Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers

Bernhard Schölkopf, Kah–Kay Sung, Chris Burges, Federico Girosi, Partha Niyogi, Tomaso Poggio, and Vladimir Vapnik

Abstract— **The Support Vector (SV) machine is a novel type of learning machine, based on statistical learning theory, which contains polynomial classifiers, neural networks, and radial basis function (RBF) networks as special cases. In the RBF case, the SV algorithm automatically determines centers, weights and threshold that minimize an upper bound on the expected test error.**

The present study is devoted to an experimental comparison of these machines with a classical approach, where the centers are determined by k–means clustering and the weights are computed using error backpropagation. We consider three machines, namely a classical RBF machine, an SV machine with Gaussian kernel, and a hybrid system with the centers determined by the SV method and the weights trained by error backpropagation. Our results show that on the US postal service database of handwritten digits, the SV machine achieves the highest recognition accuracy, followed by the hybrid system. The SV approach is thus not only theoretically well–founded, but also superior in a practical application.

Keywords—**Support Vector machines, radial basis function networks, Pattern Recognition, Clustering, Prototypes**

I. Introduction

CONSIDER Fig. 1. Suppose we want to construct a radial basis function classifier

$$D(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} w_i \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{c_i}\right) + b\right) \quad (1)$$

(b and c_i being constants, the latter positive) separating balls from circles, i.e. taking different values on balls and circles. How do we choose the *centers* $\mathbf{x}_i$? Two extreme cases are conceivable (Fig. 2):

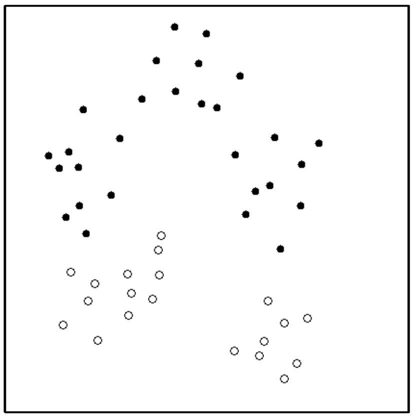

Fig. 1. A simple 2–dimensional classification problem: find a decision function separating balls from circles. The box, as in all following pictures, depicts the region $[-1,1]^2$.

The first approach consists of choosing the centers for the two classes separately, irrespective of the classification task to be solved. The classical technique of finding the centers by some clustering technique (*before* tackling the classification problem) is such an approach. The weights w_i are then usually found by either error backpropagation (Rumelhart, Hinton, & Williams, 1986) or the pseudo–inverse method (e.g. Poggio & Girosi, 1990).

An alternative approach consists of choosing as centers points which are *critical* for the classification task at hand. Recently, the *Support Vector Algorithm* was developed (Boser, Guyon, & Vapnik 1992, Cortes & Vapnik 1995, Vapnik 1995), implementing the latter idea. It is a general algorithm, based on guaranteed risk bounds of statistical learning theory, which in particular allows the construction of RBF classifiers. This is done by simply choosing a suitable kernel function for the SV machine (see Sec. II-B). The SV training consists of a quadratic programming problem which can be solved efficiently and for which we are guaranteed to find a global extremum. The algorithm automatically computes the number and location of the above centers, the weights w_i, and the threshold b, in the following way: by the use of a kernel function (in the present case, a Gaussian one), the patterns are mapped nonlinearly into a high–dimensional space. There, an optimal separating hyperplane is constructed, expressed in terms of those examples which are closest to the decision boundary (Vapnik 1979). These are the *Support Vectors* which correspond to the centers in input space.

The goal of the present study is to compare real–world results obtained with k–means clustering and classical RBF

B. Schölkopf is with the Max–Planck–Institut für biologische Kybernetik, Spemannstr. 38, 72076 Tübingen, Germany

K. Sung is with the Department of Information Systems and Computer Science at the National University of Singapore, Lower Kent Ridge Road, Singapore 0511

C. Burges is with Lucent Technologies, Bell Laboratories, 101 Crawford Corner Road, Holmdel, NJ 07733, USA

F. Girosi is with the Center for Biological and Computational Learning, Massachusetts Institute of Technology, E25-201, Cambridge, MA 02142, USA

P. Niyogi is with Lucent Technologies, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, USA

T. Poggio is with the Center for Biological and Computational Learning, Massachusetts Institute of Technology, E25-201, Cambridge, MA 02142, USA

V. Vapnik is with AT&T Research, 100 Schultz Dr., Red Bank, NJ 07701, USA

Part of the work was done while BS was with AT&T Bell Laboratories, and while KS and PN were with MIT. BS thanks MIT for hospitality during a three–week visit in March 1995, where this work was started. Support for the Center for Biological and Computational Learning is provided in part by a grant from the National Science Foundation under contract ASC–9217041. BS was supported by the Studienstiftung des deutschen Volkes; CB was supported by ARPA under ONR contract number N00014-94-C-0186. We thank K.–R. Müller, P. Simard, and A. Smola for useful discussions.

