ELSEVIER

Pattern Recognition Letters 20 (1999) 475–484

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

On domain knowledge and feature selection using a support vector machine

Ofir Barzilay [1], V.L. Brailovsky *

Department of Computer Science, Tel Aviv University, 69978 Ramat Aviv, Israel

Received 8 July 1998; received in revised form 24 November 1998

Abstract

The basic principles of a support vector machine (SVM) are analyzed. The problem of feature selection while using an SVM is specifically addressed. An approach to constructing a kernel function which takes into account some domain knowledge about a problem and thus essentially diminishes the number of noisy parameters in high dimensional feature space is suggested. Its application to Texture Recognition is described. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Domain knowledge; Feature selection; Support vector machine; Texture recognition

1. Introduction

A Support Vector Machine (SVM) is a universal learning machine developed in recent years. SVMs are known to give good results in many Pattern Recognition (PR) problems. The basic principles of an SVM according to Guyon et al. (1992) and Cortes and Vapnik (1995) and Vapnik (1995) are as follows (we consider here a two-class problem):

1. The SVM performs a nonlinear mapping of the input vectors (objects) from the input space R_d into a high dimensional feature space H; the mapping is determined by a kernel function.
2. It finds a linear decision rule in the feature space H in the form of an optimal separating plane, which is the one that leaves the widest margin between the decision boundary and the input vectors mapped into H.
3. This plane is found by solving the following constrained quadratic programming problem:

   Maximize $W(\alpha)$

   $$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \qquad (1)$$

   under the constraints $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $0 \leqslant \alpha_i \leqslant C$ for $i = 1, 2, \ldots, n$ where $x_i \in R_d$ are the training sample set (TSS) vectors, and $y_i \in \{-1, +1\}$ the corresponding class labels. C is a constant needed for nonseparable classes; $K(u, v)$ is a kernel function defined in 6.
4. The separating plane is constructed from those input vectors, for which $\alpha_i \neq 0$. These vectors are called *support vectors* and reside on the boundary margin. The number of support vectors determines the accuracy and speed of the SVM.

* Corresponding author. Tel.: +972-3-6490440; fax: +972-3-640-9357; e-mail: brail@math.tau.ac.il

[1] E-mail: barzilay@math.tau.ac.il

0167-8655/99/$ – see front matter
PII: S0167-8655(99)00014-8

5. Mapping the separating plane back into the input space R_d, gives a separating surface which forms the following nonlinear decision rule:

$$y(x) = \operatorname{sign}\left[\sum_{x_i \in \mathrm{sv}} \alpha_i y_i K(x, x_i) + b\right]. \tag{2}$$

   Here sv denotes the set of support vectors.
6. $K(u, v)$ is an inner product in the feature space H which may be defined as a kernel function in the input space. Conditions required are that the kernel $K(u, v)$ be a symmetric function which satisfies the following general positivity constraint:

$$\iint_{R_d} K(u, v) g(u) g(v)\, \mathrm{d}u\, \mathrm{d}v > 0, \tag{3}$$

   which is valid for all $g \neq 0$ for which $\int g^2(u)\, \mathrm{d}u < \infty$ (Mercer's theorem).
7. The choice of the kernel $K(u, v)$ determines the structure of the feature space H. A kernel which satisfies (3) may be presented in the form

$$K(u, v) = \sum_k a_k \psi_k(u) \psi_k(v), \tag{4}$$

   where a_k are positive scalars and the functions $\psi_k()$ represent a basis in the space H.
8. Guyon et al. (1992) and Vapnik (1995) consider three types of support vector machines:
   - Polynomial SVM:

$$K(u, v) = [(u, v) + 1]^d. \tag{5}$$

     This type of machine describes the mapping of an input space into the feature space that contains all products $u_i v_i u_j v_j u_k v_k \ldots$ up to degree d.
   - Radial Basis Function SVM:

$$K_\gamma(u, v) = \exp\{-\gamma |u - v|^2\}. \tag{6}$$

     This type of machine forms a separation rule according to radial hyperplanes.
   - Two-layer neural network SVM:

$$K(u, v) = \tanh(w(u, v) + c). \tag{7}$$

     This type of machine generates a kind of two–layer neural network without back propagation.
9. The kernel $K(u, v)$ should be chosen a priori. Other parameters of the decision rule (2) are determined by calculating (1), i.e. the set of numerical parameters $\{\alpha_i\}_1^n$ which determines the support vectors and the scalar b.

