Cleaning up incorrectly labeled data

(SPEECH)
The

(DESCRIPTION)
Text, Error Analysis. Cleaning up incorrectly labeled data.

(SPEECH)
data for your supervised learning problem comprises input X and output labels Y.

What if you going through your data and you find that some of these output labels Y are incorrect, you have data which is incorrectly labeled?

Is it worth your while to go in to fix up some of these labels? Let's take a look.

(DESCRIPTION)
New slide, Incorrectly labeled examples. Two rows are present, one marked X, the other marked Y. The X row includes seven photos horizontally. In order, they are a cat, dog, cat, cat, dog, dog, and cat. The second row, Y, is marked 1, 0, 1, 1, 0, 1, and 1.

(SPEECH)
In the cat classification problem, Y equals one for cats and zero for non cats.

So, let's say you're looking through some data and that's a cat, that's not a cat, that's a cat, that's a cat, that's not a cat, that's at a cat.

No, wait. That's actually not a cat.

So this is an example with an incorrect

(DESCRIPTION)
A square is drawn around the dog in the sixth position, with a corresponding 1 underneath it.

(SPEECH)
label.

So I've used the term, mislabeled examples, to refer to if your learning algorithm outputs the wrong value of Y.

But I'm going to say, incorrectly labeled examples, to refer to if in the data set you have in the training set or the dev set or the test set, the label for Y, whatever a human label assigned to this piece of data, is actually incorrect.

And that's actually a dog so that Y really should have been zero.

But maybe the labeler got that one wrong.

So if you find that your data has some incorrectly labeled examples, what should you do?

Well, first, let's consider the training set.

It turns out that deep learning algorithms are quite robust to random errors in the training set.

So long as your errors or your incorrectly labeled examples, so long as those errors are not too far from random, maybe sometimes the labeler just wasn't paying attention or they accidentally, randomly hit the wrong key on the keyboard.

If the errors are reasonably random, then it's probably okay to just leave the errors as they are and not spend too much time fixing them.

There's certainly no harm to going into your training set and be examining the labels and fixing them.

Sometimes that is worth doing but your effort might be okay even if you don't.

So long as the total data set size is big enough and the actual percentage of errors is maybe not too high.

So I see a lot of machine learning algorithms that trained even when we know that there are few X mistakes in the training set labels and usually works okay.

There is one caveat to this which is that deep learning algorithms are robust to random errors.

They are less robust to systematic errors.

So for example, if your labeler consistently labels white dogs as cats, then that is a problem because your classifier will learn to classify all white colored dogs as cats.

But random errors or near random errors are usually not too bad for most deep learning algorithms.

Now, this discussion has focused on what to do about incorrectly labeled examples in your training set.

How about incorrectly labeled examples in your dev set or test set?

If

(DESCRIPTION)
New slide, Error analysis. A table is presented with six columns and six rows. The top left square is shared by both rows and columns, and is marked Image. Beneath it, the rows are an elipses, 98, 99, 100, and percent of total. The columns to the right of Image are, Dog, Great Cat, Blurry, Incorrectly labeled, and Comments. Checkmarks exist in the cross-sections of 98 and incorrectly labeled, 99 and great cat, and 100 and incorrectly labeled. Comments are present in the row 98, labeler missed cat in background. And 100, drawing of a cat, not a real cat. Percentages populate each column. Dogs, 8 percent. Great Cat, 43 percent. Blurry, 61 percent. And Incorrectly labeled, 6 percent.

(SPEECH)
you're worried about the impact of incorrectly labeled examples on your dev set or test set, what they recommend you do is during error analysis to add one extra column so that you can also count up the number of examples where the label Y was incorrect.

So for example, maybe when you count up the impact on a 100 mislabeled dev set examples, so you're going to find a 100 examples where your classifier's output disagrees with the label in your dev set.

And sometimes for a few of those examples, your classifier disagrees with the label because the label was wrong, rather than because your classifier was wrong.

So maybe in this example, you find that the labeler missed a cat in the background.

So put the check mark there to signify that example 98 had an incorrect label.

And maybe for this one, the picture is actually a picture of a drawing of a cat rather than a real cat.

Maybe you want the labeler to have labeled that Y equals zero rather than Y equals one.

And so put another check mark there.

And just as you count up the percent of errors due to other categories like we saw in the previous video, you'd also count up the fraction of percentage of errors due to incorrect labels.

Where the Y value in your dev set was wrong and that accounted for why your learning algorithm made a prediction that differed from what the label on your data says.

So the question now is, is it worthwhile going in to try to fix up this 6% of incorrectly labeled examples.

My advice is, if it makes a significant difference to your ability to evaluate algorithms on your dev set, then go ahead and spend the time to fix incorrect labels.

But if it doesn't make a significant difference to your ability to use the dev set to evaluate cost buyers, then it might not be the best use of your time.

Let me show you an example that illustrates what I mean by this.

So, three numbers I recommend you look at to try to decide if it's worth going in and reducing the number of mislabeled examples are the following.

I recommend you look at the overall dev set error.

And so in the example we had from the previous video, we said that maybe our system has 90% overall accuracy.

So 10% error.

Then you should look at the number of errors or the percentage of errors that are due to incorrect labels.

So it looks like in this case, 6% of the errors are due to incorrect labels.

