2-2018

Why Learning Has Aha-Moments and Why We Should Also Reward Effort, Not Just Results

Gerargo Uranga
The University of Texas at El Paso, guranga@miners.utep.edu

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

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Comments:
Technical Report: UTEP-CS-18-08

Why Learning Has Aha-Moments and Why We Should Also Reward Effort, Not Just Results

Gerargo Uranga[1], Vladik Kreinovich[1], and Olga Kosheleva[2]
Departments of [1]Computer Science and [2]Teacher Education
University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA
guranga@miners.utep.edu, vladik@utep.edu, olgak@utep.edu

Abstract

Traditionally, in machine learning, the quality of the result improves steadily with time (usually slowly but still steadily). However, as we start applying reinforcement learning techniques to solve complex tasks – such as teaching a computer to play a complex game like Go – we often encounter a situation in which for a long time, then is no improvement, and then suddenly, the system's efficiency jumps almost to its maximum. A similar phenomenon occurs in human learning, where it is known as the aha-moment. In this paper, we provide a possible explanation for this phenomenon, and show that this explanation leads to the need to reward students for effort as well, not only for their results.

1 Formulation of the Problem

Need for machine learning. In many practical situations, we want to be able to find the values of a quantity y based on the values of some related quantities $x_1, \ldots, x_n$. For this, we can use situations $k = 1, \ldots, N$ when we know the values of both $x_1^{(k)}, \ldots, x_n^{(k)}$ and $y^{(k)}$. Determining the dependence $y = f(x_1, \ldots, x_n)$ is known as *machine learning*; see, e.g., [1, 3].

For example, we want to be able to predict the volcano eruptions based on the seismic activity preceding this eruption; see, e.g., [5, 6].

Need for reinforcement learning. In some cases, the existing patterns are not sufficient for learning; in this case, it is desirable to come up with additional patterns – and, ideally, a computer should tell us which patterns to look for to get the best learning. Such situation is known as *reinforced learning*; see, e.g., [2, 8, 10].

Reinforcement learning is especially important if we want to teach a computer to play a complex game like Go. In this case, we want to find the best move y based on the values x_i that describe the current state (and, if needed, the past states and moves).

In such situations, in addition to relying on the record of previous games, it is often desirable to test some new possible moves – by using a computer simulation of the corresponding game.

Aha-moments. Traditionally, in the process of machine learning, the efficiency of the resulting dependence increases as we continue learning. For example, as learning continues, the values $f_{\rm cur}\left(x_1^{(k)},\ldots,x_n^{(k)}\right)$ obtained by the current state of the system get closer and close to the desired values $y^{(k)}$. If we are learning how to play a game, then the quality of the corresponding strategy $f_{\rm cur}(x_1,\ldots,x_n)$ steadily increases as we continue learning.

This improvement may be slow, it may have temporary setbacks, but overall, we tend to observe a steady improvement.

However, as the learning tasks become more and more complex – e.g., to the level of playing Go – while the system eventually learns, it does not show a steady increase at all:

- for a long times, there is no visible increase in the efficiency of the resulting system;
- then, in a relatively short period of time, the system's efficiency jumps to its maximum;

see, e.g., [4] and references therein.

This is similar to so-called *aha-moment* observed during human learning (see, e.g., [9]), when:

- a student first does not grasp a concept (and thus, cannot solve related problem)
- until suddenly, he or she gets a good understanding of it.

Why aha-moment? Why do we often observe such aha-moments? In this paper, we provide a possible explanation for this phenomenon – and we also show how this explanation can effect human learning.

2 Analysis of the Problem

Case study. Let us consider a typical game-learning environment, where we want to select the best strategy. Let $a_1,\ldots,a_m$ be parameters that describe a strategy.

We want to come up with a strategy that works best "on average", to be applied in the real world. Of course, for individual games, with individual opponents, other strategies may work better – strategies that take into account the individual features of the corresponding players.

