SUPPORT VECTOR MACHINES

Jake Bodea

MTH 496: Topics of Mathematics (Final)

December 6, 2022

1. INTRODUCTION

1.1 BACKGROUND

In 2019, Elon Musk, founder of many Artificial Intelligence-driven companies such as Tesla, Neuralink, and OpenAI, spoke at the World Artificial Intelligence Conference in Shanghai and said, “There’s just a smaller and smaller corner of intellectual pursuits that humans are better at than computers, and every year that [corner] gets smaller and smaller. Soon, we’ll be far, far surpassed [by computers] in every way, guaranteed.” Musk clearly understands that the applications of computer science and mathematics could very well bring about a future run by these technologies. In fact, the idea that the use of AI in everyday life is something of the future is incredibly naïve. Picture a brief snapshot of a weekday morning: your iPhone sends you a notification to open Maps for directions to work as you turn on your car in the morning. Your job likely used AI to scan the hundreds of resumes received to recommend your application to your manager. As you drive, your car analyzes real-time data from its sensors to constantly ensure you don’t attempt to merge into another car or drift from your lane. As you look around at other office buildings, you notice a logo for a startup company, which may have been generated by LogoMaster.ai. AI is already everywhere.

1.2 CLARIFICATIONS

Before getting too far into the paper, it is important to understand the differences between the Data Science, Artificial Intelligence, and Machine Learning (ML) domains. Data Science is a very broad term, which, besides AI, includes fields such as Data Analytics and Business Intelligence. AI is a subset of Data Science that contains ML. The difference between AI and ML is that AI refers to how the mathematics of ML is used practically. For example, Apple recommending to open Maps for directions to work is AI, but the data used to prompt that

decision (phone connects to car's Bluetooth, previous drives during that time of the day, etc.) and the algorithm that turns the data into a decision is considered ML.

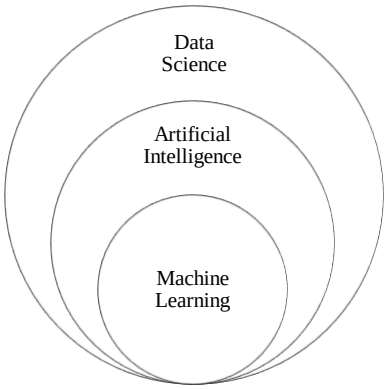

Figure 1. Diagram of the field of Data Science.

The word "algorithm" was mentioned above. All of ML, and in turn AI, is completely dependent on mathematical algorithms of varying complexity to crunch numbers and provide a statistically accurate decision. Where the power of computers comes into play is the speed at which those mathematical conclusions can be reached – especially when compared to human ability. This paper will explain the mathematics behind a ML algorithm called "Support Vector Machines" (SVM), and then demonstrate how a computer uses the algorithm in a real-world problem to provide a mathematically conceived solution.

2. THE MATHEMATICS BEHIND SVM

2.0 REFERENCES AND SOURCES

For more information on deriving the SVM algorithm, please see the following in the bibliography: the lecture from MIT posted on the OpenCourseWare channel on YouTube, *Machine Learning: The Art and Science of Algorithms that Make Sense of Data* by Peter Flach, and the video by Josh Starmer.

2.1 PURPOSE AND DEFINITIONS

First and foremost, SVM is a binary classification algorithm. This means that SVM uses some initial **training data** to predict whether new data points are either of two categories: red or blue, true or false, 1 or 0, etc. In more mathematical terms, SVM takes training vectors $\overrightarrow{x_k} \in \mathbb{R}^D$ in D dimensions (or D prediction variables) to predict whether the accompanying y_k feature is {red, blue}, {true, false}, {1, 0}, etc. To help visualize, the derivation of the algorithm will consider $\overrightarrow{x_k} \in \mathbb{R}^2$, that is, $\overrightarrow{x_k} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, though a similar method is used for any $D \in \mathbb{N}$.

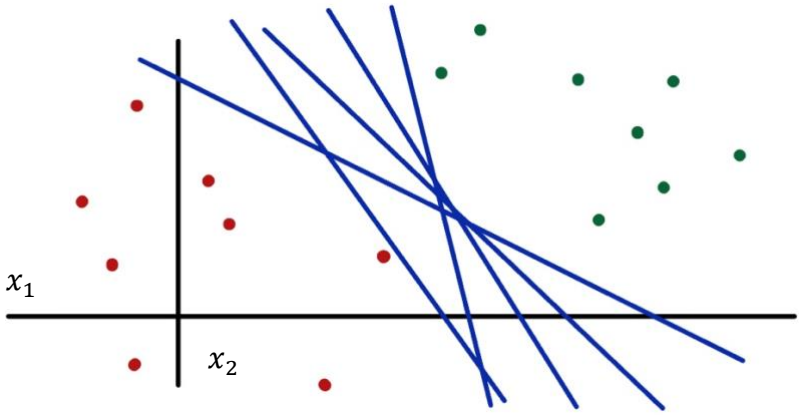

Figure 2. Two sets of binary classified data points separated by multiple hyperplanes.

