# Support Vectors Selection by Linear Programming

Vojislav Kecman, Ivana Hadzic

Department of Mechanical Engineering, University of Auckland,
Private Bag 92019, Auckland, New Zealand
E-mail: v.kecman@auckland.ac.nz

**Abstract**

*A linear programming (LP) based method is proposed for learning from experimental data in solving the nonlinear regression and classification problems. LP controls both the number of basis functions in a neural network (i.e., support vector machine) and the accuracy of learning machine. Two different methods are suggested in regression and their equivalence is discussed. Examples of function approximation and classification (pattern recognition) illustrate the efficiency of the proposed method.*

***Keywords***: *support vector machines, neural networks, linear programming, learning from experimental data, nonlinear multivariate functions approximations, nonlinear multivariate pattern recognition*

## 1. Introduction

*Support Vector Machine* (SVM) is a new type of learning machine that keeps the training error fixed (i.e., within given boundaries), and minimizes the confidence interval, i.e., it matches the machine capacity to data complexity. SVM that uses *quadratic programming* (QP) in calculating support vectors has very sound theoretical basis and it works almost perfectly for not too large data sets. When the number of points is large (say more then 2000), the QP problem becomes extremely difficult to solve with standard methods and program solvers. Recently, a lot of work was done on implementing an LP approach in support vectors selection. Kecman (1999) suggested optimal subset selection by using LP in solving regression tasks by minimizing the $L_1$ norm of the output layer weight vector **w**. Hadzic and Kecman (1999) implemented such an LP approach to classification problems. Zhang and Fuchs (1999) proposed an LP based method for selecting the hidden neurons at the initialization stage of the multilayer perceptron network. The LP approaches in solving regression problems have been implemented since 1950ties (see Charnes, Cooper and Ferguson 1955, Cheney and Goldstein 1958a, Stiefel 1960, Kelley 1957, Rice 1964). These results follow from minimizing $L_1$ norm of an error. (Interestingly, the first results on the $L_1$ norm estimators were given as early as 1757 by Yugoslav scientist Boskovic, see Eisenhart, 1962). The summary and very good presentations of mathematical programming application in statistics were given by Arthanari and Dodge (1993). An early work on LP based classification algorithms dates back to the middle of 1960s (see Mangasarian 1965). Recently, a lot of work has been done on implementing LP approach in support vectors selection (Smola et all 1998, Bennett 1999, Weston et all 1999, and Graepel et all 1999). All these papers originate from the same stream of ideas clustered around controlling the (maximal) margin. Hence, they are close to the SVM constructive algorithms.

We demonstrate the LP based approach by using the regression examples and we expand the same method for solving classification tasks. Here, the slight difference in respect to the standard SVM learning is that instead minimizing $L_2$ norm $\|\mathbf{w}\|_2$ of the weight vector **w**, we will minimize $L_1$ norm $\|\mathbf{w}\|_1$ in an LP approach. In Section 2, we develop two methods for solving regression problems by LP and discuss their equivalence. In Section 3, an LP approach is expanded for solving classification tasks. Section 4 is devoted to simulation examples, and some conclusions are drawn in Section 5.

## 2. Linear Programming in Regression

Minimization of the $L_2$ norm equals minimizing $\mathbf{w}^T\mathbf{w} = \sum_{i=1}^{n} w_i^2 = w_1^2 + w_2^2 + \cdots + w_n^2$ and this results in the QP type of problem that leads to a maximization of a margin $M$ (Vapnik, 1995). Here, instead minimizing an $L_2$ norm of the weight vector **w** we will minimize its $L_1$ norm. The geometrical meaning of the $L_1$ norm is not clear yet but the appli-

cation of the LP approach for a subset (support vectors, basis functions) selection results in a very good performance of NN and/or SVM. At the same time there is no theoretical evidence either that, the minimization of the $L_1$ norm or $L_2$ norm of the weight vector **w** produces superior generalization.

## 2.1 Sparseness regularization method

Regression tasks could be formulated in many different ways. Here we show two approaches. First method is a one-objective function minimization problem and the second one is a multi-objective function minimization task. Our problem is to design a parsimonious NN containing less neurons than data points. Such a sparseness of an NN or SVM results from minimizing $L_1$ norm of the weight vector **w**. At the same time, we want to solve **y = Gw** such that $\| \mathbf{Gw} - \mathbf{y} \|$ is small for some chosen norm. In order to perform such a task we formulate the regression problem as follows - find a weight vector

$$\mathbf{w} = \arg\min\|\mathbf{w}\|_1, \qquad \text{subject to,} \qquad \| \mathbf{Gw} - \mathbf{y} \|_\infty \le \varepsilon, \tag{1}$$

where $\varepsilon$ defines the *maximally* allowed error (that is why we used $L_\infty$ norm) and corresponds to the $\varepsilon$-insensitivity zone in an SVM learning. This constrained optimization problem can easily be transformed into a standard linear programming form. First, recall that $\min\|\mathbf{w}\|_1 = \min \sum_{p=1}^{P} | w_p |$, and this is not an LP problem formulation where we typically minimize $\mathbf{c}^T\mathbf{w} = \sum_{p=1}^{P} c_p w_p$ and **c** is some known coefficient vector. (*P* denotes the number of training data). Thus, in order to apply the LP algorithm we use the standard trick by replacing $w_p$ and $| w_p |$ as follows

$$w_p = w_p^+ - w_p^- \tag{2a}$$

$$|w_p| = w_p^+ + w_p^- \tag{2b}$$

where $w_p^+$ and $w_p^-$ are the two non-negative variables, i.e., $w_p^+ \ge 0$, $w_p^- \ge 0$. Note, that the substitutions done in (2) are unique, i.e., for a given $w_p$ there is only one pair ($w_p^+$, $w_p^-$) which fulfills both equations. Furthermore, both variables can not be bigger than zero at the same time. In fact there are only three possible solutions for a pair of variables ($w_p^+$, $w_p^-$), namely, (0, 0), ($w_p^+$, 0) or (0, $w_p^-$). Second, the constraint in (1) is not in a standard formulation either and it should also be reformulated as follows. Note that $\| \mathbf{Gw} - \mathbf{y} \|_\infty \le \varepsilon$ in (1) defines an $\varepsilon$-tube inside which should our approximating function reside. Such a constraint can be rewritten as

$$\mathbf{y} - \varepsilon\,\mathbf{1} \le \mathbf{Gw} \le \mathbf{y} + \varepsilon\,\mathbf{1} \tag{3}$$

where **1** is a (*P*, 1) column vector filled with ones. Equation (3) represents a standard set of linear constraints and our LP problem to solve is now the following. Find a pair

$$(\mathbf{w}^+, \mathbf{w}^-) = \arg\min \sum_{p=1}^{P} (w_p^+ + w_p^-), \text{ subject to, } \mathbf{y} - \varepsilon\,\mathbf{1} \le \mathbf{G}(\mathbf{w}^+ - \mathbf{w}^-) \le \mathbf{y} + \varepsilon\,\mathbf{1}, \text{ (4a)}, \ \mathbf{w}^+ \ge \mathbf{0}, \text{ (4b)}, \ \mathbf{w}^- \ge \mathbf{0}, \text{ (4c)}$$

where $\mathbf{w}^+ = [\, w_1^+, w_2^+, \cdots, w_P^+ \,]^T$ and $\mathbf{w}^- = [\, w_1^- \; w_2^- \; \ldots \; w_P^- \,]^T$. LP problem (4) can be presented in a matrix-vector formulation suitable for an LP solver as follows: $\min_{\mathbf{w}} \mathbf{c}^T\mathbf{w}$, i.e.,

