IBM Research Report

Pattern Classification using the Principle of Parsimony: A Least Square Kernel Machine with Box Constraints

Jayanta Basak
IBM Research Division
India Research Lab
4, Block-C, Institutional Area (ISID Campus),
Vasant Kunj, Phase - II,
New Delhi - 110070, India.
e-mail : bjayanta@in.ibm.com

IBM Research Division
Almaden - Austin - Beijing - Delhi - Haifa - T.J. Watson - Tokyo - Zurich

Abstract

Among various principles in the theory of pattern classification to improve generalization, one of the most widely used principle is Occam's razor or the principle of parsimony. Structural risk minimization (SRM) and minimum description length (MDL) principle and their variants are essentially two different forms of Occam's razor used in pattern classification. In this article, we present a modified view of Occam's razor and use this principle to design a kernel-based classifier. The proposed classifier has close relationships with a widely different variety of "least-square" kernel-machines such as adaptive ridge regression, least square support vector machine, regularized least square classifier, LASSO (least absolute shrinkage and selection operator), and generalized LASSO. However, unlike the existing "least-square" kernel machines, the proposed classifier uses box constraints on the priors, and the box constraints are derived from the modified principle of parsimony. Experimental results demonstrate that the proposed classifier is capable of outperforming SVM in terms of cross-validation scores on several datasets. In addition, we also prescribe some kernel functions other than the Gaussian for obtaining better performance on real-life datasets.

1 Introduction

In pattern classification [64, 19, 66] problems, the loss or discrepancy $L(t, f(x, \alpha))$ between the desired response t to a given input x and the response provided by a learning machine $f(x, \alpha)$ is minimized with respect to the distribution $F(x, t)$. The goal is to minimize the risk functional

$$R(\alpha) = \int L(t, f(x, \alpha)) dF(x, t) \tag{1}$$

and obtain an α_0 such that $f(x, \alpha_0)$ minimizes $R(\alpha)$. The general expression of the risk functional is quite broad to encompass various specific problems such as pattern classification, regression (linear or nonlinear depending on the form of $f(x, \alpha)$), or density estimation. In the present article, we restrict ourselves only to the problem of pattern classification.

In two-class classification tasks, usually the desired response t can take two different values e.g., $t \in \{-1, 1\}$ and $f(x, \alpha)$ represents a set of indicator functions (with two possible values of -1 and 1) for different values of α. The loss function is then defined as

$$L(t, f(x, \alpha)) = \begin{cases} 0 & \text{if } t = f(x, \alpha) \\ 1 & \text{if } t \neq f(x, \alpha) \end{cases} \tag{2}$$

Since the distribution $F(x, t)$ is unknown, the loss functional is often replaced by the empirical risk functional

$$R_{emp}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} L(t_i, f(x_i, \alpha)) \tag{3}$$

where $L(t_i, f(x_i, \alpha))$ is the error or loss on the observed pair (x_i, t_i). In different classifiers, this empirical risk functional is minimized in various forms. For example, in neural networks [27], the output of the network $f(x, \alpha)$ is represented as a hyperbolic tangent function of the

total activation received from the previous layer ($tanh(.)$) due to the presence of input x at the input layer. The hyperbolic tangent function ensures that the output lie in the range $[-1, 1]$ (or $[0, 1]$ if the function is sigmoidal depending on the convention of representation). In such cases, assuming the Gaussian error model for the loss function [2], the maximum likelihood estimate of the empirical risk functional reduces to the least square estimate such that

$$R_{emp}(\alpha) = \frac{1}{N}\sum_{i=1}^{N}(t_i - f(x_i, \alpha))^2 \tag{4}$$

The empirical risk minimizes the error on the observed samples (training set) by finding out one optimal α depending on $f(.)$. However, the major question remains on the performance of the model $f(x, \alpha)$ on the unobserved samples. Since the true distribution $F(x, t)$ is unknown, we have to rely only on the observed samples, however that may lead to overfitting. Simple minimization of the empirical loss does not tell us anything about the generalization performance i.e., the performance of the model on the unobserved data. In general, minimization of the empirical risk functional with increased model complexity adds to the variance part of the generalization error [19, 21, 60, 23, 33], and thus leads to poor generalization. Among the various principles to improve generalization performance (a few representatives among the vast existing literature can be found in [17, 56, 57, 30, 20, 51, 32, 70, 55, 8, 9, 10, 49]), one of the most widely used principle is Occam's razor or the principle of parsimony [59].

The principle of parsimony or Occam's razor merely states that entities or explanations should not be multiplied beyond necessity. In the pattern classification, it can be interpreted as that the classifiers that are more complicated than necessary should not be used. The 'necessary' can be determined by the quality of fit (i.e., empirical loss) on the observed data. According to Vapnik [67], "when solving a given problem, try to avoid solving a more general problem as an intermediate step." This basic philosophy gives rise to many interesting principles for classification with an aim to improve the generalization performance. These include the structural risk minimization (SRM) [68, 66] and minimum description length (MDL) principle [45, 46].

Minimum description length (MDL) principle is based on the information theoretic analysis of the randomness in the concept. In pattern classification, as discussed in [19], according to MDL, the sum of the classifier's algorithmic complexity and the description of the training data with respect to the classifier should be minimized. MDL can also be observed from a Bayesian perspective where we have a hypothesis h for a training set D such that

$$P(h|D) = \frac{P(h)P(D|h)}{P(D)} \tag{5}$$

Since $P(D)$ is independent of h, the optimal hypothesis h^* can be obtained as

$$h^* = argmin_h[(-log_2 P(h)) + (-log_2 P(D|h))] \tag{6}$$

where $-log_2 P(h)$ and $-log_2 P(D|h)$ are the cost or complexity of the description of the hypothesis (model) and the training data respectively. In general, we need to minimize the cost of description $K(h, D)$, such that

$$K(h, D) = K(h) + K(D|h) \tag{7}$$

where $K(h)$ is the complexity of the model (hypothesis) and $K(D|h)$ is the cost of description of the observed data D with respect to the model h. For example, in the case of decision trees, $K(h)$ is the number of leaf nodes (the total number of nodes is simply twice the leaf nodes minus one for binary trees), and $K(D|h)$ is the weighted sum of entropies of the data at the leaf nodes. Thus by pruning decision trees, if we can reduce $K(h)$ without significantly increasing $K(D|h)$, we can obtain better generalization. Different variations of the MDL principle such as Akaike information criteria (AIC) [1], Bayes information criteria (BIC) [50] and network information criteria (NIC) [35] are available in the literature.

In structural risk minimization (SRM), on the other hand, the complexity of a classifier is measured in terms of the VC-dimension [68, 66]. Although SRM has originally been applied in classification, it provides a generic inductive principle for model selection for learning from a finite dataset (possibly small in size). SRM principle finds a trade-off between model complexity (VC confidence) and the empirical error of model fitting. According to this principle, functions or models in the hypotheses space are arranged into a hierarchy of nested subsets of increasing VC-dimension. If we have empirical risk in each subset then a model in the subset is selected for which the sum of empirical loss and VC confidence is minimum. According to the SRM principle, the overall risk $R(\alpha)$ is bounded as

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{VC(h)(log(2N/VC(h)) + 1) - log(\delta/4)}{N}} \tag{8}$$

with probability $1 - \delta$ (PAC theory), where N is the number of training samples and $VC(h)$ is the VC-dimension of a hypotheses h. Clearly, as the $VC(h)$ of a hypotheses h increases, the overall risk $R(\alpha)$ increases for a given δ even with the same empirical risk. Later in [52], the SRM principle has been modified to "data dependent SRM" using the fat shattering dimension.

Possibly one of the most unique pattern classifier developed in the light of SRM principle is the support vector machine (SVM) [16, 67, 13, 24]. In SVM, the margin between two classes is maximized by minimizing the description of the linear classifier, i.e., $\|w\|^2$ in a higher dimensional space $\phi(.)$ under the constraint of inequality type

$$t_i(w^T\phi(x_i) + b) \geq 1 \tag{9}$$

The problem is transformed into an unconstrained problem by the method of Lagrange undetermined multipliers such that the functional

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i(t_i(w^T\phi(x_i) + b) - 1) \tag{10}$$

is minimized with respect to $\mathbf{w}$, b, and maximized with respect to $0 \leq \alpha_i \leq 1$. This leads to a quadratic optimization problem of the form

$$W_{svm}(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j t_i t_j \phi(x_i)\phi(x_j) \tag{11}$$

subject to $\alpha_i \geq 0$ and $\sum_i \alpha_i t_i = 0$. Introducing slack variables for non-separable cases, and taking into account of the inner product in the Hilbert space, the functional $W_{svm}(\alpha)$ is modified

as

$$W_{svm}(\alpha) = \sum_i \alpha_i - \frac{1}{2C} \sum_{i,j} \alpha_i \alpha_j t_i t_j K(x_i, x_j) \tag{12}$$

subject to $0 \leq \alpha_i \leq 1$ and $\sum_i \alpha_i t_i = 0$, where C is judiciously chosen constant and $K(x_i, x_j) = \phi(x_i)\phi(x_j)$ is a symmetric kernel expressible as an inner product in the higher dimensional space subject to Mercer condition [67]. Alternatively [53, 6] have shown that the quadratic optimization functional for soft-margin SVM in the non-separable cases can be expressed as

$$W_{softsvm}(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j t_i t_j (K(x_i, x_j) + \frac{1}{C}\delta_{ij}) \tag{13}$$

subject to $0 \leq \alpha_i \leq C$, where δ_{ij} is a Kronecker delta function such that the constant $1/C$ can be multiplied only with the diagonal entries. Once the functional $W_{svm}(\alpha)$ is maximized with respect to α_is to obtain nonzero α_is (support vectors), the indicator function $f(x, \alpha)$ is obtained as

$$f(x, \alpha) = sign(\sum_i \alpha_i K(x, x_i) t_i + b) \tag{14}$$

where

$$sign(u) = \begin{cases} +1 & \text{if } u > 0 \\ -1 & \text{otherwise} \end{cases} \tag{15}$$

is a signum function.

