Monograph:
Friel, Thilo-Thomas and Harrison, R. (1998) Linear Programming Support Vector Machines for Pattern Classification and Regression Estimation: and The SR Algorithm: Improving Speed and Tightness of VC Bounds in SV Algorithms. Research Report. ACSE Research Report 706 . Department of Automatic Control and Systems Engineering

Linear Programming Support Vector Machines for Pattern Classification and Regression Estimation; and The SR Algorithm: Improving Speed and Tightness of VC Bounds in SV Algorithms

Research Report Number 706
Date: 27.02.1998

Thilo-Thomas Frieß, Rob Harrison

Department of Automatic Control and Systems Engineering
The University of Sheffield
Mappin Street
Sheffield, S1 3JD, U.K.

Abstract: Three novel algorithms are presented; the linear programming (LP) machine for pattern classification, the LP machine for regression estimation, and the set-reduction (SR) algorithm. The LP machine is a learning machine which achieves solutions as good as the SV machine by only maximising a *linear* cost-function (SV machine are based on quadratic programming). The set-reduction algorithm improves the speed and accuracy of LP machines, SV machines, and other related algorithms.
An LP machine's decisions are optimal in the sense that it implements Vapnik's (Vapnik and Chervonenkis 1979, Vapnik 1995) structural risk minimisation (SRM) principle. The LP machine has a number of attractive and interesting properties like a high generalisation ability, fast learning based on linear optimisation, capacity control, and a self organisation property.
The SR algorithm is an efficient method to improve speed in a LP machine, SV machine and related algorithms. VC bounds are known to be loose bounds. The SR algorithm allows to construct *optimal* support vector machines by determining the necessary and sufficient number of support patterns. The algorithm does also give tighter VC bounds (for bounds which are a function of the number of support patterns).

Key Words: Linear Programming, LP Machines, SR Algorithm, Support Vector Machines, Learning Machines, Structural Risk Minimisation, Computational Learning Theory, VC Theory, Supervised Learning, Pattern Classification, Regression Estimation, Non-linear Component Analysis, Linear Components in Mercer-Kernel Space.

Contact author: Thilo Frieß
E-Mail: friess@acse.shef.ac.uk

1. Introduction

Vapnik's support vector (SV) machine is a highly efficient algorithm for pattern classification, regression estimation, and other applications. Compared to conventional methods, like. neural networks trained with backpropagation, the SV machine allows to construct better classification (discriminant-) functions, functions of a low capacity which lead to a high generalisation performance. SV learning is based on finding an optima of a *quadratic* cost function with respect to some constraints.

2. Linear Programming Machines

In an LP machine learning is based on calculating the maximum-margin decision boundary

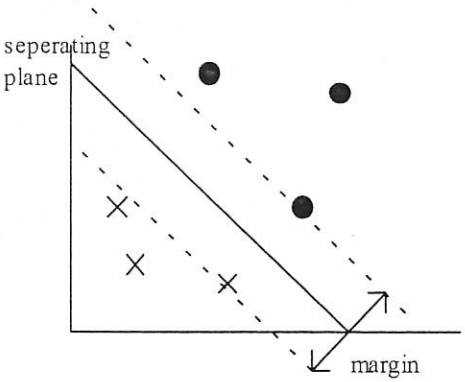


figure 1

An LP machine structured as a neural network

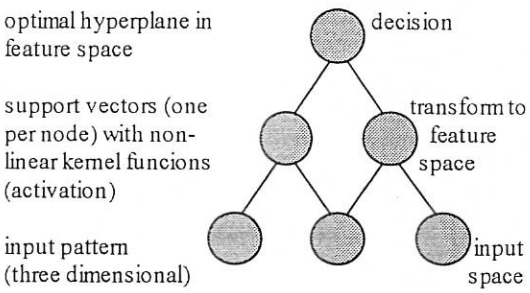


figure 2

Conceptually there are some similarities between the LP machine and Vapnik's support vector (SV) machine (Cortes and Vapnik 1995, Vapnik 1995). Both algorithms construct an optimal-margin decision plane in a very high dimensional feature space (a Hilbert space) using Mercer-kernel functions (Courant and Hilbert 1953). The risk of making a wrong decision is minimised if the margin between the two classes has a maximum value (figure 1). The LP machine (and the SV machine) can find optimal solutions in a convex cost space, therefore the algorithm does not, unlike neural networks, suffer from problem like random initialisation effects, getting stuck in local optima, oscillation, etc..
The crucial difference between the LP machine and SV machine is that the LP machine is based on finding a maximum-margin plane in a *linear* margin space, while the SV machine is based on finding the maximum-margin in a *quadratic* margin space.

2.1 Terminology

For a pattern classification task the training set $\{(x_1,y_1), (x_2,y_2), \ldots (x_l,y_l)\}$ is defined by a set of l training patterns x, $x_i \in R^d \forall i$ and of l labels y, $y_i \in \{1,-1\} \forall i$.
A linear function of the form

$$t(x) = < w, x > + b \qquad (1)$$

is used for classification of patterns (the brackets $<$, $>$ represent a dot product).

The norm of a d-dimensional vector w is indicated by: $\|w\|^1 = \sum_{i=1}^{l} |w_i|$ (L1 norm)

$$\|w\|^2 = \sqrt{\sum_{i=1}^{l} w_i^2} \quad \text{(L2 norm).}$$

If a patterns x_i is correctly classified the following inequality is satisfied:

$$\forall i: \; y_i(< w, x_i > + b) \geq 1 \tag{2}$$

For regression estimation patterns (samples) x and labels y $\{(x_1,y_1), (x_2,y_2), \ldots (x_l,y_l)\}$ form a training set. The labels y_i are real valued: $y_i \in R \; \forall i$. A function as given in (1) is learned such that it represents the functional given by the samples as good as possible (linear regression by minimising a loss function).

2.2 The Margin

In (Cortes and Vapnik 1995) it is shown that the margin between the separating plane (in figure 1) and the two classes is inversely related to the norm of the weight vector w:

$$p(w,b) = \left[\min_{\{x:y=1\}} x \cdot \left(\frac{w}{|w|} \right) + b \right] - \left[\max_{\{x:y=-1\}} x \cdot \left(\frac{w}{|w|} \right) + b \right] = \min_{\{x:y=1\}} \left(\frac{x \cdot w}{|w|} \right) - \max_{\{x:y=-1\}} \left(\frac{x \cdot w}{|w|} \right)$$
$$= \frac{(1-b)-(-1-b)}{\|w_{opt}\|} = \frac{2}{\|w_{opt}\|} \tag{3}$$

The idea of the SV machine (and the maximum margin algorithm (Cortes and Vapnik 1995)) is to minimise a function:

$$\varphi = f(\|w\|^2) \tag{4a}$$
$$\text{subject to } y_i(< w, x_i > + b) \geq 1. \tag{4b}$$

If φ has a minimum value the largest margin has been determined.

