm for face recognition. In *Pattern Recognition, 2009. CCPR 2009. Chinese Conference on*, pages 1 –5, nov. 2009.
- [97] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pages 137–142, 1998.

- [98] T. Joachims. Making large scale svm learning practical. 1999.
- [99] Juan juan Gu, Liang Tao, and H.K. Kwan. Fast learning algorithms for new l2 svm based on active set iteration method. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 5, pages V-812 – V-815 Vol.5, may 2004.
- [100] B. Julesz. A theory of preattentive texture discrimination based on first-order statistics of textons. *Biological Cybernetics*, 41(2):131–138, 1981.
- [101] B. Julesz. Texton gradients: The texton theory revisited. *Biological Cybernetics*, 54(4):245–251, 1986.
- [102] J. Austin Kanellopoulos, G. G. Wilkinson. *Neurocomputation in Remote Sensing Data. Analysis*. Springer-Verlag, 1997.
- [103] A. Karsaz, H.R. Mashhadi, and M.M. Mirsalehi. Market clearing price and load forecasting using cooperative co-evolutionary approach. *International Journal of Electrical Power & Energy Systems*, 32(5):408–415, 2010.
- [104] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *Neural Networks, IEEE Transactions on*, 11(1):124 –136, January 2000.
- [105] N. Khazenie and TF Lee. Identification Of Aerosol Features Such As Smoke And Dust. In *NOAA-AVHRR Data Using Spatial Textures. International Geoscience and Remote Sensing Symposium*, pages 726–730, 1992.
- [106] N. Kingsbury. The dual-tree complex wavelet transform: a new efficient tool for image restoration and enhancement. In *Eusipco: European signal processing conference*, pages 319–322, 1998.

- [107] D. Kinzett, M. Zhang, and M. Johnston. Using numerical simplification to control bloat in genetic programming. *Simulated Evolution and Learning*, pages 493–502, 2008.
- [108] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the second international conference on knowledge discovery and data mining*, volume 7. Menlo Park, USA: AAAI Press, 1996.
- [109] K. Labusch, E. Barth, and T. Martinetz. Simple method for high-performance digit recognition based on sparse coding. *Neural Networks, IEEE Transactions on*, 19(11):1985–1989, nov. 2008.
- [110] Man Lan, Chew Lim Tan, Jian Su, and Yue Lu. Supervised and traditional term weighting methods for automatic text categorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):721–735, april 2009.
- [111] K.I. Laws. Rapid texture identification. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 238, pages 376–380, 1980.
- [112] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [113] J.A. Lee, T.E. Gill, K.R. Mulligan, M. Dominguez Acosta, and A.E. Perez. Land use/land cover and point sources of the 15 December 2003 dust storm in southwestern North America. *Geomorphology*, 105(1-2):18–27, 2009.
- [114] RC Levy, LA Remer, YJ Kaufman, D. Tanré, S. Mattoo, E. Vermote, and O. Dubovik. Revised Algorithm Theoretical Basis Document: MODIS Aerosol Products MOD, 2006.

- [115] Haizhou Li, Bin Ma, and Chin-Hui Lee. A vector space modeling approach to spoken language identification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(1):271 – 284, jan. 2007.
- [116] Xuchun Li, Yan Zhu, and E. Sung. Sequential bootstrapped support vector machines - a svm accelerator. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, pages 1437 – 1442 vol. 3, july-4 aug. 2005.
- [117] Xun Liang, Rong-Chang Chen, and Xinyu Guo. Pruning support vector machines without altering performances. *Neural Networks, IEEE Transactions on*, 19(10):1792 – 1803, oct. 2008.
- [118] Chih-Jen Lin. Asymptotic convergence of an smo algorithm without any assumptions. *Neural Networks, IEEE Transactions on*, 13(1):248 – 250, jan 2002.
- [119] Y.C. Lin, T. Chang, and C.C.J. Kuo. Hierarchical texture segmentation with wavelet packet transform and adaptive smoothing. In *Proceedings of SPIE*, volume 2308, page 1954, 1994.
- [120] Zhenbing Liu, J.G. Liu, Chao Pan, and Guoyou Wang. A novel geometric approach to binary classification based on scaled convex hulls. *Neural Networks, IEEE Transactions on*, 20(7):1215 – 1220, 2009.
- [121] S.A. Loomer. Academic research opportunities at the National Geospatial-Intelligence Agency (NGA). In *Proceedings of SPIE*, volume 6233, page 623324, 2006.
- [122] Bao-Liang Lu, Kai-An Wang, and Yi-Min Wen. Comparison of parallel and cascade methods for training support vector machines on large-scale problems. In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 5, pages 3056 – 3061 vol.5, aug. 2004.

- [123] Z. Lu, J. Sun, and K. R. Butts. Linear programming support vector regression with wavelet kernel: A new approach to nonlinear dynamical systems identification. *Mathematics and Computers in Simulation*, 79(7):2051–2063, 2009.
- [124] A.B. Mahler, K. Thome, D. Yin, and W.A. Sprigg. Dust transport model validation using satellite-and ground-based methods in the southwestern United States. In *Proc. of SPIE Vol*, volume 6299, pages 62990L–1.
- [125] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, 2008.
- [126] O.L. Mangasarian and D.R. Musicant. Successive overrelaxation for support vector machines. *Neural Networks, IEEE Transactions on*, 10(5):1032–1037, sep 1999.
- [127] O.L. Mangasarian and D.R. Musicant. Large scale kernel regression via linear programming. *Machine Learning*, 46(1):255–269, 2002.
- [128] K.J. McGarry, S. Wermter, and J. MacIntyre. Knowledge extraction from radial basis function networks and multilayer perceptrons. In *Neural Networks, 1999. IJCNN’99. International Joint Conference on*, volume 4, pages 2494–2497. IEEE, 2002.
- [129] J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.
- [130] D.B.A. Mezghani, S.Z. Boujelbene, and N. Ellouze. Evaluation of SVM Kernels and Conventional Machine Learning Algorithms for Speaker Identification.

- [131] SD Miller. A consolidated technique for enhancing desert dust storms with MODIS. *Geophys. Res. Lett*, 30(20):2071, 2003.
- [132] K. Müller, A. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. *Artificial Neural Networks-ICANN'97*, pages 999–1004, 1997.
- [133] J. Neumann. A model of general economic equilibrium. *The Review of Economic Studies*, 13(1):1–9, 1945.
- [134] K.S. Ni, S. Kumar, N. Vasconcelos, and T.Q. Nguyen. Single image superresolution based on support vector regression. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 2, pages II –II, may 2006.
- [135] K. Nishida and T. Kurita. Ransac-svm for large-scale datasets. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.
- [136] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.
- [137] D.J. Novlan and TE Gill. A synoptic climatology of blowing dust events in El Paso, Texas from 1932-2005. In *16th Conference on Applied Climatology*, 2007.
- [138] E. Osuna and O. De Castro. Convex hull in feature space for support vector machines. *Advances in Artificial Intelligence-IBERAMIA 2002*, pages 411–419, 2002.
- [139] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 276 –285, sep 1997.

- [140] N. Oza, J.P. Castle, and J. Stutz. Classification of aeronautics system health and safety documents. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 39(6):670 –680, nov. 2009.
- [141] M. Papadonikolakis and C.-S. Bouganis. Efficient fpga mapping of gilbert algorithm for svm training on large-scale classification problems. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 385 –390, sept. 2008.
- [142] C. Papageorgiou, F. Girosi, and T. Poggio. Sparse correlation kernel reconstruction. In *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on*, volume 3, pages 1633 –1636 vol.3, mar 1999.
- [143] S.I. Park, MJT Smith, and RM Mersereau. A new directional filter bank for image analysis and classification. In *Acoustics, Speech, and Signal Processing, 1999. ICASSP'99. Proceedings., 1999 IEEE International Conference on*, volume 3, pages 1417–1420. IEEE.
- [144] Xinjun Peng. Tsvr: An efficient twin support vector machine for regression. *Neural Networks*, 23(3):365 – 372, 2010.
- [145] D.A. Peterson and M.H. Thaut. Model and feature selection in microarray classification. In *Computational Intelligence in Bioinformatics and Computational Biology, 2004. CIBCB '04. Proceedings of the 2004 IEEE Symposium on*, pages 56 – 60, oct. 2004.
- [146] Tho Hoan Pham, Dang Hung Tran, Tu Bao Ho, and K. Satou. Prediction of histone modifications in dna sequences. In *Bioinformatics and Bioengineering, 2007. BIBE 2007. Proceedings of the 7th IEEE International Conference on*, pages 959 –966, oct. 2007.

- [147] J.C. Platt. Fast training of support vector machines using sequential minimal optimization. pages 185–208, 1999.
- [148] J.C. Platt. Using analytic qp and sparseness to speed training of support vector machines. *Advances in Neural Information Processing Systems*, pages 557–563, 1999.
- [149] R. Porter and N. Canagarajah. A robust automatic clustering scheme for image segmentation using wavelets. *Image Processing, IEEE Transactions on*, 5(4):662–665, 1996.
- [150] J.M. Prospero, P. Ginoux, O. Torres, S.E. Nicholson, and T.E. Gill. Environmental characterization of global sources of atmospheric soil dust identified with the Nimbus 7 Total Ozone Mapping Spectrometer (TOMS) absorbing aerosol product. *Reviews of Geophysics*, 40(1):1002, 2002.
- [151] Yali Qi, Yeli Li, and Liuping Feng. Optimization clustersvm using improved nonlinear kernel. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 970 –974, july 2008.
- [152] S. Ramakrishnan and S. Selvan. Image texture classification using wavelet based curve fitting and probabilistic neural network. *International Journal of Imaging Systems and Technology*, 17(4):266–275, 2007.
- [153] T. Randen and J.H. Husoy. Filtering for texture classification: A comparative study. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(4):291–310, 1999.
- [154] J.A. Richards and X. Jia. *Remote sensing digital image analysis: an introduction*. Springer Verlag, 2006.

- [155] R.M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology, 2002.
- [156] B.D. Ripley. *Pattern recognition and neural networks*. Cambridge Univ Pr, 2008.
- [157] P. Rivas-Perea. Southwestern u.s. and northwestern mexico dust storm modeling through moderate resolution imaging spectroradiometer data: A machine learning perspective. *Technical Report: NASA/UMBC/GEST Graduate Student Summer Program 2009*, Aug 2009.
- [158] P. Rivas-Perea and J.G. Rosiles. A Probabilistic Model for Stratospheric Soil-Independent Dust Aerosol Detection. In *Digital Image Processing and Analysis*. Optical Society of America, 2010.
- [159] N.I. Rivera Rivera. Detection and characterization of dust source areas in the Chihuahuan Desert, southwestern North America. Master’s thesis, The University of Texas at El Paso, 2006.
- [160] NI Rivera Rivera, KA Gebhart, TE Gill, JL Hand, DJ Novlan, and RM Fitzgerald. Analysis of air transport patterns bringing dust storms to El Paso, Texas. In *Symposium on Urban High Impact Weather, AMS Annual Meeting, Phoenix*, 2009.
- [161] J. G. Rosiles and M. J. T. Smith. chapter 25. CRC Press, 2009.
- [162] JG Rosiles and MJT Smith. Texture segmentation using a biorthogonal directional decomposition. *SCI 2000*, 2000.
- [163] JG Rosiles and MJT Smith. A low complexity undecimated directional image decomposition. In *Proceedings of the IEEE International Conference on Image Processing (ICIP’03*, volume 1, pages 1049–1052, 2003.

- [164] J.G. Rosiles and M.J.T. Smith. Multichannel texture segmentation using bam-berger pyramids. *Journal on Image and Video Processing*, 2009:9, 2009.
- [165] J.G.G. Rosiles. *Image and texture analysis using biorthogonal angular filter banks*. PhD thesis, Citeseer, 2004.
- [166] P. Rullo, V.L. Policicchio, C. Cumbo, and S. Iiritano. Olex: Effective rule learning for text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 21(8):1118–1132, aug. 2009.
- [167] L. San-chao, L. Qinhuo, G. Maofang, and C. Liangfu. Detection of Dust Storms by Using Daytime and Nighttime Multi-spectral MODIS Images. In *IEEE International Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006.*, pages 294–296. IEEE, 2007.
- [168] A. Sanchez and V. David. Advanced support vector machines and kernel methods. *Neurocomputing*, 55(1-2):5–20, 2003.
- [169] N. Sapankevych and R. Sankar. Time series prediction using support vector machines: A survey. *Computational Intelligence Magazine, IEEE*, 4(2):24–38, may 2009.
- [170] B. Scholkopf and A. J. Smola. *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. The MIT Press, 2002.
- [171] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, Cambridge, UK and New York, NY, 2004.
- [172] M. Shen, L. Lin, J. Chen, and C. Q. Chang. A prediction approach for multichannel eeg signals modeling using local wavelet svm. *Instrumentation and Measurement, IEEE Transactions on*, 59(5):1485–1492, may 2010.

- [173] M. Shen, L. Lin, J. Chen, and C. Q. Chang. A prediction approach for multichannel eeg signals modeling using local wavelet svm. *Instrumentation and Measurement, IEEE Transactions on*, 59(5):1485–1492, may 2010.
- [174] S.K. Shevade, S.S. Keerthi, C. Bhattacharyya, and K.R.K. Murthy. Improvements to the smo algorithm for svm regression. *Neural Networks, IEEE Transactions on*, 11(5):1188–1193, sep 2000.
- [175] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multiscale transforms. *Information Theory, IEEE Transactions on*, 38(2):587–607, 1992.
- [176] K. Skretting and J.H. Husøy. Texture classification using sparse frame-based representations. *EURASIP journal on applied signal processing*, 2006:102–102, 2006.
- [177] A. Smola, B. Scholkopf, and G. Ratsch. Linear programs for automatic accuracy control in regression. In *Artificial Neural Networks, 1999. ICANN 99. Ninth International Conference on (Conf. Publ. No. 470)*, 1999.
- [178] A. J. Smola and B. Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.
- [179] D.F. Specht. Probabilistic neural networks for classification, mapping, or associative memory. In *IEEE International Conference on Neural Networks, 1988.*, pages 525–532. IEEE, 2002.
- [180] S. Sra. Efficient large scale linear programming support vector machines. *Machine Learning: ECML 2006*, pages 767–774, 2006.
- [181] Yuchun Tang, Yuanchen He, and S. Krasser. Highly scalable svm modeling with random granulation for spam sender detection. In *Machine Learning and*

- Applications, 2008. ICMLA '08. Seventh International Conference on*, pages 659–664, dec. 2008.
- [182] Dacheng Tao, Xiaoou Tang, Xuelong Li, and Yong Rui. Direct kernel biased discriminant analysis: a new content-based image retrieval relevance feedback algorithm. *Multimedia, IEEE Transactions on*, 8(4):716–727, aug. 2006.
 - [183] R.A. Tapia, Y. Zhang, and Y. Ye. On the convergence of the iteration sequence in primal-dual interior-point methods. *Mathematical Programming*, 68(1):141–154, 1995.
 - [184] M.C.M. Team, S. Ackerman, K. Strabala, P. Menzel, R. Frey, C. Moeller, L. Gumley, B. Baum, C. Schaaf, and G. Riggs. Discriminating Clear-Sky From Cloud With Modis Algorithm Theoretical Basis Document (Mod35). 1997.
 - [185] M. Tohme and R. Lengelle. F-svr: A new learning algorithm for support vector regression. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 2005–2008, 31 2008–april 4 2008.
 - [186] M. Topi, P. Matti, and O. Timo. Texture classification by multi-predicate local binary pattern operators. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 3, pages 939–942. IEEE, 2000.
 - [187] Y. Torii and S. Abe. Decomposition techniques for training linear programming support vector machines. *Neurocomputing*, 72(4-6):973–984, 2009.
 - [188] B. Tso and P.M. Mather. *Classification methods for remotely sensed data*. CRC, 2009.
 - [189] M. Unser. Texture classification and segmentation using wavelet frames. *Image Processing, IEEE Transactions on*, 4(11):1549–1560, 1995.

- [190] M. Unser and M. Eden. Nonlinear operators for improving texture segmentation based on features extracted by spatial filtering. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(4):804–815, 1990.
- [191] P.P. Vaidyanathan. *Multirate systems and filter banks*. Pearson Education India, 1993.
- [192] G. Valentini, M. Muselli, and F. Ruffino. Bagged ensembles of support vector machines for gene expression data analysis. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1844 – 1849 vol.3, july 2003.
- [193] J. Valenzuela and M. Mazumdar. On the computation of the probability distribution of the spot market price in a deregulated electricity market. In *Power Industry Computer Applications, 2001. PICA 2001. Innovative Computing for Power - Electric Energy Meets the Market. 22nd IEEE Power Engineering Society International Conference on*, 2001.
- [194] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.
- [195] LJP Van der Maaten, EO Postma, and HJ Van den Herik. Dimensionality reduction: A comparative review. *Preprint*, 2007.
- [196] T. Van Gestel, J.A.K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewalle. Financial time series prediction using least squares support vector machines within the evidence framework. *Neural Networks, IEEE Transactions on*, 12(4):809 –821, jul 2001.
- [197] V. Vapnik, S. Golowich, and A. Smola. *Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing*. Neural Information Processing Systems. MIT Press, Cambridge, MA, 1997.

- [198] V. N. Vapnik. *The nature of statistical learning theory*. Springer, New York, NY, 1995.
- [199] Haifeng Wang and Dejin Hu. Comparison of svm and ls-svm for regression. In *Neural Networks and Brain, 2005. ICNN B '05. International Conference on*, volume 1, pages 279–283, oct. 2005.
- [200] J. Wang, H. Lu, KN Plataniotis, and J. Lu. Gaussian kernel optimization for pattern classification. *Pattern Recognition*, 42(7):1237–1247, 2009.
- [201] L. Wang and SpringerLink (Online service). *Support Vector Machines: theory and applications*. Springer-Verlag GmbH., 2005.
- [202] Liwei Wei, Zhenyu Chen, Jianping Li, and Weixuan Xu. Sparse and robust least squares support vector machine: A linear programming formulation. In *Grey Systems and Intelligent Services, 2007. GSIS 2007. IEEE International Conference on*, pages 1134–1138, nov. 2007.
- [203] S.J. Wright. *Primal-dual interior-point methods*. Society for Industrial Mathematics, 1997.
- [204] Kui Wu and Kim-Hui Yap. Fuzzy svm for content-based image retrieval: a pseudo-label support vector machine framework. *Computational Intelligence Magazine, IEEE*, 1(2):10–16, may 2006.
- [205] Tu Xu. A new sphere-structure multi-class classifier. In *Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on*, pages 520–525, may 2009.
- [206] Zenglin Xu, Kaizhu Huang, Jianke Zhu, Irwin King, and Michael R. Lyu. A novel kernel-based maximum a posteriori classification method. *Neural Networks*, 22(7):977–987, 2009.

- [207] Zongben Xu, Mingwei Dai, and Deyu Meng. Fast and efficient strategies for model selection of gaussian support vector machine. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(5):1292–1307, oct. 2009.
- [208] Xu Yan-zi and Qin Hua. A new optimization method of large-scale svms based on kernel distance clustering. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–4, dec. 2009.
- [209] Jing Yang, Zhong-Wei Li, and Jian-Pei Zhang. A training algorithm of incremental support vector machine with recombining method [support read support]. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 7, pages 4285–4288 Vol. 7, aug. 2005.
- [210] Liu Yongping and Qiu Dongtao. Accelerating svm on large scale regression problem. In *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 65–68 vol.1, oct. 2002.
- [211] Yulai Yuan, Yongwei Wu, Guangwen Yang, and Weimin Zheng. Adaptive hybrid model for long term load prediction in computational grid. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 340–347, May 2008.
- [212] Xiangyan Zeng and Xue wen Chen. Smo-based pruning methods for sparse least squares support vector machines. *Neural Networks, IEEE Transactions on*, 16(6):1541–1546, nov. 2005.
- [213] Jian-Pei Zhang, Zhong-Wei Li, and Jing Yang. A parallel svm training algorithm on large-scale classification problems. In *Machine Learning and Cyber-*

- netics, 2005. Proceedings of 2005 International Conference on*, volume 3, pages 1637–1641 Vol. 3, aug. 2005.
- [214] Li Zhang and Weida Zhou. On the sparseness of 1-norm support vector machines. *Neural Networks*, 23(3):373–385, 2010.
 - [215] M. Zhang and L. Fu. Unbiased least squares support vector machine with polynomial kernel. In *Signal Processing, 2006 8th International Conference on*, volume 3, 16-20 2006.
 - [216] Tianyou Zhang, Xiuju Fu, R.S.M. Goh, Chee Keong Kwoh, and G.K.K. Lee. A ga-svm feature selection model based on high performance computing techniques. In *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2653–2658, oct. 2009.
 - [217] Xue-Qin Zhang and Chun-Hua Gu. Ch-svm based network anomaly detection. In *Machine Learning and Cybernetics, 2007 International Conference on*, volume 6, pages 3261–3266, aug. 2007.
 - [218] Y. Zhang and RA Tapia. Superlinear and quadratic convergence of primal-dual interior-point methods for linear programming revisited. *Journal of Optimization Theory and Applications*, 73(2):229–242, 1992.
 - [219] Y. Zhang, R.A. Tapia, F. Potra, and TX. Dept of Mathematical Sciences Rice University Houston. On the superlinear convergence of interior point algorithms for a general class of problems. 1991.
 - [220] Jun Hua Zhao, Zhao Yang Dong, Zhao Xu, and Kit Po Wong. A statistical approach for interval forecasting of the electricity price. *Power Systems, IEEE Transactions on*, 23(2):267–276, may 2008.
 - [221] Xiaofei Zhou, Wenhan Jiang, Yingjie Tian, and Yong Shi. Kernel subclass convex hull sample selection method for svm on face recognition. *Neurocomputing*,

73(10-12):2234 – 2246, 2010. Subspace Learning / Selected papers from the European Symposium on Time Series Prediction.

- [222] Dong Zhuang, Benyu Zhang, Qiang Yang, Jim Yan, Zheng Chen, and Ying Chen. Efficient text classification by weighted proximal svm. In *Data Mining, Fifth IEEE International Conference on*, page 8 pp., nov. 2005.
- [223] G. Zoutendijk. *Methods of feasible directions: a study in linear and nonlinear programming*. Elsevier Pub. Co., 1960.

Appendix A

Background in Support Vector Machines

A.1 Support Vector Machines Fundamental Principle

The seminal principle of Support Vector Machines (SVM), shown in Figure 1.3, can be explained in terms of a given training set $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$, where $d_i \in \{1, -1\}$. The goal is to find \mathbf{w} 's and b that form the hyperplane $\mathbf{w}^T \mathbf{x}_i + b = 0$ with maximum margin between the parallel hyperplanes $\mathbf{w}^T \mathbf{x}_i + b = -1$ and $\mathbf{w}^T \mathbf{x}_i + b = 1$. If we use a geometrical analysis we can trivially derive that such margin is given by $\frac{2}{\|\mathbf{w}\|_2}$ as illustrated in Figure 1.3. Therefore, in order to maximize the margin, we need to minimize $\|\mathbf{w}\|_2$, and so the SVM requires the solution of the following constrained optimization problem in its primal form:

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & \left\{ \begin{array}{l} \mathbf{w}^T \mathbf{x}_i + b = d_i \\ \text{for } i = 1, 2, \dots, N. \end{array} \right. \end{aligned} \tag{A.1}$$

Those input patterns that satisfy the equality condition

$$\mathbf{w}^T \mathbf{x}_i + b = d_i. \tag{A.2}$$

are called “support vectors.” And it must be clear from Figure 1.3 that the support vectors are the only data points relevant (necessary and sufficient) to represent the separating hyperplane, and all the remaining data can be discarded as a result.

