Using Support Vector Machine as a Binary Classifier

Nikolay Stanevski, Dimiter Tsvetkov

Abstract: *This paper presents a relatively new and less known alternative of the classical neural network architectures – Support Vector Machines (SVM) and their usage for binary data classification. After a brief description of the Statistical Learning Theory – the framework of SVM, we explore the ways to build an error-tolerant binary classifier for linearly and non-linearly separated data. A comparison between SVM and some other neural network types is performed. To illustrate the paper, a demonstration software is provided.*

Key words: *Support Vector Machines (SVM), Neural Networks (NN), Artificial Intelligence (AI)*

INTRODUCTION

A resurgence of interest in neural networks has been observed since 80s. Together with well known architectures and training algorithms, like singlelayer and multilayer perceptron, trained with backpropagation algorithm, there is a plenty of other approaches. Support Vector Machine (SVM) is among the emerging approaches here. It can be used for many AI tasks (e.g. text categorisation, image recognition, gene expression etc. [3]).

The problem we will explore here is following. We have a set $\mathbf{S}$ of pairs $(\mathbf{x}_i, y_i)$, $i = 1,2,\dots,N$, where $\mathbf{x}_i \in \Re^n$ and every vector $\mathbf{x}_i$ is labelled to belong to any of two subclasses via y_i where y_i is a scalar which value can be either 1 or -1 indicating that the corresponding vector $\mathbf{x}_i$ belongs to a particular subclass C_1 or C_2. The goal of the paper is to design a SVM which performs such binary classification.

BINARY CLASSIFICATION FOR LINEARLY SEPARABLE DATA, USING MAX MARGIN CLASSIFIER

There are a lot of approaches to linearly separate two data classes. We can use, for example, a single or multi-layer perceptrons. Solutions of the problem (i.e. neuron weight vectors $\mathbf{w}$ and biases b) are an infinite set and depend on some factors like initial weights and biases, learning algorithm and its parameters etc. The problem here is to find the *only optimal margin* of the separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$, the one that provides maximum wide boundary between the classes ($\mathbf{w}^T\mathbf{x}$ stands for the dot product of the vectors $\mathbf{w}$ and $\mathbf{x}$). This margin guaranties the lowest rate of misclassification. As mentioned above, the dataset contains two classes of data.

Suppose the set $\mathbf{S}$ is strongly linearly separable. Then there exists uniformly separating hyperplane $\mathbf{w}^T\mathbf{x} + b = 0$ and a positive $\kappa > 0$ so that $\mathbf{w}^T\mathbf{x} + b \geq \kappa$ for $\mathbf{x} \in C_1$ and $\mathbf{w}^T\mathbf{x} + b \leq -\kappa$ for $\mathbf{x} \in C_2$ therefore via normalizing any uniformly separating hyperplane $\boldsymbol{\alpha}$ can be defined by the property

$$\mathbf{w}^T\mathbf{x} + b \geq 1 \text{ for } \mathbf{x} \in C_1 \text{ and } \mathbf{w}^T\mathbf{x} + b \leq -1 \text{ for } \mathbf{x} \in C_2. \qquad (1)$$

Given a separating hyperplane $\boldsymbol{\alpha}$ which satisfies conditions (1), for the distances between $\boldsymbol{\alpha}$ and the points $\mathbf{x} \in \mathbf{S}$ we have

$$dist(\mathbf{x}, \boldsymbol{\alpha}) = \frac{\left|\mathbf{w}^T\mathbf{x} + b\right|}{\|\mathbf{w}\|} \geq \frac{1}{\|\mathbf{w}\|}, \qquad (2)$$

therefore the optimal hyperplane is the one with maximal $1/\|\mathbf{w}\|$ that guaranties an optimal margin of width $2/\|\mathbf{w}\|$ because the equality in (2) is reached at least for one point $\mathbf{x}_1 \in C_1$ and at least one point $\mathbf{x}_2 \in C_2$.

