This is a repository copy of *Mathematical Programming Potential Functions: New Algorithms for Nonlinear Function Approximation*.

White Rose Research Online URL for this paper:
http://eprints.whiterose.ac.uk/82557/

Monograph:
Frieb, Thilo-Thomas and Harrison, R.F. (1998) Mathematical Programming Potential Functions: New Algorithms for Nonlinear Function Approximation. Research Report. ACSE Research Report 738 . Department of Automatic Control and Systems Engineering

---

Reuse
Unless indicated otherwise, fulltext items are protected by copyright with all rights reserved. The copyright exception in section 29 of the Copyright, Designs and Patents Act 1988 allows the making of a single copy solely for the purpose of non-commercial research or private study within the limits of fair dealing. The publisher or other rights-holder may allow further reproduction and re-use of this version - refer to the White Rose Research Online record for this item. Where records identify the publisher as the copyright holder, users can verify any specific terms of use on the publisher's website.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

Mathematical Programming Potential Functions: New Algorithms for Non-Linear Function Approximation

Thilo-Thomas Frieß and Robert F. Harrison

The University of Sheffield
Department of Automatic Control and Systems Engineering
Mappin Street, Sheffield, S1 3JD, England
email: friess@acse.shef.ac.uk

Date: 31 December 1998

Research Report 738

200448936

SHEFFIELD UNIV. LIBRARY ST. GEORGE'S

Abstract: Linear and quadratic-programming perceptrons for regression are new potential function methods for nonlinear function approximation. Potential function perceptrons, which have been proposed by Aizerman and colleages in the early 1960s, work in the following way: in the first stage the patterns from the training set are mapped into a very high dimensional linearisation space by performing a high dimensional non-linear expansion of training vectors into the so-called linearisation space. In this space the perceptron's decision function is determined.
In the algorithms proposed in this work a non-linear prediction function is constructed using linear- or quadratic-programming routines to optimize the convex cost function. In Linear Programming Machines the $L1$ loss function is minimised, while Quadratic Programming Machines allow the minimisation of the $L2$ cost function, or a mixture of both the $L1$ and $L2$ noise models. Regularisation is implicitly performed by the expansion into linearisation space (by choosing a suitable kernel function), additionally weight decay regularisation is available. First experimental results for one-dimensional curve-fitting using linear programming machines demonstrate the performance of the new method.

1 Introduction

In the early 1960s Aizerman and colleages [1] have proposed a universal framework called the *method of potential functions* which allows, under some circumstances, linear algorithms to be applied to machine learning tasks where the relationship between the input patterns (called the training set) and the target values is non-linear[1]. All that is required from the linear algorithm is that a scalar-product (or a Euclidian distance) must be available in some sense [4]. By replacing dot products by positive semidefinite functions, the so called potential functions, in the algorithm automatically the following two steps are performed:

- ◇ expand two patterns into a very high dimensional representation $[u, v] \rightarrow [\psi(u), \psi(v)]$;
- ◇ compute the scalar product $\langle \psi(u), \psi(v) \rangle$ of two patterns in this high dimensional linearisation space.

An intrinsic property of linear perceptrons allows the linear decision function to be represented as a weighted expansion of dot products, therefore any linear perceptron can be run in the non-linear linearisation space (further information can be found in [1] [5]). Kernel Rosenblatt perceptrons [1] [17], Kernel Adalines [5], and support vector machines [22] are some examples for potential function perceptrons.

In these algorithms the weight vector of the perceptron is expanded as a weighted sum of training patterns where the weights on the expansion patterns (called multipliers) are, in

[1]Non-linear is used here in the sense that the target function to be learned cannot be expressed as a linear combination of training patterns. However the target function can be expressed as a linear combination of training patterns *after* mapping the patterns into the linearisation space.

nearly any case, not equal. In contrast in Hebbian perceptron learning[2], one of the earliest learning methods inspired by biologically plausible processes, the weight vector of the perceptron is an equally weighted sum of normalized training patterns. It is known that in correlation based learning such as Hebbian learning, only correlated patterns can be associated correctly. It is also possible to derive a non-linear version of Hebbian learning perceptrons using potential functions. An adaption of this algorithm has been studied by Specht [20] and has been called the probabilistic neural network (PNN). The properties of correlation based learning, which have been studied in the context of Hebbian learning [16], carry over to all potential function algorithms, because these methods are based on correlations in the linearisation space. Depending on the type and properties of the mapping to the linearisation space, defined by the potential function kernel, linearly dependent patterns can exist in the linearisation space (further informations can be found in [5]).

