Statistical Learning Theory & Method Report

Wenkai Sun[1)]

[1)](Shanghai Jiao Tong University School of Ocean and Civil Engineering, Shanghai 200240)

Abstract The experiment mainly targets a training set of 6666 samples, each with 1536 dimensions. Firstly, data balancing is performed through SMOTE, and then PCA is used to reduce the high-dimensional data to 63 dimensions. After data processing, models are trained using Support Vector Machines, Logistic Regression, Adaboost, and AlexNet algorithms. Logistic Regression achieves the highest accuracy rate of 94.327% on the test set.

Keywords Principle Component Analysis；Support Vector Machine；Logistic Regression；Adaboost；AlexNet

1. Data Processing

1.1 Data Preprocessing

The dataset contains a total of 6666 training samples and 2857 testing samples, each with a dimension of 1536, spanning 20 categories. The number of samples per category in the training set is as follows:

Tab. 1 The number of samples for each category in the training set

Category	Sample Number	Category	Sample Number
0	375	10	325
1	366	11	341
2	316	12	342
3	325	13	339
4	328	14	318
5	336	15	329
6	343	16	324
7	315	17	336
8	336	18	327
9	319	19	326

The dataset has an uneven number of samples across classes. To address this, the SMOTE oversampling method is employed. For each minority class sample, it calculates the distance to all other samples in the same class to find its K-nearest neighbors. The sampling rate N is set based on the imbalance ratio of the samples. Several samples are randomly selected from the K-neighbors, assuming the chosen dimension is $\tilde{x}$, and new samples are constructed using the following formula:

$$x_{new} = x + rand(0,1) \times (\tilde{x} - x) \qquad (1.1)$$

After applying SMOTE oversampling, the total number of samples obtained is 7500, with each category having 375 samples.

1.2 Data Dimensionality Reduction

The dataset contains samples with a high dimensionality of 1536 dimensions. High-dimensional spaces have a large variance; hence it is necessary to perform dimensionality reduction on the extracted 1536-dimensional features using Principal Component Analysis (PCA).

The fundamental idea is to assume a lower dimensional space of dimension d' for the sample set $D = \{x_1, x_2, \dots, x_m\}$, centralize all samples $x_i \leftarrow x_i - \frac{1}{m}\sum_{i=1}^{m} x_i$, calculate the covariance matrix of the centralized data matrix, XX^T, perform eigenvalue decomposition on the covariance matrix, and obtain the eigenvectors corresponding to the largest d' eigenvalues $\omega_1, \omega_2, \dots \omega_{d'}$, resulting in the projection matrix $W^* = (\omega_1, \omega_2, \dots, \omega_{d'})$.

The dimensionality reduction is generally specified by the user beforehand. In order to obtain a more reasonable choice of dimensionality k', cross-validation of the k-nearest neighbor classifier is performed in the low-dimensional space by taking different values of the parameter. For this purpose, 5-fold cross-validation is used to perform CV validation on dimensions from 20 to 1520 at intervals of 20, with the score being chosen based on accuracy (Accuracy). Based on the results, further CV validation is carried out for dimensions from 20 to 110 at intervals of 1.

Eventually, it is determined that a dimensionality of 63 achieves the best classification performance while ensuring that the dimensionality is as low as possible to reduce variance.

