Support Vector Machines for Classification

Alex Hostick
Dept. of Electrical and Computer Engineering
University of New Mexico
Albuquerque, New Mexico
ahostick@unm.edu

***Abstract*—Support Vector Machines (SVMs) are a powerful approach to classifying complex datasets. The algorithm can be used for data classification and has been relied on for industry since the 1990s. Implementing SVMs is effective in high dimensional spaces and builds certainty of boundaries between different data classes. We rely on Vapnik-Chervonenkis's theory to explain this process statistically. Algorithms produced by utilizing VC Theory create a practical methodology for solving multidimensional functions to transition abstract theoretical problems into a more generalized approach. In this paper, I will explore supervised linear machine learning SVM and how one can implement it with software.**

***Index Terms*–Support vector machines, VC dimensions, structural risk, Lagrange multipliers, KKT, classifiers, linear approach, machine learning, complexity**

I. Introduction

Statistical and transformative information of data systems has garnered the need for focused algorithms to classify and segregate diverse data dimensions. The introduction of a statistical method for classification and regression analysis through a Support Vector Machine (SVM) had been developed at AT&T Bell Laboratories by Vladimir Vapnik and colleagues [1]. SVMs are simple and elegant solutions for training a machine to learn and predict aspects of training data.

SVMs can classify data in a binary format for prediction and are used for linear and non-linear systems. To do this, SVMs can classify data into two categories or more in a binary manner. Data within a set of information can be defined in an N-dimensional space where the coordinates points of the data are defined as "features." Suppose a machine learning algorithm needs to classify whether the portrait image of a pet is a cat or a dog. In that case, features may include ear length and weight as two possible dimensional features for characterization. The training set given to the algorithm is labeled with the corresponding feature, asserting SVMs as supervised learning algorithms.

Hyperplanes are employed in an N-dimensional space to segregate the feature space's data points. The margin of the hyperplane seeks to find the maximum distance between the two sets of data points, separating the two categories most optimally. Any points that fall on the margin are named Support Vectors (SV). If two dimensions are used, a linear hyperplane can be established for the SVM algorithm. If three feature dimensions are needed, the hyperplane becomes a 2-D plane. Utilizing SVMs for the classification of datasets can be understood through statistical learning theory to understand unknown dependencies from known observations. We can explore the theory SVM approach through seven main techniques and criterion.

- Structural Risk and Empirical Risk
- Structural Risk Mitigation
- Vapnik-Chervonenkis (VC) Dimension
- VC Theorem and Bound on Actual Risk
- Support Vector Machine Criteria
- Dual Solution of the SVM and its Main Results
- Properties of Support Vectors

We will discuss the statistical learning theory for support vector machines for each category. The methods described will enhance the machine's ability to learn data over time and reduce the risk of misclassification. Though the information provided within this paper will cover linear SVM topics, non-linear data can still be classified by projecting it into a higher dimensional space and dividing it with the hyperplane, reducing the algorithm's complexity and cost to the machine. Keeping the statistical method simple will reduce the risk of overfitting.

Understanding the theoretical implications of statistical theory is paramount to implementing the theorems in software. This paper will also describe how the models fit into MATLAB models and how to implement an SVM for a linear machine learning example.

II. Structural Risk and Empirical Risk

Machine learning utilizes a set of functions to train the machine. The function $f(x,\alpha)$ uses alpha as the set of boundaries needed for training but also minimizes the expected risk. The machine is also assumed to be deterministic; changing α always generates the same output of $f(x,\alpha)$. The concept is illustrated in Fig. 1, with the unknown probability function F(x) and the conditional distribution function F(x|y), while x_n and y_n are data within the set.

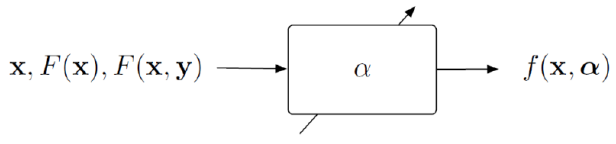


Fig. 1. Parametric Estimation Diagram

Adjusting the parameter relies on the problem to be solved. With $f(x,\alpha)$ estimating y, the estimation function $f(xn,\alpha)=yn+en$ yields e_n as the estimated error if α is inaccurate or there is an addition of erroneous data. Risk minimization criteria intend to minimize the convex function of the error, with $dF(x,y)$ representing the estimation of the true mean error of the data set. While 1 defines the expected loss, 2 designates the risk function.

$$E\{L\big((x,y)h\big)\} \tag{1}$$

$$R(\alpha) = \int_{x,y} L\big(y, f(x,a)\big)dF(x,y) \tag{2}$$

Truly, risk estimation is only possible as the true error is only available after testing and measuring the samples. This risk function must have a minimum to the point where a function of the error is minimized over all samples and state how far the model is from the actual value being detected or classified. The loss function L(·) represents the expectation of the error function with respect to x and y. Utilizing this loss function over another depends on the problem. After testing the model, one can evaluate the empirical risk to understand how accurate and effective the employed estimated risk function had been.

For empirical risk, R_{emp} (α) is the measured mean error rate on a fixed, finite number of observations embodying the training set. Empirical risk approximates the actual risk or how well the machine will perform with the data set. We see in 3 the empirical risk function.

$$R_{emp}(\alpha) = \sum_{n=1}^{N} L(y_n, f(x_n, \alpha)) \tag{3}$$

A model that can predict data points with negligible error is ideal. Minimizing the risk trains the machine to utilize accurate prediction models by estimating the risk by the empirical risk.

III. Complexity and Overfitting

Minimizing the risk can lead to better accuracy of a trained model; however, doing so can lead to over-fitting if the estimation function is too complex. Limiting the overall complexity of the machine with optimization techniques leads to Structural Risk Minimization (SRM). Models with overfitting can estimate the training data or the original data set used to train the machine but are inaccurate with any new data presented. Data that is too small, too large with irrelevant data, or machines that train too long on a single data set can suffer from overfitting.

Besides optimization and "lower"-complexity models, different methods are employed to reduce overfitting. In 2023, Amazon compiled a list of techniques for minimizing overfitting [2], denoted in

Table ***I***.

