Multi-task learning

(DESCRIPTION)
Text, Learning from multiple tasks. Multi-task learning.

(SPEECH)
So whereas in transfer learning, you have a sequential process where you learn from task A and then transfer that to task B.

In multi-task learning, you start off simultaneously, trying to have one neural network do several things at the same time.

And then each of these task helps hopefully all of the other task.

Let's look at an example.

(DESCRIPTION)
New slide, Simplified autonomous driving example. A photo appears of a black sedan at a stop sign intersection.

(SPEECH)
Let's say you're building an autonomous vehicle, building a self driving car.

Then your self driving car would need to detect several different things such as pedestrians, detect other cars, detect stop signs.

And also detect traffic lights and also other things.

So for example, in this example on the left, there is a stop sign in this image and there is a car in this image but there aren't any pedestrians or traffic lights.

So if this image is an input for an example, x(i), then Instead of having one label y(i), you would actually a four labels.

In this example, there are no pedestrians, there is a car, there is a stop sign and there are no traffic lights.

And if you try and detect other things, there may be y(i) has even more dimensions.

But for now let's stick with these four.

(DESCRIPTION)
The vertical digits zero, one, one, and zero are bracketed for emphasis.

(SPEECH)
So y(i) is a 4 by 1 vector.

And if you look at the training test labels as a whole, then similar to before, we'll stack the training data's labels horizontally as follows, y(1) up to y(m).

Except that now y(i) is a 4 by 1 vector so each of these is a tall column vector.

And so this matrix Y is now a 4 by m matrix, whereas previously, when y was single real number, this would have been a 1 by m matrix.

So what you can do is now train a neural network to predict these values of y.

(DESCRIPTION)
New slide, Neural network architecture. Arrows move on a horizontal plane, with an X variable at the left end, and a Y variable at the right end. In the middle are seven rectangles, representing traffic lights, with circles in their interior. In order, the number of circles in rectangles are three, five, five, three, three, three, and four.

(SPEECH)
So you can have a neural network input x and output now a four dimensional value for y.

Notice here for the output there I've drawn four nodes.

And so the first node when we try to predict is there a pedestrian in this picture.

The second output will predict is there a car here, predict is there a stop sign and this will predict maybe is there a traffic light.

So y hat here is four dimensional.

So to train this neural network, you now need to define the loss for the neural network.

And so given a predicted output y hat i which is 4 by 1 dimensional.

The loss averaged over your entire training set would be 1 over m sum from i = 1 through m, sum from j = 1 through 4 of the losses of the individual predictions.

So it's just summing over at the four components of pedestrian car stop sign traffic lights.

And this script L is the usual logistic loss.

So just to write this out, this is -yj i log y hat ji- 1- y log 1- y hat.

And the main difference compared to the earlier binding classification examples is that you're now summing over j equals 1 through 4.

And the main difference between this and softmax regression, is that unlike softmax regression, which assigned a single label to single example.

This one image can have multiple labels.

So you're not saying that each image is either a picture of a pedestrian, or a picture of car, a picture of a stop sign, picture of a traffic light.

You're asking for each picture, does it have a pedestrian, or a car a stop sign or traffic light, and multiple objects could appear in the same image.

In fact, in the example on the previous slide, we had both a car and a stop sign in that image, but no pedestrians and traffic lights.

So you're not assigning a single label to an image, you're going through the different classes and asking for each of the classes does that class, does that type of object appear in the image?

So that's why I'm saying that with this setting, one image can have multiple labels.

If you train a neural network to minimize this cost function, you are carrying out multi-task learning.

Because what you're doing is building a single neural network that is looking at each image and basically solving four problems.

It's trying to tell you does each image have each of these four objects in it.

And one other thing you could have done is just train four separate neural networks, instead of train one network to do four things.

But if some of the earlier features in neural network can be shared between these different types of objects, then you find that training one neural network to do four things results in better performance than training four completely separate neural networks to do the four tasks separately.

So that's the power of multi-task learning.

And one other detail, so far I've described this algorithm as if every image had every single label.

It turns out that multi-task learning also works even if some of the images we'll label only some of the objects.

So the first training example, let's say someone, your labeler had told you there's a pedestrian, there's no car, but they didn't bother to label whether or not there's a stop sign or whether or not there's a traffic light.

And maybe for the second example, there is a pedestrian, there is a car, but again the labeler, when they looked at that image, they just didn't label it, whether it had a stop sign or whether it had a traffic light, and so on.

And maybe some examples are fully labeled, and maybe some examples, they were just labeling for the presence and absence of cars so there's some question marks, and so on.

So with a data set like this, you can still train your learning algorithm to do four tasks at the same time, even when some images have only a subset of the labels and others are sort of question marks or don't cares.

And the way you train your algorithm, even when some of these labels are question marks or really unlabeled is that in this sum over j from 1 to 4, you would sum only over values of j with a 0 or 1 label.

