IBM Research Report

Pattern Classification using the Principle of Parsimony

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

Principle of parsimony or the Occam's razor is a key principle to improve the generalization performance in pattern classification. The principle is essentially equivalent to reducing the variance of a classifier at the cost of increased boundary bias. In this paper, we provide a quantitative way to express this principle in terms of the outcome of a classifier instead of explicitly regularizing the model complexity in terms of model parameters. We then use this principle to design a new kernel machine and a modified k-nearest neighbor algorithm. Experimentally we validate the performance of these two classifiers over real-life datasets. We also discuss the relationship of the proposed kernel machine with several other existing kernel machines.

1 Introduction

In pattern classification (Tsypkin, 1973; Duda, Hart, & Stork, 2001; Vapnik, 1995), 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, and 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). Assuming the Gaussian error model for the loss function (Amari, 1967), 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 $f(.)$. In general, minimization of the empirical risk functional with

increased model complexity increases the generalization error i.e., the error on the unobserved data. (Duda et al., 2001; Friedman, 1997; Tibshirani, 1996a; Geman, Bienenstock, & Doursat, 1992; Kohavi & Wolpert, 1996).

Various principles are applied to improve the generalization performance which include (Cover, 1969; Stone, 1974, 1977; Jain, Dubes, & Chen, 1987; Efron, 1983; Shao, 1993; Kohavi, 1995; Zhang, 1992; Smyth & Wolpert, 1999; Breiman, 1996a, 1996b, 1998; Schapire & Singer, 1999)). In general, the basic spirit of all these approaches is to reduce the unnecessary model complexities while performing the classification by minimizing the empirical risk functional. This basic principle of reducing unnecessary complexity is often referred to as Occam's razor or the principle of parsimony (Duda et al., 2001). The different forms for realizing this principle include Tikhonov regularization (Tikhonov & Arsenin, 1977; Tsypkin, 1973), minimum description length principle (Rissanen, 1978), and structural risk minimization (Vapnik & Chervonenkis, 1974; Vapnik, 1995). For example, in Tikhonov regularization, normally an additional cost dependent on the model complexity is added to the empirical loss functional. Such a cost functional imposes a smoothing on the classification boundary. Minimum description length (MDL) principle is based on the information theoretic analysis of the randomness in the concept. As discussed in (Duda et al., 2001), according to MDL, the sum of the classifier's algorithmic complexity and the description of the training data with respect to the classifier is minimized. Different variations of the MDL principle such as Akaike information criteria (AIC) (Akaike, 1978), Bayes information criteria (BIC) (Schwarz, 1978) and network information criteria (NIC) (Murata, Hoshizawa, & Amari, 1993) 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 (Vapnik & Chervonenkis, 1974; Vapnik, 1995). 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.

As articulated in (Friedman, 1997), the overall classification error as measured by the deviation of a model's performance with respect to a theoretically optimal classifier (Bayes classifier), can be expressed as a nonlinear relationship between the boundary bias and the variance. The performance of a classifier is more dependent on the variance component and a reduced variance can enhance the classification performance. However, the reduction in variance can cause an increase in the boundary bias due to over-smoothing. The increase in boundary bias does not necessarily deteriorate the generalization performance of a classifier so long as the sign of boundary bias remains consistent. In the existing realizations of the principle of parsimony, essentially the variance is reduced by increasing the bias, and the bias is increased to the extent such that the sign remains consistent.

In this paper, we propose an alternate approach of reducing the variance component and provide a simple quantitative way of expressing the principle of parsimony for classification. According to this principle, we essentially reduce the superfluity in the outcome (variance) constrained by a certain regularization parameter. We first articulate the principle and then explain the principle in the perspective of the bias-variance decomposition. We then design a kernel based classifier using this principle. We also use the same principle for designing a modified k-nearest neighbor classifier. For the kernel based classifier, we draw the relationships with other existing

kernel machines. We show the effectiveness of this principle by demonstrating the classification performance of our kernel machine and the enhanced k-nearest neighbor classifier.

2 Pattern Classification with the Principle of Parsimony

2.1 Principle

We present an interpretation of the principle of parsimony which we use in the subsequent design of our classifier. We state the principle as

Minimize the superfluity in the outcome with a given margin.

Here we use this principle to regularize the outcome of a classifier which in turn regularizes the model complexity. Using this principle we design classifiers without directly considering the minimization of model complexity in the loss functional. Rather we design classifiers such that the outcomes of the models follow this principle. At a later part of this paper, we show that the principle can be connected to the model complexity under certain constraints. Note that, this principle is not new; however, we express the principle of regularization in this form which helps us in designing two different forms of classifiers. We first explain the notion of the "superfluity" that we use in the sequel restricting ourselves to two-class classification tasks. We then draw the connection with the bias-variance perspective of the same principle. Subsequently, we design two different classifiers using the same principle.

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

$$L(t, f(x, \alpha)) = \begin{cases} 0 & \text{if } t = f(x, \alpha) \\ 1 & \text{if } t \neq f(x, \alpha) \end{cases} \tag{5}$$

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

$$f(x, \alpha) = sign(g(x, \alpha)) \tag{6}$$

where $sign(.)$ is $+1$ or -1 if the argument is positive or negative respectively.

For example, in a k-nearest neighbor classifier (Hart, 1968; Duda et al., 2001), 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

$$g(x, \alpha) = \left(\frac{2N_+(x)}{k} - 1 \right) \tag{7}$$

where $N_+(x)$ is the number of positive training samples in $\Omega_k(x)$. 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 (Breiman, Friedman, Olshen, & Stone, 1983; Quinlan, 1993), 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

$$g(x, \alpha) = \left(\frac{2N_+(l_T(x))}{N_+(l_T(x)) + N_-(l_T(x))} - 1 \right) \tag{8}$$

where $N_+(l_T(x))$ and $N_-(l_T(x))$ are respectively the number of positive and negative training samples allocated to the leaf node $l_T(x)$. In the case of support vector machines (and in various forms of kernel machines), the soft-indicator function $g(x, \alpha)$ is defined as

$$g(x, \alpha) = \sum_i \alpha_i K(x, x_i) t_i + b \tag{9}$$

In the case of parametric classifiers such as logistic regression models (Hosmer & Lemeshow, 2000) and semi-parametric classifiers such as neural networks (Haykin, 1999) and hierarchical mixture of experts (Jordan & Jacobs, 1994), 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. According to the principle, we fix a margin (regularization parameter) λ and minimize the superfluity in the outcome $g(x, \alpha)$ with the given margin. 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. For a training sample x with a class label $+1$ $(t(x) = +1)$ there is an empirical error in matching the class label if $g(x, \alpha) < \lambda$. Similarly, there is an empirical error if $g(x, \alpha) > -\lambda$ and $t(x) = -1$. We can express the empirical error as

$$L_1(x, t) = \begin{cases} h_1((\lambda t - g(x, \alpha))t) & \text{if } (g(x, \alpha) < \lambda) and (t = 1) \text{ or } (g(x, \alpha) > -\lambda) and (t = -1) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $h_1(.)$ is a monotonically increasing function. Due to superfluity, the error for x is

$$L_2(x, t) = \begin{cases} h_2((g(x, \alpha) - \lambda t)t) & \text{if } (g(x, \alpha) > \lambda) and (t = 1) \text{ or } (g(x, \alpha) < -\lambda) and (t = -1) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

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

We minimize the error due to superfluity and the empirical error together such that the loss functional can be written as

$$L(x, t) = L_1(x, t) + L_2(x, t) \tag{12}$$

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.