TABLE I
STRATEGIES FOR REDUCING OVERFITTING

Strategy	Description
Early Stopping	Pause the training phase before the machine learning model learns the noise in the data.
Pruning	Identifying the most important features within the training set and eliminating the irrelevant ones.
Regularization	Collection of training/optimization techniques for reducing overfitting. Eliminate factors that do not impact of predict outcomes by grading features based on importance.
Ensembling	Combines predictions from several machine learning algorithms. Combine weaker methods to gain more accurate results and pick the most accurate outcome.
Data Augmentation	Change the sample data slightly every time the model is processed to make the data appear unique on each iteration.

Understanding how to minimize the estimation complexity can be put into practice by examining the ridge regression function. Ridge regression is used to analyze data with multicollinearity with L2 regularization. When the issue of multicollinearity occurs, least squares are unbiased, variances are significant, and the results in predicted value can be far away from the actual values [3]. The number of input variables can significantly exceed the number of observations. A geometric example of ridge regression, illustrated in Fig. 2, with the residual sum of squares (RSS), which utilizes an ellipse with a smaller RSS to minimize ordinal least square (OLS) estimates.

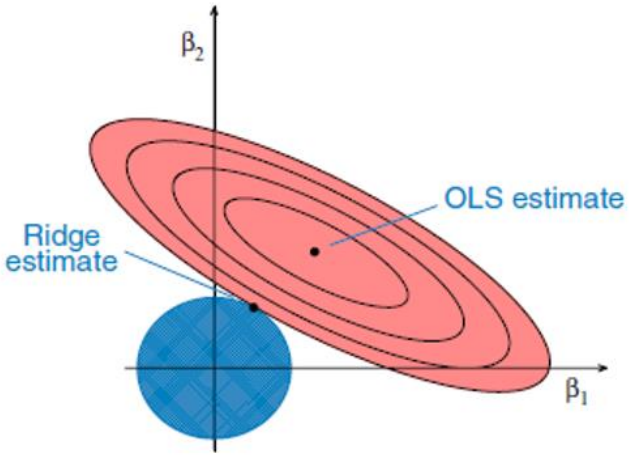


Fig. 2. Geometric Ridge Regression

A common ridge regression method is denoted in 4, a function for signal processing as a form of regularized loss function.

$$E(e^2) + \lambda \cdot \|w\|^2 \tag{4}$$

In this equation, E(e^2) denotes the sum of the squared errors in the predicted and the true outputs of the model $\sum_{n=1}^{N}(y_i - \hat{y}_i)$ estimation function. The $\lambda \cdot \|w\|^2$ is a regularization to penalize the complexity of the model, with λ as the regularization of the model. Minimizing the vector w can reduce the complexity of the machine and leave the bias $w^\top x + b$ line function.

III. Vapnik-Chervonenkis (VC) Dimensions

Vapnik-Chervonenkis (VC) theory is a computational learning theory explaining the machine learning process from a statistical viewpoint. VC theory provides generalization conditions for learning algorithms. When discussing VC dimensions, the term measures the model's capacity. This notion will help us understand how well its complexity can fit different data sets. Consider Fig. 3 with a set of real numbers $\mathbb{R}^2$. The hyperplane can linearly shatter the three possible data points if the other two are linearly separated.

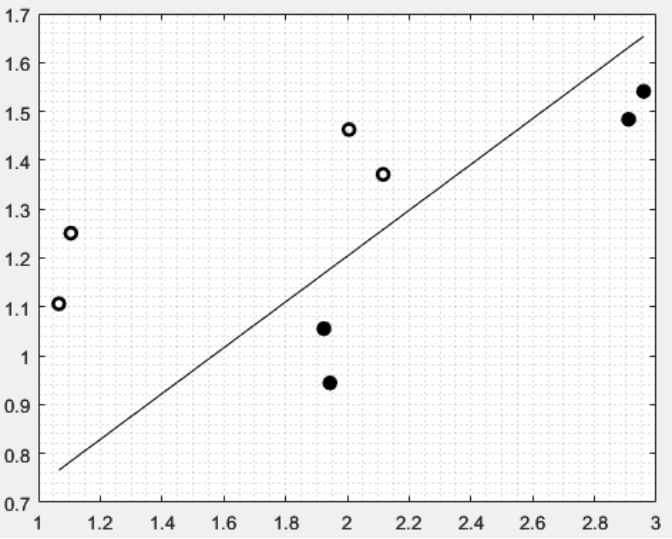


Fig. 3. Vectors Shattered by Hyperplane

If a fourth point is added, the original linear function cannot be achieved. The maximum number of vectors shattered by a hyperplane (h) in a space $\mathbb{R}^n$ is the VC dimension $h = n + 1$. This notion states we will always need one more dimension than n points to keep the vectors linearly independent. Of course, more complexity included in the VC Dimension can increase the overfitting risk. If the VC estimator exceeds the number of vectors that need classification, the estimator will overfit.

Exemplifying the linear empirical risk function is minimized to find the regression estimate with the least square method [4]. The empirical risk function in this context, noted in 5, where $f(\cdot)$ is defined so that the loss function $|y - f(x, a)|$ can only take the values 0 or 1.

$$R_{emp}(\alpha) = \frac{1}{2N}\sum_{n=1}^{N}|y - f(x, a)| \tag{5}$$

This notion leads to structural risk (R_s), where structural risk minimization consists of choosing a machine whose dimension is sufficiently small so the bound on the risk minimizes. We use 6 for defining the bound with the structural risk probability 1 – η regarding the second term on the right side.

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}} \tag{6}$$

Empirical risk leads to zero if VC dimensions are too high, increasing N. The h in Equation 5 is unknown but will increase the empirical risk as h increases. Vapnik's theorem is used for multiple classes of machines but suits linear machines as the VC dimensions can be computed and minimized.

IV. VC Theorem and Bound on Actual Risk

VC Theorem incorporates the VC dimension of a separating hyperplane minimized if the norm of its parameters minimizes. With the linear estimation function $f(x) = sign(w^\top x + b)$, the normal vector w can limit the VC dimension of the hyperplane. The VC dimension only separates the patterns xn according to their labels and maximizes the margin (d) between the two classes, as illustrated in Fig. 4.

