South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Panan Wu

Supervisor:
Mingkui Tan

Student ID:
201630665908

Grade:
Undergraduate

October 27, 2018

Logistic Regression and Support Vector Machine

Abstract—**In this experiment, I conduct two linear classifiers: logistic regression (LR) and support vector machine (SVM) using a9a dataset in LIBSVM Data. I evaluated the two methods mentioned above by plotting the loss value and accuracy during the parameters update process. Experimental results show that support vector machine converge faster than logistic regression and have higher accuracy on test sets, and logistic regression's accuracy rate increase and loss value decrease are more stable and smooth than SVM.**

I. Introduction

CLASSIFIER problem is a classical problem in machine learning. We have mutiple ways to solve these kind of problems, like logistic regression, support vector machine, decision tree, etc. The principle and ideas of each method are different, so the characteristics, efficiency and accuracy of classification methods are different.

In this experiment, I implement two kinds of classification model: logistic regression and support vector machine. By plotting the loss of both training set and test set, and the accuracy of test set in training , we can intuitively know the differences between these two methods.

The motivations of this experiment are: (1) compare and understand the difference between gradient descent and batch random stochastic gradient descent, (2) compare and understand the differences and relationships between Logistic regression and linear classification and (3) further understand the principles of SVM and practice on larger data.

II. Methods and Theory

A. Linear Classification

A linear classifier makes a classification decision based on the value of a linear combination of input features.

Given training data ($\mathbf{x}_i$; y_i) for i = 1...n, with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{-1, 1\}$, learn a classfier f($\mathbf{x}$) such that:

$$f(x) \begin{cases} \geq 0 & y_i = +1 \\ < 0 & y_i = -1 \end{cases}$$

For a correct classification, $y_i f(\mathbf{x}_i) > 0$. And a linear classifier $f(\mathbf{x})$ is:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$

.

In two-dimensional space, the discriminant is a line. **w** is the normal to the line, b is the bias. And **w** is known as the weight vector. While in three-dimensional space, the discriminant is a plane. In m-dimensional space, it becomes a hyperplane.

B. Logistic Regression

The logistic regression uses a logistic function to model a binary dependent variable. It outputs the likelihood of one class. And the logistic function is defined as:

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

Our hypothesis is given by the combination of linear trasformation and logistic function:

$$h_{\mathbf{w}}(\mathbf{x}) = g(\sum_{i=1}^{m} w_i x_i) = g(\mathbf{w}^T\mathbf{x}) \tag{2}$$

And it represents the predict probability of true class. We cannot measure a probability. We can only see the occurence of an event and try to infer a probability. And the probability we infered is described as:

$$P(y|\mathbf{x}) = \begin{cases} h_{\mathbf{w}}(\mathbf{x}) & y = +1 \\ 1 - h_{\mathbf{w}}(\mathbf{x}) & y = -1 \end{cases} \tag{3}$$

To evaluate our model, we use cross entropy error:

$$\mathbf{E}_{in}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_n \cdot \mathbf{w}^T\mathbf{x}}) \tag{4}$$

It is based on an intuitive probabilistic interpretation of h. And it's very convenient and mathematically friendly (easy to minimize).

In order to keep the model simple and avoid overfitting, we introduce regularization into our loss function:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} log(1 + e^{-y_n \cdot \mathbf{w}^T\mathbf{x}}) + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{5}$$

And the regularization parameter λ a trade off between fitting the training set well and keeping the model relatively simple.

Conditional probabilities can be expressed in a different form to facilitate implementation:

$$P(y_i|\mathbf{x}_i) = h_{\mathbf{w}}(\mathbf{x}_i)^{y_i} \cdot (1 - h_{\mathbf{w}}(\mathbf{x}_i))^{1-y_i} \tag{6}$$

And we can change loss function without regularization into another form:

$$J(\mathbf{w}) = \frac{1}{n}[\sum_{i=1}^{n} y_i log h_{\mathbf{w}}(\mathbf{x}_i) + (1 - y_i) log(1 - h_{\mathbf{w}}(\mathbf{x}_i))] \tag{7}$$

Then we calculate the gradient of the loss function to update our parameters:

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= -\frac{1}{\partial \mathbf{w}}\partial[y log h_{\mathbf{w}}(\mathbf{x}) + (1 - y) log(1 - h_{\mathbf{w}}(\mathbf{x}))] \\ &= (h_{\mathbf{w}}(\mathbf{x}) - y)\mathbf{x} \end{aligned} \tag{8}$$

Now we can use gradient descent to update parameters according to the result of parital derivative:

$$\mathbf{w} := \mathbf{w} - \frac{1}{n}\sum_{i=1}^{n}\alpha(h_{\mathbf{w}}(\mathbf{x}_i) - y_i)\mathbf{x}_i \tag{9}$$

C. Support Vector Machine

The main idea of support vector machine (SVM) are to select two parallel hyperplanes that separate the two classes of data and let the distance between them as large as possible. The region bounded by these two hyperplanes is called the "margin".