The idea of the LP machine is to minimise the functional φ of the form:

$$\varphi = f(\|w\|^1) \tag{5a}$$
$$\text{subject to } y_i(< w, x_i > + b) \geq 1. \tag{5b}$$

This is equivalent to finding a maximum margin plane in the L1 "margin space", while the SV machine does choose the maximum margin plane in a L2 "margin space". To put it into other words, the SV machine is finding a maximum margin plane with respect to the L2 norm of w, while the LP machine is choosing a maximum margin plane with respect to the L1 norm of w.

2.3 LP Perceptron in Input Space for Pattern Classification

The perceptron with linear error function can be implemented using constrained linear programming as follows:

$$\text{minimise:} \quad f(s) = \sum_{i=1}^{d} s_i \tag{6a}$$

$$\text{subject to:} \quad s_i >= 0 \ \text{ and } s_i - \lambda + y_i \, w \, x_i \geq 0 \tag{6b}$$

(as presented in Ripley 1996, Minnick, 1961; Muroga et al, 1961, F.W. Smith 1968, Grinold 1969). The scalar λ was introduced to avoid the trivial solution $w_i = 0 \ \forall i$. To satisfy (2) it is necessary to define: $\lambda = 1$. This assumption removes a degree of freedom in (1) (re-scaling of w and b) such that minimising of the norm of w does result in a large margin classifier.

If the two classes are not linearly separable in a classification task (after optimisation $f(s) > 0$ in (6)) a decision plane has been found which minimises the linear error function (6b).

The constant b can be determined if augmented patterns are used, or if the following equation is optimised:

$$\text{minimise:} \quad f(s,b) = \sum_{i=1}^{d} s_i \tag{7}$$

$$\text{subject to:} \quad s_i >= 0 \ \text{ and } s_i - \lambda + y_i < w, x_i^a > + b \geq 0$$

This is equivalent to augment patterns by: if $i \in \{1,..l\}$ $x_i^a = x_i$, $x_{l+1}^a = 1$ and use (6).

It is known that in any pereptron (Rosenblatt 1969, Minsky and Papert 1969, Vapnik 1995) the weight vector w is a weighted sum of patterns. If we expand for augmented patterns the weight vector ($w = \sum_{i=1}^{l} < \alpha_i, x_i^a >$, $b = \alpha_{l+1}$) equation (1) can be written as:

$$t(x) = b + < w, x > \ =$$

$$b + \left(\sum_{i=1}^{l} < \alpha_i, x_i > \right) x \ = b + \left(\sum_{i=1}^{l} \alpha_i < x_i, x > \right) = b + \left(\sum_{i=1}^{l} \alpha_i \, K(x_i, x) \right) \tag{8}$$

(function $K(\ ,\ )$ does represents a scalar product).

The decision plane t can be represented in two equivalent ways. Expression (8) will be called the explicit form, expression (1) the implicit form.

The algorithm for the perceptron in input space by linear programming can be rewritten in explicit form. Now the multipliers[1] α are determined, while in the implicit form the weights w are found[2]. The LP perceptron in input space is given in explicit form by:

$$\text{minimise:} \quad f(s,b) = \sum_{i=1}^{l} s_i \tag{9}$$

---

[1] Mutipliers are components of vector α.

[2] Finding alphas can be considered as finding weights in a hidden unit space.

subject to: $s_i - \lambda + y_1\alpha_1 < x_i, x_1 > + y_2\alpha_2 < x_i, x_2 > + \ldots, y_l\alpha_l < x_i, x_l > + b \geq 0$
and $s_i >= 0, \quad \alpha_i >= 0$

2.4 LP Machines for Pattern Classification in Input Space

To implement the idea of the LP machine, that is to realise structural risk minimisation by minimising the L1 norm of w, a cost function is optimised which does additionally punish large components of w (in explicit form):

$$\text{minimise:} \quad f(s,\alpha,b) = \sum_{i=1}^{l} s_i + C\sum_{i=1}^{l}\alpha_i \tag{10}$$

subject to: $s_i - \lambda + y_1\alpha_1 < x_i, x_1 > + y_2\alpha_2 < x_i, x_2 > + \ldots, y_l\alpha_l < x_i, x_l > + b \geq 0$
and $s_i >= 0, \quad \alpha_i >= 0$

C denotes a constant for capacity control which has to be chosen. The usage of C is demonstrated in the experimental section below.

2.5 Kernel Functions

There are binary (two valued) functions which represent dot products in Hilbert spaces, these functions are called kernel functions. Using kernels is equivalent to expanding two patterns (e.g x_1, x_2) at first into a very high dimensional feature space $(\gamma(x_1), \gamma(x_2))$, and then calculating a dot product there.

$$K(x_1, x_2) = \gamma(x_1)\gamma(x_2) \tag{11}$$

Any function can be used as a kernel function K if it is positive semidefinite, continuous, symmetric, and there exists a series expansion of the following form (Mercer's series expansion theorem (Courant and Hilbert 1953)):

$$\forall n: n > 1 \quad K^{(n)}(s,t) = \sum_{i=1}^{\infty} \frac{\kappa_i(s)\,\kappa_i(t)}{\upsilon_i^{\,n}} \tag{12}$$

where the υ are eigenvalues and κ are eigenfunctors of the integral operator defined by the integral kernel function (Courant and Hilbert 1953). Further discussions of kernel functions in SV machines and related algorithms are discussed in (Aizerman, Braverman, Rozonoer, 1964).

Examples of kernel functions are:

$$K(u,v) = (u \cdot v' + 1)^{\zeta}, \ \zeta = 1,2, \ldots \tag{13}$$

$$K(u,v) = (u \cdot v')^{\zeta}, \quad \zeta = 1,2, \ldots \tag{14}$$

$$K(u,v) = \exp\left(\frac{-\sum_{i=1}^{l}(u_i - v_i)^2}{\sigma^2}\right) \tag{15}$$

It is known that for some choices of the kernel function K this feature space may be of an *infinite* dimensioality (e.g. for the radial basis function (RBF) kernel (15)).

2.6 LP Machines for Pattern Classification in Feature Space

To find the large margin plane in a very high dimensional Mercer-kernel feature space dot products are replaced by kernel functions. This is the only modification of the algorithm which is necessary to implement the ability to learn non-linear classification functions. Such a non-linear function in input space is equivalent to a linear in a high dimensional feature space.