Obviously maximizing that distance means minimizing following dot product

$$\frac{1}{2}\mathbf{w}^T\mathbf{w}\,. \tag{3}$$

On the other hand for all points we have

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0\,. \tag{4}$$

Put together, (3) and (4) represent a quadratic constrained optimization problem which can be stated as

$$\textit{minimize}\ \frac{1}{2}\mathbf{w}^T\mathbf{w}\,, \tag{5}$$

$$\textit{subject to}\ \ y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0\,,\ i = 1,2,\ldots,N\,.$$

One important note here is that the target function is convex. The solution of the optimization task for convex functions is always its global minima.

The problem, described by (5) is known as *primal problem*. This problem has its corresponding *dual problem*. One common way to solve it is by using of the Lagrangian function

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N}\alpha_i\left[y_i\left(\mathbf{w}^T\mathbf{x_i} + b\right) - 1\right], \tag{6}$$

where α_i are known as Lagrange multipliers. $L(\mathbf{w}, b, \alpha)$ reaches its minima at the point which satisfies both

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0\,,\ \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0\,, \tag{7}$$

together with the Kuhn-Tucker conditions [2]. Applying these conditions we reduce our problem to the solution of the *dual problem*

$$\textit{maximize}\ \left(\textit{with respect to}\ \alpha\right)\ \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i{}^T\mathbf{x}_j\,, \tag{8}$$

$$\textit{subject to}\ \sum_{i=1}^{N}\alpha_i y_i = 0\ \text{and}\ \alpha_i \geq 0\,,\ i = 1,2,\ldots,N\,. \tag{9}$$

After calculating α_i we can find the weight vector $\mathbf{w}$ and the bias b we need by the formulas

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i\,,\ b = -\sum_{i=1}^{N}\alpha_i\mathbf{w}^T\mathbf{x}_i \Bigg/ \sum_{i=1}^{N}\alpha_i\,. \tag{10}$$

The picture below shows how our demo SVM classified linearly separable data.

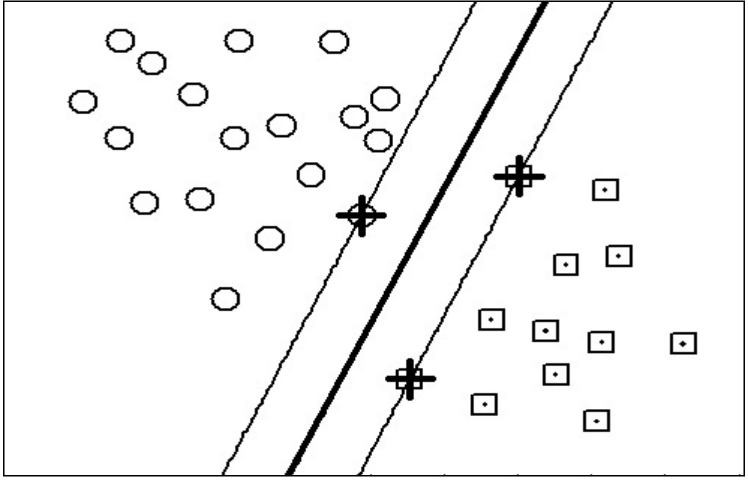

Figure 1*: SVM working on linearly separable data. Circles show one class, squares – the other. Support vectors are pointed out by crosses.*

There are two important properties we should note here.

- Only the points $\mathbf{x} \in \mathbf{S}$ with $dist(\mathbf{x}, \boldsymbol{\alpha}) = 1/\|\mathbf{w}\|$ (laying on the margin hyperplanes) have their corresponding α_i nonzero. These points are so-called *support vectors* (hence the name *Support Vector Machine*).
- The input patterns are always represented as a set of *dot products*. This will make it possible to easily switch to nonlinear SVMs as further described.