Notice the two classes of training data, where each value of y_k is demonstrated by the color of the point, and each point is located at (x_1, x_2) for each vector. Really, these points symbolize the endpoint of the $\overrightarrow{x_k}$ vector from the origin. Notice that the two groups of data can be separated by infinite lines. Each of these lines is known as a **hyperplane**. The goal of SVM is to find what is known as the **optimal hyperplane**, which is the hyperplane that best separates the two classifications.

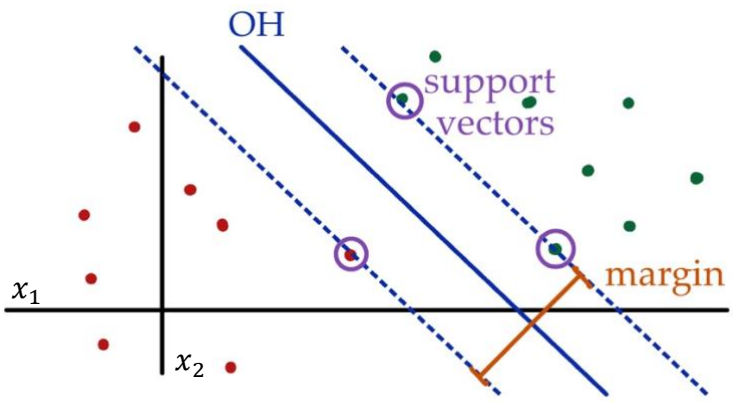

Figure 3. The optimal hyperplane of the data with the support vectors and margin.

Notice also that the optimal hyperplane is surrounded by a **margin**, twice the distance from the vectors closest to the optimal hyperplane to the optimal hyperplane itself. Those vectors are known as **support vectors**. The optimal hyperplane, then, is the hyperplane that maximizes the margin – in other words, the hyperplane that creates the largest distance between the support vectors of the two classifications of data points. The SVM algorithm finds the equation to this optimal hyperplane.

Note: since $\overrightarrow{x_k} \in \mathbb{R}^2$, the optimal hyperplane is in 1 dimension, meaning that the optimal hyperplane is a line for this example. If we looked at 1 feature, the optimal hyperplane would be a point. If we looked at 3 features, the optimal hyperplane would be a plane. If we looked at 4 features, the optimal hyperplane would be a rectangular prism, and so forth. SVM can calculate the optimal hyperplane of infinite dimensional data, should one exist.

Thus, SVM intends to find the optimal hyperplane that maximizes the margin given a training dataset, and then uses that optimal hyperplane to decide how to classify new data points.

2.2 THE MATH

2.2.1 ESTABLISHING THE DECISION RULE

The simplest way to understand how SVM is calculated is to begin by observing what we want the final product to look like, then deconstructing how to get there. Let us begin by observing a training dataset in two features, with the classifications being either $+1$ or -1. That is, let $\{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \dots, (\overrightarrow{x_N}, y_N)\}$ be a set of training data, where $\overrightarrow{x_k} \in \mathbb{R}^2$ and $y_k \in \{+1, -1\}$ for each $k \in \mathbb{N}$ such that $k \leq N$. Define $\vec{w}$ to be a normal vector to the optimal hyperplane. We do not yet know the length of $\vec{w}$, but we do know that it is perpendicular to the optimal hyperplane. Note: the figure below is intended to show the direction of $\vec{w}$, not its magnitude.

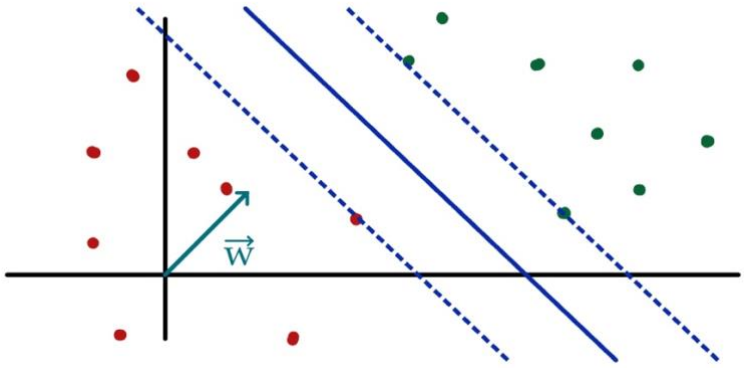

Figure 4. The SVM result with some $\vec{w}$ normal to the OH shown.

Now let $\vec{u} \in \mathbb{R}^2$ be a new vector whose associated classification is unknown (this is the unknown data point we are trying to classify as $\{+1, -1\}$).

