華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Chen Han

Supervisor:
Mingkui Tan

Student ID:
201936380086

Grade:
Class 1, Grade 2019

October 7, 2021

Logistic Regression and Support Vector Machine

***Abstract*—Further understand the principles of Logistic Regression and Support Vector Machine and practice on larger data.**

I. Introduction

IN order to further understand the conception of Logistic Regression and Support Vector Machine, we compare differences and relationships between Logistic regression and linear classification. We use dataset a9a of LIBSVM Data. Using SGD and Adam to optimize the efficiency.

II. Methods and Theory

A. Logistic Regression and Batch Stochastic Gradient Descent

We use probability function below:

$$\begin{aligned} h_w(x) &= g(z) \\ &= g(\sum_{i=1}^{m} w_i x_i) \\ &= g(w^T x) \end{aligned}$$

Here, $z = w^T x$, $g(\cdot)$ is a logistic function:

$$g(z) = \frac{1}{1+e^{-z}}$$

The function is a continuous function. If $z \to +\infty$, then $g(z) \to 1$ and if $z \to -\infty$, then $g(z) \to 0$.
We define the threshold at 0.5, if $g(W^T X_i)$>0.5 , we thought it's positive, or it's negative.
We would like to get:

$$\begin{aligned} \max \prod_{i=1}^{n} &\iff \max \log(\prod_{i=1}^{n} P(y_i|x_i)) \\ &\iff \min -\frac{1}{n}\sum_{i=1}^{n} \log P(y_i|x_i) \\ &\iff \min \frac{1}{n}\sum_{i=1}^{n} \frac{1}{\log P(y_i|x_i)} \\ &\equiv \quad min \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{g(y_i w^T x_i)} \\ &\equiv \quad min \frac{1}{n}\sum_{i=1}^{n} \log(1+e^{-y_i w^T x_i}) \\ &\equiv \quad \min \mathcal{L}(W) \end{aligned}$$

So we have loss function below:

$$\min \mathcal{L}oss(W) = \frac{1}{n}\sum_{i=1}^{n} log(1+e^{-y_i W^T X_i})$$

In order to avoid overfitting, we rewrite this equation by regular item $\frac{\lambda}{2} \parallel W \parallel^2$, then we have:

$$\min \mathcal{J}(W) = \frac{1}{n}\sum_{i=1}^{n} log(1+e^{-y_i W^T X_i}) + \frac{\lambda}{2} \parallel W \parallel^2$$

We calculate gradient with formula below:

$$\frac{\partial \mathcal{J}(W)}{\partial W} = -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i X_i e^{-y_i W^T X_i}}{1+e^{-y_i W^T X_i}} + \lambda W$$

B. Support Vector Machine

We want to optimize the problem:

$$\min_{w,b,\xi_i} \frac{\parallel w \parallel^2}{2} + \frac{C}{n}\sum_{i=1} n\xi_i$$

$$s.t. \qquad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \forall \xi_i \geq 0, i = 1,2,\cdots,n$$

where C is a hyperparameter:

-small C makes constraints easy to be ignored
-large C makes constraints hard to be ignored

$$\begin{aligned} &if \quad 1 - y_i(w^T x_i + b) \geq 0, \quad \xi_i = 1 - y_i(w^T x_i + b) \\ &if \quad 1 - y_i(w^T x_i + b) \leq 0, \quad \xi_i = 0 \end{aligned}$$

So, we rewrite ξ_i as:

$$\xi_i = \max(0, 1 - y_i(w^T x_i + b))$$

The optimization problem becomes:

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

Then we have the loss function:

$$\min_{W,b} \mathcal{L}(W,b) = \frac{\parallel W \parallel^2}{2} + \frac{C}{n}\sum_{i=1}^{n} max(0, 1 - y_i(W^T X_i + b))$$

