华南理工大学

South China University of Technology

The Experiment Report of

Machine Learning

College Software College

Subject Software Engineering

Members 孙杏杏

Student ID 201721045350

E-mail qqcnout@163.com

Tutor QingYaoWu

Date submitted 2017. 12 .15

1. Topic:

Linear Regression, Linear Classification and Gradient Descent

2. Time: 2017-12-15

3. Reporter: SunXingxing

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:

6.1 Linear Regression and Gradient Descent

1. Load the experiment data and divide the dataset into training set and validation set.
2. Initialize linear model parameters. Set all parameter into zero, initialize it randomly or with normal distribution.
3. Define the loss function of the linear regression to be Least squared loss, and the loss function of the linear classification to be Hingle loss.
4. Compute the gradient of the loss function with respect to the weight W and bias b.
5. Update the parameters W and b.
6. Repeat above steps for several times until convergence.

6.2 Linear Classification and Gradient Descent

7. Load the experiment data.
8. Divide dataset into training set and validation set.
9. Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
10. Choose loss function and derivation: Find more detail in PPT.
11. Calculate gradient toward loss function from all samples.Denote the

opposite direction of gradient as .

12. Update model: . is learning rate, a hyper-parameter that we can adjust.
13. 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 trainin set and by validating under validation set.
14. Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

7. Code:

7.1 Linear Regression and Gradient Descent

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_svmlight_file
from sklearn.externals.joblib import Memory
from sklearn.model_selection import train_test_split

mem = Memory("./mycache")
#get dataset
@mem.cache
def get_data():
    data =
load_svmlight_file("D:\Task\MachineLearning\Test\LR\LR__Housing\data\housing_scale"
    return data[0], data[1]


#divide the dataset into training set and validation set, the random state is 20
def spilt_data(X, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=20)
    return x_train, x_test, y_train, y_test


#compute cost function
def computeCost(x, y, theta):
    loss =  (1 / (2 * x.shape[0])) * ((np.dot(x, theta) - y).transpose()*(np.dot(x, theta
- y))
    return np.sum(loss)

X, y = get_data()
X_train, x_test, y_train, y_test = spilt_data(X, y)

X_train =  X_train.todense()
```

```
x_test = x_test.todense()
y_train = y_train.reshape(y_train.shape[0], 1)
y_test = y_test.reshape(y_test.shape[0], 1)

#initialize the parameters
theta_array = np.zeros((X_train.shape[1], 1))

#set the learning rate and iterate number
leraning_rate = 0.01
iterate_number = 200

loss_train = []
loss_valid = []


for i in range(iterate_number):
    gradient = (1 / (X_train.shape[0])) * (X_train.transpose() * (X_train * theta_arra
- y_train))
    #print(gradient)
    theta_array = theta_array - leraning_rate * gradient
    loss_train.append(computeCost(X_train, y_train, theta_array))
    loss_valid.append(computeCost(x_test, y_test, theta_array))

#print("loss_train", loss_train)
#print("loss_valid", loss_valid)

#ploting the loss value
plt.plot(loss_train, color="r", label="Loss_train")
plt.plot(loss_valid, color="g",label="Loss_valid")
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.title("Linear Regression---Housing")
plt.legend()
plt.show()
```

7.2 Linear Classification and Gradient Descent

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
```

```
def get_data(file):
    data = load_svmlight_file(file)
    return data[0], data[1]

def compute_loss(X, W, y, C = 1):
    L2 =  0.5 * np.dot(W.T, W)
    prediction_y = np.dot(X, W)
    #print("prediction_y", prediction_y)
    diff = np.ones(y.shape[0]) - y * prediction_y
    diff[diff < 0] = 0
    #print("diff", diff)
    hingeloss = C *(np.sum(diff)) / X.shape[0]
    loss = hingeloss + L2
    return loss

def get_gradient(X, W, y, C = 1):
    prediction_y = np.dot(X, W)
    diff = np.ones(y.shape[0]) - y * prediction_y
    #print("diff", diff.shape)
    #print("y", y.shape)
    y_copy = y.copy()
    y_copy[diff <= 0] = 0
    gradient = W - C * np.dot(y_copy, X) / X.shape[0]
    #print("Gradient", gradient)
    return gradient

X, y = get_data("D:/Task/MachineLearning/Test/LR/线性分类
__australian/data/australian_scale")

X = X.toarray()
#add anothre column for x
column = np.ones((X.shape[0]))
X = np.column_stack((X,column))
X_train, x_valid, y_train, y_valid = train_test_split(X, y, test_size =
0.30, random_state = 42)

#print("X_train", X_train.shape)
#print("y_train", y_train.shape)

W = np.random.normal(size=X_train.shape[1])

#set the learning rate and iterate number
```

```
leraning_rate = 0.01
iterate_number = 1000

loss_train = []
loss_valid = []

for i in range(iterate_number):
    gradient = get_gradient(X_train, W, y_train)
    #print(gradient)
    W = W - leraning_rate * gradient
    loss_train.append(compute_loss(X_train, W, y_train))
    loss_valid.append(compute_loss(x_valid, W, y_valid))

#ploting the loss value
plt.plot(loss_train, color="r", label="Loss_train")
plt.plot(loss_valid, color="g",label="Loss_valid")
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.title("Linear SVM---Australian")
plt.legend()
plt.show()
```

