

# Novas Estratégias para Treinamento de *Least Squares Support Vector Machines*

Bernardo Penna Resende de Carvalho<sup>1</sup>, Antônio de Pádua Braga<sup>2</sup>

<sup>1</sup>Universidade Federal de Minas Gerais – Lab. Inteligência Computacional  
Av. Antônio Carlos, 6.627 - Campus UFMG Pampulha  
30.161-970, Belo Horizonte, MG, Brasil  
Tel:+55 (31) 3499-5466, Fax:+55 (31) 3499-4850

<sup>2</sup>Universidade Federal de Minas Gerais – Depto. Engenharia Eletrônica  
Av. Antônio Carlos, 6.627 - Campus UFMG Pampulha  
30.161-970, Belo Horizonte, MG, Brasil  
Tel:+55 (31) 3499-4869, Fax:+55 (31) 3499-4850

bpenna@gmail.com, apbraga@cpdee.ufmg.br

**Abstract.** *We present in this work two new training strategies for the LS-SVMs, in order to eliminate their greatest drawback when comparing to SVMs, the inexistence of the support vectors' automatic detection. Least Squares Support Vector Machines (LS-SVMs) were created in 1999, corresponding to a modified version of Support Vector Machines (SVMs), developed in 1992. The main characteristic of the LS-SVMs is the low computational complexity comparing to the SVMs, without quality loss in the solution, because the principles that both have been based are the same. In this work, we considered the strategies Ada – Inv and Ada – Pinv as a manner to enjoy the benefits that LS-SVMs offer, in a way that all the commonly extracted information by the SVM are also detected by the LS-SVM, through the automatic detection of support vectors.*

**Resumo.** *Propomos neste trabalho duas novas estratégias para o treinamento das LS-SVMs, de forma a eliminar sua maior desvantagem 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 das Support Vector Machines (SVMs), desenvolvidas em 1992. A principal característica das LS-SVMs é a sua menor complexidade computacional em relação às SVMs, sem que ocorra perda na qualidade de suas soluções, uma vez que os princípios em que ambas se baseiam são os mesmos. Neste trabalho, propusemos as estratégias Ada – Inv e Ada – Pinv como formas de usufruir das vantagens que as LS-SVMs oferecem, de modo que todas as informações normalmente extraídas pela SVM também possam ser obtidas pela LS-SVM, através da detecção automática dos vetores de suporte.*

## 1. Introdução

*Support Vector Machines (SVMs)* são máquinas de aprendizagem desenvolvidas em 1992 [Boser et al., 1992], cuja fase de aprendizagem é realizada por meio de um treinamento supervisionado. Elas podem ser consideradas máquinas com apenas uma camada escondida, que se baseiam na Teoria de Aprendizagem Estatística, utilizando em sua formulação o Princípio de Minimização do Risco Estrutural [Vapnik, 1995]. Seu treinamento é realizado através da resolução de um QP (*quadratic programming*), que possui

um custo computacional elevado. A principal característica das SVMs é a determinação automática dos dados de treinamento mais relevantes para o problema abordado, chamados vetores de suporte.

*Least Squares Support Vector Machines* (LS-SVMs) foram criadas em 1999 [Suykens and Vandewalle, 1999], a partir de duas modificações na formulação das SVMs: a utilização de uma função objetivo de mínimos quadrados e o uso de restrições de igualdade no problema 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 os princípios em que ambas se baseiam são os mesmos. Porém, a desvantagem das LS-SVMs reside no fato de que 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.

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 [Hamers et al., 2001] e [Keerthi and Shevade, 2003], 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. Desenvolvemos em [Carvalho and Braga, 2004] algumas estratégias de soluções diretas do sistema de equações lineares das LS-SVMs, sem a necessidade de geração de dois novos sistemas, para sua posterior resolução.

