5th Seminar on Neural Network Applications in Electrical Engineering, *NEUREL 2000*
Faculty of Electrical Engineering, University of Belgrade, Yugoslavia
Proceedings, pp. 18-23, 2000

Support Vector Machines Trained by Linear Programming: Theory and Application in Image Compression and Data Classification

Ivana Hadzic, Vojislav Kecman

Department of Mechanical Engineering
University of Auckland, Auckland
New Zealand
E-mail: *ihad004@student.auckland.ac.nz, v.kecman@auckland.ac.nz*

Abstract

This paper formulates the learning of support vector machines (SVMs) as a linear programming problem. An SVM has the property that it chooses the minimum number of data points to use as the centres for the Gaussian kernel functions in order to approximate the training data within a given error. A linear programming (LP) based method is proposed for solving both regression and classification problems. Examples of function approximation and class separation illustrate the efficiency of the proposed method. In addition, the paper explores the possibility of using SVMs with radial basis function kernels to compress an image. Our results show that image compression of around 20:1 is achievable while maintaining good image quality.

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

1. Introduction

Support Vector Machine (SVM) is a new type of learning machine that keeps the training error fixed (ie., within given boundaries), and minimises the confidence interval, ie., machine capacity. SVM that uses quadratic programming 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.

The possible application of SVMs (i.e., neural networks) for image compression is based on the fact that most images contain spatial redundancy due to the high probability that the adjacent pixels will be of a similar colour. As images are two dimensional, a colour surface can be created from the image which maps the (x, y) coordinates of a pixel to a specific colour.

The compression problem can be stated as how to model the colour surface using the fewest parameters while still retaining a good approximation of the surface. This can be formulated as a *nonlinear regression problem* as we are attempting to model a function of the form $C= f(x, y)$ where C stands for a colour value.

Recently, a lot of work was done on implementing linear programming (LP) approach in support vectors selection. Kecman (1999) suggested optimal subset selection by using LP in solving regression. Zhang and Fuchs (1999) proposed an LP based method for selecting the hidden neurones at the initialisation stage of the multilayer perceptron network. One of the very first implementation of mathematical programming to statistical learning from data was the paper Charnes, Cooper and Ferguson (1955). Arthanari and Dodge (1993) gave the summary and very good presentation of mathematical programming in statistics.

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). They are all close to the SVM constructive algorithms.

Interestingly, the first results on the L_1 norm estimators were given as early as 1757 by Yugoslav scientist Boskovic (see Eisenhart, 1962).

Here, the slight difference in respect to the standard SVM learning (that uses QP programming) is that instead minimizing L_2 norm of the weight vector $\|\mathbf{w}\|_2$ we will minimise L_1 norm $\|\mathbf{w}\|_1$ in an LP approach here. In Section 2 we develop methods for solving regression problems by LP. In Section 3 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

Minimisation of the L_2 norm equals minimizing $\mathbf{w}^{\mathrm{T}}\mathbf{w} = \sum_{i=1}^{n} w_i^2 = w_1^2 + w_2^2 + \cdots + w_n^2$ and this results in the QP type of problem that leads to a maximisation of a margin M (Vapnik 1995). The application 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 minimisation of the L_1 norm or L_2 norm of the weight vector $\mathbf{w}$ produces superior generalisation.

Regression problem solving could be formulated in many different ways. Here we show two approaches. First method is a one-objective function minimisation problem and the second one is multi-objective function minimisation task.

2.1 Sparseness regularisation method

Our problem is to design a parsimonious NN containing less neurones than data points. Such a sparseness of a NN or SVM is controlled by minimizing L_1 norm of the weight vector $\mathbf{w}$. At the same time, we want to solve $\mathbf{y} = \mathbf{G}\mathbf{w}$ such that $\| \mathbf{G}\mathbf{w} - \mathbf{y} \|$ is small for some chosen norm. In order to perform this task we formulate the regression problem as follows.