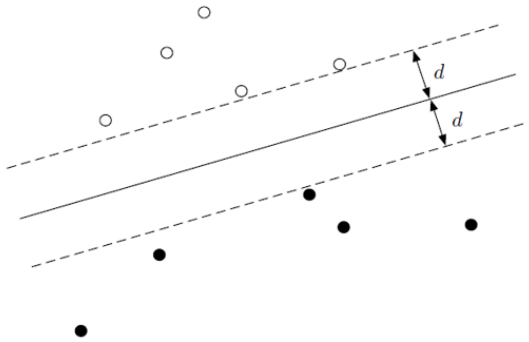


Fig. 4. Margin Between Two Clusters of Data

Setting bounds clarifies the performance capabilities and upper limits on the errors in the learning algorithm. We concluded in 5 that with high probability, actual risk R(α) is bounded by the empirical risk Remp. The bound does not depend on the distribution of data or probability distribution. The probability risk 1 – η is the bound for Eq. 5. Although we do not know the value of h, we can minimize h so that a machine has its bound on the risk minimized. Cross-validation, or the technique used to evaluate the performance of unseen data,

divides the available data into multiple folds or subsets. One of the folds is used as a validation set, training the model on the remaining folds, and repeats numerous times. The average provides a robust estimate and reduces the actual risk. For Eq. 5, with the bound probability risk 1 – η, we can reduce the actual risk by the Principle of Structural Risk Minimization. The bound on the risk is illustrated in Fig. 5, with N = 100 and η = 10^{-3}.

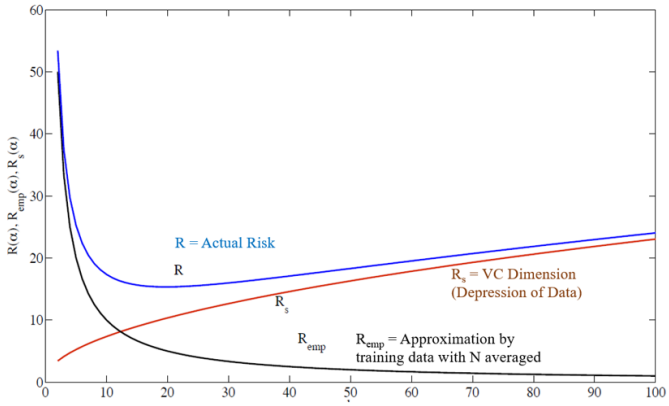


Fig. 5. Structured Risk vs Empirical Risk

We can see that with the VC dimension increasing, the complexity increases. The actual risk is rising due to the complexity; thus, errors and overfitting may occur. Having a minimal h value can approximate the empirical risk to the actual risk, lowering the complexity of the machine. Cross-validation, with averaging N, will assist with choosing a small value for h as more data sets are used in the training method.

V. Support Vector Machine Criteria

By separating the closest patterns by a plane, the machine can adequately classify them. If the margins are defined as functions $w^\top x + b = \pm 1$, we can see the separation from the hyperplane with w as a vector normal to the hyperplane in Fig. 6.

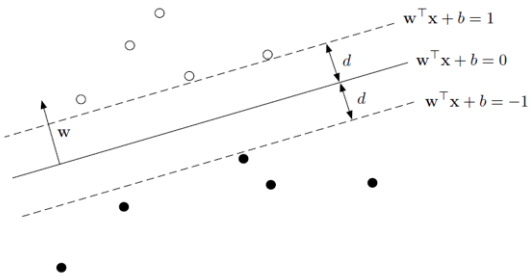


Fig. 6. Margins Defined for Clusters of Data

To find the margin distance d, we can define a point x_0 such that $w^\top x_0 + b = 0$, and then trace a line perpendicular to the plane with $x = x_0 + \rho w$ as illustrated in Fig. 7. The intersection with the upper margin for x_1 yields the distance of $d = \|x_1 - x_0\| = \rho\|w\|$.

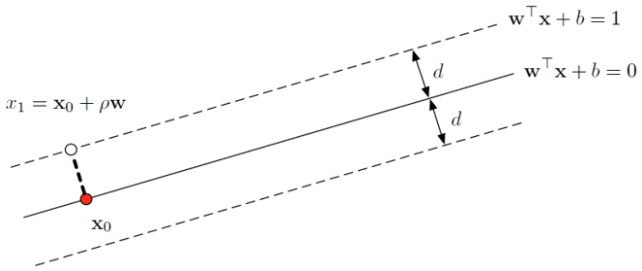


Fig. 7. Margin Distance (d) Differential Between X_1 and X_0

As we find the distance to the margin, we can maximize the term by minimizing the norm of the parameter vector *w,* subject to the constraint of correctly classifying all samples. We defined x0 on the hyperplane and found $w^\top x_0 + b = 0$. The maximum of the margin distance d is equivalent to $d = \rho\|w\| \xrightarrow{yields} \frac{1}{w}$. If we minimize w, then the following constructs the machine with the minimum possible complexity by restricting the possible shatters of the plane to 1, as shown in 7. We can use this reduction in complexity and decrease in possible shatters when defining the criteria for SVM.

$$\begin{aligned} &minimize\ \|w\|^2 \\ &\text{Subject to } y_n(w^\top x_n + b) > 1 \end{aligned} \tag{7}$$

The SVM criterion consists of defining an empirical risk and minimizing it together with maximizing the margin in a function. Understanding how to classify each data point accordingly will yield a better model. For example, if we have a set of data with the functional $y_n = (w^\top x_n + b) > 1 + \xi_n$l, with ξ_n as the slack variable of $\xi_n \geq 0$, we can also minimize the empirical and structural risks through margin maximization. This depreciation is shown in 8, where C is a free tradeoff parameter and N are the primal variables, we are trying to find the best model for. Lagrange multipliers are used to change the constrained problem into an unconstrained one as the slack variable is added to the function to transform its inequality expression into an equality.

$$\begin{aligned} &L_p(w, \xi_n) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \\ &\text{Subject to } \begin{cases} y_n(w^\top x_n + b) > 1 - \xi_n \\ \xi_n \geq 0 \end{cases} \end{aligned} \tag{8}$$

SVM criterion includes the basic methodology of finding the hyperplane that decreases the empirical risk (training error) and structural risk (VC dimension) while maximizing the distance to the margin. The machine has low complexity if the margin is large. Adversely, if the margin is not large, the machine may be overfitted with a maximum VC dimension due to its complexity.

VI. Dual Solution of the SVM and its Main Results

The Support Vector Machine minimizes the previous empirical risk and structural risks through margin maximization, as discussed in the SVM criterion—the primal function expressed in 8 utilized Lagrange multipliers to change the constrained problem into an unconstrained problem. Using Lagrange Optimization minimizes the optimal point at the moment where two gradients will be proportional. In Fig. 8, we see that using the minimization with the constraints $F(w)$ with the subject to gradient $g(w) = 0$ yields the optimal point where both gradients are proportional.