Support vector machines are constructed on the basis of maximizing the margin between two classes in some higher dimensional input space. In this article, we propose a pattern classification scheme using a modified form of the principle of parsimony. The way we view the principle of parsimony is different from the principle of SVM or MDL principle. It does not consider the margin maximization as performed in the SVM. We use the information that is optimum for classification of the patterns. We view the task of pattern classification from the point of view of approximating class-conditional densities (such as Parzen window [39]) subject to the modified form of the principle of parsimony.

Instead of maximizing the margin between two classes in a higher dimensional space as performed in SVM, we fix a margin and minimize the redundancy in estimating the class posteriors. In SRM or MDL, the redundancy is viewed as the redundancy present in the model i.e., model complexity (either information theoretic or in the VC framework). Here we do not view the redundancy as the model complexity rather consider the redundancy in the form of superfluity in the outcome. First, we show how to use our principle by formulating a two-class classification problem. We then consider the multiclass classification using the one-against-all strategy [43].

Our formulation for pattern classification leads to a classifier which is very similar to the existing family of "least square" kernel machines [12, 48, 58, 65, 44, 61, 38, 47] with the exception that we design a least square kernel machine with box constraint. Box constraint has not been applied in designing least square kernel machine for pattern classification so far. We derive the motivation for applying box constraint in designing a least square kernel machine from the modified principle of parsimony as we stated in the next section. Experimental results show that our technique is

capable of outperforming SVM on several datasets that we have used. We also do not require the kernels to satisy the Mercer condition as it is required in SVM.

In Section 2, we state a modified view of the principle of parsimony (Occam's razor) and then provide the basic formalism for a two-class classification task by using this principle. We then use this classification scheme for performing multi-class classification tasks using one-against-all classification [43]. In Section 3, we show the relationship of the proposed classification technique with a wide variety of existing "least-square" kernel machines such as adaptive ridge regression [12, 48], least square support vector machines or LSSVM [58, 65], Regularized least square classification or RLSC [44], least absolute shrinkage and selection operator or LASSO [61, 38], and generalized LASSO [47]. In Section 4, we demonstrate the effectiveness of our classification method on certain real-life datasets in terms of 10-fold cross-validation score, and also compare the results with that of SVM. Finally, we discuss and conclude our paper in Section 5.

2 Pattern Classification with the Principle of Parsimony

Principle: Here we present a modified form of Occam's razor principle (the principle of parsimony) which we use in the subsequent design of our classifier. We state our principle as

Minimize the superfluity in the outcome with a given margin.

The whole question now boils down into 'how to measure the superfluity in the outcome'? We address this question for a pattern classification problem. Let us first restrict ourselves in two-class classification tasks.

Two Class Classification: In two-class classification task, let the desired response t can take two different values such that $t \in \{-1, 1\}$ and $f(x, \alpha)$ represent a set of indicator functions in $\{-1, +1\}$ for different values of α. The loss function is

$$L(t, f(x, \alpha)) = \begin{cases} 0 & \text{if } t = f(x, \alpha) \\ 1 & \text{if } t \neq f(x, \alpha) \end{cases} \tag{16}$$

In any two-class classifier (either parametric or non-parameteric), the indicator function $f(x, \alpha)$ can be viewed as a signum of a soft-indicator output $g(x, \alpha)$ such that

$$f(x, \alpha) = sign(g(x, \alpha)) \tag{17}$$

where $sign(u)$ has the same definition as in Equation (15).

For example, in a k-nearesr neighbor classifier [25, 19], a test sample x is assigned a label $+1$ (i.e., $f(x, \alpha) = +1$) if the number of positive training samples (i.e., training samples with $t = +1$) is more than the number of negative training samples (i.e., training samples with $t = -1$) in a neighborhood $\Omega_k(x)$ of x comprising of k nearest neighbors. The corresponding soft-indicator function $g(x, \alpha)$ can be expressed as

$$g(x, \alpha) = \left(\frac{2N_+(x)}{k} - 1 \right) \tag{18}$$

where $N_+(x)$ is the number of positive training samples in $\Omega_k(x)$. Note that, we scaled the soft-indicator function $g(x, \alpha)$ such that $g(x, \alpha) \in [-1, +1]$, although it can be scaled differently. The model α here is defined by k. In a decision tree [11, 42], a leaf node is assigned a positive or negative label depending on the majority of the labels of the training samples allocated to that leaf node. Thus if a test sample x is allocated to a leaf node $l_T(x)$ of a decision tree T then the soft-indicator function $g(x, \alpha)$ is given as

$$g(x, \alpha) = \left(\frac{2N_+(l_T(x))}{N_+(l_T(x)) + N_-(l_T(x))} - 1 \right) \tag{19}$$

where $N_+(l_T(x))$ and $N_-(l_T(x))$ are respectively the number of positive and negative training samples allocated to leaf node $l_T(x)$. In the case of support vector machines, the soft-indicator function $g(x, \alpha)$ directly comes from Equation (14) such that

$$g(x, \alpha) = \sum_i \alpha_i K(x, x_i) t_i + b \tag{20}$$

In the case of parametric classifiers such as logistic regression models [29] and semi-parameteric classifiers such as neural networks [27], hierarchical mixture of experts [31], and online adaptive decision trees [4, 5], the classifiers themselves produce the soft-indicator function $g(x, \alpha)$. The final outcome of such classifiers is interpreted as $f(x, \alpha) = sign(g(x, \alpha))$. In short, the outcome of any two-class classifier can be viewed as a soft-indicator function $g(x, \alpha)$ which is then mapped onto $f(x, \alpha) \in \{-1, +1\}$ for making a decision.

The nature of $g(x, \alpha)$ is driven by the underlying model α and the generalization performance of the classifier depends on the nature of $g(x, \alpha)$. In this paper, we refer to $g(x, \alpha)$ as the "outcome" of a classifier (as in the modified principle). Evidently, we can map the "outcome" into a decision by comparing it with zero. At the boundary i.e., $g(x, \alpha) = 0$, we fix a margin λ (as in principle) as a tolerence parameter such that for all training samples,

$$f(x, \alpha) = \begin{cases} +1 & \text{if } g(x, \alpha) \geq \lambda \\ -1 & \text{if } g(x, \alpha) \leq -\lambda \end{cases} \tag{21}$$

The parameter λ can be connected to regularization such that if λ is very small then a small perturbation in x can cause $g(x, \alpha)$ to be mapped from one class to the other. Thus for better performance on the training samples, we need to a select as large λ as possible. On the other hand, if we select a very large λ then the model α will be highly perturbed by the noisy training samples resulting in a poor generalization performance.

According to the modified principle of parsimony, we fix a margin λ and minimize the superfluity in the outcome $g(x, \alpha)$ with the given margin. Note that, it is different from either MDL or SRM principle in the sense that we do not take into account the model complexity (description length) or maximize the margin (defined by the model). Rather we concentrate only on the outcome $g(.)$ for a given margin λ. If the outcome $g(x, \alpha)$ is greater than λ for a training sample x with a label $+1$ or if $g(x, \alpha) < -\lambda$ for a training sample x with label -1 then we say that there is superfluity in the outcome since $g(x, \alpha) = \pm\lambda$ is sufficient to make a decision for a given margin λ.

Let us now proceed to formulate a loss function for two-class classification problem using the modified principle of parsimony. With a given margin λ, we can say that there is an error if

$g(x,\alpha) < \lambda$ for a training sample x with a classlabel $+1$ i.e., $t(x) = +1$. Similarly, there is an error if $g(x,\alpha) > -\lambda$ for a training sample x with a classlabel $t(x) = -1$. Therefore, with the convention of representing classlabels as $t \in \{-1, +1\}$, we can measure the empirical error for a training sample x as

$$L_1(x,t) = \begin{cases} h_1((\lambda t - g(x,\alpha))sign(t)) & \text{if } (g(x,\alpha) < \lambda) and (t = 1) \text{ or } (g(x,\alpha) > -\lambda) and (t = -1) \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where $h_1(.)$ is a monotonically increasing function.

On the other hand, as we mentioned, with the given margin λ, if for a positive training sample x, $g(x,\alpha) > \lambda$, or for a negative training sample x, $g(x,\alpha) < -\lambda$ then there is superfluity in the outcome $g(x,\alpha)$. We can measure the superfluity in the outcome for a training sample x as

$$L_2(x,t) = \begin{cases} h_2((g(x,\alpha) - \lambda t)sign(t)) & \text{if } (g(x,\alpha) > \lambda) and (t = 1) \text{ or } (g(x,\alpha) < -\lambda) and (t = -1) \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

where $h_2(.)$ is a monotonically increasing function.