2.2 Bias-Variance Perspective

As articulated by (Friedman, 1997) (also available in (Duda et al., 2001; Bishop, 2006; Le Borgne, 2005)), the overall probability of classification error can be expressed as

$$P[f(x, \alpha) \neq t] = (1 - 2P[t_B \neq t])P[g(x, \alpha) \neq t_B] + P[t_B \neq t] \tag{13}$$

where t is the actual observed outcome (in terms of ± 1) for a sample x, t_B is the outcome produced by the optimal Bayes classifier. The second term in Equation (13) is the noise component and independent of the classifier. The first term is the product of the error committed by the classifier with respect to the optimal Bayes classifier and a noise term which attenuates the total error of the classifier. If the Bayes classifier output is exactly the same as the actual outcome then the resultant error is the same as that committed by the classifier. However, if the outcome of Bayes classifier deviates from the actual outcome then it imposes a wrongly right effect since $f(x)$ can differ from t_B and t_B can differ from t and therefore $f(x)$ produces the same outcome as t. The optimal Bayes classifier is obtained as

$$t_B = sign(F(x) > l) \tag{14}$$

where

$$F(x) = P[t = 1|x] = 1 - P[t = -1|x] \tag{15}$$

The error of the classifier with respect to the optimal Bayes classifier is expressed as

$$P[f(x, \alpha) \neq t_B] = 1(F(x) < l) \int_0^{\infty} p(g)dg + 1(F(x) > l) \int_{-\infty}^{0} p(g)dg \tag{16}$$

where $1(.)$ indicates a $1/0$ function where $1(y) = 1$ if $y > 0$ and 0 otherwise. Intuitively the classifier commits an error when the optimal Bayes classifier assigns a class $+1$ and the classifier assigns $f(x, \alpha) = -1$ i.e., $g(x, \alpha) \leq 0$, and vice-versa. The parameter l is the decision boundary for the Bayes classifier such that

$$l = \frac{c_{-1,1}}{c_{-1,1} + c_{1,-1}} \tag{17}$$

where c_{ij} is the cost of misclassifying a sample in class i to class j. As provided in (Friedman, 1997), $p(g)$ is assumed to be Gaussian, and the resulting error committed by the classifier with respect to the optimal Bayes model is given as

$$P[f(x, \alpha) \neq t_B] = \Phi\left[sign(F(x) - l)\mu_g \sigma_g^{-1}\right] \tag{18}$$

where

$$\mu_g = \mathcal{E}_{\mathcal{D}}[g(x, \alpha; \mathcal{D}] \tag{19}$$

is the expectation of $g(x, \alpha)$ over the dataset $\mathcal{D}$. Similarly, σ_g is the standard deviation or the square-root of the variance of $g(x, \alpha)$ computed over the dataset $\mathcal{D}$. The function Φ is a nonlinear function given as

$$\Phi(y) = \frac{1}{2}[1 - erf(y/\sqrt{2})] \tag{20}$$

The bias term is dictated by μ_g and the resulting classification bias increases as μ_g decreases. On the other hand, the classification variance decreases as $sigma_g$ decreases. The bias and variance are related by certain nonlinear way in the case of classification. As defined by Friedman (Friedman, 1997), the bias

$$bias = sign(l - F(x))\mu_g \tag{21}$$

does not affect the classification so long as the bias is negative. The classification becomes incorrect when the bias becomes positive. This bias is called the boundary bias in classification. The variance is more important aspect of classification and even if the reduction of variance causes high bias, the classification remains correct so long as the boundary bias remains negative. Reduction in the variance and subsequent increase in the bias leads to over smoothing of the classification boundary.

Instead of considering unimodal Gaussian on $g(.)$ if we consider bimodal distribution (mixture of two Gaussians) over $g(.)$ for a two-class problem then the resultant error committed by the classifier with respect to the optimal Bayes model is given as

$$P[f(x, \alpha) \neq t_B] = k_1\Phi[sign(F(x) - l)\mu_1\sigma_1^{-1}] + k_2\Phi[sign(l - F(x))\mu_2\sigma_2^{-1}] \tag{22}$$

where $k_1 + k_2 = 1$ are the weights of each individual Gaussian in the mixture, μ_1 and $-\mu_2$ are the means of individual Gaussians, and σ_1, σ_2 are the respective standard deviations. So long as the sign of $(F(x) - l)$ is consistent with that of μ_1 and sign of $(l - F(x))$ is consistent with $-\mu_2$, the classification is correct. The accuracy increases with the reduction of σ_1 and σ_2.

In our approach, we also perform the variance reduction by minimizing the superfluity in the outcome. In other words, from Equations (10) and (11), we observe that the labels t are multiplied by a factor λ which essentially shrinks $g(x, \alpha)$ and thus the variance is reduced. However, if we simply use linear scaling of the model $g(x, \alpha)$ then the variance reduction is completely cancelled by the reduction of mean (gain in bias) and there is effectively no change in the classification performance. If we identify the model parameters within the same setting (without any linear scaling) to restrict the superfluity in the outcome then the reduction of variance is not cancelled by the gain in bias. In the sequel, we show the use of this principle to design two different classifiers where we gain in terms of variance with respect to certain parameter λ.

2.3 Relationship with the Regularization using Model Complexity

As a special case, if we restrict $h_1(u) = h_2(u) = u^2$ then the resultant loss functional for a training sample x (Equation (12)) becomes

$$L(x, t) = (g(x, \alpha) - \lambda t(x))^2 \tag{23}$$

Thus the total normalized loss over the training dataset $\mathcal{D}$ is

$$L(\mathcal{D}|\alpha) = \frac{1}{|\mathcal{D}|}\sum_x (g(x,\alpha) - \lambda t(x))^2 \tag{24}$$

The Equation (24) can be expressed as

$$L(\mathcal{D}|\alpha) = \frac{1}{|\mathcal{D}|}\sum_x [\lambda(g(x,\alpha) - t(x))^2 + (1-\lambda)g^2(x,\alpha) - \lambda(1-\lambda)t^2(x)] \tag{25}$$

Since the third term in Equation (25) is independent of α, we can equivalently express $L(\mathcal{D}|\alpha)$ as

$$L(\mathcal{D}|\alpha) = \frac{1}{|\mathcal{D}|}\sum_x (g(x,\alpha) - t(x))^2 + \frac{1-\lambda}{\lambda}\frac{1}{|\mathcal{D}|}\sum_x g^2(x,\alpha) \tag{26}$$

The second term in Equation (26) imposes explicit regularization on the outcome of the classifier and $\frac{1-\lambda}{\lambda}$ is the Lagrangian multiplier of the regularization functional. Since $\frac{1}{|\mathcal{D}|}\sum_x g^2(x,\alpha)$ is a function of α, we can equivalently express Equation (26) as

$$L(\mathcal{D}|\alpha) = \frac{1}{|\mathcal{D}|}\sum_x (g(x,\alpha) - t(x))^2 + \gamma\Gamma(\alpha) \tag{27}$$

where $\gamma = (1-\lambda)/\lambda$ is the Lagrangian multiplier and

$$\Gamma(\alpha) = \frac{1}{|\mathcal{D}|}\sum_x g^2(x,\alpha) \tag{28}$$

is a function of the model parameters α; $\Gamma(\alpha)$ represents the model complexity. Thus although we express the principle in terms of reducing the superfluity in the outcome, it is connected to the explicit regularization of the model complexity. However, the model complexity in Equation (27) is not the same as that defined in MDL or SRM. For example, in SRM principle, the model complexity is driven by the worst-case performance over the training samples whereas in the present framework the model complexity is the average over all training samples. Often the squared norm of the model parameter vector is considered in the Tikhonov regularization. In our approach, we do not explicitly consider any function $\Gamma(.)$ for regularization, rather the regularization function is derived from the outcome of the classifier itself. Also we observe that the contribution of the regularization term increases as we decrease λ and effectively bias increases. On the other hand for $\lambda = 1$, the regularization term vanishes.

3 Design of Classifiers using the Principle of Parsimony

In this section, we design two different classifiers using the stated principle.

