Available online at www.sciencedirect.com
ScienceDirect
Neurocomputing 71 (2008) 1550–1560
NEUROCOMPUTING
www.elsevier.com/locate/neucom

Incremental margin algorithm for large margin classifiers

Saul C. Leite, Raul Fonseca Neto*

Laboratório Nacional de Computação Científica, Av. Getulio Vargas, 333, Quitandinha CEP:25651-075, Petrópolis, Rio de Janeiro, Brazil

Received 12 January 2006; received in revised form 5 October 2006; accepted 1 May 2007
Communicated by S. Hu
Available online 7 June 2007

Abstract

In this contribution, we introduce a new on-line approximate maximal margin learning algorithm based on an extension of the perceptron algorithm. This extension, which we call fixed margin perceptron (FMP), finds the solution of a linearly separable learning problem given a fixed margin. It is shown that this algorithm converges in $(R^2 - \gamma_f^2)/(\gamma^* - \gamma_f)^2$ updates, where $\gamma_f < \gamma^*$ is the fixed margin, γ^* is the optimum margin and R is the radius of the ball that circumscribes the data. The incremental margin algorithm (IMA) approximates the large margin solution by successively using FMP with increasing margin values. This incremental approach always guarantees a good solution at hands. Also, it is easy to implement and avoids quadratic programming methods. IMA was tested using several different data sets and it yields results similar to those found by an SVM.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Perceptron algorithm; Large margin classifiers; Kernel methods

1. Introduction

One of the fundamental problems in learning theory is the construction of classifiers that are capable of good generalization performance. In this sense, the development of margin-based VC Theory, introduced by Vapnik [16], was an important step toward the construction of improved classifiers. Under this theory, the support vector machines (SVM) [1] were developed. This algorithm is based on the idea that one can improve generalization of a linear classifier by finding a hyperplane that maximizes its distance from data of opposite classes. In order to construct this hyperplane, one needs to solve a quadratic programming problem in the natural formulation of SVMs.

Considerable attention in the literature has been given to finding simple and efficient algorithms to construct large margin classifiers that avoid the complexity of quadratic programming (e.g. [4,10,12,15]). Some of these algorithms are motivated by the fact that, usually, one only needs an approximation of the maximal margin solution to have good generalization performance. In this paper, we present a simple and efficient on-line method, meaning that it processes one example at a time, which constructs an approximation of the maximal margin solution.

The method is composed of two different algorithms. The first, called fixed margin perceptron (FMP), is an extension of Rosenblatt's perceptron algorithm [13]. It finds the solution of a linearly separable problem given a fixed margin. Or in other words, given a classification problem, one can stipulate a fixed margin value that will be attended by the final solution, as long as the problem continues to be linearly separable. Also, a bound for the number of updates based entirely on the perceptron convergence theorem is presented, as well as the algorithm in dual variables. The second algorithm, the incremental margin algorithm (IMA), is built on top of the first. The algorithm gradually increases the fixed margin value used by FMP and finds a large margin solution similar to that found by an SVM. One advantage of this approach is that we always have a good solution at hands. This way, after the first FMP is finished, one can interrupt the algorithm at any point and it will return a

*Corresponding author. Tel.: +55 24 2233 6133; fax: +55 24 2233 6167.
E-mail address: raulfonsecaneto@ig.com.br (R.F. Neto).

feasible solution, this might be important for time dependent applications. Also, IMA can be used in primal and dual variables, making it more flexible for different types of problems.

The layout of the paper is as follows: in the next section we will briefly introduce the binary classification problem as well as some definition and notation that will be used throughout this article. In Section 3, we will introduce the FMP in its primal formulation, followed by the IMA in Section 4 and FMP in dual variables in Section 5. In Section 6, we present some comparative tests and results. The final conclusions and discussion are given in Section 7.

2. Preliminaries

Briefly, a classification problem can be stated as follows: given a training set $z_m = \{(x_i, y_i)\}_{i=1}^m$, $(x_i, y_i) \in \mathscr{X} \times \mathscr{Y}$, find a function $h \in \mathscr{Y}^{\mathscr{X}} = \mathscr{H}$ which will assign labels to future "unseen" data points. Usually, we call the points x_i *examples*, the labels y_i *classes*, the space $\mathscr{X}$ *input space* and $\mathscr{H}$ *hypothesis space*. Often, we use a function $\Phi : \mathscr{X} \to \mathscr{F}$ that takes points in input space, where data might be considered "raw," and puts them to *feature space* $\mathscr{F}$. Also, for simplicity, we denote any point in feature space in bold letters, for instance $\Phi(x_i) = \boldsymbol{x}_i$.

In this paper, we will only consider binary linear classification problems, where $\mathscr{Y} = \{-1, +1\}$ and our hypothesis space is restricted to $\mathscr{H} = \{h \in \mathscr{Y}^{\mathscr{X}} | h = \text{sign}(\langle \boldsymbol{x}, \boldsymbol{w} \rangle), \boldsymbol{x}, \boldsymbol{w} \in \mathscr{F}\}$. In this sense, we are looking for a hyperplane that separates the data into two classes in feature space, where $\boldsymbol{w}$ is the hyperplane's normal vector. Notice that we only consider hyperplanes passing through the origin, since we assume the points $\boldsymbol{x}_i$ to be augmented (i.e. $\boldsymbol{x}_i$ is replaced with $(\boldsymbol{x}_i, 1)$ and the normal vector $\boldsymbol{w}$ is replaced with $(\boldsymbol{w}, b)$) to incorporate the bias term as the last component of the normal vector.

The training set z_m is said to be *linearly separable* if $\mathscr{V}(z_m) = \{\boldsymbol{w} \in \mathscr{F} | y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle > 0,\ i = 1, \ldots, m\} \neq \emptyset$, where $\mathscr{V}(z_m) \subset \mathscr{F}$ is called the *version space*, following the definition given by Herbrich [7].[1] In words, a problem is said to be linearly separable if there exists a hyperplane that separates the data. Observe that given this set we can define the following $\mathscr{V}_{\mathscr{H}}(z_m) = \{h \in \mathscr{Y}^{\mathscr{X}} | h = \text{sign}(\langle \boldsymbol{x}, \boldsymbol{w} \rangle), \boldsymbol{x} \in \mathscr{F}, \boldsymbol{w} \in \mathscr{V}(z_m)\} \subset \mathscr{H}$, which is the set of hypothesis which are consistent with the training set. Also, given a normal vector $\boldsymbol{w} \in \mathscr{F}$ we can always define a hypothesis $h_{\boldsymbol{w}}(\cdot) = \text{sign}(\langle \boldsymbol{w}, \cdot \rangle) \in \mathscr{H}$.

For a hypothesis $h_{\boldsymbol{w}} \in \mathscr{V}_{\mathscr{H}}(z_m)$, we define the *functional margin* to be $\Gamma(h_{\boldsymbol{w}}) = \Gamma(\boldsymbol{w}) = \min\{\gamma_i | \gamma_i = y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle,\ i = 1, \ldots, m\}$ and the *geometric margin* to be $\gamma(h_{\boldsymbol{w}}) = \gamma(\boldsymbol{w}) = \Gamma(h_{\boldsymbol{w}})/\|\boldsymbol{w}\|$, as long as $\boldsymbol{w} \neq 0$. Also, given a *fixed margin* $\gamma_{\text{f}} \in \mathbb{R}_{\geqslant 0}$, we can define a new version space as $\mathscr{V}(z_m, \gamma_{\text{f}}) = \{\boldsymbol{w} \in \mathscr{F} | y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle / \|\boldsymbol{w}\| \geqslant \gamma_{\text{f}}, \boldsymbol{w} \neq 0,\ i = 1, \ldots, m\}$, where now if $\mathscr{V}(z_m, \gamma_{\text{f}}) \neq \emptyset$ we say that the problem is linearly separable given the fixed margin γ_{f}.

[1]Except that we do not require $\|\boldsymbol{w}\|$ to be one.

3. Fixed margin perceptron (FMP)

Rosenblatt's perceptron is formulated to find one solution that is consistent with the training set. This is done creating an error function $J : \mathscr{F} \subseteq \mathbb{R}^n \to \mathbb{R}$ defined as

$$J(\boldsymbol{w}) = \sum_{(x_i, y_i) \in \mathscr{M}} -y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle,$$

where the set $\mathscr{M} = \mathscr{M}(z_m, \boldsymbol{w}) = \{(x_i, y_i) \in z_m | y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle \leqslant 0\}$. This error function is used to guide the search in the hypothesis space, and a $\boldsymbol{w} \in \mathscr{F} \subseteq \mathbb{R}^n$ that minimizes $J(\boldsymbol{w})$ is chosen. The perceptron algorithm is an on-line mistake driven algorithm, meaning that it only sees one example at the time and will only change its hypothesis if the given example yields a mistake. This way, for every mistake, the normal vector $\boldsymbol{w}$ is corrected in the negative direction of the gradient. Therefore, every time $y_i \langle \boldsymbol{w}^t, \boldsymbol{x}_i \rangle \leqslant 0$ we update $\boldsymbol{w}^t$ by

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \eta y_i \boldsymbol{x}_i,$$

where $\eta > 0$ is the learning rate. This functional minimization strategy is called stochastic gradient descent.

