

The Experiment Report of

Deep Learning

**College Software College**

**Subject Software Engineering**

**Members**   **陈奕男**

**Student ID 201710106543**

**E-mail easyhard@qq.com**

**Tutor**   **Li Dong**

**Date submitted** **2017.12.8**

# Topic: Linear Regression, Linear Classification and Gradient Descent

# Time: 2017-12-08 12:00 AM

# Reporter: 陈奕男

# Purposes:

1. Further understand of linear regression and gradient descent.
2. Conduct some experiments under small scale dataset.
3. Realize the process of optimization and adjusting parameters.

# Data sets and data analysis:

Linear Regression uses [Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 506 samples and each sample has 13 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.   
Linear classification uses [australian](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html" \l "australian" \t "_blank) in [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 690 samples and each sample has 14 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.

# Experimental steps:

The experimental code and drawing are completed on jupyter.

*Linear Regression and Gradient Descent*

1. Load the experiment data. You can use [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html" \t "_blank) function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html" \t "_blank) function. Test set is not required in this experiment.
3. Initialize linear model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient  toward loss function from all samples.
6. Denote the opposite direction of gradient  as .
7. Update model: .  is learning rate, a hyper-parameter that we can adjust.
8. Get the loss  under the training set and  by validating under validation set.
9. Repeate step 5 to 8 for several times, and **drawing graph of  as well as  with the number of iterations**.

*Linear Classification and Gradient Descent*

1. Load the experiment data.
2. Divide dataset into training set and validation set.
3. Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient  toward loss function from all samples.
6. Denote the opposite direction of gradient  as .
7. Update model: .  is learning rate, a hyper-parameter that we can adjust.
8. **Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative.** Get the loss  under the trainin set and  by validating under validation set.
9. Repeate step 5 to 8 for several times, and **drawing graph of  as well as  with the number of iterations**.

# Code:

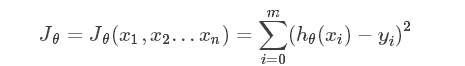
**(see ClassificationExperiment.ipynb and RegressionExperiment.ipynb on github:** <https://github.com/easyhard007/ML2017-lab-01.git>**)**

(Fill in the contents of 8-12 respectively for linear regression and linear classification)

linear regression

# Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):

Use loss function on validation set to validate

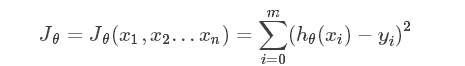


# The initialization method of model parameters:

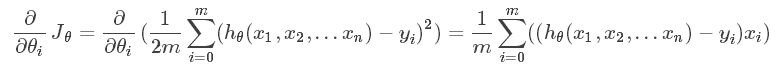
All-zero Initialization

# The selected loss function and its derivatives:

loss function:



Derivatives:



# Experimental results and curve:

## Hyper-parameter selection (η, epoch, etc.):

eta = 0.001 # Learning Rate η

iter = 100 # Iteration times

## Assessment Results (based on selected validation):

Loss on training datasets converge to around: 23.05

Loss on validation datasets converge to around: 26.10

## Predicted Results (Best Results):

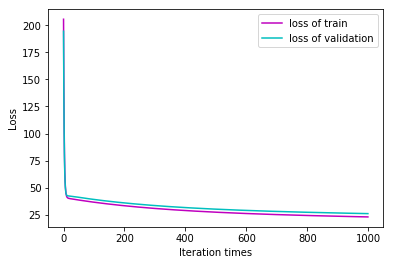
Use an iteration of 10000 and eta=0.01 can get better result, but take much more time

Loss on training datasets converge to around: 8.42

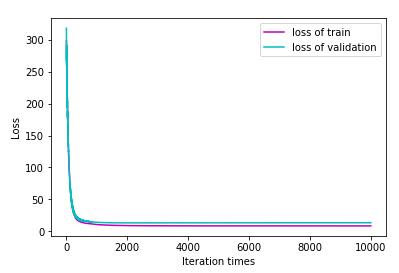
Loss on validation datasets converge to around: 13.42

## Loss curve:

Iter = 1000 , eta = 0.001



Iter = 10000 , eta = 0.01



# Results analysis:

The results of the experiment is consistent with expected. The loss curve descent down like “J” and the loss on training datasets is a bit smaller than it on validation datasets.

## linear classification

# Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.):

Use loss function and accuracy rate on validation set to validate:

Accuracy rate:

# The initialization method of model parameters:

All-zero Initialization

# The selected loss function and its derivatives:

loss function:

Derivatives:

# Experimental results and curve:

## Hyper-parameter selection (η, epoch, etc.):

eta = 0.01 # Learning Rate η

iter = 200 # Iteration times

## Assessment Results (based on selected validation):

Loss on training datasets converge to around: 87.47

Accuracy Rate on training datasets converge to around: 87.54%

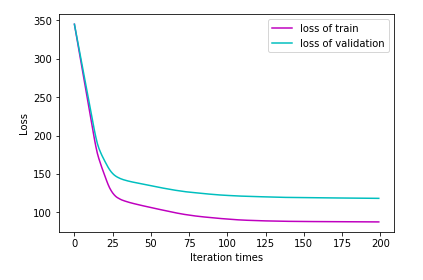
Loss on validation datasets converge to around: 118.17

Accuracy Rate on validation datasets converge to around: 83.48%

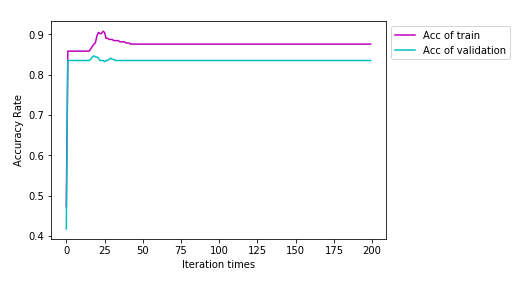
## Predicted Results (Best Results):

（None）

Loss curve:



Accuracy rate curve:



# Results analysis:

The results of the experiment is consistent with expected. The loss curve descent down like “J” and the loss on training datasets is a bit smaller than it on validation datasets.

The accuracy rate raised up and converge to about 83% to 88% on the graph, and the accuracy rate on training set is a bit higher than it on validation set.

# Similarities and differences between linear regression and linear classification:

Similarities: The procedure of the two experiment is similar, including the building-up of Loss Function, the iteration and gradient descent of the loss function.

Differences:

1. The loss function chosen for two experiments are different.
2. As the iteration times grows up, the loss function of linear regression becomes smaller and smaller, but the loss function of linear classification vibrates near a specific value.

# Summary:

Through this project, we mastered the basic steps of linear regression and linear classification, and also have a deeper understand of machine learning. The most interesting things is that, during the linear regression or classification, we don’t need to know what the meaning of the data-set is, what the meaning of each feature is. All we need is to throw the data set in, and get the best weight of features.