DETERMINATION OF HYPER-PARAMETERS FOR KERNEL BASED CLASSIFICATION AND REGRESSION

By Andreas Christmann[1], Karsten Luebke[2], Marcos Marin-Galiano[3], Stefan Rüping[4]

[1,2,3] University of Dortmund, Department of Statistics
[4] University of Dortmund, Department of Computer Science

We investigate methods to determine appropriate choices of the hyper-parameters for kernel based methods. Support vector classification, kernel logistic regression and support vector regression are considered. Grid search, Nelder-Mead algorithm and pattern search algorithm are used.

1. Introduction

The optimization of the hyper-parameters of a statistical procedure or machine learning task is a crucial step for obtaining a minimal error. Unfortunately, the optimization of hyper-parameters usually requires many runs of the procedure and hence is very costly. A more detailed knowledge of the dependency of the performance of a procedure on its hyper-parameters can help to speed up this process.

In this paper, we investigate the case of kernel-based classifiers and regression estimators which belong to the class of convex risk minimization methods from machine learning. In an empirical investigation, the response surfaces of nonlinear support vector machines and kernel logistic regression are analyzed and the performance of several algorithms for determining hyper-parameters is investigated.

The rest of the paper is organized as follows: Section 2 briefly outlines kernel based classification and regression methods. Section 3 gives details on several methods for optimizing the hyper-parameters of statistical procedures. Then, some numerical examples are presented in Section 4. Section 5 contains a discussion. Finally, all figures are given in the appendix.

2. SVM, KLR, and SVR

In statistical machine learning the major goal is the estimation of a functional relationship $y_i \approx f(x_i)$ between an outcome $y_i \in \mathbb{R}$ and a vector of explanatory variables $x_i = (x_{i,1}, \ldots, x_{i,d})' \in \mathbb{R}^d$. The function f is unknown. The estimate of f is used to get predictions of an unobserved outcome y_{new} based on an observed value x_{new}. One needs the implicit assumption that the relationship between x_{new} and y_{new} is – at least almost – the same as in the training data set (x_i, y_i), $i = 1, \ldots, n$. Otherwise, it is useless to extract knowledge on f from the training data set. The classical assumption in machine learning is

1. Supported in part by the Deutsche Forschungsgemeinschaft (SFB-475) and DoMuS
2. Supported in part by the Deutsche Forschungsgemeinschaft (SFB-475)
3. Supported in part by the Deutsche Forschungsgemeinschaft (SFB-475) and DoMuS
4. Supported in part by the Deutsche Forschungsgemeinschaft (SFB-475)
5. *AMS 2000 subject classification. Primary 62G08, 62G35; secondary 68Q32, 62G20.*
6. *Keywords and Phrases.* convex risk minimization, kernel logistic regression, statistical machine learning, support vector machine, support vector regression.

that the training data (x_i, y_i), $i = 1, \ldots, n$, are independent and identically generated from an underlying unknown distribution P for a pair of random variables (X_i, Y_i). In practical applications the training data set is often quite large, high-dimensional and complex. The quality of the predictor $f(x_i)$ is measured by some loss function $L : Y \times \mathbb{R} \to [0, \infty)$ via $L(y_i, f(x_i))$. The goal is to find a predictor $f_{\mathrm{P}}(x_i)$ that minimizes the expected loss, *i.e.*

$$\mathbb{E}_{\mathrm{P}}\, L(Y, f_{\mathrm{P}}(X)) = \min_f \mathbb{E}_{\mathrm{P}}\, L(Y, f(X)), \tag{1}$$

where $\mathbb{E}_{\mathrm{P}}\, L(Y, f(X)) = \int L(y, f(x)) d\mathrm{P}(x, y)$ denotes the expectation of L with respect to the distribution P.

In this paper we are interested in binary classification, where $y_i \in Y := \{-1, +1\}$. The straightforward prediction rule is: predict $y_i = +1$ if $f(x_i) \geq 0$, and predict $y_i = -1$ otherwise. The loss function for the classification error is given by

$$I(y_i, f(x_i)) = \mathbb{I}(y_i f(x_i) < 0) + \mathbb{I}(f(x_i) = 0)\mathbb{I}(y_i = -1),$$

where $\mathbb{I}$ denotes the indicator function. Inspired by the law of large numbers one might estimate f_{P} with the minimizer f_{emp} of the empirical classification error, that is

$$f_{emp} = \arg\min_f \frac{1}{n} \sum_{i=1}^{n} I(y_i, f(x_i)) \ . \tag{2}$$

To avoid over-fitting one usually has to restrict the class of functions f considered in (2). Unfortunately, the classification function I is not convex and the minimization of (2) is often NP-hard, *cf.* Höffgen *et al.* (1995). To circumvent this problem, one can replace the classification error function $I(y_i, f(x_i))$ in (2) by a convex upper bound $L : Y \times \mathbb{R} \to \mathbb{R}$ *cf.* Vapnik (1998) and Schölkopf and Smola (2002). Furthermore, using Reproducing Kernel Hilbert Spaces (RKHS) and an additional regularization term has some algorithmic advantages and reduces the danger of over-fitting. These modifications lead to the following empirical regularized risks:

$$\hat{f}_{n,\lambda} = \arg\min_{f \in H} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \|f\|_H^2, \tag{3}$$

$$(\hat{f}_{n,\lambda}, \hat{b}_{n,\lambda}) = \arg\min_{f \in H,\, b \in \mathbb{R}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i) + b) + \lambda \|f\|_H^2, \tag{4}$$

where $\lambda > 0$ is a small regularization parameter, H is a RKHS of a kernel $k : X \times X \to \mathbb{R}$, and $b \in \mathbb{R}$ is called *offset*. The decision functions are $\mathrm{sign}(\hat{f}_{n,\lambda})$ or $\mathrm{sign}(\hat{f}_{n,\lambda} + \hat{b}_{n,\lambda})$. Note that in practice usually (4) is solved while many theoretical papers deal with (3) since the unregularized offset b often causes technical difficulties in the analysis.

In practice the dual problems of (3) and (4) are solved. In these problems the RKHS does not occur explicitly, instead the corresponding kernel is involved. The choice of the kernel k enables the above methods to efficiently estimate not only linear, but also non-linear functions. Of special importance is the Gaussian radial basis function (RBF) kernel

$$k(x, x') = \exp(-\gamma \|x - x'\|^2), \quad \gamma > 0, \tag{5}$$

which is a universal kernel on every compact subset of $\mathbb{R}^d$ in the sense of Steinwart (2001). Furthermore, this kernel is a bounded kernel, as $|k(x, x')| \leq 1$ for all $x, x' \in \mathbb{R}^d$. Polynomial kernels $k(x, x') = (c + \langle x, x' \rangle)^m$ are also popular in practice, but are unbounded for $m \geq 1$ and $X = \mathbb{R}^d$. Here we restrict attention to the Gaussian RBF-kernel and omit polynomial kernels because the Gaussian RBF-kernel in combination with a loss function with bounded first derivative offers good robustness properties which is not true for unbounded kernels such as polynomial ones, see Christmann and Steinwart (2004, 2005).

In this paper we consider three popular loss functions. Popular loss functions for binary classification problems depend on y and f via $v = yf(x)$ or $t = y(f(x) + b)$, where $b \in \mathbb{R}$ is an intercept term. The support vector machine (SVM) uses $L_{SVM}(y, t) = \max(1 - t, 0)$, $t \in \mathbb{R}$, i.e. this loss function penalizes points linearly if $t < 1$ and is constant and equal to zero for $t \geq 1$. Kernel logistic regression (KLR) uses the twice continuously differentiable loss function $L_{KLR}(y, t) = \ln(1 + \exp(-t))$, $t \in \mathbb{R}$. Bartlett and Tewari (2004) proved that KLR can be used to estimate all conditional probabilities $P(Y = 1|X = x)$, $x \in \mathbb{R}^d$, which is *not* possible with SVM. For regression models, i.e. $y \in \mathbb{R}$, we consider Vapnik's $\varepsilon-$insensitive loss function given by $L_\varepsilon(y, t) = \max\{|y - t| - \varepsilon, 0\}$, $t \in \mathbb{R}$, for some value $\varepsilon > 0$.

Problems (3) and (4) can be interpreted as a stochastic approximation of the minimization of the theoretical regularized risk given in (6) or (7), respectively (Vapnik, 1998, Zhang, 2004, Steinwart, 2005):

$$f_{\mathrm{P},\lambda} = \arg\min_{f \in H} \mathbb{E}_{\mathrm{P}}\, L(Y, f(X)) + \lambda \|f\|_H^2 \,, \tag{6}$$

$$(f_{\mathrm{P},\lambda}, b_{\mathrm{P},\lambda}) = \arg\min_{f \in H,\, b \in \mathbb{R}} \mathbb{E}_{\mathrm{P}}\, L(Y, f(X) + b) + \lambda \|f\|_H^2 \,. \tag{7}$$

3. Determination of hyper-parameters

In this section we present different optimization methods which are used to find optimal hyper-parameters (C, γ) or (C, γ, ε) for kernel based methods. A vast amount of strategies and algorithms for function optimization has been proposed in the past. The target function for evaluating the hyper-parameters at is either

- (estimated) misclassification rate (i.e. $\hat{y} \neq y$) for classification problems
- (estimated) L_2 error (i.e. $(\hat{y} - y)^2$) for regression problems.

