The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:

Peng Cheng

Supervisor:

Mingkui Tan

Student ID:

201830020255

Grade:

Undergraduate

2020-10-24

Logistic Regression and Support Vector Machine

***Abstract*—**This paper is based on batch stochastic gradient descent to solve logistic regression and linear classification problems. Through the SGD optimization method, the parameters of the model are continuously updated to achieve the optimization of the logistic regression model and the classification model.

I. INTRODUCTION

The two most common linear classification algorithms are logistic regression (Logistic regression) and linear support vector machine (linear SVM). In this paper, we will introduce the two methods and experiment on the dataset.

II. METHODS AND THEORY

2.1 Logistic Regression

2.1.1 Probability Function

Logistic regression applies a logistic function on the basis of linear regression:

$$\boldsymbol{h_w(x)} = g(z) = g(\boldsymbol{w}^T\boldsymbol{x})$$

Here, g(•) is a logistic function:

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

2.1.2Loss Function

The least square loss is no longer valid here since $\boldsymbol{h_w(x)}$ is a probability function with $\boldsymbol{h_w(x)} \leq 1$ and it is not a convex function, which is not conducive to the use of gradient descent method to solve. Here we introduce a new loss called logistic loss as below:

$$L(\boldsymbol{w}) = \frac{1}{n}\log\left(1+e^{-y_i w^t x_i}\right)$$

Then it turns out to be an optimization problem:

$$min\ L(\boldsymbol{w}) = \frac{1}{n}\log\left(1+e^{-y_i w^t x_i}\right)$$

2.1.3Stohastic Gradient method

We first determine the size of batch size and randomly take some sample, which the subset is written as S_K and calculate the gradient of $L(\boldsymbol{w})$ with respect to $\boldsymbol{w}$ from partial samples:

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\frac{1}{||S_K||}\sum_{i\in S_K}\frac{y_i\boldsymbol{x_i}e^{-y_i\boldsymbol{w^t x_i}}}{1+e^{-y_i\boldsymbol{w^t x_i}}}$$

And then we update **w** with learning rate η:

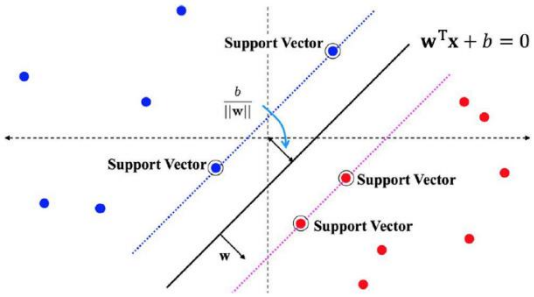
$$w := w - \eta\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

2.2 Linear classification and SVM

2.2.1 Binary classifier

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

2.2.2 SVM



The basic idea of SVM learning is to solve the separation hyperplane that can correctly divide the training data set and have the largest geometric interval.$\boldsymbol{w}^T\boldsymbol{x}+b=0$ is the hyperplane and for any hyperplane, the data points on both sides have a minimum distance (vertical distance) from it, and the sum of these two minimum distances is the interval. As shown in the diagram above, the margin is $\frac{2}{||w||}$, and then SVM can be formulated as an optimization:

$$min_w = \frac{||w||^2}{2}$$

However, there exists outliners which have an impact on the model. So making a trade-off between margin and number of mistakes on the train data is indeed indispensable. The slack variable ξ_i is introduced, which represents how much example i is on the wrong side of margin boundary. Then the optimization problem becomes:

$$\min\ \frac{||w||^2}{2} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t.\ y_i(\boldsymbol{w^t x_i}) \geq 1-\xi_i$$

And it can be represented as

$$\min\ J(\boldsymbol{w}) = \frac{||w||^2}{2} + \frac{C}{n}\sum_{i=1}^{n}\max\left(0, 1-y_i(\boldsymbol{w^t x_i})\right)$$

As described in 2.1.3, rather than using all samples, batch SGD is employed, which comes out as follows:

$$\min J(\boldsymbol{w}) = \frac{||w||^2}{2} + \frac{C}{|S_K|}\sum_{i\in S_K} \max(0, 1 - y_i(\boldsymbol{w}^t \boldsymbol{x_i}))$$

