Lab Course Machine Learning
Exercise Sheet 4

Prof. Dr. Dr. Lars Schmidt-Thieme, Hadi Samer Jomaa
Information Systems and Machine Learning Lab
University of Hildesheim

November 20th, 2016
Submission on November 27th, 2016 at 8:00 am, (on moodle, course code 3113)

Instructions

Please read the lab related instructions, i.e. submission, report format and policies, at `https://www.ismll.uni-hildesheim.de/lehre/prakAIML-16w/exercises/ml_lab_instructions.pdf`

Datasets

1. **Classification Datasets**
   (a) Bank Marketing: `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`
   (b) Occupancy Detection: `https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+`

You are required to pre-process given datasets.

1. convert any non-numeric values to numeric values. For example you can replace a country name with an integer value or more appropriately use hot-one encoding. [Hint: use hashmap (dict) or *pandas.get_dummies*]. Please explain your solution.
2. If required drop out the rows with missing values or NA. In next lectures we will handle sparse data, which will allow us to use records with missing values.
3. Split the data into a train(80%) and test(20%) .

Exercise 1: Linear Classification with Stochastic Gradient Descend/Ascend (10 Points)

In this part you are required to implement linear classification algorithm with stochastic gradient descent/ascend algorithm. Reference lecture `https://www.ismll.uni-hildesheim.de/lehre/ml-16w/script/ml-03-A2-linear-classification.pdf`
For each classification dataset given above

1. A set of training data $D_{train} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})\}$, where $\mathbf{x} \in \mathcal{R}^M$, $y \in \{0, 1\}$, N is number of training examples and M is number of features
2. Linear Regression model is given as $\hat{y}^n = \sigma(\beta^T \mathbf{x}^n)$, σ is a logistic function $\frac{1}{1+e^{-\beta^T \mathbf{x}^n}}$

3. Optimize the loglikelihood function $l(x, y)$ using Gradient Descent algorithm. **Implement (*log-reg-SGA/SGD* and *SGA/SGD* algorithms**). Choose i_{max} between 100 to 1000.

4. You will use *steplengthbolddriver* for step length choose.

   (a) In each iteration of the *SGA/SGD* algorithm calculate $|f(x_i - 1) - f(x_i)|$ and at the end of learning, plot it against iteration number i. Explain the graph.

   (b) In each iteration step also calculate logloss on test set `https://www.kaggle.com/wiki/LogarithmicLoss`, plot it against iteration number i. Explain the graph.

Exercise 2: Implement AdaGrad for adaptive step length (learning rate) (10 Points)

This task you have to implement *AdaGrad* algorithms given in the lecture slides.

1. In each iteration of the *SGA/SGD* algorithm calculate $|f(x_i - 1) - f(x_i)|$ and at the end of learning, plot it against iteration number i. Explain the graph.

2. In each iteration step also calculate logloss on test set `https://www.kaggle.com/wiki/LogarithmicLoss`, plot it against iteration number i. Explain the graph.

Compare AdaGrad with *steplengthbolddriver* algorithm

Compare the logloss graphs of AdaGrad and *steplengthbolddriver* Algorithms. Explain your graph.

Annex

1. You can use numpy or scipy in-build methods for doing linear algebra operations.

2. You can use pandas to read and processing data

3. You can use matplotlib for plotting.

4. You should not use any machine learning library for solving the problem i.e. scikit-learn etc. If you use them you will not get any points for the task.

5. RMSE is explained at `https://www.kaggle.com/wiki/RootMeanSquaredError`