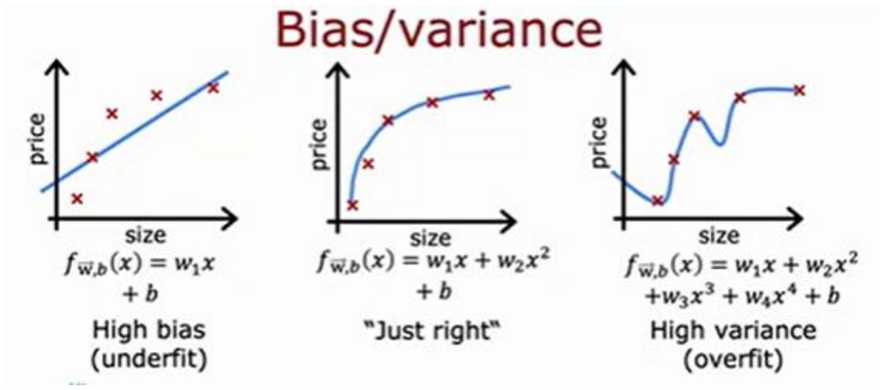
Diagnosing bias and variance

The typical workflow for developing a machine learning system goes like this: you have an idea, you train a model, and you almost always find that it's not yet performing as well as you'd like. When I train a machine learning model, it almost never works that well the first time. So, the key to the process of building a machine learning system is knowing how to decide what to do next to improve its performance.

These issues appear in many different applications, and examining the bias and variance of a learning algorithm gives you very good clues about what to try next. Let's look at what this means. You might remember that example from the first lecture on linear regression, where, given this dataset, if you were to fill a straight line to it, it wouldn't work very well. And we said that this algorithm had a high bias or that it wasn't suitable for this dataset . or if you were to fit a four-part polynomial, then it has a high variance or it is overfitted.

And in the middle, if you integrate a quadratic polynomial, then it looks pretty good, and he said it was just perfect. Since this is a problem with a single feature x, we could plot the function f and look at it like this. But if you had more features, you wouldn't be able to plot f and visualize whether it works well so easily.

So, instead of trying to look at graphs like that, a more systematic way to diagnose or discover if your algorithm has high bias or high variance would be to examine the performance of your algorithm on the training set and on the validation dataset. Let's look more specifically at the example on the left. If you were to compute JTRAIN, how would the algorithm perform on the training set? (AI, UO) Not very well.

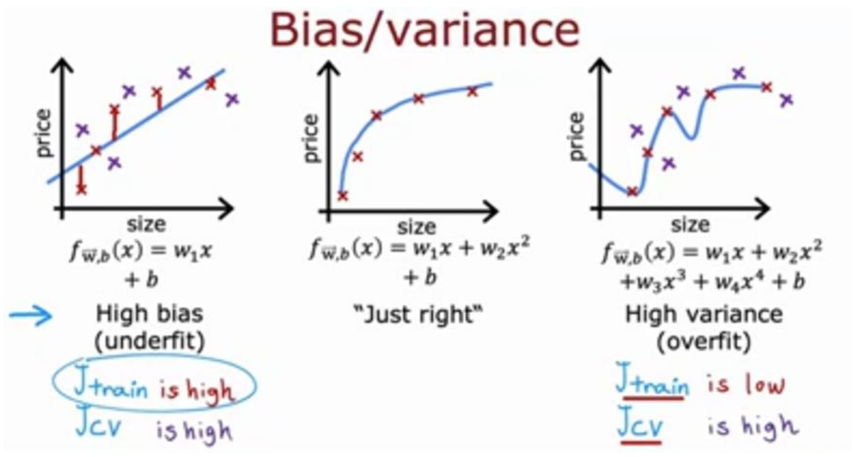


So I would say that JTRAIN would be high here because there are actually quite significant errors between the examples and the model's actual predictions. And what about JCV? Well, JCV would be high if you had some new examples, perhaps examples like this one, that the algorithm hadn't seen before. And here, the algorithm doesn't perform very well on examples it hasn't seen before either. So the JCV would also be high.

