

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **梁婧**

**Student ID 201530741368**

**E-mail qwers97@126.com**

**Tutor**   **Mingkui Tan**

**Date submitted** **2017. 12 .2**

**1. Topic: Linear Regression and Classification with Batch Gradient Descent**

**2. Time: 2017/12/2**

**3. Reporter: 梁婧**

**4. Purposes:**

**1.Regression with linear model.**

**2.Classification with support vector machine linear classification model.**

**3.To use batch gradient descent model to find the better weights for the linear model and gain less loss.**

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

**Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features.**

**Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. You are expected to download scaled edition.**

**The ratio to divide the training set and validation set is 0.4**

**What’more, I add a all-one column to the set to avoid calculation b specially.**

**6. Experimental steps:**

***Linear Regression and Gradient Descent***

1. **Load the experiment data.**
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 toward loss function from all samples.**
6. **Denote the opposite direction of gradient**
7. **Update model: η learning rate, a hyper-parameter that we can adjust.**
8. **Get the loss: loss under the training set and loss by validating under validation set.**
9. **Repeate step 5 to 8 for several times, and drawing graph of losses 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**
6. **Denote the opposite direction of gradient**
7. **Update model: ηis 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 of traing and validation set.**
9. **Repeate step 5 to 8 for several times, and drawing graph of losses with the number of iterations**

**7. Code:**

**See .ipynb files, with all the code, comments and graphs。**

**Regression:**

**gradientdecsent():**

**hypothesis=np.dot(x,w) #所有样本**

**k,j=np.shape(x)**

**x2=x.transpose()**

**gradient=np.dot(x2,hypothesis-y) /k**

**w=w-rate\*gradient**

**Classification:**

**gradientdecsent():**

**m,n=np.shape(x)**

**gradient=0**

**C=1/m**

**for i in range(0,m): #对所有的样本**

**condition=1-y[i]\*(np.dot(w.T,x[i])+b) #梯度情况的判定**

**if condition>=0:**

**gradient=gradient -np.dot(y.T,x)\*C**

**else:**

**gradient=gradient**

**w=w-rate\*gradient/m**

(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.):**

**I divide the set to training set and validation set, which is a way of hold-out method.**

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

**For regression experiment, I initialize the parameters with normal distribution. For classification experiment, I initialize it all to zero.**

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

**Linear regression:**

**I select the MSE as the loss function:**

MSE=1/2n∑(Yi−f(x))2

**the derivatives is**

XT(Xw-y)/n

**Linear Classification:**

**I select the hinge loss as the loss function:**

L(y)=C\* ∑max(0,1−y\*wx)

**the derivatives is:**

**-yixi 1-yi\*wTxi>=0**

**0 1-yi\*wTxi<0**

**11. Experimental results and curve:**

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

the learning rate of regression is 0.003

the iteration times is 2500

the learning rate of classification is 0.0001

the iteration times is 3000

the threshold is 0.01

## Assessment Results (based on selected validation):

Regression:

Loss under 40 after 2000 iterations

Classification:

Loss under 0.5 after 1700 iterations

## Predicted Results (Best Results):

Regression:

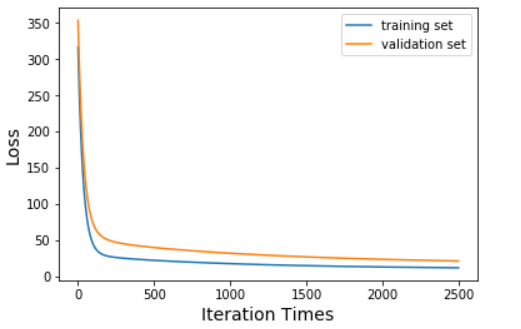
Loss under 20 after 1800 iterations

Classification:

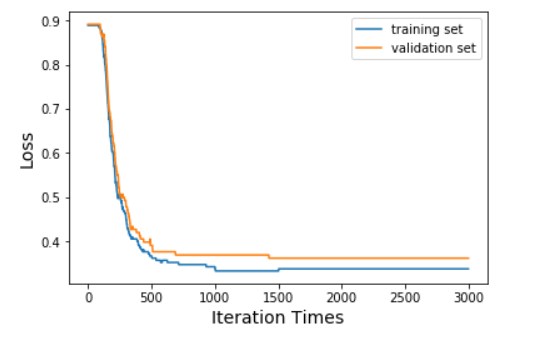
Loss under 0.3 after 1500 iterations

## Loss curve:

Regression:



Classification:



1. **Results analysis:**

Regression:

The loss curve fall severely for the first 100 iterations. Then, the loss curve become flat and finally converge after 2000 iteration times.

Classification:

The loss curve fall severely for the first 500 iterations. Then, the loss curve become flat and finally converge after 1500 iteration times. We can also see that there are some small waves in our loss curve.

What’s more, we can both curve, the training set is below the validation set, that is because we update weights with training set only.

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

**They are all linear model and output a linear form: wTx+b**

**They are all output continuous values directly.**

**However, the linear regression try to get a linear relationship between our feature and our label. The model try to describe and fit the points in our samples.**

**The linear classification, though the model directly output a continuous value, we need a threshold to classify all the samples to two parts or we can that is two classes. Finally, the output is discrete.**

1. **Summary:**

**In this experiment, we just start to use linear model to solve two important questions: regression and classification, using MSE and hinge loss respectively.**

**To optimize our weights, we use batch gradient descent method. That is, using all samples to update our gradient then use average of gradient to update weights until convergence.**