Duda et al. [3] suggest an alternative version of the perceptron algorithm where a margin ρ is imposed, and a mistake occurs when $y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle \leqslant \rho$. However, this margin is by no means "fixed" since the norm $\|\boldsymbol{w}\|$ is not bounded. In other words, for every different $\boldsymbol{w}$ with a different norm $\|\boldsymbol{w}\|$, the geometric margin associated with this ρ is different. Also, as long as the problem is linearly separable, we can always find a solution that solves the problem for any ρ. Therefore, this margin does not create any kind of additional constraint for the original problem.

In this sense, a new formulation was developed, where a fixed margin is imposed creating a more constrained problem. So given a fixed margin γ_{f}, a new mistake occurs if $y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle / \|\boldsymbol{w}\| < \gamma_{\text{f}}$ or, equivalently, if $y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle < \gamma_{\text{f}} \|\boldsymbol{w}\|$. Therefore, we have the new error function

$$J(\boldsymbol{w}) = \sum_{(x_i, y_i) \in \mathscr{M}} (\gamma_{\text{f}} \|\boldsymbol{w}\| - y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle)$$

with $\mathscr{M} = \mathscr{M}(z_m, \boldsymbol{w}, \gamma_{\text{f}}) = \{(x_i, y_i) \in z_m | y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle < \gamma_{\text{f}} \|\boldsymbol{w}\|\}$, which yields the new update rule

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \eta \left(y_i \boldsymbol{x}_i - \gamma_{\text{f}} \frac{\boldsymbol{w}^t}{\|\boldsymbol{w}^t\|} \right).$$

If we define the *margin vector* to be $\boldsymbol{\gamma} = \gamma_{\text{f}}(\boldsymbol{w}/\|\boldsymbol{w}\|)$ for a geometrical interpretation, we can have a better understanding of this new update rule. Notice that the margin vector is subtracted from the training example before the correction is made. This could be understood as if the error generated by the example $\boldsymbol{x}_i$ was increased in the opposite direction of the normal vector, see Fig. 1.

Harrington et al. [6] introduced the idea of a generalized perceptron where it is considered the use of a fixed geometric margin parameter ρ in the perceptron formulation. Similar to FMP, this algorithm searches for a solution $\boldsymbol{w}$ such that $\gamma(\boldsymbol{w}) > \rho$. This is done by updating the normal vector every time $y_i\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle / \|\boldsymbol{w}\| \leqslant \rho$ and using the perceptron update rule. However, there is no guarantee that this algorithm will converge for $\rho > 0$, since the norm $\|\boldsymbol{w}\|$ is not controlled.

It is important to notice that this new update rule has a singularity at $\boldsymbol{w} = 0$. In this case, we use the original perceptron update rule. This will not cause any additional problems, since the norm $\|\boldsymbol{w}\|$ tends to grow over time. Therefore, it should only be a problem when the algorithm starts. Also, if we rewrite the update rule as

$$\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t \lambda_t + \eta y_i \boldsymbol{x}_i, \quad \text{where}$$
$$\lambda_t = \begin{cases} 1 - \dfrac{\eta\gamma_{\mathrm{f}}}{\|\boldsymbol{w}^t\|} & \text{if } \|\boldsymbol{w}^t\| \neq 0, \\ 1 & \text{otherwise,} \end{cases} \tag{1}$$

it becomes clear that we will have a problem if $\lambda_t < 0$, which would end up scaling the normal vector by a negative factor and would completely change its direction. However, the following lemma ensures that the above condition will never happen.

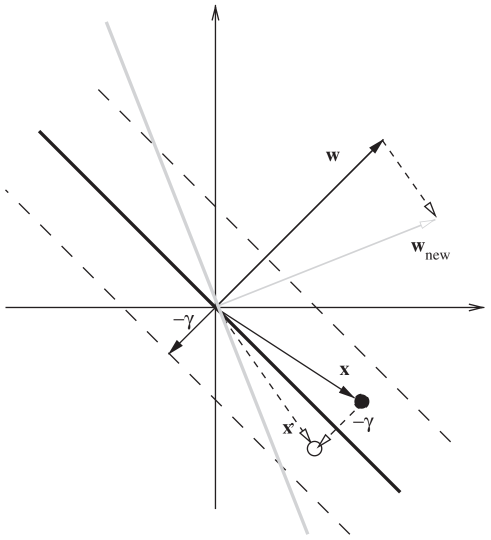


Fig. 1. The bold line crossing the axis with the normal vector $\boldsymbol{w}$ is the hyperplane before correction, and the gray line is the hyperplane after correction. The dashed lines parallel to the hyperplane represent the fixed margin size. The margin vector is denoted by $\boldsymbol{\gamma}$, which is in the direction of the normal vector $\boldsymbol{w}$. Notice that the training example $\boldsymbol{x}$ is a margin error since it is inside the margin. For interpretation purposes, we can imagine that the new update rule "forces" the point to cross the hyperplane (this is done by setting $\boldsymbol{x}' = \boldsymbol{x} - y\boldsymbol{\gamma}$) and follows with the regular perceptron update rule, correcting $\boldsymbol{w}_{\text{new}} \leftarrow \boldsymbol{w} + \eta y \boldsymbol{x}'$.

Algorithm 1. Primal FMP algorithm.

```
1:  Input: $z_m$, $w_{\text{init}}$, $\gamma_{\mathrm{f}}$, $\eta$, $T\_MAX$
2:  $w^0 \leftarrow w_{\text{init}}$, $t \leftarrow 0$
3:  repeat
4:    for ($i = 1, \ldots, m$) do
5:      if ($y_i\langle x_i, w^t\rangle < \gamma_{\mathrm{f}}\|w^t\|$) then
6:        $w^{t+1} \leftarrow w^t\lambda_t + \eta y_i x_i$
7:        $t \leftarrow t + 1$
8:      end if
9:    end for
10: until (no mistakes were made) or ($t > T\_MAX$)
11: return $w^t$
```

Lemma 3.1. *For a training set z_m that is linearly separable given a fixed margin $\gamma_{\mathrm{f}} > 0$ and using update rule* (1), *the following inequality holds for all number of updates $0 < t < T$:*

$$\|\boldsymbol{w}^t\| > \eta\gamma_{\mathrm{f}}$$

where T is the number of updates until convergence (*i.e.* $\boldsymbol{w}^T \in \mathscr{V}(z_m, \gamma_{\mathrm{f}})$).

Proof. The proof is done by induction. We assume that $T > 1$, otherwise there would be no need for the proof. Also, let us consider the initial normal vector as zero (i.e. $\boldsymbol{w}^0 \equiv 0$), therefore the normal vector after the first update for a mistake in the i_1th pattern is given by

$$w^1 = \eta y_{i_1} \boldsymbol{x}_{i_1},$$

where the regular perceptron update rule was used since $\|\boldsymbol{w}^0\| = 0$. Computing the inner product between $\boldsymbol{w}^*$ and $\boldsymbol{w}^1$, where $\boldsymbol{w}^* \in \mathscr{V}(z_m, \gamma_{\mathrm{f}})$, and using the fact that $y_{i_1}\langle \boldsymbol{x}_{i_1}, \boldsymbol{w}^*\rangle \geqslant \gamma_{\mathrm{f}}\|\boldsymbol{w}^*\|$, we have

$$\langle \boldsymbol{w}^1, \boldsymbol{w}^*\rangle = \eta y_{i_1}\langle \boldsymbol{x}_{i_1}, \boldsymbol{w}^*\rangle \geqslant \eta\gamma_{\mathrm{f}}\|\boldsymbol{w}^*\|.$$

Applying Cauchy–Schwarz inequality, we obtain

$$\|\boldsymbol{w}^1\|\|\boldsymbol{w}^*\| > \langle \boldsymbol{w}^1, \boldsymbol{w}^*\rangle \geqslant \eta\gamma_{\mathrm{f}}\|\boldsymbol{w}^*\| \Rightarrow \|\boldsymbol{w}^1\| > \eta\gamma_{\mathrm{f}},$$

where the inequality is strict since $\boldsymbol{w}^1$ is not a scalar multiple of $\boldsymbol{w}^*$, otherwise $\boldsymbol{w}^1/\|\boldsymbol{w}^1\| = \boldsymbol{w}^*/\|\boldsymbol{w}^*\|$ since $\langle \boldsymbol{w}^1, \boldsymbol{w}^*\rangle \geqslant 0$, which would mean that $\boldsymbol{w}^1 \in \mathscr{V}(z_m, \gamma_{\mathrm{f}})$ and $T = 1$.

