Support Vector Machines: An Introduction

Seminar Data Mining

Chengjie ”Jay” Zhou
Department of Informatics
Technische Universität München
Email: chengjie.zhou@mytum.de

***Abstract*— Text of the abstract**

***Keywords*— Data Mining, Support Vector Machines**

I. Introduction

One of the most important ascpects 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 early 90s, Vladimir Vapnik and his colleagues developed a new algorithm called *Support Vector Machine (SVM)*, which is optimized for classification and regression analysis. 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, evalution methods, application and many different aspects of SVM.

In order to thoroughly understand the classification problem, we first need to look at a simple example in one dimension. Suppose there are two groups of data points that are separately distributed on a one-dimension number line. The classification then becomes obvious: The threshold that separates both groups simply lies in the middle of the 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.

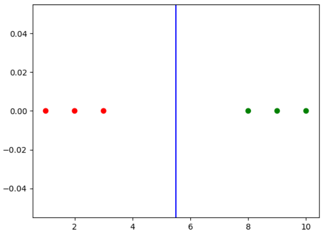

Fig. 1. Maximum Margin Classfier

Nevertheless, the maximum margin classifier is not always applicable, since the data points are not always idealy distributed in a separated way. 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 has higher bias and is less sensitive to outliers and the classifier performs better when there is new data. This margin that allows some misclassification is called the *soft margin*. The determination of soft margin could be tricky, since there are limitless points of thresholds to choose from. One way to find the optimal threshold is to use Cross Validation. Cross Validation is a method that splits the dataset into several parts. For each repetition, one part of the dataset is used for testing and the rest is used for training. After training through all the repetition, the average position of the threshold represents the most ideal position of the soft margin. This classifier is called soft margin classifier, also known as support vector classifier. The name ”Support Vector” derives from the fact that the data points that are closest to the threshold are called support vectors.

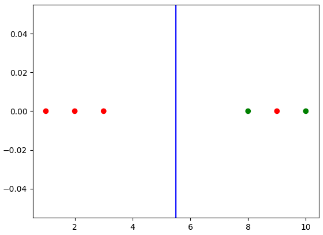

Fig. 2. Support Vector Classfier

In reality, chances are that the data is not only one-dimensional. In fact, almost all of the data that we work with is multi-dimensional. In order to represent the data in a multi-dimensional space, we need to use vectors. In this case, the idea of hyperplane is introduced. A *hyperplane* is a threshold that separates the data in a multi-dimensional space, which is formally defined as a flat affine subspace. [1] The support vector 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. The exact algorithms and the mathematical derivation will be discussed in the next section.

However, the support vector classifier is still not suitable for data that is not linearly separable, even though the support vector classifiers allows 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

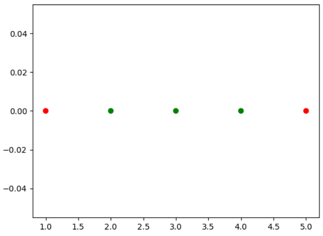

Fig. 3. Non Linearly Separable Data

space, so that the data becomes linearly separable. The SVM implements the idea of 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. [2]

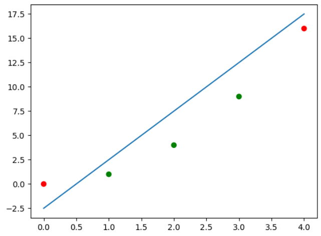

Fig. 4. Kernel Transformation

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 [1]:

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

We can then denote X and β as vectors [3]:

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

With hyperplane being defined, we are now able to observe the problem of separating 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 main two mechanics 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 minizies 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. [4]

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 implemented 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. [Vapnik, 1996] 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 [1]:

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

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

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

We arbitrarily set $M = \frac{1}{||\beta||}$. The problem can be then reconstructed as [4]:

$$\begin{aligned} &\underset{\beta, \beta_0}{\text{maximize}} \quad \frac{1}{2}||\beta||^2 \\ &\text{subject to} \\ &y_i(x_i^T \beta + \beta_0) \geq 1, \; i = 1, ..., n \end{aligned} \tag{11}$$

The two constraints of this optimization problem ensures that each observation is on the right side and has at least a distance of M from the hyperplane. It can be then solved in an efficient way using Lagrange functions. The mathematical deduction is beyond the scope of this paper, but the result is as follows:

$$\beta = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{12}$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i \tag{13}$$

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0 \ \forall i \tag{14}$$

where α is a vector of the weights of all the training points as support vectors under the condition of $\alpha_i \geq 0, \ i = 1, ..., n$. From these three equations, we can derive the following two properties [4]:

- if $\alpha_i > 0$, 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 $\alpha_i = 0$. This means that the observation is not on the margin.

However, the maximum margin classifier has zero tolerance for misclassification. As mentioned before, the support vector is designed to be more susceptible to misclassification. With the mathematical knowledge given above, we can also define the support vector classifer. 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 [1]:

$$\begin{aligned}
&\underset{\beta, \beta_0, \epsilon, M}{\text{maximize}} \quad M \\
&\text{subject to} \\
&||\beta|| = 1, \\
&y_i(x_i^T\beta + \beta_0) \geq M(1 - \epsilon_i), \ i = 1, ..., n, \\
&\epsilon_i \geq 0, \ \sum_{i=1}^{n} \epsilon_i \leq C
\end{aligned} \tag{15}$$

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 paramter. 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 varianace. In practice, C is chosen using cross-validation.

Based on the optimization problem given above, we can derive the formal definition of SVM. In comparison to support vector classifier, the SVM transform 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 [1]:

$$\begin{aligned}
&\underset{\beta_0, \beta_{11}, \beta_{12}, ..., \beta_{p(p-1)'}, \beta_{pp'}, \epsilon_1, ..., \epsilon_n, M}{\text{maximize}} \quad M \\
&\text{subject to} \\
&y_1(\beta_0 + \sum_{j=1}^{p}\sum_{k=1}^{p'} \beta_{jk} x_{ij}^k) \geq M(1 - \epsilon_i) \\
&\sum_{i=1}^{n} \epsilon_i \leq C, \ \epsilon_i \geq 0, \ \sum_{j=1}^{p}\sum_{k=1}^{p'} \beta_{jk}^2 = 1
\end{aligned} \tag{16}$$

In order to perform the transformation and simplify this problem, we use a kernel function $K(x, x')$. It is defined as [3]:

$$K(x, x') = \langle h(x), h(x') \rangle \tag{17}$$

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 pracitcal applications are:

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

Let us first take a look at 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.

$$K(x, x') = (1 + \langle x, x' \rangle)^d = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{18}$$

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 points will be large. As a result, the exponential term of the negative value of the Euclidean distance will be close to zero. In this case, 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^* [1].

$$K(x, x') = \exp(-\gamma||x - x'||^2) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{19}$$

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. [1]

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 that SVM is dominantly utilized in: Face recognition, text classification, image classification, bioinformatics and medical diagnosis, as well as handwriting recognition.

Face Recognition

Face recognition is a biometric method that is used 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 angel between two parts, to distinguish one person from another. The difficulty of this method is that in reality, there are people who look similar to 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 the facial recognition use case. 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 nearest center classification (NCC), which has been long applied in the face recognition field. Nevertheless, SVM still outperformed NCC, having scored a 8.79% minimum error rate in comparison to NCC's 15.14%. [5]

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.

Given a set of data points in a multidimensional space, we call the set of hyperplanes to separate the data points 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 a SVM with a set of labeled data points. For transductive learning cases such as relevance feedback, the classifier learns a 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. [6]

Image Classification

The problem of object detection in images is a challenging task in the data mining field. Some of the challenges include: Variations in shape due to distortions, variations in color due to illumination conditions, and variations in size due to perspective.

For this problem, we observe a specific use case: The detection of license plates in images. The research group in Korea Advanced Institute of Science and Technology has proposed an algorithm based on SVM to analyze the color textural properties of the license plates. The algorithm goes as follows:

- Set up initial locations and sizes of search windows in images.
- Generate the license plate probability distribution using SVM.
- Estimate parameters of the license plate using mathematical equations. Parameters include: location, size, orientation, and so on.
- Modify the search windows based on the estimated parameters.
- Reestimate the parameters using the new search windows.
- Output bounding windows as the result.

This algorithm is called continuously adaptive meanshift algorithm (CAMShift). The main usage of SVM here is to predict the probability of a pixel belonging to a license plate, which enables to essential parameters to be accurately estimated. It produces solid result within seconds, with an average miss rate of around 8.78% and an average false detection rate of around 14.88%. Nevertheless, this algorithm is still not perfectly responsive to bad illumination and excessive burring. [7]

IV. Summary and Outlook

blabla

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] K. Kim, K. Jung, and J. Kim, "Color texture-based object detection: An application to license plate localization," vol. 2388, 01 2002, pp. 293–309.