The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Jiale Deng

Supervisor:
Mingkui Tan

Student ID：
201530611371

Grade:
Undergraduate

December 15, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent.

Abstract—realizing logistic regression and linear classification with Stochastic Gradient Descent and using four optimized me —thods (NAG, RMSProp, AdaDelta and Adam) to updating the model parameters.

I. INTRODUCTION

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.

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.

II. METHODS AND THEORY

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 *LNAG*，*LRMSProp*，*LAdaDelta* and *LAdam*.
7. Repeate step 4 to 6 for several times, and **drawing graph of** *LNAG*，*LRMSProp*，*LAdaDelta* **and** *LAdam* **with the number of iterations**.

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 *LNAG*，*LRMSProp*，*LAdaDelta* and *LAdam*.
7. Repeate step 4 to 6 for several times, and **drawing graph of** *LNAG*，*LRMSProp*，*LAdaDelta* **and** *LAdam* **with the number of iterations**.

Selection of validation (hold-out, cross-validation, k-folds cross-validation, etc.): Both are hold-out.

The initialization method of model parameters: All parameters are set into zero in both experiment.

The selected loss function and its derivatives:

Logistic Regression and Stochastic Gradient Descent

Loss function:

$$\mathrm{J}(w) = \frac{1}{n}\sum_{i=1}^{n}[y_i log h_w(x_i) + (1-y_i)\log(1-h_w(x_i))]$$

Derivatives:

$$\frac{\partial \mathrm{J}(w)}{\partial \mathrm{w}} = (h_w(x) - y)x$$

Or (quicker and regularization added)

$$J(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n}\log(1 + e^{-y_i \cdot w^T x_i})$$

Linear Classification and Stochastic Gradient Descent

Loss function:

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

Derivatives:

$$\frac{\partial L_D(w,b)}{\partial \mathrm{w}} = w + C\sum_{i=1}^{n} g_w(x_i)$$

$$\frac{\partial L_D(w,b)}{\partial \mathrm{b}} = w + C\sum_{i=1}^{n} g_b(x_i)$$

And

$$g_w(x_i) = \begin{cases} -y_i x_i & 1 - y_i(w^T x_i + b) >= 0 \\ 0 & 1 - y_i(w^T x_i + b) < 0 \end{cases}$$

$$g_b(x_i) = \begin{cases} -y_i & 1 - y_i(w^T x_i + b) >= 0 \\ 0 & 1 - y_i(w^T x_i + b) < 0 \end{cases}$$

III. EXPERIMENT

Code:

Optimized Methods to Update Model Parameters

```
def NAG(parameters, gradients, eta=0.05, momentum=0.9):
    for i in range(para_num):
        next_v[i] = momentum * v[i] + eta * gradients[i,0]
        updates[i,0] = parameters[i,0] - momentum * next_v[i] -
eta * gradients[i,0]
    return updates

def RMSProp(parameters, gradients, eta=0.002,
momentum=0.9, epsilon=1e-8):
    for i in range(para_num):
        next_G[i] = momentum * G[i] + (1 - momentum) *
np.square(gradients[i,0])
        updates[i,0] = parameters[i,0] - eta * gradients[i,0] /
np.sqrt(next_G[i] + epsilon)
    return updates

def AdaDelta(parameters, gradients, momentum=0.95,
epsilon=1e-6):
    for i in range(para_num):
        next_G[i] = momentum * G[i] + (1 - momentum) *
np.square(gradients[i,0])
        next_dx[i] = np.sqrt(dx[i] + epsilon) / np.sqrt(next_G[i] +
epsilon)
        updates[i,0] =parameters[i,0] - next_dx[i] * gradients[i,0]
    return updates

def Adam(parameters, gradients, eta=0.002,
momentum=0.999, beta=0.9, epsilon=1e-8):
    for i in range(para_num):
        next_m[i] = beta * m[i] + (1 - beta) * gradients[i,0]
        next_G[i] = momentum * G[i] + (1 - momentum) *
np.square(gradients[i,0])
        updates[i,0] = parameters[i,0] - eta * np.sqrt(1 -
momentum) / (1 - beta) * next_m[i] / np.sqrt(next_G[i] +
epsilon)
        return updates
```

