华南理工大学

South China University of Technology

# The Experiment Report of Machine Learning

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

**SUBJECT:** SOFTWARE ENGINEERING

Author:
Zhiwei Lin

Supervisor:
Mingkui Tan

Student ID：
201530612316

Grade:
Undergraduate

December 15, 2017

# Logistic Regression, Linear Classification and Stochastic Gradient Descent

**Abstract—This report is about the second experiment that Logistic Regression, Linear Classification and Stochastic Gradient Descent. I mainly focus on the logistic regression and the 4 optimized method.**

## I. INTRODUCTION

**1. Topic:** Logistic Regression, Linear Classification and Stochastic Gradient Descent
**2. Time: 2017.12.2**
**3. Reporter: Zhiwei Lin**
**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.

## II. METHODS AND THEORY

**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. Please download the training set and validation set.
**6. Experimental steps:**
The experimental code and drawing are completed on jupyter.
*Logistic Regression and Stochastic Gradient Descent*

1. Load the training set and validation set.
2. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
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. Repeat 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.

*Linear Classification and Stochastic Gradient Descent*

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

## III. EXPERIMENT

**7. Code:**
**7.1 Logistic Regression**
**1)calculate the gradient**

```
#since M is batch size
   for i in range(M):
      g += (c*_y[i]*x[i]/(1+np.exp(_y[i]*x[i]*w))).T
   g/=M
   g =0.5*w -g
```

**2)SGD**

```
#no more operations
```

**3)NAG**

```
#phi is a parameter, usually set as 0.9,and v is a variable.
   v = phi*v + c*g
   g = w - v
```

**4)RMSProp**

```
#phi is a parameter, usually set as 0.9, Gt is a variable like v
#c is a parameter the learning rate, usually set as 0.001
   g =0.5*w - g
   Gt = phi*Gt + (1-phi)* np.multiply(g , g)
   w = w - np.multiply(c/np.sqrt(Gt + 1e-8),g)
```

**5)AdaDelta**

```
# phi is a parameter, usually set as 0.95, Gt is a variable like v
#DeltaW is used to change w, always minus
   Gt = phi*Gt + (1-phi)* np.multiply(g , g)
   DeltaW = - (np.multiply((np.sqrt(Delt + 1e-8)/np.sqrt(Gt +
1e-8)),g))
   w = w + DeltaW
   Delt = phi*Delt + (1-phi)* np.multiply(DeltaW , DeltaW)
```

**6) Adam**

```
    Gt = phi*Gt + (1-phi)* np.multiply(g , g)
    mt = beta*mt + (1-beta)*g
    alpha = c * np.sqrt(1 - phi**k)/np.sqrt(1-beta**k)
    w = w - alpha * mt /np.sqrt(Gt + 1e-8)
    c = c/np.sqrt(k)
```

**7)calculate the loss**

```
    for i in range(M):
        l+=np.log(1+np.exp(-y[i].T*x[i]*w))
    l/=M
    l+=C*(w.T*w)/2
```

**8)calculate the accuracy**

```
#m is mount of samples
    right = np.sign(x*w)
    for i in range(m):
        if (right[i][0] == y[i][0]):
            right_num+=1
    return right_num/m
```

**7.2 Linear Classification**

**1)calculate the gradient**

```
#
    for i in range(M):
        if(1 - _y[i,0]*(x[i]*w))<0:
            _y[i] = 0
    g = x.T.dot(_y)
    g = w + c*g/M
```

**2)SGD** the code 2-6 is the same as **7.1**

**3)NAG**

**4)RMSProp**

**5)AdaDelta**

**6) Adam**

**7) calculate the loss**

```
    for i in range(M):
        if(1 - (y[i,0]*(x[i]*w))[0,0])<0:
            _y[i]=0
        else :
            _y[i]=1 - (y[i]*(x[i]*w))[0,0]
        l+=_y[i]
    l/=M
    l*=c
    l = (w.T*w)/2 +l
```

**8) calculate the accuracy**

(8.1-11.1 respectively for logistic regression and 8.2-11.2 for linear classification)

**8.1 The initialization method of model parameters:**

All – zeros initialization

**9. 1The selected loss function and its derivatives:**

**Loss function:**

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i\cdot\mathbf{w}^\top\mathbf{x}_i}) + \frac{\lambda}{2}||\mathbf{w}||_2^2$$

**Derivatives:**

$$\mathbf{w}' \to \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = (1-\eta\lambda)\mathbf{w} + \eta\frac{1}{n}\sum_{i=1}^{n}\frac{y_i\mathbf{x}_i}{1+e^{y_i\cdot\mathbf{w}^\top\mathbf{x}_i}}$$

