2016

The Optimization behind Support Vector Machines and an Application in Handwriting Recognition

Caitlin Snyder
Xavier University - Cincinnati, snyderc6@xavier.edu

The Optimization behind Support Vector Machines and an Application in Handwriting Recognition

Caitlin Snyder

April 29, 2016

Abstract

Support Vector Machines(SVMs) are a unique tool for classification of data and are used for solving real world problems such as image classification and text analysis. This project explores the underlying mathematics optimization problem behind the algorithm in a Support Vector Machine. The original minimization problem is described and an equivalent maximization formulation is derived. Various two and three dimensional examples are given to illustrate how the optimization gives a useable result with both linearly and nonlinearly separable data. Finally, this tool is applied in a handwriting character recognition problem. Popular SVM kernels are explored and their respective accuracy percentages in identifying between characters are compared.

1 Introduction

My motivation to learn about Support Vector Machines stemmed from my REU over the summer. We used SVMs to classify Implicit Association Test takers as cheaters or non-cheaters. Implicit Association Tests are utilized to determine if a person has a bias by measuring reaction times when classifying objects (terms or pictures). A person is able to cheat the test by slowing down their reaction times if they are aware of how bias is measured. A problem exists with simply looking at reaction times and classifying the slow times as cheaters. People with naturally slower reaction times or those who may become distracted during the test may be incorrectly classified as cheaters. The SVM is useful in this type of classification because it is able to determine a connection between cheaters that a human eye may not see. It is also helpful with Implicit Association Tests because the test taker is asked to classify up to 200 objects and can analyze the times much quicker. For this project, I explore the underlying optimization method and study examples in 2 and 3 dimensions of small data sets. I also focus on a different application for SVMs: handwriting and letter recognition.

2 Background - What is a SVM?

Support Vector Machines are a machine learning tool utlilized for data classification and regression. The implementation of a Support Vector Machine requires two sets of data: training and testing. The training data is classified into the correct categories by the user. From this data, a model is produced through an optimization problem. This model creates a hyperplane which linearly separates the testing data set into the correct classifications.

3 Example in 2D with 20 data points

Support Vector Machines are usually applied to large data sets in high dimensions that are hard, or impossible, to separate by hand. In order to understand how a SVM works, it is helpful to look at an example that can be solved relatively easily. In this example there are 20 data points:

$x_1 = (4, 3), y_1 = 1$
$x_2 = (3, 0), y_2 = 1$
$x_3 = (4, 0), y_3 = 1$
$x_4 = (2, -1), y_4 = 1$
$x_5 = (3, -2), y_5 = 1$
$x_6 = (2, -3), y_6 = 1$
$x_7 = (1, -4), y_7 = 1$
$x_8 = (0, -5), y_8 = 1$
$x_9 = (2, -5), y_9 = 1$
$x_{10} = (5, -3), y_{10} = 1$
$x_{11} = (-3, 3), y_{11} = -1$
$x_{12} = (0, 3), y_{12} = -1$
$x_{13} = (0, 1), y_{13} = -1$
$x_{14} = (1, 2), y_{14} = -1$
$x_{15} = (2, 3), y_{15} = -1$
$x_{16} = (-1, 1), y_{16} = -1$
$x_{17} = (2, 5), y_{17} = -1$
$x_{18} = (-2, -1), y_{18} = -1$
$x_{19} = (-1, -2), y_{19} = -1$
$x_{20} = (1, -1), y_{20} = -1$