Now consider the normal vector after the tth update, $t < T$, for a mistake in the i_tth pattern

$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1}\lambda_{t-1} + \eta y_{i_t}\boldsymbol{x}_{i_t}$$
$$\Rightarrow \|\boldsymbol{w}^t\|\|\boldsymbol{w}^*\| > \langle \boldsymbol{w}^t, \boldsymbol{w}^*\rangle = \langle \boldsymbol{w}^{t-1}, \boldsymbol{w}^*\rangle\lambda_{t-1} + \eta y_{i_t}\langle \boldsymbol{x}_{i_t}, \boldsymbol{w}^*\rangle.$$

Using inequalities $y_{i_t}\langle \boldsymbol{x}_{i_t}, \boldsymbol{w}^*\rangle \geqslant \gamma_{\mathrm{f}}\|\boldsymbol{w}^*\|$ and $\|\boldsymbol{w}^{t-1}\| > \eta\gamma_{\mathrm{f}}$, which implies $\lambda_{t-1} > 0$, we conclude that $\|\boldsymbol{w}^t\| > \eta\gamma_{\mathrm{f}}$. □

With this lemma, we can introduce the primal FMP algorithm, described by Algorithm 1, and the theorem below which extends the perceptron convergence theorem presented by Novikoff [11] to the fixed margin case. It guarantees that FMP will converge in a fixed number of iteration.

Theorem 3.2. (*FMP Convergence Theorem*): *For a training set z_m that is linearly separable given a fixed margin $\gamma_f > 0$, the number of updates made by the FMP algorithm is bounded by*

$$t \leqslant \frac{R^2 - \gamma_f^2}{(\gamma(\boldsymbol{w}^*) - \gamma_f)^2}. \tag{2}$$

where $R = \max_{i \in \{1,\ldots,m\}} \|\boldsymbol{x}_i\|$, $\gamma(\boldsymbol{w}^*) = \max_{\boldsymbol{w} \in \mathscr{V}(z_m)}(\gamma(\boldsymbol{w}))$.

Notice that $\boldsymbol{w}^*$, the maximal margin solution, is not unique given the above requirements; however, each $\boldsymbol{w}^*$ differs only in the size of the norm $\|\boldsymbol{w}^*\|$. Also, it is not necessary that $\boldsymbol{w}^*$ is the maximal margin solution. Actually, any $\boldsymbol{w} \in \mathscr{V}(z_m, \gamma_f)$ suffices. Nevertheless, we chose to use the maximal margin solution for interpretation purposes. The proof of the theorem is given below.

Proof. This proof follows the steps of the perceptron convergence theorem. Let $\boldsymbol{w}^t$ be the normal vector for the hyperplane after the tth update. The update rule for the FMP for a mistake in the ith pattern is given by Eq. (1), which yields the following equation for the dot product $\langle \boldsymbol{w}^*, \boldsymbol{w}^t \rangle$:

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^t \rangle = \langle \boldsymbol{w}^*, \boldsymbol{w}^{t-1} \rangle - \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{t-1} \rangle \eta \gamma_f}{\|\boldsymbol{w}^{t-1}\|} + \eta y_i \langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle.$$

Using the geometric margin definition we have

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^t \rangle \geqslant \langle \boldsymbol{w}^*, \boldsymbol{w}^{t-1} \rangle - \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^{t-1} \rangle \eta \gamma_f}{\|\boldsymbol{w}^{t-1}\|} + \eta \gamma(\boldsymbol{w}^*) \|\boldsymbol{w}^*\| \tag{3}$$

$$\geqslant \cdots \geqslant -\sum_{k=1}^{t-1} \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^k \rangle \eta \gamma_f}{\|\boldsymbol{w}^k\|} + t\eta\gamma(\boldsymbol{w}^*)\|\boldsymbol{w}^*\|, \tag{4}$$

where we applied the expression recursively and used the fact that $\boldsymbol{w}^0 \equiv 0$ (the first update was done using the regular perceptron update rule). Applying the Cauchy–Schwarz inequality:

$$\sum_{k=1}^{t-1} \frac{\langle \boldsymbol{w}^*, \boldsymbol{w}^k \rangle \eta \gamma_f}{\|\boldsymbol{w}^k\|} \leqslant \sum_{k=1}^{t-1} \frac{\|\boldsymbol{w}^*\| \|\boldsymbol{w}^k\| \eta \gamma_f}{\|\boldsymbol{w}^k\|} = (t-1)\|\boldsymbol{w}^*\| \eta \gamma_f, \tag{5}$$

substituting (5) into Eq. (4):

$$\langle \boldsymbol{w}^*, \boldsymbol{w}^t \rangle \geqslant \eta \|\boldsymbol{w}^*\| (t(\gamma(\boldsymbol{w}^*) - \gamma_f) + \gamma_f).$$

Similarly, from the update rule we have the following:

$$\|\boldsymbol{w}^t\|^2 = \|\boldsymbol{w}^{t-1}\|^2 \lambda_{t-1}^2 + 2\eta y_i \langle \boldsymbol{w}^{t-1}, \boldsymbol{x}_i \rangle \lambda_{t-1} + \eta^2 \|\boldsymbol{x}_i\|^2,$$

since there was a mistake for the ith pattern, we know that $y_i \langle \boldsymbol{w}^{t-1}, \boldsymbol{x}_i \rangle < \gamma_f \|\boldsymbol{w}^{t-1}\|$, also substituting R we get

$$\begin{aligned}
|\boldsymbol{w}^t\|^2 &\leqslant \|\boldsymbol{w}^{t-1}\|^2 \lambda_{t-1}^2 + 2\eta\gamma_f \|\boldsymbol{w}^{t-1}\| \lambda_{t-1} + \eta^2 \|\boldsymbol{x}_i\|^2 \\
&= \|\boldsymbol{w}^{t-1}\|^2 \lambda_{t-1} \left(\lambda_{t-1} + 2\frac{\eta\gamma_f}{\|\boldsymbol{w}^{t-1}\|} \right) + \eta^2 \|\boldsymbol{x}_i\|^2 \\
&\leqslant \|\boldsymbol{w}^{t-1}\|^2 \left(1 - \frac{\eta\gamma_f}{\|\boldsymbol{w}^{t-1}\|}\right)\left(1 + \frac{\eta\gamma_f}{\|\boldsymbol{w}^{t-1}\|}\right) + \eta^2 R^2 \\
&= \|\boldsymbol{w}^{t-1}\|^2 \left(1 - \frac{\eta^2\gamma_f^2}{\|\boldsymbol{w}^{t-1}\|^2}\right) + \eta^2 R^2 \\
&\leqslant \|\boldsymbol{w}^{t-1}\|^2 + \eta^2(R^2 - \gamma_f^2) \leqslant \cdots \leqslant (t-1)\eta^2(R^2 - \gamma_f^2) \\
&\quad + \eta^2 R^2 = \eta^2(t(R^2 - \gamma_f^2) + \gamma_f^2),
\end{aligned}$$

where we used the fact that $\lambda_{t-1} > 0$ and that $w^0 \equiv 0$. Using the Cauchy–Schwarz inequality again, we have

$$\begin{aligned}
&\eta\|\boldsymbol{w}^*\|(t(\gamma(\boldsymbol{w}^*) - \gamma_f) + \gamma_f) \leqslant \langle \boldsymbol{w}^*, \boldsymbol{w}^t \rangle \\
&\quad \leqslant \|\boldsymbol{w}^*\| \|\boldsymbol{w}^t\| \leqslant \|\boldsymbol{w}^*\| \eta \sqrt{t(R^2 - \gamma_f^2) + \gamma_f^2} \\
&\quad \Rightarrow (t(\gamma(\boldsymbol{w}^*) - \gamma_f) + \gamma_f) \leqslant \sqrt{t(R^2 - \gamma_f^2) + \gamma_f^2} \\
&\quad \Rightarrow t^2(\gamma(\boldsymbol{w}^*) - \gamma_f)^2 + \gamma_f^2 \leqslant (t(\gamma(\boldsymbol{w}^*) - \gamma_f) + \gamma_f)^2 \\
&\quad \leqslant t(R^2 - \gamma_f^2) + \gamma_f^2,
\end{aligned}$$

from which the bound follows. □

It is interesting to notice that this bound reflects an intuitive interpretation of the problem. The larger the fixed margin is, the more iteration FMP will take to converge, since the feasible solution space becomes more restricted making the problem more "difficult," and it will tend to infinity as $\gamma_f \to \gamma(\boldsymbol{w}^*)$. Also, if the given fixed margin is set to zero, the bound is the same as for the perceptron algorithm.