And one of the characteristics of an algorithm with a high bias, something that is underfitted, is that it doesn't even perform very well on the training set. So when JTRAIN is high, it gives you a strong indicator that this algorithm has a high bias. Now let's look at the example on the right.

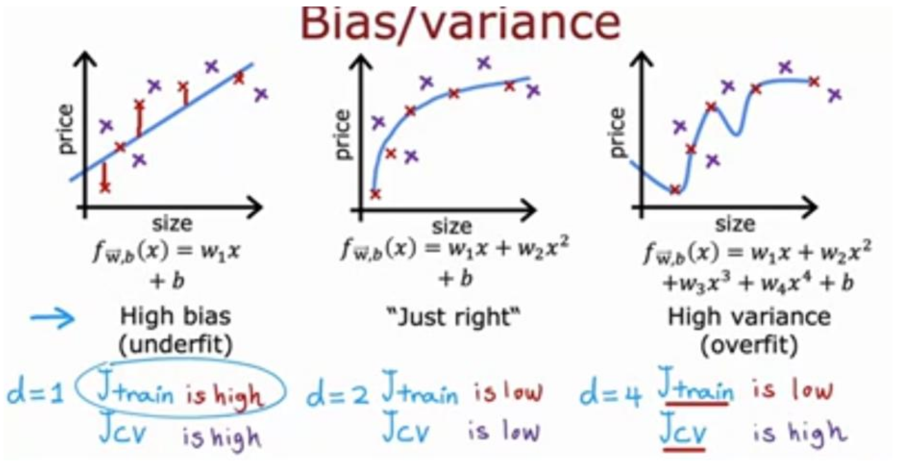
If you were to calculate JTrain, how does it perform on the training set? Actually, it performs very well on the training set. It fits the AI training data really well. So, JTrain here would be weak, but if you were to evaluate this model on other houses not part of the training set, you would find that

JCV, the cross-validation error, would be quite high. Thus, a characteristic signature or characteristic index indicating that your algorithm exhibits a high variance would be if JCV is significantly higher than JTrain. In other words, it performs much better with the data it has seen than with the data it hasn't. And this turns out to be a strong indicator that your algorithm has high variance.



By using the authentication server, AI, and CI, you can improve your unit of operation (UoO). Again, the goal of what we're doing is that by calculating JTRAIN and JCV and seeing if JTRAIN is high or if JCV is much higher than JTRAIN, this gives you an idea, even if you can't plot the function f, whether your algorithm has high bias or high variance. Finally, the middle case: the authentication server. If you look at JTRAIN, it's quite low because it performs rather well on the training set.

And if you were to examine some new examples, such as those from your cross-validation set, you'd find that JCV is also quite low. Therefore, the fact that JTRAIN isn't too high indicates that there's no high bias problem. And the fact that JCV isn't much worse than JTRAIN simply indicates that it also doesn't have a high variance problem. This is why this model, the quadratic model, seems to be a very good model for this application. So, to summarize, when IA is equal to one for a linear polynomial, Jtrain was high and JCV was high. When D is equal to 4, Jtrain was low, but JCV is high.



And when D equals 2, both were quite low. Let's now take a different look at bias and variance, using the Authentication Server and Business Intelligence (BI). Specifically, on the next slide, I'd like to show you how JTRAIN and JCV vary depending on the degree of the polynomial you're fitting. So let me draw a figure where the horizontal axis of this figure will be the degree of the polynomial we're fitting to the data.