According to our modified principle of parsimony, we need to minimize the superfluity in the outcome (Equation (23)) for better generalization as well as need to minimize the empirical error (Equation (22)) for better training with a given margin λ. Thus the resulting loss function for a training sample x is given as

$$L(x,t) = L_1(x,t) + L_2(x,t) \tag{24}$$

Restricting $h_1(u) = h_2(u) = u^2$, we have

$$L(x,t) = (g(x,\alpha) - \lambda t)^2 \tag{25}$$

for a sample x. In this context we mention that it is not necessary that $h_1(.)$ and $h_2(.)$ have to be identical. However, restricting them to be identical we express the total error as quadratic loss. Note that, the expression in Equation (25) looks similar to the empirical loss for regression [67, 19]. In regression, the quadratic expression represents only the empirical error underlying Gaussian noise or non-Gaussian noise expressible as Huber loss. However, the loss functional in regression has nothing to do with the redundancy in the form of superfluity in the outcome. Here Equation (25) represents the empirical error and redundancy in the form of superfluity in the outcome together for a two-class classification problem subject to a given margin λ.

Let us now consider the kernel based classifiers such that the function $g(x,\alpha)$ is given as

$$g(x,\alpha) = \sum_j \alpha_j K(x,x_j) t_j \tag{26}$$

where x_j is an observed sample and t_j is the given label of the observed sample, and $0 \leq \alpha_j \leq 1$ indicates the contribution of the sample j in the classifier (equivalent to undetrmined Lagrangian as used in SVM), and the indicator function $f(.)$, $f(x,\alpha) = sign(g(x,\alpha))$. $K(x,x_j)$ is a kernel function such as Gaussian kernel of the form

$$K(x,x_j) = \exp(-\frac{\|x - x_j\|^2}{2\sigma^2}) \tag{27}$$

Therefore, if we consider the empirical error and superfluity in the outcome on all observed(training) samples together, we can get

$$L = \frac{1}{2}\sum_i (\sum_j \alpha_j K(x_i, x_j)t_j - \lambda t_i)^2 \tag{28}$$

For a given λ, the minimization of L with respect to α is equivalent to maximization of the functional

$$W(\alpha) = \lambda t^T K D(t)\alpha - \frac{1}{2}\alpha^T D(t)K^T K D(t)\alpha \tag{29}$$

subject to the condition that $0 \leq \alpha_i \leq 1$ for all i. The entry $D(t)$ is a diagonal matrix whose diagonal entries are the same as that of t.

The effects of αs are similar to that used in the SVM where αs define the support vectors. However the quadratic function to be maximized in SVM is derived on the basis of margin maximization with respect to optimal separating hyperplane in some higher dimensional space $\phi(.)$ such that K is expressible as a inner product, $K(x, x_i) = \phi(x)\phi(x_i)$, and αs are the Lagrangians. In our case, we do not explicitly consider the margin maximization with respect to separating hyperplane in the space of $\phi(.)$, rather we minimize the superfluity in the outcome for a given margin λ with respect to the difference in the class posteriors. This leads to a difference in the quadratic term $K^T K$ in our formulation in Equation (29) from that in the SVM. Apart from the quadratic term, we also observe the difference with SVM in the linear term. In SVM, due to margin maximization all α_is contribute equally in the linear term as in

$$W_{svm}(\alpha) = \mathbf{1}^T\alpha - \frac{1}{2C}\alpha^T D(t)K D(t)\alpha \tag{30}$$

with an additional constraint that $\alpha^T t = 0$ and $\alpha_i \geq 0$ for all i. In the modified framework of soft margin SVM [53, 6], the funcitonal is expressed as

$$W_{softsvm}(\alpha) = \mathbf{1}^T\alpha - \frac{1}{2}\alpha^T D(t)(K + \frac{1}{C}I)D(t)\alpha \tag{31}$$

where I is the identity matrix with the same constrint i.e., $\alpha^T t = 0$. We do not require this constraint (Equation (29)) since different α_is contribute differently in the linear term depending on the kernel function and the distribution of the patterns.

Once the pattern vectors with non-zero α values are obtained by maximizing the functional $W(\alpha)$ with respect to α_is (Equation 29), we classify any new sample x, i.e., assign a classlabel t to x as

$$t = \begin{cases} 1 & \text{if } \sum_i \alpha_i K(x, x_i)t_i \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{32}$$

In the sequel, we refer to our method of classification as 'B' machine as opposed to SV machine which we consider as 'A' machine. The non-zero αs determine the 'B' vectors as opposed to the support vectors which we consider as the 'A' vectors. Alternatively, B-machine can also be called B-SVM to represent the 'Box' constrained least square SVM.

Multi-class Classification: We formulated the classification problem for a two-class classification task. In the case of multi-class classification, we consider the one-against-all strategy,

i.e., classify the patterns of each class against all other classes and obtain the B-vectors for each class separately. Here we mention that we take the stand-point as provided in [43] where the authors have shown that one-vs-all classification scheme is as accurate as any other multiclass classification approach.

Formally let there be l classlabels $L = \{1, 2, \cdots, l\}$. Each time we consider one classlabel at a time. Let the classlabel of concern be c. In that case $t_i \in L$ is transormed into a vector $\{t_{i1}, t_{i2}, \cdots, t_{il}\}$ such that

$$t_{ic} = \begin{cases} 1 & \text{if } t_i = c \\ -1 & \text{otherwise} \end{cases} \tag{33}$$

For each classlabel c, we compute the B-vectors separately, and the classlabel t to a new sample x is assigned as

$$t = argmax_{c \in L} \{ \sum_i \alpha_{ic} K(x, x_i) t_{ic} \} \tag{34}$$

where α_{ic}s are the B-vectors obtained for class c.

3 Relationship with "Least Square" Kernel Machines

The proposed B-machine has a close relationship with a class of existing variants of kernel machines which we refer to as "least-square" kernel machines. Ridge regression framework [34] which is variant of the well-known least square method where the objective functional is of the form

$$L = \|t - w^T x\|^2 \tag{35}$$

for a linear regression fit. In general, for any functional form, considering additive Gaussian noise the regression functional becomes

$$L = \|t - f(x)\|^2 \tag{36}$$

In the ridge regression framework [34], the least square framework is duly modified to include a regularization factor such that

$$L = \|t - w^T x\|^2 + a\|w\|^2 \tag{37}$$

where a is a Lagrangian. The ridge regression framework introduces priors over the coefficients w for a smooth regression function estimation, the priors being subjected to a spherical Gaussian prior distribution. In adaptive ridge regression [48], automatic relevance determination is performed in such a way that each variable is penalized by the method of automatic balancing while keeping the average penalty constant. In relevance vector machines [63] also, an automatic relevance determination over the priors is intorduced by having prior variance for each expansion coefficient (a Bayesian framework). In this framework, an iterative method is employed. It starts with a current estimate of α vectors, and based on this estimate the most probable hyperprior (prior varaince) parameters are chosen. The prior variance parameters are then used in the posterior to get new estimates of α. Both adaptive ridge regression and RVM (relevance vector machine) follow iterative processes to estimate the coefficients and hyperpriors, although they differ in their treatment.

The proposed B-machine has similarity with the class of "least-square" kernel machines namely least-square support vector machine or LS-SVM [58, 65] and regularized least-square classification or RLSC [44]. In RLSC [44], the objective functional is derived as a simple square-loss function and regularized using the Tikhonov regularization [62] with a quadratic term in α vectors. In RLSC reproducing Hilbert kernels are considered such that the objective functional takes a form

$$L = \frac{1}{N}(t - K\alpha)^T(t - K\alpha) + \frac{\lambda}{2}\alpha^T K\alpha \tag{38}$$

where N is the number of samples, λ is a regularization parameter, and α are the coefficients (the notation of α is same throughout this article). This convex optimization problem is then transformed into a set of linear equations (similar to least square regression) by equating the first order derivative to zero based on the assumption that the underlying noise is generated by a Gaussian model. The coefficients are then derived from the resulting set of linear equations such that

$$(K + \lambda NI)\alpha = t \tag{39}$$

An approach similar to RLSC namely, least-square support vector machine (LSSVM) was proposed by [58, 65]. In LSSVM also, a squared error term is considered in addition to a regularization term $w^T w$ where w represents the directionality of the separating hyperplane. The convex optimization functional is then transfromed into a set of linear equations by equating the first derivative to zero. Here also the underlying assumption is that the noise is generated by a Gaussian model and thus a regression framework has been built in the kernelized representation.

In the framework of LASSO (least absolute shrinkage and selection operator) [38], a similar quadratic function is minimized which simply reduces to L_1-constrained least square problem given as

$$L = \|t - K\alpha\|_2^2 \tag{40}$$

subject to $\|\alpha\|_1 < \lambda$ where λ is a constant. As discussed in [47], the value of the constant λ determines the approximation accuracy and therefore can be viewed as a model selection parameter. The LASSO type models can be connected to adaptive ridge regression (AdR) by considering the exponetial hyperpriors instead of the Gaussian hyperpriors. Considering the exponential hyperpriors, the LASSO model can be expressed in the form

$$L = \|t - K\alpha\|_2^2 + b\|\alpha\|_1 \tag{41}$$

where b is a Lagrange parameter determined by the exponential hyperpriors.