So whenever there's a question mark, you just omit that term from summation but just sum over only the values where there is a label.

And so that allows you to use datasets like this as well.

So when does multi-task learning makes sense?

(DESCRIPTION)
New slide, When multi-task learning makes sense.

(SPEECH)
So when does multi-task learning make sense?

I'll say it makes sense usually when three things are true.

One is if your training on a set of tasks that could benefit from having shared low-level features.

So for the autonomous driving example, it makes sense that recognizing traffic lights and cars and pedestrians, those should have similar features that could also help you recognize stop signs, because these are all features of roads.

Second, this is less of a hard and fast rule, so this isn't always true.

But what I see from a lot of successful multi-task learning settings is that the amount of data you have for each task is quite similar.

So if you recall from transfer learning, you learn from some task A and transfer it to some task B.

So if you have a million examples of task A then and 1,000 examples for task B, then all the knowledge you learned from that million examples could really help augment the much smaller data set you have for task B.

Well how about multi-task learning?

In multi-task learning you usually have a lot more tasks than just two.

So maybe you have, previously we had 4 tasks but let's say you have 100 tasks.

And you're going to do multi-task learning to try to recognize 100 different types of objects at the same time.

So what you may find is that you may have 1,000 examples per task and so if you focus on the performance of just one task, let's focus on the performance on the 100th task, you can call A100.

If you are trying to do this final task in isolation, you would have had just a thousand examples to train this one task, this one of the 100 tasks that by training on these 99 other tasks.

These in aggregate have 99,000 training examples which could be a big boost, could give a lot of knowledge to argument this otherwise, relatively small 1,000 example training set that you have for task A100.

And symmetrically every one of the other 99 tasks can provide some data or provide some knowledge that help every one of the other tasks in this list of 100 tasks.

So the second bullet isn't a hard and fast rule but what I tend to look at is if you focus on any one task, for that to get a big boost for multi-task learning, the other tasks in aggregate need to have quite a lot more data than for that one task.

And so one way to satisfy that is if a lot of tasks like we have in this example on the right, and if the amount of data you have in each task is quite similar.

But the key really is that if you already have 1,000 examples for 1 task, then for all of the other tasks you better have a lot more than 1,000 examples if those other other task are meant to help you do better on this final task.

And finally multi-task learning tends to make more sense when you can train a big enough neural network to do well on all the tasks.

So the alternative to multi-task learning would be to train a separate neural network for each task.

So rather than training one neural network for pedestrian, car, stop sign, and traffic light detection, you could have trained one neural network for pedestrian detection, one neural network for car detection, one neural network for stop sign detection, and one neural network for traffic light detection.

So what a researcher, Rich Carona, found many years ago was that the only times multi-task learning hurts performance compared to training separate neural networks is if your neural network isn't big enough.

But if you can train a big enough neural network, then multi-task learning certainly should not or should very rarely hurt performance.

And hopefully it will actually help performance compared to if you were training neural networks to do these different tasks in isolation.

So that's it for multi-task learning.

In practice, multi-task learning is used much less often than transfer learning.

I see a lot of applications of transfer learning where you have a problem you want to solve with a small amount of data.

So you find a related problem with a lot of data to learn something and transfer that to this new problem.

But multi-task learning is just more rare that you have a huge set of tasks you want to use that you want to do well on, you can train all of those tasks at the same time.

Maybe the one example is computer vision.

In object detection I see more applications of multi-task any where one neural network trying to detect a whole bunch of objects at the same time works better than different neural networks trained separately to detect objects.

But I would say that on average transfer learning is used much more today than multi-task learning, but both are useful tools to have in your arsenal.

So to summarize, multi-task learning enables you to train one neural network to do many tasks and this can give you better performance than if you were to do the tasks in isolation.

Now one note of caution, in practice I see that transfer learning is used much more often than multi-task learning.

So I do see a lot of tasks where if you want to solve a machine learning problem but you have a relatively small data set, then transfer learning can really help.

Where if you find a related problem but you have a much bigger data set, you can train in your neural network from there and then transfer it to the problem where we have very low data.

So transfer learning is used a lot today.

There are some applications of transfer multi-task learning as well, but multi-task learning I think is used much less often than transfer learning.

And maybe the one exception is computer vision object detection, where I do see a lot of applications of training a neural network to detect lots of different objects.

And that works better than training separate neural networks and detecting the visual objects.

But on average I think that even though transfer learning and multi-task learning often you're presented in a similar way, in practice I've seen a lot more applications of transfer learning than of multi-task learning.

I think because often it's just difficult to set up or to find so many different tasks that you would actually want to train a single neural network for.

Again, with some sort of computer vision, object detection examples being the most notable exception.

So that's it for multi-task learning.

Multi-task learning and transfer learning are both important tools to have in your tool bag.

And finally, I'd like to move on to discuss end-to-end deep learning.

So let's go onto the next video to discuss end-to-end learning.