5-2020

Neural Networks

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-20-53

Neural Networks

Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso, El Paso, TX, USA
vladik@utep.edu

Definition

A neural network is a general term for machine learning tools that emulate how neurons work in our brains.

Ideally, these tools do what we scientists are supposed to do: we feed them examples of the observed system's behavior, and hopefully, based on these examples, the tool will predict the future behavior of similar systems. Sometimes they do predict – but in many other cases, the situation is not so simple.

The goal of this entry is to explain what these tools can and cannot do – without going into too many technical details.

How Machine Learning Can Help

What Are the Main Problems of Science and Engineering

The main objectives of science and engineering are:

- to determine the current state of the world,
- to predict the future behavior of different systems and objects, and,
- if it is possible to affect this behavior, to find the best way to do it.

For example, in geosciences:

- we want to find mineral deposits – oil, gas, water, etc.
- we want to be able to predict potentially dangerous activities such as earthquakes and volcanic eruptions, and
- we want to find the best ways to extract minerals that would not lead to undesirable side effects such as pollution or triggered seismic activity.

Let us describe the three classes of general problems in precise terms.

To determine the current state of the world, we can use the measurement results x. Based on these measurement results, we want to describe the actual state y of the system. For example, to find the geological structure y in a certain area – e.g., the density (and/or speed of sound) values at different depths and at different locations, we can use the seismic data x, both passive (measuring seismic waves generated by actual earthquakes) and active (measuring seismic signals generated by specially set explosions or vibrations).

To predict the future state y of the system – or at least the future values y of the quantities of interest – we can use the information x about the current and past state of the system. For example, by using the accurate GPS measurements x, we can find how fast the continents drifts, and thus, predict their future location y.

To find the best control y, we can use all the known information x about the current state of the system. For example, based on our knowledge of the geological structure x of the area, we would like to find the parameters (e.g., pressure) y of the fracking technique that would leave the pollution below the desired level.

Sometimes We Know the Equations, Sometimes We Don't

In some cases, we know the equations relating the available information x and the desired quantities y. In some of these cases, the relation is straightforward: e.g., simple linear extrapolation formulas enable us to predict the future continent drift. In other cases, the equations are not easy to solve: for example, it is relatively easy, giving the density structure y, to describe how the seismic signals will propagate and what signals x to expect – but to find y based on x (i.e., to solve the *inverse* problem) is often not easy.

In most cases, however, we do not know the equations relating x and y. In this, geosciences are drastically different from physics – especially fundamental

physics – where the corresponding equations are usually known (they may be difficult to solve, as in predicting chemical properties of atoms and molecules from Schroedinger's equation, but they are known).

How Machine Learning Can Help: General Case

Usually, there are cases when we know both the input x and the desired output y. In other words, we know several pairs (x_i, y_i) corresponding to different situations.

Such cases are ubiquitous for prediction problems: for example, if we are trying to predict seismic activity a week ahead, then every time a week has passed, we have a pair (x_i, y_i) consisting of the measurement results x_i obtained before the passed week and the passed week's seismic activity y_i.

Such cases are ubiquitous in control problems: every time we do something and succeed, we get a pair (x_i, y_i) consisting of the previous situation x_i and of the successful action.

Based on such pairs, machine learning tools produce an algorithm, that, given the input x, provides an estimate for the desired output y. For example, for prediction problems, we can use the current values x to get some predictions of the future values y.

Producing such an algorithm usually takes time – but once the algorithm has been produced, it usually works very fact.

How Machine Learning Can Help: Case When We Know Equations.

When we know equations, the difficulty is usually in solving the inverse problem – finding y based on x. In contrast, the "forward" problem – finding x based on y – is usually easy to solve. So, what we can do is select several realistic examples $y_1, \ldots, y_n$ of y, compute the corresponding x's $x_1, \ldots, x_n$, and feed the resulting pairs (x_1, y_1), $\ldots$, (x_n, y_n) into a machine learning tool.

As a result, we get an algorithm, that given x, produces y – i.e., that solves the inverse problem.

Limitations

There Are Limitations

So far, it may have seemed that machine learning is a panacea, that it can solve almost all of our problems. But, of course, the reality is not always rosy. To effectively use machine learning, we need to solve three major problems:

- First, the ability of machine learning tools to process multi-D data is limited. In most cases, these tools cannot use *all* the measurement results that form the input x_i and the output y_i. So, we need to come up with a small number of informative characteristics. This selection is up to us, it is difficult, and if we do not select these characteristics correctly, we lose information – and the machine learning program will not be able to learn anything.
- Second, to make accurate predictions, we need to have a large number of pairs: thousands, sometimes millions. Sometimes we have many such pairs – e.g., when we are solving an inverse problem. But in many other important cases – e.g., in predicting volcanic eruptions or strong earthquakes – there are – thankfully – simply not that many such events. Of course, we can add to the days of observed eruptions days when nothing happened, but then the machine learning program will simply always predict "no eruption" – and be accurate in the spectacular 99.99% of the cases!
- Third, training a machine learning program requires a lot of time, so much that often it can only be done on a high performance computer.

