華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

College	**Software College**
Subject	**Software Engineering**
Members	庞俊腾
Student ID	201720145174
E-mail	554242688@qq.com
Tutor	吴庆耀
Date submitted	2017. 12 . 15

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

2. Time: 2017.12.9

3. Reporter: 庞俊腾

4. Purposes:

a. Further understand of linear regression and gradient descent.

b. Conduct some experiments under small scale dataset.

c. Realize the process of optimization and adjusting parameters.

5. Data sets and data analysis:

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. Then, it’s divided into training set and validation set.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features. Then, it’s divided into training set and validation set.

6. Experimental steps:

The experimental code and drawing are completed on jupyter.

Linear Regression and Gradient Descent

a. Use load_svmlight_file function in sklearn library to load the experiment data.

b. Divide dataset into training set and validation set using train_test_split function.

c. Initialize linear model parameters. Set all parameter into one.

d. Choose loss function and derivation.

e. Calculate gradient G toward loss function from all samples.

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

g. Update model: .$W_t = W_{t-1} + \eta D$, η is learning rate, set as 0.1 .

h. Get the loss L_{train} under the training set and $L_{validation}$ by validating under validation set.

i. Repeat step e to h for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

Linear Classification and Gradient Descent

a. Use load_svmlight_file function in sklearn library to load the experiment data.

b. Divide dataset into training set and validation set using train_test_split function.

c. Initialize SVM model parameters. Set all parameter into one.

d. Choose loss function and derivation.

e. Calculate gradient G toward loss function from all samples.

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

g. Update model: $W_t = W_{t-1} + \eta D$, η is learning rate, set as 0.1 .

h. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative.

Get the loss L_{train} under the training set and $L_{validation}$ by validating under validation set.

i. Repeat step e to h for several times, and drawing graph of L_{train} as well as $L_{validation}$ with the number of iterations.

7. Code:

Linear Regression:

```
def linearRegression(delta,n,X_train,y_train,X_test,y_test):
    #Initialize linear model parameters. Set all parameter into one
    A = np.ones((X.shape[1],1))
    b = 1

    trainCost = []
    validationCost = []
    for i in range(n):
        trainCost.append(np.sum(np.square(X_train.dot(A) + b - y_train))/
(2*X_train.shape[0]))
        validationCost.append(np.sum(np.square(X_test.dot(A) + b - y_test))/
(2*X_test.shape[0]))
        GA = X_train.T.dot((X_train.dot(A) + b - y_train)) / X_train.shape[0]
        Gb = np.sum((X_train.dot(A) + b - y_train)) / X_train.shape[0]
        DA = -GA
        Db = -Gb
        A = A + delta*DA
        b = b + delta*Db
    return trainCost,validationCost
```

Linear Classification:

```
def SVM(threshold,delta,n,X_train,y_train,X_test,y_test):

    #Initialize linear model parameters. Set all parameter into one
    W = np.ones((X.shape[1],1))
    b = 1
    C = 10

    trainCost = []
    validationCost = []
```

```
accuracy = []
for i in range(n):
    tc = 0
    vc = 0
    GW = 0
    Gb = 0
    accurateCnt = 0
    for j in range(X_train.shape[0]):
        if (1-y_train[j]*(X_train[j].dot(W)+b)) > 0 :
            tc += C*(1-y_train[j]*(X_train[j].dot(W)+b))
            Gb += C*-1*y_train[j]
            GW += C*-1*y_train[j]*X_train[j]

    for j in range(X_test.shape[0]):
        if (X_test[j].dot(W)+b >= threshold) and y_test[j] == 1 :
            accurateCnt +=1
        if (X_test[j].dot(W)+b < threshold) and y_test[j] == -1 :
            accurateCnt +=1
        if (1-y_test[j]*(X_test[j].dot(W)+b)) > 0 :
            vc += C*(1-y_test[j]*(X_test[j].dot(W)+b))

    trainCost.append(np.sum(tc/X_train.shape[0] + 0.5*np.sum(W.T.dot(W))))
    validationCost.append(np.sum(vc/X_test.shape[0] + 0.5*np.sum(W.T.dot(W))))
    accuracy.append(accurateCnt/X_test.shape[0])
    GW = GW.T/X_train.shape[0] + W
    Gb = Gb/X_train.shape[0]
    DW = -GW
    Db = -Gb

    W = W + delta*DW
    b = b + delta*Db
return trainCost,validationCost,accuracy
```

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

Linear Regression: hold-out

Linear Classification: hold-out

9. The initialization method of model parameters:

Linear Regression: set both parameters as one.

