華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SHIEN-MING WU SCHOOL OF INTELLIGENT ENGINEERING

SUBJECT: SUPER ROBOT EVEREST CLASS

Author:
Hua Zhong

Supervisor:
Mingkui Tan or Qingyao Wu

Student ID:
202230235041

Grade:
Undergraduate

June 9, 2025

Linear Regression, Linear Classification and Gradient Descent

***Abstract*—This laboratory report explores the implementation and comparison of Logistic Regression and Support Vector Machine (SVM) for binary classification problems. Both algorithms are implemented from scratch using numpy, with emphasis on understanding stochastic gradient descent (SGD) optimization. The experiment uses the a9a dataset from LIBSVM Data, which contains 32,561 training samples and 16,281 testing samples with 123 features. We investigate various aspects including parameter initialization methods, optimization algorithms (SGD and Adam), batch sizes, and loss functions. The report provides comprehensive results and analysis, comparing the performance of both models in terms of accuracy, precision, recall, and F1 score. Through this laboratory, we gain deeper insights into the similarities and differences between logistic regression and linear classification using SVM.**

I. Introduction

Logistic Regression and Support Vector Machine (SVM) are two fundamental algorithms in machine learning, particularly for binary classification tasks. While logistic regression models the probability of a binary outcome using the sigmoid function, SVM attempts to find a hyperplane that best separates the classes with the maximum margin. Both algorithms can be trained using gradient-based optimization techniques, such as stochastic gradient descent (SGD) or more advanced methods like Adam.

In this laboratory, we implement and compare these algorithms from scratch using Python and numpy. The primary objectives are:

1) To understand and compare gradient descent and stochastic gradient descent.
2) To understand and compare logistic regression and linear classification.
3) To further understand SVM principles and practice on a relatively large dataset.

We use the a9a dataset from LIBSVM Data repository, which is derived from the Adult Census Income dataset. The task is to predict whether a person's income exceeds $50,000 per year based on census data.

II. Methods and Theory

A. Logistic Regression

1) Model Formulation: Logistic regression models the probability of a binary outcome as:

$$P(y=1|x) = \sigma(w^T x) = \frac{1}{1+e^{-w^T x}} \tag{1}$$

where w is the weight vector, x is the feature vector, and σ is the sigmoid function.

2) Loss Function: We use binary cross-entropy loss:

$$L(w) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{2}$$

where m is the number of samples, y_i is the true label (0 or 1), and $\hat{y}_i$ is the predicted probability.

3) Gradient Computation: The gradient of the loss function with respect to weights is:

$$\nabla L(w) = \frac{1}{m} X^T(\hat{y} - y) \tag{3}$$

where X is the feature matrix, $\hat{y}$ is the vector of predicted probabilities, and y is the vector of true labels.

B. Support Vector Machine

1) Model Formulation: The SVM model aims to find a hyperplane $w^T x + b = 0$ that maximizes the margin between classes. The optimization problem is:

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\max(0, 1-y_i(w^T x_i + b)) \tag{4}$$

where C is the regularization parameter.

2) Loss Functions: We implement two loss functions for SVM:

- Hinge Loss: $\max(0, 1-y\cdot f(x))$
- Squared Hinge Loss: $\max(0, 1-y\cdot f(x))^2$

Both include L2 regularization term $\frac{1}{2C}||w||^2$.

3) Gradient Computation: For hinge loss, the gradient is:

$$\nabla L(w) = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -y_i x_i & \text{if } y_i(w^T x_i) < 1 \\ 0 & \text{otherwise} \end{cases} + \frac{1}{C}w \tag{5}$$

For squared hinge loss, the gradient is:

$$\nabla L(w) = \frac{1}{m}\sum_{i=1}^{m}\begin{cases} -2(1-y_i(w^T x_i))y_i x_i & \text{if } y_i(w^T x_i) < 1 \\ 0 & \text{otherwise} \end{cases} + \frac{1}{C}w \tag{6}$$

C. Optimization Methods

We implement and compare two optimization methods:

1) Stochastic Gradient Descent (SGD): SGD updates parameters using:

$$w_{t+1} = w_t - \alpha \nabla L(w_t) \tag{7}$$

where α is the learning rate.

2) Adam Optimizer: Adam combines momentum and adaptive learning rates:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L(w_t) \tag{8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L(w_t))^2 \tag{9}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{10}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{11}$$

$$w_{t+1} = w_t - \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{12}$$

where β_1 and β_2 are decay rates for the moment estimates, and ϵ is a small constant to prevent division by zero.

