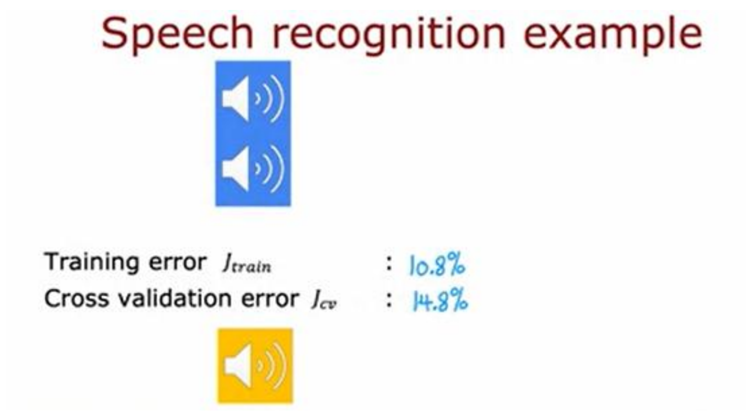
Establish a benchmark performance level

Let's look at some concrete numbers for what J-train and JCV might be, and see how you can judge if a learning algorithm has high bias or high variance. For the examples in this video, I'm going to use as a running example the application of speech recognition which is something I've worked on multiple times over the years.

Let's take a look. A lot of users doing web search on a mobile phone will use speech recognition rather than type on the tiny keyboards on our phones because speaking to a phone is often faster than typing. Typical audio that's a web search engine we get would be like this, "What is today's weather?" Or like this, "Coffee shops near me." It's the job of the speech recognition algorithms to output the transcripts whether it's today's weather or coffee shops near me.



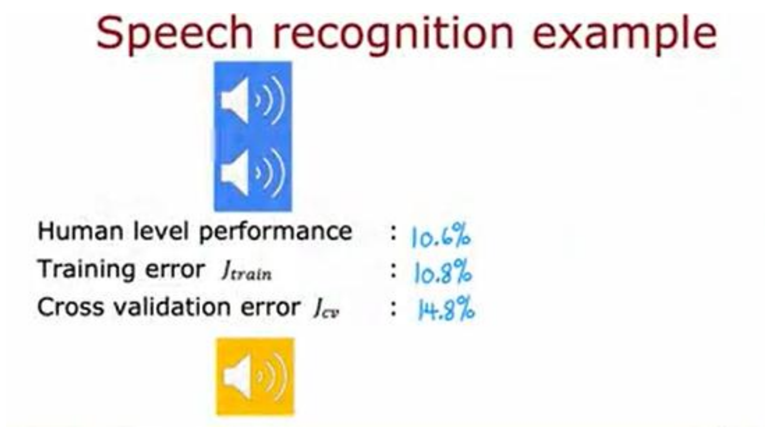
Speech recognition example

Training error J_{train} : 10.8%
Cross validation error J_{cv} : 14.8%

Now, if you were to train a speech recognition system and measure the training error, and the training error means what's the percentage of audio clips in your training set that the algorithm does not transcribe correctly in its entirety. Let's say the training error for this data-set is 10.8 percent meaning that it transcribes it perfectly for 89.2 percent of your training set, but makes some mistake in 10.8 percent of your training set.

If you were to also measure your speech recognition algorithm's performance on a separate cross-validation set, let's say it gets 14.8 percent error. If you were to look at these numbers it looks like the training error is really high, it got 10 percent wrong, and then the cross-validation error is higher but getting 10 percent of even your training set wrong that seems pretty high.

It seems like that 10 percent error would lead you to conclude it has high bias because it's not doing well on your training set, but it turns out that when analyzing speech recognition it's useful to also measure one other thing which is what is the human level of performance?



