华南理工大学

South China University of Technology

The Experiment Report of

Machine Learning

College Software College

Subject Software Engineering

Members 洪海滨

Student ID 201530611586

E-mail Ocean233@live.com

Tutor 谭明奎

Date submitted 2017.12.6

1. Topic:

Logistic Regression, Linear Classification and Stochastic Gradient Descent

2. Time:

2017.12.2

3. Reporter:

洪海滨

4. Purposes:

Compare and understand the difference between gradient descent and stochastic gradient descent.

Compare and understand the differences and relationships between Logistic regression and linear classification.

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:

Experiment Step

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. 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**.

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. 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:

```
"""NAG"""
#initialize parameters with zero
W=np.zeros((m,1))
L_NAG=[]
#set hyper_parameter
eta=1e-6
gamma=1e-5
batch=100
C=10

v=0
#begin training
for epoch in range(1500):
    #calculate gradient g from partial samples
    random.seed()
    i=random.randint(0,n-1-batch)
    g=gradient(x_train[i:i+batch].reshape((batch,m)),y_train[i:i+batch].reshape((batch,1)),W-gamma*v,C)
    #update parameter W
    v=gamma*v+eta*g
    W=W-v
    #calculate test loss
    l_test=loss(x_test,y_test,W,C)
    L_NAG.append(l_test)
print('finished training')
```

```
"""RMSProp"""
#initialize parameters with zero
W=np.zeros((m,1))
L_RMSProp=[]
#set hyper_parameter
eta=1e-4
gamma=0.9
epsilon=1e-6
batch=100
C=10

G=0
#begin training
for epoch in range(1500):
    #calculate gradient g from partial samples
    random.seed()
    i=random.randint(0,n-1-batch)
    g=gradient(x_train[i:i+batch].reshape((batch,m)),y_train[i:i+batch].reshape((batch,1)),W,C)
    #update parameter W
    G=gamma*G+(1-gamma)*(g*g)
    W=W-eta/np.sqrt(G+epsilon)*g
    #calculate test loss
    l_test=loss(x_test,y_test,W,C)
    L_RMSProp.append(l_test)
print('finished training')
```

```
"""AdaDelta"""
#initialize parameters with zero
W=np.zeros((m,1))
L_AdaDelta=[]
#set hyper_parameter
gamma=0.95
epsilon=1e-9
batch=100
C=10

G=0
dt=0
#begin training
for epoch in range(1500):
    #calculate gradient g from partial samples
    random.seed()
    i=random.randint(0,n-1-batch)
    g=gradient(x_train[i:i+batch].reshape((batch,m)),y_train[i:i+batch].reshape((batch,1)),W,C)
    #update parameter W
    G=gamma*G+(1-gamma)*g*g
    dw=-np.sqrt(dt+epsilon)/np.sqrt(G+epsilon)*g
    W=W+dw
    dt=gamma*dt+(1-gamma)*dw*dw
    #calculate test loss
    l_test=loss(x_test,y_test,W,C)
    L_AdaDelta.append(l_test)
print('finished training')
```

```
"""Adam"""
#initialize parameters with zero
W=np.zeros((m,1))
L_Adam=[]
#set hyper_parameter
beta=0.9
gamma=0.9
eta=1e-5
epsilon=1e-9
batch=100
C=10

M=0
#begin training
for epoch in range(1500):
    #calculate gradient g from partial samples
    i=random.randint(0,n-1-batch)
    g=gradient(x_train[i:i+batch].reshape((batch,m)),y_train[i:i+batch].reshape((batch,1)),W,C)
    #update parameter W
    M=beta*M+(1-beta)*g
    G=gamma*G+(1-gamma)*g*g
    alpha=eta*np.sqrt(1-math.pow(gamma,epoch))/(1-beta)
    W=W-alpha*M/np.sqrt(G+epsilon)
    #calculate test loss
    l_test=loss(x_test,y_test,W,C)
    L_Adam.append(l_test)
print('finished training')
```

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

hold-out

9. The initialization method of model parameters:

Initialize parameters with zero

10. The selected loss function and its derivatives:

Logistic Regression:

loss function:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} log\, h_\theta\left(x^{(i)}\right) + \left(1-y^{(i)}\right) log\, \left(1-h_\theta\left(x^{(i)}\right)\right)\right]$$

derivatives:

$$\frac{\partial J(\theta)}{\partial \theta} = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y\right)x^{(i)}$$

Linear Classification:

loss function:

$$\frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N} max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

derivatives:

$$\nabla f(\beta) = \begin{cases} \mathbf{w}^\top - C\mathbf{y}^\top \mathbf{X} & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ \mathbf{w}^\top & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases}$$