Neste trabalho, introduzimos duas novas estratégias para que a LS-SVM seja capaz de realizar a detecção automática de vetores de suporte em seu processo de treinamento. Ambas as propostas correspondem a processos de aprendizagem de duas fases, sendo a primeira realizada por meio das estratégias introduzidas em [Carvalho and Braga, 2004]. A segunda fase tem como objetivo selecionar, dentre todo o conjunto de treinamento, apenas aqueles que influenciam na construção da superfície de separação, ou seja, os vetores de suporte. Desta forma, introduzimos a detecção automática dos vetores de suporte nas LS-SVMs, usufruindo ao mesmo tempo das vantagens que elas oferecem, como a facilidade de implementação e o menor custo computacional.

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.

## **2. *Least Squares Support Vector Machines***

A aprendizagem das LS-SVMs, da mesma forma que nas SVMs, consiste na minimização dos seguintes erros [Vapnik, 1998]:

Erro empírico: erro dos dados de treinamento, responsável pelos ajustes da superfície de separação aos dados apresentados durante o processo de aprendizagem, comum na maioria das máquinas de aprendizagem.

Erro estrutural: erro de generalização, responsável pela alta capacidade de generalização das SVMs e LS-SVMs, pois possibilita o desenvolvimento de um limite inferior de generalização.

A obtenção de um equilíbrio entre os dois erros acima descritos significa a possibilidade de superar tendências de excesso de ajustes (overfitting) obtendo, ao mesmo tempo, a capacidade de uma boa generalização.

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  $\omega$  é o vetor de pesos,  $b$  é o termo de polarização e  $\varphi(\cdot)$  é o mapeamento realizado em um espaço de dimensão elevada.

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 maneira, 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  $e_i$ , responsáveis por considerar a distância deste padrão em relação ao hiperplano associado a sua classe.

O processo de treinamento consiste na obtenção de valores para os pesos 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

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

Através da aplicação do Lagrangeano [Luenberger, 1973], é obtida uma expressão dual

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

onde  $\alpha_i$  é o multiplicador de Lagrange correspondente ao padrão de entrada  $i$ .

Pelas condições de otimalidade, o sistema de equações lineares KKT (Karush-Kuhn-Tucker) [Fletcher, 1987] obtido é

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

onde  $\gamma$  é um parâmetro de treinamento fornecido pelo usuário, que controla o equilíbrio entre a variável de folga e a norma do vetor de pesos.

O sistema de equações lineares (6) pode ser representado pela forma matricial

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

onde

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

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

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

Substituindo a primeira e a terceira expressões de (6) na última, obtêm-se

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

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

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

A função  $K(x, x_i) = \varphi(x)^T \varphi(x_i)$ , inserida na matriz  $H$  por meio de  $ZZ^T$  em (9), é chamada de função de Kernel, responsável pela realização de um produto no espaço de entrada, ao invés de se fazer no espaço de características. Assim LS-SVM faz um mapeamento implícito dos dados de entrada, como a SVM [Suykens and Vandewalle, 1999]. Na Tabela 1 se encontram as principais funções usadas como Kernel.

A solução do sistema de equações lineares (9) é a mesma do problema primal (4) [Luenberger, 1973]. 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.

Pela condição  $\alpha_i = \gamma e_i$  em (6), nota-se a perda da escassez do vetor de multiplicadores de Lagrange nas LS-SVMs, pois os valores de  $\alpha_i$  são proporcionais às variáveis de folga, raramente nulas.

Pode-se verificar que 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.

**Tabela 1: Funções de Kernel**

KERNEL:	Expressão:	Parâmetros:
RBF	$e^{-\ x_i - x_j\ ^2 / 2\sigma^2}$	$\sigma^2$
Polinomial	$(x_i^T x_j + a)^b$	$a, b$
Sigmóide	$\tanh(\beta_0 x_i^T x_j + \beta_1)$	$\beta_0, \beta_1$

O fato de todos os dados do conjunto de treinamento serem considerados vetores de suporte implica na redução da utilidade 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 [Carvalho, 2005].