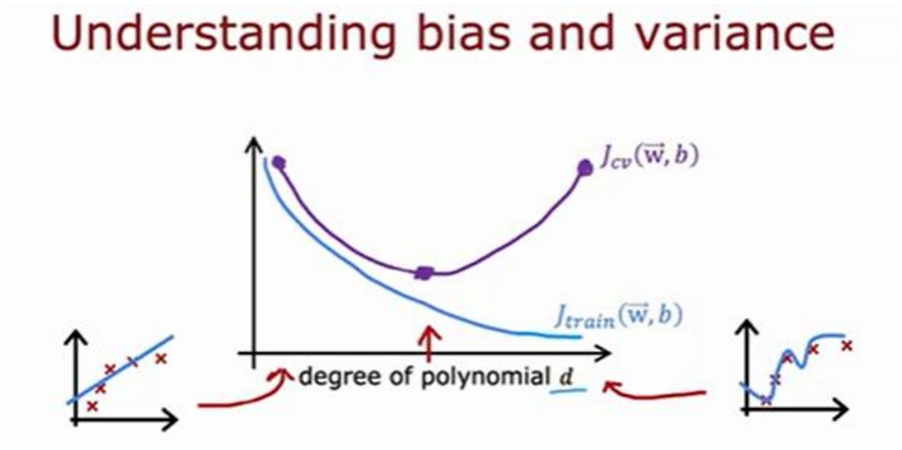
On the left will correspond to a small value of d, such as d equal to 1, which corresponds to a fill line. And on the right will correspond, say, to d equal to 4, or even higher values of d, where we're fitting this higher-order polynomial. So if you were to plot J-train of WB as a function of the degree of the polynomial, what you would find is that as you fit a polynomial of increasingly higher degree, here I'm assuming we're not using regularization, but as you As you adjust a polynomial of increasingly higher order, the training error will tend to decrease, because when you have a very simple linear function, it does not fit the training data very well.

When you fit a quadratic function, a third-order polynomial, or a fourth- order polynomial, it fits the training data better and better.

So as the degree of the polynomial increases, IA tends to decrease. Next, let's look at JCV, which is how well it performs with the data it couldn't fit. What we saw is that when D equals one when the degree of the polynomial is very low, JCV was quite high because it was underfitted, so it didn't perform well on the entire validation dataset.

And here on the right as well, when the degree of the polynomial is very high, say 4, it also doesn't perform well on the validation dataset, so it's also high. But if D were somewhere in between, say a second-order polynomial, then it would perform much better. So, if you were to vary the degree of the polynomial, you would actually get a curve that looks like this, going down and then up again, where...

If the degree of the polynomial is too low, it is underfitted and therefore does not perform well on the cross-validation set. If it is too high, it is overfitted and does not perform well on the cross-validation set.



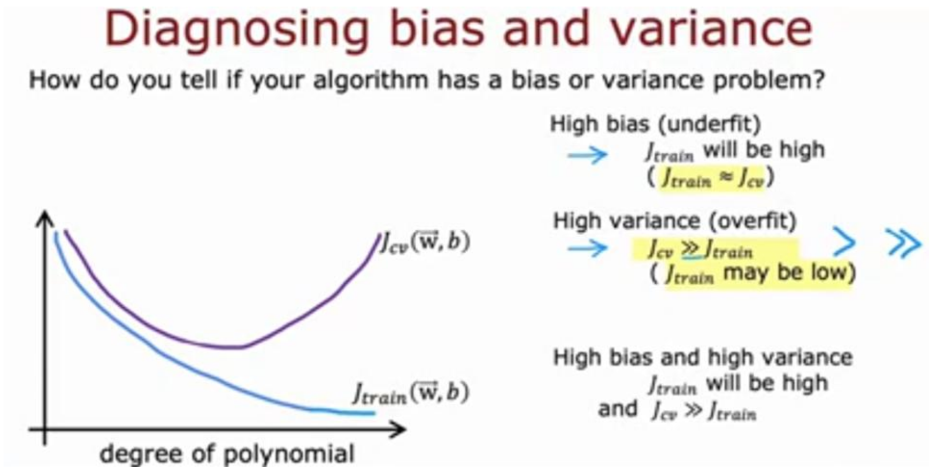
And it's only if it's somewhere in the middle that it's correct; that's why the second- order polynomial in our example ends up with a lower cross-validation error and neither high bias nor high variance. So, to summarize, how do you diagnose bias and variance in your learning algorithm ? If your learning algorithm has high bias or unvalidated data, the key indicator will be whether the J train is high.

