Support Vector Machines

Steven Busuttil

Department of Computer Science and AI,
University of Malta

Abstract. In this report we present an introductory overview of Support Vector Machines (SVMs). SVMs are supervised learning machines that can be analysed theoretically using concepts from computational learning theory while being able to achieve good performance when applied to real-world problems.

1 Introduction

The study of Support Vector Machines (SVMs) can be said to have been started by Vladimir Vapnik in the late seventies [15]. However it was only until the late nineties that the subject started to receive increasing attention [4, **?**,**?**].

Support Vector Machines, are supervised learning machines based on statistical learning theory that can be used for pattern recognition and regression. Statistical learning theory can identify rather precisely the factors that need to be taken into account to learn successfully certain simple types of algorithms, however, real-world applications usually need more complex models and algorithms (such as neural networks), that makes them much harder to analyse theoretically. SVMs can be seen as lying at the intersection of learning theory and practice. They construct models that are complex enough (containing a large class of neural networks for instance) and yet that are simple enough to be analysed mathematically. This is because an SVM can be seen as a linear algorithm in a high-dimensional space [13].

In this document, we will primarily concentrate on Support Vector Machines as used in pattern recognition. In the first section we will introduce pattern recognition and hyperplane classifiers, simple linear machines on which SVMs are based. We will then proceed to see how SVMs are able to go beyond the limitations of linear learning machines by introducing the kernel function, which paves the way to find a nonlinear decision function. Finally, we sum it all up and mention some areas in which Support Vector Machines have been applied and given excellent results.

2 Pattern Recognition and Hyperplane Classifiers

In pattern recognition we are given training data of the form

$$(\mathbf{x_1}, y_1), \ldots, (\mathbf{x}_\ell, y_\ell) \in \mathbb{R}^n \times \{+1, -1\}, \tag{1}$$

that is n–dimensional patterns (vectors) $\mathbf{x}_i$ and their labels y_i. A label with the value of $+1$ denotes that the vector is classified to class $+1$ and a label of -1 denotes that the vector is part of class -1. We thus try to find a function $f(\mathbf{x}) = y : \mathbb{R}^n \to \{+1, -1\}$ that apart from correctly classifying the patterns in the training data (a relatively simple task), correctly classifies unseen patterns too. This is called generalisation.

Statistical learning theory or VC (Vapnik-Chervonenkis) theory [16], shows that it is imperative that we restrict the class of functions that our machine can learn, otherwise learning the underlying function is impossible. It is for this reason that SVMs are based on the class of hyperplanes

$$\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0; \mathbf{w} \in \mathbb{R}^n, b \in \mathbb{R}, \tag{2}$$

which basically divide the input space into two: one part containing vectors of the class -1 and the other containing those that are part of class $+1$ (see Figure 1). If there exists such a hyperplane, the data is said to be linearly separable (nonseparable otherwise). To find the class of a particular vector $\mathbf{x}$, we use the following decision function

$$f(\mathbf{x}) = sign(\langle \mathbf{w} \cdot \mathbf{x} \rangle + b). \tag{3}$$

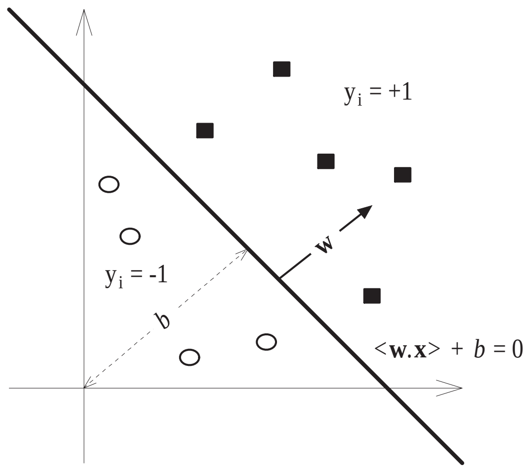

Fig. 1. A separating hyperplane $(\mathbf{w}, b)$ for a two dimensional (2D) training set.

2.1 The Optimal Hyperplane

As can be understood, there may be more than one hyperplane that correctly classifies the training examples (for instance, in Figure 1 the hyperplane could be closer to class -1). It has been shown that the hyperplane that guarantees the best generalisation performance is the one with the maximal margin of separation between the two classes [6, ?]. This type of hyperplane is known as the optimal or maximal margin hyperplane and is unique.

The optimal hyperplane can be constructed by solving a convex (no local minima, therefore any solution is global) optimisation problem that is minimising a quadratic function under linear inequality constraints. The solution to this problem has an expansion in terms of a subset of the training examples that lie on the margin, called support vectors (see Figure 2). Support vectors contain all the information needed about the classification problem, since even if all the other vectors are removed the solution will still be the same. The details of the calculations will be omitted but can be found in a number of our references (see for instance [6]).