Por este motivo, o objetivo deste trabalho é o de 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, porém as estratégias que abordamos possuem algumas vantagens em relação às demais, como uma maior semelhança no número de vetores de suporte detectados comparado às SVMs e também uma alta capacidade de generalização, observada pelos resultados apresentados na seção 4.

### 3. Estratégias para o treinamento das LS-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, introduzimos duas novas maneiras de se atingir este objetivo, ambas baseadas na solução direta das LS-SVMs proposta em [Carvalho and Braga, 2004].

#### 3.1. Estratégias existentes

##### 3.1.1. *Pruning*

A primeira estratégia para que a quantidade de vetores de suporte obtida pelas LS-SVMs seja reduzida foi sugerida em [Suykens et al., 2000]. 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 ( $\alpha_i$ ) 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. 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.

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

1. Treinar LS-SVM normalmente, com todos os vetores de treinamento.
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.

##### 3.1.2. $LS^2 - SVM$

Em [Valyon and Horváth, 2004] foi proposto um método de duas etapas que elimina automaticamente os vetores de treinamento considerados menos relevantes. A primeira etapa consiste em reduzir a matriz do sistema de equações lineares (9), representada em (11), de forma que algumas de suas colunas sejam eliminadas, mantendo-se porém todas as suas linhas.

$$\begin{bmatrix} 0 & -Y^T \\ Y & H \end{bmatrix} \quad (11)$$

A realização da primeira etapa segue os seguintes passos:

1. Operações elementares devem ser efetuadas na matriz (11) 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, apresentada em (11). 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  $LS^2 - SVM$ , consiste na resolução do novo sistema utilizando-se a função pseudo-inversa

$$A^+ = (A^T A)^{-1} A^T \quad (12)$$

em que

$$A^+ A = I.$$

## 3.2. Estratégias propostas

### 3.2.1. *Ada - Inv*

A primeira estratégia que propomos neste trabalho consiste na realização do treinamento das LS-SVMs através de duas fases. A primeira ocorre por meio da resolução direta do sistema de equações lineares (9) utilizando-se uma rede neural Adaline (Adaptive Linear Neuron), como proposto em [Carvalho and Braga, 2004].

A saída da rede Adaline [Widrow and Hoff, 1960] possui a forma

$$Y = WX + b. \quad (13)$$

Forçando o termo de polarização em (13) a ser sempre nulo ( $b = 0$ ), tem-se

$$Y = WX. \quad (14)$$

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

$$Ax = B. \quad (15)$$

Fazendo com que a entrada da rede Adaline seja  $X = A^T$  e sua saída desejada  $Y = B^T$ , obtém-se a solução de (15) como sendo

$$x = W^T = \begin{bmatrix} b \\ \alpha \end{bmatrix}. \quad (16)$$

O algoritmo de aprendizagem utilizado para o treinamento da rede Adaline é o gradiente descendente [Rumelhart et al., 1986]. Na segunda fase desta estratégia, utilizamos o critério proposto em [Suykens et al., 2000], no qual os vetores que possuem os menores valores de  $|\alpha_i|$  são eliminados do conjunto de treinamento. A quantidade de vetores a serem eliminados corresponde a um valor pré-especificado pelo usuário.

A justificativa para o uso deste critério é obtida pela análise de (17), que indica a proporcionalidade entre a variável de folga do vetor  $i$  e seu valor de  $\alpha_i$  correspondente. Quanto menor a variável de folga, mais corretamente seu vetor é classificado, portanto ele não possui grande relevância para a construção da superfície de separação das classes.

$$\alpha_i = \gamma e_i \quad (17)$$

Para a obtenção dos valores finais dos multiplicadores de Lagrange do conjunto de treinamento reduzido, utiliza-se a função inversa

$$x = A^{-1}B. \quad (18)$$

### 3.2.2. *Ada – Pinv*

A outra estratégia que propomos neste trabalho possui a primeira fase idêntica à anterior, uma vez que a utilização da rede Adaline, além de ser de fácil implementação, é capaz de resolver (9) de forma direta.

