

Modificações no Modelo Adaline para o Treinamento de Least Squares Support Vector Machines

Bernardo Penna Resende de Carvalho
Mestrando do PPGEE
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, Belo Horizonte, MG
bpenna@ufmg.br

Antônio de Pádua Braga
Departamento de Engenharia Eletrônica
Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, Belo Horizonte, MG
apbraga@cpdee.ufmg.br

Abstract—Two new strategies are introduced in this work for solving the system of linear equations intrinsic to the LS-SVMs, that has difficult solution by the fact that it isn't positive definite. These strategies are based on two modifications at the neural model Adaline, in order to eliminate the LS-SVMs' greatest drawback when comparing to SVMs: the inexistence of the support vectors' automatic detection. Least Squares Support Machines Vector (LS-SVMs) were created in 1999, corresponding to a modified version of Support Machines Vector (SVMs), developed in 1992. The optimization problem for the LS-SVMs can be solved from a system of linear equations, instead of quadratic programming as it occurs in the SVMs, without quality loss in the solution. In this work, we have proposed the strategies *Adaline2* and *Adaline3* as a manner to enjoy the benefits that LS-SVMs offer, adding to them the automatic detection of support vectors.

Resumo—São introduzidas neste trabalho duas novas estratégias para a resolução do sistema de equações lineares intrínseco às LS-SVMs, que possui difícil solução pelo fato de não ser definido positivo. Estas estratégias se baseiam em duas modificações no modelo neural Adaline, que têm como objetivo eliminar a maior desvantagem das LS-SVMs em relação às SVMs: a inexistência da detecção automática dos vetores de suporte. Least Squares Support Vector Machines (LS-SVMs) foram criadas em 1999, correspondendo a uma versão modificada de Support Vector Machines (SVMs), desenvolvidas em 1992. O problema de otimização gerado pelas LS-SVMs pode ser resolvido a partir de um sistema de equações lineares, ao invés de programação quadrática como é realizado pelas SVMs, sem que ocorra perda na qualidade de suas soluções. Neste trabalho, foram propostas as estratégias *Adaline2* e *Adaline3* como formas de usufruir das vantagens que as LS-SVMs oferecem, acrescentando a elas a detecção automática dos vetores de suporte.

I. INTRODUÇÃO

Least Squares Support Vector Machines (LS-SVMs) são máquinas de aprendizagem desenvolvidas em [14], a partir de duas modificações na formulação das *Support Vector Machines* (SVMs): a utilização de uma função objetivo de mínimos quadrados e o uso de restrições de igualdade no problema de otimização primal. A principal característica das LS-SVMs é a sua menor complexidade computacional em relação às SVMs, sem perda na qualidade das soluções, uma vez que ambas se baseiam na Teoria de Aprendizagem

Estatística, utilizando em suas formulações o Princípio de Minimização do Risco Estrutural [19].

As LS-SVMs, entretanto, possuem uma grande desvantagem: todos os dados de treinamento são considerados vetores de suporte, diferentemente das SVMs que detectam apenas uma pequena fração deste conjunto durante a fase de treinamento.

Desde sua criação, as LS-SVMs têm sido utilizadas com sucesso em diversas aplicações de áreas como controle de processos [16], engenharia biomédica [11], economia financeira [17], além de terem apresentado resultados muito bons em bases de dados de difícil solução, como bases caóticas [15] e espirais [12].

O treinamento das LS-SVMs é realizado através da solução de um sistema de equações lineares ao invés de programação quadrática, como nas SVMs. Este sistema não pode ser resolvido diretamente pelos métodos clássicos de otimização pelo fato de não ser definido positivo. Em [1] e [9], o sistema de equações das LS-SVMs foi modificado de forma a gerar dois novos sistemas, ambos definidos positivos. A resolução destes dois novos sistemas implica na obtenção indireta da solução do sistema original.

Foi proposta em [4] a utilização do modelo neural Adaline [20] como estratégia para a solução direta do sistema de equações lineares das LS-SVMs, sem a necessidade de geração de dois novos sistemas, para sua posterior resolução. Neste trabalho, são propostas duas modificações do modelo Adaline, de modo que possamos introduzir na LS-SVM a capacidade de realizar a detecção automática de vetores de suporte em seu processo de treinamento.