The quadratic functional in all these related models, is directly motivated by the underlying Gaussian noise model. However, if the noise deviates from the Gaussian model then quadratic loss functional becomes sub-optimal. In order to incorporate a more robust error measure, generalized LASSO [47] has been proposed. Generalized LASSO takes into account of the Huber loss function [54] which is quadratic for smaller deviation, however follows a linear pattern for larger deviations. In generalized LASSO, iteratively reweighted least square (IRLS) technique is incorporated with this robust Huber's loss criteria. The generalized LASSO using IRLS technique has been successfully applied to two-class classification problems where logistic regression [29]

has been formulated with binary target variables using a Bernoulli error model. The generalized LASSO model for classification finally boils down to an optimization functional

$$L = \|\hat{t} - \hat{K}_\tau \alpha_\tau\|_2^2 \tag{42}$$

subject to $\|\alpha_\tau\|_1 < \kappa$. The subscript τ represents a vector preserving only those samples for which the coefficients α do not vanish. The transformed variables $\hat{t}$ and $\hat{K}$ are given as

$$(\hat{t}, \hat{K}) = (W^{1/2}q, W^{1/2}K) \tag{43}$$

where W is a diagonal matrix (dependent on α coefficients) such that $W = diag(\pi_1(1-\pi_1), \pi_2(1-\pi_2), \cdots, \pi_N(1-\pi_N))$ for N samples, π is the success probability such that $\pi(x) = p(y=1|x)$. The variable q is expressed as $q = K\alpha + e$ where e is the error vector such that $e_j = (\pi_j - y_j)/W_{jj}$. Essentially, in generalized LASSO an equivalent problem is solved in the form of iterative optimization by following the gradient of the logistic regression functional for two-class classification.

Interestingly, the quadratic loss has been considered in a wide variety of similarly aligned algorithms considering the Gaussian noise model as well as non-Gaussian model such as generalized LASSO. In the B-machine construct, we also consider the quadratic loss both to measure the error in observation as well as the redundancy as measured in terms of the superfluity in the outcome for labeled samples. Although we deal with quadratic loss functional, we derive it from a different viewpoint of the principle of parsimony. We can revisit our B-machine construct in the following way. As in equation (28), the loss for B-machine is given as

$$L = \frac{1}{2}\sum_i (\sum_j \alpha_j K(x_i, x_j) t_j - \lambda t_i)^2 \tag{44}$$

subject to $0 \leq \alpha_i \leq 1$ for all i. The loss can be reformulated as

$$L = \frac{\lambda^2}{2}\sum_i (\sum_j \frac{\alpha_j}{\lambda} K(x_i, x_j) t_j - t_i)^2 \tag{45}$$

which is equivalent to minimizing a functional

$$L = \frac{1}{2}\sum_i (\sum_j K(x_i, x_j) \hat{\alpha_j} t_j - t_i)^2 \tag{46}$$

subject to the constraints $0 \leq \hat{\alpha_i} \leq \frac{1}{\lambda}$. From the Equation (46) and the respective constraints, we observe that B-machine employs a uniform prior over the coefficients α. The uniform prior can be better observed if we replace $\hat{\alpha_i} t_i$ by β_i then we obtain the quadratic loss as

$$L = \frac{1}{2}\sum_i (\sum_j K(x_i, x_j) \beta_j - t_i)^2 \tag{47}$$

subject to the constraint that $0 \leq |\beta_i| \leq \frac{1}{\lambda}$, and $sign(\beta_i) = t_i$ where t_i is a binary observation in $\{-1, 1\}$. Thus the B-machine construct can be observed in a Bayesian framework where the model selection parameter is governed by λ, and the coefficients have uniform prior over $[0, \pm 1]$ depending on the classlabel of the observed sample. In other words, B-machine operates on

a quadratic optimization functional similar to the "least square" kernel machines except that B-machine employs box constraint on the parameter values.

We observe the similarity of the first part in Equation (38) of RLSC with the optimization functional of B-machine as indicated in Equation (47). However, in RLSC, the coefficients are derived based on the assumption that the noise is generated by a Gaussian model. Since the prior is Gaussian, it is possible to take the first-order derivative of the optimization functional. B-machine, on the contrary, does not assume the Gaussian prior rather it considers the uniform prior over the coefficients. Secondly, RLSC derivation is based on the assumption of reproducing Hilbert kernels. B-machine is not limited only to Mercer kernels or reproducing Hilbert kernels since in Equation (29) of B-machine, the quadratic term is always positive semi-definite. B-machine differs from LSSVM in the same aspect as it differs from RLSC, i.e., in B-machine uniform prior is constructed over the coefficients, and the generating noise is no more attributed to Gaussian model. Generalized LASSO also deviates from the Gaussian noise model and develops on a more robust Huber loss measure. From equation (47) and the generalized LASSO model in equation (42), we observe the similarity in the loss functionals except that the B-machine employs uniform prior. As we view the expanded quadratic loss in Equation (29), we also observe the similarity of B-machine with ν-SVM [15], where the optimization functional is given as

$$L = -\frac{1}{2}\alpha^T D(t) K D(t) \alpha \tag{48}$$

subject to $0 \leq \alpha_i \leq \frac{1}{\lambda}$, λ being a regularization parameter $\lambda \in [0, 1]$, and $\alpha^T t = 0$, $\sum \alpha_i \geq \nu$. In ν-SVM the quadratic optimization in ν-SVM does not contain any linear term as in standard SVM. The performance of ν-SVM [15] depends on two parameters namely ν and λ. However, in ν-SVM also, no box constraint is imposed on the parameters as it is in the case of B-machine.

4 Experimental Results

4.1 Nature of the Classifier

We first illustrate the B-vectors generated by B-machine for two-dimensional Gaussian parity problem. We select the Gaussian kernel of different sizes. First, in Figure 1, we illustrate the B-vectors for $2\sigma^2 = 1$. The B-vectors are marked by 'o', and the samples from two different classes are marked by 'x' and '.' respectively. We observe that even though we do not perform margin maximization as performed in SVM, the B-vectors concentrate towards the boundary between opposite classes. However, this behavior of the B-vectors is specific to the choice of kernel. In the next figure, we observe a different behavior of the B-vectors as we change the kernel width.

Figure 2 illustrates the change in the B-vectors as we change σ of Gaussian kernel as 0.2, 0.5, 1 and 2. We observe that for a low σ, the B-vectors are not necessarily concentrated near the opposite classes. This is due to the fact that B-machine does not use the principle of margin maximization, rather it finds B-vectors such that existence of other points becomes interpretable subject to a given margin. As we increase σ, the B-vectors starts getting concentrated towards

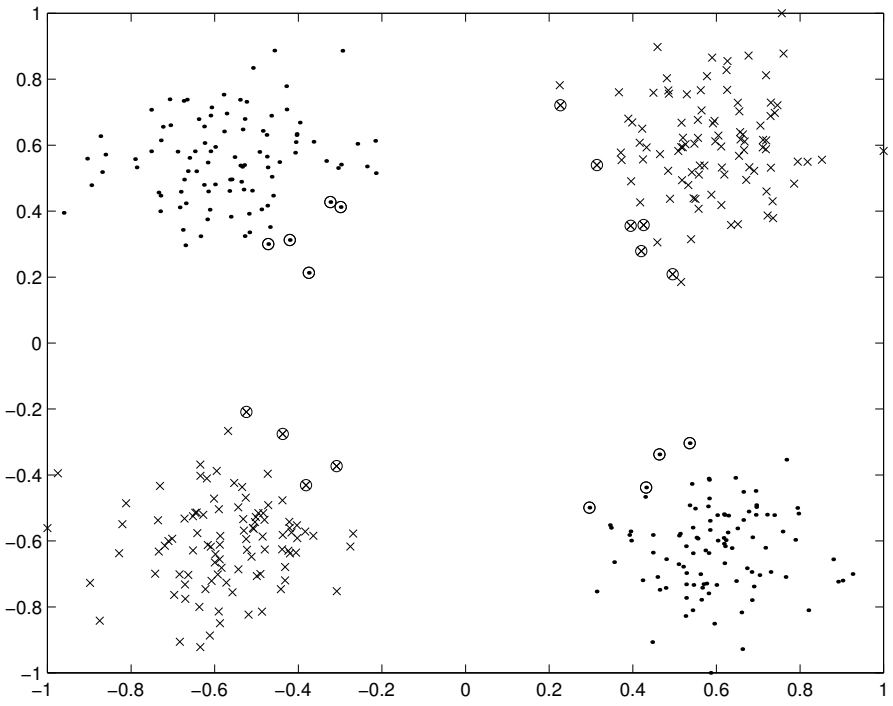

Figure 1: B-vectors obtained by the B-machine for a two-dimensional Gaussian parity problem. The two classes are represented by 'x' and '.' respectively. The B-vectors are marked by 'o'. The Gaussian kernel of the B-machine is chosen such that $2\sigma^2 = 1$.

the opposite class samples. This is due to higher interaction between samples from other classes due to larger width of the kernel function.