Finally, another very important property of hyperplane classifiers that needs to be emphasised, is that both the optimisation problem (used to find the optimal hyperplane) and the decision function

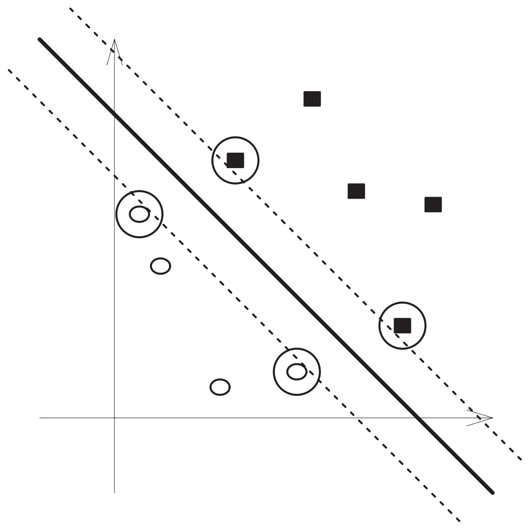

Fig. 2. A maximal margin hyperplane with its support vectors encircled.

(used for the actual classification of vectors) can be expressed in dual form which depend only on dot products between vectors. The dual representation of the decision function is

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{\ell} y_i \alpha_i \langle \mathbf{x} \cdot \mathbf{x}_i \rangle + b\right), \tag{4}$$

where $\alpha_i \in \mathbb{R}$ is a real-valued variable that can be viewed as a measure of how much informational value $\mathbf{x}_i$ has. Thus for vectors that do not lie on the margin (i.e. non support vectors) this value will be zero.

3 Feature Spaces and Kernels

Linear learning machines (such as the hyperplane classifier), while being mathematically compelling because of their ease of analysis, have limited computational power and thus limited real-world value [9]. In general, complex real-world applications require a learning machine with much more expressive power.

One proposed solution to this problem was to create a network of simple linear classifiers (in the form of neural networks) and thus be able to represent nonlinear decision surfaces. However, neural networks have a number of inherent problems, including local minima and many tunable parameters. In addition, it is very complex to analyse a neural network mathematically.

Another solution is to map the input vectors into a richer (usually high-dimensional) feature space where they are linearly separable using a nonlinear mapping ϕ. In feature space, build a separating hyperplane using a well-understood linear learning machine such as the optimal hyperplane classifier (see Figure 3). This yields a nonlinear decision surface in input space and is the approach taken by Support Vector Machines.

As we have already noted in Section 2.1, the optimal hyperplane classifier uses only dot products between vectors in input space. In feature space this will translate to $\langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle$. Clearly, this is

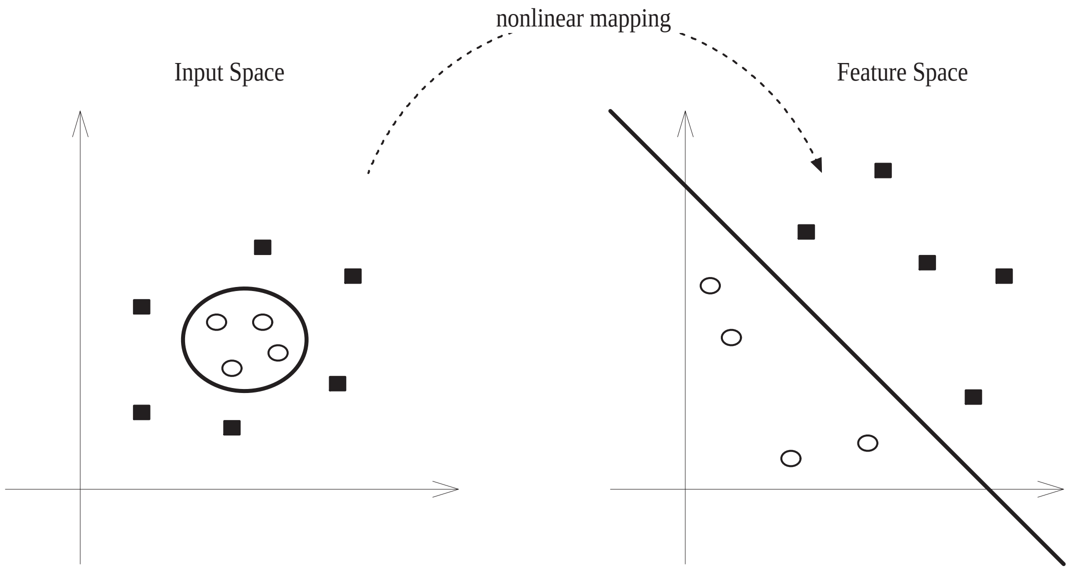

Fig. 3. A nonlinear mapping from input space to feature space can simplify the classification task.

very computationally expensive, especially if the mapping is to a high-dimensional space. Boser, Guyon and Vapnik [2], showed that a rather old trick [1]—kernel functions—can be used to accomplish the same result in a very simple and efficient way. A kernel is a function $k(\mathbf{x}, \mathbf{y})$ that given two vectors in input space, returns the dot product of their images in feature space