**10. 1Experimental results and curve:**

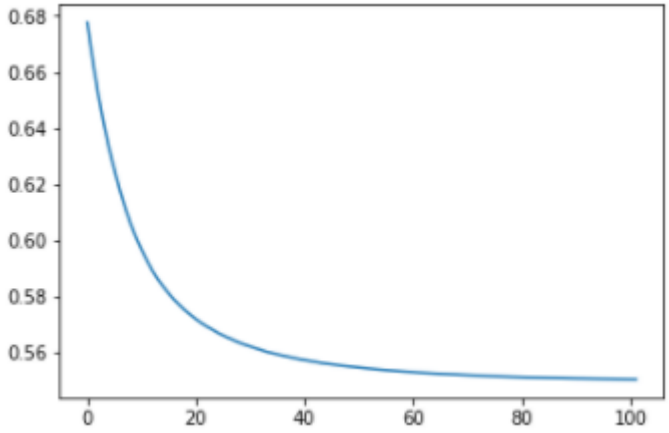
**10.1.1 NAG**

*Hyper-parameter selection:*

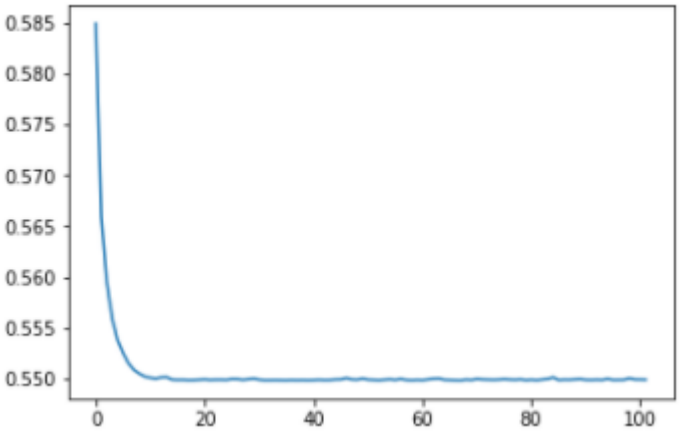
Phi:0.9

Learning rate:

0.03：



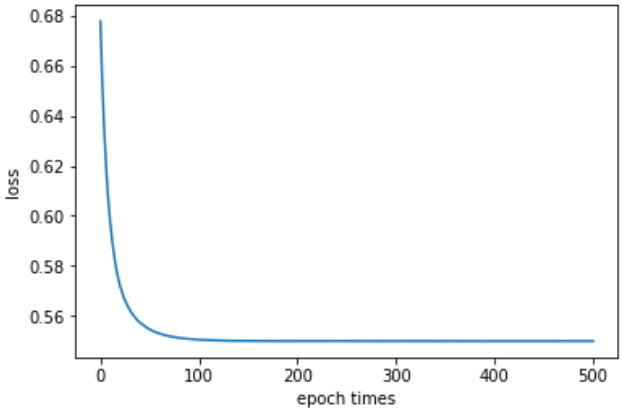

0.3

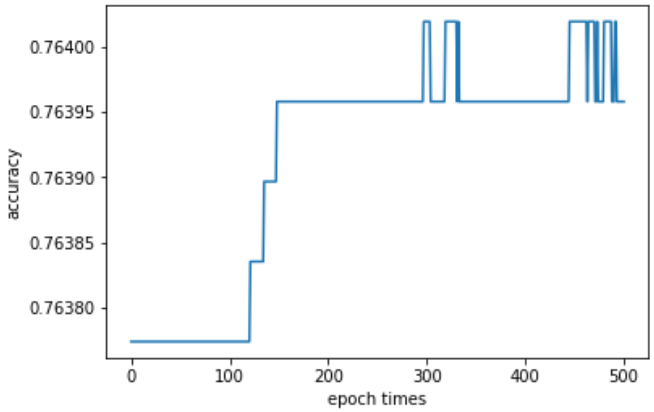


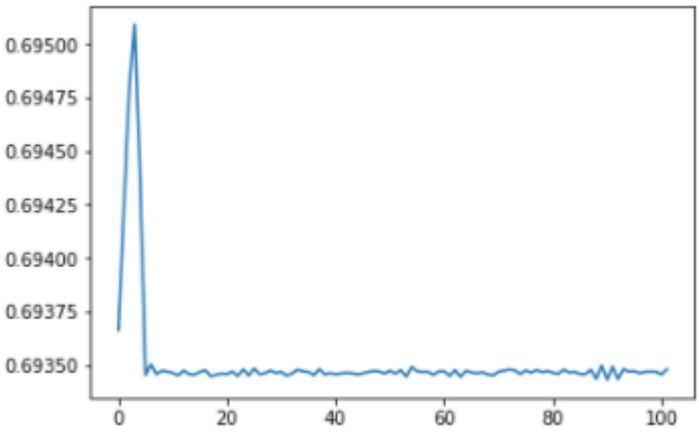

*Predicted Results (Best Results):*

Accuracy:0.76

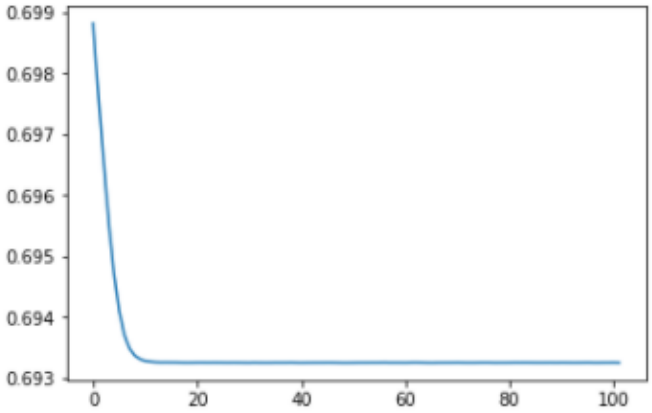
*Loss curve:*



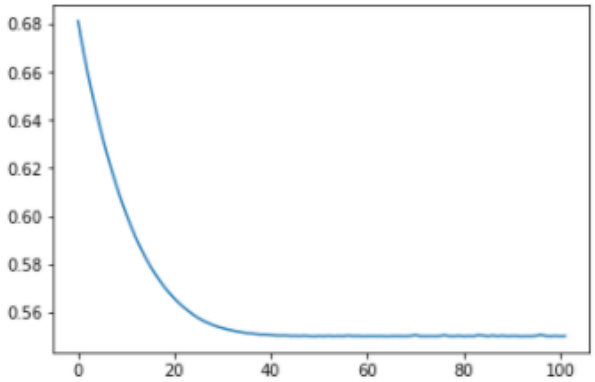
0.0001



**10.1.2 RMSProp**

*Hyper-parameter selection:*

Phi:0.9
Learning rate:
0.001

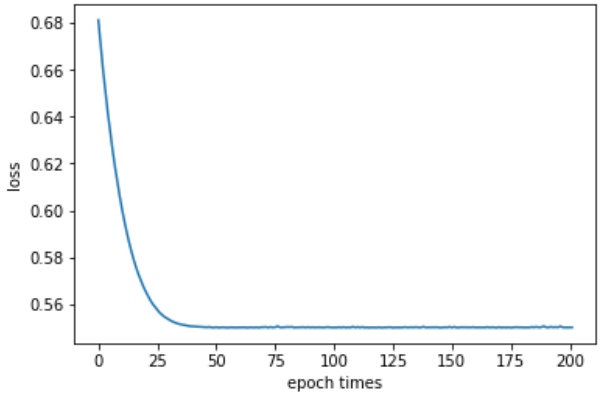


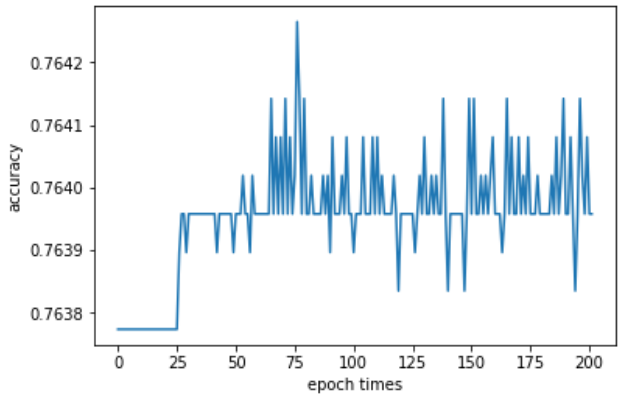
0.003

*Predicted Results (Best Results):*

Accuracy:0.7643

*Loss curve:*

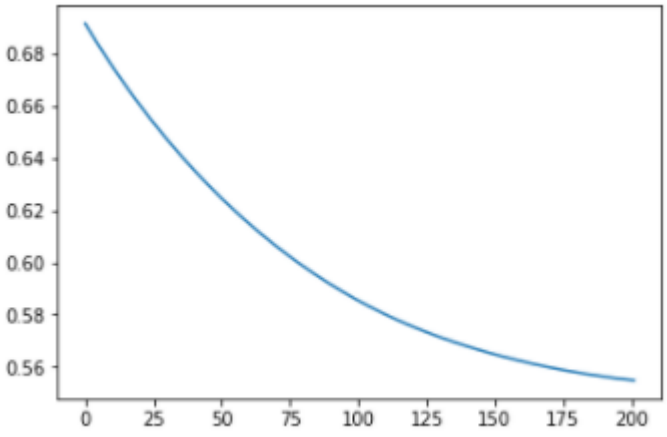



### 10.1.3 AdaDelta

*Hyper-parameter selection:*

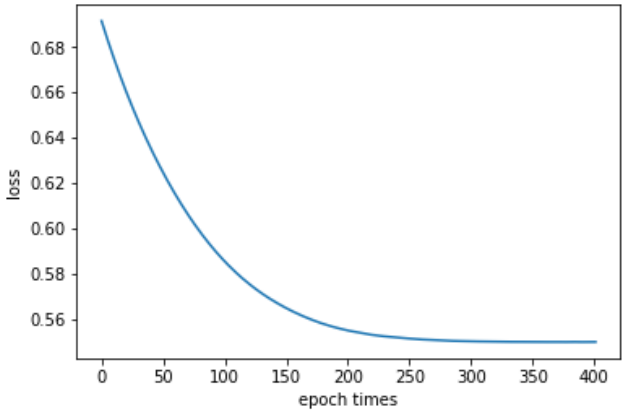