To minimize a loss function $L(w,b)$, use the iterative update:

$$\begin{aligned} w &= w - \eta \bigtriangledown_w L(w,b) \\ b &= b - \eta \bigtriangledown_b L(w,b) \end{aligned} \tag{1}$$

where η is the learning rate.
We define threshold value at 0, if $\{(W^T X_i + b)$>0, it's positive, or it's negative.
Then we have gradient descent with batch algorithm that uses all examples:

$$\frac{\partial \mathcal{L}(W,b)}{\partial W} = \begin{cases} -y_i X_i, & 1 - y_i(W^T X_i + b) \geq 0 \\ 0, & 1 - y_i(W^T X_i + b) < 0 \end{cases}$$

$$\frac{\partial \mathcal{L}(W,b)}{\partial b} = \begin{cases} -y_i, & 1 - y_i(W^T X_i + b) \geq 0 \\ 0, & 1 - y_i(W^T X_i + b) < 0 \end{cases}$$

III. Experiments

A. Dataset

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

B. Implementation

1. Load the experiment data. Using load_svmlight_file function in sklearn library, adding one column by yourself or specify the n_features to be 123 when using the function to avoid values of last column all zero to be ignored.
2. write Loss-calculating function for Logistic Regression.
3. Write sigmoid function.
4. Write validation calculating function for Logistic Regression.
5. Calculating gradient with formula above for Logistic Regression.
6. Write Logistic Regression function.

```python
def Logistic_Regression(X, Y, learning_rate, lamda, epoch, batch):
    loss = []
    acc = []
    W = np.random.rand(X.shape[1], 1)
    for i in tqdm(range(epoch), desc="Training pass: ", leave=True):
        bat = np.random.choice(X.shape[0], batch)
        X_batch = X[bat]
        Y_batch = Y[bat]
        gradient = calc_gradient_A(X_batch, Y_batch, W, lamda)
        W = W - learning_rate*gradient
        loss.append(calc_loss_A(X, Y, W))
        acc.append(calc_validation_A(W))
    return W, loss, acc
```

7. Write Logistic Regression function with Adam optimization.

```python
def Logistic_Regression_Adam(X, Y, learning_rate, lamda, epoch, batch, eps=1e-8, beta1=0.9, beta2=0.999):
    loss = []
    acc = []
    m = 0
    v = 0
    W = np.random.rand(X.shape[1], 1)
    for i in tqdm(range(epoch), desc="Training pass: ", leave=True):
        bat = np.random.choice(X.shape[0], batch)
        X_batch = X[bat]
        Y_batch = Y[bat]
        gradient = calc_gradient_A(X_batch, Y_batch, W, lamda)
        m = beta1 * m + (1 - beta1) * gradient
        v = beta2 * v + (1 - beta2) * (gradient ** 2)
        W = W - learning_rate * m / (np.sqrt(v) + eps)
        loss.append(calc_loss_A(X, Y, W))
        acc.append(calc_validation_A(W))
    return W, loss, acc
```

8. Define the function to calculate loss for SVM.
9. Write validation calculating function for SVM.
10. Calculating gradient with formula above for SVM.
11. Define the SVM classifier with Stochastic Gradient Descent.

```python
def SVM_classifier(X, Y, c, epoch, batch, learning_rate):
    loss = []
    acc = []
    W = np.random.rand(X.shape[1], 1)
    b = 0
    for i in tqdm(range(epoch), desc="Training pass: ", leave=True):
        bat = np.random.choice(X.shape[0], batch)
        X_batch = X[bat]
        Y_batch = Y[bat]
        gradient_W, gradient_b = calc_gradient_B(X_batch, Y_batch, W, b, c)
        W = W - learning_rate * gradient_W
        b = b - learning_rate * gradient_b
        loss.append(calc_loss_B(X, Y, W, b, c))
        acc.append(calc_validation_B(W, b))
    return W, loss, acc
```

12. Rewrite SVM classifier with Adam.

```python
def SVM_classifier_Adam(X, Y, c, epoch, batch, learning_rate, eps=1e-8, beta1=0.9, beta2=0.999):
    loss = []
    acc = []
    W = np.random.rand(X.shape[1], 1)
    b = 0
    mW = 0
    vW = 0
    mb = 0
    vb = 0
    for i in tqdm(range(epoch), desc="
        Training␣pass:␣", leave=True):
         bat = np.random.choice(X.shape
             [0], batch)
         X_batch = X[bat]
         Y_batch = Y[bat]
         gradient_W, gradient_b =
             calc_gradient_B(X_batch,
             Y_batch, W, b, c)
        mW = beta1 ∗ mW + (1 − beta1)
             ∗ gradient_W
        vW = beta2 ∗ vW + (1 − beta2)
             ∗ (gradient_W ∗∗ 2)
        W = W − learning_rate ∗ mW / (
             np.sqrt(vW) + eps)
        mb = beta1 ∗ mb + (1 − beta1)
             ∗ gradient_b
        vb = beta2 ∗ vb + (1 − beta2)
             ∗ (gradient_b ∗∗ 2)
        b = b − learning_rate ∗ mb / (
             np.sqrt(vb) + eps)
         loss.append(calc_loss_B(X, Y,
            W, b, c))
         acc.append(calc_validation_B(W
             , b))
    return W, loss, acc
```

