Assignment 1: Neural Networks
Machine Learning

Deadline: Friday 19 Oct 2018, 21:00

Introduction

In this assignment, you will learn how to implement an Artificial Neural Network with Python and Numpy and train it on toy data. To get started with the assignment, you will need to use the starter code provided with this assignment. For each of the tasks below it should be clear that you have to implement the algorithms yourself, i.e. you cannot use a neural network library and you are not allowed to reuse code from any other sources. The code has to be accompanied with a latex based report in the PDF format.

Your report and source code must be archived in a file named "firstname.lastname" and uploaded to the iCorsi website before the deadline expires. Your report should also contain your name. Late submissions will be subject to a point penalty according to USI regulations.

Where to get help

We encourage you to use the tutorials to ask questions or to discuss exercises with other students. However, do not look at any source code written by others or share your source code with others. Violation of that rule will result in 0 points for all students involved.

Grading

The assignment consists of three parts totalling at 100 points. Bonus points are not necessary to achieve the maximal grade. You will be awarded 5 bonus points for a well-presented report (clear writing, annotated equations), as well as, a good programming style (clear code, useful comments, simple to execute). Further bonus points are elaborated in the text below.

1 The Perceptron (20 points)

Before working on a neural network we will study the perceptron; a linear classifier without an activation function or a very simple single layer network (if you will) as given in Figure 1.

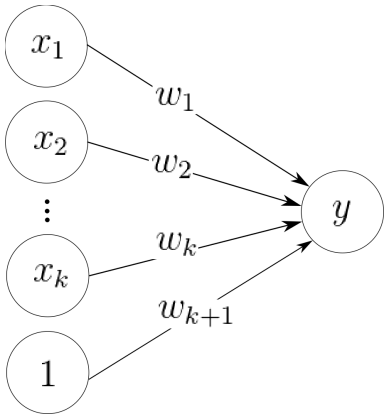

Figure 1: The linear perceptron

x_1	x_2	x_3	t
6	4	1	1
3	1	7	1
6	10	9	1
4	2	4	1
1	5	10	1
5	3	7	1
3	3	4	-1
10	3	3	-1
3	4	9	-1
4	6	6	-1

Table 1: The training dataset

You are given a dataset of size n: $\mathcal{D} = \{(\overline{x_1}, t_1), (\overline{x_2}, t_2), \ldots, (\overline{x_n}, t_n)\}$, where input is represented as k-dimensional vectors $\overline{x_i} \in \mathbb{R}^k$, and the targets $t_i \in \mathbb{R}$ are scalar. The weight vector is therefore $k+1$-dimensional (note the bias term).

1. (3 points) Write down the vectorized equation for the forward pass.
2. (3 points) Write down the vectorized equation for the MSE of the perceptron.
3. (4 points) Determine the derivative of the error function w.r.t weights.
4. (4 points) Write down the equation of the weight update by gradient descent, and explain why it can be used as a learning algorithm. Provide a positive and negative argument.
5. (3 points) Suppose $k = 3$, $n = 10$, with the dataset given in Table 1. The initial weights are $w_1 = -0.1$, $w_2 = -0.3$, $w_3 = 0.2$ and the bias weight is $w_4 = 2$. Compute the weights after one step of gradient descent with learning rate of 0.02. *Hint: MATLAB/Octave/Python might come in handy here.*
6. (3 points) Learning in multi-layer neural networks is usualy done with help of two methods: backpropagation and gradient descent. Describe briefly what role in the learning process each of the two has.

2 Simple Machine Learning Framework (45 points)

In this part we are going to implement a very simple machine learning framework. The goal is that at the end of this part, you will have a rough idea what is happening inside a real world ML framework (e.g. PyTorch, TensorFlow, etc). Using this framework you will be able to easily create arbitrary fully connected neural networks. It will have

the following building blocks: linear layer, sequential layer, tanh and sigmoid activation function and mean-squared error loss. The skeleton is given, you will have to fill in the missing parts. Pay attention to the description in the skeleton file: it clarifies the task further, and also gives some useful hints.

You will test your framework on synthetic two-class spiral dataset. You will use a simple, 3 layer network with tanh activation function on hidden layers and a sigmoid output layer. The output sizes are 50, 30, 1, respectively.

1. (10 points) Implement the activation functions and their derivatives in the code skeleton (*sigmoid*, *tanh*).
2. (10 points) Implement forward and backward pass for the linear layer.
3. (10 points) Implement forward and backward pass for the MSE loss.
4. (5 points) Implement the sequential layer.
5. (10 points) Create the network and implement single step of training.

Your task is to fill missing parts of the framework.py file. Run it with Python 3 in order to verify whether your solution is correct.

3 Handwritten Digit Recognition (35 points)

In this part you will use your ML framework to train a network that can classify handwritten digits. The network is a very simple 2 layer network, with layer output sizes of 20, 10 respectively. Again, the skeleton is given, and you have to fill in the missing parts. After this part, you will have a rough idea how to build a proper training pipeline. Pay attention to the description in the skeleton file: it clarifies the your task further, and also gives some useful hints.

1. (10 points) Split the data into a training and a validation set.
2. (5 points) Implement the main training loop.
3. (10 points) Implement one function for testing the network on validation set and one for testing it on test set.
4. (10 points) Implement early stopping.

Your task is to fill missing parts of the minst.py file. Run it with Python 3 in order to verify whether your solution is correct. Note: The mnist.py and framework.py must be in the same folder for this to work.