The previous optimization problem is a generic SVM; however, the SVM is designed for two-classes, and so the problem becomes

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & \begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq 1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1 \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned} \tag{A.3}$$

where the restrictions can be formulated as a function of d_i as follows

$$d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \tag{A.4}$$

and the primal problem finally becomes

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 \\ \text{s.t.} \quad & \begin{cases} d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned} \tag{A.5}$$

The dual problem can be formulated using the Lagrange multipliers method, where a Lagrangian function is defined as:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|_2^2 - \sum_{i=1}^N \alpha_i (d_i(\mathbf{w}^T \mathbf{x}_i + b) - 1) \tag{A.6a}$$

and the associated stationary points are defined as

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i = \mathbf{0}, \quad (\text{A.6b})$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = - \sum_{i=1}^N \alpha_i d_i = 0, \quad (\text{A.6c})$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = -d_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0. \quad (\text{A.6d})$$

Then, the dual problem using the method of Lagrange multipliers is stated as

$$\begin{aligned} & \max_{\mathbf{w}, b, \boldsymbol{\alpha}} \quad L(\mathbf{w}, b, \boldsymbol{\alpha}) \\ & \text{s.t.} \quad \begin{cases} \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{0} \\ \frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = 0 \\ \boldsymbol{\alpha} \geq \mathbf{0} \end{cases} \end{aligned} \quad (\text{A.7})$$

and the expanded problem is

$$\begin{aligned} & \max_{\mathbf{w}, b, \boldsymbol{\alpha}} \quad \frac{1}{2} \|\mathbf{w}\|_2^2 - \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i - b \sum_{i=1}^N \alpha_i d_i + \sum_{i=1}^N \alpha_i \\ & \text{s.t.} \quad \begin{cases} \mathbf{w} - \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i = \mathbf{0} \\ - \sum_{i=1}^N \alpha_i d_i = 0 \\ \boldsymbol{\alpha} \geq \mathbf{0} \end{cases} \\ & \text{for} \quad i = 1, 2, \dots, N. \end{aligned} \quad (\text{A.8})$$

However, it can be noticed from (A.6b) that we can solve for \mathbf{w} as

$$\mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{x}_i \quad (\text{A.9})$$

and so the term $\|\mathbf{w}\|_2^2$ can be denoted as

$$\|\mathbf{w}\|_2^2 = \mathbf{w}^T \mathbf{w} = \sum_{i=1}^N \alpha_i d_i \mathbf{w}^T \mathbf{x}_i = \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \quad (\text{A.10})$$

which can be plugged in the Lagrangian function and associate similar terms, and also we can remove the first constraint. Also, the term $-b \sum_{i=1}^N \alpha_i d_i$ can be removed since it becomes zero by the definition (A.6c); and the well known dual problem becomes [78]:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N \alpha_i d_i = 0 \\ \boldsymbol{\alpha} \geq \mathbf{0} \end{cases} \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned} \quad (\text{A.11})$$

A.1.1 Soft-Margin Support Vector Machines

To this point, the primal and dual constrained optimization problems are designed under the following assumptions: (i) the input patterns are linearly separable and (ii) the optimal hyperplane exists. However, most pattern recognition problems do not fit into the linearly separable model. That is, some problems are linearly non-separable because the non-linear manifold of the class input-space, or the hyperplane that minimizes the function of the primal, does not exist (*i.e.*, the solution is unfeasible), most likely because the problem becomes poorly posed in terms of the training data set. An example of this situation is shown in Figure A.1, a case where outliers may arise for different reasons such as measurement errors, acquisition errors, floating point representation errors, etc. Therefore, a new set of nonnegative scalar variables, $\{\xi_i\}_{i=1}^N$, is introduced to avoid an unfeasibility in the primal and dual problem formulations. These variables are known as “slack variables.”

Considering the slack variables $\{\xi_i\}_{i=1}^N$ in the problem (A.1) and using (A.4), we

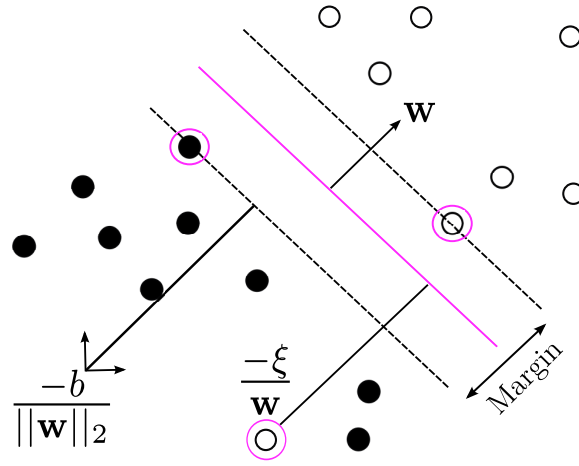


Figure A.1: Linearly non-separable data points, due to outliers in the data. Here the slack variables $\{\xi_i\}_{i=1}^N$ help in solving the problem in the presence of outliers that otherwise produce an unfeasibility.

can define the optimization problem for the linearly non-separable case as follows

$$\begin{aligned}
 & \min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N \xi_i & (\text{A.12}) \\
 & \text{s.t.} \quad \begin{cases} d_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \\ \xi \geq \mathbf{0} \end{cases} \\
 & \text{for} \quad i = 1, 2, \dots, N.
 \end{aligned}$$

where C is a parameter empirically specified by the classification scheme designer. The parameter C controls the trade-off between the complexity of the SVM and the number of points that can't be linearly separated by the hyperplane. This parameter is also known as the reciprocal of the “regularization” parameter. Parameter C should be large if the designer has high confidence in the quality of the training samples; however, parameter C should be small if the training samples are considered noisy. Empirically it suggested by the community to test exponentially growing C sequences good parameter identification *e.g.*, $C \in \{2^{-5}, 2^{-3}, \dots, 2^{13}, 2^{15}\}$.

Following the method of Lagrange multipliers, one can also formulate the dual

problem for the linearly non-separable patterns case by finding the Lagrange multipliers $\{\alpha_i\}_{i=1}^N$ in the problem

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad & \left\{ \begin{array}{l} \sum_{i=1}^N \alpha_i d_i = 0 \\ \boldsymbol{\alpha} \geq \mathbf{0} \\ \boldsymbol{\alpha} \leq C \mathbf{1} \end{array} \right. \\ \text{for} \quad & i = 1, 2, \dots, N. \end{aligned} \tag{A.13}$$

The relaxed SVM formulation in (A.12)-(A.13) is known as “Soft-Margin” SVM. These models solve linearly separable problems in the presence of outliers.

A.1.2 Kernel Expansion and Kernel Functions

Typically, real-life problems are non-linear, and as we try to use SVMs to model such problems, we find that the problems (A.12)-(A.13) only solve linearly or linearly non-separable problems. Consequently, the second key idea behind SVMs is an input pattern mapping (input space) to a higher dimensional space (kernel-induced feature space) where the data points $\mathbf{x}_i \in \mathcal{X}$ are more likely to be linearly separable. This is achieved by using a kernel function $k(\cdot, \cdot)$ mapping in the following way:

$$\mathbf{x}_i^T \mathbf{x}_j = k(\mathbf{x}_i, \mathbf{x}_j) = \phi^T(\mathbf{x}_i) \phi(\mathbf{x}_j), \tag{A.14}$$

$$\mathbf{x}_i = k(\mathbf{x}_i, \cdot) = \phi(\mathbf{x}_i), \tag{A.15}$$

where the map

$$\phi : \mathcal{X} \mapsto \mathcal{H} \tag{A.16}$$

is known as feature map from the data space \mathcal{X} into the feature space \mathcal{H} . The feature space is assumed to be a Hilbert space of real valued functions defined on \mathcal{X} (more

information in Appendix D). In the case of SVMs the data space $\mathbf{x}_i \in \mathcal{X} \in \mathbb{R}^{N \times M}$. The picture in Figure 1.4 illustrates the particular example where the feature map $\phi(x_1, x_2) : \mathbb{R}^2 \mapsto \mathbb{R}^3$.

Mercer's theorem (see Theorem D.1 in Appendix D) [41, 129] provides a useful tool known as *kernel expansion* that allows us to define the hyperplane as linear combinations with parameters α_i as follows:

$$\mathbf{w}^T \phi(\mathbf{x}) = \sum_{i=1}^N \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b, \quad (\text{A.17})$$

where α, b can take on any real value, and $k(\cdot, \cdot)$ is a valid kernel function. Mercer's theorem also explains the properties a valid kernel function $k(\cdot, \cdot)$ must follow. Mercer shows that it is sufficient for $k(\mathbf{x}_j, \mathbf{x})$ to be continuous, symmetric and positive definite. In our case if $\mathcal{X} \in \mathbb{R}^{N \times M}$ uses a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}^{N \times N}$ that

1. $k(\mathbf{x}_j, \mathbf{x}_i)$ is continuous,
2. $k(\mathbf{x}_j, \mathbf{x}_i) = k(\mathbf{x}_i, \mathbf{x}_j)$, is symmetric,
3. $k(\mathbf{x}_j, \mathbf{x}_i)$ is P.D. (positive definite),

then such function is said to be a *Mercer kernel*.

The typical kernel functions are as follows:

$$\text{Polynomial} : (\mathbf{x}_i^T \mathbf{x}_j + 1)^p, \quad (\text{A.18})$$

$$\text{Radial} : e^{-\frac{1}{2\sigma^2} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2}, \quad (\text{A.19})$$

$$\text{Sigmoidal} : \tanh(\kappa_1 \mathbf{x}_i^T \mathbf{x}_j + \kappa_2). \quad (\text{A.20})$$

The kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ may also be referred as the ij -th element of the symmetric $N \times N$ matrix

$$\mathbf{K} = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^N. \quad (\text{A.21})$$

The matrix \mathbf{K} is called “kernel matrix.” It is positive definite since it satisfies the condition $\mathbf{a}^T \mathbf{K} \mathbf{a} \geq 0$ for any real valued vector \mathbf{a} of dimension compatible with \mathbf{K} . The matrix notation of \mathbf{K} is specially useful when posing the optimization problem in matrix-vector form.

In summary, a kernel function allows us to map an input space into a kernel-induced feature space of higher dimension where the mapped input data is (expected to be) linearly separable with an appropriate selection of a valid kernel function.

Therefore, the usage of kernel functions represent a breakthrough for SVMs since they provide the means to solve non-linearly (non-)separable problems. As a result we can just modify the constraint in the SVM primal problem (A.12) to be as follows:

$$\begin{aligned} \min_{\mathbf{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^N \xi_i \\ \text{s.t.} \quad & \begin{cases} d_i(\mathbf{w}^T k(\mathbf{x}_i, \cdot) + b) \geq 1 - \xi_i \\ \boldsymbol{\xi} \geq \mathbf{0} \end{cases} \\ \text{for} \quad & i, j = 1, 2, \dots, N. \end{aligned} \tag{A.22}$$

and in the case of the dual formulation (A.13), the objective function is modified to be as follows

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \begin{cases} \sum_{i=1}^N \alpha_i d_i = 0 \\ \boldsymbol{\alpha} \geq \mathbf{0} \\ \boldsymbol{\alpha} \leq C \mathbf{1} \end{cases} \\ \text{for} \quad & i, j = 1, 2, \dots, N. \end{aligned} \tag{A.23}$$

The example on Figure A.2 shows three different views on the same data points: Figure A.2a, a linear separation does not work well since a the optimal margin requires misclassifying one point; Figure A.2b, a better separation is allowed by using a kernel function in the input space, which corresponds to a linear function in

the feature space; Figure A.2c, the input space and the kernel-induced feature space are related by the kernel function.

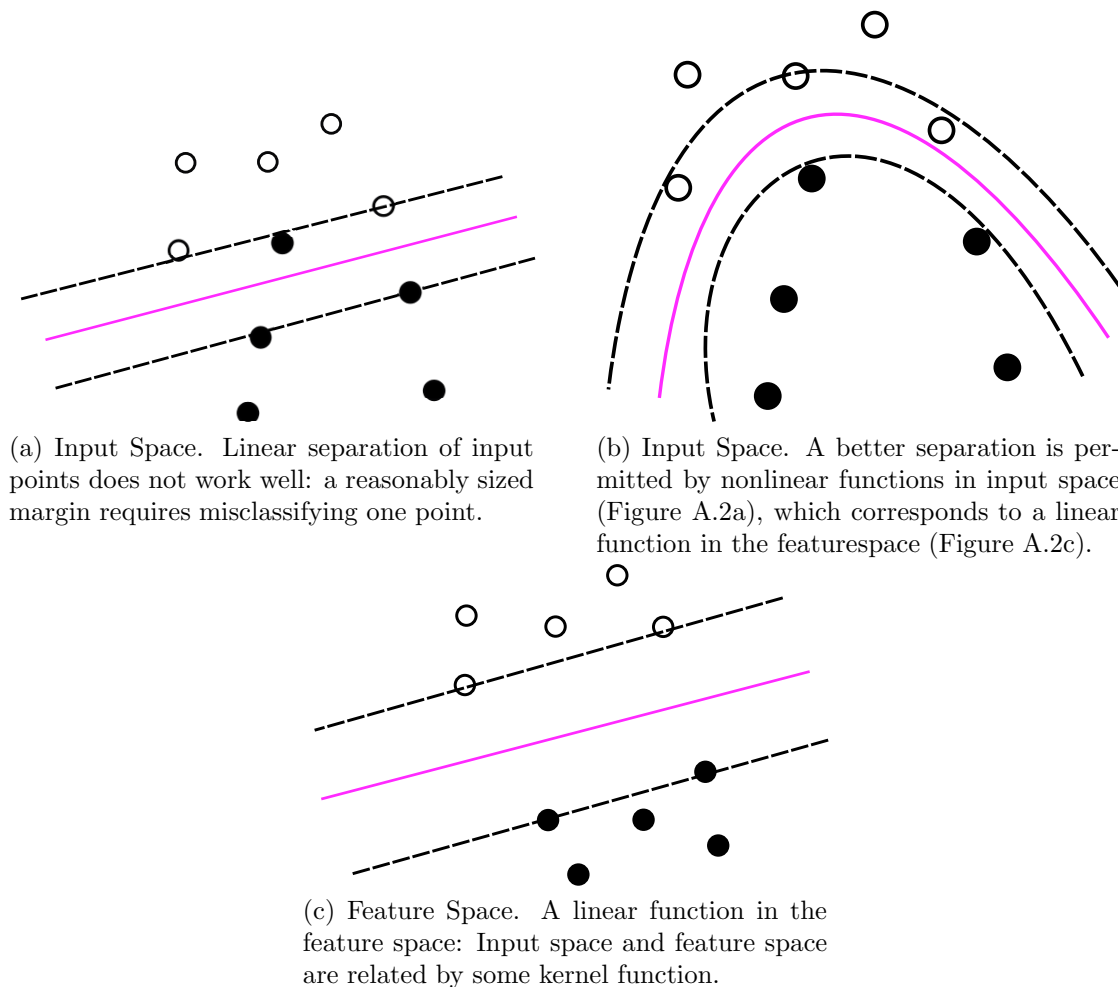


Figure A.2: Three different views of an arbitrary two-class classification problem.

Figure A.3 shows that using valid kernel function, the linearly non-separable data (in the input space) can be mapped to linearly separable in a higher dimensional space (the kernel-induced feature space).

A.1.3 Toy Example: XOR Problem

To actually show the solution procedure and elucidate the idea of SVMs, we consider the case of the XOR (Exclusive OR) problem, following [78]. The problem is

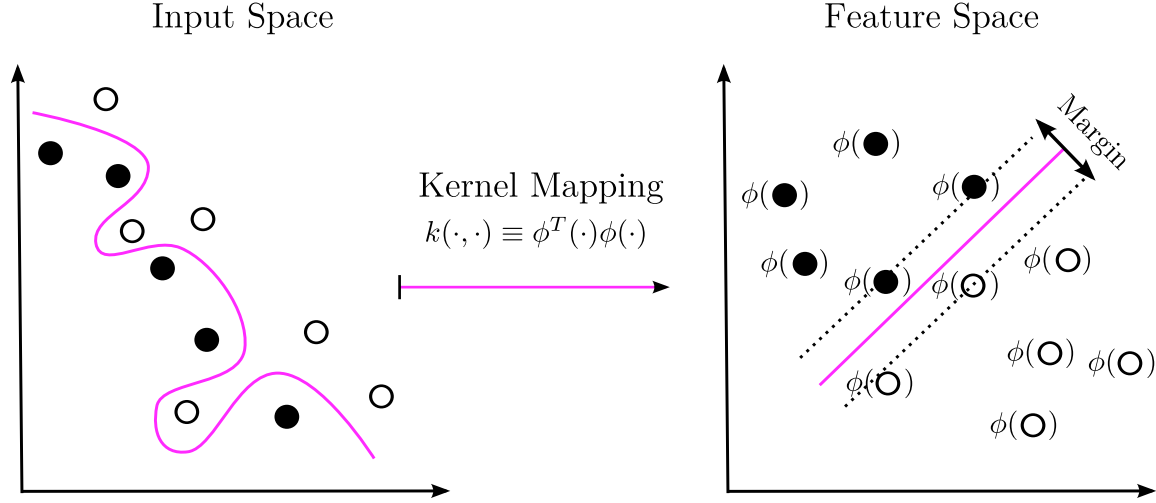


Figure A.3: With a suitable kernel function, the linear-inseparable data (in input space) can be transferred to linear separable in a higher dimensional space (in feature space).

summarized in Table A.1 that shows the inputs and the desired outputs.

Table A.1: Summary of XOR problem characterization.

Input vector \mathbf{x}	Desired response d
$(+1, +1)$	-1
$(+1, -1)$	$+1$
$(-1, +1)$	$+1$
$(-1, -1)$	-1

For this problem, the kernel proposed in [35] is a polynomial in the form of (A.18) for $p = 2$ as follows:

$$k(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}^T \mathbf{x}_i)^2, \quad (\text{A.24})$$

where $\mathbf{x} = [x_1, x_2]^T$ and $\mathbf{x}_i = [x_{i1}, x_{i2}]^T$. Thus, the kernel function can be expressed as

$$k(\mathbf{x}, \mathbf{x}_i) = 1 + x_1^2 x_{i1}^2 + 2x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2x_1 x_{i1} + 2x_2 x_{i2}. \quad (\text{A.25})$$

Following the kernel matrix definition in (A.21) we obtain

$$\mathbf{K} = \begin{pmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{pmatrix}.$$

The objective function for the dual non-linear SVM defined in (A.23) produces the following objective function:

$$\begin{aligned} \min_{\alpha} \quad & \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2} (9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 - 2\alpha_1\alpha_4 \\ & + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2), \end{aligned}$$

which need to be optimized with respect to the Lagrange multipliers as defined by the system of equations

$$\begin{aligned} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 &= 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 &= 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 &= 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 &= 1 \end{aligned}$$

from where it follows that the optimum values for the Lagrange multipliers are

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \frac{1}{8},$$

which implies that the set of input vectors $\{\mathbf{x}_i\}_{i=1}^4$ are support vectors. This solution yields that the optimum value for the objective function is $\frac{1}{4}$.

Using the duality theorem, the solution found also solves the primal problem in

(A.1) using the kernel trick, and we can rewrite

$$\frac{1}{2}||\mathbf{w}||_2^2 = \frac{1}{4},$$

or

$$||\mathbf{w}||_2^2 = \frac{1}{\sqrt{2}}.$$

Now we can find the optimum weight vector that by definition must satisfy the equation

$$\mathbf{w} = \sum_{i=1}^4 \alpha_i d_i k(\mathbf{x}, \mathbf{x}_i),$$

that applying (A.25) and the solution for the Lagrange multipliers results in

$$\begin{aligned} \mathbf{w} &= \frac{1}{8} (-k(\mathbf{x}, \mathbf{x}_1) + k(\mathbf{x}, \mathbf{x}_2) + k(\mathbf{x}, \mathbf{x}_3) - k(\mathbf{x}, \mathbf{x}_4)) \\ &= \frac{1}{8} \left(- \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 \\ 0 \\ -\frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{aligned}$$

Now, we can test the optimality of the hyperplane evaluating the property

$$\mathbf{w}^T k(\mathbf{x}, \mathbf{x}_i) + b = 0,$$

for a bias $b = 0$ we arrive to

$$\begin{pmatrix} 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{pmatrix} = 0,$$

that results into

$$-x_1x_2 = 0.$$

For both input variables $x_1 = x_2 = -1$ and $x_1 = x_2 = +1$, the output is $y = -1$. Also it can be followed that for $x_1 = -1, x_2 = +1$ and $x_1 = +1, x_2 = -1$, the output is $y = +1$. Therefore, this problem is said to be solved.

A.2 Summary of Support Vector Machines Learning Methods

The most relevant training strategies for SVMs are presented below in chronological order. The review begins with Osuna's, *et al.* and finishes with Sra's work, to cover a period of 20 years.

A.2.1 Osuna's Decomposition Algorithm

In 1997 Osuna, *et al.* proposed a decomposition method that achieves optimality by solving a sequence of smaller sub-problems from randomly selected samples [139]. The authors start by formulating the Quadratic Programming (QP) fundamental

problem for the two class SVM as follows:

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & Q(\boldsymbol{\alpha}) = -\sum_{i=1}^N \alpha_i + \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) \\
\text{s.t.} \quad & \begin{cases} \sum_{i=1}^N \alpha_i d_i = 0 \\ -\boldsymbol{\alpha} \leq \mathbf{0} \\ \boldsymbol{\alpha} - C \leq \mathbf{0} \end{cases} \\
\text{for} \quad & i = 1, 2, \dots, N.
\end{aligned} \tag{A.26}$$

which is a slightly modification of problem (A.13), where $d_i \in \{-1, 1\}$. Then, since the kernel function is a positive definite matrix and the constraints are linear, then it follows to define the Karush-Kuhn-Tucker (KKT) conditions for optimality:

$$\nabla Q(\boldsymbol{\alpha}) + \sum_{i=1}^N v_i - \sum_{i=1}^N \pi_i + \mu \sum_{i=1}^N d_i = \mathbf{0} \tag{A.27a}$$

$$\sum_{i=1}^N v_i \alpha_i - C = 0 \tag{A.27b}$$

$$\sum_{i=1}^N \pi_i \alpha_i = 0 \tag{A.27c}$$

$$\mathbf{v} \geq \mathbf{0} \tag{A.27d}$$

$$\boldsymbol{\pi} \geq \mathbf{0} \tag{A.27e}$$

$$\sum_{i=1}^N \alpha_i d_i = 0 \tag{A.27f}$$

$$-\boldsymbol{\alpha} \leq \mathbf{0} \tag{A.27g}$$

$$\boldsymbol{\alpha} - C \leq \mathbf{0} \tag{A.27h}$$

$$i = 1, 2, \dots, N. \tag{A.27i}$$

where μ , v , and π are the Lagrange multipliers associated with the KKT conditions.

Then the authors assume that there exists some α_i that satisfies $0 \leq \alpha_i \leq C$, and therefore there are three possible cases:

1. Case: $0 \leq \alpha_i \leq C$

From the first tree equations of the KKT conditions we can derive

$$\mu \sum_{i=1}^N d_i - 1 + \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) = 0 \quad (\text{A.28a})$$

which the authors show that this implies $\mu = b$.

2. Case: $\alpha_i = C$

From the first tree equations of the KKT conditions we can derive

$$\mu \sum_{i=1}^N d_i - 1 + \sum_{i=1}^N v_i + \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) = 0 \quad (\text{A.28b})$$

from which we can define by convenience the following function:

$$g(\mathbf{x}_i) = \sum_{j=1}^N \alpha_j d_j k(\mathbf{x}_i, \mathbf{x}_j) + b \quad (\text{A.28c})$$

and recalling that $\mu = b$ and the restriction of v_i being strictly positive we can show that the following condition must hold:

$$d_i g(\mathbf{x}_i) \leq 1 \quad (\text{A.28d})$$

3. Case: $\alpha_i = 0$

From the first tree equations of the KKT conditions we can derive

$$\mu \sum_{i=1}^N d_i - 1 - \sum_{i=1}^N \pi_i + \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) = 0 \quad (\text{A.28e})$$

which leads us to

$$d_i g(\mathbf{x}_i) \geq 1 \quad (\text{A.28f})$$

The previous conditions are part of the stopping criteria of the decomposition algorithm proposed by Osuna *et al.* This algorithm suggests partitioning the index set $\{1, 2, \dots, N\}$ in two sets \mathcal{B} and \mathcal{M} , $\{\mathcal{B}, \mathcal{M}\} = \{1, 2, \dots, N\}$. The set \mathcal{B} is defined as the *working set*. It allows to decompose α_i into two vectors $\alpha_{\mathcal{B}}$ and $\alpha_{\mathcal{M}}$ where the former is the one that is changing.