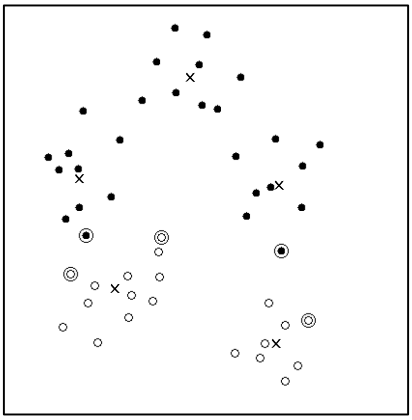

Fig. 2. RBF centers automatically found by the SV algorithm (indicated by extra circles), using $c_i = 1$ for all i (cf. Eq. 1). The number of SV centers accidentally coincides with the number of identifiable clusters (indicated by crosses found by k–means clustering with $k = 2$ and $k = 3$ for balls and circles, respectively) but the naive correspondence between clusters and centers is lost; indeed, 3 of the SV centers are circles, and only 2 of them are balls. Note that the SV centers are chosen with respect to the classification task to be solved.

training to those obtained when the centers, weights and threshold are automatically chosen by the SV algorithm. To this end, we decided to undertake a performance study by combining expertise on the SV algorithm (AT&T Bell Laboratories) and on the classical RBF networks (Massachusetts Institute of Technology). We report results obtained on a US postal service database of handwritten digits.

We have organized the material as follows. In the next Section, we describe the algorithms used to train the different types of RBF classifiers compared in this paper. Following that, we present an experimental comparison of the approaches. We conclude with a discussion of our findings.

II. Constructing Radial Basis Function Classifiers

We describe three RBF systems, trained in different ways. In Sec. II-A, we discuss the first system trained along more classical lines. In the following section (II-B), we describe the SV algorithm, which constructs an RBF network whose parameters (centers, weights, threshold) are automatically optimized. In Sec. II-C, finally, we use the SV algorithm merely to choose the centers of the RBF network and then optimize the weights separately.

A. *Classical Spherical Gaussian RBFs*

We begin by first describing the classical Gaussian RBF system. A N-dimensional spherical Gaussian RBF network with K centers has the form

$$g(\mathbf{x}) = \sum_{i=1}^{K} w_i \mathcal{G}_i(\mathbf{x}) + b$$
$$= \sum_{i=1}^{K} w_i \frac{1}{(2\pi)^{N/2}\sigma_i^N} \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{2\sigma_i^2}\right) + b$$

where $\mathcal{G}_i$ is the ith Gaussian basis function with center $\mathbf{c}_i$ and variance σ_i^2. The weight coefficients w_i combine the Gaussian terms into a single output value and b is a bias term. Building a Gaussian RBF network for a given learning task involves (a) determining the total number of Gaussian basis functions to use for each output class and for the entire system, (b) locating the Gaussian basis function centers, (c) computing the cluster variance for each Gaussian basis function, and (d) solving for the weight coefficients and bias in the summation term. One can implement a binary pattern classifier on input vectors $\mathbf{x}$ as a Gaussian RBF network by defining an appropriate output threshold that separates the two pattern classes.

In this first system, we implement each individual digit recognizer as a spherical Gaussian RBF network, trained with a classical RBF algorithm. Given a specified number of Gaussian basis functions for each digit class, the algorithm separately computes the Gaussian centers and variances for each of the 10 digit classes to form the system's RBF kernels. The algorithm then solves for an optimal set of weight parameters between the RBF kernels and each output node to perform the desired digit recognition task. The training process constructs all 10 digit recognizers in parallel so one can re–use the same Gaussian basis functions among the 10 digit recognizers. To avoid overfitting the available training data with an overly complex RBF classifier connected to every Gaussian kernel, we use a "bootstrap" like operation that selectively connects each recognizer's output node to only a "relevant" subset of all basis functions. The idea is similar to how we choose relevant near–miss clusters for each individual digit recognizer in the original system. The training procedure proceeds as follows (for further details, see Sung, 1996):

1. The first training task is to determine an appropriate number k of Gaussian kernels for each digit class. This information is needed to initialize our clustering procedure for computing Gaussian RBF kernels. We opted for using the same numbers of Gaussian kernels as the ones automatically computed by the SV algorithm (see Table I).
2. Our next task is to compute the Gaussian kernels for each digit class. We do this by separately performing classical k–means clustering (e.g. Lloyd, 1982) on each digit class in the training database. Each clustering operation returns a set of Gaussian centroids and their respective variances for the given digit class. Together, the Gaussian clusters from all 10 digit classes form the system's RBF kernels.
3. For each single-digit recognizer, we build an *initial* RBF network using only Gaussian kernels from its target class, using backpropagation of mean squared error to train the weights (the desired output is set to 1 or 0 for positive and negative examples, respectively). We then separately collect all the false positive mistakes each initial digit recognizer makes on the training database.
4. In the final training step, we augment each initial digit recognizer with additional Gaussian kernels from outside its target class to help reduce misclassification errors. We determine which Gaussian kernels are "relevant" for each recognizer as follows: for each *false positive* mistake the

initial recognizer makes during the previous step, we look up the misclassified pattern's actual digit class and include the nearest Gaussian kernel from its class in the "relevant" set. The *final* RBF network for each single-digit recognizer thus contains every Gaussian kernel from its target class, and several "relevant" kernels from the other 9 digit classes, trained by error backpropagation. Because our final digit recognizers have fewer weight parameters than a naive system that fully connects all 10 recognizers to every Gaussian kernel, we expect our system to generalize better on new data.