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

1.first we divided the dataset into training set and validation set
2. later, we used cross-validation method

9. The initialization method of model parameters:

1.in linear regression we set all parameter into zero at first
2.in linear classification , we initialized parameter randomly.

10. The selected loss function and its derivatives:

1. Linear regression
    1) Loss function

$$L = \frac{1}{2n}\sum_{i=1}^{n}(y_i - W^T x_i)^2$$

2) Gradient with the respect of the W

$$G = \frac{1}{n}\sum_{i=1}^{n}(-x_i) * (y_i - W^T x_i)$$

2. Linear classification

1) Loss function

$$L = \frac{\lambda}{2}||W||^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i(W^T x_i + b))$$

2) Gradient with the respect of the W

$$G_W = \begin{cases} \lambda w & y_i(W^T x_i + b) \geq 1 \\ \lambda W + \frac{1}{n}\sum_{i=1}^{n} - y_i x_i & y_i(W^T x_i + b) < 1 \end{cases}$$

11. Experimental results and curve:

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

Learning-rate and regularization parameter both try the value in [10-4, 10-3, 10-2, 10-1, 1, 10], and the result suggests that learning_rate = 10-2 and epoch=200 is the best choice

Assessment Results (based on selected validation):

1.linear regression:

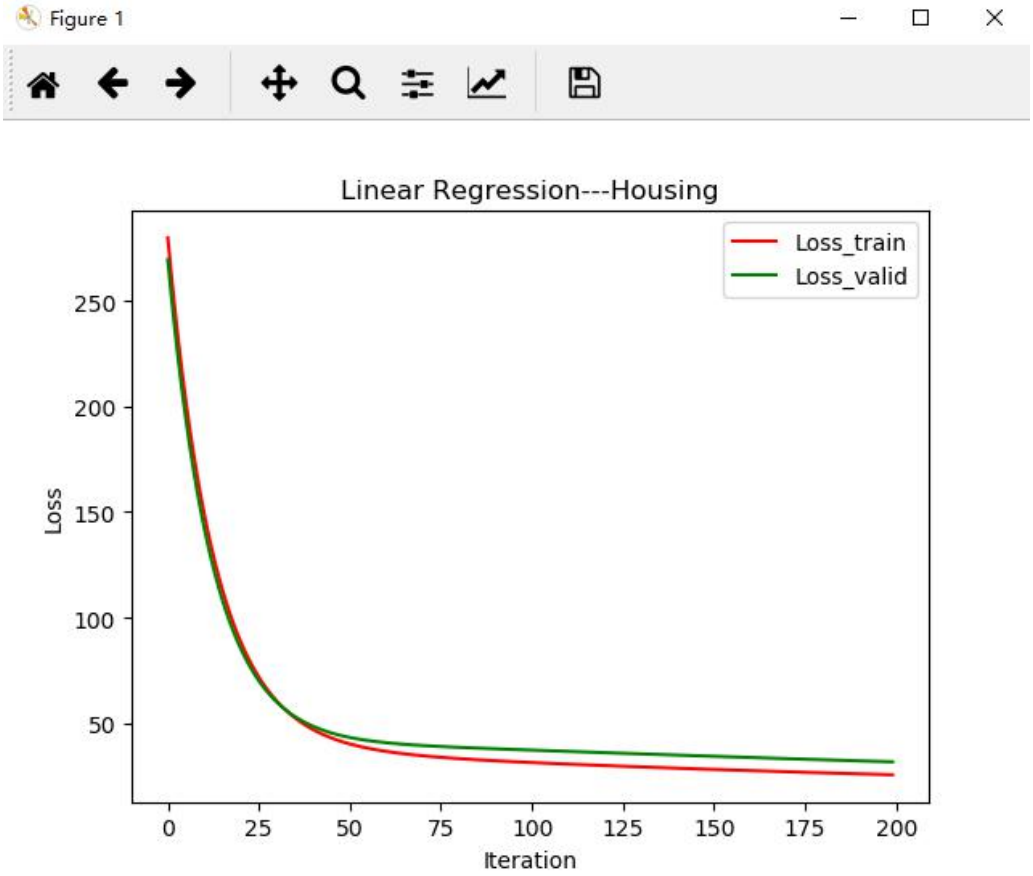
avg_train_loss:53.42402454058121,
avg_val_loss:59.55045672462349

