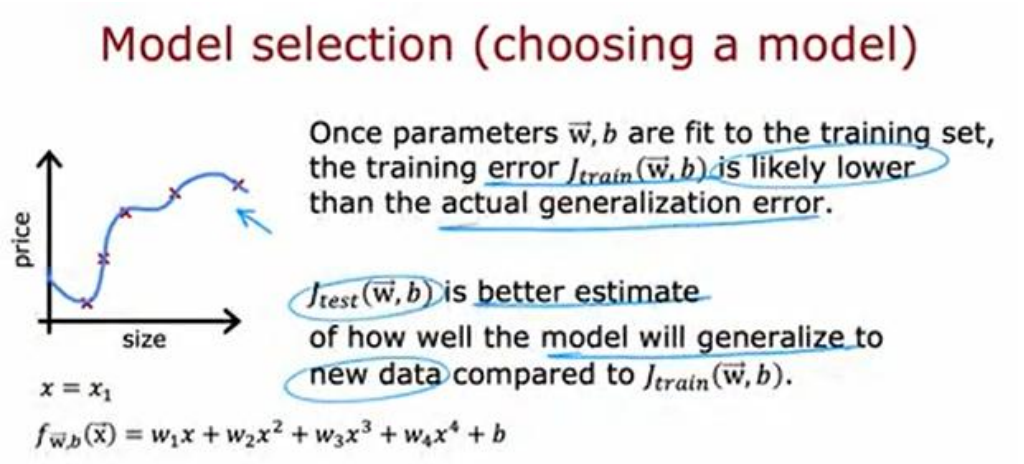
Model selection and training/cross-validation/test sets

In the last video, you saw how to use the test set to evaluate the performance of a model. Let's make one further refinement to that idea in this video, which allow you to use the technique, to automatically choose a good model for your machine learning algorithm. One thing we've seen is that once the model's parameters w and b have been fit to the training set.

The training error may not be a good indicator of how well the algorithm will do or how well it will generalize to new examples that were not in the training set, and in particular, for this example, the training error will be pretty much zero. That's likely much lower than the actual generalization error, and by that I mean the average error on new examples that were not in the training set.

What you saw on the last video is that J test the performance of the algorithm on examples, is not trained on, that will be a better indicator of how well the model will likely do on new data. By that I mean other data that's not in the training set.



Let's take a look at how this affects, how we might use a test set to choose a model for a given machine learning application. If a fitting a function to predict housing prices or some other regression problem, one model you might consider is to fit a linear model like this. This is a first-order polynomial and we're going to use d equals 1 on this slide to denote fitting a one or first-order polynomial.

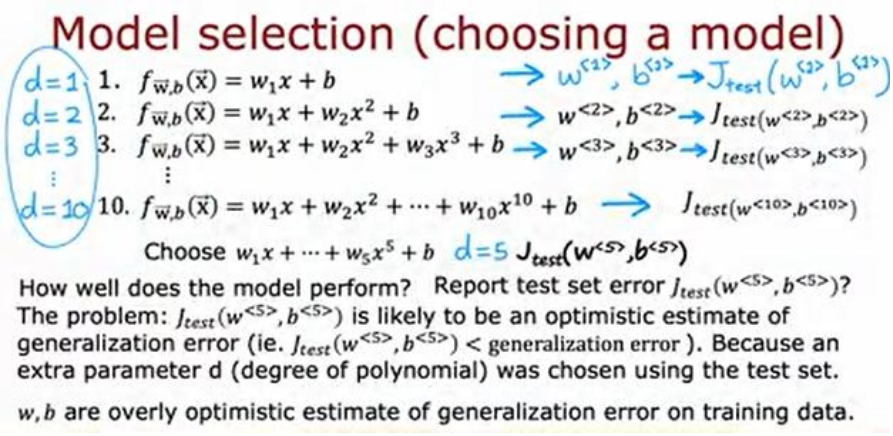
If you were to fit a model like this to your training set, you get some parameters, w and b, and you can then compute J tests to estimate how well does this generalize to new data? On this slide, I'm going to use w^1, b^1 to denote that these are the parameters you get if you were to fit a first order polynomial, a degree one, d equals 1 polynomial.

Now, you might also consider fitting a second-order polynomial or quadratic model, so this is the model. If you were to fit this to your training set, you would get some parameters, w^2, b^2, and you can then similarly evaluate those parameters on your test set and get J test w^2, b^2, and this will give you a sense of how well the second-order polynomial does.

