

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  **Jin Chengneng**

**Student ID 201530611838**

**E-mail jinchengneng@gmail.com**

**Tutor**   **Dr. Tan**

**Date submitted**  **2017. 12 .07**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

1. **Time:**

2017/12/02 —— 2017/12/07

1. **Reporter:**

Jin Chengneng

1. **Purposes:**

① Further understand linear regression, linear classification and gradient descent;

② Gain a basic concept about machine learning, and learn some tips of adjusting hyper-parameters of gradient descent.

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

Ⅰ.Linear Regression:

Source: UCI / **Housing** (Boston)

# of data: 506

# of features: 13

Link:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html#housing

Ⅱ.Linear Classification:

Source: Statlog / **Australian**

# of classes: 2

# of data: 690

# of features: 14

Link:https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/australian\_scale

1. **Experimental steps:**

**LinLinear Regression and Gradient Descent**

① Load the experiment data. You can use load\_svmlight\_file function in sklearn library.

② Divide 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: Wt=Wt−1+ηD. η is learning rate, a hyper-parameter that we can adjust.

⑧ Get the loss Ltrain under the training set and Lvalidation by validating under validation set.

⑨ Repeat step 5 to 8 for several times, and drawing graph of Ltrain as well as Lvalidation 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.

1. Choose loss function and derivation: Find more detail in PPT.

⑤ Calculate gradient G toward loss function from all samples.

1. Denote the opposite direction of gradient G as D.

⑦ Update model: Wt=Wt−1+ηD. 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 train set and Lvalidation by validating under validation set.

⑨ Repeat step 5 to 8 for several times, and drawing graph of Ltrain as well as Lvalidation with the number of iterations.

**7. Code:**

**LinLinear Regression and Gradient Descent**

**Loss function:**

**def** cal\_loss(x,y,theta):

**return** (1/2 \*(np.linalg.norm(x\*theta-y))\*\*2)/ (x.shape[0])

**gradient descent:  
for** i **in** range(0, maxIteration):

gradient =2\* train\_features.T \* (train\_features \* theta - train\_targets)

theta = theta - alpha \* gradient

**Linear Classification and Gradient Descent**

**Loss function and gradient function:**

**def** cal\_hinge\_loss\_and\_grad(theta,x,y):

loss,grad = 0,0

**for** j **in** range(x.shape[0]):

v = y[j]\*((theta.T).dot(x\_todense\_T[j])[0,0])

loss += max(0,1-v)

grad += 0 **if** v > 1 **else** -y[j]\*x[j].todense()

**return** (loss/x.shape[0],grad)

**Gradient descent:**

**def** gradientDescent(w):

**for** i **in** range(maxIteration):

current\_train\_loss, gradient = cal\_hinge\_loss\_and\_grad(w,train\_features,train\_targets)

train\_loss.append(current\_train\_loss)

evaluation\_loss.append( cal\_hinge\_loss(w,test\_features,test\_targets))

w = w-learning\_rate\*gradient.T

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

Hold-out method is used in selection of validation.

Codes are shown as below:

**LinLinear Regression and Gradient Descent**

**features, labels = load\_svmlight\_file("housing\_scale")**

**train\_features, test\_features, train\_targets, test\_targets = train\_test\_split(features, labels, test\_size=0.33)**

**Linear Classification and Gradient Descent**

**features, labels =load\_svmlight\_file("housing\_scale")**

**train\_features, test\_features, train\_targets, test\_targets = train\_test\_split(features, labels, test\_size=0.33)**

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

**LinLinear Regression and Gradient Descent**

theta = np.ones((n, 1))

alpha = 0.00005

maxIteration = 1000

**Linear Classification and Gradient Descent**

theta = np.ones((features.shape[1], 1))

maxIteration = 100

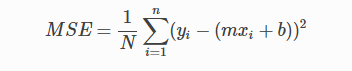
c = 0.5

learning\_rate = 0.001

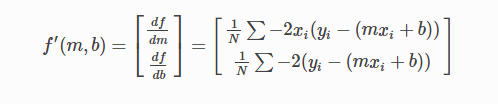
1. **The selected loss function and its derivatives:**