Note that the estimate function $\hat{f} \in H$ and the predictions $\hat{y} = \hat{f}(x)$ depend on the hyper-parameters of the kernel based methods. As the derivative of both target functions on the hyper-parameters is not known the optimal parameters have to be found numerically. We compare five methods used to find the set of (almost) optimal parameters:

- random search
- grid search
- Nelder-Mead
- Cherkassky/Ma (for regression)

- pattern search

All these methods do not need any derivatives of the objective function. While the first three are general-purpose optimization methods the last two were proposed explicitly for optimization of hyper-parameters in kernel based methods.

3.1 Random search

The simplest version of random search is implemented: a random point of the parameter space is chosen and the value of the objective function is evaluated. This is repeated N times and the best point is taken as the result. Of course, the result of this approach heavily depends on the way the random points are chosen and how many iterations are used. The trial points for which the objective function is evaluated are drawn from a multivariate normal distribution with the center of the search space as the mean.

3.2 Grid search

Optimization by grid search is also very simple: after the search space is specified each search dimension is split into n_i parts. Often these splits are equidistant. The intersections of the splits – which form a (multi-)dimensional grid – are the trial points which are evaluated.

3.3 Nelder-Mead simplex search

The Nelder-Mead (or downhill simplex) algorithm proposed by Nelder and Mead (1965) constructs a simplex of $m+1$ points for an m dimensional optimization problem. In our case we have $m = 2$ for the classification case and $m = 3$ for the regression case. The functional values are calculated for the vertices of the simplex and the worst point is reflected through the opposite side of the simplex. If this trial point is best, the new simplex is expanded further out. If the function value is worse, than the second worst point of the simplex is contracted. If no improvement at all is found, the simplex is shrunken towards the best point. This procedure terminates if the differences in the function values between the best and worst points are negligible.

3.4 Parameter choices by Cherkassky and Ma

One way to determine the hyper-parameters for support vector regression directly from the data has recently been proposed by Cherkassy and Ma (2004). Their proposal is based on both theoretical considerations and empirical results. The following suggestions for the regularization parameter C, the width of the non-penalized tube ϵ and the bandwidth parameter γ of the RBF-kernel are suited for the case that all input variables are scaled to the interval $[0, 1]$. They can be easily adjusted to non-scaled data.

Regarding the regularization parameter C, Cherkassky and Ma agree with the findings of Mattera and Haykin (1999) that C should be chosen according to the range of the values of the response variable in the training data. Since the range is not robust against outliers, Cherkassy and Ma (2004) use the following expression to determine C:

$$C = \max\left\{|\bar{y} - 3\sigma_y|, |\bar{y} + 3\sigma_y|\right\}. \tag{8}$$

In this equation, $\bar{y}$ and σ_y denote the mean and the standard deviation of the response variable y in the training data, respectively.

The width of the ϵ tube, in which all errors are not penalized, can be chosen according to the sample size and the noise of the data, denoted by σ. Cherkassky and Ma propose the value

$$\epsilon = 3\sigma\sqrt{\frac{\ln n}{n}}. \tag{9}$$

In practice, σ will be unknown and must be estimated. To accomplish this, Cherkassy and Ma (2004) propose a k-nearest neighbor regression with k set to a low value in the range of $3-7$. The noise will then be estimated using the residuals of this regression.

As Cherkassy and Ma (2004) base all their considerations on the RBF-kernel, the kernel parameter γ must be determined in addition. It is chosen in dependence of the number of input variables of the regression problem, its dimension d, as

$$\gamma = \frac{1}{2(\sqrt[d]{c})^2}, \tag{10}$$

where c is a constant between 0.2 and 0.5, for which good SVM performance can be achieved.

The method of Cherkassy and Ma (2004) has the advantage that the parameters can be accessed directly from the data. The authors give many numerical examples which show the power of their approach when used on artificial data. It is not known, however, if the heuristic choice of (C, ϵ, γ) always gives meaningful parameter combinations when applied to real-life data.

3.5 Pattern search

Momma and Bennett (2002) proposed the pattern search algorithm as a directed search method to determine the hyper-parameters for support vector machines. It examines points in the parameter space which are arranged in a pattern around the actual optimal point. The pattern depends on the number of parameters in the SVM. For classification SVM with RBF-kernel using the logarithms of the parameter value, the pattern with four elements

$$P = \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}, \tag{11}$$

can be used to construct a pattern in the parameter space (C, γ). For the three hyper-parameters C, ϵ and γ in an SVR, this pattern can be expanded to

$$P^* = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}. \tag{12}$$

The columns of P and P^* describe the change applied to a given parameter vector $q = (C, \gamma)^T$ or $q^* = (C, \epsilon, \gamma)^T$. This means that only one parameter is changed at a time. The pattern search algorithm itself works as follows:

i) Initialization: Choose a start pattern center $q^{(0)}$ and compute the value of the function to be minimized $f(q^{(0)})$. Furthermore, choose a factor $\Delta^{(0)}$ which denotes the expansion of the pattern and τ, the expansion at which the algorithm should stop.

ii) Optimization step:

- Compute $q_i^{(k+1)} = q^{(k)} + \Delta^{(k)} \cdot p_i$ for all columns p_i of P and the corresponding $f(q_i^{(k+1)})$.
- If $\min f(q_i^{(k+1)}) < f(q^{(k)})$, set $q^{(k+1)} = \arg\min f(q_i^{(k+1)})$ and $\Delta^{(k+1)} = \Delta^{(k)}$. Else, set $q^{(k+1)} = q^{(k)}$, $\Delta^{(k+1)} = \Delta^{(k)}/2$ and proceed to the stopping rule.

iii) Stopping rule: If $\Delta^{(k)} < \tau$, stop the algorithm. Otherwise, perform another optimization step.

The algorithm searches the parameter space pattern-wise and memorizes the best hyper-parameter combination it comes across. If the center of the pattern is optimal, the pattern will be made smaller which corresponds to a finer grid search. If the pattern is small enough, the algorithm will stop.

Principally the pattern search works similar to a grid search, but it only makes calculations for a subset of the grid points. By choosing the direction of the steepest descent among pattern points it will omit a lot of grid points which may lead to unsatisfactory results when their respective parameter combinations are applied to the data. Furthermore, a more exhaustive search will be automatically done in the region of interest. Whereas this should lead to computational savings, there are also disadvantages. First, the user must choose a starting point for the pattern search. This can be done randomly or by heuristics like in Section 3.4. Another important drawback could be that the algorithm is in danger to run into a local minimum.

4. Numerical examples

In this section we give some numerical results to compare the methods for hyper-parameter determination described in the previous section.

4.1 Criteria

The main criterion in many applications is the accuracy. In a classification problem the accuracy is measured by the misclassification rate, in a regression problem by the L_2 error.

Cross validation is used to estimate the accuracy. The data set is divided into ℓ disjoint sets and each set is once used as the test set whereas the other $\ell - 1$ sets are put together to form the training set. The average accuracy on the ℓ test sets is used as the estimator for the accuracy of the method with the given hyper-parameters. In our study we set $\ell = 10$.

A second criterion for optimization methods is the number of evaluations needed. Here we counted the number of combinations of the hyper-parameters which are tested in order to find the best combination of (C, γ) or of (C, γ, ε)

4.2 Description of data sets

We use several benchmark data sets, but we also consider some additional data sets.

4.3 Data sets for classification

17 data sets of varying size and dimension were used in this investigation. 13 of the data sets come from the UCI Machine Learning Repository, see Blake and Merz (1998) Additionally, 4 other real-word data sets were used. These data sets will be described in this section. A short overview over the data sets can be found in the following table.

Name	Size (n)	Dimension (d)
balance	576	4
breast	683	9
covtype	4951	48
dermatology	184	33
diabetes	768	8
digits	776	64
ionosphere	351	34
iris	150	4
liver	345	6
mushroom	8124	126
promoters	106	228
voting	435	16
wine	178	13
b3	157	13
directmailing	5626	81
insurance	10000	135
medicine	6610	18

B3

This West German Business Cycles data (1955-1994) is analyzed by the project B3 of the SFB475 (Collaborative Research Centre "Reduction of Complexity for Multivariate Data Structures"), supported by the Deutsche Forschungsgemeinschaft. It consists of 13 economic variables with 157 quarterly observations from 1955/4 to 1994/4 of the German business cycle, see Heilemann and Münch (1996). The German business cycle is classified in a four phase scheme: upswing, upper turning point, downswing and lower turning point. There were 6 complete cycles in the time period. The four phases are split into six one-against-one test situations so it is possible to investigate how the optimal hyper-parameters vary within a data set.

