South China University of Technology

The Experiment Report of *Machine Learning*

College Software College

Subject Software Engineering

Members Bo Wang

Student ID 20130381731

E-mail 765246150@qq.com

Tutor MingKui Tan

Date submitted 2017.12 .15

1. Topic:

Logistic Regression, Linear Classification and Stochastic Gradient Descent

2. Time:

2017/12/8

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.

4. 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.

5. Experimental steps:

Logistic Regression and Stochastic Gradient Descent

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 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. Initialize SVM model parameters, you can consider initializing 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.

6. Code:

Logisic Regression

```
In  [55]: import numpy
          import random
          import math
          from sklearn.datasets import load_svmlight_file
          from sklearn.model_selection import train_test_split
          from matplotlib import pyplot
```

```
In  [56]: def grad(x, y, w):                                    #逻辑回归
              gradient = x * (y - x.dot(w.T))                   #求梯度    与线性回归一样
              return gradient

          def compute_loss(x, y, w, i):
              loss = 0                                          #SGD的loss函数
              for m in range(len(i)):
                  loss += 0.5 * ((y[i[m]] - x[i[m],:].dot(w.T)) ** 2)
              return loss/len(i)
```

```
In  [57]: def NAG(x, y, x_test, y_test, w, C, r, gamma, threshold, count):
              vt = numpy.zeros(w.shape)
              loss_history = []
              test_loss_history = []
              random_index = []
              random_test_index = []
              for i in range(count):
                  random_num = random.randint(0, x.shape[0]-1)
                  random_test_num = random.randint(0, x_test.shape[0]-1)
                  random_index.append(random_num)
                  random_test_index.append(random_test_num)
              for i in range(count):
                  gradient = grad(x[random_index[i],:], y[random_index[i]], w-gamma*vt)
                  vt = gamma*vt - r*gradient
                  w -= vt
                  loss = compute_loss(x, y, w, random_index)
                  loss_history.append(loss)
                  test_loss_history.append(compute_loss(x_test, y_test, w, random_test_index))
                  if loss < threshold :
                      break
              return w, loss_history, test_loss_history

          def RMSProp(x, y, x_test, y_test, w, C, r, gamma, threshold, count):
              Gt = 0
              loss_history = []
              test_loss_history = []
              random_index = []
              random_test_index = []
              for i in range(count):
```

(Fill in the contents of 8-11 respectively for logistic regression and linear classification)

Logisic Regression

8. The selected loss function and its derivatives:

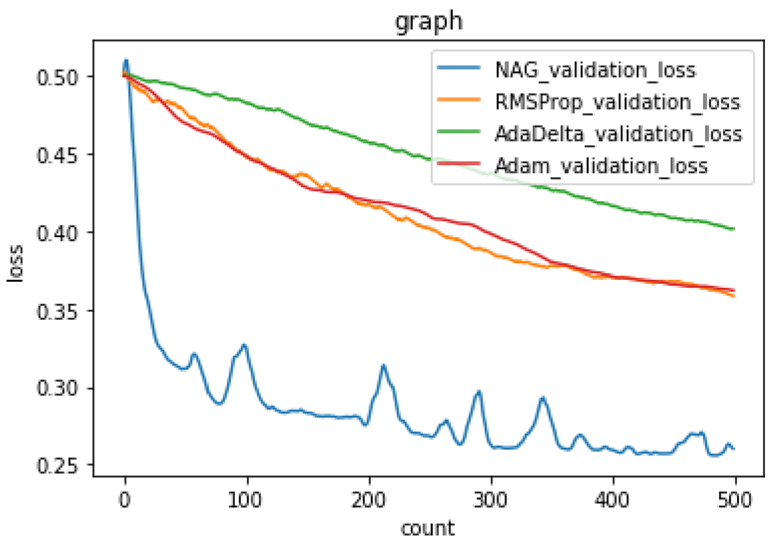
$$h_\theta(X) = \frac{1}{1+e^{-\theta^T \cdot X}}$$

$$L(\theta) = -\frac{1}{m}\sum_{i=0}^{n}(y_i \log(h_\theta(X_i)) + (1-y_i)\log(1-\log(h_\theta(X_i))))$$

$$\frac{\partial L}{\partial \theta} = \frac{1}{n}\sum_{i=0}^{n} X_i(h_\theta(X) - y_i)$$

9. Experimental results and curve:

We get the following loss graphs as results after running the program.



9. Results analysis:

From the graph we find that NAG_loss reaches the local optimal solution fastest, but it also has vibration. RMSProp and Adam are closely overlapped.

Linear classification

10.The selected loss function and its derivatives:

$$L(\theta) = \frac{1}{2n}\sum_{i=0}^{n}(y_i - h_\theta(X_i))^2$$

$h_\theta(X) = \sum_{i=0}^{n} \theta_i X_i$

$$\frac{\partial L}{\partial \theta} = \frac{1}{n}\sum_{i=0}^{n}(y_i - h_\theta(X_i)) \cdot X_i$$

9. Experimental results and curve:

11.Results analysis:

.

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

12. Summary: