YILDIZ TECHNICAL UNIVERSITY DATA SCIENCE PROGRAM

-

PROJECT REPORT

A comprehensive survey on support vector machine classification: Applications, challenges and trends

ALİCAN TUNÇ /235B7055

ABSTRACT

In this advanced project, I aimed to sum a comprehensive survey of support vector machine classification: applications, challenges, and trends. Throughout this review, I will discuss the strengths and weaknesses of support vector machines, and after touching on the theory, I will examine their current usage areas, trends and problems.

Contents

1. Introduction

The highly interdisciplinary field of machine learning draws inspiration from numerous scientific and mathematical fields, including optimisation, computer science, statistics, and cognitive science. Classification in machine learning is an approach to supervised learning that is used to analyse a given data set and create a model that divides the data into a desired number of distinct classes. There are various classification techniques available in the literature, such as k-nearest-neighbor classifiers, bayesian networks, artificial neural networks, decision trees, and SVM(J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., 2019 p. 1). Although K-nearest-neighbor (KNN) algorithms are widely praised for their ease of use, their high computational requirements often result in poor efficiency when dealing with large input datasets. Furthermore, they exhibit high sensitivity to superfluous parameters. On the other hand, decision tree(which are faster than neural networks during the training phase) are frequently employed for classification tasks. Decision trees, however, are not flexible enough to represent intricate relationships between parameters. On the other hand, neural networks are widely used as a flexible tool in many different applications. However, building a neural network that works well necessitates giving careful thought to a number of factors, including the architecture, number of neurons and layers, learning algorithm, and data representation. Neural networks also react very strongly to noise in the training set. Among these techniques, Support Vector Machine (SVM) stands out as one of the most effective methods for optimizing the expected solution. SVM, introduced by Vapnik, is a kernel-based machine learning model used for classification and regression tasks. Its remarkable generalization capability, optimal solution, and discriminative power have garnered significant attention from the data mining, pattern recognition, and machine learning communities in recent years. SVM has been widely adopted as a powerful tool for solving practical binary classification problems, outperforming other supervised learning methods. Thanks to its strong theoretical foundations and excellent generalization capacity, SVMs have become one of the most commonly used classification methods.

2. Theoretical Background of SVM

The main purpose of a pattern classification model is to get the best performance out of the training data. With traditional training methods, the goal is to classify every input-output pair in its class correctly. However, if a classifier is trained too hard on the training data, instead of learning how to generalize, it may start to memorize the data, resulting in a decrease in classifier generalization. SVM stands for Support Vector Machines. The basic idea behind SVM is to divide different classes in a training set using a surface that allows for the largest margin between them. In other words, the goal of SVM is to maximize the model's generalization capability. This is in line with SRM, which focuses on minimizing a bound on a model's overall generalization error rather than minimizing the average squared error of the training data, which is often done with empirical risk minimization techniques.

2.1 Linearly separable case

Training a Support Vector Machine (SVM) requires a set of n examples, where each example is a pair consisting of an input vector xi and its associated label yi. The training set X can be represented as:

$$(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \qquad (1)$$ [1]

Here, xi represents the input vector for the n-th example, and yi is its associated label. The goal of SVM training is to learn a model that can correctly classify these examples into their respective classes. For the purpose of visualization, let's consider the case of a two-dimensional input, where x ∈ R^2. If the data is linearly separable, there are many hyperplanes that can perfectly separate the input data set. Figure 1 illustrates several decision hyperplanes that achieve this separation. It's evident that there are infinitely many hyperplanes that could achieve this separation. However, the generalization ability of the classifier depends on the position of the separation hyperplane, with the hyperplane that maximizes the margin being optimal. The decision boundary, or hyperplane, that separates the input space can be defined by the equation

$$\mathbf{w}^{\mathsf{T}}\mathbf{x} - b = 0,$$

(J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., 2019 p. 2)

Here, w is the weight vector perpendicular to the hyperplane, x is the input vector, and b is the bias term.

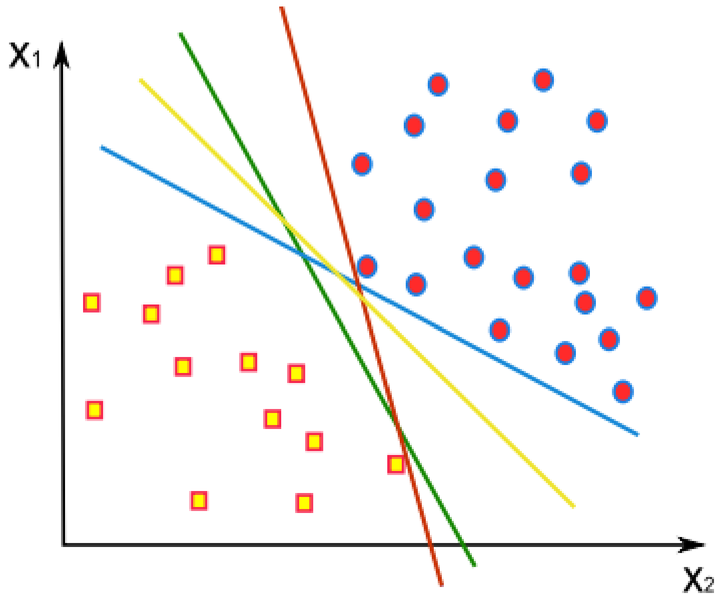


Fig. 1. Separation hyperplanes. [2]

$$\langle w \cdot x^{+} \rangle + b = 1$$
$$\langle w \cdot x^{-} \rangle + b = -1 \quad (2)$$

These can be combined into a set of inequalities:

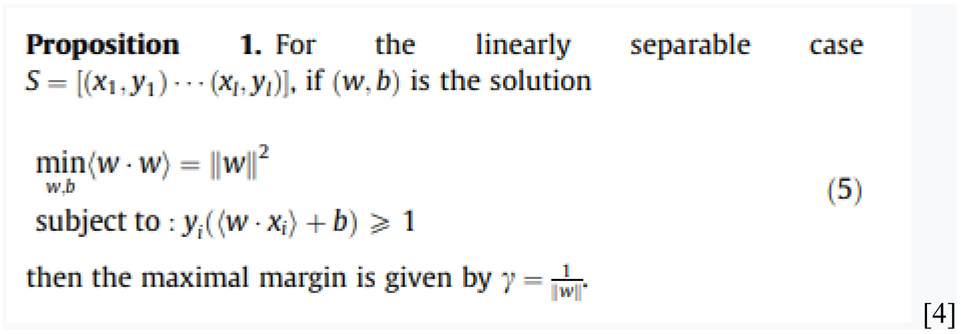
$$y_i(\langle w \cdot x_i \rangle + b) \geqslant 1 \forall i \quad (3)$$

The geometric margin of x^{+} y x^{-} is

$$\begin{aligned} \gamma_i &= \frac{1}{2}\left(\left\langle \frac{w}{\|w\|} \cdot x^{+} \right\rangle - \left\langle \frac{w}{\|w\|} \cdot x^{-} \right\rangle\right) \\ &= \frac{1}{2\|w\|}\left[\langle w \cdot x^{+} \rangle - \langle w \cdot x^{-} \rangle\right] \\ &= \frac{1}{\|w\|} \end{aligned} \quad (4)$$

[3]

In the context of Support Vector Machines (SVMs), w defines the optimal separation hyperplane and b is the bias term. The distance between the hyperplane and the training data point that is closest to it is known as the margin. Maximising the ability to generalise is achieved by choosing the ideal separation hyperplane. Minimising the weight vector's norm is necessary to optimise the geometric margin. Our goal when using SVMs to solve the quadratic programming problem is to identify both the optimal and two parallel hyperplanes (H1 and H2). To ensure that there is no data between the two parallel hyperplanes (H1 and H2), the objective of an SVM is to maximise the margin—the distance between them. Some data points may lie on or above H1, while others may lie on or below H2, when the margin between H1 and H2 is maximised. Because these data points have a direct impact on the separation hyperplane's position, they are referred to as support vectors. The remaining data points, which do not fall between H1 and H2, are modifiable or removeable without affecting the hyperplanes' position or the classifier's capacity to generalise. Thus, this small set of support vectors defines the solution of an SVM. Briefly, we say that with this method we can determine the vector that separates the data in the best way.