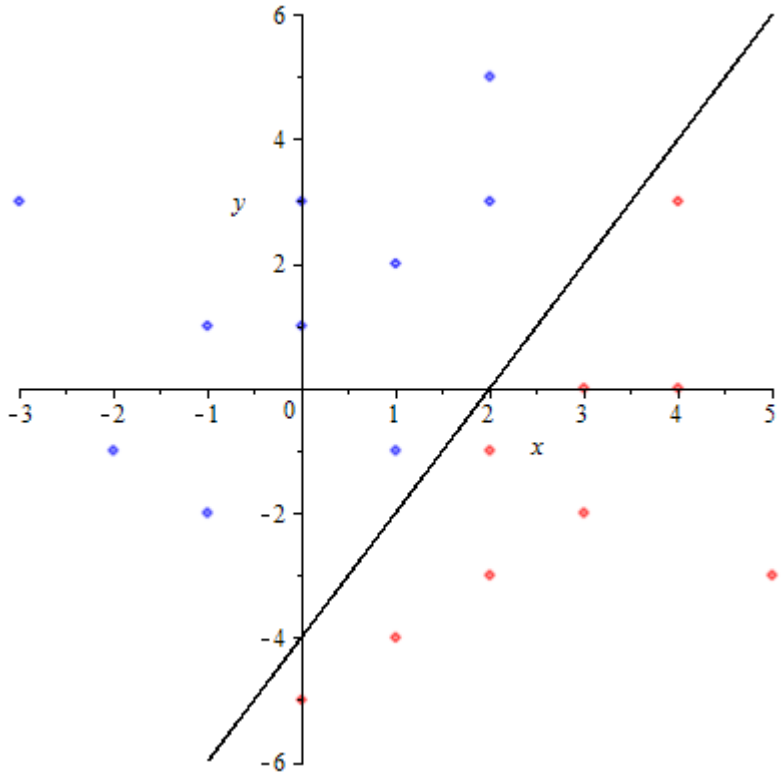
The x_i variables denote the data point and the y_i variables denote which category each point is classified as. It is common practice to use 1 and -1 to denote the two different categories. In a problem existing in the n^{th} dimension, the x_i variables will be vectors of length n while the y_i variables will always be 1 or -1. Each entry in the x_i vector will be assigned information. For example, say we would like to separate data points that represent people that make over \$75,000 a year and those that make less than \$75,000 a year. The y_i would be 1 for those who make over \$75,000 and -1 would represent those who make less than \$75,000. Each entry in the x_i vector would represent different information like age, gender, career, location, etc.

In order to separate the data points, a hyperplane is used (discussed in the next section). The data in this example can be separated using the line:

$$2x - y - 4 = 0$$

as seen in Figure 1 where the $y_i = 1$ are in red and y_i = -1 are in blue.

Figure 1: Example in 2D with 20 points



4 Hyperplane

The equation for the hyperplane created by the SVM is based on a training data set. The training data set is composed of labeled vectors that are of the form $(\mathbf{x}_i, y_i)$ such that $\mathbf{x}_i$ is the i^{th} vector in the data set and y_i is the label associated with $\mathbf{x}_i$. The label y_i indicates which category the i^{th} vector belongs in.

The equation of a hyperplane comes from a linear discriminant function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{1}$$

where b is the bias term and $\mathbf{w}$ is the weight vector. When $f(\mathbf{x}) = 0$ we call this the seperating hyperplane because it divides the space into two classes dependent upon the sign of the discriminant function. In two dimensions, $\mathbf{w} =< w_1, w_2 >$ and $\mathbf{x} =< x, y >$ will result in a line, $f(\mathbf{x}) = w_1x + w_2y + b$.

Consider the example in Section 3, $\mathbf{w} =< 2, -1 >, \mathbf{x} =< x, y >$, and $b = -4$.

$$\mathbf{w}^T\mathbf{x} + b = 0 \implies 2x - y - 4 = 0$$

This line is the separating hyperplane for this data set. Consider the vectors $\mathbf{x_1} =< 4, 3 >$ and $\mathbf{x_{11}} =< -3, 3 >$ from this example. Evaluating the discriminant function with each results in:

$$f(\mathbf{x_1}) = f(< 4, 3 >) =< 2, -1 >^T < 4, 3 > -4 = 1$$

$$f(\mathbf{x_2}) = f(< -3, 3 >) =< 2, -1 >^T < -3, 3 > -4 = -1$$

The different signs of $f(\mathbf{x_1})$ and $f(\mathbf{x_{11}})$ indicate that $\mathbf{x_1}$ and $\mathbf{x_{11}}$ belong to two different class, which is true since $f(\mathbf{x_1})$ has the label $y_1 = 1$ and $f(\mathbf{x_{11}})$ has the label $y_{11} = -1$.

