An Approach to Multi-class Classification Problems Using the Support Vector Machine(SVM) and the Min-Max-Module Strategy

Zhang Yifei*
Sang Rui†

May 16, 2020

*Student ID: 518021911091
†Student ID: 518021911085

Abstract

This article is for the Course of Machine Learning, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. In this paper, we will show that how to implement a multi-class classification task by using Support Vector Machine(SVM) and the Min-Max-Module strategy. Also, the advantages of the Min-Max-Module strategy are illustrated.

Please note that the codes in this article are based on Python 3.7.

Keywords: SVM; multi-class classification; Min-Max-Module strategy; machine learning

1 Introduction

1.1 Classification

With the rapid growth of computer technology and the emerging online information, classication job has become a crucial techique for the classification of various and massive data. Normally, classification algorithms are used to distinguish between two or more types of data , for example, to determine whether an email is a spam.

There are multiple approaches that is commonly used for classification: Logistic Regression, K Nearest Neighbor, Decision Tree, Naive Bayes and Support Vector Machine. Though the differrent means of classification may vary, the key processes are mostly the same as illustrated in Figure 1.

Figure 1: The key process of classification

The classification is the automatic processing and interpretation of patterns/features through the computer using mathematical methods. It contains two parts: feature selection/extraction and data classification. However, the diversity and complexity of the original data bring much difficulty to the classification, especially for the case when a large amount of data is implemented.

1.2 Support Vector Machine (SVM)

The Support Vector Machine(SVM) was formally proposed by Vapnik at the 1995 Computer Learning Theory Conference. With 25 years of development, it has become a classic classification method. The theoretical basis of the method is the principle of VC dimension and structural risk minimization. It overcomes the local minima and dimensional disasters, thus has a great advantage for solving practical problems such as nonlinearity, small samples, high dimensionality and local minimum points[xuegong2000introduction].

In simple terms, SVM achieves classification by finding a hyperplane of given data. The hyperplane may be in the space where the data is located or mapped to a higher dimensional space by the kernel function. The division of the hyperplane may be strict (hard) or not so strict (soft).

While learning from large-scale samples, SVM still has some deficiencies which manifested in excessive memory consumption, slow training speed and low accuracy[platt1999fast].

1.3 Multiclass Problems

SVM solves two-class problems, that is, it divides the given data into two classes. If we want to apply SVM in multiclass problems, two strategies can be conducted: one-vs-rest strategy and one-vs-one strategy.

1.3.1 One-vs-Rest Strategy

One-vs-rest strategy means that each time, consider one of the K classes and the rest of the K classes as two elements of a two-class problem. It divides the K-class problem into K two-class problems. For each two-class problem, a SVM is trained. The final result depends on the discriminant functions of each SVM. For a given data x as input, y is the output value of the discriminant function. And x belongs to the class whose discriminant function has the largest y.

1.3.2 One-vs-One Strategy

One-vs-one strategy means that considering any two classes in the multiclass problem as two classes in a two-class problem. Therefore, the K-class problem is divided into K(K-1)/2 two-class problems. For each two-class problem, a SVM is trained. When a data x is given, each SVM will predict a class in which x belongs to. Generally, voting strategy is used to determine which class x belongs to.

1.4 Min-Max-Module Strategy

In order to solve large-scale multiclass problems efficiently and effortlessly, Lv and Ito(1999) proposed a Min-Max-Modular(M^3) network. The subsequent research has verified its validity.

For multiclass problem, a module selection procedure for combination of binary classifiers is proposed for a general combination procedure under the one-vs-one task decomposition strategy**赵海 2005 最小最大模块化分类器研究**.

Generally, the overview of the Min-Max-Module is shown in Figure 2**lu2000emergence**. More details will be discussed in the next section.