You can go on to try d equals 3, that's a third order or a degree three polynomial that looks like this, and fit parameters and similarly get J test. You might keep doing this until, say you try up to a 10th order polynomial and you end up with J test of w^10, b^10. That gives you a sense of how well the 10th order polynomial is doing.

One procedure you could try, this turns out not to be the best procedure, but one thing you could try is, look at all of these J tests, and see which one gives you the lowest value. Say, you find that, J test for the fifth order polynomial for w^5, b^5 turns out to be the lowest. If that's the case, then you might decide that the fifth order polynomial d equals 5 does best, and choose that model for your application.

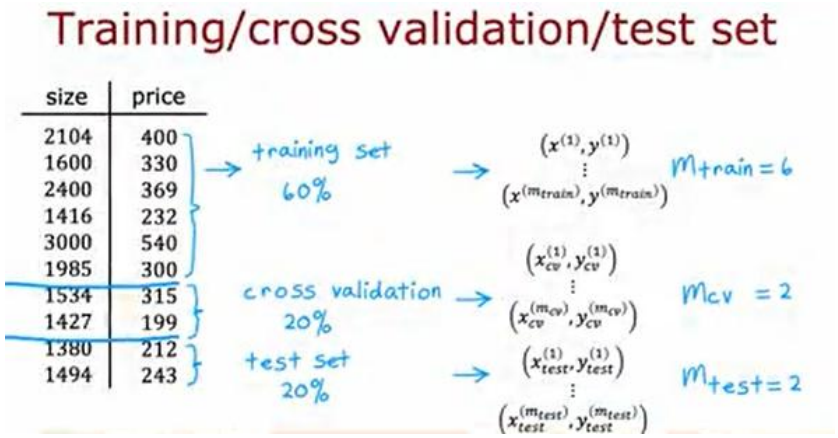
If you want to estimate how well this model performs, one thing you could do, but this turns out to be a slightly flawed procedure, is to report the test set error, J test w^5, b^5.



The reason this procedure is flawed is J test of w^5, b^5 is likely to be an optimistic estimate of the generalization error. In other words, it is likely to be lower than the actual generalization error, and the reason is, in the procedure we talked about on this slide with basic fits, one extra parameter, which is d, the degree of polynomial, and we chose this parameter using the test set.

On the previous slide, we saw that if you were to fit w, b to the training data, then the training data would be an overly optimistic estimate of generalization error. It turns out too, that if you want to choose the parameter d using the test set, then the test set J test is now an overly optimistic, that is lower than actual estimate of the generalization error.

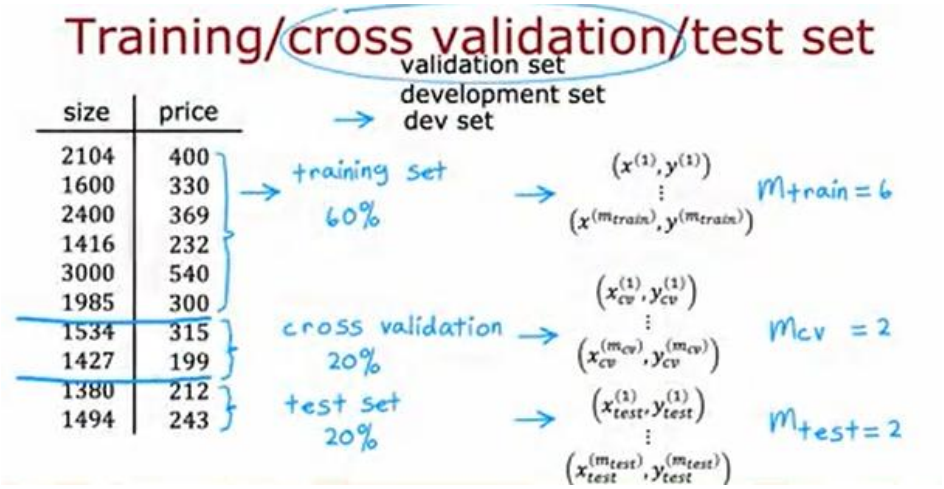
The procedure on this particular slide is flawed and I don't recommend using this. Instead, if you want to automatically choose a model, such as decide what degree polynomial to use. Here's how you modify the training and testing procedure in order to carry out model selection.



Whereby model selection, I mean choosing amongst different models, such as these 10 different models that you might contemplate using for your machine learning application. The way we'll modify the procedure is instead of splitting your data into just two subsets, the training set and the test set, we're going to split your data into three different subsets, which we're going to call the training set, the cross-validation set, and then also the test set.

