The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Jin Cao

Supervisor:
Mingkui Tan

Student ID:
201530611111

Grade:
Undergraduate

December 13, 2017

Logistic Regression, Linear Classification and Gradient Descent

Abstract—In this experiment, we implement the logistic regression and linear classification, we use mini-batch stochastic gradient descent to update the parameters, we use four optimization methods which are NAG, RMSProp, AdaDelta, Adam, and we will show the results of this experiment.

I. Introduction

In this experiment, we implement the logistic regression and linear classification. Through the experiment, we can compare the differences between gradient descent and stochastic gradient descent. We also need to compare and understand the differences between logistic regression and linear classification. Through this experiment, you can also see the differences and relationships between logistic regression and linear classification. In this experiment, we expects the loss of different optimization methods are similar. Because optimization methods are methods of updating parameters, different optimization methods may result in different convergence times, but the value of loss should be same.

II. Methods and Theory

In this experiments, we use four optimization method to optimize the parameters.We will give some introduction of this experiment.

1.The Chosen Methods:

(1)NAG
(2)RMSProp
(3)AdaDelta
(4)Adam

2.The Related Theroies:

(1)NAG

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1} - \gamma \mathbf{v}_{t-1})$$
$$\mathbf{v}_t \leftarrow \gamma \mathbf{v}_{t-1} + \eta \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \mathbf{v}_t$$

(2)RMSProp

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t$$

(3)AdaDelta:

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\Delta\boldsymbol{\theta}_t \leftarrow -\frac{\sqrt{\Delta_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}_t$$
$$\Delta_t \leftarrow \gamma\Delta_{t-1} + (1-\gamma)\Delta\boldsymbol{\theta}_t \odot \Delta\boldsymbol{\theta}_t$$

(4)Adam

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\alpha \leftarrow \eta \frac{\sqrt{1-\gamma^t}}{1-\beta^t}$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\mathbf{m}_t}{\sqrt{G_t + \epsilon}}$$

(5) the loss of Logistic regression:

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} \log\left(1 + e^{-yiW^T X_i}\right) + \frac{\lambda}{2}||w||_2^2$$

(6) the update of W in Logistic Regression:

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

(6) the loss of Linear Classification

$$\frac{1}{2}||w||_2^2 + \frac{C}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(W^T X_i + b))$$

(7) the update of W in Linear Classification:

$$\mathbf{w} := \mathbf{w} - \eta\nabla_{\mathbf{w}} L(\mathbf{w}, b) = \mathbf{w} - \eta\frac{1}{n}\sum_{i=1}^{n}\nabla_{\mathbf{w}} L_i(\mathbf{w}, b)$$

3. The Derivation Process:

(1) the derivation of updating W in Linear Classification:

The hinge loss is $\xi_i = \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$

Let $g_{\mathbf{w}}(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial \mathbf{w}}$

if $1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0$:

$$\begin{aligned} g_{\mathbf{w}}(\mathbf{x}_i) &= \frac{\partial(-y_i(\mathbf{w}^\top \mathbf{x}_i + b))}{\partial \mathbf{w}} \\ &= -\frac{\partial(y_i \mathbf{w}^\top \mathbf{x}_i)}{\partial \mathbf{w}} \\ &= -y_i \mathbf{x}_i \end{aligned}$$

if $1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0$:

$$g_{\mathbf{w}}(\mathbf{x}_i) = 0$$

so we have:

$$g_{\mathbf{w}}(\mathbf{x}_i) = \begin{cases} -y_i\mathbf{x}_i & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases}$$

Let $g_b(\mathbf{x}_i) = \frac{\partial \xi_i}{\partial b}$

$$g_b(\mathbf{x}_i) = \begin{cases} -y_i & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) >= 0 \\ 0 & 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b) < 0 \end{cases}$$

Optimization problem:

$$\begin{aligned} \min_{\mathbf{w},b} \; L(\mathbf{w}, b) &= \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b)) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{\|\mathbf{w}\|^2}{2} + C\max(0, 1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} L_i(\mathbf{w}, b) \end{aligned}$$

So we have:

$\nabla_w L_i(w, b) = w + C g_w(x_i)$

$\nabla_b L_i(w, b) = C g_b(x_i)$

III. Experiment

A. Dataset

Experiment uses a9a of LIBSVM Data, training set includes 32561 training samples and testing set includes 16281 testing samples, each training sample and testing sample has 123 features. The label of training set and testing set are 1 or -1.