4. Incremental margin algorithm (IMA)

The maximal margin problem can be stated as finding $\boldsymbol{w}^* \in \mathscr{F}$ such that

$$\boldsymbol{w}^* = \arg \max_{\boldsymbol{w} \in \mathscr{V}(z_m)} \gamma(\boldsymbol{w}).$$

A well-known result for this problem states that $\gamma^+(\boldsymbol{w}^*) = \gamma^-(\boldsymbol{w}^*) = \gamma(\boldsymbol{w}^*)$, where

$$\gamma^+(\boldsymbol{w}) = \min\{\gamma_i | \gamma_i = y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle / \|\boldsymbol{w}\|, y_i = +1, i = 1, \ldots, m\},$$

$$\gamma^-(\boldsymbol{w}) = \min\{\gamma_i | \gamma_i = y_i \langle \boldsymbol{x}_i, \boldsymbol{w} \rangle / \|\boldsymbol{w}\|, y_i = -1, i = 1, \ldots, m\}.$$

This result can be verified by the Karush–Kuhn–Tucker conditions obtained from Vapnik's SVM formulation [2]. Therefore, if we have an element $\boldsymbol{w} \in \mathscr{V}(z_m)$ such that $\gamma^+(\boldsymbol{w}) \neq \gamma^-(\boldsymbol{w})$ we know it is not optimum. Additionally, $\boldsymbol{w}^*$ is also a solution for the problem of maximizing the distance between classes, given by

$$\boldsymbol{w}^* = \arg \max_{\boldsymbol{w} \in \mathscr{V}(z_m)} (\gamma^+(\boldsymbol{w}) + \gamma^-(\boldsymbol{w})).$$

Consequently, it follows that

$$2\gamma(\boldsymbol{w}^*) \geqslant \gamma^+(\boldsymbol{w}) + \gamma^-(\boldsymbol{w}), \quad \forall \boldsymbol{w} \in \mathscr{V}(z_m). \tag{6}$$

Based on these results, the IMA was constructed. The main idea of IMA is to create a series of increasing fixed margin values, which are obtained through the solution of successive FMP problems. Initially, the fixed margin is set to zero and it is consistently increased until it approximates the maximal margin value.

Notice that the FMP can be thought as an algorithm that finds an element $\boldsymbol{w}$ in version space $\mathscr{V}(z_m, \gamma_{\rm f})$. In this way, IMA constructs a series of version spaces $\mathscr{V}(z_m, \gamma_{\rm f}^0) \supset \mathscr{V}(z_m, \gamma_{\rm f}^1) \supset \mathscr{V}(z_m, \gamma_{\rm f}^2) \supset \mathscr{V}(z_m, \gamma_{\rm f}^3) \supset \cdots \supset \mathscr{V}(z_m, \gamma_{\rm f}^d)$, where $0 = \gamma_{\rm f}^0 < \gamma_{\rm f}^1 < \gamma_{\rm f}^2 < \gamma_{\rm f}^3 < \cdots < \gamma_{\rm f}^d$ and $\gamma_{\rm f}^d \approx \gamma(\boldsymbol{w}^*)$. The FMP is used successively to find a new element $\boldsymbol{w}^j$ inside each version space $\mathscr{V}(z_m, \gamma_{\rm f}^j)$, $j = 1, \ldots, d$, where each new fixed margin $\gamma_{\rm f}^j$ is obtained from the previous solution $\boldsymbol{w}^{j-1}$ using

$$\gamma_{\rm f}^j = \tfrac{1}{2}[\gamma^+(\boldsymbol{w}^{j-1}) + \gamma^-(\boldsymbol{w}^{j-1})].$$

The result given by (6) guarantees that the new margin $\gamma_{\rm f}^j$ always satisfies $\gamma_{\rm f}^j \leqslant \gamma(\boldsymbol{w}^*)$, hence the new version space $\mathscr{V}(z_m, \gamma_{\rm f}^j)$ will not be empty.

There are still two remaining problems. The condition $\gamma^+(\boldsymbol{w}^*) = \gamma^-(\boldsymbol{w}^*)$ for $\boldsymbol{w}^*$ to be optimum is necessary but not sufficient. Therefore, it cannot be used as a stop criterion. Furthermore, if by chance $\gamma_{\rm f}^j = \gamma(\boldsymbol{w}^*)$, FMP may never converge. Thus, a stop criterion must be defined. Also, we might have a non-optimum solution $\boldsymbol{w}^j$ where the condition is attended and therefore the new margin $\gamma_{\rm f}^{j+1}$ will not be different from $\gamma_{\rm f}^j$.

In order to solve the latter problem, we consider a new margin update rule given by

$$\gamma_{\rm f}^j = \max(\tfrac{1}{2}[\gamma^+(\boldsymbol{w}^{j-1}) + \gamma^-(\boldsymbol{w}^{j-1})], (1+\delta)\gamma_{\rm f}^{j-1}), \tag{7}$$

where $\delta \in (0, 1)$ is a minimum margin increment assigned by the user. Using this new margin update, we guarantee that the fixed margin values are monotonic increasing since $\boldsymbol{w}^i \in \mathscr{V}(z_m, \gamma_{\rm f}^i)$ for every $i \geqslant 1$, which implies that $\gamma^+(\boldsymbol{w}^i), \gamma^-(\boldsymbol{w}^i) \geqslant \gamma_{\rm f}^i$. Therefore, using the fact that $\frac{1}{2}[\gamma^+(\boldsymbol{w}^{j-1}) + \gamma^-(\boldsymbol{w}^{j-1})] \geqslant \gamma_{\rm f}^{j-1}$ and Eq. (7), we have

$$\gamma_{\rm f}^j \geqslant \max(\gamma_{\rm f}^{j-1}, (1+\delta)\gamma_{\rm f}^{j-1}) = (1+\delta)\gamma_{\rm f}^{j-1} > \gamma_{\rm f}^{j-1}.$$

Observe that by adopting this increment, we might end up with an empty version space and the FMP will not converge. With this in mind, also trying to solve the stop criterion problem, we stipulate a maximum number of iteration for the FMP to converge. If the algorithm does not converge in the given number of iterations, IMA will return the solution for the last problem solved. Therefore, the value δ should be carefully chosen in order not to interfere in the incremental process. An adequate choice for this parameter would be a value proportional to the desired margin approximation, that is, if the user wants to have an α-approximation of the optimum margin (i.e. the final margin is greater or equal than $(1-\alpha)\gamma(\boldsymbol{w}^*)$, $\alpha \in (0, 1)$), then δ should be set to the value of α. To prove this statement, consider that for some $j \geqslant 1$ we have $\gamma_{\rm f}^{j-1} \in (0, \gamma(\boldsymbol{w}^*))$, $\gamma_{\rm f}^j = (1+\alpha)\gamma_{\rm f}^{j-1}$ and $\gamma_{\rm f}^j \geqslant \gamma(\boldsymbol{w}^*)$. Hence, the algorithm will return the last feasible solution $\boldsymbol{w}^{t-1}$. Clearly, $\gamma(\boldsymbol{w}^{j-1}) \geqslant \gamma_{\rm f}^{j-1}$ since $\boldsymbol{w}^{j-1} \in \mathscr{V}(z_m, \gamma_{\rm f}^{j-1})$. Therefore,

$$\gamma(\boldsymbol{w}^{j-1}) \geqslant \gamma_{\rm f}^{j-1} \geqslant \frac{\gamma(\boldsymbol{w}^*)}{(1+\alpha)} > (1-\alpha)\gamma(\boldsymbol{w}^*).$$

One more observation to be made is that after each FMP solution we use the last normal vector $\boldsymbol{w}^{j-1}$ as a starting point for the next FMP problem. This strategy has the advantage that FMP needs to make fewer corrections to satisfy the new margin. The IMA is given in Algorithm 2. Also, the following result establishes a bound for the number of updates made by IMA.