$$\min_{\mathbf{w}} \left[ \underbrace{1 \; 1 \ldots 1}_{P \text{ columns}} \quad \underbrace{1 \; 1 \ldots 1}_{P \text{ columns}} \right] \left[ w_1^+ \, w_2^+ \, \cdots w_P^+ \, w_1^- \, w_2^- \cdots w_P^- \right]^T \quad \text{subject to,} \quad \begin{bmatrix} \mathbf{G} & -\mathbf{G} \\ -\mathbf{G} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} \le \begin{bmatrix} \mathbf{y} + \varepsilon\mathbf{1} \\ -\mathbf{y} + \varepsilon\mathbf{1} \end{bmatrix}, \tag{5}$$

$$\mathbf{w}^+ \ge 0, \ \mathbf{w}^- \ge 0, \tag{6), (7}$$

where both **w** and **c** are the (2*P*, 1)-dimensional vectors. **c** = **1**(2*P*, 1), and $\mathbf{w} = [\mathbf{w}^{+T} \; \mathbf{w}^{-T}]^T$.

## 2.2 Expanded sparseness regularization method

Here, the original problem is not changed in the sense that we want to solve $\mathbf{y} = \mathbf{G}\mathbf{w}$ such that $\|\mathbf{G}\mathbf{w} - \mathbf{y}\|$ is small for some chosen norm. However, the formulation is different. Instead of minimizing $\|\mathbf{w}\|_1$ that enforces a model sparseness only, we additionally require that the Chebyshev norm $\|\mathbf{G}\mathbf{w} - \mathbf{y}\|_\infty$ is minimal as well. In other words, we want to minimize the $\varepsilon$-tube inside which our regression function reside. Thus, we want to minimize regularized risk

$$R_{reg} = \lambda \|\mathbf{w}\|_1 + C_1 \|\mathbf{G}\mathbf{w} - \mathbf{y}\|_\infty . \tag{8}$$

If we divide (8) with $\lambda$ an optimal solution doesn't change and regularized risk functional becomes

$$R_{reg} = 1 \|\mathbf{w}\|_1 + C \|\mathbf{G}\mathbf{w} - \mathbf{y}\|_\infty \tag{9}$$

Same as in section 2.1 we introduce new variables $\mathbf{w}^+$ and $\mathbf{w}^-$. Second term on the RHS of our objective function (9) is not in a standard LP formulation either. Let's introduce new variables $r_i^+, r_i^-$ (Rice, 1964)

$$\begin{aligned} r_i^+ &= y_i - \mathbf{G}_i\mathbf{w} + d, \\ r_i^- &= -(y_i - \mathbf{G}_i\mathbf{w}) + d \end{aligned} \tag{10}$$

where $\mathbf{G}_i$ denotes the $i$-th row of the design matrix $\mathbf{G}$, $r$-s are deviations such that $r_i^+, r_i^- \geq 0$, $y_i$ is the known observation and $d$ is maximal deviation of the approximation, i.e.,

$$\begin{aligned} d &\geq -(y_i - \mathbf{G}_i\mathbf{w}) \\ d &\geq y_i - \mathbf{G}_i\mathbf{w} \end{aligned} \quad (11), \qquad \text{where } i = 1, 2, \ldots, P, \text{ or } \quad d \geq \max_i \left| y_i - \mathbf{G}_i\mathbf{w} \right| \tag{12}$$

The problem may now be reformulated as follows. Minimize $d$ subject to the $2P$ constraints $r_i^+, r_i^- \geq 0$. $d$ is equivalent to the $\varepsilon$ in the sparseness regularization method presented in 2.1. Linear programming problem now becomes

$$\min_{\mathbf{w},d} \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathrm{C} \end{bmatrix} \begin{bmatrix} \mathbf{w}^+ & \mathbf{w}^- & d \end{bmatrix}^T, \text{ subject to, } \begin{bmatrix} \mathbf{G} & -\mathbf{G} & -1 \\ -\mathbf{G} & \mathbf{G} & -1 \end{bmatrix} \begin{bmatrix} \mathbf{w}^+ & \mathbf{w}^- & d \end{bmatrix}^T \leq \begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix}, \tag{13}$$

$$\mathbf{w}^+ \geq \mathbf{0}, \ \mathbf{w}^- \geq \mathbf{0}, d \geq 0 \tag{14), (15), (16}$$

**Proposition:** *If expanded sparseness regularization leads to the solution* $\bar{\mathbf{w}}, \bar{d}$ *then sparseness regularization with* $\varepsilon$ *set apriori to* $\bar{d}$ *and with same parameters for basis functions, has the solution* $\bar{\mathbf{w}}$.