5 What is the Optimal Hyperplane?

The goal of a SVM is to create a model that produces a hyperplane which will linearly separate a data set into specific categories. However, there are many different hyperplanes that could achieve this goal for any given data set. Figure 2 below illustrates this point for the example described above. The question emerges: which hyperplane is optimal?

5.1 Margin

The margin of a hyperplane [1] is defined as

$$m_D(\mathbf{w}) = \frac{1}{2}\hat{\mathbf{w}}^T(\mathbf{x}_+ - \mathbf{x}_-) \tag{2}$$

where $\hat{\mathbf{w}}$ is the unit vector of the weight vector, $\mathbf{w}$, and $\mathbf{x}_+, \mathbf{x}_-$, the support vectors, are the points closest to the hyperplane.

From the definition of the hyperplane,

$$f(\mathbf{x}_+) = \mathbf{w}^T\mathbf{x}_+ + b$$

$$f(\mathbf{x}_-) = \mathbf{w}^T\mathbf{x}_- + b.$$

Assume that $\mathbf{x}_+, \mathbf{x}_-$ are equidistant from the hyperplane, such that $f(\mathbf{x}_+) = a$ and $f(\mathbf{x}_-) = -a$ for some postive, constant a. [1]

Set $a = 1$ by manipulating the data points by a fixed number until this is true (Note: this doesn't really change the margin, just the units). Subtracting $f(\mathbf{x}_+)$ and $f(\mathbf{x}_-)$ leads to the equation:

$$\mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-) = 2a$$

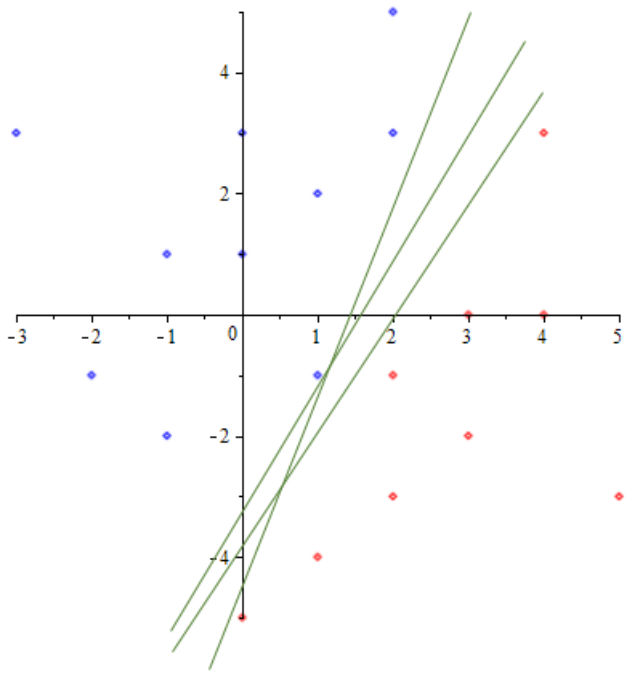


Figure 2: Many Possible Separating Hyperplanes

but $a = 1$ so,

$$\mathbf{w}^T(\mathbf{x}_+ - \mathbf{x}_-) = 2$$

Dividing this equation by $||\mathbf{w}||$ results in:

$$\left(\frac{\mathbf{w}}{||\mathbf{w}||}\right)^T (\mathbf{x}_+ - \mathbf{x}_-) = \frac{2}{||\mathbf{w}||}.$$

By the definition of unit vector and dividing by 2, this simplifies to:

$$\frac{1}{2}\hat{\mathbf{w}}^T(\mathbf{x}_+ - \mathbf{x}_-) = \frac{1}{||\mathbf{w}||}. \tag{3}$$

From the definition of margin (2) and (3) ,

$$m_D(\mathbf{w}) = \frac{1}{||\mathbf{w}||}. \tag{4}$$

Figure 3 below illustrates the margin of the separating hyperplane in the example described in Section 3. The circled points indicate the support vectors.