13. Get all the solutions.

TABLE I
TRAINING SPEED

function	speed
Logistic_Regression	11.54it/s
Logistic_Regression_Adam	11.42it/s
SVM_classifier	6.44it/s
SVM_classifier_Adam	6.22it/s

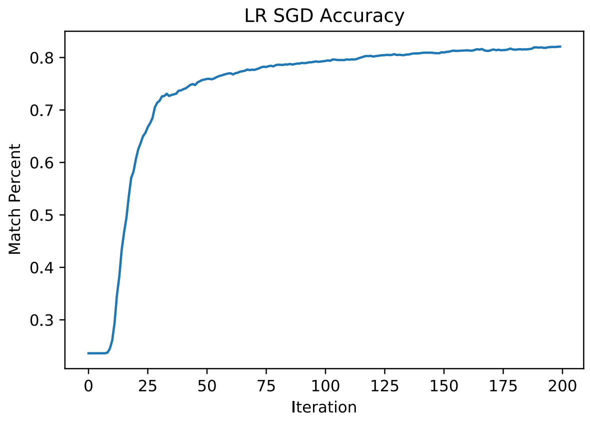

Fig. 1. Accuracy of Logistic_Regression with SGD.

IV. Conclusion

In this experiment, I learnt the difference between Logistic_Regression and SVM while learnt the principle of them.

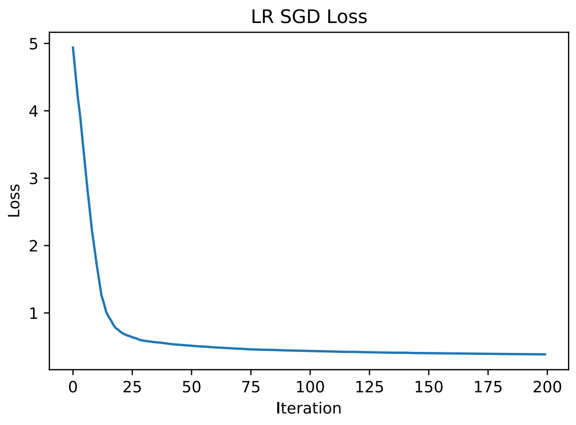

Fig. 2. Loss of Logistic_Regression with SGD.

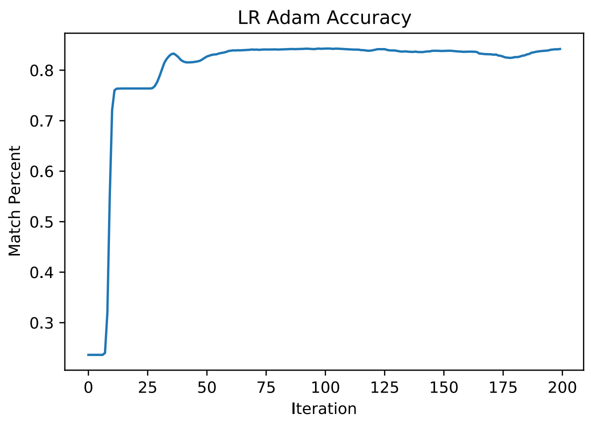

Fig. 3. Accuracy of Logistic_Regression with adam.

