The Experiment Report of Machine Learning

SCHOOL: Automation Science and Engineering

SUBJECT: automatization

Author:

Zhirui Zhang

Supervisor:

Mingkui Tan

Student ID:

202130462158

Grade:

Undergraduate

2024-3-29

Logistic regression and Support vector machines

***Abstract*—In this experiment, we compared the differences between gradient descent and stochastic gradient descent on larger datasets, as well as the distinctions and connections between logistic regression and linear classification. We studied the behavior of logistic regression and linear classification models trained with batch stochastic gradient descent and furthered our understanding and application of support vector machines (SVMs).**

I. INTRODUCTION

This experiment focuses on exploring the subtle differences between gradient descent algorithms, specifically gradient descent and stochastic gradient descent, and understanding their implications in logistic regression and linear classification. We aim to address fundamental questions such as convergence behavior, computational efficiency, and optimization strategies associated with these algorithms. Additionally, we extend our exploration to support vector machines (SVMs), aiming to understand their underlying principles and practical applications on larger datasets, particularly the a9a dataset. Through this experiment, we aim not only to apply theoretical concepts but also to gain practical experience in implementing machine learning models and evaluating their performance.

II. METHODS AND THEORY

1. Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification and regression tasks. Its main idea is to find an optimal hyperplane in the feature space that separates samples from different classes while maximizing the margin between the closest training samples and the hyperplane. In classification tasks, SVM aims to find a decision boundary that maximizes the margin while ensuring the accuracy of classifying the training samples. If the data is not linearly separable, it can be mapped to a higher-dimensional feature space using kernel trick to make it linearly separable.

The fundamental idea behind SVM learning is to find the separating hyperplane that correctly divides the training dataset and maximizes the geometric margin. For linearly separable datasets, there are infinitely many such hyperplanes (as in the perceptron); however, the separating hyperplane with the maximum geometric margin is unique.

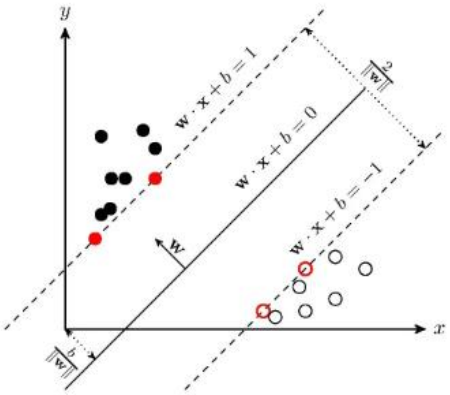

Figure. 1. Hyperplane and its distance

Thus the problem can be transformed into solving the optimization problem:

$$\min_{\mathbf{w},b}\frac{\|\mathbf{w}\|^2}{2}$$

$$s.t.\quad y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\geq 1,\quad i=1,2,3,...,n$$

When faced with linearly inseparable sample data, optimization of the algorithm is necessary. Therefore, we can utilize a soft margin, which, in comparison to the strict conditions of a hard margin, allows for individual sample points to fall within the margin. We permit certain sample points to not satisfy the constraint conditions:

$$1-y_i(w^T x_i+b)\leq 0$$

To quantify the softness of this margin, we introduce a slack variable for each sample, transforming the optimization objective into:

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

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

Where C is a positive constant greater than 0, which can be understood as the penalty imposed on misclassified samples. If C is infinite, the penalty becomes infinitesimal, thus transforming the linear SVM back into a linearly separable SVM. Only when C takes a finite value does it allow for some samples to violate the constraint conditions.

By taking the gradient of the loss function, we can derive the gradient descent algorithm equation:

$$\mathbf{w}=\mathbf{w}-n\nabla_{\mathbf{w}}L(\mathbf{w},b)$$

$$b=b-\eta\nabla_b L(w,b)$$

$$\nabla_w L(\mathbf{w},b)=\mathbf{w}+\frac{C}{n}\sum_{i=1}^{n}g_w(\mathbf{x}_i)$$

$$\nabla_b L(\mathbf{w},b)=\frac{C}{n}\sum_{i=1}^{n}g_b(\mathbf{x}_i)$$

2. Logistic Regression

Logistic regression is a statistical learning method used to solve classification problems. It maps the output of a linear regression model to a probability value, and then makes classification predictions based on this probability value. Specifically, logistic regression uses the sigmoid function to transform the linear combination of features into probability values between 0 and 1, representing the probability of a sample belonging to a certain class. Then, based on a predefined threshold, the probability values are transformed into class labels. Logistic regression can be used for both binary and multi-class classification problems and finds applications in various fields such as credit scoring, medical diagnosis, and natural language processing.

Because the logistic regression outputs probability values ranging from 0 to 1, the mean squared error function is no longer applicable, and the logistic loss function is used instead.

$$\mathcal{L}(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-\gamma_i \mathbf{w}^{\mathrm{T}}\mathbf{x}_i})$$

Converting the optimization problem to maximum likelihood estimation of probabilities is equivalent to minimizing the logistic loss function.

$$\max_{\mathbf{w}}\prod_{i=1}^{n}P(y_i \mid \mathbf{x}_i) = \boxed{\min_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \min_{\mathbf{w}}\frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-\gamma_i \cdot \mathbf{w}^{\mathrm{T}}\mathbf{x}_i})}$$

3. SGD and the Adam Optimization Algorithm

Stochastic Gradient Descent (SGD) for linear regression is a variant of the gradient descent algorithm primarily utilized for handling large-scale datasets or a large number of training samples. In contrast to traditional gradient descent algorithms, SGD updates parameters by randomly selecting samples, thereby reducing computational complexity and enhancing efficiency.

