Deep Learning Lab Report

Edward Kreutzarek

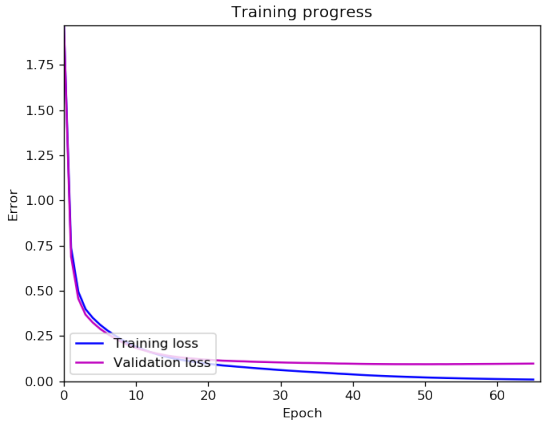
October 2018

Overview

General Architecture

- 1. Input Layer (28*28)
- 2. Fully Connected Layer (250 Units, Sigmoid)
- 3. Fully Connected Layer (100 Units, ReLu)
- 4. Fully Connected Layer (10 Units, no activation function)
- 5. Softmax Output Layer

Epochs: 65, loss: 0.0091, train error: 0.0015, validation error: 0.0231
test error: 0.0231



Implementation

Difficulties

At first, i found it difficult to navigate through the code already written in the exercise. Variable names like “output grad” or “grad input” where confusing in contrast to the syntax of a “normal” pen and paper backpropagation. It was quite difficult to implement the backpropagation.

Overall, i found it challenging to transfer the general structure of a neural network (input layer, output layer, fully connected layer, etc.), the role of the different types of layers and how these layers interact with each other to the code.

Negative log likelihood really confused me and i'm still not sure what the right implementation is, especially since one has to take multiclass classification into account in our case. Searching on ie. stackoverflow gave a lot of different results.

Playing around and doing trial and error with the different hyperparameters (number of epochs, learning rate, number of hidden layers and their activation functions, etc.) was fun at first but got kind of frustrating, since i do not have a lot of experience/ knowledge to base them on. I did some research on the basic dos and don'ts of neural network architecture, but i guess it requires a lot of experience to get a feeling for these things.

Overfitting

I tried to avoid overfitting, as seen by looking at the small training error, by implementing L2 regularization and/or Dropout, but unfortunately, i could not get either of them working. In theory, i know how both of them work, however, when it comes to writing the code on the right places, it gets quite tricky. I'd like to implement them eventually, as i've spent a lot of time working on this neural network and i'd like to really understand to code behind it.

Learning rate

I noticed that there really was no point in leaving the learning rate constant through all epochs. It would be much better to start with a high learning rate and then eventually lower it, significantly reducing the number of epochs needed. I tried this by hard coding a switching learning rate. This was a great lesson for understand the intention behind things like momentum and optimizers. I tried to implement momentum and RMSprop, but ran out of time to properly implement them. However, i plan on eventually finishing it.

Uneccesary things i spent too much time on:

1. Live plotting the training and validation error
2. Visualizing single elements of the MNIST data