O trabalho é dividido em cinco seções, sendo a primeira esta introdução. Na segunda, é apresentada uma pequena revisão bibliográfica das LS-SVMs. Na terceira, são detalhadas as estratégias propostas, bem como outras já existentes na literatura. Na próxima, são apresentados os resultados e suas discussões. Por fim, na última seção são exibidas as conclusões deste trabalho.

II. Least Squares Support Vector Machines

Dado o conjunto de treinamento $(x_i, y_i)_{i=1}^N$ com dados de entrada $x_i \in \mathbb{R}^n$ e saída binária correspondente $y_i \in \{-1, +1\}$, a superfície de decisão criada pelas LS-SVMs

é representada por

$$\omega^T \varphi(x) + b = 0 \quad (1)$$

onde ω é o vetor de pesos, b é o termo de polarização e $\varphi(\cdot)$ é o mapeamento realizado em um espaço de dimensão elevada, como indica a Fig. 1.

A LS-SVM cria dois hiperplanos, paralelos entre si no espaço de características, um para as classes positivas e outro para as negativas. Um vetor de classe +1 é considerado corretamente classificado quanto menor for sua distância em relação ao hiperplano positivo. O mesmo se aplica a um vetor de classe -1.

Desta forma, a classificação do padrão i é dada por

$$\begin{cases} \omega^T \varphi(x_i) + b = +1 & \text{se } y_i = +1 \\ \omega^T \varphi(x_i) + b = -1 & \text{se } y_i = -1. \end{cases} \quad (2)$$

Pode-se expressar a equação acima por

$$y_i[\omega^T \varphi(x_i) + b] = 1 - e_i \quad (3)$$

para o padrão de entrada i , através da utilização de variáveis de folga e_i .

A variável de folga e_i da LS-SVM pode possuir qualquer valor real, e sua interpretação é:

- $e_i < 0$: o vetor se encontra do lado oposto ao hiperplano que não é da sua classe, ou seja, sua classificação é correta.
- $e_i = 0$ (caso raro): o vetor se encontra exatamente no hiperplano paralelo correspondente a sua classe, seja +1 ou -1.
- $e_i > 0$: o vetor se encontra entre os dois hiperplanos paralelos. Quanto maior o valor de e_i , mais longe do hiperplano correto e mais próximo do hiperplano incorreto o vetor está.

O processo de treinamento consiste na obtenção de valores para os pesos ω e para o termo de polarização b de forma a minimizar uma função de custo $J(\omega, e)$.

O problema primal das LS-SVMs é

$$\min_{\omega, b, e} J(\omega, e) = \frac{1}{2} \omega^T \omega + \gamma \frac{1}{2} \sum_{i=1}^N e_i^2 \quad (4)$$

sujeito a

$$y_i[\omega^T \varphi(x_i) + b] = 1 - e_i, \quad i = 1, \dots, N$$

onde γ é um parâmetro que controla o equilíbrio entre a variável de folga e a norma do vetor de pesos.

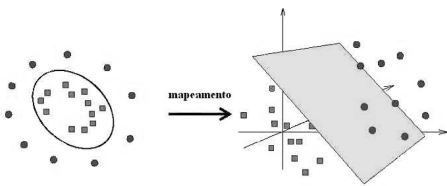


Fig. 1. Mapeamento das Funções de Kernel

Através da aplicação do Lagrangeano, é obtida uma expressão dual [10]

$$L(\omega, b, e; \alpha) = J(\omega, e) - \sum_{i=1}^N \alpha_i \{y_i[\omega^T \varphi(x_i) + b] - 1 + e_i\} \quad (5)$$

onde α_i é o multiplicador de Lagrange correspondente ao padrão de entrada i .