This document is structured as follows: In the first section Minnick's and Muroga's linear programming perceptron is reviewed [11] [13]. In the next section linear programming machines (LPM) and quadratic programming machines (QPM) for pattern classification [3] [4] are discussed. In the third section LPM's and QPM's for linear and non-linear regression are proposed; different noise-models are considered.
Finally first experimental results using linear programming machines for regression (LPM-R) are given.

2 Linear Models for Pattern Classification

To classify a set of labeled training patterns $x_i \in R^n, i \in \{1..l\}$ by a linear function $f(x) = \langle w, x \rangle$ a linear program of the following form can be solved:

$$\begin{aligned} min: \quad & \sum_{i=1}^{l} s_i \\ subject\ to: \quad & y_i \langle w, x_i \rangle + s_i \geq t \quad \forall i \in \{1..l\} \\ & s_i \geq 0 \quad \forall i \in \{1..l\} \end{aligned} \tag{1}$$

The aim is to find a weight vector, w, such that the prediction of the model, $sign(f(x))$, matches the given class label $y_i \in \{+1, -1\}$ of as many patterns, x_i, as possible (that is $sign(f(x_i)) = y_i$ for most i). For those patterns, x_i, which cannot be classified correctly the associated slack variable, s_i, is minimized.
This algorithm was studied in the early 1960s by Minnick and Muroga et al. [11] [13]. For the constant, t, any nonnegative number can be chosen. To avoid the solution $w_i = 0 \ \forall i \in \{1..l\}$ constant, t, should be greater than zero, e.g. $t = 1$. A bias parameter can be introduced into the decision function by replacing the first constraint in (1) by a constraint of the form $y_i(b + \langle w, x_i \rangle) + s_i \geq t \quad \forall i \in \{1..l\}$.

[2] referring to the pattern associator model as described in [16]

In order to minimize the $L2$ (MSE) cost function one can use quadratic programming to minimse the cost function $Y = \sum_{i=1}^{l} s_i^2$ subject to the constraints of (1). Also a mixture of $L1$ and $L1$ loss is possible just by minimizing a combined, but still quadratic, cost function $Y = m_1 \sum_{i=1}^{l} s_i^2 + m_2 \sum_{i=1}^{l} s_i$ subject to the constraints given in (1). Constants m_1 and m_2 allow a tradeoff the two noise-models in the cost function.

3 Non-Linear Models for Pattern Classification

3.1 Linear Programming Machines (LPM)

A nonlinear version of these algorithms (called the linear programming machine, and the quadratic programming machine, respectively) has been derived by replacing the linear decision function, in the first constraint of (1), $f(x_i) = b + \langle w, x_i \rangle = b + \sum_{i=1}^{l} \alpha_i \langle x, x_i \rangle$, by a potential decision function:

$$f(x_i) = b + \langle \psi(w), \psi(x_i) \rangle = b + \sum_{j=1}^{l} \beta_i K(x_j, x_i) \tag{2}$$

where K is a nonlinear positive semidefinite potential function. Alternatively one can replace the β_i in the formulation above each by $\alpha_i y_i$, $\alpha_i \geq 0 \forall i \in \{1..l\}$.
It is interesting to note that the evaluations of functional $K(x_i, x_j) = h(i,j)$ $\forall i,j \in \{1..l\}$ can be fixed before the optimization process because both the training patterns and function K is available a priori. One example of a potential kernel is the radial basis function $K(u, v, \sigma) = exp(\frac{-\sum_{i=1}^{l}(u_i - v_i)^2}{\sigma^2})$. This kernel, which has been used in the computer experiments below, has a parameter σ, which specifies the kernel width. For a discussion on the choice and design of potential function kernels see [1]; where it has been discussed how to run Rosenblatt perceptrons to learn non-linear relationships between training patterns and their target values using potential kernel functions.
Linear programming machines for pattern classification were proposed for the first time in [3]. There it is discussed how overfitting can be avoided using a weight decay regularisation technique by penalizing additionally the sum of multipliers α in the cost function. This penalizer represents $L1$ regularisation on the multipliers α.