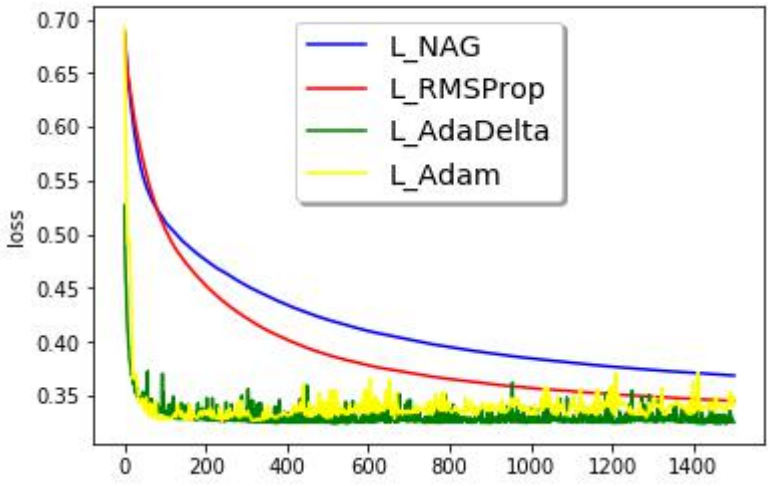
11. Experimental results and curve:

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

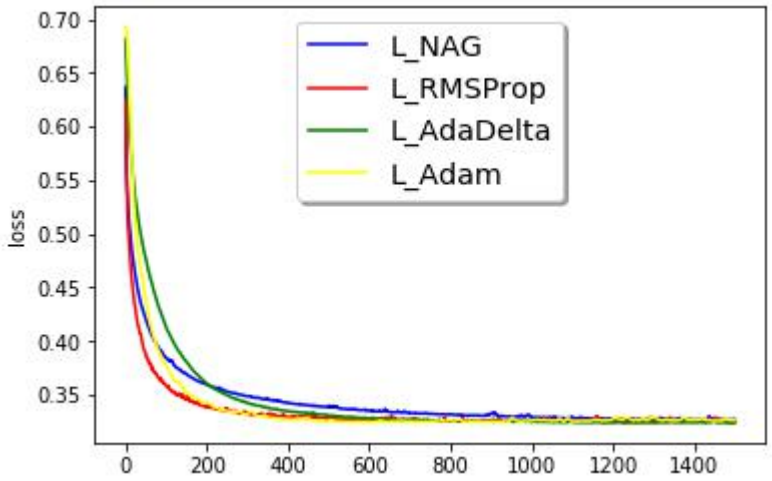
batch=100,epoch=1500
Logistic Regression:
NAG:eta=0.1,gamma=0.1,
RMSProp:eta=0.01,gamma=0.9,epsilon=1e-3
AdaDelta:gamma=0.9,epsilon=1e-6
Adam:eta=0.001,beta=0.9,gamma=0.999,epsilon=1e-6
Linear Classification:
NAG:eta=1e-6,gamma=1e-5
RMSProp:eta=1e-4,gamma=0.9,epsilon=1e-6
AdaDelta:gamma=0.95,epsilon=1e-9
Adam:eta=1e-5,beta=0.9,gamma=0.999,epsilon=1e-9

Assessment Results (based on selected validation):

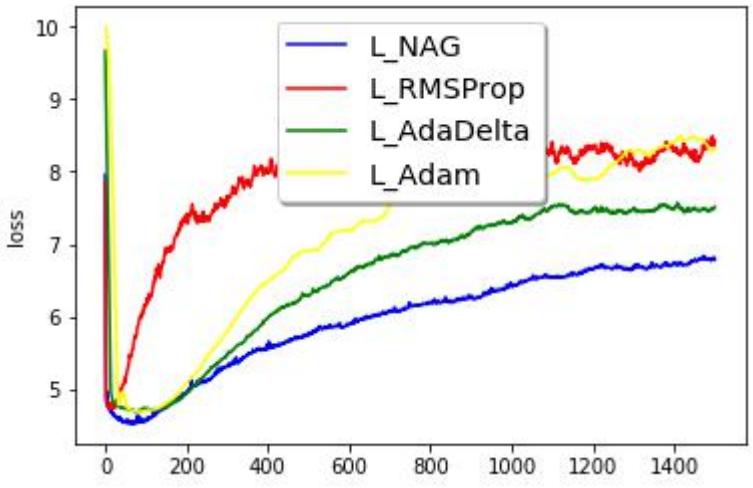
Logistic Regression：batch=100
NAG:eta=0.01,gamma=0.1
RMSProp:eta=0.001,gamma=0.9,epsilon=1e-3
AdaDelta:gamma=0.9,epsilon=1e-3
Adam:eta=0.1,beta=0.9,gamma=0.999,epsilon=1e-6