Pelas condições de otimalidade, o sistema linear KKT (Karush-Kuhn-Tucker) obtido é [7]

$$\begin{cases} \frac{\partial L}{\partial \omega} = 0 \rightarrow \omega = \sum_{i=1}^N \alpha_i y_i \varphi(x_i) \\ \frac{\partial L}{\partial b} = 0 \rightarrow \sum_{i=1}^N \alpha_i y_i = 0 \\ \frac{\partial L}{\partial e_i} = 0 \rightarrow e_i = \frac{\alpha_i}{\gamma}, \quad i = 1, \dots, N \\ \frac{\partial L}{\partial \alpha_i} = 0 \rightarrow y_i[\omega^T \varphi(x_i) + b] - 1 + e_i = 0. \end{cases} \quad (6)$$

O sistema linear (6) pode ser representado matricialmente como

$$\begin{bmatrix} I & 0 & 0 & -Z^T \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \gamma I & -I \\ Z & Y & I & 0 \end{bmatrix} \begin{bmatrix} \omega \\ b \\ e \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vec{1} \end{bmatrix} \quad (7)$$

onde

$$I = \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \quad \vec{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

$$Z = \begin{bmatrix} \varphi_1(x_1)y_1 & \dots & \varphi_N(x_1)y_1 \\ \vdots & \ddots & \vdots \\ \varphi_1(x_N)y_N & \dots & \varphi_N(x_N)y_N \end{bmatrix}$$

$$\omega = \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_N \end{bmatrix} \quad e = \begin{bmatrix} e_1 \\ \vdots \\ e_N \end{bmatrix} \quad \alpha = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix}$$

Substituindo a primeira e a terceira expressões de (6) na última, são obtidas as equações

$$\begin{cases} \sum_{i=1}^N \alpha_i y_i = 0 \\ \alpha \sum_{i=1}^N \sum_{j=1}^N (y_i y_j \varphi(x_i)^T \varphi(x_j) + \frac{1}{\gamma}) + y b = 1 \end{cases} \quad (8)$$

que podem ser escritas na forma matricial

$$\begin{bmatrix} 0 & -Y^T \\ Y & H \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix} \quad (9)$$

com

$$H = Z Z^T + \frac{I}{\gamma}.$$

A função $K(x, x_i) = \varphi(x)^T \varphi(x_i)$, inserida na matriz H por meio do termo $Z Z^T$ em (9), é chamada de função de Kernel, responsável pela realização de um produto no próprio espaço de entrada, ao invés de se fazer no espaço de características. Na Tabela I se encontram as principais funções utilizadas como função de Kernel.

TABELA I
FUNÇÕES DE KERNEL

Kernel:	Expressão:	Parâmetros:
Radial Basis Function	$e^{-\ x_i - x_j\ ^2 / 2\sigma^2}$	σ^2
Polinomial	$(x_i^T x_j + a)^b$	a, b
Sigmóide	$\tanh(\beta x_i^T x_j + \beta_1)$	β_0, β_1

A solução do sistema de equações lineares (9) é a mesma do problema primal (4). O primeiro elemento do vetor solução de (9) consiste no termo de polarização. Os demais elementos correspondem aos multiplicadores de Lagrange associados aos vetores de treinamento, que definem os vetores de suporte.

A saída obtida pelas LS-SVMs é da forma

$$f(x) = \text{sign}\left[\sum_{i=1}^N \alpha_i y_i K(x, x_i) + b\right] \quad (10)$$

que é a mesma gerada pelas SVMs.

Pela condição $\alpha_i = \gamma e_i$ em (6), nota-se que diferentemente da SVM, na qual grande parte dos multiplicadores de Lagrange é nula após o processo de treinamento, os valores de α_i da LS-SVM são proporcionais às variáveis de folga e_i , raramente nulas.

III. ESTRATÉGIAS PARA O TREINAMENTO DAS LS-SVMs

As SVMs, além de serem empregadas como classificadores em problemas de reconhecimento de padrões, podem também ser utilizadas na extração das informações mais relevantes em uma base de dados, por meio dos vetores de suporte detectados [3].

Por este motivo, este trabalho tem como objetivo acrescentar esta nova característica nas LS-SVMs, para ampliar a gama de problemas em que ela pode ser aplicada. Já existem métodos desenvolvidos com esta finalidade [13], [18], [5], porém as estratégias que propomos possuem algumas vantagens em relação às demais, como uma maior semelhança entre os vetores de suporte detectados, quando comparado às SVMs.

Nesta seção, serão descritas duas das principais estratégias existentes para o treinamento das LS-SVMs, que resultam em um número reduzido de vetores de suporte. Além disso, são introduzidas duas modificações no modelo Adaline que são usadas para se atingir este mesmo objetivo.

