Universität des Saarlandes
FR 4.7 Sprach- und Signalverarbeitung

Prof. Dietrich Klakow
Pattern and Speech Recognition

Exercise Sheet 5

Neural Networks

Deadline: 15.12.2016, 23:56

Exercise 5.1 *(2+1 = 3 points)*

In the lecture you have encountered the following activation functions:

- $\mathrm{ReLU}(z) = \max\{0, z\}$
- $\mathrm{g}(z) = \frac{1}{1+\mathrm{e}^{-z}}$

a) Compute the derivative of each of the functions.

b) What do you notice for $\mathrm{g}'(z)$? Which influence might this have for backpropagation?

Exercise 5.2 *(5 + 12 = 17 points)*

In this exercise we want to implement a simple Neural Network. For the functions `predict` and `train` you are not allowed to use an existing implementation.

You can assume that we only deal with fully connected networks as indicated below in figure 1. Furthermore, for simplicity, you can assume that we have only one hidden layer in the network as indicated in figure 1. Therefore you do not have to implement nodes explicitly, i.e. it suffices if you describe the layers by a matrix $\mathbf{W_h}$ and $\mathbf{W_o}$ containing the weights of nodes from the hidden layer and the output layer respectively. Furthermore, you will have a vector $\mathbf{b_h}$ containing the biases of the hidden nodes and $\mathbf{b_o}$, containing the biases of the output layer nodes.

Your task now is to implement a neural network in a script `nn.py`, based on the description above. The implementation should provide the following functionalities:

a) `predict(x)`:
The function `predict` should compute the output of your network. This reflects a Forward Propagation through the network. The output of `predict` is the computed vector $\hat{\mathbf{y}}$ for the input $\mathbf{x}$. As an activation function use the sigmoid function

$$\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}.$$

b) `train(X, epochs, ϵ)`:
The function `train` determines the parameters of the network using the training set `X` consisting of input vectors $\mathbf{x}$ and corresponding output vectors $\mathbf{y}$. To do so, proceed as follows:

For each sample $\mathbf{x}$ in your training data you should

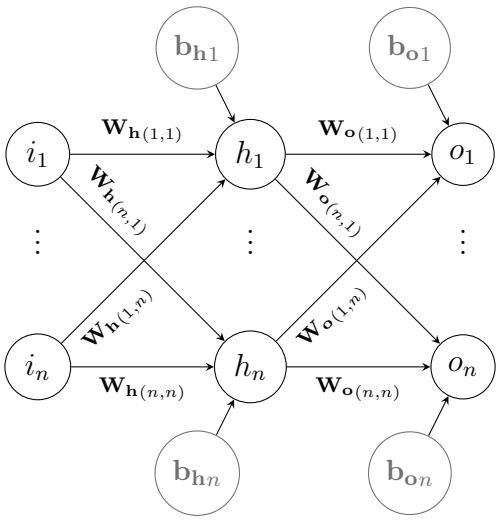

Figure 1: Exemplary Neural Network.

- compute the value $\hat{\mathbf{y}}$ predicted by the network.
- adjust your parameters based on the error using backpropagation. Here you try to minimize the obtained error

$$\text{MSE} = \frac{1}{2}\sum_{i=1}^{n}\left(\hat{\mathbf{y}}_i - \mathbf{y}_i\right)^2.$$

with respect to each of the individual weights in $\mathbf{W_h}$ and $\mathbf{W_o}$, where $\mathbf{y}$ is the true value corresponding to $\mathbf{x}$. I.e. you compute

$$\frac{\partial}{\partial \mathbf{W}_{\mathbf{h}(i,j)}}\text{MSE} \text{ and } \frac{\partial}{\partial \mathbf{W}_{\mathbf{o}(i,j)}}\text{MSE}$$

for every (i, j), to obtain the slope of MSE for each (i, j). Using this information, update each of the weights using the gradient descent method with learning rate ϵ. So your updated weight $\mathbf{W}_{\mathbf{m}(i,j)}$, where $\mathbf{m} \in \{h, o\}$ becomes

$$\mathbf{W}_{\mathbf{m}(i,j)} = \mathbf{W}_{\mathbf{m}(i,j)} - \epsilon \cdot \frac{\partial}{\partial \mathbf{W}_{\mathbf{m}(i,j)}}\text{MSE}.$$

Similarly you have to update the biases of your model.

After you have adjusted the parameters for each sample in $\mathbf{X}$ you have completed a so called epoch.

As you have no parameters available during the first epoch initialize those randomly.

After each epoch you should compute the error your network produces using the adjusted weights on the training data again using the MSE (now for all samples in your training data).

In total your algorithm should run `epochs` epochs.

c) Provide a plot of the cost for each of the epochs. I.e. the x-axis denotes the epoch and the y-axis denotes the error.

As training data use the `mnist` dataset. The python script provided on the webpage shows an implementation of training a neural network to classify the data provided by the

dataset. You can use this script and modify it for your purposes, i.e. you have to implement the optimization part on your own. For any functionality you need that is not related to the backpropagation implementation (e.g. importing the dataset, etc.) feel free to use any library or the given script.

Submission instructions

The following instructions are mandatory. If you are not following them, tutors can decide to not correct your exercise.

Submission architecture

You have to generate a **single ZIP file** respecting the following architecture:

```
tutorial5_<matriculation1>_<matriculation2>_<matriculation3>
|
+--- source
|       |
|       +------- file 1
|       +------- file 2
|       +------- ...
+--- rapport.pdf
+--- README.txt
```

where

- **source** contains the source code of your project,
- **rapport.pdf** is the report where you present your solution with **the explanations** and the plots,
- **README** which contains group member informations (name, matriculation numbers and emails) and a **clear** explanation about how to compile and run your source code

The ZIP filename has to be :

```
tutorial5_<matriculation1>_<matriculation2>_<matriculation3>.zip
```

Some hints

We advice you to follow the following guidelines in order to avoid problems :

- Avoid building complex systems. The exercises are simple enough.
- Do not include any executables in your submission, as this will cause the e-mail server to reject it.

Grading

Send your assignment to the tutor who is responsible of your group:

- Merlin Köhler s9mnkoeh@stud.uni-saarland.de
- Yelluru Gopal Goutam goutamyg@lsv.uni-saarland.de

- Ahmad Taie ataie@lsv.uni-saarland.de

If you are assigned to different tutorials send your assignment to the tutor to whom most of you are assigned to.
The email subject should start with `[PSR TUTORIAL 5]`