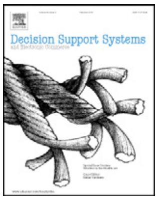
Decision Support Systems 52 (2012) 717–728

Contents lists available at SciVerse ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss



# Novel linear programming approach for building a piecewise nonlinear binary classifier with a priori accuracy

Ubaldo M. García-Palomares [a,b,*,1], Orestes Manzanilla-Salazar [b,c]

[a] Dep. Ingeniería Telemática, Universidad de Vigo, 36310 Vigo, Spain
[b] Dep. Ingeniería de Sistemas, Universidad Simón Bolívar, Caracas 89000, Venezuela
[c] CESMa, Universidad Simón Bolívar, Caracas 89000, Venezuela

ARTICLE INFO

*Article history:*
Received 19 October 2010
Received in revised form 22 September 2011
Accepted 3 November 2011
Available online 11 November 2011

*Keywords:*
Binary classification
Support vector machines
Artificial neural networks
Classification trees
Linear programming
Piecewise discriminator

ABSTRACT

This paper describes a novel approach to build a piecewise (non)linear surface that separates individuals from two classes with an a priori classification accuracy. In particular, total classification with a good generalization level can be obtained, provided no individual belongs to both classes. The method is iterative: at each iteration a new piece of the surface is found via the solution of a Linear Programming model. Theoretically, the larger the number of iterations, the better the classification accuracy in the training set; numerically, we also found that the generalization ability does not deteriorate on the cases tested. Nonetheless, we have included a procedure that computes a lower bound to the number of errors that will be generated in any given validation set. If needed, an early stopping criterion is provided. We also showed that each piece of the discriminating surface is equivalent to a neuron of a feed forward neural network (FFNN); so as a byproduct we are providing a novel training scheme for FFNNs that avoids the minimization of non convex functions which, in general, present many local minima.
We compare this algorithm with a new linear SVM that needs no pre tuning and has an excellent performance on standard and synthetic data. Highly encouraging numerical results are reported on synthetic examples, on the Japanese Bank dataset, and on medium and small datasets from the Irvine repository of machine learning databases.

## 1. Introduction

Let us first mention minor peculiarities of our notation: $R^n$ denotes the $n$-th dimensional Euclidean space, lowercase Latin letters are vectors in $R^n$, $x_i^k$ is the $k$-th component of $x_i$, uppercase cursive letters $\mathcal{P},\mathcal{N}$, etc., denote sets in $R^n$. Given the sets $\mathcal{D},\mathcal{H}$ the difference set $\mathcal{D}-\mathcal{H}$ consists of those members of $\mathcal{D}$ that do not belong to $\mathcal{H}$. Lowercase Greek letters denote scalars. The rest of the notation is rather standard.

This paper is concerned with the binary classification problem (BCP) of determining a discriminating function $h(\cdot):R^n\rightarrow R$ that separates *individuals* belonging to two different classes, say, class $\mathcal{P}$ and class $\mathcal{N}$. This is a classical problem that still has vast applications in several areas [4]. This work solves the BCP by successive linear programming (LP) models, which generate nodes of a decision tree. Each node classifies a group of individuals of one class.

We assume that an individual $x$ is characterized by its $n$ features $x^1,\ldots,x^n$ and its class indicator $c(x)$,

$$c(x)=\begin{cases}1, & x\in\mathcal{P}\\ -1, & x\in\mathcal{N}.\end{cases}$$

We extend this definition to the class indicator of a set $\mathcal{S}$,

$$c(\mathcal{S})=\begin{cases}1, & \mathcal{S}\subseteq\mathcal{P}\\ -1, & \mathcal{S}\subseteq\mathcal{N}\\ 0, & \text{otherwise}.\end{cases}$$

The discriminating function $h(\cdot):\mathcal{D}\rightarrow R$, also called discriminating surface, decision rule or learning rule, satisfies

$$h(x)\begin{cases}>0, & x\in\mathcal{P}\\ <0, & x\in\mathcal{N}.\end{cases}$$

Albeit a discriminating function always exists, provided $(\mathcal{P}\cap\mathcal{N})=\varnothing$, there are practical issues that prevent the practitioners from obtaining

[*] Corresponding author.
*E-mail addresses:* garciap@usb.ve, ubaldo@det.uvigo.es (U.M. García-Palomares).
[1] Partial support by Ministerio de Ciencia e Innovación, Spain.

0167-9236/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.11.006

a solution. They rather search for the best *separating* function that minimizes a measure of the error set $\mathcal{E}$ defined as

$$\mathcal{E} = \{x \in \mathcal{N} : h(x) \geq 0\} \cup \{x \in \mathcal{P} : h(x) \leq 0\}. \tag{1}$$

On top of this, practitioners must formulate a model simple enough to generate an accurate solution within a reasonable time and allocated budget. Instead of striving for a discriminating function two less ambitious objectives are pursued. Given a class of functions $\mathcal{F}$ we want

O1 To find the best *separating* function in $\mathcal{F}$ for a training set $(\mathcal{T} \subseteq \mathcal{D})$.
O2 To accurately predict the class indicator of those individuals in $(\mathcal{D} - \mathcal{T})$.

A variety of optimization models to minimize a measure of the error set $\mathcal{E}$ have been suggested. Linear programming optimization models (LPs) are mostly used, because they find a solution in polynomial time, are robust and may deal with large problems. The simplest strategy is to obtain the best linear separating function. In the late sixties, [26] proposed the multisurface method (MSM), which finds a piecewise linear discriminating function for the training set $\mathcal{T}$. A clear description of MSM is given by [33] and a more updated version of MSM is given by [29].