Algorithm 2. Incremental margin algorithm.

```
1: **Input**: $z_m$, $\eta$, $\delta$, $T\_MAX$
2: $\boldsymbol{w} \leftarrow 0$, $\gamma_{\rm f} \leftarrow 0$
3: **repeat**
4:     $\boldsymbol{w} \leftarrow$ FMP($z_m$,$\boldsymbol{w}$,$\gamma_{\rm f}$,$\eta$,$T\_MAX$)
5:     $\gamma_{\rm f} \leftarrow \max((\gamma^+(\boldsymbol{w}) + \gamma^-(\boldsymbol{w}))/2, (1+\delta)\gamma_{\rm f})$
6: **until** the convergence of FMP in $T\_MAX$ iterations is
   not achieved
7: return last feasible $\boldsymbol{w}$
```

Corollary 4.1. *Given a linearly separable training set z_m, suppose that our solution $\boldsymbol{w}_{\rm f}$ satisfies $\gamma(\boldsymbol{w}_{\rm f}) = (1-\alpha)\gamma(\boldsymbol{w}^*)$, $\alpha \in (0, 1)$, then the number of updates made by IMA is in the order of*

$$\mathcal{O}\left(\frac{R^2}{\alpha^2 \gamma(\boldsymbol{w}^*)^2}\right),$$

where $R = \max_{i \in \{1,\ldots,m\}} \|\boldsymbol{x}_i\|$, $\gamma(\boldsymbol{w}^*) = \max_{\boldsymbol{w} \in \mathscr{V}(z_m)}(\gamma(\boldsymbol{w}))$.

Proof. The proof follows straight from Theorem (3.2) substituting $\gamma_{\rm f} = (1-\alpha)\gamma(\boldsymbol{w}^*)$ and using the fact that the bound for the number of updates for the last FMP is larger than the bound for any other previous FMP. □

One interesting observation that can be made from the above corollary is that for every choice of $\alpha \in (0, 1)$ we have that

$$t \leqslant \frac{C^2(z_m)}{\alpha^2} \Rightarrow (1-\alpha) \geqslant 1 - \frac{C(z_m)}{\sqrt{t}},$$

where $C(z_m) = \widetilde{C}(R/\gamma(\boldsymbol{w}^*)) > 0$, is a constant depending only on the set z_m. This gives us a nice relationship between the quality of approximation $(1-\alpha)$ and number of updates t.

Gentile [4] proposed a large margin on-line algorithm, called ALMA$_p$, which is similar to the algorithm proposed here. Internally, ALMA$_p$ uses a generalization of the perceptron algorithm named p-norm algorithm [5]. For $p = 2$, its update rule is equivalent to the perceptron's followed by a normalization step to control the growth of the norm. This normalization is necessary since the functional margin $\varGamma(\boldsymbol{w})$ is used instead of the geometric margin $\gamma(\boldsymbol{w})$. One important difference between the two algorithms is that IMA does its margin updates in batch mode after each FMP solution is found as opposed to ALMA, which updates its margin value in an on-line fashion after each correction. Contrary to IMA, ALMA$_p$

starts out imposing a high margin value and it is iteratively reduced until a feasible solution is found. One significant advantage of IMA's batch strategy to gradually increase the margin is that it always provides a feasible solution at hands.

5. Algorithm in dual variables

In this section, we will describe the extension of the FMP in dual variables. Learning algorithms in dual variables are usually employed when the function $\Phi : \mathcal{X} \to \mathcal{F}$ is not known explicitly. All that is known is a mercer kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is the inner product in feature space, i.e. $k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle_{\mathcal{F}} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle_{\mathcal{F}}$, for $x_i, x_j \in \mathcal{X}$. In order to use this dual representation, we write $\boldsymbol{w}$ as follows:

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i,$$

where the nonnegative values $\{\alpha_i\}_{i=1}^{m}$, $\alpha_i \in \mathbb{R}_{\geqslant 0}$ were introduced (see representer theorem, e.g. [7]). Therefore, the corresponding hypothesis is written as

$$h_w(\cdot) = \text{sign}\left(\left\langle \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i, \cdot \right\rangle\right) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i \langle \boldsymbol{x}_i, \cdot \rangle\right)$$
$$= \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i k(x_i, \cdot)\right).$$

For the perceptron algorithm, the update rule in dual variables is given by

$$\alpha_i \leftarrow \alpha_i + \eta, \tag{8}$$

for a mistake in the ith example. In this sense, the multipliers α_i can be thought as the number of corrections made for each example in z_m. We construct the new update rule for FMP from the update rule in primal variables, given by Eq. (1). Notice that the vector $\boldsymbol{w}^t$ is scaled by a factor of λ_t before a correction takes place. This multiplication in dual variables is given by

$$\boldsymbol{w}^t \lambda_t = \sum_{i=1}^{m} \lambda_t \alpha_i y_i \boldsymbol{x}_i.$$

Since the α_i's are the only non-constant values in this equation, this multiplication is equivalent to multiplying all α_i by the scale factor λ_t. After all multipliers α_i are scaled, the correction is equivalent to the perceptron update rule, given by (8).

The scale factor λ_t brings us an interesting effect. By scaling down each multiplier before every update, we are in essence choosing the support vectors. In other words, those multipliers that were changed when the algorithm started and are not associated to any support vector tend to be corrected less often and their values are eventually brought to zero.

In order to have more control on the decay of the multipliers α_i we propose another version of the dual algorithm where a new parameter σ is introduced. Instead of multiplying the α_i values, we subtract σ from them before every new perceptron iteration is started, making sure that the property $\alpha_i \geqslant 0$ is respected. With this parameter, the user has more control on the decay of the α_i values, and therefore, on the number of support vectors.

One thing still remains to be defined completely in dual variables. In order to compute the norm $\|\boldsymbol{w}\|$, one can do the following:

$$\|\boldsymbol{w}\|^2 = \left\langle \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i, \sum_{j=1}^{m} \alpha_j y_j \boldsymbol{x}_j \right\rangle = \sum_{i,j=1}^{m} \alpha_i y_i \alpha_j y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$$
$$= \sum_{i,j=1}^{m} \alpha_i y_i \alpha_j y_j k(x_i, x_j).$$

For computational efficiency, it is possible to compute the above value after each update. For example, after the update for a mistake in the ith pattern, we can write $\|\boldsymbol{w}^{t+1}\|$ using the update rule (1) as

$$\|\boldsymbol{w}^{t+1}\|^2 = \lambda_t^2 \langle \boldsymbol{w}^t, \boldsymbol{w}^t \rangle + 2\eta\lambda_t y_i \langle \boldsymbol{w}^t, \boldsymbol{x}_i \rangle + \eta^2 \langle \boldsymbol{x}_i, \boldsymbol{x}_i \rangle$$
$$= \lambda_t^2 \|\boldsymbol{w}^t\|^2 + 2\eta\lambda_t y_i \langle \boldsymbol{w}^t, \boldsymbol{x}_i \rangle + \eta^2 k(x_i, x_i),$$

where $y_i \langle \boldsymbol{w}^t, \boldsymbol{x}_i \rangle = y_i \sum_{j=1}^{m} \alpha_j^t y_j k(x_j, x_i)$ was already computed. The final algorithm is given by Algorithm 3.

Algorithm 3. Dual FMP algorithm.

```
1:  **Input**: $z_m$, $\boldsymbol{\alpha}_{\text{init}}$, $\gamma_f$, $\eta$, $T\_MAX$
2:  $\boldsymbol{\alpha}^0 \leftarrow \boldsymbol{\alpha}_{\text{init}}$, $t \leftarrow 0$
3:  **repeat**
4:    **for** ($i = 1, \ldots, m$) **do**
5:      **if** ($y_i \sum_{j=1}^{m} \alpha_j^t y_j k(x_j, x_i) < \gamma_f \|\boldsymbol{w}^t\|$) **then**
6:        $\boldsymbol{\alpha}^{t+1} \leftarrow \lambda_t \boldsymbol{\alpha}^t$
7:        $\alpha_i^{t+1} \leftarrow \alpha_i^t + \eta$
8:        $t \leftarrow t + 1$
9:      **end if**
10:   **end for**
11: **until** (no mistakes were made) or ($t > T\_MAX$)
12: return $\boldsymbol{\alpha}^t$
```

Notice that given a training set z_m we can construct a matrix $K \in \mathbb{R}^{m \times m}$ such that $K_{ij} = k(x_i, x_j)$ for $j, i = 1, \ldots, m$. In order to consider the soft margin case, which treats the possibility of margin violation for data outliers, we follow the approach described by Smola and Scholkopf [14]. A new parameter $\lambda^{\text{diag}} \in \mathbb{R}_{>0}$ is introduced and we redefine the kernel matrix by $K = K + I\lambda^{\text{diag}}$, where I is the identity matrix. According to Smola and Scholkopf, this procedure is equivalent to the minimization of the Euclidean norm of the slack vector in the SVM formulation.