A. Estratégias existentes

1) *Pruning*: A primeira estratégia para que a quantidade de vetores de suporte obtida pelas LS-SVMs seja reduzida foi sugerida em [13]. Os autores propuseram a utilização de um processo de pruning, ou poda da rede, no qual vetores de treinamento são eliminados de acordo com o valor do multiplicador de Lagrange (α_i) associado a cada um deles.

A eliminação dos multiplicadores de Lagrange ocorre de forma recursiva, de modo que em cada iteração uma pequena quantidade de vetores de treinamento é eliminada.

O funcionamento do *Pruning* pode ser descrito pelos seguintes passos:

- 1) Treinar LS-SVM com todos os vetores.
- 2) Remover uma pequena parcela dos vetores de treinamento (por exemplo, 5%), cujos valores de $|\alpha_i|$ sejam os menores dentre os existentes.
- 3) Re-treinar LS-SVM no conjunto de treinamento reduzido.
- 4) Ir para 2, se a performance no conjunto de validação não diminuir. Caso contrário, finalizar o processo com o conjunto de treinamento da iteração anterior.

Neste método, é utilizado como conjunto de validação um subconjunto dos vetores de treinamento. O critério de parada para se determinar quando a redução deve terminar é a diminuição da acurácia da máquina já treinada com o conjunto reduzido de vetores, em relação aos dados de validação.

2) *LS² - SVM*: Em [18] foi proposto um método de duas etapas que elimina automaticamente os vetores de treinamento considerados menos relevantes. A primeira etapa consiste em se reduzir a matriz do sistema de equações lineares (9) de forma que algumas de suas colunas sejam eliminadas, mantendo-se porém todas as suas linhas.

A realização da primeira etapa segue os seguintes passos:

- 1) Operações elementares devem ser efetuadas na matriz (9) com o objetivo de que ela seja transformada em sua forma escalonada reduzida.
- 2) Os valores menores que um limiar pré-determinado devem ser transformados em zero.
- 3) Ir para 1, se a forma escalonada reduzida não tiver sido obtida. Caso contrário, finalizar o processo de escalonamento retornando a matriz obtida.

Como alguns valores foram transformados em zero, a matriz resultante possui colunas com valores totalmente nulos, que correspondem aos vetores linearmente dependentes, devendo portanto ser descartados.

As colunas linearmente dependentes, detectadas na primeira fase, são então eliminadas da matriz original. As linhas entretanto, devem ser mantidas. Desta forma, a nova matriz não é uma matriz quadrada, e a solução do novo sistema de equações lineares não pode ser resolvido por meio de métodos que exigem esta condição, como a inversa ou quaisquer métodos iterativos.

A segunda fase desta estratégia, cujos autores chamaram de *LS² - SVM*, consiste na resolução do novo sistema utilizando-se a função pseudo-inversa

$$A^+ = (A^T A)^{-1} A^T \quad (11)$$

em que

$$A^+ A = I.$$

B. Estratégias propostas

A utilização da rede Adaline (Adaptive Linear Neuron) para a resolução de sistemas lineares é uma maneira de se resolver iterativamente o sistema linear gerado pela LS-SVM, sem a necessidade de se criar dois outros sistemas definidos positivos [4]. A seguir será mostrado como o modelo Adaline é usado para a resolução da LS-SVM, bem como as modificações propostas para o modelo.

Forçando o termo de polarização a ser sempre nulo, a saída da rede Adaline [20] possui a forma

$$Y = WX. \quad (12)$$

Podemos expressar o sistema de equações lineares (9) através de

$$Ax = B. \quad (13)$$

Fazendo com que a entrada da rede Adaline seja $X = A^T$ e sua saída desejada $Y = B^T$, obtém-se a solução de (13) como sendo

$$x = W^T = \begin{bmatrix} b \\ \alpha \end{bmatrix}. \quad (14)$$

O algoritmo de aprendizagem utilizado para o treinamento da rede Adaline é o gradiente descendente [6]. Os multiplicadores de Lagrange resultantes (α) são proporcionais às variáveis de folga (e), e portanto este tipo de abordagem não proporciona uma detecção automática de vetores de suporte durante o treinamento.