3.1 A Kernel Machine

We consider a kernel based classifier such that the function $g(x,\alpha)$ is given as

$$g(x,\alpha) = \sum_j \alpha_j K(x,x_j)t_j \tag{29}$$

where x_j is an observed sample and t_j is the given label of the observed sample, and α_j indicates the contribution of the sample j in the classifier. $K(x, x_j)$ is a kernel function such as Gaussian kernel of the form

$$K(x, x_j) = \exp(-\frac{\|x - x_j\|^2}{2\sigma^2}) \tag{30}$$

The kernel is not necessarily restricted to Gaussian kernels. We restrict to $h_1(u) = h_2(u) = u^2$ (Equation (12)) and therefore the resultant loss functional (not normalized by the number of training samples) is given as

$$L = \frac{1}{2}\sum_i (\sum_j \alpha_j K(x_i, x_j) t_j - \lambda t_i)^2 \tag{31}$$

Using unrestricted parameter values, we can minimize L with respect to α using linear regression technique and obtain one classifier for one λ. In deriving the parameter values $L(\alpha)$ can be differentiated with respect to α and we obtain exactly $|\mathcal{D}|$ equations such that we can obtain an exact solution for α. However, the solution for α is simply linear in λ, and therefore λ linearly scales $g(x, \alpha)$. As discussed in Section 2.2, if we simply perform linear scaling of $g(x, \alpha)$ then the gain in the variance reduction is completely cancelled by the increase in bias. Therefore if we use unconstrained linear regression then for any value of λ we obtain the same classifier as obtained by minimizing the empirical error.

In order to effectively gain in variance reduction, we constrain the parameters in $0 \leq \alpha \leq 1$. Thus for a given λ, the minimization of L with respect to α is equivalent to maximization of the functional

$$W(\alpha) = \lambda t^T K D(t) \alpha - \frac{1}{2}\alpha^T D(t) K^T K D(t) \alpha \tag{32}$$

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. Once we maximize the functional $W(.)$ with respect to α we obtain certain non-zero αs which define the classifier. Since we constrain $0 \leq \alpha \leq 1$, as we increase λ, maximization of the objective functional $W(.)$ generates new non-zero vectors which changes the nature of the distribution. The introduction of new non-zero vectors causes an additional increase in the variance. Thus by extending this logic, we notice that decrease in λ reduces μ_g causing an increase in the bias, whereas the variance in decreased further (decrease in the number of non-zero vectors). If we decrease λ towards zero then it introduces infinite bias and no variance, and there is no classification (all the samples are treated as same). Therefore, for a particular classification task, there exist an optimal range of λ over which the classifier provides an optimal performance. If two classes are completely separable then we can obtain a wide range of λ whereas for a highly overlapped class structure, we can identify a specific narrow range of λ over which the classifier performs optimally.

Once the pattern vectors with non-zero α values are obtained by maximizing the functional $W(\alpha)$ with respect to α_is (Equation 32), we classify any new sample x, i.e., assign a class label t to x as

$$t = \begin{cases} 1 & \text{if } \sum_i \alpha_i K(x, x_i) t_i \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{33}$$

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 model coefficients α for each class separately. Here we mention that we take the stand-point as provided in (Rifkin & Klautau, 2004) where the authors have shown that one-against-all classification scheme is as accurate as any other multi-class classification approach.

Formally let there be l class labels such that the set of class labels is denoted by $CL = \{1, 2, \cdots, l\}$. Each time we consider one class label at a time. Let the class label of concern be $c \in CL$. In that case $t_i \in CL$ is transformed into a vector $\{t_{i1}, t_{i2}, \cdots, t_{il}\}$ such that

$$t_{ic} = \begin{cases} 1 & \text{if } t_i = c \\ -1 & \text{otherwise} \end{cases} \tag{34}$$

For each class label c, we compute the coefficients separately, and assign the class label t to x as

$$t = argmax_{c \in L}\{\sum_i \alpha_{ic} K(x, x_i) t_{ic}\} \tag{35}$$

where α_{ic}s are the coefficients obtained for class c.

3.1.1 Interpretation of the Kernel Machine

As we mentioned in Section 2.3, the objective functional for the kernel machine can be expressed as a combination of the empirical error and the model complexity in the form

$$L(\mathcal{D}|\alpha) = \frac{1}{|\mathcal{D}|} \sum_x (g(x, \alpha) - t(x))^2 + \gamma \Gamma(\alpha) \tag{36}$$

where γ is the regularization parameter ($\gamma = (1 - \lambda)/\lambda$) and $\Gamma(\alpha)$ is the model complexity given as

$$\Gamma(\alpha) = \frac{1}{|\mathcal{D}|} \sum_x g^2(x, \alpha) \tag{37}$$

Considering

$$g(x, \alpha) = \sum_j \alpha_j K(x, x_j) t_j \tag{38}$$

the model complexity is given as

$$\Gamma(\alpha) = \alpha^T D(t) K^T K D(t) \alpha \tag{39}$$

In other words, we minimize a quadratic loss functional equivalent to the empirical error in addition to a regularization function governed by the model complexity; the model complexity being dependent on the outcome of the classifier.

As we impose the constraints on the priors α, the loss can be reformulated as

$$L = \frac{\lambda^2}{2} \sum_i (\sum_j \frac{\alpha_j}{\lambda} K(x_i, x_j) t_j - t_i)^2 \tag{40}$$

which is equivalent to minimizing a functional

$$L = \frac{1}{2}\sum_i (\sum_j K(x_i, x_j)\hat{\alpha_j} t_j - t_i)^2 \tag{41}$$

subject to the constraints $0 \leq \hat{\alpha_i} \leq \frac{1}{\lambda}$. From the Equation (41) and the respective constraints, we observe the kernel 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

$$L = \frac{1}{2}\sum_i (\sum_j K(x_i, x_j)\beta_j - t_i)^2 \tag{42}$$

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 $\{-1, 1\}$. In other words, we minimize a quadratic loss functional equivalent to the empirical error subject to the box constraints on the model coefficients; the size of the box depends on the parameter λ (the margin according to the principle).

Alternatively combining Equations (36) and (42), we can express

$$L = \|t - K\beta\|^2 + \gamma \|K\beta\|^2 \tag{43}$$

With normalization,

$$L = \frac{1}{|\mathcal{D}|}(t - K\beta)^T (t - K\beta) + \frac{\hat{\gamma}}{2}\beta^T K^T K\beta \tag{44}$$

where $\beta \in [-1, 1]$ and $\beta_i t_i \geq 0$ and $\hat{\gamma} = \frac{2(1-\lambda)}{|\mathcal{D}|\lambda}$. Thus the coefficients have uniform prior over $[0, \pm 1]$ depending on the class label of the observed sample and regularized by the model complexity with a regularization parameter γ or $\hat{\gamma}$. We express the objective functional in this form in order to show the similarity with other existing kernel machines in the next section.

3.1.2 Relationship with Other Kernel Machines

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 outcome for a given parameter λ.

This leads to a difference in the quadratic term $K^T K$ in our formulation in Equation (32) 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

$$W_{svm}(\alpha) = \mathbf{1}^T \alpha - \frac{1}{2C}\alpha^T D(t) K D(t) \alpha \tag{45}$$

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 (Shawe-Taylor & Cristianini, 2000; Bie, Lanckriet, & Cristianini, 2003), the

funcitonal is expressed as

$$W_{softsvm}(\alpha) = \mathbf{1}^T\alpha - \frac{1}{2}\alpha^T D(t)(K + \frac{1}{C}I)D(t)\alpha \tag{46}$$

where I is the identity matrix with the same constraint i.e., $\alpha^T t = 0$. We do not require this constraint (Equation (32)) since different α_is contribute differently in the linear term depending on the kernel function and the distribution of the patterns.

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. Since in the formulation of the kernel based classifier, we do not use the principle of margin maximization, we do not claim this as a variant of SVM.

