# Neural Network Theory and Applications | Assignment#2

Shangyu Liu (020033910052)

April 12, 2021

## 1 Introduction

In this assignment, support vector machine (SVM) and Min-Max-Modular SVM will be used to deal with multi-class classification problems. SVMs usually handle binary classification tasks. If there are several classes to recognize, some strategies are needed. The most common methods are one-vs-one strategy, one-vs-rest strategy, and part-vs-part strategy.

Two problems are given below. The dataset used in this homework is the SJTU Emotion EEG Dataset (SEED), which is a three-class classification problem. 37367 samples are included in the training data, and 13588 samples in the test data. The input vector of each sample consists of 62 channels and 5 extracted frequency features for each channel.

## 2 Multi-class SVM

### 2.1 Problem Definition

Solving the three-class classification problem in the given dataset using SVM classifiers and the one-vs-rest strategy. SVM classifiers are provided in LibSVM package and other machine learning libraries (sklearn). You can use these libraries to solve this problem. Notice: the SVM provided in these third-party modules can handle multi-class classification. However, you are required to write the one-vs-rest strategy by yourself in this assignment.

### 2.2 Problem Solution

To solve the problem, we implemented a three-class SVM model with python using one-vs-rest strategy. The commonly used third party module sklearn is adopted as the basic binary classification model. To boost the performance, we use the popular radial basis function (RBF) as kernel function to transform the input features to a new high-dimensional space.

**Model Selection.** Taking the outliers into account, the SVM model with RBF kernel could be defined as an optimization problem

$$\min_{w,b} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t. } y_i(\sum_{j:\alpha_j>0}\alpha_j y_j K_{ij} + b) \geq 1 - \xi_i$$

where the kernel function is defined as

$$K_{ij} = exp(-\frac{\|X_i - X_j\|_2^2}{2\gamma^2})$$

According to this definition, there are two key hyper parameters, $\gamma$ and $C$, involved in this method.

Theoretically, $\gamma$ determines how much the decision boundary is based on the support vectors. The RBF kernel could be viewed as a Gaussian distribution, the center of which is a support vector, while $\gamma$ represents the variance of this distribution. When $\gamma$ is very small, the shape of the distribution is sharp. The model is then equivalent to k nearest neighbors method among support vectors. Therefore, $\gamma$ could be

simply viewed as the invert of the influence radius of the support vectors.

$C$ is the penalty coefficient of the slack variables. When $C$ is small, the slack variables could grow large, and there would be more allowance for outliers. The decision boundary is consequently smooth. Otherwise, when $C$ is large, the model is more accurate but complex.

As a result, we can control the value of $\gamma$ and $C$ to optimize the trade-off between accuracy and generality. In practice, we adopted a 5-fold cross-validation to select these hyper parameters based on the accuracy score on both the training set and the validation set. The result is shown in Table 1, Column "3-class SVM". After extensive experiments, the hyper parameters we finally chose are

$$\begin{aligned} \gamma_0 &= 0.010, & C_0 &= 2.0 \\ \gamma_1 &= 1.000, & C_1 &= 1.0 \\ \gamma_2 &= 0.008, & C_2 &= 1.0 \end{aligned}$$

The subproblem in table 1 means the three one-vs-rest problems. In detail, model 0 solves class 0 versa class 1 and -1. Model 1 solves class 1 versa 0 and -1. Model 1 solves class -1 versa 0 and 1. The model * in the table represents the final model for the original problem.

Obviously, the accuracy on the validation set is much higher than the testing set. According to the result of cross-validation, our models do not suffer from over fitting. Actually, we can adjust hyper parameters to have these models predict totally correct answers over the validation set. However, the accuracy on the testing set could easily goes down to about 0.35. We guess this phenomenon is caused by the difference between training set and testing set. As we know, the electrical signals could vary violently among different subjects (i.e. people invited to participate the experiments).

Additionally, to speed up the training process, we adopted some improvement. According to the user guide of sklearn, the built-in predicting function of probability distribution uses Platt scaling for calibration. During this procedure, another two hyper-parameters are involved implicitly. And the module would perform a cross validation over the training set to automatically search the optimal ones, which is quite time-consuming. Instead, we use a simple sigmoid function to transform the confidence score output by decision function to fit the form of probability. Furthermore, we use a subset of the training dataset and testing dataset. We use 10k samples for training and 2k samples for testing. The subset is randomly sampled from the original dataset, thus do not affect the performance results.

There is also an interesting phenomenon that scaling operation will increase the accuracy of these models. We simply scaled the feature vector $X_i$ of each sample in both training set and testing set in the way of

$$X_{ij} \leftarrow (X_{ij} - 10)/20, \ \forall i, j$$

based on the observation of the data distribution that the original $X_{ij}$ varies from 10 to 30.

# 3 Min-Max-Modular SVM

## 3.1 Problem Definition

Solving the three-class classification problem using Min-Max-Module SVM and part-vs-part task decomposition method. You should divide the three-class problem into three two-class problems using one-vs-rest method and then decompose these imbalance two-class problems into balance two-class problems following random task decomposition and task decomposition with prior knowledge strategies.