We first choose normalization such that $\mathbf{w}^T\mathbf{x}_+ + b = +1$ and $\mathbf{w}^T\mathbf{x}_- + b = -1$ for the positive and negative support vectors respectively. Then the magin is given by:

$$\frac{\mathbf{w}}{||\mathbf{w}||}\cdot(\mathbf{x}_+ - \mathbf{x}_-) = \frac{\mathbf{w}^T(\mathbf{x}_+) - \mathbf{x}_-}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||} \tag{10}$$

Learning the SVM can be formulated as an optimization:

$$\begin{aligned} &\max_{\mathbf{w},b} \frac{2}{||\mathbf{w}||} \\ &s.t. \quad \mathbf{w}^T\mathbf{x}_i + b \begin{cases} \geq 1 & y = +1 \\ \leq -1 & y = -1 \end{cases} \end{aligned} \tag{11}$$

In general there is a trade off between the margin and the number of mistakes on the training data. Moreover, training data may not be linearly separable! Thus we introduce variable $\xi_i \geq 0$, for each i, which represents how much example i is on wrong side of margin boundary. If $\xi_i = 0$ then it is ok. If $0 < \xi_i < 1$ it is correctly classified, but with a smaller margin than $\frac{1}{||\mathbf{w}||}$. If $\xi_i > 1$ then it is incorrectly classified.

So the optimization problem becomes:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + C\sum_{i=1}^{n}\xi_i \\ &s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, 2...n \end{aligned} \tag{12}$$

After that, we introduce hingle loss:

$$Hingle\ loss = \xi_i = max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{13}$$

And the optimization problem becomes:

$$\begin{aligned} &\min_{\mathbf{w},b} \frac{||\mathbf{w}||^2}{2} + C\sum_{i=1}^{n}\xi_i \\ &s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad i = 1, 2...n \end{aligned} \tag{14}$$

Since we finally have a good form of optimization problem, we now can compute the gradients:

$$\begin{aligned} \frac{\partial f(\mathbf{w}, b)}{\mathbf{w}} &= \mathbf{w} + C\sum_{i=1}^{N} g_{\mathbf{w}}(\mathbf{x}_i) \\ \frac{\partial f(\mathbf{w}, b)}{b} &= C\sum_{i=1}^{N} g_b(\mathbf{x}_i) \end{aligned} \tag{15}$$

III. Experiments

A. Dataset

The dataset we used in this experiment is a9a dataset from LIBSVM Data. There are 32561 instances in the training dataset and 16281 instances in the test dataset. Both datasets have 123 features.

B. Implementation

1) Logistic Regression: .

First of all, I load the a9a dataset using $load\_svmlight\_file$ function in sklearn library. Since the dataset has already been splited into training set and test set, I just simply store data into different variables.

When it comes to initialize parameters. I choose to set all parameter into zero, initialize it randomly and with normal distribution separately to determine which one is the best. After that, I set learning rate to 0.2, batch size to 256, threshold to 0.5 and update parameters. And then I calculate loss of both training set and validation set. I repeat this process 800 times and output the loss during each process.

Finally, I plot both training loss and validation loss during parameter update. As we can see from Fig. 1, the loss of validation set are stable decline. And then I plot the validation accuracy varing with the number of iterations, the accuracy of valiadation is stable increase. Finally, I output and plot the mean loss with different parameter initializers, as shown in TABLE I and Fig. 3, the initializers in logistic regression have some impact on validation accuracy, and zero intializer seems to have the best effect on validation accuracy.

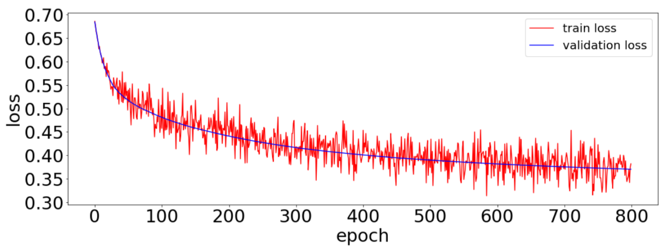

Fig. 1. The loss varing with the number of iterations.

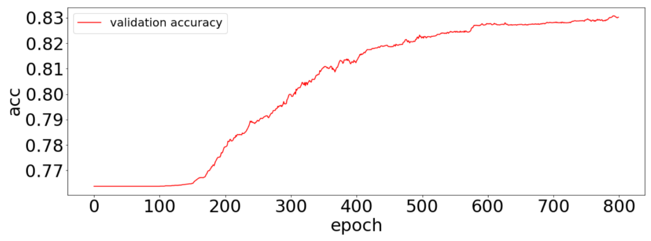

Fig. 2. The accuracy varing with the number of iterations.

2) Support Vector Machine: .

