Review of Support Vector Machines

Reid E. Markland,[1] Logan M. Heydon,[2] and Kyle J. Camlic[2]
[1)] *Department of Physics, Auburn University*
[2)] *Department of Mathematics, Auburn University*

(Dated: December 2022)

I. INTRODUCTION

A Support Vector Machine (SVM) is a supervised machine learning algorithm used in solving classification or regression problems. They are constructed from the most simple methods in classification—drawing a line between two sets— and through abstraction develop a wide variety of applications in the the field of classification. They are used anywhere from facial recognition and image classification to bioinformatics or linear regression.

In this paper a general overview of SVMs will be given. Staring with a theoretical treatment motivated by simple ideas in classification, it will be shown that the core problem lies in the world of constrained optimization. Through abstraction the simple SVM will be developed into a general model which can be applied to most classification applications. The theory will be followed by two examples of real world problems which can be solved through implementation of SVMs.

II. THEORETICAL OVERVIEW

Let $\{\boldsymbol{x}_i\}_{i=1}^m \subset \mathbb{R}^n$ be our dataset with characterizing values $\boldsymbol{y} = \{y_i\}_{i=1}^m$, where y_i is either -1 or 1, from which we want to build a prediction model. A simple way of doing this is to have a $n-1$ dimensional curve (or **hypersurface**) dividing the two subsets based on the distinction given by $\boldsymbol{y}$. The simplest such hypersurface accomplishing this is the **hyperplane** $\boldsymbol{w}^T\boldsymbol{x} = b$ where $\boldsymbol{w}$ is the normal vector to our hyperplane and $\boldsymbol{x}$ characterizes our vector space.

A. Simple Classification

For now, let's suppose this $n-1$ dimensional hyperplane can accomplish this (the data is **linearly separable**). Then a very natural method of making predictions would be using our hyperplane

$$g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} - b = 0 \tag{1}$$

and having

$$\hat{y}(\boldsymbol{x})_i = \begin{cases} 1, & g(\boldsymbol{x}_i) > 0 \\ -1, & g(\boldsymbol{x}_i) \leq 0 \end{cases} \tag{2}$$

for any new data we want to predict the characterization of.

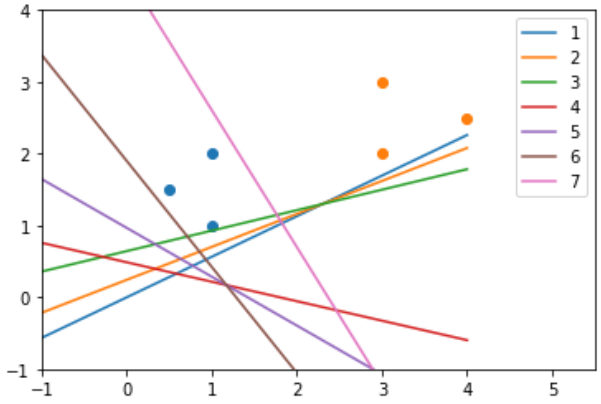


FIG. 1. A visualization of the simple classifier finding an appropriate hyperplane. As any hyperplane separating the datasets would be valid, this is not a unique solution.

Now, to have a prediction model, we need only to find some such $g(\boldsymbol{x})$ with these properties. We can develop an algorithm for obtaining a valid $g(\boldsymbol{x})$ given any initial hyperplane by defining a minimization problem. As our model will be informed by our dataset, we want it to describe our dataset as well as possible. This means minimizing $\sum_{i=1}^m |y_i - \hat{y}_i|$ where $\hat{\boldsymbol{y}} = \{\hat{y}(\boldsymbol{x_i})\}_{i=1}^m$. Such an algorithm has a fairly simple solution for a linearly separable dataset. Given a step size $\eta \in (0,1)$ and using the sign of $y_i - \hat{y}_i$ to guide the adjustment direction, we can inch towards a solution by adjusting $\boldsymbol{w}$, b with

$$\begin{aligned} \boldsymbol{w} &\leftarrow \boldsymbol{w} + \eta(y_i - \hat{y}_i)\boldsymbol{x_i} \\ b &\leftarrow b - \eta(y_i - \hat{y}_i). \end{aligned} \tag{3}$$

As can be seen in Fig 1, this has the effect of rotating the hyperplane toward some $\boldsymbol{x_i}$ by adjusting $\boldsymbol{w}$ and then sliding towards it by adjusting b. With a linearly separable dataset, Eq (3) can be iterated over the dataset until $\hat{\boldsymbol{y}}^T\boldsymbol{y} = 0$. However, as this algorithm requires only that the data is separated by the hyperplane, this is not a unique solution to the problem.

