Discrimination of reflected sound signals based on Support Vector Machine

Master Information Technology
Machine Learning
SANG NGUYEN
1185021
Email: phuoc@stud.fra-uas.de

***Abstract*—Support Vector Machine (SVM) is a well-founded and popular technique in machine learning. It is famous for its robustness and well-established mathematical formulation, hence for its efficiency in practical applications. This report is intended to describe a very simple demonstration of this technique in machine learning with the help of Matlab. The training data sets are reflected time-domain ultrasonic signals. The data is first transformed by Principle Component Analysis to obtain a reduced representation of the signal. It will then be used to fit a simple SVM model with polynomial kernel. The trained model will be tested with testing data and also be cross validated.**

Keywords—Support Vector Machine, SVM, Principle Component Analysis, PCA, Matlab, ultrasonic, demonstration

I. Introduction

In a world, where we are flooded with information, artificial intelligence proved to be one of the most useful and powerful tools for human being. It aids human in many important tasks, e.g. finding pattern inside data distributions, recognition and classification of data, … Machine learning is a smaller area inside this broad topic. It deals with mathematics and algorithms for machines to emulate the ability of human brain. It could also be further divided, based on technical details and applications, into supervised, unsupervised and reinforcement learning. Supervised learning machines need to be carefully "trained" by human instruction before being deployed. Unsupervised learning machines need to be carefully designed and will improve itself while being deployed. Reinforcement learning machines will also improve itself during deployment, but with partial help from feedback of human defined components.

One of the most efficient and powerful supervised machine learning methods is Support Vector Machine (SVM). The mathematics behind SVM is well-founded and rigorous [1]. Therefore, it has found extremely extensive use in practical machine learning applications. SVM attempts to classify a simple data distribution with a mathematically best "border". For more sophisticated data distributions, SVM tries to look at the data distributions from a different perspective (with the help of the so-called kernel) in order to bring about that best "border". Many popular software tools for machine learning included SVM as an important part of its features. This helps the deployment of SVM become easier and more widespread.

To make the machine learning algorithm's performance more efficient, some preprocessing procedures of the data are required. For example, Principle Component Analysis (PCA) could be used to reduce the data's dimension. PCA attempts to capture the variation in the data distribution in a systematic manner so that the resulting transformed data could be adjusted to balance its verbosity and significance [2]. Less verbosity (less significance) means less input to the machine learning algorithm, thus faster training, but it also means less fidelity to the original data. More verbosity (more significance) means slower training but more fidelity to the original data.

In this project, the data to be classified is the reflected time-domain ultrasonic signals from different objects. The raw data is first compressed dimensionally into smaller data with PCA. Then it will be used to fit a SVM with polynomial kernel. The trained SVM model will then be evaluated using both test data and cross validation.

The Matlab version that was used in this project is R2019a.

II. Description

A. *Scope of the report*

This report is intended to provide information about development and validation of a simple SVM model to classify time-domain ultrasonic signal. Additionally, some information about PCA is also included. This project only serves to investigate and present findings about the feasibility of using SVM and PCA to classify ultrasonic signal.

B. *Overview*

The main goal of this project is to implement a simple SVM model to label time-domain ultrasonic signals reflected from a certain object versus the others. In this project, to classify dataset of ultrasonic signals, SVM and PCA are used. These tools, as well as developing platform for user interface, are also available in Matlab R2019a, thus this whole project will be implemented in Matlab R2019a.

The data set is first divided into training and testing set. The training dataset is then fed to the PCA component. The output of the component is a new representation of the original dataset, which is a set of smaller (less features) signals. The training set is used to fit the SVM model with polynomial kernel function. The performance of the trained SVM model is then tested with the reduced testing set. The general model is also cross validated.

C. *Dataset*

The dataset used in this project consists of labeled time-domain ultrasonic signals reflected from different objects.

TABLE I. SUMMARIZATION OF DATASET

Object	Summarization			
	Number of XLSX files	*Total data entry count*	*Length of one data entry*	*Metadata length*
1	9	315	3406	6
2	4	200		
3	8	400		
4	6	300		
5	5	250		

The dataset is saved in the form of Excel XLSX files. Each file contains a table. Each table contains a set of rows and columns. Essentially, each row is a sampled array of time signal of the reflected ultrasonic wave plus some additional information. Looking from the perspective of the SVM model, each row represents one *observations* or data point. Each column represents a feature or *predictor*. Each XLSX file is contained in folder with name "Object 1", "Object 2",… The data inside these folder is labeled accordingly and the label is also called the *response variable*.

The main task of the SVM model is to differentiate between "Object 1" labeled signals and the rest.

The dataset is first read from the XLSX files, divided to training and testing set and then fed to the PCA component.

D. *Principle component analysis (PCA)*

PCA attempts to reduce the dimension of the data distribution by searching for a coordinate that captures the most variation in the data distribution [3]. Specifically, PCA first tries to center the data distribution at its average. Then it will try to find the best fit line to data as its first principle component (PC). When data is projected onto this PC, its variation has the largest value. The second PC is constructed in the same manner with an additional constraint that it is orthogonal to the first one. The third one is constructed similarly and must be orthogonal to the first and the second one and so on.