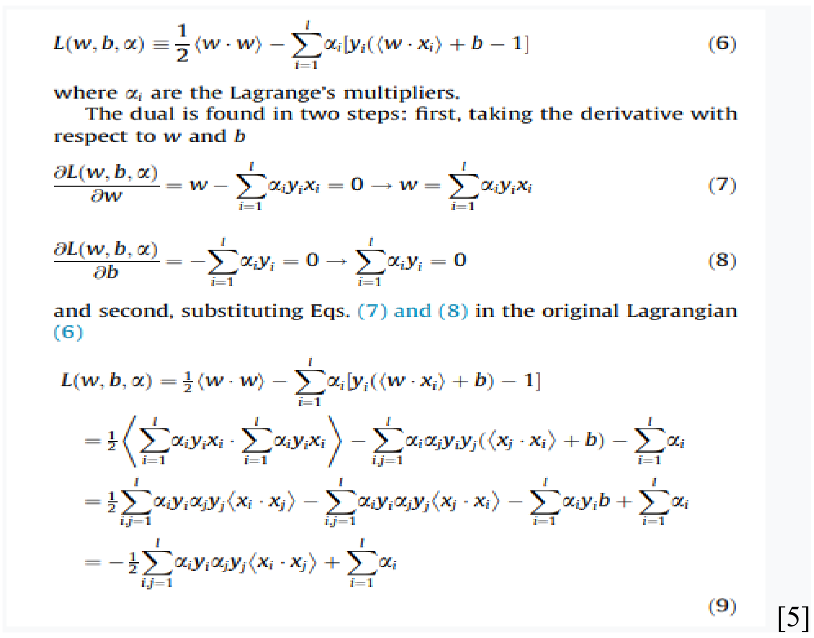
Proposition 1. For the linearly separable case $S = [(x_1, y_1) \cdots (x_l, y_l)]$, if (w, b) is the solution

$$\begin{aligned} &\min_{w,b} \langle w \cdot w \rangle = \|w\|^2 \\ &\text{subject to} : y_i(\langle w \cdot x_i \rangle + b) \geqslant 1 \end{aligned} \qquad (5)$$

then the maximal margin is given by $\gamma = \frac{1}{\|w\|}$.

[4]

We transform the SVM optimization problem into its dual form using the Lagrange formulation for two main reasons. Firstly, the conditions specified in the original problem can be replaced by Lagrange multipliers, which are easier to work with. Secondly, reformulating the problem in this way allows the training data to only appear in the form of dot products between vectors. This is a crucial property that enables the generalization of the SVM procedure to the nonlinear case. The Lagrangian for the SVM is given by:



$$L(w, b, \alpha) \equiv \frac{1}{2}\langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i [y_i(\langle w \cdot x_i \rangle + b - 1] \qquad (6)$$

where α_i are the Lagrange's multipliers.

The dual is found in two steps: first, taking the derivative with respect to w and b

$$\frac{\partial L(w, b, \alpha)}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \rightarrow w = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad (7)$$

$$\frac{\partial L(w, b, \alpha)}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i = 0 \rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (8)$$

and second, substituting Eqs. (7) and (8) in the original Lagrangian (6)

$$\begin{aligned} L(w, b, \alpha) &= \tfrac{1}{2}\langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1] \\ &= \tfrac{1}{2}\left\langle \sum_{i=1}^{l} \alpha_i y_i x_i \cdot \sum_{i=1}^{l} \alpha_i y_i x_i \right\rangle - \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j (\langle x_j \cdot x_i \rangle + b) - \sum_{i=1}^{l} \alpha_i \\ &= \tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j \rangle - \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_j \cdot x_i \rangle - \sum_{i=1}^{l} \alpha_i y_i b + \sum_{i=1}^{l} \alpha_i \\ &= -\tfrac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l} \alpha_i \end{aligned} \qquad (9)$$

[5]

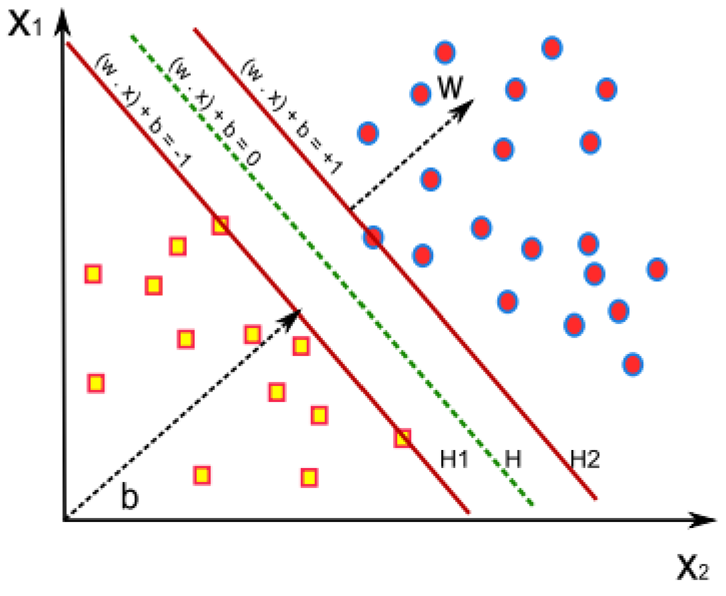
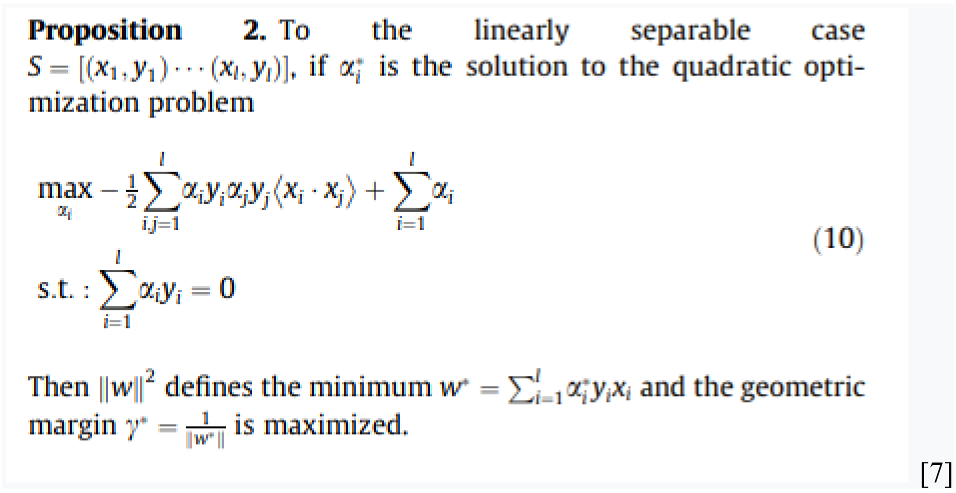


Fig. 2. Optimal classifier.