The success of the SVM in many applied PR problems surprised the PR community since the principles of the SVM somehow contradict the generally held intuitions and beliefs. The usual approach suggests that there is a small number of features which may be obtained with the help of feature selection and extraction methods. We map the TSS into the space based on these features, i.e. into a space with dimensionality much smaller than the original one, and then try to find a good classification rule in this feature space.

In the SVM a kind of inverse procedure is used. Instead of decreasing dimensionality we map the TSS into a space of very high dimensionality, i.e. we add a great number of new features without checking their efficiency, and find a classification rule with some optimal properties in this feature space. The success of the SVM in solving PR problems raises the question: Are feature selection and extraction procedures needed when using SVM?

This article examines the above question. In Section 2 we analyze a synthetic problem in which we check how fast results deteriorate when the dimensionality of input space increases, provided the TSS size is fixed. We compare the classification results of the SVM with those obtained by traditional classifiers.

In Sections 3 and 4 we consider some natural problems of Texture Recognition (TR) when the texture is generated with the help of generally accepted MRF model. So, it is another example of a problem with many weak features in the input space (a feature is a pixel value).

We consider how to define a kernel $K(u, v)$ that takes into account domain knowledge (that the solution is determined by the interaction of neighbor pixels), and as a result, essentially diminishes the number of "noisy features" in the feature space H (in comparison to the standard definition of kernel functions (5)–(7)).

In recent years there was a number of articles devoted to study of connection between domain knowledge and the form of kernel function. E.g. in Scholkoph et al. (1997) some kernels that take into

account interactions only within a certain neighborhood of pixels are considered (so called piramidal receptive fields) and an improvement of the results of a digit recognition problem is demonstrated. Our approach is close to one accepted in this paper. At the same time we consider another problem (TR) and other specific forms of the kernel functions. We demonstrate that the usage of such kernels permits solving some TR problems that cannot be solved with the help of standard kernel functions.

As demonstrated by Vapnik (1995), the ratio of the number of support vectors to the TSS size is an upper bound for the error rate estimate obtained using the leave-one-out procedure. It may lead one to estimate the quality of a solution by this ratio. In Section 3 we show that there are some "natural problems" like TR where this upper bound is very far from the real error estimate and therefore is an inadequate estimate.

2. Patterns with many weak features

To check the ability of an SVM to cope with a problem where a large number of weak features are involved, we created a synthetic database of normally distributed vectors.

2.1. Normally distributed vectors

Two classes of vectors were generated according to the general multivariate normal density function:

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \times \exp\left[-\frac{1}{2}(x-\mu_i)^t\Sigma^{-1}(x-\mu_i)\right], \quad i = 1,2, \tag{8}$$

where x is a d component vector, μ_i are the d component mean vectors, and Σ is the $d \times d$ covariance matrix which is identical for both classes.

In the case of two classes with equal covariance matrices and a priori probabilities, the optimal decision rule can be stated very simply (see Duda and Hart, 1973): To classify a vector x, measure the squared Mahalanobis distance $(x-\mu_i)^t\Sigma^{-1}(x-\mu_i)$ from x to each of the mean vectors, and assign x to the category of the closest mean. Formally the optimal classification function is

$$F(x) = \operatorname{sign}[(\mu_1-\mu_2)^t\Sigma^{-1}x - \frac{1}{2}(\mu_1^t\Sigma^{-1}\mu_1 - \mu_2^t\Sigma^{-1}\mu_2)]. \tag{9}$$

This classification rule is called linear discriminant analysis (LDA), because the decision function is linear.

Usually in classification problems, the values of the mean vectors and the covariance matrix are not known. One way of evaluating these values is by using the maximum likelihood method which gives the well known (see Anderson, 1958) estimates for μ_i and Σ:

$$\hat{\mu}_i = \frac{1}{n}\sum_{x_k \in w_i} x_k, \tag{10}$$

$$\hat{\Sigma} = \frac{1}{2n}\sum_{i=1}^{2}\sum_{x_k \in w_i}(x_k-\hat{\mu}_i)(x_k-\hat{\mu}_i)^t, \tag{11}$$

where n is the number of samples from each class w_1 and w_2. According to Duda and Hart (1973), as the TSS size $(2n)$ decreases the estimates for μ_i and Σ lose accuracy, and when $2n$ is less than the dimension of the input vectors, the sample covariance matrix is singular. This, of course, causes a high error rate and failure of the classifier since the covariance matrix cannot be inverted.