And update **w**:

$$w := w - \eta(\boldsymbol{w} + \frac{C}{|S_K|}\sum_{i\in S_K} \boldsymbol{g_w}(\boldsymbol{x_i}))$$

$\boldsymbol{g_w}(\boldsymbol{x_i})$ is as follows, which can be got by derivation:

$$\boldsymbol{g_w}(\boldsymbol{x_i}) \begin{cases} -y_i \boldsymbol{x_i} & 1 - y_i(\boldsymbol{w}^t \boldsymbol{x_i}) \geq 0 \\ 0 & 1 - y_i(\boldsymbol{w}^t \boldsymbol{x_i}) < 0 \end{cases}$$

III. EXPERIMENT

A. Dataset

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features. And $y_i \in \{-1,1\}$.

B. Implementation

TABLE I
INITIALIZATION OF PARAMETERS IN LOGISTIC REGRESSION

Learning rate	η = 0.1
Max epoch number	e = 50
Penalty factor	C = 0.5
Batch_size	50
threshold	0

TABLE II
INITIALIZATION OF PARAMETERS IN LINEAR CLASSIFICATION

Learning rate	η = 0.0001
Max epoch number	e = 100
Penalty factor	C = 10
Batch_size	2048
threshold	0

We first set **w** to vectors with random value in logistic regression and with all zeros in linear classification and after constant iterations, the loss of training set and validation gradually drops.

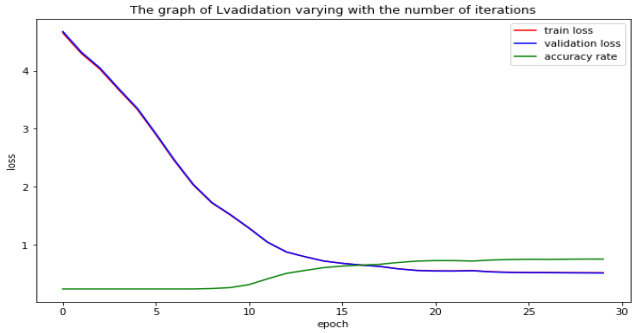


Figure. 1. The graph of Lvalidation varying with the number of iterations(Logistic regression)

As shown in fig 1, Lvalidation (loss of validation set) gradually decrease and after 30 iterations, it drops to 0.593. Meanwhile the accuracy rate rises to 0.710, which indicates that the training is effective.

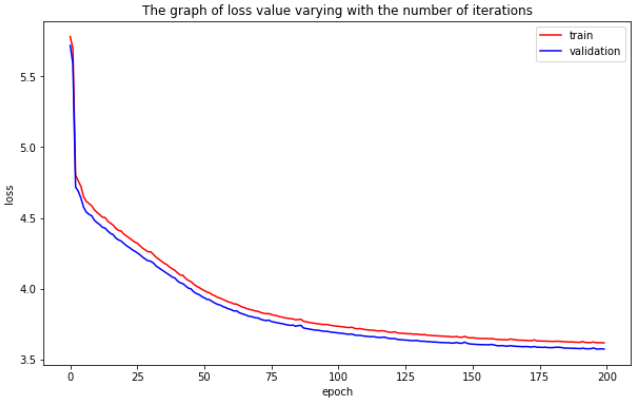


Figure. 2. The graph of Lvalidation varying with the number of iterations(Linear classification)

As shown in fig 2, Lvalidation(loss of validation set) gradually decrease and after 200 iterations, it drops to 3.572. And the final results comes out as follow:

TABLE III
CLASSIFICATION REPORT

	precision	recall	F1-score	support
Positive	0.868	0.938	0.902	12435
Negative	0.730	0.537	0.618	3846
Accuracy			0.844	16281
Macro avg	0.799	0.738	0.760	16281
Weighted avg	0.835	0.844	0.835	16281

IV. CONCLUSION

In the course of the experiment, it was found that the convergence speed of using the batch gradient descent method is more efficient than using all samples without loss of accuracy.

In this process, the internal connection between logistic regression and linear classification becomes clearer. Both of them can be used to solve the two classification problems.

However, some problems exist. Slight changes in some parameters will cause more obvious changes, indicating that the robustness of the model is not strong. Some optimizations can be employed by adding regular terms to logistic regression to prevent overfitting and in batch SGD, different batches can be chose in each iteration to cover as many samples as possible to make full use of all the features of all sample given.