Draft Version

MACHINE LEARNING YEARNING

Technical Strategy for AI Engineers, In the Era of Deep Learning

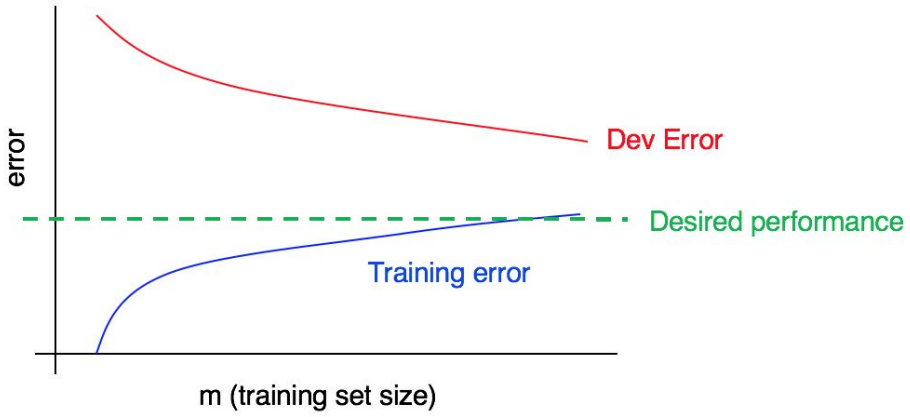
ANDREW NG

deeplearning.ai

Machine Learning Yearning is a
deeplearning.ai project.

31 Interpreting learning curves: Other cases

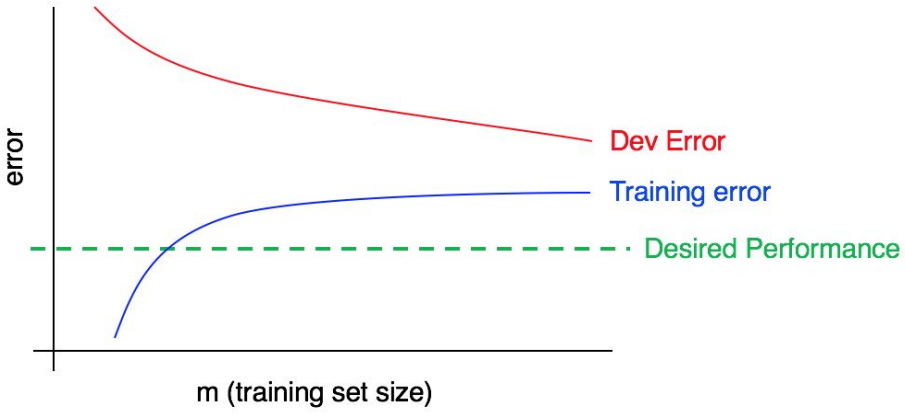
Consider this learning curve:



Does this plot indicate high bias, high variance, or both?

The blue training error curve is relatively low, and the red dev error curve is much higher than the blue training error. Thus, the bias is small, but the variance is large. Adding more training data will probably help close the gap between dev error and training error.

Now, consider this:



This time, the training error is large, as it is much higher than the desired level of performance. The dev error is also much larger than the training error. Thus, you have significant bias and significant variance. You will have to find a way to reduce both bias and variance in your algorithm.

32 Plotting learning curves

Suppose you have a very small training set of 100 examples. You train your algorithm using a randomly chosen subset of 10 examples, then 20 examples, then 30, up to 100, increasing the number of examples by intervals of ten. You then use these 10 data points to plot your learning curve. You might find that the curve looks slightly noisy (meaning that the values are higher/lower than expected) at the smaller training set sizes.

When training on just 10 randomly chosen examples, you might be unlucky and have a particularly "bad" training set, such as one with many ambiguous/mislabeled examples. Or, you might get lucky and get a particularly "good" training set. Having a small training set means that the dev and training errors may randomly fluctuate.

If your machine learning application is heavily skewed toward one class (such as a cat classification task where the fraction of negative examples is much larger than positive examples), or if it has a huge number of classes (such as recognizing 100 different animal species), then the chance of selecting an especially "unrepresentative" or bad training set is also larger. For example, if 80% of your examples are negative examples (y=0), and only 20% are positive examples (y=1), then there is a chance that a training set of 10 examples contains only negative examples, thus making it very difficult for the algorithm to learn something meaningful.

If the noise in the training curve makes it hard to see the true trends, here are two solutions:

- Instead of training just one model on 10 examples, instead select several (say 3-10) different randomly chosen training sets of 10 examples by sampling with replacement[1] from your original set of 100. Train a different model on each of these, and compute the training and dev set error of each of the resulting models. Compute and plot the average training error and average dev set error.
- If your training set is skewed towards one class, or if it has many classes, choose a "balanced" subset instead of 10 training examples at random out of the set of 100. For example, you can make sure that 2/10 of the examples are positive examples, and 8/10 are

[1] Here's what sampling *with replacement* means: You would randomly pick 10 different examples out of the 100 to form your first training set. Then to form the second training set, you would again pick 10 examples, but without taking into account what had been chosen in the first training set. Thus, it is possible for one specific example to appear in both the first and second training sets. In contrast, if you were sampling *without replacement*, the second training set would be chosen from just the 90 examples that had not been chosen the first time around. In practice, sampling with or without replacement shouldn't make a huge difference, but the former is common practice.

negative. More generally, you can make sure the fraction of examples from each class is as close as possible to the overall fraction in the original training set.

I would not bother with either of these techniques unless you have already tried plotting learning curves and concluded that the curves are too noisy to see the underlying trends. If your training set is large—say over 10,000 examples—and your class distribution is not very skewed, you probably won't need these techniques.

Finally, plotting a learning curve may be computationally expensive: For example, you might have to train ten models with 1,000, then 2,000, all the way up to 10,000 examples. Training models with small datasets is much faster than training models with large datasets. Thus, instead of evenly spacing out the training set sizes on a linear scale as above, you might train models with 1,000, 2,000, 4,000, 6,000, and 10,000 examples. This should still give you a clear sense of the trends in the learning curves. Of course, this technique is relevant only if the computational cost of training all the additional models is significant.