The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
王跃

Supervisor:
Mingkui Tan

Student ID：
201530612941

Grade:
Undergraduate

December 9, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—

I. INTRODUCTION

In order to compare and understand the difference between gradient descent and stochastic gradient d descent, compare and understand the differences and relationships between Logistic regression and linear classification and further understand the principles of SVM and practice on larger data.We do this experiments about linear regression,linear classification and gradient descent.

II. METHODS AND THEORY

The experimental data set used a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

The amplify steps of linear regression and gradient descent are as fellow:

1. Load the training set and validation set.
2. Initialize logistic regression model parameters, you can consider initializing zeros, random numbers or normal distribution.
3. Select the appropriate loss function and calculate its derivation.
4. Calculate gradient G toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).
6. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss L_NAG，L_RMSProp，L_AdaDelta and L_Adam.
7. Repeated step 4 to 6 for several times, and drawing graph of L_NAG，L_RMSProp，L_AdaDelta and L_Adam with the number of iterations.

The amplify steps of linear classification and gradient descent are as fellow:

1. Load the training set and validation set.
2. Initialize SVM model parameters, you can consider initializing zeros, random numbers or normal distribution.
3. Select the appropriate loss function and calculate its derivation.
4. Calculate gradient toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).
8. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss L_NAG，L_RMSProp，L_AdaDelta and L_Adam.
9. Repeated step 4 to 6 for several times, and drawing graph of L_NAG，L_RMSProp，L_AdaDelta and L_Adam with the number of iterations.

The selected loss function and its derivatives of linear regression and gradient descent:

$$loss = -1/n*\left(\sum_{i=1}^{n} yi*log(1/(1+e^{-w.T*x})) + (1-yi)*log(1-1/(1+e^{-w.T*x}))\right)$$

$$gradient = \left(1 \div \left(1+e^{w.T*x}\right) - yi\right)*x$$

The selected loss function and its derivatives of linear classification and gradient descent :

$$loss = 1/n*((\|w\|^2)/2 + C\sum_{i=1}^{n} max(0, 1-yi*(w.T*xi+b)))$$

If $(1-yi*(w.T*xi+b)) > 0$

$$gradient = w - yi*xi$$

If $(1-yi*(w.T*xi+b)) \le 0$

$$gradient = w$$

According to the above method and theory, we conducted the following experiment

III. EXPERIMENT

A. Hyper-parameter selection:

Logistic regression: $\eta = 0.01$ $\beta = 0.9$ $\gamma = 0.95$
Linear classification: $\eta = 0.01$ $\beta = 0.9$ $\gamma = 0.95$

B. Predicted Results (Best Results):

The forecast results are shown graphically.
The vital code of linear regression and gradient descent:
The method of computing the gradient:

```
def gradient(w,x,y,rand_list):
    grad=np.zeros((124,))
    for i in rand_list:
        grad
+=(1/(1+math.exp(-np.dot((w.T),(x[i].T))))-y[i])*(x
[i].T)
    return grad*(1/100)
```

The method of computing the loss :

```
def get_loss(w,x,y):
    los = 0
    for i in range(x.shape[0]):
        h = 1/(1+math.exp((-1)*np.dot((w.T),(x[i].T))))
        l =
(-1)*(y[i]*math.log(h)+(1-y[i])*math.log(1-h))
        los += l
    return los/(x.shape[0])
```

The general gradient descent :

```
 grad = gradient(w1,X_train,y_train,rand_list)
    w1 = w1 - 0.001*grad
    loss.append(get_loss(w1,X_train,y_train))
    loss_test.append(get_loss(w1,X_test,y_test))
```

The NAG optimization:

```
v =
0.9*v+0.001*(gradient(w2-0.9*v,X_train,y_train,ra
nd_list))
    w2 = w2 - v
    loss_NAG.append(get_loss(w2,X_train,y_train))

loss_test_NAG.append(get_loss(w2,X_test,y_test))
```