A diferença desta abordagem para *Ada – Inv* se encontra na segunda fase. O critério de relevância dos vetores de treinamento também é o mesmo da estratégia anterior: o valor do módulo do multiplicador de Lagrange, como proposto em [Suykens et al., 2000]. Porém, apenas as colunas da matriz  $A$  em (15) são eliminadas. As linhas são mantidas, da mesma forma que em [Valyon and Horváth, 2004].

A justificativa para a não eliminação das linhas se deve ao fato de que na formulação das LS-SVMs, cada linha da matriz  $A$  corresponde a uma restrição da função de custo primal (4). A eliminação de uma linha implica na total eliminação daquela restrição e portanto na perda de informações para o processo de aprendizagem. Já as colunas correspondem aos próprios vetores de treinamento, ou seja, sua eliminação tem como consequência impedir que estes vetores sejam suporte, o que de fato é o nosso objetivo.

A grande vantagem deste método, bem como de *Ada – Inv*, em relação à  $LS^2 - SVM$  é que não é necessário fazer o escalonamento para se obter a forma reduzida, utilizando-se um parâmetro de difícil sintonia, a tolerância usada para zerar os menores valores no processo de escalonamento. Basta-se treinar uma LS-SVM com o conjunto de treinamento original por meio da rede Adaline, para obter um sistema de equações lineares com a matriz  $A$  não quadrada.

Pode-se então usar a função pseudo-inversa, representada em (12), para se resolver o sistema reduzido e obter uma solução das LS-SVMs com uma quantidade pré-estabelecida de vetores de suporte.

## 4. Resultados

Nesta seção, as estratégias *Ada – Inv* e *Ada – Pinv* introduzidas na seção anterior são comparadas com os demais métodos de treinamento de LS-SVMs descritos,  $LS^2 - SVM$  e *Pruning*. Além desses métodos, os resultados são compostos por uma implementação

**Tabela 2: Dados de distribuições gaussianas (kernel RBF)**

Método	Tempo de treinamento(s)	Acurácia (%)	Quantidade de vetores de suporte (%)
<i>SVM(QP)</i>	0,782 ± 0,220	100,0 ± 0,0	4,0 ± 1,0
<i>LS – SVM(Adaline)</i>	0,568 ± 0,065	100,0 ± 0,0	100,0 ± 0,0
<i>Ada – Inv</i>	0,591 ± 0,073	81,3 ± 0,8	6,0 ± 0,8
<i>Ada – Pinv</i>	0,571 ± 0,078	100,0 ± 0,0	6,0 ± 1,0
<i>Pruning</i>	0,468 ± 0,068	100,0 ± 0,0	53,0 ± 8,2
<i>LS<sup>2</sup> – SVM</i>	0,535 ± 0,041	70,0 ± 5,0	11,0 ± 3,1

**Tabela 3: Bupa liver disorder (kernel linear)**

Método	Tempo de treinamento(s)	Acurácia (%)	Quantidade de vetores de suporte (%)
<i>SVM(QP)</i>	24,34 ± 1,80	67,3 ± 2,3	72,0 ± 3,2
<i>LS – SVM(Adaline)</i>	451,56 ± 8,11	68,7 ± 2,5	100,0 ± 0,0
<i>Ada – Inv</i>	460,72 ± 9,37	47,8 ± 1,4	75,0 ± 6,1
<i>Ada – Pinv</i>	447,36 ± 13,06	66,9 ± 2,6	75,0 ± 4,2
<i>Pruning</i>	5,58 ± 0,30	67,4 ± 3,4	84,0 ± 5,0
<i>LS<sup>2</sup> – SVM</i>	12,32 ± 1,2	67,1 ± 3,3	76,2 ± 1,1