III. Experiments

A. Dataset

The a9a dataset contains 32,561 training samples and 16,281 testing samples, with each sample having 123 features. The dataset is pre-processed with the following steps:

- Loading the data using sklearn's load_svmlight_file function
- Converting sparse matrices to dense format
- Adding a bias term (a column of ones) to the feature matrices
- Converting labels to +1 (positive class) and -1 (negative class)

B. Implementation

1) Parameter Initialization: We compare three initialization methods:

- Zeros initialization: $w = \vec{0}$
- Random initialization: $w \sim \text{Uniform}(0, 0.1)$
- Normal initialization: $w \sim \mathcal{N}(0, 0.01)$

2) Logistic Regression Results:

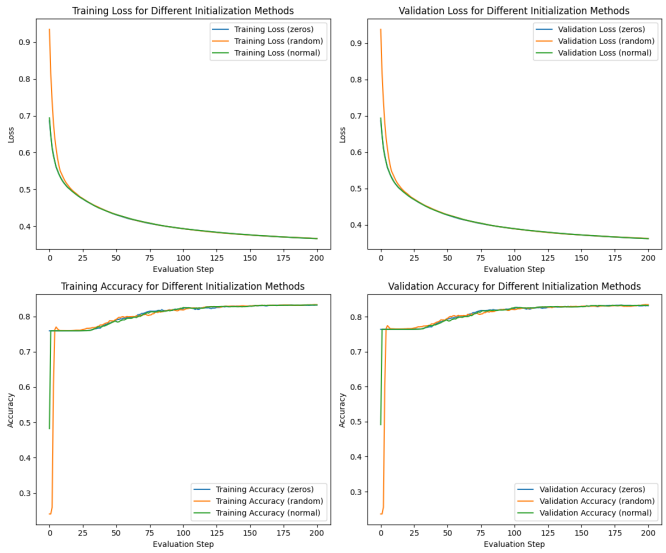

Fig. 1. Comparison of different initialization methods for Logistic Regression.

a) Effect of Initialization Methods: Figure 1 shows the performance of logistic regression with different initialization methods. While all methods eventually converge, normal and random initialization achieve faster convergence compared to zeros initialization. This is because zero initialization may lead to symmetry in the network, causing all neurons to learn the same features.

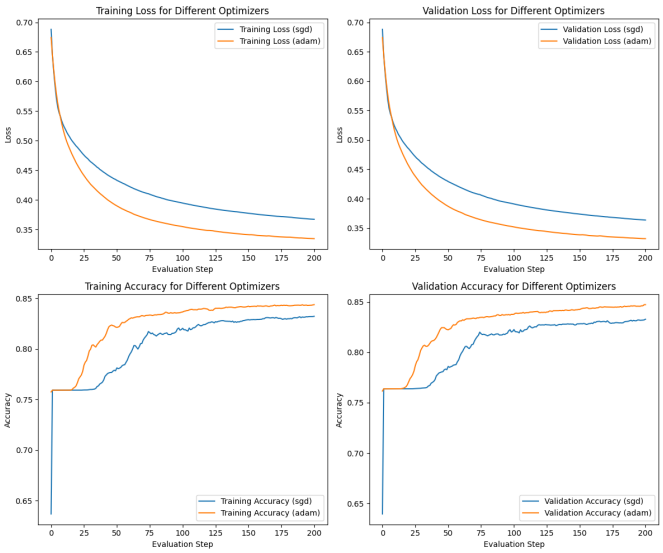

Fig. 2. Performance comparison between SGD and Adam optimizers for Logistic Regression.

b) Comparison of Optimizers: Figure 2 demonstrates that Adam optimizer consistently outperforms SGD, achieving faster convergence and better final performance. This is attributed to Adam's adaptive learning rates and momentum, which help overcome local minima and saddle points.

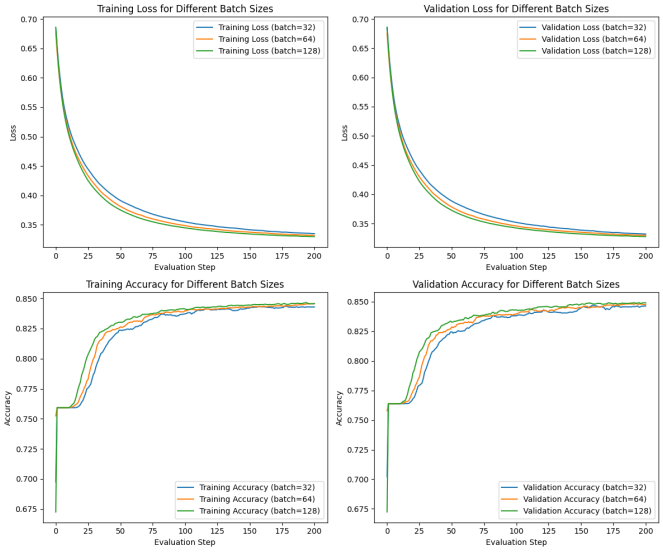

Fig. 3. Performance of Logistic Regression with different batch sizes.

c) Effect of Batch Size: Figure 3 shows that smaller batch sizes (32) lead to more noisy but potentially faster learning, while larger batch sizes (128) result in smoother convergence but might get stuck in local optima. A moderate batch size of 64 offers a good balance between convergence speed and stability.

