Bias and Variance with mismatched data distributions

(DESCRIPTION)
Text, Mismatched training and dev, test, data. Bias and Variance with mismatched data distributions.

(SPEECH)
Estimating the bias and variance of your learning algorithm really helps you prioritize what to work on next.

But the way you analyze bias and variance changes when your training set comes from a different distribution than your dev and test sets.

Let's see how.

Let's keep using our cat classification example and let's say humans get near perfect performance on this.

So, Bayes error, or Bayes optimal error, we know is nearly 0% on this problem.

So, to carry out error analysis you usually look at the training error and also look at the error on the dev set.

So let's say, in this example that your training error is 1%, and your dev error is 10%.

If your dev data came from the same distribution as your training set, you would say that here you have a large variance problem, that your algorithm's just not generalizing well from the training set which it's doing well on to the dev set, which it's suddenly doing much worse on.

But in the setting where your training data and your dev data comes from a different distribution, you can no longer safely draw this conclusion.

In particular, maybe it's doing just fine on the dev set, it's just that the training set was really easy because it was high res, very clear images, and maybe the dev set is just much harder.

So maybe there isn't a variance problem and this just reflects that the dev set contains images that are much more difficult to classify accurately.

So the problem with this analysis is that when you went from the training error to the dev error, two things changed at a time.

One is that the algorithm saw data in the training set but not in the dev set.

Two, the distribution of data in the dev set is different.

And because you changed two things at the same time, it's difficult to know of this 9% increase in error, how much of it is because the algorithm didn't see the data in the dev set, so that's some of the variance part of the problem.

And how much of it, is because the dev set data is just different.

So, in order to tease out these two effects, and if you didn't totally follow what these two different effects are, don't worry, we will go over it again in a second.

But in order to tease out these two effects it will be useful to define a new piece of data which we'll call the training-dev set.

So, this is a new subset of data, which we carve out that should have the same distribution as training sets, but you don't explicitly train in your network on this.

So here's what I mean.

(DESCRIPTION)
A line is drawn down the middle of the slide. A long rectangle is drawn at the top, and split into three columns. The first column is marked train, and makes up the majority of the rectangle. At the end are two smaller columns, marked dev, and test.

(SPEECH)
Previously we had set up some training sets and some dev sets and some test sets as follows.

And the dev and test sets have the same distribution, but the training sets will have some different distribution.

What we're going to do is randomly shuffle the training sets and then carve out just a piece of the training set to be the training-dev set.

(DESCRIPTION)
A third, small, column is made in the rectangled, and marked train dash dev.

(SPEECH)
So just as the dev and test set have the same distribution, the training set and the training-dev set, also have the same distribution.

But, the difference is that now you train your neural network, just on the training set proper.

You won't let the neural network, you won't run that obligation on the training-dev portion of this data.

To carry out error analysis, what you should do is now look at the error of your classifier on the training set, on the training-dev set, as well as on the dev set.

So let's say in this example that your training error is 1%.

And let's say the error on the training-dev set is 9%, and the error on the dev set is 10%, same as before.

What you can conclude from this is that when you went from training data to training dev data the error really went up a lot.

And only the difference between the training data and the training-dev data is that your neural network got to sort the first part of this.

It was trained explicitly on this, but it wasn't trained explicitly on the training-dev data.

So this tells you that you have a variance problem.

Because the training-dev error was measured on data that comes from the same distribution as your training set.

So you know that even though your neural network does well in a training set, it's just not generalizing well to data in the training-dev set which comes from the same distribution, but it's just not generalizing well to data from the same distribution that it hadn't seen before.

So in this example we have really a variance problem.

Let's look at a different example.

Let's say the training error is 1%, and the training-dev error is 1.5%, but when you go to the dev set your error is 10%.

So now, you have actually a pretty low variance problem, because when you went from training data that you've seen to the training-dev data that the neural network has not seen, the error increases only a little bit, but then it really jumps when you go to the dev set.

So this is a data mismatch problem, where data mismatched.

So this is a data mismatch problem, because your learning algorithm was not trained explicitly on data from training-dev or dev, but these two data sets come from different distributions.

But whatever algorithm it's learning, it works great on training-dev but it doesn't work well on dev.