de *SVM* e outra de *LS – SVM*. A *SVM* é resolvida por meio de um problema QP e a *LS – SVM* pela rede 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 [Blake and Merz, 1998]. Ela corresponde a dados adquiridos pela BUPA Medical Research, com dados sobre doenças que atacam o fígado: bupa liver disorder. Suas seis dimensões contêm exames de sangue que indicam a tendência de doenças do fígado, que podem ser apresentadas devido ao consumo excessivo de álcool.

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 [Gestel et al., 2000]. Experimentos foram feitos com kernel linear e RBF (Radial basis function). Todos os métodos foram implementados em *Matlab<sup>TM</sup>* e utilizamos 10-fold cross validation para a obtenção dos resultados dispostos nas Tabelas 2 a 4.

Pela análise da Tabela 2, todos os métodos comparados se caracterizam por um

**Tabela 4: Bupa liver disorder (kernel RBF)**

Método	Tempo de treinamento(s)	Acurácia (%)	Quantidade de vetores de suporte (%)
<i>SVM(QP)</i>	49,19 ± 4,51	58,9 ± 7,6	81,3 ± 1,7
<i>LS – SVM(Adaline)</i>	451,82 ± 21,85	55,1 ± 2,4	100,0 ± 0,00
<i>Ada – Inv</i>	458,30 ± 22,21	49,1 ± 3,2	85,4 ± 7,4
<i>Ada – Pinv</i>	452,64 ± 21,94	66,6 ± 4,1	85,1 ± 0,8
<i>Pruning</i>	9,22 ± 0,54	64,3 ± 4,0	94,2 ± 3,2
<i>LS<sup>2</sup> – SVM</i>	12,80 ± 2,2	57,3 ± 3,8	86,0 ± 1,7

tempo de treinamento semelhante, destacando-se *Pruning* e *SVM*, o primeiro com o menor tempo e o segundo com o mais elevado. Apenas dois métodos obtiveram acurácias inferiores a 100%: *Ada – Inv* e *LS<sup>2</sup> – SVM*. Em relação à quantidade de vetores de suporte, as duas estratégias que propusemos neste trabalho se aproximaram do valor obtido pela *SVM*. A obtenção de 100% de acurácia e de um número de vetores de suporte muito próximo aos da *SVM*, indica que a estratégia *Ada – Pinv* apresentou a melhor performance nesta base de dados, utilizando o kernel RBF.

Pelas Tabelas 3 e 4, podemos observar uma característica dos métodos baseados na rede Adaline: o tempo gasto para seu treinamento excede de 10 a 20 vezes os demais para dados de dimensões mais elevadas, como é o caso do "bupa liver disorder".

Observando a Tabela 3, constatamos que com a utilização do kernel linear, a melhor solução foi obtida pela *LS – SVM*, com acurácia de 68.7%, sendo que apenas *Ada – Inv* apresentou um resultado abaixo de 65%, indicando resultados bem próximos uns dos outros. Além da *LS – SVM*, que não possui a capacidade de detectar vetores de suporte, apenas *Pruning* não selecionou um número próximo destes vetores em relação à *SVM*, excedendo em mais de 10% este valor.

Utilizando o kernel RBF, como visto na Tabela 4, *Ada – Pinv* e *Pruning* apresentaram acurácia superior a 64%, tendo as demais estratégias acurácia inferior a 60%. Da mesma forma que na Tabela 3, o *Pruning* selecionou um número de vetores de suporte muito acima dos obtidos pela *SVM*, demonstrando que apesar de obter acurácias elevadas, possui é capaz de selecionar um número adequado de vetores de suporte. Já *LS<sup>2</sup> – SVM*, nem sempre obteve resultados acurados, como observa-se na Tabela 2.