Find a weight vector

$$\mathbf{w} = \arg\min \|\mathbf{w}\|_1, \text{ subject to,}$$

$$\| \mathbf{G}\mathbf{w} - \mathbf{y} \|_\infty \le \varepsilon, \tag{1}$$

where ε defines the *maximally* allowed error (that is why we used L_∞ norm) and corresponds to the ε-insensitivity zone in SVM. This constrained optimisation problem can easily be transformed into a standard linear programming form. First, recall that $\|\mathbf{w}\|_1 = \sum_{p=1}^{P} | w_p |$, and this is not an LP problem formulation where we typically minimise $\mathbf{c}^T\mathbf{w} = \sum_{p=1}^{P} c_p w_p$ and $\mathbf{c}$ is some known coefficient vector. 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, ie., $w_p^+ \ge 0$, $w_p^- \ge 0$. P equals to the number of the training data. Note, that the substitutions done in (2) are unique, ie., for a given w_p there is only one pair (w_p^+, w_p^-) which fulfils 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 $\| \mathbf{G}\mathbf{w} - \mathbf{y} \|_\infty \le \varepsilon$ in (1) defines an ε-tube inside which should our approximating function reside. Such a constraint can be rewritten as

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

where $\mathbf{1}$ is a (P, 1) column vector filled with ones. (3) represents a standard set of linear constraints and our LP problem to solve is now the following.

Find a pair

$$(\mathbf{w}^+, \mathbf{w}^-) = \arg\min \sum_{p=1}^{P} (w_p^+ + w_p^-),$$

subject to,

$$\mathbf{y} - \varepsilon\,\mathbf{1} \le \mathbf{G}(\mathbf{w}^+ - \mathbf{w}^-) \le \mathbf{y} + \varepsilon\,\mathbf{1}, \tag{4a}$$

$$\mathbf{w}^+ \ge \mathbf{0}, \quad \mathbf{w}^- \ge \mathbf{0}, \tag{4b), (4c)}$$

where $\mathbf{w}^+ = [\, w_1^+ \;\; w_2^+ \;\; \ldots \;\; w_P^+ \,]^T$ 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 program solver as follows

$\min_{\mathbf{w}} \mathbf{c}^T\mathbf{w}$, i.e.,

$$\min_{\mathbf{w}} \begin{bmatrix} 1 & 1 \underset{P}{\ldots} 1 & 1 & 1 \underset{columns}{\ldots} 1 \end{bmatrix} \begin{bmatrix} w_1^+ \\ w_2^+ \\ \vdots \\ w_P^+ \\ w_1^- \\ w_2^- \\ \vdots \\ w_P^- \end{bmatrix}$$

subject to,

$$\begin{bmatrix} \mathbf{G} & -\mathbf{G} \\ -\mathbf{G} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{w}^+ \\ \mathbf{w}^- \end{bmatrix} \le \begin{bmatrix} \mathbf{y}+\varepsilon\mathbf{1} \\ -\mathbf{y}+\varepsilon\mathbf{1} \end{bmatrix}, \qquad (5)$$

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

where both $\mathbf{w}$ and $\mathbf{c}$ are the $(2P, 1)$-dimensional vectors. $\mathbf{c} = \mathbf{1}(2P, 1)$, ie., $\mathbf{c}$ is a $(2P, 1)$ vector filled with ones and $\mathbf{w} = [\mathbf{w}^{+T} \ \ \mathbf{w}^{-T}]^T$. Examples that illustrate the efficiency of the method are given in the Section 4.

Instead of minimizing $\|\mathbf{w}\|_1$ that enforces a sparseness of the model only, we have an additional requirement that the Chebyshev norm $\|\mathbf{Gw} - \mathbf{y}\|_\infty$ is minimal as well. In other words, we want to minimize the ε-tube, ie., we want to minimize the residual or error function.

Thus, given data samples, we try to obtain a small value of risk by minimizing regularised risk functional

$$R_{reg} = 1\,\|\mathbf{w}\|_1 + C\|\mathbf{Gw} - \mathbf{y}\|_\infty \qquad (8)$$

It can be shown [8] that this expanded sparseness regularisation method is really equivalent to the sparseness method given in (4).