B. The Support Vector Machine

B.1 Structural Risk Minimization

For the case of two–class pattern recognition, the task of *learning from examples* can be formulated in the following way: given a set of functions

$$\{f_\alpha : \alpha \in \Lambda\}, \quad f_\alpha : \mathbf{R}^N \to \{-1, +1\}$$

and a set of examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell), \quad \mathbf{x}_i \in \mathbf{R}^N, y_i \in \{-1, +1\},$$

each one generated from an unknown probability distribution $P(\mathbf{x}, y)$, we want to find a function f_{α^*} which provides the smallest possible value for the average error committed on novel examples randomly drawn from P, called the *risk*

$$R(\alpha) = \int \frac{1}{2}|f_\alpha(\mathbf{x}) - y| \; dP(\mathbf{x}, y).$$

The problem is that $R(\alpha)$ is unknown, since $P(\mathbf{x}, y)$ is unknown. Therefore an *induction principle* for risk minimization is necessary.

The straightforward approach to minimize the *empirical risk*

$$R_{emp}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} \frac{1}{2}|f_\alpha(\mathbf{x}_i) - y_i|$$

turns out not to guarantee a small actual risk (i.e. a small error on the training set does not imply a small error on a test set), if the number ℓ of training examples is limited. To make the most out of a limited amount of data, novel statistical techniques have been developed during the last 25 years. The *Structural Risk Minimization* principle (Vapnik, 1979) is based on the fact that for the above learning problem, for any $\alpha \in \Lambda$ with a probability of at least $1 - \eta$, the bound

$$R(\alpha) \leq R_{emp}(\alpha) + \phi\left(\frac{h}{\ell}, \frac{\log(\eta)}{\ell}\right) \tag{2}$$

holds, ϕ being defined as

$$\phi\left(\frac{h}{\ell}, \frac{\log(\eta)}{\ell}\right) = \sqrt{\frac{h\left(\log\frac{2\ell}{h} + 1\right) - \log(\eta/4)}{\ell}}.$$

The parameter h is called the *VC(Vapnik–Chervonenkis)–dimension* of a set of functions. It describes the capacity of a set of functions implementable by the learning machine. For binary classification, h is the maximal number of points which can be separated into two classes in all possible 2^h ways by using functions of the learning machine; i.e. for each possible separation there exists a function which takes the value 1 on one class and -1 on the other class.

According to (2), given a fixed number ℓ of training examples one can control the risk by controlling two quantities: $R_{emp}(\alpha)$ and $h(\{f_\alpha : \alpha \in \Lambda'\})$; Λ' denoting some subset of the index set Λ. The empirical risk depends on the *function* chosen by the learning machine (i.e. on α), and it can be controlled by picking the right α. The VC–dimension h depends on the *set of functions* $\{f_\alpha : \alpha \in \Lambda'\}$ which the learning machine can implement. To control h, one introduces a structure of nested subsets $S_n := \{f_\alpha : \alpha \in \Lambda_n\}$ of $\{f_\alpha : \alpha \in \Lambda\}$,

$$S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots, \tag{3}$$

whose VC–dimensions, as a result, satisfy

$$h_1 \leq h_2 \leq \ldots \leq h_n \leq \ldots$$

For a given set of observations $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_\ell, y_\ell)$ the *Structural Risk Minimization principle* chooses the function $f_{\alpha_\ell^n}$ in the subset $\{f_\alpha : \alpha \in \Lambda_n\}$ for which the guaranteed risk bound (the right hand side of (2)) is minimal.

The remainder of this section follows Schölkopf, Burges & Vapnik (1995) in briefly reviewing the SV algorithm. For details, the reader is referred to (Vapnik, 1995).

B.2 A Structure on the Set of Hyperplanes

Each particular choice of a structure (3) gives rise to a learning algorithm. The SV algorithm is based on a structure on the set of separating hyperplanes. To describe it, first note that given a dot product space $\mathbf{Z}$ and a set of vectors $\mathbf{z}_1, \ldots, \mathbf{z}_r \in \mathbf{Z}$, each hyperplane $\{\mathbf{z} \in \mathbf{Z} : (\mathbf{w}\cdot\mathbf{z}) + b = 0\}$ corresponds to a canonical pair $(\mathbf{w}, b) \in \mathbf{Z} \times \mathbf{R}$ if we additionally require

$$\min_{i=1,\ldots,r} |(\mathbf{w} \cdot \mathbf{z}_i) + b| = 1, \tag{4}$$

i.e. that the scaling of $\mathbf{w}$ and b be such that the point closest to the hyperplane has a distance of $1/||\mathbf{w}||$. Let R be the radius of the smallest ball $B_R(\mathbf{a}) = \{\mathbf{z} \in \mathbf{Z} : ||\mathbf{z} - \mathbf{a}|| < R\}$ $(\mathbf{a} \in \mathbf{Z})$ containing the points $\mathbf{z}_1, \ldots, \mathbf{z}_r$, and

$$f_{\mathbf{w},b} = \operatorname{sgn}\left((\mathbf{w} \cdot \mathbf{z}) + b\right) \tag{5}$$

the decision function defined on these points. The possibility of introducing a structure on the set of hyperplanes is based on the result (Vapnik, 1995) that the set $\{f_{\mathbf{w},b} : ||\mathbf{w}|| \leq A\}$ has a VC–dimension h satisfying

$$h < R^2A^2 + 1. \tag{6}$$

Note. Dropping the condition $||\mathbf{w}|| \leq A$ leads to a set of functions whose VC–dimension equals $N + 1$, where N is the dimensionality of $\mathbf{Z}$. Due to $||\mathbf{w}|| \leq A$, we can get VC–dimensions which are much smaller than N, enabling us to work in very high dimensional spaces.