**LinLinear Regression and Gradient Descent**

Loss function:

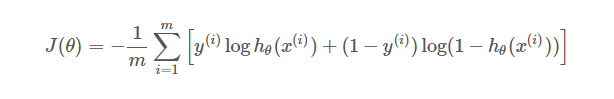


It’s derivatives:

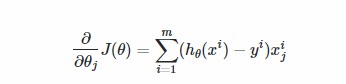


**Linear Classification and Gradient Descent**

Loss function:



It’s derivatives:



**10. Experimental results and curve:**

**Linear Regression and Gradient Descent**

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

η = 0.00005

epoch = 1000

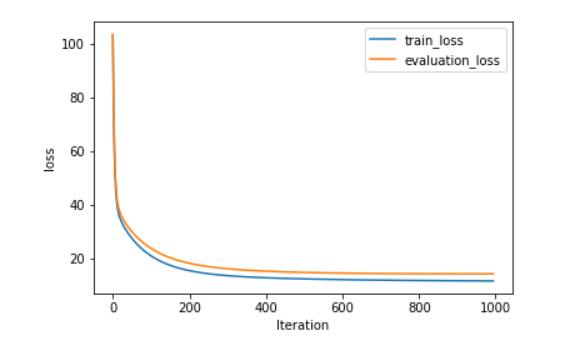
## Assessment Results (based on selected validation):

Min\_loss = 12.9073

## Predicted Results (Best Results):

Min\_loss = 11.0944

## Loss curve:



**Linear Classification and Gradient Descent**

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

η= 0.001

epoch = 100

C = 1

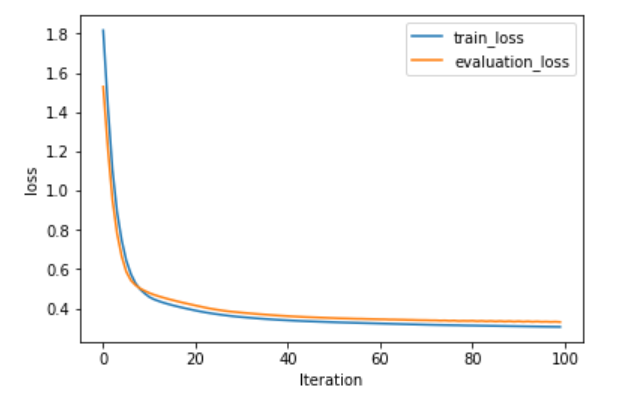
## Assessment Results (based on selected validation):

Min\_loss = 0.3061

## Predicted Results (Best Results):

Min\_loss = 0.3307

## Loss curve:



**12. Results analysis:**

The gradient descent is very smooth, which means that the descent process meets the requirements of this lab. In addition, the descent process of the loss of train and the loss of evaluation are similar, which means that the hyper-parameters chosen are suitable.

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

Linear regression and linear classification have lots things in common. From literal meaning, they both use linear model, which is the simplest model in machine learning. Secondly, they both use gradient descent method, which uses the derivative of loss function to reduce the loss.

However, they do have something different. First, the purposes of linear regression and linear classification are different. Linear regression wants to draw a line to fit the train features and labels, while linear classification tries to use a hyperplane to separate two different kinds of samples. Moreover, their loss functions are different.

**14. Summary:**

This experiment is really interesting! Honestly, I have never try to write codes to implement the basic methods of machine learning. What I always do is writing codes like “from sklearn import svm”. I did not understand what happened in a linear regression model or SVM model. This experiment does help me a lot and now I have a better understand about the meanings and effects of hyper-parameters such as learning rate and max iterations.