CmpE 462 Spring - 2020 Project2 Report

Alperen Bağ - 2018400279
Cihat Kapusuz - 2016400126
İbrahim Özgürcan Öztaş - 2016400198

May 17, 2020

Contents

Chapter 1

Introduction

This report is composed to explain the process that is how we understand, analyze and apply support vector machine approach to a real life problem, given as Project 2 in CmpE 462, Machine Learning.

By understanding the main idea behind support vector machines allows us to cope with high dimensional data with simple yet concise methods. It is known that support vector machines are powerful machines that calculates huge amount of data with less than expected, since it is applicable to use duality feature, which transforms any primal solution to dual solution that includes transformations between different spaces. That creates a huge amount of efficiency in bigger sized data sets.

By explaining little information about support vector machines, let's start our project. There may be some referral to the theoretical part of the topic, yet it is planned as brief and compact as possible.

Have a nice reading!

Chapter 2

Hard Margin Support Vector Machine[1][2][3]

Support vector machines are constructed on several vectors given in the training data set, which are selected with respect to maximum margin possible between two or more classes to be separated. That aims to have the maximum tolerance between the vectors and the separation line between classes.

With previous explanation, let's proceed further. In our project, we're requested to train a hard margin support vector machine according to the given data set, **'data.mat'**.

In **cell** [**1**], we've initialized necessary libraries and modules to be used in further cells. We used **numpy** to play the features of data, briefly data manipulation. In addition to that, we've used **scipy.io** to load out data with extension **'.mat'**, and we've used **matplotlib** to plot our data to illustrate the outcomes if necessary. For the last, we've used svm to train our support vector machine.

Afterwards, in **cell** [**2**], we've loaded our data set as **'mat'** and split our data into two parts, one as training and the other one is test. For **training**, we've used **150 out of 270**, and for **testing**, we've used the remaining **120 out of 270**.

Furthermore, we've set our options of the planned support vector machine. To create a hard margin, we've set the **'-c'** option, which is the constant of penalty, as 10^{10} that leads to a strict margins between distinct classes in data set. With that much penalty value, it usually leads to a over fitting in data that results in decrease in testing accuracy. Hence, we're ready to train our model and the last line of **cell 2** shows the training process.

Finally, with every necessary action is taken and handled successfully, it is time to see the results of training. In the first line of **cell 3**, we try to predict the remaining 120 data vector, and we've got an accuracy as %76.6, which seems to have a little bit discouraging, due to the fact that we've been expecting at least 80% accuracy to claim the model successful. After that, we've executed one more prediction. This time, it predicts the training data itself that results in 100% success rate, that is totally expected. But, we've not thrown the towel about testing accuracy, and we've got several ideas to increase the accuracy to make it at least 80%.

Chapter 3

Soft Margin Support Vector Machine[4]

This time, we've requested to set several options to see the difference at each parameter change. For this to happen, we've set two lists, one is for kernel type, and the other one is for penalty constants.

For kernel type, we've initialized 4 types of kernel, namely **Linear**, **Polynomial**, **RBS(Radial Basis Function Kernel)**, and **Sigmoid.** Hence, we've set five constants for penalty, **0.1**, **1**, **2**, **5**, and **10**.

For each different kernel type - penalty constant pair, we've trained our model and output the resulting accuracy. It is easy to see that for this data set, both **Linear kernel and c = 1** and **Sigmoid kernel and c = 2** pairs have given the best accuracy which is **%85**, that is sufficient to satisfy our expectations.

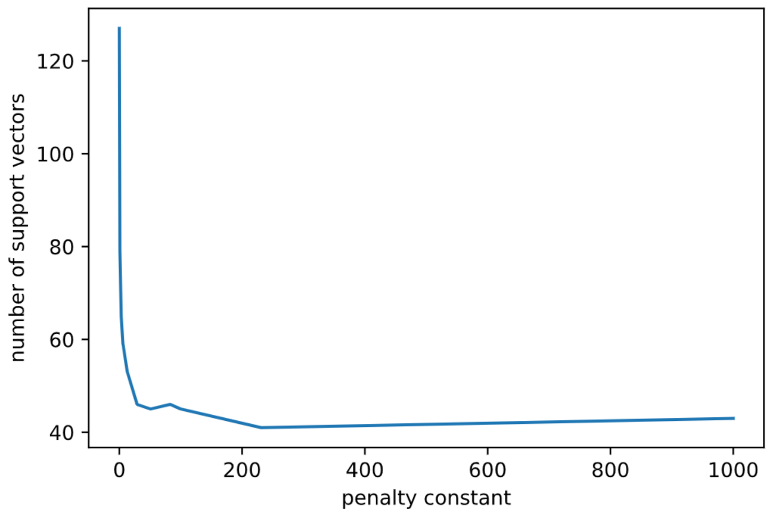
Chapter 4

Different Kernels in SVM[5]

Now, we're determined to analyze the model features, such as number of support vectors in a trained model and its relation with penalty constant **c**, if exists.

To achieve such task, we've created a list of penalty constants that are selected by our will. Furthermore, for each constant, we've trained our model and kept the number of support vectors per trained model.

Finally, to see the correlation between the penalty constants and number of support vectors, we've plotted our data via **matplotlib**.



Chapter 5

Effects of Support and Non-Support Vectors[6]-[16]

In this part, we've dug deep into model features and realized the question **'What happens if we remove one of the support vectors inside model?'** To simulate such state, we've trained a dummy model and extracted the indices of the support vectors of the model.

Then, we've removed the first support vector from the training data set, and trained a new model to see the changes if exists.

Furthermore, we've removed a non support vector from the training data set, and trained a new model to see the changes if exists.

It is reasonable to say the fact that by removing a support vector, we've increased the accuracy by %1.7 percent, which is an improvement not so tiny to ignore. Yet, removing a non support vector resulted in no change, as we see.

	Main Model	SV Removed Model	Non-SV Removed Model
Weight 1	-0.06778313	-0.05855969	-0.06778313
Weight 2	0.29866279	0.28665850	0.29866279
Weight 3	0.58209869	0.63597603	0.58209869
Weight 4	0.10258802	0.13676829	0.10258802
Weight 5	0.08010807	0.05694247	0.08010807
Weight 6	0.00090010	-0.00680732	0.00090010
Weight 7	0.05038589	0.08009280	0.05038589
Weight 8	0.51048120	-0.50702316	0.51048120
Weight 9	0.17760370	0.20768610	0.17760370
Weight 10	0.13125647	0.15059528	0.13125647
Weight 11	0.15436003	0.16007441	0.15436003
Weight 12	0.88238350	0.94541053	0.88238350
Weight 13	0.47563777	0.42195553	0.47563777

Model weights are listed above.

As we can see, when we removed a non-support vector data point, hyperplane did not change. On the contrary, removing support vector data point altered the hyperplane, since one pivot point of previous hyperplane has been removed from the data set, thus a new configuration for each feature has been appeared as expected.

The values of training models:
Main model accuracy = %83.3
Support vector **removed** model accuracy = %85
Non support vector **removed** model accuracy = %83.3

Number of support vectors inside models:
Main model has **74** support vectors.
Support vector removed model has **73** support vectors.
Non support vector removed model has **74** support vectors.