Project 1

NUMN21/FMNN25: Advanced Numerical Algorithms in Python

Robert Klöfkorn and Andreas Langer

This describes the first bigger programming project in the course, devoted to artificial neural networks.

Application: Recognizing handwritten numbers

Most people effortlessly recognise these digits as 504192. This ease is deceptive. The difficulty of visual pattern recognition becomes obvious when trying to write a computer program to recognise digits like the above. What seems easy when we do it ourselves suddenly becomes extremely difficult. Simple notions about how we recognise shapes - "a 9 has a loop at the top and a vertical line at the bottom right" - turn out to be not so easy to express algorithmically. If you try to specify such rules, you quickly get lost in a quagmire of exceptions, restrictions and special cases. It seems hopeless.

Neural networks approach the problem differently. The idea is to use a large number of handwritten digits, called training examples, and then develop a system that can learn from these training examples. In other words, the neural network uses the examples to automatically derive rules for recognising handwritten digits. In addition, by increasing the number of training examples, the network can learn more about handwriting and thus improve its accuracy.

Task 1

Implement a feedforward neural network (as a class) consisting of 3 layers (input, hidden, output layer), where each layer can contain any number of neurons. Use the sigmoid function as the activation function.

Task 2

Reading in MNIST data (provided in canvas). The data is separated into training data (50 000), validation data (10 000), and test data (10 000).

Task 3

Implement the stochastic gradient method (SGD) to train the network. The implementation of the SGD should allow for different mini-batch sizes and different numbers of epochs. An epoch is the complete pass of the training data through the learning algorithm.

Task 4

Implement the backpropagation algorithm (used in SGD to effectively calculate the derivative).

Task 5

Train and test the accuracy of the network for the following parameters:

- Input layer with 784 + 1 neurons
- hidden layer with 30 + 1 neurons
- Output layer with 10 neurons

As loss function use the quadratic function (square loss)

$$\frac{1}{2n}\sum_{x}\|h_w(x)-y\|_2^2,$$

where (x, y) is a pair of training data, n the amount of used training data, and h_w represents the neural network.

Task 6

Print an output of the learning success per epoch.

Extra Task 7 (in case you want an extra challenge)

Implement an attack on the trained neural network.

Lycka till!