South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: ELECTRONIC AND INFORMATION

Author:
Weiwen Hu

Supervisor:
Mingkui Tan

Student ID:
202021045611

Grade:
Graduate

January 3, 2021

Logistic Regression and Support Vector Machine

***Abstract*—This experiment is about two basic linear classification method: logistic regression and support vector machine, also the Batch Stochastic Gradient Descent method.**

I. Introduction

We are conducting this experiment in hope of: 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. 3. Further understand the principles of SVM and practice on larger data. We are expecting to get a classification model for each of these two methods.

II. Methods and Theory

A. Logistic Regression

In statistics, the logistic model is a widely used statistical model that, in its basic form, uses a logistic function to model a binary dependent variable.

The logistic function calculates a probability. It is defined as follows:

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

For example, given a sample with features $\mathbf{x} \in \mathbb{R}^N$, the probability that it is a positive sample can be calculated as:

$$P(y=1|\mathbf{x}) = g(\mathbf{w}^\mathsf{T}\mathbf{x}). \tag{2}$$

Then we can maximize the possibility of correct prediction.

$$\begin{aligned}
\max \prod_{i=1}^{n} P(y_i|\mathbf{x}_i) &\Leftrightarrow \max \log\left(\prod_{i=1}^{n} P(y_i|\mathbf{x}_i)\right) \\
&\equiv \max \sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i) \\
&\Leftrightarrow \min -\frac{1}{n}\sum_{i=1}^{n} \log P(y_i|\mathbf{x}_i) \\
&\equiv \min \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{P(y_i|\mathbf{x}_i)} \\
&\equiv \min \frac{1}{n}\sum_{i=1}^{n} \log \frac{1}{g(y_i \cdot \mathbf{w}^\top \mathbf{x}_i)} \\
&\equiv \min \frac{1}{n}\sum_{i=1}^{n} \log\left(1+e^{-y_i\cdot\mathbf{w}^\top\mathbf{x}_i}\right) \\
&\equiv \min E_{in}(\mathbf{w})
\end{aligned} \tag{3}$$

In this way, we derived the logistic loss function $E_{in}(\mathbf{w})$. We can use gradient descent method to minimize it. The gradient is:

$$\frac{\partial E_{in}(\mathbf{w})}{\partial \mathbf{w}} = -\frac{1}{n}\sum_{i=1}^{n} \frac{y_i\mathbf{x}_i}{1+e^{y_i\cdot\mathbf{w}^\mathsf{T}\mathbf{x}_i}} \tag{4}$$

B. Support Vector Machine

An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible, which is formulated by:

$$\begin{aligned}
&\max_{\mathbf{w},b} \tfrac{2}{\|\mathbf{w}\|} \\
&\text{s.t.} \quad \mathbf{w}^\mathsf{T}\mathbf{x}_i + b \begin{cases} \geqslant 1 & y_i = +1 \\ \leqslant -1 & y_i = -1 \end{cases}
\end{aligned} \tag{5}$$

But since noise and outliner is unavoidable, we must tolerant some classification mistake, so we use the hinge loss function:

$$\min_{\mathbf{w},b} \frac{\|\mathbf{w}\|^2}{2} + \frac{C}{n}\sum_{i=1}^{n} \max\left(0, 1-y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i+b\right)\right), \tag{6}$$

where C is a hyper-parameter to balance the margin and the tolerance to outliner.

Like before, we use gradient descent to minimize this loss. The gradient is calculated to the follow:

$$g_{\mathbf{w}}(\mathbf{x}_i) = \begin{cases} -y_i\mathbf{x}_i & 1-y_i\left(\mathbf{w}^\top\mathbf{x}_i+b\right) >= 0 \\ 0 & 1-y_i\left(\mathbf{w}^\top\mathbf{x}_i+b\right) < 0 \end{cases} \tag{7}$$

$$\frac{\partial f(\mathbf{w},b)}{\partial \mathbf{w}} = \mathbf{w} + \frac{C}{n}\sum_{i=1}^{n} g_{\mathbf{w}}(\mathbf{x}_i) \tag{8}$$

III. Experiments

A. Dataset

We use *a9a* of *LIBSVM* Data, including 32561 trainning and 16281 testing samples, each sample has 123 features.

B. Implementation

I use NumPy to implement all the formulas above.