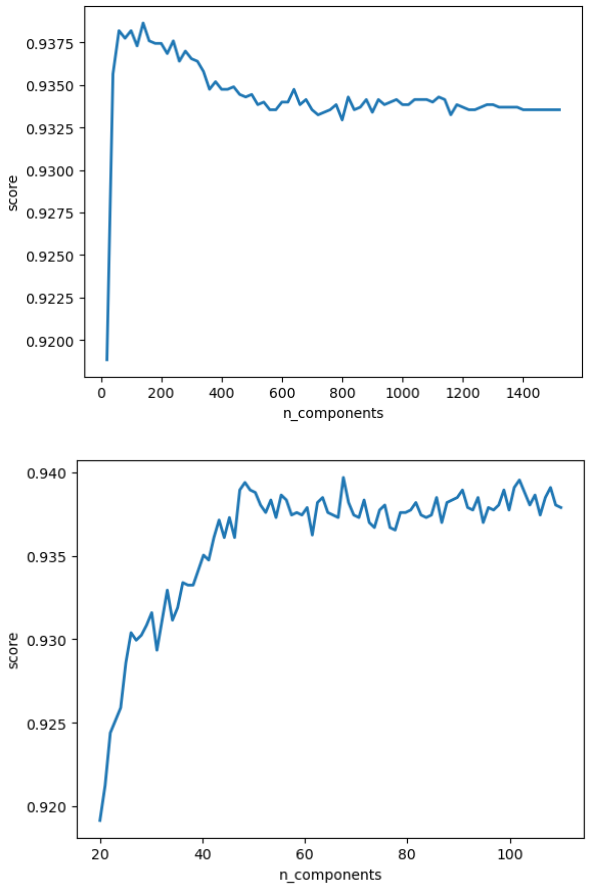

Fig. 1 The accuracy of the KNN model under different numbers of principal components in PCA

2. Model I SVM

2.1 The Principle of Support Vector Machine

Support Vector Machines (SVM) primarily include linear SVMs and nonlinear SVMs that use kernel tricks. For most practical data, nonlinear SVMs based on kernel functions are used to transform the problem into a linearly separable one. The advantages of SVM are that it is very efficient in high-dimensional spaces; it is still effective when the dimensionality of the data is greater than the number of samples; and it uses a subset of the training set for the decision function (known as support vectors), so it is also memory-efficient.

For multi-class problems, Support Vector Classification (SVC) is a tolerant "classification model" that, compared to strict classification problems, sets a certain margin band. Loss is not calculated for all samples falling within the margin band, and the model is optimized by minimizing the width of the margin band and the total loss. For linear SVMs, given a linearly separable training set, the separating hyperplane and decision function are learned by maximizing the margin or equivalently by solving the corresponding convex quadratic programming problem.

$$\omega^* \cdot x + b^* = 0 \tag{2.1}$$

$$f(x) = sign(\omega^* \cdot x + b^*) \tag{2.2}$$

The equivalent optimization problem is shown as follows:

$$\min_{\omega,b} \frac{1}{2}\|\omega\|^2 \tag{2.3}$$

$$\text{s.t. } y_i(\omega \cdot x_i + b) - 1 \geq 0, i = 1,2,\cdots,N \tag{2.4}$$

From a nonlinear classification training set, the classification decision function learned through kernel functions and soft margin maximization, or convex quadratic programming is called a nonlinear support vector machine.

$$f(x) = \text{sign}(\sum_{i=1}^{N} \alpha_i^* y_i K(x, x_i) + b^*) \tag{2.5}$$

Due to the excellent characteristics of nonlinear classifiers when in use, the following presents the algorithm for solving the nonlinear support vector machine for binary classification problems.

Algorithm 1. Non-linear Support Vector Function

Input: Training dataset $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, in which $x_i \in X = \mathrm{R}^n, y_i \in Y = \{-1, 1\}, i = 1, 2, \ldots, N$;

Output: Classification Decision Function

(1) Select an appropriate kernel function $K(x, z)$ and appropriate parameters C, construct and solve the optimization problem

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{N} \alpha_i \tag{2.6}$$

$$\text{s.t. } \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{2.7}$$

$$0 \leq \alpha_i \leq C, i = 1, 2, \ldots, N \tag{2.8}$$

Obtain the optimal solution $\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)^T$

（2） Select a positive component of α^*, $0 < \alpha_j^* < C$, calculate

$$b^* = y_j - \sum^{N} \alpha_i^* y_i K(x_i, x_j) \tag{2.9}$$

（3） Construct the decision function.

$$f(x) = \text{sign}(\sum_{i=1}^{N} \alpha_i^* y_i K(x, x_i) + b^*) \tag{2.10}$$

2.2 SVM Model Parameters Selection

The model parameters of the SVM are shown as follows:

Tab. 2 SVM Model Parameters

Parameter	Value
Punishment Parameter C	Default 1
Kernel Function	'linear', 'poly', 'rbf', 'sigmoid'
Kernel Function Parameter gamma	Default 1/n_features

Within this, the penalty parameter is equivalent to penalizing the slack variable. The smaller the value of the slack variable, the greater the penalty for misclassification, which increases the fitting accuracy on the training set but reduces the generalization ability. Conversely, the smaller the penalty parameter, the more it allows for errors, considering them as noise points, which enhances the generalization ability.