Two parameters are chosen in advance, that is the the kernel parameter and the C parameter which specifies the degree of weight decay regularisation. Methods of model selection can be applied to find optimal values for the two constants. The LPM's solution is found by solving the following linear program:

$$\begin{aligned} min: \quad & \sum_{i=1}^{l} s_i + C \sum_{i=1}^{l} \alpha_i \\ subject\ to: \quad & y_i(b + \alpha_1 h(i,1) + \alpha_2 h(i,2) ... + \alpha_l h(i,l)) + s_i \geq t \quad \forall i \in \{1..l\} \\ & \alpha_i \geq 0 \ ; \ \ s_i \geq 0 \quad \forall i \in \{1..l\} \end{aligned} \tag{3}$$

3.2 Quadratic Programming Machines (QPM)

It is possible to use different methods for complexity regularisation, e.g. a weight decay regularizer which penalizes the $L2$ norm of the vector of the multipliers α.
This can be expressed by the following cost function:

$$Y = m_2 \sum_{i=1}^{l} s_i^2 + m_1 \sum_{i=1}^{l} s_i + n_1 \sum_{i=1}^{l} \alpha_i + n_2 \sum_{i=1}^{l} \alpha_i^2 \tag{4}$$

For the case that the constant n_1 is zero, the cost function Y represents $L1$ error minimization with a quadratic penalty on the vector of multipliers α. The mixture of the penalizers (both for the empirical error measures and regularizers) is adjusted by choosing values for constants n_1, n_2 and m_1, m_2. Since now quadratic programming is required to minimize the function Y, subject to the constraints given above in (3), the method will be called the *quadratic programming machine* [4]. The decision function of the algorithm is given by (2).

4 Linear and Non-Linear Models for Regression

4.1 Linear Regression using Linear Programming

In the following it is shown that the problem of estimating an unknown function, given some training patterns x_i, $i \in \{1..l\}$, and "labels" y_i, $i \in R$, can be solved by linear programming.
Again the slack variables, that is the discrepancy between a pattern's true label, y_i, and the model's prediction, $f(x_i)$, are minimized in the $L1$ norm using linear programming.
Two cases are possible, either (a) $f(x_i) > y_i$, or (b) $f(x_i) \leq y_i$. Therefore, in regression, two slack variables, an "upper" slack and a "lower" slack, are required for each pattern. It is clear that, for each pattern, only one of the two variables can be greater than zero, depending on which case, (a) or (b), is satisfied. The linear program which must be solved for linear regression is given by:

$$\begin{aligned} & min: \quad \sum_{i=1}^{l} s_i^{up} + \sum_{i=1}^{l} s_i^{lo} \\ subject\ to: \quad & b + \langle w, x_i \rangle - y_i + s_i^{up} = 0 \quad \forall i \in \{1..l\} \qquad (a) \\ & b + (-1) * (b + \langle w, x_i \rangle) + y_i + s_i^{lo} = 0 \quad \forall i \in \{1..l\} \qquad (b) \\ & s_i^{lo}, s_i^{up} \geq 0 \quad \forall i \in \{1..l\} \end{aligned} \tag{5}$$

Note that in this algorithm a linear decision function of the form $f(x) = b + \langle w, x \rangle$ is used for prediction.

4.2 LP Machines for Regression (LPM-R)

The algorithm above can be generalised such that regression of non-linear signals is possible. In each constraint of the algorithm above a linear function, the prediction function,

is used. If the linear decision function is replaced by its nonlinear equivalent (2) then the non-linear algorithm can be derived. Again it is crucial to understand that this can be done only because in the linear decision function, $f(x_i) = b + \langle w, x_i \rangle = b + \sum_{j=1}^{l} \beta_i \langle x_i, x_j \rangle$, the dot products can be replaced by potential functions. For all training points, x_i, the potential can be computed by calculating the correlation matrix in the linearisation space $M_{i,j} = h(i,j) = K(x_i, x_j)$. Therefore the, so far linear, constraints in the algorithm can be replaced by the constraints which stem from the non-linear decision function. The non-linear version of the algorithm is given by:

$$
\begin{aligned}
& min: \quad \sum_{i=1}^{l} s_i^{up} + \sum_{i=1}^{l} s_i^{lo} \\
subject\ to: \quad & b + \beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i) - y_i + s_i^{up} = 0 \quad \forall i \in \{1..l\} \\
& b + (-1) * (\beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i)) + y_i + s_i^{lo} = 0 \quad \forall i \in \{1..l\} \\
& s_i^{lo}, s_i^{up} \geq 0 \quad \forall i \in \{1..l\} \qquad (6)
\end{aligned}
$$