Using this decomposition, the problem in (A.26) has been reformulated in terms of the index set variables \mathcal{B}, \mathcal{M} as follows:

$$\begin{aligned} \min \quad Q(\alpha_{\mathcal{B}}) = & -\sum_{i \in \mathcal{B}} \alpha_i + \frac{1}{2} \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{B}} \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (\text{A.29}) \\ & + \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{M}} \alpha_i \alpha_j d_i d_j k(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} \quad & \begin{cases} \sum_{i \in \mathcal{B}} \alpha_i d_i + \sum_{j \in \mathcal{M}} \alpha_j d_j = 0 \\ -\sum_{i \in \mathcal{B}} \alpha_i \leq 0 \\ \sum_{i \in \mathcal{B}} \alpha_i - C \leq 0 \end{cases} \end{aligned}$$

The procedure in Algorithm A.1 selects the support vector coefficients that are active on either side of the optimal hyperplane, then proceeds iteratively until the completion of the problem size. That is, the algorithm suggests adding one example and subtracting one example at each iteration. This algorithm performs satisfactory in applications of up to 100,000 data points; however, the algorithm has deficiencies. Platt's paper refers to these deficiencies as follows [147]: "Numerical QP is tricky to get right; there are many numerical precision issues that need to be addressed." Also, the author fails to prove convergence, as Smola, *et al.* [178] confirm when they say "... this [algorithm] does not prove convergence (contrary to statement in Osuna, Freund and Girosi (1997) [139]) ..."

The problem with this algorithm is that it requires a numerical QP solver at each

iteration. It is well known that numerical QP solvers lack precision.

Algorithm A.1 Osuna’s algorithm

```

1: procedure  $\alpha_{\mathcal{B}} = \text{TRAINSVM}(\mathbf{x}_i, d_i, C, |\mathcal{B}|, N)$ 
2:    $[\mathcal{B}, \mathcal{M}] \leftarrow \text{RandIdx}(|\mathcal{B}|, N)$   $\triangleright$  Randomly choose  $|\mathcal{B}|$  points from  $N$ 
3:    $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C)$   $\triangleright$  Solve sub-problem (A.29) for the variables in  $\mathcal{B}$ 
4:   for all  $j \in \mathcal{M}$  do
5:     if  $\left( \begin{array}{lll} (\alpha_j = 0 & \text{and} & g(\mathbf{x}_j)d_j \leq 1) & \text{or} \\ (\alpha_j = C & \text{and} & g(\mathbf{x}_j)d_j \geq 1) & \text{or} \\ (0 \leq \alpha_j \leq C & \text{and} & g(\mathbf{x}_j)d_j \neq 1) \end{array} \right)$  then
6:        $[\mathcal{B}, \mathcal{M}] \leftarrow \text{ReplaceIdx}(i \in \mathcal{B}, j \in \mathcal{M})$   $\triangleright$  Replace any  $\alpha_i$ , with  $\alpha_j$ 
7:        $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C)$   $\triangleright$  Solve (A.29) for new  $\mathcal{B}$ 
8:     end if
9:   end for
10:  return  $\alpha_{\mathcal{B}}$ 
11: end procedure

```

A.2.2 Joachims’ SVM^{Light}

Later in 1998, Joachims introduced the concept of “shrinking” the problem in (A.26) under the SVM training context [98]. The authors follow Osuna’s algorithm up to the point of breaking down the problem to find the new *working set*. Unlike Osuna, Joachims uses a more formal method. The author proposes to follow Zoutendijk’s method [223] on a first order approximation to the target function. In this method we want to obtain a feasible direction vector p that has steep descent and at most q non-zero entries. The direction of the vector p is

$$-\mathbf{1} + \mathbf{R}\boldsymbol{\alpha} = \nabla Q(\boldsymbol{\alpha}) \tag{A.30}$$

where $\mathbf{R} = \sum_{i=1}^N \sum_{j=1}^N d_i d_j k(\mathbf{x}_i, \mathbf{x}_j)$. However the direction in (A.30) may not be sparse or feasible. Thus, it follows to define the problem:

$$\begin{aligned} \min \quad & \sum_{i=1}^N p_i \nabla Q(\alpha_i) \\ \text{s.t.} \quad & \left\{ \begin{array}{l} \sum_{i=1}^N p_i d_i = 0 \\ p_i \geq 0 \quad \text{for } i : \alpha_i = 0 \\ p_i \leq 0 \quad \text{for } i : \alpha_i = C \\ p_i \leq 1 \\ -p_i \geq 1 \\ |\{d_i : d_i \neq 0\}| = q \end{array} \right. \end{aligned} \quad (\text{A.31})$$

where the vector p_i can take on values $\{-1, 0, 1\}$ that establish which samples will be part of the new working set \mathcal{B} .

The other idea that Joachims introduced is “shrinking.” To define this concept, the author uses the following decomposition: Let \mathcal{X} be the indexes associated to unbounded support vectors, \mathcal{Y} define the indexes associated to bounded support vectors, and \mathcal{Z} the indexes of non-support vectors. Then, the author defines

$$\boldsymbol{\alpha} = \begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{X}} \\ \boldsymbol{\alpha}_{\mathcal{Y}} \\ \boldsymbol{\alpha}_{\mathcal{Z}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{X}} \\ C\mathbf{1} \\ \mathbf{0} \end{pmatrix} \quad \mathbf{d} = \begin{pmatrix} \mathbf{d}_{\mathcal{X}} \\ \mathbf{d}_{\mathcal{Y}} \\ \mathbf{d}_{\mathcal{Z}} \end{pmatrix} \quad \mathbf{R} = \begin{pmatrix} \mathbf{R}_{\mathcal{X}\mathcal{X}} & \mathbf{R}_{\mathcal{X}\mathcal{Y}} & \mathbf{R}_{\mathcal{X}\mathcal{Z}} \\ \mathbf{R}_{\mathcal{Y}\mathcal{X}} & \mathbf{R}_{\mathcal{Y}\mathcal{Y}} & \mathbf{R}_{\mathcal{Y}\mathcal{Z}} \\ \mathbf{R}_{\mathcal{Z}\mathcal{X}} & \mathbf{R}_{\mathcal{Z}\mathcal{Y}} & \mathbf{R}_{\mathcal{Z}\mathcal{Z}} \end{pmatrix} \quad (\text{A.32})$$

where $\mathbf{R} = \sum_{i=1}^N \sum_{j=1}^N d_i d_j k(\mathbf{x}_i, \mathbf{x}_j)$. Then Joachims defines the decomposition as

follows

$$\begin{aligned} \min \quad & Q(\boldsymbol{\alpha}_{\mathcal{X}}) = -\boldsymbol{\alpha}_{\mathcal{X}}^T (\mathbf{1} - C\mathbf{R}_{\mathcal{X}\mathcal{Y}}\mathbf{1}) + \frac{1}{2}\boldsymbol{\alpha}_{\mathcal{X}}^T \mathbf{R}_{\mathcal{X}\mathcal{X}}\boldsymbol{\alpha}_{\mathcal{X}} \quad . \quad (\text{A.33}) \\ \text{s.t.} \quad & \begin{cases} \boldsymbol{\alpha}_{\mathcal{X}}^T \mathbf{y}_{\mathcal{X}} + C\mathbf{1}^T \mathbf{y}_{\mathcal{Y}} = \mathbf{0} \\ \boldsymbol{\alpha}_{\mathcal{X}} \leq C\mathbf{1} \\ -\boldsymbol{\alpha}_{\mathcal{X}} \leq \mathbf{0} \end{cases} \end{aligned}$$

Then Joachims define a heuristic approach to identify certain variables that will be at boundaries (*i.e.*, support vectors); this approach is based on Lagrange multiplier estimates. The idea is based on the fact that a strictly positive value of a Lagrange multiplier of a bound constraint implies variable optimality at such bound. The point is to use a Lagrange multiplier estimate whenever a non-optimal point arises. Let \mathcal{A} denote the current set of α_i that satisfy $0 \leq \alpha_i \leq C$, then, by solving (A.27) for μ and averaging for all α_i in \mathcal{A} we have

$$\mu = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \left(d_i - \sum_{j=1}^N \alpha_j d_j k(\mathbf{x}_i, \mathbf{x}_j) \right), \quad (\text{A.34})$$

and since it is not possible that the variables α_i exist at the upper and lower bound simultaneously, we can define the lower bound as

$$\pi_i = d_i \left(\left[\sum_{j=1}^N \alpha_j d_j k(\mathbf{x}_i, \mathbf{x}_j) \right] + \mu \right) - 1 \quad (\text{A.35})$$

and the upper bound as

$$v_i = -d_i \left(\left[\sum_{j=1}^N \alpha_j d_j k(\mathbf{x}_i, \mathbf{x}_j) \right] + \mu \right) + 1 \quad (\text{A.36})$$

If the Lagrange multiplier estimates are positive for last h iterations, then it is likely that they correspond to the optimal solution; this procedure is explained in

Algorithm A.2.

The only issues with this algorithm are the inherent QP solution process and the additional steepest descent process.

Algorithm A.2 Joachims' algorithm

```

1: procedure  $\alpha_{\mathcal{B}} = \text{TRAINSVM}(\mathbf{x}_i, d_i, C, |\mathcal{B}|, N, h)$ 
2:    $[\mathcal{B}, \mathcal{M}] \leftarrow \text{GetShrinkIdx}(|\mathcal{B}|, N)$   $\triangleright$  Choose  $|\mathcal{B}|$  points from  $N$  using (A.30)
3:    $[\alpha_{\mathcal{B}}, \mu_{\mathcal{B}}, \pi_{\mathcal{B}}, v_{\mathcal{B}}] \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C)$   $\triangleright$  Solve (A.33)-(A.36) for  $\mathcal{B}$ 
4:    $\text{Iter} \leftarrow 0$ 
5:   repeat
6:     if  $\left( \begin{array}{lll} (\alpha_j = 0 & \text{and} & g(\mathbf{x}_j)d_j \leq 1) & \text{or} \\ (\alpha_j = C & \text{and} & g(\mathbf{x}_j)d_j \geq 1) & \text{or} \\ (0 \leq \alpha_j \leq C & \text{and} & g(\mathbf{x}_j)d_j \neq 1) \end{array} \right)$  then
7:        $\text{Iter} \leftarrow \text{Iter} + 1$ 
8:        $[\mathcal{B}, \mathcal{M}] \leftarrow \text{ReplaceIdx}(|\mathcal{B}|, \mathcal{B}, \mathcal{M}, N)$   $\triangleright$  Get new  $\mathcal{B}$  from  $N$  using (A.30)
9:        $[\alpha_{\mathcal{B}}, \mu_{\mathcal{B}}, \pi_{\mathcal{B}}, v_{\mathcal{B}}] \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C)$   $\triangleright$  Solve (A.33)-(A.36) for  $\mathcal{B}$ 
10:    end if
11:    until  $\left( \begin{array}{ll} \mu(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 & \text{and} \\ \pi(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 & \text{and} \\ v(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 \end{array} \right)$ 
12:  return  $\alpha_{\mathcal{B}}$ 
13: end procedure

```

A.2.3 Platt's Sequential Minimal Optimization

Also in 1999, Platt extended Osuna's work for the case of $|\mathcal{B}| = 2$, and he called his algorithm "sequential minimal optimization" (SMO) [148]. This algorithm has been implemented in several commercial and open-source software applications such as "Weka" [74]. The key idea behind Platt's method is to decompose the large QP problem into a series of two dimensional QP sub-problems; therefore, the problem is so small that it can be solved without a QP solver, thus much faster. In Platt's mathematical derivation, we are interested in optimizing both α_i, α_j leaving the remaining α 's fixed. Therefore, we can reduce the inequalities in problem (A.29) for

the two variable case dual problem as follows:

$$d_i \alpha_i + d_j \alpha_j = - \sum_{\substack{i=1 \\ i \notin \{j,k\}}}^N d_k \alpha_k = \gamma$$

$$\alpha_j^{\text{new}} = \frac{\gamma - d_i \alpha_i}{d_j} \quad (\text{A.37})$$

Therefore, this is used to remove α_j from the problem, and then it is just matter of solving the quadratic equation for α_i by clipping the previous value as follows:

$$\alpha_i^{\text{new}} = \alpha_i - \frac{d_i(E_j - E_i)}{2k(\mathbf{x}_i, \mathbf{x}_j) - k(\mathbf{x}_i, \mathbf{x}_i) - k(\mathbf{x}_j, \mathbf{x}_j)} \quad (\text{A.38})$$

where $E_i = y_i - d_i$, that is the error at point i . Then, to decide which α 's are selected as the first pair, the author uses heuristics that make single passes over the whole training set, and then multiple passes over the unbounded support vectors since they are more likely to require re-optimization. The author uses a different heuristic to maximize the step size towards the solution. The SMO stores the value of E_i for each unbounded support vector i , and if the heuristics chose i as the first element, then j will be the one that maximizes $|E_i - E_j|$, these ideas are shown in Algorithm A.3.

The SMO also stores the value of $\mathbf{w} = \sum_i^N d_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_j)$ and updates it as follows:

$$\mathbf{w}^{\text{new}} = \mathbf{w} + d_i(\alpha_i^{\text{new}} - \alpha_i)k(\mathbf{x}_i, \mathbf{x}_j) \quad (\text{A.39})$$

which increases training speed. This is because instead of having to compute a kernel inner product for all the training samples, it only has to compute a single dot product. The problem with SMO is that the algorithm was highly questionable. It received criticism because the algorithm as reported was not as fast as promised and as reported by the author. Rifkin from MIT in his dissertation [155] comments that "... Platt's algorithm as written is not particularly fast, and the timing results

spring from a poor implementation of [other methods]...” Rifkin reports that his implementation of the SMO was found to be quite slow.

Algorithm A.3 Platt’s algorithm

```

1: procedure  $\alpha_{\mathcal{B}} = \text{TRAINSVM}(\mathbf{x}_i, d_i, C, |\mathcal{B}| = 2, N = |N| - 2)$ 
2:    $[\mathcal{B}, \mathcal{M}] \leftarrow \text{FindBestIdx}(|\mathcal{B}|, N)$   $\triangleright$  Choose best two points from  $N$ 
3:    $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$   $\triangleright$  Solve sub-problem (A.29) for the variables in  $\mathcal{B}$ 
4:   for all  $j \in \mathcal{M}$  do
5:     if  $\left( \begin{array}{lll} (\alpha_j = 0 & \text{and} & g(\mathbf{x}_j)d_j \leq 1) & \text{or} \\ (\alpha_j = C & \text{and} & g(\mathbf{x}_j)d_j \geq 1) & \text{or} \\ (0 \leq \alpha_j \leq C & \text{and} & g(\mathbf{x}_j)d_j \neq 1) \end{array} \right)$  then
6:        $[\mathcal{B}, \mathcal{M}] \leftarrow \text{FindNewBestIdx}(i \in \mathcal{B}, j \in \mathcal{M})$   $\triangleright$  Replace any  $\alpha_i$ , with  $\alpha_j$ 
7:        $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$   $\triangleright$  Solve using procedure below
8:     end if
9:   end for
10:  return  $\alpha_{\mathcal{B}}$ 
11: end procedure
12: procedure  $\alpha_{\mathcal{B}} = \text{SOLVEQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$ 
13:    $\alpha_j \leftarrow \text{GetAlphaJ}(\mathcal{B}[1], d_{\mathcal{B}})$   $\triangleright$  Solve  $\alpha_j$  with (A.37)
14:    $\alpha_i \leftarrow \text{GetAlphaI}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}[2], C)$   $\triangleright$  Solve  $\alpha_i$  with (A.38)
15:    $\alpha_{\mathcal{B}} \leftarrow \{\alpha_i, \alpha_j\}$ 
16:  return  $\alpha_{\mathcal{B}}$ 
17: end procedure

```

A.2.4 Rifkin’s SVMFu

Rifkin, in his PhD dissertation work, presented the “SVMFu” algorithm in 2002 [155]. This work reviews and synthesizes Osuna’s, Platt’s, and Joachims’ work, as shown in Algorithm A.4.

However, the author redefines the gradient function to be in terms of problem (A.33), so the gradient is no longer denoted as

$$\nabla Q(\boldsymbol{\alpha}) = -\mathbf{1} + \mathbf{R}\boldsymbol{\alpha}$$

Algorithm A.4 Rifkin's algorithm

```

1: procedure  $\alpha_{\mathcal{B}} = \text{TRAIN SVM}(\mathbf{x}_i, d_i, C, |\mathcal{B}|, N)$ 
2:    $[\mathcal{B}, \mathcal{M}] \leftarrow \text{FindBestIdx}(|\mathcal{B}|, N)$   $\triangleright$  Choose best two points from  $N$ 
3:    $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$   $\triangleright$  Solve sub-problem (A.29) for the variables in  $\mathcal{B}$ 
4:   for all  $j \in \mathcal{M}$  do
5:     if  $\left( \begin{array}{lll} (\alpha_j = 0 & \text{and} & g(\mathbf{x}_j)d_j \leq 1) & \text{or} \\ (\alpha_j = C & \text{and} & g(\mathbf{x}_j)d_j \geq 1) & \text{or} \\ (0 \leq \alpha_j \leq C & \text{and} & g(\mathbf{x}_j)d_j \neq 1) \end{array} \right)$  then
6:        $[\mathcal{B}, \mathcal{M}] \leftarrow \text{FindNewPair}(\mathcal{B}, \mathcal{M}, N)$   $\triangleright$  Use (A.40) and procedure below
7:        $\alpha_{\mathcal{B}} \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$   $\triangleright$  Solve using procedure below
8:     end if
9:   end for
10:  return  $\alpha_{\mathcal{B}}$ 
11: end procedure
12: procedure  $[\mathcal{B}, \mathcal{M}] = \text{FIND NEW PAIR}(\mathcal{B}, \mathcal{M}, N)$ 
13:   $[i, j] \leftarrow \arg_{i,j} \max |\nabla Q(\alpha_j) - \nabla Q(\alpha_i)|$   $\triangleright$  with  $i, j$  moving towards gradient
14:   $\mathcal{B} \leftarrow \{i, j\}; \mathcal{M} \leftarrow \{N \setminus \mathcal{B}\}$ 
15:   $E \leftarrow |\nabla Q(\alpha_j) - \nabla Q(\alpha_i)|$ 
16:  if  $(E \geq \text{Threshold})$  then
17:    return  $[\mathcal{B}, \mathcal{M}]$ 
18:  else if  $(|\mathcal{B}| < |N|)$  then
19:     $\mathcal{B} \leftarrow N; \mathcal{M} \leftarrow \emptyset$ 
20:    return  $[\mathcal{B}, \mathcal{M}]$ 
21:  else  $\triangleright$  It is done
22:     $\mathcal{B} \leftarrow \emptyset; \mathcal{M} \leftarrow \emptyset$ 
23:    return  $[\mathcal{B}, \mathcal{M}]$ 
24:  end if
25: end procedure
26: procedure  $\alpha_{\mathcal{B}} = \text{SOLVE QP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}, C)$ 
27:   $\alpha_j \leftarrow \text{GetAlphaJ}(\mathcal{B}[1], d_{\mathcal{B}})$   $\triangleright$  Solve  $\alpha_j$  with (A.37)
28:   $\alpha_i \leftarrow \text{GetAlphaI}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, \mathcal{B}[2], C)$   $\triangleright$  Solve  $\alpha_i$  with (A.38)
29:   $\alpha_{\mathcal{B}} \leftarrow \{\alpha_i, \alpha_j\}$ 
30:  return  $\alpha_{\mathcal{B}}$ 
31: end procedure

```

but instead as

$$\nabla Q(\boldsymbol{\alpha}) = -\mathbf{1} + \mathbf{R}_{\mathcal{X}\mathcal{Y}}\boldsymbol{\alpha}_{\mathcal{Y}} + \mathbf{R}_{\mathcal{X}\mathcal{X}}\boldsymbol{\alpha}_{\mathcal{X}} \quad (\text{A.40})$$

which adds robustness to the sub-problem since the linear interactions between $\boldsymbol{\alpha}_{\mathcal{X}}$ and $\boldsymbol{\alpha}_{\mathcal{Y}}$ are considered.

The author also use “shrinking” methods to reduce the number of points, *e.g.*, to remove those points that have not been selected as part of the working set for a number of k iterations. The key contribution is the idea of dividing the large problem in sub-problems such that their Hessian matrices fit within memory limitations. Rather than storing the kernel inner products with the α ’s, he stores only the inner products of the kernel rows and columns associated to non-zero α ’s. The drawbacks of this algorithm is that in the case that there are not non-zero α ’s (*i.e.*, all data are support vectors) the complete Hessian is stored; also another issue arises with the usage of explicit “shrinking,” since this technique no longer guarantees optimality in the final solution.

A.2.5 Nystrom Method for Kernel Approximation

In 2005, Drineas, *et al.* developed an algorithm to approximate a kernel in the form of (A.21) with low rank matrix. They authors give mathematical proof that if $O(k/\varepsilon^4)$ columns are selected, then the following equation holds

$$\|\mathbf{K} - \mathbf{C}\mathbf{W}_k^+\mathbf{C}^T\|_{\xi} \leq \|\mathbf{K} - \mathbf{K}_k\|_{\xi} + \varepsilon \sum_{i=1}^N \mathbf{K}_{ii}^2, \quad (\text{A.41})$$

for both $\xi = \{2, F\}$, and for all $k : 0 \leq k \leq \text{rank}(\mathbf{W})$; where $\mathbf{K} \in \mathbb{R}^{N \times N}$ is a kernel in the form of (A.21), $\mathbf{C} \in \mathbb{R}^{N \times c}$ is a matrix formed by randomly selecting a small c number of columns and rows of \mathbf{K} , $\mathbf{W}_k \in \mathbb{R}^{c \times c}$ is the best rank- k approximation to \mathbf{W} , which is formed by the intersection of those c columns and rows of \mathbf{K} . Also,

it is important to recall that the notation \mathbf{W}^+ is used to denote the Moore-Penrose generalized inverse, also that the spectral norm $\|\cdot\|_2$ of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as

$$\|\mathbf{A}\|_2 = \sup_{\mathbf{x} \in \mathbb{R}^n, x \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \quad (\text{A.42})$$

where $\|\mathbf{x}\|_2 = (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$. Also, let us recall that the Frobenious norm $\|\cdot\|_F$ is defined as

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n \|\mathbf{A}_{i,j}\|_2^2 \right)^{\frac{1}{2}} \quad (\text{A.43})$$

Finally, the kernel approximation $\tilde{\mathbf{K}}$ is given by

$$\tilde{\mathbf{K}} = \mathbf{C}\mathbf{W}^+\mathbf{C}^T \quad (\text{A.44})$$

and it is computed as shown in Algorithm A.5.

Using this approximation, the amount of computation required to find the solution of an SVM can be reduced from $O(N^3)$ to $O(N)$ using the kernel approximation. The authors report an interesting derivation of their ideas based on the Nystron method [53].

A.2.6 Hush's QP SVM Dual to Primal Mapping

Continuing with QP-based methods, Hush, *et al.* in 2006, proposed an algorithm that produces approximate solutions without compromising accuracy in a QP SVM problem [90]. The authors propose a first stage where the algorithm provides an approximate solution for the QP dual and a second stage where that maps the QP-dual to the QP-primal, based on the duality theorem. The authors claim that this mapping is performed in $O(N \log N)$ time and with an accuracy of $(2\sqrt{2\mathbf{K}_N} + 8)^{-2}\epsilon_p^2$

Algorithm A.5 Nyström method for kernel approximation

```

1: procedure  $\tilde{\mathbf{K}} = \text{APPROXIMATEKERNEL}(\mathbf{K}, c \leq N, k \leq c)$ 
2:    $p_i \leftarrow \frac{\mathbf{K}_{ii}^2}{\sum_{i=1}^N \mathbf{K}_{ii}^2}$ 
3:    $\mathbf{S} \leftarrow \mathbf{0}_{N \times c}$  ▷ Define the matrix  $\mathbf{S} \in \mathbb{R}^{N \times c}$ 
4:    $\mathbf{D} \leftarrow \mathbf{0}_{c \times c}$  ▷ Define the matrix  $\mathbf{D} \in \mathbb{R}^{c \times c}$ 
5:   for  $(t = 1, \dots, c)$  do
6:     if  $(\text{Pr}(i_t = i) = p_i)$  then ▷ Pick  $i_t \in [N]$ , where  $\text{Pr}(i_t = i) = p_i$ 
7:        $\mathbf{D}_{tt} \leftarrow (cp_{i_t})^{-\frac{1}{2}}$ 
8:        $\mathbf{S}_{i_t} \leftarrow 1$ 
9:     end if
10:  end for
11:   $\mathbf{C} \leftarrow \mathbf{KSD}$ 
12:   $\mathbf{W} \leftarrow \mathbf{DS}^T \mathbf{KSD}$ 
13:   $\mathbf{W}_k \leftarrow \text{GetRank}k\text{Approx}(\mathbf{W})$ 
14:   $\tilde{\mathbf{K}}_k \leftarrow \mathbf{CW}_k^+ \mathbf{C}^T$ 
15:  return  $\tilde{\mathbf{K}}_k$ 
16: end procedure

```

in the dual, and with an accuracy ε_p back in the primal. The parameter \mathbf{K}_N is defined as

$$\mathbf{K}_N = \max_{1 \leq i \leq N} k(\mathbf{x}_i, \mathbf{x}_i). \quad (\text{A.45})$$

The proposed method is shown in Algorithm A.6, where $\mathbf{u} = \{u_i\}_{i=1}^N$ is functionally equivalent to the constant C in problem (A.12) but here there is a specific constant for each sample; and ε_p is the maximum error tolerated in the solution to the primal problem. Although the authors prove that the ε -accuracy is guaranteed, the process involves the solution to a QP problem, several sorting algorithms, and a limited selection of a decomposition strategy.