B.3 The Support Vector Algorithm

Now suppose we want to find a decision function $f_{\mathbf{w},b}$ with the property $f_{\mathbf{w},b}(\mathbf{z}_i) = y_i$, $i = 1, \ldots, \ell$. If this function exists, canonicality (4) implies

$$y_i((\mathbf{w} \cdot \mathbf{z}_i) + b) \geq 1, \quad i = 1, \ldots, \ell. \tag{7}$$

In practice, a separating hyperplane often does not exist. To allow for the possibility of examples violating (7), Cortes & Vapnik (1995) introduce slack variables

$$\xi_i \geq 0, \quad i = 1, \ldots, \ell, \tag{8}$$

to get

$$y_i((\mathbf{w} \cdot \mathbf{z}_i) + b) \geq 1 - \xi_i, \quad i = 1, \ldots, \ell. \tag{9}$$

The SV approach to minimizing the guaranteed risk bound (2) consists of the following: minimize

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + \gamma \sum_{i=1}^{\ell} \xi_i \tag{10}$$

subject to the constraints (8) and (9). Due to (6), minimizing the first term is related to minimizing the VC–dimension of the considered class of learning machines, thereby minimizing the second term of the bound (2) (it also amounts to maximizing the separation margin, cf. the remark following (4)). The term $\sum_{i=1}^{\ell} \xi_i$, on the other hand, is an upper bound on the number of misclassifications on the training set — this controls the empirical risk term in (2). For a suitable positive constant γ, this approach therefore constitutes a practical implementation of Structural Risk Minimization on the given set of functions.

Introducing Lagrange multipliers α_i, and using the Kuhn–Tucker theorem of optimization theory, the solution can be shown to have an expansion

$$\mathbf{w} = \sum_{i=1}^{\ell} y_i \alpha_i \mathbf{z}_i, \tag{11}$$

with nonzero coefficients α_i only where the corresponding example $(\mathbf{z}_i, y_i)$ precisely meets the constraint (9). These $\mathbf{z}_i$ are called *Support Vectors*. All remaining examples of the training set are irrelevant: their constraint (9) is satisfied automatically (with $\xi_i = 0$), and they do not appear in the expansion (11). The coefficients α_i are found by solving the following quadratic programming problem: maximize

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j (\mathbf{z}_i \cdot \mathbf{z}_j) \tag{12}$$

subject to

$$0 \leq \alpha_i \leq \gamma, \quad i = 1, \ldots, \ell, \quad \text{and} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0. \tag{13}$$

By linearity of the dot product, the decision function (5) can thus be written as

$$f(\mathbf{z}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot (\mathbf{z} \cdot \mathbf{z}_i) + b \right). \tag{14}$$

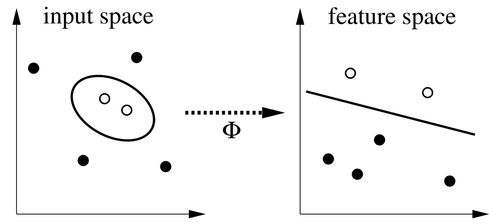

Fig. 3. By the use of a nonlinear kernel function (15), it is possible to compute a separating hyperplane with maximum margin in a feature space without explicitly mapping into that space. The feature space is related to input space via a nonlinear map Φ, causing the decision surface to be nonlinear in input space (e.g. of a Gaussian RBF type).

So far, we have described the case of linear decision surfaces (Vapnik & Chervonenkis, 1974). To allow for much more general decision surfaces, one can first nonlinearly transform a set of input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ into a high–dimensional feature space by a map $\Phi : \mathbf{x}_i \mapsto \mathbf{z}_i$ and then do a linear separation there. Maximizing (12) and evaluating (14) then requires the computation of dot products $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i))$ in a high–dimensional space. We are interested in cases where these expensive calculations can be reduced significantly by using a suitable function K such that

$$(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i)) = K(\mathbf{x}, \mathbf{x}_i), \tag{15}$$

leading to decision functions of the form

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}, \mathbf{x}_i) + b \right). \tag{16}$$

In practise, we need not worry about conceiving the map Φ. We will choose a K which is the continuous Kernel of a positive definite integral operator, and Mercer's theorem of functional analysis then tells us that K can be expanded in a uniformly convergent series in terms of Eigenfunctions ψ_j and positive Eigenvalues λ_j,

$$K(\mathbf{x}, \mathbf{x}_i) = \sum_{j=1}^{\infty} \lambda_j \psi_j(\mathbf{x}) \psi_j(\mathbf{x}_i),$$

corresponding to a dot product $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{x}_i))$ in the (possibly infinite–dimensional) range of

$$\Phi : \mathbf{x} \mapsto (\sqrt{\lambda_1}\psi_1(\mathbf{x}), \sqrt{\lambda_2}\psi_2(\mathbf{x}), \ldots)$$