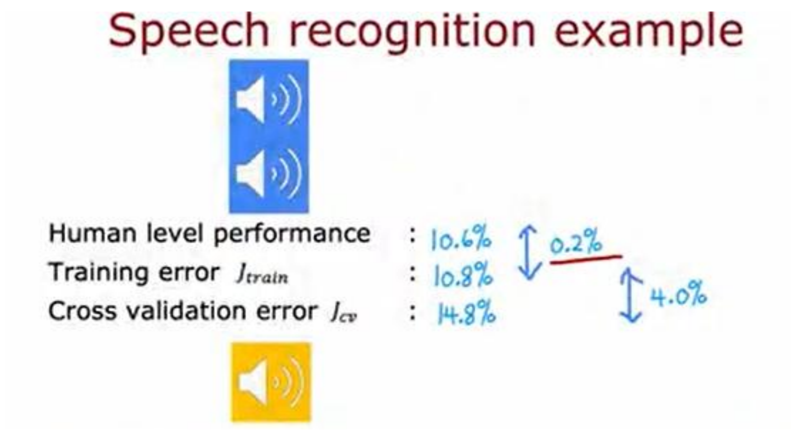
Speech recognition example

Human level performance : 10.6%
Training error J_{train} : 10.8%
Cross validation error J_{cv} : 14.8%

In other words, how well can even humans transcribe speech accurately from these audio clips? Concretely, let's say that you measure how well fluent speakers can transcribe audio clips and you find that human level performance achieves 10.6 percent error.

Why is human level error so high? It turns out that for web search, there are a lot of audio clips that sound like this, "I'm going to navigate to [inaudible]." There's a lot of noisy audio where really no one can accurately transcribe what was said because of the noise in the audio. If even a human makes 10.6 percent error, then it seems difficult to expect a learning algorithm to do much better.

In order to judge if the training error is high, it turns out to be more useful to see if the training error is much higher than a human level of performance, and in this example it does just 0.2 percent worse than humans. Given that humans are actually really good at recognizing speech I think if I can build a speech recognition system that achieves 10.6 percent error matching human performance I'd be pretty happy, so it's just doing a little bit worse than humans. But in contrast, the gap or the difference between JCV and J-train is much larger.

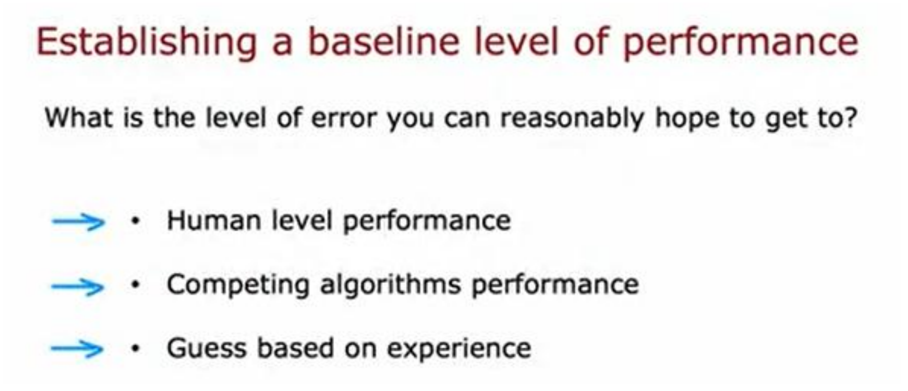

Speech recognition example
Human level performance : 10.6%
Training error J_{train} : 10.8%
Cross validation error J_{cv} : 14.8%
0.2%
4.0%

There's actually a four percent gap there, whereas previously we had said maybe 10.8 percent error means this is high bias. When we benchmark it to human level performance, we see that the algorithm is actually doing quite well on the training set, but the bigger problem is the cross-validation error is much higher than the training error which is why I would conclude that this algorithm actually has more of a variance problem than a bias problem.

It turns out when judging if the training error is high is often useful to establish a baseline level of performance, and by baseline level of performance I mean what is the level of error you can reasonably hope your learning algorithm to eventually get to. One common way to establish a baseline level of performance is to measure how well humans can do on this task because humans are really good at understanding speech data, or processing images or understanding texts.

Human level performance is often a good benchmark when you are using unstructured data, such as: audio, images, or texts. Another way to estimate a baseline level of performance is if there's some competing algorithm, maybe a previous implementation that someone else has implemented or even a competitor's algorithm to establish a baseline level of performance if you can measure that, or sometimes you might guess based on prior experience.

If you have access to this baseline level of performance that is, what is the level of error you can reasonably hope to get to or what is the desired level of performance that you want your algorithm to get to?



Establishing a baseline level of performance

What is the level of error you can reasonably hope to get to?