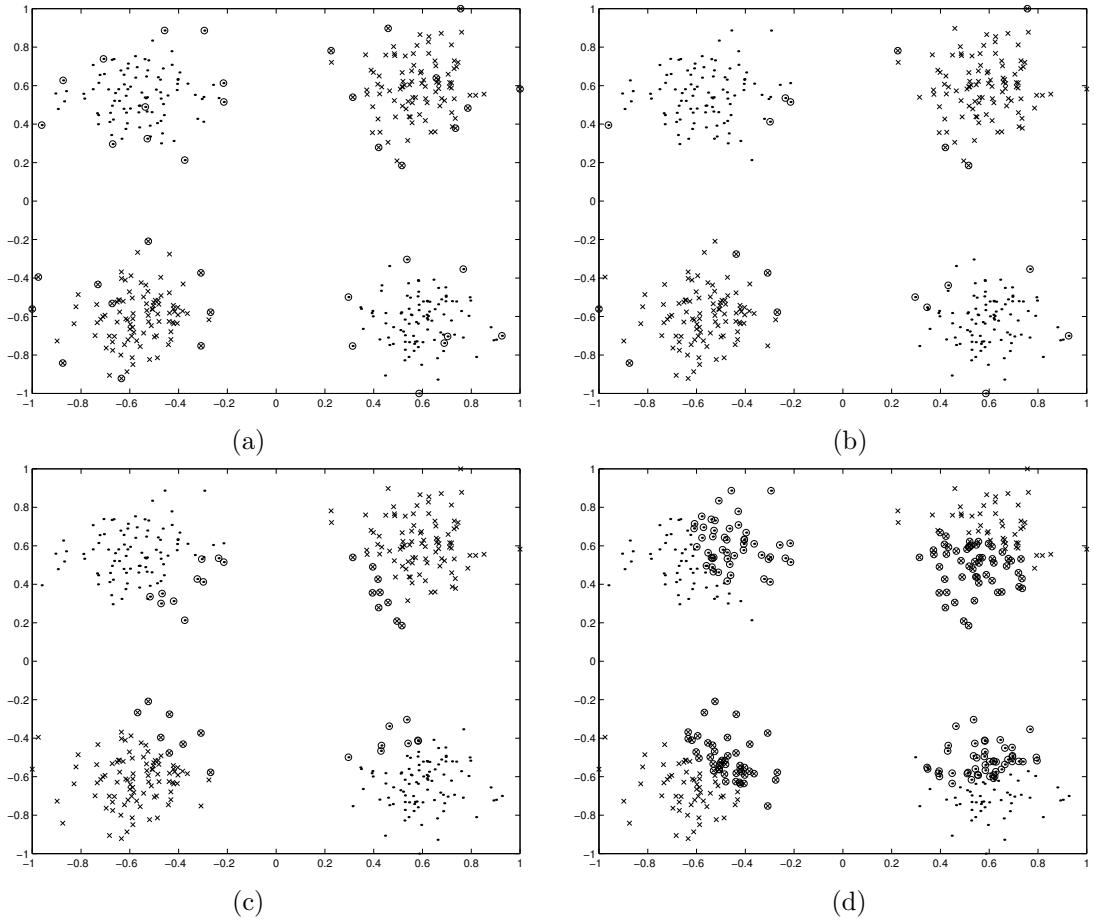

Figure 2: B-vectors obtained by B-machine for the same two-dimensional Gaussian parity problem as in Figure 1 with different kernel sizes. (a), (b), (c), and (d) illustrate the B-vectors for $\sigma = 0.2$, $\sigma = 0.5$, $\sigma = 1.0$ and $\sigma = 2.0$ respectively.

We now illustrate the effect of λ on the number of B-vectors. If we lower the margin λ then we observe that the number of B-vectors decreases and it increases with the increase in λ. This is due to the fact that if we reduce the margin, we require less number of B-vectors to support the existence of other samples from the same class subject to the margin. Figure 3 illustrates the B-vectors for the same Gaussian parity problem with $\lambda = 0.1$, 0.2, 0.5, and 1 respectively for $\sigma = 1$.

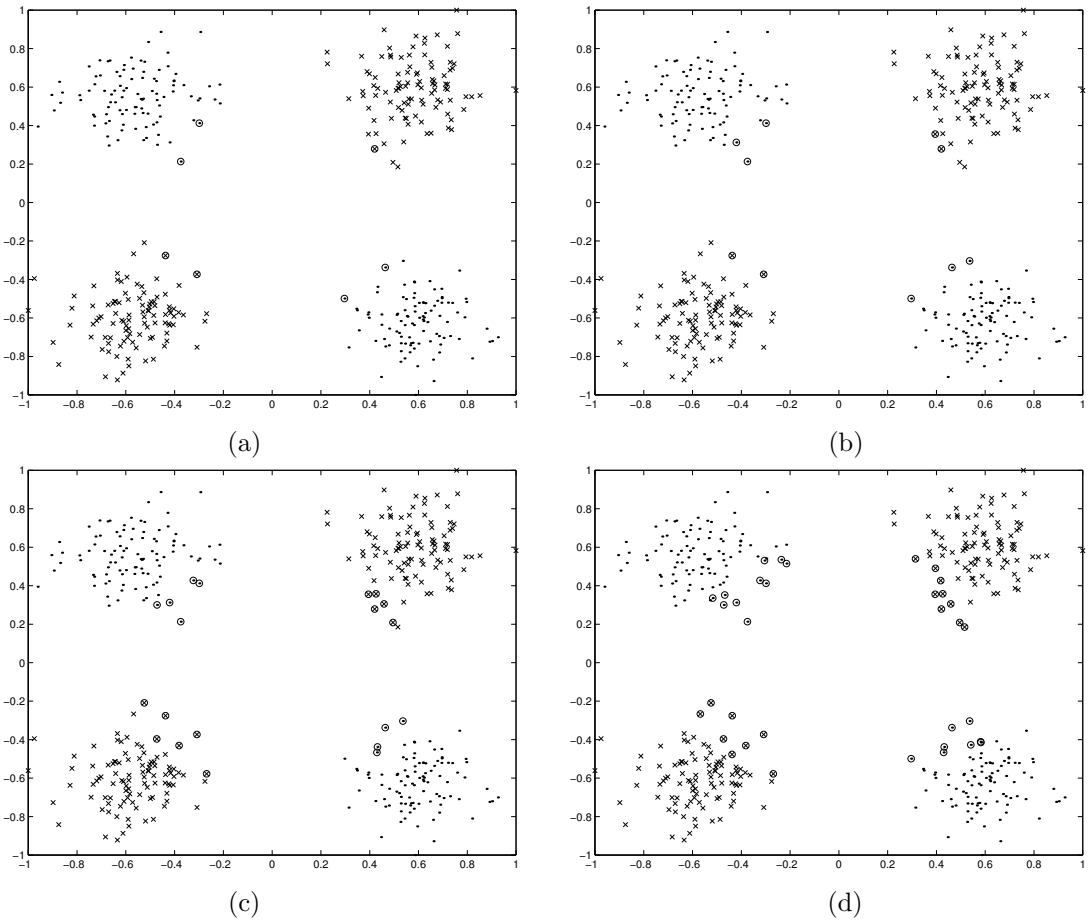

Figure 3: B-vectors obtained by B-machine for the same two-dimensional Gaussian parity problem as in Figure 1 for different margins (λ) with a fixed size of Gaussian kernel $\sigma = 1$. (a), (b), (c), and (d) illustrate the B-vectors for $\lambda = 0.1$, $\lambda = 0.2$, $\lambda = 0.5$, and $\lambda = 1.0$ respectively.

4.2 Selection of the Kernels

Apart from Gaussian kernels, we select two other kernel functions which are centrally peaked with longer tails in nature, such that the distant B-vectors have greater interaction between. The first one is given as

$$K_1(x,\mu) = \frac{1}{1+\left(\frac{\|x-\mu\|}{\sigma}\right)^2} \tag{49}$$

The kernel in Equation (49) is derived from the Cauchy distribution which has a long tail, and we call this kernel function as Cauchy kernel. Note that, Cauchy kernel has also been applied in the formation of sparse codes for natural scenes leading to a complete family of localized, oriented, bandpass receptive fields [37]. Cauchy kernel is continuous and it is differentiable except at the point $x = \mu$.

The second form of kernel is a mixture of Guassian and exponential kernel such that for smaller deviation it is squared loss and for larger deviation the loss is linear. In the second kernel function, we use Gaussian kernel near the center of the kernel (the logarithm of Gaussian is essentially squared deviation) and exponential decay away from the center (the logarithm of exponential represents linear deviation). The second kernel function is given as

$$K_2(x,\mu) = \begin{cases} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right) & \text{for } \|x-\mu\| \leq \epsilon \\ \exp\left(-\frac{\|x-\mu\|}{\sqrt{2}\sigma}\right) & \text{otherwise} \end{cases} \tag{50}$$

Note that, K_2 is continuous only for $\epsilon = \sqrt{2}\sigma$ and we choose the same value although in the framework of B-machine it is not necessary that the kernel function needs to be continuous. We refer to the kernel K_2 as the *Gaussian + Exponential* kernel which is not differentiable on the hyperplane $\|x-\mu\| = \sqrt{2}\sigma$. In the next section, we demonstrate the performance of B-machine with *Cauchy* kernel and *Gaussian + Exponential* kernel in addition to *Gaussian* kernel.

4.3 Data Sets

We demonstrate the effectiveness of our classifier on certain real-life data sets. We have considered the data sets available in the UCI machine learning repository [7]. Table 1 summarizes the data set description. We considered the data sets from UCI machine learning repository [7] mostly based on the fact that the data should not contain any missing variable and consist of only numeric attributes. Note that, the 'Iris' dataset that we used is the 'BezdekIris' data in the UCI repository. We modified the 'Ecoli' data originally used in [36] and later in [28] for predicting protein localization sites. The original dataset consists of eight different classes out of which three classes namely, outer membrane lipoprotein, inner membrane lipoprotein, and inner membrane cleavable signal sequence have only five, two, and two instances respectively. Since we report the 10-fold cross-validation score, we omitted samples from these three classes and report the results for the rest five different classes. Note that, in the original work [36] also, the results are not cross-validated. We normalize all input patterns such that any compponent of $\mathbf{x}$ lies in the range $[-1,+1]$.