I initialized parameter $\mathbf{w}$ to all zero. Parameter C in SVM is set to 100. I add an extra all 1 feature to every sample, so that I don't need to worry about the bias parameter b. Other hyper-parameters are listed in table I

TABLE I
Hyper Parameters

	Logistic regression	SVM
Learning rate	0.3	0.001
Epoch	30	20
Batch size	512	512

TABLE II
RESULTS

	Logistic regression	SVM
Train loss	0.3275	38.05
Test loss	0.3246	37.73
Accuracy (%)	85.20	84.64

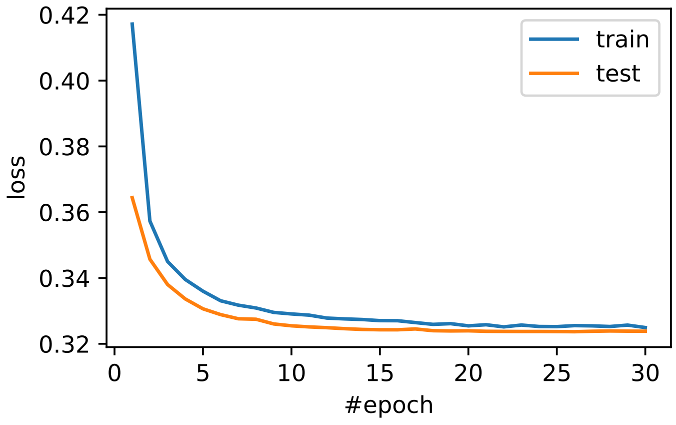

Fig. 1. Logistic Regression Loss

C. Results

The experiment result is summarizes in table II

Figure 1, figure 2, figure 3, figure 4 show the loss fuction value and classification accuracy changing with training epoch number, respectively. The results of logistic regression and SVM are similar, both achieved arround 85% test classification accuracy, logistic regression does slightly better.

Despite no explict regularization term added, there are no sign of over fitting. Instead, the test accuracy is even slightly higher than train accuracy. This may be caused by more noise in training set.

When tuning the parameter C in SVM, I found the higher the C, the better the result. The large C also resulted in large absolute value of the loss. This could be because these examples are not quite linearly separable, and the "clear gap" motivation of SVM does not make many sense in this scenario.

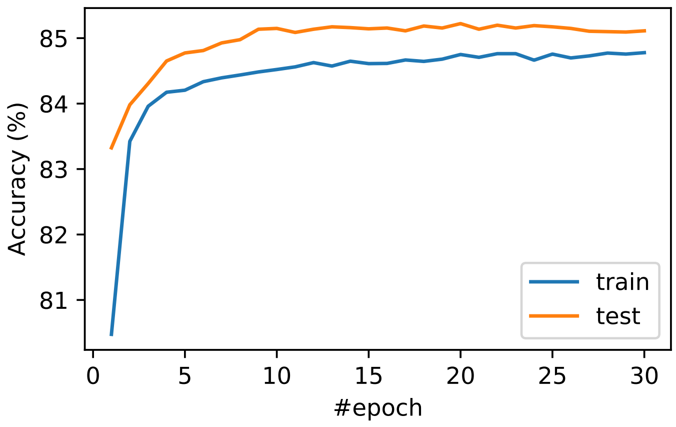

Fig. 2. Logistic Regression Accuracy

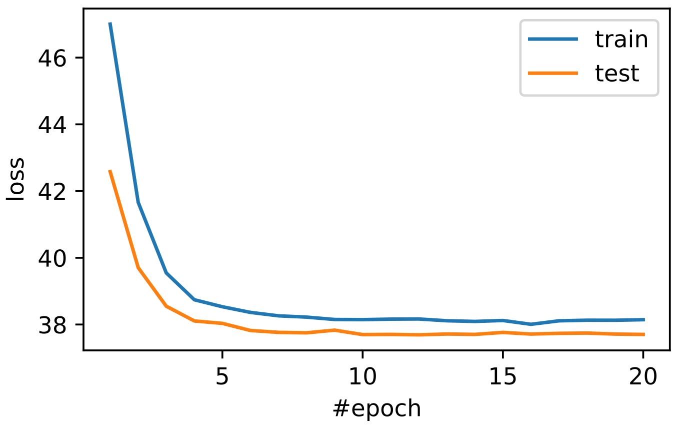

Fig. 3. SVM Loss

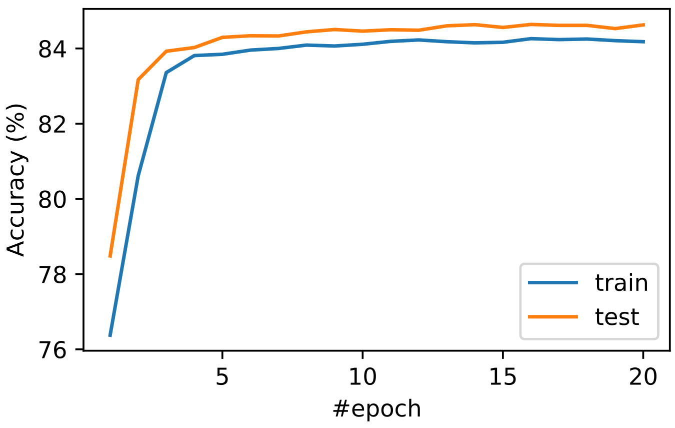

Fig. 4. SVM Accuracy

IV. Conclusion

This experiment is giving expected results. I have learned the basic procedure of machine learning project and the powerful gradient descent method, as well as the logistic regression and SVM classification methods. I also get a sense of how they works compared with each other.