Friedman (1989) suggested a method to regularize the covariance matrix, using the following equation:

$$\hat{\Sigma}(\lambda) = (1-\lambda)\hat{\Sigma} + \frac{\lambda}{d}\operatorname{trace}[\hat{\Sigma}]I, \tag{12}$$

where λ is a regularization parameter $(0 \leqslant \lambda \leqslant 1)$, d is the dimension of the input space, $\operatorname{trace}[\Sigma]$ is the sum of elements on the diagonal of the covariance matrix Σ, and I is the identity matrix.

This approach is called regularized discriminant analysis (RDA). When $\lambda = 0$ we get the LDA classifier, and when $\lambda = 1$, the RDA approach corresponds with the nearest means classifier where the input pattern is assigned to the class with the closest mean.

2.2. Test results

We generated a training sample set containing two classes of vectors. The vectors were distributed according to multivariate normal density functions. Both classes had the same covariance matrix Σ but different mean vectors μ_1 and μ_2:

$$\Sigma = \mathrm{diag}\{83, 83, 83, 120, 333, 333, 50, \sigma^2, \ldots, \sigma^2\},$$
$$\mu_1 = (-5, 0, 10, -2, 25, -5, 7.75, \mu, \ldots, \mu),$$
$$\mu_2 = (5, 10, 0, 10, 5, -25, 0, \mu, \ldots, \mu),$$

where $\mu = 10$ and $\sigma^2 = 100$.

The first seven coordinates in μ_1 and μ_2 are different and serve as strong features; all other coordinates are identical and serve as noisy features. The two classes are not separable; there are vectors from one class that lie within the decision area of the other class and vice versa. The minimal error rate for the above distribution parameters may be easily calculated (Duda and Hart, 1973) and is about 7.4%. We applied an orthonormal matrix to the generated vectors. The resulting vectors had features that are actually weighted sums of the features of the original vectors, meaning that all their features were weak.

We used vectors with up to 900 features in order to check the feature extraction of the SVM. In all cases the generated TSS contained 90 vectors from each class. The support vector machine built a classification rule according to the TSS. The classification was checked using a test set of 1000 vectors from each class, distributed according to the same multivariate normal density functions.

We compared the error rate with the classification results obtained by using LDA and RDA classifiers, where the values of the mean vector and the covariance matrix were estimated by (10)–(12) accordingly. For the RDA method we chose for each test the regularization parameter λ which produced best classification results.

Table 1 presents the results obtained, rows represent the classification methods and columns the number of features. The results displayed under each feature number are the average error rate percentage over 10 experiments.

We see that the quality of decision rules is deteriorating while the number of the input features is increasing. When the number of these features reaches some hundreds there is no major difference between results obtained by RDA and SVM classifiers. When the number of the input features is relatively small (10–50) all the classifiers give much better results.

So, if one faces a problem with many input features of the type which is modeled here, and there is a domain knowledge which permit to perform feature extraction (selection) and to diminish the number of the input features, it should be done for all types of classifiers including SVM.

3. Domain knowledge consideration

In the next two sections we demonstrate that in many problems there is a need to diminish the number of features in a feature space. This problem may be reduced to the problem how to choose the kernel $K(u, v)$ that defines a scalar product.

We present an important theorem which can help to build kernel functions that take into consideration the domain knowledge of a problem, and demonstrate such a kernel function for texture image in which classification is determined according to the interaction of neighborhood pixels.

Table 1
Classification results of normally distributed vectors

Feature number	10	50	90	130	170	210	300	500	700	900
LDA	9	12	18	29	43	–	–	–	–	–
RDA	10	11	12	14	15	15	17	19	20	23
Polynomial SVM $d=1$	11	13	15	16	17	17	20	21	21	22
Polynomial SVM $d=2$	11	13	14	16	16	16	18	19	19	20
Radial SVM $\gamma=30$	11	11	13	14	15	16	18	19	19	19

3.1. Sum of kernel functions

Theorem. *If a kernel function $K(x,y)$ can be presented in the form*

$$K(x,y) = \sum_{i=1}^{m} K_i(x,y), \tag{13}$$

where each of the functions $K_i(x,y)$ complies with the condition of Mercer's theorem (3), *then $K(x,y)$ also complies with that condition and may be used as a kernel function for an SVM.*

The proof is straightforward.