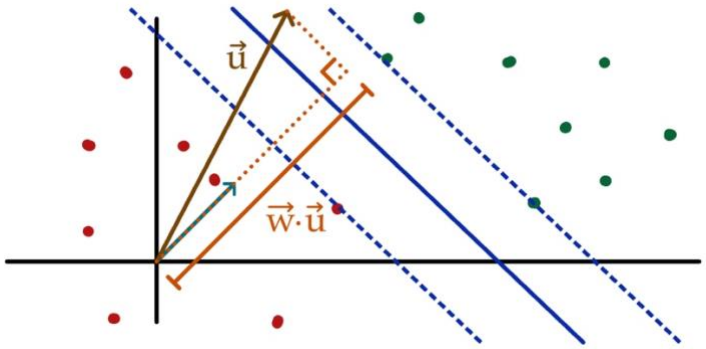

Figure 5. The geometric result of $\vec{w} \cdot \vec{u}$, where $\vec{w}$ is considered a unit vector.

Recall that the dot product $\vec{w} \cdot \vec{u}$ geometrically projects $\vec{u}$ onto $\vec{w}$ and result in a scalar: the magnitude of $\vec{u}$ in the direction perpendicular to the optimal hyperplane multiplied by the magnitude of $\vec{w}$. In the example above, $\vec{w} \cdot \vec{u}$ is shown geometrically with $\vec{w}$ as a unit vector, the classification of $\vec{u}$ seems easy to observe. But if, say, $||\vec{w}|| = 2$, then $\vec{w} \cdot \vec{u}$ would result geometrically as twice the magnitude of $\vec{u}$ in the direction of $\vec{w}$ and thus $\vec{w} \cdot \vec{u}$ may appear to cross on one side of the optimal hyperplane but really be classified on the other.

As $||\vec{w}||$ is a constant, we notice that $\vec{w} \cdot \vec{u}$ represents the magnitude of $\vec{u}$ in the direction of $\vec{w}$ multiplied by $||\vec{w}||$. So, if $\vec{w} \cdot \vec{u}$ is greater than a certain value, we would classify the point as $+1$, otherwise it would be classified as -1. Let us then define some value c such that

$$\text{if } \vec{w} \cdot \vec{u} \geq c, \text{ then } +1.$$

Then define $b = -c$ to simplify that

$$\text{if } \vec{w} \cdot \vec{u} + b \geq 0, \text{ then } +1.$$

We call this our **decision rule**. We will now solve for $\vec{w}$ and b, so that we can use this rule to determine the classification of a new data point $\vec{u}$.

2.2.2 MAXIMIZING THE MARGIN

Let $\overrightarrow{x_+}$ represent a data point in the training data set classified as $+1$ and $\overrightarrow{x_-}$ represent a point classified as -1. We can then construct our decision rule to satisfy the following for each $\overrightarrow{x_k}$ in the training data:

$$\vec{w} \cdot \overrightarrow{x_+} + b \geq +1,$$

$$\text{and } \vec{w} \cdot \overrightarrow{x_-} + b \leq -1.$$

Now recall that $y_k = +1$ for each $\overrightarrow{x_+}$ and $y_k = -1$ for each $\overrightarrow{x_-}$. Then, for the entire training dataset, when $y_k = 1$,

$$(y_k)(\vec{w} \cdot \overrightarrow{x_+} + b) \geq (y_k)(+1),$$

$$\text{so } (y_k)(\vec{w} \cdot \overrightarrow{x_+} + b) \geq 1 ,$$

and when $y_k = -1$

$$(y_k)(\vec{w} \cdot \overrightarrow{x_+} + b) \leq (y_k)(+1),$$

$$\text{so } (y_k)(\vec{w} \cdot \overrightarrow{x_+} + b) \geq 1.$$

Thus,

$$y_k(\vec{w} \cdot \overrightarrow{x_k} + b) - 1 \geq 0, \quad \forall k \in \mathbb{N} \text{ with } k \leq N.$$

As support vectors lie on the border of the margin, let us add the additional constraint for our training set that all support vectors satisfy the equation

$$y_k(\vec{w} \cdot \overrightarrow{x_k} + b) - 1 = 0.$$

Recall that we are trying to maximize the margin. Let us then find an equation for the margin that we can mathematically maximize.

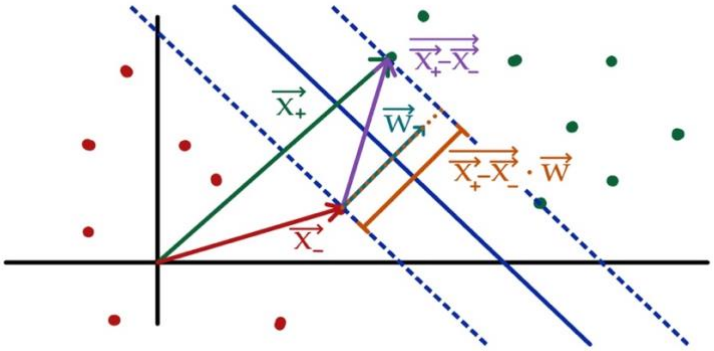

Figure 6. The different vectors used to find an equation for the width of the margin, where $\vec{w}$ is visualized as a unit vector and shown from the edge of the margin instead of the origin.