B. Unique Classification

In a true SVM, a unique hyperplane is chosen by maximising the distance of the hyperplane from both datasets. The distance from our plane to a point $\boldsymbol{x}$ is given by $d = (\boldsymbol{w}^T\boldsymbol{x} - b)/\|\boldsymbol{w}\| = g(\boldsymbol{x})/\|\boldsymbol{w}\|$. We can

rescale our $\boldsymbol{w}$, b such that $|g(\boldsymbol{x})| = 1$ for the nearest point in each dataset, so that the distance between them, or **margin**, is

$$D = \frac{2}{\|\boldsymbol{w}\|}. \tag{4}$$

Further, as we rescaled our plane, we have that

$$\begin{aligned} g(\boldsymbol{x}_i) \geq 1, \ \text{ if } y_i = 1 \\ g(\boldsymbol{x}_i) \leq -1, \ \text{ if } y_i = -1 \end{aligned} \tag{5}$$

or

$$y_i g(\boldsymbol{x}_i) \geq 1. \tag{6}$$

Then our problem becomes to

$$\begin{aligned} \text{minimize } \ f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{subject to } \ y_i g(\boldsymbol{x}_i) \geq 1 \end{aligned} \tag{7}$$

in order to maximize the margin (Eq 4).

Now that our problem is framed as a constrained optimization problem, we can write the Lagrangian,

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\lambda}) = f(\boldsymbol{w}) - \sum_{i=1}^{m} \lambda_i (y_i g(\boldsymbol{x}_i) - 1). \tag{8}$$

The Karush-Kuhn-Tucker (KKT) first order necessary conditions are then:

$$\begin{aligned} &(i) \ \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m} \lambda_i y_i \boldsymbol{x}_i = 0 \\ &(ii) \ \frac{\partial \mathcal{L}}{\partial b} = \sum_{i=1}^{m} \lambda_i y_i = 0 \\ &(iii) \ c_i(\boldsymbol{w}) = y_i g(\boldsymbol{x}_i) - 1 \geq 0 \\ &(iv) \ \lambda_i \geq 0 \\ &(v) \ \lambda_i c_i(\boldsymbol{w}) = \lambda_i (y_i g(\boldsymbol{x}_i) - 1) = 0. \end{aligned} \tag{9}$$

Expanding $\mathcal{L}$ and noticing that by KKT condition (i) $\boldsymbol{w} = \sum_{i=1}^{m} \lambda_i y_i \boldsymbol{x}_i$, we can obtain the dual Lagrangian

$$\mathcal{L}^*(\boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j. \tag{10}$$

We now have the simplified problem of finding the $\{\lambda_i\}_{i=1}^m$ which maximize $\mathcal{L}^*$ constrained to

$$\begin{aligned} &(i) \ \sum_{i=1}^{m} \lambda_i y_i = 0 \\ &(ii) \ \lambda_i \geq 0. \end{aligned} \tag{11}$$

After solving this simplified problem numerically the nonzero optimum $\{\lambda_i\}_{i \in I}$ values are known as the **support vectors** and can be used to solve for $\boldsymbol{w}$ using KKT condition (i). A solution for b is obtained using condition (iv) to derive $b = \boldsymbol{w}^T \boldsymbol{x}_i - y_i$ for $i \in I$.

C. Soft-Margin SVM

Now that we have a foundation for a useful SVM, lets see if we can modify our definition to suit more datasets. One of the limiting factors in the generalization of our SVM beyond the training data is the strict requirement that the model hyperplane perfectly separates our subsets. This is known as a **hard-margin** SVM. Certainly, though, there are datasets which dont have such hard deliniation between subsets. By modifying our constraints from Eq 7 to

$$y_i g(\boldsymbol{x}_i) \geq 1 - \zeta_i, \ \zeta_i \geq 0 \tag{12}$$

and adding some extra cost based on these ζ_i, we can define a **soft-margin** SVM:

$$\begin{aligned} \text{minimize } \ f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\left(\sum_{i=1}^{m} \zeta_i\right) \\ \text{subject to } \ y_i g(\boldsymbol{x}_i) \geq 1 - \zeta_i, \ \zeta_i \geq 0. \end{aligned} \tag{13}$$

Where C is related to the strictness of the model. This soft-margin SVM allows for some overlap in the data classification, helpful if the subsets aren't so clearly separated. By assigning a ζ_i to each data point which depends on position in the dataset, we can characterize whether a point is on the right side of the hyperplane or very close to the model hyperplane. Factoring these ζ_i into the cost provides a means of tracking how many data points spill into the wrong region. A larger C will result in a SVM similar to the hard-margin model by amplifying the cost of the ζ_i.