Logistic Regression and Stochastic Gradient Descent

```
#gradient descent
    idx = random.sample(range(0,n),batch_size)
    x_train_batch = x_train[idx]
    y_train_batch = y_train[idx]
    h = sigmoid(x_train_batch*W)
    deltaW = x_train_batch.T*(h-y_train_batch)
    W = W+eta*deltaW
        h = sigmoid(x_train*W)
        loss_train.append(-
        (1/x_train.shape[0])*(y_train*np.log(h)+(1-
        y_train)*np.log(1-h)).sum())

#Or a quicker way

        deltaW=-
        learning_rate*C*W+learning_rate/n*sum(y_train_batch.
        T*x_train_batch/(1+np.exp(np.multiply(y_train_batch,x_
        train_batch*W))))

        loss_train.append(C/2*np.linalg.norm(W)**2+1/n*np.lo
        g1p(np.exp((-y_train.T*x_train*W)[0,0])))
```

Linear Classification and Stochastic Gradient Descent

```
#gradient descent
    g = 1-np.multiply(y_train_batch,y_train_pred_batch)
    idx_1 = np.where(g>=0)[0]
    idx_2 = np.where(g<0)[0]
    deltaW_2 = -W.T
    deltaW_1 = C*y_train_batch.T*x_train_batch+deltaW_2
    deltaW =
eta*((deltaW_1*len(idx_1)+deltaW_2*len(idx_2))/batch_size)
    W = W-eta*deltaW.T
    y_train_pred = x_train*W
    g = 1-np.multiply(y_train,y_train_pred)
        loss_train.append(((np.linalg.norm(W))**2/2+C*sum(np
        .maximum(zeros((n,1)),g)))/n)
```

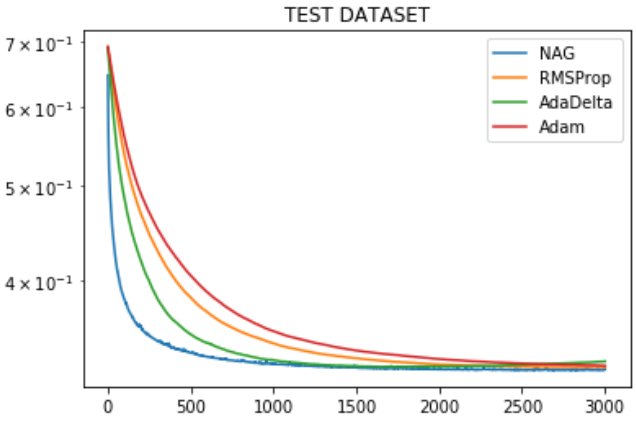
Experimental results and curve:

Logistic Regression and Stochastic Gradient Descent

A. Hyper-parameter selection (η , epoch, etc.):

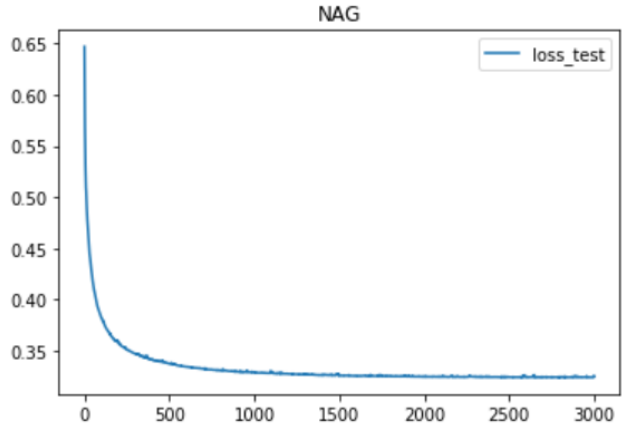
η= 0.0005（NAG） , η= 0.0002（RMSProp） , η= 0.0005（Adam）, epoch = 3000 .