Notice that the difference vector $\overrightarrow{x_+ - x_-}$ of one positive and one negative support vector spans the width of the margin. Notice also that the dot product between that difference vector and a unit vector perpendicular to the optimal hyperplane would result in the length of the margin. Since $\vec{w}$ was defined to be normal to the optimal hyperplane, notice that the unit vector perpendicular to the optimal hyperplane is $\frac{\vec{w}}{||\vec{w}||}$, and thus, the width of the margin is equal to $\overrightarrow{x_+ - x_-} \cdot \frac{\vec{w}}{||\vec{w}||}$.

Now recall that for positive support vectors,

$$y_k(\vec{w} \cdot \overrightarrow{x_k} + b) = 1.$$

$$\Longrightarrow \vec{w} \cdot \overrightarrow{x_+} = 1 - b \text{ since } y_k = 1,$$

and for negative support vectors

$$y_k(\vec{w} \cdot \overrightarrow{x_k} + b) = 1.$$

$$\Longrightarrow \vec{w} \cdot \overrightarrow{x_-} = -1 - b \text{ since } y_k = -1.$$

Thus, the margin equals

$$\overrightarrow{x_+ - x_-} \cdot \frac{\vec{w}}{||\vec{w}||} = \frac{(\overrightarrow{x_+} \cdot \vec{w} - \overrightarrow{x_-} \cdot \vec{w})}{||\vec{w}||}$$

$$= \frac{(1-b) - (-1-b)}{||\vec{w}||}$$

$$= \frac{2}{||\vec{w}||}.$$

Recall again that we are trying to maximize this margin. Note that maximizing $\frac{2}{||\vec{w}||}$ is the same as minimizing $||\vec{w}||$. Because it will be easier computationally, we will instead minimize $\frac{1}{2}||\vec{w}||^2$.

2.2.3 LAGRANGE MULTIPLIERS

Now recall the concept of Lagrange Multipliers. This method allows us to find local minima or maxima of a function subject to some constraints, where we take the function we wish to maximize and subtract the summation of some new variables λ_k multiplied by each constraint. Since $y_k(\vec{w} \cdot \overrightarrow{x_k} + b) - 1 = 0$ are the constraints for $\vec{w}$ and involve some unknown b, let $L(\vec{w}, b, \lambda_1, \lambda_2, \dots, \lambda_N) = L$ be the Lagrange function looking to maximize the margin, where λ_k are the Lagrange multipliers. So,

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \lambda_k [y_k(\vec{w} \cdot \overrightarrow{x_k} + b) - 1]$$

is our decision function. Then, the local extremum can be found by computing the partial derivatives, setting each equal to 0 and solving for each λ_k.

Deriving by a vector is a concept not covered in undergraduate studies, so it is best perhaps to cover how this is done. $$\frac{d}{d\vec{x}} f(\vec{x}) = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix}$$

Also, recall that in two-dimensional space,

$$\frac{\partial}{\partial \vec{x}}(||\vec{x}||^2) = \frac{\partial}{\partial \vec{x}}\left(\left(\sqrt{(x_1)^2 + (x_2)^2}\right)^2\right) = \frac{\partial}{\partial \vec{x}}((x_1)^2 + (x_2)^2) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 2\vec{x}.$$

Also, notice that

$$\frac{\partial}{\partial \vec{x}}(\vec{x} \cdot \vec{y}) = \frac{\partial}{\partial \vec{x}}(x_1 y_1 + x_2 y_2) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \vec{y}.$$

Then, we can calculate that since

$$L = \frac{1}{2}||\vec{w}||^2 - \sum \lambda_k [y_k(\vec{w} \cdot \overrightarrow{x_k} + b) - 1],$$

it follows that

$$\frac{\partial L}{\partial \vec{w}} = \frac{1}{2}(2\vec{w}) - \sum \lambda_k y_k \overrightarrow{x_k}$$

$$= \vec{w} - \sum \lambda_k y_k \overrightarrow{x_k}.$$

Since we want to find the local extrema, we set

$$\vec{w} - \sum \lambda_k y_k \overrightarrow{x_k} = 0.$$

$$\Longrightarrow \vec{w} = \sum \lambda_k y_k \overrightarrow{x_k}.$$

The above equation yields an enormous conclusion: the magnitude of $\vec{w}$ is solely dependent on the training data and some calculated Lagrange multipliers. Before, there was no certainty that $\vec{w}$ had any dependence on the training data.