Again constructing the Lagrangian,

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, b, , \boldsymbol{\zeta}, \boldsymbol{\lambda}, \boldsymbol{\mu}) =& \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\left(\sum_{i=1}^{m} \zeta_i\right) \\ &- \sum_{i=1}^{m} \lambda_i (y_i g(\boldsymbol{x}_i) - 1 + \zeta_i) - \sum_{i=1}^{m} \mu_i \zeta_i. \end{aligned} \tag{14}$$

We can find the dual Lagrangian through similar means, giving

$$\mathcal{L}^*(\boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j. \tag{15}$$

This, like the hard-margin SVM, simplifies the problem to maximizing $\mathcal{L}$ constrained to

$$\begin{aligned} &(i) \ \sum_{i=1}^{m} \lambda_i y_i = 0 \\ &(ii) \ 0 \leq \lambda_i \leq C. \end{aligned} \tag{16}$$

This problem is remarkably similar to the dual hard-margin problem. It can also be solved numerically to find the support vectors $\{\lambda_i\}_{i \in I}$. As before,

$$\boldsymbol{w} = \sum_{i \in I} \lambda_i y_i \boldsymbol{x}_i. \tag{17}$$

Of the support vectors, the $\lambda_i < C$ lie on the margin and are used to calculate the value b. Unlike the hard-margin SVM, we'll take the average distance of the data lying on the accepted margin to calculate

$$b = \frac{1}{M} \sum_{i \in I | \lambda_i < C} \boldsymbol{w}^T \boldsymbol{x} - y_i. \tag{18}$$

Where M is the number of $i \in I$ so that $\lambda_i < C$.

D. Kernels and Higher Dimensions

The next step in generalizing our SVM to the widest variety of possible classification problems is doing away with the assumption that the data is linearly separable at all. Then in order to distinguish the data with our hyperplane—an inherently linear problem—we must change the representation of the data:

$$\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^m \Rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = \{\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), ..., \phi_N(\boldsymbol{x})\}. \tag{19}$$

This transformation has the effect of increasing the dimension of the space from $\mathbb{R}^n$ to $\mathbb{R}^N$ such that the representation of the data $\boldsymbol{\phi}(\boldsymbol{x})$ is solvable using a hyperplane. This N-dimensional space is known as a feature space. The problem of finding $\phi_k(\boldsymbol{x})$ so that this method is useful lies in the complexity of the inner product on our feature space. To simplify this, a matrix of kernel functions is calculated. For a symmetric function $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x_i})^T \phi(\boldsymbol{x_j})$, it is a kernel function iff the matrix

$$\boldsymbol{K}_{ij} = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) \tag{20}$$

is positive semidefinite. Each of these functions carries information about the inner product in its respective space, though finding $\phi_k(\boldsymbol{x})$ which satisfies the kernel is exceedingly difficult.

The most popular kernel functions used today are:
Degree d polynomial

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + c)^d; \ c > 0 \tag{21}$$

Gaussian radial basis (RBF)

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{2\sigma^2}\right)^2; \ \sigma > 0 \tag{22}$$

Sigmoid

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = tanh(\beta \boldsymbol{x}_i^T \boldsymbol{x}_j + \gamma). \tag{23}$$

The implementation of these kernel functions is just a reframing of the dataset via the $\phi_k(\boldsymbol{x})$. From there the determination of a classifying hyperplane look similar to the treatment of soft-margin SVMs.

III. NUMERICAL EXAMPLE I

The data in this example comes from the Iris flower data set. This data was made famous by work done by Ronald Fisher, a British statistician and biologist. Fisher used four different measurements obtained from different species of Iris's. These four measurements include the length and width of the sepal and the length and width of the petal. Using these measurements, SVM can be used to classify the species of Iris and determine which Iris attribute has the most effect on the classification.