Directmailing

This data set contains the demographic description of households which did or did not answer to a direct mailing campaign of a large company.

Insurance

This data set contains simulated data with a similar structure than a huge data set from 15 German insurance companies from around 4.6 million customers described in Christmann (2004).

MEDICINE

This data set is based upon an application in intensive care medicine described in Morik *et al.* (2000). The classification task is to predict whether the dosage of a certain drug should be increased on the basis of vital signs of intensive care patients.

Data sets for regression

In addition to the numerous classification data sets, we also examined two regression data sets, a real life one and an artificial one. The real life data set is the well-known Boston Housing data set from the UCI Machine Learning Repository, see Blake and Merz (1998). It contains the data from 506 suburbs of Boston. The task is to estimate the median value of owner-occupied homes depending of 13 input variables like e.g. crime rate, highway accessibility, number of teachers and air pollution.

The artificial data set is one of the problems described in Friedman (1991). For this regression problem, we created 400 realizations of the random variables $x_1 - x_5$ which are all uniformly distributed on [0,1]. The target variable y depends on $x_1 - x_5$ through the functional $y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e$, with $e \sim N(0,1)$.

4.4 Numerical results

For safety reasons we used a very fine grid (25×25 grid points) which is much finer than usually used in practice. The grid values used in our research were $\gamma = e^{c_i}$ and $C = 10^{c_i}$. The coefficients c_i were 25 equidistant points spreaded over the interval $[-5, 5]$ (i.e. $-5\ ,\ -5+1\cdot 10/24\ ,\ \ldots\ ,\ -5+23\cdot 10/24\ ,\ 5$). Therefore the performance of the grid search should be close to the optimum and may be better than observed in real applications.

In order to plot response surfaces for SVR we constantly set $\epsilon = 0$ so that the optimization is still in two dimensions. The starting point for pattern search was $(C, \gamma) = (1, 1)$ and the starting simplex for the Nelder-Mead algorithm consisted of the points $(C_1, \gamma_1) = (1, 1)$, $(C_2, \gamma_2) = (10, 1)$ and $(C_3, \gamma_3) = (1, e)$. For the stopping rules we used $\tau = 10^{-3}$ for pattern search and a relative change of $1.5 \cdot 10^{-8}$ for the Nelder-Mead-algorithm.

The computational effort was immense due to the fine grid. Overall it took approximately three months (!) of pure computation time on an Opteron 248 Processor with 2.2 GHz.

The main results are:

- No method performs best everytime. Based on our research, pattern search offers a good compromise between accuracy and computational effort if an extensive grid search can not be done.
- Sometimes the grid search could be beaten by other methods slightly, although we used a rather fine grid.
- Often one can approximate the result of the grid search by other methods and simultaneously reduce the computation time by a large factor (say around 5 to 30).
- Nelder-Mead sometimes performs very good w.r.t. both criteria, but sometimes it needs too much computation time. Fine-tuning of the parameters for inflation or deflation of the Nelder-Mead algorithm may be helpful.

		SVM		KLR	
		error rate	No. Eval	error rate	No.Eval
b3	Grid	*0.115*	625	*0.115*	625
	Random	0.140	100	0.153	100
	Nelder-Mead	0.140	*13*	0.134	756
	Pattern	0.146	41	0.153	*33*
balance	Grid	*0.000*	625	*0.000*	625
	Random	0.010	100	0.007	100
	Nelder-Mead	0.003	759	*0.000*	*9*
	Pattern	*0.000*	*53*	0.014	21
breast	Grid	*0.028*	625	0.029	625
	Random	0.031	100	0.034	100
	Nelder-Mead	0.041	*7*	0.031	769
	Pattern	*0.028*	37	*0.028*	*41*
covtype	Grid	0.188	625	0.186	625
	Random	0.191	100	0.187	100
	Nelder-Mead	0.191	*44*	*0.185*	770
	Pattern	*0.186*	69	0.187	*77*
dermatology	Grid	*0.000*	625	*0.000*	625
	Random	0.027	100	0.038	100
	Nelder-Mead	0.391	*3*	0.391	*3*
	Pattern	*0.000*	29	0.391	5
diabetes	Grid	*0.224*	625	*0.220*	625
	Random	0.229	100	0.236	100
	Nelder-Mead	0.286	*9*	0.223	*27*
	Pattern	0.227	57	0.225	49
digits	Grid	*0.003*	625	0.003	625
	Random	0.043	100	0.031	100
	Nelder-Mead	*0.003*	*17*	*0.001*	756
	Pattern	*0.003*	33	0.004	*45*
directmailing	Grid	*0.037*	625	*0.037*	625
	Random	*0.037*	100	*0.037*	100
	Nelder-Mead	*0.037*	*7*	*0.037*	*3*
	Pattern	*0.037*	17	*0.037*	5

Table 1: Comparison of numerical results of four methods to determine useful hyper-parameters.

		SVM		KLR	
		error rate	No. Eval	error rate	No. Eval
insurance	Grid	0.004	625	0.004	625
	Random	0.007	100	0.012	100
	Nelder-Mead	0.002	756	*0.002*	*15*
	Pattern	*0.001*	*53*	*0.002*	89
ionosphere	Grid	*0.043*	625	0.043	625
	Random	0.046	100	0.048	100
	Nelder-Mead	0.046	761	0.046	760
	Pattern	*0.043*	*45*	*0.040*	*41*
iris	Grid	*0.000*	625	*0.000*	625
	Random	*0.000*	100	*0.000*	100
	Nelder-Mead	*0.000*	*9*	*0.000*	*9*
	Pattern	0.007	*9*	0.007	17
liver	Grid	*0.270*	625	0.264	625
	Random	0.278	100	0.293	100
	Nelder-Mead	0.339	*34*	*0.261*	764
	Pattern	0.290	65	0.272	*61*
medicine	Grid	0.196	625	0.193	625
	Random	0.197	100	0.196	100
	Nelder-Mead	*0.195*	786	0.193	*39*
	Pattern	0.197	*77*	*0.191*	61
mushroom	Grid	*0.000*	625	*0.000*	625
	Random	0.002	100	0.003	100
	Nelder-Mead	0.482	*3*	0.482	*3*
	Pattern	*0.000*	49	*0.000*	57
promoters	Grid	*0.123*	625	0.132	625
	Random	0.566	100	0.594	100
	Nelder-Mead	0.566	*5*	0.651	*3*
	Pattern	0.566	9	*0.066*	73
voting	Grid	*0.032*	625	0.037	625
	Random	0.034	100	0.039	100
	Nelder-Mead	0.041	*23*	0.037	*15*
	Pattern	*0.032*	41	*0.034*	53
wine	Grid	*0.011*	625	*0.011*	625
	Random	*0.011*	100	0.028	100
	Nelder-Mead	0.022	*15*	0.022	*13*
	Pattern	*0.011*	37	*0.011*	37

Table 2: Comparison of numerical results of four methods to determine useful hyper-parameters.

		SVM	
		msep	No. Eval
benchartificial	Grid	0.0425	625
	Random	0.0437	100
	Nelder-Mead	*0.0422*	235
	Pattern	0.0427	97
	Cherkassky/Ma	0.0790	1
boston	Grid	0.1183	625
	Random	0.2293	100
	Nelder-Mead	*0.1167*	1573
	Pattern	*0.1167*	81
	Cherkassky/Ma	0.2367	1

Table 3: Comparison of numerical results of five methods to determine useful hyper-parameters.

- The response surfaces (as shown in the appendix) often show plateaus for inappropriate choices of the hyper-parameters. This could be expected as for bad choices of the hyper-parameters the estimated response $\hat{y}$ in the classification case collapses to a default one.

- Sometimes, the results differ grossly between SVM and KLR. Nevertheless the response surfaces of SVM and KLR have similar shapes.

- Most often there is no single optimal choice of the hyper-parameters but a connected region of optimal values.

- Although for every fixed combination of hyper-parameters the optimization problem is convex the above results show that we do not have a convex optimization problem in terms of the hyper-parameters.

- For the tested examples the response surfaces look very similar to each other with a kind of "J" shape.

- In most cases the level change in the target function is approximately parallel to the input parameters. Hence pattern search often performs as good as the more flexible Nelder-Mead method as it performs axes parallel optimization steps.

- The optimal region is always in the upper-left quadrant of the contour plot. A further increasing of C would not change the result as long as C is larger than all Lagrangian multipliers in the internal optimization of SVM and KLR (for details see Schölkopf and Smola (2002) and Keerthi *et al.* (2004)). A further reduction of γ would not change the result as with a small γ the kernel values converge to 1 (5).

5. Conclusion

The response surface for optimization of hyper-parameters for convex risk minimization methods turns out to be non-convex. No method performs best in every case but pattern search seems to be a good compromise between accuracy and computational effort. On the other hand grid- and random search are easily parallelized so if computational time is important and many processors are available grid search may be the method of choice.