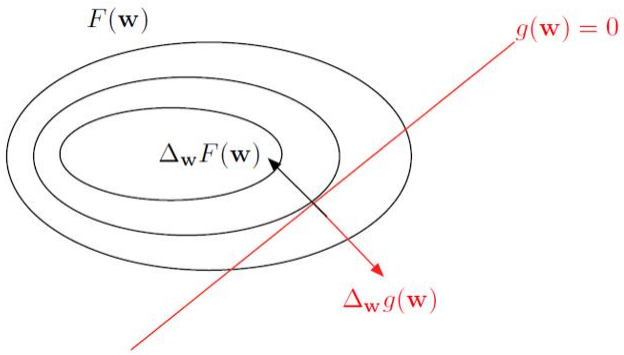


Fig. 8. Optimal Point where Both Gradients Proportional

The functional $L_{Lagrange} = F(w) - \alpha g(w)$ where $\alpha \geq 0$ is a Lagrange multiplier and is known as the dual variable. Optimization consists of computing the gradient w.r.t. with the primal variables *w* and nulling it as expressed in 9. This will lead to the Karush Kuhn Tucker (KKT) conditions.

$$\Delta_w F(w) - \alpha g(w) = 0 \tag{9}$$

Optimizing by computing the gradient will also yield the Lagrangian, as shown in 10.

$$\begin{aligned} L_L(w, \xi_n, \alpha_n, \mu_n) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n \\ - \sum_{n=1}^{N} \alpha_n (y_n (w^\top x_n + b) - 1 \\ + \xi_n) - \sum_{n=1}^{N} \mu_n \xi_n \end{aligned} \tag{10}$$

By nulling the gradient in respect to w, we find $\Delta_w L_L(w, \xi_n, \alpha_n, \mu_n) = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0$. The equation can be reduced to $w = \sum_{n=1}^{N} \alpha_n y_n x_n$, or the matrix notation $w = XY\alpha^\top$, where Y is a diagonal matrix containing all the labels and α includes all the multipliers. After minimizing the Lagrangian functional, we can null the derivative *w.r.t.*, the slack variable ξ_n, and b, as expressed 11 and 12.

$$\frac{\delta}{\delta \xi_n} L_p(w, \xi_n, \alpha_n, \mu_n) = C - \alpha_n - \mu_n = 0 \tag{11}$$

$$\frac{d}{db} L_p(w, \xi_n, \alpha_n, \mu_n) = \sum_{n=1}^{N} \alpha_n y_n = 0 \tag{12}$$

We must also force the complementarity property over the constraints with 13:

$$\begin{aligned} \mu_n \xi_n = 0 \\ \alpha_n (y_n (w^\top x_n + b) - 1 + \xi_n) = 0 \end{aligned} \tag{13}$$

Nulling these properties of the Lagrangian function will yield the KKT conditions. A summary of the KKT conditions is available in Table II.

Table II
The KKT Conditions

No.	Condition
1	$w = \sum_{n=1}^{N} \alpha_n y_n x_n$
2	$C - \alpha_n - \mu_n = 0$
3	$\sum_{n=1}^{N} \alpha_n y_n = 0$
4	$\mu_n \xi_n = 0$
5	$\alpha_n (y_n (w^\top x_n + b) - 1 + \xi_n) = 0$
6	$\alpha_n \geq 0, \mu_n \geq 0, \xi_n \geq 0$

The KKT conditions will yield results respective to the samples. For clarity, the support vector will either be inside, on, or outside the margin based on the KKT conditions. Table III clarifies conditions 2, 4, and 5 based on the location of the sample.

Table III
Sample Constraints

Condition	Classification
$\xi_n \geq 0, \alpha_n = C$ (2)(4)	Sample is inside the margin or misclassified, thus $\alpha_n = C$
$0 < \alpha_n < C$ (5)	Sample is on the margin.
$\xi_n = 0, \alpha_n = 0$ (5)	Sample is classified and outside the margin.

Finally, we can write the estimator function. The estimator function is a dual expression of the classifier, as noted in 14. The primal SVM functional expression leads to the construction of the Lagrange functional.

$$y_k = \sum_{n=1}^{N} y_n \alpha_n x_n{}^\top x_k + b = a_n{}^\top Y X^\top x_k + b \tag{14}$$

Finding the *duality* of the system allows for optimization from two perspectives. The solution to the dual problem provides a lower bound to the solution to the primal minimization problem. We want to minimize the primal problem, while the dual problem is something we want to maximize. In finding the solution to a linear model, the primal problem is ideal, whereas the dual problem is helpful for non-linear solutions. Using the Lagrange Multiplier, we determined when we could find a strong duality. The SVM is a linear machine whose criterion is to minimize the primal.

Finding the dual problem solution will be the next step. Using the estimator function from 14 and the Lagrangian Equation, we can see the terms noted in Table IV. We can use KKT parameters to minimize the function as indicated in Table 4, and we reduce the Lagrangian Equation to 15 and its matrix notation to 16. Note that with the dual problem's matrix notation, $X^\top X$ is a Gram matrix of dot products, with $K_{i,j} = x_i^\top x_j$ and the dual problem is written in 17.

TABLE IV
LAGRANGIAN TERMS

Term	Equation
A	$\frac{1}{2}\|w\|^2 = \sum_{n=1}^{N}\sum_{n'=1}^{N} y_{n'}\,\alpha_{n'} x_{n'}{}^\top x_n\,\alpha_n y_n$
B	$-\sum_{n=1}^{N}\sum_{n'=1}^{N} y_{n'}\,\alpha_{n'} x_{n'}{}^\top x_n\,\alpha_n y_n$
	$-\sum_{n=1}^{N} a_n y_n b + \sum_{n=1}^{N} a_n$
	$-\sum_{n'=1}^{N} a_n \xi_n$
C	$-\sum_{n=1}^{N} \mu_n \xi_n$
D	$C\sum_{n=1}^{N} \xi_n$

$$L_d = -\frac{1}{2}\sum_{n=1}^{N}\sum_{n'=1}^{N} y_n\,\alpha_n x_n{}^\top x_{n'}\,\alpha_{n'} y_{n'} + \sum_{n=1}^{N}\alpha_n \qquad (15)$$

$$L_d = -\frac{1}{2}\alpha^\top Y X^\top X Y \alpha + \alpha^\top 1 \qquad (16)$$
$$With\ the\ constraint\ of\ \alpha \geq 0$$

$$L_d = -\frac{1}{2}\alpha^\top Y K Y \alpha^\top + \alpha 1 \qquad (17)$$
$$\alpha^\top Y K Y \alpha > 0$$

VII. PROPERTIES OF SUPPORT VECTORS

SVMs will train their models based on finding decision surfaces determined by points in the training set termed support vectors (SVs). Typically, the SVs are obtained using the solution of quadratic programming using regularization parameters. The objective is to find a hyperplane in an N-dimensional space that classifies the data points. Hyperplanes are decision boundaries that help classify data points. Points on either side of the hyperplane can be attributed to different classes, and the dimension of the hyperplane depends upon the number of features.