Data Set	No. of Instances	No. of Features	No. of Classes
Indian diabetes (Pima)	768	8	2
Diagnostic Breast Cancer (Wdbc)	569	30	2
Prognostic Breat Cancer (Wpbc)	198	32	2
Liver Disease (Bupa)	345	6	2
Flower (Iris)	150	4	3
Bacteria (Ecoli)	326	7	5
AI (Monks1)	556	6	2
AI (Monks2)	601	6	2
AI (Monks3)	554	6	2

Table 1: Description of the pattern classification datasets obtained from the UCI machine learning repository [7].

4.4 Results

We report the 10-fold cross-validation performance of B-machine for Gaussian kernel, Cauchy kernel, and Gaussian+Exponential kernel respectively. In computing the 10-fold cross-validation scores, we randomly partitioned the data into 10 disjoint groups, and used 90% (i.e., 9 disjoint groups) samples as the training sample and the rest 10% samples as the test samples, and iterated this procedure using each group in the partition as the test set and the rest as the training set. We used 10 such trials of random partitioning the data. For each trial, we compute the mean and variance of the classification score over the ten different test sets, and then compute the overall mean and variance scores over the ten different trials. We performed this random partitioning of the dataset and averaged over ten different trials in order to reduce any inherent bias generated due to sample ordering present in the data set. We also compare the performance of B-machine with that of SVM using Gaussian kernel and third degree polynomial kernels. In addition we compare the performance with the LSSVM using Gaussian kernels. In comparing the performance, we use the same training set and the test set for each trial and each fold throughout for all these classifiers.

Table 2 summarizes the 10-fold cross-validation score of B-machine on the datasets in Table 1 for Gaussian kernels, Cauchy kernels, and Gaussian+Exponential kernels. We report the mean accuracy as well as the standard deviation in the accuracy as the indicator of significance. As a comparison, we also provide the same scores of SVM for Gaussian kernel and polynomial kernel, and LSSVM with Gaussian kernel in Table 2. For implementing SVM, we used the publicly available code in [14]. Note that, WEKA [22, 69] also provides an implementation of SVM, particularly the sequential minimal optimization (SMO) [41]. However, SMO algorithm as provided in WEKA cannot handle more than two class labels. We therefore, implememted the

multi-class SVM using the software provided in [14]. We implemented LSSVM using the Matlab code available in [40]. In implementing LSSVM for multiclass classification (such as 'Iris' and 'Ecoli'), we used 'minimum output coding' as provided in [14]. Note that for B-machine, we used 'one-against-all' classification scheme, however, for LSSVM we observed that 'minimum output coding' provides much better scores as compared to 'one-vs-all' coding. We also tested LSSVM using 'error-correcting-output-coding' as provided in [14]. However, 'minimum output coding' provided the best results for the datasets that we considered.

The performance of SVM depends on the choice of the kernel and the constant C as in Equation 30. For LSSVM also, the performance is dependent on the kernel width (Gaussian kernel), and the regularization parameter (γ) used for performing the regression. Similarly, the performance of B-machine is dependant on the choice of kernel and the margin λ. In Table 2, we have reported the best performances of all the classifiers namely, SVM, LSSVM, and B-machine. Here we mention that we obtained the best scores of SVM by searching over different possible configurations of (σ, C). A better way of searching an optimum configuration for (σ, C) is provided in [26], which is a smarter search as compared to exhaustive search. For B-machine also, we report the best results for all three types of kernels by searching over different possible combinations of (σ, λ). We obtained the set of parameters for LSSVM in the same way, i.e., searching over different possible combinations. Note that, in Table 2, we provide one set of parameters for each classifier and each dataset. However, we observed that all these three classifiers are able to obtain similar scores over a significantly large neighborhood of the parameter values as provided in Table 2. In other words, these classifiers are not very sensitive on the choice of parameter values over a neighborhood of the candidate values that we report, although the performance can significantly differ if the choice of paramter is widely different.

As we observe from Table 2, B-machine outperforms SVM in terms of 10-fold cross-validation score on several datasets but not all that we have used. In particular, both SVM and LSSVM outperform the proposed B-machine on the artificial datasets 'Monks1', 'Monks2', and 'Monks3'. For the real-life datasets like 'Pima', 'Bupa', 'Wpbc', 'Iris' and 'Ecoli', B-machine outperforms SVM and LSSVM, particularly when we observe the performance of B-machine with Cauchy kernel. For 'Wdbc' dataset, the best performance of SVM is marginally better than that of B-machine although LSSVM is much worse than the other two. Interestingly, we observe that the performance of B-machine often improves significantly with long-tailed kernels such as Cauchy kernel.

In order to establish the comparison between these classifiers in a more quantitiave way, we performed resampled paired t-test as provided in [18]. As described in [18], we randomly divided the dataset such that two-thrid of the dataset constituted a training set, and the rest one-third constituted a test set. We then used the same training set and test set for all the variants of the three classifiers. We then computed the rate of misclassification on the test set. We repeated this experiment for 30 different trials, where in every trial we randomly partitioned the data and computed the rate of misclassification by each classifier. For every trial, we computed the difference in the rate of misclassification. For example, if c_1 and c_2 are two different classifiers with rates of misclassification $p_1^{(i)}$ and $p_2^{(i)}$ respectively for trial i, then the difference in misclassification

Classifier	Pima	Bupa	Wdbc	Wpbc	Iris	Ecoli	Monks1	Monks2	Monks3
SVM	76.88	69.39	98.03	80.78	96.13	88.17	90.43	92.07	96.40
(Gaussian)	($\pm$4.74)	($\pm$6.53)	($\pm$1.80)	($\pm$6.15)	($\pm$4.87)	($\pm$4.53)	($\pm$3.30)	($\pm$3.63)	($\pm$2.60)
(σ, C)	$(1, 2)$	$(0.3, 1)$	$(2, 20)$	$(1, 2)$	$(1, 1)$	$(0.5, 1)$	$(2, 1)$	$(0.5, 1)$	$(2, 1)$
SVM	75.58	71.72	96.20	74.39	96.13	87.25	100	99.07	97.97
(Polynomial)	($\pm$4.74)	($\pm$6.40)	($\pm$2.26)	($\pm$8.72)	($\pm$5.06)	($\pm$4.88)	(0)	($\pm$1.49)	($\pm$1.95)
(C)	(1)	(5)	(1)	(1)	(5)	(3)	(3)	(10)	(1)
LSSVM	76.28	68.87	93.78	78.13	84.53	77.40	99.46	95.34	96.97
(Gaussian)	($\pm$5.18)	($\pm$7.21)	($\pm$2.93)	($\pm$3.93)	($\pm$7.90)	($\pm$5.54)	($\pm$0.83)	($\pm$2.78)	($\pm$2.39)
(σ^2, γ)	$(5, 0.1)$	$(1, 2)$	$(4, 0.3)$	$(5, 4)$	$(0.2, 15)$	$(1, 10)$	$(2, 20)$	$(2, 20)$	$(5, 20)$
BM	77.51	72.42	97.68	81.05	95.87	87.93	85.26	89.46	96.14
(Gaussian)	($\pm$5.41)	($\pm$6.37)	($\pm$1.65)	($\pm$7.11)	($\pm$5.02)	($\pm$4.79)	($\pm$4.84)	($\pm$3.96)	($\pm$2.56)
(σ, λ)	$(5, 0.2)$	$(2, 0.2)$	$(1.5, 1)$	$(5, 0.6)$	$(1, 0.4)$	$(1, 0.5)$	$(0.5, 0.4)$	$(0.5, 0.3)$	$(2, 0.4)$
BM	76.28	73.17	97.61	80.83	97.07	88.77	94.38	80.64	96.29
(Gauss+Expo)	($\pm$4.78)	($\pm$6.59)	($\pm$1.76)	($\pm$6.35)	($\pm$4.70)	($\pm$4.56)	($\pm$2.46)	($\pm$5.23)	($\pm$2.42)
(σ, λ)	$(2, 0.2)$	$(2, 0.6)$	$(0.2, 0.2)$	$(3, 1)$	$(1, 0.6)$	$(2, 0.4)$	$(1, 0.2)$	$(1, 0.3)$	$(3, 0.6)$
BM	77.10	72.77	97.81	82.33	97.60	89.16	99.06	92.48	96.54
(Cauchy)	($\pm$5.40)	($\pm$6.55)	($\pm$1.65)	($\pm$6.51)	($\pm$4.21)	($\pm$4.37)	($\pm$1.56)	($\pm$3.43)	($\pm$2.55)
(σ, λ)	$(5, 0.2)$	$(3, 0.4)$	$(2, 0.7)$	$(3, 0.5)$	$(2, 0.3)$	$(3, 0.8)$	$(1, 0.2)$	$(1, 0.5)$	$(3, 0.5)$

Table 2: Classification performance in terms of the 10-fold cross-validation scores on the datasets described in Table 1. Classification performance summarizes the mean classification accuracy and the standard deviation of the accuracies as a significance check. In B-machine, we use three different types of kernels namely, Gaussian kernel, Gaussian kernel augmented with exponential kernels, and Cauchy kernels. Note that, the later two do not follow the Mercer condition. For SVM, we use Gaussian and third degree polynomial kernels, and for LSSVM we use Gaussian kernel. All scores are obtained with normalized input such that each variable in each pattern is in the range $[-1, +1]$. For each variant of B-machine, SVM, and LSSVM, the best results obtained as determined by the cross-validation score is reported. We also show the corresponding parameter values i.e., (σ, C) for SVM, (σ^2, γ) for LSSVM, and (σ, λ) for BM. Certain candidate parameter values for which we obtained the best results are reported. However, in each case SVM, LSSVM, and BM produce the best results over a neighborhood of the parameter values as shown here. We implemented SVM with Gaussian and polynomial kernels using the Matlab code available in [14], and LSSVM with Gaussian kernel using the code available in [40].

is $p^{(i)} = p_1^{(i)} - p_2^{(i)}$, and the statistic is obtained as

$$t = \frac{\bar{p}.\sqrt{N}}{\sqrt{\frac{\sum_{i=1}^{N}(p^{(i)}-\bar{p})^2}{N-1}}} \tag{51}$$

where $N(=30)$ is the number of trials, and $\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p^{(i)}$ is the average difference in the misclassification rate. Evidently, if $t < 0$ then classifier c_1 is better than the classifier c_2 and vice-versa. The significance to which the two classifiers are different is obtained from the measure t. As provided in [18] (as obtained from the Student's cumulative t-distribution function with $N-1(=29)$ degrees of freedom), if $|t| > 2.0452$ then the classifiers are different from each other with a confidence level of 97.5%, and the classifiers are different with a 95% confidence level if $|t| > 1.6991$.

