

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**   **Zanjie Fang**

**Student ID 201530611456**

**E-mail fangzanjie@163.com**

**Tutor**   **Mingkui Tan**

**Date submitted** **2017.12.07**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

**2. Time:**

2017.12.07

**3. Reporter:**

Zanjie Fang

**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.

**5. 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.

**6. Experimental steps:**

**(1)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: = +D. is learning rate, a hyper-parameter that we can adjust.

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

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

**(2)** **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: = +D. 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 under the training set and by validating under validation set.

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

**7. Code:**

1. Linear Regression and Gradient Descent

|  |
| --- |
|  |

|  |
| --- |
| # 计算loss  def validation(X\_test, y\_test, w, b):  loss = 0  for i in range(X\_test.shape[0]):  y\_ = np.dot(X\_test[i].data, w) + b  loss += (y\_test[i] - y\_) \* (y\_test[i] - y\_)\* 0.5  return loss/X\_test.shape[0]  for num in range(epoch):  # loss\_function 对w，b的偏导  G\_w = 0  G\_b = 0    # sigma求和  for i in range(X\_train.shape[0]):  y\_ = np.dot(X\_train[i].data, w) + b  loss += 1/2 \* (y\_train[i] - y\_) \* (y\_train[i] - y\_)  G\_w += (y\_train[i] - y\_) \* (-X\_train[i].data)  G\_b += y\_ - y\_train[i]    # 求偏导的平均数  loss /= X\_train.shape[0]  G\_w /= X\_train.shape[0]  G\_b /= X\_train.shape[0]    # 更新参数  w = w - learning\_rate \* G\_w  b = b - learning\_rate \* G\_b    # 计算验证集loss  val\_loss = validation(X\_test, y\_test, w, b)    # 将loss加入列表  losses.append(loss[0])  val\_losses.append(val\_loss[0]) |

1. Linear Classification and Gradient Descent

|  |
| --- |
| # 计算HingeLoss  def calHingeLoss(Yi, Xi, W):  temp = Yi\*np.dot(W.T, Xi.T)  result = 1 - (Yi\*np.dot(W.T, Xi.T))[0][0]  if result > 0:  hingeLoss = result  else:  hingeLoss = 0  return hingeLoss    # 计算梯度  def calDirection(X\_train, y\_train,W):  temp = np.zeros((X\_train.shape[1], 1))  for index in range(len(X\_train)):  Yi = y\_train[index][0]  Xi = X\_train[index].reshape(1, X.shape[1])  hingeLoss = calHingeLoss(Yi, Xi, W)  if hingeLoss > 0:  temp = temp + Yi \* Xi.T  else:  temp = temp+np.zeros((X.shape[1], 1))  direction = W - c \* temp  return direction  # 计算loss  def calLoss(W, X\_t, y\_t):  temp = 0  for index in range(len(X\_t)):  Yi = y\_t[index][0]  Xi = X\_t[index].reshape(1, X\_t.shape[1])  temp += calHingeLoss(Yi, Xi, W)  temp = c \* temp  Loss = 1/2 \* np.dot(W.T, W) + temp  return Loss/X\_t.shape[0]      for i in range(epoch):  #计算训练集的梯度  direction = calDirection(X\_train, y\_train, W)  # 更新参数  W = W - learning\_rate\*direction    # 计算训练集的loss  Ltrain = calLoss(W, X\_train, y\_train)  lossTrain.append(Ltrain[0][0]) # 将训练集的loss加入列表  error = 0  for index in range(len(X\_train)):  Ypredict = np.dot(X\_train[index], W)  if Ypredict > 0:  Ypredict = 1  else:  Ypredict = -1  if Ypredict != y\_train[index][0]:  error += 1  errorRate = error/float(len(X\_train))  lossTrainClassification.append(errorRate)    # 计算测试集的loss  Ltest = calLoss(W, X\_test,y\_test)  lossValidation.append(Ltest[0][0])# 将测试集的loss加入列表  error = 0  for index in range(len(X\_test)):  Ypredict = np.dot(X\_test[index], W)  if Ypredict > 0:  Ypredict = 1  else:  Ypredict = -1  if Ypredict != y\_test[index][0]:  error += 1  errorRate = error/float(len(X\_test))  lossValidationClassification.append(errorRate) |

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

1. Linear Regression and Gradient Descent

Hold-out.

1. Linear Classification and Gradient Descent

Hold-out.

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

1. Linear Regression and Gradient Descent

Initialization of nomal distribution.

(2) Linear Classification and Gradient Descent

Initialization of all zero.

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

1. Linear Regression and Gradient Descent

Loss Function:

Derivative:

1. Linear Classification and Gradient Descent

Loss Function:

Derivative:

The derivative of loss:

**11. Experimental results and curve:**

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

(1)Linear Regression and Gradient Descent

learning\_rate = 0.002, epoch = 1000.

(2)Linear Classification and Gradient Descent

learning\_rate = 0.00006, epoch = 200, LAMBDA = 1, c = 1.

## Assessment Results (based on selected validation):

1. Linear Regression and Gradient Descent

In the iteration time, the min loss of the training set is about 24.6996 and the min loss of the validation set is 24.0038.

(2) Linear Classification and Gradient Descent

In the iteration time, the min loss of the training set is about 0.2923 and the min loss of the validation set is 0.3085.

## Predicted Results (Best Results):

(1)Linear Regression and Gradient Descent

The result is about 24.0038.

1. Linear Classification and Gradient Descent

The result is about 0.3085

## Loss curve:

(1)Linear Regression and Gradient Descent

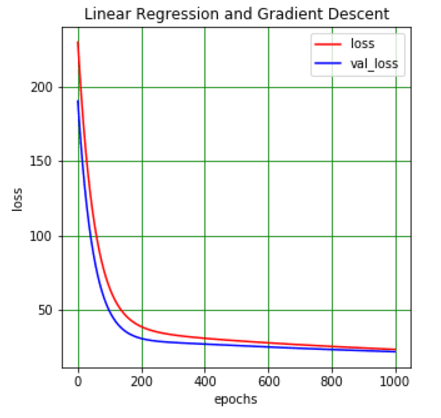


Fig 1

(2) Linear Classification and Gradient Descent

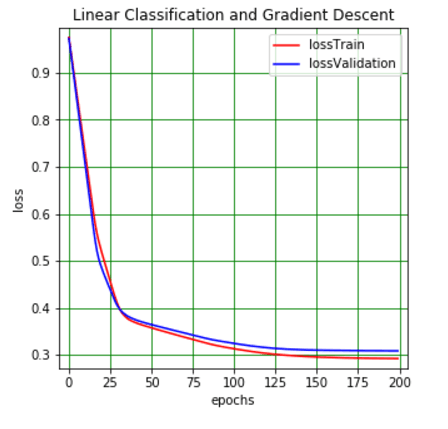


Fig 2

**12. Results analysis:**

For the Linear Classification and Gradient Descent, the errorRate is shown below:

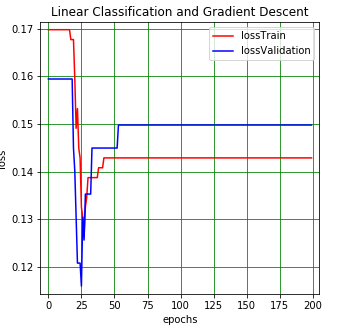


Fig 3

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

Both of them use the gradient to find the best arguments.

**14. Summary:**

From this experiment, I know how to optimization the arguments to find the best one, and so to find the min loss.