Properties of the support vectors are expressed according to where they are positioned around the margin. As illustrated in Fig. 9, support vectors are defined by their Lagrangian position to the margin. The summary of their classifications is noted in Table V.

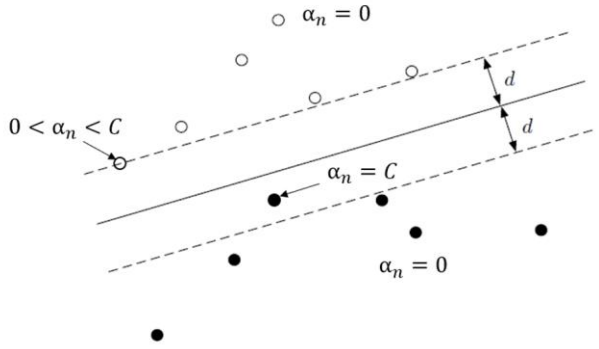


Fig. 9. Point Lagrange Parameter Based on Location to Margin

TABLE V
SUPPORT VECTOR CLASSIFICATION

Expression	Condition
$a_n = 0$	Data well classified outside of the margin.
$a_n = C$	Data inside the margin (or misclassified)
$0 < a_n < C$	Data on the margin

As discussed in this paper, we want to maximize the margins with respect to a_n and minimize the error with respect to w. Support vectors are the most critical points in determining the optimal decision boundary (hyperplane) that separates the classes. The support vectors are the points closest to the hyperplane and margins and can be the most difficult to classify. Determining the support vector positions will decide the decision boundary for model predictions.

VIII. SOFTWARE EXPERIMENTS

We can generate a set of data to train a Support Vector Machine. Data generated will need to contain labels for

classification parallel to using features to describe the data. The SVM can then classify the data into two classifications and train on segregating the data's aspects. Training the data can be done with the LIBSVM functional library developed by Chih-Chung Chang and Chih-Jen Lin [5]. The LIBSVM library is used within common programming languages such as MATLAB, Java, Python, Ruby, and others. The LIBSVM library can be used with CUDA and Cell accelerated processor technology.

I will use the MATLAB LIBSVM library in this experiment to express how machine learning can be simulated and simplified with a software approach. The approach will consist of a linear, two-dimensional example.

A. Data and SVM Training

Data for the software approach can be delivered to the machine learning process in a variety of inputs. Files containing structured data or real-time inputs, with a maximum of two features, are commonly used as inputs to the SVM. This experiment will produce 10 sample points (N=10) around four centroids (*c*). Changing a sigma variable will update the set's normal distribution around the centroid with each iteration of the code. The sample MATLAB code is:

```
N=10; % number of samples
c=[1,1;2,1.5;2,1;3,1.5]; %centroids
sigma=0.1; % Gaussian sigma of sample
set

sigma=0.1;
for i=1:4
      X=[X;sigma*randn(N,2)+repmat(c(i,
:),N,1)];
end

Y = [ones(1,2*N) -ones(1,2*N)]';

figure('Name', 'Data')
plot(X(1:end/2,1),X(1:end/2,2),'ok',
'LineWidth', 2)
hold on

plot(X(end/2+1:end,1),X(end/2+1:end,2),
'ok', 'LineWidth', 2,
'MarkerFaceColor', 'k')
```

This code produces two data clusters as illustrated in Fig. 10. As noted, 10 data points are spread around the centroid points defined in variable *c*.

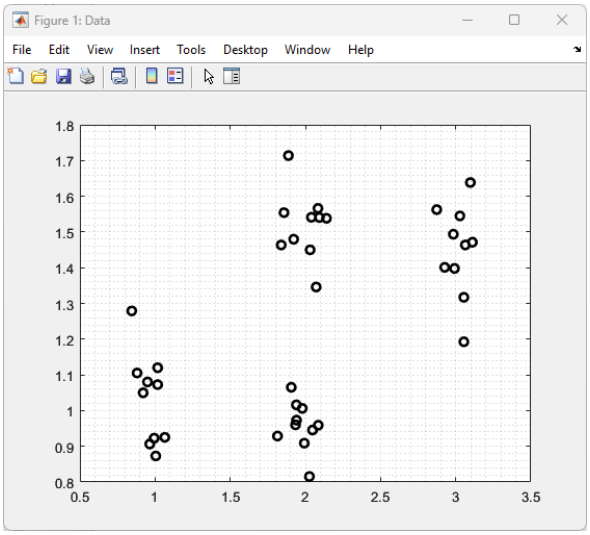


Fig. 10. Training Data for SVM Classification

We can now train the SVM by utilizing the LIBSVM functions provided in its library. Doing so is as simple as using the *svmtrain* function provided in the library:

```
%train the SVM
model = svmtrain(Y,X,'-s 0 -t 0 -c
100');
```

Where:

- -s svm_type
  - '0' is C-SVC
- -t kernal_type
  - '0' is linear: u'*v
- -c cost
  - Sets the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)

With the *svmtrain* function, we can analyze the results of the data with the library's output as denoted in Fig. 11. This output notes how the linear classifier will train with the provided data. Changing the value C within the *svmtrain* function will determine the capacity of the machine and the complexity of data it can decipher.

```
optimization finished, #iter = 71
nu = 0.033057
obj = -66.133582, rho = 1.871725
nSV = 3, nBSV = 0
Total nSV = 3
```

Fig. 11. *svmtrain* MATLAB Output Parameters

B. Probability Estimation

Along with training the data, we can use the library to output its probability of correctly classifying the data. The *svmtrain*

function trains the machine to separate data points with one class of information on one side versus the other. This is the decision boundary the machine applies to minimize loss with the maximum amount of margin. Due to the SVM not inherently predicting its training accuracy, we can use the *svmpredict* function to output the machine's probability of accurate classification.