As we view the expanded quadratic loss in Equation (32), we observe the similarity of B-machine with ν-SVM (Chen, Lin, & Schölkopf, 2005), where the optimization functional is given as

$$L = -\frac{1}{2}\alpha^T D(t)KD(t)\alpha \tag{47}$$

subject to $0 \leq \alpha_i \leq \frac{1}{\lambda}$, λ being a regularization parameter $\lambda > 0$, 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 (Chen et al., 2005) depends on two parameters namely ν and λ. ν-SVM uses a principle of regularizing the margin of separation between the classes (with an additional regularization factor) where the parameter ν controls the penalization due to separation of the classes in the primal formulation of the loss functional. The separation between the classes is governed by the separation margin ρ and a regularization functional $-\nu\rho$ is added to the primal formulation. The minimization of the loss functional is subjected to the maximization of the separation margin ρ and the effect of ρ depends on the parameter ν. In this respect, B-machine also uses a parameter λ to control the separation between the classes in terms of the outcome of the classifier. However, the primal formulation of ν-SVM is derived from the margin maximization principle as in the SVM whereas in B-machine, we do not explicitly consider the margin maximization as a primal formulation; however, we consider the minimization of the loss functional with a given margin (regularization parameter) directly in its dual form.

In the ridge regression framework (Marquardt, 1970), the least square regression includes a regularization factor such that

$$L = \|t - w^T x\|^2 + a\|w\|^2 \tag{48}$$

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 (Saunders, Gammerman, & Vovk, 1998), 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 RLSC (Rifkin, Yeo, & Poggio, 2003), the objective functional is derived as a simple square-loss function and regularized using the Tikhonov regularization (Tikhonov & Arsenin, 1977) with a quadratic term in α vectors. In RLSC reproducing Hilbert kernels are considered such that the objective

functional takes a form

$$L = \frac{1}{N}(t - K\alpha)^T(t - K\alpha) + \frac{\gamma}{2}\alpha^T K\alpha \tag{49}$$

where N is the number of samples, γ 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

$$(K + \gamma NI)\alpha = t \tag{50}$$

In least-square support vector machine (LSSVM) (Suykens & Vandewalle, 1999; Van Gestel et al., 2004), 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 transformed into a set of linear equations by equating the first derivative to zero using unconstrained priors α.

We observe that the B-machine as expressed in Equations (43) and (44) is similar to the adaptive ridge regression (AdR) and RLSC. However, in AdR, RLSC, and LSSVM, the form of model complexity that has been used to regularize the model parameters is different from that in the B-machine. Therefore minimization of L with unconstrained priors (α) in such models does not yield simple scaling of the outcome $g(x, \alpha)$. In the B-machine on the other hand, unconstrained minimization results into simple scaling and we constrain the the parameters α. Moreover, RLSC and other related family of kernel machines are restricted to Hilbert kernels subjected to Mercer condition which is relaxed in the B-machine. The B-machine is not limited only to Hilbert kernels since the quadratic term in Equation (32) is always positive semi-definite.

Constrained priors have been used in LASSO (least absolute shrinkage and selection operator) (Tibshirani, 1996b; Osborne, Presnell, & Turlach, 2000), and generalized LASSO (Roth, 2004; Hosmer & Lemeshow, 2000). In LASSO, the loss functional considered is given as

$$L = \|t - K\alpha\|_2^2 \tag{51}$$

subject to $\|\alpha\|_1 < \lambda$ where λ is a constant which determines the approximation accuracy. The LASSO model is very similar to the B-machine as interpreted in Equation (42) except that the constraints over the priors are different. Considering the exponential hyperpriors, the LASSO model can be expressed in the form

$$L = \|t - K\alpha\|_2^2 + b\|\alpha\|_1 \tag{52}$$

where b is a Lagrange parameter determined by the exponential hyperpriors. The generalized LASSO uses a very similar form of loss functional as in Equation (52) except that it uses a more robust form of Huber loss (Smola & Schölkopf, 1998) which is quadratic for smaller deviation and linear for larger deviations. Subsequently iteratively reweighted least square (IRLS) technique is used to optimize the functional. The constraints that we use in the B-machine (Equation (43)) are different from the constraints used in LASSO and generalized LASSO.

In relevance vector machines (Tipping, 2001), empirical error is defined based on the logistic regression model where the outcome is given as $g(x) = 1/(1+\exp(-w^T\phi(x))$. In a Bayesian framework zero-mean Gaussian priors are defined over the parameter vectors w such that $p(w|\alpha) = \prod_i \mathcal{N}(0, 1/\alpha_i)$ where $\mathcal{N}$ represents the Gaussian distribution. 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 variance) parameters are chosen. The prior variance parameters are then used in the posterior to get new estimates of α. Thus relevance vector machine explicitly regularizes the model parameters w within a Bayesian framework with respect to the priors α and iteratively estimates the priors. In the B-machine, we do not use the logistic regression model for classification for measuring the empirical error and instead of Gaussian hyperpriors, we use uniform priors over the parameters α.

3.2 A Modified k-Nearest Neighbor Algorithm

As we discussed in Section 2.1 (Equation (7)) that the outcome for a test samples can also be expressed as a soft indicator function, we apply the proposed principle to design a modified k-nearest neighbor classifier using the soft indicator function. In the case of k-NN classifier, the model parameter is k or the number of nearest neighbors being considered. Therefore according to this principle, we decide the value of k for every test sample with a given value of λ. On the contrary, in the standard k-NN classifier, all test samples are classified for a given k (the value of k can be decided by cross-validation). First, we scale the outcome of the classifier for a test sample in $[0, 1]$ for l different class labels. We assign a function $g(x, k)$ such that

$$g(x, k) = \frac{lN_t(x) - k}{k(l-1)} \tag{53}$$

where $N_t(x) > N_i(x)$ for all $i \neq t$ and $i, t \in CL$. $N_i(x)$ is the number of training samples with a class label i in the k nearest neighboring training samples of x and $CL = \{1, 2, \cdots, l\}$ is the set of class labels. We scale the outcome for a test samples in $[0, 1]$ so that the value of λ can be chosen in the range $[0, 1]$ irrespective of the number of class labels. We then compute the outcome for the test sample x for a large range of k and choose that particular k for which $g(x, k)$ is closest to λ and assign the respective class label to x. Formally, the algorithm is presented in Figure 1. In this algorithm, we do not use any kernel function to implement the k-nearest neighbor algorithm. Rather the model parameter k is decided from the outcome of the classifier for a given margin λ. Since the performance of this classifier is solely dependent on the parameter λ, we refer to this classifier as λ-kNN classifier in the sequel. Interestingly, we note that the value of k in λ-kNN can differ for different test samples. If the classifier performs best of $k = 1$ for all test samples (i.e., 1-nearest neighbor classifier) then the value of optimal λ will be unity.

Input: Training dataset ($\mathcal{D}$); Test dataset;
Output: class label $t(x)$ for every test sample x;
Input Variable: λ;
begin
  for every test samples x *do*
  begin
    for $k := 1$ *to* k_max *do*
    begin
      Compute the class label $f(x,k) = t$ if $N_t(x) > N_i(x)$ for all $i \neq t$, $i, t \in CL$
      Compute the outcome $g(x,k) = \frac{lN_t(x)-k}{k(l-1)}$
    end
    Find $k_{opt} = argmin_k|g(x,k) - \lambda|$
    Assign class label to x such that $t(x) = f(x, k_{opt})$
  end
end

Figure 1: Modified k-nearest neighbor classifier (λ-kNN).

4 Experimental Results

4.1 Nature of the B-Machine

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 2, 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 3 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 the opposite class samples. This is due to higher interaction between samples from other classes due to larger width of the kernel function.

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 4 illustrates the B-vectors for the same Gaussian parity problem with $\lambda = 0.1$, 0.2, 0.5, and 1 respectively for $\sigma = 1$.