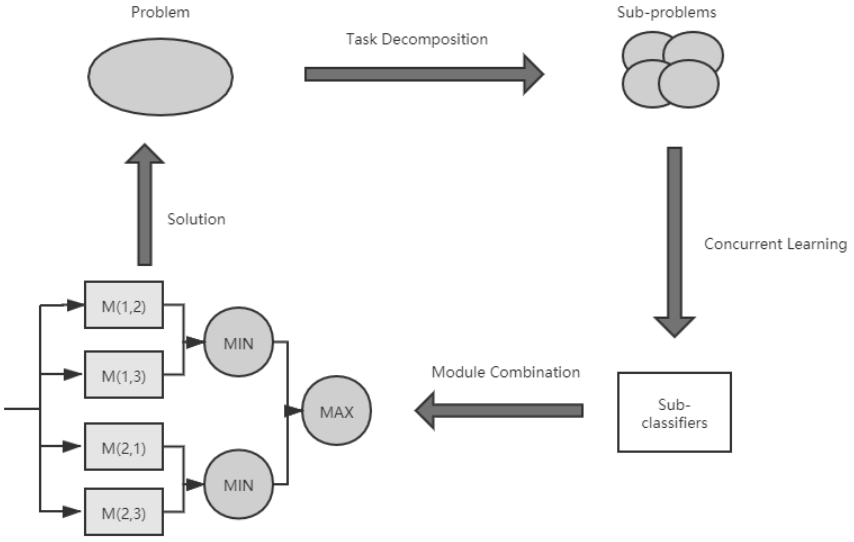

Figure 2: An overview of the Min-Max-Module strategy

2 Algorithm and Code Explanation

2.1 Data Pretreatment

We use sklearn, a popular third-party package for machine learning, as well as numpy tools to help solving these problems. We are provided with four data files[1] in npy format: test_data, test_label, train_data and train_label. Among these test_data and train_data are sets of 310 features and have the following format after loading into numpy array:

feature1 feature2 feature3 ...feature310 (of data1)
feature1 feature2 feature3 ...feature310 (of data2)
.....
feature1 feature2 feature3 ...feature310 (of datan)

Test_label and train_label are the label sets corresponding to the two feature sets respectively, and are one-dimensional arrays when loaded into numpy array. This is a three-class problem and there are three kinds of labels: 1,0 and -1.

In order to facilitate the later data selection and processing, we disrupt the order of both the training set and test set (note that same random seeds should be used for feature sets and label sets to ensure the correspondence does not change). Considering the similar value range of each feature, we do not use data normalization in the pretreatment.

2.2 Problem 1

The first problem is to solve the three-class classification problem in the given dataset using SVM Classifiers and the one-vs-rest strategy. The principle of classification has been explained in Introduction. We use sklearn's function SVC(decision_function_shape='ovr') to train the one-vs-rest SVM model and solve this problem directly.

2.3 Problem 2

The second problem is to solve the three-class classification problem using min-max Module SVM and part-vs-part task decomposition method. Explanation of the our algorithm is given below. At the end of this section we also draw a process diagram to make the whole process intuitive. Note that the following algorithm is based on the paper**lu2000emergence王开安 2005 最小最大模块化支持向量机及其在文本分类中的应用**.

2.3.1 Training Feature Set are Divided by Labels

According to the corresponding labels, the training feature set is divided into three small feature sets: set1,set2 and set3, with corresponding labels 1, 0 and

[1]The data that we use is kindly offered by Professor Chen. It can be downloaded by: https://pan.baidu.com/s/1AfZKtPzbjoFSL67KzldD8A Password: rtx0

-1 respectively. For easy indexing, three small feature sets are placed in a list called sets.

2.3.2 Three-class Problem To Two-class Problems

We divide this three-class problem into three two-class problems, which are two-class problem between set1 and set2, two-class problem between set2 and set3, and two-class problem between set1 and set3. Consider the label of one of the sets in each pair (which in the code we call "positive set") as 1, and the label of the other set (which in the code we call "negative set") as -1. That is:

$$X^+ = \left\{\left(x_i^+, +1\right)\right\}_{i=1}^{l^+}$$

$$X^- = \left\{\left(x_i^-, -1\right)\right\}_{i=1}^{l^-}$$

l^+ is the number of samples of positive set , l^- is the number of samples of negative set. X^+ is the"positive class", X^- is"negative class".

2.3.3 Further Decomposition of Positive Set and Negative Set

Further decompose positive set and negative set. The positive set is decomposed into N^+ subsets, and the negative set is decomposed into N^- subsets. N^+ and N^- are determined by the number of samples l^+ and l^- and the preset parameters ρ. Their relationship is as follows:

$$N^r = \left[\frac{l^r}{\rho}\right], r \in \{+, -\}$$

where $[x]$ means to round x.