(see Boser, Guyon, & Vapnik, 1992; Vapnik, 1995). Consequently, everything that has been said about the linear case also applies to nonlinear cases obtained by using a suitable kernel K instead of the Euclidean dot product (Fig. 3). We are now in a position to explain how the SV algorithm can construct RBF classifiers: we simply use

$$K(\mathbf{x}, \mathbf{x}_i) = \exp\left(-\|\mathbf{x} - \mathbf{x}_i\|^2 / c\right) \tag{17}$$

(see Aizerman, Braverman & Rozonoer, 1964). Other possible choices of K include $K(\mathbf{x}, \mathbf{x}_i) = (\mathbf{x} \cdot \mathbf{x}_i)^d$, yielding polynomial classifiers of degree d, and $K(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa \cdot (\mathbf{x} \cdot \mathbf{x}_i) + \Theta)$ for constructing neural networks.

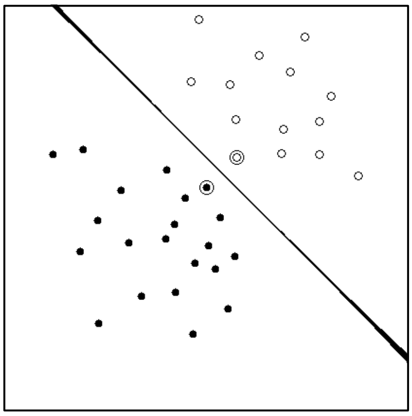

Fig. 4. A simple two–class classification problem as solved by the SV algorithm ($c_i = 1$ for all i; cf. Eq. 1). Note that the RBF centers (indicated by extra circles) are closest to the decision boundary.

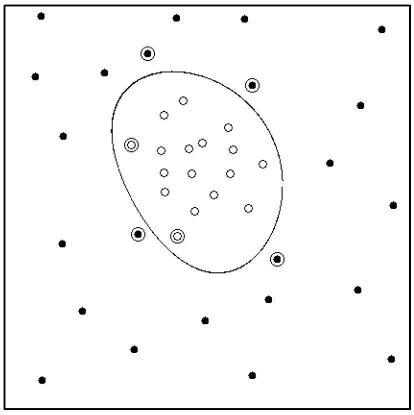

Fig. 5. Two–class classification problem solved by the SV algorithm ($c_i = 1$ for all i; cf. Eq. 1).

To find the decision function (16), we maximize (cf. (12))

$$W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{18}$$

subject to the constraints (13). Since K is required to satisfy Mercer's conditions, it corresponds to a dot product in another space (15), thus $(y_i y_j K(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ is a positive matrix, providing us with a problem that can be solved efficiently. To compute the threshold b, one takes into account that due to (9), for Support Vectors $\mathbf{x}_j$ for which $\xi_j = 0$ we have $\sum_{i=1}^{\ell} y_i \alpha_i \cdot K(\mathbf{x}_j, \mathbf{x}_i) + b = y_j$.

B.4 Further Remarks on SV Research

The SV algorithm has been empirically shown to exhibit good generalization ability (Cortes & Vapnik, 1995). This applies for different types of SV machines obtained by using different kernel functions (cf. (17)); moreover, different machines have been found to use largely the same SVs; e.g. most of the centers of an SV machine with Gaussian kernel coincide with the first–layer weights of neural network SV classifiers (Schölkopf, Burges, & Vapnik, 1995).

The accuracy can be further improved by incorporating invariances of the problem at hand, as with the Virtual SV method. There, one generates artificial examples by transforming SVs and retrains on these in a second training run, with a computational complexity comparable to the first one (Schölkopf, Burges, & Vapnik, 1996). In addition, the decision rule (16), which requires the evaluation of kernel functions for the test example and all SVs, can be sped up with the reduced set technique (Burges, 1996) by approximating (16) with an expansion using fewer terms. These methods have led to substantial improvements for polynomial SV machines — Burges & Schölkopf (1997) cut the error rate on a character recognition task by a third, and obtained a speed–up by a factor of 50 —, and they are directly applicable also to Gaussian SV machines.

Besides pattern recognition, modified versions of the SV algorithm have been used for regression estimation and time series prediction (Vapnik, 1995; Vapnik, Golowich, & Smola, 1997; Mukherjee, Osuna, & Girosi, 1997; Müller et al., 1997). Moreover, the kernel method for computing dot products in high–dimensional feature spaces has been applied in other domains, as for instance nonlinear principal component analysis (Schölkopf, Smola, & Müller, 1996).[1]

C. A Hybrid System: RBF with SV centers

The previous section discusses how one can train RBF like networks using the SV algorithm (cf. (16), (17)). This involves the choice of an appropriate kernel function K and solving the quadratic optimization problem (18). The SV algorithm thus automatically determines the centers (which are the Support Vectors), the weights (given by $y_i\alpha_i$), and the threshold b for the RBF machine.

To assess the relative influence of the automatic SV center choice and the SV weight optimization, respectively, we built another RBF system, constructed with centers that are simply the Support Vectors arising from the SV optimization, and with the weights trained separately.