So far the algorithm minimizes the $L1$ error on the training set. To avoid overfitting effects the smoothing parameter (the kernel width) of the potential function must be adjusted by choosing parameter σ. Another method for complexity regularisation is weight decay, as used in the kernel Adaline. Since the current algorithm is a method of batch-learning early stopping is not possible, however it is possible to force the weight vector to have small components by adding the following constraints to formulation (6):

$$\beta_i < C \quad , \quad \beta_i > -C \quad \forall i \in \{1..l\} \qquad (7)$$

Since the mapping from the row-space of the kernel correlation matrix to the feature space (and back) is fixed weight decay regularisation on the multipliers β automatically implies weight decay regularisation for the weight vector, w, which resides in an infinite dimensional feature space (given the fact that the radial basis potential function defined above is used). Experimental results with the kernel Adaline [6] have demonstrated that this type of regularizer enhances the performance of kernel perceptron based methods for regression.

If different patterns are associated with a different cost (as is usual in many applications, e.g. in the medical domain where type I and type II errors may be of a different cost) a confidence factor c_i; $0 \leq c_i \leq 1$, for each pattern can be introduced. Constraint (7) can be replaced by:

$$\beta_i < C\ c_i \quad , \quad \beta_i > -C\ c_i \quad \forall i \in \{1..l\} \qquad (8)$$

to take the different cost (and risk) of individual patterns in the algorithms into account. Alternatively the cost function of the algorithms can be modified such that the loss of individual patterns is weighted directly by their associated cost, which leads to the following cost function:

$$Y = m_1 \sum_{i=1}^{l} c_i s_i + m_2 \sum_{i=1}^{l} c_i s_i^2 \qquad (9)$$

Another modification of the algorithm is to introduce epsilon-insensitive loss; this loss function has been proposed in the context of support vector machines [22]. In the LPM-R algorithm an insensitive zone for the loss function is obtained by replacing the first two constraints in (6) by:

$$\begin{aligned} b + \beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i) - y_i + s_i^{up} = \epsilon \quad \forall i \in \{1..l\} \\ b + (-1) * (\beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i)) + y_i + s_i^{lo} = \epsilon \quad \forall i \in \{1..l\} \end{aligned} \tag{10}$$

where constant, ϵ, defines the insensitive interval for the loss function.
In the final algorithm, which is called LPM-R, weight decay regularisation as discussed above, and epsilon insensitive loss is available. If desired the insensitive zone of the loss function can be deactivated by choosing $\epsilon = 0$. The type of practical application, with its underlying noise model, defines which loss function is most suitable for a given task.

4.3 QP Machines for Regression (QPM-R)

In the LPM-R algorithm an $L1$ loss function is used, but also the $L2$ loss function can be minimized if quadratic programming is available. The regression method for quadratic and linear loss will be called *QPM-R*. Again two constants m_1 and m_2 can be used (4) to adjust the degree to which the different noise models should take part in the cost function. Methods for model selection can be applied to determine these parameters. Algorithm QPM-R uses the following cost function and constraints:

$$\begin{aligned} min: \quad & m_1 \sum_{i=1}^{l} s_i^{up} + m_1 \sum_{i=1}^{l} s_i^{lo} + m_2 \sum_{i=1}^{l} (s_i^{up})^2 + m_2 \sum_{i=1}^{l} (s_i^{lo})^2 \\ subject\ to: \quad & b + \beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i) - y_i + s_i^{up} = \epsilon \quad \forall i \in \{1..l\} \\ & b + (-1) * (\beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i)) + y_i + s_i^{lo} = \epsilon \quad \forall i \in \{1..l\} \\ & s_i^{lo}, s_i^{up} \geq 0, \beta_i < C * c_i, \ \beta_i > -C * c_i \quad \forall i \in \{1..l\} \end{aligned} \tag{11}$$

Weight decay regularisation and an insensitive zone is available, as described above for the algorithm LPM-R. If parameter m_1 is zero the algorithm implements a kernel Adaline using batch-learning; if parameter m_2 is zero the algorithm minimizes the same cost function as in the LPM-R algorithm.

Note that the formulation for affine decision functions using a bias term, b, in the decision function is different to the one used in the kernel Adaline. In the latter approaches for sequential learning the bias parameter is specified by augmenting the training patterns in the linearisation space, $x \rightarrow [\psi(x)\ 1]^t$, which leads to the constraint $b = \sum_{i=1}^{l} \beta_i$. More information about this can be found in the Appendix.

5 Computer Experiments

The experiments presented in this section are to serve only as a proof of concept of the algorithms discussed above. Two small examples for non-linear one-dimensional curve

fitting have been prepared to demonstrate some properties of the method. For all experiments radial basis function potential kernels and the algorithm LPM-R has been used.