It is necessary to optimise (explicit form):

$$\text{minimise:} \quad f(s,\alpha,b) = \sum_{i=1}^{l} s_i + C\sum_{i=1}^{l}\alpha_i \tag{16}$$

$$\text{subject to: } s_i - \lambda + y_1\alpha_1 Kx_i, x_1) + y_2\alpha_2 K(x_i, x_2) + \ldots, y_l\alpha_l K(x_i, x_l) + b \geq 0$$
$$\text{and} \quad s_i >= 0, \quad \alpha_i >= 0$$

to find a linear classification function in kernel space. This can be implemented by using standard packages for constrained linear optimisation. Software is available in public archives.

2.7 LP Machines for Regression Estimation in Feature Space

Only a few changes in expression (16) are necessary to implement a regression learning machine. Again kernel functions are used to learn non-liner functions which are linear in some feature space. The LP machine for regression implements Vapnik's epsilon insensitive loss function (Vapnik 1995).

It is necessary to optimise (explicit form):

$$\text{minimise:} \quad f(s,\alpha,b) = \sum_{i=1}^{l} s_i + C\sum_{i=1}^{l}\alpha_i \tag{17}$$

$$\text{subject to:} \quad -s_i + \alpha_1 Kx_i, x_1) + \alpha_2 K(x_i, x_2) + \ldots, \alpha_l K(x_i, x_l) + b - y_i \leq \frac{1}{2}\varepsilon \quad ,$$
$$-1(s_i + \alpha_1 Kx_i, x_1) + \alpha_2 K(x_i, x_2) + \ldots, \alpha_l K(x_i, x_l) + b + y_i) \leq \frac{1}{2}\varepsilon \quad ,$$
$$s_i >= 0, \quad \alpha_i >= 0$$

Two parameters must be chosen, C and ε. The choice of parameter ε defines the size of the epsilon interval. More details about the epsilon insensitive loss function can be found in (Vapnik 1995, Smola 1997).
By choosing a linear or non-linear kernel function (e.g. (13), (14), (15), or a simple scalar product < , >) any linear or non-linear function can be approximated. The capacity of the function is controlled by choosing parameter C. If the parameter epsilon is properly chosen most patterns lie an epsilon interval around function (1), these are patterns with a zero multiplier α. Only a few patterns outside (and on the border) will have a non-zero multiplier α, these are the support patterns. There are

two types of support patterns, bound support patterns which have a distance of exactly epsilon / 2 to the linear regression plane (1) in feature space, and unbound support patterns which have a larger distance to the plane. These patterns are more likely to be outliers.

2.8 LP Machine for Pattern Classification in Prediction Mode

After training a LP machine for pattern classification it can be observed that most components of α are zero. Patterns with a non-zero multiplier α_i are the so called "support vectors" (hidden layer units in figure 2). A decision in a LP machine depends only on the weights of support patterns and on α. The LP machine, can therefore be considered as a compression scheme. In an LP machine a classification for a pattern v is achieved by:

Pattern Classification:
$$LPM(v) = sign\left(b + \left(\sum_{i=1}^{l} y_i \alpha_i \, K(x_i, v)\right)\right) \tag{18}$$

The process of classification in the LP machine can be considered as propagating a pattern through a network where the hidden units consist of support vectors and the hidden activation functions are the kernel functions (see figure 2).

2.9 LP Machine for Regression Estimation in Prediction Mode

The prediction function for algorithm (17) is given by the explicit form of (1):

Regression Estimation:
$$LPM(v) = b + \left(\sum_{i=1}^{l} \alpha_i \, K(x_i, v)\right) \tag{19}$$

Again this function represents a neural-network like structure (figure 2) where hidden nodes are support patterns (the patterns with non-zero multiplier α).

3. Capacity Control

The LP machine used in experiments presented below implements expression (9). Values for the parameter C and kernel parameters are given below. The question how to choose C is discussed in (Cortes and Vapnik 1995).
The parameter C in the LP machine is equivalent to the quantity $1/C$ in a SV machine. The LP machine has the scalar C on the term which represents the norm of w, the SV machine on the term of the error function. This difference is only of a cosmetic nature because only the weighting between the two terms is important to perform SRM.

4. Set-Reduction: Improving the Tightness of VC Bounds and Increasing the Speed of Support Vector Machines and Related Algorithms

In (Cortes 1995) and (Schölkopf 1997) it is explained that the number of support patterns found by a quadratic programming algorithm in a support vector machine is only an upper bound on the number of necessary and sufficient (optimal) support patterns. It is explained that the reason for this effect are linear dependencies between support patterns in feature space.

The set-reduction (SR) algorithm allows to find the optimal number of support patterns in a SV based classifier. The algorithm is applicable for most support vector (kernel-) based algorithms like LP machines, SV machines etc..

The idea of the SR algorithm is to identify linear dependencies in a kernel based data dependent hierarchy (a network structure as given in figure 2). A method to find linear dependencies between patterns (which can be organised as rows in a matrix P) is to calculate the rank of this matrix. The matrix of patterns will be called P

$$P_{ij} = {}^{i}x_{j}$$

(where ${}^{i}x_{j}$ is the j'th component of the i'th pattern in the training set, $i \in 1..l, j \in 1..d$).

Theorem 1: The rank of matrix P expresses the number of necessary and sufficient number of support patterns.
Proof: The proof is trivival. By definition the rank of a matrix gives the number of linear dependent patterns (rows of P). ■

In the theorem above it has been assumed that patterns are accessible, such that a matrix P can be build up. This is not the case if non-linear kernel functions are used (e.g. in a RBF SV machine where patterns in feature space are of infinite dimensionality). However a dot product between patterns in feature space is available by the kernel function.

