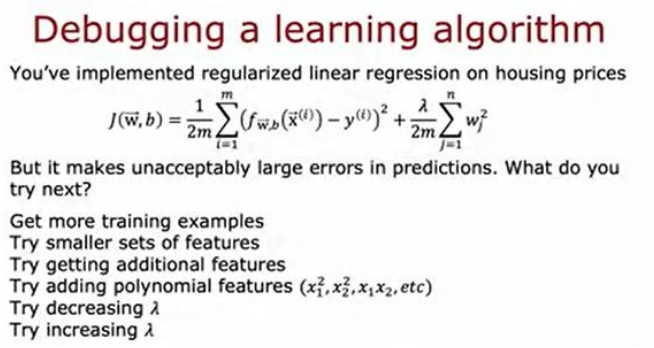
Deciding what to try next, revisited

You've seen how by looking at J train and Jcv, that is the training error and cross-validation error, or maybe even plotting a learning curve. You can try to get a sense of whether your learning algorithm has high bias or high variance. This is the procedure I routinely do when I'm training a learning algorithm more often look at the training error and cross-validation error to try to decide if my algorithm has high bias or high variance.

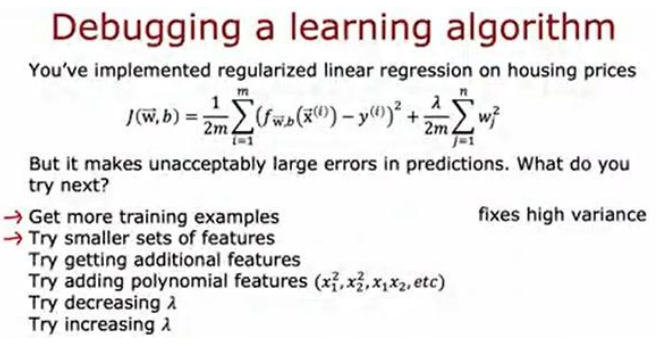
It turns out this will help you make better decisions about what to try next in order to improve the performance of your learning algorithm. Let's look at an example. This is actually the example that you saw earlier. If you've implemented regularized linear regression on predicting housing prices, but your algorithm makes unacceptably large errors in its predictions, what do you try next?



These were the six ideas that we had when we had looked over this slide earlier. Getting more training examples, try small set of features, additional features, and so on. It turns out that each of these six items either helps fix a high variance or a high bias problem. In particular, if your learning algorithm has high bias, three of these techniques will be useful. If your learning algorithm has high variance than a different three of these techniques will be useful. Let's see if we can figure out which is which. First one is get more training examples.

We saw in the last video that if your algorithm has high bias, then if the only thing we do is get more training data, that by itself probably won't help that much. But in contrast, if your algorithm has high variance, say it was overfitting to a very small training set, then getting more training examples will help a lot. This first option or getting more training examples helps to fix a high variance problem. How about the other five?

Do you think you can figure out which of the remaining five fix high bias or high variance problems? I'm going to go through the rest of them in this video in a minute but if you want it, you're free to pause the video and see if you can think through these five other things by yourself. Feel free to pause the video.

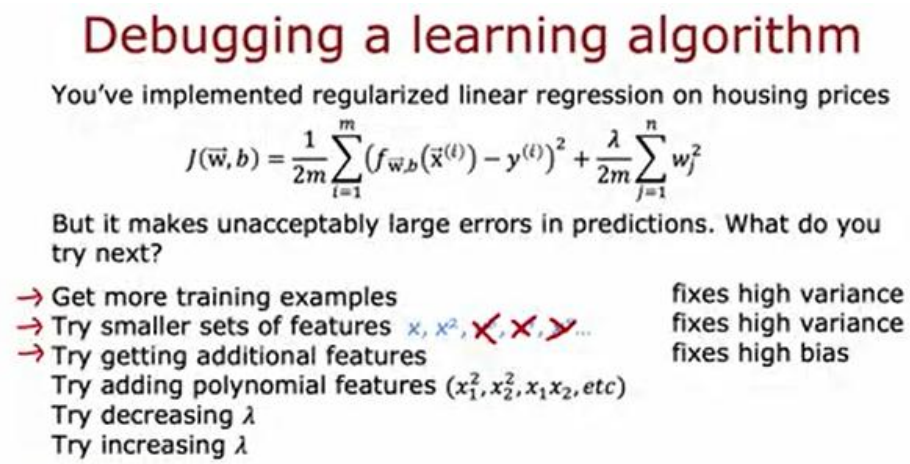


Just kidding, that was me pausing and not your video pausing. But seriously, if you want it, go ahead and pause the video and think through that you want or not and we'll go over these review in a minute. How about trying a smaller set of features? Sometimes if your learning algorithm has too many features, then it gives your algorithm too much flexibility to fit very complicated models. This is a little bit like if you had x, x squared, x cubed, x^4, x^5, and so on.

If only you were to eliminate a few of these, then your model won't be so complex and won't have such high variance. If you suspect that your algorithm has a lot of features that are not actually relevant or helpful to predict housing price, or if you suspect that you had even somewhat redundant features, then eliminating or reducing the number of features will help reduce the flexibility of your algorithm to overfit the data.