Next, we can find the partial derivative to find an extremum with respect to b to conclude:

$$\frac{\partial L}{\partial b} = \sum \lambda_k y_k = 0. \Rightarrow \sum \lambda_k y_k = 0.$$

So then,

$$L = \frac{1}{2}\left(\sum \lambda_k y_k \overrightarrow{x_k}\right)\left(\sum \lambda_j y_j \overrightarrow{x_j}\right) - \left(\sum \lambda_k y_k \overrightarrow{x_k}\right)\left(\sum \lambda_j\, y_j \overrightarrow{x_j}\right) - \sum \lambda_k y_k b + \sum \lambda_k\,.$$

$$L = -\frac{1}{2}\sum\sum\left(\lambda_k \lambda_j y_k y_j \overrightarrow{x_k} \cdot \overrightarrow{x_j}\right) - b\left(\sum \lambda_k y_k\right) + \sum \lambda_k\,.$$

But since we established that $\sum \lambda_k y_k = 0$,

$$L = \sum \lambda_k - \frac{1}{2}\sum\sum\left(\lambda_k \lambda_j y_k y_j \overrightarrow{x_k} \cdot \overrightarrow{x_j}\right).$$

This is now the final decision function for our optimal hyperplane, expressed exclusively in terms of the training data and constants $\lambda_1, \ldots, \lambda_N$ with the only step left being to maximize the function under the constraints $\sum \lambda_k y_k = 0$ and $\lambda_k \geq 0$. This is called the **dual optimization problem**, and requires partial derivatives and systems of equations to optimize the values of $\lambda_1, \ldots \lambda_N$. Once that is completed, solving for $\vec{w}$ can be done using the equation $\vec{w} = \sum \lambda_k y_k \overrightarrow{x_k}$ derived earlier, and then solving for b is a matter of substituting a support vector in the equation $y_k(\vec{w} \cdot \overrightarrow{x_k}) + b = 1$. Once those values are obtained, SVM yields the following conclusion for some vector $\vec{u}$ whose binary classification is unknown:

$$\text{if } \vec{w} \cdot \vec{u} + b \geq 0, \text{ then } +1, \text{ otherwise } -1.$$

Looking at our solution for the decision function L, notice that searching for the maximum-margin decision boundary is solely dependent on finding the support vectors, as they are the training examples with non-zero Lagrange multipliers that give $\vec{w} = \sum \lambda_k y_k \overrightarrow{x_k}$ and are used to then calculate b. Thus, since $\vec{w}$ is calculated by the support vectors and b is calculated with $\vec{w}$ and an arbitrary support vector, the support vectors completely determine the decision boundary.

3. A MATHEMATICAL EXAMPLE

To better understand the mathematics, let us go through a simple example of how SVM would calculate the optimal hyperplane given training data. For similar examples of using the

SVM algorithm with matrices, please see *Machine Learning: The Art and Science of Algorithms that Make Sense of Data* by Peter Flach in the bibliography.

Let the following be a sample of 3 pairs of data, given as rows of the matrix X. Note that the entry of each row of Y gives the classification of the corresponding row of X.

$$X = \begin{bmatrix} 0 & 1 \\ -1 & 0 \\ -1 & 1 \end{bmatrix}, \qquad Y = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}$$

Define the matrix X' to consist of the rows of X multiplied in each coordinate by their corresponding classification in Y, which equates to $y_k\overrightarrow{x_k}$. So,

$$X' = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ -1 & 1 \end{bmatrix}.$$

In linear algebra, the Gram Matrix (XX^T) contains the entries of all the dot products. Notice then that the Gram matrix of X' is

$$X'(X')^T = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix},$$

where the (k, j)th entry of $X'(X')^T$ is equivalent to $\left(y_k y_j \overrightarrow{x_k} \cdot \overrightarrow{x_j}\right)$. Thus, the dual optimization problem becomes:

$$L = \sum \lambda_k - \frac{1}{2}\sum\sum\left(\lambda_k \lambda_j y_k y_j \overrightarrow{x_k} \cdot \overrightarrow{x_j}\right).$$

Since the coefficient of $\lambda_k\lambda_j$ is the (k, j)th entry of $X'(X')^T$, we have that

$$\begin{aligned} L = \lambda_1 + \lambda_2 + \lambda_3 - \frac{1}{2}[(\lambda_1)^2 \quad &+ 0 \quad - \lambda_1\lambda_3 \\ 0 \quad &+ (\lambda_2)^2 - \lambda_2\lambda_3 \\ -\lambda_1\lambda_3 &- \lambda_2\lambda_3 + 2(\lambda_3)^2]. \end{aligned}$$

Recall that the dual optimization problem follows the constraints that $\sum \lambda_k y_k = 0$ and $\lambda_k \geq 0$. To enforce this constraint, let us solve for λ_1 and substitute into the equation for L. Since $-\lambda_1 - \lambda_2 + \lambda_3 = 0$, we have that $\lambda_1 = -\lambda_2 + \lambda_3$. Then

$$L = (-\lambda_2 + \lambda_3) + \lambda_2 + \lambda_3$$
$$- \frac{1}{2}[(-\lambda_2 + \lambda_3)^2 - (-\lambda_2 + \lambda_3)\lambda_3 + (\lambda_2)^2 - \lambda_2\lambda_3 - (-\lambda_2 + \lambda_3)\lambda_3 - \lambda_2\lambda_3$$
$$+ 2(\lambda_3)^3]$$
$$= 2\lambda_3 - \frac{1}{2}[((\lambda_2)^2 - 2\lambda_2\lambda_3 + (\lambda_3)^2) + \lambda_2\lambda_3 + (\lambda_3)^2 + (\lambda_2)^2 - \lambda_2\lambda_3 + \lambda_2\lambda_3 - (\lambda_3)^2 - \lambda_2\lambda_3$$
$$+ 2(\lambda_3)^2]$$
$$= 2\lambda_3 - \frac{1}{2}[2(\lambda_2)^2 - 2\lambda_2\lambda_3 + (\lambda_3)^2]$$
$$= -(\lambda_2)^2 + \lambda_2\lambda_3 - \frac{1}{2}(\lambda_3)^2 + 2\lambda_3.$$