There is a close relation between FMP and some regularized risk minimization algorithms such as Norma, introduced by Kivinen et al. [9]. In order to present this relation, we will briefly introduce the regularized risk functionals and reproducing kernel Hilbert space formalism. Let $f : \mathcal{X} \to \mathbb{R}$ with $f \in \mathbf{H}$. Here, $\mathbf{H}$ is a reproducing kernel Hilbert space (RKHS) [14], which means that there exists a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(\cdot,\cdot)$ has the reproducing property and $\mathbf{H}$ is the closure of the span of all $k(x,\cdot)$. The inner product in $\mathbf{H}$ is defined as

$$\langle f, g\rangle_{\mathbf{H}} = \sum_{i=1}^{m}\sum_{j=1}^{m'} \beta_i\beta'_j k(x_i, x'_j),$$

for $f(\cdot) = \sum_{i=1}^{m}\beta_i k(x_i,\cdot)$ and $g(\cdot) = \sum_{j=1}^{m'}\beta'_j k(x_j,\cdot), f, g \in \mathbf{H}$. Regularized risk minimization algorithms are based on finding a solution that minimizes the regularized risk, given by

$$R_{\text{reg}}[f, z_m] = R_{\text{emp}}[f, z_m] + \lambda\Omega[f], \tag{9}$$

where $\lambda > 0$ is a constant left for the user to choose appropriately for each problem and $\Omega : \mathbf{H} \to \mathbb{R}_{\geqslant 0}$ is called the regularizer. The empirical risk is defined as

$$R_{\text{emp}}[f, z_m] = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i),$$

where $l : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is the loss function.

In order to derive Norma for classification, Kivinen et al. [9] proposed the choice of $\Omega[f] = \frac{1}{2}\|f\|^2_{\mathbf{H}}$ and the loss function $l(f(x), y) = \max(0, \rho - yf(x))$. Following the stochastic gradient approach, the update rule for Norma became

$$f \leftarrow \begin{cases} f(1-\eta\lambda) + \eta y_i \boldsymbol{x}_i & \text{if } y_i f(x_i) \leqslant \rho, \\ f(1-\eta\lambda) & \text{otherwise,} \end{cases}$$

which turns out to be very similar to the update rule for FMP. This is because both algorithms control the growth of the norm during the update. If we, instead of using $\Omega = \frac{1}{2}\|f\|^2_{\mathbf{H}}$, use $\Omega = \|f\|_{\mathbf{H}}$ and pick $\lambda = \gamma_{\text{f}}$ the update rule turns out to be even more similar to FMP's. Nevertheless, there is still something that makes FMP different from other regularized risk minimization algorithms, specially since it only updates the solution if the pattern yields a mistake as opposed to Norma. This difference comes from that fact that the regularizer $\Omega[\cdot]$ is part of the loss function, and for that reason, the FMP cannot be written in the usual form given by Eq. (9). Therefore, although FMP is essentially an empirical risk minimization algorithm it has imbedded in its loss function a regularizer.

6. Data sets and results

In order to test IMA, we applied it to several different linearly and nonlinearly separable data sets. Four linearly separable problems were considered. The first problem used was the iris data set from UCI repository, using Setosa versus others. The data set consists of 150 examples and four attributes. The following two data sets were obtained from UCI KDD archive, namely the robot execution failure (data set lp4) and the synthetic control chart time series data sets, which we will call robot and synthetic data sets, respectively. The robot data set is composed of 117 instances and 90 attributes, with the classes divided into normal versus collision and obstruction. The synthetic data set has 600 examples and 60 attributes, and the classes were split into types A, D and F versus B, C and D. The last set was an artificial one sampled from normal distributions with standard deviation

Table 1
Comparison of algorithms IMA, ALMA_2 and SVM by number of iterations (passes through the data set), updates (normal vector corrections) and margin value

Algorithm	Iterations	Updates	Margin
Iris			
SVM	–	–	0.12348
IMA	2	2	0.10506
IMA	921	2462	0.11693
IMA	2970	7391	0.12296
ALMA(.1)	1691	6890	0.10843
ALMA(.2)	656	1871	0.09250
ALMA(.3)	369	906	0.07769
ALMA(.4)	240	547	0.06410
ALMA(.5)	174	371	0.05229
ALMA(.6)	138	273	0.04065
ALMA(.7)	108	213	0.02948
ALMA(.8)	89	175	0.01960
Synthetic			
SVM	–	–	0.01692
IMA	48	990	0.00760
IMA	372	5389	0.01193
IMA	422	5961	0.01220
IMA	606	8107	0.01265
ALMA(.1–.6)	–	10 000	–*
ALMA(.7)	2625	7550	0.00512
ALMA(.8)	2037	5530	0.00360
Robot			
SVM	–	–	0.15122
IMA	50	180	0.11795
IMA	350	1452	0.13588
IMA	1177	4947	0.13844
ALMA(.06)	–	10 000	–*
ALMA(.07)	1335	9894	0.13424
ALMA(.09)	872	6035	0.13062
ALMA(.1)	728	4877	0.13078
ALMA(.2)	247	1253	0.11632
ALMA(.3)	142	574	0.10839
ALMA(.4)	81	299	0.09484
ALMA(.5)	57	190	0.07641
Toy			
SVM	–	–	0.03920
IMA	37	750	0.02934
IMA	222	4216	0.03596
IMA	396	6818	0.03753
ALMA(.1–.2)	–	10000	–*
ALMA(.3)	749	5422	0.03226
ALMA(.4)	920	3946	0.02468
ALMA(.5)	331	1968	0.02330
ALMA(.6)	238	1349	0.01896
ALMA(.7)	155	918	0.01736
ALMA(.8)	171	771	0.00939

*Convergence was not achieved in the desired number of updates.

$\sigma = 5.0$ and mean $\mu = -1$ for the negative class and $\mu = +1$ for the positive class. A total of 1000 points were sampled for each class, each with 10 features. All the data sets were normalized before training, and the parameters used were set to $\eta = 1$ and $\delta = 10^{-3}$. The training set was also randomly shuffled before every pass through the data set.

We compared the margin values for IMA in primal variables to the maximal margin found by Joachim's SVM-light [8], and to the values obtained with ALMA$_p$ using Euclidean norm ($p = 2$) for varying values of α, reporting results up to be best value obtained under 10^4 normal vector updates. For ALMA$_2$, we used the parameters $B = 1/\alpha$ and $C = \sqrt{2}$, which are suggested by Gentile [4]. The results are given in Table 1. Notice that IMA is reported at several stages before it reaches 10^4. These stages are the solutions obtained using FMP with different fixed margins acquired at the same run. From the results, we can see that IMA returns superior results under fewer updates than the ones obtained by ALMA$_2$. Also, ALMA$_2$ is completely dependent on the correct choice of parameter α, if set too high it might yield a poor approximation and if set too low it might not converge in the desired time.

In order to give an idea of the running times for IMA in these linearly separable data sets, we report the time taken by IMA to reach the highest margin value reported in Table 1. For data sets, iris, robot, synthetic, and toy the times were 0.04, 0.07, 0.28 and 0.21 s, respectively, where the system overhead time was not accounted for. The algorithm was implemented in C, compiled using GNU's gcc and was tested on a Mobile Intel Celeron CPU 2.40 GHz.

Six nonlinearly separable data sets from UCI's repository were also used: bupa, ionosphere, Pima Indians diabetes, sonar, WDBC (Wisconsin Breast Cancer Databases) and wine. Each of these sets are composed of 345, 351, 768, 208, 569 and 178 examples and 6, 34, 8, 60, 30, and 13 attributes, respectively. For the wine data set, we divided the classes into class 1 and 3 versus class 2. For these sets we used the Gaussian kernel

Table 2
Report of number of FMP calls, iterations (passes through the data set), updates (number of corrections), margin values, number of support vectors and running time in seconds for the nonlinearly separable data sets

Algo.	FMPs/It./Up.	Margin	SVs	Sec.
Bupa				
SVM	–	0.03327	283	0.21
IMA	5/96/1123	0.01078	311	0.01
IMA	30/1202/9569	0.02768	322	0.06
IMA	42/3666/29513	0.03039	297	0.17
Pima				
SVM	–	0.03479	724	0.45
IMA	55/180/1428	0.01572	702	0.09
IMA	115/648/9997	0.02920	747	0.17
IMA	164/1711/47 948	0.03212	738	0.44
WDBC				
SVM	–	0.04975	324	0.26
IMA	19/124/1176	0.03452	358	0.06
IMA	39/591/8120	0.04362	436	0.12
IMA	50/1711/28 722	0.04652	378	0.25
Ionosphere				
SVM	–	0.07662	224	0.07
IMA	14/73/738	0.05491	241	0.01
IMA	32/351/5215	0.06870	256	0.03
IMA	44/816/14 532	0.07201	244	0.07
Sonar				
SVM	–	0.07720	152	0.05
IMA	20/197/2595	0.06688	185	0.01
IMA	31/631/10 556	0.07185	173	0.03
IMA	37/1138/19 501	0.07308	165	0.05
Wine				
SVM	–	0.03249	93	0.20
IMA	3/95/1065	0.01957	129	0.00
IMA	11/811/9552	0.02752	154	0.02
IMA	24/6941/87 132	0.03078	120	0.20

