Deep Learning Lab, Report for Submission 1

Implementation Status

In general my submissions contains all the requested features. It additionally contains the following bonus features:

- Random dropout of weights and biases
- L2-Regularization (I did not add the penalty term to the actual loss function but only used the regularization when computing the gradients. This is possible because the penalty term of a given parameter only affects the gradient of this parameter).
- RMSprop Optimizer. This needs an additional parameter for each weight (the average squared history) and thus increases the memory usage.

Given the task of the exercise sheet my submission currently has the following limitations:

- There currently are no stopping criteria other than "train for the given number of epochs". I found out during my experiments that this feature could be useful to avoid unnecessary epochs that are expensive in bigger nets.
- The SGD optimizer currently only supports a constant learning rate and no more advanced learning rate schedule techniques. I mostly used my RMSprop optimizer for the experiments which to my understanding worked well.

I hope that my implemented bonus features will make up for those missing features.

Experiments and Results

After playing around a bit initially I limited myself to the following tasks:

- Find a network that is powerful enough to represent the connection between the training data and the output
- See how much this network overfits and apply regularization techniques to keep the training and the validation loss connected.

For most of the experiments I used the **RMSprop** optimizer with a forgetting parameter of 0.9 and a learning rate of 0.001. These are according to literature the default parameters for this optimizer and I found them to lead to fast convergence also in my application.
I found that a net with 2 hidden layers of 800 neurons each was powerful enough to learn the training set almost completely. The training and validation loss of this experiment can be seen in figure 1 In this figure we can observe the following:

- This model clearly overfits, as the training loss goes to zero while the validation loss increases at first.

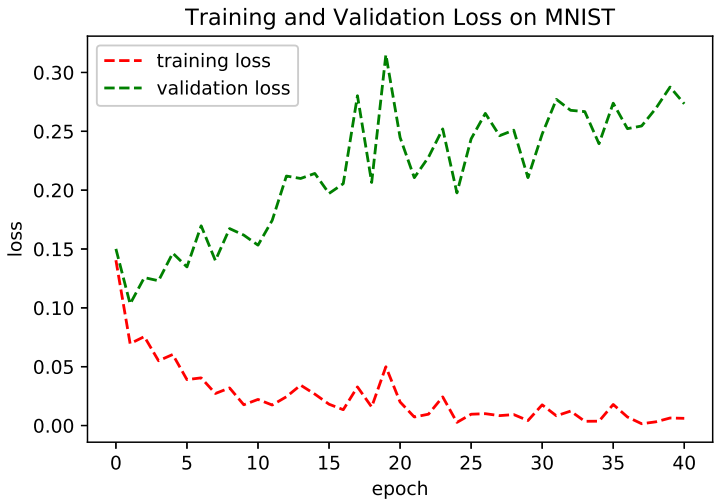

Figure 1: 2 hidden layers with 800 neurons each, ReLU activation, no regularization, 40 epochs

- The overfitting only starts at the third epoch. This means that stopping the training earlier can reduce overfitting.

As a next step I applied a dropout of 50% and got the results from figure 2. We see that the absolute loss values stay worse than in the unregularized case and that we still have overfitting.
I also made the following (strange) observation: My training routine also plots the validation error and not only the loss and I found out that those two are not really closely coupled. Although the validation loss slightly increases in figure 1, the error on the validation set keeps going down. I am not sure if this is a typical behavior of the softmax cross entropy loss or if there might be something wrong in my code.
For my final observations I merged the training and validation set and compared against the test data. The losses of this run can be seen in figure 3. What can be observed in this figure is that the two loss functions are still connected (peaks in the training loss are also peaks in the validation loss) which could be a sign for the overfitting not being too bad.
The final result on the test set yielded an error rate of **1.73%** which I consider to be pretty decent for my first tries with deep learning in practice.

Feedback

I am attending the "Foundations of Deep Learning" lecture in parallel where we also have to implement a feed forward network this week. The stubs that are provided differ in several aspects. While the one provided for this lab is simpler (less arguments, less abstract base classes) it is harder to implement extra features. While I found it pretty

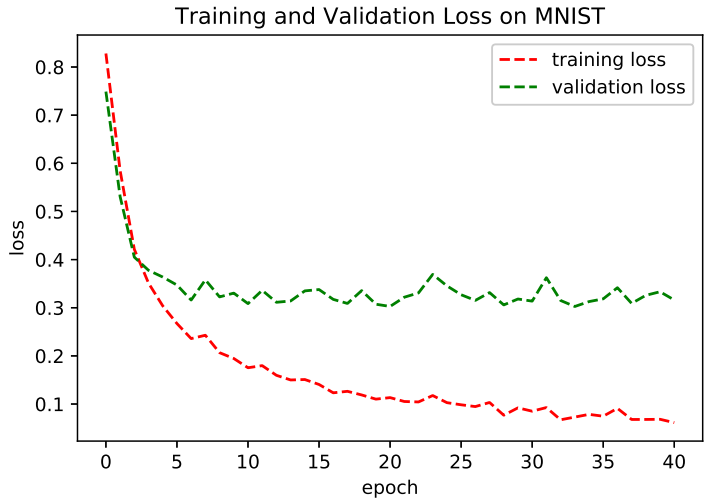

Figure 2: 2 hidden layers with 800 neurons each, ReLU activation, dropout factor 0.5, 40 epochs

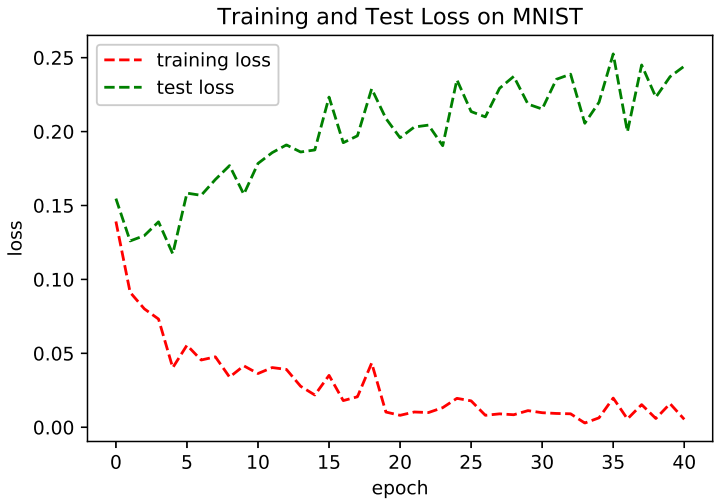

Figure 3: 2 hidden layers with 800 neurons each, ReLU activation, no regularization, 40 epochs

straightforward to code the basic features I sometimes found it hard to include extra features like regularization or more optimizers in a architecturally clean way. For these aspects the stub from the Deep Learning lecture is probably a bit better. There they try to mimic the API of an existing framework (pytorch in their case) which will be used later in the course. I am not sure how long we will work with our own implementation in this lab and if or when we will start using existing framework but maybe this point could be taken into consideration for upcoming versions of this course.

But overall I really liked this sheet and was really pleased when the gradient checks worked for the first time and when I was able to get error rates below 2%.