B. Assessment Results (based on selected validation):

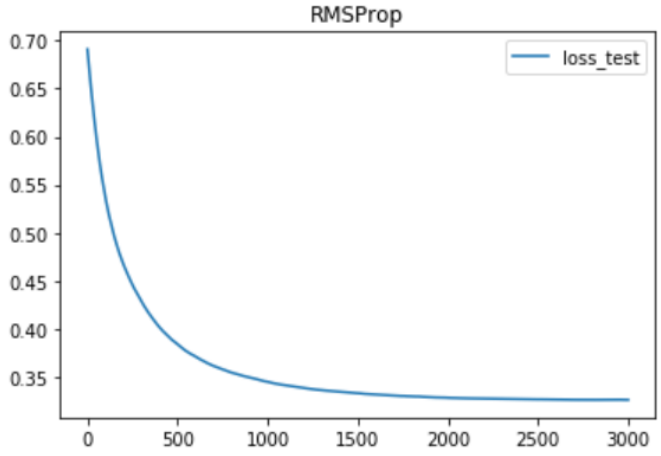


C. Loss curve:

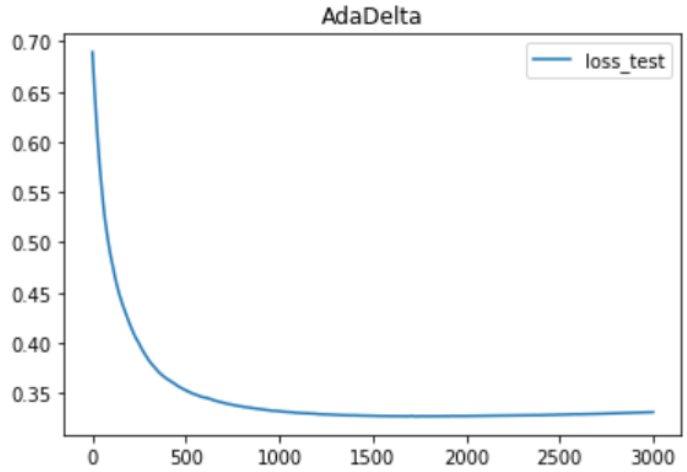
The accuracy: 0.8490264725753947



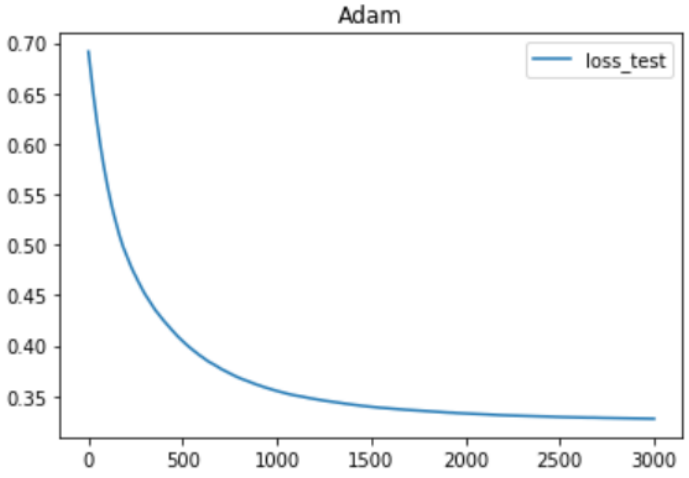
The accuracy: 0.8506234260794792



The accuracy: 0.8498863706160555



The accuracy: 0.8500092131932928

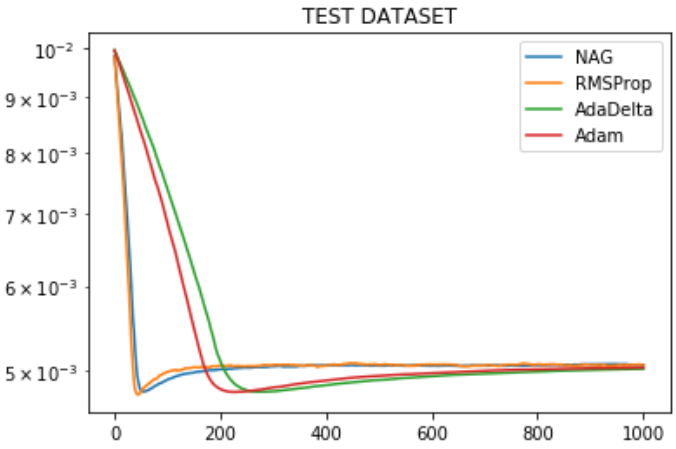


Linear Classification and Stochastic Gradient Descent

A. Hyper-parameter selection (η , epoch, etc.):

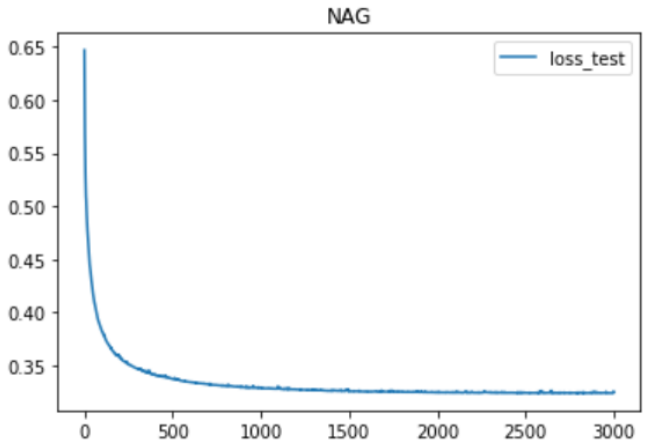
η= 0.05（NAG） , η= 0.002（RMSProp） ,
η= 0.002（Adam）, epoch = 3000 , C = 0.01.

