华南理工大学

South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Yingting Huang

Supervisor:
Qingyao Wu

Student ID：
201720145136

Grade:
Graduate

December 13, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract—
To further understand of Logistic Regression, Linear Classification and Stochastic Gradient Descent, I conduct some experiments under large scale dataset. During the process of optimization and adjusting parameters, I relate theory with practice. Now I am more be acquainted with the models as well as the Gradient descent optimization algorithms.

Key word: Logistic Regression, Linear Classification, Stochastic Gradient Descent

I. INTRODUCTION

SGD usually achieves to find a minimum, but it might take significantly longer than with some of the optimizers, is much more reliant on a robust initialization and annealing schedule, and may get stuck in saddle points rather than local minima. Consequently, if we care about fast convergence and train a deep or complex neural network, we are supposed to choose one of the adaptive learning rate methods.

In brief, RMSprop is an extension of Adagrad that deals with its radically diminishing learning rates. It is identical to Adadelta, except that Adadelta uses the RMS of parameter updates in the numinator update rule. Adam, finally, adds bias-correction and momentum to RMSprop. Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances.

II. METHODS AND THEORY

A. The selected loss function and its derivatives

1. Logistic Regression and Stochastic Gradient Descent

1) Loss Function：

$$L(a,y) = -\frac{1}{N}(\quad ylog a + (1-y)log(1-a)\quad) \tag{1}$$

Where

$$\hat{y} = a = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$z = W^T X \tag{2,3}$$

2) Gradient Matrix：

$$\frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z} = a - y$$

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial W} = X(a-y)^T \tag{4-6}$$

2. Linear Classification and Stochastic Gradient Descent

1) Loss Function：

$$L = \frac{1}{N}\sum_i \sum_{j\neq y_i}[max(0, f(x_i;W)_j - f(x_i;W)_{y_i} + \Delta] + \lambda \sum_k \sum_l W_{k,l}^2 \tag{7}$$

2) Gradient Matrix：

$$\frac{\partial L}{\partial w_{y_i}} = -(\sum_{j\neq y_i} 1(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0))x_i \qquad (j = y_i)$$

$$\frac{\partial L}{\partial w_j} = 1(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0))x_i \qquad (j \neq y_i) \tag{8-9}$$

Where l(x) is an indicator function：

$$l(x == T) = 1$$

$$l(x == F) = 0 \tag{10-11}$$

3. Gradient descent optimization algorithms

3.1 NAG

$$d_t = \gamma * d_{t-1} + g_{t-1} + \gamma * (g_{t-1} - g_{t-2})$$

$$\Delta W = -\eta * d_t$$

$$W_t = W_{t-1} + \Delta W \tag{12-15}$$

3.2 RMSProp

$$n_t = \gamma * n_{t-1} + (1-\gamma) * g_t^2$$

$$\Delta W = \frac{-\eta}{\sqrt{n_t + \epsilon}} * g_t$$

$$W = W + \Delta W \tag{16-19}$$

3.3 AdaDelta

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2$$

$$\Delta W_t = -\frac{\sqrt{E[\Delta W^2]_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

$$W = W + \Delta W$$

$$E[\Delta W_t^2]_t = \gamma E[\Delta W_t^2]_{t-1} + (1-\gamma)\Delta W_t^2 \quad (20\text{-}24)$$

3.4 Adam

$$m_t = \mu * m_{t-1} + (1-\mu) * g_t$$

$$n_t = \nu * n_{t-1} + (1-\nu) * g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\mu^t + \epsilon}$$

$$\hat{n}_t = \frac{n_t}{1-\nu^t + \epsilon}$$

$$\Delta W = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t + \epsilon}} * \eta \quad (25\text{-}29)$$

III. EXPERIMENT

A. Data sets and data analysis

In this Experiment, we use a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

B. Experimental steps

1. Logistic Regression and Stochastic Gradient Descent

1) Load the training set and validation set.
2) Initialize logistic regression model parameters.
3) Select the loss function and calculate its derivation.
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.