To start, we must compute the primal variable *w* and the bias *b*. This is one of the dual computations discussed within this paper:

```
% Compute Primal w (weight)
W = (model.sv_coef' * full(model.SVs));

%negative rho is the bias(b)
bias = -model.rho;
```

We then use the bias and the primal variable for predicting the accuracy of the model used in classification:

```
classifier = sign(X * W' + bias);

fprintf("Prediction: ");
[labels, precision, vals] =
svmpredict(Y,X,model);
```

This yields a probability output within the MATLAB terminal as denoted in Fig. 12:

```
optimization finished, #iter = 71
nu = 0.033057
obj = -66.133582, rho = 1.871725
nSV = 3, nBSV = 0
Total nSV = 3
Prediction: Accuracy = 100% (40/40) (classification)
```

Fig. 12. Prediction Accuracy Added to MATLAB Terminal

C. Graphical Expression of the SVM

We can graph the SVM's hyperplane used for the training data. Graphing the data will visualize the machine's *svmtrain* function as it aims to classify the data points. The expressions can use the primal variable w and the bias b to define the hyperplane and margins as noted in Fig. 4. To define the hyperplane, we can use the MATLAB code:

```
%Hyperplane: y = w'*x+b = 0
input = linspace(min(X(:,1)),
max(X(:,1)),100);
output_h = - (W(1) * input +
bias)/W(2);
plot(input, output_h, 'm')
hold on
```

And the first margin as:

```
%Margin1: y = w'*x+b = 1
Margin1 = - (W(1)*input + bias)/W(2) +
1/W(2);
plot(input, Margin1, '--k',
color='red');
hold on
```

And the second margin as:

```
%Margin2 = w'*x+b = -1
Margin2 = - (W(1)*input + bias)/W(2) -
1/W(2);
plot(input, Margin2, '--k',
color='blue');
```

These functions will plot the hyperplane and its margins. The data is separated by color and shape; thus, red and blue are chosen to graphically represent the two classes of data as illustrated in Fig. 13. Support Vector classification is also noted.

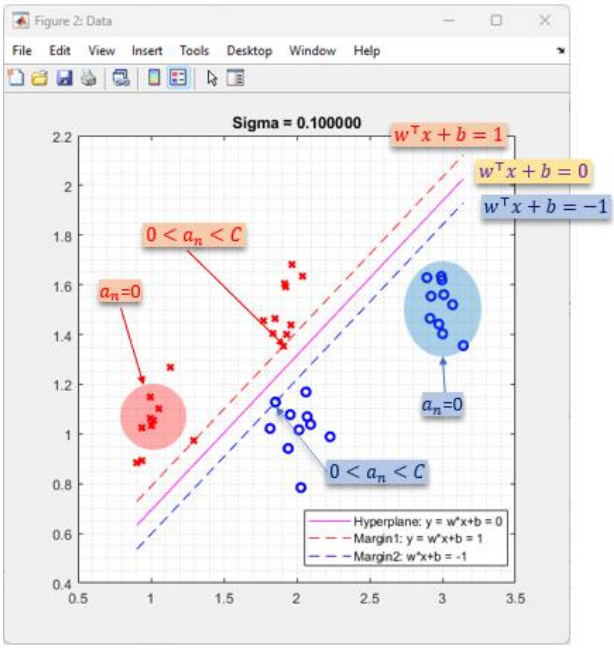


Fig. 13. Classification with Sigma = 0.1

Fig. 13 has support vectors and classified data outside the margins. We can increase the sigma value of the N points among the centroids to see data within the margins, as noted in Fig. 14.

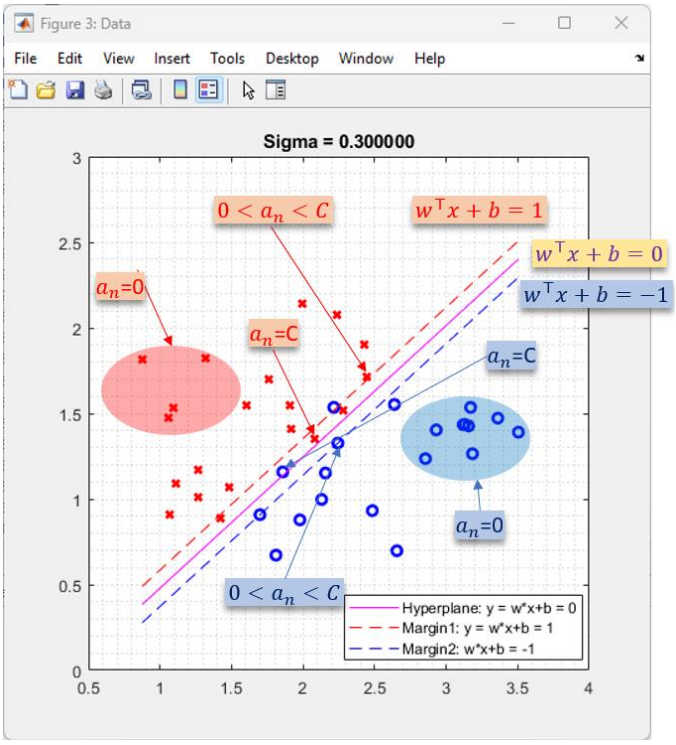
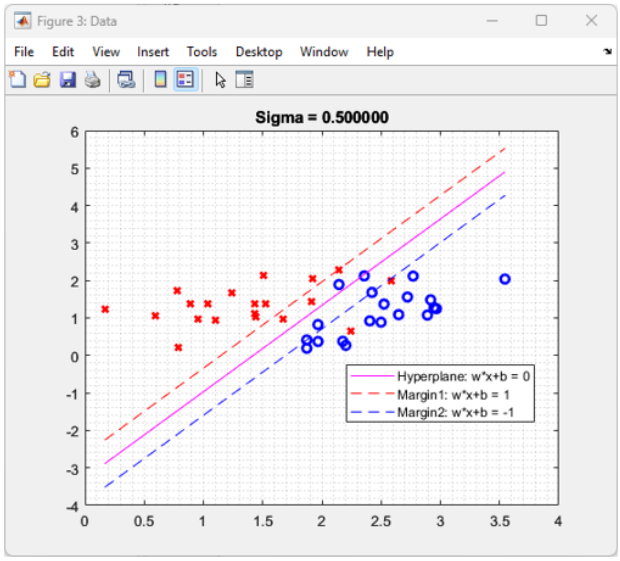


