Error Predictive Learning

by Sven Nilsen, 2018

In this paper I represent a simple modification to the error function that yields significant speedups for curve-fitting problems in machine learning. It is inspired by the theory of zen rational agents who can imagine themselves having more time to think. The general idea might be used in other applications as well. The basic principle is to add terms to the error function that require a prediction of the error, another term for predicting the error in the prediction of the error, and so on. This approach is very flexible because error predictions depend on how the learning algorithm climbs the gradient. Since the error prediction is consistent with the gradient climbing algorithm through a computational feedback relationship, it might be possible to apply this idea to many kinds of algorithms for performance gains.

In a curve-fitting problem, a typical way of defining an error of \`f\` fitted to \`g\` is:

$$\mathrm{error}_0 := \sum x \{ (f(x) - g(x))^2 \}$$

In error predictive learning, instead of training on the \`$\mathrm{error}_0$\` function, one picks a higher error function that extends the error recursively by adding some positive valued function of the difference between the lower error function and the predicted error:

$$\begin{aligned}
&\mathrm{error}_1 := \mathrm{error}_0 + \mathrm{abs}(\mathrm{error}_0 - \mathrm{predicted\_error}_0) \\
&\mathrm{error}_2 := \mathrm{error}_1 + \mathrm{abs}(\mathrm{error}_1 - \mathrm{predicted\_error}_1) \\
&\ldots \\
&\mathrm{error}_n := \mathrm{error}_{n-1} + \mathrm{abs}(\mathrm{error}_{n-1} - \mathrm{predicted\_error}_{n-1})
\end{aligned}$$

This class of error functions has the property that in order to drive the error to zero, the learning algorithm must also learn to predict that the error is zero, that this prediction has zero error, and so on. With other words, it does not just learn to give the correct answer, but it learns to be confident that the answer is correct, that the confidence is justified, and so on.

In many problems, the predicted error (\`$\mathrm{predicted\_error}_0$\`) never vanishes because the function to be learned is non-deterministic. However, this can be context dependent. Under some circumstances, the predicted error can easily be determined, for example when rolling a dice. If one believes that rolling 6 happens 1/6 of the time on average and you receive answer \`1\` if the dice rolls 6 and \`0\` when it does not, then the expected error is \`5/6\` when guessing \`1\`, but only \`1/6\` when guessing \`0\`. This means that in order to minimize the total error in a guessing game where you have only one chance, the algorithm learns to guess \`0\` with a predicted error \`1/6\`, but also with a predicted error in that error equal to \`1/6\`, because the nature of the dice does not change over time. While it is correct that a dice rolls the guess \`0\` one sixth of the times in the long run, one can not use the confidence in the belief that the dice has this long term behavior directly, because at any given moment, believing that you are \`1/6\` wrong about your guess means you should guess \`1/6\` instead. On the other hand, you expect to be \`1/6\` wrong one sixth of the time, which also is expected to be wrong sixth of the time, etc. This is the same as believing the guess \`0\` is wrong \`1/6\` of the time with 100% confidence. However, it is not possible to encode this 100% belief in the error function, because with an infinite number of error predictive layers the error adds up to infinity for any non-deterministic problem. Instead, one only adds sufficiently enough error predictive layers and estimate the long term belief by taking the difference.