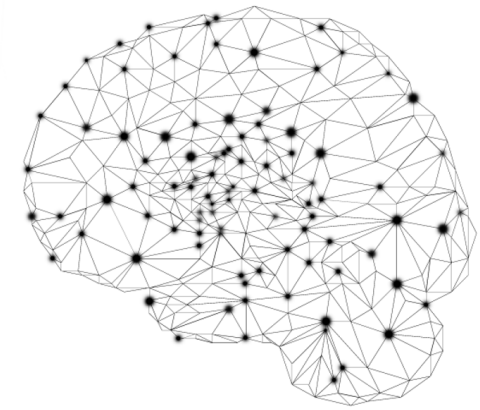

Neural Networks and Data Science

Lab #8

29.11.2023
Deadline: 06.12.2023, 12:10

Dr. Marcel Völschow

HAW HAMBURG

Problem 1

Feeding in the entire training dataset into the network is called an *training epoch*. As every image only slightly changes the neural network weights, networks are typically trained for more than one epoch.

a) Create a `cnn` module and design a network `testNet` with 784 input nodes, 100 hidden nodes, 10 output nodes and a learning rate of 0.6.

b) Download the `mnist.zip` file, extract its contents and use `np.load` to load the `*.npy` files of the training and test images and labels.

c) Write a method `batchTrain(self, dataTrain, labelsTrain)` that takes a training data array and a labels array as arguments and trains the network with every image of the training data array.

d) Train your network for at least 10 epochs, but feel free to go even further. After every epoch, evaluate the fraction of test images that the networks classifies correctly and store it in a list or array `score`. Use `time.time()` from the `time` library to measure the total execution time of your multi-epoch training and test. Open the Task Manager and inspect the utilization of your system (CPU, RAM).

e) Generate a plot to illustrate the evolution of the network's performance as a function of the number of training epochs. Discuss your result.

Problem 2

The number of hidden nodes has a big impact on the performance of the network. However, too many nodes just increase the computational effort and slow down the convergence of the training process. What's the sweet spot?

a) Create a list of integers `nodes` that contains values between 40 and 480 with a step size of 20. Use that list and list comprehension to create a list of neural networks `networks` where the number of hidden nodes are given by the list `nodes`. Keep the other parameters constant.

b) Use the `batchTrain` method to train all networks. For every network, evaluate the fraction of test images that it classifies correctly and store the result in a list or array `score`.

c) Generate a plot to illustrate the classification performance as a function of the number of hidden nodes. Discuss your result.

Problem 3

The learning rate controls the speed of the gradient descent which in turn affects the speed and quality of the network's convergence during the training process.

a) Create a list or array `lRates` with 30 floats between 0.1 and 0.9. Use list comprehension and these learning rates to create a list of neural networks `networks`.

b) Use the `batchTrain` method to train all networks. For every network, evaluate the fraction of test images that it classifies correctly and store the result in a list or array `score`.

c) Generate a plot to illustrate the classification performance as a function of the learning rate. Discuss your result.

Problem 4

In the first three problems, you have explored the parameter space for the three free parameters of our custom neural network. Time for an actual application!

a) Create a neural network `readNet`. Use the set of parameters that yields the best performance.

b) Train the network and use the `saveWeights` method to save the weights. What's the performance of your network?

c) Read in the handwritten digit(s) you have created for problem sheet 6 as a numpy array. Normalize the array entries such that they fit the network's requirements. Query your network, apply the softmax function to the output vector and discuss the result.

Problem 5

This task is intended for those of you who would like to go an extra mile ...

Another classic dataset is MNIST-Fashion by *Zalando Research* (yes, that IS actually a thing):

https://github.com/zalandoresearch/fashion-mnist

It contains images of 60,000 fashion pieces that belong to one of ten classes.

a) In the cloud you will find a ZIP file `fashion.zip`. Download it, unpack it and load the `*.npy` files into a notebook.

b) Create a list `names` with the names of the ten classes (see the link). Plot one (random) representative of each class using subplots.

c) Set up a network `fashionNet`. Start with the parameters we used for the MNIST dataset and evaluate the performance. Feel free to optimize it ...