A promising direction for further research can be the modelling of the typical shape of the response surface. An optimization method that exploits the typical shape of the response surface might result in additional improvements.

Acknowledgements

The financial support of the Deutsche Forschungsgemeinschaft (SFB-475) and of DoMuS (University of Dortmund, "Model building and simulation") are gratefully acknowledged. We also thank Maria Eveslage for her help with the figures and results.

References

Bartlett, P. and Tewari, A. (2004). Sparseness versus estimating conditional probabilities: Some asymptotic results. In *Proceedings of the 17th Annual Conference on Learning Theory, Vol. 3120*, Lecture Notes in Computer Science, Vol. 3120, pages 564–578, New York. Springer.

Blake, C. and Merz, C. (1998). UCI repository of machine learning databases.

Cherkassy, V. and Ma, Y. (2004). Practical selecion of svm parameters and noise estimation for svm regression. *Neural Networks*, **7**, 113–126.

Christmann, A. (2004). An approach to model complex high-dimensional insurance data. *Allg. Statist. Archiv*, **88**, 375–397.

Christmann, A. and Steinwart, I. (2004). On robust properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*, **5**, 1007–1034.

Christmann, A. and Steinwart, I. (2005). Consistency and robustness of kernel based regression. University of Dortmund, SFB-475, TR-01/05. Submitted.

Friedman, J. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*, **19**(1), 1–67.

Heilemann, U. and Münch, H. (1996). West german business cycles 1963-1994: A multivariate discriminant analysis. In *CIRET-Conference in Singapore*, CIRET-Studien 50.

Höffgen, K., Simon, H.-U., and van Horn, K. (1995). Robust trainability of single neurons. *Journal Computer and System Sciences*, **50**, 114–125.

Keerthi, S., Duan, K., Shevade, S., and Poo, A. (2004). A fast dual algorithm for kernel logistic regression. In *Machine Learning: Proceedings of the Ninetheenth International Conference (Eds: C. Sammut, A.G. Hoffmann)*, pages 299–306. Kaufmann, San Francisco.

Mattera, D. and Haykin, S. (1999). Support vector machines for dynamic reconstruction of a chaotic system. In: B. Schölkopf, J. Burger and A. Smola (editors): Advances in Kernel Methods: Support Vector Machine. MIT Press, Cambridge, MA.

Momma, M. and Bennett, K. (2002). A pattern search method for model selection of support vector regression. In: Proceedings of SIAM Conference on Data Mining.

Morik, K., Imhoff, M., Brockhausen, P., Joac hims, T., and Gather, U. (2000). Knowledge discovery and knowledge validation in intensive care. *Artificial Intelligence in Medicine*, **19**(3), 225–249.

Nelder, J. and Mead, R. (1965). A simplex method for functional minimization. *Computer Journal*, **7**, 308–313.

Schölkopf, B. and Smola, A. (2002). *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, Massachusetts.

Steinwart, I. (2001). On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, **2**, 67–93.

Steinwart, I. (2005). Consistency of support vector machines and other regularized kernel machines. *To appear in: IEEE Trans. Inform. Theory*.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, New York.

Zhang, T. (2004). Statistical behaviour and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, **32**, 56–134.

Andreas Christmann
University of Dortmund
Department of Statistics
44221 Dortmund
GERMANY
E-mail:
christmann@statistik.uni-dortmund.de

Karsten Luebke
University of Dortmund
Department of Statistics
44221 Dortmund
GERMANY
E-mail:
luebke@statistik.uni-dortmund.de

Marcos Marin-Galiano
University of Dortmund
Department of Statistics
44221 Dortmund
GERMANY
E-mail:
marcos.marin-galiano@uni-dortmund.de

Stefan Rüping
University of Dortmund
Department of Computer Science
44221 Dortmund
GERMANY
E-mail:
rueping@kimo.cs.uni-dortmund.de