Linear Classification: set both parameters as one.

10. The selected loss function and its derivatives:

Linear Regression:

$$LossFunction = \frac{1}{2m}\sum_{i=1}^{m}(X_i * A + b - y_i)^2$$

$$G_A = \frac{1}{m}\sum_{i=1}^{m}(X_i * A + b - y_i) * X_i$$

$$G_b = \frac{1}{m}\sum_{i=1}^{m}(X_i * A + b - y_i)$$

Linear Classification:

SVM

$$LossFunction = \frac{\|w\|^2}{2} + \frac{C}{m}\sum_{i=1}^{m} max(0, 1 - y_i(W^T x_i + b)$$

$$G_W = W + \frac{C}{m}\sum_{i=1}^{m} -y_i * x_i \text{ ,if } 1 - y_i(W^T x_i + b) > 0$$

$$G_b = \frac{C}{m}\sum_{i=1}^{m} -y_i \text{ ,if } 1 - y_i(W^T x_i + b) > 0$$

11. Experimental results and curve:

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

Linear Regression:

η = 0.1, epoch = 50

Linear Classification:

η = 0.01, epoch = 50

Assessment Results (based on selected validation):

Linear Regression: train loss may be smaller than validation loss.

Linear Classification: train loss may be smaller than validation loss.

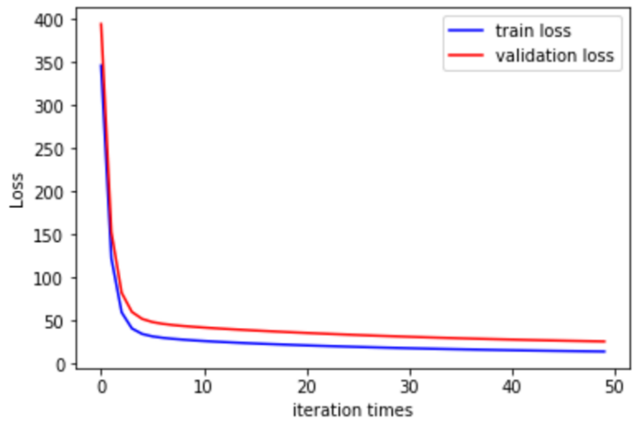
Predicted Results (Best Results):

Linear Regression: validation loss can be smaller than 10.

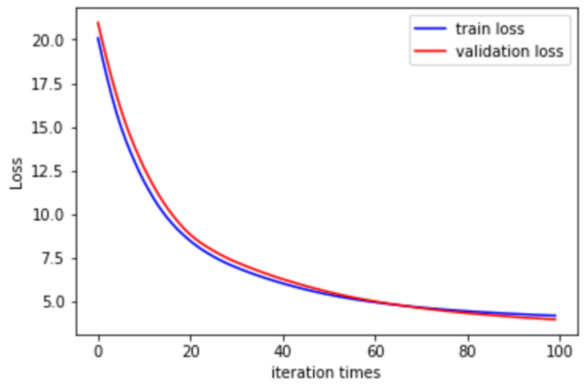
Linear Classification: validation loss can be smaller than 2.

Loss curve:

Linear Regression:



Linear Classification:



12. Results analysis:

Linear Regression:

As iteration goes on, both the train loss and the validation loss are becoming smaller and smaller, but none of them can reach zero because the data isn’t mapped on a line.

The train loss would be smaller than the validation loss, because the training is based on the training data, and it will make sure the model suits the training data well.

Linear Classification:

As iteration goes on, both the train loss and the validation loss are becoming smaller and smaller. The train loss would be smaller than the validation loss, because the training is based on the training data, and it will make sure the model suits the training data well.

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

Similarities: Both linear regression and linear classification are training the model parameters so that the model can fit the data well.

Differences: Linear regression fit a x of small domain of x to a y. While linear classification fit a larger domain to a y.

14. Summary:

Through this experiment, I learn the basic way to train a model. The gradient descent is a good method in such domain. Also, I have the opportunity to realize the linear regression model and have a deeper understanding in deep learning.