Please compare the performance of SVMs obtained in Problem One and the Min-Max-Module SVMs here.

| Fold Number | Subproblem | 3-class SVM | | MMM-RD SVM | | MMM-PK SVM | |
|---|---|---|---|---|---|---|---|
| | | valid acc | test acc | valid acc | test acc | valid acc | test acc |
| Fold 0 | Model 0 | 0.8225 | 0.6845 | 0.9170 | 0.7220 | 0.6725 | 0.6660 |
| | Model 1 | 1.0000 | 0.6765 | 0.9950 | 0.6765 | 1.0000 | 0.6765 |
| | Model 2 | 0.6905 | 0.6575 | 0.6795 | 0.6575 | 0.6825 | 0.6575 |
| | Model * | 0.8270 | 0.5590 | **0.8905** | 0.5190 | 0.8000 | **0.5710** |
| Fold 1 | Model 0 | 0.8225 | 0.6765 | 0.9140 | 0.7165 | 0.6825 | 0.6660 |
| | Model 1 | 1.0000 | 0.6765 | 0.9980 | 0.6765 | 1.0000 | 0.6765 |
| | Model 2 | 0.6990 | 0.6575 | 0.6865 | 0.6575 | 0.6890 | 0.6575 |
| | Model * | 0.8425 | 0.5725 | **0.8770** | 0.5420 | 0.8190 | **0.5870** |
| Fold 2 | Model 0 | 0.8390 | 0.6935 | 0.9305 | 0.7110 | 0.6745 | 0.6660 |
| | Model 1 | 1.0000 | 0.6765 | 0.9960 | 0.6765 | 1.0000 | 0.6765 |
| | Model 2 | 0.7010 | 0.6575 | 0.6885 | 0.6575 | 0.6900 | 0.6575 |
| | Model * | 0.8395 | 0.5385 | **0.8895** | 0.5375 | 0.8020 | **0.5625** |
| Fold 3 | Model 0 | 0.8335 | 0.6895 | 0.9200 | 0.7125 | 0.6935 | 0.6660 |
| | Model 1 | 1.0000 | 0.6765 | 0.9935 | 0.6765 | 1.0000 | 0.6765 |
| | Model 2 | 0.6820 | 0.6575 | 0.6675 | 0.6575 | 0.6740 | 0.6575 |
| | Model * | 0.8315 | 0.5765 | **0.8825** | **0.6060** | 0.8245 | 0.5750 |
| Fold 4 | Model 0 | 0.8125 | 0.6770 | 0.9230 | 0.7110 | 0.6725 | 0.6660 |
| | Model 1 | 1.0000 | 0.6765 | 0.9965 | 0.6890 | 1.0000 | 0.6765 |
| | Model 2 | 0.6870 | 0.6575 | 0.6735 | 0.6575 | 0.6750 | 0.6575 |
| | Model * | 0.8355 | **0.5715** | **0.8770** | 0.5595 | 0.7975 | 0.5515 |

Table 1: Cross-validation Result of the 3-class SVM

### 3.2 Problem Solution

To solve this problem, we first implement a Min-Max-Module SVM using subsets which are randomly divided. We use the shuffle function provided by third-party module Numpy with fixed random seed. For each subproblem, we divide the training set into two equal parts. We select positive samples from one subset and negative samples from the other to form two new subsets. Then we train four SVM over them to predict the probability of positive class. We do a min max operation to get the final confidence score. At last, we select the positive class of the most confidential subproblem as the final prediction. The performance result is shown in Table 1, Column "MMM-RD SVM". The best hyper-parameters we find are

$$\gamma_0 = 0.10, \qquad C_0 = 2.2$$
$$\gamma_1 = 1.00, \qquad C_1 = 1.0$$
$$\gamma_2 = 0.01, \qquad C_2 = 1.2$$

Then we implement a Min-Max-Module SVM using subsets divided with prior knowledge. We investigate the collection process of the dataset and manage to find out the gender of different subjects. We put this information into the dataset and shuffle it with the same random seed as mentioned above. We divide it into two subsets based on different genders in the first stage, and repeat the same process as described in last paragraph. The performance result is shown in Table 1, Column "MMM-PK SVM". The best hyper-parameters we find are

$$\gamma_0 = 0.001, \qquad C_0 = 1.0$$
$$\gamma_1 = 1.000, \qquad C_1 = 1.0$$
$$\gamma_2 = 0.010, \qquad C_2 = 1.0$$

According to Table 1, we can see the largest accuracy over different models on the validation set and the testing set is displayed in bold. It's obvious that the Min-Max-Module SVM defeat the baseline method in most situation. The random problem decomposition method obtained the best score on validation set among all folds. And the Min-Max-Module SVM with prior knowledge is distinctly superior on the testing set among most folds.

In spite of these positive results, we can not assure the goodness of the Min-Max-Module method. The performance of these models is inevitably sensitive to the hyper parameters. And the optimal hyper parameters shifts violently when the model structure changes or the dataset is different. Due to the slow training process and huge searching space, we have little confidence in their optimality. As a result, the performance scores could be misleading. What's worse, the decomposition of binary classification problems involves considerably more hyper-parameters. This will definitely contribute to the difficulty of optimization which limits the application of the Min-Max-Module algorithms.