D. Addenda

D.1 Computational Complexity

By construction, the resulting classifiers after training will have the same architecture and comparable sizes. Thus the three machines are comparable in classification speed and memory requirements.

Differences are, however, noticeable in training. Regarding training time, the SV machine was faster than the RBF system by about an order of magnitude, due to the formulation as a quadratic programming problem which can be solved efficiently. The optimization, however, requires to work with potentially large matrices. In the implementation that we used, the training data is processed in chunks, and matrix sizes were of the order 500×500. For problems with very large numbers of SVs, a modified training algorithm has recently been proposed by Osuna, Freund, and Girosi (1997).

[1]More information on SV methods can be obtained via http://www.mpik-tueb.mpg.de/people/personal/bs/svm.html.

TABLE I

NUMBERS OF CENTERS (SUPPORT VECTORS) AUTOMATICALLY EXTRACTED BY THE SV MACHINE. THE FIRST ROW GIVES THE TOTAL NUMBER FOR EACH BINARY CLASSIFIER, INCLUDING BOTH POSITIVE AND NEGATIVE EXAMPLES; IN THE SECOND ROW, WE ONLY COUNTED THE POSITIVE SUPPORT VECTORS. THE LATTER NUMBER WAS USED IN THE INITIALIZATION OF THE k–MEANS ALGORITHM, CF. SEC. II-A.

digit class	0	1	2	3	4	5	6	7	8	9
# of Support Vectors	274	104	377	361	334	388	236	235	342	263
# of positive Support Vectors	172	77	217	179	211	231	147	133	194	166

TABLE II

BINARY CLASSIFICATION: NUMBERS OF TEST ERRORS (OUT OF 2007 TEST PATTERNS) FOR THE SYSTEMS DESCRIBED IN SECTIONS II-A – II-C.

digit class	0	1	2	3	4	5	6	7	8	9
classical RBF	20	16	43	38	46	31	15	18	37	26
RBF with SV centers	9	12	27	24	32	24	19	16	26	16
full SV machine	16	8	25	19	29	23	14	12	25	16

D.2 Different Ways of Training an RBF Classifier

Due to (9) and (10), the SV algorithm puts emphasis on correctly separating the training data. In this respect, it is different from the classical RBF approach of training in the least–squares metric, which is more concerned with the general problem of estimating posterior probabilities than with directly solving a classification task at hand. There exist, however, studies investigating the question how to select RBF centers or exemplars to minimize the number of misclassifications, see for instance (Chang & Lippmann, 1993), (Duda & Hart, 1973), (Reilly, Cooper, & Elbaum, 1982), (Barron, 1984). A classical RBF system could also be made more discriminant by using moving centers (e.g. Poggio & Girosi, 1990), or a different cost function, as the classification figure of merit (Hampshire & Waibel, 1990). In fact, it can be shown that Gaussian RBF regularization networks are equivalent to SV machines if the regularization operator and the cost function are chosen appropriately (Smola & Schölkopf, 1997).

It is important to stress that the SV machine does not minimize the empirical risk (misclassification error on the training set) alone. Instead it minimizes the sum of an upper bound on the empirical risk and a penalty term that depends on the complexity of the classifier used.

III. EXPERIMENTAL RESULTS

A. *Toy examples*

What are the Support Vectors? They are elements of the data set that are "important" in separating the two classes from each other. Support Vectors with zero slack variables (8) lie on the boundary of the decision surface, as they precisely satisfy the inequality (9) in the high–dimensional space. Figures 4 and 5 illustrate that for the used Gaussian kernel, this is also the case in input space. This raises an interesting question from the point of view of interpreting the structure of trained RBF networks. The traditional view of RBF networks has been one where the centers were regarded as "templates" or stereotypical patterns. It is this point of view that leads to the clustering heuristic for training RBF networks. In contrast, the SV machine posits an alternate point of view, with the centers being those examples which are critical for a given classification task.

B. *US Postal Service Database*

We used the USPS database of 9300 handwritten digits (7300 for training, 2000 for testing), collected from mail envelopes in Buffalo (cf. LeCun et al., 1989). Each digit is a 16×16 image, represented as a 256–dimensional vector with entries between -1 and 1. Preprocessing consisted of smoothing with a Gaussian kernel of width $\sigma = 0.75$. The SV machine results reported in the following were obtained with $\gamma = 10$ and $c = 0.3 \cdot N$ (cf. (10), (17)), where $N = 256$ is the dimensionality of input space.[2] In all experiments, we used the SV algorithm with standard quadratic programming techniques (conjugate gradient descent).

Two–class classification. Table I shows the numbers of Support Vectors, i.e. RBF centers, extracted by the SV algorithm. Table II gives the results of binary classifiers separating single digits from the rest, for the systems described in Sections II-A, II-B, and II-C.

Ten–class classification. For each test pattern, the arbitration procedure in all three systems simply returns the digit class whose recognizer gives the strongest response.[3] Table III shows the 10-class digit recognition error rates for our original system and the two RBF-based systems.

The fully automatic SV machine exhibits the highest test accuracy of the three systems.[4] Using the SV algorithm to choose the centers for the RBF network is also better than

[2] The SV machine is rather insensitive to different choices of c. For all values in $0.1, 0.2, \ldots, 1.0$, the performance is about the same (in the area of 4% – 4.5%).

