

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  Junwen Mo

**Student ID 201530612545**

**E-mail 545096186@qq.com**

**Tutor**   **Mingkui Tan**

**Date submitted** **2017. .**

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

**2. Time: 2017.12.02**

**3. Reporter:Junwen Mo**

**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.

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

1)Linear Regression uses Housing in LIBSVM Data, 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.

2)Linear classification uses australian in LIBSVM Data, 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.

1. **Experimental steps:**

**Linear Regression and Gradient Descent**

1. Load the experiment data. You can use load\_svmlight\_file function in sklearn library.
2. Devide dataset. You should divide dataset into training set and validation set using train\_test\_split 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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: . η is learning rate, a hyper-parameter that we can adjust.
8. Get the loss Ltrain under the training set and Ltest by validating under validation set.
9. Repeat step 5 to 8 for several times, and drawing graph of Ltrain as well as Ltest 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 G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: . η 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 Ltrain under the training set and Ltest by validating under validation set.
9. Repeat step 5 to 8 for several times, and drawing graph of Ltrain as well as Ltest with the number of iterations.
10. **Code:**

**Linear Regression:**

for i in range(epoch):

g=lamda\*w-x\_train.T\*(y-x\_train\*w)

w=w-eta\*g;

**Linear Classification:**

for i in range(epoch):

l=(1-np.multiply(y,x\*w))>0

tmp=np.multiply(y,l)

g=-C\*np.sum(np.multiply(x\_train,tmp),0).T

w=w-eta\*g

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

**Linear regression**: cross-validation

**Linear classification**: cross-validation

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

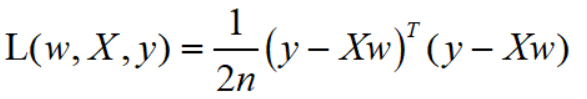
**Linear regression**: Standard normal distribution initialize

**Linear classification**: Standard normal distribution initialize

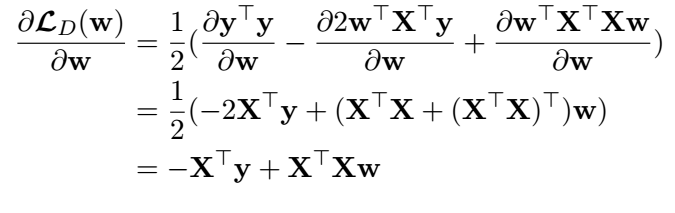
1. **The selected loss function and its derivatives:**

**Linear regression:**

Loss function:

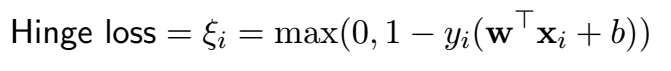


Derivatives:

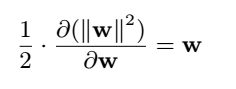


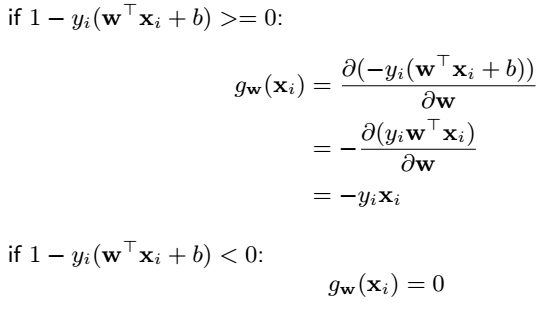
**Linear classification:**

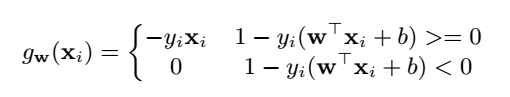
Loss function:

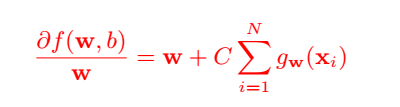


Derivatives:









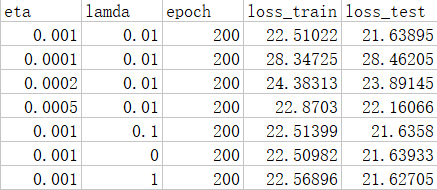
1. **Experimental result and curve:**

**Linear Regression:**

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

## η=0.001 lambda=1 epoch=200

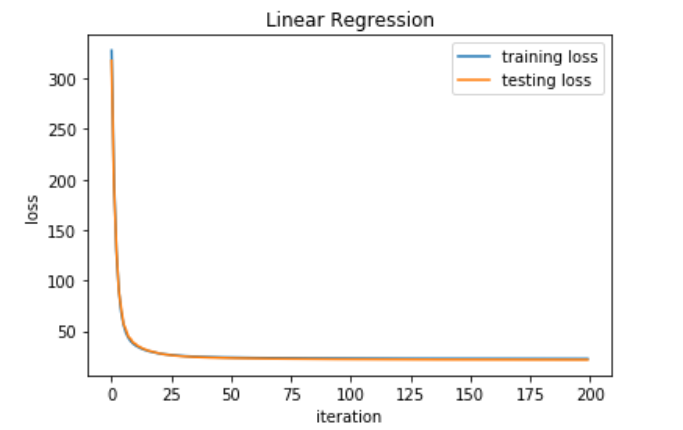
## Assessment Results (based on selected validation):



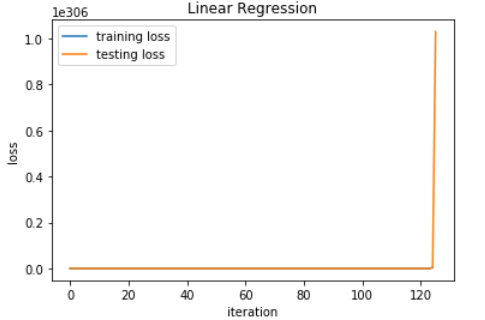
## Predicted Results (Best Results):The best validation loss is 21.62705.

## Loss curve:

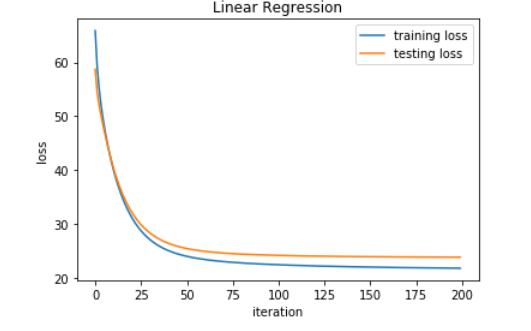
Eta=0.001 lambda=1 epoch=200



Eta=0.01 lambda=0.01 epoch=200



Eta=0.0005 lambda=0.01 epoch=200



**Linear Classification:**

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

## η=0.0005 C=1 epoch=600 threshold=0.5

## Assessment Results (based on selected validation):

## When η=0.0005 C=1 epoch=600 threshold=0.5,

## Training accuracy is 85.69% while validation accuracy is 84.05%.

## When η=0.0002 C=1 epoch=600 threshold=0.5,

## Training accuracy is 84.42% while validation accuracy is 83.33%.

## When η=0.0001 C=1 epoch=600 threshold=0.5,

## Training accuracy is 83.87% while validation accuracy is 82.60%.

## When η=0.0005 C=0.8 epoch=600 threshold=0.5,

## Training accuracy is 85.53% while validation accuracy is 83.33%.

## When η=0.0005 C=2 epoch=600 threshold=0.5,

## Training accuracy is 85.14% while validation accuracy is 83.33%.

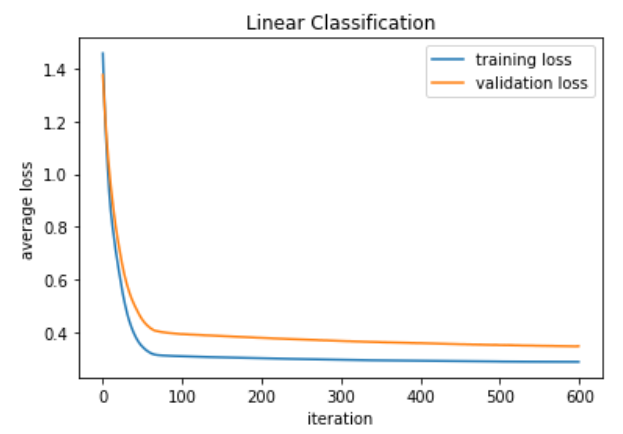
## When η=0.0005 C=1 epoch=600 threshold=0,

