A Mathematical Essay on Support Vector Machines

Aryan Pandey
Department of Ocean Engineering
Indian Institute of Technology, Madras
Chennai, India
na19b030@smail.iitm.ac.in

***Abstract*—This document is an overview of the mathematical aspects of Support Vector Machine as well as its application on a sample data set. The algorithm has been applied on a data set of stars and the prediction task is to predict if a star is a pulsar start or not - which is a rare type of Neutron star that produces radio emissions detectable here on Earth**

***Index Terms*—Support Vector Machine, Pulsar**

I. Introduction

SVMs (short for Support Vector Machines) are machine learning algorithms that are used for classification and regression. SVMs are a type of machine learning method that can be used for classification, regression, and outlier detection. SVM classifiers create a model that allocates new data points to one of the predetermined categories. As a result, it can be considered a binary linear non-probabilistic classifier. In 1963, Vladimir N Vapnik and Alexey Ya. Chervonenkis created the first SVM algorithm. The algorithm was still in its early stages at the time. Drawing hyperplanes for linear classifiers is the sole option. Bernhard E. Boser, Isabelle M Guyon, and Vladimir N Vapnik proposed using the kernel method on maximum-margin hyperplanes to generate non-linear classifiers in 1992. Corinna Cortes and Vapnik proposed the current standard in 1993, and it was published in 1995. In this paper, the aim is to apply Support Vector Machines algorithm to predict if a star is a pulsar start or not. The paper systematically goes through first the mathematical details of SVM, the nature of the data set which has been given to us, then the problem we have in hand and how it has been solved, and finally the conclusions which were drawn. Useful insights and figures have been presented whenever necessary.

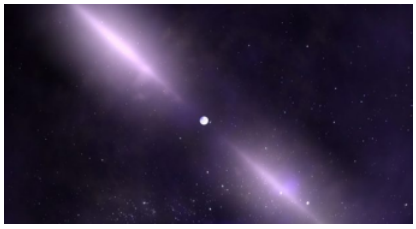

Fig. 1. Pulse Star

II. Datasets

The problem asks us to predict if a star is a pulsar start or not. This section walks you through the entire process followed to make the predictions

A. Outline

We will follow the standard procedure followed to build any Machine Learning model. We will first analyze and visualize the data. We will then try to figure out which features are to be kept intact and which features are to be dropped. Then we will convert the features into the form which the algorithm understands. Then we will split the train data into train/test sub-parts in order to figure out the most optimum parameters. And then finally we will fit the original train and test data to produce the predictions.

B. Data Visualization

All of the features appear to be continuous, which is to be expected.

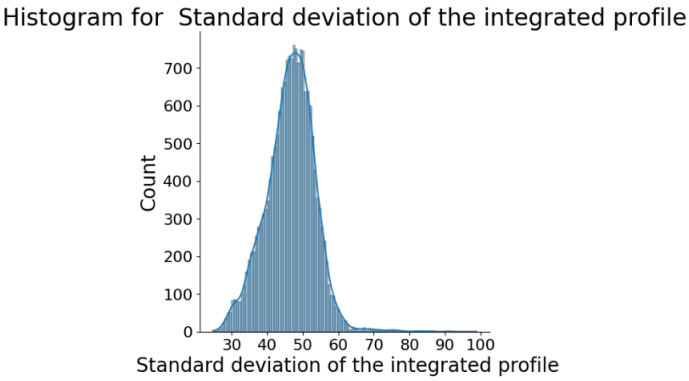

Fig. 2. Histogram for the Standard Deviation of the Integrated Profile

Most features appear to be skewed to some degree, with some containing a large number of outliers. It's also worth noting that the variables are on various scales, which should be taken into account if we're going to employ regularisation techniques (hint: we won't). It's also worth noting that some characteristics are related to one another; for example, both kurtosis and skewness are functions of a distribution's mean and variance, and their functional definitions are extremely similar.

A pair plot of the combined profile features is shown in Fig.3. The mean and kurtosis, as well as the mean and skewness, have decreasing non-linear connections. The link between standard deviation and kurtosis, as well as standard deviation and skewness, is similar, however the patterns are slightly noisier.