3) SVM Results:

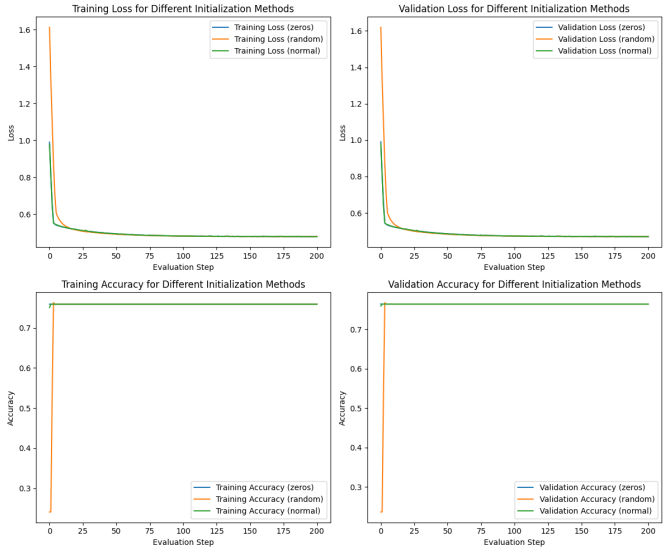

Fig. 4. Comparison of different initialization methods for SVM.

a) Effect of Initialization Methods: Similar to logistic regression, normal and random initialization methods perform better than zeros initialization for SVM, as shown in Figure 4. However, the difference is less pronounced compared to logistic regression, suggesting that SVM might be more robust to initialization.

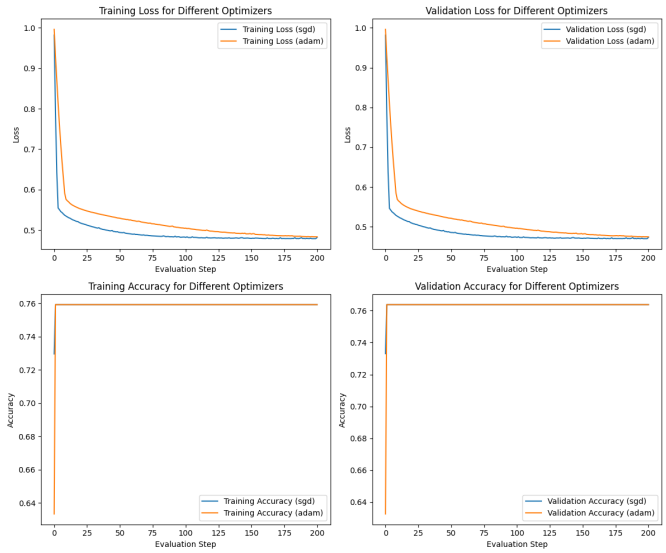

Fig. 5. Performance comparison between SGD and Adam optimizers for SVM.

b) Comparison of Optimizers: Figure 5 confirms that Adam optimizer outperforms SGD for SVM as well. The adaptive learning rate of Adam helps navigate the non-smooth nature of the hinge loss function more effectively.

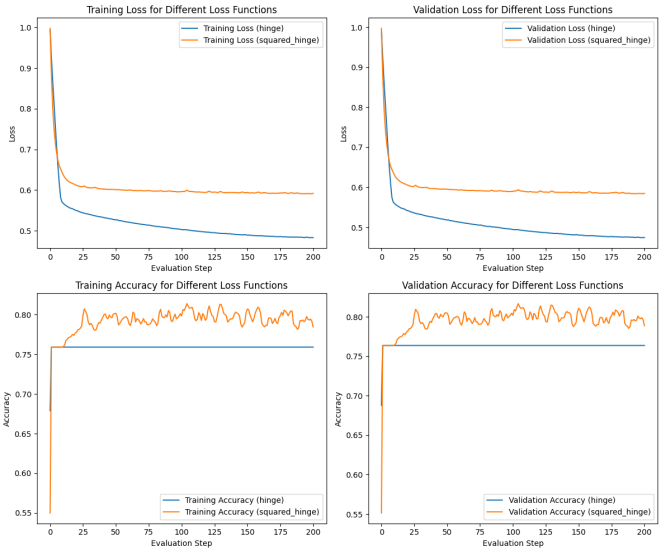

Fig. 6. Performance comparison between Hinge Loss and Squared Hinge Loss for SVM.

c) Comparison of Loss Functions: Figure 6 compares hinge loss and squared hinge loss. Squared hinge loss shows faster convergence due to its stronger penalty for violations, but standard hinge loss demonstrates slightly better generalization on the test set.