[6]

The figure would show a geometric representation of the quadratic programming problem for SVMs. It would illustrate the optimal separator H (the hyperplane that maximizes the margin), as well as the two parallel hyperplanes H1 and H2 that define the margin.



Proposition 2. To the linearly separable case $S = [(x_1, y_1) \cdots (x_l, y_l)]$, if α_i^* is the solution to the quadratic optimization problem

$$\max_{\alpha_i} -\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i y_i \alpha_j y_j \langle x_i \cdot x_j \rangle + \sum_{i=1}^{l}\alpha_i$$
$$\text{s.t.} : \sum_{i=1}^{l}\alpha_i y_i = 0 \tag{10}$$

Then $\|w\|^2$ defines the minimum $w^* = \sum_{i=1}^{l}\alpha_i^* y_i x_i$ and the geometric margin $\gamma^* = \frac{1}{\|w^*\|}$ is maximized.

[7]

2.1.1 Karush-Kuhn-Tucker conditions

The Karush-Kuhn-Tucker conditions (KKT) are fundamental to the theory of optimisation since they provide the necessary conditions for obtaining the best possible solution to a wide range of optimisation problems.

Theorem 1. *Given an optimization problem with convex domain* $\Omega \subseteq \mathbb{R}^n$,

$$\begin{aligned} &\text{minimize} \quad f(w), \quad w \in \Omega \\ &\text{s.t.} \qquad g_i(w) \leqslant 0, i = 1, \ldots, k, \\ &\qquad\qquad h_i(w) = 0, i = 1, \ldots, m, \end{aligned} \tag{11}$$

with $f \in C^1$ convex, the necessary and sufficient conditions for a normal point w^* to be optimal are the existence of α^*, β^* such that

$$\begin{aligned} &\tfrac{\partial L(w^*,\alpha^*,\beta^*)}{\partial w} = 0 \\ &\tfrac{\partial L(w^*,\alpha^*,\beta^*)}{\partial \beta} = 0 \\ &\alpha_i^* g_i(w^*) = 0, i = 1, \ldots, k, \\ &g_i(w^*) \leqslant 0, i = 1, \ldots, k, \\ &\alpha_i^* \geqslant 0, i = 1, \ldots, k. \end{aligned} \tag{12}$$

From KKT conditions if the training set is linearly separable, it is verified that

$$\|w^*\|^2 = \langle w^* \cdot w^* \rangle = (\sum_{i \in sv} \alpha_i^*)^{-\frac{1}{2}} \tag{13}$$

Therefore, the maximum distance of a hyperplane is:

$$\frac{1}{\|w^*\|} = \left(\sum_{i \in sv} \alpha_i^*\right)^{-\frac{1}{2}} \tag{14}$$

[8]

2.2 Soft margin hyperplanes and non-linear classifier

The learning problem described earlier assumes that the data is linearly separable, meaning that the training data points do not intersect. However, such perfectly separable datasets are rare in real life. In practice, there are cases where a linear separation hyperplane can still yield good results even when the data points intersect.

However, the quadratic programming solutions discussed earlier cannot be directly applied in cases where the data points intersect. This is because the condition $y_i(\langle w \cdot x_i \rangle + b) \geqslant 1,$ for all i cannot be satisfied when there are intersections (see Figure 3). Points that lie in the intersection cannot be correctly classified, and for any misclassified data point xi, its corresponding alphai (Lagrange multiplier) will tend towards infinity. (J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., 2019 p. 4)

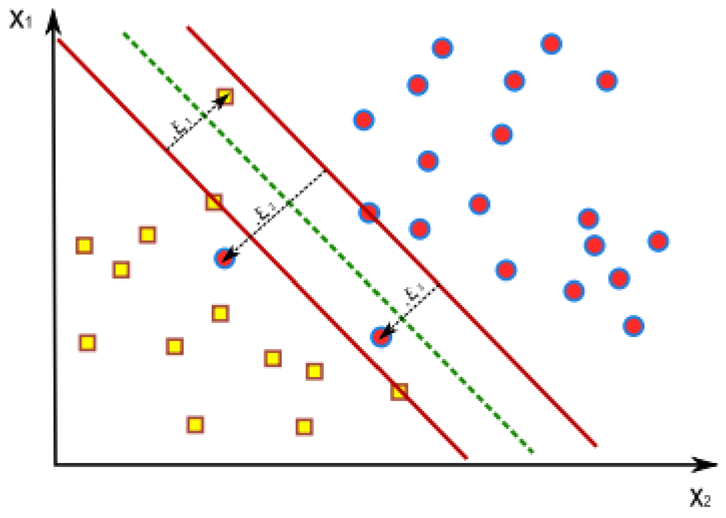


Fig. 3. Soft margin hyperplanes. [9]

To accommodate cases where the data is not perfectly separable, the algorithm can be modified to allow for a soft margin (see Figure 4). This involves introducing non-negative slack variables ni > 0 in the equation:

$$y_i(\langle w^T \cdot x_i \rangle + b) \geqslant 1 - \xi_i \quad \forall i \qquad (15)$$ [10]

Using these slack variables, a feasible solution always exists. For training data xi, if 0< ni < 1, the data points are not at the maximum margin but can still be correctly classified. The width of this soft margin can be controlled by the penalty parameter C , which determines the trade-off between maximizing the margin and minimizing the training error.