In the first task 28 points generated by the function: $f(x) = sin(0.8x)$ are learned; there is no additional noise in this signal. The diagrams in figure (1) demonstrate that, for too large choices of σ, underfitting effects can be observed. As pointed out in [1], the kernel parameter can serve as a method of capacity control.

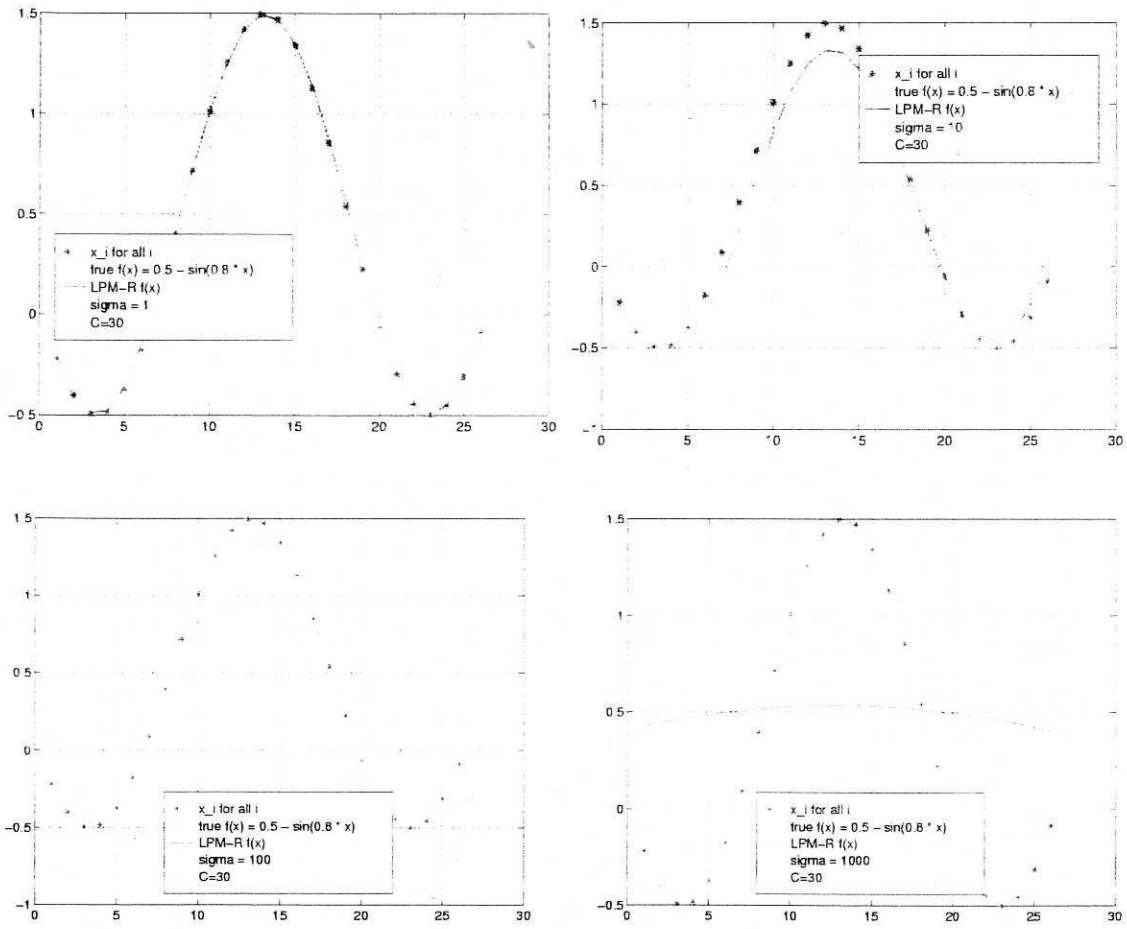

Figure 1 Curve fitting using one dimensional data (x-axis) and their target values and predictions (y-axis). In the four cases illustrated above parameters were chosen in the following way: $C = 30, \epsilon = 0$ in each case, while the value of σ has been changed in each case; $\sigma = 1$ and $\sigma = 10$ for the top left and top right diagrams, and $\sigma = 100$, and $\sigma = 1000$ for the bottom left and bottom right plot. In the diagrams the true signal is plotted by the dotted line, the training samples are marked each by a star, and the predicted (estimated) signal is indicated by a solid line.