Algorithm A.6 Hush's, *et al.* method for primal QP-SVM guaranteed accuracy

```

1: procedure  $[\hat{\alpha}, \hat{\mathbf{b}}] = \text{PRIMALQP}(\mathcal{T} = \{(\mathbf{x}_i, d_i)\}_{i=1}^N, k(\cdot, \cdot), \mathbf{u}, \epsilon_p)$ 
2:   if  $(d_i = d_1, \forall i)$  then
3:      $\hat{\alpha} \leftarrow \begin{cases} 0, & d_1 = 1 \\ u_1, & d_1 = -1 \end{cases}$ 
4:      $\hat{\mathbf{b}} \leftarrow d_1$ 
5:   end if
6:    $Q_{ij} \leftarrow \frac{k(x_i, x_j)}{2}, l_i \leftarrow \frac{(1-d_i)u_i}{2}, w \leftarrow Ql + d, c \leftarrow l \cdot \mathbf{1}$  ▷ From the dual
7:    $\epsilon \leftarrow \frac{\epsilon_p^2}{(2\sqrt{2}\mathbf{K}_N + 8)^2}$  ▷ Compute desired accuracy for the dual
8:    $\alpha^0 \leftarrow l$  ▷ Initialize variable for the dual
9:    $[\hat{\alpha}, g] \leftarrow \text{Decomposition}(Q, w, c, \mathbf{u}, \epsilon, \alpha^0)$  ▷  $\epsilon$ -approximate dual solution  $\hat{\alpha}$ 
10:   $\hat{\mathbf{b}} \leftarrow \text{Offset}(g, d, \mathbf{u})$  ▷ Compute offset parameter
11:  return  $[\hat{\alpha}, \hat{\mathbf{b}}]$ 
12: end procedure

```

A.2.7 Sra's LP to QP SVM Mapping

Sra in 2006 proposed an algorithm that explicitly takes advantage of the LP formulation for SVMs to produce an efficient training method for large-scale problems [90]. The author formulates the two class (*i.e.* $d_i \in \{\pm 1\}$) SVM problem for the ℓ_1 -norm instead of the typical ℓ_2 -norm as follows:

$$\begin{aligned}
& \min_{\mathbf{w}, b} \quad ||\mathbf{w}\|_1 + C \sum_{i=1}^N \xi_i & (A.46) \\
& \text{s.t.} \quad \begin{cases} d_i(\mathbf{w}^T \mathbf{x} + b) \geq 1 - \xi_i \\ \xi_i \geq 0 \end{cases} \\
& \text{for} \quad i = 1, 2, \dots, N.
\end{aligned}$$

Then the author converts the LP problem into a QP problem using the principle establishing that there exists an $\epsilon_0 > 0$, such that for all $\epsilon \leq \epsilon_0$,

$$\arg \min_{\mathbf{g} \in G} ||\mathbf{g} + \epsilon^{-1} \mathbf{c}||_2^2 = \arg \min_{\mathbf{g} \in G^*} ||\mathbf{g}||_2^2 \quad (A.47)$$

where G is the feasible set, and G^* is the set of optimal solutions to the SVM QP problem, and is a unique minimizer. Here $\mathbf{g} = \begin{pmatrix} \mathbf{w} & b & |\mathbf{w}| & \boldsymbol{\xi} \end{pmatrix}$, $\mathbf{c} = \begin{pmatrix} \mathbf{0} & 0 & \mathbf{1} & C\mathbf{1} \end{pmatrix}$. In the solution of the latter QP problem, they apply Bregman's [27] and Hildreth's [81] decomposition, then the QP problem is solved. The authors report solving a problem with as many as 20,000,000 samples. The drawbacks are using QP formulation instead of taking advantage of the efficiency of an LP formulation and also sacrificing performance to gain speed.

A.3 Linear Programming Support Vector Machines

The methods for QP-SVM described in previous sections cannot be trivially extended to the LP-SVM and LP-SVR problems since the nature of QP and LP problems is different. In spite of this, there have been recent attempts to address an efficient technique to decompose a large-scale SVM problem using Linear Programming (LP) techniques [187], which could be extended to the LP-SVR case.

By the time this dissertation topic was already under development, Torii, *et al.* published his work on LP-SVM decomposition methods in 2009. His decomposition strategy for SVM is very similar to the one presented in this document; however, a different problem is addressed; that is, the LP-SVR problem is developed in this dissertation. Furthermore, they address a totally different formulation for the linear program, as the reader will notice later in Chapter 3. Torii, *et al.* [187] proposed what is called a “decomposition method” for LP-SVMs. The authors decompose the input training set indexes into two subsets: \mathcal{B} is the subset of indexes that represent the *working set*, while the remaining indexes are denoted as \mathcal{M} , where $\mathcal{B} \cap \mathcal{M} = \emptyset$. The authors follow the definition of problem (A.22) and then start to formulate it in a standard LP form. First, the ℓ_2 -norm is changed by the ℓ_1 -norm, and introduce $\boldsymbol{\alpha}$ to allow the usage of the kernel expansion. Then authors divide the $\boldsymbol{\alpha}$ in its positive $\boldsymbol{\alpha}^+$ and negative $\boldsymbol{\alpha}^-$ parts. To remove the inequality in the constraint, a

slack variable \mathbf{u} is added to arrive at the following formulation:

$$\begin{aligned}
& \max_{\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\xi}, b^+, b^-, \mathbf{u}} && \sum_{i \in \mathcal{B}} (\alpha_i^+ + \alpha_i^-) + C \sum_{i=1}^N \xi_i && (\text{A.48}) \\
& \text{s.t.} && \left\{ \begin{array}{l} d_i (\sum_{i \in \mathcal{B}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_i, \mathbf{x}_j) + b^+ \\ -b^- + \sum_{i \in \mathcal{M}} (\alpha_i^+ - \alpha_i^-) k(\mathbf{x}_i, \mathbf{x}_j)) + \xi_j = 1 + u_j \\ \boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, \boldsymbol{\xi} \geq \mathbf{0} \\ b^+, b^- \geq 0 \end{array} \right. \\
& \text{for} && j = 1, 2, \dots, N
\end{aligned}$$

which the authors claim is equivalent to the following LP-SVM problem:

$$\begin{aligned}
& \min && \sum_{i \in \mathcal{B}} c_i z_i + \mathbf{d}^T \boldsymbol{\xi} && (\text{A.49}) \\
& \text{s.t.} && \left\{ \begin{array}{l} \sum_{j \in \mathcal{B}} A_{i,j} z_j = b_i + u_i - \xi_i \\ \mathbf{z}, \mathbf{u}, \boldsymbol{\xi} \geq \mathbf{0} \end{array} \right. \\
& \text{for} && i \in \mathcal{B},
\end{aligned}$$

where \mathbf{d} is a vector, and $\boldsymbol{\xi}, \mathbf{u}$ are a slack variable vectors.

The corresponding dual is then defined as follows:

$$\begin{aligned}
& \max && \mathbf{b}^T \boldsymbol{\lambda} && (\text{A.50}) \\
& \text{s.t.} && \left\{ \begin{array}{l} \sum_{j=1}^N A_{j,i} z_j + v_i = c_i \\ \boldsymbol{\lambda} + \mathbf{t} = \mathbf{d} \\ \mathbf{z}, \mathbf{v}, \mathbf{t} \geq \mathbf{0} \end{array} \right. \\
& \text{for} && i \in \mathcal{B},
\end{aligned}$$

where $\boldsymbol{\lambda}, \mathbf{v}$ are vectors, and \mathbf{t} is a vector of slack variables.

The solution $(\mathbf{z}^*, \boldsymbol{\xi}^*, \mathbf{u}^*, \boldsymbol{\lambda}^*, \mathbf{v}^*, \mathbf{t}^*)$ is said to be optimal if it satisfies the corresponding optimality conditions, *i.e.*, the Karush-Kuhn-Tucker (KKT) conditions. If the solution to (A.49)-(A.50) is not optimal, they propose to move the indexes of inactive variables from \mathcal{B} to \mathcal{M} , and those variables that violate the optimality con-

ditions are moved from \mathcal{M} into \mathcal{B} . The process is repeated until a solution satisfies the optimality conditions for the working set and for the remaining variables. The authors propose a theorem that states for this method to produce a sequence of non-increasing values bounded (in its lower limit) by the global minimum of (A.49); however, Torii, *et al.* do not report mathematical proof or reference to an existing one.

Appendix B

Algorithms and Theorems Proofs

Some of the algorithms used in this dissertation are included in this appendix. Also, some Theorem proofs are included.

B.1 Algorithms

Algorithm B.1 Collobert's algorithm

```

1: procedure  $\alpha_{\mathcal{B}} = \text{TRAINSVR}(\mathbf{x}_i, d_i, C, \xi, \epsilon, q, N, h)$ 
2:    $[\mathcal{B}, \mathcal{M}] \leftarrow \text{GetShrinkIdx}(q, N)$   $\triangleright$  Choose  $q$  points from  $N$  using (2.16)
3:    $[\alpha_{\mathcal{B}}, \alpha_{\mathcal{B}}^*, \mu_{\mathcal{B}}, \pi_{\mathcal{B}}, v_{\mathcal{B}}] \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C, \xi, \epsilon)$   $\triangleright$  Solve (2.15) for  $\mathcal{B}$ 
4:    $\text{Iter} \leftarrow 0$ 
5:   repeat
6:     if  $\left( \begin{array}{lll} (\alpha_j = 0 & \text{and} & g(\mathbf{x}_j)d_j \leq 1) & \text{or} \\ (\alpha_j = C & \text{and} & g(\mathbf{x}_j)d_j \geq 1) & \text{or} \\ (0 \leq \alpha_j \leq C & \text{and} & g(\mathbf{x}_j)d_j \neq 1) \end{array} \right)$  then
7:        $\text{Iter} \leftarrow \text{Iter} + 1$ 
8:        $[\mathcal{B}, \mathcal{M}] \leftarrow \text{ReplaceIdx}(q, \mathcal{B}, \mathcal{M}, N)$   $\triangleright$  Get new  $\mathcal{B}$  from  $N$ 
9:        $[\alpha_{\mathcal{B}}, \alpha_{\mathcal{B}}^*, \mu_{\mathcal{B}}, \pi_{\mathcal{B}}, v_{\mathcal{B}}] \leftarrow \text{SolveQP}(\mathbf{x}_{\mathcal{B}}, d_{\mathcal{B}}, C, \xi, \epsilon)$   $\triangleright$  Solve (2.15) for  $\mathcal{B}$ 
10:    end if
11:    until  $\left( \begin{array}{ll} \mu(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 & \text{and} \\ \pi(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 & \text{and} \\ v(\text{Iter} - h : \text{Iter})_{\mathcal{B}} \geq 0 \end{array} \right)$ 
12:  return  $\alpha_{\mathcal{B}}, \alpha_{\mathcal{B}}^*$ 
13: end procedure

```

Algorithm B.2 Bradley and Mangasarian's LPC algorithm

```

1: procedure  $\mathbf{z} = \text{LPC}(\mathbf{c}, \mathbf{A}, \mathbf{b}, \ell)$ 
2:    $\begin{pmatrix} \mathbf{A}^1 & \mathbf{b}^1 \\ \mathbf{A}^2 & \mathbf{b}^2 \\ \vdots & \vdots \\ \mathbf{A}^\ell & \mathbf{b}^\ell \end{pmatrix} \leftarrow \text{BlockPartition}(\ell, \mathbf{A}, \mathbf{b})$  ▷ Partition into  $\ell$  blocks
3:    $j \leftarrow 0$  ▷ Iterations counter
4:   repeat
5:      $j \leftarrow j + 1$ 
6:      $\mathbf{z}^j \leftarrow \arg \text{vertex} \min \left\{ \mathbf{c}^T \mathbf{z} \mid \begin{array}{l} \mathbf{A}^{(j \bmod \ell)} \mathbf{z} \geq \mathbf{b}^{(j \bmod \ell)} \\ \bar{\mathbf{A}}^{(j \bmod \ell)-1} \mathbf{z} \geq \bar{\mathbf{b}}^{(j \bmod \ell)-1} \end{array} \right\}$ 
7:   until  $(\mathbf{c}^T \mathbf{z}^j = \mathbf{c}^T \mathbf{z}^{j+4})$  ▷ Stops if there is no change during four iterations
8:   return  $\mathbf{z}^j$ 
9: end procedure

```

B.2 Theorems and Proofs

B.2.1 Theorem on the ESV Bound

Theorem B.1 (Zhang, *et al.* [214]: ESV Bound). *Given an optimal solution \mathbf{z}^* to (3.2), the number of nonzero α_i coefficients of (3.2) has the following upper bound:*

$$|\mathcal{A}| \leq |\mathcal{V}_E|. \quad (\text{B.1})$$

for all $i : \alpha_i \neq 0$.

Proof. Consider the LP problem

$$\min_{\mathbf{z}^*} \quad \mathbf{c}^T \mathbf{z}^* \quad (\text{B.2a})$$

$$\text{s.t. } \mathbf{A} \mathbf{z}^* = \mathbf{b} \quad (\text{B.2b})$$

$$\mathbf{z}^* \geq \mathbf{0}, \quad (\text{B.2c})$$

which is equivalent to (3.2), where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{z}^* \in \mathbb{R}^n$. Then, for $\boldsymbol{\alpha}^*$ we can

Algorithm B.3 Unfeasible interior point method used

```

1: procedure  $[\mathbf{z}_B, \boldsymbol{\lambda}_B, \mathbf{s}_B] = \text{SOLVELP}(\mathbf{x}_B, d_B, B, C, \epsilon, \sigma, \mu)$ 
2:    $\mathbf{A}_B \leftarrow \begin{pmatrix} -\mathbf{K}_B & \mathbf{K}_B & -1 & 1 & -\mathbf{I}_B & \mathbf{I}_B \\ \mathbf{K}_B & -\mathbf{K}_B & 1 & -1 & -\mathbf{I}_B & \mathbf{I}_B \end{pmatrix}$ 
3:    $\mathbf{b}_B \leftarrow \begin{pmatrix} \mathbf{1}\epsilon_B - \mathbf{d}_B \\ \mathbf{1}\epsilon_B + \mathbf{d}_B \end{pmatrix}$ 
4:    $\mathbf{c}_B \leftarrow (\mathbf{1}_B \quad \mathbf{1}_B \quad 0 \quad 0 \quad 2\mathbf{C}_B \quad \mathbf{0}_B)^T$ 
5:    $(\mathbf{z}_0, \boldsymbol{\lambda}_0, \mathbf{s}_0) \leftarrow \text{UnfeasibleInitialPoint}(\mathbf{A}_B, \mathbf{b}_B, \mathbf{c}_B)$   $\triangleright$  With  $(\mathbf{z}_0, \mathbf{s}_0) > 0$ 
6:   for  $t = 0, 1, \dots$ , until  $[\mathbf{r}_c, \mathbf{r}_b, \mathbf{XS1}, \text{stop}] = \text{Convergence}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}, \mathbf{A}, \mathbf{b}, \mathbf{c})$  do
7:      $[\Delta \mathbf{z}_p, \Delta \boldsymbol{\lambda}_p, \Delta \mathbf{s}_p] \leftarrow \text{PredictorDirection}(\mathbf{A}_B, \mathbf{z}_t, \mathbf{s}_t, \mathbf{r}_c, \mathbf{r}_b, \mathbf{XS1})$ 
8:      $\mu_{\text{cen}} \leftarrow \text{CenteringParameter}(\mathbf{z}, \Delta \mathbf{z}_p, \mathbf{s}, \Delta \mathbf{s}_p, \tau, \mathbf{XS1})$ 
9:      $[\Delta \mathbf{z}_{\text{pc}}, \Delta \boldsymbol{\lambda}_{\text{pc}}, \Delta \mathbf{s}_{\text{pc}}] \leftarrow \text{PCDir}(\mathbf{A}_B, \mathbf{z}_t, \mathbf{s}_t, \mathbf{r}_c, \mathbf{r}_b, \mathbf{XS1}, \mu_{\text{cen}}, \Delta \mathbf{z}_p, \Delta \boldsymbol{\lambda}_p, \Delta \mathbf{s}_p)$ 
10:     $[\mathbf{z}_{t+1}, \boldsymbol{\lambda}_{t+1}, \mathbf{s}_{t+1}] \leftarrow \text{Globalization}(\mathbf{z}_t, \boldsymbol{\lambda}_t, \mathbf{s}_t), \Delta \mathbf{z}_{\text{pc}}, \Delta \boldsymbol{\lambda}_{\text{pc}}, \Delta \mathbf{s}_{\text{pc}}, \mu_{\text{cen}}, \tau)$ 
11:    if  $(\text{DetectInfeasibility}(\mathbf{r}_c, \mathbf{r}_b, t) == \text{true})$  then
12:      return  $[\mathbf{z}_{B,t-1}, \boldsymbol{\lambda}_{B,t-1}, \mathbf{s}_{B,t-1}]$ 
13:    end if
14:  end for
15:   $\mathbf{z}_B \leftarrow \mathbf{z}^* | \mathbf{z} = (\boldsymbol{\alpha}_B^+ \quad \boldsymbol{\alpha}_B^- \quad b^+ \quad b^- \quad \boldsymbol{\xi}_B \quad \mathbf{u}_B)^T$ 
16:   $[\boldsymbol{\lambda}_B, \mathbf{s}_B] \leftarrow [\boldsymbol{\lambda}^*, \mathbf{s}^*]$ 
17:  return  $[\mathbf{z}_B, \boldsymbol{\lambda}_B, \mathbf{s}_B]$ 
18: end procedure
19: procedure  $[\mathbf{r}_c, \mathbf{r}_b, \mathbf{XS1}, \text{STOP}] = \text{CONVERGENCE}(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}, \mathbf{A}, \mathbf{b}, \mathbf{c})$ 
20:    $\mathbf{XS1} \leftarrow \text{GetGap}(\mathbf{z}, \mathbf{s})$   $\triangleright$  Duality gap
21:    $[\mathbf{r}_c, \mathbf{r}_b] \leftarrow \text{GetResiduals}(\mathbf{A}, \mathbf{b}, \mathbf{c}, \mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$   $\triangleright$  Residuals at current solution
22:    $\text{stop} \leftarrow \text{CheckCriteria}(\mathbf{r}_c, \mathbf{r}_b, \mathbf{b}, \mathbf{c}, \mathbf{XS1})$   $\triangleright$  Returns a boolean
23:   return  $[\mathbf{r}_c, \mathbf{r}_b, \mathbf{XS1}, \text{stop}]$ 
24: end procedure
25: procedure  $\mu = \text{CENTERINGPARAMETER}(\mathbf{z}, \Delta \mathbf{z}_p, \mathbf{s}, \Delta \mathbf{s}_p, \tau, \mathbf{XS1})$ 
26:    $\varphi_{\text{cen}}^{\text{pri}} \leftarrow \arg \max \{ \varphi \in [0, 1] \mid \mathbf{z}_t + \varphi \Delta \mathbf{z}_p \geq 0 \}$   $\triangleright$  Steps length
27:    $\varphi_{\text{cen}}^{\text{dual}} \leftarrow \arg \max \{ \varphi \in [0, 1] \mid \mathbf{s}_t + \varphi \Delta \mathbf{s}_p \geq 0 \}$ 
28:    $\sigma \leftarrow ((\mathbf{z}_t + \varphi_{\text{cen}}^{\text{pri}} \Delta \mathbf{z}_p)^T (\mathbf{s}_t + \varphi_{\text{cen}}^{\text{dual}} \Delta \mathbf{s}_p) / \mathbf{XS1})^3$ 
29:    $\mu_{\text{cen}} \leftarrow \sigma(\mathbf{XS1} / |B|)$   $\triangleright$  Affine-scaling direction
30:   return  $\mu_{\text{cen}}$ 
31: end procedure

```

define the following equality

$$|\mathcal{A}| = |\{\alpha_j^* : \alpha_j^* \neq 0, j = 1, 2, \dots, N\}| \quad (\text{B.3})$$

$$|\{\alpha_j^{+*} : \alpha_j^{+*} \neq 0, j = 1, 2, \dots, N\}| \dots \quad (\text{B.4})$$

$$+ |\{\alpha_j^{-*} : \alpha_j^{-*} \neq 0, j = 1, 2, \dots, N\}|. \quad (\text{B.5})$$

Now, let

$$NPE(\mathbf{z}^*) \leq m, \quad (\text{B.6})$$

denote an upper bound to the number of positive elements in \mathbf{z}^* . By (B.6), there are at most $2N$ basic variables in \mathbf{z}^* that can take nonzero values; the other non-basic variables take zeros. Among these basic variables, there are $|\mathcal{V}_S|$ of $\xi^* > 0$ and $|\mathcal{V}_S|$ of $\mathbf{u}^* > 0$, $|\mathcal{V}_N|$ of $\mathbf{u}^* > 0$, and $|\mathcal{V}_E|$ of $\mathbf{u}^* = 2\epsilon$ apart from the nonzero coefficients in $\alpha_j^{+*}, \alpha_j^{-*}$, $j = \{1, 2, \dots, N\}$. As a result, the number of nonzero coefficients is

$$|\mathcal{A}| = |\{\alpha_j^* : \alpha_j^* \neq 0, j = 1, 2, \dots, N\}| \quad (\text{B.7})$$

$$\leq 2N - 2|\mathcal{V}_S| - |\mathcal{V}_N| - |\mathcal{V}_E|. \quad (\text{B.8})$$

Since $N = |\mathcal{V}_S| + |\mathcal{V}_E| + |\mathcal{V}_N|$, we have that

$$|\mathcal{A}| \leq |\mathcal{V}_E| + |\mathcal{V}_N|. \quad (\text{B.9})$$

This completes the proof. □

B.2.2 Theorem on The Rank Bound

Theorem B.2 (Zhang, *et al.* [214]: Rank Bound). *Given an optimal solution \mathbf{z}^* to (3.2), the number of nonzero coefficients of (3.2) has the following upper bound:*

$$|\mathcal{A}| \leq \text{rank}(\mathbf{K}), \quad (\text{B.10})$$

and the column vectors $k(\mathbf{x}_j, \mathbf{x}_1), k(\mathbf{x}_j, \mathbf{x}_2), \dots, k(\mathbf{x}_j, \mathbf{x}_i)$, are linearly independent for all $j \in \mathcal{A}$.

Proof. For the LP-SVR problem (3.2), that is equivalent to (B.2), let us denote the column vector matrix associated with the variables $\alpha_j^{+*}, \alpha_j^{-*}$, $j = \{1, 2, \dots, N\}$ by

$$\mathbf{B}_\alpha = \begin{pmatrix} -\mathbf{K} & \mathbf{K} \\ \mathbf{K} & -\mathbf{K} \end{pmatrix}. \quad (\text{B.11})$$

According to [214], the number $|\mathcal{A}|$ of nonzero variables in $\alpha_j^{+*}, \alpha_j^{-*}$, $j = \{1, 2, \dots, N\}$ is at most equal to the number of columns in the corresponding basic column vector matrix \mathbf{B}_α^* which are shared by \mathbf{B}_α and the optimal basic matrix \mathbf{B}^* corresponding to the optimal solution \mathbf{z}^* . Hence,

$$|\mathcal{A}| \leq \text{rank}(\mathbf{B}_\alpha^*) \leq \text{rank}(\mathbf{B}_\alpha) = \text{rank}(\mathbf{K}), \quad (\text{B.12})$$

thus,

$$|\mathcal{A}| \leq \text{rank}(\mathbf{K}). \quad (\text{B.13})$$

Since the optimal basic matrix \mathbf{B}^* is linearly independent, so is \mathbf{B}_α^* . Now the column vectors $\{(\phi_j(\mathbf{x}_1), \phi_j(\mathbf{x}_2), \dots, \phi_j(\mathbf{x}_i)) : \alpha_j^* \neq 0, i, j = 1, 2, \dots, N\}$ are linearly independent, since $\alpha_j^{+*} \geq 0$ (or $\alpha_j^{-*} \geq 0$ associated with $\alpha_j^* \neq 0$ must be the basic variables. \square

B.2.3 KKT Conditions of Constraints Decomposition Method

This proposition is inspired by a Lemma introduced by Bradley, *et al.* [22], in which the authors were giving proof of termination of the LPC algorithm.