Since both methods essentially lead to the same optimal solutions we present only first method in the 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 input-output relationship (dependency or function) $f(\mathbf{x})$ 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 labels (or system responses) $y \in \Re$ are continuous values for regression tasks and discrete (eg., Boolean) for classification problems. 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.

LINEAR CLASSIFICATION:

The LP problem to solve is,

$$\min \sum_{i=1}^{P} (w_i^+ + w_i^-) + C\sum_{i=1}^{P} \xi_i \qquad (9)$$

subject to,

$$\begin{aligned} & y_i f(\mathbf{x}_i) \ge 1 - \xi_i \\ & w_i^+, w_i^-, \xi_i \ge 0 \end{aligned} \qquad (10)$$

where decision function is given as

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \qquad (11)$$

and ξ_i measures the deviation of a data point from the ideal condition of pattern separability. It corresponds to the slack variable in classic QP learning in SVMs.

A matrix notation for a *two-dimensional feature vector* $\mathbf{x}$ that includes a bias $b = 1$ as a 3rd component of $\mathbf{x}$ is given below,

$$\min[1 \ 1 \ 0 \ 1 \ 1 \ 0 \ C\cdot\mathbf{1}(P,1)] \begin{bmatrix} w_1^+ \\ w_2^+ \\ w_3^+ \\ w_1^- \\ w_2^- \\ w_3^- \\ \xi \end{bmatrix} \qquad (12)$$

such that,

$$\left[diag(\mathbf{y}) \cdot \mathbf{x} \quad -diag\ \mathbf{y} \bullet \mathbf{x} \quad \mathbf{I}\ P \right] \begin{bmatrix} \mathbf{w}^{+} \\ \mathbf{w}^{-} \\ \xi \end{bmatrix} \geq \mathbf{1}(13)$$

where C is some trade-off constant or a penalty parameter that should be chosen by user during the design stage, ξ is a (P, *1*) vector of deviations and $\mathbf{I}_y$ is an identity (P, P) matrix.

NONLINEAR CLASSIFICATION:

Classes that are not only overlapped, but have nonlinear separating boundaries, would be separated by a NN or SVM having hidden layer of neurones. Such a classifier is still linear but in an imaginary hidden space. Now, in a nonlinear case the LP formulation of the problem is as follows,

$$\min[\mathbf{1}(P,1)\ \ 0\ \ \mathbf{1}(P,1)\ \ 0] \begin{bmatrix} w_1^{+} \\ w_2^{+} \\ \vdots \\ w_{P+1}^{+} \\ w_1^{-} \\ w_2^{-} \\ \vdots \\ w_{P+1}^{-} \end{bmatrix}^{T} \qquad (14)$$

such that,

$$\left[diag(\mathbf{y})\mathbf{G} \quad -diag\ \mathbf{y}\ \mathbf{G} \right] \begin{bmatrix} \mathbf{w}^{+} \\ (\ ) \\ \mathbf{w}^{-} \end{bmatrix} \geq \mathbf{1}\ P \quad , \qquad (15)$$

where $\mathbf{G}$ is a so-called design or kernel matrix, $\mathbf{1}$(P, 1) is again a row vector filled with ones and P (same as previously) represents the number of the training data points. In the formulation above, a design matrix $\mathbf{G}$ is augmented by a bias column, ie., $\mathbf{G}$ is an (P, P +1) matrix.

4. Simulation Examples

NONLINEAR REGRESSION

In Fig 1, the SVs selection based on an LP learning algorithm (5) is shown for a Hermitian function $f(x) = 1.1(1 - x + 2x^2)\exp(-0.5x^2)$ polluted with a 10% Gaussian zero-mean noise. The training set contained 41 training data pairs and the LP algorithm has selected 10 SVs shown as encircled data points. The resulting graph is similar to the standard QP based learning outcomes.