The need to solve these three problems – especially the first two – severely limits the usefulness of machine learning tools.

Let us explain, on a qualitative level, where these problems come from.

Machine Learning Is, in Effect, a Nonlinear Regression

Machine learning is not magic. It is, in effect, a nonlinear regression: just like linear regression enables us

to find the coefficient of a linear dependence based on samples, nonlinear regression enables us to find the parameters of a nonlinear regression. For neural networks, such parameters are known as *weights*.

In many practical situations – especially in geosciences – linear models provide a very crude approximation. So, crudely speaking, in addition to parameters describing linear terms, we also need parameters describing quadratic terms, cubic terms, etc. The more parameters we use, the more accurately we can describe the actual dependence. This is similar to how we can approximate, e.g., an exponential function $\exp(x) = 1 + x + \dfrac{x^2}{2!} + \ldots$ by the first few terms in its Taylor expansion: $\exp(x) = 1 + x + \ldots + \dfrac{x^m}{m!}$ (this is, by the way, how computers actually compute $\exp(x)$): the more terms we use, the more accurate is the result.

We Cannot Use All the Information

And here lies the problem. The numbers of nonlinear terms drastically grows with the number of inputs. For example, if the input x consists of m values $x = (X_1, \ldots, X_m)$, then, to describe a generic linear dependencies $Y = c_0 + \sum\limits_{i=1}^{m} c_i \cdot X_i$, we need $m + 1$ parameters. To describe a generic quadratic dependence $Y = c_0 + \sum\limits_{i=1}^{m} c_i \cdot X_i + \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{m} c_{ij} \cdot X_i \cdot X_j$, we already need $\approx m^2$ parameters. This already leads to a problem. For example, suppose that we are processing images. Describing an image x means describing the intensity X_i at each of its pixels i. A typical image consists of about $1000 \times 1000 = 10^6$ pixels, so here we have $m \approx 10^6$. It is easy to store and process a million values, but finding 10^{12} unknown coefficients – even in the simplest case, when we have a system of 10^{12} linear equations with 10^{12} unknown – is way beyond the abilities of modern computers.

As a result, we cannot simply feed the image into a machine learning tool – and, similarly, we cannot simply feed the seismogram into this tool. We need to select a few important parameters characterizing this image (or this seismogram). And here computers are not much help, we the scientists need to do the selection. This explains the first problem.

It should be mentioned that the situation is not so bad with images. Everyone knows that images can be compressed into a much smaller size without losing much information – e.g., a small-size photo on a webpage is still quite recognizable – and modern machine learning techniques used such methods automatically. However, for seismograms, no such no-information-loss drastic compressions are known.

What about computation time? The more parameters we need, the more computation time we need to find the values of these parameters. Even if the system is linear, to find the values of q parameters – i.e., to solve a system of q equations for q unknowns – we need time $\approx q^3$, at least as much as we need to multiply two $q \times q$ matrices A and B – where to compute each of q^2 elements of the product $a_{ij} = a_{i1} \cdot b_{1j} + \ldots + a_{iq} \cdot b_{qj}$, we need q computational steps (and $q^2 \cdot q = q^3$).

Even if we compress the image from a million to $m = 300$ values, if we take the simplest – quadratic – terms into account, we will need $q \approx m^2 = 10^5$ variables, so we need at least $q^3 \approx 10^{15}$ computational steps. On a usual GigaHertz PC that performs 10^9 operations per second, this means 10^6 seconds – about two weeks. And we only took into account quadratic terms – and just like an exponential function or a sinusoid do not look like graphs of x^2, real-life dependencies are not quadratic either. So machine learning requires a lot of computation time – often so much time that only high-performance computers can do it. This explains the third problem.

We Need a Large Number of Samples

And this is not all. The more accurately we want to predict y, the more parameters we need. How many samples do we need? Crudely speaking, each pair (x_i, y_i) with $y_i = (Y_{i1}, \ldots, Y_{ir})$ provides r equations $Y_{ij} = f(X_{i1}, \ldots, X_{iq})$ for determining the unknown parameters, for some small r. So, we need approximately as many samples as parameters. In the above example of $q \approx 10^5$ unknowns, we need hundreds of thousands of example – and usually, even more. This explains the second problem.

There is an additional reason why we need many pairs. The reason is that it is not enough to just train the tool, we need to test how well the trained machine learning tool works. For that purpose, the usual idea is to divide the original pairs into the training set and the testing set, train the tool on the training set only, and then test the resulted training on the training set.