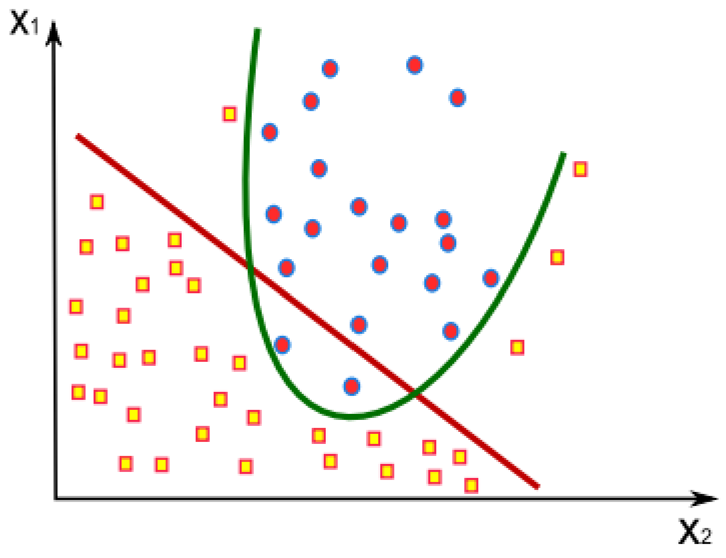
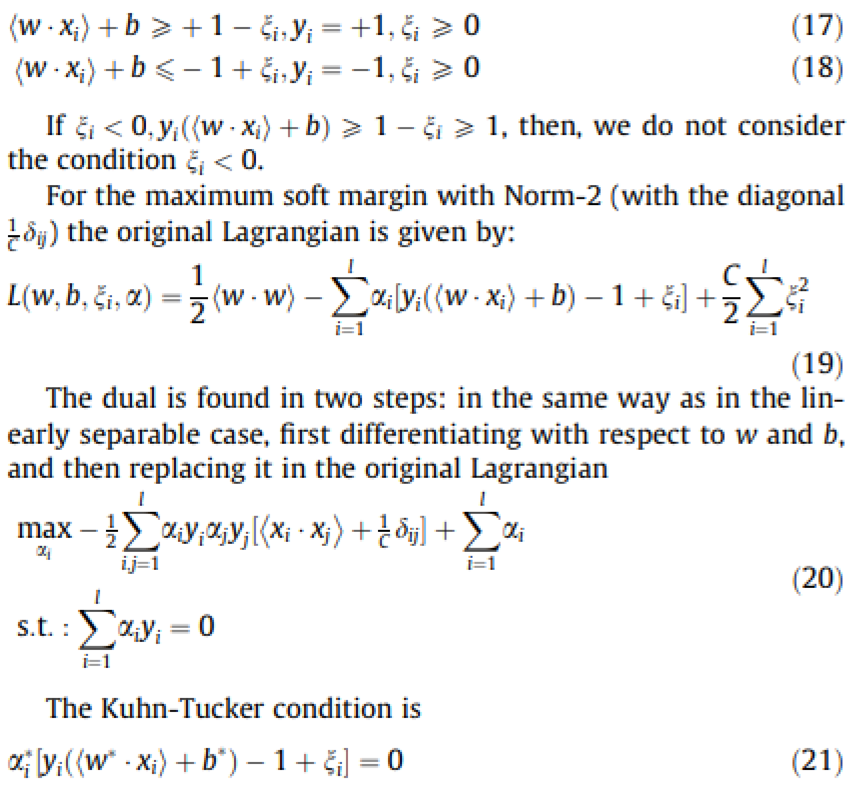


Fig. 4. Non linearly classifier. [11]

A learning machine's capacity for a given set of functions is characterised by the VC dimension. In the context of binary classification, h represents the greatest number of points that, through the application of the learning machine's functions, can be divided into two classes in all 2^h conceivable ways.

In the context of SVMs, selecting a large C results in a modest number of classification errors but may produce a big ||w||^2. It is noteworthy that the value of C = 1 indicates that there should be no misclassified data points. This would not always be possible, though, as the issue might only be solved for some C < 1.

$$\begin{aligned} &\min_{w,b,\xi_i} \langle w \cdot w \rangle + C\sum_{i=1}^{l} \xi_i^2 \\ &\text{s.t.}: y_i(\langle w \cdot x_i \rangle + b) \geqslant 1 - \xi_i \\ &\xi_i \geqslant 0 \end{aligned} \tag{16}$$



$$\langle w \cdot x_i \rangle + b \geqslant +1 - \xi_i, y_i = +1, \xi_i \geqslant 0 \tag{17}$$

$$\langle w \cdot x_i \rangle + b \leqslant -1 + \xi_i, y_i = -1, \xi_i \geqslant 0 \tag{18}$$

If $\xi_i < 0, y_i(\langle w \cdot x_i \rangle + b) \geqslant 1 - \xi_i \geqslant 1$, then, we do not consider the condition $\xi_i < 0$.

For the maximum soft margin with Norm-2 (with the diagonal $\frac{1}{C}\delta_{ij}$) the original Lagrangian is given by:

$$L(w, b, \xi_i, \alpha) = \frac{1}{2}\langle w \cdot w \rangle - \sum_{i=1}^{l} \alpha_i [y_i(\langle w \cdot x_i \rangle + b) - 1 + \xi_i] + \frac{C}{2}\sum_{i=1}^{l} \xi_i^2 \tag{19}$$

The dual is found in two steps: in the same way as in the linearly separable case, first differentiating with respect to w and b, and then replacing it in the original Lagrangian

$$\begin{aligned} &\max_{\alpha_i} -\tfrac{1}{2}\sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j [\langle x_i \cdot x_j \rangle + \tfrac{1}{C}\delta_{ij}] + \sum_{i=1}^{l} \alpha_i \\ &\text{s.t.} : \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned} \tag{20}$$

The Kuhn-Tucker condition is

$$\alpha_i^* [y_i(\langle w^* \cdot x_i \rangle + b^*) - 1 + \xi_i] = 0 \tag{21}$$

[12]

In practice, the quadratic optimization problem for SVMs with a soft margin is very similar to the one for the separable case, with the main difference being the adjusted values of the Lagrange multipliers ai due to the introduction of the slack variables ni. The parameter C is a user-defined parameter that controls the trade-off between having a smaller margin and misclassifying training points.

Choosing an appropriate C is typically done experimentally using cross-validation techniques. Cross-validation involves splitting the dataset into multiple subsets, training the model on one subset, and validating it on another. This process helps to find the C value that provides the best balance between model complexity and generalization performance.

2.3 Kernels

In an SVM, the optimal hyperplane is chosen to maximize the model's ability to generalize to unseen data. However, when the training data are not linearly separable, the classifier may not have high generalization ability even if the hyperplanes are optimally determined to maximize the margin between classes. In such cases, the original input space is transformed into a higher-dimensional space called the "feature space."(J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., 2019 p. 4)

The feature space transformation is achieved by using a kernel function, which implicitly maps the original input space into a higher-dimensional space where the data may be more easily separable. This allows the SVM to find a hyperplane that separates the classes with a larger margin in the transformed space, even if the classes are not linearly separable in the original space. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels.

The basic idea in designing non-linear SVMs is to transform the input vectors $x \in \mathbb{R}^n$ into vectors $\Phi(x)$ of a highly dimensional feature space [30] F (where Φ represents the mapping: $\mathbb{R}^n \to \mathbb{R}^f$) and solve the problem of linear classification in this feature space

$$x \in \mathbb{R}^n \to \Phi(x) = [\phi_1(x), \phi_2(x), \ldots, \phi_n(x)]^T \in \mathbb{R}^f \qquad (22)$$

The set of hypotheses considered will be

$$f(x) = \sum_{i=1}^{l} w_i \phi_i(x) + b \qquad (23)$$

[13]

The learning procedure involves two steps: first, a non-linear mapping transforms the data into the feature space F, and then a Support Vector Machine (SVM) is used to classify the data in this feature space. One important property of linear learning machines is that they can be represented in a dual form. This means that the decision rule can be expressed as a linear combination of the training data points. As a result, the decision rule can be evaluated using dot products between the input data points.

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i \langle \phi(x_i) \cdot \phi(x) \rangle + b \qquad (24)$$

[14]

If it's possible to directly compute the product <ϕ(xi).ϕ(x)> in the feature space as a function of the original input data, this allows us to combine the two essential steps in constructing a non-linear learning machine. This direct computation method is referred to as a kernel function.(J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., 2019 p. 4)

$$K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle \tag{25}$$

where ϕ is a mapping of $X$to a feature space F.