SOFT MARGIN SVM

The case described above is sometimes useless, because there are some factors (e.g. as a result of noise, fluctuations etc.) and it is not possible to find a hyperplane that strictly divides the data. Here, we can find an error-tolerant margin, called *soft margin*. To describe it, we first introduce the so-called *slack variables* ξ_i which measure the deviation between the real and the ideal position of each point. If $\xi_i = 0$ the point is on the right side, $\xi_i > 1$ means wrong side, and $0 < \xi_i \leq 1$ means the marginal band. Then (4) can be generalized as

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \ i = 1,2,...,N . \quad (11)$$

Now the primal problem is to minimize (with respect to $\mathbf{w}$) the following function

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i , \quad (12)$$

subject to the same constraints. The parameter C controls how much error-tolerant the machine is.

The corresponding dual problem does not use slack variables. The only difference here is that the Lagrange multipliers should satisfy $0 \leq \alpha_i \leq C$ instead of $\alpha_i \geq 0$ for each training sample $(\mathbf{x_i}, y_i)$. For the optimal weight and bias here we have

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i , \ b = \left(\sum_{i=1}^{N}(C - \alpha_i)\alpha_i y_i - \sum_{i=1}^{N}(C - \alpha_i)\alpha_i \mathbf{w}^T \mathbf{x}_i \right) \Big/ \sum_{i=1}^{N}(C - \alpha_i)\alpha_i . \quad (13)$$

The following picture is an illustration of this case.

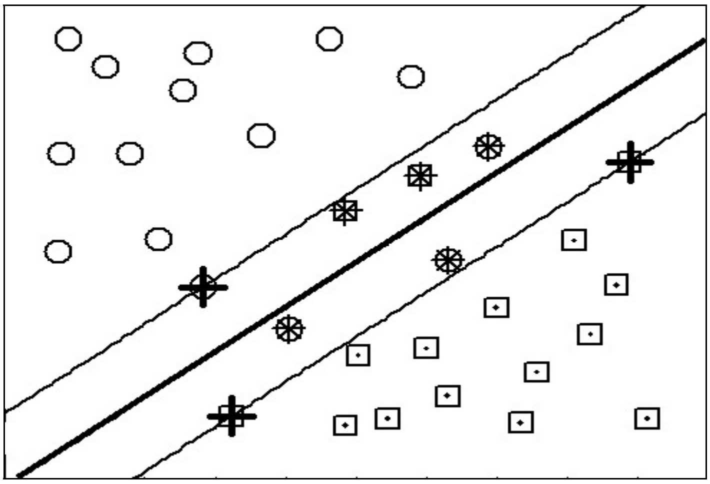

***Figure 2**: Soft margin SVM: Filled by stars samples are misclassified by the linear soft margin*

NONLINEAR BINARY CLASSIFCATION USING SVM

The most exciting property of SVM is the easy way it can switch from linear to nonlinear margin. It uses Cover's theorem [2], which states that input data, which are linearly non separable into input space I (under certain assumptions) can be mapped to another space (called feature space) F , in which data can be linearly separable.

The main problem here is to find a map function $\Phi : I \rightarrow F$. Given that fact, the dot product is now transformed as $\mathbf{x}_i{}^T \mathbf{x}_j \rightarrow \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$. If there is a *kernel* function K which satisfies $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi^T(\mathbf{x}_i)\Phi(\mathbf{x}_j)$ then the calculations will be entirely on K. Thus the problem becomes to find a linear margin into new feature space F. Then the conditions (8) can be generalized to

$$maximize\ (with\ respect\ to\ \alpha)\ \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \qquad (14)$$

$$subject\,to\ \sum_{i=1}^{N}\alpha_i y_i = 0 \text{ and } 0 \le \alpha_i \le C,\ i = 1,2,\dots,N. \qquad (15)$$

An important issue here is how to choose kernel function. It turns out that a function can be used as a kernel if it satisfies conditions of the Mercer's theorem [2]. Most used kernels are described below.

