Feedforward neural networks

Final project for Computational Physics II, WS 2018 *January 30, 2019*

Students:	Aron Castro Anna-Belle Garten Marius Pille Max Pfeifer
Deadline for written report and code:	27.02.2019
Mid-project meeting	13.02.2019 (to be confirmed)
Project goals	• Implement feedforward neural networks with one and possibly two layers of hidden neurons. • Implement and test the backpropagation algorithm for the calculation of the gradient of the cost function with respect to the connection weights. • Implement and test the gradient descent algorithm for supervised training. • Train the one-layer neural network with a subset of MNIST and test the ability of the network to recognize handwritten digits.
Main resources	Intro, chapters 1 and 2 of the online book *Neural Networks and Deep Learning* `http://neuralnetworksanddeeplearning.com`
Other resources	Syllabus of the *CS229: Machine Learning* course at Stanford `http://cs229.stanford.edu/syllabus.html`

Assignment

1. Study introduction, chapters 1 and 2 of the online book *Neural Networks and Deep Learning*, which can be found at `http://neuralnetworksanddeeplearning.com`. My advice is to study this as you progress with the other tasks. Some of the terminology in the next points will become clear only after you have learned something about neural networks.

2. Consider a feedforward neural network with an arbitrary number of layers and neurons in each layer. Given the connection weights, the neural network takes some input data (one bit per input neuron) and produces some output data (one real number per output neuron). Write the function that calculates the output of the neural network, given the connection weights and the input data.

   For a generic feedforward neural network you will need to think of what is the best way to store the weights into an appropriate data structure. A discussion of the data structure should be included in the final report.

   In order to calculate the output of the neural network, you will need to choose an activation function. Write the code in such a way that it is easy to change the activation function.

   The function that calculates the output of the neural network should have a recursive structure over the number of layers. Discuss the structure of this function in the final report.

3. In order to train the neural network, one needs to choose a cost function. Training the neural network means to minimize the cost function as a function of the weights, over the training data set. In order to do so, one needs to choose a minimization algorithm. Let us work with the quadratic cost function, and the gradient descent algorithm.

   In order to implement the gradient descent, you will need to calculate the gradient of the cost function with respect to the weights. Backpropagation is a method to write this gradient for a feedforward neural network with an arbitrary number of layers.

   Use the backpropagation method to write the gradient of the feedforward neural network with only one hidden layer, and (if you have time) with two hidden layers. Write the program that calculates these gradients.

4. Given a function $f(x)$ of n variables $x_1, x_2, \ldots, x_n$, its gradient is given by the vector of partial derivatives $g_k(x) = \frac{\partial f}{\partial x_k}$ with $k = 1, \ldots, n$. If you have a code that calculates $f(x)$ and $g(x)$, a way to check that $g(x)$ has been implemented correctly is to show numerically that, given a random vector v_k with $k = 1, \ldots, n$,

$$\frac{f(x + \epsilon v) - f(x)}{\epsilon} - \sum_k v_k g_k(x) = O(\epsilon) \ . \tag{1}$$

   In practice this can be done by calculating the left hand side of the above equation for a sequence of decreasing ϵ and showing appriximate linearity with ϵ. Use this trick to test that the implementation of the gradient of the cost function is correct.

5. Implement the gradient descent algorithm to minimize the cost function. Choose a sensible stopping criterium for the gradient descent algorithm.

6. Fix the size of the neural network (keep it relatively small), and produce a random training data set, and a random set of initial weights. Run the gradient descent algorithm and show numerically that the cost function decreases along the iterations of the gradient descent. Keep the training data set fixed, and generate new initial weights. Is the output of the gradient descent independent of the initial weights?

7. Train your network with one hidden layer on a subset of MNIST, discuss the parameters (training speed, stopping criterium, size of training set...) on which the training depends. Measure the performance of the trained network on a test set. Discuss critically your findings.

Guidelined for code and report

- Code and report can be sent by email.
- The code has to be working, and well organized. Avoid operations that are not necessary, split logically-different parts in modules/functions, avoid code repetition, make the code readable. In the report, include a short description of all functions in the code (i.e. description of what they do, their input and output).
- Spend time thinking about data structure, i.e. how all the parameters of the problem are mapped into variables in the code. Provide a description of the data structure in the report.
- Provide a description of the algorithms implemented in your code. You need to convince me that you know how they work. You can consider the use of pseudocode with extended comments to illustrate the algorithms.
- Describe the tests you perform: how they work, what they test exactly, what parameters you have used to run the tests, and what is the outcome. In some cases it may be good to include tables of output or plots of the outcome of the tests. Often, in order to perform tests, you will need to write dedicated code. Make sure to hand in this code as well.
- Describe your experience with the MNIST set in a quantitative way.
- For the report: short is better. However you have to provide me with enough material to evaluate the work you have done!