Proposition B.1 (Decomposition KKT Conditions). *If \mathbf{z} solves the linear program $\min_{\mathbf{z}} \{\mathbf{c}^T \mathbf{z}\}$ subject to $\{\mathbf{A}\mathbf{z} = \mathbf{b}, \mathbf{z} \geq \mathbf{0}\}$, and $(\mathbf{z}, \boldsymbol{\lambda}) \in \mathbb{R}^{n+m}$ is a primal-dual optimal pair, such that $\boldsymbol{\lambda}_{\mathcal{I}} > \mathbf{0}$, where $\mathcal{I} \subset \{1, 2, \dots, m\}$ and $\boldsymbol{\lambda}_{\mathcal{J}} = \mathbf{0}$, where $\mathcal{J} \subset \{1, 2, \dots, m\}$, $\mathcal{I} \cup \mathcal{J} = \{1, 2, \dots, m\}$, then*

$$\mathbf{z} \in \arg \min_{\mathbf{z}} \{\mathbf{c}^T \mathbf{z}\} \quad \text{s.t.} \quad \{\mathbf{A}_{\mathcal{I}} \mathbf{z} = \mathbf{b}_{\mathcal{I}}, \mathbf{z} \geq \mathbf{0}\}, \quad (\text{B.14})$$

where $\mathbf{A}_{\mathcal{I}}$ has rows of \mathbf{A}_i , for all $i \in \mathcal{I}$, and $\mathbf{b}_{\mathcal{I}}$ has elements b_i , for all $i \in \mathcal{I}$.

Proof. The KKT conditions for a primal-dual optimal pair $(\mathbf{z}, \boldsymbol{\lambda})$ are:

$$\begin{aligned} \mathbf{c} &= \mathbf{A}^T \boldsymbol{\lambda}, \\ \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{z} - \mathbf{b}) &= 0 \\ \mathbf{A}\mathbf{z} - \mathbf{b} &\geq \mathbf{0}, \\ \mathbf{z}, \boldsymbol{\lambda} &\geq \mathbf{0}, \end{aligned}$$

which under the condition $\boldsymbol{\lambda}_{\mathcal{I}} > \mathbf{0}$ imply that

$$\begin{aligned} \mathbf{A}_{\mathcal{I}} \mathbf{z} &= \mathbf{b}_{\mathcal{I}}, \\ \boldsymbol{\lambda}_{\mathcal{J}} &= \mathbf{0}, \\ \mathbf{A}_{\mathcal{J}} \mathbf{z} &\geq \mathbf{b}_{\mathcal{J}}. \end{aligned}$$

It can be claimed [22] that \mathbf{z} is also a solution for (B.14) because the primal-dual

optimal pair $(\mathbf{z}, \boldsymbol{\lambda})$ satisfies the KKT conditions:

$$\mathbf{c} = \mathbf{A}_{\mathcal{I}}^T \boldsymbol{\lambda}_{\mathcal{I}},$$

$$\boldsymbol{\lambda}_{\mathcal{I}} \geq \mathbf{0},$$

$$\mathbf{A}_{\mathcal{I}} \mathbf{z} = \mathbf{b}_{\mathcal{I}},$$

which are necessary and sufficient.

□

Appendix C

Interior Point Methods, Datasets, Performance Metrics, and Additional Tables

C.1 Primal Dual Interior Point Methods for Linear Programming

In this dissertation, an Interior Point Methods (IPM)-based solver is used. The basic idea behind IPM is explained in this section; however, for a more comprehensive explanation, the reader may refer to the text in [203]. More specifically, an *infeasible* IPM within the *path-following* framework is used, which means that the algorithm will follow the path to the solution instead of looking at the vertex of each constraint *e.g.* the simplex method. For promoting a fast rate of convergence a *predictor-corrector* strategy in computing the Newton step was chosen.

An IPM aims to satisfy the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions are a set of equalities and inequalities that are necessary and sufficient conditions to establish optimality of the model. The KKT conditions of a linear

programming problem with primal (2.18) and dual (2.19) are the following:

$$\mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \quad (\text{C.1a})$$

$$\mathbf{A}\mathbf{z} = \mathbf{b}, \quad (\text{C.1b})$$

$$z_i s_i = 0, \quad (\text{C.1c})$$

$$(\mathbf{z}, \mathbf{s}) \geq \mathbf{0}, \quad (\text{C.1d})$$

$$\text{for } i = 1, 2, \dots, n,$$

where the equality $z_i s_i$ implies that one of both variables must be zero. This equality will be referred to as the *complementarity condition*. Note that the problem depends on the variables $(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$, and if the set of solutions $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ satisfy all the conditions, the problem is said to be solved. The vector $(\mathbf{z}^*, \boldsymbol{\lambda}^*, \mathbf{s}^*)$ is known as a *primal-dual solution*.

IPM considers the KKT conditions (C.1a)-(C.1d) as the following function:

$$F(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{pmatrix} \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ \mathbf{A}\mathbf{z} - \mathbf{b} \\ \mathbf{Z}\mathbf{S}\mathbf{1} \end{pmatrix} = \mathbf{0}, \quad (\text{C.2a})$$

$$(\mathbf{z}, \mathbf{s}) \geq \mathbf{0} \quad (\text{C.2b})$$

where $\mathbf{Z} = \text{diag}(z_1, z_2, \dots, z_n)$, and $\mathbf{S} = \text{diag}(s_1, s_2, \dots, s_n)$. The IPM generates a set of solutions $(\mathbf{z}^t, \boldsymbol{\lambda}^t, \mathbf{s}^t)$ at each iteration t . The key idea is to find solutions $(\mathbf{z}^t, \boldsymbol{\lambda}^t, \mathbf{s}^t)$ that satisfy $F(\mathbf{z}^t, \boldsymbol{\lambda}^t, \mathbf{s}^t) = 0$ and more importantly $(\mathbf{z}^t, \mathbf{s}^t)$ being strictly positive, except at the solution where \mathbf{z} or \mathbf{s} may be equal to zero.

Then, IMP uses a quasi-Newton's method to approach the solution of a linear programming problem (2.18). The most remarkable difference between Newton's method and IPM, is that the former does not care of keeping $(\mathbf{z}, \mathbf{s}) \geq \mathbf{0}$, while the latter does. IPM surrounds the current point in a linear model in order to obtain

the step direction $(\Delta \mathbf{z}, \Delta \boldsymbol{\lambda}, \Delta \mathbf{s})$ as follows:

$$J(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) \begin{pmatrix} \Delta \mathbf{z} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = -F(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}), \quad (\text{C.3})$$

where $J(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$ is the Jacobian of $F(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$. Then the step direction (using a predictor-corrector strategy) becomes as follows:

$$\begin{pmatrix} \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{S} & \mathbf{0} & \mathbf{Z} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{z} \\ \Delta \boldsymbol{\lambda} \\ \Delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{ZS}\mathbf{1} - \Delta \mathbf{Z}^{\text{aff}} \Delta \mathbf{S}^{\text{aff}} \mathbf{1} + \sigma \mu \mathbf{1} \end{pmatrix}, \quad (\text{C.4})$$

where $\mathbf{r}_c = \mathbf{A}^T \boldsymbol{\lambda} + \mathbf{s} - \mathbf{c}$ and $\mathbf{r}_b = \mathbf{A} \mathbf{z} - \mathbf{b}$ are residuals, $\Delta \mathbf{Z}^{\text{aff}}, \Delta \mathbf{S}^{\text{aff}}$ are the affine-scaling direction, μ is the duality gap, and σ is an adaptive line-search parameter depending on μ . The new iterate is therefore defined as follows:

$$(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s}) + \alpha(\Delta \mathbf{z}, \Delta \boldsymbol{\lambda}, \Delta \mathbf{s}), \quad (\text{C.5})$$

where $\alpha \in (0, 1]$ is appropriately chosen in order to maintain (\mathbf{z}, \mathbf{s}) strictly positive.

As mentioned before, the predictor-corrector strategy promotes a very fast rate of convergence, which is desirable. In fact, theoretical studies demonstrate that IPM for linear programs is *q-quadratically* convergent to a feasible solution, *i.e.* it is equivalent to the Newton method. Even if $J(\mathbf{z}, \boldsymbol{\lambda}, \mathbf{s})$ is degenerate, the IPM is *q-super-linearly* convergent. In contrast, the simplex method which is typically used in most decomposition strategies in large-scale SVM, is of exponential complexity.

C.2 Data Sets Used in this Dissertation

For an objective numerical assessment, the algorithms developed in this dissertation are evaluated against the commonly used data-sets for classification and regression problems. A benchmarking procedure is performed on most data-sets described in [39, 53, 90, 98, 139, 144, 148, 155, 206].

The well-known *Ripley* dataset problem [138, 156] consists of two classes where the data for each class have been generated by a mixture of two Gaussian distributions. The data is shown in Figure C.1.

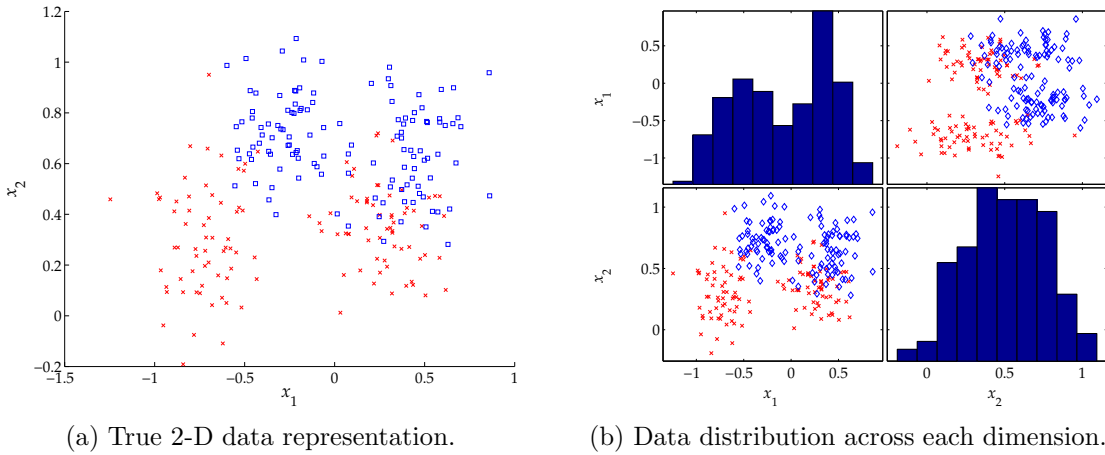


Figure C.1: Ripley dataset. Two classes non-separable.

The *Wine* dataset [64, 107] contains results of wine chemical analysis within the Italy region but was derived from three different vines. The analysis consists of 13 attributes of two different groups of wine. This dataset is part of the UCI machine-learning repository [65]. For displaying purposes, we used the dimensionality reduction technique called t-SNE [194, 195]. Figure C.2 shows the dataset using t-SNE to reduce the data to two and three dimensions.

ADA is a marketing-related dataset [108]. The goal of ADA is to discover high revenue people from census data. This is a two-class classification problem. The raw data from the census bureau is known as the Adult database [98, 148] in the UCI machine-learning repository [65]. The 48 features include age, workclass, education,

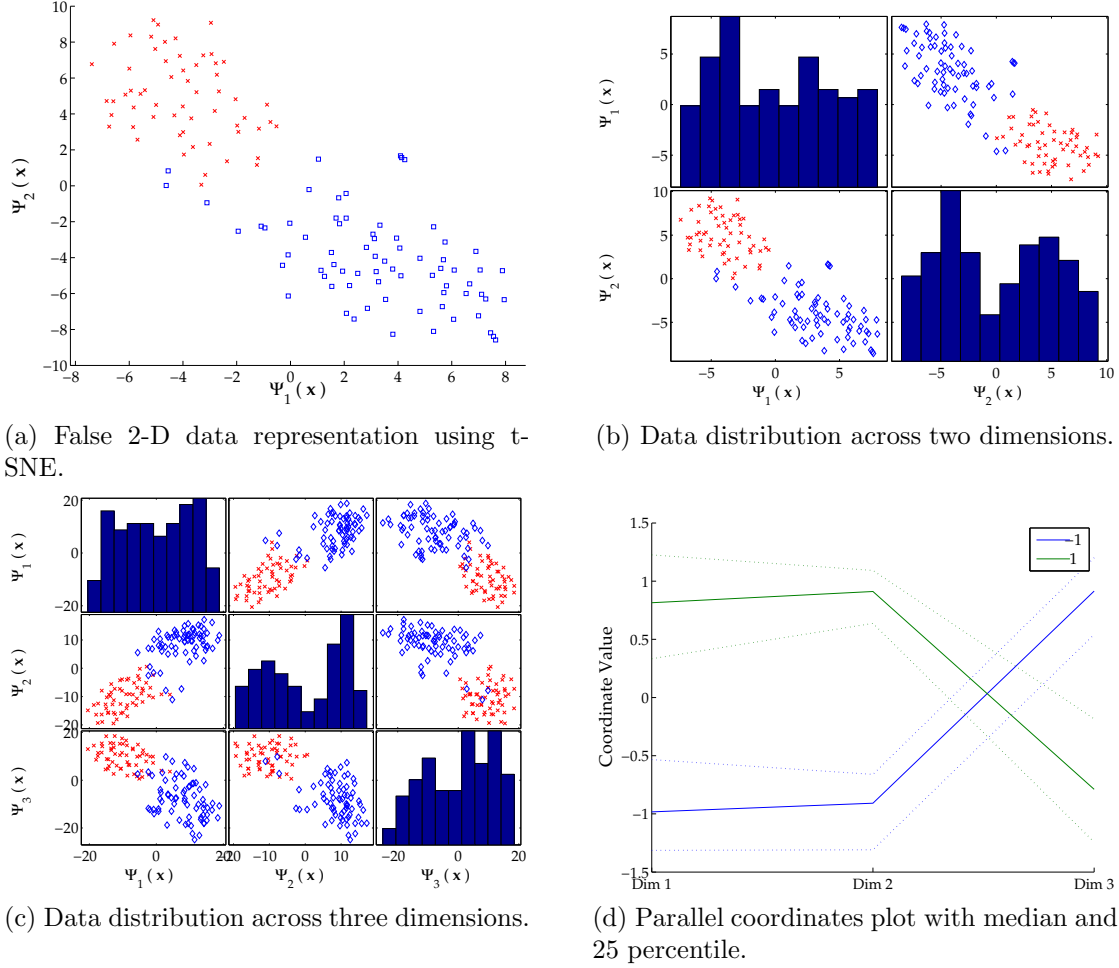


Figure C.2: Wine dataset. Two classes non-separable.

education, marital status, occupation, native country, etc. An illustration of the non-separability of the data is shown in Figure C.3.

GINA is a digit recognition-related dataset that is commonly known as the MNIST database of handwritten digits [112]. *GINA* aims to provide features for handwritten digit recognition [39, 148]. The problem consists of separating two-digit odd numbers from two-digit even numbers. Only the unit digit is informative for that task; therefore, at least $\frac{1}{2}$ of the features are distracters. Additionally, the pixels that are almost always blank were removed and the pixel order was randomized to hide the feature identity. This is a two class classification problem with non-sparse

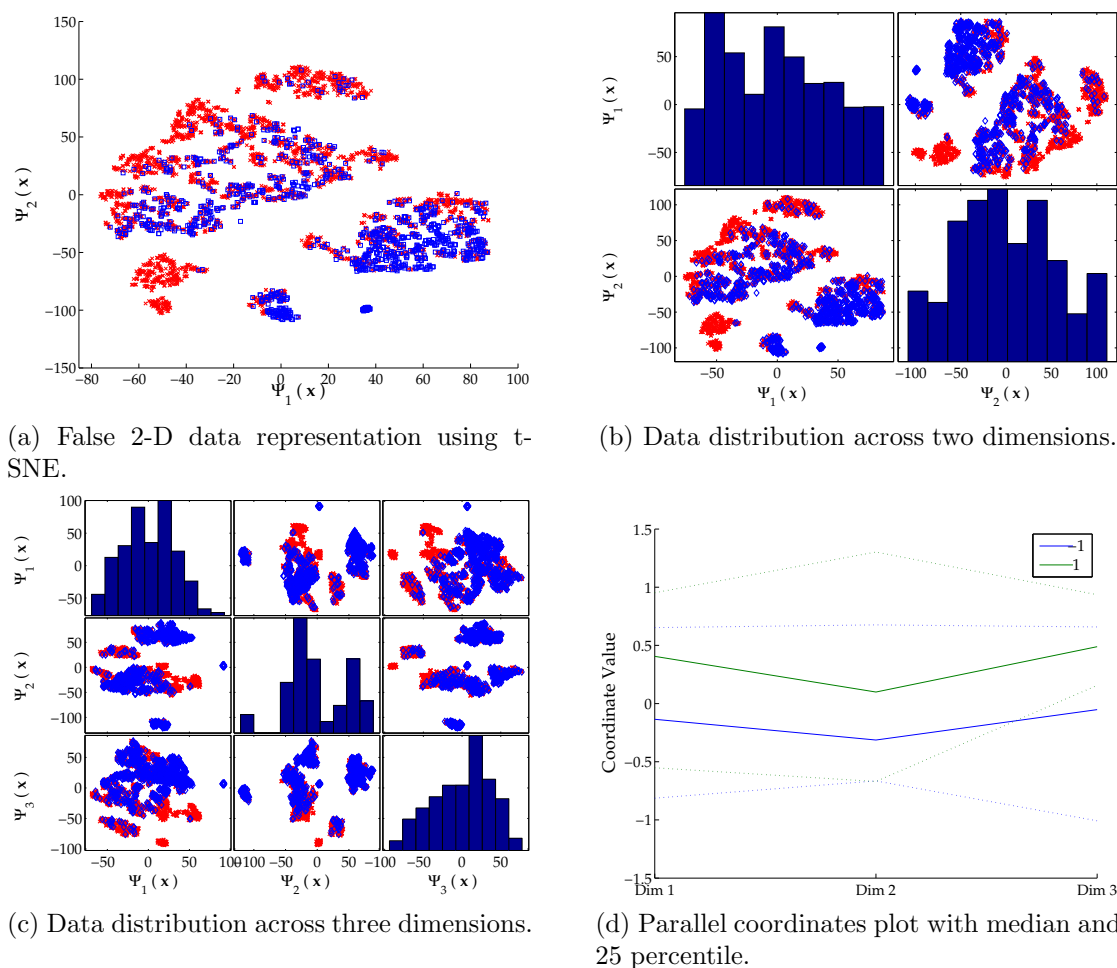


Figure C.3: ADA dataset. Two classes non-separable.

continuous input variables, in which each class is composed of several clusters. It is a problem with heterogeneous classes. The representation of the data is shown in Figure C.4.

HIVA is a dataset related to HIV infections. The goal of *HIVA* is to provide features for prediction of active compounds within an AIDS HIV infection. The dataset represents a two-class classification problem (active vs. inactive) consisting of 2000 sparse binary input variables. The variables represent properties of the molecule inferred from its structure. The problem is to relate structure to activity *i.e.*, a quantitative structure-activity relationship (QSAR) problem, to screen new

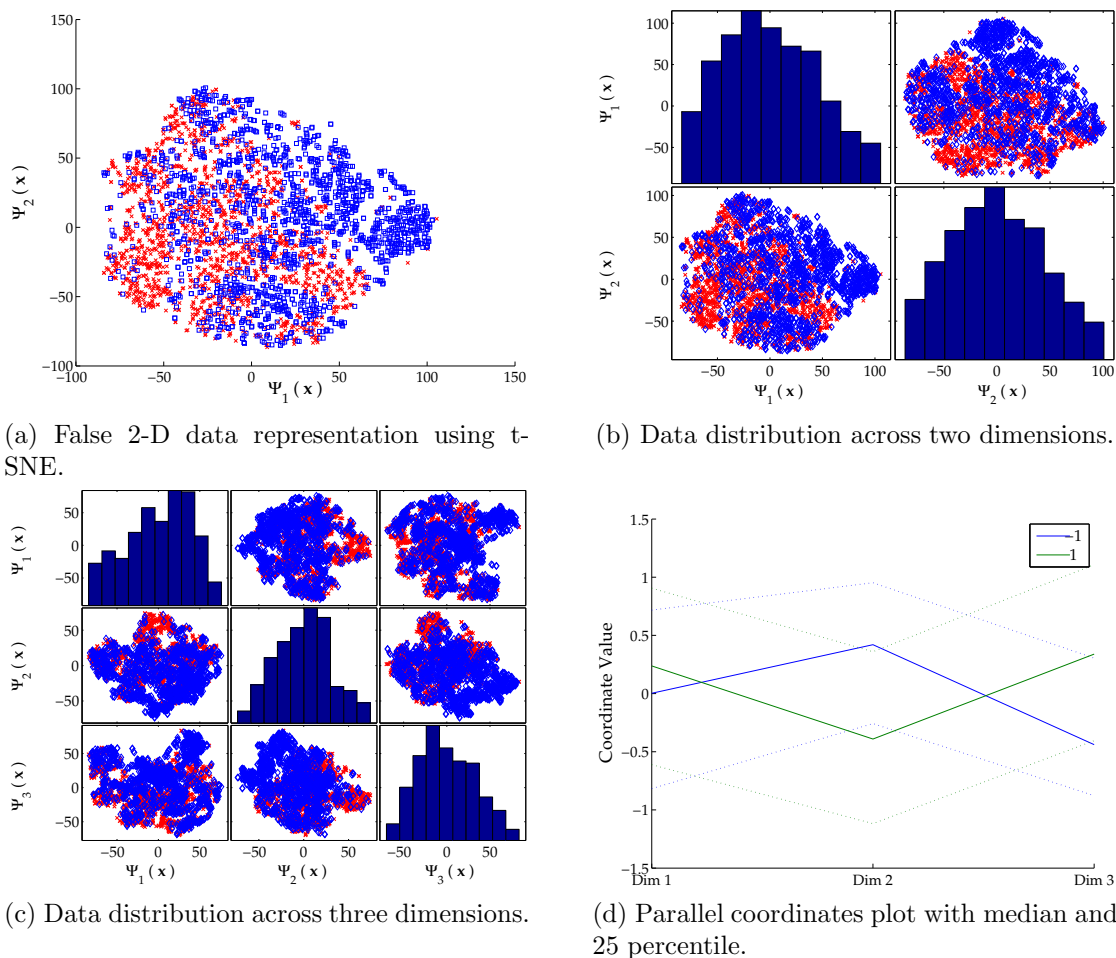


Figure C.4: GINA dataset. Two classes non-separable.

compounds before actually testing them, *i.e.*, a high-throughput screening (HTS) problem. The data is made available by the National Cancer Institute (NCI), and has been used in [26]. A representation of the dataset appears in Figure C.5.

NOVA is a text classification dataset. The data of *NOVA* come from the UCI repository [65] which is also known as Twenty-Newsgroup dataset [97]. Each text to classify corresponds to email text. The features consist of a sparse binary representation of a vocabulary of approximately 17,000 words. Figure C.6.