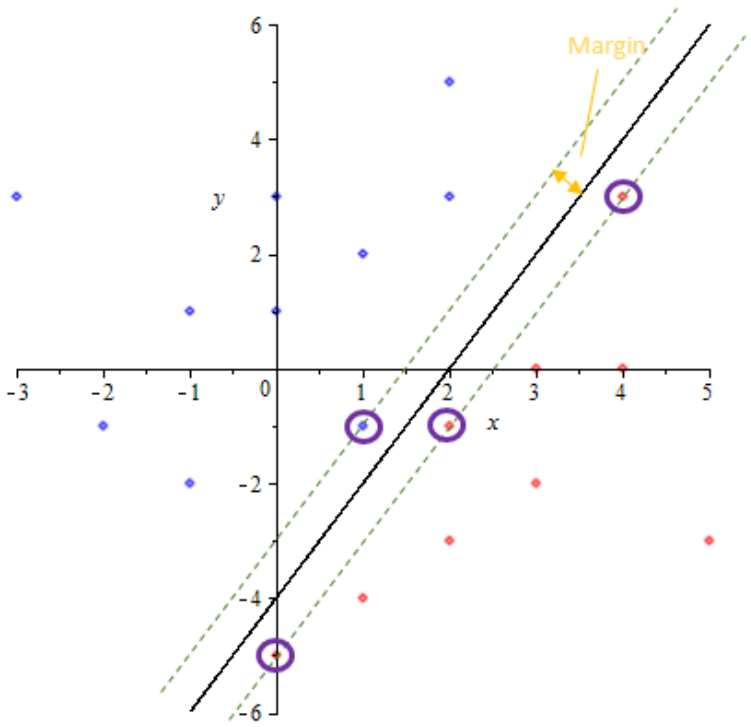


Figure 3: Margin and Support Vectors

5.2 Why do we care about the margin?

The goal of a SVM is to create a model that produces a hyperplane which will linearly separate data into specific categories. An ideal hyperplane would not only separate the training set of vectors but would correctly classify other data sets. The optimal hyperplane will have a maximized margin. If the distance between the closest vector and the hyperplane is large there is a smaller chance that a vector will be incorrectly classified.

Maximizing equation (4) is equivalent to the optimization problem:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1. \end{aligned} \tag{5}$$

The contraints force every vector to be classified into the correct category (indicated by y_i's). Using the definition of the discriminant function (1), we could re-write the constraints to be

$$y_i f(\mathbf{x}) \geq 1.$$

The constraints requires the label of the vectors and evaluation of the discriminant function to have the same sign. Consider the hyperplane example in Section 4 where $f(\mathbf{x}) =< 2, -1 >^T \mathbf{x} - 4$. Suppose the vector $\mathbf{x_1} =< 4, 3 >$ has label $y_1 = -1$ and is incorrectly classified. Evaluating the discriminant fuction results in $f(\mathbf{x}) = f(< 4, 3 >) = 1$. This violates the constraints since $y_i f(\mathbf{x}) = -1 \ngeq 1$. Thus incorrectly classifying vectors is impossible with the given constraints.

5.3 Soft-Margin SVM

The requirement of correct classification for every training vector in the optimization problem descibed above defines it as a Hard-Margin SVM. The goal of a SVM is to create a model that will linearly separate data sets into correct classifications. The model is created from the training vectors but a good model could be used on other similar data sets. For example, earlier we described a problem where we wanted to classify a person based on if they earning more or less than $75,000. We would input a data set where each person was classified as earning more or less than $75,000 and the SVM would output a model of a hyperplane to separate them. We already know if each person from the input data set made more or less than $75,000 so it wouldn't make sense to use this model to classify one of them. The model is actually useful for people that we have no idea how much they make. In this case, we don't really care if all of the training vectors are classified as correct, just enough to make the model accurate enough.

