华南理工大学

South China University of Technology

The Experiment Report of
Machine Learning

College **Software College**

Subject **Software Engineering**

Members 阮子琦

Student ID 201530612668

E-mail 609387481@qq.com

Tutor Prof.Mingkui Tan

Date submitted 2017. 12 . 14

1. Topic: Comparison of Various Stochastic Gradient Descent Methods for Solving Classification Problems

2. Time: 2017.12.8

3. Reporter:阮子琦

4. Purposes:

1) Compare and understand the difference between gradient descent and stochastic gradient descent.
2) Compare and understand the differences and relationships between Logistic regression and linear classification.
3) Further understand the principles of SVM and practice on larger data.

5. Data sets and data analysis:

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

6. Experimental steps:

1）Logistic Regression and Stochastic Gradient Descent:

a) Load the training set and validation set.
b) Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.

c) Select the loss function and calculate its derivation, find more detail in PPT.
d) Calculate gradient toward loss function from partial samples.
e) Update model parameters using different optimized methods(NAG， RMSProp， AdaDelta and Adam).
f) 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}.
g) Repeate 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.

2） Linear Classification and Stochastic Gradient Descent:

a) Load the training set and validation set.
b) Initalize SVM model parameters, you can consider initalizing zeros, random numbers or normal distribution.
c) Select the loss function and calculate its derivation, find more detail in PPT.

d) Calculate gradient toward loss function from partial samples.

e) Update model parameters using different optimized methods(NAG， RMSProp， AdaDelta and Adam).

f) 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} Repeate 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.

7. Code:

There are the main code for each method in each experiment:

1) Logistic Regression and Stochastic Gradient Descent:

a) NAG:

```
train_x_part1, train_x_part2,train_y_part1, train_y_part2 =
train_test_split(x_train_array, train_y, test_size=0.1)
v_pre = v
descent = compute_descent(train_x_part2,train_y_part2,theta)
v = 0.9 * v - rate * descent
theta = theta - 0.9 * v_pre + ( 1 + 0.9 ) * v
loss_NAG.append(compute_loss())
```

b) RMSProp:

```
train_x_part1, train_x_part2,train_y_part1, train_y_part2 =
train_test_split(x_train_array, train_y, test_size=0.02)
descent = compute_descent(train_x_part2,train_y_part2,theta)
g = 0.9 * g + 0.1 * descent * descent
theta = theta - (rate / ( np.power(g + np.exp(-8),1/2))) * descent
loss_RMSProp.append(compute_loss())
```

c) AdaDelta:

```
train_x_part1, train_x_part2,train_y_part1, train_y_part2 =
train_test_split(x_train_array, train_y, test_size=0.02)
descent = compute_descent(train_x_part2,train_y_part2,theta)
g = 0.95 * g + 0.05 * descent * descent
change_theta = - (np.power(t + np.exp(-8) ,1/2 ) / ( np.power(g +
np.exp(-8),1/2))) * descent
theta = theta + change_theta
t = 0.95 * t + 0.05 * change_theta * change_theta
loss_AdaDelta.append(compute_loss())
```

d) Adam:

```
train_x_part1, train_x_part2,train_y_part1, train_y_part2 =
train_test_split(x_train_array, train_y, test_size=0.01)
descent = compute_descent(train_x_part2,train_y_part2,theta)
m = 0.9 * m + 0.1 * descent
```

g = 0.999 * g + 0.001 * descent * descent

learning_rate = 0.001 * np.power(1-0.999**(i+1),1/2) / (1 - 0.9**(i+1))

theta = theta - learning_rate * m / np.power(g + np.exp(-8),1/2)

loss_Adam.append(compute_loss())

2) Linear Classification and Stochastic Gradient Descent:

In this experiment, we compute the w and b respectively because of the difference of the result of descent compute.