Kernel function	Description
$K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$	Dot product kernel
$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p$	Polynomial kernel
$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\lVert\mathbf{x}_i - \mathbf{x}_j\rVert^2 / 2\sigma^2}$	Radial basis (Gaussian) kernel
$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(k\,\mathbf{x}_i^T \mathbf{x}_j - \delta)$	Sigmoidal kernel (satisfies Mercer's theorem only for some values (k, δ))

***Table 1**: Most used kernel functions*

How can a trained SVM be used for classifying an unknown vector? It depends on the sign of the following expression

$$f(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b, \qquad (16)$$

where

$$b = \left(\sum_{i=1}^{N}(C - \alpha_i)\alpha_i y_i - \sum_{i=1}^{N}(C - \alpha_i)\alpha_i \sum_{j=1}^{N}\alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \Big/ \sum_{i=1}^{N}(C - \alpha_i)\alpha_i. \qquad (17)$$

In the sums above it holds $\alpha_i > 0$ only for the support vectors hence they play the main role during the usage phase.

Nonlinear SVM is illustrated below. In the right side it is shown the shape of the separating surface (in addition zero plane is shown).

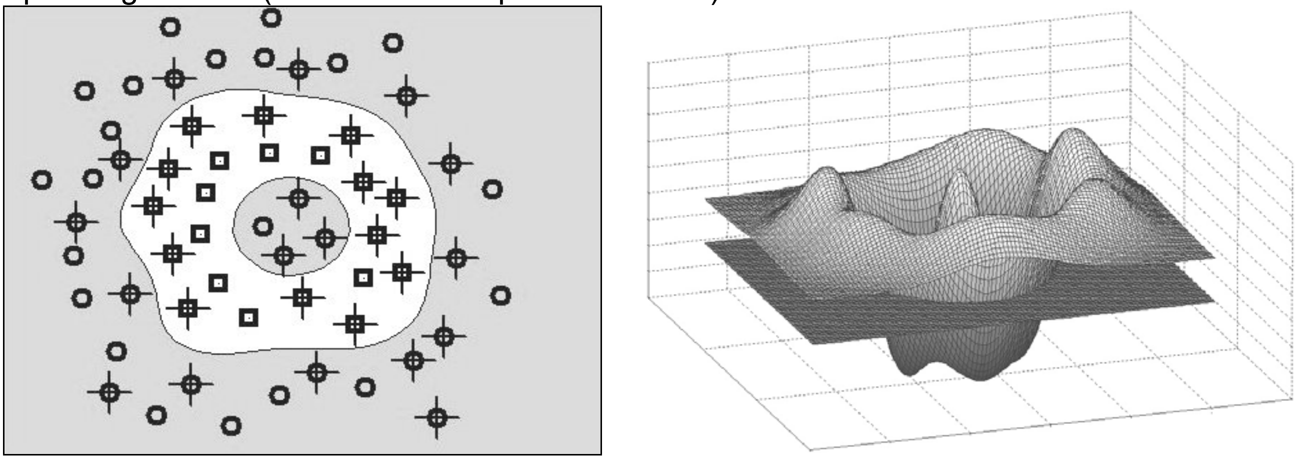

***Figure 3**: Nonlinear SVM: Gaussian kernel* $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\lVert\mathbf{x}_i - \mathbf{x}_j\rVert^2}{2\sigma^2}}$ *with* $\sigma = 1$ *is used.*

When the Gaussian kernel is used it appears a great amount of support vectors. In the example above we have $N = 59$ samples and $N_s = 29$ support vectors.

In the next example we use a polynomial kernel.