Appendix: Figures of Response Surfaces

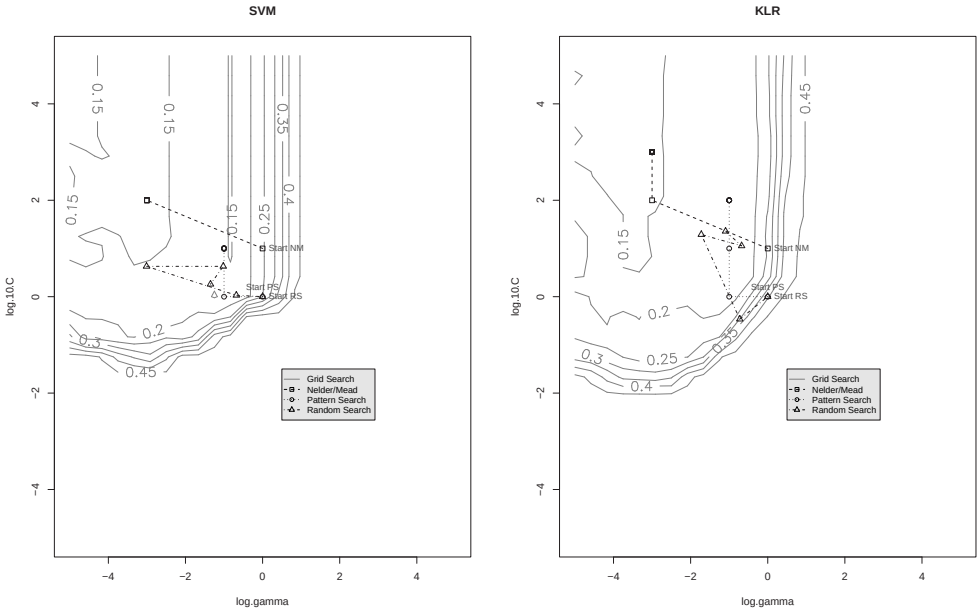

Figure 1: Response surface for b3 data

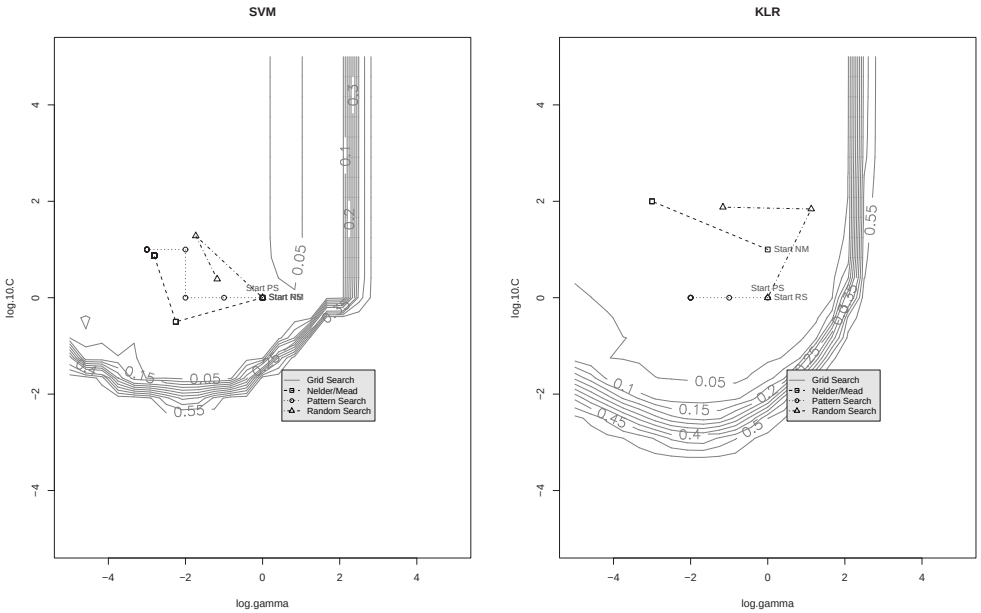

Figure 2: Response surface for balance data

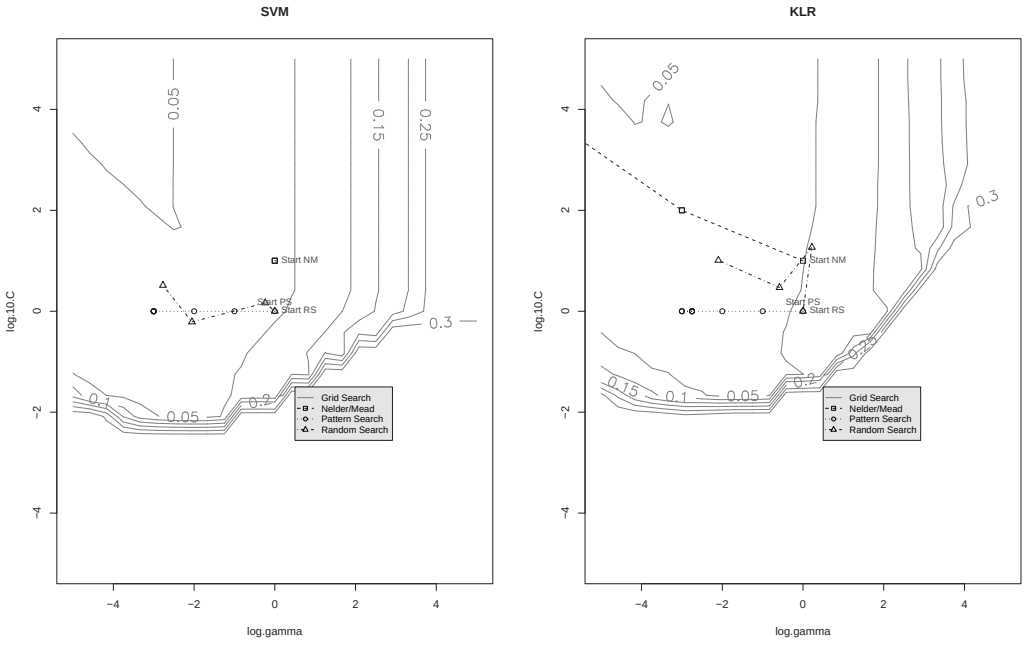

Figure 3: Response surface for breast data

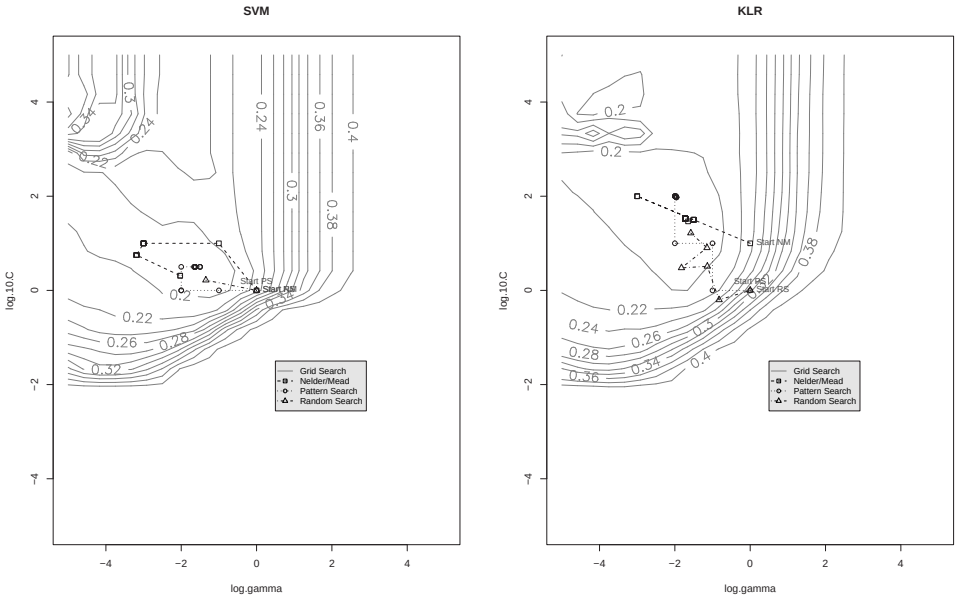

Figure 4: Response surface for covtype data

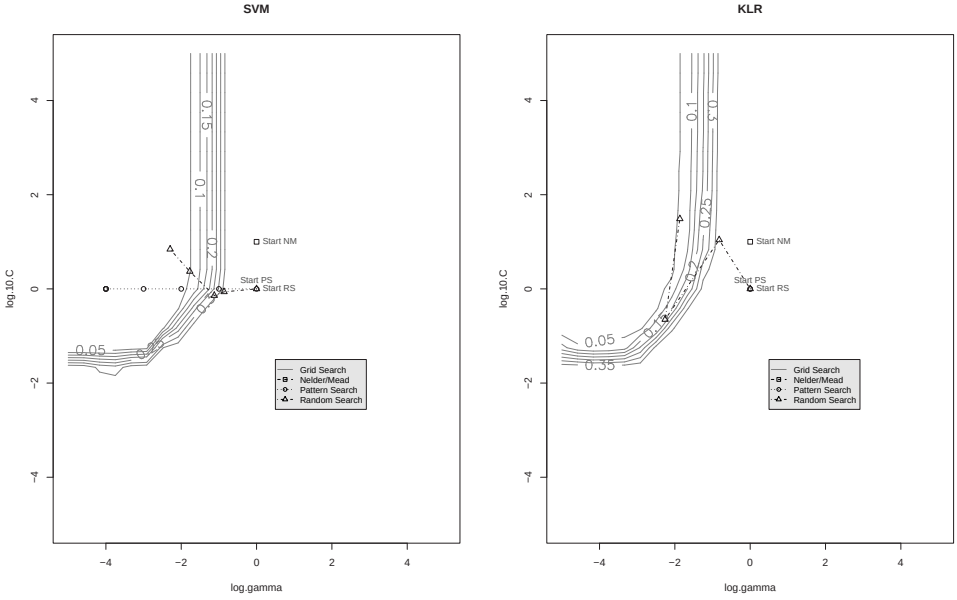