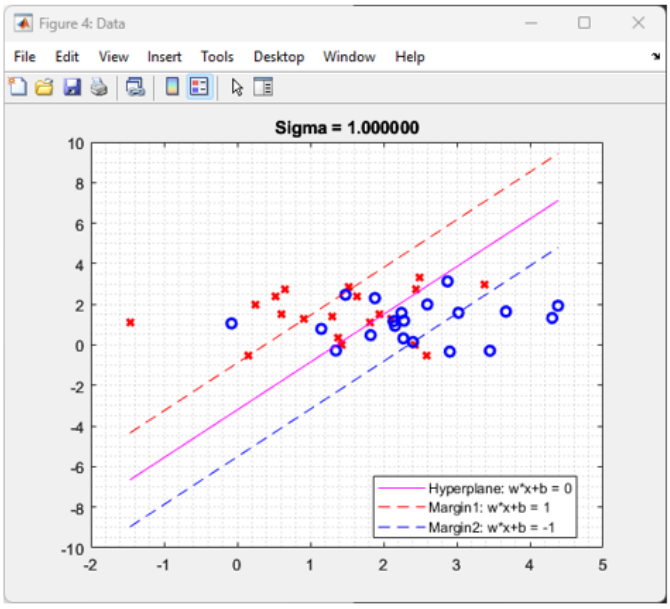
Fig. 14. Classification with Sigma = 0.3

Various changes in sigma will introduce more variance 'noise' to the generated data. We can see how well the *svmtrain* and *svmpredict* algorithms can classify and characterize the data different sigma values applied in the following figures. Note that the prediction accuracy does decrease with noise added to the system.



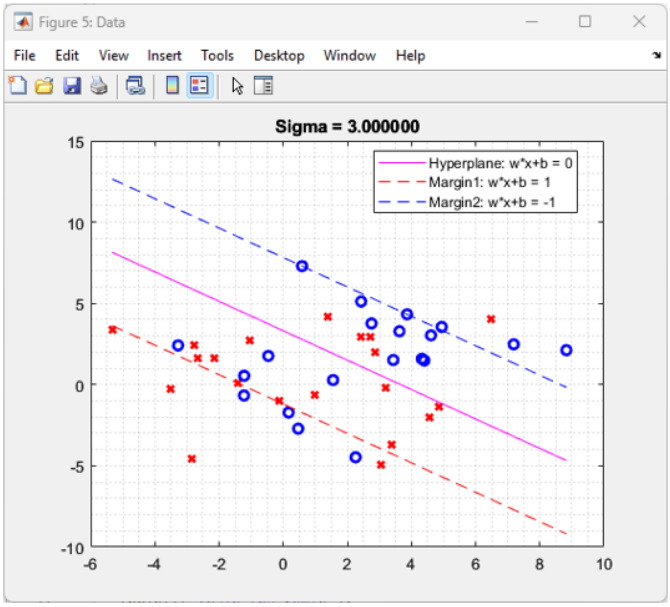

```
optimization finished, #iter = 478
nu = 0.217566
obj = -862.270332, rho = -5.203626
nSV = 10, nBSV = 7
Total nSV = 10
Prediction: Accuracy = 92.5% (37/40) (classification)
```

Fig. 15. Sigma = 0.5




```
optimization finished, #iter = 8179
nu = 0.686638
obj = -2745.896979, rho = -1.386627
nSV = 28, nBSV = 25
Total nSV = 28
Prediction: Accuracy = 77.5% (31/40) (classification)
```

Fig. 16. Sigma = 1.0




```
optimization finished, #iter = 64602
nu = 0.753733
obj = -3014.913798, rho = -0.734643
nSV = 33, nBSV = 30
Total nSV = 33
Prediction: Accuracy = 67.5% (27/40) (classification)
```

Fig. 17. Sigma = 3.0

Fig. 7 noted that if we minimize w, then the following constructs the machine with the minimum possible complexity by restricting the possible shatters of the plane to 1. Distance to the margin increases as the normal vector w decreases due to $d = \rho\|w\| \xrightarrow{yields} \frac{1}{w}$.

Why would we want to increase sigma for the experiment? Increasing sigma minimizes the VC dimension and the inherent risk in contrast to lower model prediction accuracy. Finding the balance of the sigma variable for support vectors is paramount for accuracy and correct classification.

IX. Estimating Structural Risk

Minimizing risk is imperative for SVM modeling and implementation. Models trained on data must consider the variance of data, especially input derived from real-world samples. There will be a tradeoff between the VC complexity dimension and the fitting training data quality. Utilizing the LIBSVM library, we can use the *svmtrain* and *svmpredict* functions to estimate the structural and actual risks with the difference of empirical risk. Consider Fig. 5 and how criterion for structural risk minimization can be applied when presented with data. We can use Fig. 18 as an example of how to write the experiment's MATLAB code.