**Proof:** *It is easy to show that for a fixed d =* $\varepsilon$*, equation (13) is equal to (5).*

Since both methods lead to the same optimal solution, we present only first method in simulation examples section.

# 3. Linear Programming in Classification

The supervised learning algorithm embedded in a learning machine (which can be any of these - multilayer perceptron NN, SV machine, RBF network or neuro-fuzzy model) typically learns the separation boundary $f(\mathbf{x})$ by using a training data set $D = \{[\mathbf{x}(i), y(i)] \in \Re^n \times \Re, i = 1,...,P\}$ consisting of $P$ pairs $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_P, y_P)$, where the inputs $\mathbf{x}$ are $n$-dimensional vectors ($\mathbf{x} \in \Re^n$) and the class labels $y$ are discrete (e.g., Boolean) values for *classification problems*. $y_i$ are also called the desired values in classic supervised learning. Here, because of space constraints we do not present a full derivation of the LP based learning in the case of classification. Instead, we give only the final expressions for designing *linear* and *nonlinear separation boundaries* between classes. An LP learning algorithm for linear classification is given below

$$\min \sum_{i=1}^{P}(w_i^+ + w_i^-) + C\sum_{i=1}^{P}\xi_i, \quad \text{subject to,} \quad \begin{matrix} y_i f(\mathbf{x}_i) \geq 1-\xi_i \\ w_i^+, w_i^-, \xi_i \geq 0 \end{matrix} \tag{17}$$

where $f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b$ and $\xi_i$ measures the deviation of a data point from the ideal condition of pattern separability. It corresponds to the slack variable in a classic QP learning in SVMs. In a matrix notation, matrix of input data $\mathbf{X}$ is augmented by a constant input $+\mathbf{1}$, i.e., $\mathbf{X}=[\mathbf{x}_1^T\ 1;\ \mathbf{x}_2^T\ 1;\ \ldots,\ \mathbf{x}_P^T\ 1]$ and bias $b$ becomes the $n+1$-st component of the weights vector $\mathbf{w}$. For a *two-dimensional feature vector* $\mathbf{x}$, LP learning algorithm is given below,

$$\min[1\ 1\ 0\ 1\ 1\ 0\ C\mathbf{1}(1,P)]\left[w_1^+\ \ w_2^+\ \ w_3^+\ \ w_1^-\ \ w_2^-\ \ w_3^-\ \ \xi\right]^T, \text{ s.t., } \left[diag(\mathbf{y})\mathbf{X}\ \ -diag(\mathbf{y})\mathbf{X}\ \ \mathbf{I}(P)\right]\begin{bmatrix}\mathbf{w}^+\\ \mathbf{w}^-\\ \xi\end{bmatrix} \geq \mathbf{1}(P,1) \tag{18}$$

A trade-off constant $C$ is a design parameter that is typically adjusted separately during the design experiments. Classes that have nonlinear separating (hyper)boundaries have classifier constructed from some (in the case of LP approach, not necessarily kernel) basis functions. Here, we use a polynomial (which is, by chance, a kernel function) having 'design matrix' $G$. The LP learning formulation for a nonlinear classification tasks is given as

$$\min[\mathbf{1}(1,P)\ \ 0\ \ \mathbf{1}(1,P)\ \ 0]\left[w_1^+\ \ w_1^+\ \ \cdots\ \ w_{P+1}^+\ \ w_1^-\ \ w_2^-\ \ \cdots\ \ w_{P+1}^-\right]^T, \text{ s.t., } \left[diag(\mathbf{y})\mathbf{G}\ \ -diag(\mathbf{y})\mathbf{G}\right]\begin{bmatrix}\mathbf{w}^+\\ \mathbf{w}^-\end{bmatrix} \geq \mathbf{1}(P,1) \tag{19}$$

Similarly to (18), in a nonlinear classification formulation given by (19), one can also introduce $\xi_i$ variable and its regularizing i.e., trade-off parameter $C$. However, this should be done with the care. Actually, the impact of such regularization is, at the moment, under research in Hadzic (1999).