For this example, the quadprog function in MATLAB was used to solve the optimization problem. The quadprog function solves quadratic programming optimization problems with linear constraints. For this example, the quadprog function produces the optimal weight values applied to each attribute by solving either the primal or dual constrained problem (Eq 13 or Eq 16). This results in the appropriate classifier line or plane that separates the data points.

Fig 2 shows the relationships between sepal length and petal length among two different species of Iris. The data from 50 Iris's from two species has been plotted. From Fig 2, it can be seen that the petal length of the Iris-Virginica is often times greater than that of an Iris-Versicolor of the same sepal length. This creates a relatively clear divide in the data points with most of the Iris-Virginica being above the Iris-Versicolor. The classifier line acts as this divide. The solution for the classifier line is such that the fewest number of misclassifications of a species possible are allowed. For example, in Fig 2, there are two Iris-Versicolor points that are above the classifier line, indicating two misclassifications. This means that an Iris-Versicolor would have been identified as an Iris-Virginica. The weights obtained from the solved optimization problem determine the placement of the classifier line. That is how the number of misclassifications is minimized. Figs 2-4 are the three graphs that show the relationships between sepal length and the three other attributes.

As is seen in Fig 4, the relationship between sepal length and sepal width does not have a clear divide amongst the two different species. This results in many more misclassifications when compared to other combinations of attributes. This result indicates that comparing sepal length and sepal width is not the ideal attributes to use together to classify the Iris species. To decrease the number of misclassifications, another attribute can be used. Using three attributes results in a three-dimensional graph and a classifier plane. Figs 5-8 are the four different combinations of three attributes.

It is evident from Figs 5-8 that, in general, using more attributes results in fewer misclassifications. The combination of sepal width, petal length, and petal width resulted in the fewest misclassifications. The two combinations that include sepal length and sepal width perform worse than the other two. Intuitively, that makes sense because sepal length and sepal width performed

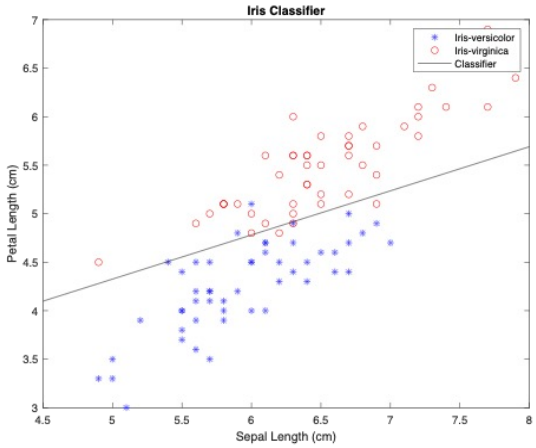


FIG. 2. Relationship between Petal Length and Sepal Length for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

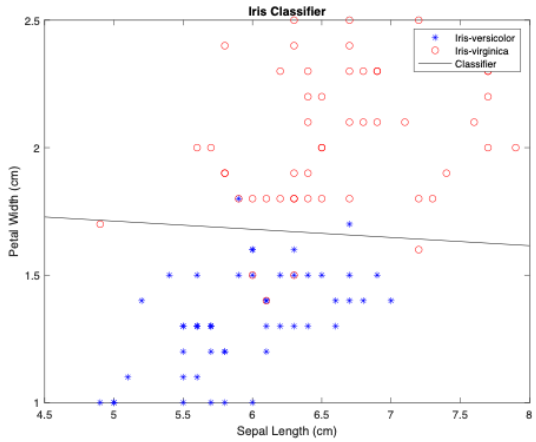


FIG. 3. Relationship between Petal Width and Sepal Length for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

poorly as a pair. That relationship contributes to the number of misclassifications, even when a third attribute is introduced. The impact that sepal length and width have on the number of misclassifications is also evident when using all four attributes. In general, increasing the number of attributes would decrease the number of misclassifications. Below is a chart detailing the number of misclassifications for each combination of attributes.