A estratégia *Ada – Inv*, pelo fato de não levar em consideração todas as restrições do problema primal, nas três tabelas apresentou baixa acurácia, mesmo tendo selecionado uma quantidade de vetores de suporte semelhante aos obtidos pela *SVM*. *Ada – Pinv*, por sua vez, conseguiu equilibrar acurácia e quantidade de vetores de suporte. Porém, em relação ao tempo gasto no processo de treinamento, apresentou bons resultados para a base de dados bidimensional apenas, ou seja, se comportou bem na presença de poucas dimensões. Entretanto, para bases de dados com dimensões extremamente elevadas, ela pode se beneficiar do fato de ser iterativa, de forma a não ser necessário armazenar todos os dados de uma só vez na memória, como seria o caso de *Pruning* e *SVM*.

## 5. Conclusões

Com a intenção de evitar a necessidade de se resolver problemas QPs, foram criadas as LS-SVMs, cuja solução pode ser obtida pela resolução de 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, propusemos 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, como uma primeira fase do processo de treinamento, abordagens desenvolvidas para a resolução direta do sistema de equações lineares das LS-SVMs, sem a necessidade de geração de dois novos sistemas, para sua posterior resolução.

As estratégias introduzidas foram comparadas com outras já existentes, sendo que *Pruning* não se mostrou capaz de detectar um número semelhante de vetores de suporte comparado à SVM, apesar de ter apresentado acurácias elevadas. Já *LS<sup>2</sup> – SVM* obteve acurácia e quantidade de vetores de suporte insatisfatórios na primeira base de dados.

Dentre os métodos propostos, *Ada - Inv* foi capaz de detectar corretamente os vetores de suporte, porém com uma grande perda na qualidade dos resultados. Já *Ada - Pinv* esteve entre os melhores métodos em todos os testes realizados, tanto em termos de acurácia quanto de quantidade de vetores de suporte.

Um fato a se ressaltar porém, é que ambas as estratégias que propusemos neste trabalho apresentaram um tempo de treinamento elevado, em relação aos outros métodos, quando utilizadas em bases de dados com dimensões mais elevadas. Em bases bidimensionais entretanto, elas se mostraram mais rápidas que a *SVM*.

## Referências

- Blake, C. L. and Merz, C. J. (1998). UCI repository of machine learning databases.
- Boser, B. E., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152.
- Carvalho, B. P. R. (2005). 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.
- Carvalho, B. P. R. and Braga, A. P. (2004). Estratégias neurais para treinamento de *Least Squares Support Vector Machines*. In *VIII Simpósio Brasileiro de Redes Neurais (SBRN 2004)*, São Luis, MA.
- Fletcher, R. (1987). *Practical Methods of Optimization*. Wiley.
- Gestel, T. V., Suykens, J., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B. D., and Vandewalle, J. (2000). Benchmarking least squares support vector machine classifiers. Technical report, ESAT-SISTA, K.U. Leuven.
- Hamers, B., Suykens, J. A. K., and Moor, B. D. (2001). A comparison of iterative methods for least squares support vector machine classifiers. Technical report, ESAT-SISTA, K.U. Leuven.
- Keerthi, S. S. and Shevade, S. K. (2003). Smo algorithm for least-squares svm formulations. *Neural Computations*, 15(2):487–507.
- Luenberger, D. G. (1973). *Introduction to Linear and Nonlinear Programming*. Addison-Wesley.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1(1):318–362.
- Suykens, J. A. K., Lukas, L., and Vandewalle, J. (2000). Sparse least squares support vector machine classifiers. In *ESANN'2000 European Symposium on Artificial Neural Networks*, pages 37–42.
- Suykens, J. A. K. and Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300.
- Valyon, J. and Horváth, G. (2004). A sparse least squares support vector machine classifier. In *International Joint Conference on Neural Networks (IJCNN'2004)*.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons;
- Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, 4:96–104.