The more commonly used Support Vector Machine, called a Soft-Margin SVM, allows for error in the classification of the training data. It is defined as follows:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i. \end{aligned}$$

where ξ_i's are slack variables that allow an example to be in the margin($1 \geq \xi_i \geq 0$) or misclassified ($\xi_i \geq 1$) and the constant C sets the relative importance of maximizing the margin and minimizing the amount of slack.

The Soft-Margin SVM requires a balance between not misclassifying too many training vectors and maximizing the margin. The constant C which controls this balance, is one of the user inputs the SVM requires.

6 Kernels

So far we have been looking at data sets that are linearly separable. However, most data sets are not linearly separable in their given form. How does the SVM handle this?

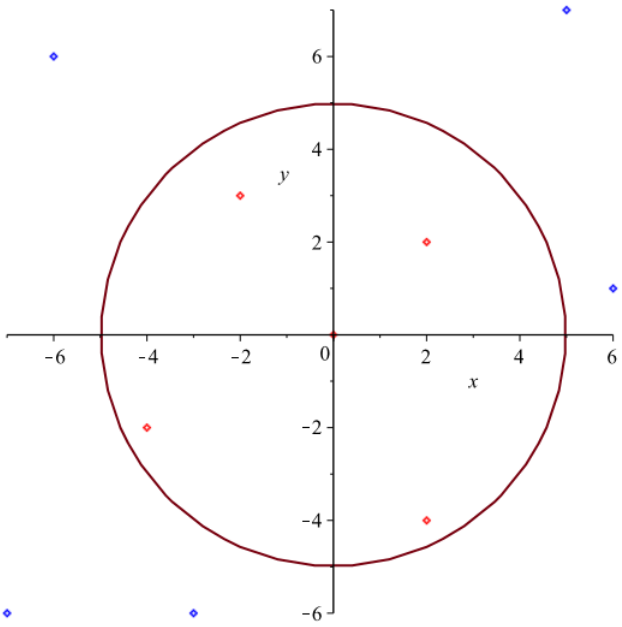
6.1 Kernel Example

Consider the data points:

$x_1 = (0, 0), y_1 = 1$
$x_2 = (2, 2), y_2 = 1$
$x_3 = (2, -4), y_3 = 1$
$x_4 = (-4, -2), y_4 = 1$
$x_5 = (-2, 3), y_5 = 1$
$x_6 = (6, 1), y_6 = -1$
$x_7 = (5, 7), y_7 = -1$
$x_8 = (-6, 6), y_8 = -1$
$x_9 = (-7, -6), y_9 = -1$
$x_{10} = (-3, -6), y_{10} = -1$

Performing the same optimization process we used for the example in Section 3 will result in a linearly separating hyperplane, a line in this case. As shown in Figure 4, these data points are separated by a circle and a line would incorrecly classify many points.

Figure 4: Kernel Example



A solution to this problem emerges if we map these 2-dimensional data points into 3 dimensions. Consider the function

$$\phi : \mathbb{R}^2 \mapsto \mathbb{R}^3$$

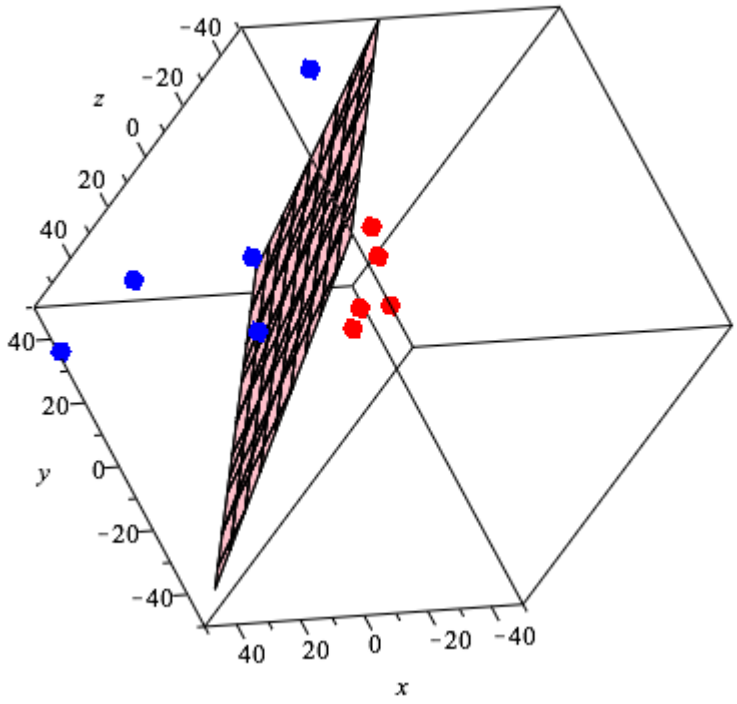
$$\phi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Applying this mapping to the previous data points results in:

$x'_1 = \phi(0,0) = (0,0,0)$
$x'_2 = \phi(2,2) = (4,4,4\sqrt{2})$
$x'_3 = \phi(2,-4) = (4,9,-6\sqrt{2})$
$x'_4 = \phi(-4,-2) = (4,16,-8\sqrt{2})$
$x'_5 = \phi(-2,3) = (16,4,8\sqrt{2})$
$x'_6 = \phi(6,1) = (9,36,18\sqrt{2})$
$x'_7 = \phi(5,7) = (25,49,35\sqrt{2})$
$x'_8 = \phi(-6,6) = (36,1,6\sqrt{2})$
$x'_9 = \phi(-7,-6) = (36,36,-36\sqrt{2})$
$x'_{10} = \phi(-3,-6) = (49,36,42\sqrt{2})$

Note: The labels, $y_i s$, do not change.
The figure below illustrates the points and the hyperplane that separates them.

Figure 5: Projected Points in 3D



6.2 Dual Formulation

For the optimization problem in (5) the Lagrangian is defined as [2]:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i (y_i(\mathbf{x}_i^T \mathbf{w} + b) - 1) \tag{6}$$

where the $\alpha_i s$ are the Lagrange multipliers.

The general form of the optimization problem can be defined as:

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) \\ &\text{subject to} && c_i(x) \geq 0 \text{ for all } i \in [n]. \end{aligned} \tag{7}$$

Theorem 1 (Karush-Kuhn-Tucker (KKT) for Differentiable Convex Problems)**.** *A solution to the optimization problem (7) with convex, differentiable f, c_i is given by $\bar{x}$, if there exists some $\bar{\alpha} \in \mathbb{R}^n$ with $\alpha_i \geq 0$ for all $i \in [n]$ such that the following conditions are satisfied:*

$$\partial_x L(\bar{x}, \bar{\alpha}) = \partial_x f(\bar{x}) + \sum_{i=1}^{n} \bar{\alpha_i} \partial_x c_i(\bar{x}) = 0 \quad \textit{(Saddle Point in } \bar{x}) \tag{8}$$

$$\partial_{\alpha_i} L(\bar{x}, \bar{\alpha}) = c_i(\bar{x}) \leq 0 \quad \textit{(Saddle Point in } \bar{\alpha}) \tag{9}$$

$$\sum_{i=1}^{n} \bar{\alpha_i} c_i(\bar{x}) = 0 \quad \textit{(Vanishing KKT-Gap).} \tag{10}$$

From Theorem 1, the Lagrangian (6) must be maximized with respect to α_i and minimized with respect to $\mathbf{w}$ and b. In other words, from (8)

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0,$$

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0.$$

Taking the partial of the Lagrangian with respect to b leads to:

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\partial}{\partial b} \left(\frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{n} (\alpha_i y_i \mathbf{x}_i^T \mathbf{w} + \alpha_i y_i b - \alpha_i) \right) = \sum_{i=1}^{n} \alpha_i y_i$$

Similarly, taking the partial of the Lagrangian with respect to $\mathbf{w}$ leads to:

$$\frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{\partial}{\partial \mathbf{w}} \left(\frac{1}{2} ||\mathbf{w}||^2 - \sum_{i=1}^{n} (\alpha_i y_i \mathbf{x}_i^T \mathbf{w} + \alpha_i y_i b - \alpha_i) \right) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

Setting both to 0 results in:

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{11}$$

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{12}$$

Finally, substituting (11) and (12) into the Lagrangian and taking into account we still need to maximize with respect to α, we arrive at the dual form of the optimization problem:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, 0 \leq \alpha_i \leq C. \end{aligned}$$

6.3 What is a kernel?

In the example in Section 6.1, each vector in $\mathbb{R}^2$ was explicitly mapped to its corresponding vector $\mathbb{R}^3$. This is not an issue in problems with smaller data sets like this example. However, when using extremely large data sets in high dimensions this is unrealistic in terms of both memory and computation time.