So somehow your algorithm has learned to do well on a different distribution than what you really care about, so we call that a data mismatch problem.

(DESCRIPTION)
A line is drawn under the explained, hand-written numbers.

(SPEECH)
Let's just look at a few more examples.

I'll write this on the next row since I'm running out of space on top.

So Training error, Training-Dev error, and Dev error.

Let's say that training error is 10%, training-dev error is 11%, and dev error is 12%.

Remember that human level proxy for Bayes error is roughly 0%.

So if you have this type of performance, then you really have a bias, an avoidable bias problem, because you're doing much worse than human level.

So this is really a high bias setting.

And one last example.

If your training error is 10%, your training-dev error is 11% and your dev error is 20 %, then it looks like this actually has two issues.

One, the avoidable bias is quite high, because you're not even doing that well on the training set.

Humans get nearly 0% error, but you're getting 10% error on your training set.

The variance here seems quite small, but this data mismatch is quite large.

So for for this example I will say, you have a large bias or avoidable bias problem as well as a data mismatch problem.

So let's take what we've done on this slide and write out the general principles.

(DESCRIPTION)
New slide, Bias and variance on mismatched training and dev, test, sets.

(SPEECH)
The key quantities I would look at are human level error, your training set error, your training-dev set error.

So that's the same distribution as the training set, but you didn't train explicitly on it.

Your dev set error, and depending on the differences between these errors, you can get a sense of how big is the avoidable bias, the variance, the data mismatch problems.

So let's say that human level error is 4%.

Your training error is 7%.

And your training-dev error is 10%.

And the dev error is 12%.

So this gives you a sense of the avoidable bias.

because you know, you'd like your algorithm to do at least as well or approach human level performance maybe on the training set.

This is a sense of the variance.

So how well do you generalize from the training set to the training-dev set?

This is the sense of how much of a data mismatch problem have you have.

And technically you could also add one more thing, which is the test set performance, and we'll write test error.

You shouldn't be doing development on your test set because you don't want to overfit your test set.

But if you also look at this, then this gap here tells you the degree of overfitting to the dev set.

So if there's a huge gap between your dev set performance and your test set performance, it means you maybe overtuned to the dev set.

And so maybe you need to find a bigger dev set, right?

So remember that your dev set and your test set come from the same distribution.

So the only way for there to be a huge gap here, for it to do much better on the dev set than the test set, is if you somehow managed to overfit the dev set.

And if that's the case, what you might consider doing is going back and just getting more dev set data.

Now, I've written these numbers, as you go down the list of numbers, always keep going up.

Here's one example of numbers that doesn't always go up, maybe human level performance is 4%, training error is 7%, training-dev error is 10%, but let's say that we go to the dev set.

You find that you actually, surprisingly, do much better on the dev set.

Maybe this is 6%, 6% as well.

So you have seen effects like this, working on for example a speech recognition task, where the training data turned out to be much harder than your dev set and test set.

So these two were evaluated on your training set distribution and these two were evaluated on your dev/test set distribution.

So sometimes if your dev/test set distribution is much easier for whatever application you're working on then these numbers can actually go down.

So if you see funny things like this, there's an even more general formulation of this analysis that might be helpful.

Let me quickly explain that on the next slide.

(DESCRIPTION)
New slide, More general formulation.

(SPEECH)
So, let me motivate this using the speech activated rear-view mirror example.

It turns out that the numbers we've been writing down can be placed into a table where on the horizontal axis, I'm going to place different data sets.

So for example, you might have data from your general speech recognition task.

(DESCRIPTION)
A cross is drawn. The right-hand column is marked General Speech Recognition.

(SPEECH)
So you might have a bunch of data that you just collected from a lot of speech recognition problems you worked on from small speakers, data you have purchased and so on.

And then you all have the rear view mirror specific speech data, recorded inside the car.

(DESCRIPTION)
Further to the right is written Rearview Mirror, Speech Data.

(SPEECH)
So on this x axis on the table, I'm going to vary the data set.

On this other axis, I'm going to label different ways or algorithms for examining the data.

So first, there's human level performance, which is how accurate are humans on each of these data sets?

Then there is the error on the examples that your neural network has trained on.