Next, we take partial derivatives of L with respect to each λ_k and set these equal to 0. This results in a linear system of equations represented by the matrix below:

$$\begin{bmatrix} -2 & 1 & 0 \\ 1 & -1 & -2 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 4 \end{bmatrix}$$

So $\lambda_2 = 2$ and $\lambda_3 = 4$, and $\lambda_1 = -\lambda_2 + \lambda_3 = -2 + 4 = 2$. As none of the Lagrange multipliers are 0, each of the vectors are considered support vectors.

Now, we must solve for $\vec{w}$ and b. Notice that

$$\vec{w} = \sum \lambda_k y_k \overrightarrow{x_k}$$

$$= 2\begin{bmatrix} 0 \\ -1 \end{bmatrix} + 2\begin{bmatrix} 1 \\ 0 \end{bmatrix} + 4\begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ -2 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} + \begin{bmatrix} -4 \\ 4 \end{bmatrix}$$

$$= \begin{bmatrix} -2 \\ 2 \end{bmatrix}.$$

To solve for b, we can use the fact that $\overrightarrow{x_3} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ is a positive support vector and solve for b. So

$$y_3(\vec{w} \cdot \overrightarrow{x_3}) + b = 1.$$

$$\Longrightarrow (1)\big((-2)(-1) + (2)(1)\big) + b = 1.$$

$$\Longrightarrow (2 + 2) + b = 1.$$

$$\Longrightarrow 4 + b = 1.$$

$$\Longrightarrow b = -3.$$

Thus, the equation for our optimal hyperplane is

$$0 = \vec{w} \cdot \vec{x} + b.$$

$$\Longrightarrow 0 = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 3.$$

$$\Longrightarrow 0 = -2x_1 + 2x_2 - 3.$$

$$\Longrightarrow -2x_2 = -2x_1 - 3.$$

$$\Longrightarrow x_2 = x_1 + \frac{3}{2}.$$

Recall also that the margin is equal to

$$\frac{2}{||\vec{w}||} = \frac{2}{\sqrt{(-2)^2 + (2)^2}} = \frac{2}{\sqrt{4+4}} = \frac{2}{\sqrt{8}} = \frac{\sqrt{2}}{2}.$$

Below is the solution graphically, with a margin of $\frac{\sqrt{2}}{8}$ included to emphasize the three support vectors.

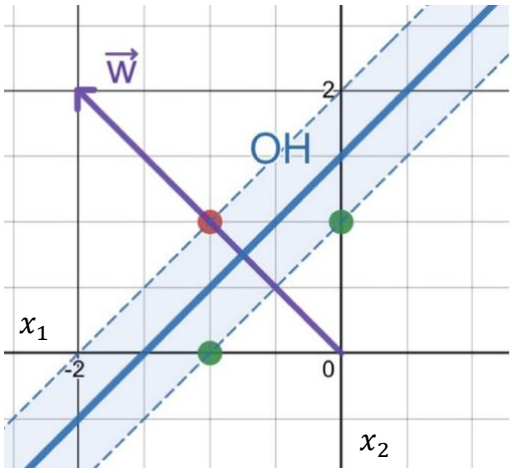

Figure 7. The result of SVM in this simple example.

Based on the training data, our model would classify some $\overrightarrow{u_1} = [0 \quad 0]$ as -1 or green and $\overrightarrow{u_2} = [-2 \quad 1/2]$ as $+1$ or red. But say some $\overrightarrow{u_3} = [-1/2 \quad 3/4]$ is evaluated. The algorithm would predict that $\overrightarrow{u_3}$ is -1 or green since $\overrightarrow{u_3}$ lies on the same side of the optimal hyperplane as the green data points. However, should $\overrightarrow{u_3}$ be considered a new training vector to better improve the model, since $\overrightarrow{u_3}$ lies within the margin, a new optimal hyperplane and new support vectors would likely be calculated to better fit the new point – identifying it as a new support vector for the model. Both $\vec{w}$ and b would likely have different values (in other words, the position and slope of the optimal hyperplane would change), and thus our model would be more accurate.

4. REAL-WORLD APPLICATIONS

Now that a fundamental understanding of the SVM algorithm has been achieved, we can look at some practical implementations of the algorithm. As the headers will outline, this section will span two projects: Australian weather prediction and video game rating prediction. We will

split the data, found on Kaggle and referenced in the Bibliography, into 75% training data and 25% testing data to evaluate SVM's performance. The coding itself will be done in the Python programming language and will utilize the Pandas and Scikit-Learn libraries.