B. Assessment Results (based on selected validation):

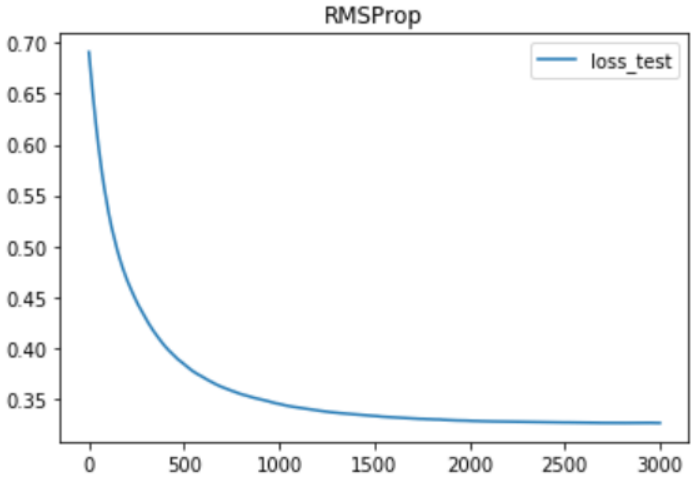


C. Loss curve:

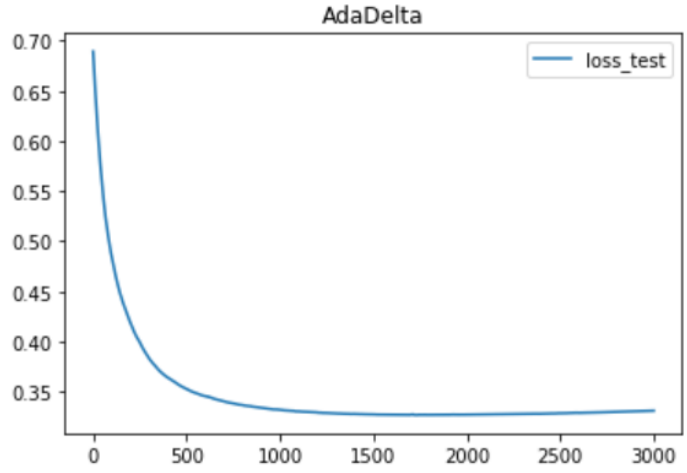
The accuracy: 0.8490264725753947



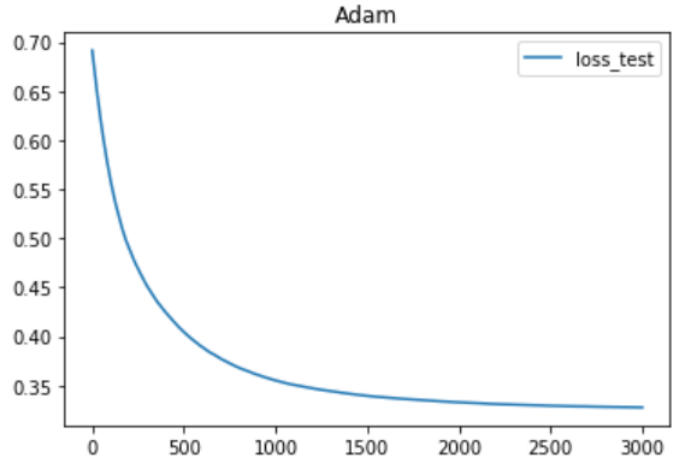
The accuracy: 0.8506234260794792



The accuracy: 0.8498863706160555

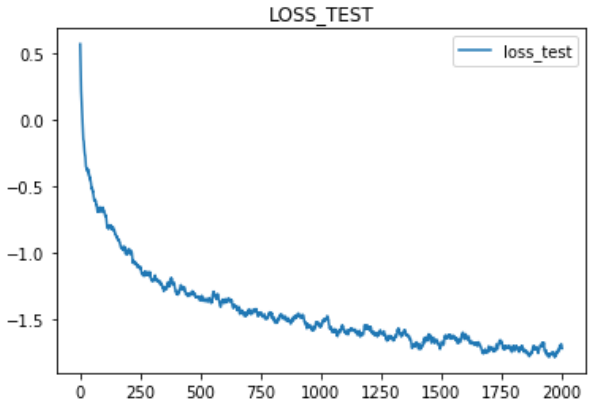


The accuracy: 0.8500092131932928

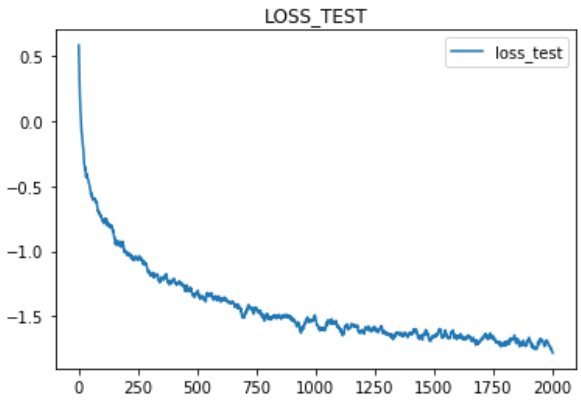


Results analysis:

With the increase of iterations, both two loss lines reduces rapidly initially and slope becomes less. Finally, the slope shock around zero value and the loss lines seem to be flat.
Because of using mini-batch sgd, the curves may shock frequently, and with the increase of batch size, the curves become more smooth but the shock still exists:



sgd,batch_size=64



sgd,batch_size=128

Besides, there may be a problem in my code: the curves decrease initially but then increase before becoming steady.

IV. CONCLUSION

Similarities and differences between logistic regression and linear classification：

Similarities:

(1) The methods to update the model parameters are the same.

Differences:

(1) The selected loss function and its gradient.
(2) The loss in logistic classification is easier to diverge , hence smaller hyper parameters than that in linear classification.

Summary:

In this experiment, I have learned the coding of logistic classification and knew the difference between two classification method. Besides, I also have learned four Optimized Methods to Update Model Parameters.