Lab Course Machine Learning
Exercise Sheet 5

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

November 27th, 2017
Submission on December 4th, 2017 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. **Regression Datasets**
   (a) Wine Quality: (use winequality-red.csv)`http://archive.ics.uci.edu/ml/datasets/Wine+Quality`
2. **Classification Datasets**
   (a) Bank Marketing: (use bank.csv) `https://archive.ics.uci.edu/ml/datasets/Bank+Marketing`

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 $pandas.get\_dummies$].
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. normalize our data

Exercise 1: Regularization (8 Points)

You have to implement *Ridge Regression* using mini-Batch Gradient Descent (mini-BGD) algorithm. Now your SGD algorithm will have three hyperparameters i.e. 1) learning rate (stepsize) α, 2) regularization constant λ and 3) number of mini Batches *batchsize*.

1. Implement Ridge Regression using mini-BGD algorithm
2. you can use any algorithm for selecting learning rate i.e. (AdaGrad, BoldDriver or fixed stepsize)

3. Pick three values of α_0 and λ, these values should be picked from relatively small to large. You should keep a fixed *batchsize* $= 50$.

4. Train you model for each combination of the picked values of α and λ, and for each training epoch (an epoch is equal to going over all mini-batches once) record RMSE on training and test data.

5. For each combination of α_0 and λ, plot $RMSE^{train}$ and $RMSE^{test}$ per iteration. [Hint: you can plot $RMSE^{train}$ on positive axis and $RMSE^{test}$ on negative axis of same plot].

Exercise 2: Hyper-parameter tuning and Cross validation (12 Points)

In this section you will implement *grid search* with *k-fold cross-validation* for model selection i.e. choosing best hyperparameters. You will use your implementation from Exercise 1: *Ridge Regression* using mini-Batch Gradient Descent (mini-BGD) algorithm.

- Pick a range of α_0 and λ defined on grid. You can choose fixed *batchsize*=50.
- Implement *k-fold cross-validation* protocol for grid search. For each combination of α_0 and λ you will perform *k-fold cross-validation*. let $k = 5$ in this case.
- Keep track of mean performance (i.e. RMSE value) across *k-folds* for each set of hyperparameters. Plot on the grid α_0 vs λ the RMSE score for all combinations. [Hint: you can use a 3D plot with axes={α_0, λ, RMSE}]
- Finally, for the optimal value of α_0 and λ, train your model on complete training data and evaluate on test data.
- Plot $RMSE^{train}$ and $RMSE^{test}$ per iteration. [Hint: you can plot $RMSE^{train}$ on positive axis and $RMSE^{test}$ on negative axis of same plot]. Compare your result with results in previous plots.

[Hint: If you were unable to complete Exercise 1, you can still complete Exercise 2 by using linear regression implementation from Exercise Sheet 3 and adding regularization term. There will be some penalty for this.]

Bonus: Implement Newton's Method (5 Points)

This is a bonus exercise. In this exercise you are required to implement Newton's method to solve logistic regression with regularization. The algorithm is given at `https://www.ismll.uni-hildesheim.de/lehre/ml-16w/script/ml-03-A2-linear-classification.pdf`.

1. Pick three values of α and λ, these values should be picked from relatively large to small. You can choose fixed *batchsize*=50.
2. split your data into train and test
3. implement Newton's method [Hint: use scipy or numpy for inverting a matrix or other matrix operations, do not use *scipy.optimize.newton*)
4. Train you model for each combination of the picked values of α and λ, and for each iteration (one iteration is equal to updating all model parameters once) record RMSE on training and test data.
5. Plot $RMSE^{train}$ and $RMSE^{test}$ per iteration. [Hint: you can plot $RMSE^{train}$ on positive axis and $RMSE^{test}$ on negative axis of same plot].

Annex

1. Following lecture is relevant this exercise `https://www.ismll.uni-hildesheim.de/lehre/ml-16w/script/ml-04-A3-regularization.pdf`
2. You can use numpy or scipy in-build methods for doing linear algebra operations.
3. You can use pandas to read and processing data
4. You can use matplotlib for plotting.
5. 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.
6. RMSE is explained at `https://www.kaggle.com/wiki/RootMeanSquaredError`.
7. LogLoss is explained at `https://www.kaggle.com/wiki/LogarithmicLoss`.