Kernels provide a way around this problem by providing a way to calculate the dot product of two vectors without explicitly mapping them to a specific higher dimension. Consider the dual formulation of the optimization problem. We only care about the dot product of the the training vectors, $\mathbf{x_i}$ and $\mathbf{x_j}$, not their respective mappings.

A kernel is a fuction K, such that for all $\mathbf{x}$, $\mathbf{z}$ ϵ X

$$K(\mathbf{x}, \mathbf{z}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{z})$$

where $\boldsymbol{\phi}$ is a mapping from X to an (inner product) feature space F.

In the example in Section 6.1, we define the kernel as $K(\mathbf{x_1}, \mathbf{x_2}) = \boldsymbol{\phi}(\boldsymbol{x_1})^T \boldsymbol{\phi}(\boldsymbol{x_2}) = (\mathbf{x_1}^T \mathbf{x_2})^2$. So instead of mapping every training vector to $\mathbb{R}^3$, we only have to square the dot product in $\mathbb{R}^2$.

6.4 Popular Kernels

Exploring and discovering new kernels is an area of research on its own. There are four popular kernels that are used.

- Linear:
$$K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T \mathbf{x_j}$$
- Polynomial:
$$K(\mathbf{x_i}, \mathbf{x_j}) = (\gamma \mathbf{x_i}^T \mathbf{x_j} + r)^d, \gamma > 0$$
- Radial Basis Function :
$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma ||\mathbf{x_i} - \mathbf{x_j}||^2), \gamma > 0$$
- Sigmoid:
$$K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\gamma \mathbf{x_i}^T \mathbf{x_j} + r)$$

The d, γ, r variables are all parameters the user inputs into the SVM. These variables are chosen based on the characteristics of the input data sets.

7 LIBSVM

LIBSVM is one of the popular Support Vector Machine tools. It was developed by Chih-Chung Chang and Chih-Jen Lin at the National Taiwan University. LIBSVM supports vector classification, vector regression, and one-class SVM. For this research, the vector classification component was utilized. Each of the four kernels described in Section 6.4 are implemented by LIBSVM as well as an option to input a user calculated kernel function.

7.1 Input and Output

LIBSVM requires user input of training and testing data sets. Each data set must be in its own .txt file where each line in the file represents one vector as follows: $< label >< index1 >:< value1 >< index2 >:< value2 >$ .... Each label is the y_i associated with the vector while each value is the i^{th} component of the vector. Any value of the vector that is 0 is not included in the .txt file.

After training the SVM, a model of the separating hyperplane is generated. This model output and the input of the testing vectors results in an output of an accuracy percentage, the number of support vector machines, the number of iterations of the SMO algorithm and other output particular to the SMO algorithm.

7.2 Sequential Minimal Optimization Algorithm

LIBSVM implements the SMO Algorithm in order to solve the dual optimization problem. The SMO algorithm was designed in order to solve the quadratic programming problem. This problem is characterized by an optimization of a quadratic function with multiple inputs that is constrained by linear restrictions. The dual optimization problem falls into the quadratic programming problem.

The SMO algorithm is iterative nature, focusing on the linear constraints, specifically the Lagrange multipliers. The main idea is that two Lagrange multipliers, α_i and α_j, are picked. Then the dual formulation is optimized with respect to α_i and α_j while the other Lagrange multipliers are held constant. This process is repeated until all the KKT conditions are satisfied.