2.linear classification:

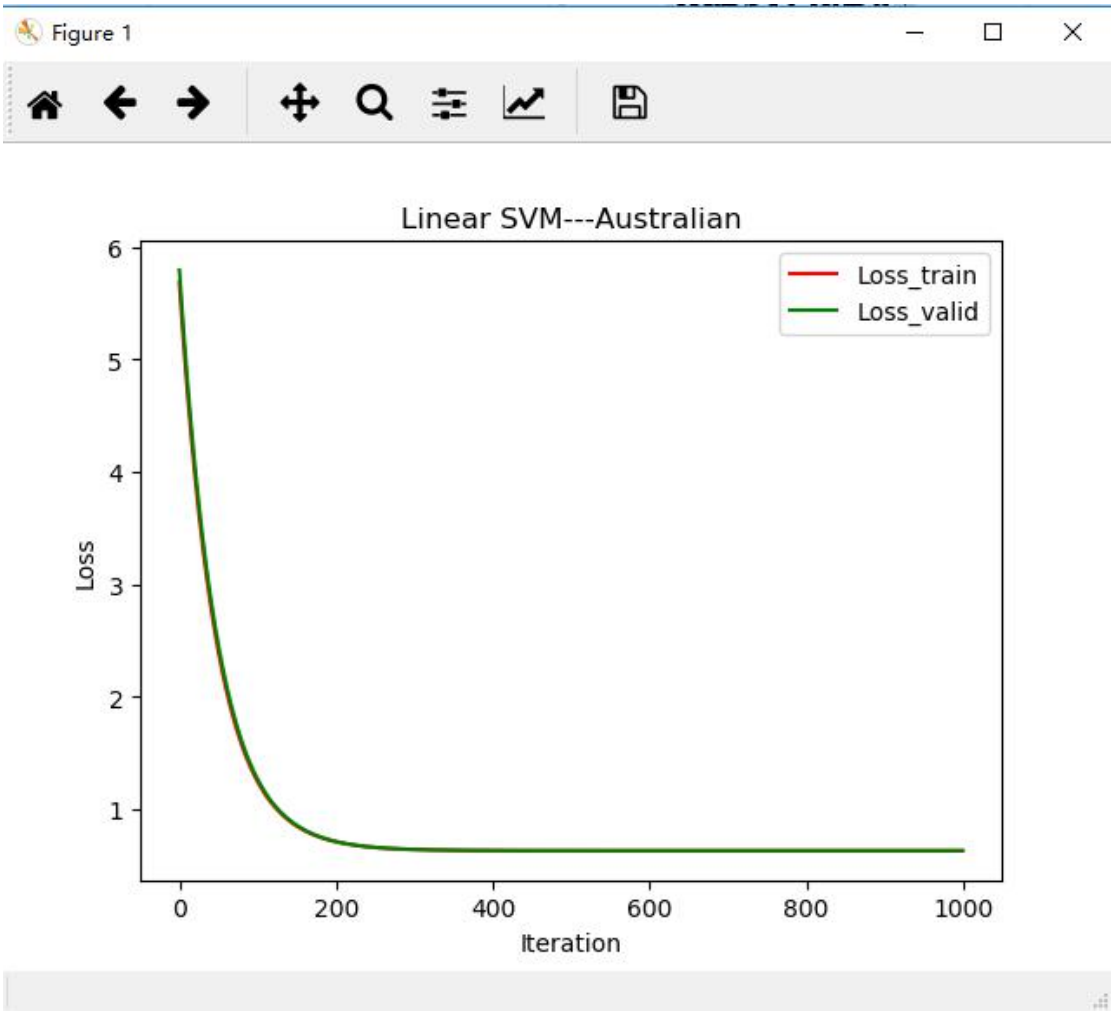
avg_train_loss:0.613414613342641,
avg_val_loss:0.6618234029995596

Predicted Results (Best Results):

Loss curve:



Linear Regression



Linear SVM

12. Results analysis:

1 .if the learning rate is too small, we may have a slow convergence problem.

2. if the learning rate is too large, J(theta) may not decrease on every iteration and it may not even converge. In some cases if the learning rate is too large, slow convergence is also possible
3. if we have high variance, it is high possible get overfitting

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

1. In linear regression, the outcome is continuous, but in linear classification the outcome has only a limited number of possible

2. Linear classification output as probabilities, but linear regression is a prediction

3. Linear regression gives an equation which is of the form Y = mX + C equation with degree 1.

However, linear classification gives an equation which is of the form Y = e^X/1 + e^-X

14. Summary:

In this experiment, I understand the linear regression and gradient descent method. Trying solve the regression and classification problem, I figure the difference between them.Besides, I also learn how to realize the process of optimization and adjusting parameters.