The RMSProp optimization:

```
g=gradient(w3,X_train,y_train,rand_list)
    G = 0.9*G+(1-0.9)*(g*g)
    G_e= G + e
    for i in range(G.shape[0]):
        G_e[i]=0.001/math.sqrt(G_e[i])
    w3 = w3-G_e*(g)

loss_RMSProp.append(get_loss(w3,X_train,y_train
))

loss_test_RMSProp.append(get_loss(w3,X_test,y_t
est))
```

The AdaDelta optimization :

```
g=gradient(w4,X_train,y_train,rand_list)
    G=0.95*G+(1-0.95)*(g*g)
    t_e=t+e
    G_e=G+e
    for i in range(G.shape[0]):

G_e[i]=-((math.sqrt(t_e[i]))/(math.sqrt(G_e[i])))
    wt=G_e*g
    w4=w4+wt
    t=0.99*t+(1-0.99)*(wt*wt)

loss_AdaDelta.append(get_loss(w4,X_train,y_train)
)

loss_test_AdaDelta.append(get_loss(w4,X_test,y_te
st))
```

The Adam optimization :

```
g=gradient(w5,X_train,y_train,rand_list)
    m=0.9*m+(1-0.9)*g
    G=0.999*G+(1-0.999)*(g*g)
    G_e=G+e
    for i in range(G.shape[0]):
        G_e[i]=m[i]/(math.sqrt(G_e[i]))
    Alpha=
0.001*((math.sqrt(1-math.pow(0.999,k)))/(math.sqrt
(1-math.pow(0.9,k))))
    w5=w5-Alpha*G_e
    loss_Adam.append(get_loss(w1,X_train,y_train))

loss_test_Adam.append(get_loss(w1,X_test,y_test))
```

The vital code of linear classification and gradient descent:
The method of computing the gradient:

```
def gradient(w,x,y,rand_list):
    grad = np.zeros((124,))
    for i in rand_list:
```

```
        if (1-y[i]*np.dot(w.T,x[i].T))>0 :
            grad += w - y[i]*(x[i].T)
        if (1-y[i]*np.dot(w.T,x[i].T))<=0 :
            grad += w
    grad = (1/100)*grad
    return grad
```

The method of computing the loss :

```
def get_loss(w,x,y):
    los = 0
    for i in range(x.shape[0]):
        if (1-y[i]*np.dot(w.T,x[i].T))>0 :
            los += 1-y[i]*np.dot((w.T),(x[i].T))
        if (1-y[i]*np.dot(w.T,x[i].T))<=0 :
            los += 0
    return (1/x.shape[0])*(0.5*np.dot((w.T),w)+los)
```

The general gradient descent :

```
grad = gradient(w,X_train,y_train,rand_list)
    w = w - 0.05*grad
    loss.append(get_loss(w,X_train,y_train))
    loss_test.append(get_loss(w,X_test,y_test))
```

The NAG optimization:

```
v=0.9*v+0.01*gradient(w2,X_train,y_train,rand_lis
t)
    w2=w2-v
    loss_NAG.append(get_loss(w2,X_train,y_train))

loss_test_NAG.append(get_loss(w2,X_test,y_test))
```

The RMSProp optimization:

```
g=gradient(w3,X_train,y_train,rand_list)
    G = 0.9*G+(1-0.9)*(g*g)
    G_e= G + e
    for i in range(G.shape[0]):
        G_e[i]=0.001/math.sqrt(G_e[i])
    w3 = w3-G_e*(g)

loss_RMSProp.append(get_loss(w3,X_train,y_train
))

loss_test_RMSProp.append(get_loss(w3,X_test,y_t
est))
```