2. Linear Classification and Stochastic Gradient Descent

1) Load the training set and validation set.
2) Initialize SVM model parameters.
3) Select the loss function and calculate its derivation.
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.

C. The initialization method of model parameters

1. Logistic Regression and Stochastic Gradient Descent

All parameters are set into zero in the linear model.

2. Linear Classification and Stochastic Gradient Descent

All parameters are set into zero in the SVM model.

D. Experimental results and curve

1 Linear Regression and Gradient Descent

1.1 NAG

1) Hyper-parameter selection
- ✓ *threshold*, marks the sample whose predict scores greater than the threshold as positive, on the contrary as negative, is set to 0.5
- ✓ *eta*, the learning rate, is set to 0.05
- ✓ *maxIterations*, the maximum number of iterations, is set to 10000
- ✓ *gamma*, the momentum factor, is set to 0.9

2) Predicted Results (Best Results):
- ✓ *test_accuracy_NAG*, the accuracy of test data, is equal to 0.7637737239727289

3) Loss curve:

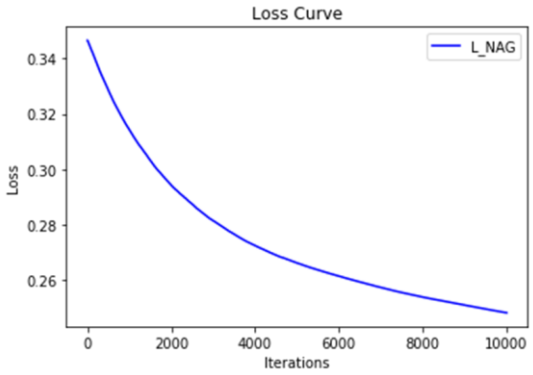

Fig 1.1

1.2 RMSProp

1) Hyper-parameter selection:
- ✓ *threshold*, marks the sample whose predict scores greater than the threshold as positive, on the contrary as negative, is set to 0.5
- ✓ *eta*, the learning rate, is set to 0.05
- ✓ *maxIterations*, the maximum number of iterations, is set to 10000
- ✓ *gamma*, the decay factor, is set to 0.9
- ✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 0.001

2) Predicted Results (Best Results):

✓ *test_accuracy_RMSProp*, the accuracy of test data, is equal to 0.7841041705054972

3) Loss curve:

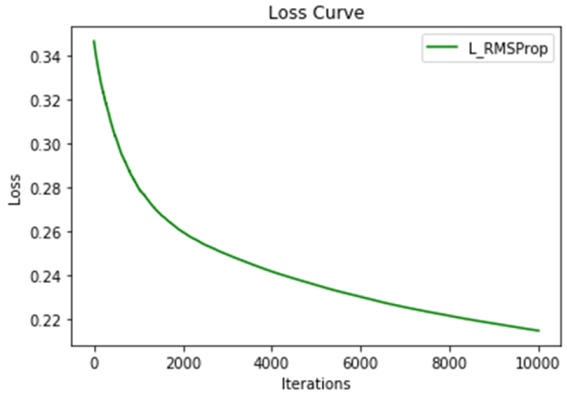

Fig 1.2

1.3 AdaDelta

1) Hyper-parameter selection:

✓ *threshold*, marks the sample whose predict scores greater than the threshold as positive, on the contrary as negative, is set to 0.5

✓ *maxIterations*, the maximum number of iterations, is set to 10000

✓ *gamma*, the decay factor, is set to 0.9

✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 0.001

2) Predicted Results (Best Results):

✓ *test_accuracy_AdaDelta*, the accuracy of test data, is equal to 0.7651249923223389

3) Loss curve:

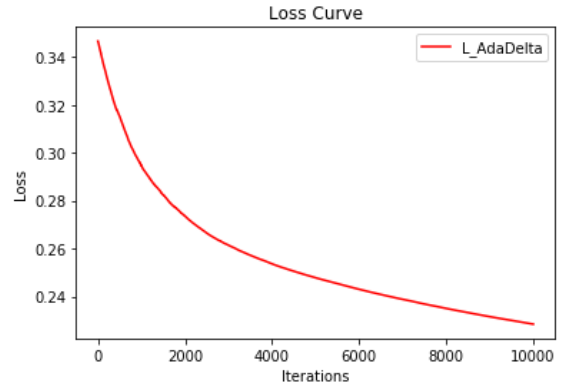

Fig 1.3

1.4 Adam

1) Hyper-parameter selection:

✓ *threshold*, marks the sample whose predict scores greater than the threshold as positive, on the contrary as negative, is set to 0.5

✓ *maxIterations*, the maximum number of iterations, is set to 10000

✓ *mu*, the decay factor of m (the first rank moment estimation of gradient), is set to 0.9

✓ *v*, the decay factor of n (the second rank moment estimation of gradient), is set to 0.9

✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 0.001

2) Predicted Results (Best Results):

✓ *test_accuracy_Adam*, the accuracy of test data, is equal to 0.8501934770591487

3) Loss curve:

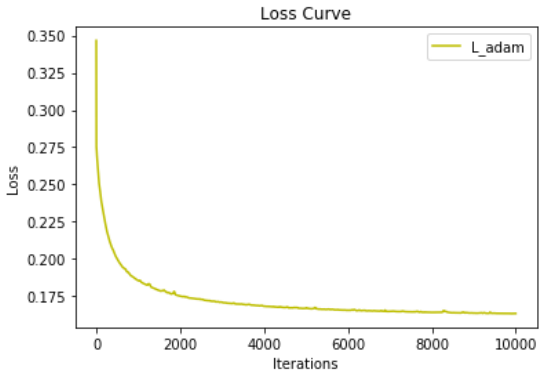

Fig 1.4

1.5 Compare the loss curve

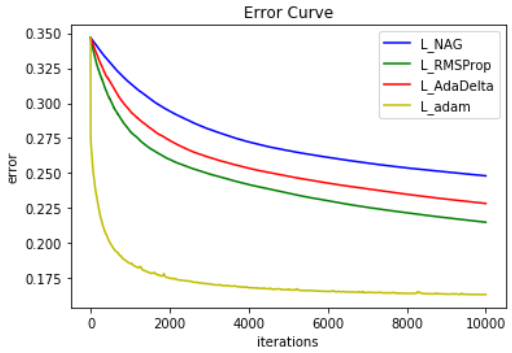

Fig 1.5

2 Linear Classification and Stochastic Gradient Descent

2.1 NAG

1) Hyper-parameter selection

✓ *eta*, the learning rate, is set to 0.05

✓ *maxIterations*, the maximum number of iterations, is set to 200

✓ *gamma*, the momentum factor, is set to 0.9

2) Predicted Results (Best Results):

✓ *test_accuracy_NAG*, the accuracy of test data, is equal to 0.7646950433020084

3) Loss curve:

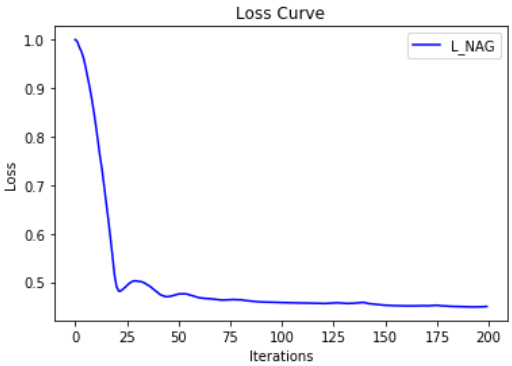

Fig 2.1

2.2 RMSProp

1) Hyper-parameter selection:
- ✓ *eta*, the learning rate, is set to 0.001
- ✓ *maxIterations*, the maximum number of iterations, is set to 200
- ✓ *gamma*, the decay factor, is set to 0.9
- ✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 0.001