There are many different strategies to split the positive set and the negative set. We split them using the numpy function numpy.array.split(). The effect of this function is to decompose the original set into mutually disjoint subsets, and the number of samples of all the subsets differ from each other by no more than 1. For example, for a set of 10 samples, the number of samples of the subsets using function numpy.array.split() is 3,3,2,2. Thus, it is very simple to split the sets using function numpy.array.split(), and the result is very uniform.

The principle of the numpy.array.split() function can be expressed as follows:

$$l_i^r = \begin{cases} \left\lfloor \frac{l^r}{N^r} \right\rfloor & i > (l^r \bmod N^r) \\ \left\lfloor \frac{l^r}{N^r} \right\rfloor + 1 & \text{others} \end{cases}, r \in \{+, -\}$$

2.3.4 Sub-two-class Problems and Max-Mins Strategy

After the positive set is decomposed into N^+ subsets and the negative set is decomposed into N^- subsets, the original two-class problem becomes $N^+ \times N^-$ sub-two-class problems. Then set T of size $N^+ \times N^-$ can be obtained. The training set of sub-two-class problem $T_{i,j}$ (including feature set and label set) can be intuitively represented by Figure 3, where $1 \le i \le N^+, 1 \le j \le N^-$.

Figure 3: Training set of sub-two-class problem $T_{i,j}$

For each sub-two-class problem, a small SVM is trained by using sklearn's SVC module. Then the set (matrix) of small vector machines of size $N^+ \times N^-$ can be obtained, which we call set M. If the feature set of test set is input into M, the predict label set is obtained, which we call set Predict_val. The maximum and minimum strategy are adopted for Predict_val, that is, taking the minimum value of each row of Predict_val and then taking the maximum value of the result. Then the predict label of the original two-class problem can be obtained. The plus or minus of this label determines the result of the original two-class problem.

2.3.5 Vote and Accuracy

Among set1, set2 and set3, the set with the most votes is the final predict set of the feature set for the test data.

The accuracy can be calculated by comparing the final predict set with the set to which the feature set originally belongs.

2.3.6 Process Diagram

The process diagram of the whole process is shown in Figure 4.