SYLVA is an ecology-related dataset that is part of the UCI repository [65] under the name of Covertype Data Set [17,39]. The *SYLVA* dataset aims to provide features

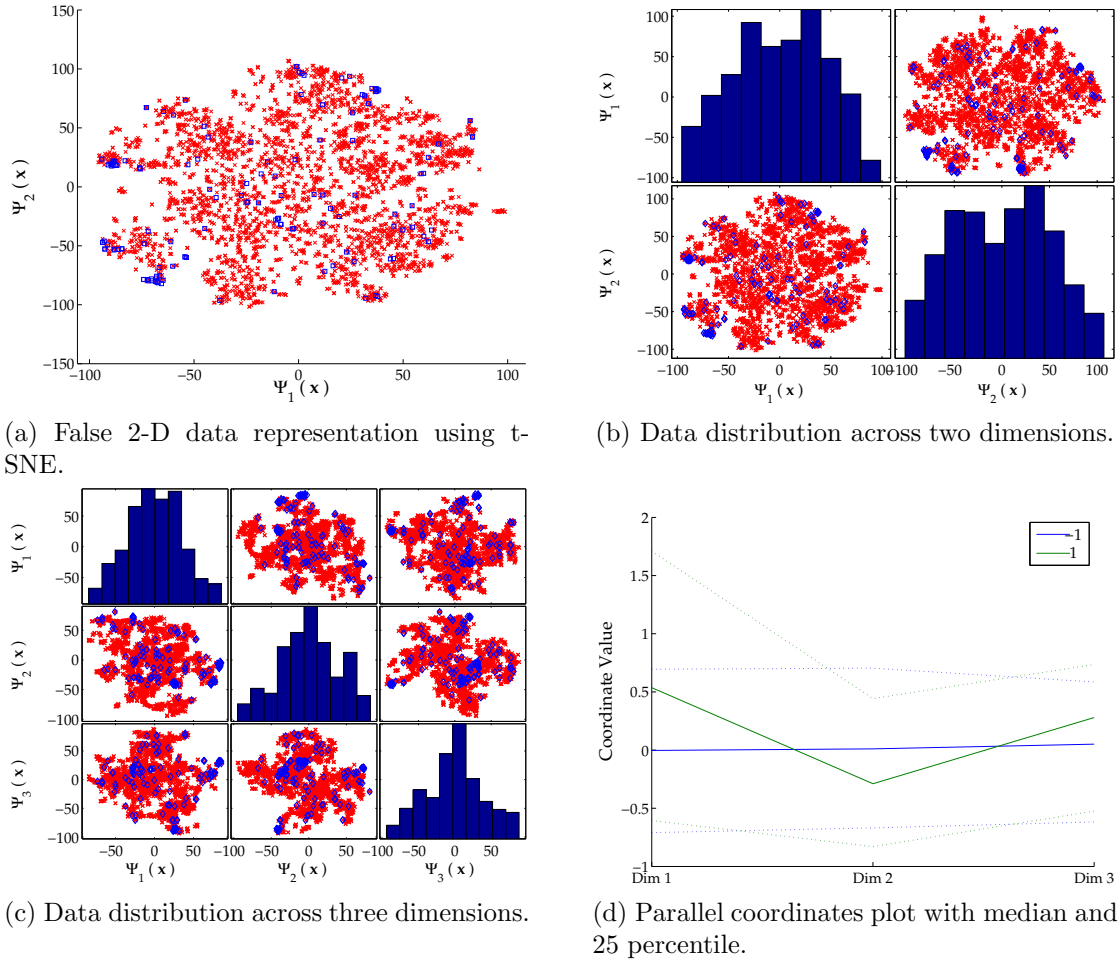


Figure C.5: HIVA dataset. Two classes non-separable.

for forest cover type classification. The data is obtained from 30×30 meter cells by the US Forest Service (USFS) Region 2 Resource Information System (RIS). It represents a two-class classification problem, that classifies Ponderosa pine against everything else. The features consists of 216 input variables. Each pattern is composed of four records: two records matching the target and two records chosen at random. Thus $\frac{1}{2}$ of the features are distracters. The representation of the dataset is depicted in Figure C.7.

The *Iris* dataset is perhaps the best known database to be found in the pattern recognition literature and is also part of the UCI dataset [65]. The dataset contains

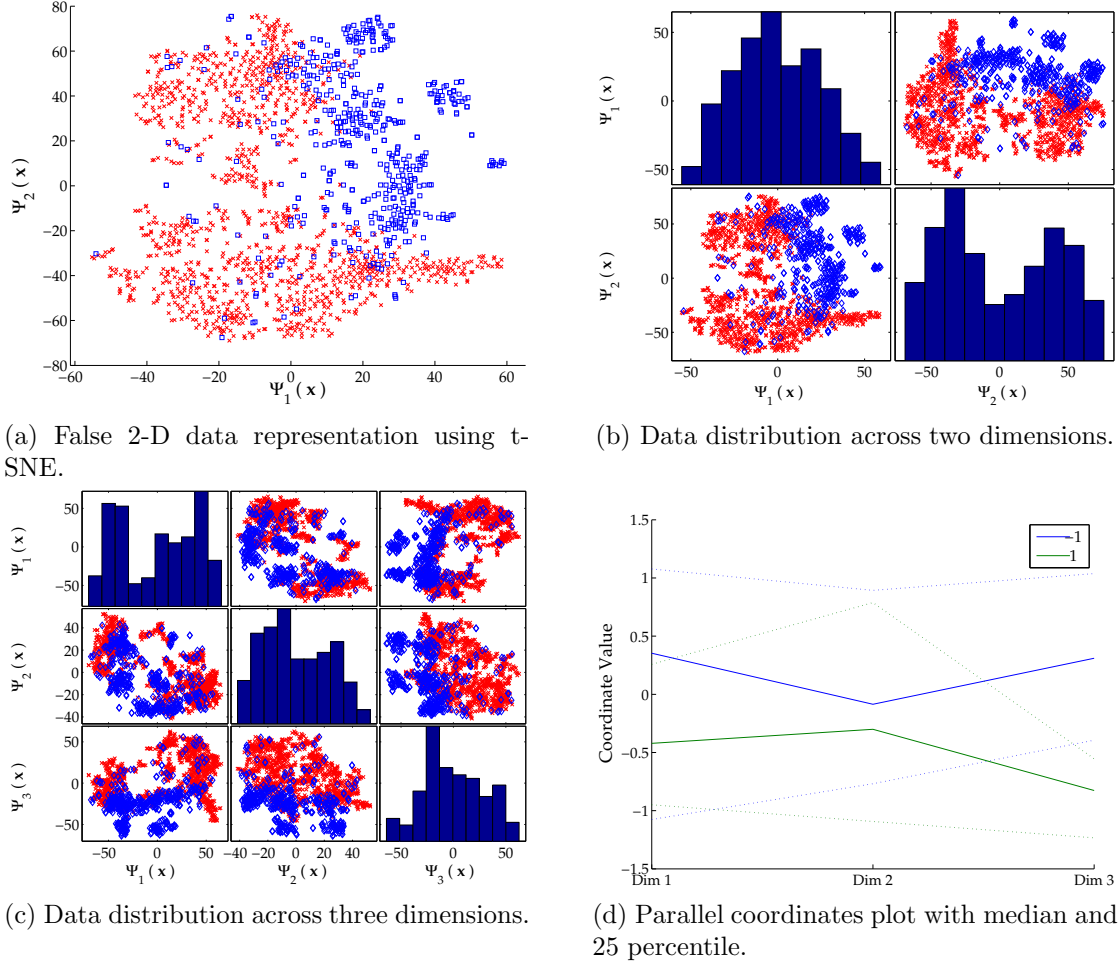


Figure C.6: NOVA dataset. Two classes non-separable.

three classes of 50 instances each, where each class refers to a type of Iris plant. One of the classes is linearly separable from the rest, which are not linearly separable [63, 77] as illustrated in Figure C.8.

The *Spiral* dataset is a synthetic dataset that consist of a two class problem with an extremely non-linear decision surface, and is typically used to test the ability of classifiers in finding such difficult decision functions [206].

Similarly, the remaining datasets are synthetic. The *Synthetic S* is a non-linearly separable three-class problem whose classes are normally distributed as shown in Figure C.10. The *Synthetic NS* is identical, however, the classes are non-separable.

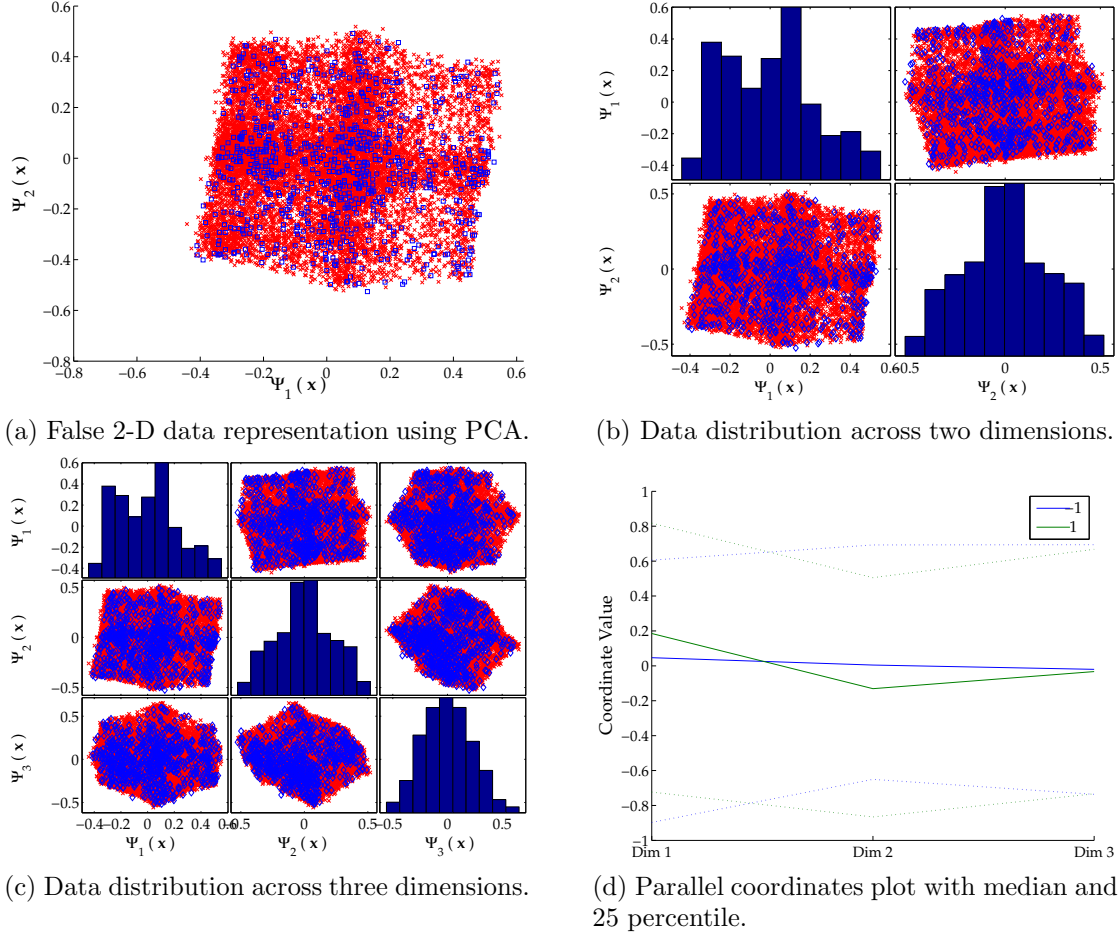


Figure C.7: SYLVA dataset. Two classes non-separable.

The two last examples are for regression purposes. The datasets objective is to fit the “sinc” function, which is a typical function to approximate [144]. The $f(x) = \text{sinc}(x)$ consists of unevenly sampled points from the sinc function. Similarly, $f(x) = \text{sinc}(x) \times \pi$ consists of unevenly spaced points from the sinc function that are affected by multiplicative white Gaussian noise (WGN); this makes it a very difficult function to fit. Figure C.12 illustrates both cases.

The summary of the properties of these datasets are shown in Table C.1. Note that the simulations include classification in two and multiple classes, as well as regression problems. These datasets are widely used in the pattern recognition community, except by the “MODIS” and “Power Load”. The MODIS dataset was ob-

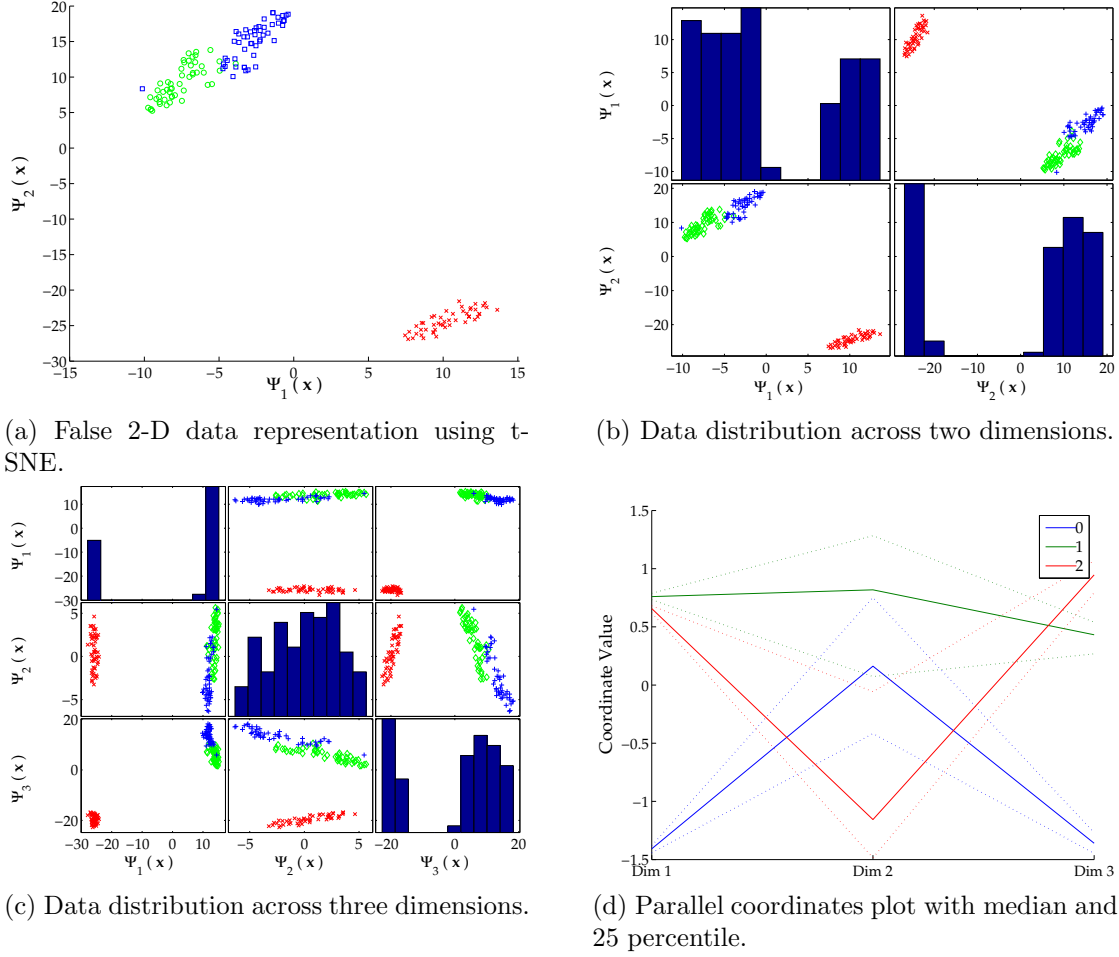


Figure C.8: Iris dataset. Two classes non-separable.

tained from a dust storm detection project developed by the author while at NASA Goddard Space Flight Center. And the “Power Load” dataset is a relatively new dataset exploited in MATLAB[®] tutorials for neural networks-based regression, and regression trees. The MODIS dataset is illustrated in Figure C.13 while the Power Load dataset is depicted in Figure C.14 and C.15.

A double line in Table C.1 indicates that the datasets are synthetic. That is, they were generated to test specific capabilities of classifiers. The spiral dataset is a highly non-linear dataset in which many classifiers fail. The datasets Synthetic S and NS, are three-class separable and non-separable problems respectively, which

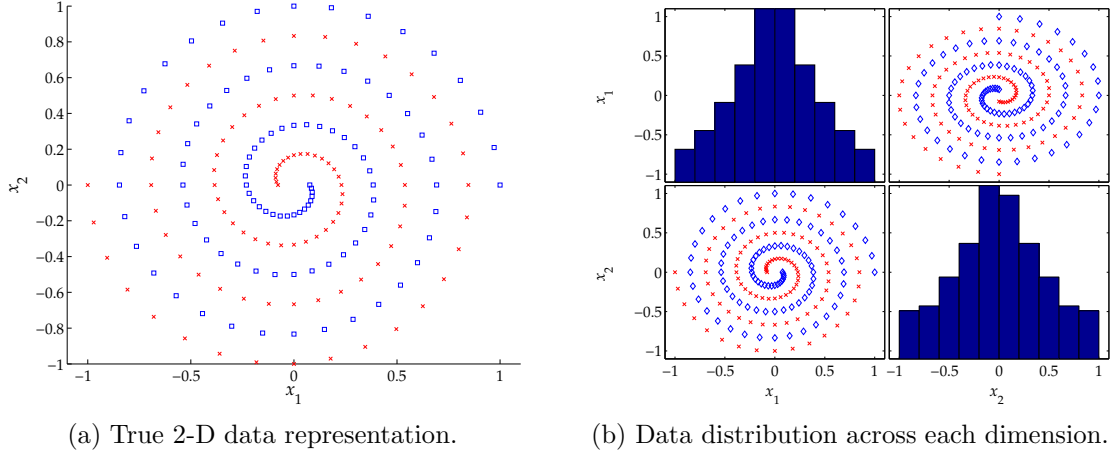


Figure C.9: Spiral dataset. Two classes separable but with a highly non-linear decision function.

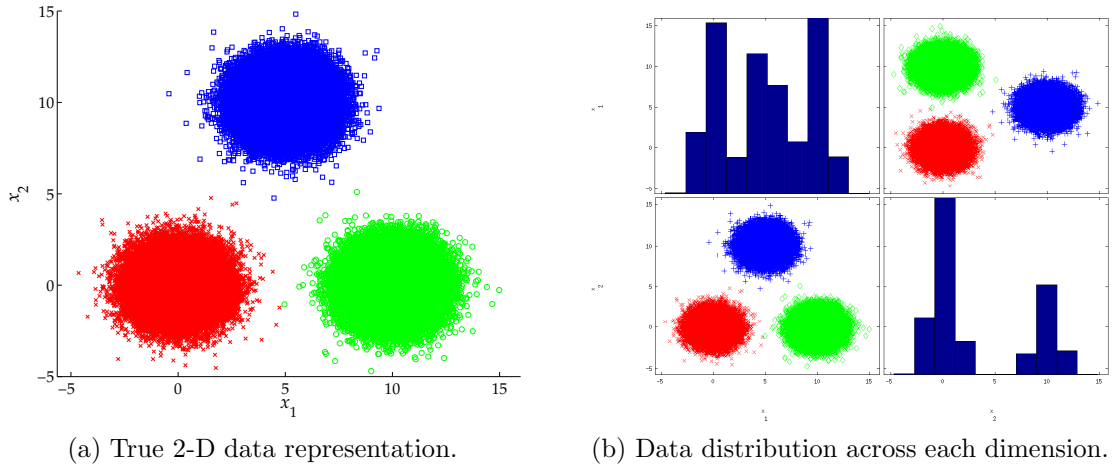


Figure C.10: Synthetic S dataset. Three classes separable.

are normally distributed in \mathbb{R}^2 . The π symbol in the last row, states for white Gaussian noise (WGN). Thus, it means that the sinc function was contaminated with multiplicative WGN.

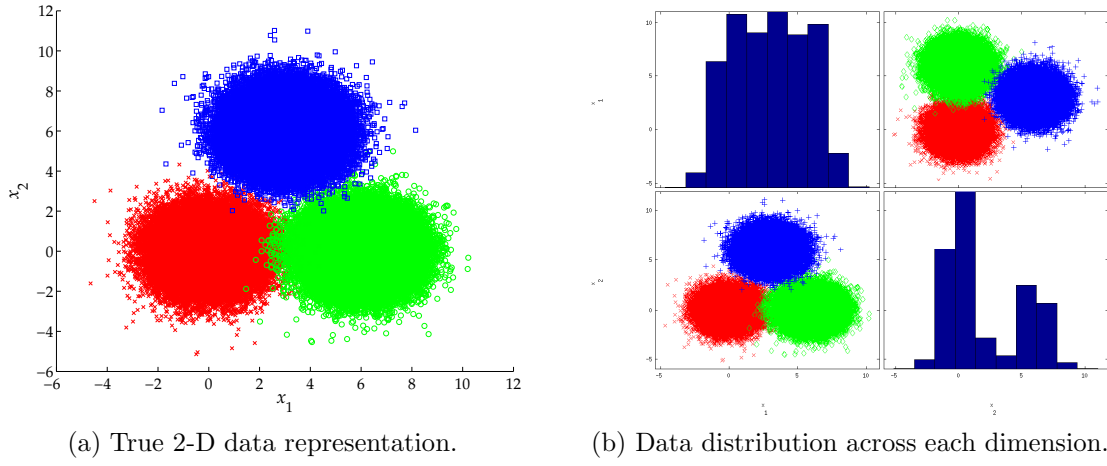
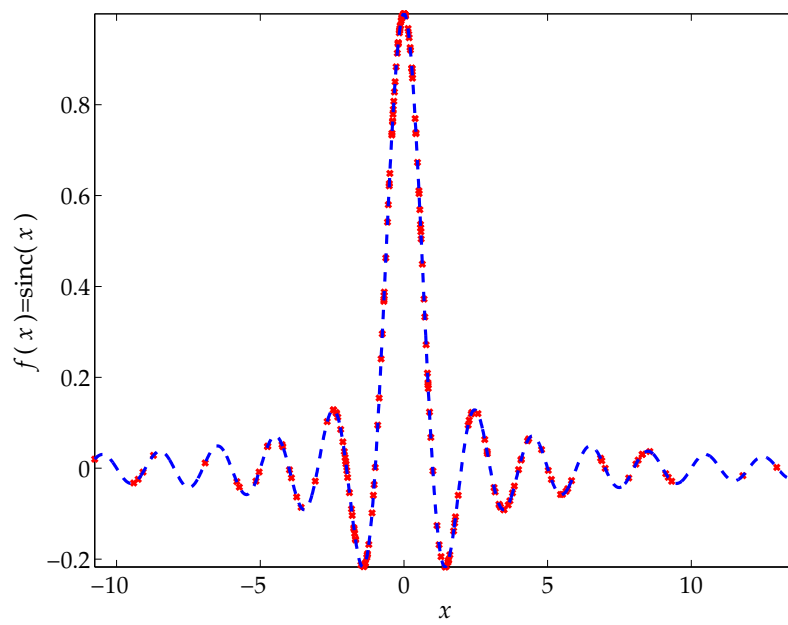


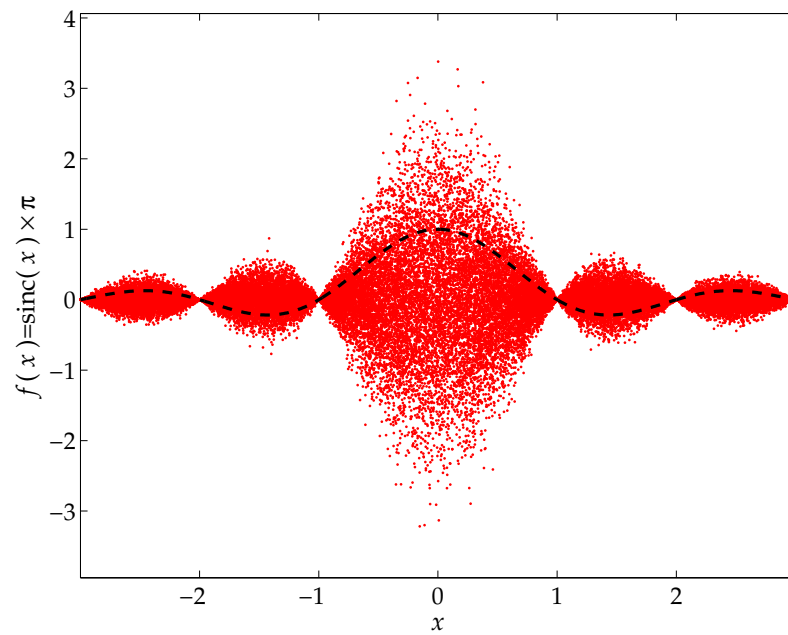
Figure C.11: Synthetic NS dataset. Three classes non-separable.

Table C.1: Summary of the Dimensions and Properties of the Datasets.