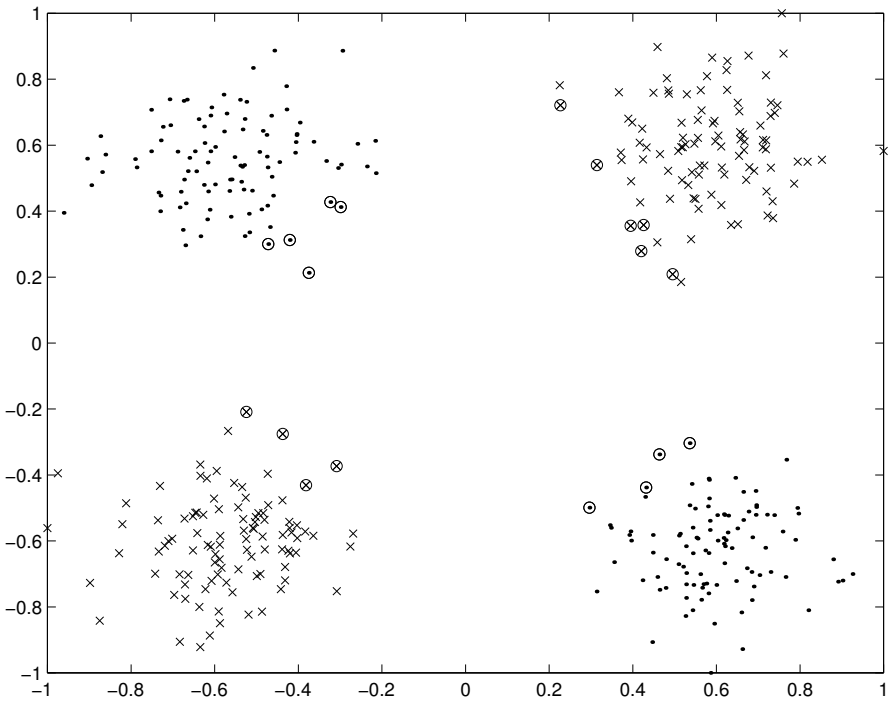

Figure 2: B-vectors obtained by the B-machine for a two-dimensional Gaussian parity problem. The two classes are represented by 'x' and '.' respectively. The B-vectors are marked by 'o'. The Gaussian kernel of the B-machine is chosen such that $2\sigma^2 = 1$.

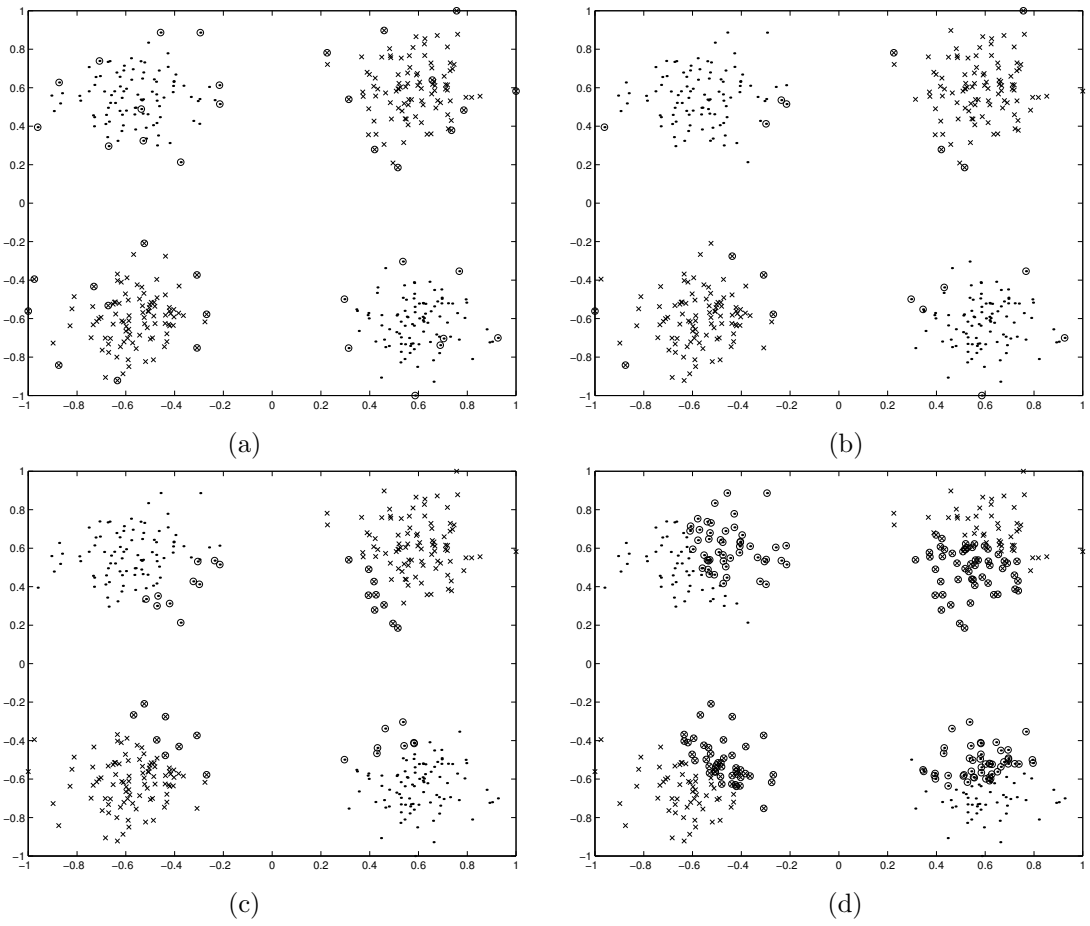

Figure 3: B-vectors obtained by B-machine for the same two-dimensional Gaussian parity problem as in Figure 2 with different kernel sizes. (a), (b), (c), and (d) illustrate the B-vectors for $\sigma = 0.2$, $\sigma = 0.5$, $\sigma = 1.0$ and $\sigma = 2.0$ respectively.