The key to the approach is to find a kernel function that can be evaluated efficiently. Once we have such a decision function, the rule can be evaluated

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i K\langle x_i \cdot x_j \rangle + b \tag{26}$$

A kernel function must respect the following properties: for any $x, y, z \in X$ and $\alpha \in R$

1. $x \cdot x = 0$ only if $x = 0$
2. $x \cdot x > 0$ otherwise
3. $x \cdot y = y \cdot x$
4. $(\alpha x \cdot y) = \alpha(x \cdot y)$
5. $(z + x) \cdot y = (z \cdot y) + (x \cdot y)$

[15]

Kernel functions used in SVMs must satisfy the condition given by Mercer's theorem. Some of the most commonly used kernel functions include:

1. **Linear kernel:** $K(x_i, x_j) = (x_i \cdot x_j)$;
2. **Polynomial kernel:** $K(x_i, x_j) = (x_i \cdot x_j + 1)^p$;
3. **Gaussian kernel:** $K(x_i, x_j) = e^{\frac{-\|x_i - x_j\|^2}{2\sigma^2}}$;
4. **RBF kernel:** $K(x_i, x_j) = e^{-\gamma(x_i - x_j)^2}$;
5. **Sigmoid kernel:** $K(x_i, x_j) = tanh(\eta x_i \cdot x_j + v)$;

[16]

It is computationally advantageous that we can compute the dot product in the feature space without explicitly mapping the data to that space thanks to these kernel functions.

The SVM's performance in real-world applications can be greatly impacted by the kernel selection. Because it works well in a variety of applications, the Gaussian RBF kernel is the most widely used. Nonetheless, the selection of a kernel is contingent upon the particular features of the data, and in order to attain satisfactory outcomes, it is frequently imperative to ascertain the ideal parameters for the selected kernel.

Normally, in order to complete the theoretical part and understand it better, we need to open the kernel function with Mercers theorem and look at the non-linearly separable case more theoretically. However, since we have explained the mathematics behind SVM sufficiently for the review, we will look at its strengths, weaknesses, trends and usage areas in the following section. Finally, the table below shows a summary of the four main kernels.

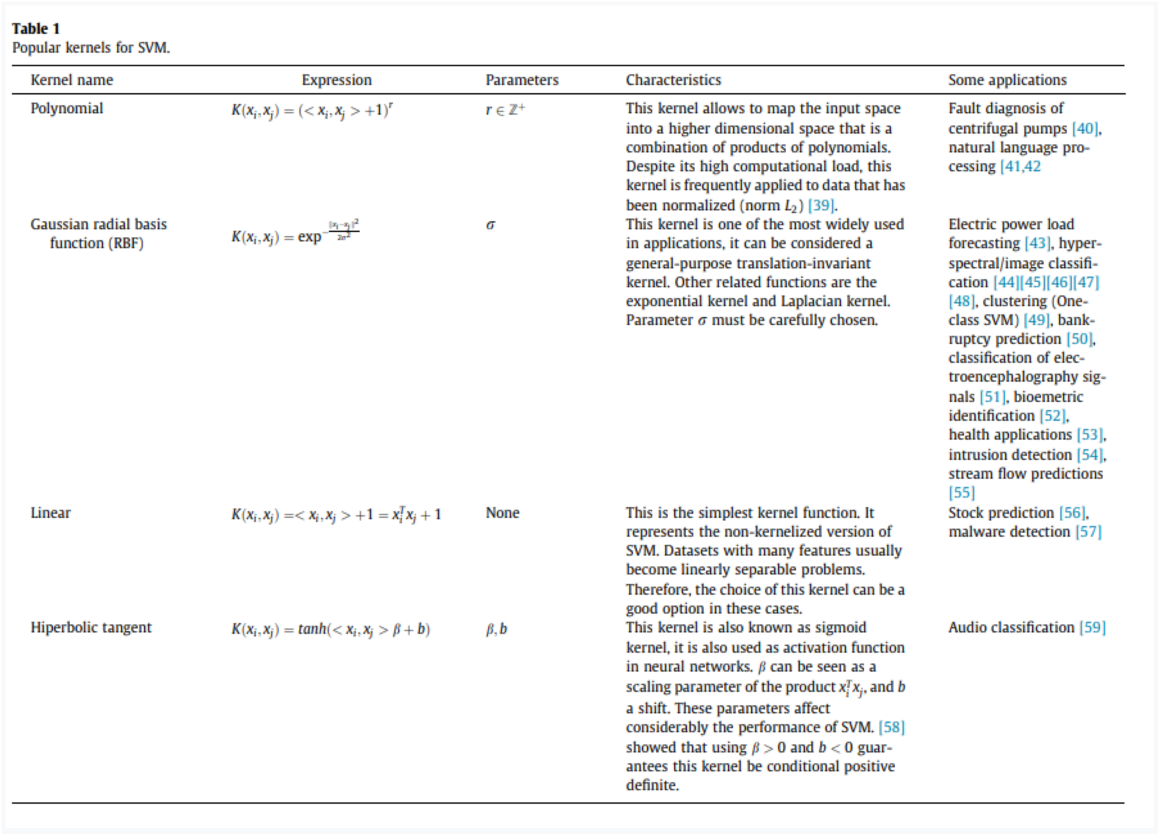


Table 1
Popular kernels for SVM.

Kernel name	Expression	Parameters	Characteristics	Some applications
Polynomial	$K(x_i,x_j) = (< x_i,x_j > +1)^r$	$r \in \mathbb{Z}^+$	This kernel allows to map the input space into a higher dimensional space that is a combination of products of polynomials. Despite its high computational load, this kernel is frequently applied to data that has been normalized (norm L_2) [39].	Fault diagnosis of centrifugal pumps [40], natural language processing [41,42
Gaussian radial basis function (RBF)	$K(x_i,x_j) = \exp^{-\frac{\|x_i-x_j\|^2}{2\sigma^2}}$	σ	This kernel is one of the most widely used in applications, it can be considered a general-purpose translation-invariant kernel. Other related functions are the exponential kernel and Laplacian kernel. Parameter σ must be carefully chosen.	Electric power load forecasting [43], hyperspectral/image classification [44][45][46][47][48], clustering (One-class SVM) [49], bankruptcy prediction [50], classification of electroencephalography signals [51], bioemetric identification [52], health applications [53], intrusion detection [54], stream flow predictions [55]
Linear	$K(x_i,x_j) =< x_i,x_j > +1 = x_i^T x_j + 1$	None	This is the simplest kernel function. It represents the non-kernelized version of SVM. Datasets with many features usually become linearly separable problems. Therefore, the choice of this kernel can be a good option in these cases.	Stock prediction [56], malware detection [57]
Hiperbolic tangent	$K(x_i,x_j) = tanh(< x_i,x_j > \beta + b)$	β, b	This kernel is also known as sigmoid kernel, it is also used as activation function in neural networks. β can be seen as a scaling parameter of the product $x_i^T x_j$, and b a shift. These parameters affect considerably the performance of SVM. [58] showed that using $\beta > 0$ and $b < 0$ guarantees this kernel be conditional positive definite.	Audio classification [59]

[17]

3. Weaknesses and Strengths of SVM

J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., (2019) highlighted several weaknesses of Support Vector Machines (SVMs), despite their generalization capacity and many advantages. These weaknesses include the selection of parameters, algorithmic complexity affecting the training time of the classifier in large datasets, development of optimal classifiers for multi-class problems, and the performance of SVMs in unbalanced datasets.