[3] In the SV case, we constructed ten two–class classifiers, each trained to separate a given digit from the other nine. Ten–class classification was done according to the maximal output before applying the sgn function, among the two–class classifiers. These outputs can be used for reject decisions (e.g. Cortes & Vapnik, 1995).

[4] An analysis of the errors showed that about 85% of the errors committed by the SV machine were also made by the other systems. This makes the differences in error rates very reliable.

TABLE III

10-class digit recognition error rates for three RBF classifiers constructed with different algorithms. The first system is a classical one, choosing its centers by k–means clustering. In the second system, the Support Vectors were used as centers, and in the third one, the entire network was trained using the SV algorithm.

USPS Database	Classification Error Rate		
	classical RBF	RBF with SV centers	full SV machine
Training (7291 patterns)	1.7%	0.0%	0.0%
Test (2007 patterns)	6.7%	4.9%	4.2%

the baseline procedure of choosing the centers by a clustering heuristic. It can be seen that in contrast to the k–means cluster centers, the centers chosen by the SV algorithm allow zero training error rates.

The considered recognition task is known to be rather hard — the human error rate is 2.5%, almost matched by a memory–based Tangent–distance classifier (2.6%, Simard, LeCun, & Denker, 1993). Other results on this data base include a Euclidean distance nearest neighbour classifier (5.9%, Simard, LeCun, & Denker, 1993), a two-layer perceptron, 5.9%, and a convolutional neural network (5.0%, Le Cun et al., 1989). By incorporating translational and rotational invariance using the Virtual SV technique (Schölkopf, Burges, & Vapnik, 1996; cf. our remark at the end of Sec. II-B.3), we were able to improve the performance of the considered Gaussian kernel SV machine (same values of γ and c) from 4.2% to 3.2% error.

IV. Summary and Discussion

The SV algorithm provides a principled way of choosing the number and the locations of RBF centers. Our experiments on a real–world pattern recognition problem have shown that *in contrast to a corresponding number of centers chosen by k–means, the centers chosen by the SV algorithm allowed a training error of zero*, even if the weights were trained by classical RBF methods. The interpretation of this finding is that the SV centers are specifically chosen for the classification task at hand, whereas k–means does not care about picking those centers which will make a problem separable.[5]

In addition, *the SV centers yielded lower test error rates than k–means.* It is interesting to note that using SV centers, while sticking to the classical procedure for training the weights, improved training and test error rates by approximately the same margin (2 per cent). In view of the guaranteed risk bound (2), this can be understood in the following way: the improvement in test error (risk) was solely due to the lower value of the training error (empirical risk); the confidence term (the second term on the right hand side of (2)), depending on the VC–dimension and thus on the norm of the weight vector, did not change, as we stuck to the classical weight training procedure. However, when we also trained the weights with the SV algorithm, we minimized the norm of the weight vector (see (10), (6)) in feature space, and thus the confidence term, while still keeping the training error zero. Thus, consistent with (2), *the Support Vector machine achieved the highest test accuracy of the three systems.*

[5] As pointed out in Sec. II-D.2, the classical system could be made more discriminant, too, by using different cost functions, e.g. the classification figure of merit (Hampshire & Waibel, 1990), or by using moving centers (e.g. Poggio & Girosi, 1990).

References

[1] Aizerman, M.; Braverman, E.; Rozonoer, L. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, **25**:821-837.

[2] Barron, A. 1984. Predicted squared error: a criterion for automatic model selection. In: S. Farlow (ed.), *Self-organizing Methods in Modeling*, New York, Marcel Dekker.

[3] Boser, B. E.; Guyon, I. M.; and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh ACM 144–152.

[4] Burges, C. J. C. 1996. Simplified Support Vector Decision Rules. In: Saitta, L., *Proceedings, 13th Intl. Conf. on Machine Learning*, Morgan Kaufmann, San Mateo, CA.

[5] Burges, C.J.C., and Schölkopf, B. 1997. Improving the Accuracy and Speed of Support Vector Machines. In: Mozer, M.; Jordan, M.; Petsche, T., *Advances in Neural Information Processing Systems 9.*, MIT Press, Cambridge, MA.

[6] Chang, E. I., and Lippmann, R. L. 1993. A boundary hunting radial basis function classifier which allocates centers constructively. In: Hanson, S. J.; Cowan, J. D.; Giles, C. L., *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, CA.

[7] Cortes, C.; and Vapnik, V. 1995. Support Vector Networks. *Machine Learning*, **20**:273 – 297.

[8] Duda, R. O.; and Hart, P. E. 1973. *Pattern Classification and Scene Analysis*, Wiley, New York.

[9] Hampshire, J. B.; and Waibel, A. 1990. A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks. *IEEE Trans. Neural Networks*, **1**:216 – 228.

[10] Le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. J. 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation* **1**: 541 – 551.

[11] Lloyd, S. P. 1982. Least squares quantization in PCM. *IEEE Trans. Information Theory*, IT-28:129 – 137.