So 6% of 10% is 0.6%.

And then you should look at errors due to all other causes.

So if you made 10% error on your dev set and 0.6% of those are because the labels is wrong, then the remainder, 9.4% of them, are due to other causes such as misrecognizing dogs being cats, great cats and their images.

So in this case, I would say there's 9.4% worth of error that you could focus on fixing, whereas the errors due to incorrect labels is a relatively small fraction of the overall set of errors.

So by all means, go in and fix these incorrect labels if you want but it's maybe not the most important thing to do right now.

Now, let's take another example.

Suppose you've made a lot more progress on your learning problem.

So instead of 10% error, let's say you brought the errors down to 2%, but still 0.6% of your overall errors are due to incorrect labels.

So now, if you want to examine a set of mislabeled dev set images, set that comes from just 2% of dev set data you're mislabeling, then a very large fraction of them, 0.6 divided by 2%, so that is actually 30% rather than 6% of your labels.

Your incorrect examples are actually due to incorrectly label examples.

And so errors due to other causes are now 1.4%.

When such a high fraction of your mistakes as measured on your dev set due to incorrect labels, then it maybe seems much more worthwhile to fix up the incorrect labels in your dev set.

And if you remember the goal of the dev set, the main purpose of the dev set is, you want to really use it to help you select between two classifiers A and B.

So you're trying out two classifiers A and B, and one has 2.1% error and the other has 1.9% error on your dev set.

But you don't trust your dev set anymore to be correctly telling you whether this classifier is actually better than this because your 0.6% of these mistakes are due to incorrect labels.

Then there's a good reason to go in and fix the incorrect labels in your dev set.

Because in this example on the right is just having a very large impact on the overall assessment of the errors of the algorithm, whereas example on the left, the percentage impact is having on your algorithm is still smaller.

Now, if you decide to go into a dev set and manually re-examine the labels and try to fix up some of the labels, here are a few additional guidelines or principles to consider.

First,

(DESCRIPTION)
New slide, Correcting incorrect dev, test set examples.

(SPEECH)
I would encourage you to apply whatever process you apply to both your dev and test sets at the same time.

We've talk previously about why you want to dev and test sets to come from the same distribution.

The dev set is tagging you into target and when you hit it, you want that to generalize to the test set.

So your team really works more efficiently to dev and test sets come from the same distribution.

So if you're going in to fix something on the dev set, I would apply the same process to the test set to make sure that they continue to come from the same distribution.

So we hire someone to examine the labels more carefully.

Do that for both your dev and test sets.

Second, I would urge you to consider examining examples your algorithm got right as well as ones it got wrong.

It is easy to look at the examples your algorithm got wrong and just see if any of those need to be fixed.

But it's possible that there are some examples that you haven't got right, that should also be fixed.

And if you only fix ones that your algorithms got wrong, you end up with more bias estimates of the error of your algorithm.

It gives your algorithm a little bit of an unfair advantage.

We just try to double check what it got wrong but you don't also double check what it got right because it might have gotten something right, that it was just lucky on fixing the label would cause it to go from being right to

being wrong, on that example.

The second bullet isn't always easy to do, so it's not always done.

The reason it's not always done is because if you classifier's very accurate, then it's getting fewer things wrong than right.

So if your classifier has 98% accuracy, then it's getting 2% of things wrong and 98% of things right.

So it's much easier to examine and validate the labels on 2% of the data and it takes much longer to validate labels on 98% of the data, so this isn't always done.

That's just something to consider.

Finally, if you go into a dev and test data to correct some of the labels there, you may or may not decide to go and apply the same process for the training set.

Remember we said that at this other video that it's actually less important to correct the labels in your training set.

And it's quite possible you decide to just correct the labels in your dev and test set which are also often smaller than a training set and you might not invest all that extra effort needed to correct the labels in a much larger training set.

This is actually okay.

We'll talk later this week about some processes for handling when your training data is different in distribution than you dev and test data.

Learning algorithms are quite robust to that.

It's super important that your dev and test sets come from the same distribution.

But if your training set comes from a slightly different distribution, often that's a pretty reasonable thing to do.

I will talk more about how to handle this later this week.

So I'd like to wrap up with just a couple of pieces of advice.

First, deep learning researchers sometimes like to say things like, "I just fed the data to the algorithm.

I trained in and it worked." There is a lot of truth to that in the deep learning error.

There is more of feeding data in algorithm and just training it and doing less hand engineering and using less human insight.

But I think that in building practical systems, often there's also more manual error analysis and more human insight that goes into the systems than sometimes deep learning researchers like to acknowledge.

Second is that somehow I've seen some engineers and researchers be reluctant to manually look at the examples.

Maybe it's not the most interesting thing to do, to sit down and look at a 100 or a couple hundred examples to counter the number of errors.

But this is something that I so do myself.

When I'm leading a machine learning team and I want to understand what mistakes it is making, I would actually go in and look at the data myself and try to counter the fraction of errors.

And I think that because these minutes or maybe a small number of hours of counting data can really help you prioritize where to go next.

I find this a very good use of your time and I urge you to consider doing it if those machines are in your system and you're trying to decide what ideas or what directions to prioritize things.

So that's it for the error analysis process.

In the next video, I want to share a view of some thoughts on how error analysis fits in to how you might go about starting out on the new machine learning project.