Support Vector Machines (SVM) are primarily flawed due to their high computational overhead, particularly when working with large datasets. The main cause of this is that SVM training times on big datasets are sluggish because the training kernel matrix grows quadratically with dataset size. Strong theoretical foundations and good generalisation capabilities notwithstanding, SVM is not a good choice for classification of large datasets. In order to find a separation hyperplane, SVM training is typically formulated as a quadratic programming (QP) problem involving a density matrix n x n, where n is the number of points in the dataset. The training complexity of SVM is highly dependent on the size of the dataset as a result, resulting in significant computational time and memory requirements for large datasets. SVMs can be computationally intensive, especially with large datasets. Data selection methods aim to reduce the dataset size by focusing on instances that are crucial for defining the separation hyperplane (support vectors). These methods include simple random sampling and more sophisticated approaches that update instance probabilities based on misclassifications. Distance-based methods measure the distance between instances and the hyperplane, with metrics like Euclidean and Mahalanobis distances. Clustering methods and statistics-based approaches are also used to reduce the computational burden of SVMs on large datasets, but they may have limitations in handling noisy data or maintaining the sparse property of SVMs.Decomposition methods aim to speed up Support Vector Machine (SVM) training by focusing on active constraints of the Quadratic Programming (QP) problem. These methods, like Chunking and Sequential Minimal Optimization (SMO), iteratively solve smaller QP problems. Other approaches, such as parallel optimization and genetic programming, also help handle large datasets. Variants like Least Squares SVM (LS-SVM) modify the QP problem formulation to trade off speed for accuracy. But all these methods are kind of like a trade-off. As a result, we can say that SVM's inadequacy in large data sets is its biggest disadvantage compared to other methods. Furthermore, accurately classifying objects belonging to minority classes can be a difficult problem in data sets that are imbalanced. For these skewed data sets, standard classification techniques like support vector machines perform poorly because it is challenging to obtain the ideal separation hyperplane for an SVM trained with unbalanced data. Vapnik introduced Support Vector Machines (SVMs) as a kernel-based machine learning model for regression and classification applications. In recent years, theorists and practitioners have become more interested in them due to their capacity for discrimination and generalisation. SVMs have solid theoretical underpinnings and, in practical applications, typically show high classification accuracy. SVM performance, however, dramatically declines when applied to imbalanced datasets, especially when the ratio between the majority and minority class is high, according to recent experiments. When applied to unbalanced datasets, SVMs have the drawback of producing margins that are skewed in favour of the minority class. Over the past ten years, the machine learning community has been studying classification for imbalanced data sets. Various techniques are utilised for unbalanced data sets in an effort to enhance classifier performance. These techniques are typically separated into two groups: internal and exterior techniques. Preprocessing training data sets using external methods ensures that they are balanced. Internal techniques work with altering the learning algorithms to lessen their sensitivity to unequal class distribution. Stated differently, internal approaches take into account the costs associated with misclassification and incorporate these costs into the model, whereas external methods consider the amount of samples for each class in an attempt to balance the data sets. It is nearly impossible to choose randomly instances that aid in improving performance in this area without using a genetic algorithm due to the large search space. While creating new instances in the minority class can improve the performance in SVM classification, this process may introduce noise into the data set.

Despite all this Support Vector Machines (SVMs) are widely recognized for their robust performance in various machine learning tasks. One of their key strengths lies in their ability to handle high-dimensional data effectively. SVMs aim to find the hyperplane that best separates different classes in the feature space while maximizing the margin between the classes. This margin maximization leads to a model that generalizes well to unseen data, making SVMs less prone to overfitting compared to other algorithms. Additionally, SVMs can handle non-linear decision boundaries by using kernel functions to map the input data into a higher-dimensional space where the classes become separable. This flexibility allows SVMs to capture complex patterns in the data, making them suitable for a wide range of applications, including text categorization, image recognition, and bioinformatics. Despite their computational complexity, SVMs are highly effective in scenarios where the data is well-structured and the goal is to achieve high classification accuracy.

4. SVM implementations and real-world problems

Support Vector Machines (SVMs) are powerful machine learning models used for classification and regression tasks. They are particularly effective in high-dimensional spaces and when the number of dimensions is greater than the number of samples. SVMs work by finding the hyperplane that best separates different classes in the input data.

The main advantage of SVMs lies in their ability to find the optimal hyperplane that maximizes the margin between classes, which helps in generalizing well to unseen data. This margin maximization is achieved by identifying a subset of training examples, known as support vectors, which lie closest to the hyperplane. However, SVMs can be computationally expensive, especially when dealing with large datasets. The computational complexity of SVMs is almost cubic, meaning that training time increases rapidly with the size of the dataset. To address this challenge, several approaches have been proposed:

1. Data Reduction: SVM solutions often rely on a small subset of data points (support vectors). By eliminating data points that are less likely to be support vectors, the training time can be reduced.

2. Chunking: This approach breaks down the SVM optimization problem into smaller, more manageable chunks. Each chunk is solved iteratively, and the process continues until all data points are considered. This reduces the complexity of the problem and speeds up training.

3. Decomposition Methods: These methods optimize a subset of variables in each iteration, known as the working or active set, while keeping the remaining variables fixed. This reduces memory requirements and improves efficiency, but careful selection of the active set is crucial.

4. Sequential Minimal Optimization (SMO): SMO is a popular algorithm for training SVMs that optimizes only two points in each iteration, making it computationally efficient for large datasets. It reduces the time spent on kernel computations, leading to faster convergence.

5. Shrinking Heuristics and Working Set Selection: These techniques further speed up the optimization process by reducing the number of kernel values needed to update the gradient vector and selecting an initial set of variables for optimization that leads to faster convergence.

In summary, SVMs are powerful models with strong theoretical foundations. While they can be computationally expensive, especially for large datasets, various techniques can be used to improve their training time and efficiency, making them suitable for a wide range of real-world applications. Support Vector Machines (SVMs) have found widespread applications in various real-world problems, particularly in text and hypertext categorization. Text categorization has been successfully handled by Support Vector Machines (SVMs), which exhibit reliable performance in a variety of applications. One method involves training SVMs with the Euclidean distance during the classification phase and the SVM approach during the training phase. During training, support vectors for every category are found, and the category of support vectors that is closest to the new data point is used to make classification decisions. SVMs have demonstrated excellent efficiency in learning from well-organized samples of moderate size when evaluated alongside other machine learning techniques in Chinese document categorization. These examples highlight the utility and adaptability of SVMs in addressing a variety of challenges in text categorization tasks, demonstrating their efficacy in various aspects of text classification. SVMs are widely used in text and image classification. In text categorization, they are effective in identifying support vectors for a particular category, leading to classification decisions. SVMs excel in Chinese document categorization, particularly with well-organized samples. In text classification, term-frequency transformations often impact SVM performance more than the choice of kernel. SVMs are employed in active learning to reduce data labeling efforts, especially in batch mode active learning for text classification. Using linear SVM with distributional clustering of words reduces the dimensionality of text documents without compromising classification performance. SVMs have also been successfully applied in very short document term weighting, active learning for text classification, and various other text categorization tasks. In image classification, SVMs are used alongside other techniques like median filters and feature extraction methods for tasks such as kidney stone image analysis, facial expression recognition, and hyperspectral image classification. Overall, SVMs are a versatile tool in both text and image analysis tasks, offering efficient solutions for various real-world problems. Support vector machines (SVM) are also used in bioinformatics, mainly for protein and cancer classification. SVM is used for semi-supervised feature selection in protein classification; compared to supervised methods, it achieves higher accuracy and requires less processing time. SVM is used in cancer classification to select and classify genes and to find important features in data on prostate cancer gene expression. Moreover, SVM and deep neural networks are combined to increase classification accuracy. SVM is used in the field of handwritten character recognition to recognise Arabic characters and digits as well as hierarchical classification schemes in Bangla characters. Further applications of SVM include offline writer identification and online handwritten character recognition services. When compared to other classifiers, including neural networks, k-NN, and decision trees, SVM's effectiveness is demonstrated in a number of applications with encouraging outcomes.