The Principle Components (PC) are ordered decreasingly with respect to how much variation in the data distribution they captured. This provides a convenient way to somehow separate intrinsic informative variations of the data distribution from the noise-induced ones [2].

The second advantage of PCA is that it helps to reduce the dimension of the data. This will significantly improve the speed of the learning algorithm, which makes use of these reduced data.

E. *SVM model*

The training set, after being transformed, is used to fit the SVM model. This SVM model will make use of polynomial kernel and tries to construct a "border" to differentiate signal with label "Object 1" from the rest. To avoid overfitting, the model will be configured to aim for soft margin "border", which allows some misclassification in the training phase.

The testing set is used to validate the generalizability of the trained model to new data. From the predictions made by the trained SVM model on the new dataset, some statistics will be gathered and discussed.

In order to obtain a more correct evaluation of the model, 5-fold cross validation will also be carried out.

F. *User interface*

Matlab contains an application called App Designer, which integrates many functions to create a simple user interface for program written in Matlab code [10]. This application is utilized to create a simple user interface for the SVM program.

III. IMPLEMENTATION

Following is the description of solution to each project requirement. In this project, all codes are written in Matlab.

A. *Creating graphical user interface*

The user interface is created based on the App Designer tool from Matlab [10].

B. *Reading and formatting data for training phase*

Dataset is organized into different folders with its label as the corresponding containing folder's name. The data is first read into memory using Matlab function *datastore()* and *readall()* [4][5]. The metadata will then be truncated from the obtained data. During deployment, the starting and ending position for data reading can be specified by user.

After the data has been read, it will be labeled according to the corresponding containing folder.

The whole dataset will be divided into training and testing dataset with ratio 8:2 respectively using function *cvpartition()* [7].

The training dataset will then be transformed using PCA. A transformation matrix is constructed from this dataset using function *pca()* [6]. The columns in the transformation matrix is ordered from high to low according to level of original data's variation captured. The dimension of the new transformed data can be adjusted (by choosing the number of columns in the transformation matrix). This matrix will then be used to transform the training dataset and later on to preprocess data during deployment.

C. *Training the SVM model*

An SVM model could be created using the Matlab function *fitcsvm()* [8]. This model will be trained with the transformed training dataset. Soft margin will be used to prevent overfitting. After some testing, polynomial is the chosen kernel for the model.

Following is the parameters for the utilized Matlab function.

TABLE II. COMMON SETTING FOR MATLAB FUNCTIONS

Function	**Parameter**	**Value**
pca()	—	—
cvpartition()	Holdout	0.2
fitcsvm()	Kernel function	Polynomial
	Standardize	True

D. *Statistics gathering*

After finishing training, the model will be tested with transformed testing dataset. The following statistics will be calculated from the result: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN), False Discovery Rate (FDR), Negative Predictive Value (NPV), True Positive Rate (TPR), True Negative Rate (TNR), F1-scrore, receiver operating characteristic (ROC) curve and misclassification rate.

Cross validation will also be carried out to investigate the generalizability of the model to new data.

IV. Testing Result and Discussion

A. *Evaluation with holdout (0.2) test*

Following is the test result when PCA is used in the preprocessing phase.

TABLE III. HOLDOUT (0.2) TEST RESULT WITH PCA

TP	FP	TN	FN	FDR
43	10	220	20	0.18868
NPV	**TPR**	**TNR**	**F1**	**Miss Rate**
0.91667	0.68254	0.95652	0.74138	0.11407

TABLE IV. CONFUSION MATRIX OF PCA TEST

True class	**Object 1**	43	20
	Not Object 1	10	220
		Object 1	**Not Object 1**
		Predicted class	

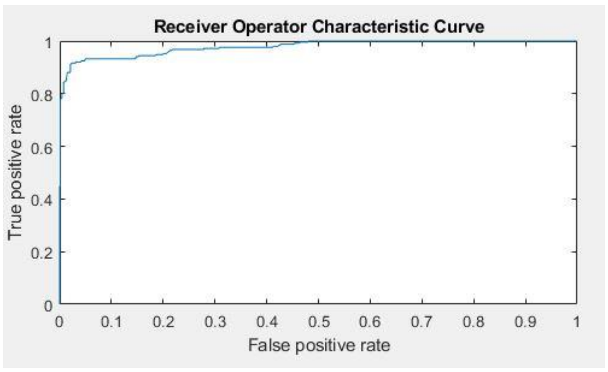

Figure 1 ROC curve for PCA test (calculated with *perfcurve()[9]*)

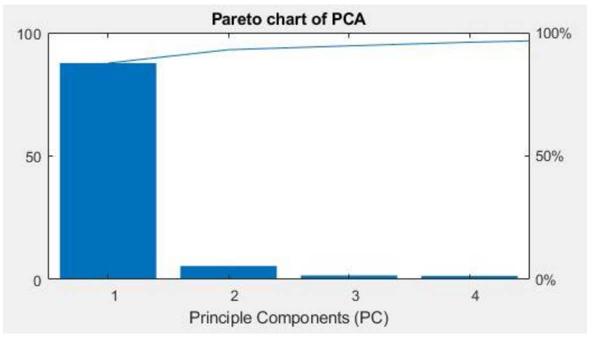

Figure 2 Variation captured by PCs (only first 4 components are shown)

