South China University of Technology

# The Experiment Report of Machine Learning

**SCHOOL:** SCHOOL OF SOFTWARE ENGINEERING

**SUBJECT:** SOFTWARE ENGINEERING

Author:
ZhanCongCong

Supervisor:
Mingkui Tan

Student ID：
201530613573

Grade:
Undergraduate

December 15, 2017

# Logical Regression, Linear Classification and Gradient Descent

**Abstract—**

## I. INTRODUCTION

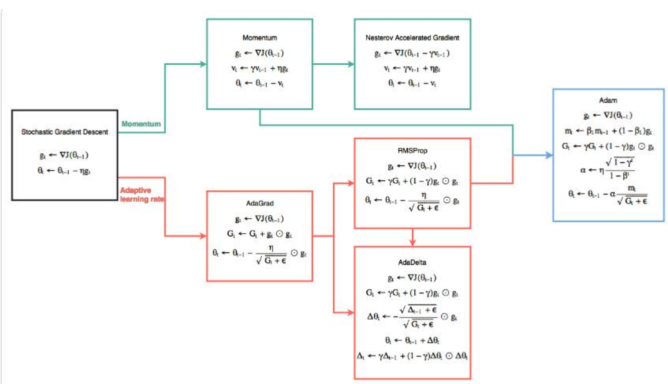
Compare and understand the difference between gradient descent and stochastic gradient descent.

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

Further understand the principles of SVM and practice on larger data.

## II. METHODS AND THEORY

**Improved optimization algorithm:**




**Logical Regression:**

Loss Function:

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

Update parameters with rate $\eta$

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

**Liear Classification:**

Loss Function:

$$\mathcal{L}(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n}a_i(1-y_i(\mathbf{w}^T\mathbf{x}_i+b))$$

Gradient Computation:

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

$$\frac{\partial f(\mathbf{w},b)}{b} = \begin{cases}-C\sum_{i=1}^{N}y_i & 1-y_i(\mathbf{w}^\top\mathbf{x}_i+b) >= 0\\ 0 & 1-y_i(\mathbf{w}^\top\mathbf{x}_i+b) < 0\end{cases}$$

## III. EXPERIMENT

Dataset:

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.

### Logical Regression:

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.**

Initialization:

```
#超参数
step_size_nag=0.003
r_nag=0.9

step_size_rms=0.019
r_rms=0.9

r_adaDelta=0.95

r_adam=0.95
b_adam=0.9
e_adam=0.025
```

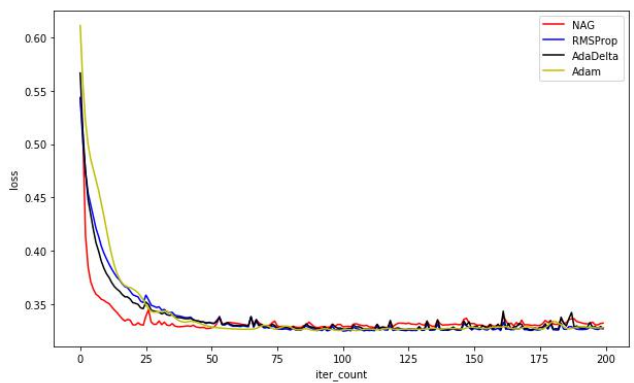
```
max_iter_count=200)
```

minibacth = 300

```
#NAG
w_nag = np.zeros((dim,), dtype=np.float32)
v_nag = np.zeros((dim,), dtype=np.float32)
#RMSProp
w_rms = np.zeros((dim,), dtype=np.float32)
Gt_rms = np.zeros((dim,), dtype=np.float32)
#AdaDelta
w_adaDelta = np.zeros((dim,), dtype=np.float32)
dt_adaDelta = np.zeros((dim,), dtype=np.float32)
Gt_adaDelta = np.zeros((dim,), dtype=np.float32)
#Adam
w_adam = np.zeros((dim,), dtype=np.float32)
Gt_adam = np.zeros((dim,), dtype=np.float32)
mt_adam = np.zeros((dim,), dtype=np.float32)
```

Result:

```
NAG准确率为：
0.8474909403599288
RMSProp准确率为：
0.8485965235550642
AdaDelta准确率为：
0.8468767274737424
Adam准确率为：
0.8473066764940729
```




Main Codes:

```
#NAG
v_nag[j] = r_nag*v_nag[j]+ step_size_nag*G_nag[j]
w_nag[j] -= v_nag[j]
#RMS
Gt_rms[j] = r_rms*Gt_rms[j]+(1-r_rms)*(G_rms[j]*G_rms[j])
w_rms[j] = w_rms[j]-step_size_rms * G_rms[j]/np.sqrt(Gt_rms[j]+1e-4)
#ADADELTA
Gt_adaDelta[j] = r_adaDelta*Gt_adaDelta[j]+(1-r_adaDelta)*(G_adaDelta[j]*G_adaDelta[j])
ds = -np.sqrt(dt_adaDelta[j]+1e-4)*G_adaDelta[j]/np.sqrt(Gt_adaDelta[j]+1e-4)
w_adaDelta[j] += ds
dt_adaDelta[j] = r_adaDelta*dt_adaDelta[j]+(1-r_adaDelta)*ds*ds
#ADAM
mt_adam[j] = b_adam*mt_adam[j]+(1-b_adam)*G_adam[j]
Gt_adam[j] = r_adam*Gt_adam[j] + (1-r_adam)*G_adam[j]*G_adam[j]
w_adam[j] = w_adam[j] - e_adam*np.sqrt(1-r_adam**(iter_count+1)
                                      )/(1-b_adam**(iter_count+1))*mt_adam[j]/np.sqrt(Gt_adam[j]+1e-8)
```

## Linear Claasification:

Steps:

*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**.

Initialization:

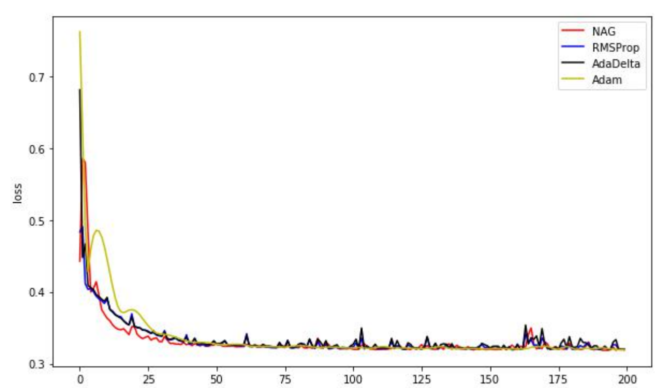
```
#超参数
step_size_nag=0.001
r_nag=0.6

step_size_rms=0.019
r_rms=0.9

r_adaDelta=0.90

r_adam=0.99
b_adam=0.9
e_adam=0.02
max_iter_count=200, C=0.9)
```

minibacth = 300

```
#NAG
w_nag = np.zeros((dim,), dtype=np.float32)
v_nag = np.zeros((dim,), dtype=np.float32)
#RMSProp
w_rms = np.zeros((dim,), dtype=np.float32)
Gt_rms = np.zeros((dim,), dtype=np.float32)
#AdaDelta
w_adaDelta = np.zeros((dim,), dtype=np.float32)
dt_adaDelta = np.zeros((dim,), dtype=np.float32)
Gt_adaDelta = np.zeros((dim,), dtype=np.float32)
#Adam
w_adam = np.zeros((dim,), dtype=np.float32)
Gt_adam = np.zeros((dim,), dtype=np.float32)
mt_adam = np.zeros((dim,), dtype=np.float32)
iter_count = 0
```

Result:

```
NAG准确率为：
0.8474909403599288
RMSProp准确率为：
0.8501320557705301
AdaDelta准确率为：
0.8490264725753947
Adam准确率为：
0.847613782937166
```




Main Codes:

```
#NAG
v_nag[j] = r_nag*v_nag[j]+ step_size_nag*(w_nag[j]-r_nag*v_nag[j]+C*G_nag[j])
w_nag[j] -= v_nag[j]
#RMS
Gt_rms[j] = r_rms*Gt_rms[j]+(1-r_rms)*((w_rms[j]+C*G_rms[j])**2)
w_rms[j] = w_rms[j]-step_size_rms * (w_rms[j]+C*G_rms[j])/np.sqrt(Gt_rms[j]+1e-4)
#ADADELTA
Gt_adaDelta[j] = r_adaDelta*Gt_adaDelta[j]+(1-r_adaDelta)*((w_adaDelta[j]+C*G_adaDelta[j])**2)
ds = -np.sqrt(dt_adaDelta[j]+1e-4)*(w_adaDelta[j]+C*G_adaDelta[j])/np.sqrt(Gt_adaDelta[j]+1e-4)
w_adaDelta[j] += ds
dt_adaDelta[j] = r_adaDelta*dt_adaDelta[j]+(1-r_adaDelta)*(ds**2)
#ADAM
mt_adam[j] = b_adam*mt_adam[j]+(1-b_adam)*(w_adam[j]+C*G_adam[j])
Gt_adam[j] = r_adam*Gt_adam[j] + (1-r_adam)*((w_adam[j]+C*G_adam[j])**2)
w_adam[j] = w_adam[j] - e_adam*np.sqrt(1-r_adam**(iter_count+1))/(1-b_adam**(
    iter_count+1))*mt_adam[j]/np.sqrt(Gt_adam[j]+1e-8)
```

## IV. CONCLUSION

Through this experiment harvested the following points:

1: Have a more in-depth understanding of its principles about Logical Regression and also have a certain understanding above the main use of the function $g\left(z\right)=\frac{1}{1+e^{-z}}$, and the logical regression is actually a classification problem.

2: Also understand the small batch gradient decline, which used in the big data sets can quickly converge, greatly reducing the running time.

3: Through this experiment, the biggest gain is to know the method of updating parameters, not only this method $\mathbf{g}_t \leftarrow \frac{1}{n}\sum_{i}^{n} \nabla J_i(\boldsymbol{\theta}_{t-1})$ $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta \mathbf{g}_t$, but there are many more excellent update algorithms, such as Adam, RMSProp, AdaDelta, etc., and found the Adam update algorithm best beacuse of stable gradition and fast convergence in the experiment.