In addition, skewnees and kurtosis have a definite link. Let us summarize the dataset all at once:

- There are 9 numerical variables in the dataset
- 8 are continuous variables and 1 is discrete variable
- The discrete variable is targetclass variable. It is also the target variable

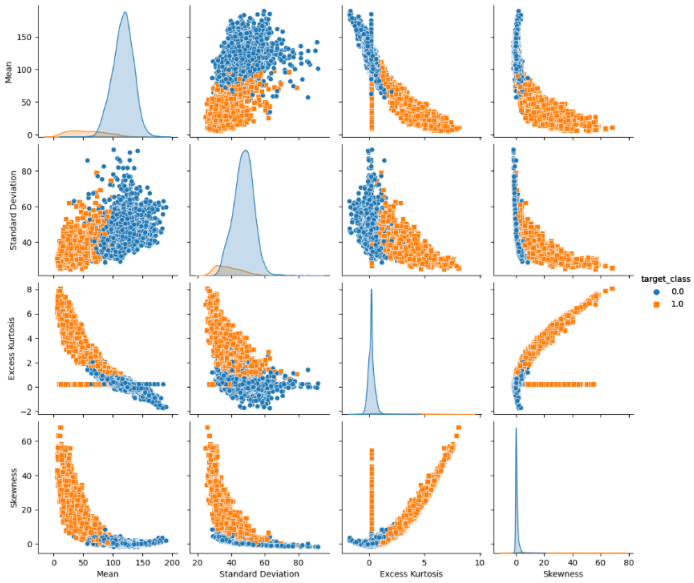

Fig. 3. Pair Plot for features of the integrated Profile separated by target class

On closer inspection, we can suspect that all the continuous variables may contain outliers.

III. Models

This section discusses the mathematical and conceptual aspects of the SVM algorithm.

A. Intuition

Now, we should be familiar with some SVM terminology

- Hyperplane - A hyperplane is a decision boundary that divides a set of data points with differing class labels into two groups. The SVM classifier uses a hyperplane with the most margin to separate data points. The maximum margin hyperplane and the linear classifier it specifies are known as the maximum margin hyperplane and maximum margin classifier, respectively.
- Support Vectors - The sample data points nearest to the hyperplane are called support vectors. By calculating margins, these data points will better define the separation line or hyperplane.
- Margin - A margin is the distance between the two lines on the data points that are closest to each other. It is determined as the perpendicular distance between the line and the nearest data points or support vectors. We strive to maximise this separation distance in SVMs to get the most margin.

The graphic below visually depicts these notions

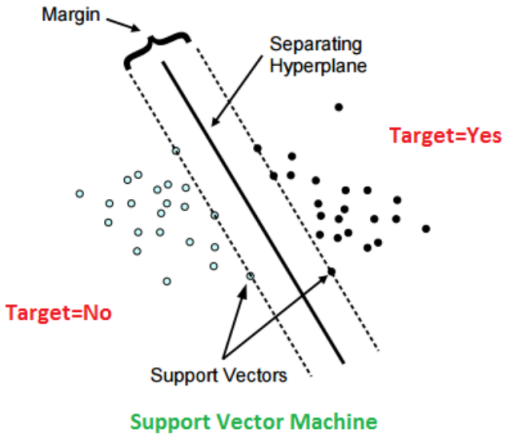

Fig. 4. Margins in a Support Vector Machine

The primary goal of SVMs is to find a hyperplane with the greatest feasible margin between support vectors in a given dataset. In the following two-step method, SVM looks for the largest margin hyperplane

- Create hyperplanes that separate the classes as much as possible. There are numerous hyperplanes that might be used to categorise the data. The optimal hyperplane that reflects the widest separation, or margin, between the two classes should be chosen.
- As a result, we choose the hyperplane with the greatest distance between it and the support vectors on each side. If such a hyperplane exists, it is referred to as a maximum margin hyperplane, and the linear classifier it defines is also referred to as a maximum margin classifier.