- Human level performance
- Competing algorithms performance
- Guess based on experience

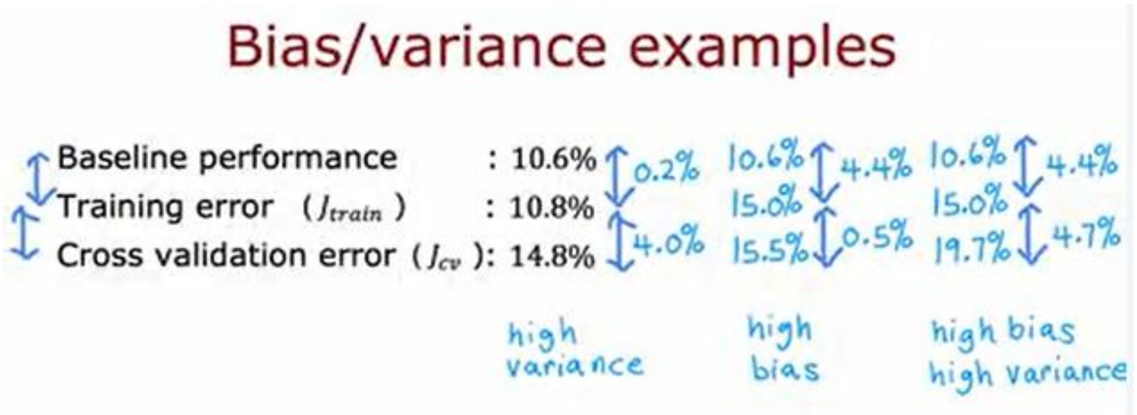
Then when judging if an algorithm has high bias or variance, you would look at the baseline level of performance, and the training error, and the cross-validation error. The two key quantities to measure are then: what is the difference between training error and the baseline level that you hope to get to.

This is 0.2, and if this is large then you would say you have a high bias problem. You will then also look at this gap between your training error and your cross-validation error, and if this is high then you will conclude you have a high variance problem. That's why in this example we concluded we have a high variance problem, whereas let's look at the second example.

If the baseline level of performance; that is human level performance, and training error, and cross validation error look like this, then this first gap is 4.4 percent and so there's actually a big gap. The training error is much higher than what humans can do and what we hope to get to whereas the cross-validation error is just a little bit bigger than the training error.

If your training error and cross validation error look like this, I will say this algorithm has high bias. By looking at these numbers, training error and cross validation error, you can get a sense intuitively or informally of the degree to which your algorithm has a high bias or high variance problem. Just to summarize, this gap between these first two numbers gives you a sense of whether you have a high bias problem, and the gap between these two numbers gives you a sense of whether you have a high variance problem.

Sometimes the baseline level of performance could be zero percent. If your goal is to achieve perfect performance than the baseline level of performance it could be zero percent, but for some applications like the speech recognition application where some audio is just noisy then the baseline level of a performance could be much higher than zero. The method described on this slide will give you a better read in terms of whether your algorithm suffers from bias or variance. By the way, it is possible for your algorithms to have high bias and high variance.



Bias/variance examples

Baseline performance	: 10.6%	10.6%	10.6%
	0.2%	4.4%	4.4%
Training error (J_{train})	: 10.8%	15.0%	15.0%
	4.0%	0.5%	4.7%
Cross validation error (J_{cv})	: 14.8%	15.5%	19.7%
	high variance	high bias	high bias high variance

Concretely, if you get numbers like these, then the gap between the baseline and the training error is large. That would be a 4.4 percent, and the gap between training error and cross validation error is also wide. This is 4.7 percent. If it looks like this you will conclude that your algorithm has high bias and high variance, although hopefully this won't happen that often for your learning applications. T

o summarize, we've seen that looking at whether your training error is large is a way to tell if your algorithm has high bias, but on applications where the data is sometimes just noisy and is infeasible or unrealistic to ever expect to get a zero error then it's useful to establish this baseline level of performance. Rather than just asking is my training error a lot, you can ask is my training error large relative to what I hope I can get to eventually, such as, is my training large relative to what humans can do on the task?

That gives you a more accurate read on how far away you are in terms of your training error from where you hope to get to.

Then similarly, looking at whether your cross-validation error is much larger than your training error, gives you a sense of whether or not your algorithm may have a high variance problem as well. In practice, this is how I often will look at these numbers to judge if my learning algorithm has a high bias or high variance problem. Now, to further hone our intuition about how a learning algorithm is doing, there's one other thing that I found useful to think about which is the learning curve.