Phi:0.95

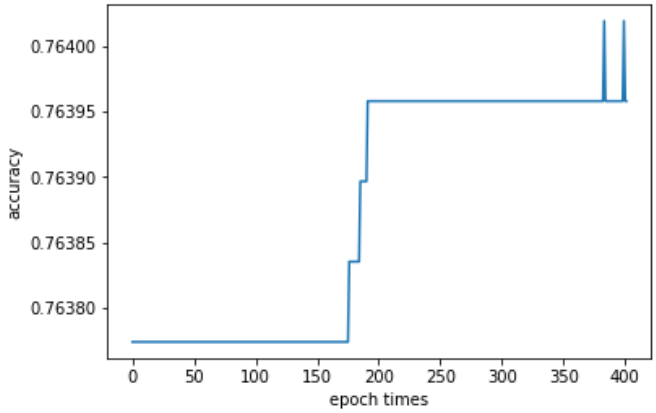


*Predicted Results (Best Results):*

Accuracy:0.76

*Loss curve:*

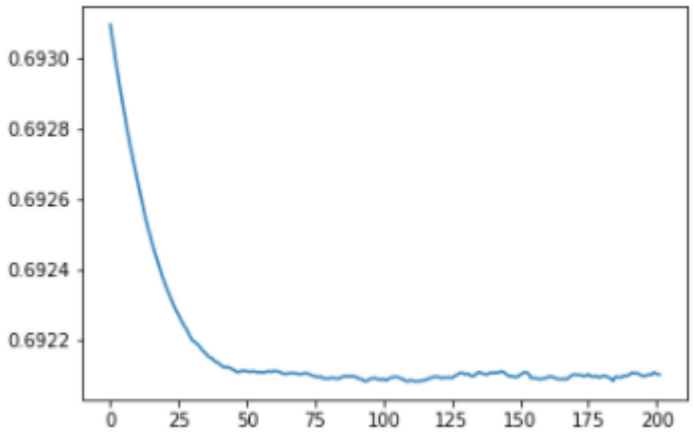



### 10.1.4 Adam

*Hyper-parameter selection:*

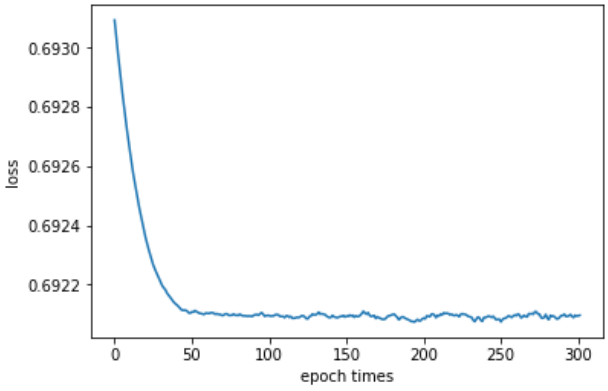
Beta:0.9
Phi:0.999
Learning rate:
0.001

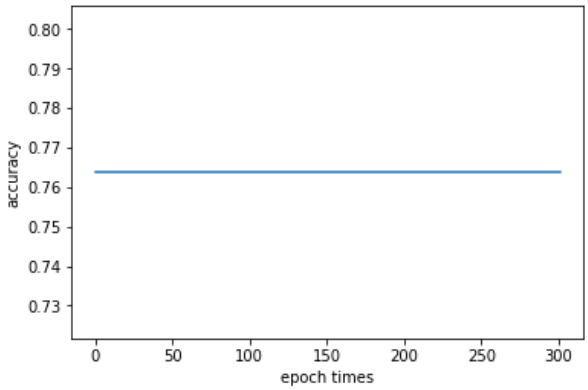


*Predicted Results (Best Results):*

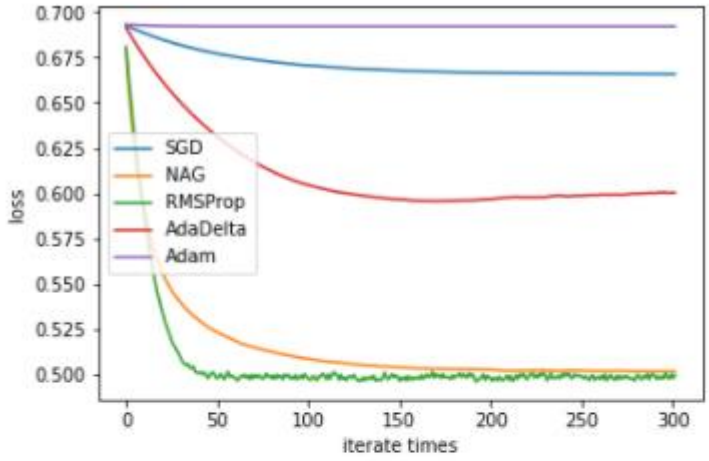
Accuracy:0.764

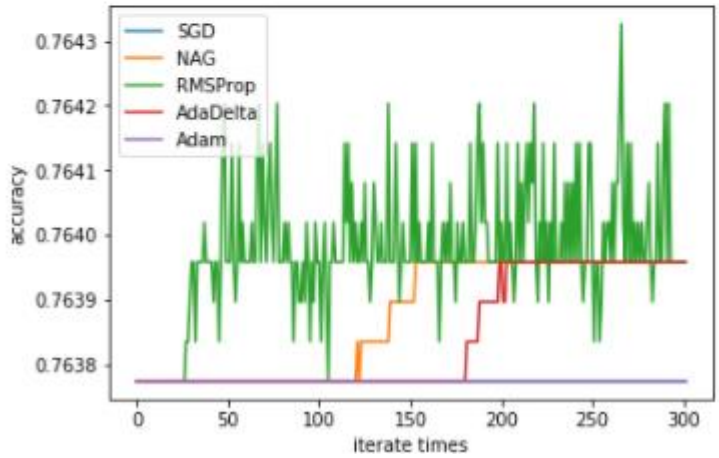
*Loss curve:*




**11. 1Results analysis:**




This graph is made by run the 5 method together, so, it can see the differences between them because they use the same training data set.The RMSProp converge faster but has a big oscillation, I think it's because I choose a wrong learning rate. It may be too big.

**8. 2The initialization method of model parameters:**

All – zeros initialization

**9. 2The selected loss function and its derivatives:**

**Loss function:**

$$\min_{\mathbf{w},b} \quad L : \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{|\boldsymbol{\mathcal{S}}_k|} \sum_{i \in \boldsymbol{\mathcal{S}}_k} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

**Derivatives:**

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \mathbf{w} + \frac{C}{|\boldsymbol{\mathcal{S}}_k|} \sum_{i \in \boldsymbol{\mathcal{S}}_k} g_{\mathbf{w}}(\mathbf{x}_i)$$