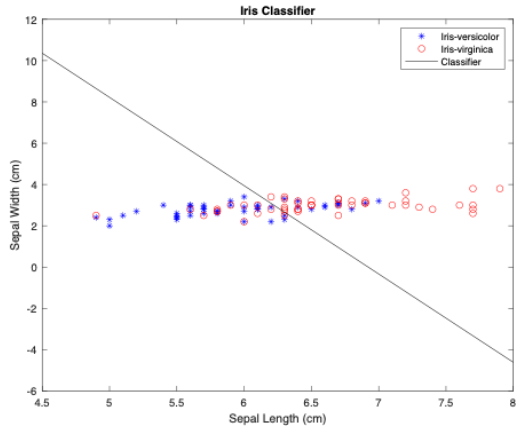


FIG. 4. Relationship between Sepal Width and Sepal Length for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

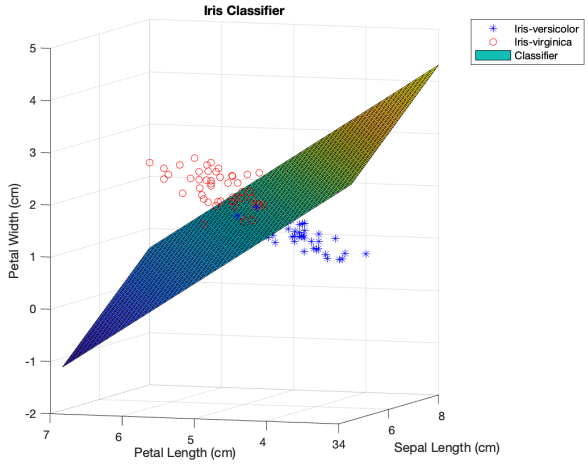


FIG. 5. Relationship between Petal Width, Petal Length, and Sepal Length for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

Attribute Combinations	No. Misclass
Sepal Length, Petal Length	5
Sepal Length, Petal Width	7
Sepal Length, Sepal Width	25
Petal Length, Petal Width	5
Sepal Width, Petal Length	7
Sepal Width, Petal Width	5
Sepal Length, Petal Length, Petal Width	3
Sepal Length, Sepal Width, Petal Length	6
Sepal Length, Sepal Width, Petal Width	5
Sepal Width, Petal Length, Petal Width	2
All Four Attributes	3

Overall, the SVM performed relatively well for using linear classification. For better performance, a nonlinear

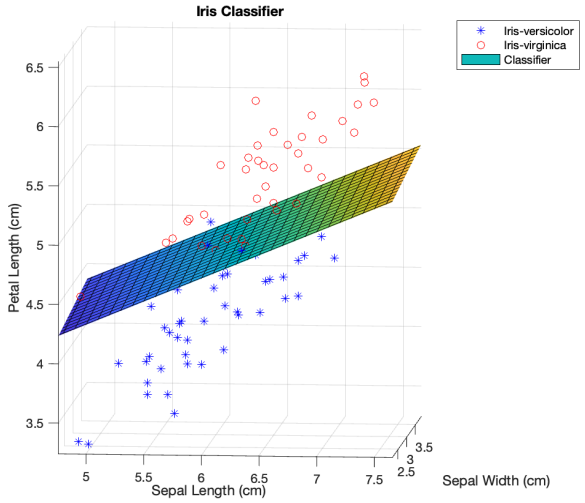


FIG. 6. Relationship between Petal Length, Sepal Length, and Sepal Width for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

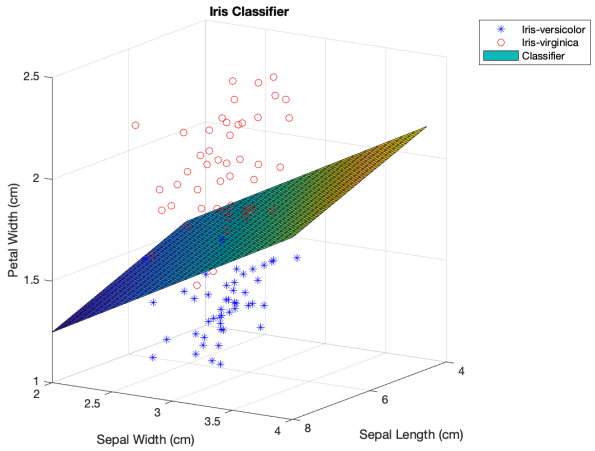


FIG. 7. Relationship between Petal Width, Sepal Width, and Sepal Length for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

approach can be taken, as in Section IV.