And then finally there's error on the examples that your neural network has not trained on.

So turns out that what we're calling on a human level on the previous slide, there's the number that goes in

(DESCRIPTION)
Human level is written in the first row and column cross-section.

(SPEECH)
this box, which is how well do humans do on this category of data.

Say data from all sorts of speech recognition tasks, the thousand utterances that you could into your training set.

And the example in the previous slide is this 4%.

This number here was our, maybe the training error.

(DESCRIPTION)
Training error is written in the second row and, first column.

(SPEECH)
Which in the example in the previous slide was 7% Right, if you're learning algorithm has seen this example, performed gradient descent on this example, and this example came from your training set distribution, or some general speech recognition distribution.

How well does your algorithm do on the example it has trained on?

Then here is the training-dev set error.

It's

(DESCRIPTION)
Training, dev set error is written in the third row, first column.

(SPEECH)
usually a bit higher, which is for data from this distribution, from general speech recognition, if your algorithm did not train explicitly on some examples from this distribution, how well does it do?

And that's what we call the training dev error.

And then if you move over to the right, this box here is the dev set error, or maybe also the test set error.

(DESCRIPTION)
Dev, test error is written in the third row, second column.

(SPEECH)
Which was 6% in the example just now.

And dev and test error, it's actually technically two numbers, but either one could go into this box here.

And this is if you have data from your rearview mirror, from actually recorded in the car from the rearview mirror application, but your neural network did not perform back propagation on this example, what is the error?

So what we're doing in the analysis in the previous slide was look at differences between these two numbers, these two numbers, and these two numbers.

(DESCRIPTION)
Lines are drawn from human level, 4 percent, to training error, 7 percent. From training error, 7 percent, to training, dev, error, 10 percent. And from training, dev, error, 10 percent, to dev, test error, 6 percent.

(SPEECH)
And this gap here is a measure of avoidable bias.

This gap here is a measure of variance, and this gap here was a measure of data mismatch.

(DESCRIPTION)
Two vertical parameters are drawn between along the first row, the second row, and horizontally along the second column.

(SPEECH)
And it turns out that it could be useful to also throw in the remaining two entries in this table.

And so if this turns out to be also 6%, and the way you get this number is you ask some humans to label their rearview mirror speech data and just measure how good humans are at this task.

And maybe this turns out also to be 6%.

And the way you do that is you take some rearview mirror speech data, put it in the training set so the neural network learns on it as well, and then you measure the error on that subset of the data.

But if this is what you get, then, well, it turns out that you're actually already performing at the level of humans on this rearview mirror speech data, so maybe you're actually doing quite well on that distribution of data.

When you do this more subsequent analysis, it doesn't always give you one clear path forward, but sometimes it just gives you additional insights as well.

So for example, comparing these two numbers in this case tells us that for humans, the rearview mirror speech data is actually harder than for general speech recognition, because humans get 6% error, rather than 4% error.

But then looking at these differences as well may help you understand bias and variance and data mismatch problems in different degrees.

So this more general formulation is something I've used a few times.

I've not used it, but for a lot of problems you find that examining this subset of entries, kind of looking at this difference and this difference and this difference, that that's enough to point you in a pretty promising direction.

But sometimes filling out this whole table can give you additional insights.

Finally, we've previously talked a lot about ideas for addressing bias.

Talked about techniques on addressing variance, but how do you address data mismatch?

In particular training on data that comes from different distribution that your dev and test set can get you more data and really help your learning algorithm's performance.

But rather than just bias and variance problems, you now have this new potential problem of data mismatch.

What are some good ways that you could use to address data mismatch?

I'll be honest and say there actually aren't great or at least not very systematic ways to address data mismatch.

But there are some things you could try that could help.

Let's take a look at them in the next video.

So what we've seen is that by using training data that can come from a different distribution as a dev and test set, this could give you a lot more data and therefore help the performance of your learning algorithm.

But instead of just having bias and variance as two potential problems, you now have this third potential problem, data mismatch.

So what if you perform error analysis and conclude that data mismatch is a huge source of error, how do you go about addressing that?

It turns out that unfortunately there are super systematic ways to address data mismatch, but there are a few things you can try that could help.

Let's take a look at them in the next video.