2) Predicted Results (Best Results):
- ✓ *test_accuracy_RMSProp*, the accuracy of test data, is equal to 0.7637737239727289

3) Loss curve:

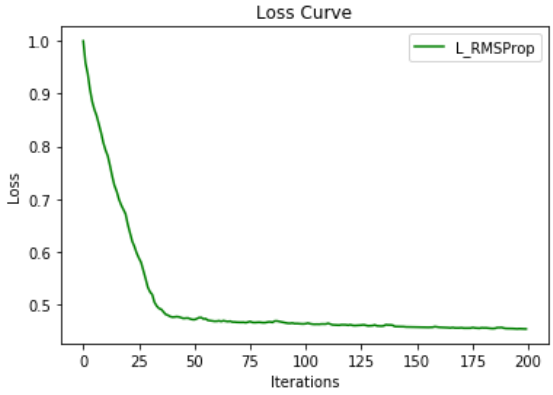

Fig 2.2

2.3 AdaDelta

1) Hyper-parameter selection:
- ✓ *maxIterations*, the maximum number of iterations, is set to 200
- ✓ *gamma*, the decay factor, is set to 0.8
- ✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 1e-6

2) Predicted Results (Best Results):
- ✓ *test_accuracy_AdaDelta*, the accuracy of test data, is equal to 0.7976782752902156

3) Loss curve:

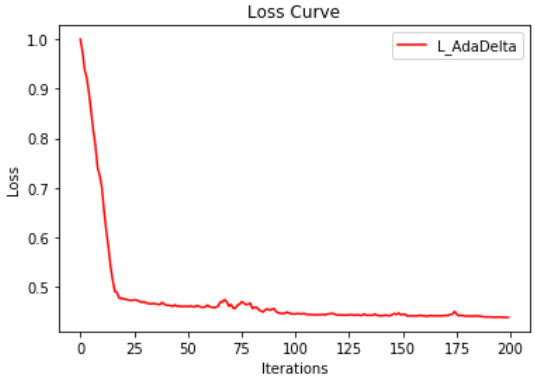

Fig 2.3

2.4 Adam

1) Hyper-parameter selection:
- ✓ *maxIterations*, the maximum number of iterations, is set to 200
- ✓ *mu*, the decay factor of m (the first rank moment estimation of gradient), is set to 0.9
- ✓ *v*, the decay factor of n (the second rank moment estimation of gradient), is set to 0.9
- ✓ *epsilon*, used to prevent the denominator equaling to 0, is set to 0.001

2) Predicted Results (Best Results):
- ✓ *test_accuracy_Adam*, the accuracy of test data, is equal to 0.7679503715987961

3) Loss curve:

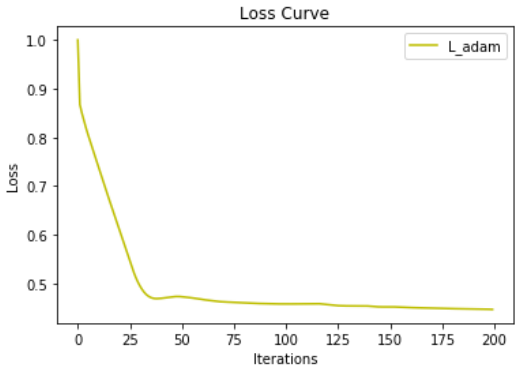

Fig 2.4

2.5 Compare the loss curve

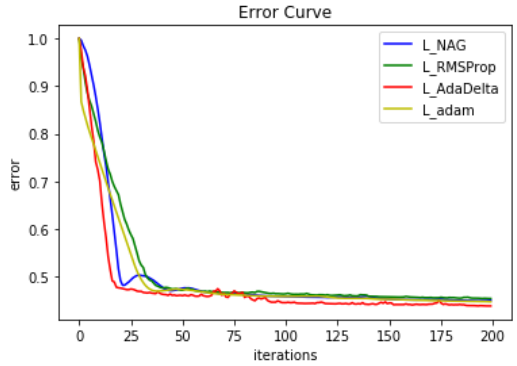

Fig 2.5

F. Results analysis

1 Logistic Regression and Stochastic Gradient Descent

From the predicted results in D.1, we can see that all the accuracies are high, which mean the predicting effect of the model is good.

