EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

# Deep Neural Networks

## Assignment 4

Assignment due by: 05.06.2018, Discussions on: 12.06.2018

**Note:** All programming exercises in this course would be done in python. To ensure that we can run your code, you are not allowed to use any external python libraries except for *numpy*, *scipy*, *matplotlib* (and later *tensorflow* when we get to the corresponding exercises). By default we will expect python 3. If you would like to use python 2.7, then change the first line of the python file accordingly.

### Question 1 Logistic Regression - Theory (2+2+4 Points)

Logistic regression is a simple model for binary classification, i.e. it gives a binary label $y \in \{0, 1\}$ for each example $\boldsymbol{x}$. The probability distribution is defined as:

$$P(y = 1|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{1 + \exp(-\boldsymbol{\theta}^\top \boldsymbol{x})} = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x})$$

with $P(y = 0|\boldsymbol{x}, \boldsymbol{\theta}) = 1 - P(y = 1|\boldsymbol{x}, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ are the parameters of the model that are to be determined.

(a) Calculate the derivative of the sigmoid function $\sigma(x) = (1 + \exp(-x))^{-1}$. Show that the derivative can be written as $\sigma(x)(1 - \sigma(x))$.

(b) The likelihood is the probability that a distribution will produce a certain dataset or point of data. Find an expression for the likelihood of a single data point i.e $P(y = y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\theta})$ and the likelihood of the whole dataset i.e $P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_i P(y = y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\theta})$. This expression should match with the one in part (c) when you take the logarithm.

(c) To find the optimal values of $\boldsymbol{\theta}$, we can maximize the likelihood by differentiating with respect to $\boldsymbol{\theta}$. However it is usually easier to work with the logarithm of the likelihood:

$$\log P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \sum_i y^{(i)} \log \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}^{(i)}))$$

Take the derivative of the log-likelihood and simplify it as far as possible to get an equation that determines the optimal values for $\boldsymbol{\theta}$.

**Question 2 Logistic Regression Implementation (2+2+2+3+3 Points)**

In this exercise you will build a logistic regression model to predict whether a student gets admitted into a university based on the results on two exams. You have historical data from previous applicants that you can use as a training set for logistic regression. Your task is to build a classification model that estimates an applicant' s probability of admission based on the scores from those two exams. An outline of the code is provided in `logistic_regression.py`, and the data is provided in `ex_data.csv`

(a) (Visualization) In this part, your code should load the data and create a 2-D plot where the axes are the two exam scores, and the positive and negative examples are shown with different markers. Don't forget to add ticks, labels and a legend to your plot.

(b) (Sigmoid function) Before you start with the actual cost function, recall that the logistic regression hypothesis is defined as:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x})$$

where $\sigma$ is the logistic sigmoid function, which is defined as in the previous question.
Your first step is to implement the logistic sigmoid function, so it can be called by the rest of your program. When you are finished, try testing a few values to make sure your implementation is right. Your code should also work with vectors and matrices, for which your function should perform the sigmoid function on every element (try to take advantage of $numpy$ broadcasting to get this easily).

(c) (Loss function and gradient) Now you will implement the loss function and gradient for logistic regression. Complete the code in section (c) of `logistic_regression.py` to return the loss and gradient. The loss function for logistic regression is the negative log likelihood

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{m} \left[ y^{(i)} log(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) + (1 - y^{(i)}) log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \right]$$

and the gradient of the loss is a vector of the same length as $\boldsymbol{\theta}$, where the jth element is defined as follows:

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_j} = \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

(d) (Gradient descent) Next, you will implement gradient descent. As you program, make sure you understand what you are trying to optimize and what is being updated.

(e) (Evaluation) After training the model, you now want to evaluate how well it performs. First, find a formula for the decision boundary in terms of $\boldsymbol{\theta}$, i.e. the line where being accepted becomes more likely than being rejected and plot it on the same graph that you plotted the points on in (a). You should see the line separating the two different classes fairly well. Second, you should see how well the model has learned the training data, so find out how many of the examples are classified correctly by the trained model (i.e. find the percentage accuracy).