The expressions for the four types of kernel functions are as follows:

- linear: $u^T v$
- poly: $(gamma \cdot u^T v + coef\,0)^{\deg ree}$
- rbf: $\exp(-gamma|u-v|^2)$
- sigmoid: $\tanh(gammau^T v + coef\,0)$

For the above parameters, the grid search method (GridSearchCV) is adopted, the cross-validation set is set to 5, and the specific search grid parameters are as follows:

parameters={'kernel':['linear','rbf','sigmoid','poly'],'C':np.linspace(0.1,20,50),'gamma':np.linspace(0.1,20,20)

The best validation set accuracy is 94.75%, and the final selected parameters are as follows:

```
In [16]: model.best_params_
Out[16]: {'C': 0.1, 'gamma': 0.1, 'kernel': 'linear'}
```

2.3 SVM Model Evaluation

The accuracy of the support vector machine is 99.46% on the training set, 95.03% on the validation set, and 93.697% on the test set. In addition, the final prediction results on the test set are obtained from 200 meta-classifiers, and their roc_auc_score is shown below.

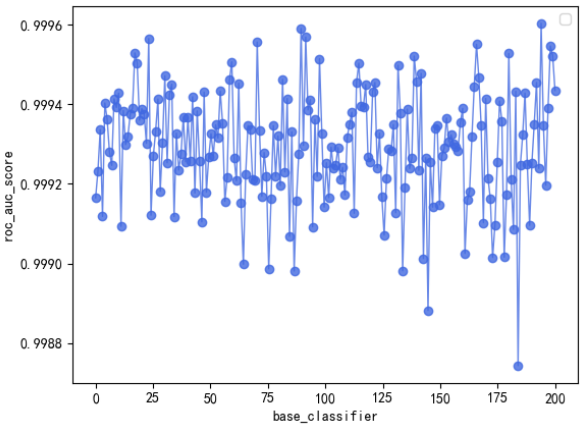

Fig. 2 SVM Meta-Classifier ROC_AUC_SCORE

3. Model II Logistic Regression

3.1 The Principle of Logistics Regression

Logistic Regression is a classic classification method in statistical learning. The polynomial Logistic regression model is applicable to multi-class situations, and the regression model is the following conditional probability distribution:

$$P(Y=k|x) = \frac{\exp(\omega_k \cdot x)}{1+\sum_{k=1}^{K-1}\exp(\omega_k \cdot x)}, \quad k=1,2,\ldots,K-1 \tag{2.11}$$

$$P(Y=K|x) = \frac{1}{1+\sum_{k=1}^{K-1}\exp(\omega_k \cdot x)}, \quad x \in \mathrm{R}^{n+1}, \omega_k \in \mathrm{R}^{n+1} \tag{2.12}$$

Logistic regression uses the log-likelihood method to estimate model parameters.

3.2 Logistic Parameter Selection

In this experiment, the Logistic Regression model's solving algorithm chosen is Stochastic Average Gradient (SAG). Each iteration only uses a part of the samples to compute the gradient, which is suitable for cases with a large number of sample data.

To enhance the generalization performance of Logistic Regression, regularization is applied to the

Logistic Regression by using L2 constraints. The penalty coefficient C is determined by 10-fold cross-validation, with the scoring metric being the mean squared error, and the results are visualized as shown below.

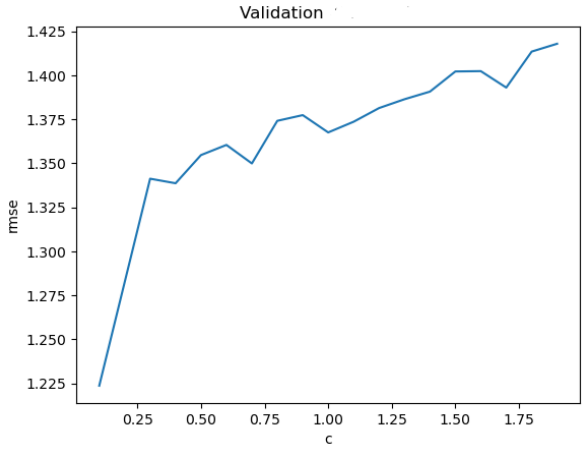

Fig. 3 Logistics penalty parameter cross-validation selection

3.2 Logistic Model Evaluation

The Logistic model uses an ensemble method, combining votes from 200 Logistic regression classifiers to determine the final model on the test set, which is the best model of this experiment. The average accuracy of the 200 models on the training set is 98.88%, the average accuracy on the validation set is 96.08%, and the accuracy on the test set is 94.33%.