[12] Mukherjee, S.; Osuna, E.; Girosi, F. 1997. Nonlinear Prediction of Chaotic Time Series using a Support Vector Machine. To appear in: *NNSP'97.*

[13] Müller, K.-R.; Smola, A.; Rätsch, G.; Schölkopf, B.; Kohlmorgen, J.; Vapnik, V. 1997. Predicting Time Series with Support Vector Machines. Submitted to *ICANN'97.*

[14] Osuna, E.; Freund, R.; Girosi, F. 1997. Improved Training Algorithm for Support Vector Machines. To appear in: *NNSP'97.*

[15] Poggio, T.; & Girosi, F. 1990. Networks for approximation and learning. *Proc. IEEE*, **78**:1481 – 1497.

[16] Reilly, D.; Cooper, L. N.; Elbaum, C. 1982. A neural model for category learning. Biol. Cybern., **45**:35 – 41.

[17] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. 1986. Learning representations by back–propagating errors. *Nature*, **323**:533 – 536.

[18] Schölkopf, B.; Burges, C.; and Vapnik, V. 1995. Extracting support data for a given task. In: Fayyad, U. M., and Uthurusamy, R. (eds.): *Proceedings, First Intl. Conf. on Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA.

[19] Schölkopf, B., Burges, C.J.C., Vapnik, V. 1996. Incorporating Invariances in Support Vector Learning Machines. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, and B. Sendhoff, editors, *Artificial Neural Networks — ICANN'96*, pages 47 – 52, Berlin. (Springer Lecture Notes in Computer Science, Vol. 1112.)

[20] Schölkopf, B.; Smola, A.; and Müller, K.-R. 1996. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max–Planck–Institut für biologische Kybernetik, Tübingen, Germany. Submitted to *Neural Computation*.

[21] Simard, P.; Le Cun, Y.; Denker, J. 1993. Efficient Pattern Recognition Using a New Transformation Distance. In Moody, J. E., Hanson, S. J., Lippmann, R. P., *Advances in Neural Information Processing Systems 5*, Morgan Kaufmann, San Mateo, CA.

[22] Smola, A.; and Schölkopf, B. 1997. On a kernel–based method for pattern recognition, regression, approximation and operator inversion. Technical Report 1064, GMD, Berlin.

[23] Sung, K. 1995. Learning and Example Selection for Object and Pattern Detection. Ph.D. Thesis, Massachusetts Institute of Technology.

[24] Vapnik, V. 1979. *Estimation of Dependences Based on Empirical Data*, [in Russian] Nauka, Moscow; English translation: Springer–Verlag, New York, 1982.

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

[26] Vapnik, V.; Chervonenkis, A. 1974. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979.)

[27] Vapnik, V.; Golowich, S.; Smola, A. 1997. Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing. In: Mozer, M.; Jordan, M.; Petsche, T., *Advances in Neural Information Processing Systems 9.*, MIT Press, Cambridge, MA.

Bernhard Schölkopf is a doctoral student with Heinrich Bülthoff (Max–Planck–Institut für biologische Kybernetik) and Vladimir Vapnik (AT&T Research). He received an M.Sc. in mathematics and the Lionel Cooper Memorial Prize from the University of London in 1992. Two years later, he received the Diplom in physics from the Eberhard–Karls–Universität, Tübingen, Germany. His scientific interests include machine learning and perception.

Kah–Kay Sung received his Ph.D. degree in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1995. He is currently a lecturer at the Department of Information Systems and Computer Science, National University of Singapore. His research interests include computer vision and machine learning.

Chris J. C. Burges has a B.A., First Class, in Physics from Oxford University, and a Ph.D. in particle physics from Brandeis University. After a two year postdoctoral position at MIT in theoretical particle physics, he joined AT&T, and developed the routing algorithm AT&T now uses to ensure physical diversity of its signaling links. He then started working on applying neural networks to handwriting recognition, and developed several methods now used by banks in automated check readers. For the last two years he has been working on developing Support Vector algorithms with V. Vapnik and others, concentrating on the pattern recognition problem, methods to speed up support vector machines, and the theory of kernel based methods.

Federico Girosi, a Principal Research Scientist in the Department of Brain and Cognitive Sciences at MIT, is Associate Director of the Center for Biological & Computational Learning at MIT. Dr. Girosi graduated with a Ph.D. in Theoretical Physics from the University of Genoa, and has worked in the fields of neural networks, approximation theory and computer vision. His current research topics include: the study of neural networks architectures, the complexity of the approximation problem, Support Vector Machines and their applications to computer vision.

Partha Niyogi received the B.Tech. degree from IIT, New Delhi and the Ph.D. in Electrical Engineering and Computer Science from MIT. His research interests are in pattern recognition and learning theory and its application to problems in speech and language processing. He is currently with Bell Laboratories in Murray Hill, New Jersey.

Tomaso Poggio ...

Vladimir Vapnik was born in Russia and received his Ph.D. degree in theoretical cybernetics from the Institute of Control Sciences at the Academy of Science of the USSR (Moscow) in 1964. Since 1991, he has been working for AT&T Bell Laboratories (since 1996, AT&T Labs Research). His research interests include statistical learning theory, theoretical and applied statistics, theory and methods of solving stochastic ill–posed problems, and methods of multidimensional function approximation. His main results in the last three years are related to the development of the Support Vector method.

He is author of many publications, including 7 monographs on various problems of statistical learning theory.