Introduction to TensorFlow

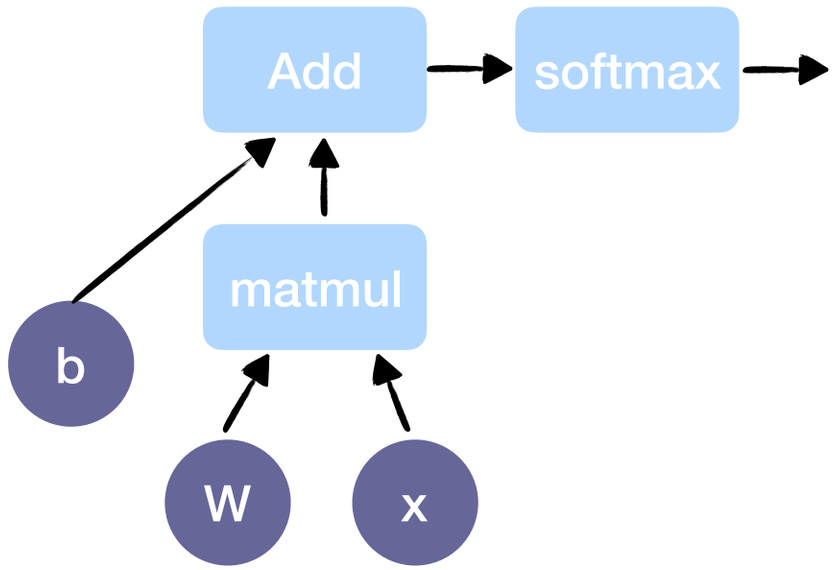
 Romeo Kienzler

In the next module we will cover TensorFlow, one of the hottest DeepLearning frameworks out now.

- **Open Source**
- **(static) data flow graphs**
- **Express algorithms**
- **Execute at scale or embedded**

- Open Source library for numerical computation
- ..using data flow graphs
- It allows us to express machine learning and deep learning algorithms
- ..and brings along an execution engine which allows these algorithms run at scale on multiple nodes in a cluster, backed by CPUs, GPUs, TPUs or on mobile devices

Express numerical computation as graph



- so every numerical computation is a graph where the nodes are computations and on the edges are flowing tensors between them - therefore the name tensorflow

a3_m1_s2_v1_tfintro

 Romeo Kienzler

- so let’s start with a little coding example within ibm data science experience. We create a new notebook. We give it a name and select the correct spark service although we do not yet need it and click on create. Make sure that python is selected as programming language.

So first of all we download some data. TensorFlow has a couple of built-in data sets and we use the mnist handwritten digit classification set. Now the data gets downloaded and we obtain a complex python object called mint

let’s import tensorflow first but then examine the data we’ve just downloaded
we use the jupyter magic command matplotlib inline to display the plots directly in the notebook

then we access the first image out of 55 thousand from the mnist training set using a built-in iterator which is quite useful for later usage but now we just access one image at a time

images in the mnist data set are 28 by 28 pixels, but we are obtaining a only a vector of 784 pixels length .. therefore we have to reshape accordingly

then we tell pyplot to plot greyscale and finally plot the image

so this is a three and if we run again we see a one…note that every time we call next_batch we obtain new image

sometimes we are not really sure what number is meant, therefore we can also print the label

Cross-Entropy

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

Now we define the cost function as cross entropy, therefore let me just walk you through the formula and we will see later how to implement it in tensorflow.

Cross-Entropy

$$-\sum_i y'_i \log(y_i)$$

 Romeo Kienzler

Now we define the cost function as cross entropy, therefore let me just walk you through the formula and we will see later how to implement it in tensorflow. so we take the predicted value of y a

Cross-Entropy

$$-\sum_i y'_i \log(y_i)$$

and multiply it to the log of the desired value of y

Cross-Entropy

$$-\sum_i y'_i \log(y_i)$$

and sum those values up.

a3_m1_s2_v1_tfintro

 Romeo Kienzler

So we start with the reduce mean function of tensor flow because we are now calculating ten individual cross entropy values, one for each softmax regression model.

Then we use reduce sum to calculate the sum over individual values of a tensor.

and this tensor is the product of the desired value and the log of the actual prediction

reduction indicies defines the dimensions of the tensor where the aggregation should take place. Since y is a matrix of 10 columns and n rows - where n stands for the number of training examples - we sum over the columns to obtain a value for each digit

this result is now passed as an argument to reduce mean so that the overall prediction error is calculated out of the individual prediction errors for each number between zero and nine

Now we use TensorFlows GradientDescentOptimizer with a learning rate of zero dot 5 to tweak double u and b with respect to the cross entropy function. So TensorFlow will take care of calculating the back propagation and gradients for this tasks automatically. A feature called “automatic differentiation” does the job for us.

Now we create a tensor flow session - since we are in a interactive context within a jupyter notebook we use the interactive session. A session is the way to deploy the TensorFlow execution graph into a specific execution context like a CPU or GPU.

then we initialise all global variables since this hasn't been done. remember we just have only expressed the computational graph, now it’s time to bring it to live

Summary

So as you have seen, training of neural networks can get quite complicated. But it’s ok if you didn’t understand all the details. Just note that gradient descent doesn’t make any guarantees that it converges to the global optimum unless the cost function is convex, which is not the case in neural networks. So it might get stuck in local optimas and even guarantees to converge at all. Therefore a lot of hyper parameter tuning might be necessary.

Introduction to TensorBoard

 Romeo Kienzler

In the next module we will cover TensorBoard - a visual way to debug your neural networks