Using our example from before of these 10 training examples, we might split it into putting 60 percent of the data into the training set and so the notation we'll use for the training set portion will be the same as before, except that now M train, the number of training examples will be six and we might put 20 percent of the data into the cross-validation set and a notation I'm going to use is x_cv of one, y_cv of one for the first cross-validation example. So cv stands for cross-validation, all the way down to x_cv of m_cv and y_cv of m_cv.

Where here, m_cv equals 2 in this example, is the number of cross-validation examples. Then finally we have the test set same as before, so x1 through x m tests and y1 through y m, where m tests equal to 2. This is the number of test examples.



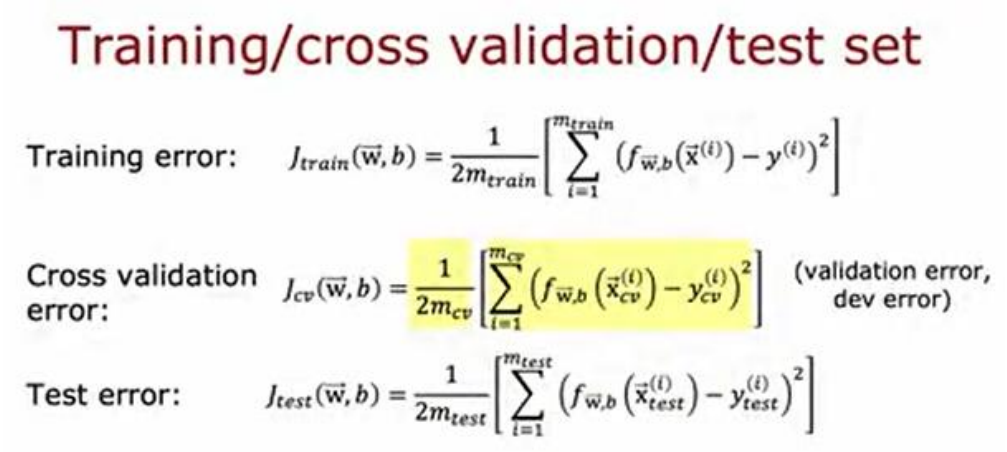
We'll see you on the next slide how to use the cross-validation set. The way we'll modify the procedure is you've already seen the training set and the test set and we're going to introduce a new subset of the data called the cross-validation set. The name cross-validation refers to that this is an extra dataset that we're going to use to check or cross check the validity or really the accuracy of different models.

I don t think it's a great name, but that is what people in machine learning have gotten to call this extra dataset. You may also hear people call this the validation set for short, it's just fewer syllables than cross-validation or in some applications, people also call this the development set. Means basically the same thing or for short. Sometimes you hear people call this the dev set, but all of these terms mean the same thing as cross-validation set.

I personally use the term dev set the most often because it's the shortest, fastest way to say it but cross-validation is pretty used a little bit more often by machine learning practitioners. Onto these three subsets of the data training set, cross-validation set, and test set, you can then compute the training error, the cross-validation error, and the test error using these three formulas.

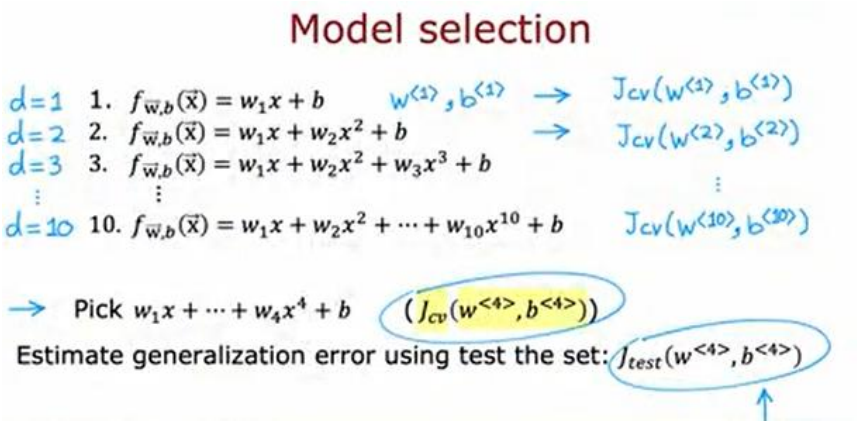
Whereas usual, none of these terms include the regularization term that is included in the training objective, and this new term in the middle, the cross-validation error is just the average over your m_cv cross-validation examples of the average say, squared error. This term, in addition to being

called cross-validation error, is also commonly called the validation error for short, or even the development set error, or the dev error.