## Training accuracy is 85.87% while validation accuracy is 84.06%.

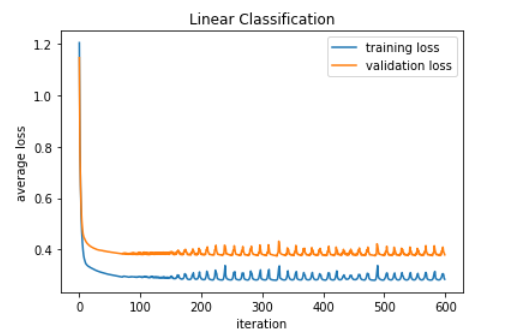
## Predicted Results (Best Results):The best validation accuracy is 84.06% and the best training accuracy is 85.87%.

## Loss curve:

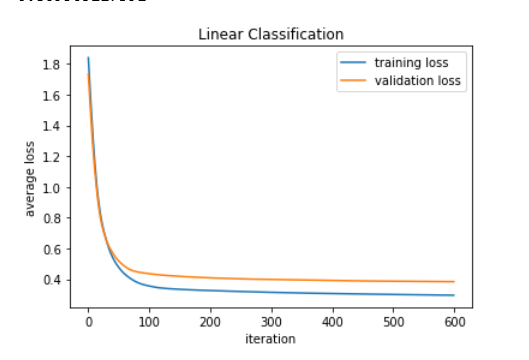
Eta=0.0005 C=1 epoch=600



Eta=0.0005 C=2 epoch=600



Eta=0.0001 C=1 epoch=600



1. **Results analysis:**

**Linear Regression:** The assessment results shows that the best learning rate is 0.001, because the value of the loss function declined more quickly with this learning rate than the leaning rate which is smaller than it. When I try using the gradient descent with the learning rate of 0.002, the value of the loss function went up quickly. Also, we can find that with regularization, the training loss is a little bigger than the training loss without regularization, while the test loss is a little smaller, which shows that the regularization did have the ability to avoid the overfitting. Last, we can observe that the training loss is a little smaller than validation loss sometimes, which may be caused by the similarity of the distribution of the training set and the validation set.

**Linear Classification:** The factor C I used is 1 because when the bigger C makes the loss unstable, while the smaller C makes the accuracy smaller. The learning rate is 0.0005 because the bigger learning rate makes the loss unstable, which means it goes up and down during the learning process, while the smaller learning rate makes it converge slowly. In this experiment, we can find that the average loss of the training set and the one of the validation set are more different. Last, we can found that with the smaller threshold, the accuracy is a little better, because the the threshold is the measurement about the tolerance of the samples within the margin. The smaller threshold means the better acceptance to the samples with a little error.

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

**Similarity:** They learn the same parameter w because they are both linear. The format of the equation is similar, they are both y=wx. In this experiment, they both use the gradient descent to learn.

**Differences:** Their goal functions and the loss functions are different. And the assessment methods are different because the Linear Classification use the accuracy as the evaluation criterion mainly while the Linear Regression use the loss as the evaluation criterion.

**14. Summary:**  We do the experiment of the Linear Regression and Linear Classification with a small dataset, which make me know more about them. In this experiment, I found that tuning is boring but important in the machine learning. With a better hyper-parameter, the algorithm will perform better. For example, higher learning rate may cause divergence but accelerate the learning process while smaller learning rate will slow down the process. And the initialization of the w is also important to the learning process and the result. Setting the w to the zero vector will cause slower convergence in this experiment, while the standard normal initialization makes convergence faster.