## 4. Simulation Examples

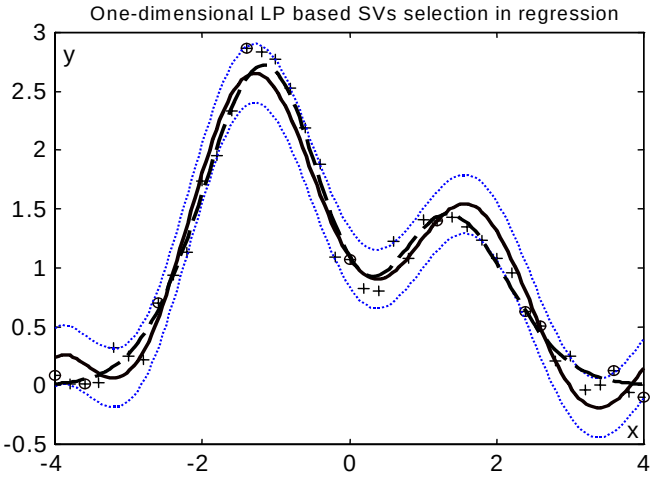



**Figure 1. Nonlinear regression:** The SVs selection based on an LP algorithm (5). Hermitian function f(x) = 1.1(1 – x + 2x$^2$)exp(-0.5x$^2$) (dashed) is polluted with a 10% Gaussian zero-mean noise. So obtained training set contains 41 training data points (crosses). An LP algorithm has selected 10 SVs shown as encircled data points. Resulting approximation curve (solid). Insensitivity zone is bounded by dotted curves.

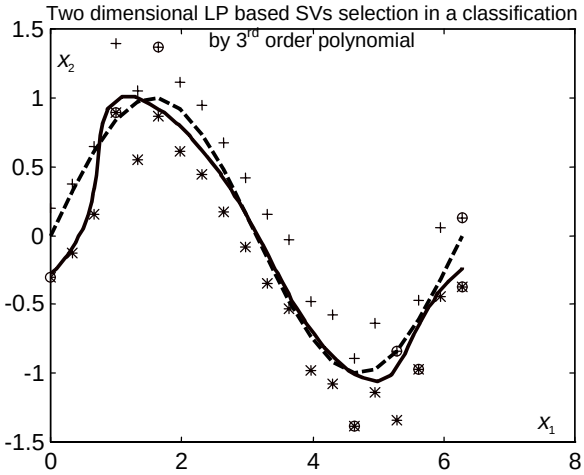



**Figure 2. Nonlinear Classification, i.e., Pattern Recognition:** The SVs selection based on an LP learning algorithm (19). Separation boundary is sinus function (dashed). Data are shown as crosses. Selected SVs are shown as encircled crosses. Out of 40 training data 8 were selected as SVs. Resulting separation boundary curve (solid).

*NONLINEAR REGRESSION*

In Fig 1, the SVs selection based on an LP learning algorithm (5) is shown for a Hermitian function f(x) = 1.1(1 – x + 2x$^2$)e(-0.5x$^2$) polluted with a 10% Gaussian zero-mean noise. The training set contained 41 training data pairs and

the LP algorithm has selected 10 data points as the SVs shown as encircled crosses. Basis functions are Gaussians $G(x_i, c_j = x_j) = \exp(-0.5\| x_i - c_j\|^2/(\sigma)^2)$ where $\sigma = k_S \Delta c$ where $\Delta c$ denotes the average distance between the adjacent centers of the basis functions. The resulting graph is similar to the standard QP based learning outcomes.