How do we know that it works well? One correct prediction may be a coincidence. However, if we have several good predictions, this makes us confident that the trained model works well. So, to gain this confidence, we need a significant number of such pairs. And in many important problems, we do not have that many pairs. For example, even if a volcano has been very active, had 5 eruptions, there is not enough data to confirm the model.

An Additional Problem

All the above arguments assumed that once we arranged an appropriate compression, gathered millions of sample, and rented time on high-performance computer, the machine learning tool will always succeed. Unfortunately, this is not always the case.

It is known that, in general, prediction problems, inverse problems, etc. are NP-hard, which means that (unless P = NP, which most computer scientists believe to be impossible) no feasible algorithm is possible that would always solve these problems. In other words, any feasible algorithm – and algorithms implemented in the machine learning tools are feasible – will sometimes not work.

Good news is that computer scientists are constantly inventing new algorithms, new tools. So if you encounter such a situation, a good idea is to team up with such a researcher. Maybe his/her new algorithm will work well when the algorithms from the software package you used did not work.

This brings us to the question of what machine learning tools are available.

What Tools Are Available

Traditional neural networks used 3 layers of neurons. Corresponding, the number of parameters was not high, so while such neural networks can be easily implemented on a usual PC, they are not very accurate. So, if you have a few samples – not enough for more accurate training – you can use traditional neural networks, and get some reasonable (but not very accurate) results.

In the latest decades, the most popular are deep neural networks, that have up to several dozen layers, and thus, a much larger number of adjustable parameters. Often, they lead to spectacular results and accurate predictions. However, due to the high number of parameters, deep neural networks need a large (sometimes unrealistically large) number of sample pairs. For the same reason, deep neural networks require a lot of time to train – so much that this training is rarely possible on a usual computer. So, if you have a large number of samples – and you know how to compress the original information – it is a good idea to try to use deep learning.

There are also other efficient machine learning tools, such as Support Vector Machine (SVM) – that used to be the most efficient tool until deep learning appeared – but these tools are outside the scope of this article.

In Case You Are Curious

So how do neural networks work? An artificial neural network consists of *neurons*. Each neuron takes several inputs $v_1, \ldots, v_k$ and returns an output $u = s(w_0 + w_1 \cdot v_1 + \ldots + w_k \cdot v_k)$, where w_i are coefficients ("weights") that need to be determined during training, and $s(z)$ is an appropriate nonlinear function. Traditional neural networks used the so-called sigmoid functions $s(z) = 1/(1+\exp(-z))$, while deep neural networks use the "rectified linear" function $s(z) = \max(0, z)$.

Neurons usually form layers:

- Neurons from the first layer use the measurement results (or whatever we feed them as x_i) as inputs.
- Neurons from the second layer use outputs of the first layer neurons as inputs, etc.

How are the coefficients w_i trained? In a nutshell, by using *gradient descent* – the very first optimization method that students learn in their numerical methods class. The main idea behind this method is very straightforward: to do down the mountain as

fast as possible, you follow the direction in which the descent is the steepest. Neural networks use some clever algorithmic tricks that help find this steepest direction, but they *do* use gradient descent.

Readers interested in technical details are welcome to read (Goodfellow et al., 2016) – at present (2020) the main textbook on deep neural networks.

Examples of Successful Applications

Successful applications of *traditional* (3-layers) neural networks have been summarized in a widely cited book (Dowla and Rogers, 1996). There have been many interesting applications since then, especially in petroleum engineering, where even a small improvement in prediction accuracy can lead to multi-million gains. The paper (Thornhauser, 2015) provides a survey of such applications. For example, neural neworks are efficient in classifying geological layers based on the well log (Bhatt and Helle, 2002) – this is one of the cases when we have a large amount of data, sufficient to train a neural network.

Applications of *deep learning* are a rapidly growing research area. There is a large number of papers, there are surveys – e.g., (Bergen at al., 2019). However, new applications and new techniques appear all the time – this research area grows faster that surveys can catch up. Because of this fast growth, this part of the article is more fragmentary than comprehensive – we will just cite examples of typical applications.

Probably the most active application areas is processing images – e.g., satellite images. One of the main reasons why these applications of deep learning are most successful is that, as we have mentioned, for images we can apply efficient almost-no-loss compression and thus, naturally prepare the data for neural processing. A typical recent example of such an application is (Ullo et al., 2019).

One of the most interesting applications of deep learning to satellite images is the possibility to predict volcanic activities based on images produced by satellites equipped with interferometric synthetic aperture radar (InSAR) – which can detect centimeter-scale deformations of Earth's surface (Gaddes et al., 2019).

Many examples of applications are related to solving inverse problems – since, as we have mentioned, in this case, we can easily generate a large number of examples; see, e.g., (Araya-Polo et al., 2018), (Mosser et al., 2018). The use of state-of-the-art machine learning techniques for solving inverse problem has already lead to interesting discoveries. For example, it turns out that, contrary to the previously assumed simplified models of seismic activity – according to which only usual ("fast") earthquakes release the stress, a significant portion of the stress is released through slow slip and slow earthquakes; see, e.g., (Pratt, 2019).