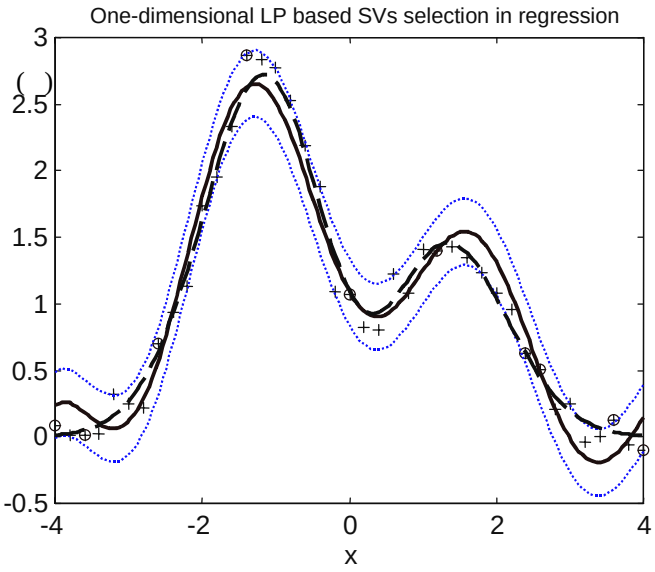


Figure 1. Nonlinear regression: The SVs selection based on an LP learning algorithm (5). Hermitian function f(x) = 1.1(1 – x + 2x^2)exp(-0.5x^2) polluted with a 10% Gaussian zero-mean noise (dashed). The training set contains 41 training data points (crosses). An LP algorithm has selected 10 SVs shown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

It was interesting to see differences between QP and LP based methods. We investigated sinus function polluted with 20% noise. In QP solution, an insensitivity zone was chosen to be e=0.1, C=20·10^6 and the Gaussian bells parameter σ =3Δc_{ij}, where Δc_{ij} denotes the distance between the adjacent centres of the basis functions. Simulations were repeated on randomly selected data set 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. However, accuracy is slightly better for QP based algorithms.

SVM Compression Method

J. Robinson proposed in [12] to use NN in image compression treating such a problem in original color domain as a nonlinear regression (approximation) problem. His work is still in progress trying to apply QP based SVM learning for these tasks. Here we present some results on applying LP learning algorithm for image compression. We used popular benchmark - Lena image here. The image is subdivided into 8 × 8 blocks (64 pixels per block). Each block is used as training data for a SVM. In a 256 × 256 image there are 1024 blocks, thus 1024 subnetworks will be produced. The value of design parameters ε and σ depends highly on the 'colour surface' that is to be modelled by the SVM.

SVM Results

The Lena picture was used as a test image and some of the LP method results are presented in figure 2. In the higher compression image, a 'blocking' effect can be seen. This is caused by the average colour of each block being different.

Original Image
65536 Pixels

Error = 20, σ = 10

No of Neurons = 3697

Compression = 18:1

Error = 10, σ = 10

No of Neurones = 7269

Compression = 9:1

Figure 2. The benchmark 256 × 256 Lena picture compressed for various values of error and σ.

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 $x_2 = x_1^2$. Kernel (i.e., basis) function is a standard Gaussian RBF $G(x_i, c_j = x_j) = \exp(-0.5\| x_i - c_j\|^2/(k_S \Delta c)^2)$, and parameter that defines its width k_S =5. Resulting decision boundary, as seen in Fig 3, is almost parabolic and data are correctly classified. For smaller values of coefficient k_S, kernel function has narrower bells and algorithm picks up more SVs. In that case, decision boundary is more data dependant.