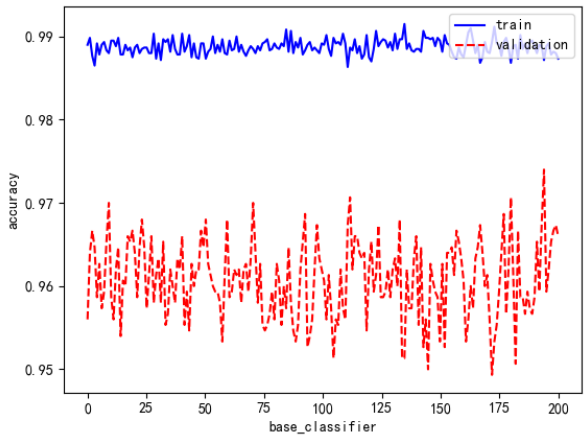

Fig. 4 The accuracy of 200 Logistic classifiers

4. Model III Adaboost

4.1 The Principle of Adaboost

Adaboost is a boosting algorithm that learns a series of weak classifiers from training data and combines these weak classifiers linearly into a strong classifier. The specific learning algorithm is shown below (taking binary classification as an example)

Algorithm 1. Adaboost

Input：Training Dataset $T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$，in which $x_i \in X=\mathrm{R}^n, y_i \in Y=\{-1,1\}, i=1,2,\ldots,N$；weak learning algorithm

Output：Classifier $G(x)$。

（1） Initialize the weight distribution of the training data

$$D_1=(\omega_{11},\ldots,\omega_{1i},\ldots,\omega_{1N}),\omega_{1i}=\frac{1}{N},i=1,2,\ldots,N \quad (2.13)$$

（2） For m=1, 2, …, M

a) Learn using a training dataset with weighted distribution D_m, obtaining base classifiers

$$G_m(x):X\to\{-1,+1\} \quad (2.14)$$

b) Calculate the classification error rate of $G_m(x)$ on the training set

$$e_m=\sum_{i=1}^{N}P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}\omega_{mi}I(G_m(x_i)\neq y_i) \quad (2.15)$$

c) Calculate the coefficient of $G_m(x)$

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m} \quad (2.16)$$

d) Update the weight distribution of the training dataset

$$D_{m+1}=(\omega_{m+1,1},\ldots,\omega_{m+1,i},\ldots,\omega_{m+1,N}) \quad (2.17)$$

$$\omega_{m+1,i}=\frac{\omega_{m,i}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)) \quad (2.18)$$

which makes D_{m+1} a probabilistic distribution

（3） Construct the linear combination of base classifiers

$$f(x)=\sum^{M}\alpha_m G_m(x) \quad (2.19)$$

Finally, we can get the classifier

$$\begin{aligned}G(x)&=sign(f(x))\\&=sign(\sum_{m=1}^{M}\alpha_m G_m(x))\end{aligned} \quad (2.20)$$

In this experiment, decision trees were chosen as the base classifiers, hence this method is also referred to as Boosting Tree. The decision model of Boosting Tree can be represented as the additive model of decision trees.

$$f_M(x) = \sum_{m=1}^{M} T(x;\Theta_m) \tag{2.21}$$

4.2 Adaboost Parameter Selection

Test the predictive performance of Adaboost Classifier as influenced by the number of base classifiers, the number and type of base classifiers, the learning rate, and the combination of the learning rate and algorithmic parameters, as shown in the following figures.

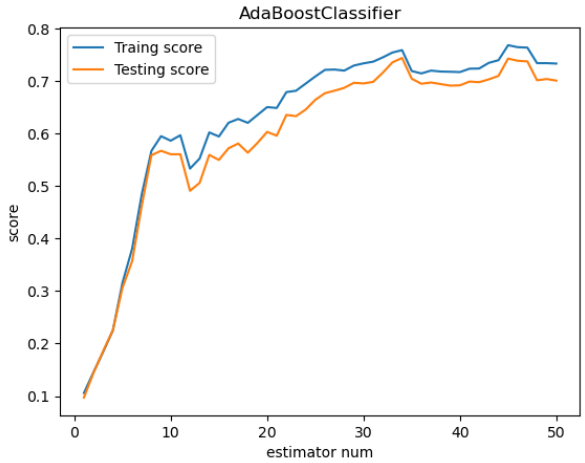

Fig. 5 The Influence of the Number of Base Classifiers on the Performance of the AdaBoost Classifier

