Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 70 (2007) 1089–1095
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Letters

Support vector perceptrons ☆

A. Navia-Vázquez*

DTSC, Univ. Carlos III de Madrid, Avda Universidad 30, 28911-Leganés, Madrid, Spain

Received 8 June 2006; received in revised form 12 July 2006; accepted 14 August 2006
Communicated by G.-B. Huang
Available online 10 October 2006

Abstract

Due to their excellent performance, support vector machines (SVMs) are now used extensively in pattern classification applications. In this paper we show that the standard sigmoidal kernel definition lacks the capability to represent the family of perceptrons, and we propose an improved SVM with a sigmoidal kernel called support vector perceptron (SVP). We show by means of both synthetic and real world data sets that the proposed SVP is able to provide very accurate results in many classification problems, providing maximal margin solutions when classes are separable, and also producing very compact architectures comparable to classical multilayer perceptrons.

Keywords: Support vector; Sigmoidal; Kernel; Multilayer; Perceptron

1. Introduction: support vector machines with sigmoidal kernels

Recently support vector machines (SVMs) have been extensively used by the machine learning community because they effectively deal with high dimensional data, provide good generalization properties and define the classifier architecture in terms of the so-called support vectors (SVs), once the hyperparameters are set (usually by means of a cross-validation procedure) [14,17]. Nonlinear SVMs are obtained by mapping input patterns to a feature space F, such that all operations comprising inner products in F can be computed using a kernel function. Most successful SVMs use Gaussian kernels (yielding networks analogous to radial basis functions), while sigmoid kernels (potentially leading to single-hidden-layer perceptrons) are seldom successfully used in real applications. Sigmoid kernels may lead to kernel matrices which are non-positive-semi-definite (PSD) [14], and PSD kernel matrices are required by the SVM framework to obtain the solution by means of quadratic programming (QP) techniques. Some authors have tried to circumvent this problem by either developing linear approximations to the sigmoidal kernel [3] (with some additional drawbacks) or by implementing special solver methods robust to non-PSD kernel matrices. Among the latter, sequential minimal optimization (SMO [13]) decomposition methods are used in [9] to solve non-convex dual problems, leading to the successful, robust and popular SVM software implementation known as LibSVM [8], which is always able to provide a solution with sigmoid kernels. In the experimental part of this paper we will use LibSVM as a reference method for benchmarking our proposed algorithm, since other software implementations sometimes fail to provide a solution, possibly because they do not take into account the case of non-PSD kernel matrices, often encountered in practice when using sigmoid kernels. LibSVM authors have explicitly taken into account these non-PSD situations, as explained in [9], and therefore their LibSVM package seems to be robust when working with the sigmoid kernel.

We have observed that the average performance obtained with the sigmoidal kernel was systematically worse than that obtained with Gaussian kernels. By simple inspection, it can be directly observed that sigmoid-SVMs are actually working with a subset of all possible perceptrons. Before discussing this kernel and its

☆This work has been partially supported by Spain CICYT Grant TEC2005-04264/TCM and CAM Grant PRO.MULTIDIS S-0505/TIC/0223.
*Tel.: +34 91 6245977; fax: +34 91 6248749.
E-mail address: navia@tsc.uc3m.es.

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2006.08.001

limitations, let us briefly recall the foundations of SVMs. Consider a binary classification problem defined by a set of labelled input patterns $\{\mathbf{x}_i, y_i\}_{i=1}^N$ with $\mathbf{x}_i \in \mathbb{R}^P$ and $y_i \in \{-1, +1\}$. The SVM first projects the input data onto a (usually high-dimensional) space, F, by means of a nonlinear projection $\boldsymbol{\phi}(\cdot)$, in a way that inner products between projected vectors can be computed by means of a kernel function $k(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_j)$ (the so-called 'kernel trick'). Then the SVM finds a maximal margin linear classifier in F, $f(\mathbf{x}) = sign(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) + b)$, where $\mathbf{w}$ is the solution to

$$\min_{\mathbf{w},\xi_i}\left\{\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w} + C\sum_{i=1}^{N}\xi_i\right\}$$
$$\text{s.t.}\begin{vmatrix} y_i(\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_i) + b) - 1 + \xi_i \geqslant 0, & \forall i = 1,\ldots,N, \\ \xi_i \geqslant 0, & \forall i = 1,\ldots,N, \end{vmatrix} \quad (1)$$

where ξ_i are positive slack variables introduced to deal with non-separable problems and C is the penalization for patterns incorrectly classified or inside the margin. The solution to the optimization problem in (1) is a linear combination of S patterns called SVs: $\mathbf{w} = \sum_{i=1}^{S} y_i\alpha_i\boldsymbol{\phi}(\mathbf{x}_i)$. Then the classifier (before taking the sign) becomes, for the sigmoid case

$$o(\mathbf{x}_j) = \sum_{i=1}^{S} y_i\alpha_i\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j) + b = \sum_{i=1}^{S}\beta_i\tanh(\mathbf{v}_i^{\mathrm{T}}\mathbf{x}_j + \delta_0) + b, \quad (2)$$

where $\beta_i = y_i\alpha_i$, $\mathbf{v}_i = \gamma\mathbf{x}_i$, and γ and δ_0 are the kernel hyperparameters, such that (2) is equivalent to a MLP with a hidden layer of S neurons. In the standard SVM γ and δ_0 are obtained by cross-validation, and the same values are used for all network nodes, which is a severe limitation. This will prevent the sigmoid-SVM from having the same representational potential as a true MLP. Furthermore, not all (γ, δ_0) parameter combinations lead to a valid kernel function, sometimes deriving in computational problems since we obtain non-PSD kernel matrices. To avoid this, it is recommendable to use positive values for γ and negative values for δ_0, as discussed in [9], a restriction which further limits the capability of the sigmoid-SVM.