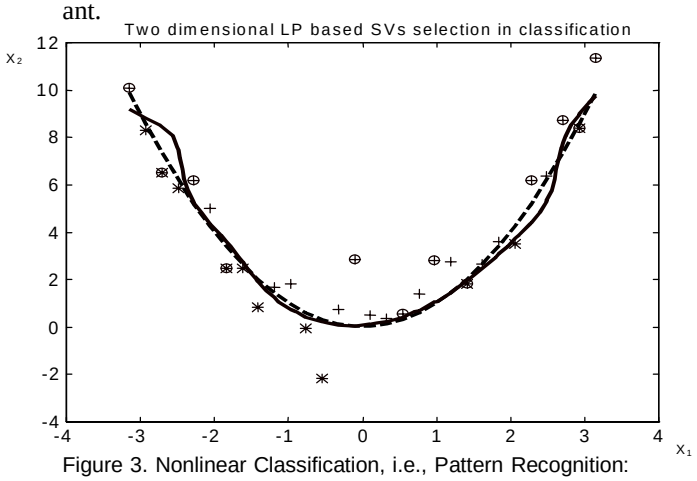
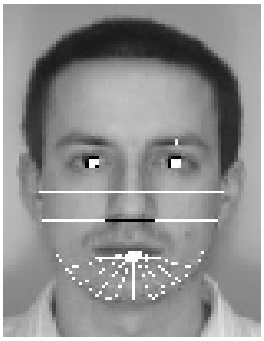


Figure 3. Nonlinear Classification, i.e., Pattern Recognition: The SVs selection based on an LP learning algorithm. Separation boundary is a parabola (dashed). Data are shown as crosses. Selected SVs are shown as encircled crosses. Resulting separation curve (solid).

In addition, we show results for a nonlinear classification example through a decision boundary estimation of the data belonging to male and female faces.



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. Nonlinear Classification, ie., gender recognition problem. In the upper picture standard male and female faces are presented. Our task was to distinguish sexes upon 18 dimensional data set.

Brunelli R., Poggio T. [3], developed NN - Hyper Basis Function (BF) Networks for Gender Classification method for solving nonlinear classification problems. On the same data set as us, for gender recognition problem they had following results: on

the vectors of the training set they were 90% correct, on novel faces of people in the training set 86% correct and on faces of people not represented in the training set 79% correct. Human performance is 90% correct. We had on the novel faces of people not represented in the training set with QP based method very good precision, 96.4% were correct. On the novel faces of people not represented in the training set and classified with LP based method we had reasonably good correctness rate - 92,8% were correctly classified.

Same as in previous example, for smaller values of coefficient σ, kernel function has narrow bells and algorithm picks up more SVs. Thus, decision boundary is more data dependent.

5. Conclusions

In this paper, we present an LP approach for solving *nonlinear* regression and classification tasks. The results that we have got in Lena example are promising but we still have blocking effect and matrices are not sparse. The method proposed here is still in the development phase. The approach and results show that an LP based learning (that resulted from the minimisation of the L_1 norm of the weight vector) produces sparse networks. This may be of particular interest in modern days applications while processing a huge amount of data and when the contemporary QP solvers seem to be not that suitable. We primarily refer to the following possible benefits of applying LP based learning: LP algorithms are faster and more robust than QP ones, they tend to minimise number of weights (meaning SV) chosen, 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

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

[2] 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

[3] Brunelli R., Poggio, 1993. Caricatural Effects in Automated Face Perception. Biological Cybernetics.

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

[5] 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

[6] 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.

[7] Kecman, V., *Learning and Soft Computing by SVM, NN and FLS*, The MIT Press, Cambridge, MA, 2000

[8] Kecman, V., I. Hadzic, Support Vectors Selection by Linear Programming, Proceedings of International. Joint Conference on NN (IJCNN 2000), Vol. 5, pp. 193-198, Como, Italy, 2000

[9] 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

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

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

[12] Robinson J, 2000. The Use of Support Vector Machines in Image Compression. Ph.D. thesis in progress, Mechanical Engineering Department, School of Engineering, Auckland University

[13] Schölkopf, B., P. Bartlett, A. Smola, R. Williamson. Support Vector Regression with Automatic Accuracy Control. In: L. Niklasson and M. Boden and T. Ziemke, Editors. Proceedings of the 8th International Conference on Artificial NN, pp. 111 - 116, Springer Verlag, Perspectives in Neural Computing, Berlin, 1998.

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

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