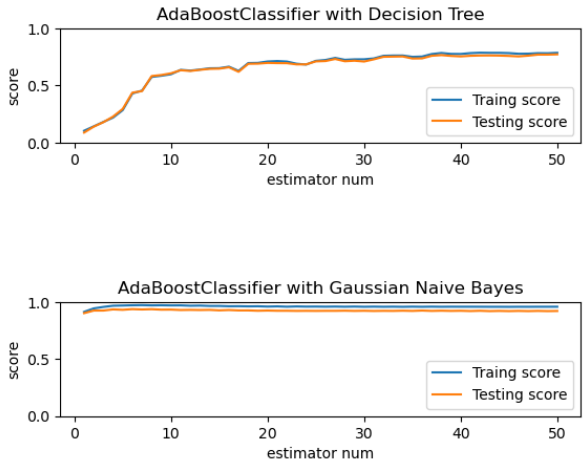

Fig. 5 The Influence of the Number and the Type of Base Classifiers on the Performance of the AdaBoost

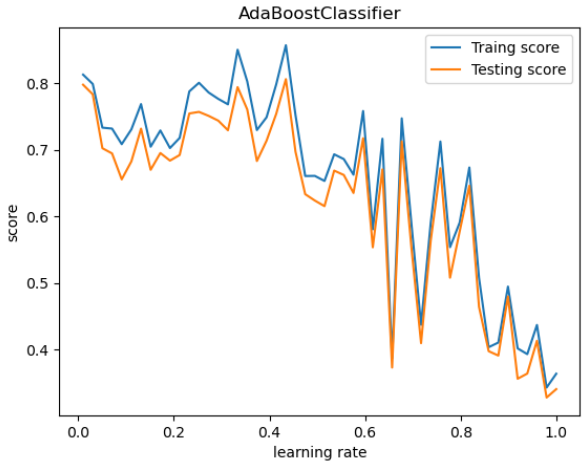

Fig. 6 The Influence of the Learning Rate on the Performance of the AdaBoost

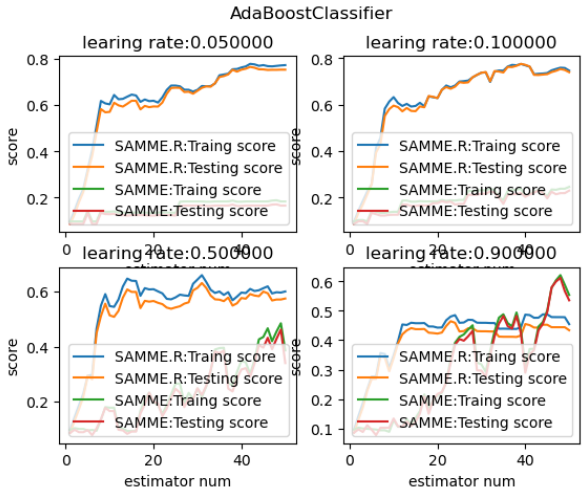

Fig. 7 The Influence of Learning Rate, Algorithm, and Base Classifiers on the Performance of AdaBoost

Additionally, due to the numerous adjustable parameters of the base classifier, the decision tree classifier (max_depth=15, min_samples_split=15) was ultimately selected. The number of learners is 100, the learning rate is 0.05, and the algorithm chosen is SAMME.R.

4.3 Adaboost Model Evaluation

The AdaBoost model evaluation uses the Bootstrap method, which simulates cross-validation. The leave-one-out bootstrap is used to estimate the expected prediction error, which is defined as follows:

$$\sum Err^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L\left(y_i, \hat{f}^{*b}(x_i)\right) \tag{2.22}$$

Since the leave-one-out bootstrap may produce an overestimate of the true error, it is necessary to use the '.632+' estimate, which is a compromise between the error rate of the leave-one-out bootstrap method and an overfitting procedure, with the specific estimation method as follows:

$$\sum Err^{(.632+)} = (1-\widehat{w}) \cdot \overline{err} + \widehat{w} \cdot \sum Err^{(1)} \quad (2.23)$$

$$\widehat{w} = \frac{.632}{1-.632\hat{R}} \quad (2.24)$$

$$\hat{\gamma} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'} NL\left(y_i, \hat{f}(x_{i'})\right) \quad (2.25)$$

$$\hat{R} = \frac{\sum Err^{(1)} - \overline{err}}{\hat{\gamma} - \overline{err}} \quad (2.26)$$

in which, γ is non-information error rate, $\hat{R}$ is relative overfitting rate.