Dataset	Classes	Features M	Training N	Testing N^*	Reference
Ripley	2	2	250	1,000	[138]
Sonar	2	60	104	104	[71]
Wine	2	13	110	20	[107]
ADA	2	48	4,147	415	[98, 148]
GINA	2	970	3,153	315	[39, 148]
HIVA	2	1,617	3,845	384	[26]
NOVA	2	16,969	1,754	175	[26]
SYLVA	2	216	13,086	1,308	[26, 39]
Iris	3	4	130	20	[128]
MODIS	4	4	374,566	85+ million	[157]
Power Load	\mathbb{R}	8	35,064	8,784	[103]
Spiral	2	2	200	101	[206]
$f(x) = \text{sinc}(x)$	\mathbb{R}	1	200	200	[144]
Synthetic S	3	2	3 million	3 million	—
Synthetic NS	3	2	3 million	3 million	—
$f(x) = \text{sinc}(x) \times \pi$	\mathbb{R}	1	1 million	1 million	—

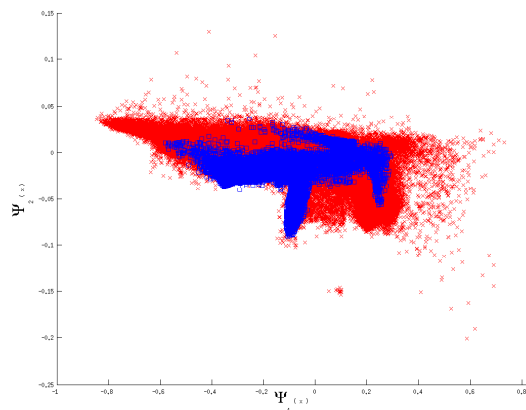


(a) Plot of the $f(x) = \text{sinc}(x)$ dataset.

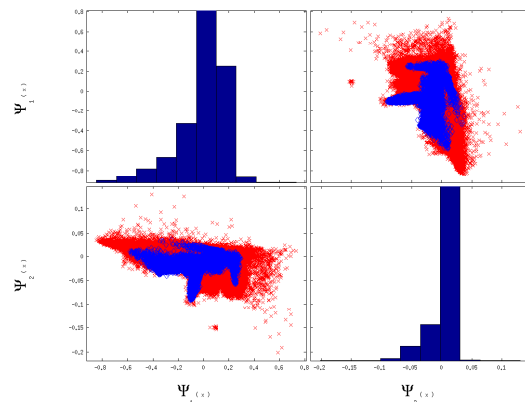


(b) Plot of the $f(x) = \text{sinc}(x) \times \pi$ dataset.

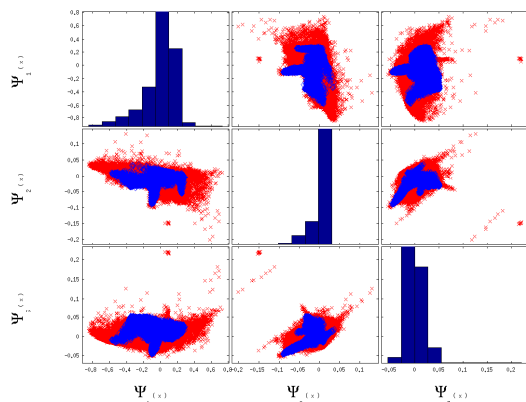
Figure C.12: Sinc function datasets. True function to approximate and the data provided.



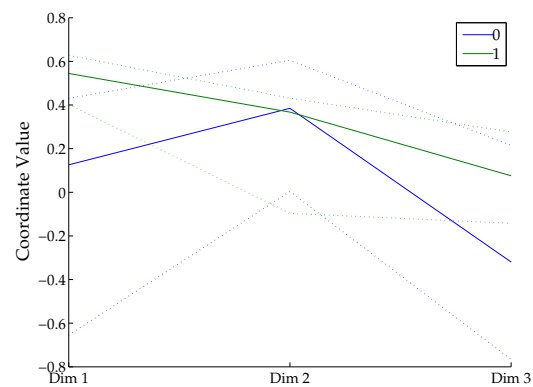
(a) False 2-D data representation using PCA.



(b) Data distribution across two dimensions.

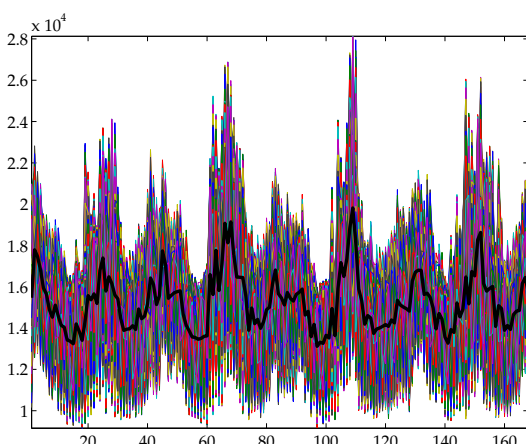


(c) Data distribution across three dimensions.

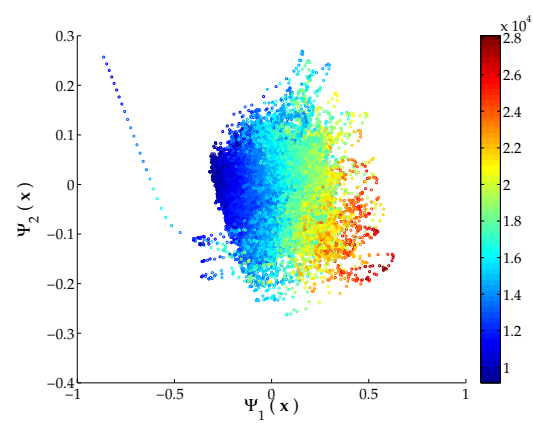


(d) Parallel coordinates plot with median and 25 percentile.

Figure C.13: MODIS dataset. Two classes non-separable.



(a) Dataset targets with average in dark color.



(b) False 2-D data representation using t-SNE.

Figure C.14: Power Load dataset. Regression problem.

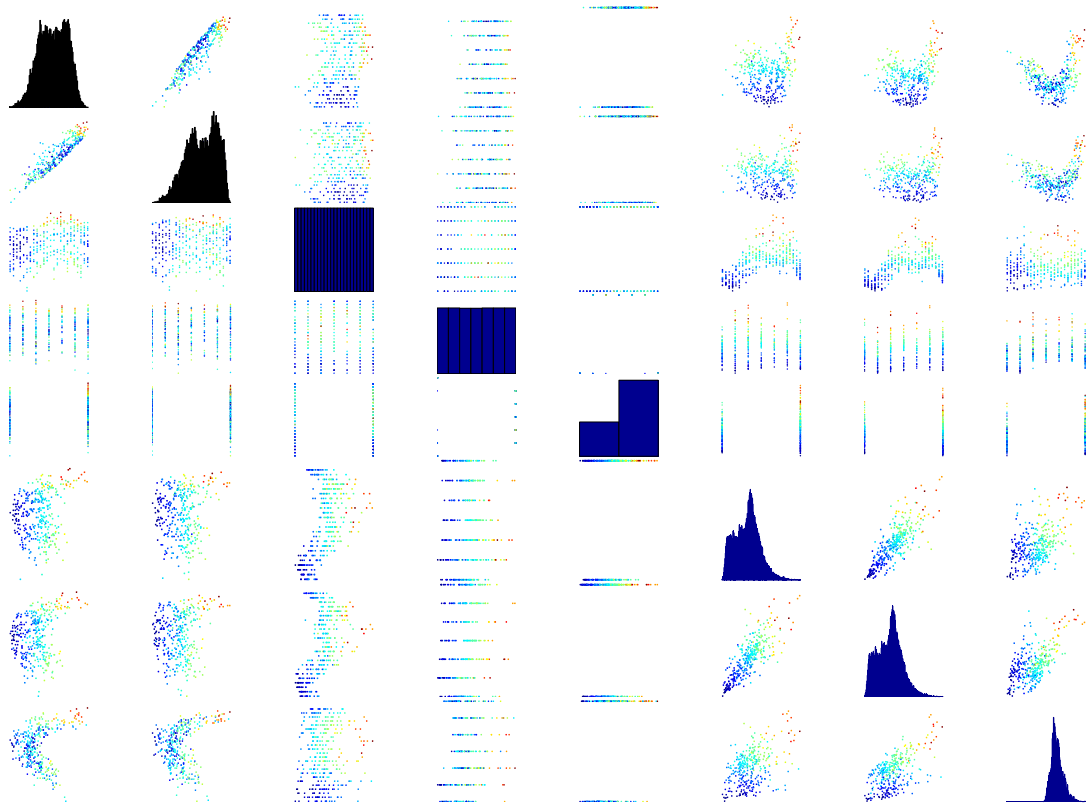


Figure C.15: Power Load dataset. Sample statistical distribution across dimensions.

C.3 Performance Metrics

C.3.1 Regression

The most common parameters to measure good performance in regression are the following: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Normalized Root Mean Squared Error (NRMSE). These parameters are defined as follows:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - d_i|, \quad (\text{C.6})$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - d_i)^2}, \quad (\text{C.7})$$

$$\text{NRMSE} = \frac{1}{\sigma} \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - d_i)^2}, \quad (\text{C.8})$$

where y is the actual output of the classifier when the input vector \mathbf{x} is presented at its input, and sigma is the standard deviation of y .

Other error functions to be used are two: sum of squared error (SSE), and statistical metrics (STAT). The SSE metric is given by the following equation:

$$\text{SSE} = \sum_{i=1}^N (y_i - d_i)^2. \quad (\text{C.9})$$

Statistical properties of the residual error are based on the residual error: $y_i - d_i$. From estimation theory it is known that if one have the residual error expected value equal to zero, and a unit variance, one have achieved the least-squares solution to the regression problem, either linear or non-linear. Furthermore, it is understood that as the variance of the residual error approaches zero, the regression problem is

better solved. Let us denote the expected value of the residual error as follows:

$$\mu = \mathcal{E}[y_i - d_i] = \frac{1}{N} \sum_{i=1}^N y_i - d_i, \quad (\text{C.10})$$

and the variance of the residual error $\text{Var}[y_i - d_i - \mu]$ is defined as follows:

$$\sigma^2 = \mathcal{E}[y_i - d_i - \mu]^2 = \frac{1}{N-1} \sum_{i=1}^N (y_i - d_i - \mu)^2, \quad (\text{C.11})$$

from where it is desired that $\mu, \sigma^2 \rightarrow 0$. Hence, the second error metric is defined as follows:

$$\text{STAT} = \sigma + |\mu|. \quad (\text{C.12})$$

C.3.2 Classification

This document shows the usage of standard metrics for classification. These metrics exist provided a training set (input) $\{(\mathbf{x}_i, d_i)\}_{i=1}^N$ and from there, exist four possible outcomes. Suppose one know the input (\mathbf{x}_i) correspond to the class 0 ($d_i = 0$) and it is classified as 0 ($y_i = 0$), then one call this a true positive (TP); if it is classified as 1 ($y_i = 1$), it becomes a false negative (FN). If the input is not 0 ($d_i = 1$) and it is not classified as 0 ($y_i = 1$), it is counted as a true negative (TN); if it is classified as 0, it is counted as a false positive (FP). This is exemplified Table C.2, using a confusion matrix.

Table C.2: Illustration of TP, FP, TN, and FN using a confusion matrix.

Known Class, d	SVR's Output, y	
	0	1
0	TP	FN
1	FP	TN

For any given classification method and a number of j classes, a j -by- j confusion

matrix exists. This matrix be constructed representing the dispositions of the test set. The numbers along the major diagonal represent the correct decisions made, and the numbers off this diagonal represent the errors (the confusion) between the different classes. An example of a multi-class confusion matrix and its interpretation for two classes is shown in Table C.3 and Table C.4.

Table C.3: Illustration of TP, FP, TN, and FN for class 0, using a multi-class confusion matrix.

Known Class, d	SVR's Output, y				
	0	1	2	\cdots	j
0	TP	FN	FN	FN	FN
1	FP	TN	FN	FN	FN
2	FP	FN	TN	FN	FN
\vdots	FP	FN	FN	TN	FN
j	FP	FN	FN	FN	TN

Table C.4: Illustration of TP, FP, TN, and FN for class 2, using a multi-class confusion matrix.

Known Class, d	SVR's Output, y				
	0	1	2	\cdots	j
0	TN	FN	FP	FN	FN
1	FN	TN	FP	FN	FN
2	FN	FN	TP	FN	FN
\vdots	FN	FN	FP	TN	FN
j	FN	FN	FP	FN	TN

The confusion matrix is very useful in the estimation of the performance metrics used in this research. These metrics are defined as follows:

$$\text{TP rate} = \text{TPR} = \frac{TP}{TP + FN}, \quad (\text{C.13a})$$

$$\text{FP rate} = \text{FPR} = \frac{FP}{FP + TN}, \quad (\text{C.13b})$$

$$\text{Accuracy} = \text{ACC} = \frac{TP + TN}{TP + FN + FP + TN}, \quad (\text{C.13c})$$

$$\text{Specificity} = \text{SPC} = \frac{TN}{FP + TN}, \quad (\text{C.13d})$$

$$\text{Positive Predictive Value} = \text{PPV} = \frac{TP}{TP + FP}, \quad (\text{C.13e})$$

$$\text{Negative Predictive Value} = \text{NPV} = \frac{TN}{TN + FN}, \quad (\text{C.13f})$$

$$\text{False Discovery Rate} = \text{FDR} = \frac{FP}{FP + TP}, \quad (\text{C.13g})$$

$$\begin{aligned} \text{Matthews Correlation Coefficient} = \text{MCC} = \\ \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}, \end{aligned} \quad (\text{C.13h})$$

$$F_1 - \text{Score} = 2 \times \frac{\left(\frac{TP}{TP+FP}\right) \times \left(\frac{TP}{TP+FN}\right)}{\left(\frac{TP}{TP+FP}\right) + \left(\frac{TP}{TP+FN}\right)}. \quad (\text{C.13i})$$

$$\begin{aligned} \text{Balanced Error Rate} = \text{BER} = \\ \frac{1}{2} \left(\frac{FP}{TN + FP} + \frac{FN}{FN + TP} \right) \end{aligned} \quad (\text{C.13j})$$

$$\text{Estimate of Scaled Error Rate} = \text{ESER} = \sum_{i=1}^N \zeta \Psi\{(y_i - d_i) - 0.5\} \quad (\text{C.13k})$$

Remark: In the literature, one might also find the above measures with different names; *i.e.* TPR is also known as Sensitivity, SPC is also known as TN rate, PPV is also known as Precision, and the F_1 –Score is also known as the F –Measure.

In literature, one can find other typical performance metrics used in this research, such as the Receiver Operating Characteristics [60] (ROC). The ROC graphs are two-dimensional graphs in which TP rate is plotted on the *ordinates* axis and FP rate is plotted on the *abscissas* axis, as shown in Figure C.16, constructed from the data [60] in Table C.5.

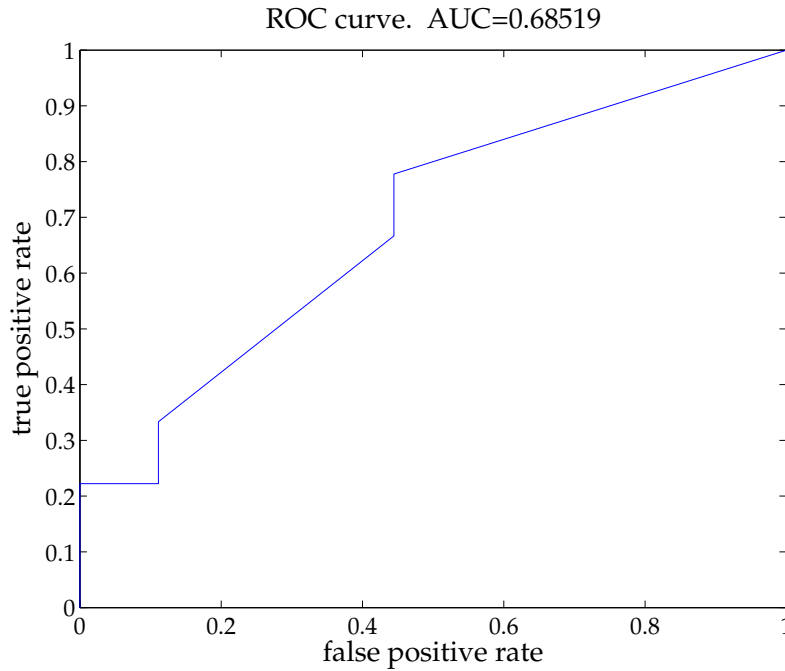


Figure C.16: An example of the ROC graph, and the corresponding AUC for the example shown in Table .

A ROC graph shows the relative trade-offs between benefits (true positives) and costs (false positives). The SVR methods here will be treated as discrete classifiers

Table C.5: Classification targets and predicted values. Borrowed from [60].

$d = ($	1	1	0	1	1	1	0	0	1	0	1	0	1	0	0	0	1	0	$)^T$
$y = ($.9	.8	.7	.6	.5	.5	.5	.5	.5	.5	.4	.3	.3	.3	.3	.3	.3	.3	$)^T$

that outputs only a class label. Each discrete classifier produces an (fp rate, tp rate) pair corresponding to a single point in the ROC space. Several aspects in the ROC space are important to note. The lower left point (0; 0) represents the strategy of never issuing a positive classification: a classifier commits no false positive errors but also gains no true positives. The opposite strategy, of unconditionally issuing positive classifications, is represented by the upper right point (1; 1). The point (0; 1) represents perfect classification. Informally, one point in ROC space is better than another if it is to the northwest (tp rate is higher, fp rate is lower, or both) with respect to the first. Classifiers appearing on the left hand-side of an ROC graph, near the ordinates axis, may be thought of as "conservative": they make positive classifications only with strong evidence so they make few false positive errors, but they often have low true positive rates as well. Classifiers on the upper right-hand side of a ROC graph may be thought of as "liberal": they make positive classifications with weak evidence so they classify nearly all positives correctly, but they often have high false positive rates. The diagonal line represents the strategy of randomly guessing a class.

For simplicity in the comparison of the proposed method among others, the ROC performance is expressed in a simple scalar value. A common method is to calculate the area under the ROC curve, abbreviated AUC. Since the AUC is a portion of the area of the unit square, its value will always be between zero and one. However, because random guessing produces a diagonal line between (0; 0) and (1; 1), which has an area of 0.5, no realistic classifier should have an AUC less than 0.5.

Although ROC curves may be used to evaluate classifiers, care should be taken

when using them to make conclusions about classifier superiority. Some researchers have assumed that an ROC graph may be used to select the best classifiers simply by graphing them in ROC space and seeing which ones dominate. This is misleading; it is analogous to taking the maximum of a set of accuracy figures from a single test set. Therefore, the AUC of the proposed LP-SVR model is computed for the different benchmark classification problems. A ROC plot with the result of each problem will be included in this document.

Table C.6: Summary of Performance Metrics and Their Interpretation.

Metric	Interval	Desired
MAE	\mathbb{R}^+	The smallest value.
RMSE	\mathbb{R}^+	The smallest value.
NRMSE	\mathbb{R}^+	The smallest value.
SSE	\mathbb{R}^+	The smallest value.
STAT	\mathbb{R}^+	The smallest value.
TP	$[0, \mathbb{Z}]$	The largest integer.
TN	$[0, \mathbb{Z}]$	The largest integer.
FP	$[0, \mathbb{Z}]$	Zero.
FN	$[0, \mathbb{Z}]$	Zero.
TPR	$[0, 1]$	One.
FPR	$[0, 1]$	Zero.
ACC	$[0, 1]$	One.
SPC	$[0, 1]$	One.
PPV	$[0, 1]$	One.
NPV	$[0, 1]$	One.
FDR	$[0, 1]$	Zero.
MCC	$[-1, 1]$	One.
F_1 -Score	$[0, 1]$	One.
BER	$[0, 1]$	Zero.
ESER	$[0, 1]$	Zero.
AUC	$[0, 1]$	One.
AUHC	$[0, 1]$	One.

C.4 Additional Tables from Chapter 3

Table C.7 shows the false positive counts and Table C.8 the false negative counts. The data is shown in proportion to the total number of samples. The proposed model is comparable to the other classifiers in both tables, and shows better results for larger problems. Table C.9 shows a comparison of the true positives rate

Table C.7: False Positives in Proportion to the Dataset Size.

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.039	0.036	0.032	0.035	0.041
Wine	0	0	0	0	0.05
ADA	0.022	0.036	—	0.034	0.031
GINA	0.003	0	—	0.003	0
HIVA	0.008	0.008	—	—	—
NOVA	0	0	0	—	—
SYLVA	0	0.001	—	—	0.002
Iris	0	0	0	0	0
Spiral	0	0	0	0	0
Synthetic S	0.011	0.01	—	—	—
Synthetic NS	0.048	0.044	—	—	—
Avg.	0.087	0.012	—	—	—

(a.k.a. sensitivity). In the average case, the proposed model shows better average performance, followed by the LS SVM.

Table C.10 shows a comparison of the false positives rate. The proposed approach exhibits low false positives rate, although is not the lowest.

Table C.11 shows a comparison of the specificity (a.k.a. true negatives rate).

Table C.12 shows a comparison of the positive predictive value (a.k.a. precision).

Table C.13 shows a comparison of the negative predictive value.

Table C.14 shows a comparison of the false discovery rate.

Table C.15 shows a comparison of the Mathews correlation coefficient.

Table C.16 shows a comparison of the F_1 -score (a.k.a. F -Measure).

Table C.8: False Negatives in Proportion to the Dataset Size.

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.045	0.049	0.058	0.059	0.051
Wine	0	0	0	0	0
ADA	0.137	0.113	—	0.123	0.13
GINA	0	0.003	—	0.006	0.003
HIVA	0.122	0.122	—	—	—
NOVA	0	0	0	—	—
SYLVA	0.002	0	—	—	0.002
Iris	0	0	0.15	0.15	0.15
Spiral	0	0	0	0	0.02
Synthetic S	0.022	0.02	—	—	—
Synthetic NS	0.095	0.089	—	—	—
Avg.	0.038	0.036	—	—	—

Table C.9: True Positives Rate

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.91	0.902	0.884	0.882	0.898
Wine	1	1	1	1	1
ADA	0.817	0.849	—	0.837	0.827
GINA	1	0.994	—	0.988	0.994
HIVA	0.873	0.873	—	—	—
NOVA	1	1	1	—	—
SYLVA	0.998	1	—	—	0.998
Iris	1	1	0.998	0.998	0.998
Spiral	1	1	1	1	0.979
Synthetic S	0.979	0.98	—	—	—
Synthetic NS	0.969	0.971	—	—	—
Avg.	0.959	0.961	—	—	—

Table C.17 shows a comparison of the estimate of scaled error rate.

Table C.18 shows a comparison of the area under the ROC curve.

Table C.19 shows a comparison of the area under the ROC curve convex hull.

Table C.20 shows a comparison of the root mean squared error and normalized

Table C.10: False Positives Rate

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.078	0.072	0.064	0.07	0.082
Wine	0	0	0	0	0.016
ADA	0.087	0.146	—	0.136	0.126
GINA	0.006	0	—	0.006	0
HIVA	0.214	0.214	—	—	—
NOVA	0	0	0	—	—
SYLVA	0	0.013	—	—	0.025
Iris	0	0	0	0	0
Spiral	0	0	0	0	0
Synthetic S	0.005	0.005	—	—	—
Synthetic NS	0.008	0.007	—	—	—
Avg.	0.036	0.041	—	—	—

Table C.11: Specificity

Dataset	Classifiers				
	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.922	0.928	0.936	0.93	0.918
Wine	1	1	1	1	0.984
ADA	0.913	0.854	—	0.864	0.874
GINA	0.994	1	—	0.994	1
HIVA	0.786	0.786	—	—	—
NOVA	1	1	1	—	—
SYLVA	1	0.988	—	—	0.975
Iris	1	1	1	1	1
Spiral	1	1	1	1	1
Synthetic S	0.995	0.995	—	—	—
Synthetic NS	0.992	0.993	—	—	—
Avg.	0.964	0.959	—	—	—

root mean squared error.