The diagram below clearly explains the concepts of maximum margin and maximum margin hyperplane

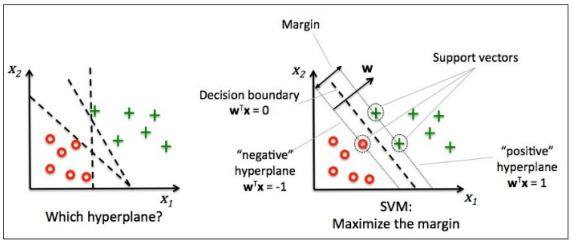

Fig. 5. Maximum Margin Hyperplane

B. The Algorithm

Let's say we have some data and we (the SVM algorithm) are asked to distinguish between males and females by first analysing the features of both genders and then appropriately labelling the unseen data as male or female. In this case, the qualities that would aid in gender differentiation are referred to as features in machine learning. We comprehend x's domain

when we define it in real space, and we receive range and co-domain when we map a function for y = f(x). As a result, we are provided the data to be split by the algorithm at the start. The data to be separated/classified is represented as a single point in space, with each point being represented by a feature vector x.

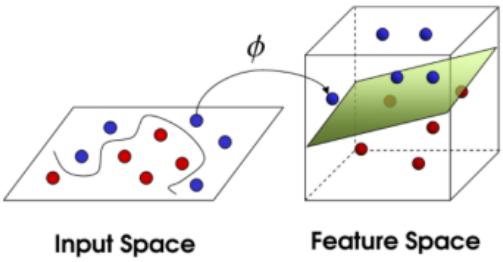

Fig. 6. Input Space gets mapped to the Feature Space

Further, mapping the point on a complex feature space x. The transformed feature space for each input feature mapped to a transformed basis vector (x) can be defined as $\phi(x) = R^D \rightarrow R^M$ Now that we've physically depicted our points, the next step is to divide them with a line, which is where the phrase "decision boundary" comes into play. Decision The key separator for splitting the points into their different classes is the boundary. The equation of the main separator line is called a hyperplane equation.

The hyperplane equation dividing the points (for classifying) can now easily be written as $H : w^T(x) + b = 0$

Here: b = Intercept and bias term of the hyperplane equation In D dimensional space, the hyperplane would always be D -1 operator. Now that we've seen how to represent data points and how to fit a separating line between them, let's look at how to fit a separating line between them. However, while fitting the dividing line, we obviously want one that can segregate the data points in the best feasible way with the fewest mistakes/misclassification errors. So, in order to have the fewest errors in data point categorization, we must first determine the distance between a data point and the separating line.

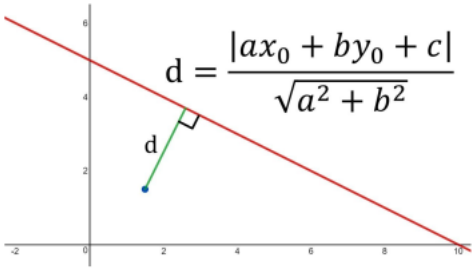

Fig. 7. Distance Measure from the Hyperplane

The distance of any line, ax + by + c = 0 from a given point say, (x0 , y0) is given by d. Similarly, the distance of a hyperplane equation: $w^T(x)+b = 0$ from a given point vector (x0) can be easily written as:

$$d_H(\phi(x_0)) = \frac{|w^T(\phi(x_0)) + b|}{||w||_2}$$

Fig. 8. Distance of a Hyperplane from a point

Here $||w||_2$ is the Euclidean norm for the length of w Finding a hyperplane with the greatest margin (margin is a protected space around the hyperplane equation) and attempting to have the greatest margin with the fewest points (known as support vectors). "The goal is to maximise the minimum distance," to put it another way. If the point from the positive group is substituted in the hyperplane equation while generating predictions on the training data that was binary classified as positive and negative groups, we will get a value larger than 0. (zero) The product of a predicted and actual label would be greater than 0 (zero) on correct prediction, otherwise less than zero. For perfectly separable datasets, the optimal hyperplane classifies all the points correctly, further substituting the optimal values in the weight equation.