Interestingly, sometimes even for the forward problem, neural networks produce results faster than the traditional numerical techniques; see, e.g., (Moseley at al., 2018).

The papers (Magaña-Zook and Ruppert, 2017) and (Linville et al., 2019) uses deep learning to solve another important problem: how to distinguish earthquakes from explosions based on their seismic waves. This problem was actively developed in the past because of the need to distinguish nuclear weapons tests from earthquakes. Now, the main case study is separating small earthquakes from small explosions like quarry blasts – and in this appplication, there is a large number of examples, which makes neural networks very successful.

Summary

In a nutshell, neural networks are interpolation and extrapolation tools. When we have a large number (x_i, y_i) of pairs (x, y) of related tuples of quantities, machine learning techniques – such as neural networks – produce a program that, given a generic tuple x, estimates y.

In many cases, neural networks has led to successful applications in geoscience. However, neural networks are not a panacea.

For a neural network application to be successful, we really need a very large number of examples – which is not always possible in geosciences; we need to compress the data without losing information –

which is also often difficult in geosciences; we usually need to spend a large amount of computation time; and we need to be patient: sometimes these techniques work, sometimes they don't.

We hope that this article will help geoscientists to select problems in which all these conditions are satisfied, and get great results by applying neural networks! (And do not hesitate to collaborate with computer scientists if it does not work the first time around.)

Acknowledgments. This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science) and HRD-1242122 (Cyber-ShARE Center of Excellence).

References

Araya-Polo, M., Jennings, J., Adler, A., Dahlke, T., 2018. Deep-learning tomography, *Leading Edge (Tulsa Okla.)*, **37**, 58–66.

Bergen, K. J., Johnson, P. A., de Hoop. M. V., and Beroza, G. C., 2019. Machine learning for data-driven discovery in solid Earth geoscience, *Science*, **363**(6433), Paper eaau0323.

Bhatt, A., and Helle, H. B., 2002. Determination of facies from well logs using modular neural networks, Petroleum Geoscience, **8**, 217–228.

Dowla, F. U., and Rogers, L. L., 1996. *Solving Problems in Environmental Engineering and Geosciences with Artificial Neural Networks*, Cambridge, Massachusetts: MIT Press.

Gaddes, M. E., Hooper, A., and Bagnard, M., 2019. Using machine learning to automatically detect volcanic unrest in a time series of interferograms, *Journal of Geophysical Research: Solid Earth*, **124**(11), 12304–12322.

Goodfellow, I., Bengio, Y., and Courville, A., 2016. *Deep Learning*, Cambridge, Massachusetts: MIT Press.

Linville, L., Pankow, K., and Draelos, T., 2019. Deep learning models augment analyst decisions for event discrimination, *Geophysical Research Letters*, **46**(7), 3643–3651.

Magaña-Zook, S. A., and Ruppert, S. D., 2017. Explosion monitoring with machine learning: A LSTM approach to seismic event discrimination,*Abstract of the American Geophysical Union (AGU) Fall 2017 Meeting*, New Orleans, Louisiana, December 11–15, 2017, Abstract S43A-0834.

Moseley, B., Markham A., and Nissen-Meyer, T., 2018. Fast approximate simulation of seismic waves with deep learning, *Proceedings of the 2018 Conference on Neural Information Processing Systems NeurIPS'2018*, Montreal, Canada, December 3–8, 2018.

Mosser, L., Kimman, W., Dramsch, J., Purves, S., De la Fuente, A., Ganssle, G., 2018. Rapid seismic domain transfer: Seismic velocity inversion and modeling using deep generative neural networks, *Proceedings of the 80th European Association of Geoscientists and Engineers (EAGE) Conference*, Copenhagen, Denmark, June 11–14, 2018.

Pratt, S. E., 2019. Machine fault, *Earth & Space Science News (EoS)*, December 2019, 28–35.

Thonhauser, G., 2015. Application of artificial neural networks in geoscience and petroleum industry. In: Craganu, C., Luchian, H., and Breaban, M. E., Artificial Intelligent Approaches in Petroleum Geosciences, Cham, Switzerland: Springer, 127–166.

Ullo, S. L., Langenkamp, M. S., Oikarinen, T. P., Del Rosso, M. P., Sebastianelli, A., Piccirillo, F. Sica, S., 2019. Landslide geohazard assessment with convolutional neural networks using Sentinel-2 imagery data, *Proceedings of the 2019 IEEE International Geoscience and Remote Sensing Symposium IGARSS'2019*, Yokohama, Japan, July 28 – August 2, 2019, 9646–9649.