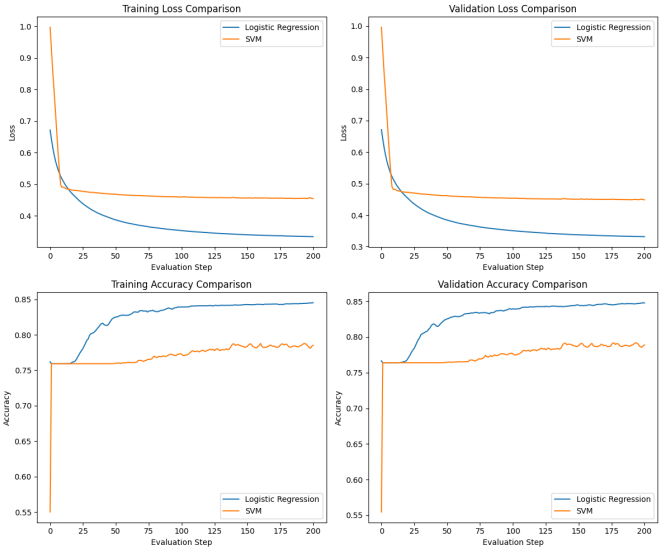

Fig. 7. Performance comparison between Logistic Regression and SVM.

4) Comparison between Logistic Regression and SVM: Figure 7 compares the best configurations of logistic regression and SVM. Both models achieve similar accuracy on the test set, but they show different training characteristics:

TABLE I
PERFORMANCE COMPARISON BETWEEN LOGISTIC REGRESSION AND SVM

Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.8476	0.7238	0.5736	0.6400
SVM	0.8383	0.7536	0.4691	0.5782

Table I shows the performance of both models. Logistic Regression achieves a higher F1 score, Accuracy and Recall, while SVM shows better Precision. Overall, Logistic Regression demonstrates a better balance between precision and recall in this experiment.

IV. CONCLUSION

Our experiments demonstrate the effectiveness of stochastic gradient descent (SGD) compared to full-batch gradient descent. By using mini-batches, SGD provides several advantages:

- Faster convergence due to more frequent parameter updates
- Reduced memory requirements as only a subset of data is processed at once
- Better generalization through the inherent noise in sampling, which helps escape local minima

The optimal batch size depends on the specific problem, but our experiments suggest that moderate batch sizes (32-64) often provide a good balance between computational efficiency and convergence stability.

While both algorithms aim to find a linear decision boundary, they differ in their optimization objectives:

- Logistic regression minimizes the log-loss, focusing on correctly modeling the probability of class membership
- SVM maximizes the margin between classes, focusing on finding the most robust decision boundary

Our results show that SVM slightly outperforms logistic regression in terms of classification metrics. This is consistent with theory, as SVM's margin maximization approach often leads to better generalization, especially in high-dimensional spaces with limited training data.

However, logistic regression provides probability estimates, which can be valuable in applications requiring risk assessment or confidence scores. Additionally, logistic regression can be more easily extended to multi-class problems without requiring multiple binary classifiers.

Our comparison of SGD and Adam optimizers consistently shows the superiority of Adam in terms of convergence speed and final performance. Adam combines the advantages of:

- AdaGrad, which adapts learning rates based on historical gradients
- RMSProp, which uses a moving average of squared gradients
- Momentum, which accelerates convergence in relevant directions

This combination makes Adam particularly effective for non-convex optimization problems like neural networks, but our experiments show it also benefits classical algorithms like logistic regression and SVM.

In this laboratory, we implemented and compared logistic regression and SVM algorithms from scratch. Our experiments on the a9a dataset provide the following key insights:

- Parameter initialization significantly affects convergence speed, with random and normal initialization outperforming zeros initialization
- Adam optimizer consistently outperforms SGD for both logistic regression and SVM
- Smaller batch sizes lead to faster but noisier learning, while larger batches provide more stable but potentially slower convergence
- SVM with hinge loss slightly outperforms logistic regression in classification metrics, confirming its theoretical advantage in generalization

These findings deepen our understanding of the connections and differences between logistic regression and linear classification using SVM, as well as the impact of different optimization strategies on model performance.

Future work could explore kernel methods to extend SVM to non-linearly separable problems, multi-class extensions of these algorithms, and more sophisticated regularization techniques to further improve generalization.