Support Vector Machines: Introduction, Implementation and Application

Seminar Data Mining

Chengjie "Jay" Zhou
School of Computation, Information and Technology
Technische Universität München
Email: chengjie.zhou@mytum.de

***Abstract*— This paper provides an overview of the support vector machine (SVM) algorithm and its applications. SVM is a machine learning algorithm that is widely used for classification and pattern recognition problems. In this paper, a brief introduction to the motive of SVM is given. Then, the mathematical background and definitions of various concepts within the SVM algorithm are stated. Later on, we discuss the implementation of SVM in various applications, including image classification, bioinformatics, and so on. Finally, we conclude the paper with a summary of the advantages and disadvantages of SVM.**

***Keywords*— Data Mining, Support Vector Machines, Maximal Margin Classifier, Support Vector Classifier, Hyperplane, Kernel, SVM in Practice**

I. Introduction

One of the most important aspects of machine learning is classification. Typical algorithms and models that are used for classification include logistic regression, Naive Bayes, decision trees, and so on. In the 90s, Vladimir Vapnik and his colleagues developed a new algorithm called *Support Vector Machine (SVM)*. The name of this algorithm comes from the fact that the algorithm is based on the concept of *support vectors*, which is the distance of the observation on the edge and within the margin. This algorithm is dedicated to the classification of both linearly separable and non-linearly separable data points. In this paper, we will focus on the main usages of SVM, including the generalization of linear decision boundaries for classification. We will also discuss the role of kernel in SVM, as well as the implementation, evaluation methods, application and many aspects of SVM.

First, it would be helpful to take a look at an example of a simple data classification problem in a two-dimensional space. Assume that there are two sets of data points that are separately distributed into two clusters. The classification then becomes obvious: The threshold that separates both groups simply lies in the middle of the most outward data points from both groups. This classification method is called the *maximum margin classifier*, since the margin that separates both groups is maximized.

Nevertheless, the maximum margin classifier is not always applicable, since the data points are not always ideally separable. If an outlier happens to appear in the dataset, it would push the threshold to one side, since the data point is much closer to the other group. This would result in a severe misclassification, since the data that are close to one group now belongs to the other group because of the shift of the threshold. A solution to this problem is to allow some misclassification, so that the threshold is less sensitive to outliers and the classifier performs better when there is new data. This margin that allows some misclassifications is called the *soft margin*, and the classifier that is used to separate the data points is thus called the *soft margin classifier*.

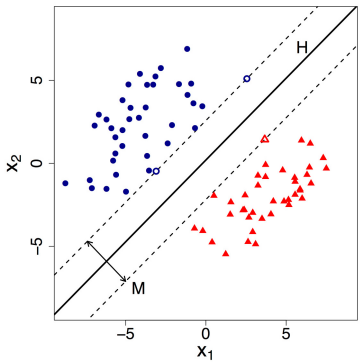


Fig. 1. Maximum Margin Classifier [1]

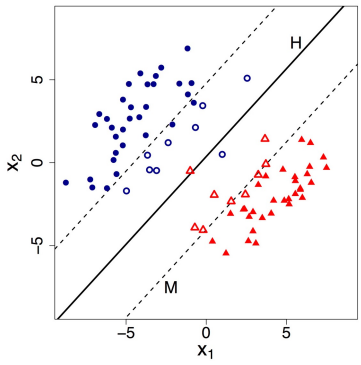


Fig. 2. Soft Margin Classifier [1]

For the real-life application, nevertheless, the most datasets that we work with are most likely not going to be one-dimensional. Due to the multidimensional property of the dataset, we use vectors and matrices to represent the data. In this case, the idea of a hyperplane is introduced.

A *hyperplane* is a flat affine subspace that is one dimension lower than the ambient space. [2] It is used to separate the data in multidimensional space. The soft margin classifier is able to deal with this case as well. It finds the hyperplane that separates the data in a way that the margin is maximized. We will discuss the exact algorithm and the mathematics in the next section.

However, the soft margin classifier is still not suitable for data that is not linearly separable, even though the soft margin classifiers allow misclassifications and is less sensitive to outliers. Usually, we use a *kernel* to deal with this case, which is a function that maps the data into a higher dimensional space so that the data becomes linearly separable. The SVM implements the idea of the kernel without transforming the data into a higher dimensional space. This concept is called *the kernel trick* and is a crucial part of SVM. This trick allows SVM to cast nonlinear variants to dot products, enabling easier computation and better performance. [3]

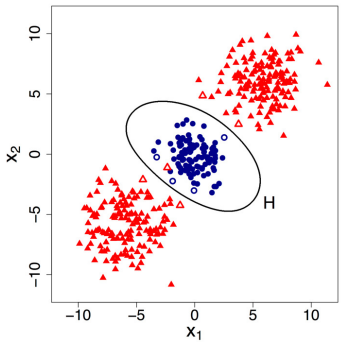


Fig. 3. Kernel Trick [1]

II. Implementation

The problem of support vector machine is an optimization problem. The goal is, as mentioned, to find the optimal hyperplane that separates the data in a way that the margin is maximized. Therefore, it is important to first define the hyperplane.

Since the hyperplane is a flat affine subspace of dimension p, we can easily define a hyperplane as [2]:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p = 0 \tag{1}$$

We can then denote X and β as vectors [4]:

$$\{X \in \mathbb{R}^p : X^T \beta + \beta_0 = 0\} \tag{2}$$

With the hyperplane being defined, we are now able to observe the problem of separating the spaces of the hyperplane. In other words, we need to construct linear decision boundaries that attempt to separate the data into two or more classes as precisely as possible. The two main approaches to this problem are Rosenblatt and Optimal.

The *Rosenblatt's perceptron learning algorithm* aims to find the optimal hyperplane in an iterative way. The algorithm minifies the distance of misclassified points to the decision boundary. The goal of this algorithm is to minimize the following equation

$$D(\beta, \beta_0) = \sum_{i \in M} -y_i (x_i^T \beta + \beta_0) \tag{3}$$

where M is the set of misclassified points and y_i is the class label of x_i, being either 1 or -1. Using stochastic gradient descent, the algorithm updates the coefficients and intercept.

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta} = -\sum_{i \in M} y_i x_i \tag{4}$$

$$\partial \frac{D(\beta, \beta_0)}{\partial \beta_0} = -\sum_{i \in M} y_i \tag{5}$$

However, the perceptron learning algorithm is not deterministic, since the result depends on the starting values. Furthermore, the algorithm does not converge if the data is not linearly separable, resulting in an infinite loop if the case was not detected beforehand. [5]

In 1996, Vapnik proposed the *optimal separating hyperplane* algorithm in order to cope with the problems of the perceptron learning algorithm. This algorithm is widely known as maximum margin classifier. The goal of the algorithm is to find a hyperplane that maximize the distance to the closet point from either class. The observation of both classes is denoted as $y_i = 1$ and $y_i = -1$. The separating hyperplane then has the following property:

$$X^T \beta + \beta_0 \geq 1, \text{if } y_i = 1 \tag{6}$$

$$X^T \beta + \beta_0 \leq -1, \text{if } y_i = -1 \tag{7}$$

To summarize:

$$y_i (X^T \beta + \beta_0) \geq 1, \ i = 1, ..., n \tag{8}$$

To solve this problem, we construct the following optimization problem [2]:

$$\begin{aligned} &\underset{\beta, \beta_0, ||\beta|| = 1}{\text{maximize}} \quad M \\ &\text{subject to} \\ &y_i (x_i^T \beta + \beta_0) \geq M, \ i = 1, ..., n \end{aligned} \tag{9}$$

where M is the margin. Since the margin is M units away from the hyperplane on either side, the margin is then $2M$ units wide. The constraint $||\beta|| = 1$ can be also left out by replacing the condition by

$$\frac{1}{||\beta||} y_i (x_i^T \beta + \beta_0) \geq M \tag{10}$$

If we arbitrarily set $M = \frac{1}{||\beta||}$, then the problem can be then reconstructed as [5]:

$$\begin{aligned} &\underset{\beta, \beta_0}{\text{maximize}} \quad \frac{1}{||\beta||} \\ &\text{subject to} \\ &y_i (x_i^T \beta + \beta_0) \geq 1, \ i = 1, ..., n \end{aligned} \tag{11}$$

The two constraints of this optimization problem ensure that each observation is on the right side and has at least a distance of M from the hyperplane. It can be then solved efficiently using Lagrange functions.

$$L_P(\beta, \lambda) = \frac{1}{||\beta||} - \sum_{i=1}^{N} \lambda_i [y_i(x_i^T \beta + \beta_0) - 1] \tag{12}$$

The goal of the Lagrange function is to find the local maxima and minima of a function that is subjected to equation constraints. Here, $\frac{1}{||\beta||}$ is the objective function, which is the function that is being maximized or minimized, with the constraints being $y_i(x_i^T\beta+\beta_0)-1$. λ is a vector of the weights of all the training points as support vectors under the condition of $\lambda_i \geq 0,\ i = 1, ..., n$. From this Lagrange function, we can derive the following two properties [5]:

- if $\lambda_i > 0$, then $y_i(x_i^T\beta+\beta_0)-1 = 0$, which is equivalent to $y_i(x_i^T\beta + \beta_0) = 0$. This means that the observation is on the margin.
- if $y_i(x_i^T\beta + \beta_0) - 1 > 0$, or $y_i(x_i^T\beta + \beta_0) > 1$, then $\lambda_i = 0$. This means that the observation is not on the margin.

However, the maximum margin classifier has zero tolerance for misclassification. As mentioned before, the soft margin is introduced to be more susceptible to misclassification. With the mathematical knowledge given above, we can also define the soft margin classifier. The hyperplane is chosen to correctly separate most of the dataset into two classes with the tolerance of a few errors. Thus, we introduce a slack variable ϵ_i, which allows individual observations to be on the wrong side of the margin. The optimization problem is then defined as [2]:

$$\begin{aligned} &\underset{\beta, \beta_0, \epsilon}{\text{maximize}} \quad \frac{1}{||\beta||} \\ &\text{subject to} \\ &y_i(x_i^T\beta + \beta_0) \geq \frac{1}{||\beta||}(1 - \epsilon_i),\ i = 1, ..., n, \\ &\epsilon_i \geq 0,\ \sum_{i=1}^{n} \epsilon_i \leq C \end{aligned} \tag{13}$$

The interpretation of the slack variable helps us to determine whether an observation is on the correct side of the margin. The ith observation is on the correct side of the margin if $\epsilon_i = 0$. Otherwise, the ith observation is on the wrong side of the margin if $\epsilon_i > 0$ and on the wrong side of the hyperplane if $\epsilon_i > 1$. The parameter C, in this case, is the tuning parameter. It determines the budget of violations that could happen in this classification problem. The choice of C plays a decisive role in the learning process. If C is large, then many observations violate the margin, which causes the involvement of many observations in determining the hyperplane. On the other hand, smaller Cs result in a classifier with lower bias but higher variance. In practice, cross-validation is used to choose the ideal C.

Based on the optimization problem given above, we can derive the formal definition of SVM. In comparison to soft margin classifier, the SVM transforms the data into a higher dimension space p', so that the data could be linearly separable. For instance, we transform the features $X_1, X_2, ..., X_p$ into $X_1, X_1^{p'} X_2^{p'}, ..., X_p^{p'}$. The optimization problem with the transformed data can then be altered to [2]:

$$\begin{aligned} &\underset{\beta_0, \beta_{11}, \beta_{12}, ..., \beta_{p(p-1)'}, \beta_{pp'}, \epsilon_1, ..., \epsilon_n}{\text{maximize}} \quad \frac{1}{||\beta||} \\ &\text{subject to} \\ &y_1(\beta_0 + \sum_{j=1}^{p}\sum_{k=1}^{p'} \beta_{jk} x_{ij}^k) \geq \frac{1}{||\beta||}(1 - \epsilon_i) \\ &\sum_{i=1}^{n} \epsilon_i \leq C,\ \epsilon_i \geq 0,\ \sum_{j=1}^{p}\sum_{k=1}^{p'} \beta_{jk}^2 = 1 \end{aligned} \tag{14}$$

In order to perform the transformation and simplify this problem, we use a kernel function $K(x, x')$. It is defined as [4]:

$$K(x, x') = \langle h(x), h(x') \rangle \tag{15}$$

where $h(x)$ is a transformation function. The transformation process leaves us with a lot of freedom, and we are able to freely decide how we should transform the data. The two popular choices of kernel functions in practical applications are:

- dth-Degree Polynomial kernel: $K(x, x') = (1 + \langle x, x' \rangle)^d$
- Radial kernel: $K(x, x') = \exp(-\gamma||x - x'||^2)$

First, let us discuss the polynomial kernel. The polynomial kernel transforms each data point into a polynomial of degree d and fits a support vector classifier into this higher-dimensional space. This transformation leads to a flexible decision boundary.

$$K(x, x') = (1 + \langle x, x' \rangle)^d = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{16}$$

Another popular choice is the radial kernel. In order to understand the radial kernel, we first need to understand the following scenario. If a given test observation x^* is far away from the observation x_i, then the Euclidean distance between the two observations will be large. As a result, the exponential term of the negative value of the Euclidean distance will be close to zero, which means that the presence of x^* will not have much effect on the transformation. On the other hand, if x^* is close to x_i, then the exponential term will be close to one, which will exert a strong influence on the transformation. The radial kernel utilizes this property of the exponential function to emphasize the points that are close to the test observation x^* [2].

$$K(x, x') = \exp(-\gamma||x - x'||^2) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{17}$$

The implementation of kernels in SVM saves computational power, since for each pair of data points, we only need

to calculate the transformed value once. Furthermore, the transformative property of the kernel leads to a flexible and precise boundary. [2]

III. Application

With the mathematical definition and deduction being discussed, we can now look at the application of SVM in real life. Being a classification method, SVM is widely used in many fields in order to distinguish data classes from another. In this section, we will discuss five fields in which SVM is dominantly utilized in: Face recognition, text classification, bioinformatics and medical diagnosis, as well as handwriting recognition.

Face Recognition

Face recognition is a biometric method that is widely used today in order to identify a specific person. It captures facial features such as eyes, chins, nose, et cetera and uses the properties, for example, the distance or the angle between two parts, to distinguish a person from others. In reality, though, people do not always look completely different from each other. The usage of SVM, in this case, strengthens the accuracy by its pattern recognition ability.

Since the basic theory of SVM is used to classify data into two classes, we need to modify the learning algorithm in order to apply it to facial recognition. The key here is to combine multiple SVM models in order to construct a bottom-up binary tree. We compare each pair of nodes in the tree and the winner of the comparison will be promoted to the upper level. In the end, the unique class, which is the winner, will appear on the top of the tree. Given C classes, this algorithm learns $C(C-1)/2$ SVM models at the training phase and performs $C-1$ comparisons at the testing phase.

This algorithm is used on the Cambridge ORL face database for testing purposes, which contains 40 people with 10 face images per person. The result shows that SVM performs better than other algorithms including pseudo two-dimensional HMMs (Hidden Markov Model) with a 5% error rate and CNN (Convolutional Neural Network) with a 3.83% error rate. SVM has only classified 3.0% of the images incorrectly. In another experiment, this algorithm is used to compete against the eigenface method with the nearest center classification (NCC), which has been long applied in the face recognition field. Nevertheless, SVM still outperformed NCC, having scored an 8.79% minimum error rate in comparison to NCC's 15.14%. [6]

Text Classification

Dealing with text data has long been a challenge in the machine learning field, and this task is no exemption for SVM. In order to suit the algorithm for text classification, we need to implement pool-based active learning to the SVM model. The idea is that the learner has access to a pool of unlabeled data and is allowed to request the label of some data points in the pool.

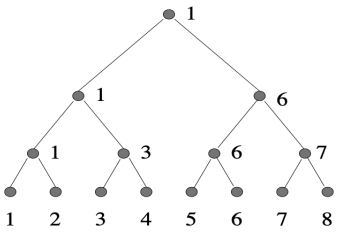
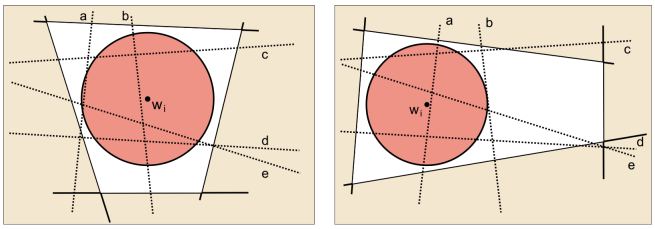


Fig. 4. The binary tree structure for 8 classes face recognition. The numbers 1-8 encode the classes. Each class is first compared to the nearby node, and the winner is promoted to the upper level. After $\log_2(8) = 3$ comparisons, the unique class number appears on the top of the tree. [6]

Given a set of data points in a multidimensional space, we call the set of hyperplanes to separate the data points in the version space. The goal of the newly implemented algorithm is to reduce the version space as fast as possible. Here, we introduce the idea of active learning, which consists of a boolean classifier, a query strategy and a set of unlabeled data points. In this case, we can choose the next query in a greedy way such that the version space is reduced as fast as possible. There are three ways to instantiate this idea:

- **Simple Margin**: The algorithm picks the unlabeled instances in the pool whose hyperplane comes closest to the SVM unit vector, which is the center of the largest hypersphere that can fit inside the version space.
- **MinMax Margin**: For each unlabeled instance, compute the margins of the SVMs obtained when we label the instance as positive and negative respectively. Then, we choose the instance with the smallest margin.
- **Ratio Margin**: The algorithm works like MinMax margin, but uses the normalized margin instead of the absolute margin.

For inductive learning cases such as email filtering, a classifier is trained based on an SVM with a set of labeled data points. For transductive learning cases such as relevance feedback, the classifier learns an SVM with both labeled and unlabeled data points. In reality, it turns out that combining MinMax margin and ratio margin methods delivers the best result. [7]



(a) (b)

Fig. 5. A simple comparison of simple margin and MinMax margin. For this example, the simple margin will query **b**, whereas the MinMax margin will query **a**. [7]

Bioinformatics

In the bio-informatics field, two of the most common tasks are cancer diagnosis and protein secondary structure prediction (PSSP). Since both of these tasks are classification problems, SVM can be easily applied to them and is capable of generating high accuracy results.

For cancer diagnosis, it is important to identify the types of cancer and the responsive genes. However, all genes tend to have high dimensionality, while there are only a few samples available. For this problem, the research group from Nanyang Technological University applied SVM with Welch's t-test. The t-test shows the degree of differences between two groups of data points. Here, we first apply the t-test to the data points based on which they are ranked. Afterwards, we select the top data points and apply SVM to them. The result after applying this hybrid algorithm to several cancer datasets shows a very high accuracy score. For the SRBCT and the Lymphoma dataset, the algorithm achieved a 100% accuracy score. For the Leukemia dataset, the algorithm only misclassified one instance out of 34 samples.

The PSSP problem is to predict the secondary structure of a protein. The protein sequence is a linear array of amino acids, which is the basic unit of a protein sequence. It consists of 3 consecutively ordered DNA bases (A, T, G, C). The secondary structure of the protein is formed between small segments of protein sequences with hydrogen bonds, which are mainly divided into three types: α-*helix*, β-*sheet*, and *coil*. In PSSP, we should classify the residues into these three types. The algorithm consists of two phases: Sequence-Structure Prediction (Q2T) and Structure-Structure Prediction (T2T). Q2T predicts the secondary structure from the protein sequences, whereas T2T helps correct the errors caused by Q2T. For Q2T, the SVM parameters alongside the window size are tuned to train the model. For the window size of 13 and 15, the highest accuracy score of 74.2% is generated. T2T uses the output of Q2T as the input and corrects the errors. The highest accuracy score achieved is 78.2%. [8]

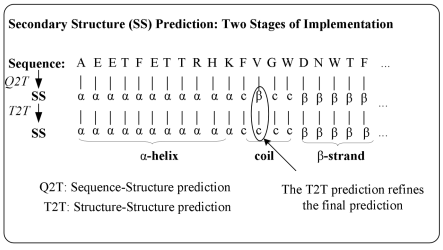


Fig. 6. Two phases of PSSP [8]

Handwriting Recognition

The process of handwriting recognition consists of three phases: pre-processing, feature extraction and classification. In the pre-processing phase, the images undergo noise removal and binarization, skew detection and correction, segmentation, as well as normalization and thinning. Then, the feature selection technique is implemented to determine the width/height ratio, number of endpoints, number of cross points and number of branch points. Finally, the SVM comes in to recognize the pattern. Like the usage of SVM in face recognition, the SVM is used in a one-against-all fashion, so that it can distinguish between different classes.

In an experiment conducted on a dataset of 10,000 handwritten isolated Malayalam characters with 44 distinct characters, the dataset is divided into 80% training set and 20% testing set and trained in five iterations using cross validation. The SVM with polynomial kernel reports an average accuracy of 90.83%, whereas the SVM with radial basis function kernel reports an average accuracy of 92.24%, which is substantial subjected to this problem. [9]

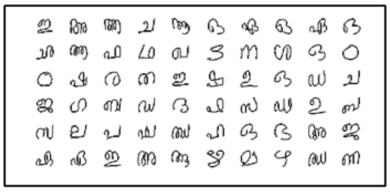


Fig. 7. Handwriting after binarization preprocessing [9]

IV. Relative Merits

Having discussed the application of SVM in real life, we can now conclude the advantages and disadvantages of SVM as well as the point of using it as a classification method.

The advantages can be described as follows:

- **Suitable for All Data**: The incorporation of kernel functions allows SVM to be used on data with any form of distribution. Kernel functions are able to transform the data in a way that it is linearly separable. Therefore, one shall not worry about the property of the data while applying SVM.
- **Human Knowledge Not Required**: Since the implementation of kernel functions means that SVM is able to deal with any form of data, the user does not need to have prior knowledge about the background of the data. For instance, SVM is easily used in the field of bioinformatics without the users having to understand the biological meaning of the data.
- **Unique Solution**: The property of convex optimization allows SVM to deliver unique solutions to one single dataset. For other algorithms, such as Convolutional Neural Network, the result may vary based on the position of the datasets, since it alters the local minimum of the function.
- **Flexibility in Kernel Choice**: Even though we use the polynomial kernel and the radial basis kernel in most occasions, the choice of kernel is versatile. Therefore, users can use their knowledge of the data to transform it in order to suit real world situations. In this case, the

SVM model will be fine-tuned in a way that it delivers an overall better result.

Like every other machine learning algorithm though, SVM does have its disadvantages, even though it is widely used in many fields. The main disadvantage is that the result of SVM is not interpretable and transparent enough. The relationship between the data and the result is hard to explain, since the data has experienced complicated transformations through the kernel function. Therefore, it would be hard to be used to explain the relationship between the data and the result. For instance, in the case of face recognition, SVM is able to distinguish a person from another, but it cannot tell the user why it is able to do so in such an accurate way. In this case, however, we are able to use data visualization to graphically interpret the relationship between the input and the output. Bi-dimensional graphs with color coding are, for instance, used popularly in this case, since it shows the property of the data distribution in a relational way. [10]

V. Conclusion

In a nutshell, SVM is an algorithm that is capable of solving classification problems with high accuracy. The main idea of SVM is to find the optimal hyperplane that separates the data into two classes. In cases where the data points are not linearly separable, SVM is able to transform the dataset in a way that they are linearly separable using kernel functions. It is a highly automated algorithm that does not require users to deal with the data distribution properties before applying it. However, its flexibility also allows users to customize their own kernel functions to transform the data in a way that suits the real life situation, so that an even better result could be delivered using these fine-tuning mechanics.

With the properties discussed above, SVM does not really require the users to have deep background knowledge in the field, since it is highly automated. The algorithm often delivers high accuracy in binary classification. But it can be modified to deal with more complicated problems, such as multi-class classification and pattern recognition. Its flexibility in usage makes it a popular choice among researches in many fields. Popular applications of SVM include image preprocessing, authentication, natural language processing, and bioinformatics. The drawback of SVM is that the interpretation of the result might be difficult. Nevertheless, we can use data visualization techniques to help us understand the relationship between the input and the output.

References

[1] A. Kirchner and C. S. Signorino, "Using support vector machines for survey research," *Survey Practice*, vol. 11, no. 1, 1 2018.

[2] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, second edition ed. New York: Springer, 2021, ch. 9, Support Vector Machines.

[3] T. Hofmann, B. Schölkopf, and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: The MIT Press, 2002, ch. 2, Kernels.

[4] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, second edition ed. New York: Springer, 2009, ch. 12, Support Vector Machines and Flexible Discriminants.

[5] ——, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, second edition ed. New York: Springer, 2009, ch. 4, Linear Methods for Classification.

[6] G. Guo, S. Li, and K. Chan, "Face recognition by support vector machines," *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 02 1970.

[7] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 12 2001.

[8] F. Chu, G. Jin, and L. Wang, *Cancer Diagnosis and Protein Secondary Structure Prediction Using Support Vector Machines*, 11 2005, vol. 177, pp. 578–578.

[9] D. Nasien, H. Haron, and S. Yuhaniz, "Support vector machine (svm) for english handwritten character recognition," *Computer Engineering and Applications, International Conference on*, vol. 1, pp. 249–252, 01 2010.

[10] L. Auria and R. A. Moro, "Support Vector Machines (SVM) as a Technique for Solvency Analysis," *SSRN Electronic Journal*, 2008. [Online]. Available: http://www.ssrn.com/abstract=1424949