The final prediction error estimate for the model is 7.42%.

```
#无信息错误率
prediction=clf0.predict(train_feature)
prediction0=list(prediction)
train_labels0=list(train_labels)
count1=np.zeros((20,1))
count2=np.zeros((20,1))
for i in range (20):
    count1[i]=prediction0.count(i)/6666
    count2[i]=train_labels0.count(i)/6666
gamma=0
for i in range(20):
    gamma+=count1[i]*(1-count2[i])
R=(Err1-err)/(gamma-err)
w=0.632/(1-0.368*R)
Err_632=(1-w)*err+w*(1-Err1)
print("Err_632=",Err_632)
```

```
Err_632= [0.07416818]
```

5. Model IV AlexNet NN

5.1 The Pinciple of AlexNet

AlexNet has an 8-layer structure, with the first 5 layers being convolutional layers and the last three layers being fully connected layers. The structural features of AlexNet are as follows:

1) AlexNet uses the non-linear activation function ReLU, which has a faster rate of gradient descent during the training phase compared to functions like sigmoid or tanh.
2) AlexNet operates on dual GPUs, with each GPU responsible for half of the network's computation.
3) Local Response Normalization (LRN) is adopted, which, when added to the ReLU layer, can form a kind of lateral inhibition, thereby improving the generalization ability of the network;
4) Overfitting is prevented through Dropout, Data augmentation, and overlapping pooling.

5.2 AlexNet Parameter Selection

AlexNet was initially used for processing data with spatial information such as images, therefore it includes convolutional layers. Given that the data is extracted features, flattened kernel parameters are used. The specific network structure is as follows:

Tab. 3 AlexNet Network Structural Parameters

Layer	Structural Parameters
Convolutional Layer 1	filters=96, kernel_size=(11,1), strides=4, activation='relu' pool_size=(3,1),strids=2
Convolutional Layer 2	filters=256, kernel_size=(5,1), strides=1, activation='relu' pool_size=(3,1),strids=2
Convolutional Layer 3	filters=384, kernel_size=(3,1), strides=1, activation='relu'
Convolutional Layer 4	Filters=384, kernel_size=(3,1), strides=1, Activation='relu' Pool_size=(3,1),strides=2
Convolutional Layer 5	Filters=256, kernel_size=(3,1), strides=1, Activation='relu' Pool_size=(3,1),strides=2
FC 1	4096 neurons+ ReLU，dropout=0.5
FC 2	2048 neurons + ReLU，dropout=0.5
FC 3	20 neurons +Softmax

5.3 AlexNet Model Selection

When training, n_epoch=40 and n_batch=250. After 40 epochs of training, the error on the training set is 0.0675 with an accuracy of 0.9826, while the error on the validation set is 0.2585 with an accuracy of 0.9220. The curves of training error and accuracy with respect to epoch are shown in the following figure. The accuracy stabilizes approximately after 10 epochs, while the error reaches its minimum around 20 epochs.

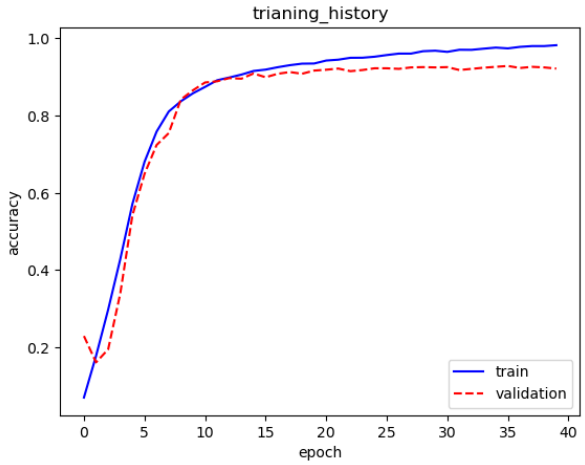

Fig. 8 The relationship between epoch and accuracy for the training and validation sets