Theorem 2: The rank of a matrix M ($M_{ij} = <x_i,x_j>$).is equal to the rank of matrix P.
Proof: Given are l patterns x, $i \in 1..l$, then

$$P = \begin{bmatrix} {}^{1}x_1 & {}^{1}x_2 & \dots & {}^{1}x_l \\ {}^{2}x_1 & {}^{2}x_2 & \dots & {}^{2}x_l \\ \dots & & & \dots \\ {}^{l}x_1 & {}^{l}x_2 & \dots & {}^{l}x_l \end{bmatrix} \quad \text{and} \quad M = \begin{bmatrix} < x_1, x_1 > & < x_1, x_2 > & \dots & < x_1, x_l > \\ < x_2, x_1 > & < x_2, x_2 > & \dots & \\ \dots & & & \dots \\ < x_1, x_l > & < x_2, x_l > & \dots & < x_l, x_l > \end{bmatrix}$$

Lets assume a linear dependency between two patterns: $x_i = \mu\ x_j$
Then row i of M is defined by: $< x_i, x_1 \asymp x_i, x_2 >..< x_i, x_l >$
and row j of M is defined as: $< x_j, x_1 \asymp x_j, x_2 >..< x_j, x_l >$
Now row i can be rewritten by using $x_i = \mu\ x_j$:

$$< \mu\ x_j, x_1 \asymp \mu\ x_j, x_2 >..< \mu\ x_j, x_l > =$$
$$\mu < x_j, x_1 > \mu < x_j, x_2 >.. \mu < x_j, x_l > \quad ■$$

Theorem 3: If Mercer Kernel functions are used instead of dot products in theorem 2 then the rank of M gives the number of necessary and sufficient support vectors.
Proof: The same as in theorem 2, but with K(,) instead of < , >. ■

The SR algorithm is an implementation of theorem 3, it computes the rank and linear dependencies of matrix M.
This allows to optimise LP machines, SV machines, and similar algorithms, after the training phase. The linear dependencies between the rows in M are used to update the

multipliers α such that linear dependent patterns vanish (hidden units in figure 2 are removed). Let us image two rows in a matrix M which are linearly dependent , row k = η row g.

In the SR algorithm an update is performed such that:

multiplier(k) = η + multiplier(k).
multiplier(g) = 0;

The algorithm updates multipliers until it is impossible to increase the number of multipliers with a value equal to zero.

In the SV machine it is not necessary to find linear combinations between patterns in feature space (that is between $\gamma(x_k)$ and $\gamma(x_g)$), but to find linear dependencies between the patterns $y_k\gamma(x_k)$ and $y_g\gamma(x_g)$. If each pattern $x_k, k \in 1..l$ is replaced by the product (y_k x_k) (or any other kind of pattern required) before running the RS algorithm this kind of linear dependencies in kernel space can be removed in a similar way.
Bounds from VC theory, e.g. Vapnik's leave one out bound (Vapnik 1995) is a function of the number of support patterns which has been found by an algorithms. VC bound are upper bounds and known to be loose bounds. The RS algorithm provides a way to speedup and optimise SV based learning machines, compute *tighter VC bounds*, and therefore to *improve* the method of *model selection* given by VC theory.

5. Experiments

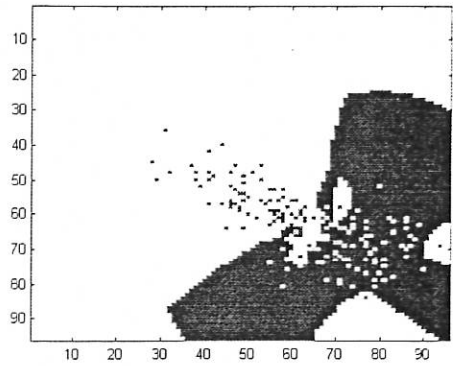
5.1 Empirical and Structural Risk Minimisation with an LP machine using Gaussian Patterns;

Description: The patterns from this benchmark were drawn from two gaussian generators and are overlapping. Covariance, θ, and mean, μ, of the two classes are given by:

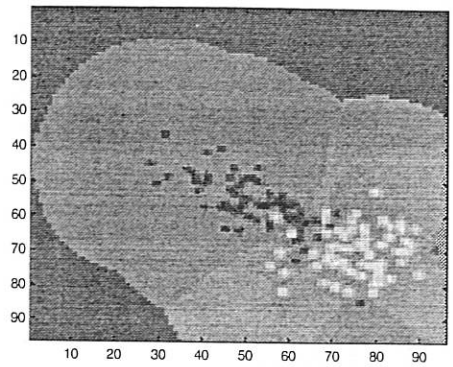
$$\theta_1 = \begin{vmatrix} 0.113 & 0 \\ 0 & 0.116 \end{vmatrix}, \mu_1 = \begin{bmatrix} 1.5 \\ 1.7 \end{bmatrix}, \quad \theta_2 = \begin{vmatrix} 0.1866 & 0.2 \\ 0.2 & 0.43 \end{vmatrix}, \mu_2 = \begin{bmatrix} 0.7 \\ 0.5 \end{bmatrix}.$$

Examples (a) to (h) show classification results for the LP-kernel perceptron, the norm of the weight vector was not minimised (C = 0), *so the margin in feature space is small*. Plots for RBF-LP machines for different values of sigma are given.
Decision space plots are given, potential plots show the relative confidence. The white area in a potential plot shows that the LP machine predicts with high confidence one class, the black area shows high-confidence predictions for the other class. It is known that the distance from a plane in feature space (the potential) can be directly considered as a measure of confidence.
In examples (i) to (l) it is demonstrated how the capacity can be controlled by choosing constant C such that *large margin* classifiers are obtained.

(a) C = 0, sigma = 0.05, RBF kernel, (ERM)

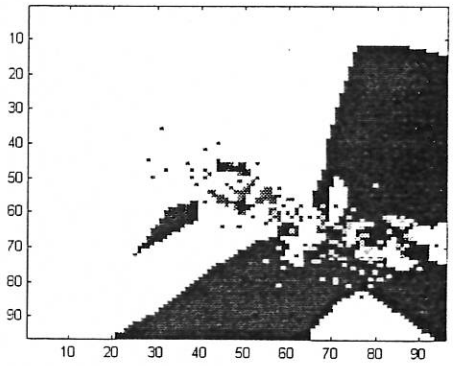


decision space plot

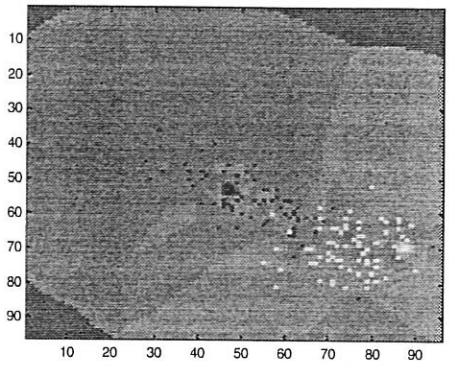


potential plot

(b) C = 0, sigma = 0.075, RBF kernel, (ERM)

