South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Deng Huang

Supervisor:
Mingkui Tan or Qingyao Wu

Student ID:
201530211588

Grade:
Undergraduate

December 15, 2017

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—Empirical risk minimization (ERM) is always a problem in machine learning. This paper focus on the comparison of four improvements of Stochastic gradient descent (SGD) - NAG, RMSProp, AdaDelta and Adam, and their performance.**

I. Introduction

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

A. NAG

NAG (Nesterov accerlerated gradient) uses Momentum to predict next gradient, instead of using current θ

$$\begin{aligned} g_t &\leftarrow \nabla J(\theta_{t-1} - \gamma v_{t-1} \\ v_t &\leftarrow \gamma v_{t-1} + \eta g_t \\ \theta_t &\leftarrow \theta_{t-1} - v_t \end{aligned}$$

B. RMSProp

To solve the problem in AdaGrad, RMSProp was put forward by Hinton

$$\begin{aligned} g_t &\leftarrow \nabla J(\theta_{t-1}) \\ G_t &\leftarrow \gamma G_t + (1-\gamma) g_t \odot g_t \\ \theta_t &\leftarrow \theta_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \end{aligned}$$

C. AdaDelta

Also solves the same problem as RMProp does but has no learning rate to be set.

$$\begin{aligned} g_t &\leftarrow \nabla J(\theta_{t-1}) \\ G_t &\leftarrow \gamma G_t + (1-\gamma) g_t \odot g_t \\ \Delta\theta_t &\leftarrow -\frac{\sqrt{\Delta_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} \odot g_t \\ \theta_t &\leftarrow \theta_{t-1} + \Delta\theta_t \\ \Delta_t &\leftarrow \gamma \Delta_{t-1} + (1-\gamma) \Delta\theta_t \odot \Delta\theta_t \end{aligned}$$

D. Adam

Adaptive estimates of lower-order moments. Has both the advantage of AdaGrad and RMSProp.

$$\begin{aligned} g_t &\leftarrow \nabla J(\theta_{t-1}) \\ m_t &\leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t \\ G_t &\leftarrow \gamma G_t + (1-\gamma) g_t \odot g_t \\ \alpha &\leftarrow \eta \frac{\sqrt{1-\gamma^t}}{1-\beta^t} \\ \theta_t &\leftarrow \theta_{t-1} - \alpha \frac{m_t}{\sqrt{G_t + \epsilon}} \end{aligned}$$

III. Experiments

A. 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.

B. Implementation

1) Core Implementation: The python implementation was show as Figure **??**.

```
if method == 'gd':
    grad = gradient(X_batch, y_batch, w)
    w -= learning_rate * grad

elif method == 'NAG':
    grad = gradient(X_batch, y_batch, w - gamma * v)
    v = gamma * v + learning_rate * grad
    w -= v

elif method == 'RMSProp':
    grad = gradient(X_batch, y_batch, w)
    G = gamma * G + (1 - gamma) * np.square(grad)
    w -= learning_rate * 100 * grad / np.sqrt(G + epsilon)

elif method == 'AdaDelta':
    grad = gradient(X_batch, y_batch, w)
    G = gamma * G + (1 - gamma)* np.square(grad)
    dw = -np.sqrt(delta + epsilon) / np.sqrt(G + epsilon) * grad
    w += dw
    delta = gamma * delta + (1 - gamma) * np.square(delta)

elif method == 'Adam':
    t = i + 1
    grad = gradient(X_batch, y_batch, w)
    m = beta * m + (1 - beta) * grad
    G = gamma * G + (1 - gamma) * np.square(grad)
    alpha = learning_rate * 100 * np.sqrt(1 - gamma ** t) / (1 - beta ** t)
    w -= alpha * m / np.sqrt(G + epsilon)
```

Fig. 1. python implementation

2) Logistic Regression and Linear SVM Initialization:

- weights: vector filled with ones in Logistic Regresssion and zeros in Linear SVM
- V: vector filled with zeros
- G: vector filled with zeros
- m: vector filled with zeros