B. Implementation

Initialization Parameters:

The parameters are initialized as follow:

1. logistic regression:

	NAG	RMSProp	AdaDelta	Adam
γ	0.9	0.9	0.999	0.999
ε	-	1e-8	1e-8	1e-8
η	0.01	0.003	-	0.005
β	-	-	-	0.9
V_t(vector)	0	-	-	-
G_t(vector)	-	0	0	0
Δ_t(vector)	-	-	0	-
m_t(vector)	-	-	-	0
W(vector)	0	0	0	0
λ	1	1	1	1

2.Linear Classification:

	NAG	RMSProp	AdaDelta	Adam
γ	0.9	0.95	0.9995	0.999
ε	-	1e-8	1e-8	1e-8
η	0.01	0.002	-	0.005
β	-	-	-	0.9
V_t(vector)	0	-	-	-
G_t(vector)	-	0	0	0
Δ_t(vector)	-	-	0	-
m_t(vector)	-	-	-	0
W(vector)	0	0	0	0
λ	1	1	1	1

Process:

The implementation of gradient, loss and four optimization methods are showing as follows:

1. logistic regression:

(1)Gradient:

```
def calGradient(W, X_t, y_t):
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    temp = np.zeros(W.shape)
    for i in range(y_samples.shape[0]):
        Xi = X_samples[i].reshape(1, X_samples.shape[1]).T
        yi = y_samples[i][0]

        temp += yi / (1 + math.exp(yi * np.dot(W.T, Xi)[0][0])) * Xi
    gradient = LAMBDA * W - 1 / sampleBatchNum * temp
    return gradient
```

Fig1.the gradient of logistic regression

(2)Loss:

```
def calLoss(W, X_t, y_t):
    loss = 0
    for index in range(X_t.shape[0]):
        Xi = X_t[index].reshape(1, X_t.shape[1]).T
        yi = y_t[index][0]
        loss += math.log(1 + math.exp(-yi * np.dot(W.T, Xi)[0][0]))
    loss = LAMBDA / 2 * np.dot(W.T, W)[0][0] + 1 / X_t.shape[0] * loss
    return loss
```

Fig2. loss

(3) NAG:

```
def calNAG(W, Vt, gt, eta, X_t, y_t):
    gamma = 0.9
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    temp = np.zeros(W.shape)
    W_ = W - gamma * Vt
    gt = calGradient(X_samples, y_samples, W_)
    Vt = gamma * Vt + eta * gt
    W = W - Vt
    return W, Vt, gt
```

Fig3. Implementation of NAG

(4) RMPSProp:

```
def calRMSProp(W, Gt, eta, X_t, y_t):
    gamma = 0.9
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    Gt = gamma * Gt + (1 - gamma) *(gt * gt)
    W = W - (eta / (np.sqrt(Gt + xigema))) * gt
    return W, Gt
```

Fig4. Implementation of RMSProp

(5) AdaDelta:

```
def calAdaDelta(W, Gt, Delta_t, X_t, y_t):
    gamma = 0.999
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    Gt = gamma * Gt + (1-gamma) * (gt * gt)
    Delta_W = -1 * (np.sqrt(Delta_t + xigema) / np.sqrt(Gt + xigema)) * gt
    W = W + Delta_W
    Delta_t = gamma * Delta_t + (1 - gamma) * (Delta_W * Delta_W)
    return W, Gt, Delta_t
```

Fig5. Implementation of AdaDelta

(6) Adam:

```
def calAdam(W, mt, Gt, eta, X_t, y_t):
    gamma = 0.999
    beta1 = 0.9
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    mt = beta1 * mt + (1 - beta1) * gt
    Gt = gamma * Gt + (1- gamma) * (gt * gt)
    alpha = eta * (np.sqrt(1 - gamma) / np.sqrt(1-beta1))
    W = W - alpha * mt /np.sqrt(Gt + xigema)
    return W, mt, Gt
```

Fig6. Implementation of Adam

2.Linear Classification:

(1)Gradient:

```
#计算梯度函数
def calGradient(X_t, y_t, W):
    temp = np.zeros(W.shape)
    for index in range(X_t.shape[0]):
        Yi = y_t[index][0]
        Xi = X_t[index].reshape(1,X_t.shape[1])
        hingeLoss = calHingeLoss(Yi, Xi, W)
        if hingeLoss > 0:
            temp = temp - Yi * Xi.T
        else:
            temp = temp + np.zeros((X_t.shape[1], 1))
    gradient = W + C / X_t.shape[0] * temp
    return gradient
```

Fig7.the gradient of linear classification

(2)Loss:

```
def calHingeLoss(Yi,Xi,W):
    middle = Yi * np.dot(W.T, Xi.T)
    result = 1 - middle[0][0]
    if result > 0:
        hingeLoss = result
    else:
        hingeLoss = 0
    return hingeLoss
```

Fig8. Hinge Loss

```
def calLoss(W, X_t, y_t):
    temp=0
    for index in range(y_t.shape[0]):
        Yi = y_t[index][0]
        Xi = X_t[index].reshape(1,X_t.shape[1])
        temp += calHingeLoss(Yi, Xi, W)
    temp = C * temp
    loss = 1/2 * np.dot(W.T,W)[0][0] + temp/float(y_t.shape[0])
    return loss
```

Fig9. Loss

(3) NAG:

```
def calNAG(W, Vt, gt, eta, X_t, y_t):
    gamma = 0.9
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    temp = np.zeros(W.shape)
    W_ = W - gamma * Vt
    gt = calGradient(X_samples, y_samples, W_)
    Vt = gamma * Vt + eta * gt
    W = W - Vt
    return W, Vt, gt
```