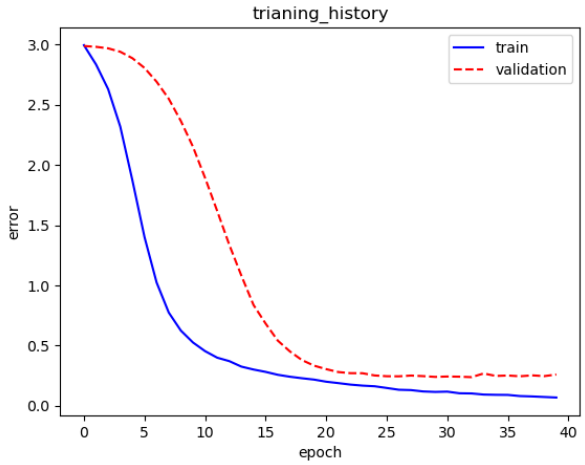

Fig. 9 The relationship between epoch and error rate for the training and validation sets

6. Test Set Noise Processing

Since the training set data is pure, while the test set data has noise added, to enhance the algorithm's generalization ability and improve model robustness, the main methods for test set noise handling are as follows:

6.1 Dimensionality Reduction of Data

The sample dimension is 1536 dimensions, and the bias term in the root mean square error is very large due to the high feature dimensionality, which greatly affects the model's susceptibility to noise interference. This report adopts PCA dimensionality reduction, and similar methods include subset selection, independent component analysis (ICA), etc. Taking the SVM model as an example, in this experiment, when PCA dimensionality reduction is used, the model achieves an accuracy of up to 95% on the validation set, while the accuracy on the test set is only 31.09%. However, after using PCA dimensionality reduction, the accuracy on the validation set remains at 95%, but there is a significant improvement in accuracy on the test set, reaching 93.7%. Therefore, data dimensionality reduction can significantly improve the model's generalization ability.

6.2 Training Set Noise Processing

Since the training set is pure data without noise interference, the accuracy on the validation set can generally reach 95% and above. However, the accuracy on the test set is generally slightly lower than on the validation set. To enhance the model's robustness, this Logistics model adopts an ensemble method, training a total of 200 classifiers. Each trainer's data is subjected to additive noise following a normal distribution, with a mean of 0 and a standard deviation 0.05 times that of each feature's standard deviation. Finally, voting is conducted on the test set, and the mode of the votes is selected as the predicted category. The Logistics classifier, when not using this strategy, only had an accuracy of 92.16% on the test set. After applying this strategy, the accuracy on the test set increased to 94.33%, which is also the highest accuracy training model in this experiment.

7. Conclusion

In this experiment, four algorithms were tested: Support Vector Machine, Logistic Regression, AdaBoost, and AlexNet network. The highest accuracy achieved on the test set was 94.327% by Logistics Regression. More evaluation and improvements were primarily focused on the first two basic algorithms. Compared to AlexNet, the models of the first two algorithms have lower complexity, higher overall training efficiency, and better training results. The training time of AlexNet network is significantly higher than that of the first two algorithms, and the accuracy

on the test set is also relatively low. Additionally, AlexNet contains convolutional layers mainly used for processing image data containing spatial information. However, in this experiment, the samples are 1×1536-dimensional data, so the convolutions used are only flattened convolutional kernels. It was also attempted to set the "flattened" features as two-dimensional "images", which showed relatively good performance on the validation set. However, the accuracy on the test set was only around 15%, indicating severe overfitting of the model.

Tab. 4 Summary of Accuracies of 4 Classifiers

Classifiers	Accuracy	
SVM	Training Set	99.46%
	Validation Set	95.03%
	Test Set	93.697%
Logistic Regression	Training Set	98.88%
	Validation Set	96.08%
	Test Ste	94.33%
Adaboost	Training Set	100%
	Validation Set	92.95%
	Test Set	90.36%
AlexNet	Training Set	98.26%
	Validation Set	92.20%
	Test Set	89.15%

In addition, the generalization ability of a model is the criterion for evaluating its quality. Through this experiment, I have learned methods for model evaluation and selection. I deeply understand how data dimensionality reduction improves the generalization ability of a model and how adding noise to the training dataset and using ensemble methods enhance the model's generalization ability.