Figure 5: Response surface for dermatology data

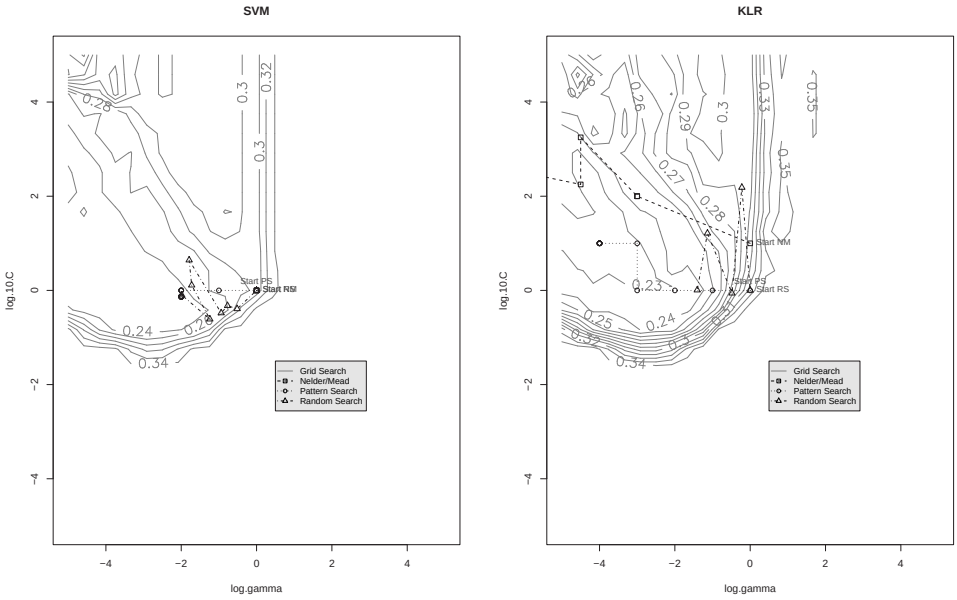

Figure 6: Response surface for diabetes data

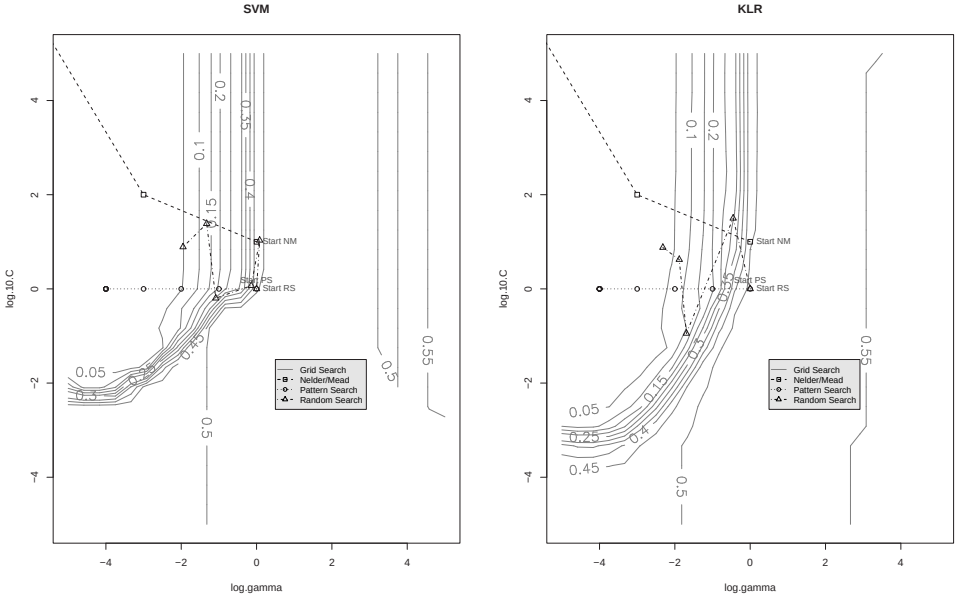

Figure 7: Response surface for digits data

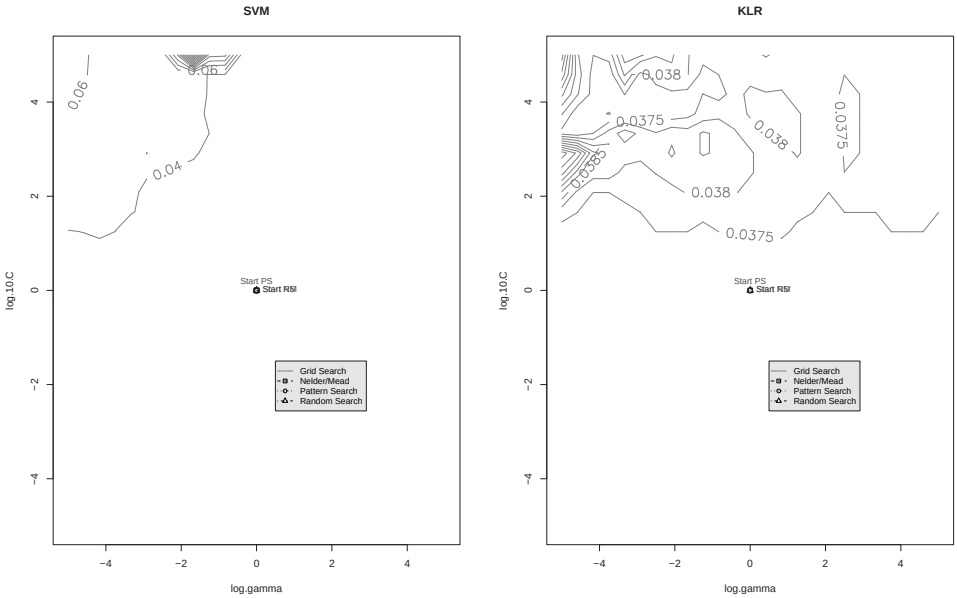

Figure 8: Response surface for directmailing data

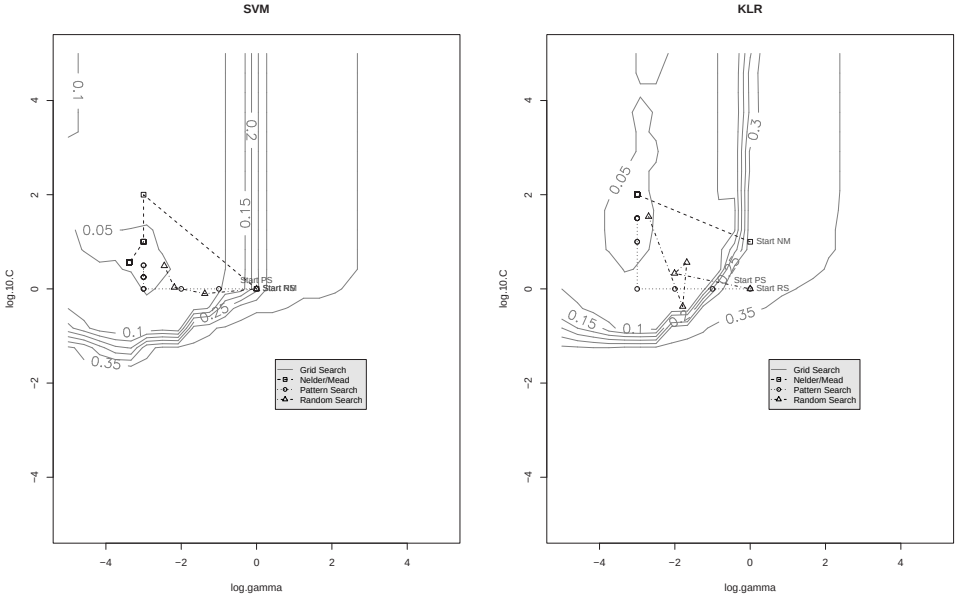

Figure 9: Response surface for ionosphere data

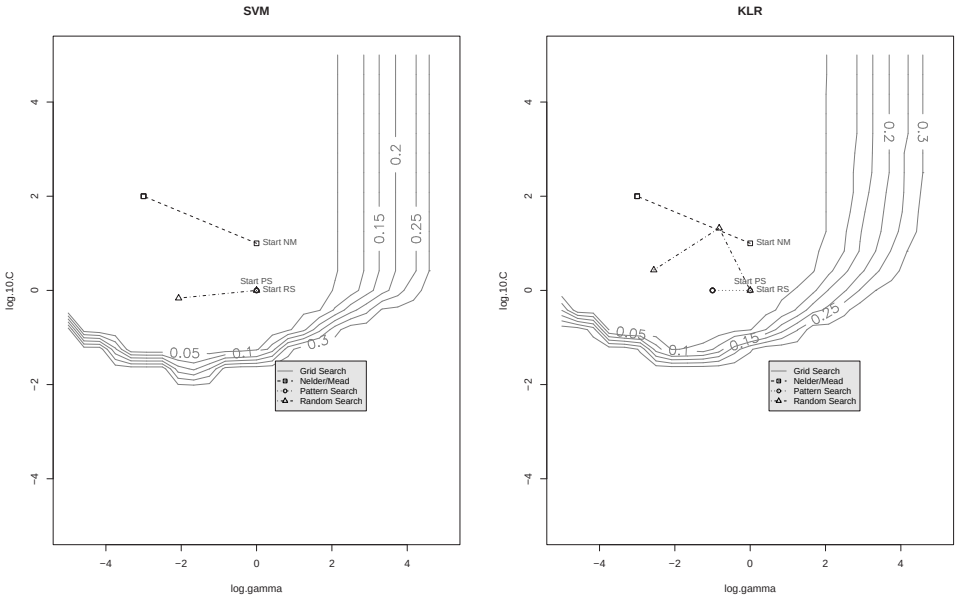