1) *Adaline2*: O primeiro método que propomos tem como principal característica o fato de, a cada iteração, o número de vetores utilizados no treinamento ser reduzido. Esta redução se dá com o valor de erro de treinamento

$$erro_i = Y_{i(desejado)} - Y_{i(obtido)} \quad (15)$$

associado a cada entrada do modelo Adaline.

Vetores cujos erros de treinamento são muito pequenos podem ser eliminados, pois quanto menor este erro, mais corretamente o vetor a ele associado é classificado, portanto ele não possui grande relevância para a construção da superfície de separação das classes.

O treinamento desta versão modificada da rede Adaline ocorre seguindo os seguintes passos:

- 1) Os vetores de pesos são iniciados com valores aleatórios.
- 2) A saída da rede na iteração k (atual) é calculada por meio de $Y_k = W_k X$.
- 3) O erro atual para cada vetor de treinamento X_i é obtido $D_{ki} - Y_{ki}$.
- 4) O vetor de pesos da iteração k é atualizado com $W_k = W_{k-1} + \eta(D_k - Y_k)X$.
- 5) A soma dos erros quadráticos $E_k = \frac{1}{2} \sum_{i=1}^N (D_{ki} - Y_{ki})^2$ é calculada.
- 6) Caso o valor do erro de treinamento correspondente ao vetor X_i seja inferior a θ , a linha i e a coluna i da matriz de dados de treinamento X são eliminadas.

- 7) O peso W_i associado à linha i também é eliminado.
- 8) A saída Y_i desejada associada à coluna i também é eliminada.
- 9) Se a soma E_k for menor que δ , o treinamento termina. Caso contrário, volta ao item (2).

Após o processo de treinamento, todos os vetores que restam são considerados vetores de suporte, ou seja, apenas uma parcela do conjunto inicial corresponde aos vetores de suporte detectados. Além dos parâmetros comuns à LS-SVM, esta estratégia tem como parâmetros θ e δ , responsáveis respectivamente pela redução do conjunto de treinamento e pelo término do processo de treinamento.

2) *Adaline3*: A outra estratégia que propomos também realiza uma redução do conjunto de treinamento durante o próprio processo de treinamento. A diferença entre os métodos é que neste são eliminadas apenas as linhas da matriz com os vetores de treinamento. Como a entrada do modelo Adaline é $X = A^T$, a eliminação de uma linha em X corresponde à eliminação de uma coluna em A .

Na formulação das LS-SVMs, cada linha da matriz A corresponde a uma restrição da função de custo (4). A eliminação de uma linha implica na total eliminação daquela restrição. Já as colunas correspondem aos próprios vetores de treinamento, ou seja, sua eliminação tem como consequência impedir que estes vetores sejam suporte.

O método proposto possui os seguintes passos:

- 1) Os vetores de pesos são iniciados com valores aleatórios.
- 2) A saída da rede na iteração k (atual) é calculada por meio de $Y_k = W_k X$.
- 3) O erro atual para cada vetor de treinamento X_i é obtido $D_{ki} - Y_{ki}$.
- 4) O vetor de pesos da iteração k é atualizado com $W_k = W_{k-1} + \eta(D_k - Y_k)X$.
- 5) A soma dos erros quadráticos $E_k = \frac{1}{2} \sum_{i=1}^N (D_{ki} - Y_{ki})^2$ é calculada.
- 6) Caso o valor do erro de treinamento correspondente ao vetor X_i seja inferior a θ , a linha i da matriz de dados de treinamento X é eliminada.
- 7) O peso W_i associado à linha i também é eliminado.
- 8) Se a soma E_k for menor que δ , o treinamento termina. Caso contrário, volta ao item (2).

Ao final do processo de treinamento, todos os vetores que correspondem às linhas de X não eliminadas são considerados vetores de suporte. Além dos parâmetros comuns à LS-SVM, esta estratégia utiliza os parâmetros θ e δ , com as mesmas finalidades que possuem na estratégia anterior.