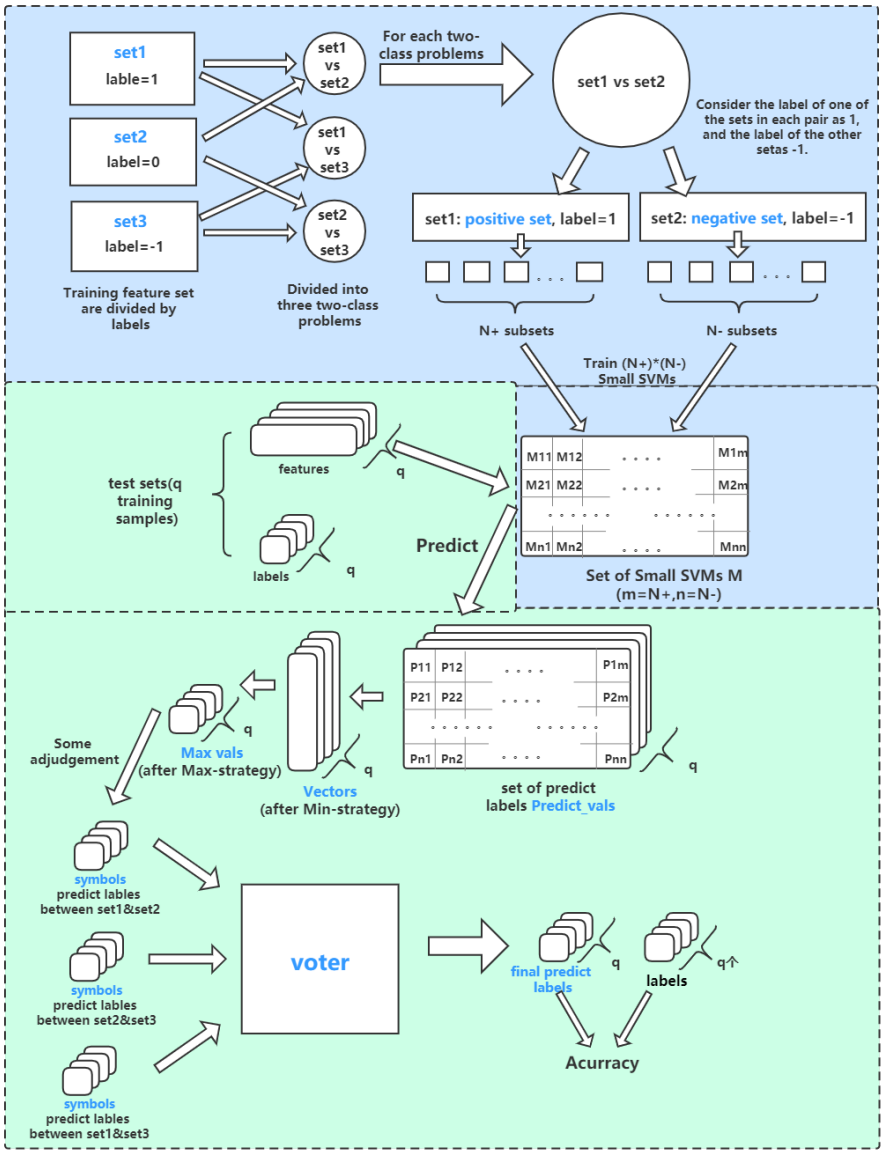

Figure 4: Process diagram of problem 2
Notes: Words in blue colour are names of variables or matrixs used in the code; Blue background means training phase and green background means testing phase

3 Model Optimization

3.1 Problem 1 Optimization

From previous training, we learned that the time complexity is particularly high for the SVM algorithm. The process of training the model takes a lot of time due to the overfitting phenomenon and the intricacy of the dataset, while the accuracy rate is hovering at 45%. By adjusting the parameters respectively, we can have a better view of how to optimize the model to get higher accuracy.

3.1.1 Impact of The Number of Training Samples on Accuracy

We assign kernel = 'linear', set other parameters as default. By changing different size of training samples, we obtain the accuracy respectively as shown in Figure 5.

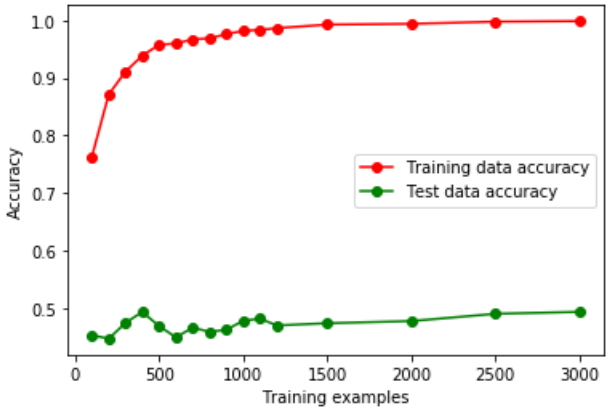

Figure 5: Impact of the number of training samples on accuracy for problem 1

As the number of the training data increases, the training data accuracy increases as well and flattened.

Further analysis show that when kernel = 'linear', and the number of training samples is 400, the optimal accuracy is 49.30%.

3.1.2 Impact of The Kernel Function on Accuracy

We tested four kinds of kernels: linear, rbf, poly and sigmoid. The accuracies are shown in the Figure 6.

The accuracy obtained by the linear kernel and the polynomial kernel is relatively close, while the sigmoid kernel offered poor performance.

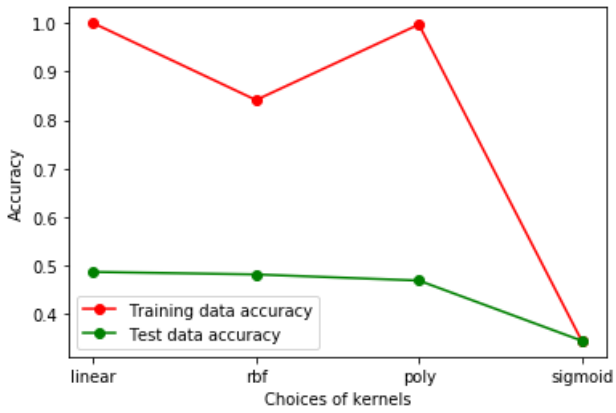

Figure 6: Impact of the kernel on accuracy for problem 1

3.1.3 Detailed Parameters Adjustments of Kernel Radial Basis Function(rbf)

Here, we would like to go far into the case of kernel 'rbf', to find out how the rbf kernel performs under different parameters C and γ.