This corresponds to the leftmost part of the curve, where the J train is high. And generally, the J train and J CV will be close to each other. And how do you diagnose if you have high variance? (AI, UO) Well, the key indicator for high variance will be whether JCV is much larger than JCI. This double oversigned in mathematics refers to 'much larger than' (UO).

So, this is greater, and that means much larger. And this rightmost part of the graph is where JCV is much greater than JTRAIN. And generally, JTRAIN will be quite low. But the key indicator is whether

JCV is much greater than JTRAIN. And that's what happened when we fitted a very high-degree polynomial to this small set of small data.

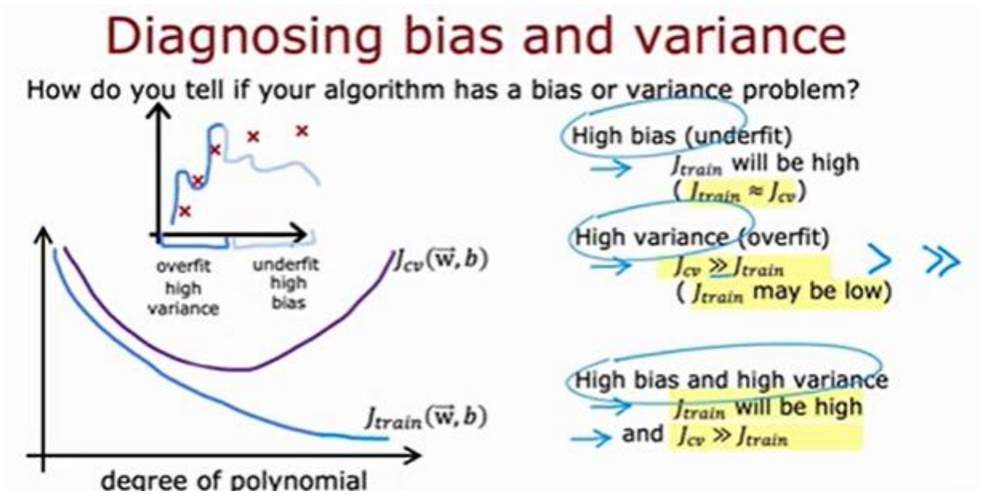
And even though we just looked at bias and variance, it turns out that in some cases it's possible to have both high bias and high variance simultaneously. You won't see this happen often with linear regression, but it turns out that if you training a neural network, there are some applications where you unfortunately have both high bias and high variance.



One way to recognize this situation is if JTRAIN is high, meaning you're not performing very well on the training set, but even worse, the cross-validation error is much larger than the training set. The notion of high bias and high variance doesn't really occur for linear models applied to 1D. But to give you an idea of what it looks like, it would be as if, for part of the input, you had a very complicated model that overfits.

So, it overfits that portion of the input. But then, for some reason, for other parts of the input it doesn't even fit the training data well, and it's therefore underfitted. For part of the input, in this example, which seems artificial because it's a single-function input, we fit the training set very well and we overfit part of the input, and we don't even fit the training data well and we underfit part of the input.

And that's how, in some applications, you can unfortunately end up with high bias and high variance. And the indicator for this will be whether the algorithm performs poorly on the training set, or even much worse than on the training set. For most machine learning applications, you're probably mainly dealing with a high bias or high variance, rather than both at the same time.



But sometimes it's possible to have both at the same time. So I know there's a lot to cover. There are a lot of concepts on the slides. But the key takeaways are that high bias means it doesn't even perform well on the training set, and high variance means it performs much worse on the cross-validation set than on the training set.

Whenever I train a machine learning algorithm, I almost always try to determine how much the algorithm has high bias or underfitting versus high variance or an overfitting problem. And that would give good clues, as we'll see later this week, about how you can improve your algorithm's performance. But first, let's look at how regularization affects the bias and variance of a learning algorithm , because that will help you better understand when you need to use regularization.