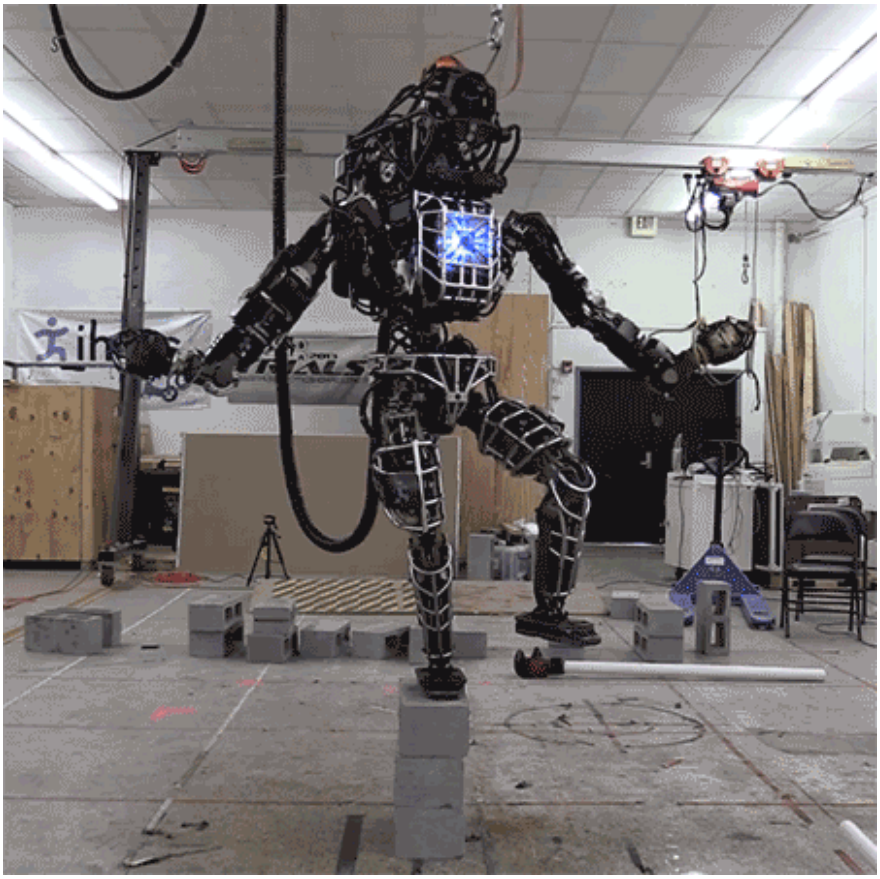
Learning Machine Learning — Part 6: Bias, Variance and Error Metrics

Ryan Gotesman Follow
Apr 25, 2018 · 6 min read



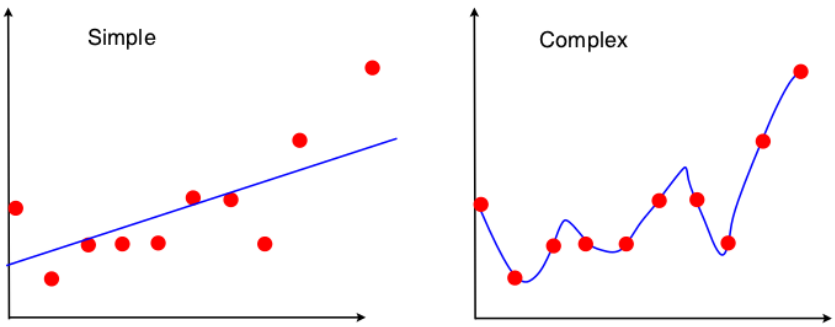
Week 6 of the course was much lighter on math and focused on pragmatic decisions you will have to make when designing ML programs. I found this week very interesting and suspect it conveyed some of the most useful information of the course.

The first topic that we covered was the idea of machine learning diagnostics which are various ways of assessing the characteristics of your ML model. To calculate these characteristics we need to split our data into 3 sets: a training, validation and test set. Normally we allocate the majority of the data, say 60%, to the training set and split the remainder between the validation and test sets.

Say we were interested in developing linear and quadratic functions to model our data. First we find the optimal parameters of these 2

candidate models based off the training set. We then test these models on the validation set and pick the one with the highest accuracy. Finally, we run this chosen model on our test set to see how generalizable it is to other data. You might ask why we split the data in this way and the reason is to ensure we preserve some data we have never seen before that can be used to test the accuracy of our model. If you test the accuracy of your model on any data that was used to train it, the accuracy will likely be exaggerated and we want to have an unbiased measure of the model's performance before testing it in the field. Splitting the data allows us to do that.