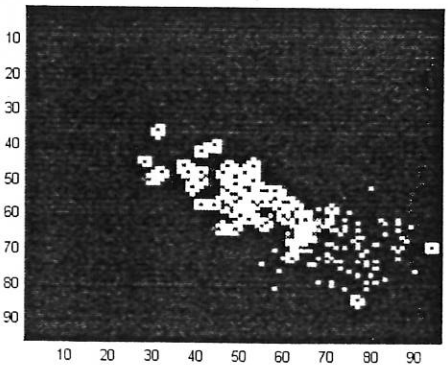


decision space plot

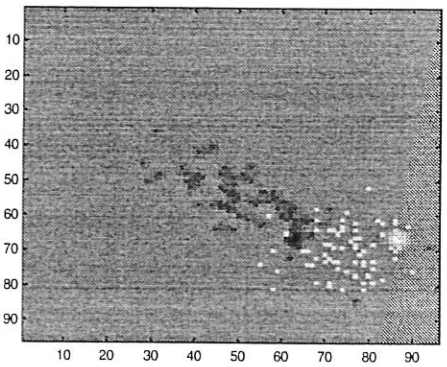


potential plot

(c) C = 0, sigma = 0.1 , RBF kernel, (ERM)

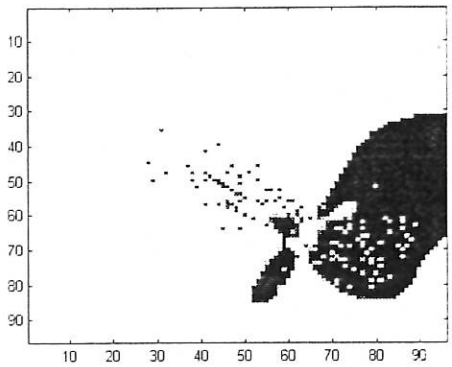


decision space plot

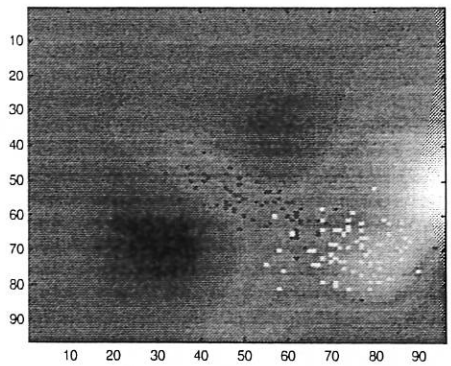


potential plot

(d) C = 0, sigma = 1, RBF kernel, (ERM)

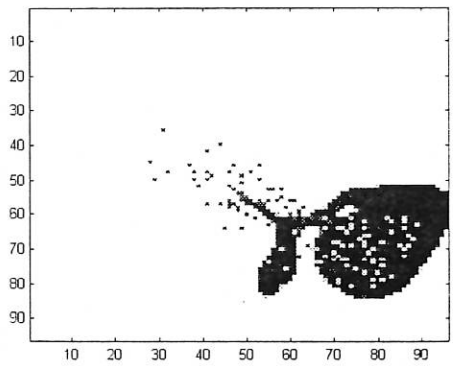


decision space plot

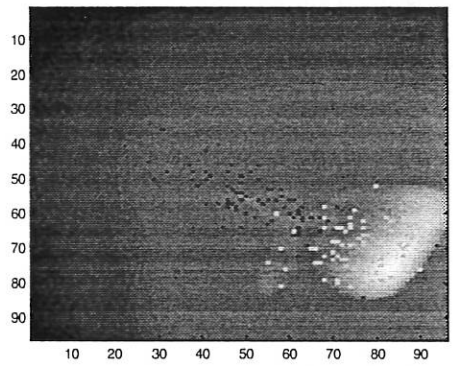


potential plot

(e) C = 0, sigma = 2 , RBF kernel, (ERM)

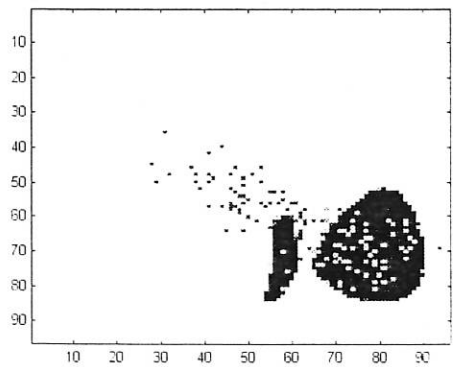


decision space plot

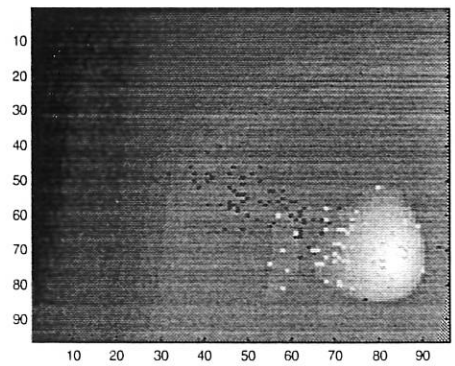


potential plot

(f) C = 0, sigma = 3, RBF kernel, (ERM)

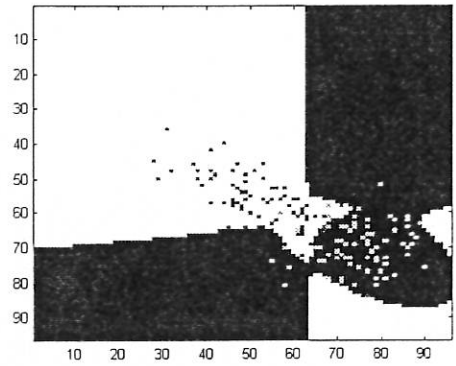


decision space plot

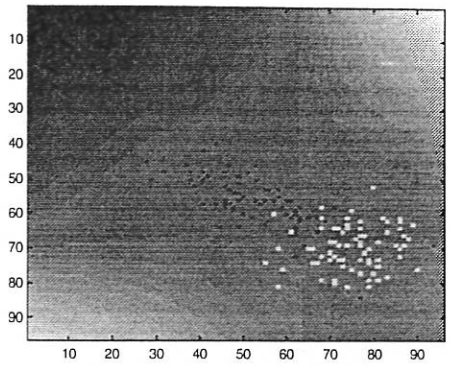


potential plot

(g) C = 0, sigma = 4 , RBF kernel, (ERM)