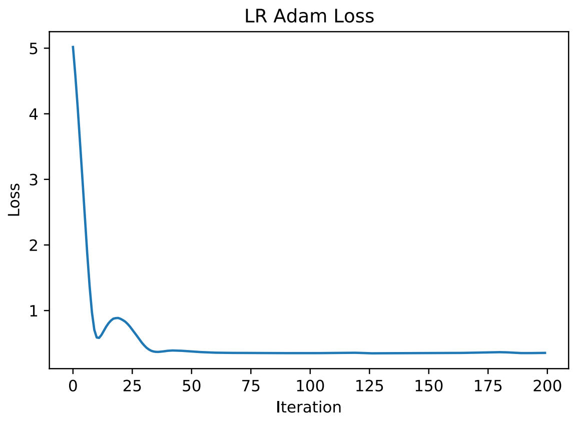

Fig. 4. Loss of Logistic_Regression with adam.

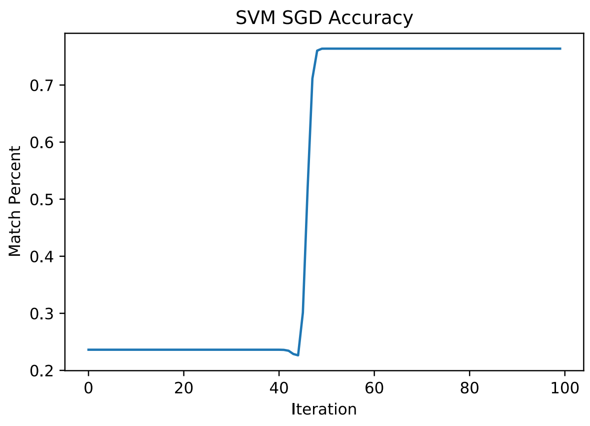

Fig. 5. Accuracy of SVM with SGD.

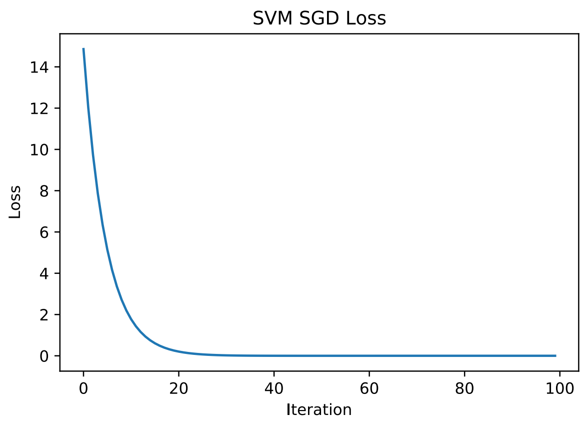

Fig. 6. Loss of SVM with SGD.

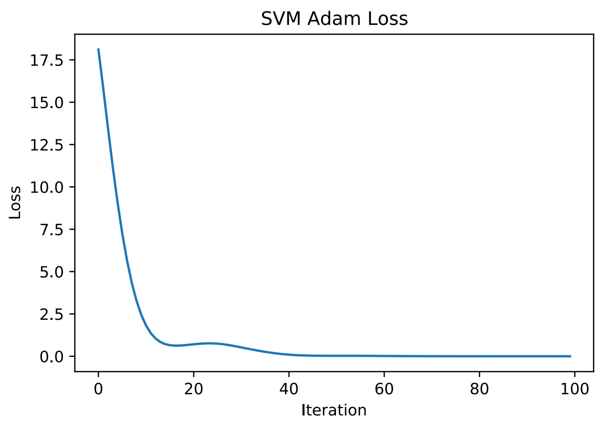

Fig. 8. Loss of SVM with adam.

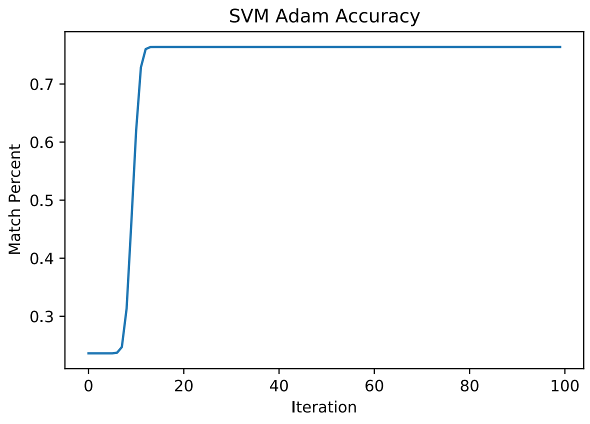

Fig. 7. Accuracy of SVM with adam.