IV. RESULTADOS E DISCUSSÕES

Nesta seção, as estratégias *Adaline2* e *Adaline3* introduzidas são comparadas com os demais métodos de treinamento de LS-SVM descritos: *LS² - SVM* e *Pruning*. Além destes métodos, os resultados são compostos por uma implementação de *SVM* e outra de *LS - SVM*. A

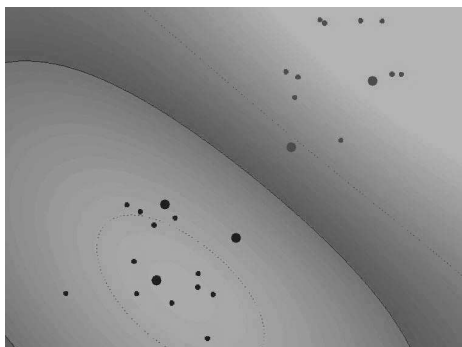


Fig. 2. Solução de *Pruning*

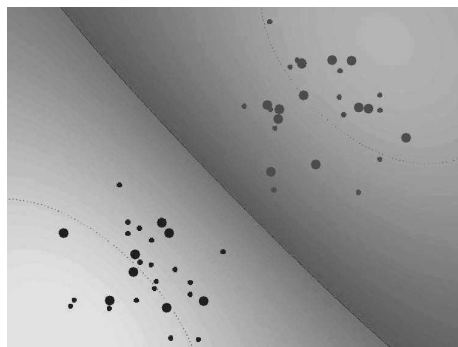


Fig. 4. Solução de *Adaline2*

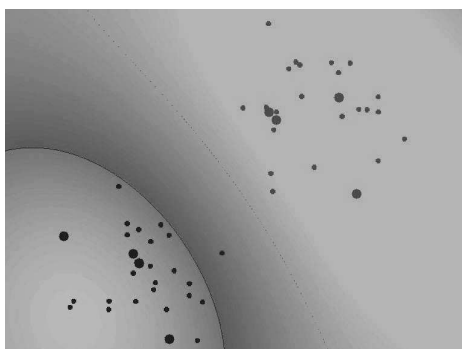


Fig. 3. Solução de LS^2-SVM

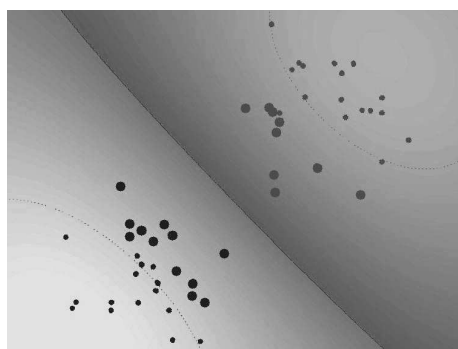


Fig. 5. Solução de *Adaline3*

SVM é resolvida por meio de um problema QP (quadratic programming) e a $LS-SVM$ pelo modelo *Adaline*.

Para a realização dos testes, utilizamos duas bases de dados. A primeira, gerada artificialmente, é constituída de duas distribuições gaussianas de dados bidimensionais, cada distribuição correspondendo a uma classe. A segunda base de dados foi obtida no repositório da Universidade da Califórnia [2]. Ela corresponde a dados adquiridos pelo Laboratório de Física Aplicada, da Universidade John Hopkins. Os dados da base representam elétrons livres na ionosfera, que é ionizada pela radiação solar ultravioleta. Cada vetor possui 34 dimensões, sendo que a classe positiva corresponde à presença de estruturas na ionosfera, e a negativa indica sua ausência.

Para a primeira base de dados, foram feitos inicialmente os ajustes dos parâmetros a serem utilizados. Para a segunda, os parâmetros que usamos foram sugeridos em [8]. Todos os métodos foram implementados em *MatlabTM* e cada um dos resultados das Tabelas II e III correspondem a valores médios de 10 experimentos.

As Figuras 2 a 5 apresentam as soluções obtidas com os quatro métodos de resolução de $LS-SVM$, que detectam automaticamente os vetores de suporte, comparados neste trabalho. Nas figuras, os vetores de suporte são representados por pontos maiores que os vetores comuns. Como *Pruning* utiliza parte dos dados de treinamento como conjunto de validação, o número de pontos na Figura 2

é menor que os demais.