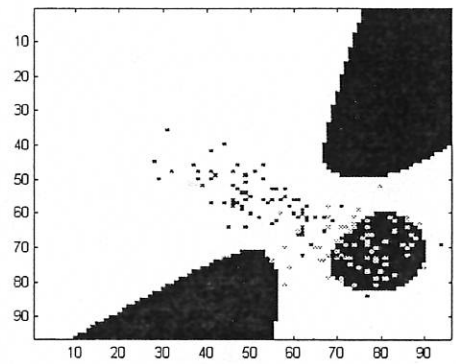


decision space plot

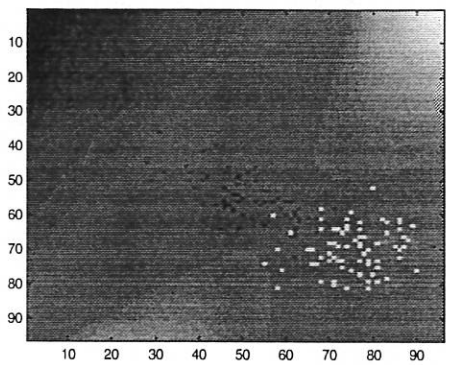


potential plot

(h) C = 0, sigma =5 , RBF kernel, (ERM)

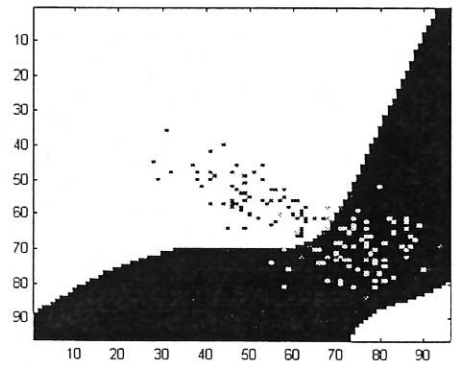


decision space plot

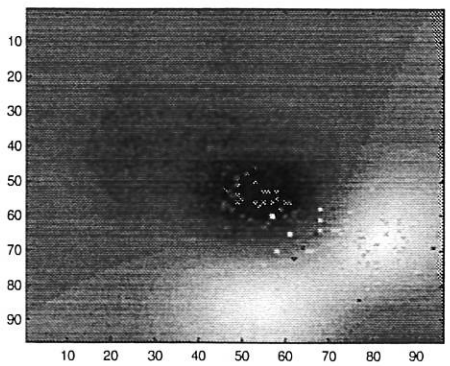


potential plot

(j) C = 0.1, sigma =1 , RBF kernel, (SRM)

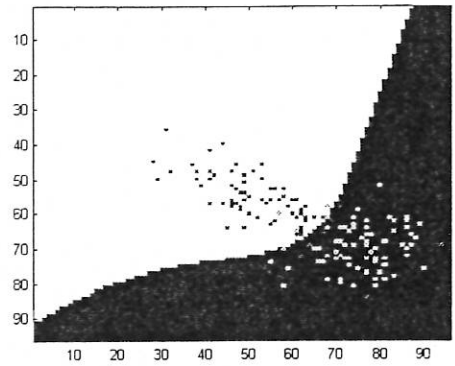


decision space plot

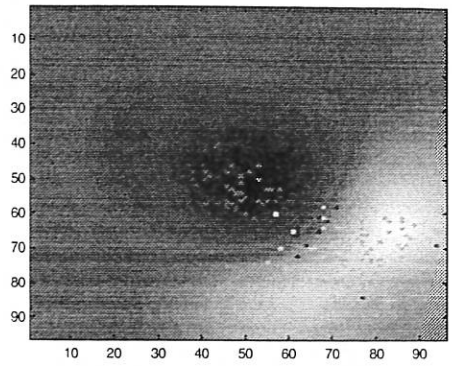


potential plot

(j) C = 1, sigma =1 , RBF kernel, (SRM)

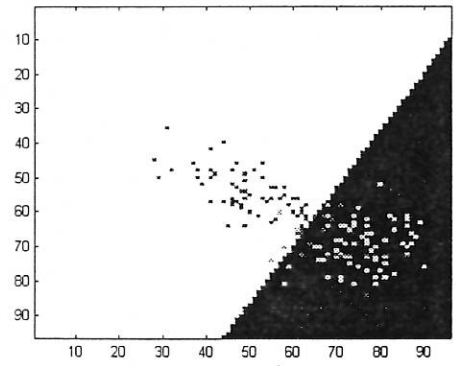


decision space plot

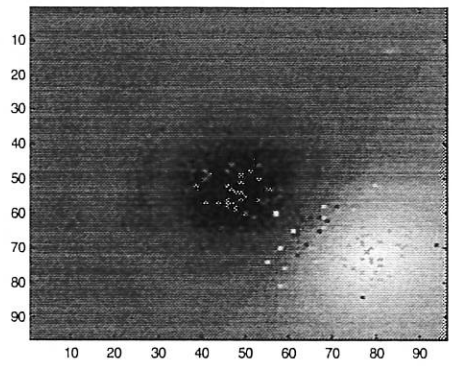


potential plot

(k) C = 10, sigma =1 , RBF kernel, (SRM)



decision space plot



potential plot

Discussion: In cases (a) to (h), the constant C is set to zero. In this cases the LP machine does achieve a high accuracy, most patterns are correctly classified. Note also that the expected generalisation ability is low, the shape of the two gaussian pattern-generators is not represented by the decision (potential-) function. If parameter C is set to zero the LP machine ignores capacity control and is therefore a LP kernel-perceptron which implements the empirical risk minimisation (ERM) principle (Vapnik 1995). To put it into other words, in examples (i) to (h) in kernel space *the margin* between the two classes *is small*.

In cases (i) to (l) the capacity of the LP machine has been controlled by choosing values for C unequal to zero (this implies SRM). The plots illustrate that smooth[1] functions which *generalise well* are learned, the complexity of the constructed learning

[1] Recently it was proven by Beliczynski and Lukianiuk (Beliczynski and Lukianiuk.1998) that it is equivalent to:
- remove high-frequceny components from a (classification-, regression-) function, or to
- minimise the norm of the weight vector w.

machine (the VC dimension) is influenced (regularised) by the choice of C. So in these cases a classifier with a *large margin* (-distribution) has been constructed.

5.2 Sonar: "Real World" Pattern Classification in the Computer Vision Domain by LP Machines

Description: The benchmark stems from the Carnegie Mellon University AI-repository and consists of 208 patterns, each pattern is 60 dimensional. The patterns are split in 104 training-set patterns and 104 test-set patterns.
In the experiments a RBF-kernel LP machine has classified the training patterns, then the test-set has been classified to estimate the generalisation ability empirically.

sigma	C	e_{trn}	e_{tst}	*acc*
0.6	0	0/0	0/12	88.5
0.6	0.025	0/0	4/2	94.2
0.6	0.05	0/0	4/2	94.2
0.6	0.1	0/0	4/2	94.2
0.6	0.3	0/0	4/2	94.2
0.6	1	7/18	11/11	78.8

Table 1: Error rates on the sonar data using different values for C. The constant e_{trn} denotes training error, e_{tst} denotes errors on the test-set. The error is given in a format where the type 1 error is followed by a slash and the type 2 error. The accuracy on the test-set is additionally given in percent by *acc*.