By setting the size of training data at 400 and adjusting C and γ respectively, the ouput accuracies are shown in the Table 1 and Figure 7. We select C = [0.1, 1, 2, 3, 4, 5,6, 7, 9, 11] and γ = [0.00001, 0.0001, 0.01].

C	**γ=0.00001**	**γ=0.0001**	**γ=0.01**
0.1	0.329703	0.329703	0.329703
1	0.329703	0.460627	0.329703
2	0.329703	0.464822	0.335443
3	0.424934	0.482043	0.336032
4	0.420812	0.517883	0.335369
5	40.42957	0.521195	0.335369
6	50.398366	0.519797	0.335369
7	60.39844	0.518104	0.335369
9	70.45577	0.514645	0.335369
11	90.472917	0.520459	0.335369

Table 1: Accuracy of different C and γ of rbf kernel

From the Table 1 and Figure 7 and our further testing, we can conclude that when training size = 1200, C = 6.0, γ = 0.0001, the test data has a highest accuracy of 57.81%, which is close to the highest accuracy obtained by the linear kernel.

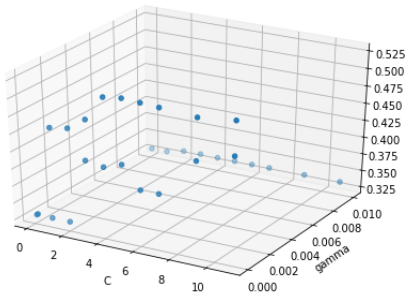

Figure 7: Accuracy of different C and gamma of rbf kernel

3.2 Problem 2 Optimization

3.2.1 Impact of The Total Number of Training Samples and The Size of Sub-Groups on Accuracy

Assigned linear kernel, we set different size of training samples and different size of sub-groups to test how they influence the final accuracy. The size of training samples is designated respectively as [330, 600, 900, 1200, 1600, 2000, 3000, 4000, 5000, 6000, 8000, 10000], and the size of sub-groups is designated respectively as [20, 40, 60, 80].

		The size of sub-group			
		20	40	60	80
The size of training sample	330	0.4808	0.5394	0.5394	0.4429
	600	0.5258	0.5664	0.4962	0.4287
	900	0.5076	0.4725	0.5399	0.5190
	1200	0.4981	0.5753	0.5388	0.5333
	1600	0.5077	0.5073	0.5204	0.4870
	2000	0.5789	0.5448	0.5394	0.5012
	3000	0.5098	0.5187	0.5094	0.5176
	4000	0.5568	0.5391	0.5483	0.5123
	5000	0.5286	0.4511	0.4823	0.5217
	6000	0.5237	0.5239	0.5367	0.5036
	8000	0.5061	0.4614	0.4538	0.5025
	10000	0.5086	0.5198	0.4895	0.4848

Table 2: The impact of the size of training samples and the size of sub-groups on accuracy

From Table 2. and our further testing, the highest accuracy 62.1% occurs when the size of training samples is 1200 and the size of sub-groups is 30.

3.2.2 Impact of The Kernel Function on Accuracy

Same as problem 1, we set four kernels: linear, rbf, poly and sigmoid. The accuracies of these four kernel are shown in Figure 8.

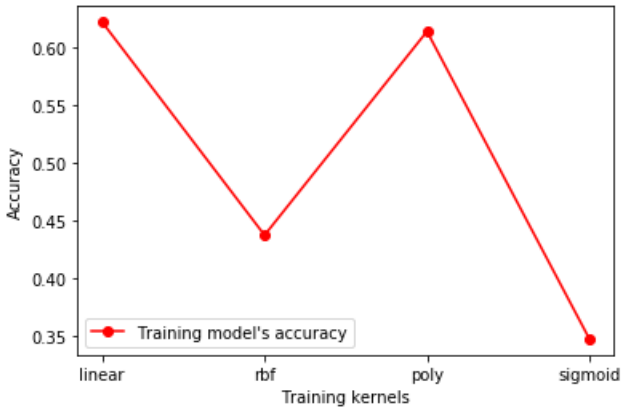

Figure 8: The accuracy of different kernels

As shown in the Figure 8, the linear and rbf kernel have similar values, while the rbf and sigmoid kernel do not behave as good as the previous two kernels.

