EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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

Deep Neural Networks

Assignment 5

Assignment due by: 27.11.2019, Discussions on: 04.12.2019

Note: All further programming exercises in this course have to be done in python 3.5. 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).

Question 1 Logistic Regression - Theory (8 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) 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 (b) when you take the logarithm. (2 points)

(b) 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. (4 points)

(c) Now you should expand the logistic regression to a polynomial logistic regression of rank d, assuming $\boldsymbol{x}$ has only 2 elements. Therefore, you have to consider all combinations of elements of $\boldsymbol{x}$, where the sum of their powers is at most d. For example in the case of d=1: $\sigma(\theta_1 x_1 + \theta_2 x_2 + \theta_3)$. How do you have to redesign $\boldsymbol{\theta}$ and $\boldsymbol{x}$?

(2 points)

Question 2 Logistic Regression Implementation (12 Points)

In this exercise you will build a logistic regression model to predict whether a high school student will get 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 `data_train.csv` and `data_test.csv`.

(a) (Visualization) In this part, your code should 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. (2 points)

(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 σ 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). (1 point)

(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:

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

(2 points)

(d) (Gradient descent) Next, implement the gradient descent routine to train the model on the training set. As you program, make sure you understand what you are trying to optimize and what is being updated. (2 points)

(e) (Evaluation) The decision of the model is based on whether the output of the model is greater or less than 0.5. Run your gradient descent and evaluate the accuracy on the test set. Also plot the decision boundary on your plot from (a) (Hint: use a contour plot). (2 points)

(f) As you can see on the plot, the points can't really be separated by a straight line. Add the polynomial extension from (1c) to the code and see how the model performs at degree 3 and 5. Please include the plot (including decision boundary) and the reached accuracy for the degree which results in the best test accuracy in your paper/pdf submission. (3 points)