Pode-se perceber que *Pruning* e *Adaline2*, apesar de obterem soluções com uma correta separabilidade dos dados, não consideram como suporte apenas os vetores na fronteira de separação ente as classes, como faz a *SVM*. *Adaline3*, por sua vez, detecta apenas os pontos na fronteira como sendo os vetores de suporte. Já LS^2-SVM obteve um resultado visivelmente inferior aos demais, tanto em relação à solução quanto na detecção dos suportes.

Analisando a Tabela II, observamos que apenas LS^2-SVM apresentou acurácia inferior a 100%. O tempo gasto no processo de treinamento foi semelhante entre os métodos, com exceção de *Pruning* e *Adaline3*, que gastaram aproximadamente o dobro de tempo em relação aos demais. As estratégias *Pruning* e LS^2-SVM foram as que mais se aproximaram do número de vetores de suporte determinados pela *SVM*. *Adaline2* e *Adaline3*, mesmo não detectando uma fração tão reduzida de vetores de suporte, foram capazes de eliminar mais de 60% de vetores do conjunto de treinamento. Já *Adaline*, como esperado, considerou todo o conjunto de treinamento como vetores de suporte.

Pelos resultados da Tabela III, notamos que os métodos baseados no modelo *Adaline*, modificado ou não, possuem um tempo de treinamento relativamente grande, para bases de dados de dimensão elevada. Mas este fato não prejudicou a acurácia de nenhum método, uma vez que todas as

TABELA II
DISTRIBUIÇÃO COM 2 AGRUPAMENTOS (KERNEL RBF)

Método	Tempo de treinamento (segundos)	Acurácia de teste (%)	Vetores de suporte (%)
<i>SVM</i>	0,78 ± 0,22	100,0 ± 0,0	4,0 ± 0,1
<i>Adaline</i>	0,56 ± 0,06	100,0 ± 0,0	100,0 ± 0,0
<i>Pruning</i>	1,37 ± 0,11	100,0 ± 0,0	9,1 ± 0,3
<i>LS² - SVM</i>	0,53 ± 0,04	94,0 ± 0,0	11,4 ± 0,3
<i>Adaline2</i>	0,75 ± 0,07	100,0 ± 0,0	36,3 ± 0,7
<i>Adaline3</i>	1,38 ± 0,10	100,0 ± 0,0	28,0 ± 0,3

TABELA III
IONOSFERA (UCI) (KERNEL RBF)

Método	Tempo de treinamento (segundos)	Acurácia de teste (%)	Vetores de suporte (%)
<i>SVM</i>	24,9 ± 4,1	94,6 ± 1,7	46,0 ± 0,1
<i>Adaline</i>	456,6 ± 26,4	95,3 ± 2,1	100,0 ± 0,0
<i>Pruning</i>	10,9 ± 7,7	93,1 ± 3,2	35,0 ± 0,9
<i>LS² - SVM</i>	13,85 ± 0,19	95,3 ± 1,7	50,0 ± 0,1
<i>Adaline2</i>	157,9 ± 5,3	93,5 ± 2,6	33,0 ± 0,2
<i>Adaline3</i>	550,5 ± 23,9	94,8 ± 1,8	33,0 ± 0,3

acurácias para esta base estiveram entre 93,1% e 95,3%. Em relação aos vetores de suporte detectados, *LS² - SVM* mais se aproximou do número detectado pela *SVM*, sendo que as outras três estratégias obtiveram resultados similares entre si.

A partir dos vetores de suporte detectados por cada uma das estratégias, foi feita a comparação de quantos vetores de treinamento foram considerados suporte tanto pela *SVM* como por cada método. Assim, foi obtida a informação de qual dos métodos se aproximou mais do resultado da *SVM*, isto é, qual detectou como suporte os mesmos vetores que a *SVM* detectou. O melhor deles foi *Adaline3*, que dentre os vetores detectados, 56% coincidem com os da *SVM*. Os métodos que o seguem são *Pruning*, com 51%, *LS² - SVM* com 46% e *Adaline2* com 43%.

V. CONCLUSÕES

As LS-SVMs foram criadas para reduzir a complexidade computacional das SVMs, substituindo a resolução via programação quadrática por um sistema de equações lineares. As soluções encontradas pela *SVM* e *LS-SVM* possuem qualidade equivalente, uma vez que ambas são baseadas no Princípio de Minimização do Risco Estrutural.