3.2.3 Detailed Parameters Adjustments of Kernel Radial Basis Function(rbf)

Same as problem 1, we assign rbf kernel. Meanwhile, adjust the value of C and γ, to test how these two parameters influence the final accuracy. We set C = [0.1, 1, 2, 3, 4, 5, 6, 7, 9, 11], γ = [0.00001, 0.0001, 0.01] for cross validation. The results are shown in Table 3 and Figure 9.

C	**γ=0.00001**	**γ=0.0001**	**γ=0.01**
0.1	0.4561	0.5483	0.4217
1	0.4427	0.4566	0.4561
2	0.4125	0.4561	0.5425
3	0.4561	0.5776	0.4204
4	0.4532	0.4325	0.4707
5	0.4294	0.4561	0.5419
6	0.4561	0.5562	0.4277
7	0.5578	0.4223	0.4722
9	0.4296	0.4561	0.5452
11	0.4561	0.5569	0.4335

Table 3: The accuracy of different C and γ for rbf kernel

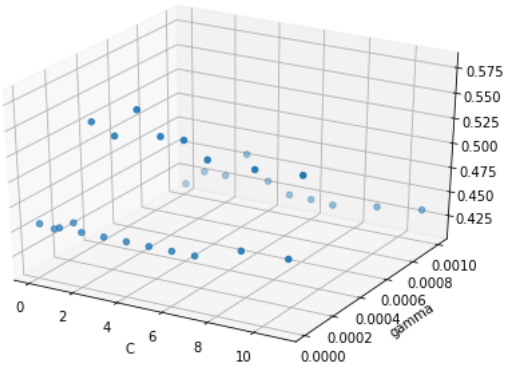

Figure 9: Impact of different C and γ on the accuracy of rbf kernel

From Table 3 and Figure 9, it can be concluded that the perferable parameter C for rbf kernel should be 3.0 and γ should be 0.0001.

4 Conclusion

After multiple parameter adjustments and trail runs, we finally ended up with the best results.

In problem 1 and problem 2, the linear kernel function performs better than the rbf kernel. By applying the Min-Max-Module, the efficiency and the generalization performance of classification algorithm have been significantly improved.

4.1 Conclusion For Problem 1

For linear kernel, we set the size of training samples 400 to get highest accuracy of 49.30%.

For rbf kernel, we set the size of training samples 1200, C = 6.0, $\gamma = 0.0001$ to get highest accuracy of 57.81%.

4.2 Conclusion For Problem 2

For linear kernel, we set the size of training samples 1200 and the size of sub-groups 30 to get highest accuracy of 62.16%.

For rbf kernel, we set the size of training samples 1200, the size of sub-groups 30, C = 3.0, $\gamma = 0.0001$ to get highest accuracy of 57.76%.

Acknowledgements

After going through a lot of difficulties, this project is finally get done well. As it is the very first time for us to solve the machine learning related problems, we carefully reviewed detailed contents in the class and searched the scientific literature for previous studies. Upon the completion of this assignment, we are grateful to those who have offered us encouragement and support during the assignment.

First, special acknowledgement is given to our professor Chen Li and assistant Zhang Xueheng, who have provided such a helpful and enlightening course, helping us along the project responsibly.

Also, particular thanks go to our classmate, Wang Haojian, without whom we could not have come up with various ideas and optimized our model.

Last but not least, we would like to express our heartfelt gratitude to Shanghai Jiao Tong University, for providing the course of machine learning.

Division of Work

Zhang Yifei:

- Write code of problem 2
- Modify code of problem 1
- Optimize problem 2
- Write part 2, 4, 6 of the article

Sang Rui:

- Write code of problem 1
- Modify code of problem 2
- Do the validation and optimization of problem 1 & 2
- Write part 1, 3, 5 of the article
- Compile the article in the form of Latex