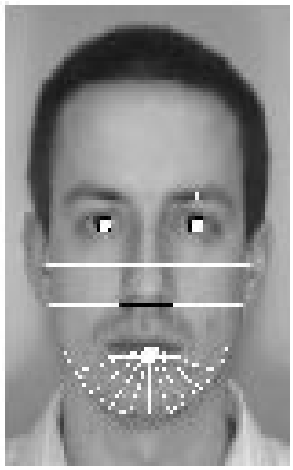
It was interesting to see differences between an LP and QP approach in solving regression problems. We investigated *sinus function* polluted with 20% noise. In a QP solution, an insensitivity zone was chosen to be $e = 0.1$, $C = 20 \cdot 10^6$ and the Gaussian bells parameter $\sigma = 3\Delta c$, (i.e., $k_S = 3$). Simulations were repeated on randomly selected data set one hundred times. Comparison of QP and LP based training algorithms shows that as number of training data increases computational time becomes significantly smaller for LP then for QP and that the number of chosen support vectors for LP is almost half their number for QP. At the same time, however, accuracy is slightly better for QP based algorithms. Compare a magnitude of the accuracy difference between the QP and the LP learning algorithms in the example on gender recognition below.

*NONLINEAR CLASSIFICATION*

Results for a nonlinear classification example are presented through a decision boundary estimation of the data belonging to two different classes separated by $x_2 = \sin(x_1)$. Here, the (kernel) function is the 3rd order polynomial. Despite the fact that the training data a polluted by 25% noise, resulting decision boundary, as seen in Fig. 2, is a good approximation to a sinus separation boundary curve and data are correctly classified.

The last example is a *gender recognition task* as given in Brunelli and Poggio, 1993 and in Poggio and Girosi, 1993. Data base contains 168 data pairs comprising 18-dimensional input vector **x** and one dimensional output +1 and –1 for male and female class respectively. Two typical faces are given in Fig 3. An input vector **x** comprises 18 geometrical features as given in Fig. 4. The results obtained in the Brunelli and Poggio's paper by applying (Hyper)Gaussian Basis Function Networks are as follows: *a*) on the vectors of the training set (classification is 90% correct), *b*) *on novel faces of people in the training set* (*86% correct*) and *c*) on faces of people not represented in the training set (79% correct). Human performance on all data sets was (90%). Our first results by applying QP and LP based learning are better in all three cases. For example, *on novel faces of people not represented in the training set*, QP based method was (96.4% correct) while an LP based method as given by (19) was (92,8% correct). Note that both the QP and the LP approach outperform human performance.

**Figure 3.** Typical photos of male and female face without gender facial hair used in gender classification experiments



| Nr | Feature |
|---|---|
| 1 | pupil to nose vertical distance |
| 2 | pupil to mouth vertical distance |
| 3 | pupil to chin vertical distance |
| 4 | nose width |
| 5 | mouth width |
| 6 | zygomatic breadth |
| 7 | bigonial breadth |
| 8–13 | chin radii |
| 14 | mouth height |
| 15 | upper lip thickness |
| 16 | lower lip thickness |
| 17 | pupil to eyebrow separation |
| 18 | eyebrow thickness |

**Figure 4.** Geometrical features (white) used in classification and their description. (Both figures are from Poggio and Girosi, 1993).

## 5. Conclusions

In this paper, we present an LP approach for solving *nonlinear* regression and classification tasks. For a regression, we introduced two methods – sparseness and expanded sparseness regularization method. In the case of nonlinear

classification, we presented two LP problem formulations – first one for a calculation of a separation hyperplane and the second formulation is aimed at calculating nonlinear separation hypersurface between two classes. The approach and results show that an LP based learning (that resulted from the minimization of the $L_1$ norm of the weight vector **w**) produces sparse networks. This may be of particular interest in modern days applications while processing a huge amount of data (say several dozens of thousands) that can not be processed by the contemporary QP solvers. The results presented here are inconclusive in the sense that much more investigations and comparisons with a QP based SVs selection on both real and 'artificial' data sets should be done. Despite the lack of such an analysis yet, our first simulational results show that the LP subset selection may be more than a good alternative to the QP based algorithms when faced with a huge training data sets. Saying that we primarily refer to the following possible benefits of applying LP based learning: *a*) LP algorithms are faster and more robust than QP ones, *b*) they tend to minimize number of weights (meaning SV) chosen, and *c*) they naturally incorporate the use of kernels for creation of nonlinear separation and regression hypersurfaces in pattern recognition and function approximation problems respectively.