Neste trabalho, foram propostas duas novas estratégias para o treinamento das LS-SVMs, de modo a torná-las capazes de realizar a detecção automática de vetores de suporte durante seu processo de treinamento, característica que a *LS-SVM* originalmente proposta não apresenta. Ambas as estratégias utilizam modificações no modelo neural *Adaline*.

As estratégias introduzidas foram comparadas com outras já existentes, sendo que *Pruning* e *LS² - SVM* não se mostraram capazes de detectar os mesmos vetores de

suporte que a *SVM*, apesar de terem apresentado acurácias elevadas.

Dentre os métodos propostos, *Adaline2* não foi capaz de detectar corretamente os vetores de suporte, apesar de apresentar acurácias semelhantes aos demais métodos. Já *Adaline3* apresentou os melhores resultados conjuntos de acurácia e semelhança de vetores de suporte em relação à *SVM*, dentre todos os métodos comparados.

REFERENCIAS BIBLIOGRÁFICAS

- [1] J. A. K. Suykens B. Hamers and B. De Moor. A comparison of iterative methods for least squares support vector machine classifiers. 2001.
- [2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
- [3] B. P. R. Carvalho. O estado da arte em métodos para reconhecimento de padrões: Support vector machine. In *Congresso Nacional de Tecnologia da Informação e Comunicação (Sucesu 2005)*, Belo Horizonte, MG, 2005.
- [4] B. P. R. Carvalho and A. P. Braga. Estratégias neurais para treinamento de *Least Squares Support Vector Machines*. In *Simpósio Brasileiro de Redes Neurais (SBRN 2004)*, 2004.
- [5] B. P. R. Carvalho and A. P. Braga. Novas estratégias para treinamento de *Least Squares Support Vector Machines*. In *Encontro Nacional de Inteligência Artificial (ENIA 2005)*, 2005.
- [6] R. J. Williams D. E. Rumelhart, G. E. Hinton. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1986.
- [7] R. Fletcher. *Practical Methods of Optimization*. 1987.
- [8] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. Technical report, ESAT-SISTA, K.U. Leuven, 2000.
- [9] S.S. Keerthi and S.K. Shevade. Smo algorithm for least-squares svm formulations. 2003.
- [10] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. 1973.
- [11] L. Lukas, A. Devos, J. A. K. Suykens, L. Vanhamme, S. Van Huffel, A. Tate, C. Majos, and C. Arus. The use of ls-svm in the classification of brain tumors based on magnetic resonance spectroscopy signals. In *European Symposium Artificial Neural Networks (ESANN'2002)*, 2002.
- [12] L. Lukas, J.A.K. Suykens, and J. Vandewalle. Least squares support vector machines classifiers: a multi two-spiral benchmark problem. In *Indonesian Student Scientific Meeting (ISSM 2001)*, pages 289–292, Manchester, United Kingdom, 2001.
- [13] J. A. K. Suykens, L. Lukas, and J. Vandewalle. Sparse least squares support vector machine classifiers. In *European Symposium on Artificial Neural Networks (ESANN 2000)*, 2000.
- [14] J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
- [15] J.A.K. Suykens and J. Vandewalle. Chaos control using least squares support vector machines. *International Journal of Circuit Theory and Applications, Special Issue on Communications, Information Processing and Control using Chaos*, 27(6):605–615, 1999.
- [16] J.A.K. Suykens, J. Vandewalle, and B. De Moor. Optimal control by least squares support vector machines. *Neural Networks*, 14(1):23–35, 2001.
- [17] Van Gestel T., Suykens J., Baestaens D., Lambrechts A., Lanckriet G., Vandaele B., De Moor B., and Vandewalle J. Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks in Financial Engineering*, 12(4):809–821, 2001.
- [18] J. Valyon and G. Horváth. A sparse least squares support vector machine classifier. In *International Joint Conference on Neural Networks (IJCNN'2004)*, 2004.
- [19] Vladimir Vapnik. *Statistical Learning Theory*. 1998.
- [20] B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE WESCON Convention Record*, 1960.