Table 3 summarizes the t-statistic as obtained by the resampled paired t-test. As we observe from Table 3 that for the 'Monks1' dataset the best B-machine (with Cauchy kernel) falls short of best SVM with polynomial kernel with a t-measure 10.77 which shows that SVM is better than the B-machine with a probability very close to unity. Similarly, LSSVM is also better than B-machine with a probability very close to unity. We observe the similar performance for the 'Monks2' dataset where SVM and LSSVM are definitely better than B-machine with probability very close to unity. For 'Monks3' dataset, SVM with polynomial kernel (the best variant) is better than B-machine with Cauchy kernel with a confidence level 69.65% ($t = 0.52$) whereas B-machine is better than LSSVM with a confidence level of 80.96% ($t = -0.89$). In 'Wdbc' dataset, the best variant of B-machine (Gaussian kernel) is worse than the best SVM (Gaussian kernel) with a confidence 58.24% ($t = 0.21$), however, the B-machine performs significantly better (with a probability very close to unity) than LSSVM. In all other datasets, we observe that the best variant of B-machine significantly outperforms the best variant of SVM and LSSVM.

5 Discussion and Summary

We presented a kernel machine called B-machine which uses quadratic loss funcitonal with box constraint on the parameter values. B-machine is related to various existing "least-square" kernel machines such as adaptive ridge regression, regularized least square classifier, least square support vector machine, LASSO (least absolute shrinkage and selection operator), and generalized LASSO by the fact that these existing kernel machines are also motivated by the quadratic loss functional. However, B-machine uses box constraint on the parameter values which makes it different from the existing ones. Interestingly, almost all existing quadratic loss kernel machines were motivated from the regression framework and therefore use Gaussian prior on the parameter values except the generalized LASSO which was motivated from a more robust Huber loss. B-machine on the contrary is motivated directly from the principle of parsimony or Occam's razor. We modified the principle of parsimony in the context of classification to state that "Minimize the superfluity in the outcome with a given margin". We discussed the meaning of 'outcome' as a soft-indicator function $g(x, \alpha)$ depending on the model α, and the outcome is interpreted as the final output $f(x, \alpha) \in \{-1, +1\}$ for any binary classifier. We viewed the redundancy (as stated in the original

Classifier Pair	Pima	Bupa	Wdbc	Wpbc	Iris	Ecoli	Monks1	Monks2	Monks3
BM (Gauss) SVM (Gauss)	-3.94	-7.33	0.21	-3.33	-3.26	2.37	8.60	8.31	3.61
BM (Gauss) SVM (Poly)	-7.0	-3.81	-6.93	-9.59	-2.72	-2.19	29.62	9.84	2.82
BM (Gauss) LSSVM	-4.27	-8.10	-14.73	-4.75	-17.83	-12.10	26.13	14.32	3.88
BM (GaussExpo) SVM (Gauss)	1.73	-8.00	1.48	-0.41	-5.43	-2.54	-6.44	25.43	5.63
BM (GaussExpo) SVM (Poly)	-2.03	-4.61	-4.91	-6.41	-4.63	-8.53	27.21	25.07	2.93
BM (GaussExpo) LSSVM	1.52	-8.56	-14.11	-2.29	-17.81	-13.34	11.13	26.31	3.61
BM (Cauchy) SVM (Gauss)	-4.84	-6.97	1.09	-2.71	-4.82	-3.25	-16.59	-4.74	0.37
BM (Cauchy) SVM (Poly)	-7.4	-3.49	-5.38	-7.45	-4.85	-6.56	10.77	5.57	0.52
BM (Cauchy) LSSVM	-4.9	-7.91	-14.68	-3.90	-17.94	-13.79	6.89	8.04	-0.89

Table 3: Resampled paired t-test scores in terms of t-statistic comparing the three variants of B-machine with the two variants of SVM and LSSVM on the datasets described in Table 1.

principle of parsimony) as the 'superfluity in the outcome' and measured the superfluity with respect to a given margin λ, the margin λ being viewed as a regularization parameter. Considering both empirical error and superfluity together we came up with the quadratic loss. We also discussed that the B-machine is designed based on the difference between the class posteriors (which is the outcome of the classifier) subject to the modified form of the principle of parsimony.

The formulation of the proposed classifier can also be viewed from the Parzen window estimation [39] of the class-conditional densities. In Parzen window estimate, the density $p(x)$ is nonparametrically estimated as

$$p(x) = \frac{1}{NV} \sum_{i=1}^{n} cK(x, x_i) \tag{52}$$

V being the volume spanned by the kernel $K(.)$, and N being the number of samples. In a Bayes classifier, samples are assigned the class labels based on the class posteriors such that if

$$P_1 p(x|\omega_1) \geq P_2 p(x|\omega_2) \tag{53}$$

then x is assigned a classlabel $+1$ else -1, where ω_1 and ω_2 represent the classes labeled as $+1$ and -1 respectively. P_1 and P_2 are the priors, and $p(x|\omega)$ represents the respective class conditional density. Letting the priors $P_1 = N_1/N$ and $P_2 = N_2/N$, N_1 and N_2 being the respective number of training samples in class ω_1 and ω_2 respectively such that $N = N_1 + N_2$, we can express the classifier in the form of the difference between class posteriors (from Equations (52) and (53)) as follows. If

$$\sum_{i=1}^{n} K(x, x_i) t_i \geq 0 \tag{54}$$

then x is assigned a class label $+1$ else -1. In Parzen classifier, we consider the contribution of each training sample to be equal in computing the class posterior difference. From classification point of view, we need to compute the difference between class posteriors "to the extent" such that the class labels are attached "properly". We therefore differentiate the effects of observations with a factor α_i as in Equation (26) such that the class labels are attached to the samples "properly to the extent" that the difference between the class posteriors is subjected to a given margin λ. In the present context, the difference between the class posteriors represents the outcome (Equation (26)) of the classifier. The term "to the extent" is explained from the modified form of the principle of parsimony and quantified with a loss L_2 in Equation (23); and the term "properly" is quantified by the empirical error L_1 as in Equation (22). We minimize the loss in Equation (28) subject to the constraints $0 \leq \alpha_i \leq 1$ for all i. Note that, the loss in Equation (28) can also be minimized in a linear regression framework considering Gaussian priors for αs. However, that does not guarantee $\alpha \geq 0$, which means that there can be negative contribution from training samples in computing the difference in posterriors. In our framework, we differentiate between the training samples such a way that they either make positive contribution or zero contribution in computing the difference between class posteriors subject to a given margin λ.

The basic difference between B-machine and SVM is that the B-machine does not perform any margin maximization as it is performed in SVM with respect to the optimal separating hyperplane in some higher dimensional space. Rather the B-machine minimizes the superfluity in the outcome (in terms of the difference between class posteriors) with a given margin λ. The principle of parsimony has been addressed in the literature of machine learning with two major interpretations namely, minimum description length (MDL) and structural risk minimization (SRM). Both MDL and SRM measures the redundancy (as in the principle of parsimony) in terms of the model complexity – information theoretic complexity in the case of MDL and VC-dimension in the case of SRM. In our interpretation of the principle, we did not consider the model complexity rather we considered the superfluity in the outcome. Thus the modified principle of parsimony that we presented does not involve any estimation of the model complexity, rather it involves suitable interpretation of the outcome of a model such as the soft-indicator function that we used in the context of a binary classifier. Although in the present context of pattern classification, we used this modified principle to design a quadratic loss kernel machine with box constrained parameters, this principle can possibly be applied to other learning machines also such as neural networks [3, 27]. We considered only symmetric square loss in designing the objective functional. However, other form of loss functional (not necessarily symmetric) can also be investigated for this purpose. The symmetric square loss leads to a quadratic loss functional that can be handled in a quadratic programming framework. The use of other kind of loss functionals may require more complex and sophisticated methods of parameter estimation.