C. Types of Kernels

Linear Kernel - When the data is linearly separable, a linear kernel is utilised. It indicates that data can be split with only one line of code. It is one of the most often utilised kernels. It is most commonly utilised when a dataset contains a significant number of features. The linear kernel is frequently used for text classification.

Polynomial Kernel - The similarity of vectors (training samples) in a feature space over polynomials of the original variables is represented by a polynomial kernel. To estimate their similarity, the polynomial kernel examines not only the supplied attributes of input samples, but also combinations of input samples.

RBF Kernel - Radial basis function kernel is a general purpose kernel. It is used when we have no prior knowledge about the data. Fig 19 visualizes the RBF Kernel.

Sigmoid Kernel - Sigmoid kernel has its origin in neural networks. We can use it as the proxy for neural networks. Fig 20 visualized the Sigmoid Kernel

IV. Modelling

After running the SVM model with all the 4 kernels mentioned before, we find that at C=100.0, we get the best accuracy with rbf and linear kernel, and the accuracy is 0.9832. Based on the aforementioned study, we can infer that the accuracy of our classification model is excellent. In terms of predicting class labels, our model performs admirably. However, this is not the case. We have an unbalanced dataset here. The issue is that in the imbalanced dataset scenario,

accuracy is an insufficient metric for assessing predictive performance. As a result, we must look into alternative metrics that can help us choose better models. We'd like to know the underlying distribution of values as well as the kind of errors our classifier makes. Confusion matrix is one such statistic for analysing model performance in imbalanced classes problems. Here is the confusion matrix for this case

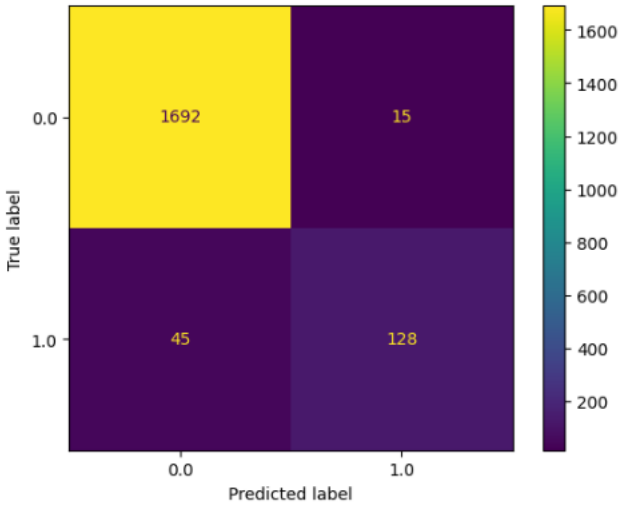

Fig. 9. Distance of a Hyperplane from a point

We can also perform hyperparameter tuning using grid-searchCV. Our original model test accuracy is 0.9832 while GridSearch CV score on test-set is 0.9835. So, GridSearch CV helps to identify the parameters that will improve the performance for this particular model.

V. Conclusion

- Our dataset contains outliers. As I increased the value of C to reduce the number of outliers, the accuracy improved. This is true for several types of kernels
- With C=100.0 and rbf and linear kernel, we get the highest accuracy of 0.9832. As a result, we may conclude that our model does an excellent job at predicting class labels. However, this is not the case. We have an unbalanced dataset here. In the imbalanced dataset problem, accuracy is an insufficient metric for assessing predictive performance. As a result, we must investigate confusion matrices that can help us choose better models
- Our model's ROC AUC is extremely close to 1. As a result, we can conclude that our classifier accurately classifies the pulsar star
- With the linear kernel, I get a better average stratified k-fold cross-validation score of 0.9789, but the model accuracy is 0.9832. As a result, the stratified cross-validation strategy is ineffective in improving model performance.
- The accuracy of our original model test is 0.9832, whereas the GridSearch CV score on the test-set is 0.9835. As a result, GridSearch CV assists in identifying the parameters that will increase the model's performance.