**10. 2Experimental results and curve:**

**10.2.1 NAG**

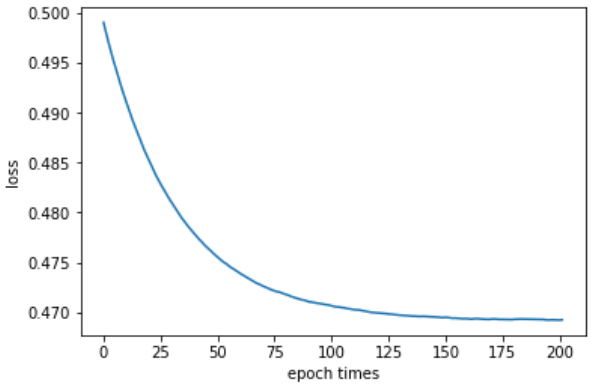
*Hyper-parameter selection:*

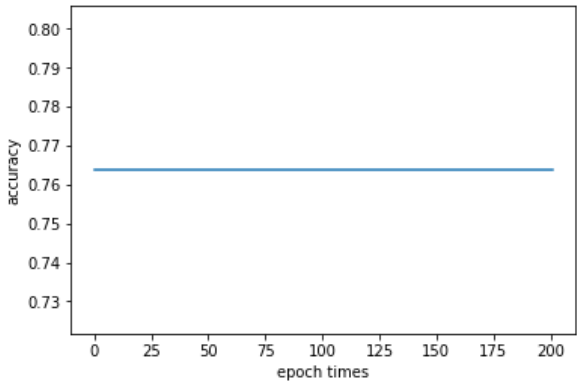
Phi:0.9
Learning rate:
0.03

*Predicted Results (Best Results):*

Accuracy:0.764

*Loss curve:*

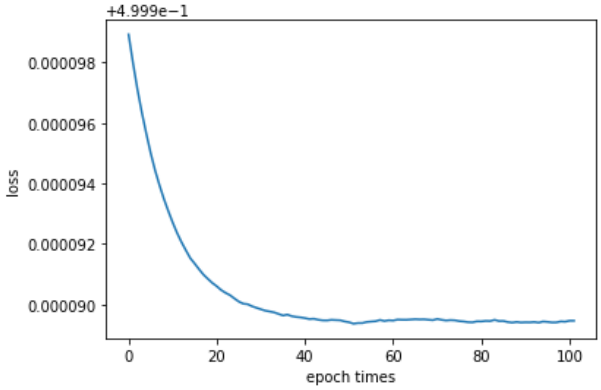


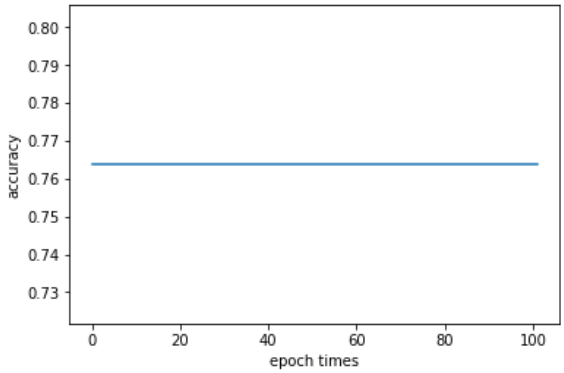




**10.2.2 RMSProp**

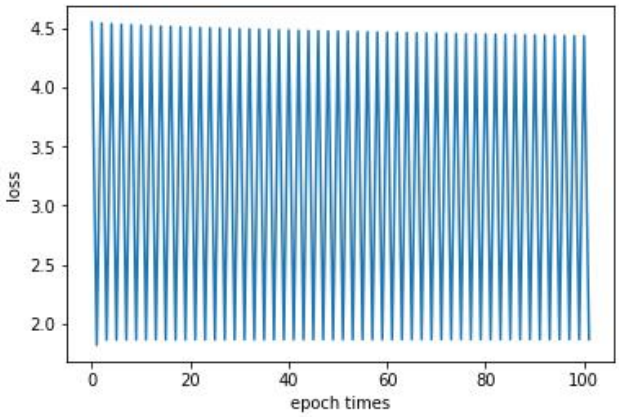
*Hyper-parameter selection:*

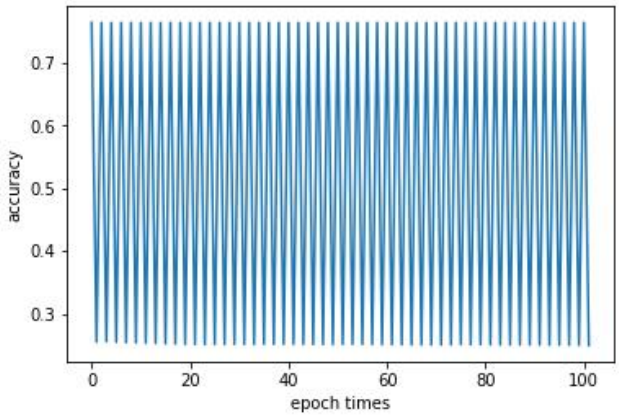
Phi:
Learning rate:
0.00001:




0.1



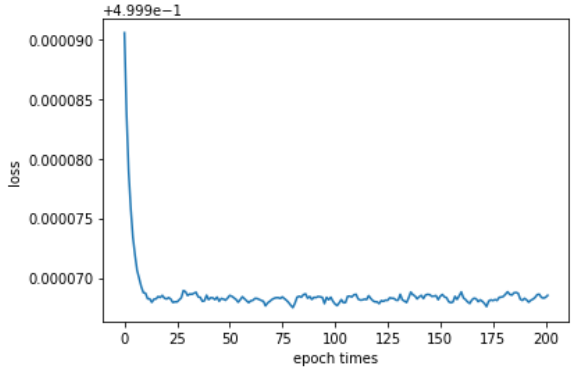



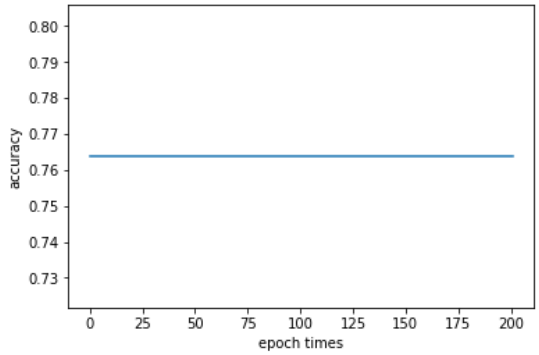

*Predicted Results (Best Results):*

Accuracy:0.764

*Loss curve:*

0.00003

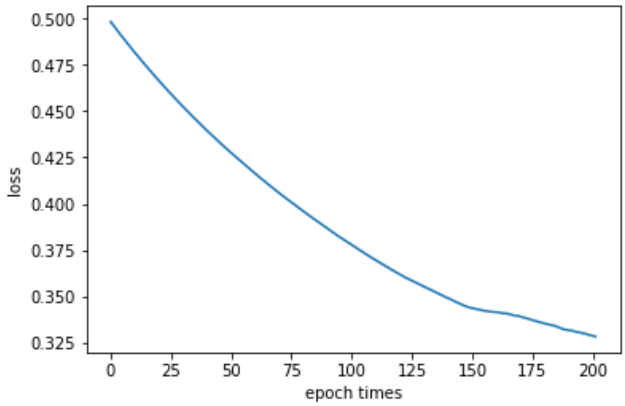


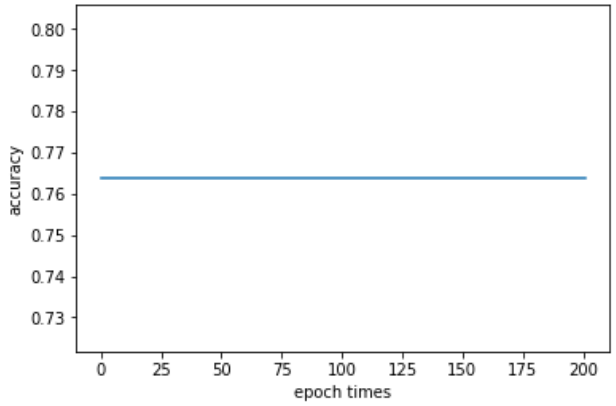




**10.2.3 AdaDelta**

*Hyper-parameter selection:*

Phi:0.95
Epoch times:
200; we could see it has not converged yet.