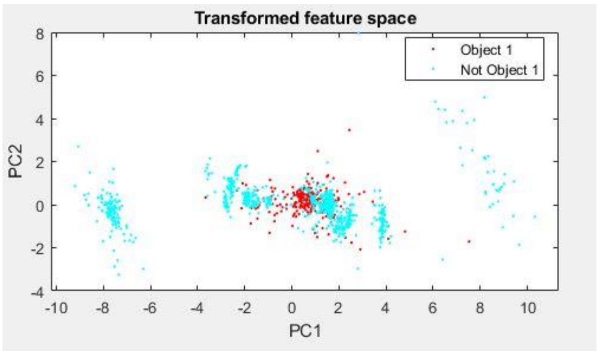

Figure 3 Transformed feature space with the first 2 PCs

From the collected data, with FDR of about 20 percent and TPR of about 70 percent, the performance of the classifier is not quite satisfactory regarding positive (“Object 1”) predicting ability. The F1-score, derived from FDR and TPR, quantifies this with a value of approximately 0.75.

However, the negative (“Not Object 1”) predicting ability is acceptable. Both NPV and TNR are higher than 90 percent.

Overall, with 10 percent of misclassification rate, the performance of the classifier is somewhat acceptable if we do not put heavy weight on positive predicting ability.

B. *Cross validation (5-fold)*

Following is the summarization of the test’s result.

TABLE V. CROSS VALIDATION RESULT

TP	FP	TN	FN	FDR
168	22	1128	147	0.11579
NPV	**TPR**	**TNR**	**F1**	**Miss Rate**
0.88471	0.53333	0.98087	0.66535	0.1304

TABLE VI. CONFUSION MATRIX FOR CROSS VALIDATION

True class	**Object 1**	168	147
	Not Object 1	22	1128
		Object 1	**Not Object 1**
		Predicted class	

As expected, for cross validation, the performance of the model regarding the positive (“Object 1) predicting ability is slightly lower.

The negative (“Not Object 1”) predicting ability related indices remain quite high and the overall misclassification rate is also relatively low. This indicates an acceptable performance regarding negative predictions.

V. CONCLUSION

SVM is a popular and robust solution for machine learning problems. SVM is included and highly optimized in many famous software tools. Therefore, it could be considered as an off-the-shelf solution to many practical applications.

PCA is a handy technique that brings about 2 main advantages: separating important variations from noise-induced ones in the dataset and improving speed of the classfifier.

In this project, SVM is used in conjunction with PCA as the data preprocessing method for ultrasonic signal. The combination between SVM and PCA gives only a moderate performance regarding positive predicting ability (accuracy and precision of the model to predict positive label data, which, in this project, is assumed to be “Object 1” data). Overall, the general misclassification rate is, however, acceptable.

Many other preprocessing methods should be tested to improve the performance of SVM-based classification of ultrasonic signals.

REFERENCES

[1] Y. S. Abu-Mostafa, “Machine Learning,” in *Learning From Data*, 29-Mar-2021.

[2] J. Shlens, “A Tutorial on Principal Component Analysis,” *pca.dvi*, 10-Dec-2005. [Online]. Available: https://www.cs.cmu.edu/~elaw/papers/pca.pdf. [Accessed: 02-Apr-2021].

[3] “StatQuest - Video Index,” *StatQuest!!!*, 30-Mar-2021. [Online]. Available: https://statquest.org/video-index/. [Accessed: 31-Mar-2021].

[4] “Datastore,” *Datastore - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/matlab/datastore.html. [Accessed: 31-Mar-2021].

[5] “readall” *Read all data in datastore - MATLAB*. [Online]. Available: https://www.mathworks.com/help/matlab/ref/matlab.io.datastore.readall.html. [Accessed: 31-Mar-2021].

[6] “pca,” *Principal component analysis of raw data - MATLAB*. [Online]. Available: https://www.mathworks.com/help/stats/pca.html. [Accessed: 02-Apr-2021].

[7] “cvpartition,” *Partition data for cross-validation - MATLAB*. [Online]. Available: https://www.mathworks.com/help/stats/cvpartition.html. [Accessed: 31-Mar-2021].

[8] “fitcsvm,” *Train support vector machine (SVM) classifier for one-class and binary classification - MATLAB*. [Online]. Available: https://www.mathworks.com/help/stats/fitcsvm.html. [Accessed: 31-Mar-2021].

[9] “perfcurve,” *Receiver operating characteristic (ROC) curve or other performance curve for classifier output - MATLAB*. [Online]. Available: https://www.mathworks.com/help/stats/perfcurve.html. [Accessed: 31-Mar-2021].

[10] “MATLAB App Designer,” *MATLAB*. [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html. [Accessed: 02-Apr-2021].