a) NAG:

train_x_part1, train_x_part2,train_y_part1, train_y_part2 = train_test_split(x_train_array, train_y, test_size=0.2)

theta_term = theta

beta_term = beta

descent_theta = compute_descent_theta(train_x_part2,train_y_part2,theta_term,beta_term)

descent_beta = compute_descent_beta(train_x_part2,train_y_part2,theta_term,beta_term)

v_theta = 0.9 * v_theta - rate * descent_theta

v_beta = 0.9 * v_beta - rate * descent_beta

theta = theta - 0.9 * theta_term + (1 + 0.9) * v_theta

beta = beta - 0.9 * beta_term + (1 + 0.9) * v_beta

loss_NAG.append(compute_loss(theta,beta))

b) RMSProp:

train_x_part1, train_x_part2,train_y_part1, train_y_part2 = train_test_split(x_train_array, train_y, test_size=0.2)

theta_term = theta

beta_term = beta

descent_theta = compute_descent_theta(train_x_part2,train_y_part2,theta_term,beta_term)

descent_beta = compute_descent_beta(train_x_part2,train_y_part2,theta_term,beta_term)

G_theta = 0.9 * G_theta + 0.1 * descent_theta * descent_theta

G_beta = 0.9 * G_beta + 0.1 * descent_beta * descent_beta

theta = theta - (rate / np.power(G_theta + np.exp(-8) ,1/2)) * descent_theta

beta = beta - (rate / np.power(G_beta + np.exp(-8) ,1/2)) * descent_beta

loss_RMSProp.append(compute_loss(theta,beta))

c) AdaDelta:

train_x_part1, train_x_part2,train_y_part1, train_y_part2 =

```
train_test_split(x_train_array, train_y, test_size=0.05)
            theta_term = theta
            beta_term = beta
            descent_theta =
compute_descent_theta(train_x_part2,train_y_part2,theta_term,beta_ter
m)
            descent_beta =
compute_descent_beta(train_x_part2,train_y_part2,theta_term,beta_ter
m)
            G_theta = gama * G_theta + (1-gama) * descent_theta *
descent_theta
            G_beta = gama * G_beta + (1-gama) * descent_beta *
descent_beta
            delta1_theta = - (np.sqrt(delta2_theta + np.exp(-8) ) /
np.sqrt(G_theta + np.exp(-8))) * descent_theta
            delta1_beta = - (np.sqrt(delta2_beta + np.exp(-8) ) /
np.sqrt(G_beta + np.exp(-8))) * descent_beta
            theta = theta + delta1_theta
            beta = beta + delta1_beta
            delta2_theta = gama * delta2_theta + (1-gama) * delta1_theta *
delta1_theta
            delta2_beta = gama * delta2_beta + (1-gama) * delta1_beta *
```

delta1_beta

loss_AdaDelta.append(compute_loss(theta,beta))

d) Adam:

train_x_part1, train_x_part2,train_y_part1, train_y_part2 = train_test_split(x_train_array, train_y, test_size=0.2)

theta_term = theta

beta_term = beta

descent_theta = compute_descent_theta(train_x_part2,train_y_part2,theta_term,beta_term)

descent_beta = compute_descent_beta(train_x_part2,train_y_part2,theta_term,beta_term)

m_theta = 0.9 * m_theta + 0.1 * descent_theta

m_beta = 0.9 * m_beta + 0.1 * descent_beta

G_theta = 0.999 * G_theta + 0.001 * descent_theta * descent_theta

G_beta = 0.999 * G_beta + 0.001 * descent_beta * descent_beta

alpha = rate * np.power(1 - np.power(0.999,i+1),1/2) / (1 - np.power(0.9,i+1))

theta = theta - alpha * m_theta / np.power(G_theta +

```
np.exp(-8),1/2)
            beta = beta - alpha * m_beta / np.power(G_beta +
np.exp(-8),1/2)
            loss = compute_loss(theta,beta)
            loss_Adam.append(loss)
```

8. The initialization method of model parameters:

For the experiment of Logistic Regression and Stochastic Gradient Descent, we set the vector w to zero vector and the learning rate to 0.001.