Figure 10: Response surface for iris data

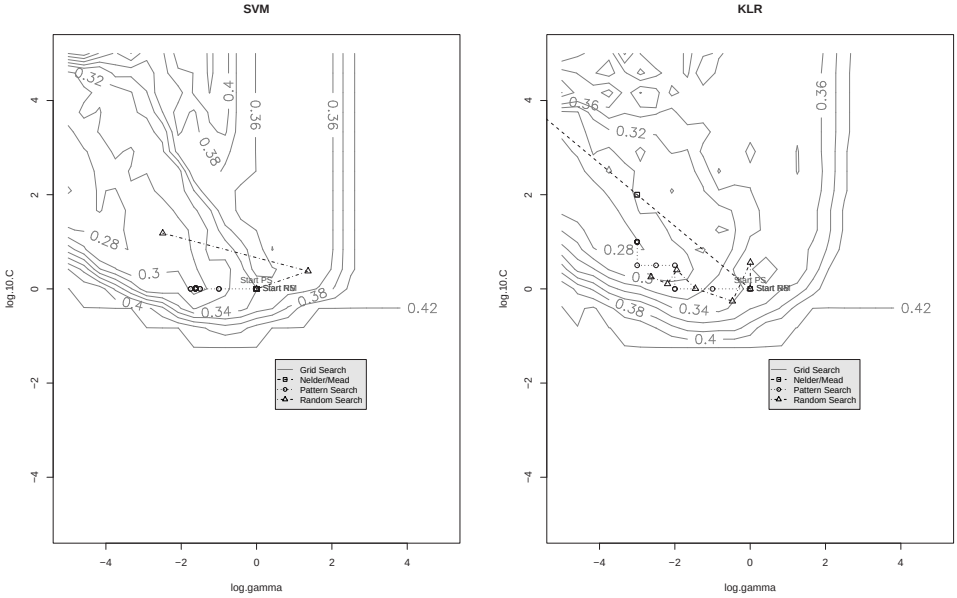

Figure 11: Response surface for liver data

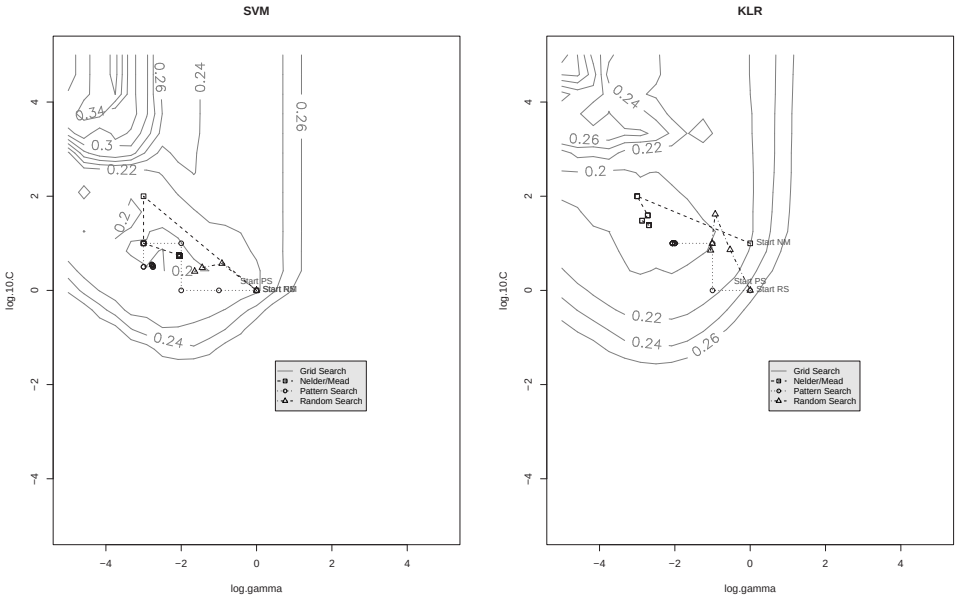

Figure 12: Response surface for medizin data

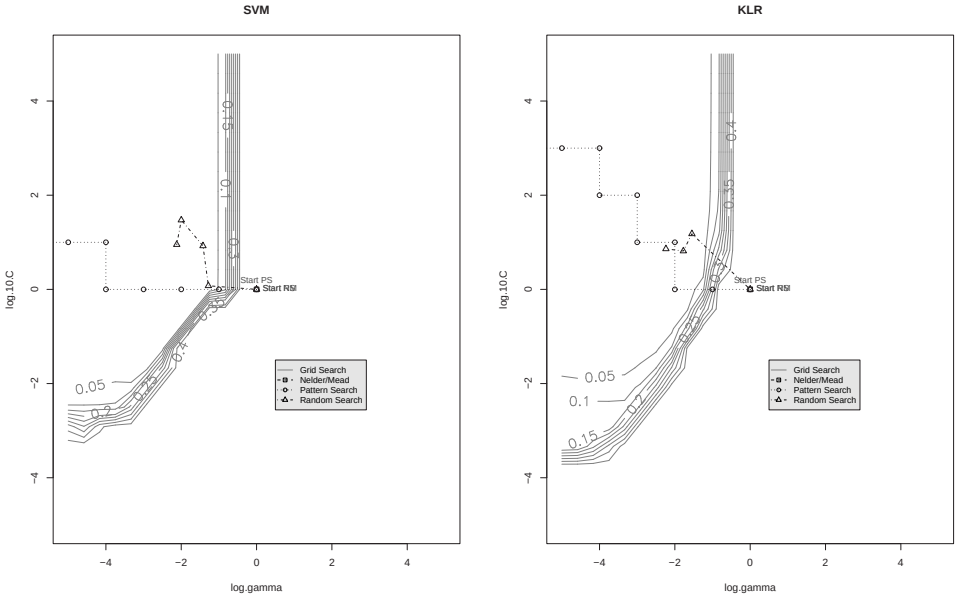

Figure 13: Response surface for mushroom data

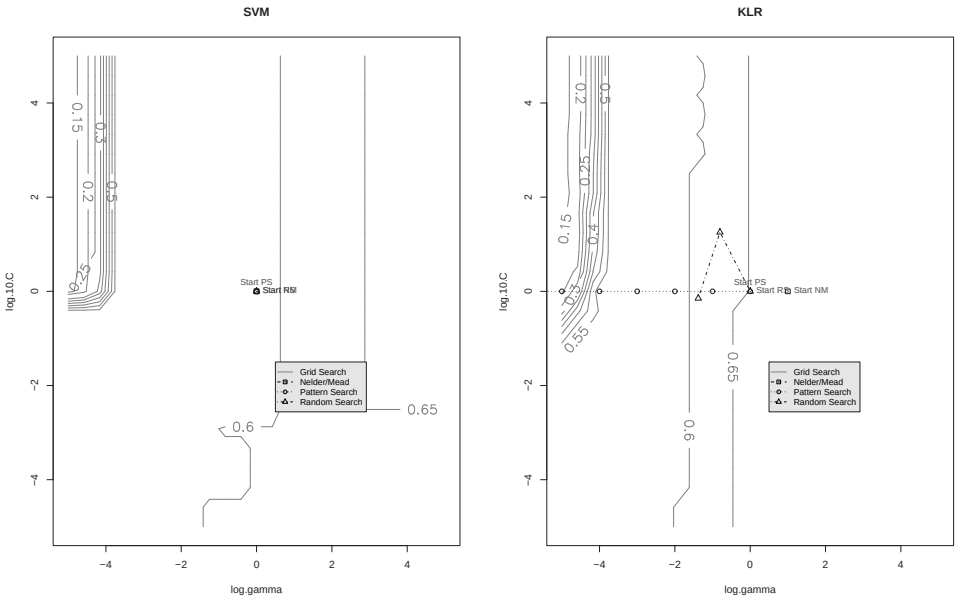

Figure 14: Response surface for promoters data