Least square techniques and nonlinear constrained optimization models are used to train (FFNNs), which can generate a piecewise linear discriminating surface for any two disjoint sets with a sufficiently large but unknown number of neurons in the hidden layer [17]. MSM may also be used to train a feed forward neural network (FFNN) via LP models [27, Section 2].

Support vector machines (SVMs) developed at Bell Laboratories [41,42] solve quadratic programming optimization models (QPs), mainly with the intention of improving the prediction ability of the solution (objective O2). SVMs use kernels, which allow the search of the best linear separating function in a space bounded by the number of individuals. Primal and Dual QP models have been tried for its solution. An account of going on research for an efficient implementation of these QP models is given in [3, Section 5]. SVMs' objective is structural risk and does not attempt total separation, which can be obtained by our approach. Hence, our algorithm will render important when this feature is indispensable for the user.

Mixed integer linear programming models (MILPs) have recently been proposed whose solution gives rise to a piecewise separating surface [4,21,22], but there is no hint on the number of pieces necessary to obtain a required classification accuracy. A good account on different optimization models that have been proposed in the open literature to obtain the best separating function is given by Bennett and Parrado-Hernandez [3] and by Smaoui et al. [38].

We should keep in mind that the larger the set of functions $\mathcal{F}$, the better the separating function $h \in \mathcal{F}$ that minimizes the error set (1) on $\mathcal{T}$; but commonly the prediction ability on $(\mathcal{D} - \mathcal{T})$ worsens. This incident has been called overfitting, and it causes a conflict between O1 and O2. A common approach is to impose a *k*-fold validation on any model that tries to satisfy the first objective: the given sample is split in *k* subsamples; one of them is taken as a testing set and the separating function is sought on the training set made up by the remainder of the sample. This is done *k* times and the rates of success obtained on the testing sets give an estimate of the prediction ability.

This paper presents an iterative algorithm that solves LP models and is able to find any explicit classification accuracy on a training set. It offers the following salient properties:

P1 The algorithm generates a piecewise (non)linear discriminating function.
P2 Within a finite number of iterations a complete or an a priori classification accuracy can always be obtained on the whole training set, or on either of the classification sets.
P3 The optimization model is linear, but may consider as many binary variables as predetermined by the computational resources.
P4 The size of the optimization model decreases with the iteration number; it is then plausible to solve MILPs at the final stages of the method.
P5 The algorithm provides a lower bound on the number of errors that will occur in a given testing set.
P6 Large systems can be handled by way of parallelism and/or decomposition of the training set.
P7 An FFNN is easily adapted, which opens up the possibility of hardware implementations with the use of field programmable gate arrays (FPGAs). See [39] and ([34], Chapters 1 and 10).
P8 Finally, only basic programming skills are needed for its implementation.

To summarize, our approach enjoys properties P2, P3, P4 which circumvent some of the deficiencies encountered in FFNNs, SVMs and MILPs mentioned earlier. It also exhibits other properties that enhance its usefulness.

Before proceeding any further, let us illustrate our approach with an easy example. Fig. 1 depicts a dot curve discriminating the sets $\mathcal{P}$ and $\mathcal{N}$. Generally this function is unknown to us, and it is in general a difficult task to find out a nonlinear function that discriminates two classes (see [20], and references therein). Let us graphically exemplify how our algorithm obtains a piecewise linear discriminating function. The first iteration of the algorithm finds a (hyper)plane that forces all individuals in $\mathcal{N}$ to be on one side, and all members of $\mathcal{P}$ located on the other side are considered well classified (Fig. 2a). The second iteration works on the reduced set of not yet classified individuals (Fig. 2b) and finds a new (hyper)plane. At this iteration, the hyperplane forces all individuals in $\mathcal{P}$ to be on the same side (Fig. 2c). In this example total classification was obtained; otherwise, the iterations proceed until $\mathcal{P}$ or $\mathcal{N}$ is exhausted. A test is included that may stop the algorithm as soon as a possibility of overfit is detected.

We now formally state the BCP. Let $\mathcal{D} \subset \mathcal{R}^n$ be a bounded set, and let $\mathcal{P}, \mathcal{N} \subset \mathcal{D}$. We assume that there exists a discriminating function $f(\cdot) : \mathcal{D} \to R$ such that $f(x) > 0$ for all $x \in \mathcal{P}$ and $f(x) < 0$ for all $x \in \mathcal{N}$. We also assume that sign$(f(x))$ is either at hand or easily computable for any $x \in \mathcal{T} \subseteq \mathcal{D}$. In other words, we can easily deduce the class indicator $c(x)$ in some subset of $\mathcal{D}$. Let us define the *hard*-margin $\rho(f)$ as:

$$\rho(f) = min(||x - z||), x \in (\mathcal{P} \cup \mathcal{N}), z \in \mathcal{Z} = \{x \in \mathcal{D} : f(x) = 0\}, \tag{2}$$

where $||\cdot||$ is any norm in $R^n$. Geometrically, we might assume there is a gray zone $\mathcal{Z}_\epsilon = \{x \in \mathcal{D} : |f(x)| \leq \epsilon\}\mathbb{F}$, where sign$(f)$ is uncertain, either because of noise or by probabilistic estimation. In any event, the bigger the *hard*-margin, the better the predictive accuracy of an algorithm [33,37].

Our ultimate objective is to elaborate on linear programming optimization models that with the sign information $c(x) = \text{sign}(f(x))$ for

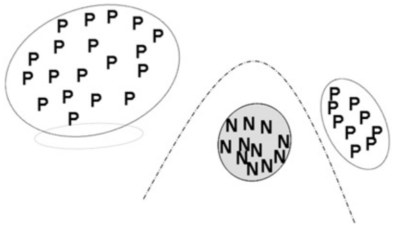


**Fig. 1.** The unknown dot curve strictly discriminates sets $\mathcal{P}$ and $\mathcal{N}$.