

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**

**Student ID** 201530613207

**E-mail 978163495@qq.com**

**Tutor**   **mingkui Tan**

**Date submitted** **2017.12 .3**

**1. Topic:** LinearRegression& Linear Classification & Gradient Descent

**2. Time:** 2017.12.2 AM 9:00-12:00

**3. Reporter:** peizhuoXie

**4. Purposes:**

* Further understand the principle of linear regression and gradient decent method.
* Practicing on a small data set
* Understand the process of optimizing the super parameters

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

In the linear regression, using the Housing data set from LIBSVM Data, which include 506 samples and 13 features in each sample.

In the linear classification, using the Australian data set from LIBSVM Data, which include 690 samples and 14 features in each sample.

**6. Experimental steps:**

* Using the function load\_svmlignt\_file from sklearn read the data set.
* Using function train\_test\_split from sklearn to spilt the data into train set and validation set.
* Initializing the parameters randomly.
* Using MSE loss/ Hinge loss and compute all the samples’ loss.
* Update the loss function by gradient decent.
* Repeat the above steps for some times and draw the figure of loss-iteration.

**7. Code:**

w = np.random.random(size=(feature\_size + 1, 1))  
  
lr = 0.001  
ite = []  
train\_loss\_set = []  
val\_loss\_set = []  
for i in range(0, 200):  
 ite.append(i)  
 gradient = -np.matmul(train\_x.transpose(), train\_y - np.matmul(train\_x, w))  
 w -= lr \* gradient  
  
 train\_loss = np.mean((train\_y - np.matmul(train\_x, w)) \*\* 2)  
 train\_loss\_set.append(train\_loss)  
 val\_loss = np.mean((val\_y - np.matmul(val\_x, w)) \*\* 2)  
 val\_loss\_set.append(val\_loss)  
  
plt.plot(ite, train\_loss\_set, label='train\_loss')  
plt.plot(ite, val\_loss\_set, label='val\_loss')  
plt.xlabel('Iteration')  
plt.ylabel('Loss')  
plt.legend()  
plt.show()

w = np.random.rand(feature\_size + 1, 1)  
  
# training  
iteration = 100  
lr = 0.01  
C = 0.1  
for i in range(0, iteration):  
 ite.append(i)  
 pred = np.matmul(train\_x, w)  
 hinge\_loss = np.maximum(1 - train\_y \* pred, 0)  
 train\_loss = np.mean(hinge\_loss\*\*2) + C \* np.sum(w\*\*2)  
 gradient = -np.matmul(train\_x.transpose(), hinge\_loss \* train\_y) / len(train\_y)  
 w -= lr \* (gradient + 2 \* C \* w)  
 train\_loss\_set.append(train\_loss)  
  
 val\_pred = np.matmul(val\_x, w)  
 val\_hinge\_loss = np.maximum(1 - val\_y \* val\_pred, 0)  
 val\_loss = np.mean(val\_hinge\_loss\*\*2) + C \* np.sum(w\*\*2)  
 val\_loss\_set.append(val\_loss)  
  
plt.plot(ite, train\_loss\_set, label='train')  
plt.plot(ite, val\_loss\_set, label='validation')  
plt.xlabel('Iteration')  
plt.ylabel('Loss')  
plt.legend()  
plt.show()

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

hold-out

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

Random initialization

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

**Linear Regression:**

The include the bias of the linear regression.

**Linear Classification:**

**11. Experimental results and curve:**

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

Linear regression:

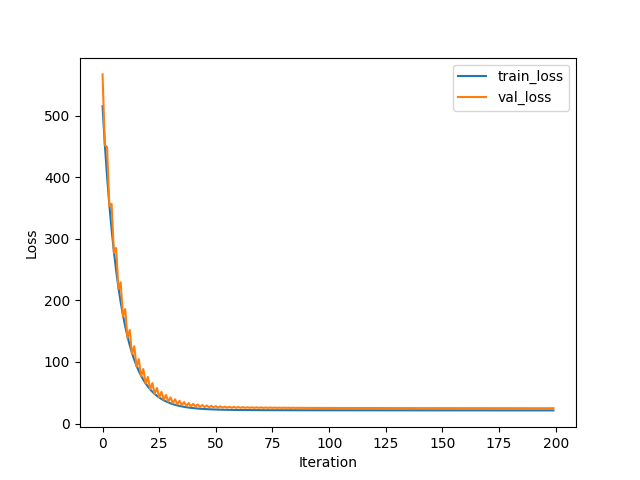
* Learning rate: 0.001
* Gamma = 0
* Iteration = 200

Linear classification:

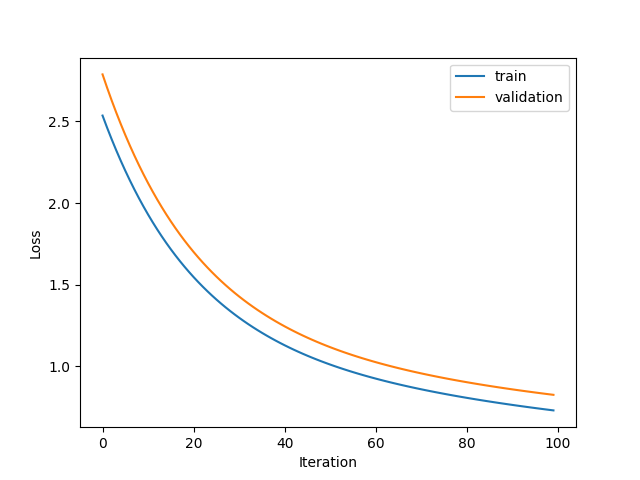
* Learning rate: 0.01
* Gamma = 0.1
* Iteration = 100

## Loss curve:

Linear regression:



Linear classification:



**12. Results analysis:**

This two methods perform well on Houing dataset and Australian dataset seperatively.

Linear regression model reaches almost its best result after 25 iterations, while the linear classification model reaches almost its best result after 100 iterations.

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

**14. Summary:**