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{y}) \rangle. \tag{5}$$

There are several different kernels, choosing one depends on the task at hand. One of the simplest is the polynomial kernel $k(\mathbf{x}, \mathbf{y}) = \langle \mathbf{x} \cdot \mathbf{y} \rangle^d$. For example, taking $d = 2$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$

$$\begin{aligned} \langle \mathbf{x} \cdot \mathbf{y} \rangle^2 &= (x_1 y_1 + x_2 y_2)^2 \\ &= (x_1 y_1 + x_2 y_2)(x_1 y_1 + x_2 y_2) \\ &= (x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2) \\ &= (x_1^2, x_2^2, \sqrt{2} x_1 x_2)(y_1^2, y_2^2, \sqrt{2} y_1 y_2) \\ &= \langle \phi_1(\mathbf{x}) \cdot \phi_1(\mathbf{y}) \rangle \end{aligned} \tag{6}$$

defining $\phi_1(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)$.

4 Support Vector Machines

Support Vector Machines are nothing more (or less) than linear learning machines expressed in dual form that map their input vectors to a feature space by the use of kernels and compute the optimal hyperplane there.

If we take Equation 4, which is the decision function for the optimal hyperplane classifier in dual form and apply the mapping ϕ to each vector it uses, we will get

$$f(\mathbf{x}) = sign \left(\sum_{i=1}^{\ell} y_i \alpha_i \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) \rangle + b \right). \tag{7}$$

As already mention in Section 3 the explicit mapping to feature space is not desirable, since it is very computational expensive. We will therefore use kernels which will give us a nonlinear decision function of the form

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{\ell} y_i \alpha_i k(\mathbf{x}, \mathbf{x}_i) + b\right). \tag{8}$$

The SVM algorithm is thus based on statistical learning theory, while being practical since it reduces to an optimisation problem with a unique solution. Up to now we have only considered the case of classification (pattern recognition). A generalisation to regression, that is, having $y \in \mathbb{R}$, can be given. In this case, the algorithm tries to construct a linear function in the feature space such that the training points lie within a distance of $\varepsilon > 0$. Similar to the pattern-recognition case, this can be written as a quadratic programming problem in terms of kernels [13] (see [6] for details).

5 Final Remark

Support Vector Machines have been applied to many real-world problems, producing state-of-the-art results. These include text categorisation [7, 8], image classification [5, 10–12], biosequence analysis and biological data mining [3] and handwritten character recognition [2].

References

1. Aizerman, M. A., Braveman, E. M. and Rozoner, L. I. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control,* 25:821–837, 1964.
2. Boser, B. E., Guyon, I. M. and Vapnik, V. A training algorithm for optimal margin classifiers. *In Fifth Annual Workshop on Computational Learning Theory,* Pittsburgh, 1992. ACM.
3. Brown, M., Grundy, D., Lin, D., Christianini, N., Sugnet, C., Furey, T., Ares, M and Haussler, D. *Knowledge-based analysis of microarray gene expression data using support vector machines.* Technical report, University of California in Santa Cruz, 1999.
4. Burges, C. J. C. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery.* 2:121–167, Kluwer Academic Publishers, Boston, 1998.
5. Chapelle, O., Haffner, P. and Vapnik, V. SVMs for histogram-based image classification. *IEEE Transaction on Neural networks,* 1999.
6. Cristianini, N. and Shawe-Taylor, J. *An introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.
7. Dumais S., Platt, J., Heckerman, D. and Sahami, M. Inductive learning algorithms and representations for text categorisation. In *7th International Conference on Information and Knowledge Management,* 1998.
8. Joachims, T. Text categorisation with support vector machines, In *Proceedings of European Conference on Machine Learning (ECML),* 1998.
9. Minsky, M. L. and Papert, S. A. *Perceptrons.* MIT Press, 1969. Expanded Edition 1990.
10. Oren, M., Papageorgiou, Sinha, P., Osuna, E. and Poggio, T. Pedestrian detection using wavelet templates. In *Proceedings of Computer Vision and Pattern Recognition,* 193–199, 1997.
11. Osuna, E., Freund, R. and Girosi, F. Training support vector machines: An application to face detection. In *Proceedings of Computer Vision and Pattern Recognition,* 130–136, 1997.
12. Pontil, M. and Verri, A. Object recognition with support vector machines. *IEEE Transaction on PAMI,* 20:637–646, 1998.
13. Schölkopf, B. SVMs—a practical consequence of learning theory. *Trends and Controversies—IEEE Intelligent Systems,* 18–21, 1998.

14. Schölkopf, B., Burges, C. J. C. and Smola, A. J. *Advances in Kernel Methods—Support Vector Learning.* MIT Press, Cambridge, 1999.
15. Vapnik, V. *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, Moscow, 1979. (English translation: Springer Verlag, New York, 1982.)
16. Vapnik, V. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, 1995.