IV. NUMERICAL EXAMPLE II

Now, suppose the goal is to solve a business problem using SVM. In this case, the goal is to perform binary classification using baseball data from Statcast and tag each pitch thrown as either a ball, or a strike. A general "decision boundary" of the strike zone will be determined based upon how pitches in certain locations are more likely to be classified with a certain tag. It is also important to evaluate the SVM model by looking at error

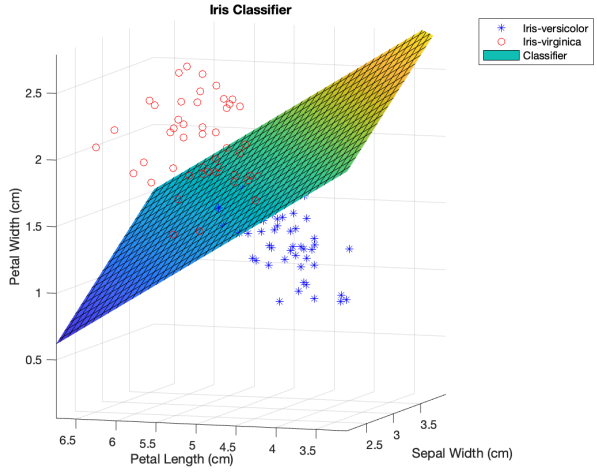


FIG. 8. Relationship between Petal Width, Petal Length, and Sepal Width for the Iris-Versicolor and Iris Virginica. Soft-margin SVM hyperplane is depicted classifying the dataset.

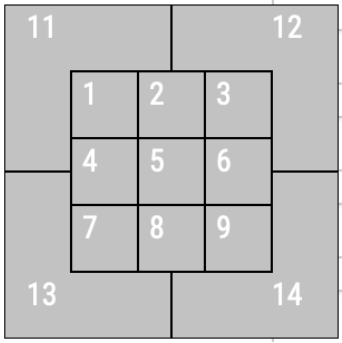


FIG. 9. Strike zone with regions 1-9 considered a strike and regions 11-14 considered a ball.

and accuracy from model output.

The objective is to find a decision boundary to best classify pitches as strikes or balls. Fig 9 represents the general strike zone. As opposed to the Iris data set, the decision boundary will not be liner because strike regions are surrounded by the ball regions.

This nonlinear relationship can also be seen in Fig 10. It's easy to tell there is no decision boundary or line that would efficiently separate the data into distinct categories. Hence, to separate the data it must be transformed and mapped to a higher dimension. The objective is to make the data linearly separable in the new higher dimension space, as in Section II (D). In the Statcast example, the data is mapped to 3 dimensions and the decision boundary or hyperplane is in 2 dimensions.

If the data for the SVM is not linearly separable, like the data from Statcast, the decision boundary would be like that of Fig 11. Here it is evident that using a linear decision boundary is not the best way to go about classifying this specific data.

To solve this, a kernel function must be used in the

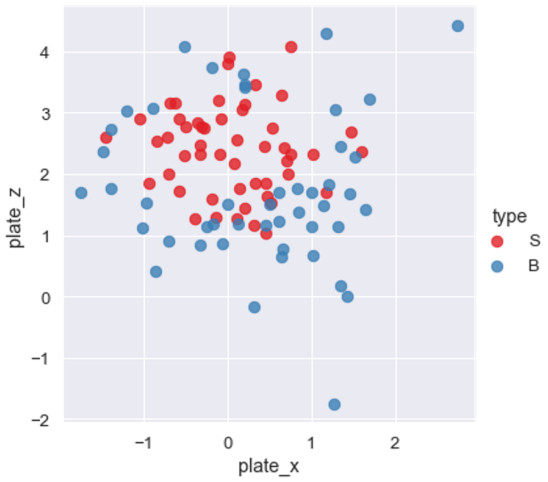


FIG. 10. A subset of the Stratcast Data used in this example, clearly not linearly separable. The red data points denote strikes and the blue are balls.

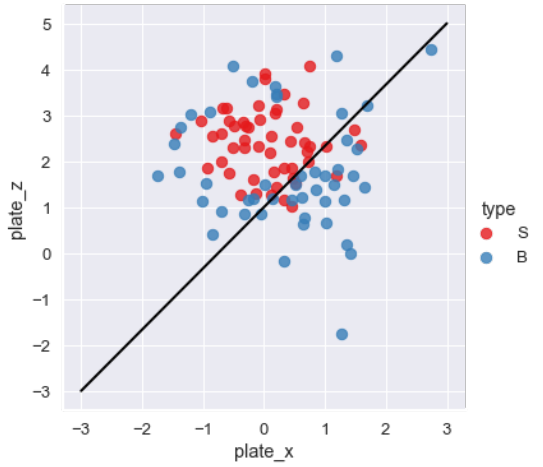


FIG. 11. An attempt to classify the Stratcast data in 2D.

