

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  王泽众 **Wesley Wang**

**Student ID 201530612958**

**E-mail w2w.wzz@foxmail.com**

**Tutor**  **谭明奎**

**Date submitted** **2017.12 . 2**

**1. Topic: Linear Regression, Linear Classification and Gradient Descent**

**2. Time: 2017.12.2**

**3. Reporter:王泽众 Wesley Wang**

**4. 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.

**5. Data sets and data analysis:**

Linear Regression uses [Housing](https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html" \l "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) 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.

**6. Experimental steps:**

*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) 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) 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**.

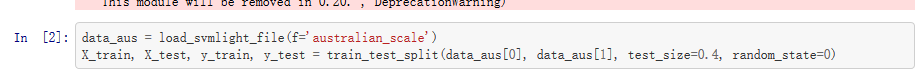
**7. Code:**

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

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

data\_aus = load\_svmlight\_file(f='australian\_scale')

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data\_aus[0], data\_aus[1], test\_size=0.4, random\_state=0)



**9. The initialization method of model parameters:**

n\_features = X.shape[1]

n\_targets = 1

self.w\_ = np.zeros([n\_features])

self.b\_ = 0

1. **The selected loss function and its derivatives:**

def calc\_error(self, X, y, w, b):

'''

error = 0.5||w||^2 + C\* sum (max(0, 1-yi(wTxi + b)))

'''

hinge = 1 - (X.dot(w) + b)\*y

hinge = np.array([max(0, x) for x in hinge])

# print(hinge)

# print('hingesum', np.sum(hinge))

# print('zhengze', w\*\*2)

error = 0.5 \* np.sum(w \*\* 2) + self.c\_ \* np.sum(hinge)

return error

1. **Experimental results and curve:**

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

Regression

Leaning rate ： 0.01

Turns： 50

Classification

Learning rate ：0.001

Tradeoff c ： 0.1

Turns： 50

## Assessment Results (based on selected validation):

Regression：

Loss = MSE

Classification：

Loss =

## Predicted Results (Best Results):

**Regression:**

---------------------------

Iteration End

w: [-4.83305464 -2.30611534 -2.0858356 -3.07407123 -2.15016283 1.60893886

0.70675394 -2.35505243 -1.55135196 -1.50787726 -0.48788494 4.22983774

-3.721055 ]

b: 5.03061596525

MSE: 57.4561790919

---------------------------

**Classification:**

---------------------------

Iteration End

w: [ 0.01697354 0.03291524 0.0550113 0.06938791 0.16891487 0.10010861

0.05819778 0.76872867 0.31006157 0.05132718 0.00762425 0.05008541

-0.03272936 0.01837882]

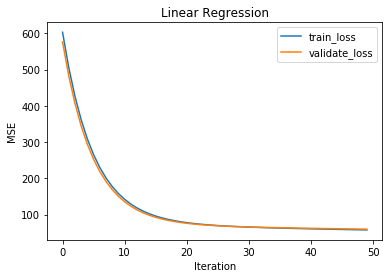
b: 0.075

Loss: 14.4064919084

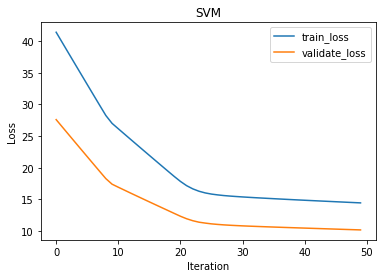
---------------------------

## Loss curve:

Regression:



Classification:



1. **Results analysis:**

Regression :

With the iteration, loss gradually decreases, and finally tents to smooth.

The learning rate have a greate influence on the performance of linear regression.

Classification :

With the iteration, loss gradually decreases, and finally tents to smooth.

The learning rate have a greate influence on the performance of linear regression either. What’s more, the tradeoff coefficient also have a great influence.

1. **Similarities and differences between linear regression and linear classification:**

Regression :

With the iteration, loss gradually decreases, and finally tents to smooth.

The learning rate have a greate influence on the performance of linear regression.

Classification :

With the iteration, loss gradually decreases, and finally tents to smooth.

The learning rate have a greate influence on the performance of linear regression either. What’s more, the tradeoff coefficient also have a great influence.

1. **Summary:**

Linear Regression and Linear Classification can deal with some of simple situations, and it’s ability is limited. It can’t tackle nonlinear problem.