As the same in logistic regression experiment, I read training data and test data into two variables. In training, I set learning rate to 0.005, soft margin trade off parameter C to 0.5 and max

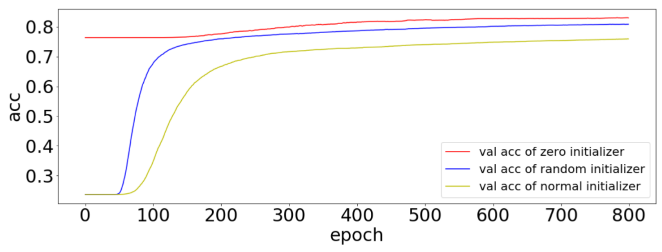

Fig. 3. The accuracy varing with the number of iterations and different initializers.

TABLE I
MEAN ACCURACY OF LAST TEN ITERATIONS WITH DIFFERENT INITIALIZER

Initializer	Mean Acc
Zero Initializer	0.830
Random Initializer	0.809
Normal Initializer	0.759

epoch to 100. I still try three types of parameter initializers. When it comes to parameters update, I first randomly choose 256 samples from training set and update parameters according to its partial derivative, and I calculate and store the training loss and validation loss with parameters just update.

As shown in Fig. 4 and Fig. 5, the training loss and validation accuracy is unstable, but the validation loss is quite stable and keep declining. Different initializers still bring different mean accuracy values, which means that choosing different initializers in svm will have a certain impact on the results.

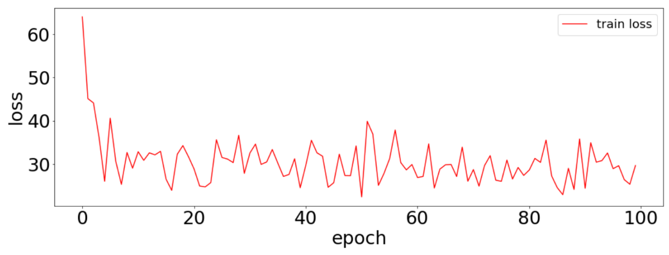

Fig. 4. The training loss varing with the number of iterations.

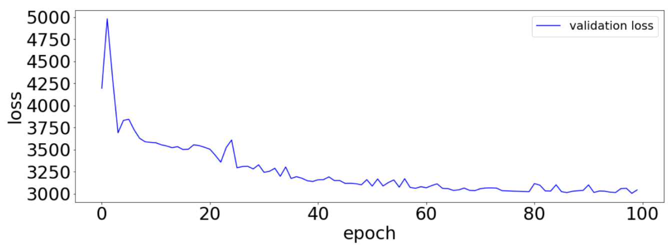

Fig. 5. The validation loss varing with the number of iterations.

Finally, I compared the result of two methods mentioned above. We can know from Table III that support vector machine has higher validation accuracy than LR. And the number of iterations it needs are much lower than LR, the learning rate of SVM is much lower than LR too. But logistic regression has its own advantage, its validation set accuracy rate increase and loss value decrease are more stable and smooth than SVM. They both have four hyper-parameters that need to be artificially adjusted.

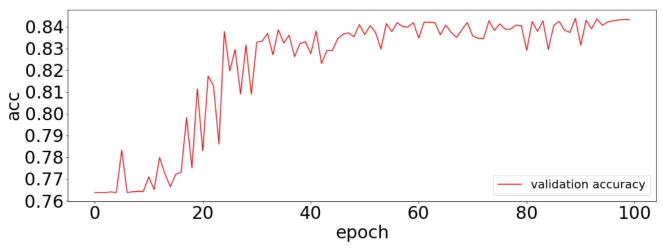

Fig. 6. The accuracy varing with the number of iterations.

TABLE II
MEAN ACCURACY WITH DIFFERENT INITIALIZER OF LAST TEN ITERATIONS

Initializer	Mean Acc
Zero Initializer	0.841
Random Initializer	0.842
Normal Initializer	0.827

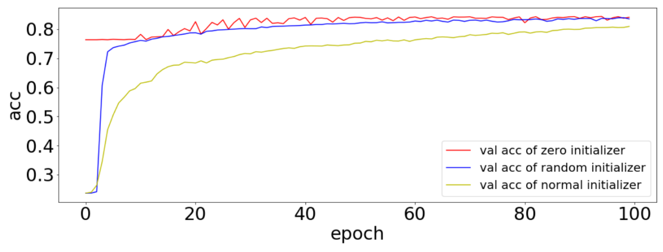

Fig. 7. The accuracy varing with the number of iterations and different intializers.

TABLE III
MEAN ACCURACY WITH DIFFERENT METHODS OF LAST TEN ITERATIONS

Methods	Mean Acc
Logistic Regression	0.842
Support Vector Machine	0.830

IV. Conclusion

In this experiment, I implement logistic regression and support vector machine, I analyse these two models mentioned above in different aspects and I make comparison between these two methods. From what has been shown above, we may safely draw the conclusion that support vector machine needs less iterations and lower learning rate than logistic regression, and the logistic regression's accuracy rate increase and loss value decrease are more stable and smooth than SVM. But we need to try other kinds of dataset to evalute these two methods better.

During this experiment, I realized what I learned in class, which enhanced my code practice ability. By comparing these two methods and writing experimental reports, I came into contact with a little scientific research. And it has increased my knowledge.