SVM model. A kernel function is a mathematical function that transforms the data into the desired form. The SVM kernel trick uses a function to take a lower dimensional space and transform it into a higher, more-detailed dimensional output space. This helps deal with nonlinear separation problems and finds the best way to separate the data points. Mathematically, kernel functions get the scalar product between two points in a more-suitable feature space.

For this baseball example, it will be best to use the Radial Basis Function (RBF) kernel described in Eq. 22. This RBF kernel is the most common kernel function in SVM and is particularly popular because of its similarity to the Gaussian distribution.

Often times, using SVM to find the line, plane, or hyperplane that best separates the data classes results in errors or misclassifications. But just because a decision boundary has less errors, does not mean it is the better

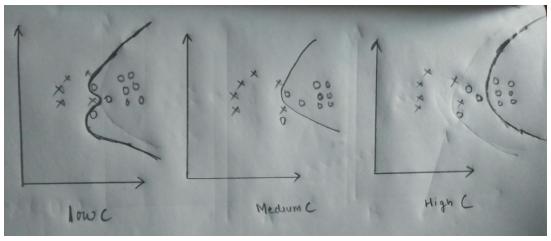
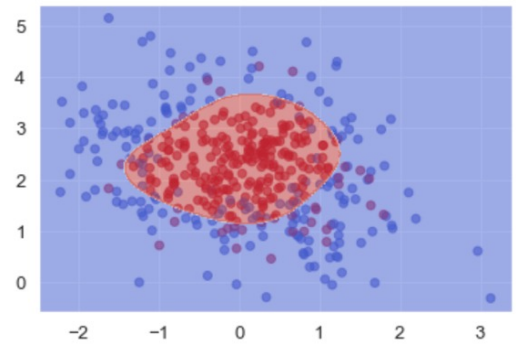


FIG. 12. Hand-drawn graphs depicting the impact of C on the classification hyperplane in a soft-margin SVM.



```
The score of SVM with gamma=1 and C=1 is:
0.7475728155339806
```

FIG. 13. Soft-margin Kernel SVM classification of Stratcast data with C=σ=1.

choice for the boundary. This is where C and σ come into play. C is the parameter used in a soft-margin SVM described in section II (C). C also controls the margin between data points, which will contribute to error calculation. There is no specific rule of thumb for a "good" C value and the C value picked is entirely dependent on the dataset in this specific business case.

Fig 12 shows a specific case where a low C gives the lowest error, a medium C value gives a medium amount of error, and a high C gives the most error. In this case, the medium C value produces the best boundary, showing that low error does not necessarily mean that it is the best boundary choice.

There is also another hyperparameter, σ. While C can be used with other kernels in SVM, σ is a parameter that is specific to the RBF kernel function (Eq. 22). The value σ is determined before training the model that decides how much curvature is in the decision boundary. A high σ has more curvature, and a low σ has a less curvature. As with C, the σ value is dependent on the specific dataset. Figs 13-17 show different values of C and σ used on the pitch location dataset. This data set contains 500 data points randomly sampled from the larger Statcast dataset to help interpret aspects of the decision boundary.

The model starts with generic values for C and σ with both parameters equal to 1. The SVM model with these parameters can be seen in Fig 14 (note red data are strikes, blue data are balls).

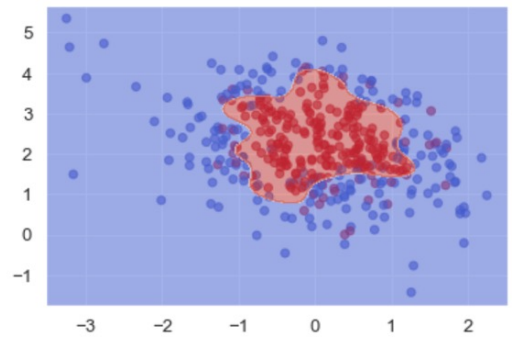


FIG. 14. Soft-margin Kernel SVM classification of Stratcast data.

