华南理工大学

South China University of Technology

# The Experiment Report of

# *Machine Learning*


**College** Software College

**Subject** Software Engineering

**Members** Jiami Huang

**Student ID** 201537611718

**E-mail** 215880454@qq.com

**Tutor** Mingkui Tan


**Date submitted** 2017.12.15

1. **Topic:**

   Comparison of Different Classification Methods

2. **Time:**

   2017.12

**3. Reporter:**

   Jiami Huang

3. **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:**

The data set a9a in LIBSVM is used this time to do logistic regression and linear classification. It includes 32561 samples and each sample has 123 features.

**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  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，，  and .
7. Repeate step 4 to 6 for several times, and drawing graph of ，，  and  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  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 ， ， and .
7. Repeate step 4 to 6 for several times, and drawing graph of ， ， and with the number of iterations.

**7. Code:**

**NAG**

```
gamma=0.9
eta=0.1
epochs=400
epochstep=100
epochset=[]
l_NAG=[]
grad=np.zeros((n+1,1))
v=np.zeros((n+1,1))
w=np.random.rand(n+1,1)
batch_size=128

def label(x):
    if x<0.5:
        return 0
    else:
        return 1

for i in range (epochs+1):
    #training
    batch_bgn=np.random.randint(0,m-batch_size-1)
    for j in range(batch_bgn,batch_bgn+batch_size):
        h=1/(1+np.exp(x_train[j]*(gamma*v-w)))
        grad=grad+((h-y_train[j])*x_train[j]).T
    grad=grad/batch_size
    v=gamma*v+eta*grad
    w=w-v
```

## RMSProp

```
gamma=0.9
eta=0.01
eps=1e-8
epochset=[]
l_RMSP=[]
G=np.zeros((n+1,1))
grad=np.zeros((n+1,1))
w=np.random.rand(n+1,1)

for i in range(epochs+1):
    #training
    batch_bgn=np.random.randint(m-batch_size-1)
    for j in range(batch_bgn,batch_bgn+batch_size):
        h=1/(1+np.exp(-x_train[j]*w))
        grad+=((h-y_train[j])*x_train[j]).T
    grad=grad/batch_size
    G=gamma*G+np.multiply((1-gamma)*grad,grad)
    w=w-np.multiply(eta/(np.sqrt(G+eps)),grad)
```

## Adadelta

```
gamma=0.999
eps=1e-8
delta_t=np.zeros((n+1,1))
epochset=[]
l_Ada=[]
G=np.zeros((n+1,1))
grad=np.zeros((n+1,1))
w=np.random.rand(n+1,1)

for i in range(epochs+1):
    #training
    batch_bgn=np.random.randint(0,m-batch_size-1)
    for j in range(batch_bgn,batch_bgn+batch_size):
        h=1/(1+np.exp(-x_train[j]*w))
        grad+=((h-y_train[j])*x_train[j]).T
    grad=grad/batch_size
    G=gamma*G+np.multiply((1-gamma)*grad,grad)
    delta_w=-np.multiply(np.sqrt(delta_t+eps),grad)/np.sqrt(G+eps)
    delta_t=gamma*delta_t+np.multiply((1-gamma)*delta_w,delta_w)
    w=w+delta_w
```

## Adam

```
beta=0.88
gamma=0.99
eta=0.005
eps=1e-8
moment=np.random.rand(n+1,1)
epochset=[]
l_Adam=[]
G=np.zeros((n+1,1))
grad=np.zeros((n+1,1))
w=np.random.rand(n+1,1)

for i in range(epochs+1):
    #training
    batch_bgn=np.random.randint(0,m-batch_size-1)

    for j in range(batch_bgn,batch_bgn+batch_size):
        h=1/(1+np.exp(-x_train[j]*w))
        grad+=((h-y_train[j])*x_train[j]).T
    grad=grad/batch_size
    moment=beta*moment+(1-beta)*grad
    G=gamma*G+np.multiply((1-gamma)*grad,grad)
    alpha=eta*np.sqrt(1-gamma)/(1-beta)
    #if(i>0):
     #   beta=beta/np.sqrt(i)
      #  eta=eta/np.sqrt(i)
    w=w-alpha*moment/np.sqrt(G+eps)
```

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

   **It's shown on the images above.(In section 7)**

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

```
for k in range(p):
    h=1/(1+np.exp(-x_valid[k]*w))
    loss+=(y_valid[k]*np.log(h)+(1-y_valid[k])*np.log(1-h))
    if label(h)==y_valid[k]:
        correct+=1
loss=np.asarray(-loss/p)
```

**10.Experimental results and curve:**

NAG

Hyper-parameter selection:

It's shown on the images above.(In section 7)

Predicted Results (Best Results):

Loss curve:

RMSProp

Hyper-parameter selection:

It's shown on the images above.(In section 7)

Predicted Results (Best Results):

Loss curve:

Adadelta

Hyper-parameter selection:

It's shown on the images above.(In section 7)

Predicted Results (Best Results):

Loss curve:

Adam

Hyper-parameter selection:

It’s shown on the images above.(In section 7)

Predicted Results (Best Results):

Loss curve:

**11. Results analysis:**

The loss function doesn’t show. I’m looking for the reason.

NAG and RMSProp behave good. Adadelta is slower , and Adam’s loss is a little bigger, maybe because the parameters are not adjust properly.

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

I have finished the linear classification part but still doing the logistic regression. Logistic regression not only gives the predicted result, but also gives the percentage of probability. linear classification only presents the classification result.

**13. Summary:**

I learned a lot from this experiment. I have a further understanding

about logistic regression and and stochastic gradient descent.