Also, SVM is used in an ensemble setting for face detection in digital videos to enhance detection performance. A locality-sensitive SVM is used in another method to provide robustness against external factors such as changing viewpoints and intense lighting. A multiclassifier system based on domain adaptation and multiple face representation improves robustness to intra-class variations in video surveillance. SVM is used to classify tasks in the fields of protein fold and remote homology detection. For instance, Golgi protein classification and feature ranking both use SVM in conjunction with random forest. In an additional work, SVM is applied to an ensemble-based transfer learning model for membrane protein discrimination with the goal of minimising data constraints in computational modelling. SVM shows its adaptability and efficiency in handling challenging issues in protein structure analysis and face detection. Moreover, SVM is utilised in surge control strategies for centrifugal compressors in the field of generalised predictive control in order to boost efficiency. The compressor's mass flow and discharge pressure are predicted using nonlinear model predictive control based on least-squared SVM. SVM has been used for complicated classification problems in a variety of fields, including the classification of plant species, the detection of credit card fraud, and the classification of melanoma. SVM is used to analyse images of plants in both controlled and uncontrolled environments in order to classify different plant species. Unbalanced data sets present problems for SVM in credit card fraud detection, necessitating the development of new algorithms to correct the imbalance. Despite its computational cost, SVM is used in computer vision analysis of dermatoscopic images for early diagnosis in melanoma classification.

Overall, SVM demonstrates its versatility in solving complex classification problems, although further research is needed to enhance its performance in certain applications.

5. Trends and challenges

There are several difficult issues with applying Support Vector Machines (SVM) to multi-class problems, small data sets, dynamic environments, and sparsely tagged data. Among these difficulties are:

1. Consolidated SVM:The direct application of SVM to multi-class problems is restricted by its traditional formulation, which is meant for binary classification. A number of strategies, including one-against-all and one-against-one tactics, as well as more modern techniques that build a piecewise-nonlinear classification function or map classes to vertices of a simplex, have been put forth to expand SVM for multi-class classification.
2.Multi-task SVM: Multi-task learning (MTL) attempts to learn several related tasks simultaneously, with the assumption that this joint learning can outperform independent learning of the tasks. The majority of current SVM techniques use multiple multitask binary problems, which may overlook class relationships.
3.Large-scale issues: SVM training requires resolving quadratic programming (QP) issues, which can require a lot of processing power when dealing with big data sets. Under-sampling, in which a subset of significant samples is chosen, and employing clustering or heuristic techniques to lower the number of support vectors are two techniques for addressing large-scale issues.
4.On-line SVM: SVMs must be easily and quickly updated for data streams that contain concept drift. To tackle these issues, techniques like Representative Prototype Areas (RPA) and learning prototypes from data streams have been suggested.
5.Kernel Selection and Parameter Optimisation: The performance of SVMs depends on the choice of kernel and the optimisation of its parameters. Hyperparameter optimisation techniques include grid search, class separability analysis, and global optimisation models.

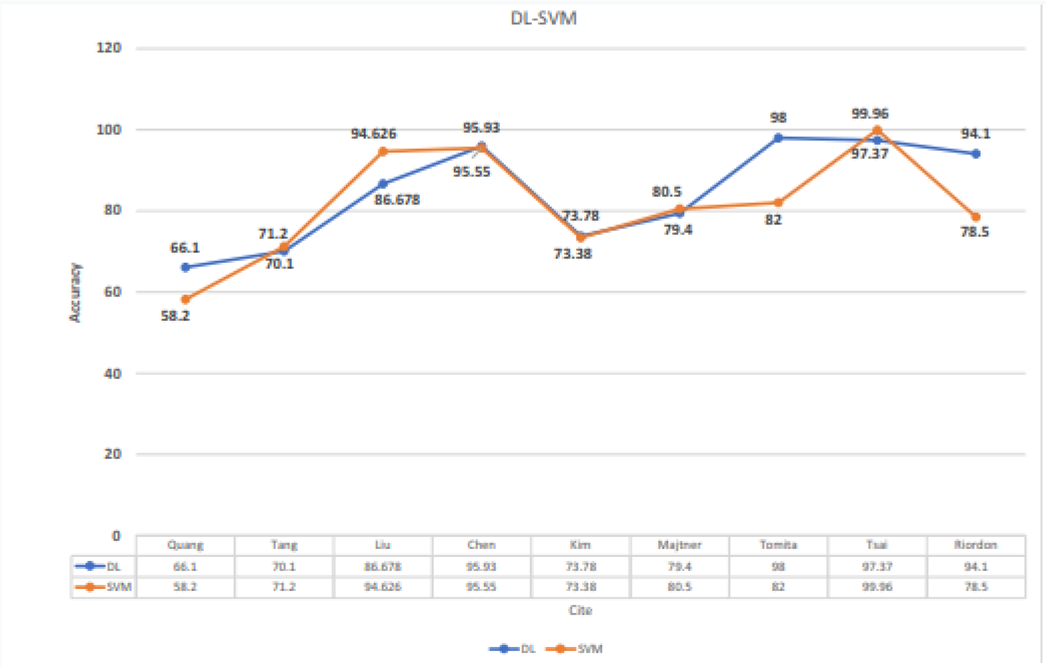
6.Transductive SVM (TSVM) and other semi-supervised techniques: To handle partially labelled data sets more efficiently, semi-supervised techniques such as TSVM and algorithms that maximise the margin distribution of TSVM have been suggested.

While these methods have shown promise in improving SVM performance, further research is needed to develop more efficient solutions to the challenges posed by SVM in various applications.