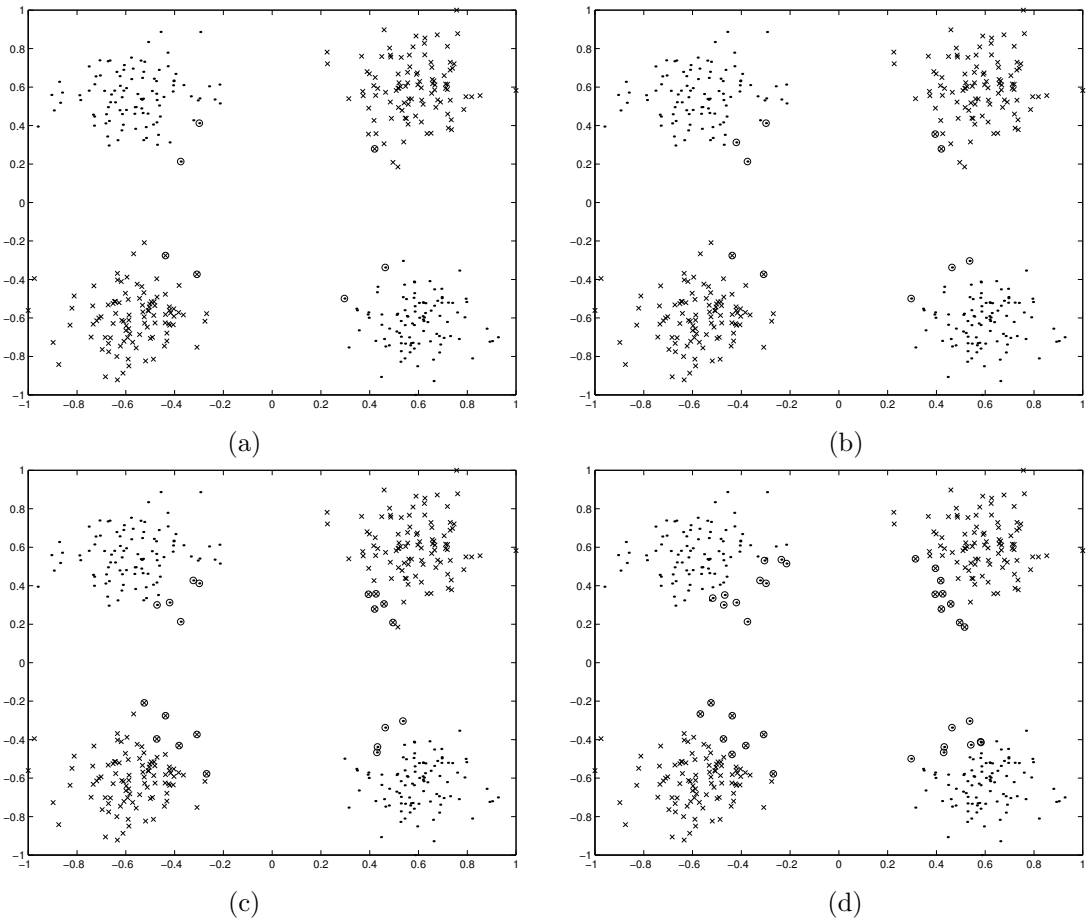

Figure 4: B-vectors obtained by B-machine for the same two-dimensional Gaussian parity problem as in Figure 2 for different margins (λ) with a fixed size of Gaussian kernel $\sigma = 1$. (a), (b), (c), and (d) illustrate the B-vectors for $\lambda = 0.1$, $\lambda = 0.2$, $\lambda = 0.5$, and $\lambda = 1.0$ respectively.

4.2 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 (Blake & Merz, 1998). Table 1 summarizes the data set description. We considered the data sets from UCI machine learning repository (Blake & Merz, 1998) 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 (Nakai & Kanehisa, 1991) and later in (Horton & Nakai, 1996) 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 (Nakai & Kanehisa, 1991) also, the results are not cross-validated. We normalize all input patterns such that any component 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

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

4.3 Performance of the B-Machine

4.3.1 Choice of the Kernel Functions

The kernel functions of the B-machine is not necessarily subjected to Mercer condition (Hilbert kernels) since the quadratic term in the objective functional of B-machine (Equation (32)) is always positive semi-definite. 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

$$K_1(x,\mu) = \frac{1}{1+\left(\frac{\|x-\mu\|}{\sigma}\right)^2} \tag{54}$$

The kernel in Equation (54) 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 (Olshausen & Field, 1996). 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

$$K_2(x,\mu) = \begin{cases} \exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right) & \text{for } \|x-\mu\| \leq \epsilon \\ \exp\left(-\frac{\|x-\mu\|}{\sqrt{2}\sigma}\right) & \text{otherwise} \end{cases} \tag{55}$$

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.

4.3.2 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 (Canu, Grandvalet, Guigue, & Rakotomamonjy, 2005). We implemented LSSVM using the Matlab code available in (Pelckmans et al., 2003). In implementing LSSVM for multiclass classification (such as 'Iris' and 'Ecoli'), we used 'minimum output coding' (Pelckmans et al., 2003) which provided the best performance for the LSSVM. In Table 2, we report the best performances of all the classifiers namely, SVM, LSSVM, and B-machine and the corresponding parameter values. We observed that all these three classifiers are able to obtain similar scores over a significantly large range of parameter values, and we report one candidate set of values in Table 2. As we observe from Table 2, B-machine outperforms SVM and LSSVM in terms of 10-fold cross-validation score on several datasets. For the 'Pima', 'Bupa', 'Wpbc', 'Iris' and 'Ecoli' datasets, 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.

Classifier	Pima	Bupa	Wdbc	Wpbc	Iris	Ecoli
SVM	76.88	69.39	98.03	80.78	96.13	88.17
(Gaussian)	($\pm$4.74)	($\pm$6.53)	($\pm$1.80)	($\pm$6.15)	($\pm$4.87)	($\pm$4.53)
(σ, C)	$(1, 2)$	$(0.3, 1)$	$(2, 20)$	$(1, 2)$	$(1, 1)$	$(0.5, 1)$
SVM	75.58	71.72	96.20	74.39	96.13	87.25
(Polynomial)	($\pm$4.74)	($\pm$6.40)	($\pm$2.26)	($\pm$8.72)	($\pm$5.06)	($\pm$4.88)
(C)	(1)	(5)	(1)	(1)	(5)	(3)
LSSVM	76.28	68.87	93.78	78.13	84.53	77.40
(Gaussian)	($\pm$5.18)	($\pm$7.21)	($\pm$2.93)	($\pm$3.93)	($\pm$7.90)	($\pm$5.54)
(σ^2, γ)	$(5, 0.1)$	$(1, 2)$	$(4, 0.3)$	$(5, 4)$	$(0.2, 15)$	$(1, 10)$
BM	77.51	72.42	97.68	81.05	95.87	87.93
(Gaussian)	($\pm$5.41)	($\pm$6.37)	($\pm$1.65)	($\pm$7.11)	($\pm$5.02)	($\pm$4.79)
(σ, λ)	$(5, 0.2)$	$(2, 0.2)$	$(1.5, 1)$	$(5, 0.6)$	$(1, 0.4)$	$(1, 0.5)$
BM	76.28	73.17	97.61	80.83	97.07	88.77
(Gauss+Expo)	($\pm$4.78)	($\pm$6.59)	($\pm$1.76)	($\pm$6.35)	($\pm$4.70)	($\pm$4.56)
(σ, λ)	$(2, 0.2)$	$(2, 0.6)$	$(0.2, 0.2)$	$(3, 1)$	$(1, 0.6)$	$(2, 0.4)$
BM	77.10	72.77	97.81	82.33	97.60	89.16
(Cauchy)	($\pm$5.40)	($\pm$6.55)	($\pm$1.65)	($\pm$6.51)	($\pm$4.21)	($\pm$4.37)
(σ, λ)	$(5, 0.2)$	$(3, 0.4)$	$(2, 0.7)$	$(3, 0.5)$	$(2, 0.3)$	$(3, 0.8)$

Table 2: Classification performance in terms of the 10-fold cross-validation scores on the datasets described in Table 1. Classification performance summarizes the best mean classification accuracy and the corresponding standard deviation of the accuracies as a significance check. Corresponding to each classifier model, the candidate parameter values for which the best performance is obtained, are shown.

In order to establish the comparison between these classifiers in a more quantitative way, we performed resampled paired t-test as provided in (Dietterich, 1998). As described in (Dietterich, 1998), 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 is $p^{(i)} = p_1^{(i)} - p_2^{(i)}$, and the statistic is obtained as

$$t = \frac{\bar{p}.\sqrt{N}}{\sqrt{\frac{\sum_{i=1}^{N}(p^{(i)}-\bar{p})^2}{N-1}}} \tag{56}$$

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 (Dietterich, 1998) (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. 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.

Classifier Pair	Pima	Bupa	Wdbc	Wpbc	Iris	Ecoli
BM (Gauss) SVM (Gauss)	-3.94	-7.33	0.21	-3.33	-3.26	2.37
BM (Gauss) SVM (Poly)	-7.0	-3.81	-6.93	-9.59	-2.72	-2.19
BM (Gauss) LSSVM	-4.27	-8.10	-14.73	-4.75	-17.83	-12.10
BM (GaussExpo) SVM (Gauss)	1.73	-8.00	1.48	-0.41	-5.43	-2.54
BM (GaussExpo) SVM (Poly)	-2.03	-4.61	-4.91	-6.41	-4.63	-8.53
BM (GaussExpo) LSSVM	1.52	-8.56	-14.11	-2.29	-17.81	-13.34
BM (Cauchy) SVM (Gauss)	-4.84	-6.97	1.09	-2.71	-4.82	-3.25
BM (Cauchy) SVM (Poly)	-7.4	-3.49	-5.38	-7.45	-4.85	-6.56
BM (Cauchy) LSSVM	-4.9	-7.91	-14.68	-3.90	-17.94	-13.79

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.