The performance of a RBF SV machine on the same data is given in the Appendix, The best accuracy of the SV machine on this data test-set lies at 93%.

Discussion: The table shows that, if C is set to zero, pure empirical risk minimisation (ERM) (Vapnik 1995) is performed. As expected the performance on the test-set is low. Increasing C does increase the generalisation performance up to a point where many training errors occur.
Compared to results with a neural network given in (Gorman and Sejnowsky 1988) the LP machine has found significantly better classifiers; the best neural networks was able to achieve an accuracy of 90% on the test-set.
The experiment shows that the LP machine's large margin (of the linear classifier in kernel "feature" space) does increase the performance significantly. The large margin is obtained by setting the constant C to a non-zero value.
Compared to the SV machine a slightly higher generalisation ability has been obtained. Further experimental studies will show if the L1-large-margin does generally give a higher generalisation performance, or if this effect occurs only under some special circumstances. In (Schapire, Bartlett 1997) it has been shown in another context that L1-large-margin classifiers (combined by the AdaBoost algorithm) have the ability to make very good predictions.

5.3 Finding the Optimal Number of Support Vectors in a SV machine by the SR algorithm:

Description: A SV machine using kernel function (13) has been trained on the two dimensional gaussian data (as described in section 5.1). The patterns from this training set are highly overlapping. Ten SV machines were constructed, for each model the number of support vectors found by the SV machine (*#SV (SVM)*) is given.

dimensionality of the feature space	*kernel parameter* (ζ)	C	*#SV (SVM)*	*#SV(SVM) >1e-12*	*#SV RS*
3	1	10	150	80	3
9	2	10	150	87	4
27	3	10	150	60	10
81	4	10	150	53	15
243	5	10	150	62	21
729	6	10	150	50	28
2187	7	10	150	40	36
6561	8	10	150	38	42
19683	9	10	150	33	46
50949	10	10	150	31	40

Table 2: The number of support vectors for ten polynomial SV machines. The dimensionality of the feature space is given by expression 3^{ζ}. In each case the constant C was set to 10. Due to the limited accuracy of the quadratic optimiser[2] the SV machine did learn many multipliers with a very small value. For this reason the number of multipliers (that is the number of support patterns) which is greater than 1e-12 is given. Finally the optimal number of support patterns determined by the RS algorithm is given (#SV RS).

Discussion: The number of real support patterns determined by the RS algorithm is much more realistic than the one determined by the SV machine. In all cases the number of real support patters lies below the dimensionality of the feature space. The subspace spanned by the real support patterns is therefore a subspace of the feature space. With an increase in the kernel parameter ζ the dimensionality of this subspace is increased.
Using the number of real support patterns, that is the rank of the hidden unit space, the capacity (roughly speaken the VC dimension) of the learning machine can be expressed more accurate. Vapnik's leave one out bound for the expected generalisation ability is a function of the number of support patterns.

It has been observed that the number support vectors found by a SV machine does not give expressive information about the rank of the hidden unit space and thus of the capacity of the learning machine (Frieß and Harrison 1998).

---

[2] In this experiment the function qp from the MATLAB optimisation toolbox used. The effect has been also been observed with other routines for quadratic programming.

By reducing the number of support vectors, that is the number of hidden units in figure 2, the speed of the prediction mode of a SV algorithm is improved significantly.

6. Conclusion

The LP machine is an elegant and very fast algorithm for pattern recognition and regression estimation. In the preliminary experiments a high performance was obtained, which is at least as good as the one of the SV machine.
A comparison study with many training-sets and other learning machines will be undertaken in the near future to benchmark and analyse the new and promising algorithms.

Both the LP machine and SV machine capture an intrinsic mechanism of regularisation which does not depend on the hypothesis classes defined by the kernel operator; this is the way to measure the norm of w or α. This can be considered as a kind of *prior* in using both learning machines.

The SR algorithm determines the number of necessary and sufficient support patterns in SV machines and related algorithms. This can be considered as *removing redundancies*, *a compression scheme*, or as a kind of *non-linear component analysis*[3]. The speed of the SV machine can be improved by magnitudes using the SR algorithm because the computational complexity of a prediction is directly related to the number of support patterns (propagating a pattern through a network as given in figure 2 where the hidden nodes are support patterns.). *VC bounds are tighter* if used with the optimal number of support patterns. Therefore the quality of the model selection procedure given by VC theory is safer and more reliable. Compared to heuristics which aim to find the optimal number of support vectors the SR algorithm is much faster in finding the true support patterns; it allows to update multipliers directly by linear dependencies. In existing approaches the SV machine must be trained on support patterns found by another SV machine.

Achknowledgemets: The financial support of the Department of Automatic Control and Systems Engineering and the Engineering and Physical Sciences Research Council (EPSRC) is gratefully acknowledged.

References

M.A. Aizerman, E.M. Braverman, L.I. Rozonoer, 1964: "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning", Automation and Remote Control, 25:821-873.

[3] A linear component analysis in a data dependent hierarchy, that is the kernel space defined by the kernel function and given patterns.

N.I. Akhiezer, I.M, Giazman (1993): "Theory of Linear Operators in Hilbert Space", Dover Publications, New York

Bartolmiej Beliczynski, Andrzej Lukianiuk (1998): "Fast Regularized Network for Dynamic System Approximation", Engineering and Intelligent Systems (EIS) 1998, Conference Proceedings, La Laguna, Spain

C.M. Bishop (1995): "Neural Networks for Pattern Recognition", Oxford University Press

R. Courant and D. Hilbert, 1964 : "Methods of Mathematical Physics", Vol. 1, Wiley, New York

Corinna Cortes (1995): "Prediction of Generalisation Ability in Learning Machines", (PhD Thesis), University of Rochester, New York, 1995

Corinna Cortes, Vladimir Vapnik, 1995.: "Support Vector Networks", Machine Learning, 20, 273-279, 1995

Thilo Frieß, Rob Harrison (1998): "Pattern Classification using Support Vector Machines", Engineering and Intelligent Systems (EIS98) , Conference Proceedings, La Laguna, pp. 497-485.