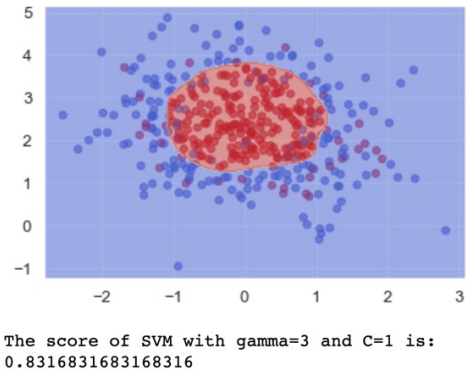


FIG. 15. Soft-margin Kernel SVM classification of Stratcast data.

For this model, the accuracy score is the total number of correct predictions divided by the total predictions. With the generic C and σ in the RBF kernel, the accuracy is about 74.8 percent. Figs 14-15 show other combinations of C and σ hyperparameters to determine the optimal values that result in greatest accuracy.

Fig 14 shows that a higher σ gives the boundary more curvature. Keeping C constant at 1,and a σ of 3 seems to give the best model. Now, what will happen if σ is kept constant at 3, which seems to be reasonable in this case. It is then possible to increase and decrease the C value and see what this looks like as well.

Keeping σ at 3, Fig 16 shows a higher C value of 1.5 gives the best accuracy but is not necessarily the best fit for the model. As shown in Fig 17, a smaller C value of 0.5 results in a lower accuracy score than that in Fig 16. The optimal hyperparameter values are when σ equals 3 and C equals 1. These values best balance the relationship between accuracy score and model practicality.

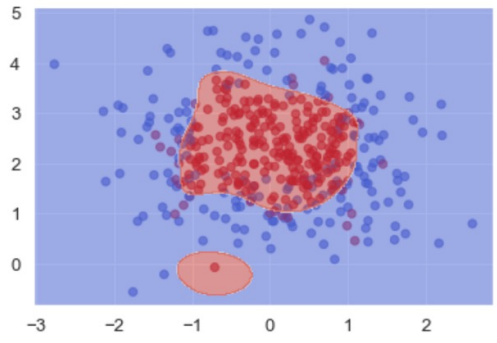


FIG. 16. Soft-margin Kernel SVM classification of Stratcast data.

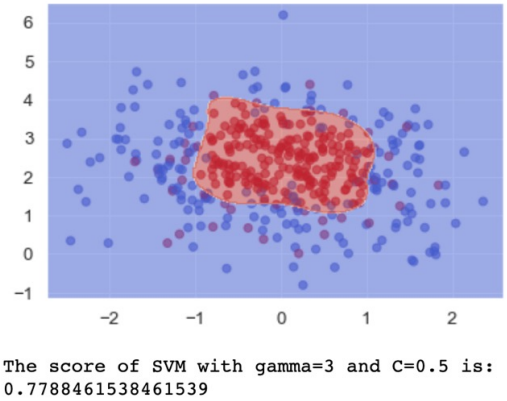


FIG. 17. Soft-margin Kernel SVM classification of Stratcast data.

V. CONCLUSION

In summary, the SVM model has solved a data driven business problem and found a good decision boundary to classify pitches as strikes or balls. This was done by understanding the business problem, the theory and mathematics behind SVM, the optimal kernel function, and the hyperparameters C and σ. Both the iris and baseball examples show that SVM provides a simple and efficient way to go about solving optimization and classification problems.

VI. BIBLIOGRAPHY

[2]
"An Introduction to Support Vector Machine Implementations in MATLAB." Andrew Zisserman, University of Oxford, David Lindsay , 10 Aug. 2003, https://www.robots.ox.ac.uk/ az/lectures/.
Kumar, A Man. "C And Gamma in SVM." Medium, Medium, 17 Dec. 2018,

https://medium.com/@myselfaman12345/c-and-gamma-in-svm-e6cee48626be. “Seven Most Popular SVM Kernels.” Dataaspirant, 17 Dec. 2020, https://dataaspirant.com/svm-kernels/t-1608054630728.
Shuzhan Fan. “Understanding the Mathematics behind Support Vector Machines.” Shuzhan Fan, 7 May 2018, https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/.
Sreenivasa, Sushanth. “Radial Basis Function (RBF) Kernel: The Go-to Kernel.” Medium, Towards Data Science, 12 Oct. 2020, https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a.
Wilimitis, Drew. “The Kernel Trick.” Medium, Towards Data Science, 21 Feb. 2019, https://towardsdatascience.com/the-kernel-trick-c98cdbcaeb3f.