In the following section we present the support vector perceptron (SVP) model, which solves these drawbacks by training the SVM without any restrictions on the hyperparameter values, and we also provide a training SVP algorithm which is free from the non-PSD problem of QP-solver-based methods.

2. The SVP algorithm

In the standard SVM formulation we have little control over the kernel hyperparameters once the QP optimization starts, since they are fixed beforehand. We therefore need a more flexible scheme to be able to select good kernel hyperparameters as learning progresses, and without the restriction $\delta_i = \delta_0, \forall i$. We propose to take advantage of a previously developed method to grow semiparametric models [10,11]. Under this paradigm, the size of the classifier can be effectively controlled by introducing a predefined model in the formulation of the SVM problem, and updating it according to some additional criteria, to be discussed later. Let us assume that the hyperplane defined by $\mathbf{w}$ can be approximated by a linear combination of several[1] mapped patterns $\boldsymbol{\phi}(\mathbf{z}_i)$, i.e., $\mathbf{w} \simeq \sum_{i=1}^{R}\beta_i\boldsymbol{\phi}(\mathbf{z}_i)$, s.t. the classifier (before taking the sign) becomes $o(\mathbf{x}_j) = \sum_{i=1}^{R}\beta_i\boldsymbol{\phi}^{\mathrm{T}}(\mathbf{z}_i)\boldsymbol{\phi}(\mathbf{x}_j) + b = \sum_{i=1}^{R}\beta_i k(\mathbf{z}_i, \mathbf{x}_j) + b$. With this approximation, the optimization in (1) is transformed into the following regularized iterative weighted least-squares (IWLS) minimization

$$\min_{\boldsymbol{\beta},b}\left\{\frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_z\boldsymbol{\beta} + \frac{1}{2}\sum_{i=1}^{N}a_i e_i^2\right\};$$
$$a_i = \begin{cases} 0; & e_i y_i < 0, \\ M; & 0 \leqslant e_i y_i \leqslant C/M, \\ \dfrac{C}{e_i y_i}; & e_i y_i > C/M, \end{cases} \quad (3)$$

where $(\mathbf{K}_z)_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$, $e_i = y_i - o_i$, M is a large regularizing constant, usually set to 10^9 and a_i are weighting values whose precise computation has already been derived in [10]. Briefly, these weights a_i serve to transform the L_1-norm functional in (1) (the error term appears as a sum of ξ_i terms) into a least squares functional (3) (where the errors have been converted to L_2-norm, and they appear as a sum of weighted squared errors of the form $a_i e_i^2$). To solve the minimization in (3), we first rewrite the error term in matrix form, to obtain

$$\min_{\boldsymbol{\beta},b}\left\{\frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_z\boldsymbol{\beta} + \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{D}_a\mathbf{e}\right\}, \quad (4)$$

where $\mathbf{D}_a = diag\{a_i\}$, $\mathbf{e} = \mathbf{y} - (\mathbf{K}\boldsymbol{\beta} + b)$, $(\mathbf{K})_{i,j} = k(\mathbf{x}_i, \mathbf{z}_j)$, $\mathbf{y} = [y_1, \ldots, y_P]^{\mathrm{T}}$, and $\mathbf{e} = [e_1, \ldots, e_P]^{\mathrm{T}}$. We have now to compute partial derivatives with respect to $\boldsymbol{\beta}$ and b and make them equal to zero:

$$\frac{\partial}{\partial\boldsymbol{\beta}}\left\{\frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_z\boldsymbol{\beta} + \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{D}_a\mathbf{e}\right\} = \mathbf{K}_z\boldsymbol{\beta} - \mathbf{K}^{\mathrm{T}}\mathbf{D}_a(\mathbf{y} - (\mathbf{K}\boldsymbol{\beta} + b)) = \mathbf{0}, \quad (5)$$

$$\frac{\partial}{\partial b}\left\{\frac{1}{2}\boldsymbol{\beta}^{\mathrm{T}}\mathbf{K}_z\boldsymbol{\beta} + \frac{1}{2}\mathbf{e}^{\mathrm{T}}\mathbf{D}_a\mathbf{e}\right\} = -\mathbf{1}^{\mathrm{T}}\mathbf{D}_a\mathbf{y} + \mathbf{1}^{\mathrm{T}}\mathbf{D}_a\mathbf{K}\boldsymbol{\beta} + \mathbf{1}^{\mathrm{T}}\mathbf{D}_a\mathbf{1}b = 0, \quad (6)$$

where $\mathbf{1} = [1, \ldots, 1]^{\mathrm{T}}$. After reordering terms and joining those respectively depending on $\boldsymbol{\beta}$ and b, we obtain a system of two equations linear in the unknowns $(\boldsymbol{\beta}, b)$, that

[1] R is usually much smaller than the number of support vectors S, as will be shown in the experimental section.