4.1 AUSTRALIA RAINY WEATHER PREDICTION

For this project, we are given a dataset containing 10 years of weather information from Australia, with the intent being to predict whether or not it will rain "tomorrow" based on the following features to predict if it will/wont rain "tomorrow" (as in, to predict if 'RainTomorrow' will be 1 or 0): location (city), minimum/maximum temperatures of the day, amount of rainfall that day in mm, the humidity at 9am and 3pm, the fraction of the sky covered in clouds at 9am and 3pm (measured in oktas), and the temperature at 9am and 3pm in degrees Celsius. Thus, we have 145,460 rows of data in 10 dimensions.

Location	MinTemp	MaxTemp	Rainfall	Humidity9am	Humidity3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	Location_Encoded	RainTomorrow
Tuggeranong	0.8	9.8	0.0	62.0	57.0	0.0	0.0	6.2	8.2	16	0
CoffsHarbour	17.2	25.3	2.0	92.0	68.0	7.0	3.0	20.0	24.6	3	1
Darwin	24.8	31.8	7.2	84.0	76.0	7.0	7.0	28.0	29.9	46	1
Cobar	13.0	26.2	0.0	42.0	25.0	0.0	1.0	17.9	24.8	2	0
Newcastle	13.4	24.7	13.4	84.0	62.0	0.0	1.0	19.5	23.2	5	0
Darwin	19.4	29.7	0.0	50.0	25.0	1.0	1.0	22.6	29.5	46	0
Newcastle	21.4	27.3	46.8	64.0	62.0	6.0	6.0	24.5	26.5	5	0
NorahHead	20.0	25.3	0.0	83.0	80.0	0.0	0.0	22.2	22.3	6	1
Ballarat	8.9	21.0	0.2	63.0	33.0	1.0	0.0	13.0	19.9	18	0
Hobart	8.5	17.5	0.2	86.0	56.0	7.0	7.0	10.4	16.6	43	0

Figure 8. A truncated sample of the Australian Weather Dataset.

As SVM only works with numeric data, notice the 'Location_Encoded' column above. This column assigns each city ('Location') a number. This process of converting text data to numeric data must be emulated when using SVM on a data set.

After the necessary data transformations were made, our data set is ready to be fed into the SVM algorithm.

```
1  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```
executed in 77ms, finished 23:00:36 2022-11-07

```
1  model = SVC()
2  model.fit(X_train, y_train)
```
executed in 7m 5s, finished 23:10:51 2022-11-07

```
1  y_pred = model.predict(X_test)
2  accuracy_score(y_test, y_pred)
```
executed in 6ms, finished 23:10:51 2022-11-07

```
0.8314863192630276
```

Figure 9. Code snippet of training and testing the SVM model on the Australian weather data.

We first split the data into *X* and *y* variables, which essentially function as the matrices used in section 2.3, then split each again into 75% training data and 25% testing data. This development of a model and then testing the model's performance is typical in the machine learning world. The testing data is used to create a model, then the model predicts classifications for the 'X_test' data and compares the accuracy to the actual classifications of those points. Notice that the second box, where we fit the model with the training data, took 7 minutes to run. In this step, the computer did calculations similar to our work in section 2.3, but on 145,460 data points of 10 dimensions instead of 3 data points in two dimensions. Doing these calculations by hand is nearly impossible, hence the value of technology in mathematics. Notice at the bottom of the figure above that the model was 83% accurate in predicting whether or not it would rain "tomorrow" based on the features. Additional investigation reveals that the model calculated 39,459 support vectors. While not covered in this paper, there are additional parameters that could go into the SVM algorithm that may change both the accuracy and number of support vectors in the data.

4.2 VIDEO GAME RATING PREDICTION

This next project is simpler and included to demonstrate the speed of SVM when predicting using binary features as opposed to the non-binary data from example 4.1.

ESRB is a company that assigns maturity ratings to video games: “E” for “everyone,” “ET” for “everyone 10 and up,” “T” for “teen,” and “M” for “mature.” The dataset consists of every rating ESRB has ever given, as well as whether features such as blood, alcohol, gambling, or sexual content are present in the game. Below is a brief snapshot of the data.

title	alcohol_reference	blood_and_gore	sexual_content	esrb_rating
DiRT Rally 2.0	0	0	0	E
The Princess Guide	0	0	0	T
Battlefield 1 Revolution	0	0	0	M
Headlander	0	0	0	T
Disco Dodgeball Remix	0	0	0	ET
The Lost Child	1	0	0	T
Cobra Kai: The Karate Kid Saga Continues	1	0	0	T
Darkwood	0	1	1	M
RAID: WORLD WAR II	0	1	0	M
Q.U.B.E. 2	0	0	0	T

Figure 10. Truncated sample of the ESRB video game dataset.

We will evaluate the consistency of the maturity ratings based on 31 features of the video games by trying to predict whether a game was given an M or not. After some light data manipulation, we can use practically the same code as earlier to train a prediction model.