Armed with these three measures of learning algorithm performance, this is how you can then go about carrying out model selection. You can, with the 10 models, same as earlier on this slide, with d equals 1, d equals 2, all the way up to a 10th degree or the 10th order polynomial, you can then fit the parameters w_1, b_1. But instead of evaluating this on your test set, you will instead evaluate these parameters on your cross-validation sets and compute J_cv of w1, b1, and similarly, for the second model, we get J_cv of w2, v2, and all the way down to J_cv of w10, b10.

Then, in order to choose a model, you will look at which model has the lowest cross-validation error, and concretely, let's say that J_cv of w4, b4 as low as, then what that means is you pick this fourth-order polynomial as the model you will use for this application. Finally, if you want to report out an estimate of the generalization error of how well this model will do on new data. You will do so using that third subset of your data, the test set and you report out Jtest of w4,b4.

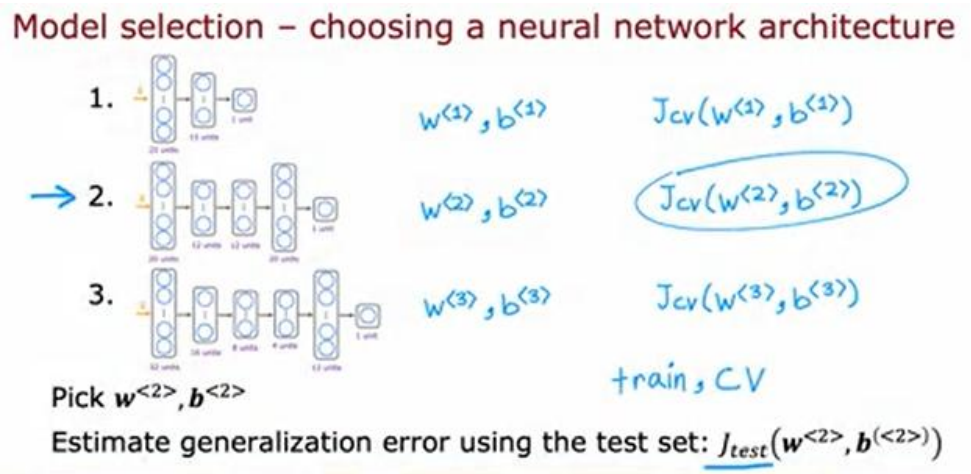


You notice that throughout this entire procedure, you had fit these parameters using the training set. You then chose the parameter d or chose the degree of polynomial using the cross-validation set and so up until this point, you've not fit any parameters, either w or b or d to the test set and that's why Jtest in this example will be fair estimate of the generalization error of this model thus parameters w4,b4.

This gives a better procedure for model selection and it lets you automatically make a decision like what order polynomial to choose for your linear regression model. This model selection procedure also works for choosing among other types of models.

For example, choosing a neural network architecture. If you are fitting a model for handwritten digit recognition, you might consider three models like this, maybe even a larger set of models than just me but here are a few different neural networks of small, somewhat larger, and then even larger. To help you decide how many layers do the neural network have and how many hidden units per layer should you have, you can then train all three of these models and end up with parameters w1, b1 for the first model, w2, b2 for the second model, and w3,b3 for the third model.

You can then evaluate the neural networks performance using Jcv, using your cross-validation set Since this is a classification problem, Jcv the most common choice would be to compute this as the fraction of cross-validation examples that the algorithm has misclassified. You would compute this using all three models and then pick the model with the lowest cross validation error.



If in this example, this has the lowest cross validation error, you will then pick the second neural network and use parameters trained on this model and finally, if you want to report out an estimate of the generalization error, you then use the test set to estimate how well the neural network that you just chose will do.

It's considered best practice in machine learning that if you have to make decisions about your model, such as fitting parameters or choosing the model architecture, such as neural network architecture or degree of polynomial if you're fitting a linear regression, to make all those decisions only using your training set and your cross-validation set, and to not look at the test set at all while you're still making decisions regarding your learning algorithm.

It's only after you've come up with one model as your final model to only then evaluate it on the test set and because you haven't made any decisions using the test set, that ensures that your test set is a fair and not overly optimistic estimate of how well your model will generalize to new data. That's model selection and this is actually a very widely used procedure. I use this all the time to automatically choose what model to use for a given machine learning application.

Earlier this week, I mentioned running diagnostics to decide how to improve the performance of a learning algorithm. Now that you have a way to evaluate learning algorithms and even automatically choose a model, let's dive more deeply into examples of some diagnostics. The most powerful diagnostic that I know of and that I used for a lot of machine learning applications is one called bias and variance.