Table 3
Report of accuracy tests for all data sets, linearly and nonlinearly separable. The margin values and times in seconds are averages over 100 trials

Algo.	Avg. Margin	Accuracy	Avg. Time
Iris			
IMA	0.0948927	100%	0.0*
SVM	0.1249025	100%	0.0*
Synthetic			
IMA	0.008059	99.5%	0.0315
SVM	0.017604	99.4%	0.0629
Bupa			
IMA	0.0305294	64.0%	0.0567
SVM	0.0361274	63.7%	0.1135
Pima			
IMA	0.031564	64.3%	0.1700
SVM	0.037049	64.4%	0.3401
WDBC			
IMA	0.0458015	92.3%	0.0925
SVM	0.0527047	92.2%	0.1852
Robot			
IMA	0.1309316	98.0%	0.0009
SVM	0.1614046	98.0%	0.0019
Toy			
IMA	0.035783	100%	0.3451
SVM	0.039582	100%	0.6903
Ionosphere			
IMA	0.0674429	92.4%	0.0225
SVM	0.0798914	92.5%	0.0451
Sonar			
IMA	0.0723901	86.3%	0.0169
SVM	0.0802006	86.2%	0.0339
Wine			
IMA	0.0316397	83.4%	0.0585
SVM	0.0346797	83.1%	0.1171

*Times for each individual run fell below 10^{-2} s of precision.

$k(x,y) = \exp(-\theta\|x-y\|^2)$. The kernel parameter θ was chosen empirically to improve accuracy tests for SVM. The values chosen were, 0.01, 1, 0.01, 1, 0.001 and 0.001 for bupa, ionosphere, pima, sonar, WDBC and wine, respectively. IMA parameters were the same as the ones used in the linearly separable case. The results are given in Table 2. Notice again that IMA is reported at different stages of the same run, each of these stages is a feasible solution for the problem. Also, we can see from the results that IMA converges rapidly to a large margin solution. The running times reported in Table 2 are under the same conditions described in the linearly separable case.

In order to show that one can have good classification performance with a large but not optimum margin we perform an accuracy test. The test was done as follows: each data set was randomly split into a training and a test file, the latter containing 10% of the entries. Each classifier was constructed using the training file and it was tested against the test file, using both SVM and IMA on the same files. For IMA, we interrupted the algorithm after it reached half of the training time taken by the SVM-light on the same training set. This procedure was repeated 100 times and the averages computed. The results are given in Table 3. Observe that, the FMP algorithm has the same computational complexity of a perceptron, that is, in primal variables each pass through the training set is of the order of $\mathcal{O}(md)$, where m is the number of examples in the data set and d its dimension. In dual variables, the algorithm can be optimized to have worst case complexity of $\mathcal{O}(m^2)$, where the worst case is when the algorithm updates its hypothesis for every example (see, for instance, [7]). Therefore, for the linearly separable data sets, we are able to choose FMP in primal or in dual variables depending on whether dimension d is considerably less than the number of examples m. In this sense, we used FMP in primal variables for the iris, synthetic and toy data sets. The source code for IMA in primal and dual variables used in this paper is available upon request.

The plots in Figs. 2 and 3 were obtained running IMA up to 10^3 passes through the training set. They show how the quality of approximation, given by $(\gamma_f/\gamma(\mathbf{w}^*)) \times 100\%$, where $\mathbf{w}^*$ is the maximal margin solution, grows with the

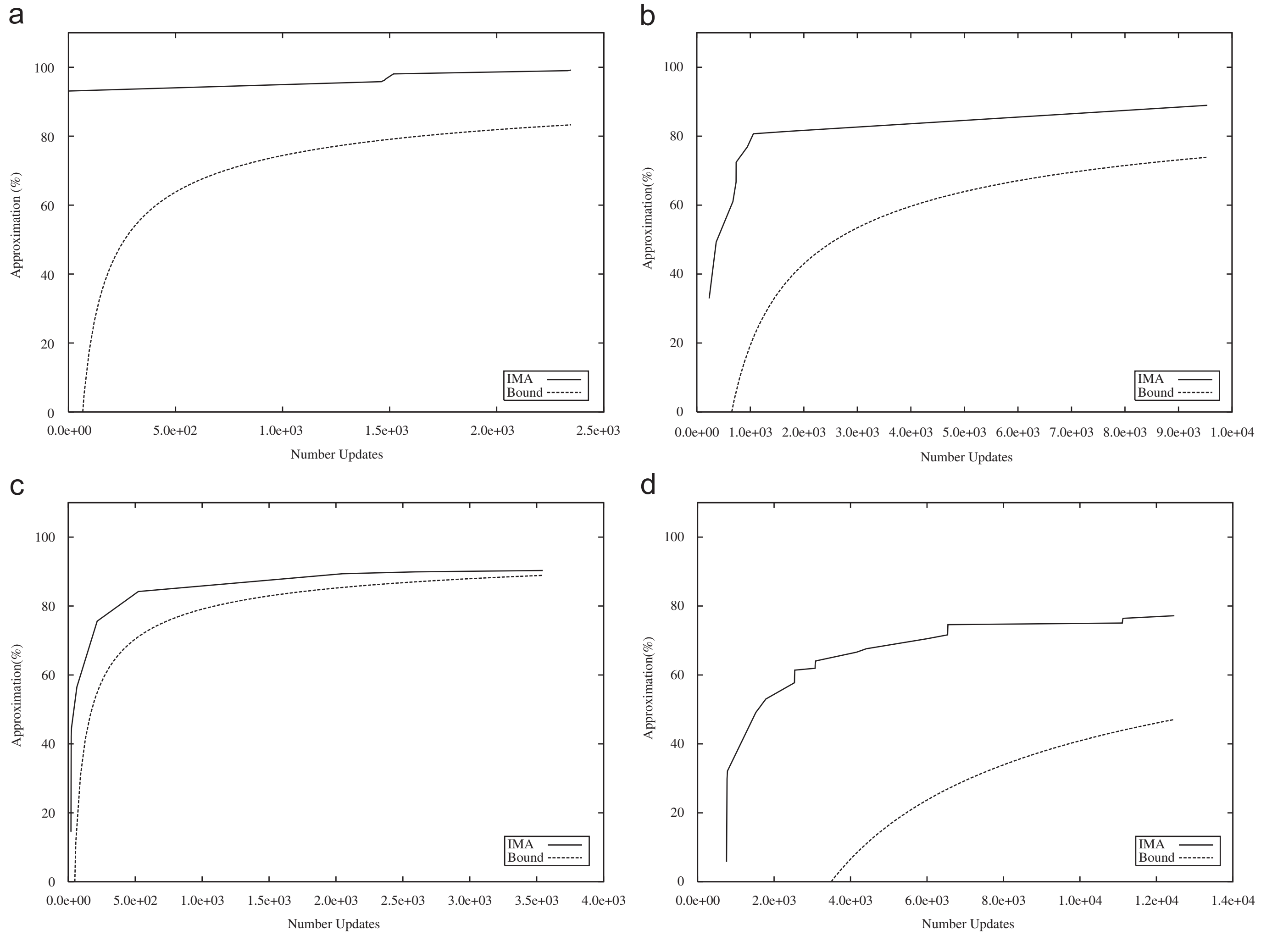


Fig. 2. Plot of quality of margin approximation, $(\gamma_f/\gamma(\mathbf{w}^*)) \times 100\%$, versus number of iterations (i.e. number of passes through the training set) for the linearly separable data sets: (a) iris; (b) toy; (c) robot; and (d) synthetic.