5.1 Deep learning vs SVM

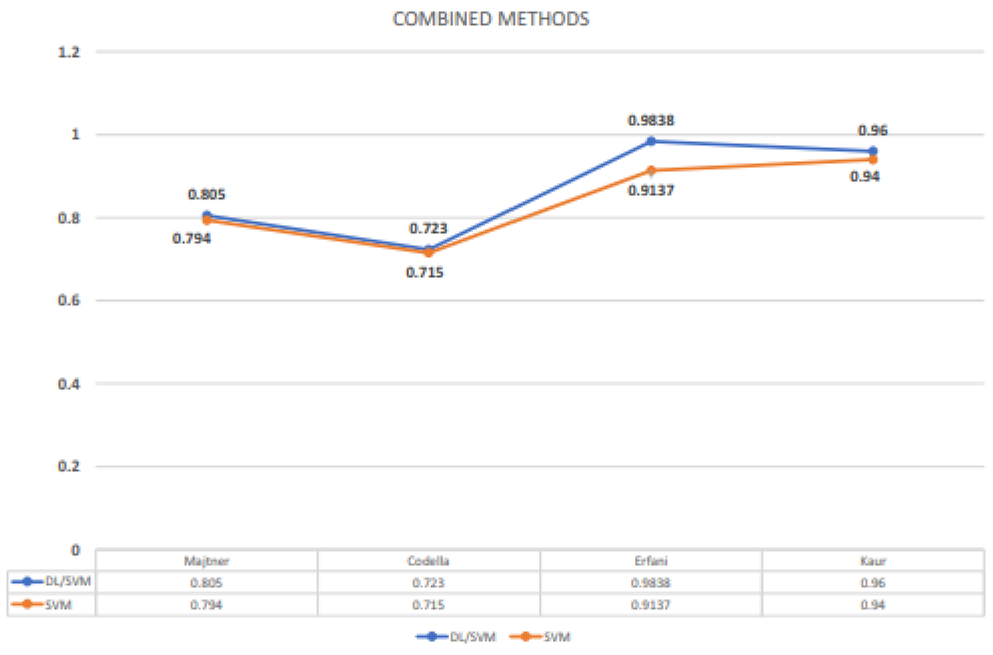
Deep learning has gained popularity due to its ability to model high-level abstractions in data using computational architectures that support multiple non-linear and iterative transformations of data. It has shown high success rates, especially with unsupervised training, and has been applied in various fields such as healthcare, finance, speech recognition, augmented reality, digital image processing, and 3D and video applications.

Comparative performance studies between deep learning and Support Vector Machines (SVM) have been conducted by several authors. Figure 5 in the text illustrates these comparisons. In many cases, deep learning outperforms SVM, especially in tasks involving complex data patterns. However, some authors have proposed combining deep learning with SVM to improve performance further. This combination involves training one of the SVM classes based on features learned by a convolutional neural network (CNN). This allows for the replacement of linear SVM kernels with non-linear ones without losing precision.



	Quang	Tang	Liu	Chen	Kim	Majtner	Tomita	Tsai	Riordon
DL	66.1	70.1	86.678	95.93	73.78	79.4	98	97.37	94.1
SVM	58.2	71.2	94.626	95.55	73.38	80.5	82	99.96	78.5

Fig. 5. Performance of SVM vs Deep Learning.



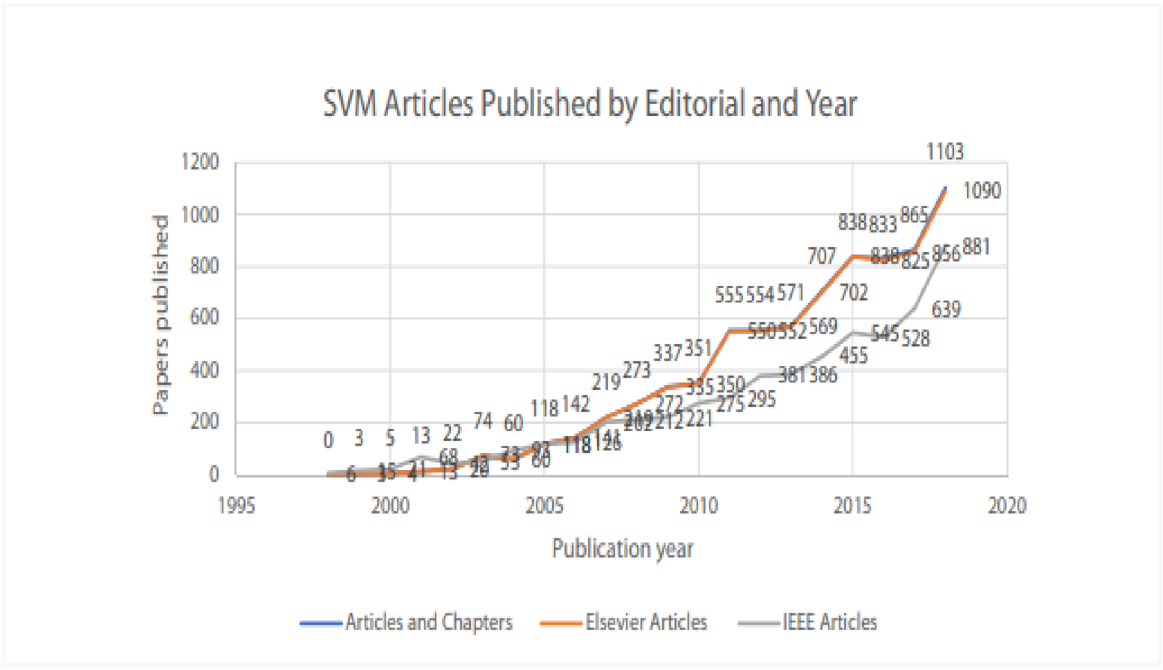
	Majtner	Codella	Erfani	Kaur
DL/SVM	0.805	0.723	0.9838	0.96
SVM	0.794	0.715	0.9137	0.94

[18]

Overall, the studies suggest that SVM and deep learning have similar average performances, and when used together, they can synergistically enhance performance across different applications.

5.2 Impact of SVM

The impact of Support Vector Machines (SVM) in academic literature was analyzed, focusing on research papers published from 1998 to 2018. The distribution of SVM-related research papers by year is illustrated in below figure . The analysis revealed that the majority of SVM papers address problems related to normal-sized and balanced datasets, where SVM performs well. Using ScienceDirect and IEEE Xplore search engines, over 13,000 publications containing the term "SVM" were retrieved from journals. To identify applications of SVM in large and imbalanced datasets, publications meeting specific criteria were selected, including being published in journals or book chapters, and containing relevant search terms in the title, abstract, or keywords.



[19]

6. Conclusions

The strong theoretical foundations and strong generalisation ability of Support Vector Machines (SVMs) have led to their widespread implementation in a wide range of real-world applications. Text classification, protein fold and remote homology detection, image classification, bioinformatics (protein and cancer classification), handwritten character recognition, face detection, generalised predictive control, and other domains are among those in which they have been used. Studies have shown that SVMs frequently perform better than alternative classification methods.

Nevertheless, SVMs are not without limitations when it comes to handling multiclass data sets, choosing parameters, algorithmic complexity, and handling imbalanced data sets. Notwithstanding these drawbacks, SVMs' strong theoretical underpinnings and strong generalisation capabilities have allowed them to be applied successfully to a wide range of real-world classification issues. Because of their potentially long training times, support for very large data sets is less common for SVMs. Additionally, SVM accuracy may suffer from imbalanced data sets. To overcome these obstacles, a number of strategies have been put forth, such as specific algorithms meant to boost SVM performance in the face of these constraints.

This paper describes algorithms designed to address the main drawbacks of SVM and offers a thorough discussion of them. It makes reference to the studies conducted by researchers who have tackled these implementation challenges with SVM.

References

1-19. J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua et al., A comprehensive survey on support vector machine classification: Applications, challenges and trends, Neurocomputing, https://doi.org/10.1016/j.neucom.2019.10.118