In the second task patterns from the same signal used above have been distorted with uniform noise. A pattern generator of the following form has been used: $f(x) = 0.5 -$

$rand() + 0.5 - sin(0.8x)$. The function $rand()$ returns a random number from a uniform distribution in the interval $[0, 1]$.

The diagrams in figure (2) demonstrate the effect of the choice of the kernel parameter. Compare the top left and top right plot in the diagram. While in the left plot ($\sigma = 1$, $C = 1$) overfitting has occured, in the right plot a much smoother function has been learned ($\sigma = 10$, $C = 1$).
Now compare the top right and bottom left diagram in figure (2); in the latter case the value for the parameter C has been altered ($C = 0.01$). It can be observed that this leads to a smoother prediction function.
Finally consider the bottom right plot. Here the value of C has been adjusted back to 1, however the setting for ϵ has been changed ($\epsilon = 0.75$). It can be observed that this also leads to a smoother decision function, compared to the top right plot in the figure.

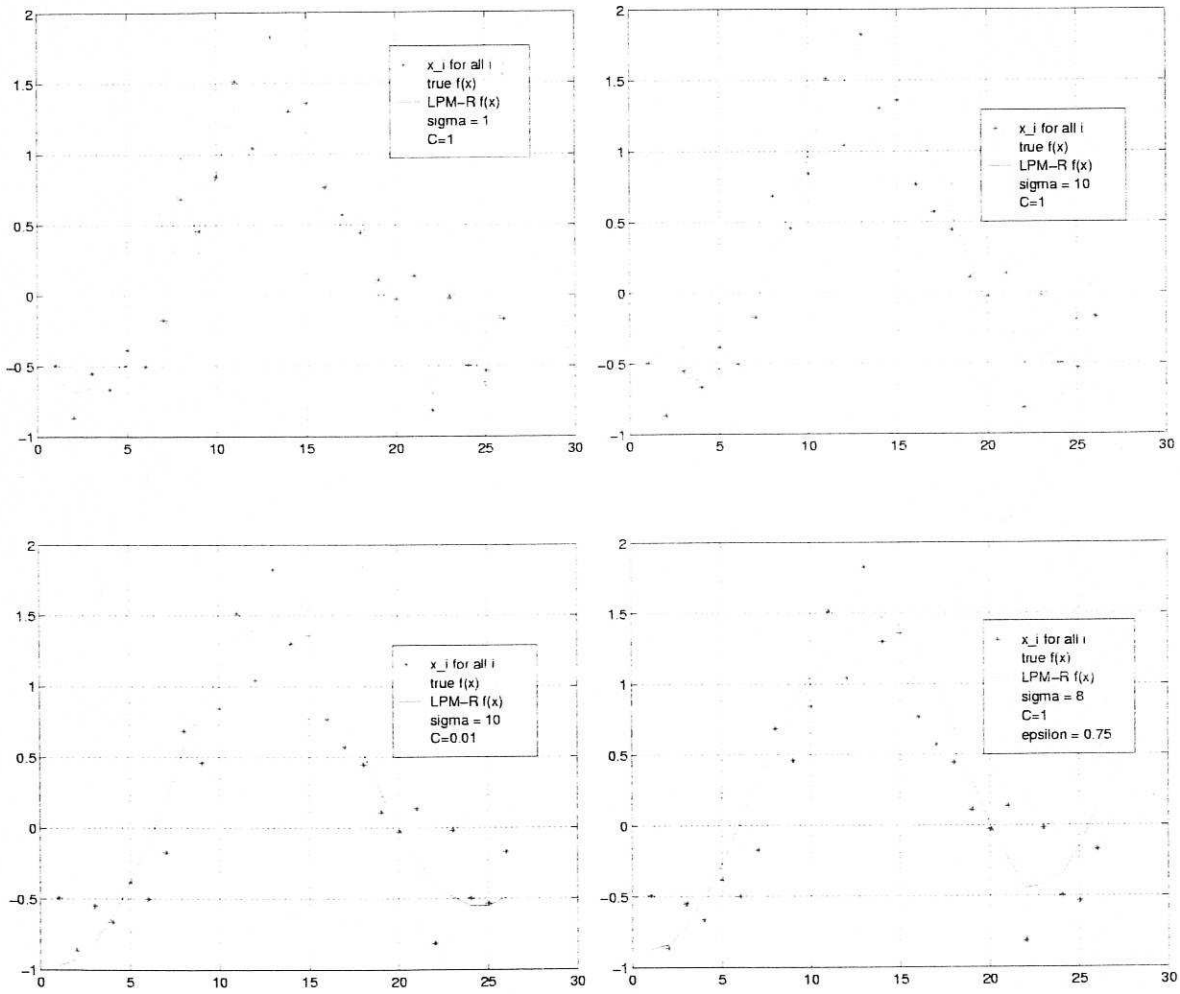

Figure 2 One dimensional function approximation using the LPM-R algorithm. The signal has been distorted with uniform noise. The diagrams show the effects of choosing different values for σ (top left and top right plot), the effect of choosing parameter C (bottom left plot), and the effect of choosing ϵ (bottom right plot) in each case.

