Problem 11.1 (Convolutional Neural Networks) (50 P.)

The goal of this week's exercise is to extend the generic multilayer neural network that we implemented last week. We will implement a new type of layer that can be regarded as a combination of a convolutional layer and a subsampling layer. Afterwards, we will train a convolutional neural network (CNN) on the MNIST dataset of handwritten digits.

Following the paper *Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis* from Patrice Y. Simard, Dave Steinkraus and John C. Platt, you will use a CNN with 3 hidden layers to classify handwritten digits. The images from the MNIST datasets are grayscale images with (1 x) 28 x 28 pixels. The first hidden layer will be a convolutional layer with 6 feature maps, the corresponding filters will have the size 5 x 5 and the stride will be 2 in each direction. In the second hidden layer, we will have 50 feature maps and the corresponding filters will again have the size 5 x 5 with a stride of 2 in each direction. The last hidden layer will be fully connected to the previous layer and contain 100 nodes.

a) For classification we need the softmax activation function $g(a_f) = \frac{\exp(a_f)}{\sum_{f'} \exp(a_{f'})}$ in the output layer and cross entropy error function $E = -\sum_n \sum_f ln(y_f^n)t_f^n$. Note that $\frac{\partial E_n}{\partial a_f} = \delta_f = y_f^n - t_f^n$ (this is the same formula like the one for sum of squared errors and identity activation function). This approach requires that labels are represented by 1-of-c encoding which means that we have c outputs for c classes and each output indicates the probability that an example belongs to a certain class. The outputs in the training data are binary, of course. Implement both and check your implementation with `test_mlnn_classification_gradient()` from `test_gradient.py`. (15 P.)

b) Implement a convolutional layer with a stride variable. The stride will indicate the displacement between consecutive filter positions during the convolution and is usually 1. You will only have to implement the forward propagation in `tools.py` (function `convolve()`). The backpropagation has been implemented for you. To make the forward propagation efficient, you must implement it either with SciPy weave and blitz or Numba (or Cython or Parakeet) because the implementation involves many loops that are inefficient in pure Python. Our solution is written with SciPy weave. Check your implementation with `test_cnn_gradient()` from `test_gradient.py`. (15 P.)

c) For the CNN architecture that has been described above, compute how many pixels there will be in the feature maps of the convolutional layers, how many connections there will be between each pair of consecutive layers and how many weights there will be between each pair of consecutive layers. How many connections would a neural network with the same number of hidden nodes and fully connected hidden layers have? (10 P.)

d) Train a CNN with this architecture on the training set of the MNIST dataset. Use the parameters `epochs=15`, `batch_size=32`, `alpha=0.01`, `alpha_decay=0.9999`, `min_alpha=0.00005`, `eta=0.5`, `eta_inc=0.00001`, `max_eta=0.9` for `MBSGD`. Report the accuracy on the test set. Plot those numbers of the test set that have not been classified correctly. (10 P.)

Note: We will give you our solution to exercise 10 as soon as the deadline is over.

On the hand-in date, **08.02.2017**, you must hand-in the following: [1]

a) a text file stating how much time you (all together) used to complete this exercise sheet
b) your solutions / answers / code

for problem 11.1

[1]upload via StudIP (if there are problems with the upload contact me **beforehand**: krell@uni-bremen.de)