We observed that B-machine outperforms SVM on several real-life datasets. B-machine provides us some flexibility of choosing kernels as it does not require any Mercer condition to be satisfied as it is in the case of support vector machines. We also observed that the incorporation of centrally peaked long-tailed kernel functions can enhance the performance of the classifier on certain datasets. In addition to Gaussian kernel, we used two other kernel functions namely, Cauchy kernel and a hybrid of Gaussian and exponential kernels. We observed that these long-tailed kernel functions provide better performance on several real-life datasets as compared to

simple Gaussian kernel due to the long-range interaction between the B-vectors.

It may be mentioned here that the formulation of SVM (Equation (12)) minimizes the objective functional under a constraint $\alpha_i t_i = 0$ which helps in avoiding the label bias problem. In other words, if there is a large difference in the number of training samples of different classes then the constraint helps in imposing a balance. The formulation of B-machine is not subjected to any such explicit balance constraint. Rather it introduces the linear term $\sum_i \alpha_i t_i (\sum_j K(x_i, x_j) t_j)$ in the objective functional (Equation (29)). Due to the explicit balance constraint in the formulation, SVM is expected to perform better in terms of generalization for the unbalanced datasets, although we observe that B-machine outperforms SVM by a significant margin (Table 3) on one such dataset ('Ecoli' dataset). One disadvantage of the B-machine is that it produces relatively non-sparse solution as compared to SVM and other least square kernel machines using Hilbert kernels. This leads to relatively slower training and testing processes for B-machine. In this respect, further investigation can be performed towards obtaining better variants of B-machine. With respect to the run-time memory requirements, on-line learning of B-machine instead of batch-mode learning for larger datasets can also be investigated as a scope of future study.

Acknowledgements: The author takes the privilege to gratefully acknowledge Prof. Shun-ichi Amari of the RIKEN Brain Science Institute, Saitama, Japan for his thoughtful and constructive suggestions which helped the author to improve this article.

References

[1] H. Akaike. A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, 30A:9–14, 1978.

[2] S. Amari. Theory of adaptive pattern classifiers. *IEEE Trans.*, EC-16:299–307, 1967.

[3] P. L. Bartlett. For valid generalization the size of the weights is more important than the size of the network. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 134–140. The MIT Press, 1997.

[4] J. Basak. Online adaptive decision trees. *Neural Computation*, 16:1959–1981, 2004.

[5] J. Basak. Online adaptive decision trees: Pattern classification and function approximation. *Neural Computation*, 18:2062–2101, 2006.

[6] T. D. Bie, G. R. G. Lanckriet, and N. Cristianini. Convex tuning of the soft margin parameter. Technical Report UCB/CSD-03-1289, University of California, Computer Science Division (EECS), Berkeley, California, USA, 2003.

[7] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. Available at: *http://www.ics.uci.edu/m̃learn/MLRepository.html*.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

[9] L. Breiman. Stacked regressions. *Machine Learning*, 24:49–64, 1996.

[10] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26:801–824, 1998.

[11] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Chapman & Hall, New York, 1983.

[12] P. J. Brown and J. V. Zidek. Adaptive multivariate ridge regression. *The Annals of Statistics*, 8:64–74, 1980.

[13] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[14] S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. SVM and kernel methods matlab toolbox. Perception Systmes et Information, INSA de Rouen, Rouen, France, 2005. Available at: *http://asi.insa-rouen.fr/ãrakotom/toolbox*.

[15] P.-H. Chen, C.-J. Lin, and Schölkopf. A tutorial on nu-support vector machines. *Applied Stochastic Models in Business and Industry*, 21:111–136, 2005.

[16] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:1–25, 1995.

[17] T. M. Cover. Learning in pattern recognition. In S. Watanabe, editor, *Methodologies of Pattern Recognition*, pages 111–132. Academic Press, New York, 1969.

[18] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10:1895–1923, 1998.

[19] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley, New York, 2001.

[20] B. Efron. Estimating the error rate of a prediction rule : Improvement on cross-validation. *Journal of the American Statistical Association*, 78(382):316–330, 1983.

[21] J. H. Friedman. On bias, variance, 0/1-loss, and curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.

[22] S. Garner. Weka: The waikato environment for knowledge analysis. In *Proc. of the New Zealand Computer Science Research Students Conference*, 1995.

[23] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[24] S. R. Gunn. Support vector machines for classification and regression. Technical Report http://www.ecs.soton.ac.uk/šrg/publications/pdf/SVM.pdf, Faculty of Engineering, Science and Mathematics, School of Elecetronics and Computer Sceince, University of Southampton, 1998.

[25] P. E. Hart. The condensed nearest neighbor rule. *IEEE Trans. Information Theory*, IT-14:515–516, 1968.

[26] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal Machine Learning Research*, 5:1391–1415, 2004.

[27] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, 1999.

[28] P. Horton and K. Nakai. A probablistic classification system for predicting the cellular localization sites of proteins. In *Intelligent Systems in Molecular Biology*, pages 109–115, St. Louis, USA, 1996.

[29] D. W. J. Hosmer and S. Lemeshow. *Applied Logistic Regression (2nd Ed.)*. John Wiley, USA, 2000.

[30] A. K. Jain, R. C. Dubes, and C. Chen. Bootstrap techniques for error estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-9(5):628–633, 1987.

[31] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[32] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[33] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML96)*, 1996.

[34] D. W. Marquardt. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, 12:591–612, 1970.

[35] N. Murata, S. Hoshizawa, and S.-I. Amari. Learning curves, model selection and complexity in neural networks. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems (NIPS92)*, volume 5, pages 607–614. MIT Press, Cambridge, MA, 1993.

[36] K. Nakai and M. Kanehisa. Expert sytem for predicting protein localization sites in gram-negative bacteria. *PROTEINS: Structure, Function, and Genetics*, 11:95–110, 1991.

[37] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[38] M. Osborne, B. Presnell, and B. Turlach. On the LASSO and its dual. *Journal Comput. Graphical Statist.*, 9:319–337, 2000.

[39] E. Parzen. On the estimation of a probability density function and the mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[40] K. Pelckmans, J. A. K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, J. Vandewalle. LS-SVMlab Toolbox Users Guide. Technical Report 02-145, Department of Electrical Engineering, ESAT-SCD-SISTA, Katholieke Universiteit Leuven, Belgium, 2003. Available at http://www.esat.kuleuven.ac.be/sista/lssvmlab/.

[41] J. C. Platt. Sequential minimal optimization : A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, USA, 1998.

[42] J. R. Quinlan. *Programs for Machine Learning*. Morgan Kaufmann, San Fransisco, CA, USA, 1993.

[43] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[44] R. Rifkin, G. Yeo, and T. Poggio. Regularized least square classification. In J.A.K. Suykens, G. Horvath, S. Basu, C. Micchelli, and J. Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications, NATO Science Series III: Computer and System Sciences*, volume 190, pages 131–153. IOS Press, Amsterdam, 2003.

[45] J. Rissanen. Modeling by shortest path description. *Automatica*, 14:465–471, 1978.

[46] J. Rissanen. *Stochastic Complexity and Statistical Inquiry*. World Scientific, 1989.

[47] V. Roth. The generalized LASSO. *IEEE Trans. Neural Networks*, 15(1):16–28, 2004.

[48] C. Saunders, A. Gammerman, and V. Vovk. Ridge regression learning algorithm in dual variables. In *Proceedings of the 15th International Conference on Machine Learning (ICML98)*, 1998.

[49] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336, 1999.

[50] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[51] J. Shao. Linear model selection via cross-validation. *Journal of the American Statistical Association*, 88(422):486–494, 1993.

[52] J. Shawe-Taylor and P. L. Bartlett. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5):1926–1940, 1998.

[53] J. Shawe-Taylor and N. Cristianini. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, UK, 2000.

[54] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC-TR-98-030, NeuroCOLT, Royal Holloway College, University of London, UK, 1998.

[55] P. Smyth and D. Wolpert. An evaluation of linearly combining density estimators via stacking. *Machine Learning*, 36:59–83, 1999.

[56] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36:111–147, 1974.

[57] M. Stone. Asymptotics for and against cross-validation. *Biometrika*, 64:29–35, 1977.

[58] J. A. K. Suykens and J. Vandewalle. Least square support vector machine classifiers. *Neural Processing Letters*, 9:293–300, 1999.

[59] W. M. Thorburn. The myth of Occam's razor. *Mind*, 27:345–353, 1918.

[60] R. Tibshirani. Bias, variance and prediction error for classification rules. Technical Report http://www.utstat.toronto.edu/~tibs, Dept. of Statistics, University of Toronto, Canada, 1996.

[61] R. Tibshirani. Regression shrinkage and selection via lasso. *Journal Royal Statistical Society, Series B*, 58:267–288, 1996.

[62] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. W. H. Winston, Washington, D. C., 1977.

[63] M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal Machine Learning Research*, 1:211–244, 2001.

[64] Y. Tsypkin. *Foundation of the Theory of Learning Systems*. Academic Press, New York, 1973.

[65] T. Van Gestel, J. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and Vandewalle J. Benchmarking least squares support vector machine classifiers. *Machine Learning*, 54:5–32, 2004.

[66] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.

[67] V. Vapnik. *Statistical Learning Theory*. Springer-Verlag, New York, USA, 1998.

[68] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moskow, Russian edition, 1974.

[69] I. H. Witten and E. Frank. *Data mining : Practical machine learning tools and techniques with Java implementations (Chapter 8)*. Morgan-Kauffmann, USA, 2000.

[70] P. Zhang. On the distributional properties of model selection criteria. *Journal of the American Statistical Association*, 87(419):732–737, 1992.