Table C.12: Positive Predictive Value

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.921	0.926	0.932	0.926	0.916
Wine	1	1	1	1	0.977
ADA	0.966	0.946	—	0.949	0.952
GINA	0.994	1	—	0.994	1
HIVA	0.991	0.991	—	—	—
NOVA	1	1	1	—	—
SYLVA	1	0.999	—	—	0.998
Iris	1	1	1	1	1
Spiral	1	1	1	1	1
Synthetic S	0.989	0.99	—	—	—
Synthetic NS	0.984	0.985	—	—	—
Avg.	0.986	0.985	—	—	—

Table C.13: Negative Predictive Value

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.911	0.904	0.89	0.887	0.9
Wine	1	1	1	1	1
ADA	0.623	0.652	—	0.636	0.625
GINA	1	0.994	—	0.987	0.994
HIVA	0.19	0.19	—	—	—
NOVA	1	1	1	—	—
SYLVA	0.964	1	—	—	0.963
Iris	1	1	0.964	0.964	0.964
Spiral	1	1	1	1	0.98
Synthetic S	0.989	0.99	—	—	—
Synthetic NS	0.984	0.985	—	—	—
Avg.	0.878	0.883	—	—	—

Table C.14: False Discovery Rate

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.079	0.074	0.068	0.074	0.084
Wine	0	0	0	0	0.023
ADA	0.034	0.054	—	0.051	0.048
GINA	0.006	0	—	0.006	0
HIVA	0.009	0.009	—	—	—
NOVA	0	0	0	—	—
SYLVA	0	0.001	—	—	0.002
Iris	0	0	0	0	0
Spiral	0	0	0	0	0
Synthetic S	0.011	0.01	—	—	—
Synthetic NS	0.016	0.015	—	—	—
Avg.	0.014	0.015	—	—	—

Table C.15: Mathews Correlation Coefficient

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.832	0.83	0.821	0.813	0.816
Wine	1	1	1	1	0.98
ADA	0.655	0.649	—	0.64	0.636
GINA	0.994	0.994	—	0.981	0.994
HIVA	0.345	0.345	—	—	—
NOVA	1	1	1	—	—
SYLVA	0.981	0.993	—	—	0.967
Iris	1	1	0.981	0.981	0.981
Spiral	1	1	1	1	0.98
Synthetic S	0.976	0.978	—	—	—
Synthetic NS	0.965	0.967	—	—	—
Avg.	0.886	0.887	—	—	—

Table C.16: F_1 –Score

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.915	0.914	0.908	0.904	0.907
Wine	1	1	1	1	0.988
ADA	0.885	0.895	—	0.889	0.885
GINA	0.997	0.997	—	0.991	0.997
HIVA	0.928	0.928	—	—	—
NOVA	1	1	1	—	—
SYLVA	0.999	1	—	—	0.998
Iris	1	1	0.999	0.999	0.999
Spiral	1	1	1	1	0.99
Synthetic S	0.984	0.985	—	—	—
Synthetic NS	0.976	0.978	—	—	—
Avg.	0.971	0.972	—	—	—

Table C.17: Estimate of Scaled Error Rate

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	264.5	237.501	230	233.5	247.5
Wine	10.415	7.647	5.314	4.026	10
ADA	105.5	110	—	124	103
GINA	21.841	20.689	—	26.551	19.501
HIVA	105	97	—	—	—
NOVA	7.1	11.213	10.563	—	—
SYLVA	94.937	109.06	—	—	100.001
Spiral	16.85	20.179	17.209	15.315	18.5
Avg.	56.92	55.75	—	—	—

Table C.18: Area Under the Receiver Operating Characteristic Curve

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.977	0.974	0.971	0.97	0.971
Wine	1	1	1	1	1
ADA	0.938	0.933	—	0.905	0.927
GINA	1	1	—	1	1
HIVA	0.934	0.954	—	—	—
NOVA	1	1	1	—	—
SYLVA	1	1	—	—	1
Spiral	1	1	1	1	1
Avg.	0.981	0.983	—	—	—

Table C.19: Area Under the Receiver Operating Characteristic Curve Convex Hull

	Classifiers				
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM	FFNN [5, 20, 2]
Ripley	0.979	0.977	0.974	0.972	0.973
Wine	1	1	1	1	1
ADA	0.943	0.939	—	0.911	0.932
GINA	1	1	—	1	1
HIVA	0.949	0.966	—	—	—
NOVA	1	1	1	—	—
SYLVA	1	1	—	—	1
Spiral	1	1	1	1	1
Avg.	0.984	0.985	—	—	—

Table C.20: Root Mean Squared Error and Normalized Root Mean Squared Error.

	Classifiers			
Dataset	LS SVR	LS LPSVR	B. Dec. Trees	FFNN [5, 20, 2]
Root Mean Squared Error				
$f(x) = \text{sinc}(x)$	0.00119	0.001156	0.001168	0.00105
$f(x) = \text{sinc}(x) \times \pi$	0.11671	0.11463	0.11747	0.11577
Normalized Root Mean Squared Error				
$f(x) = \text{sinc}(x)$	0.003479	0.00338	0.003416	0.00307
$f(x) = \text{sinc}(x) \times \pi$	0.32429	0.31848	0.3264	0.32167

Table C.21 shows a comparison of the sum of squared errors, and the statistical measure.

Table C.21: Sum of Squared Error and Statistical Metrics

	Classifiers			
Dataset	LS SVR	LS LPSVR	B. Dec. Trees	FFNN [5, 20, 2]
Sum of Squared Error				
$f(x) = \text{sinc}(x)$	0.000283	0.000267	0.000273	0.00022
$f(x) = \text{sinc}(x) \times \pi$	13622.3175	13138.9464	13800.226	13402.8679
Statistical Metrics				
$f(x) = \text{sinc}(x)$	0.001352	0.001193	0.00127	0.001082
$f(x) = \text{sinc}(x) \times \pi$	0.11678	0.11467	0.11757	0.11581

Table C.22 shows a comparison of the number of exact support vectors.

Table C.22: Exact Support Vectors

	SV-Based Classifiers			
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM
Ripley	0.14	0.06	0.14	1.00
Wine	0.15	0.15	0.21	1.00
ADA	0.16	0.08	—	1.00
GINA	0.07	0.06	—	1.00
HIVA	0.15	0.16	—	—
NOVA	0.07	0.07	0.06	—
SYLVA	0.06	0.03	—	—
Iris	0.20	0.13	0.19	1.00
Spiral	0.17	0.16	0.16	1.00
$f(x) = \text{sinc}(x)$	0.15	0.16	0.15	1.00
$f(x) = \text{sinc}(x) \times \pi$	0.01632	0.00331	—	—
Synthetic S	0.000016	0.000003	—	—
Synthetic NS	0.000082	0.000018	—	—

Table C.23 shows a comparison of the number of saturated support vectors.

Table C.23: Saturated Support Vectors

	SV-Based Classifiers			
Dataset	LS SVM	LS LPSVR	IncSVM	LSSVM
Ripley	0.16	0.07	0.16	1.00
Wine	0.16	0.19	0.17	1.00
ADA	0.19	0.10	—	1.00
GINA	0.08	0.08	—	1.00
HIVA	0.18	0.18	—	—
NOVA	0.07	0.08	0.07	—
SYLVA	0.07	0.03	—	—
Iris	0.18	0.14	0.14	1.00
Spiral	0.19	0.19	0.19	1.00
$f(x) = \text{sinc}(x)$	0.18	0.18	0.17	1.00
$f(x) = \text{sinc}(x) \times \pi$	0.01865	0.00379	—	—
Synthetic S	0.000019	0.000004	—	—
Synthetic NS	0.000093	0.000020	—	—

Appendix D

Background in the Newton Method and in Kernel Functions

D.1 Newton Method for Error Function Minimization

Let $f(\boldsymbol{\theta})$, $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a real function representing some estimate of classification of regression error; where $\boldsymbol{\theta} \in \mathbb{R}^n$ is a vector of parameters, and $\mathcal{T} = \{\mathbf{x}_i, d_i\}_{i=1}^N$ defines a training set given by N samples of the M -dimensional data vector $\mathbf{x} \in \mathbb{R}^M$, and a desired output class value $d \in \mathbb{R}$. Then, let $\mathbf{F} : \mathbb{R}^n \mapsto \mathbb{R}^m$ be denoted as

$$\mathbf{F}(\boldsymbol{\theta}) = \begin{pmatrix} f_1(\boldsymbol{\theta}) \\ f_2(\boldsymbol{\theta}) \\ \vdots \\ f_m(\boldsymbol{\theta}) \end{pmatrix}_{m \times n}. \quad (\text{D.1})$$

That is, \mathbf{F} represents m different measures of error, provided model parameters $\boldsymbol{\theta}$, and training data \mathcal{T} . Here, we aim to make (ideally) $\mathbf{F}(\boldsymbol{\theta}) = \mathbf{0} \equiv (0, 0, \dots, 0)^T$.

In this research the case when $n = m$ is addressed, that is, when the number of model parameters to estimate is equal to the number of error metrics used to find such model parameters: $\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots, \theta_n\}$, and f_1, f_2, \dots, f_n .

If $\mathbf{F} : \mathbb{R}^n \mapsto \mathbb{R}^n$, then it has a gradient usually known as Jacobian given by

$$\nabla \mathbf{F}(\boldsymbol{\theta}) \equiv \mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}) = \begin{pmatrix} \nabla f_1(\boldsymbol{\theta})^T \\ \nabla f_2(\boldsymbol{\theta})^T \\ \vdots \\ \nabla f_n(\boldsymbol{\theta})^T \end{pmatrix}_{n \times n}, \quad (\text{D.2})$$

where $\nabla f_n(\boldsymbol{\theta})$ denotes the gradient of the n -th function, given by

$$\nabla f_n(\boldsymbol{\theta})^T = \left(\frac{\partial f_n}{\partial \theta_1} \quad \frac{\partial f_n}{\partial \theta_2} \quad \cdots \quad \frac{\partial f_n}{\partial \theta_n} \right)_{1 \times n}. \quad (\text{D.3})$$

Since we want to find the vector of parameters $\boldsymbol{\theta}^*$, for which the training set \mathcal{T} produce minimal error functions, such that $\mathbf{F}(\boldsymbol{\theta}^*) = \mathbf{0}$, then we can use Newton method assuming, for now, that \mathbf{F} is continuously differentiable on \mathbb{R}^n .

D.1.1 Newton Method

This method is well known from basic calculus and optimization courses. Newton method can be summarized as in Algorithm D.1.

Algorithm D.1 Newton method to find $\boldsymbol{\theta}^*$ that satisfies $\mathbf{F}(\boldsymbol{\theta}^*) = \mathbf{0}$.

Require: \mathbf{F} to be continuously differentiable on \mathbb{R}^n

Require: A *close* initial point $\boldsymbol{\theta}^0$.

1: **for** $t = 0, 1, 2, \dots$, until convergence **do**

2: Solve for $\Delta \boldsymbol{\theta}^t$ in:

▷ Newton direction

$$\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}^t) \Delta \boldsymbol{\theta}^t = -\mathbf{F}(\boldsymbol{\theta}^t) \text{ or} \quad (\text{D.4})$$

$$\Delta \boldsymbol{\theta}^t = -(\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}^t))^{-1} \mathbf{F}(\boldsymbol{\theta}^t) \quad (\text{D.5})$$

3: Update:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \Delta \boldsymbol{\theta}^t \quad (\text{D.6})$$

4: **end for**

Newton method is known because it has a q -quadratic rate of convergence, finding a solution in very few iterations, such that $\mathbf{F}(\boldsymbol{\theta}^t) = \mathbf{0}$, if and only if such a solution exists.

This method is also known for one of its main disadvantages: it is a local method. Therefore, one needs to have in advance a vector of parameters that is close to an acceptable solution. To overcome this difficulty, the following globalization strategy is established.

D.1.2 Globalization Strategy

The proposed globalization strategy uses the following merit function:

$$M_f(\boldsymbol{\theta}) = \frac{1}{2} \|\mathbf{F}(\boldsymbol{\theta})\|_2^2, \quad (\text{D.7})$$

where $\|\cdot\|_2$ denotes the ℓ_2 -norm. Then the following property is defined.

Property D.1. $\Delta\boldsymbol{\theta}$ is a descent direction for $M_f(\boldsymbol{\theta})$. That is, $\mathbf{0} \geq \Delta\boldsymbol{\theta}$ in the linear system given by

$$\nabla M_f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta} = -M_f(\boldsymbol{\theta}). \quad (\text{D.8})$$

Proof. Let $\nabla M_f(\boldsymbol{\theta})$ be the derivative of the merit function (D.7) denoted as:

$$\nabla M_f(\boldsymbol{\theta}) = \frac{1}{2} \mathbf{J}_F(\boldsymbol{\theta})^T \mathbf{F}(\boldsymbol{\theta}). \quad (\text{D.9})$$

Then, substituting (D.9) into (D.8) results

$$\frac{1}{2} \mathbf{F}(\boldsymbol{\theta})^T \mathbf{J}_F(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} = -\frac{1}{2} \|\mathbf{F}(\boldsymbol{\theta})\|_2^2 \quad (\text{D.10})$$

which reduces to

$$\Delta\boldsymbol{\theta} = -(\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}))^{-1} \mathbf{F}(\boldsymbol{\theta}) \leq \mathbf{0}. \quad (\text{D.11})$$

Hence, $\mathbf{0} \geq \Delta\boldsymbol{\theta}$. \square

Since the merit function (D.7) is a valid function that guarantees a descent at every iterate, then we can establish the globalization strategy by defining the following property.

Property D.2. *If $\Delta\boldsymbol{\theta}$ is a descent direction of $M_f(\boldsymbol{\theta})$, then, there exists a $\beta_1 \in (0, 1]$ such that*

$$M_f(\boldsymbol{\theta} + \beta_1 \Delta\boldsymbol{\theta}) \leq M_f(\boldsymbol{\theta}) + \beta_1 \beta_2 \nabla M_f(\boldsymbol{\theta})^T \Delta\boldsymbol{\theta}. \quad (\text{D.12})$$

Proof. The proof is given by Dennis *et al.*. 1996 [51] (see Theorems 6.3.2. and 6.3.3. pp. 120-123). \square

Thus, substituting (D.7) into (D.12), one obtains the following

$$\frac{1}{2} \|\mathbf{F}(\boldsymbol{\theta} + \beta_1 \Delta\boldsymbol{\theta})\|_2^2 \leq \frac{1}{2} \|\mathbf{F}(\boldsymbol{\theta})\|_2^2 + \quad (\text{D.13})$$

$$\beta_1 \beta_2 \mathbf{F}(\boldsymbol{\theta})^T \mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta}, \quad (\text{D.14})$$

which reduces to

$$\|\mathbf{F}(\boldsymbol{\theta} + \beta_1 \Delta\boldsymbol{\theta})\|_2 \leq \|\mathbf{F}(\boldsymbol{\theta})\|_2 + \sqrt{1 - 2\beta_1 \beta_2}, \quad (\text{D.15})$$

where β_2 is a parameter controlling the speed of the globalization strategy (a.k.a. line search). Typically $\beta_2 = 1 \times 10^{-4}$ [51].

Using the line-search globalization strategy, we can modify Newton method to include a sufficient decrease condition (a.k.a. Armijo's condition). The Globalized

Newton method is as shown in Algorithm D.2.

Algorithm D.2 Globalized Newton method to find $\boldsymbol{\theta}^*$ that satisfies $\mathbf{F}(\boldsymbol{\theta}^*) = \mathbf{0}$.

Require: \mathbf{F} to be continuously differentiable on \mathbb{R}^n

Require: An initial point $\boldsymbol{\theta}^0$.

1: **for** $t = 0, 1, 2, \dots$, until convergence **do**

2: Solve for $\Delta\boldsymbol{\theta}^0$ in: ▷ Newton direction

$$\mathbf{J}_{\mathbf{F}}(\boldsymbol{\theta}^t)\Delta\boldsymbol{\theta}^t = -\mathbf{F}(\boldsymbol{\theta}^t)$$

3: Sufficient decrease: ▷ Armijo's condition

Find β_1 that satisfies:

$$\|\mathbf{F}(\boldsymbol{\theta}^t + \beta_1\Delta\boldsymbol{\theta}^t)\|_2 \leq \|\mathbf{F}(\boldsymbol{\theta}^t)\|_2 + \sqrt{1 - 2\beta_1\beta_2}$$

4: Update:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t + \beta_1\Delta\boldsymbol{\theta}^t \tag{D.16}$$

5: **end for**

Notice the new update step (D.16) that considers the sufficient decrease condition. This will produce an acceptable step towards the solution parameters: $\boldsymbol{\theta}^t \rightarrow \boldsymbol{\theta}^*$.

This material completes the background needed in order to follow the discussion presented in Section 4.5. Nonetheless, the reader can read further into details given in textbooks by Dennis, *et al.* [51], or Nocedal, *et al.* [136].

D.2 Background in Kernel Functions

D.2.1 Mercer's Theorem

Theorem D.1. *If $k(x, w)$ is a Mercer kernel then there exists a Hilbert space \mathcal{H}_k of real valued functions defined on \mathcal{X} and a feature map $\phi : \mathcal{X} \mapsto \mathcal{H}_k$ such that,*

$$\langle \phi(x), \phi(w) \rangle_k = k(x, w) \tag{D.17}$$

where $\langle \cdot, \cdot \rangle_k$ is the innerproduct on \mathcal{H}_k .

Proof. Let L be the vector space containing all the real valued functions f defined on \mathcal{X} of the form,

$$f(x) = \sum_{j=1}^m \alpha_j k(x_j, x) \quad (\text{D.18})$$

where m is a positive integer, the α_j 's are real numbers and $\{x_1, \dots, x_m\} \subset \mathcal{X}$. Define in L the inner product,

$$\left\langle \sum_{i=1}^m \alpha_i k(x_i, \cdot), \sum_{j=1}^n \beta_j k(w_j, \cdot) \right\rangle = \sum_{i=1}^m \sum_{j=1}^n \alpha_i \beta_j k(x_i, w_j) \quad (\text{D.19})$$

Since k is a Mercer kernel the above definition makes L a well defined inner product space. The Hilbert space \mathcal{H}_k is obtained when we add to L the limits of all the Cauchy sequences (with respect to the $\langle \cdot, \cdot \rangle$) in L .

Notice that the inner product in H_k was defined such that

$$\langle k(x, \cdot), k(w, \cdot) \rangle_k = k(x, w). \quad (\text{D.20})$$

We can then take the feature map to be,

$$\phi(x) = k(x, \cdot) \quad (\text{D.21})$$

□

For a more detailed explanation of this topic, consult [41, 129].

D.2.2 Reproducing Kernel Hilbert Space

The space H_k is said to be a Reproducing Kernel Hilbert Space (RKHS). Moreover, for all $f \in \mathcal{H}_k$

$$f(x) = \langle k(x, \cdot), f \rangle_k. \quad (\text{D.22})$$

This follows from the reproducing property when $f \in L$ and by continuity for all $f \in \mathcal{H}_k$. It is also easy to show that the reproducing property is equivalent to the continuity of the evaluation functionals $\delta_x(f) = f(x)$.

The Mercer kernel $k(x, w)$ naturally defines an integral linear operator that we also denote by $k : \mathcal{H}_k \mapsto \mathcal{H}_k$, where,

$$(kf)(x) = \int k(x, y)f(y)dy \quad (\text{D.23})$$

and since $k(x, y) = \mathbf{K}$ is symmetric and positive definite, it is orthogonally diagonalizable. Thus, there is an ordered orthonormal basis set $\{\phi_1, \phi_2, \dots\}$ of eigen vectors of \mathbf{K} . *E.g.*, for $j = 1, 2, \dots$

$$\mathbf{K}\phi_j = \lambda_j\phi_j \quad (\text{D.24})$$

with $\langle \phi_i, \phi_j \rangle_k = \delta_{ij}$, $\lambda_1 \geq \lambda_2 \geq \dots > 0$ and such that,

$$k(x, w) = \sum_{j=1}^{\infty} \lambda_j \phi_j(x) \phi_j(w) \quad (\text{D.25})$$

from where it follows that the feature map is,

$$\phi(x) = \sum_{j=1}^{\infty} \lambda_j^{1/2} \phi_j(x) \phi_j \quad (\text{D.26})$$

producing,

$$k(x, w) = \langle \phi(x), \phi(w) \rangle_k \quad (\text{D.27})$$

Notice also that any continuous map $\phi : \mathcal{X} \mapsto \mathcal{H}$ where \mathcal{H} is a Hilbert space, defines $k(x, w) = \langle \phi(x), \phi(w) \rangle$ which is a Mercer Kernel.

D.2.3 Representer Theorem

Theorem D.2 (Reisz Representer Theorem). *If ϕ is a bounded functional on \mathcal{H} then exists a unique $u \in \mathcal{H}$ such that $\phi(f) = \langle f, u \rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$ define a function/operator k is positive-definite if for all functions $\int f(x) k(x, x') f(x') dx dx' > 0$.*

Proof. Proof follows from Mercer's theorem and the theory of RKHS explained above.

□

Appendix E

Remote Sensing

E.1 Non-Linear Enhancement for MODIS True Color Images

For any downloaded MODIS reflectance data, if we try to make a true-color image using bands B1, B4, and B3 as the RGB values, the images will appear dark. The key to solve this problem is to scale the data differentially as proposed by Gumley *et. al.* [59]. This involves two vectors:

$$\begin{aligned}\text{input} &= [0, \quad 30, \quad 60, \quad 120, \quad 190, \quad 255] \\ \text{output} &= [0, \quad 110, \quad 160, \quad 210, \quad 240, \quad 255]\end{aligned}$$

The differential scaling consists of the following steps: First, scale the MODIS reflectance data into the range $[0, 255]$. Second, use the input vector to select pixels in the scaled image and re-scale those pixels to the range indicated by the output vector. For example, take the pixels in the scaled image whose range fall between 0 and 30, and re-scale those pixels into the range 0 to 110. Then find the pixels with values between 30 and 60, and scale those pixels into the range of 110 to 160, and so on.

E.2 List of MODIS Level 1B Data Granules

The complete list of data granules referred in Section II can be found in [157]. These events correspond to NASA Terra MODIS instrument. The reference will show level 1B files.

Curriculum Vitae

Pablo Rivas Perea was born on August 4, 1980, at a small beach town on the Pacific coast called Guaymas, in Sonora, México. He received the BS in Computer Systems Engineering degree from the Nogales Institute of Technology, Sonora, México in 2003 and the MS in Electrical Engineering from the Chihuahua Institute of Technology, Chihuahua, México, in 2007. He received his PhD in Electrical and Computer Engineering at The University of Texas El Paso (UTEP) in 2011. He has worked in the manufacturing industry for three years as a Product Engineer. He also has worked as a Software Engineer for five years. He has national and international publications in the fields of computer science and electrical engineering. His current research interests include Bible study, massive numerical optimization, cheap computers, large-scale pattern recognition, going to the movie theater, multispectral and multidimensional image processing, basketballing, large-scale computational neural networks, Christian music, fuzzy optimization, guitars, medical applications, and eating pizza.

Rivas Perea became a member of the Institute of Electrical and Electronic Engineers (IEEE) in 2002. He is a member of the Computer Society and the Computational Intelligence Society (IEEE-CIS), both IEEE. Also he is a member of the Society of Industrial and Applied Mathematics (SIAM), member of the Hispanic-American Fuzzy System Association (HAFSA), member of the International Association of Engineers (IAENG), member of the Optical Society of America (OSA), and member of the International Neural Networks Society (INNS). He has served as technical reviewer for the INNS and IEEE-CIS since 2008. He was the IEEE Chihuahua Student Branch chair in the 2006-2007 period. He has received several awards concerning scientific creativity and academics. Also he received the IEEE Student Enterprise Award in 2007 and the Research Excellence Award from the Paul L. Foster Health

Sciences School at the Texas Tech University in 2009. In 2009, Mr. Rivas Perea was an intern at NASA Goddard Space Flight Center (GSFC), becoming the first Mexican national graduate student in the NASA GSFC history. In 2011, he was awarded a life time membership to Eta Kappa Nu (HKN): the worldwide honor society for electrical and computer engineers in academic excellence standing.

During his doctoral studies at UTEP, Pablo Rivas Perea obtained the following scholarships: CONACYT doctoral scholarship, Cotton Memorial graduate scholarship, Texas Instruments Foundation Endowed Scholarship for graduate students, IEEE Computational Intelligence travel scholarship, and the SEP-DGRI scholarship for graduate students. He partially supported his studies with a teaching assistantship from the Electrical and Computer Engineering Department.

In the near future, Mr. Rivas Perea plans to take it easy and spend more time with his wife, Nancy. Also he plans to visit his relatives and in-laws more often. Pablo has planned to have seven children with Nancy, God permitting. He currently has an offer for a full-time tenured professor position from the Sonora Institute of Technology (ITSON), at México. However, he is waiting for 15 more response letters from universities and research laboratories within the US and México.

Permanent address: Leopardo 3909

Frac. Lomas del Sol

Chihuahua, Chih. CP 31167

México