## 6. References


**Arthanari, T. S., Y. Dodge,** 1993. Mathematical Programming in Statistics, J. Wiley & Sons., New York, NY

**Bennett, K.,** 1999. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C. Burges, and A. Smola, Editors, *Advances in Kernel Methods - SV Learning*, pp 307–326, MIT Press, Cambridge, MA

**Brunelli, R., T. Poggio**, 1993. Caricatural Effects in Automated Face Perception, Biological Cybernetics, 69: 235-241

**Charnes, A., W. W. Cooper, R. O. Ferguson,** 1955. Optimal Estimation of Executive Compensation by Linear Programming, Manage. Sci., 1, 138

**Cheney, E. W., A. A. Goldstein,** 1958. Note on a paper by Zuhovickii concerning the Chebyshev problem for linear equations. J. Soc. Indust. Appl. Math., Volume 6, pp. 233-239

**Eisenhart, C.,** 1962. Roger Joseph Boscovich and the Combination of Observationes, Actes International Symposium on R. J. Boskovic, pp. 19-25, Belgrade-Zagreb-Ljubljana, YU

**Graepel, T., R. Herbrich, B. Schölkopf, A. Smola, P. Bartlett, K.–R. Müller, K. Obermayer, R. Williamson,** 1999. Classification on proximity data with LP–machines, Proc. of the 9th Intl. Conf. on Artificial NN, ICANN 99, Edinburgh,7-10 Sept.

**Hadzic, I.,** 1999. Learning from Experimental Data by Linear Programming Selected Support Vectors, PhD Thesis, (work in progress), The University of Auckland, Auckland, NZ

**Hadzic, I., V. Kecman,** 1999. Learning from Data by Linear Programming, NZ Postgraduate Conference Proceedings, Auckland, Dec. 15-16

**Kecman, V.,** 1999. Learning and Soft Computing by SVM, NN and FLS, The MIT Press, (in print), Cambridge, MA

**Kelley E. J. jr.,** 1958. An application of linear programming to curve fitting (received by editors October 11, 1957), J. Soc. Indust. Appl. Math., Volume 6, pp. 15-22

**Mangasarian, O. L.,** 1965. Linear and Nonlinear Separation of Patterns by Linear Programming, Operations Research 13, pp. 444-452.

**Poggio, T., F. Girosi,** 1993. Learning, Approximation and Networks, Lectures and Lectures' Handouts - Course 9.520, MIT, Cambridge, MA,

**Rice R. J.,** 1964. The Approximation of Functions, Addison - Wesley Publishing Company, Reading, Massachusetts, Palo Alto, London

**Smola A., T. T. Friess, B. Schölkopf,** 1998. Semiparametric Support Vector and Linear Programming Machines, NeuroCOLT2 Technical Report Series, NC2-TR-1998-024

**Smola, A., B. Schölkopf, G. Rätsch,** 1999. Linear Programs for Automatic Accuracy Control in Regression, submitted to ICANN'99

**Stiefel, E.,** 1960. Note on Jordan Elimination, Linear Programming and Chebyshev Approximation, Numer. Math., 2, 1.

**Vapnik, V. N.,** 1995. The Nature of Statistical Learning Theory, Springer Verlag Inc, New York, NY

**Weston, J., A. Gammerman, M. O. Stitson, V. Vapnik, V. Vovk, C. Watkins,** 1999. Support Vector Density Estimation, In B. Schölkopf, C. Burges, and A. Smola, Editors, *Advances in Kernel Methods - SV Learning*, p.p. 307–326, MIT Press, Cambridge, MA

**Zhang, Q. H., J.-J. Fuchs,** 1999. Building neural networks through linear programming, Proceedings of 14th IFAC Triennial World Congress, Vol. K, pp. 127-132, Pergamon Press