Analysis of the case study. Let $a^{(0)}=\left(a_1^{(0)},\ldots,a_m^{(0)}\right)$ be the desired strategy which is the best on average. This strategy works the best against the "average"

player, i.e., a player with average values of appropriate characteristics – such as skill, memory, desire to take risks, etc. A deviation of any characteristic of player from its average value, in general, leads to the fact that the strategy $a = (a_1, \ldots, a_m)$ optimal for players with this deviation will be slightly different from $a^{(0)}$: $a_i = a_i^{(0)} + \Delta a_i$ for some small Δa_i.

In general, we have many different characteristic describing a player. They can be viewed as reasonably independent ones. So, for a randomly selected player, the deviation of the strategy optimal for this player from $a^{(0)}$ is a joint effect of all the differences from this person's characteristics and the average values of all these characteristics.

Resulting distribution. It is known that in general, the probability distribution of a joint effect of many similar random variables is close to Gaussian; the corresponding mathematical result is known as the Central Limit Theorem; see, e.g., [7].

Thus, we can conclude that the strategy which is optimal for a given randomly selected player is normally distributed around $a^{(0)}$.

How the quality of a given strategy $a = (a_1, \ldots, a_m)$ depends on the parameters a_i. A natural way to gauge the quality of a strategy is by measuring how successful it is when playing against a randomly selected opponent. In other words, as a quality of a strategy a, it is reasonable to take the probability $p(a)$ that this strategy defeats a randomly selected opponent.

Since we have shown that the corresponding distribution is Gaussian (normal), the dependence of this probability on a_i has the Gaussian form:

$$p(a) = \text{const} \cdot \exp\left(-Q\left(\Delta a\right)\right),$$

where $\Delta a \stackrel{\text{def}}{=} a - a^{(0)}$ (i.e., $\Delta a_i = a_i - a_i^{(0)}$), and $Q(\Delta a)$ is a quadratic form:

$$Q(\Delta a) = \sum_{i=1}^{m} \sum_{j=1}^{m} q_{ij} \cdot \Delta a_i \cdot \Delta a_j.$$

Comment. In a complex game like Go, a random strategy never wins. So:

- for most of the strategies a, the value $p(a)$ of the corresponding objective function is close to 0, and
- there is a small area for which $p(a)$ is significantly different from 0.

3 Why Learning Has Aha-Moments: A Possible Explanation

What we plan to do. The above analysis shows how the corresponding objective function $p(a)$ depends on the parameters a_i that describes a strategy $a = (a_1, \ldots, a_m)$.

Let us describe how this shape of the objective function affects the learning process.

How the objective functions are optimized in machine learning algorithms. In most machine learning algorithms, the objective function is optimized by using the gradient descent ; see, e.g., [1, 3] (in this case, since we are maximizing the probability of success, it is gradient accent). Crudely speaking, at each step, instead of the original values $a = (a_1, \ldots, a_m)$, we have new values $a_i + \delta a_i$, where

$$\delta a_i = \lambda \cdot \frac{\partial p}{\partial a_i}.$$

So how does this work for our objective function. In the beginning, we do not have a good strategy of playing Go or a similar game. Thus, we start with a strategy a_{init} for which $p(a_{\text{init}}) \approx 0$.

For the Gaussian function $p(a)$, when $p(a) \approx 0$, the partial derivatives $\dfrac{\partial p}{\partial a_i}$ are also close to 0, and for the resulting values $a + \delta a$, we still get $p(a + \delta a) \approx 0$.

So, indeed, in the beginning, there is no visible improvement of the objective function. This does not mean, however, that the gradient method does not work: it definitely brings us closer to $a^{(0)}$ and eventually, we will get close enough to $a^{(0)}$ to have $p(a) > 0$. But we will see an improvement only when we get very close to $a^{(0)}$ – within 2σ or so of the corresponding probability distribution.

Example. Let us illustrate the above behavior on the 1-D example, when $a = a_1$, $Q(a) = k \cdot \left(a - a^{(0)}\right)^2$, and

$$p(a) = \text{const} \cdot \exp\left(-k \cdot \left(a - a^{(0)}\right)^2\right).$$

(For multi-D case, the formulas are similar.)

