Deep Learning Lab Course
Second Exercise

Aaron Klein

Due: November 13, 2018

The goal of this exercise is to 1) get familiar with Tensorflow by implementing a small convolutional neural network (CNN) for MNIST and 2) learn how to set up and conduct experiments. At the end, send us a small report (1 or 2 pages) with the below described plots and the conclusions that you can draw from them.

1 Implementing a CNN in Tensorflow

First, we will implement a scaled-down version of LeNet which we have discussed during the lecture. Before you start, checkout the Tensorflow tutorial https://www.tensorflow.org/tutorials/.

Our CNN consists of two convolutional layers (16 3×3 filters and a stride of 1), each followed by ReLU activation function and a max pooling layer (pool size 2). After the convolution layers we add a fully connected layer with 128 units and a softmax layer for the classification. We train the network by optimizing the cross-entropy loss with stochastic gradient descent.

Make sure that you save the validation performance after each epoch, i.e the learning curve, since we are going to need them for the experiments in the next section.

2 Learning Rate

After implementing our neural network and making sure that it works correctly, we will have a look on the effect of the learning rate on the network's performance. Try the following values for the learning rate: $\{0.1, 0.01, 0.001, 0.0001\}$, save the results (validation performance after each epoch) and plot all learning curves in the same figure.

Which conclusions could be drawn from this figure? Which value for the learning rate works best? Do you have an idea what is happening if the learning rate is too small / high?

3 Convolution Type

In the lecture we learned that one can use different filter size for the convolutions. Try out the following filter sizes $\{1, 3, 5, 7\}$ (the same for height and width) and report the results. Make sure that for all filter sizes you use the padding strategy *'SAME'*.

Again plot the learning curves and analyze the results. Do you have an idea in which scenarios smaller filters might be better than larger filters and vice versa?

4 Random Search

As you can see, setting the hyperparameters correctly makes a huge difference. Unfortunately, the most hyperparameter are not easily transferable across datasets, for instance the optimal learning rate depends on the dataset. Besides the learning rate and the filter sizes, there are more hyperparameters that can be tuned. In the last part of this exercise we will apply random search to automatically tune the following hyperparameters:

- learning rate $\in [10^{-4}, 10^{-1}]$
- batch size $\in [16, 128]$
- number of filters $\in [2^3, 2^6]$
- filter size $\in \{3, 5\}$

Make sure that you optimize the first three on a logarithmic scale and that the filter size is a categorical hyperparameter. Run random search for 50 iterations where each function evaluation has a budget of 6 epochs. We will use the HpBandster (https://github.com/automl/HpBandSter) package for random search. You can install via *'pip install hpbandster'* .

In order to evaluate the progress of random search, plot the validation performance of the best found configuration after each iteration. Finally, evaluate the test performance of the best configuration that you found.