3) Logistic Regression and Linear SVM Parameters:

- NAG: learning rate = 0.0001, γ = 0.9
- RMSProp: learning rate = 0.01, γ = 0.9, ϵ = 1e-5
- AdaDelta: γ = 0.9, ϵ = 1e-5
- Adam: learning rate = 0.01, γ = 0.9, ϵ = 1e-5, β = 0.9

4) Logistic Regression Result: shown in Figure 2

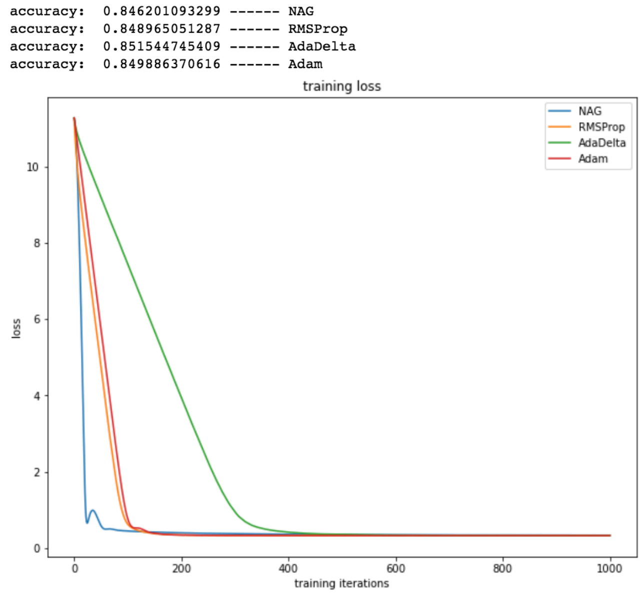

Fig. 2. Logistic Regression

5) Linear SVM Result: shown in Figure 3

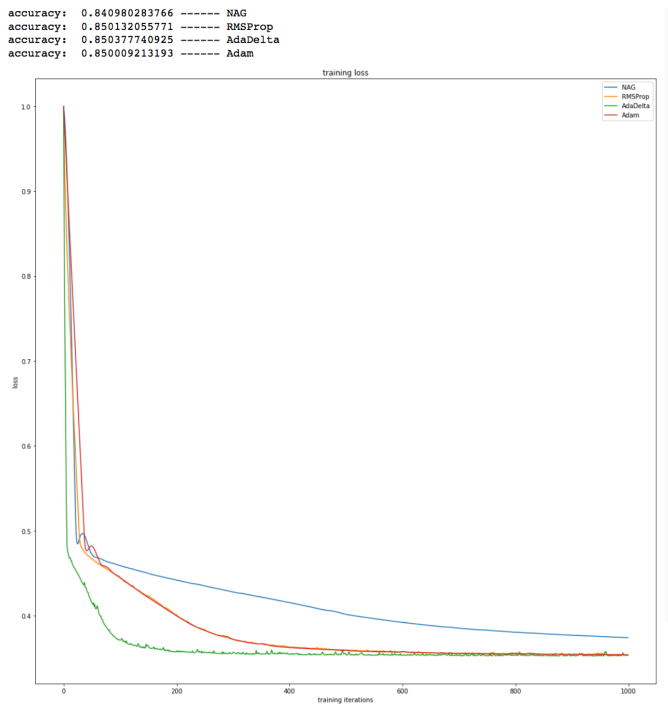

Fig. 3. Linear SVM

IV. Conclusion

1) You have to be very careful when implementing algorithms or you may misunderstanding what those equations means.
2) No matter how you initialize the weights, after gradient descent they will be steady.
3) All the models this paper implements perform similarly to the models [1] and [2] in python package sklearn.
4) RMSProp acts like Adam. NAG didn't fully converge because of the iteration rounds.
5) Despite of all these improvements, SGD still performs well.
6) Learning rates vary from different methods. They should be well optimized, but I simply times 100 and it works.

References

[1] Logistic regression.
[2] Linearsvc.