```
1 X = df.drop(columns=['console','title','esrb_rating'])
2 y = df.esrb_rating.apply(lambda x: int(x=="M"))
```
executed in 5ms, finished 12:13:26 2022-11-29

```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```
executed in 7ms, finished 12:13:28 2022-11-29

```
1 model = SVC()
2 model.fit(X_train, y_train)
```
executed in 47ms, finished 12:13:28 2022-11-29

```
SVC()
```

```
1 y_pred = model.predict(X_test)
2 accuracy_score(y_test, y_pred)
```
executed in 63ms, finished 12:13:29 2022-11-29

```
0.9499165275459098
```

Figure 11. Code snippet of transforming the ESRB video game data set and training/testing the SVM model to predict whether the rating was ‘M’ or not.

For this project, however, the model took only 47 milliseconds to train, and resulted in an accuracy of 95%, despite having triple the number of features as the previous example. The two largest factors to this speed are the quantity of data (1,885 games vs. 145,460 days of weather) and the diversity of the features' values; each of this example's features were binary choices, whereas the previous example had features such as rainfall that were continues from 0 up. Regardless, we can conclude based on the 95% accuracy of the predictions that a game's maturity rating is not at all randomized, but instead a result of a clear consistency given the presence of certain features.

For curiosity's sake, let us limit the data to results where the rating was either T or M. It would make sense that the accuracy of our model would fall, as there is likely more overlap between T and M ratings than E and M ratings (i.e., both T and M would allow for violence, but E would not).

```
df = df[df.esrb_rating == "T"].append(df[df.esrb_rating == "M"])

X = df.drop(columns=['console','title','esrb_rating'])
y = df.esrb_rating.apply(lambda x: int(x=="M"))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = SVC()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
accuracy_score(y_test, y_pred)
```

executed in 54ms, finished 12:18:48 2022-11-29

```
0.9319526627218935
```

Figure 12. Code snippet of truncating the ESRB video game dataset to only values of 'T' and 'M', then training/testing the SVM model to predict whether the rating was 'M' or not.

This yields an accuracy score of 93%. While the accuracy slightly decreases, it is still very high. This again suggests that ESRB's rating algorithm is consistent based on the presence of certain features in the game – even for ratings that should have many overlapping features.

5. CONCLUSION

A mentor once told me: “The world is constantly figuring out ways to get more data, so those who know what to do with it and how to use it for our benefit are the ones that will make it in life.” In the world where Mark Zuckerberg’s Meta pushes to change the virtual reality field, Apple becomes the world’s first trillion-dollar company, and autonomous driving cars are now driving alongside manual drivers, mathematicians and computer scientists have every opportunity to have a say in what the future holds. From the AI perspective, SVM is one of many algorithms used to shape the prediction and automation technological advances. Applications such as Natural Language Processing and Computer Vision are built on these mathematical algorithms and are constantly evolving even newer technologies (see how OpenAI’s ChatGPT makes Google Search seem outdated). In fact, this very conclusion paragraph will become outdated, and the breakthroughs in technology mentioned here will soon be replaced by their modernizations. Yet, in the grand scheme of all these technological evolutions, the reality is that a strong understanding of mathematics is required to ensure that we do not get run over by an ever-changing world, but instead can become the architects of it.

Bibliography

Alhamad, Mohammad. n.d. *Video Games Rating by 'ESRB'.* Kaggle. https://www.kaggle.com/datasets/imohtn/video-games-rating-by-esrb

Deosemroth, Marc Peter, A. Aldo Faisal, and Cheng Soon Ong. 2020. *Mathematics for Machine Learning.* Cambridge University Press.

Flach, Peter. 2012. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data.* Cambridge: Cambridge University Press.

Marius, Hucker. 2020. "Multiclass Classification with Support Vector Machines (SVM), Dual Problem and Kernel Functions." *Towards Data Science.* June 9. Accessed October 25, 2022. https://towardsdatascience.com/multiclass-classification-with-support-vector-machines-svm-kernel-trick-kernel-functions-f9d5377d6f02.

OpenCourseWare, MIT. 2014. *16. Learning: Support Vector Machines.* Video. Cambridge, January 10. https://www.youtube.com/watch?v=_PwhiWxHK8o.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2022. "Support Vector Machines." *scikit-learn.* Accessed October 25, 2022. https://scikit-learn.org/stable/modules/svm.html#.

Philippe. n.d. *Australia, Rain Tomorrow.* Kaggle. https://www.kaggle.com/datasets/filhypedeeplearning/australia-rain-tomorrow.

Ray, Sunil. 2015. "Understanding Support Vector Machine(SVM) algorithm from examples (along with code)." *Analytics Vidhya.* October 6. Accessed October 25, 2022. https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/.

Starmer, Josh. 2019. *Support Vector Machines Part 1 (of 3): Main Ideas!!!* Video. September 2019. https://www.youtube.com/watch?v=efR1C6CvhmE.