6 Conclusion

Algorithms for pattern classification using *linear* decision functions by linear programming have been studied by Minnick [11], Muroga et al. [13], Smith [19], Bennett and Mangasarian [2].
In this work algorithms for pattern classification *and regression* of *non-linear* target-functions are proposed, related work on the ideas developed in this article has been published in [3] [4].
A different method for regression of non-linear target functions, related to support vector machines, been published by Vapnik and Weston [25].
Their method is different to the approach proposed here because:

- ◇ in the algorithms above $L1$ regularisation on the weight vector is used, it is discussed how mixtures of noise models can be realized. In contrast in support vector regression, the $L2$ norm of the weight vector is minimized, subject to one single noise model.

- ◇ in the regression algorithms above the $L1$ norm of the weight vector is minimized by early stopping constraints in the positive *and negative* quadrant.

- ◇ in the algorithms above (and also in the kernel Adaline) only $l + 1$ parameters β are chosen during learning to characterize the prediction function. In contrast in support vector machines $2l + 1$ parameters are used in their learning scheme to parameterize the prediction function.

- ◇ in the algorithms presented above two positive slack variables are used for each training pattern to model the loss of the linear regression in the linearisation space in a novel way. In contrast in Vapnik's and Weston's algorithms for regression one slack variable and two multipliers are used for each slack variable.

This work shows that, in order to minimize the $L1$ loss function in a potential function regression machine, linear programming is sufficient, while for squared loss quadratic programming is required. It is known from support vector learning that constrained quadratic programming may be subject to technical and practical problems. The kernel Adaline presents an alternative to MSE regression, the algorithm is a numerically more robust alternative to support vector machines [6].
It is the author's belief that both regularisation, by the potential kernel and by weight decay, and the convexity of the potential function perceptron's cost function, can explain the good performance of these class of potential algorithms.

Acknowledgments: The financial support from the EPSRC and the University of Sheffield, Dept. of AC&SE is gratefully acknowledged.

7 Appendix

In the algorithms given above it has been discussed that an affine function, both linear and non-linear, can be found by introducing the bias parameter b. In the formulations given above b is determined by the optimization routine, such that the cost functional y is minimized subject to some constraints.

Another common approach for bias is to *augment* the training patterns of a perceptron $x^* \rightarrow [x \;\; 1]^t$, and then to use the linear learning technique. After learning a weight vector $w^* = [w \;\; b]^t$ is obtained where the last component of vector w^* is the bias parameter. In Rosenblatt perceptrons[3] constraint $w = \sum_{i=1}^{l} \beta\phi(x_i)$ is always satisfied; therefore

$$f(x^*) = \sum_{i=1}^{l} \beta\langle\psi(x_i^*), \psi(x^*)\rangle = \sum_{i=1}^{l} \beta K(x_i^*, x^*) =$$
$$\sum_{i=1}^{l} K_{+1}(x_i, x) = \sum_{i=1}^{l} K(x_i, x) + \sum_{i=1}^{l} \beta = \sum_{i=1}^{l} K(x_i, x) + b \qquad (A1)$$

where $K_{+1}(u, v) = [K(u, v) + 1]$. Equation (A1) shows that, if one uses augmented patterns, constraint $b = \sum_{i=1}^{l} \beta$ is automatically satisfied. It is possible to implement this formulation for bias into the algorithms given above, and also into support vector machines. All that is required is to add this constraint into the constraints for the linear or quadratic program which is solved in the algorithms above. For instance in the algorithm LPM-R the resulting linear program, which implements the "augmented" bias, is given by:

$$min: \quad \sum_{i=1}^{l} s_i^{up} + \sum_{i=1}^{l} s_i^{lo}$$

$$subject\ to: \qquad \beta_1 + \beta_2 + ...\beta_l = b$$
$$b + \beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i) - y_i + s_i^{up} = 0 \quad \forall i \in \{1..l\}$$
$$b + (-1) * (\beta_1 h(1,i) + \beta_2 h(2,i) ... + \beta_l h(l,i)) + y_i + s_i^{lo} = 0 \quad \forall i \in \{1..l\}$$
$$s_i^{lo}, s_i^{up} \geq 0 \quad \forall i \in \{1..l\}$$

The "augmented" bias can also be realised in the following way: An algorithm without any bias, which performs in a linearisation space, is used but the potential kernel $K(u, v)$ is replaced by a kernel of the form $K_{+1}(u, v)$. By doing this, constraint $b = \sum_{i=1}^{l} \beta\psi(x)$ will be satisfied. After learning the parameters using the potential algorithm, decision function (A1) is used for prediction.