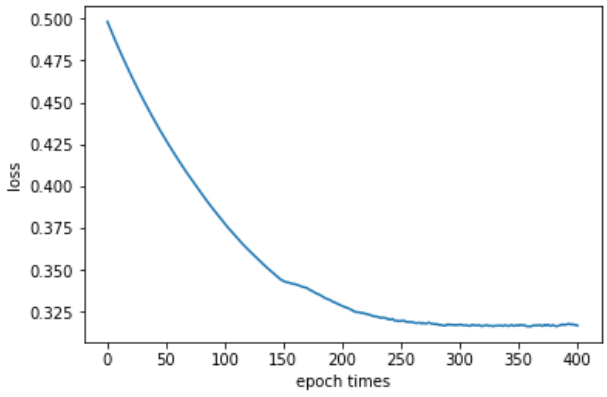


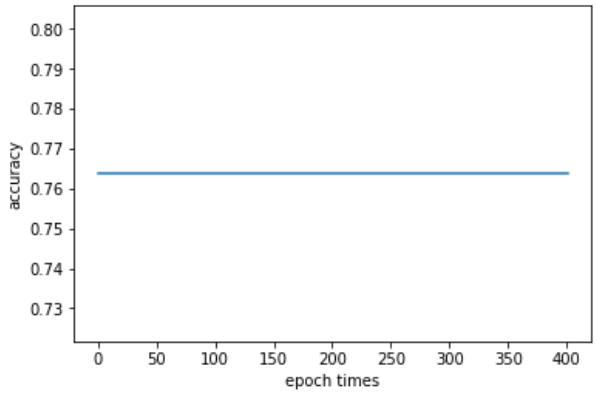




*Predicted Results (Best Results):*

Accuracy:0.764

*Loss curve:*




### 10.2.4 Adam

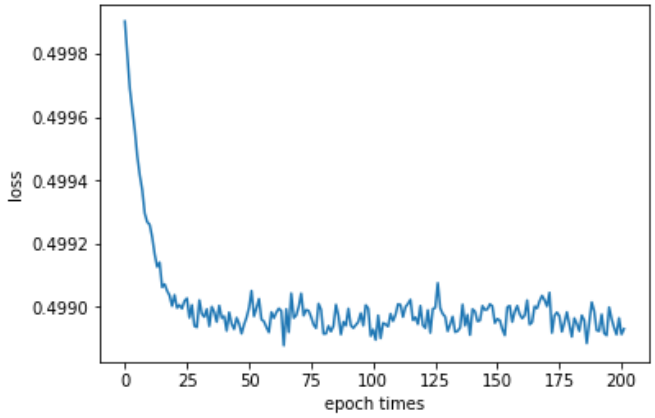
*Hyper-parameter selection:*

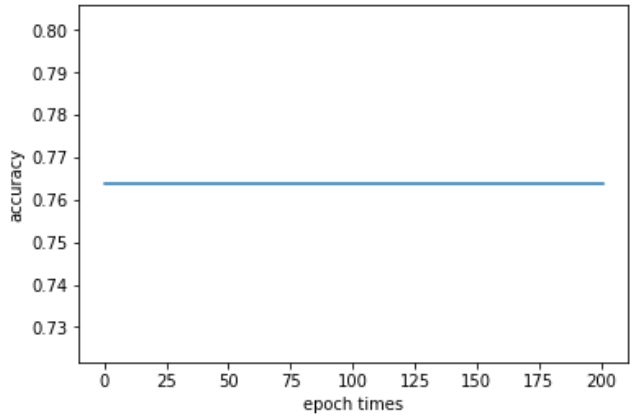
Beta:0.9
Phi:0.999
Learning rate: 0.001

*Predicted Results (Best Results):*

Accuracy:0.764

*Loss curve:*




**11. 2Results analysis:**
I still can't get the right parameter. I think it's why all my accuracy graph is a straight line.

## IV. CONCLUSION

**12. Similarities and differences between logistic regression and linear classification：**
**Similarities**: 1)logistic regression and linear classification are both the classification problem; 2)using the same method named Stochastic Gradient Descent(SGD) when deal with the big data; 3)they need to set the threshold which used to be 0;
**Differences**:1) logistic regression is used the model call sign, while the linear classification is used Support Vector Machine(SVM) model;

**13. Summary:**
In this experiment, I mainly learn that the logistic regression, linear classification and SGD.
1) I am impressive with the **logistic regression** as a classifier. The basic ideas are the sign function and the likelihood. The main tool is the SGD. First is the continuous function which can divide the data into two parts, called binary. Secondly, build up the likelihood function, maximize the likelihood as the loss function, while minimize the loss function is equal to the maximize the likelihood.

2)**linear classification** we did the last time, this time use the different algorithm.
3)SGD, mini-batch SGD
**SGD**: choose one sample to train data every time. We use SGD method when the data is very big. It helps us get the faster GD.
**Mini-Batch SGD**: choose a part of data as one sample to train every time. Somehow, larger mini-batch size takes more computation time but less iterations.
4)what about the parameters selection?
In a word, adjusting the parameters is very difficult. I don't think there is any rules to decide the best parameter. I have to adjust all parameters to test the best one. So, if I want to be good at this work, I must need to have more experiences.
**Batch size**: since I choose the 4885 size as a mini-batch, but I think that the larger batch can make the loss less oscillation, which can bring a more stable model.
**Learning rate**: more explain in 5)
5)Finally talk about the different optimized methods(NAG，RMSProp，AdaDelta and Adam).
**NAG**: when we talk about NAG, we should think about the momentum algorithm. Momentum is a theory in physics. It's very easy to understand that the momentum will become larger with an object keeping its state all the time. So, we could get the feature of the momentum, then we predict the next time the momentum. Therefore, we could know the next gradient early. It's just like a prediction.
Before explain the rest algorithms.
**AdaGrad(Adaptive Gradient)**: one of the most difficult things in training a model is to select an adapted learning rate. If the learning rate is too small, it will be too slow to get the final model. If the learning rate is too large, it will be oscillatory and may even diverge. So, there is an algorithm called AdaGrad which help us to make the learning rate selection easy.
**RMSprop**: it's one of the optimized methods. It solves the problem that the learning rate in the AdaGrad will final tend to be zero, that the w would not change anymore.
**AdaDelta**: moreover, it's like the RMSProp algorithm. But we want a more convenient method to help us choose the best learning rate. In this method, we even don't have to set the first learning rate. It uses the Delta of the learning rate before to decide the next gradient. It's just like append a parameter to make the algorithm more accurate.
**Adam**: Sometimes, it seems to be the sum of the above algorithms.
6)what is I lack of?
I think that I still have less skills about coding. My code run a time take me more minutes, while the others can finish it in seconds. It costs a lot of my time to waiting the result(which is wrong in most of times). But this time I pay more attention to this experiment, so I understand the theory deeper.

**14.Reference:**

1. Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747.
2. https://blog.slinuxer.com/2016/09/sgd-comparison