For the experiment of Linear Classification and Stochastic Gradient Descent, we set the vector w to zero vector and C to 0.1

9. The selected loss function and its derivatives:

For the experiment of Logistic Regression and Stochastic Gradient Descent, the loss function we use is as followed:

$$-\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log h_w(x_i) + (1-y_i)\log(1-h_w(x_i))\right]$$

and its derivatives is:

$$\frac{1}{n}\sum_{i=1}^{n}(h_w(x_i)-y_i)x_i$$

For the experiment of Linear Classification and Stochastic Gradient Descent, the loss function we use is:

$$\frac{w^2}{2}+\frac{C}{n}\sum_{i=1}^{n}\max\left(0,1-y_i(w^T x_i+b)\right)$$

And its derivatives is:

$$\frac{\Delta y}{\Delta w} = w + \frac{C}{n}\sum_{i=1}^{n} g_w(x_i)$$
$$\frac{\Delta y}{\Delta b} = \frac{C}{n}\sum_{i=1}^{n} g_b(x_i)$$

10. Experimental results and curve:

1) Logistic Regression and Stochastic Gradient Descent:

a) NAG:

Hyper-parameter selection: rate = 0.001, $\gamma = 0.9$,batch_size = 6513, iterations = 2000

Predicted Results (Best Results): Converge to 0.37

Loss curve:

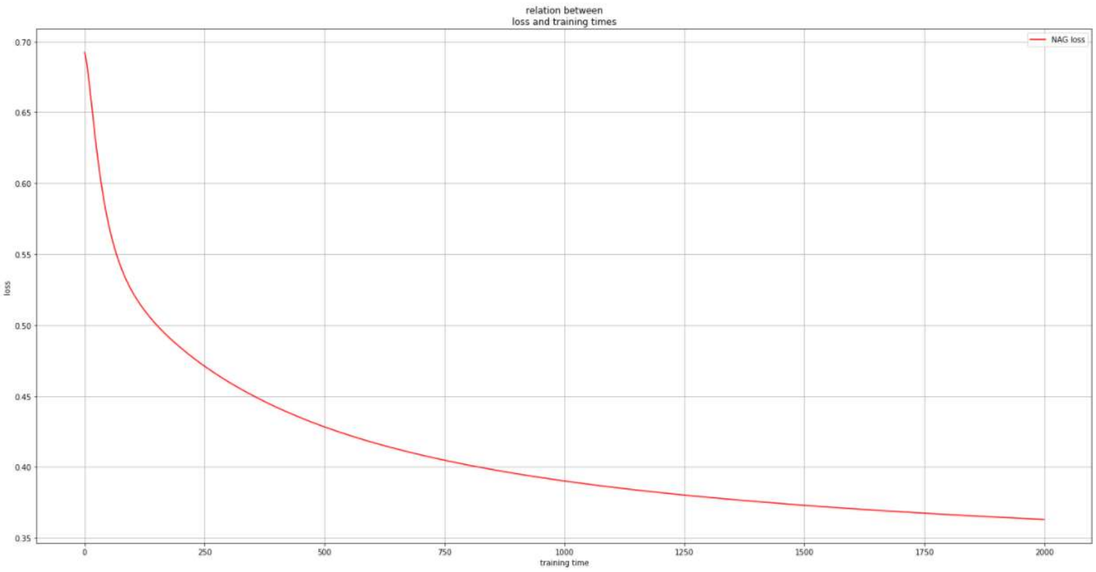

Figure 1

b) RMSProp:

Hyper-parameter selection: rate = 0.001, $\gamma = 0.9$,batch_size = 6513, iterations = 1000

Predicted Results (Best Results): Converge to 0.34

Loss curve:

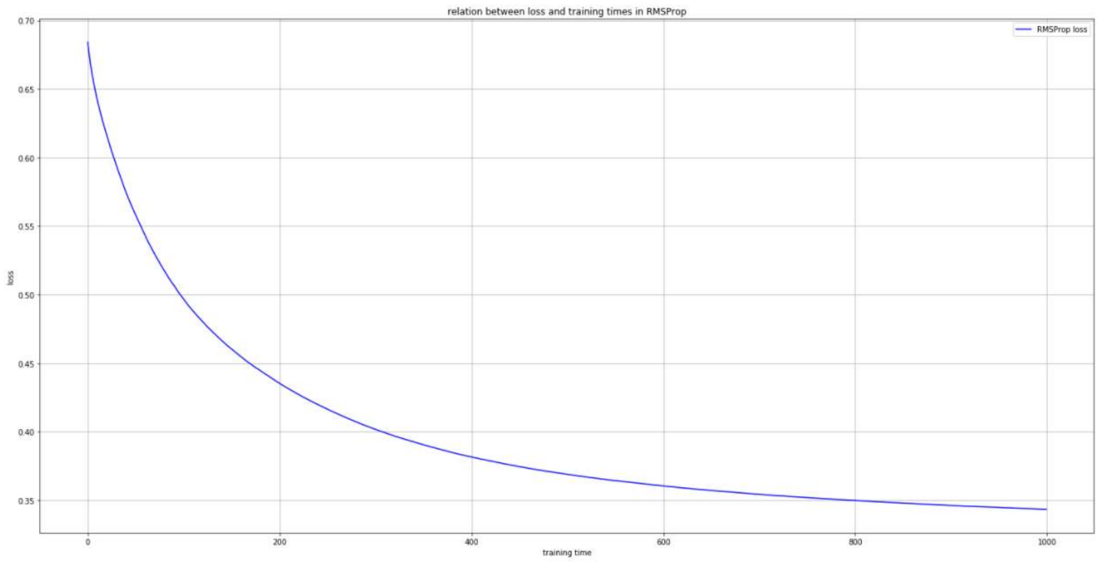

Figure 2

c) AdaDelta:

Hyper-parameter selection: $\gamma = 0.95$,batch_size = 6513, iterations = 1000

Predicted Results (Best Results): Converge to 0.12

Loss curve:

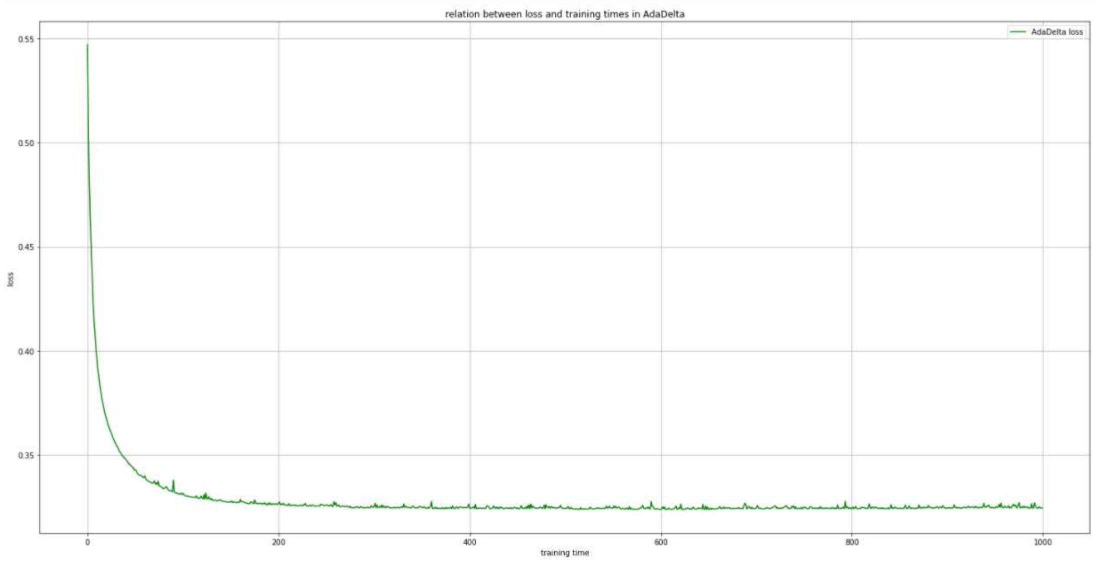

Figure 3

d) Adam:

Hyper-parameter selection: $\gamma = 0.999, \beta = 0.9$,learning_rate = 0.001, batch_size = 6513, iterations = 1000

Predicted Results (Best Results): Converge to -1.2

Loss curve:

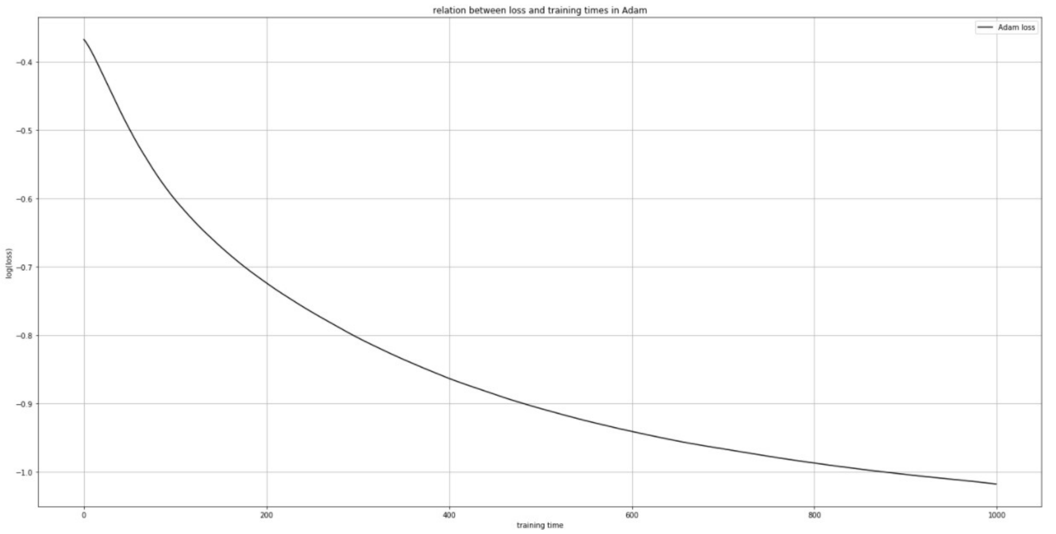

Figure 4

2) Linear Classification and Stochastic Gradient Descent:

a) NAG:

Hyper-parameter selection: rate = 0.001, $\gamma = 0.9$,batch_size = 6513, iterations = 200

Predicted Results (Best Results): Converge to 0.099955

Loss curve:

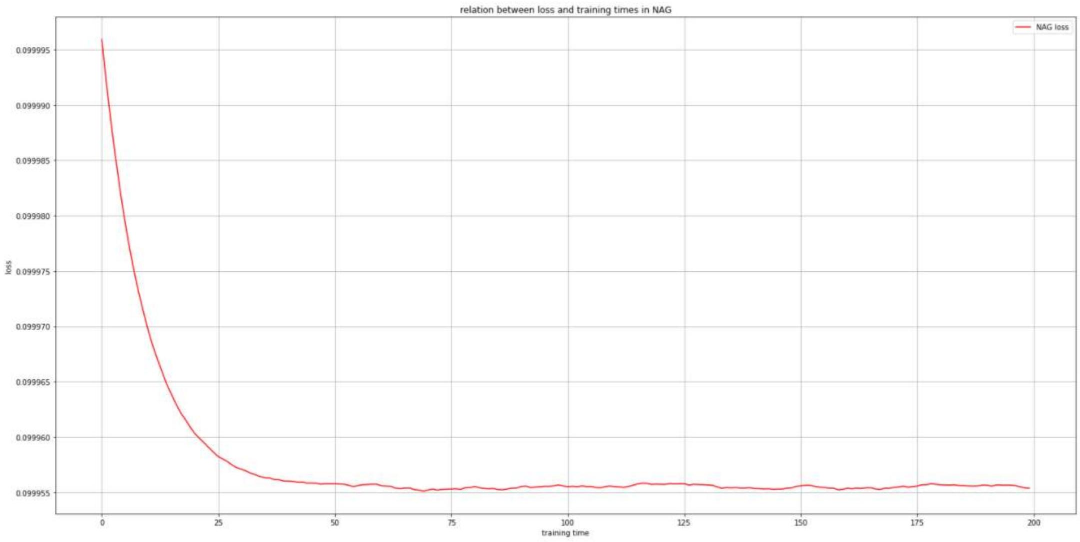

Figure 5

b) RMSProp:

Hyper-parameter selection: rate = 0.002, $\gamma = 0.9$,batch_size = 6513, iterations = 1000

Predicted Results (Best Results): Converge to 0.03

Loss curve:

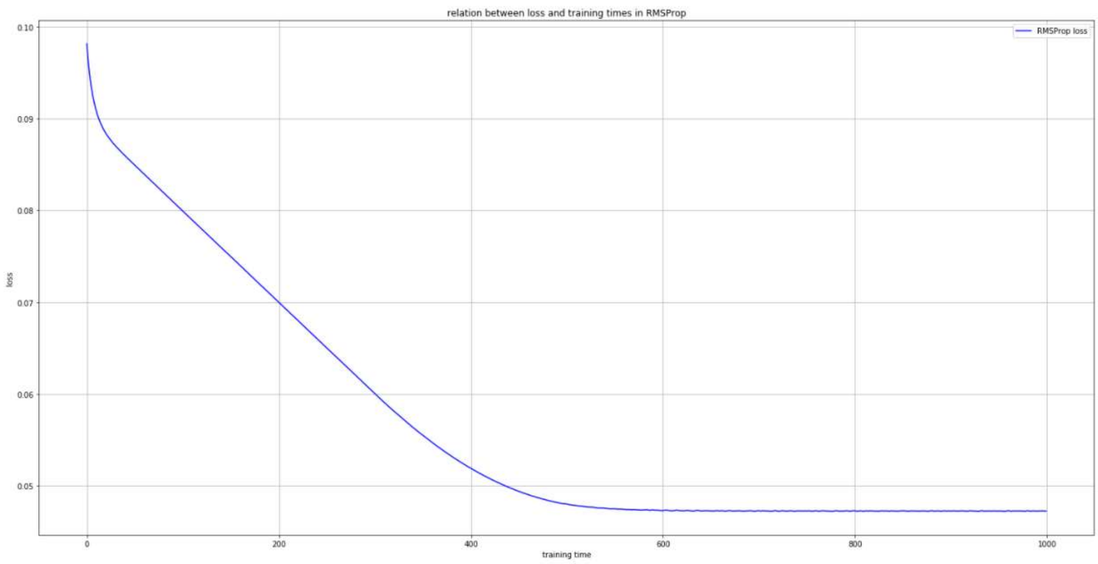

Figure 6

c) AdaDelta:

Hyper-parameter selection: $\gamma = 0.95$,batch_size = 6513, iterations = 1000

Predicted Results (Best Results): the answer fluctuates strongly no matter how I ajust the parametter

Loss curve:

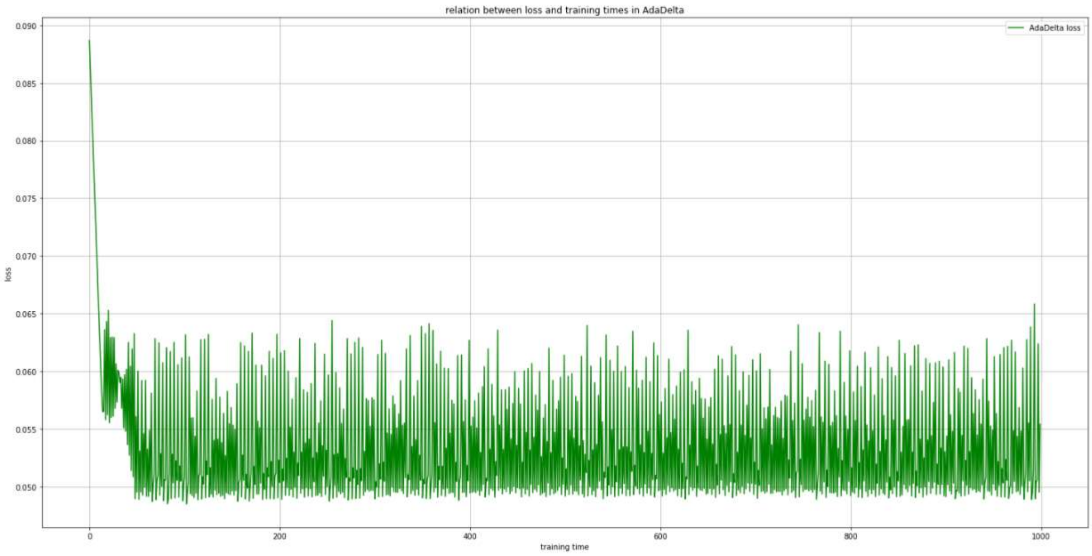

Figure 7

d) Adam:

Hyper-parameter selection: $\gamma = 0.999, \beta = 0.9$,learning_rate = 0.001, batch_size = 6513, iterations = 1000

Predicted Results (Best Results): Converge to -3.6

Loss curve:

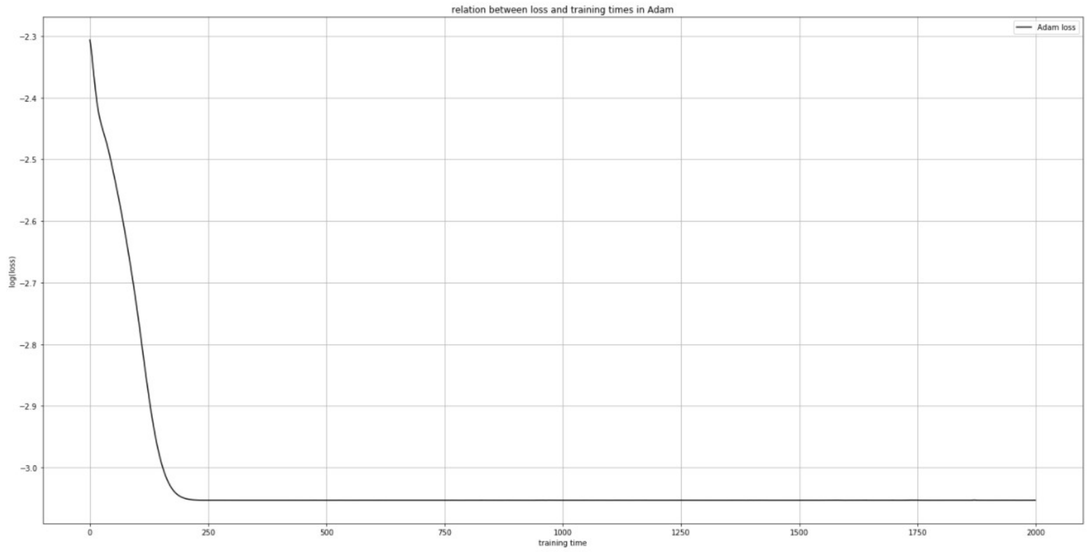

Figure 8

11. Results analysis:

In the experiment of Logistic Regression and Stochastic Gradient Descent, we found the AdaDelta has the best outcome. The loss go to the convergence after about 200 iterations.But in the experiment of Linear Classification and Stochastic Gradient Descent, the NAG shows the best behavior. Strangely, the outcome of the AdaDelta fluctuate very intensively. I tried to change the parameter, the batch size, but failed at the end. It make me confused and nervous.

12. Similarities and differences between logistic regression and linear classification:

Similarity: Both of them solve the classification problem. And both can use the Gradient Descent.

Difference:

Linear classification use a linear classifier to distinguish differernt classes. Its outcome is yes or no, which is a discrete value.

Logistic regression use a continuous function, which outcome is continuous. By set a threshold value, we use it to solve the classification problem.

13. Summary:

To sum up, the outcome is very relative to the parameter we set such as the learning rate and other argument in the formula. We has to learn to adjust the parameter if we want to have a beautiful outcome.