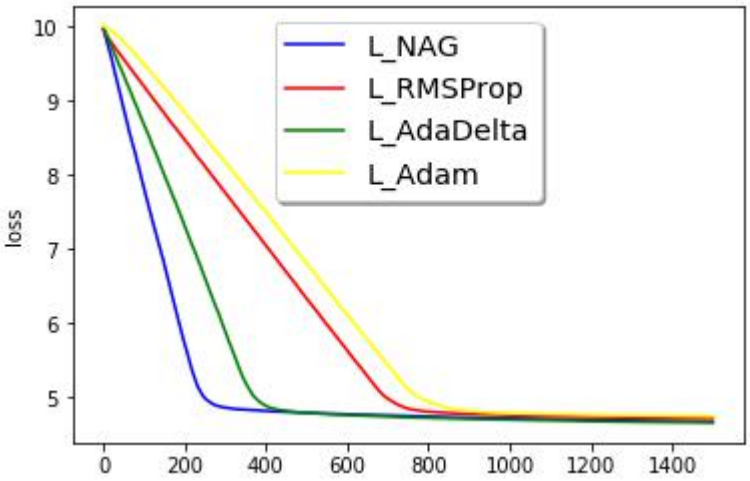
NAG:eta=0.1,gamma=0.1
RMSProp:eta=0.01,gamma=0.9,epsilon=1e-3
AdaDelta:gamma=0.9,epsilon=1e-6
Adam:eta=0.001,beta=0.9,gamma=0.999,epsilon=1e-6



Linear Classification：batch=100,C=10
NAG:eta=1e-4,gamma=1e-5
RMSProp:eta=0.01,gamma=0.9,epsilon=1e-3
AdaDelta:gamma=0.95,epsilon=1e-6
Adam:eta=1e-3,beta=0.9,gamma=0.999,epsilon=1e-6



NAG:eta=1e-6,gamma=1e-5
RMSProp:eta=1e-4,gamma=0.9,epsilon=1e-6
AdaDelta:gamma=0.95,epsilon=1e-9
Adam:eta=1e-5,beta=0.9,gamma=0.999,epsilon=1e-9

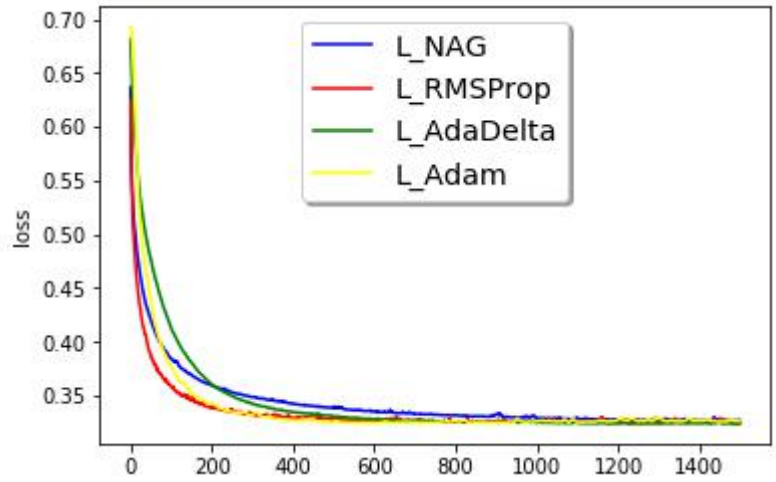


Predicted Results (Best Results):

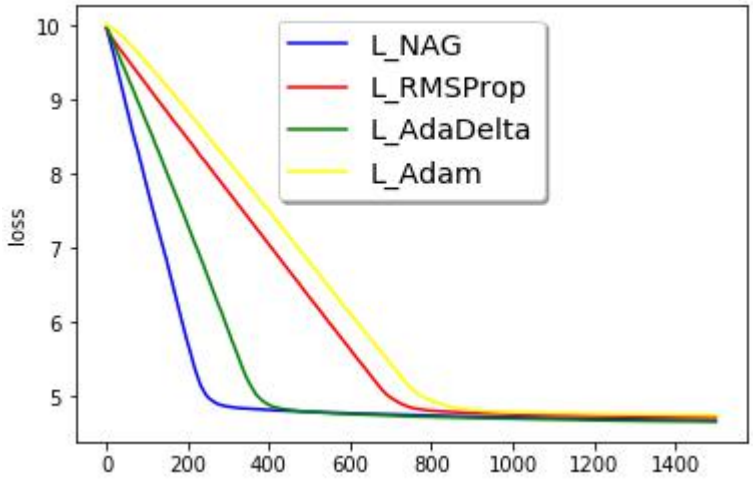
Logistic Regression：min_loss≈0.32
Linear Classification：min_loss≈4.8

Loss curve:

Logistic Regression：



Linear Classification：



12. Results analysis:

Logistic Regression:By using different optimized methods, all losses converge to the same value at similar speeds.

Linear classification:By using different optimized methods, all losses converge to the same value at different speeds.

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

Similarities:They are both used for classification.

Differences:The predicted values of logistic regression are

probabilities and are therefore restricted to (0,1). The predicted values of linear classification are 0 and 1.

14. Summary:

Logistic regression performs better than linear classification at the data set. Under the same conditions, logistic regression spend less time computing loss and gradient. Also it’s loss converge more fast.

Hyper parameters cannot be learned directly from the data in the standard model training process and need to be predefined. They can be decided by setting different values, training different models, and choosing the values that test better.