CSE 569

Fundamentals of Statistical Learning and Pattern Recognition

(Fall 2021)

Project Part 2: Report

Sunaada Hebbar Manoor Nagaraja

ASU ID: 1219580453

MS in Computer Science

Arizona State University

Email: sunaada.hebbar@asu.edu

1. Introduction

Support Vector Machines are statistical learning models that can be used to solve classification problems. SVMs can be used for both linear classification as well as non-linear classification problems. For non-linear classification, SVMs can efficiently map the inputs into higher dimensional feature spaces and then convert the problem into a liner classification problem in the new feature space.

The objective of this project is to study how to use a popular open-source SVM library called 'LIBSVM' through doing some experiments on SVM based classification tasks. A dataset consisting of 50 classes/categories, a training set of 4786 samples and a testing set of 1883 samples is used in the project. Each sample is described by the rows of the three different feature matrices i.e., X_1, X_2, and X_3 and all the three features are normalized histograms.

The project involves implementation of the following tasks:

- Step 0: Classification by individual features
- Step 1: Feature combination by fusion of classifiers
- Step 2: Feature combination by simple concatenation

2. Summary of Tasks

2.0 Using LIBSVM library

The Python interface for LIBSVM can be installed through pip using the following command on the terminal:

```
pip install -U libsvm-official
```

The following functions from the LIBSVM library are used for training SVM model, classifying the testing set and then calculating the accuracy of classification:

1. **svm_train**

   Function signature: svm_train(y, x [, options]) -> model | ACC | MSE

   Where 'y' is the labelled class vector corresponding to each training set sample. 'x' is the feature vector of each training instance. 'options' is string specifying additional parameters which must be considered for training the data. Ex: Use "-s 0" for SVM type C-SVC, "-t 0" for kernel type as linear, "-c 0" for cost parameter, "-q" for quiet mode with no outputs printed on the console, "-b 1" for specifying whether to train a SVC or SVR model for probability estimates, etc.

   The function returns a 'svm_model' object (if "-v" or cross validation is NOT specified in the 'options' parameter, otherwise it returns either accuracy or mean-squared error) which contains all the trained parameters and functions necessary for predicting classes of any given dataset.

2. **svm_predict**

   Function signature: svm_predict(y, x, m [, options]) -> (p_labels, p_acc, p_vals)

   Where 'y' is the labelled class vector corresponding to each testing set sample. 'x' is the feature vector of each testing instance. 'm' is the SVM model. 'options' can be used to provide additional parameters like "-b 1" in order to predict probability estimates and "-q" for quiet mode with no outputs printed on the console.

   The function returns a tuple consisting of p_labels, p_acc, and p_vals, where p_labels is a list of predicted labels, p_acc is a tuple including accuracy (for classification), mean-squared error, and squared correlation coefficient (for regression) and p_vals is a list of decision values or probability estimates (it is null if "-b" parameter is NOT specified)

2.1 Step 0: Classification by individual features

In this step, individual features are used from the training set to train separate SVM models for each of those features using 'svm_train' function from LIBSVM library. Further, we classify the testing set for each of the individual features and compute the accuracy of classification using 'svm_predict' function from LIBSVM library.

Case 1:

In this step, the options **"-c 10 -t 0 -q"** is used for svm_train function and options **"-q"** is used for svm_predict function. All other options are set to their default values.

So, for each of the features $\mathbf{X}_k$ (where $1 \leq k \leq 3$) from the training set, a multi-class linear SVM classifier, i.e., $\mathbf{h}_k(\mathbf{x})$ is trained. Using the previously trained models compute $\mathbf{h}_k(\mathbf{x})$ for corresponding features $\mathbf{X}_k$ in the testing set. Compare the classification predictions made on the testing set with **Y** and compute the classification accuracies for each of the features.