4.3.3 Adult and Web Datasets

In Platt (Platt, 1998), two different datasets namely 'adult' and 'web' datasets were used. In the adult dataset, the task is to classify households whether having annual income greater than USD 50K or not based on the 14 different census fields including eight categorical variables. The data is transformed into 123 sparse binary attributes where six continuous variables are quantized into quantiles. In the web dataset, the task is to predict whether a web page belongs to particular category or not based on the presence of 300 different keywords. Thus the web dataset also has 300 sparse binary attributes. The original 'adult' and 'web' datasets consist of 32562 and 49749 training samples. Since we used Matlab quadratic programming library to directly optimize the objective functional of the B-machine, the large number of samples could not be accommodated due to the limitation of the virtual memory. Originally Platt (Platt, 1998) used sequential minimal optimization for the larger datasets, however, we have not used any equivalent implementation of the B-machine. Platt (Platt, 1998) used nested sets of training samples for both the 'adult' and the 'web' datasets. We used the first two subsets of the 'adult' dataset and the first subset of the 'web' dataset. In Table 4, we show the respective details of the 'adult' and 'web' datasets that we used for the experimentation. These datasets are highly unbalanced (samples from one class largely dominate the dataset) and SVM is able to perform classification with high accuracy even for these unbalanced datasets. We compare the performance of the B-machine with that of SVM and LSSVM for different types of kernels. In Table 5 we report the results over the test samples for all classifiers with the best parameter settings. Here also we observe that the B-machine performs better than the SVM and LSSVM although the differences in performance scores are not so significant. However this shows that the B-machine is able to perform well even for the unbalanced datasets such as adult and web datasets.

Dataset	number of attributes	number of training samples	number of test samples
Adult1	123	1605	30956
Adult2	123	2265	30296
Web1	300	2477	47272

Table 4: Description of the nested subsets of the 'adult' and 'web' datasets.

Classifier	SVM (Gaussian) (σ^2, C)	SVM (Cubic) (C)	SVM (Linear) (C)	LSSVM (Gaussian) (σ^2, γ)	BM (Gaussian) (σ, λ)	BM (Gauss+Expo) (σ, λ)	BM (Cauchy) (σ, λ)
Adult1	84.22 (10,1)	80.15 (0.01)	84.26 (0.05)	82.1 (20,20)	84.14 (10,0.8)	84.14 (10,0.8)	84.28 (10,0.8)
Adult2	84.33 (10,1)	79.18 (0.01)	84.44 (0.05)	81.42 (20,20)	84.55 (10,0.8)	84.55 (10,0.8)	84.69 (10,0.8)
Web1	97.96 (10,5)	97.72 (0.01)	97.74 (1)	97.98 (100,5)	98.1 (5,0.2)	97.88 (7,0.2)	98.1 (5,0.2)

Table 5: Best classification performance on the 'adult' and 'web' test datasets for SVM, LSSVM and B-machine. Corresponding best parametric settings of each classifier is also reported.

4.4 Peformance of the λ-kNN Classifier

We report the 10-fold cross-validation results of the λ-kNN classifier as proposed in Section 3.2 for the datasets in Table 1. We computed the CV scores in the same way (random sampling) as described in Section 4.3.2. As a comparison we provide the corresponding scores of the standard k-nearest neighbor algorithm. We report the best scores obtained for both the algorithms and show the value of optimal λ for the λ-kNN and the value of optimal k for the standard kNN classifier. Table 6 summarizes the performance scores. We observe that λ-kNN is able to outperform the standard kNN algorithm in terms of mean accuracies on the 'Bupa', 'Wdbc', 'Wpbc', and 'Ecoli' datasets. The scores are same for the 'Iris' datset whereas for the 'Pima' datset, standard kNN performs better than the λ-kNN.

Classifier	Pima	Bupa	Wdbc	Wpbc	Iris	Ecoli
λ-kNN	75.86	68.18	97.44	79.75	97.33	87.51
	($\pm$4.12)	($\pm$8.46)	($\pm$2.13)	($\pm$7.11)	($\pm$4.44)	($\pm$5.40)
(λ)	(0.13)	(0.37)	(0.67)	(0.7)	(0.48)	(0.57)
kNN	76.01	65.41	97.36	78.89	97.33	87.38
	($\pm$3.71)	($\pm$7.08)	($\pm$2.13)	($\pm$6.93)	($\pm$4.44)	($\pm$4.68)
(k)	(15)	(15)	(15)	(6)	(30)	(14)

Table 6: 10-fold cross-validation scores (with random sampling) of kNN and λ-kNN over the datasets in Table 1. Scores summarize the mean accuracies and corresponding standard deviations. Corresponding best parametric settings are also reported for each classifier.

We also compare the performance of the λ-kNN classifier with the standard kNN classifier using the 'adult' and 'web' datasets in Table 7. Since the adult and web datasets contains sparse vectors of zeros and ones, we used two different similarity measures to obtain the nearest neighbors. We used the standard Euclidian distance and also used normalized cosine similarity. Normalized cosine similarity is often used to measure the similarities between text documents. We observe that for the 'adult1' dataset, the standard kNN classifier performs better than the λ-kNN classifier when we compute neighbors based on the Euclidian distance; however, λ-kNN outperforms standard kNN classifier using cosine similarity. For the 'adult2' and 'web1' datasets, λ-kNN classifier performs better than the standard kNN with both Euclidian distance and cosine similarity.

Classifier	Adult1		Adult2		Web1	
	Euclidian	Cosine	Euclidian	Cosine	Euclidian	Cosine
λ-kNN	83.02	83.25	83.40	83.43	97.49	97.93
(λ)	(0.26)	(0.26)	(0.24)	(0.26)	(0.92)	(0.98)
kNN	83.12	83.21	83.37	83.29	97.41	97.89
(k)	(36)	(42)	(42)	(41)	(5)	(3)

Table 7: Classification performance of λ-kNN and standard kNN classifiers on 'adult' and 'web' datasets using Euclidian distance and cosine similarity. The best parametric setting corresponding to each classifier is shown.

5 Summary and Conclusions

We presented an interpretation of the principle of parsimony in the context of pattern classification where we minimize the superfluity in the outcome of a classifier with a given margin. We described the meaning of the outcome in the context of different types of classifiers. We then provided an explanation of this principle from the bias-variance perspective where we have shown that the margin of the superfluity determines the boundary bias of the classifier. As the margin increases, the bias decreases and the variance increases. Similarly the bias increases with the decrease in the margin. In the limiting condition, as the margin goes to zero, it imposes infinite bias on the classifier. We also explained the connection of this principle with the regularization techniques using model complexity. The Lagrangian multiplier of the regularization functional increases with a decrease of the margin of superfluity in the outcome. This principle is different from the margin maximization principle used in the SVM classifier where margin denotes how far the samples are separated from the separating hyperplane. In our context, the margin acts as a threshold for the superfluity in the outcome of a classifier. Note that, we do not claim this principle as a new principle. This principle has been used in decision tree pruning where the leaf nodes of a tree are pruned till the impurity reaches a certain threshold. In context of decision tree growing, a leaf node is not further split if the impurity reaches that threshold. The threshold in impurity is exactly the same as the margin in our principle. However, this mechanism has not been explicitly used to design other classifiers. We explicitly state this practice in the form of a principle and use this principle to design two different classifiers namely B-machine (which is kernel based classifier) and λ-kNN (which is based on k-nearest neighbor classifier). We show that for certain values of λ, B-machine is able to outperform the standard SVM on several datasets. We also observed that λ-kNN is able to outperform standard kNN classifier for several datasets.

