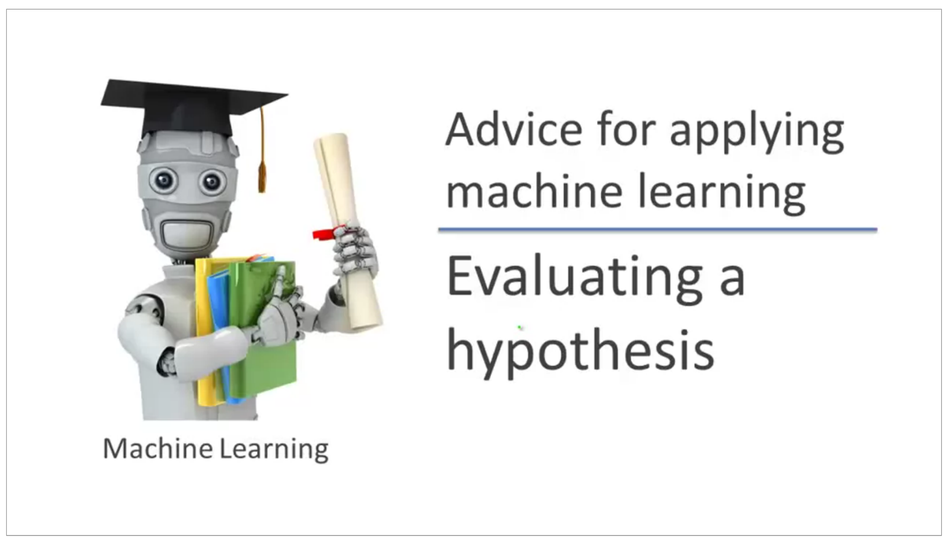
Evaluating a Hypothesis

Notes **Discuss**



Save note **Download**

Transcript

English **Help Us Translate**

0:00 In this video, I would like to talk about how to evaluate a hypothesis that has been learned by your algorithm. In later videos, we will build on this to talk about how to prevent in the problems of overfitting and underfitting as well. When we fit the parameters of our learning algorithm we think about choosing the parameters to minimize the training error. One might think that getting a really low value of training error might be a good thing, but we have already seen that just because a hypothesis has low training error, that doesn't mean it is necessarily a good hypothesis. And we've already seen the example of how a hypothesis can overfit. And therefore fail to generalize the new examples not in the training set. So how do you tell if the hypothesis might be overfitting. In this simple example we could plot the hypothesis h of x and just see what was going on. But in general for problems with more features than just one feature, for problems with a large number of features like these it becomes hard or may be impossible to plot what the hypothesis looks like and so we need some other way to evaluate our hypothesis. The standard way to evaluate a learned hypothesis is as follows. Suppose we have a data set like this. Here I have just shown 10 training examples, but of course usually we may have dozens or hundreds or maybe thousands of training examples. In order to make sure we can evaluate our hypothesis, what we are going to do is split the data we have into two portions. The first portion is going to be our usual training set

1:42 and the second portion is going to be our test set, and a pretty typical split of this all the data we have into a training set and test set might be around say a 70%, 30% split. Worth more today to grade the training set and relatively less to the test set. And so now, if we have some data set, we run a sine of say 70% of the data to be our training set where here "m" is as usual our number of training examples and the remainder of our data might then be assigned to become our test set. And here, I'm going to use the notation m subscript test to denote the number of test examples. And so in general, this subscript test is going to denote examples that come from a test set so that x1 subscript test, y1 subscript test is my first test example which I guess in this example might be this example over here. Finally, one last detail whereas here I've drawn this as though the first 70% goes to the training set and the last 30% to the test set. If there is any sort of ordinary to the data. That should be better to send a random 70% of your data to the training set and a random 30% of your data to the test set. So if your data were already randomly sorted, you could just take the first 70% and last 30% that if your data were not randomly ordered, it would be better to randomly shuffle or to randomly reorder the examples in your training set. Before you know sending the first 70% in the training set and the last 30% of the test set. Here then is a fairly typical procedure for how you would train and test the learning algorithm and the learning regression. First, you learn the parameters theta from the training set so you minimize the usual training error objective j of theta, where j of theta here was defined using that 70% of all the data you have. There is only the training data. And then you would compute the test error. And I am going to denote the test error as j subscript test. And so what you do is take your parameter theta that you have learned from the training set, and plug it in here and compute your test set error. Which I am going to write as follows. So this is basically the average squared error as measured on your test set. It's pretty much what you'd expect. So if we run every test example through your hypothesis with parameter theta and just measure the squared error that your hypothesis has on your m subscript test, test examples. And of course, this is the definition of the test set error if we are using linear regression and using the squared error metric. How about if we were doing a classification problem and say using logistic regression instead. In that case, the procedure for training and testing say logistic regression is pretty similar first we will do the parameters from the training data, that first 70% of the data. And it will compute the test error as follows. It's the same objective function as we always use but we just logistic regression, except that now is define using our m subscript test, test examples. While this definition of the test set error j subscript test is perfectly reasonable. Sometimes there is an alternative test sets metric that might be easier to interpret, and that's the misclassification error. It's also called the zero one misclassification error, with zero one denoting that you either get an example right or you get an example wrong. Here's what I mean. Let me define the error of a prediction. That is h of x. And given the label y as equal to one if my hypothesis outputs the value greater than equal to five and Y is equal to zero or if my hypothesis outputs a value of less than 0.5 and y is equal to one, right, so both of these cases basic respond to if your hypothesis mislabeled the example assuming your threshold at an 0.5. So either thought it was more likely to be 1, but it was actually 0, or your hypothesis stored was more likely to be 0, but the label was actually 1. And otherwise, we define this error function to be zero. If your hypothesis basically classified the example y correctly. We could then define the test error, using the misclassification error metric to be one of the m tests of sum from i equals one to m subscript test of the error of h of x(i) test comma y(i). And so that's just my way of writing out that this is exactly the fraction of the examples in my test set that my hypothesis has mislabeled. And so that's the definition of the test set error using the misclassification error of the 0 1 misclassification metric. So that's the standard technique for evaluating how good a learned hypothesis is. In the next video, we will adapt these ideas to helping us do things like choose what features like the degree polynomial to use with the learning algorithm or choose the regularization parameter for learning algorithm.

Like **Dislike** **Report an issue** **Share**