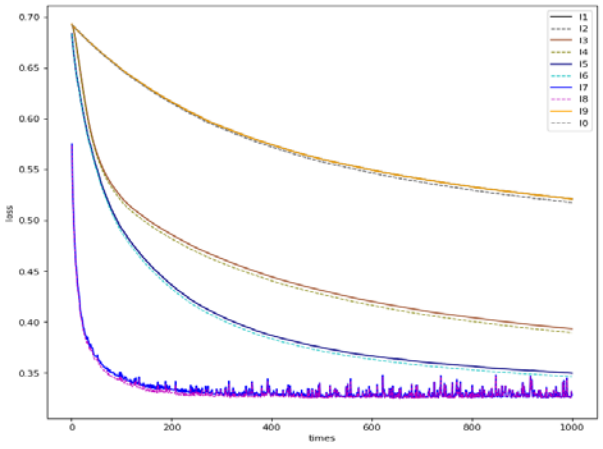
The AdaDelta optimization:

```
g=gradient(w4,X_train,y_train,rand_list)
    G=0.95*G+(1-0.95)*(g*g)
    t_e=t+e
    G_e=G+e
    for i in range(G.shape[0]):

G_e[i]=(-1)*((math.sqrt(t_e[i]))/(math.sqrt(G_e[i]))
)
    wt=G_e*g
    w4=w4+wt
    t=0.99*t+(1-0.99)*(wt*wt)

loss_AdaDelta.append(get_loss(w4,X_train,y_train)
)

loss_test_AdaDelta.append(get_loss(w4,X_test,y_te
st))
```

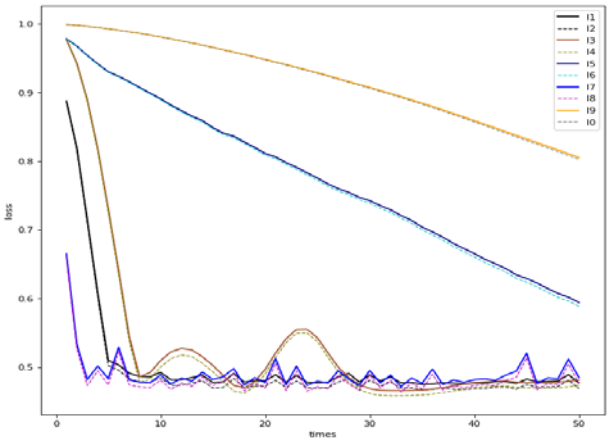
The Adam optimization:

```
g=gradient(w5,X_train,y_train,rand_list)
    m=0.8*m+(1-0.9)*g
    G=0.99*G+(1-0.999)*(g*g)
    G_e=G+e
    for i in range(G.shape[0]):
        G_e[i]=m[i]/(math.sqrt(G_e[i]))
    Alpha=
0.001*((math.sqrt(1-math.pow(0.999,k)))/(math.sqrt
(1-math.pow(0.9,k))))
    w5=w5-Alpha*G_e
    loss_Adam.append(get_loss(w5,X_train,y_train))

loss_test_Adam.append(get_loss(w5,X_test,y_test))
```

The loss curve of Logistic regression:



The loss curve of Logistic classification:



IV. CONCLUSION

Results analysis:
Logistic regression:
Through the adjustment of parameters can get a better regression results.
Linear classification:
Through the adjustment of multiple hyper-parameters, a better SVM model can be trained on the training set.

Similarities and differences between logistic regression and linear classification：
Same point:
1, LR and SVM are classification algorithm.
2, If you do not consider the kernel function, LR and SVM are linear classification algorithm, that is, the classification decision-making surface is linear.
3, LR and SVM are supervised learning algorithm.
difference:
1, in essence, its loss function is different.
2, support vector machine only consider the local boundary line near the point, while the logical regression to consider the overall

Summary:
Through the study of this experiment, further understand the nature and realization of logistic regression and linear classification. Through the learning of gradient descent method, we further understand the important content of gradient learning. And in the course of this experiment, the gradient descent is better understood through the practice of different optimization methods.