[3] the constraint is also satisfied in the LPM and QPM algorithms discussed above, as well as in support vector machines

References

[1] Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning, *Automations and Remote Control*, **25**:821-837.

[2] Bennett K., Mangasarian O.L., (1998), Multicategory Discrimination via Linear Pr ogramming, Optimization Methods and Software 3, pp. 27-39

[3] Frieß, T-T., Harrison, R.F., (1998), Linear Programming Support Vector Machines for Pattern Classification and Regresiion Estimation and the SR Algorithm, Research Report 706, *Department of Automatic Control and Systems Engineering*, The University of Sheffield.

[4] Frieß, T-T., Harrison, R.F., (1998), Perceptrons in Kernel Feature Space, Research Report 720, *Department of Automatic Control and Systems Engineering*, The University of Sheffield.

[5] Frieß, T-T., Harrison, R.F., (1998), Support Vector Neural Networks, Research Report 725, *Department of Automatic Control and Systems Engineering*, The University of Sheffield.

[6] Frieß, T-T., Harrison, R.F., (1998), The Kernel Adaline: A new Algoritm for non-linear Signal Processing and Regression Research Report 731, *Department of Automatic Control and Systems Engineering*, The University of Sheffield.

[7] Fukunaga K., (1990), Introduction to Statistical Pattern Recognition, Academic Press, San Diego

[8] Gardner E., Derrida, B. (1988), Optimal storage properties of neural network models. *Journal of Physics.* **21**, 271

[9] Hebb, D. (1949). The Organization of Behaviour, New York, Wiley

[10] Kinzel, W.,(1990) Statistical Mechanics of the Perceptron with Maximal Stability. *Lecture Notes in Physics* (Springer-Verlag) **368**:175-188.

[11] Minnick, (1961). Linear Input Logic, *IEE Transactions on Electronic Computers*, **10**, 6-16

[12] Minsky M.L. & Papert, S.A. (1969) *Perceptrons*, MIT Press: Cambridge.

[13] Muroga E., (1961). Theory of Majority Decision Elements, *Journal of Franklin Institute*, 271, 376-419

[14] Opper, M. (1989). Learning in Neural Networks: Solvable Dynamics. *Europhysics Letters*, **8**:389

[15] Pontil M., Verri A. (1998) Properties of Support Vector Machines, *Neural Computation*, MIT AI Memo 1612.

[16] Rummelhart, D. E., Mc Clelland, J. L. (1984) Parallel Distributed Processing, Cambridge Massachusetts

[17] Rosenblatt F., (1961). Principles of Neurodynamics, Spartan Press, New York

[18] Ripley B.D., (1996). Pattern Recognition and Neural Networks, Cambridge University Press, Cambridge

[19] Smith, (1968). Pattern Classifier Design by Linear Programming, *IEEE Transactions on Computers*, **17**, 367-372

[20] Specht D., (1990). Probabilistic Neural Networks, Neural Networks, 3;109-118

[21] Vanderbei R. J., (1994) LOQO: An interior point code for quadratic programming, TR SOR-94-15, *Statistics and Operations Research*, Priceton University, NJ

[22] Vapnik, V. (1995) The Nature of Statistical Learning Theory, Springer Verlag.

[23] Wahba, G., (1998). Support Vector Machines, Reproducing Kernel Hilbert Spaces and the Randomized GACV, in *Advances in Kernel Methods - Support Vector Learning*, Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola (eds.), MIT Press, Cambridge, USA.

[24] Watkin, T., Ran, A. & Biehl, M. (1993). The Statistical Mechanics of Learning a Rule,

[25] Weston J. Vapnik V.(1998). Density Estimation Using SV Machines, Technical Report CSD-TR-97-23, Dept. of Computer Science, Royal Holloway Univer sity of London

[26] Widrow, G., and Hoff, M. (1960) Adaptive Switching Circuits, Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4, 96-104