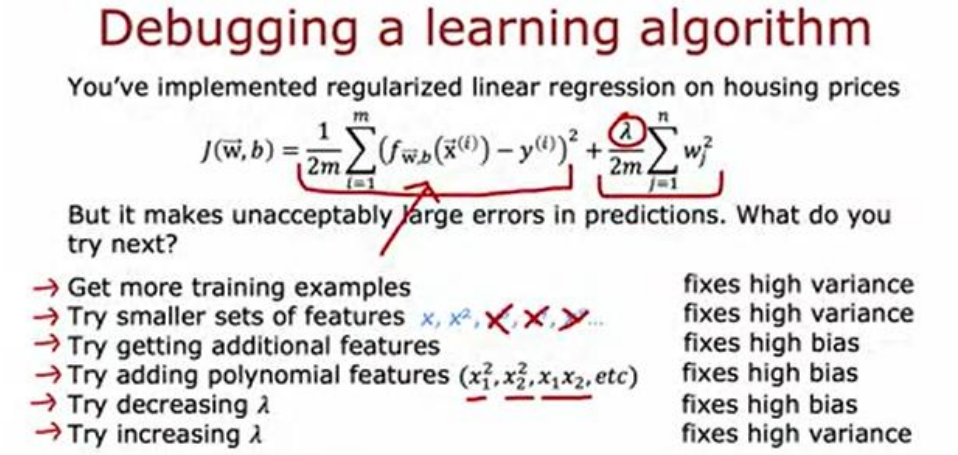
This is a tactic that will help you to fix high variance. Conversing, getting additional features, that's just adding additional features is the opposite of going to a smaller set of features. This will help you to fix a high bias problem. As a concrete example, if you're trying to predict the price of the house just based on the size, but it turns out that the price of house also really depends on the number of bedrooms and on the number of floors and on the age of the house, then the algorithm will never do that well unless you add in those additional features.

That's a high bias problem because you just can't do that well on the training set when only the size, is only when you tell the algorithm how many bedrooms are there, how many floors are there? What's the age of the house that it finally has enough information to even do better on the training set. Adding additional features is a way to fix a high bias problem.



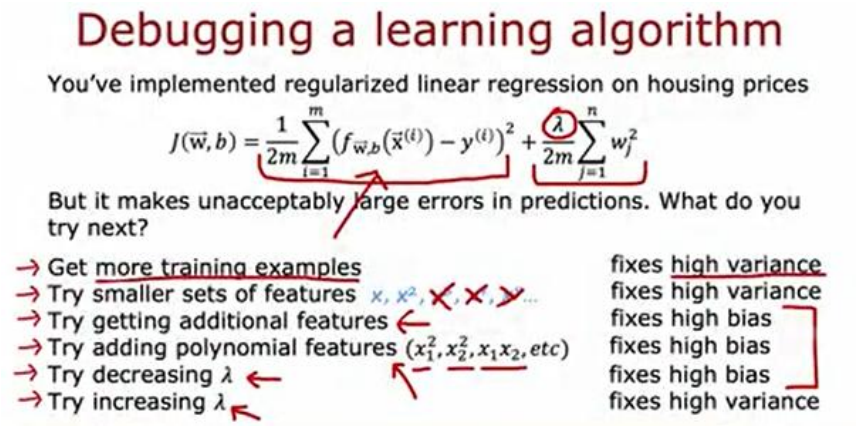
Adding polynomial features is a little bit like adding additional features. If you're linear functions, three-line can fit the training set that well, then adding additional polynomial features can help you do better on the training set, and helping you do better on the training set is a way to fix a high bias problem. Then decreasing Lambda means to use a lower value for the regularization parameter. That means we're going to pay less attention to this term and pay more attention to this term to try to do better on the training set.

Again, that helps you to fix a high bias problem. Finally, increasing Lambda, although that's the opposite of this, but that says you're overfitting the data. Increasing Lambda will make sense if is overfitting the training set, just putting too much attention to fit the training set, but at the expense of generalizing to new examples, and so increasing Lambda would force the algorithm to fit a smoother function, may be less wiggly function and use this to fix a high variance problem.



I realized that this was a lot of stuff on this slide. But the takeaways I hope you have are, if you find that your algorithm has high variance, then the two main ways to fix that are; neither get more training data or simplify your model. By simplifying model I mean, either get a smaller set of features or increase the regularization parameter Lambda. Your algorithm has less flexibility to fit very complex, very wiggly curves.

Conversely, if your algorithm has high bias, then that means is not doing well even on the training set. If that's the case, the main fixes are to make your model more powerful or to give them more flexibility to fit more complex or more wiggly functions. Some ways to do that are to give it additional features or add these polynomial features, or to decrease the regularization parameter Lambda.



Anyway, in case you're wondering if you should fix high bias by reducing the training set size, that doesn't actually help. If you reduce the training set size, you will fit the training set better, but that tends to worsen your cross-validation error and the performance of your learning algorithm, so don't randomly throw away training examples just to try to fix a high bias problem.

One of my PhD students from Stanford, many years after he'd already graduated from Stanford, once told me that while he was studying at Stanford, he learned about bias and variance and felt like he got it, he understood it. But that subsequently, after many years of work experience in a few different companies, he realized that bias and variance is one of those concepts that takes a short time to learn, but takes a lifetime to master.

Those were his exact words. Bias and variance is one of those very powerful ideas. When I'm training learning algorithms, I almost always try to figure out if it is high bias or high variance.

But the way you go about addressing that systematically is something that you will keep on getting better at through repeated practice. But you'll find that understanding these ideas will help you be much more effective at how you decide what to try next when developing a learning algorithm. Now, I know that we did go through a lot in this video and if you feel like, boy, this is a lot of stuff here, it's okay, don't worry about it.

Later this week in the practice labs and practice quizzes will also have additional opportunities to go over these ideas so that you can get additional practice. We're thinking about bias and variance of different learning algorithms. If it seems like a lot right now is okay, you get to practice these ideas later this week and hopefully deepen your understanding of them at that time. Before moving on, bias and variance also are very useful when thinking about how to train a neural network.