From the loss curve (fig 1.1- fig 1.4), the loss converge to a small number nears zero with the number of iterations. That is to say, the models we have trained are robust.

In addition, Adam, with high convergence rate as well as accuracy, is most effective method among all the algorithms I used (fig 1.5). And I don't need to tune the learning rate but likely achieve the best results with the default value. Insofar, Adam might be the best overall choice.

2 Linear Classification and Stochastic Gradient Descent

From the predicted results in D.2, we can see that all the accuracies are high, which mean the predicting effect of the model is good.

From the loss curve (fig 2.1 – fig 2.4), the loss converge to a small number nears zero with the number of iterations. That is to say, the models we have trained are robust.

In addition, Adam, with high convergence rate as well as

accuracy, is most effective method among all the algorithms I used (fig 2.5). And I don't need to tune the learning rate but likely achieve the best results with the default value. Insofar, Adam might be the best overall choice.

IV. CONCLUSION

In the end, I want to talk about the similarities and differences between logistic regression and linear classification first.

As far as I am concerned, both of them are classification methods with linear decision boundaries, and can be "kernelized". But they're motivated completely differently. They differ in the loss function — SVM minimizes hinge loss while logistic regression minimizes logistic loss.

The basic idea of logistic regression is to adapt linear regression so that it estimates the probability a new entry falls in a class. The linear decision boundary is simply a consequence of the structure of the regression function and the use of a threshold in the function to classify. So logistic regression can be integrated into other probabilistic frameworks much more seamlessly than SVMs.

The decision boundary is much more important for Linear SVM's - the whole goal is to place a linear boundary in a smart way. There isn't a probabilistic interpretation of individual classifications, at least not in the original formulation.

Like with any linear regression, every training point has a certain influence on the estimated logistic regression function. SVM's are actually formulated so that only points near the decision boundary really make a difference. Points that are "obvious" have no effect on the decision boundary. This can be very different from logistic regression, as "obvious" points can sometimes be very influential.

Besides, I want to talk about the gradient descent optimization algorithms we used in this experiment. These algorithms are widely used by the deep learning community. Therefore, I think it is important to summarize these meaningful algorithms.

As far as I am concerned, SGD usually achieves to find a minimum, but it might take significantly longer than with some of the optimizers, is much more reliant on a robust initialization and annealing schedule, and may get stuck in saddle points rather than local minima. Consequently, if we care about fast convergence and train a deep or complex neural network, we are supposed to choose one of the adaptive learning rate methods.

So, which optimizer should you now use? If your input data is sparse, then you likely achieve the best results using one of the adaptive learning-rate methods. An additional benefit is that you won't need to tune the learning rate but likely achieve the best results with the default value.

In summary, to train a Neural Network:

1) Gradient check your implementation with a small batch of data and be aware of the pitfalls.
2) As a sanity check, make sure your initial loss is reasonable, and that you can achieve 100% training accuracy on a very small portion of the data
3) During training, monitor the loss, the training/validation accuracy, and if you're feeling fancier, the magnitude of updates in relation to parameter values (it should be ~1e-3), and when dealing with ConvNets, the first-layer weights.
4) The two recommended updates to use are either SGD+Nesterov Momentum or Adam.
5) Decay your learning rate over the period of the training. For example, halve the learning rate after a fixed number of epochs, or whenever the validation accuracy tops off.
6) Search for good hyperparameters with random search (not grid search). Stage your search from coarse (wide hyperparameter ranges, training only for 1-5 epochs), to fine (narrower rangers, training for many more epochs)
7) Form model ensembles for extra performance