8 Handwriting Recognition

8.1 Image to LIBSVM Format

Since LIBSVM takes .txt files as inputs, image files must be manipulated into .txt files. Each image utilized in this project was a 100x100 pixels. In order to represent each as a .txt file, each image was first transformed into a bitmap. This results in 10,000 bits for each image and a vector that has length 10,000. In order to decrease the dimension, the entry of each vector represents 8 bits

i.e. a number between 0 to 255. Thus each vector in the training and testing data sets have 1,250 entries.

In order to remove bias when determining which letters would be included into the training and testing data sets, I wrote a program to randomly choose an equal number of A's and B's for each data set. This program, as well as the one to transform the images to .txt files, was written in Python.

8.2 Results

When actually running LIBSVM, I used a total of six different data sets. The first and second data sets had 92 and 200 images respectively. This included lowercase and uppercase A's and B's that had a white or black background and were both cursive and print. The third and fourth data sets had 50 images each of lowercase and uppercase A's and B's with white and black background. The fifth data sets had 50 images of A's with white backgrounds and 50 images of B's with white backgrounds. The final data set had 25 images each of uppercase A's and B's with white backgrounds. Each of these data sets was run with the four different kernels supported by LIBSVM. The parameters for each kernel were determined by a cross validation tool supported by LIBSVM.

Figure 6: LIBSVM Results

Description	Kernel Type	Parameters	Accuracy
First 92	Linear	C = 0.03125	54.3%
First 92	**RBF**	**C = 0.03125 γ = 3.0517578125e-05**	**69.6%**
First 92	Polynomial	C = 0.03125 d = 3	54.3%
First 92	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	58.7%
All 200	Linear	C = 0.03125	43.0%
All 200	**RBF**	**C = 0.03125 γ = 3.0517578125e-05**	**61.0%**
All 200	Polynomial	C = 0.03125 d = 3	60.0%
All 200	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	52.0%
Lowercase	Linear	C = 0.03125	58.0%
Lowercase	RBF	C = 2.0 γ = 3.0517578125e-05	50.0%
Lowercase	**Polynomial**	**C = 0.03125 d = 5**	**68.0%**
Lowercase	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	56.0%
Uppercase	Linear	C = 0.03125	44.0%
Uppercase	**RBF**	**C = 0.03125 γ = 3.0517578125e-05**	**58.0%**
Uppercase	**Polynomial**	**C = 0.03125 d = 3**	**58.0%**
Uppercase	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	54.0%
White Background	Linear	C = 0.03125	62.5%
White Background	RBF	C = 0.03125 γ = 3.0517578125e-05	54.2%
White Background	**Polynomial**	**C = 0.03125 d = 5**	**64.6%**
White Background	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	50.0%
White, Uppercase	**Linear**	**C = 0.03125**	**70.8%**
White, Uppercase	**RBF**	**C = 0.03125 γ = 3.0517578125e-05**	**70.8%**
White, Uppercase	Polynomial	C = 0.03125 d = 3	66.7%
White, Uppercase	Sigmoid	C = 0.03125 γ = 3.0517578125e-05	50.0%

References

[1] Ben-Hur, A & Weston, J 2010, "A User's Guide to Support Vector Machines" , *Methods in Molecular Biology (Clifton, N.J.)*, 609, pp. 223-239, MEDLINE with Full Test, EBSCOhost

[2] Schlkopf, Bernhard, and Alexander J. Smola. *Learning With Kernels : Support Vector Machines, Regularization, Optimization, And Beyond*. n.p.: Cambridge, Mass. : MIT Press, c2002., 2002.

[3] Cristianini, Nello, and John Shawe-Taylor. *An Introduction To Support Vector Machines : And Other Kernel-Based Learning Methods*. n.p.: Cambridge ; New York : Cambridge University Press, 2000., 2000.

[4] Chang, CC, and CJ Lin. "LIBSVM: A Library For Support Vector Machines." *Acm Transactions On Intelligent Systems And Technology* 2.3 (2011): *Science Citation Index.*

[5] Hsu, Chih-Wei, Chang, CC and CJ Lin. "A Practical Guide to Support Vector Machines" National Taiwan University 2010.