We also discussed the relationship of the B-machine with certain other kernel based classifiers. We interpreted B-machine as a least square kernel machine with box constraint where it uses uniform priors instead of Gaussian priors. From the regularization perspective, B-machine has similarity with certain other kernel machines although the regularization functional is different in the B-machine. We mentioned that ν-SVM also uses a similar principle of regularizing the margin of separation between the classes (with an additional regularization factor) where the parameter ν controls the penalization due to separation of the classes in the loss functional. However, ν-SVM also uses the basic principle of margin maximization between the classes as used in SVM whereas the basic principle in B-machine is different. We mentioned that B-machine is flexible to accommodate non-Mercer kernels unlike the SVM and related kernel machines. We observed that use of centrally peaked long-tailed kernels can enhance the performance of the B-machine. 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.

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 processes for B-machine. In this respect, further investigation can be performed

towards obtaining better variants of the B-machine. With respect to the run-time memory requirements, sequential minimal optimization of B-machine objective functional for larger datasets can also be investigated as a scope of future study.

References

Akaike, H. (1978). A Bayesian analysis of the minimum AIC procedure. *Annals of the Institute of Statistical Mathematics*, *30A*, 9-14.

Amari, S. (1967). Theory of adaptive pattern classifiers. *IEEE Trans.*, *EC-16*, 299-307.

Bie, T. D., Lanckriet, G. R. G., & Cristianini, N. (2003). *Convex tuning of the soft margin parameter* (Tech. Rep. No. UCB/CSD-03-1289). Berkeley, California, USA: University of California, Computer Science Division (EECS).

Bishop, C. M. (2006). *Pattern recognition and machine learning.* UK: Springer.

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases* (Tech. Rep. No. *http://www.ics.uci.edu/∼mlearn/MLRepository.html*). USA: University of California, Irvine, Dept. of Information and Computer Sciences.

Breiman, L. (1996a). Bagging predictors. *Machine Learning*, *26*, 123-140.

Breiman, L. (1996b). Stacked regressions. *Machine Learning*, *24*, 49-64.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, *26*, 801-824.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1983). *Classification and regression trees.* New York: Chapman & Hall.

Canu, S., Grandvalet, Y., Guigue, V., & Rakotomamonjy, A. (2005). *Svm and kernel methods matlab toolbox.* Perception Systmes et Information, INSA de Rouen, Rouen, France.

Chen, P.-H., Lin, C.-J., & Schölkopf. (2005). A tutorial on nu-support vector machines. *Applied Stochastic Models in Business and Industry*, *21*, 111-136.

Cover, T. M. (1969). Learning in pattern recognition. In S. Watanabe (Ed.), *Methodologies of pattern recognition* (p. 111-132). New York: Academic Press.

Dietterich, T. G. (1998). Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, *10*, 1895-1923.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification (2nd edition).* New York: Wiley.

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

Friedman, J. H. (1997). On bias, variance, 0/1-loss, and curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*, 55-77.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1-58.

Hart, P. E. (1968). The condensed nearest neighbor rule. *IEEE Trans. Information Theory*, *IT-14*, 515-516.

Haykin, S. (1999). *Neural networks: A comprehensive foundation.* Upper Saddle River, NJ: Prentice Hall.

Horton, P., & Nakai, K. (1996). A probablistic classification system for predicting the cellular

localization sites of proteins. In *Intelligent Systems in Molecular Biology* (p. 109-115). St. Louis, USA.

Hosmer, D. W. J., & Lemeshow, S. (2000). *Applied logistic regression (2nd ed.).* USA: John Wiley.

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

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*, 181-214.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International joint conference on artificial intelligence (ijcai-95).*

Kohavi, R., & Wolpert, D. H. (1996). Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the thirteenth international conference on machine learning (icml96).*

Le Borgne, Y. (2005). *Bias-variance trade-off characterization in a classification problem: What differences with regression?* (Tech. Rep. No. 534). Belgium: Machine Learning Group, Univ. Libre de Bruxelles.

Marquardt, D. W. (1970). Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. *Technometrics*, *12*, 591-612.

Murata, N., Hoshizawa, S., & Amari, S.-I. (1993). Learning curves, model selection and complexity in neural networks. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems (nips92)* (Vol. 5, p. 607-614). Cambridge, MA: MIT Press.

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

Olshausen, B. A., & Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, *381*, 607-609.

Osborne, M., Presnell, B., & Turlach, B. (2000). On the LASSO and its dual. *Journal Comput. Graphical Statist.*, *9*, 319-337.

Pelckmans, K., Suykens, J. A. K., Van Gestel, T., De Brabanter, J., Lukas, L., Hamers, B., De Moor, B., & Vandewalle, J. (2003). *Ls-svmlab toolbox users guide* (Tech. Rep. No. 02-145; *http://www.esat.kuleuven.ac.be/sista/lssvmlab/*). Belgium: Department of Electrical Engineering, ESAT-SCD-SISTA, Katholieke Universiteit Leuven.

Platt, J. C. (1998). *Sequential minimal optimization : A fast algorithm for training support vector machines* (Tech. Rep. No. MSR-TR-98-14). USA: Microsoft Research.

Quinlan, J. R. (1993). *Programs for machine learning.* San Fransisco, CA, USA: Morgan Kaufmann.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *Journal of Machine Learning Research*, *5*, 101-141.

Rifkin, R., Yeo, G., & Poggio, T. (2003). Regularized least square classification. In J. Suykens, G. Horvath, S. Basu, C. Micchelli, & J. Vandewalle (Eds.), *Advances in learning theory: Methods, model and applications, NATO science series iii: Computer and system sciences* (Vol. 190, p. 131-153). Amsterdam: IOS Press.

Rissanen, J. (1978). Modeling by shortest path description. *Automatica*, *14*, 465-471.

Roth, V. (2004). The generalized LASSO. *IEEE Trans. Neural Networks*, *15*(1), 16-28.

Saunders, C., Gammerman, A., & Vovk, V. (1998). Ridge regression learning algorithm in

dual variables. In *Proceedings of the 15th international conference on machine learning (icml98).*

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*, 297-336.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461-464.

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

Shawe-Taylor, J., & Cristianini, N. (2000). *An introduction to support vector machines and other kernel-based learning methods.* UK: Cambridge University Press.

Smola, A. J., & Schölkopf, B. (1998). *A tutorial on support vector regression* (Tech. Rep. No. NC-TR-98-030). Royal Holloway College, University of London, UK: NeuroCOLT.

Smyth, P., & Wolpert, D. (1999). An evaluation of linearly combining density estimators via stacking. *Machine Learning*, *36*, 59-83.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, *36*, 111-147.

Stone, M. (1977). Asymptotics for and against cross-validation. *Biometrika*, *64*, 29-35.

Suykens, J. A. K., & Vandewalle, J. (1999). Least square support vector machine classifiers. *Neural Processing Letters*, *9*, 293-300.

Tibshirani, R. (1996a). *Bias, variance and prediction error for classification rules* (Tech. Rep. No. http://www.utstat.toronto.edu/$\tilde{}$tibs). Canada: Dept. of Statistics, University of Toronto.

Tibshirani, R. (1996b). Regression shrinkage and selection via LASSO. *Journal Royal Statistical Society, Series B*, *58*, 267-288.

Tikhonov, A. N., & Arsenin, V. Y. (1977). *Solutions of ill-posed problems.* Washington, D. C.: W. H. Winston.

Tipping, M. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal Machine Learning Research*, *1*, 211-244.

Tsypkin, Y. (1973). *Foundation of the theory of learning systems.* New York: Academic Press.

Van Gestel, T., Suykens, J., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., De Moor, B., & J., V. (2004). Benchmarking least squares support vector machine classifiers. *Machine Learning*, *54*, 5-32.

Vapnik, V. (1995). *The nature of statistical learning theory.* New York: Springer-Verlag.

Vapnik, V., & Chervonenkis, A. (1974). *Theory of pattern recognition.* Moskow, Russian edition: Nauka.

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