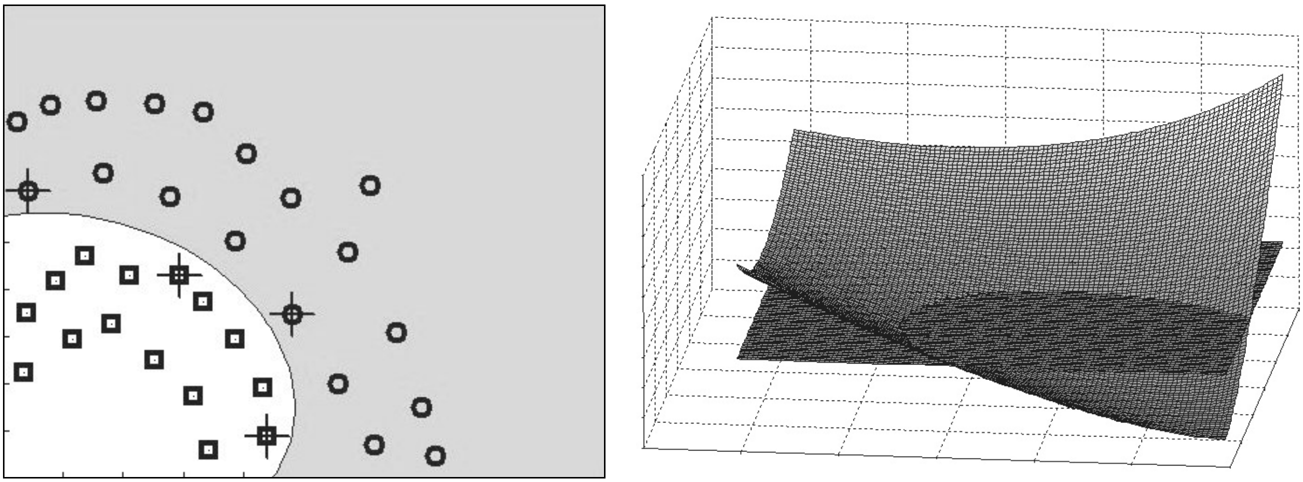

***Figure 4**: Nonlinear SVM: Polynomial kernel* $K(\mathbf{x}_i,\mathbf{x}_j)=\left(\mathbf{x}_i^T\mathbf{x}_j+1\right)^{3/2}$ *is used.*

It is interesting to compare SVM with other neural network architectures. Compared to multilayer perceptron (backprop algorithm), it is seen that sigmoidal-kernel SVM is actually a perceptron with one hidden layer with N_s neurons on it. Here the learning algorithm selects the optimal amount of support vectors, while for the perceptron it depends on the network designer how many hidden layer neurons will be used. In addition the perceptron learning algorithms (e.g. gradient descent) are slower than SVM learning. Another advantage is that quadratic programming always finds the global minima, while for backpropagation there is a danger to be trapped into a local minima.

CONCLUSIONS AND FUTURE WORK

SVM is a good alternative of the well know backpropagation neural networks. Based on quadratic optimization of convex function, this architecture can easily switch from linear to nonlinear separation, i.e. from input to feature space. This is realized by nonlinear mapping using so-called kernel functions. A topic for future work would be to explore in depth properties of different kernel functions for particular applications.

The authors thank to Assoc. Prof. Svilen Stefanov for the help in the technical preparing of the examples above.

REFERENCES

[1] Burges, C.J.C., A tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery vol.2, p. 121-167, 1998

[2] Cristianini N., J.S. Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, 2000

[3] Haykin, S., Neural Networks – A Comprehensive Foundation, Delhi, India, 1999

ABOUT THE AUTHORS

Nikolay Stoyanov Stanevski, PhD student, Military Unit 24430 – Troyan, Phone: +359 670 2 58 59, E-mail: nstanevski@gmail.com

Assoc.Prof. Dimiter Petkov Tsvetkov, PhD, Department of Mathematics, National Military University “Vassil Levski”, Veliko Tarnovo, Phone: +359 62 600 210, E-mail: dimiter99@yahoo.com