Prof. Dr. Carsten Meyer
Faculty of Engineering
Christian-Albrechts-Universität Kiel

Neural Networks and Deep Learning – Summer Term 2019

Exercise sheet 5

Submission due: Tuesday, June 18, 13:15 sharp

Exercise 1 (Overfitting):

The script **`exercise1.py`** (adapted from Michael Nielsen and Michal Daniel Dobrzanski) configures a simple feedforward neural network with a single hidden layer, the parameters of which are chosen to demonstrate the effect of overfitting on the MNIST digit recognition task. Run the script and discuss the results. Does the overfitting effect depend on the learning rate and on the mini-batch size?
Explore two possibilities to reduce the effect of overfitting (you may refer to the previous exercise for options and code...).

Exercise 2 (Loss functions and weight update formulae, theoretical considerations):

a) Show that for a single-layer perceptron with a linear activation function, the weights can be determined in closed-form from the training data, assuming a mean-squared error loss.

b) The cross-entropy loss for a single-layer perceptron on a training set $D = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(p)}, y^{(p)})\}$ is defined as

$$\mathrm{L}_{CE}(\boldsymbol{w}, y, \hat{y}) = -\frac{1}{p}\sum_{\mu=1}^{p}\left[y^{(\mu)} \ln \hat{y}(\boldsymbol{w}, \boldsymbol{x}^{(\mu)}) + \left(1 - y^{(\mu)}\right) \ln(1 - \hat{y}\left(\boldsymbol{w}, \boldsymbol{x}^{(\mu)}\right))\right],$$

where $\hat{y} = \hat{y}\left(\boldsymbol{w}, \boldsymbol{x}^{(\mu)}\right)$ is the output of the perceptron. Show that this expression is non-negative for $y, \hat{y} \in [0,1]$ and assumes a minimum (as a function of the perceptron output $\hat{y}$) if $\hat{y}\left(\boldsymbol{w}, \boldsymbol{x}^{(\mu)}\right) = y^{(\mu)}$ for all μ, i.e. if the neuron output $\hat{y}\left(\boldsymbol{w}, \boldsymbol{x}^{(\mu)}\right)$ corresponds to the target value $y^{(\mu)}$ for each training sample μ.

Exercise 3 (Convolutional neural networks):

a) Explain the following terms related to convolutional neural networks (CNN):

- Convolutional neural network
- Filter, Kernel
- Feature map
- Receptive field
- Pooling (subsampling) layer
- Fully convolutional neural network

b) (CNN with Keras – optional) The file **`exercise3b.py`** (adapted from the Keras examples, https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py) applies a CNN to the MNIST classification problem.

Some hints on installing Keras and Tensorflow (not a complete installation guide!) can be found in **`README_installation_cuda_and_tensorflow.txt`**.

Note that plotting a visualization of the model requires pydotprint to work; to this end, pydot and graphviz have to be installed from an anaconda command prompt:

```
conda install pydot
conda install graphviz
```

If the installation is not successful, comment out the following lines:

```
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
...
SVG(model_to_dot(model).create(prog='dot', format='svg'))
```

Alternatively, you may visualize the model using the line

```
plot_model(best_model, to_file='best_model.png',
show_shapes=True)
```

Further documentation about Keras can be found at https://keras.io/
In particular:

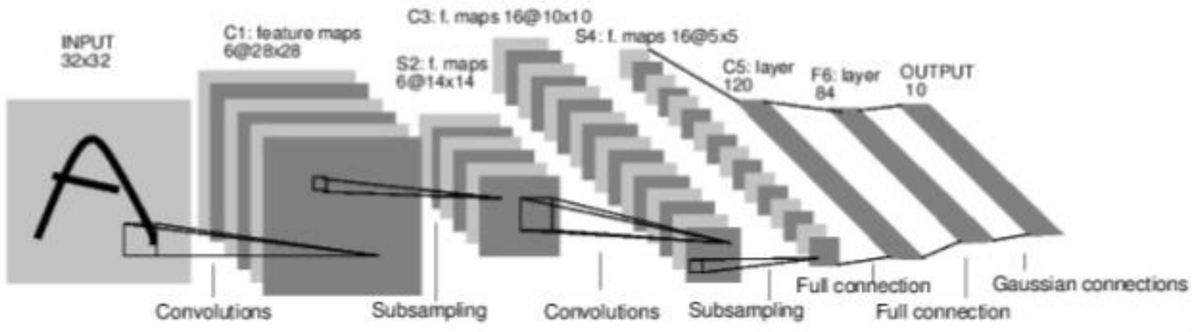
- Documentation about the Keras Sequential model:
  https://keras.io/getting-started/sequential-model-guide/
- Documentation about Keras convolutional layers:
  https://keras.io/layers/convolutional/
- Documentation about Keras pooling layers:
  https://keras.io/layers/pooling/
- Documentation about Keras core layers (dense, dropout, activation etc.):
  https://keras.io/layers/core/

Run the script **`exercise3b.py`**, visualize the model structure, discuss the values of the most important default parameters and report the accuracy of the model.
Try to improve the accuracy (and / or the number of model parameters) by modifying the model structure and / or the values of the default parameters.

Exercise 4 (Number of parameters in a fully connected and a convolutional network):

a) For a fully connected multi-layer perceptron containing two hidden layers with 100 hidden units each which is designed for the MNIST classification problem, calculate the number of learnable parameters (i.e. parameters which are learned using the backpropagation algorithm).

b) The figure shows the architecture of the famous LeNet-5 convolutional network. Calculate the number of trainable parameters and the number of connections (synaptic weights plus biases, i.e. in augmented space).
Cx: Convolutional layer x, Sx: Subsampling layer x, Fx: Fully connected layer x



d) How do these numbers change in the ZF net?