In this case,

$$\frac{\partial p}{\partial a} = \text{const} \cdot \exp\left(-k \cdot \left(a - a^{(0)}\right)^2\right) \cdot (-2k) \cdot \left(a - a^{(0)}\right),$$

hence

$$\delta a = \lambda \cdot \frac{\partial p}{\partial a} = \lambda \cdot \text{const} \cdot \exp\left(-k \cdot \left(a - a^{(0)}\right)^2\right) \cdot (-2k) \cdot \left(a - a^{(0)}\right),$$

i.e.,

$$\delta a = \lambda \cdot p(a) \cdot (-2k) \cdot \left(a - a^{(0)}\right).$$

Since $p(a) \approx 0$, we get $\delta a \approx 0$ and thus, still $p(a + \delta a) \approx 0$ as well. This means that we do not see any visible improvement.

On the other hand, for the new value $a + \delta a$, we have

$$(a + \delta a) - a^{(0)} = \left(a - a^{(0)}\right) + \delta a = \left(a - a^{(0)}\right) - 2k \cdot \lambda \cdot p(a) \cdot \left(a - a^{(0)}\right) =$$

$$\left(a - a^{(0)}\right) \cdot (1 - 2k \cdot \lambda \cdot p(a)).$$

This shows that we do get closer to $a^{(0)}$.

Since we get closer to $a^{(0)}$, the value $p(a)$ increases, so on the next iteration, we get an even better improvement, and eventually, we will reach $a^{(0)}$.

This explains the ubiquity of aha-moments. The above analysis explains the ubiquity of aha-moments:

- for a long time, we do not see any visible improvement of the objective function,
- until we get close to $a^{(0)}$, at which point we will see an improvement.

This is exactly what we observe as the aha-moment.

4 Why We Should Also Reward Effort, Not Just Results

What happens in human learning. The above analysis has an important consequence to human learning – for which similar aha-moments are also ubiquitous. What this analysis shows is that in the beginning, while the student is on his/her way to learning, there is no visible improvement in the student's ability to solve the corresponding problems.

What if we only award results. If we gauge the students's success – as often happens – only by the results of learning, i.e., by the student's ability to solve the corresponding problems, we will not see any improvement – in spite of the fact that the student actively works.

In this case, the resulting bad grade would discourage the student from further attempts to learn – although in reality, the learning process is on the right track.

Conclusion: we also need to reward efforts. A natural conclusion is that to avoid such discouragement, we need to reward *efforts*, not only results – otherwise students will be discouraged from learning and mastering complex topics.

Acknowledgments

This work was supported in part by the US National Science Foundation grant HRD-1242122.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[2] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, *RL^2: Fast Reinformcement Learning via Slow Reinforcement Learning*, arXiv:1611.02779v2, 10 Nov 2016.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts, 2016.

[4] A. Juliani, *Learning Policies for Learning Policies – Meta Reinforcement Learning (RL^2) in Tensorflow*, posted at http://hackernoon.com, https://hackernoon.com/learning-policies-for-learning-policies-meta-reinforcement-learning-rl%C2%B2-in-tensorflow-b15b592a2ddf

[5] J. Parra, O. Fuentes, E. Anthony, and V. Kreinovich, "Use of machine learning to analyze and – hopefully – predict volcano activity", *Acta Politechnica Hungarica*, 2017, Vol. 14, No. 3, pp. 209–221.

[6] J. Parra, O. Fuentes, E. Anthony, and V. Kreinovich, "Prediction of volcanic eruptions: case study of rare events in chaotic systems with delay", *Proceedings of the IEEE Conference on Systems, Man, and Cybernetics SMC'2017*, Banff, Canada, October 5–8, 2017, pp. 351–356.

[7] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman and Hall/CRC, Boca Raton, Florida, 2011.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, Massachusetts, 1998; a new deition is being prepared.

[9] E. Thompson, *Mind in Life: Biology, Phenoemonoly, and the Sciences of Mind*, Belknap Press, Cambridge, Massachusetts, 2010.

[10] J. X. Wang, Z. Kirth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumarn, and M. Botvinick, *Learning to Reinforcement Learn*, arXiv:1611.05763v3, 23 Jan 2017.