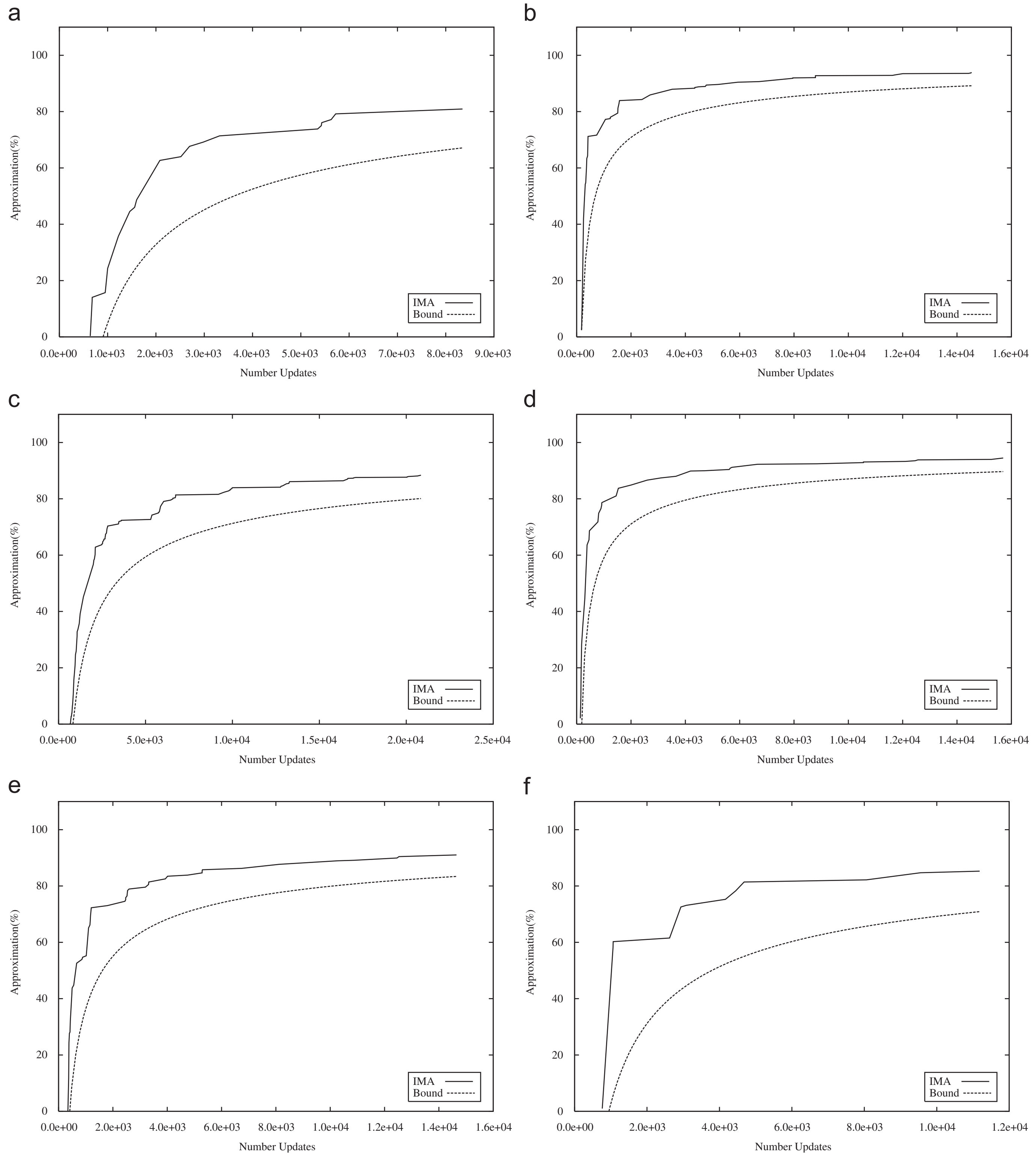


Fig. 3. Plot of quality of margin approximation, $(\gamma_f/\gamma(\boldsymbol{w}^*)) \times 100\%$, versus number of iterations (i.e. number of passes through the training set) for the nonlinearly separable data sets: (a) bupa; (b) ionosphere; (c) pima; (d) sonar; (e) WDBC; and (f) wine.

number of updates. It is interesting to notice that each one of the graphs follows the same shape of curve. This can be understood with the aid of the lower bound $\gamma_f/\gamma(\boldsymbol{w}^*) = (1-\alpha) \geqslant 1 - C(z_m)/\sqrt{t}$ derived in Section 4. In order to plot this bound, we had to compute the constant $C(z_m) = \widetilde{C}(R/\gamma(\boldsymbol{w}^*))$. From our experiments, $\gamma(\boldsymbol{w}^*)$ was known, and R was easily computed in primal and dual variables (notice that $R = \max_{i \in \{1,\ldots,m\}} k(x_i, x_i)^2$). What remained was the constant $\widetilde{C}$. From Corollary 4.1, one can see that this constant is associated with the number of times IMA calls FMP from a zero starting point, that is, beginning with weight vector $\boldsymbol{w}^0 = 0$ (or $\boldsymbol{\alpha}^0 = 0$ in dual variables). Since, in our implementation of IMA, the solution of the previous FMP is used to start the next, we supposed, simply as an approximation for these plots, that $\widetilde{C} = 1$. The resulting bound, for most cases, turns out to be very close to the real behavior of the algorithm.

7. Discussion

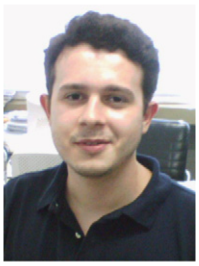
In this paper, we proposed a new algorithm for the construction of large margin classifiers in dual and primal variables. With this algorithm, it was possible to obtain excellent approximations to the maximal margin solution. Also, the algorithm is based entirely on the perceptron which makes it simple to understand and implement. From the computational experiments, the algorithm has shown to be efficient and robust. Additionally, after a few iterations, the algorithm always has a feasible solution at hand. This way, it can be interrupted at any time returning an approximation of the maximal margin solution.

Acknowledgments

The authors would like to thank CNPq, Conselho Nacional de Desenvolmento Científico e Tecnológico for the financial support and the LNCC's Bioinformatics Laboratory where most of this research took place.

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, in: Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, 1992, pp. 144–152.
[2] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discovery 2 (1998) 121–167.
[3] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed., Wiley, New York, 2000.
[4] C. Gentile, A new approximate maximal margin classification algorithm, J. Mach. Learn. Res. 2 (2001) 213–242.
[5] C. Gentile, N. Littlestone, The robustness of the *p*-norm algorithms, in: Proceedings of the 12th Annual Conference on Computer Learning Theory, 1999, pp. 1–11.
[6] E. Harrington, R. Herbrich, J. Kivinen, J.C. Platt, R.C. Williamson, Online Bayes point machines, in: Lecture Notes in Computer Science, Springer, Berlin, 2003, pp. 241–252.
[7] R. Herbrich, Learning Kernel Classifiers Theory and Algorithms, MIT Press, London, 2002.
[8] T. Joachims, Making large-scale SVM learning practical, in: B. Scholkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, 1999.
[9] J. Kivinen, A.J. Smola, R.C. Williamson, Online learning with kernels, Trans. Signal Process. 52 (8) (2004) 2165–2176.
[10] Y. Li, P. Long, The relaxed online maximum margin algorithm, in: S.A. Solla, T.K. Leen, K.R. Muller (Eds.), Advances in Neural Information Processing System, vol. 12, MIT Press, Cambridge, 2000, pp. 498–504.
[11] A. Novikoff, On the convergence proofs for perceptron, in: Report at the Symposium on Mathematical Theory of Automata, vol. 22 , 1962, pp. 615–622.
[12] J.C. Platt, Fast training of support vector machines using sequential minimal optimization, in: B. Scholkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods—Support Vector Learning, MIT Press, Cambridge, 1999.
[13] F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, Psychol. Rev. 65 (1958) 386–407.
[14] B. Scholkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, 2002.
[15] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers. Neural Process. Lett. 9 (3) (1999) 293–300.
[16] V.N. Vapnik, The Nature of Statistical Learning Theory, third ed., Springer, New York, 1995.



Saul C. Leite received his B.Sc. degree in Computer Science from the Oklahoma State University in 2002. Since 2005, he is pursuing the Ph.D. degree in Computational Modelling at the National Laboratory for Scientific Computing (LNCC), Brazil. His current interests include stochastic networks, queueing theory, linear classifiers, kernel methods, and bioinformatics.

Raul Fonseca Neto graduated in Civil Engineering from the Federal University of Juiz de Fora in 1983. In 1986, he received the Master's degree in Engineering of Transports from the Military Institute of Engineering, and in 1990, the Ph.D. degree in Engineering of Systems and Computation from the Federal University of Rio de Janeiro. Currently, he is an associate professor IV of the Federal University of Juiz de Fora and coordinator of the Computer science undergraduate course at the same University. His main interests include: artificial intelligence, optimization, network flow, planning and scheduling, algorithm complexity, machine learning, pattern recognition, and bioinformatics.