The following accuracies were obtained for each of the individual features in Case 1 of Step 0:

```
Step0, Case1 Output:
X1 Feature Test Data Accuracy: 11.36%
X2 Feature Test Data Accuracy: 17.53%
X3 Feature Test Data Accuracy: 8.60%
```

Case 2:

In this step, the options **"-c 10 -t 0 -b 1 -q"** is used for svm_train function and options **"-b 1 -q"** is used for svm_predict function. All other options are set to their default values. "-b" is the new parameter passed other than the ones passed in Case1. "-b 1" specifies that 'svm_train' function must train SVC for probability estimates and 'svm_predict' function must predict probability estimates.

So, like Case 1 the SVM models are trained for each feature with the training of probability estimates enabled. In the prediction step, probability estimates are returned along with the results, i.e., $p_k(w_i|\mathbf{x})$, the (posterior) probability of sample **x** that it belongs to the i^{th} category (w_i) according to feature $\mathbf{X}_k$ ($1 \leq k \leq 3$) is also returned by 'svm_predict' function.

The following accuracies were obtained for each of the individual features in Case 2 of Step 0:

```
Step0, Case2 Output:
X1 Feature Test Data Accuracy: 27.62%
X2 Feature Test Data Accuracy: 27.67%
X3 Feature Test Data Accuracy: 29.53%
```

From the results of Step 0, one can observe that the classification accuracies improve upon enabling training probability estimates (posterior probabilities) and predicting classification based on the probability estimates, which is an expected result.

2.2 Step 1: Feature combination by fusion of classifiers

In this step, the results for predictions obtained for individual features are combined, the classification is done for the combined results, and the classification accuracy is computed. This can be done by averaging the probability estimates for each of the classes from the 3 features that is the result of 'svm_predict' obtained in Case 2 of Step 0 and assigning the predicted class as the one with highest probability estimate.

So, the 3 SVM classifiers with probability output i.e., $p_k\,(w_i|\mathbf{x})$ ($1 \le k \le 3$), in Case 2 of Step 0 are directly combined. The 3-classifier combination by probability fusion is calculated as:

$p\,(w_i|\mathbf{x}) = \sum_k p_k\,(w_i|\mathbf{x}) / 3$

The final classification result is chosen based on:

$w_i* = \text{argmax}_i\; p\,(w_i|\mathbf{x})$

The accuracy obtained in Step 1 was:

```
Step 1 Output:
Test Data Accuracy using Feature Combination by Fusion of Classifiers: 45.03%
```

By comparison of results from Step 0 Case 2 with that of Step1, it is evident that the accuracy of classification for the testing set can be improved significantly (close to 150% improvement for the given testing set) by using fusion of classifiers as compared to using individual feature-based classifiers. These results are indeed natural because when considering only one feature for the classification we are ignoring the information about the other features and hence the accuracy can be low. Whereas by taking average of the classifiers we are taking into consideration information from all the features and if the feature spaces are highly corelated it provides particularly good accuracies.

2.2 Step 2: Feature combination by simple concatenation

In this step, all the 3 features are concatenated to get the combined feature i.e., individual features are joined together to get a single combined feature. The concatenated feature for every sample from the training set is used to train the SVM models. Similarly, the concatenated feature for every sample from the testing set is used for classification and computation of classification accuracy.

So, the 3 features $\mathbf{X}_k$, $1 \le k \le 3$ are directly concatenated to form a single feature **X** given by:

$\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_k]$

A linear SVM classifier is trained based on **X** and the classification accuracy for the testing set is calculated using **X**.

The accuracy obtained in Step 2 was:

```
Step 2 Output:
Test Data Accuracy Using Model Trained From Feature Concatenation: 39.19%
```

From the results for Step 2, the classification accuracy using concatenation of features is much better (more than 200% improvement for the given testing set) compared to that of individual classifiers in Step 0 Case 1. This result is obvious because by considering only one feature for the classification we are ignoring the information about the other features. Whereas in feature concatenation we combine all the features in the input into a single feature and the training includes information from all the features and hence the classification accuracies are better.

3. Code

Code is submitted to the GitHub repository - https://github.com/shebbar27/statistical-learning-project-part2 and all the above results can be replicated using the repository.