3.1 SGD

Stochastic Gradient Descent (SGD) is a fundamental optimization algorithm widely used in machine learning and optimization tasks. It aims to minimize the loss function by iteratively updating the model parameters based on the gradient of the loss function with respect to those parameters. Specifically, SGD operates by randomly selecting a subset of the training data (mini-batch) to compute an approximation of the gradient. This stochastic nature makes SGD computationally efficient and particularly suitable for large-scale datasets. However, the convergence of SGD can be sensitive to the choice of the learning rate hyperparameter, which requires careful tuning.

3.2 Adam

The Adam optimization algorithm, short for Adaptive Moment Estimation, is a popular variant of gradient descent that combines the benefits of both momentum and RMSProp. Adam maintains adaptive learning rates for each parameter by computing the first and second moments of the gradients. This adaptive learning rate mechanism allows Adam to automatically adjust the step size for each parameter, leading to faster convergence and improved performance compared to traditional SGD. Additionally, Adam incorporates momentum, which accelerates the optimization process by accumulating a fraction of the past gradients. The algorithm is known for its robustness and effectiveness across a wide range of deep learning tasks and datasets, requiring minimal hyperparameter tuning compared to SGD.

The update rules for the Adam optimizer are as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

III. EXPERIMENT

1. Experimental data

The experiment employs the a9a dataset from LIBSVM Data, comprising 32,561 (training) / 16,281 (testing) samples, with each sample containing 123 (training) / 123 (testing) attributes.

2. Logistic regression and linear classification

For the same dataset, both logistic regression and linear classification models were trained. Both models utilized SGD optimization with a batch size of 100 and 250 iterations. The learning rate for logistic regression was set to 0.0001, while for linear classification, it was set to 0.01, with a regularization constant (C) of 0.5.

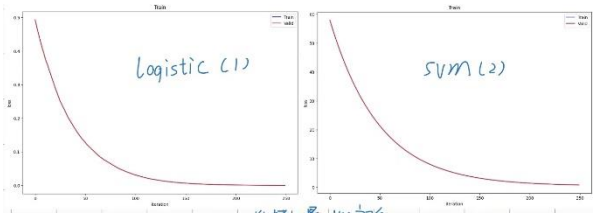

Figure. 2. Model comparison

The experimental results indicate that under these two sets of parameters, the classification scores of the two models are essentially the same, with similar performance.

3. Mini Batch Comparison

To conduct comparative experiments on different batch sizes for the logistic regression model, batch sizes of 100, 50, and 150 were defined, respectively.

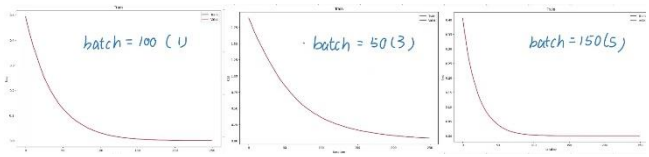

Figure. 3. Batch comparison

It can be observed that with larger batch sizes, the loss decreases more rapidly, resulting in smaller final loss values. However, excessively large batch sizes increase computational complexity, thus requiring a balance between accuracy and complexity.

4. Comparison of svm penalty levels

C is a positive constant greater than 0, which can be understood as the penalty imposed on misclassified samples.

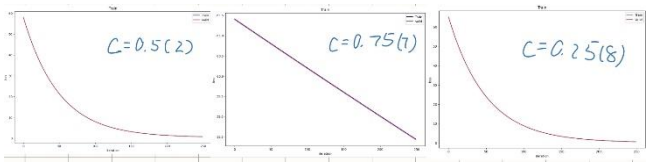

Figure. 4. C comparison

The experimental results indicate that the penalty term C affects the convergence speed of training, with larger values of C resulting in slower convergence speeds.

5. Optimization with Adam

Building upon SGD optimization, incorporate Adam optimization. Utilize Adam optimization to control the learning rate.

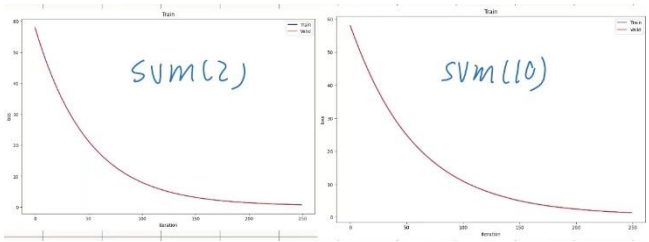

Figure. 5. Optimization with Adam

Using SVM for comparative experiments, the final results show that employing Adam optimization reduces the overall learning rate, leading to a slower decrease in loss, yet without impacting the model's performance.

IV. CONCLUSION

This experiment explored the intricacies of optimization techniques and classification algorithms, focusing on logistic regression and linear classification with Support Vector Machines (SVM). Through empirical evaluations on the a9a dataset, various aspects including parameter initialization, loss functions, and optimization algorithms were investigated. The results showcased the importance of optimization methods such as stochastic gradient descent (SGD) and Adam in achieving convergence and accuracy in model training. Additionally, the experiment highlighted the effectiveness of SVM in handling large datasets and complex classification tasks.