

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: Logistic Regression, Linear Classification and Stochastic Gradient Descent

# Time: 2017-12-15 12:00 AM

# Reporter: 陈奕男

# Purposes:

1. Compare and understand the difference between gradient descent and stochastic gradient descent.
2. Compare and understand the differences and relationships between Logistic regression and linear classification.
3. Further understand the principles of SVM and practice on larger data.

# Data sets and data analysis:

Experiment uses [a9a](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a9a) of [LIBSVM Data](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/), including 32561/16281(testing) samples and each sample has 123/123 (testing) features. Please download the training set and validation set.

# Experimental steps:

The experimental code and drawing are completed on jupyter.

*Logistic Regression and Stochastic Gradient Descent*

1. Load the training set and validation set.
2. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient  toward loss function from **partial samples**.
5. **Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).**
6. Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative**. Predict under validation set and get the different optimized method loss ，， and .
7. Repeate step 4 to 6 for several times, and **drawing graph of ，，and  with the number of iterations**.

*Linear Classification and Stochastic Gradient Descent*

1. Load the training set and validation set.
2. Initalize SVM model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient  toward loss function from **partial samples**.
5. **Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).**
6. Select the appropriate threshold, mark the sample whose predict scores **greater than the threshold as positive, on the contrary as negative**. Predict under validation set and get the different optimized method loss ，， and .
7. Repeate step 4 to 6 for several times, and **drawing graph of ，，and  with the number of iterations**.

# Code:

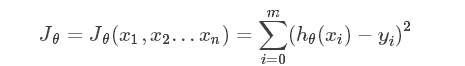
**(see ClassificationExperiment.ipynb and RegressionExperiment.ipynb on github:** <https://github.com/easyhard007/ML2017-lab-02.git>**)**

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

Logistic Regression

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

Use loss function on validation set to validate

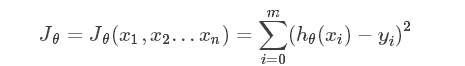


# The initialization method of model parameters:

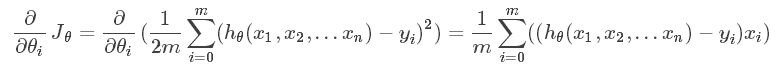
All-zero Initialization

# 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):

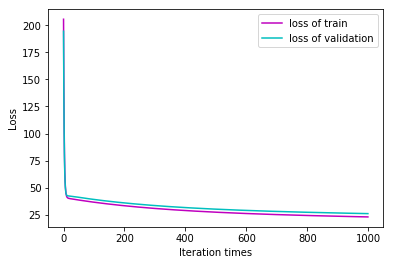
Use an iteration of 10000 and eta=0.01 can get better result, but take much more time

Loss on training datasets converge to around: 8.42

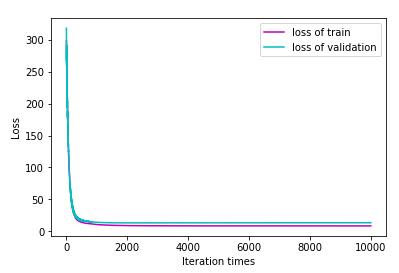
Loss on validation datasets converge to around: 13.42

## 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:

# 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

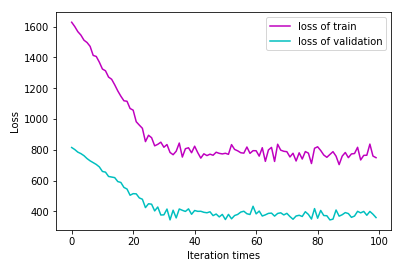
Accuracy Rate on validation datasets converge to around: 83.48%

## Predicted Results (Best Results):

（None）

Loss curve:

LAdaDelta:



# 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 much smaller than it on validation datasets for the size of training set is twice bigger as the validation one.

# 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.