Fig10. Implementation of NAG

(4) RMPSProp:

```
def calRMSProp(W, Gt, eta, X_t, y_t):
    gamma = 0.9
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    Gt = gamma * Gt + (1 - gamma) *(gt * gt)
    W = W - (eta / (np.sqrt(Gt + xigema))) * gt
    return W, Gt
```

Fig11. Implementation of RMSProp

(5) AdaDelta:

```
def calAdaDelta(W, Gt, Delta_t, X_t, y_t):
    gamma = 0.999
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    Gt = gamma * Gt + (1-gamma) * (gt * gt)
    Delta_W = -1 * (np.sqrt(Delta_t + xigema) / np.sqrt(Gt + xigema)) * gt
    W = W + Delta_W
    Delta_t = gamma * Delta_t + (1 - gamma) * (Delta_W * Delta_W)
    return W, Gt, Delta_t
```

Fig12. Implementation of AdaDelta

(6) Adam:

```
def calAdam(W, mt, Gt, eta, X_t, y_t):
    gamma = 0.999
    beta1 = 0.9
    xigema = 1e-8
    X_samples, y_samples = samples_selection(X_t, y_t, sampleBatchNum)
    gt = calGradient(X_samples, y_samples, W)
    mt = beta1 * mt + (1 - beta1) * gt
    Gt = gamma * Gt + (1- gamma) * (gt * gt)
    alpha = eta * (np.sqrt(1 - gamma) / np.sqrt(1-beta1))
    W = W - alpha * mt /np.sqrt(Gt + xigema)
    return W, mt, Gt
```

Fig13. Implementation of Adam

Results:

1. Logistic Regression:

In the logistic regression, by setting the batch number 100 and iteration time 200, we get the curve of loss as follow:

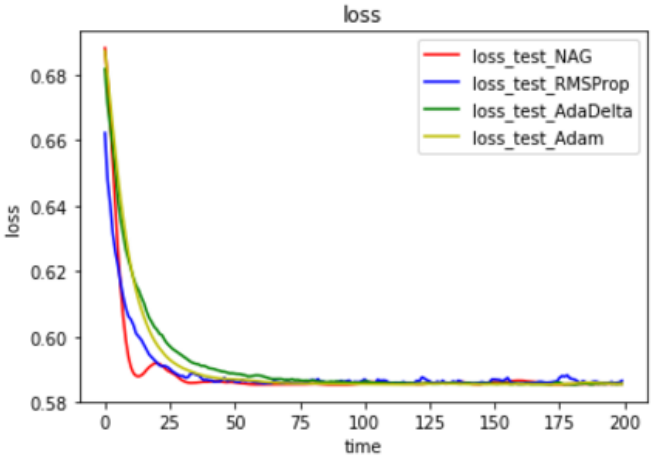

Fig14. The loss curve of logistic regression

2. Linear Classification:

In the logistic regression, by setting the batch number 100 and iteration time 200, we get the curve of loss as follow:

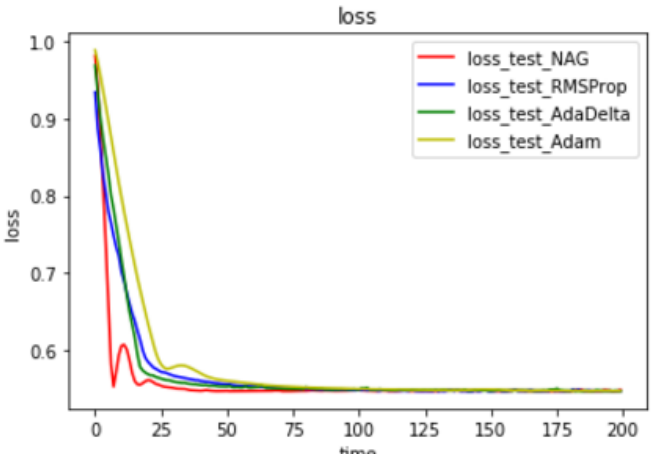

Fig15. The loss curve of logistic regression

IV. CONCLUSION

This experiment achieved logistic regression and linear classification, four different optimization methods are used for logistic regression and linear classification, the final convergence of these four methods is consistent. However, the convergence rates of the four optimization methods are different. Therefore, we can think that the final convergence value of different optimization methods is the same, but different optimization methods may result in different convergence rates due to different optimization steps. In addition, the differences between stochastic gradient descent and gradient descent is that convergence rate is different, and the computational complexity between them is different. Because the gradient descent use all the samples in training set for calculation, but stochastic gradient descent use part of the samples in training set for calculation, so the computational complexity of gradient descent is even greater. Besides, the shocks caused by the gradient and stochastic are also different, the reason is that the gradient descent use all the samples in training set for calculation, but stochastic gradient descent use part of the samples in training set for calculation, so the shock of stochastic gradient descent is greater. In this experiment, logistic regression and linear classification can both be used for classification, and both of them can achieve good classification results. Through this experiment, I understand the difference between gradient descent and stochastic gradient descent, and I understand the logistic regression and linear classification.