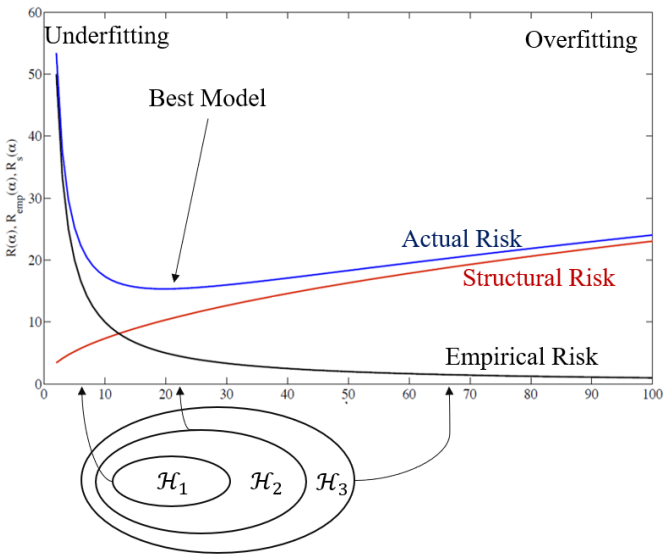


Fig. 18. Structural Risk Minimization

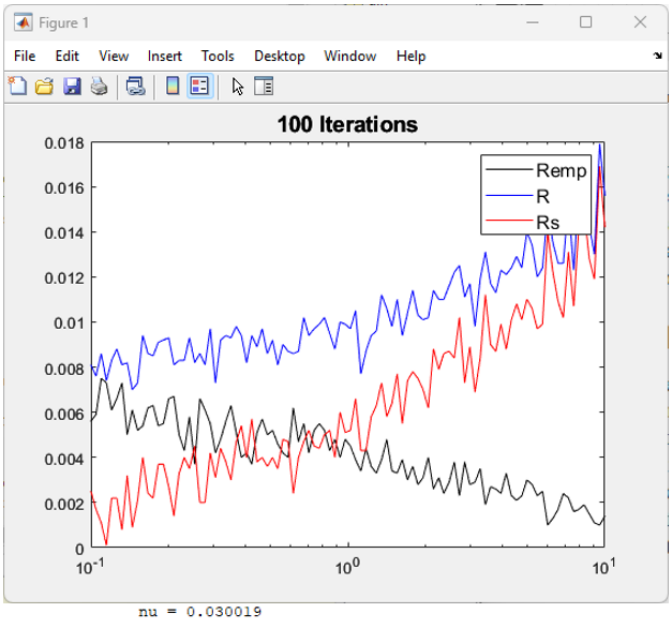
Again, we need to generate values for training data. Because we want to simulate real-world data, the value generated should be different each time the experiment is run. We can generate 100 logarithmically spaced points between 10^{-1} and 10^{1} with the MATLAB function:

```
c = logspace(-1,1,100);
```

We will also need to iterate over a defined number of data sets that differentiate each time the code is run. The SVM data function accomplishes this by adding *svmtrain* and *svmpredict* to train the model with each iteration. The updated MATLAB code is as follows:

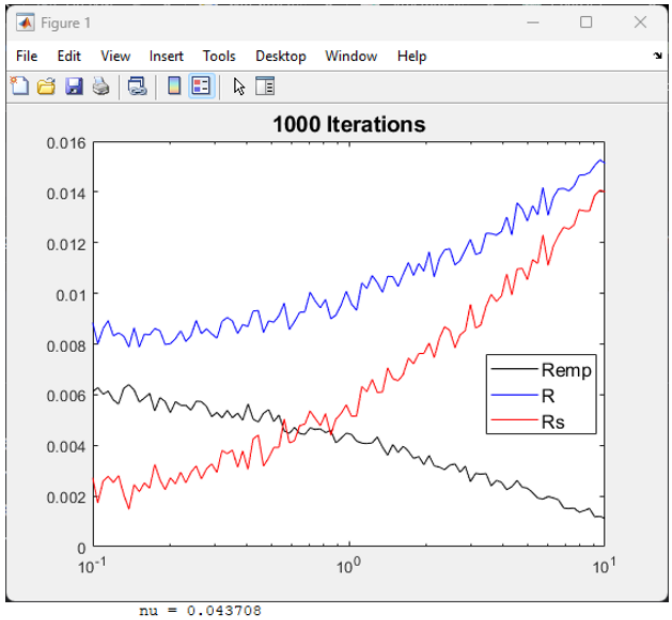
```
% c = 100 data points
c = logspace(-1,1,100);
% Matrix to store the Empirical Risk
Remp = zeros(1000,length(c));
% Matrix to store the Test risk
R = Remp;
sigma = 1;
iterations = 10;

% Iterate over number desired
for i = 1:iterations
    for j = 1:length(c)
         [X,Y] = SVMdata_func(100,1);

% SVM options
options = ['-s 0 -t 0 -c '
num2str(c(j))];

model = svmtrain(Y,X,options);

[~,precision,~] = svmpredict(Y,X,model);

% Remp
Remp(i,j) = 1-precision(1)/100;

% Produce test data
[X,Y] = SVMdata_func(100,sigma);

[~,precision,~]  =
svmpredict(Y,X,model);

R(i,j) = 1-precision(1)/100; % Risk(R)
end
```

If implemented correctly, the SVM will compute its actual risk (R), empirical risk (R_{emp}), and structural risk (R_s) as noted in Fig. 19. Sigma and number of iterations can be changed to differentiate how the risk will react to addition of noise and time, respectively.




```
nu = 0.030019
obj = -15.009841, rho = -0.576634
nSV = 10, nBSV = 0
Total nSV = 10
Accuracy = 100% (100/100) (classification)
Accuracy = 97% (97/100) (classification)
Elapsed time is 4.890043 seconds.
```

Fig. 19. Risk Estimation with Sigma = 1, i = 100




```
nu = 0.043708
obj = -24.375157, rho = 0.151133
nSV = 13, nBSV = 1
Total nSV = 13
Accuracy = 100% (100/100) (classification)
Accuracy = 99% (99/100) (classification)
Elapsed time is 46.253846 seconds.
```

Fig. 20. Risk Estimation with Sigma = 1, i = 1000

As noted in Fig. 20, the longer the machine can train on the data, the more the empirical training error decreases. While the training error decreased, the time taken for the model increased by about 41 seconds. This is an example of the tradeoff of the complexity of the machine as the increased training only marginally increased the accuracy of the machine from 97% to 99%.

XI. Conclusion

In this paper, we overviewed how Support Vector Machines are a simple and elegant solution for machine learning. SVMs are used with supervised statistical learning algorithms and work best with finite training data sets. Additionally, lowering the complexity of the SVM is a paradigm for efficient training and greater classification accuracy. Maximizing the SVM margins, or the distance between the margins and the hyperplane, will define the most significant classification distance between two data sets. As the normal vector w decreases, so do we increase the distance of the margins. Support vectors were found using statistical methods using Lagrange multipliers. We can then start to understand which data is well-classified or misclassified depending on its feature and how well it fits into the margins. Large margins decrease the complexity of the machine, while small margins may lead to overfitting. Several examples on how to reduce overfitting were provided. Using this information and graphically plotting data, we can choose better optimization parameters for the model.

References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the fifth annual workshop on Computational learning theory*, p. 144, 1992. doi:10.1145/130385.130401

[2] "What is Overfitting?," Amazon, https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine,on%20a%20known%20data%20set (accessed Oct. 26, 2023).

[3] P. Ashok, "What is ridge regression?," Great Learning Blog: Free Resources what Matters to shape your Career!, https://www.mygreatlearning.com/blog/what-is-ridge-regression/#:~:text=Ridge%20regression%20is%20a%20model,away%20from%20the%20actual%20values (accessed Oct. 26, 2023).

[4] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, Sep. 1999. doi:10.1109/72.788640

[5] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A Practical Guide to Support Vector Classification." National Taiwan University, Department of Computer Science, Taipei 106, Taiwan, May. 19, 2016