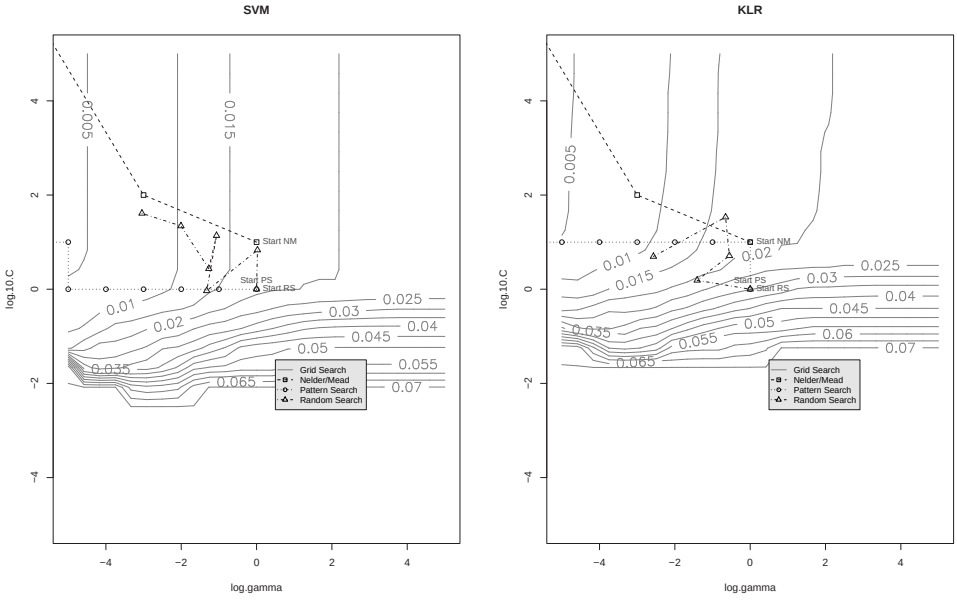

Figure 15: Response surface for versicherung data

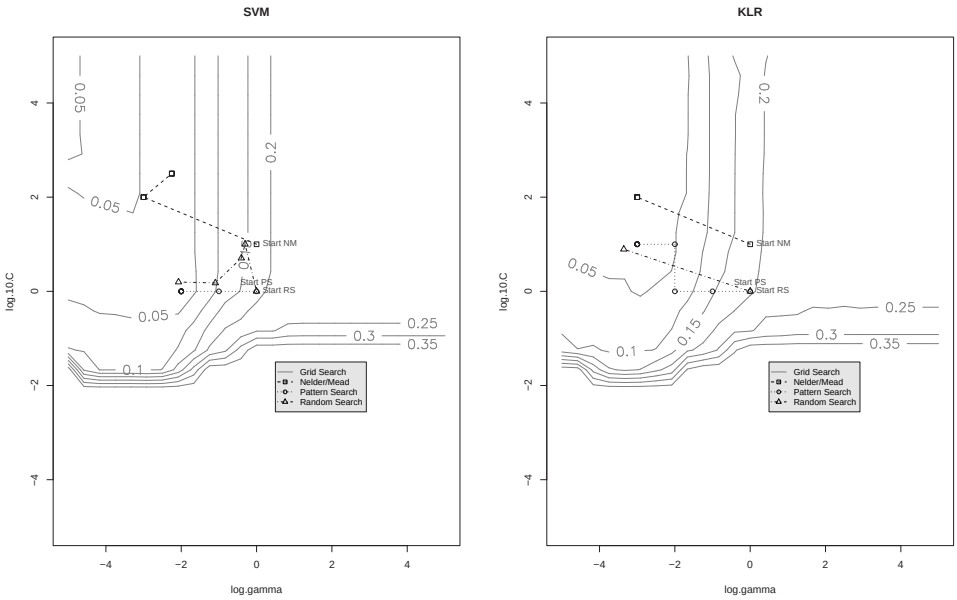

Figure 16: Response surface for voting data

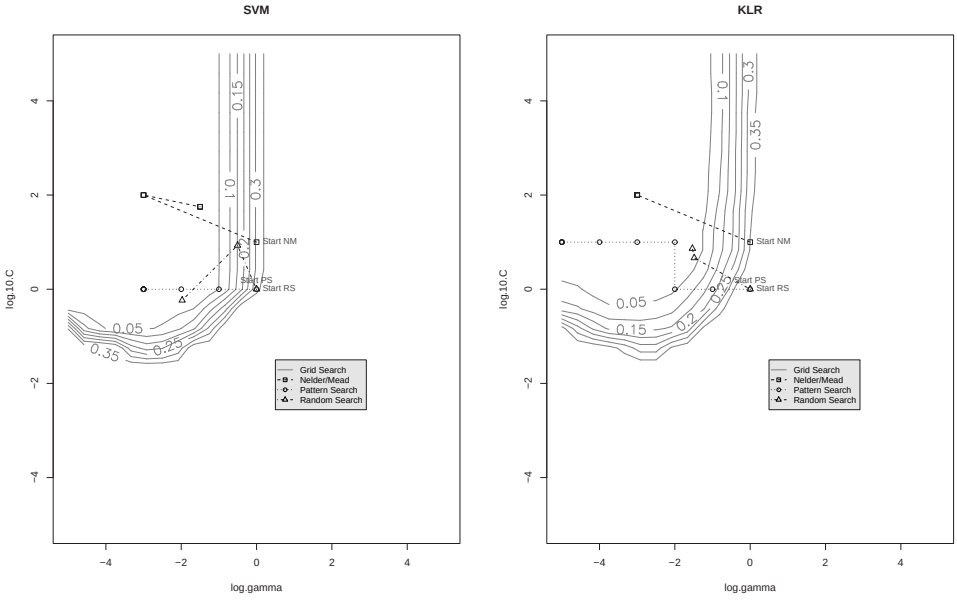

Figure 17: Response surface for wine data

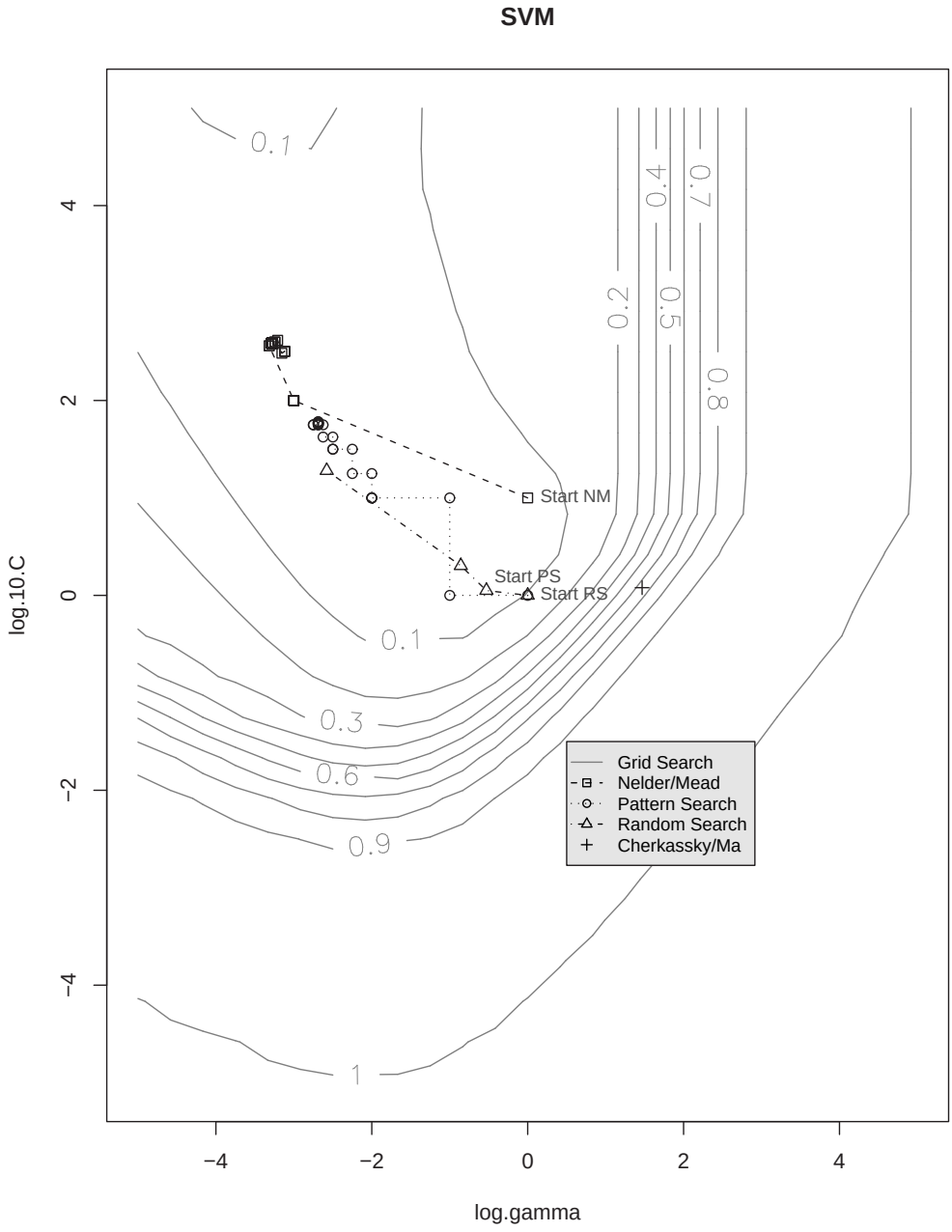

Figure 18: Response surface for benchartificial data

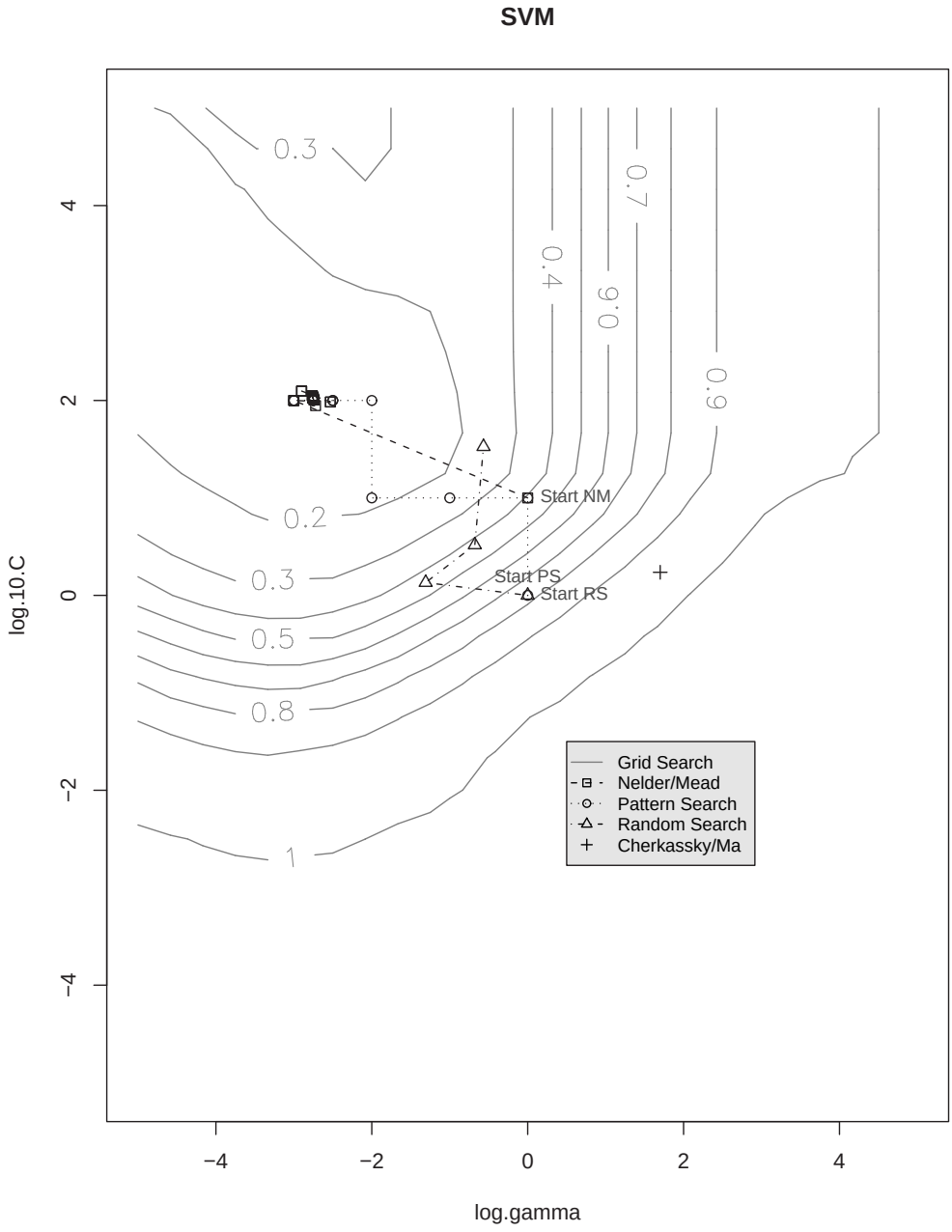

Figure 19: Response surface for boston data