We can now introduce some important terms: bias and variance. Both are bad for different reasons. We say a model has high bias when it is too simple to capture the complexity of the data. This would be like using a linear model for data that has a quadratic form. In short, high bias means our model underfits the data. In contrast, high variance is when our model is too complex like using a 100 degree polynomial for that quadratic data. In this situation we overfit the data. This can be visually understood by considering the graphs below. Bias and variance are a problem because a model that has too much of either fails to generalize. It is common to refer to the bias-variance tradeoff because when we have more of one we have less of the other.



After learning about these concepts I realized bias and variance had been with us since the start of the course. If you consider the cost function for regularized linear regression:

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

you'll see that it can be rewritten as:

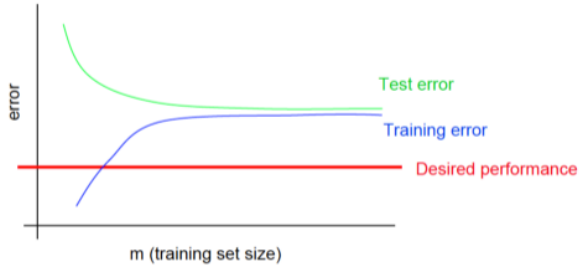
$$J = A + \lambda * B$$

where, in a way, λ dictates how we go about balancing the bias-variance tradeoff. Consider if λ is very large. Then to minimize *J* we know we must make *B* small, but this means making most of our parameters small in which case we are dealing with a model that is more prone to having high bias rather than high variance. In contrast, if λ is small, *B* is not much of a concern and instead we want to minimize *A*. The better fit our model is to the training data the smaller *A* will be and we see that in this case our model is more likely to have high variance than high bias. Finding the optimal value of λ to balance the bias-variance tradeoff is key to coming up with a good model and can be achieved by testing various values of λ in the validation set.

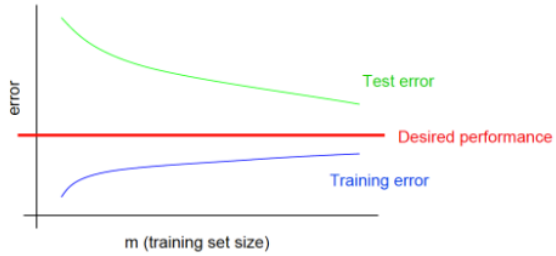
We can test if our models have high bias or variance through the use of learning curves. To create a learning curve we take *m* random examples from our training set, train our candidate model and calculate the error of said model on the *m* training examples and the entire validation set. We keep doing this for larger and larger values of *m* and plot the training and validation errors as a function of *m*. Different graphs will emerge depending on whether your model suffers from high bias or high variance.

For instance, with high bias, regardless of how much training data you have the model will be a poor fit. You can have all the data in the world but if you're only interested in horizontal lines you can't model much. So as *m* increases the training error will get large. The model will be just as bad in the validation set so we expect the validation error to be similarly large. If we consider what happens when our model has high variance we'll note that the training error will be very low, since we're overfitting, and as *m* increases the error will slowly creep upward. Turning to the validation error, since our model is overfitted it poorly generalizes so the validation error will be large. As *m* gets larger the model will learn from more representative data and the validation error will start to move down. These 2 scenarios are captured in the graphs below.

Typical learning curve for high bias:



Typical learning curve for high variance:



By simply checking the learning curves we can see if our model suffers from high bias or high variance. This is important because different strategies can be used to improve an algorithm depending on its shortcomings. For instance, more data will generally improve algorithms with high variance but not high bias. Conversely, adding more complex features usually improves models with high bias but not high variance. These 2 graphs can serve as an excellent guide when deciding how to improve the model.

The last topic covered this week was error metrics. When making changes to your model it is helpful to have a single number that captures the model's performance. This helps us quickly figure out whether the change we made improved or hurt the model. A commonly used error metric is accuracy but this metric does not work for skewed data sets.

Consider a data set where 99% of cases are negative and only 1% is positive. By always classifying an example as negative we can get 99% accuracy even though this is a terrible model. We can get around this problem by defining 2 new metrics, precision and recall as:

$$Precision\ (P) = \frac{\#\ True\ Positives}{\#\ Predicted\ Positive} \qquad Recall\ (R) = \frac{\#\ True\ Positives}{\#\ Actual\ Positives}$$

Note that if we always guess the example is negative both precision and recall will be 0 and if we always guess the example is positive, precision will be 1%. We can combine these 2 into a single error metric, called F1, defined as:

$$F1 = 2\frac{PR}{P+R}$$

Note that F1 = 1 when P = R =1 and that if either P or R is small F1 will also be small.

That covers most of everything in week 6. Next week we cover support vector machines which I have wanted to know more about for a while.

Until next time.