Duda, Hart (1973): "Pattern Classification and Scene Analysis", Wiley

Gorman and Sejnowsky (1992): "The two spiral benchmark", on Carnegie Mellon University Artifical Intellgience Repository (CMUAIR), Kantowitz: "Prime Time Freeware for AI", Calif

D. Goldfarb, A. Idnani (1983): "A Numerically Stable Dual Method For Solving Strictly Convex Quadratic Programs", Mathematical Programming 27, pp.1-33

Gill, Murray (1980): "Practical methods of optimisation", New York, Acad. Press

P.E.Gill, W. Murray, and M.H. Wright (1981), "Practical Optimisation", Academic Press, London

Kolmogorov (1957): "On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition", Doklady Akademiia Nauk SSSR

Minski, Papert (1969): "Perceptrons", Basic Books, Cambridge, MA

Parzen (1961): "On estimation of a probability density function and mode", Annals of Mathematical Statistics, Vol. 33. pp. 1065-1076

Prechelt (1994): "A study of experimental evaluations of neural network learning algorithms,: Current research practice". Technical Report 19/47

B.D. Ripley (1996): "Pattern Recognition and Neural Networks", Cambridge University Press

Frank Rosenblatt (1961): "Principles of Neurodynamics", Spartan Press, New York

Rummelhart, Hinton, Williams (1986): "learning representations by error propagation", in D.E. Rummelhart, McClelland: "Parallel Distributed processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA

Robert Schapire, Peter Bartlett, Yoav Freud, Wee Sun Lee, 1997: "Bossting the Margin: A New Explanation for the Effectiveness of Voting Methods", Machine Learning: Proceedings of the Fourteenth International Conference

B. Schölkopf, K. Sung, C. Burges, F. Girosi, P. Niyogi, T. Poggio, V. Vapnik, 1996 : "Comparing Support Vector Machines with Gaussian kernels to Radial Basis Function Classifiers", A.I. Memo No. 1599, and in Advances in Neural Information Processing Systems (NIPS), Vol9. , available electronically by anonymous ftp on: `publications.ai.mit.edu`

Bernhard Schölkopf (1997): "Support Vector Learning", PhD Dissertation, Berlin

John Shawe-Taylor, Peter L. Bartlett, Robert C. Williamson, Martin Antony, 1996: "Structural Risk Minimisation over Data-Dependent Hierarchies", NeuroCOLT report 8556, available electronically vial anonymous ftp at: `ftp.dcs.rhbnc.ac.uk`

Spellucci, P.(1993): "Numerical experiments with modern methods for large scale QP-problems", Available through http://www.mathematik.th-darmstadt.de/ags/ag8/prof/ spellucci.html.

Vladimir Vapnik (1995): "The Nature of Statistical Learning Theory", Springer, New York

Vladimir Vapnik (1979): "Estimation of dependencise based on empirical data", Springer, New York

Vladimir Vapnik, C. Chervonenkins (german translation) (1979): "Theorie der Zeichenerkennung", Akademie Verlag Berlin

Appendix 1:

Table 2: SV machine performance on Sonar data; RBF kernel:

The constant sigma denotes the kernel parameter, C the capacity constant; *acc trn* does give the training accuracy and *acc tst* the accuracy on the test-set (in percent).

sigma	*C*	*acc trn*	*acc tst*	*margin*
0.2	1e+010	100	50	0.1968
0.2	1e+008	100	50	0.1968
0.2	1e+006	100	50	0.1968
0.2	1e+004	100	50	0.1969
0.2	1000	100	50	0.1969
0.2	500	100	50	0.197
0.2	250	100	50	0.1972
0.2	100	100	50	0.1978
0.2	10	100	50	0.2064
0.2	1	100	47.12	0.2781
0.4	1e+010	100	77.88	0.2016
0.4	1e+008	100	77.88	0.2016
0.4	1e+006	100	77.88	0.2016
0.4	1e+004	100	77.88	0.2016
0.4	1000	100	77.88	0.2017
0.4	500	100	77.88	0.2018
0.4	250	100	77.88	0.202
0.4	100	100	77.88	0.2026
0.4	10	100	76.92	0.2111
0.4	1	100	71.15	0.2822
0.6	1e+010	100	87.5	0.2047
0.6	1e+008	100	87.5	0.2047
0.6	1e+006	100	87.5	0.2047
0.6	1e+004	100	87.5	0.2047
0.6	1000	100	87.5	0.2048
0.6	500	100	87.5	0.2049
0.6	250	100	87.5	0.2051
0.6	100	100	87.5	0.2057
0.6	10	100	87.5	0.2149
0.6	1	100	82.69	0.2878
0.8	1e+010	100	92.31	0.1965
0.8	1e+008	100	92.31	0.1965
0.8	1e+006	100	92.31	0.1965
0.8	1e+004	100	92.31	0.1965
0.8	1000	100	92.31	0.1967
0.8	500	100	92.31	0.1968
0.8	250	100	92.31	0.1971
0.8	100	100	92.31	0.1979
0.8	10	100	92.31	0.2092
0.8	1	99.04	83.65	0.289
1	1e+010	100	93.27	0.1781
1	1e+008	100	93.27	0.1781
1	1e+006	100	93.27	0.1781
1	1e+004	100	93.27	0.1782
1	1000	100	93.27	0.1783
1	500	100	93.27	0.1785
1	250	100	93.27	0.1789
1	100	100	92.31	0.18
1	10	100	91.35	0.1947
1	1	99.04	86.54	0.2843
1.2	1e+010	100	93.27	0.1565

1.2	1e+008	100	93.27	0.1565
1.2	1e+006	100	93.27	0.1565
1.2	1e+004	100	93.27	0.1565
1.2	1000	100	93.27	0.1568
1.2	500	100	93.27	0.157
1.2	250	100	93.27	0.1575
1.2	100	100	93.27	0.159
1.2	10	100	91.35	0.1773
1.2	1	98.08	89.42	0.2763
1.4	1e+010	100	93.27	0.1358
1.4	1e+008	100	93.27	0.1358
1.4	1e+006	100	93.27	0.1358
1.4	1e+004	100	93.27	0.1358
1.4	1000	100	93.27	0.1361
1.4	500	100	93.27	0.1364
1.4	250	100	93.27	0.1371
1.4	100	100	93.27	0.139
1.4	10	100	91.35	0.1609
1.4	1	98.08	88.46	0.2678
1.6	1e+010	100	92.31	0.1174
1.6	1e+008	100	92.31	0.1174
1.6	1e+006	100	92.31	0.1174
1.6	1e+004	100	92.31	0.1175
1.6	1000	100	92.31	0.1179
1.6	500	100	92.31	0.1183
1.6	250	100	92.31	0.1191
1.6	100	100	92.31	0.1215
1.6	10	100	90.38	0.1467
1.6	1	95.19	89.42	0.2597