3.2. SVM neighborhood kernel function

Consider a two-dimensional image consisting of a matrix of $M = N \times N$ pixels. Furthermore suppose that from the domain knowledge it can be assumed that the classification rule may be obtained as a result of neighborhood pixel interaction. Consider a central site with the index t and a relative numbering system for the geometric neighborhood of site t (see Fig. 1).

For each site t we define the partial neighborhood kernel function of degree $d = 2$:

$$K_t(x,y) = \sum_{r=-s}^{s} x_t x_{t:+r} y_t y_{t:+r}, \tag{14}$$

where $s = 2$ for order 1 neighborhoods, $s = 4$ for order 2 neighborhoods, etc., $\{x_i\}$ stand for pixel values of one image and $\{y_i\}$ for the corresponding pixels of the other image.

Eq. (14) is for a central site in the image; for edge and corner sites the formula should be corrected accordingly.

$t:-11$	$t:-7$	$t:-6$	$t:+8$	$t:+12$
$t:-9$	$t:-3$	$t:-2$	$t:+4$	$t:+10$
$t:-5$	$t:-1$	t	$t:+1$	$t:+5$
$t:-10$	$t:-4$	$t:+2$	$t:+3$	$t:+9$
$t:-12$	$t:-8$	$t:+6$	$t:+7$	$t:+11$

5	4	3	4	5
4	2	1	2	4
3	1	t	1	3
4	2	1	2	4
5	4	3	4	5

Fig. 1. Order and numbering for the geometric neighbors of site t.

On the basis of the kernel sum theorem, we introduce the neighborhood kernel function of degree $d = 2$:

$$K(x,y) = \sum_{t=1}^{M} K_t(x,y). \tag{15}$$

In this way [2] we define a kernel that describes the mapping of the input space (described by the function value in M sites of the image) into a feature space that contains the products $x_i y_i$ of the function values only in the neighborhood pixels. This significantly diminishes the dimensionality of the feature space H in comparison with the general polynomial kernel function (5) of degree 2.

When analyzing a binary image with the values of the function $x_t, y_t = \pm 1$ the multiplications in (14) indicate

$$x_t x_{t:+r} y_t y_{t:+r} = \begin{cases} +1 & \text{if } (x_t = x_{t:+r} \text{ and } y_t = y_{t:+r}) \\ & \text{or } (x_t \neq x_{t:+r} \text{ and } y_t \neq y_{t:+r}), \\ -1 & \text{otherwise.} \end{cases} \tag{16}$$

The neighborhood kernel function of degree $d = 3$ is defined by

$$K_t(x,y) = \sum_{r=1}^{s} \sum_{p=1}^{s} x_t x_{t:+r} x_{t:+p} y_t y_{t:+r} y_{t:+p}. \tag{17}$$

Higher neighborhood kernel functions are defined in the same manner.

4. Texture recognition

Texture recognition is a natural problem which requires a strong capacity for feature selection. An image need not portray any particular object or form to be recognized. We can perceive certain aspects of the overall pattern of gray value changes in an aerial image of a forest or a swamp, in a wooden table top, or in a painted surface or car body. Texture refers to this perceived variation among gray values.

[2] To avoid repeated inclusion of the same term; the summation in (14) is performed from $r = 1$.

Here we consider a few types of texture, each type generated by a certain form of Markov random field. Since our purpose is to check the classification rules built on the base of numerous weak features, we do not use different features based on first and second order statistics.

4.1. Gibbs and Markov random fields

A random field is a joint distribution imposed on a set of random variables representing objects of interest. For example, in two-dimensional images pixel intensities are random variables that impose statistical dependence spatially. Gibbs and Markov random fields may serve as models of visual texture, used to color a matrix of $M = N \times N$ pixels using a set of colors $A = \{0, 1, \ldots, G-1\}$ (see Dubes and Jain, 1989).

A discrete Gibbs random field (GRF) provides a global model for an image by specifying a probability mass function in the following form:

$$P(X = x) = \exp[-U(x)]/Z. \tag{18}$$

The function $U(x)$, where x is an M-place vector of colors, is called an energy function.

The notion of a neighborhood is central to Markov random field (MRF) models. Site j is a neighbor of site i ($i \neq j$) if the probability

$$P(X_i = x_i | X_k = x_k \ \forall k \neq i) \tag{19}$$

depends on x_j, the value of X_j. The usual neighborhood system in image analysis defines the first order neighbors of a pixel as the four pixels sharing a side with the given pixel. Second-order neighbors are the four pixels sharing a corner. Higher order neighbors are defined analogously (see Fig. 1).

A random field, with respect to a neighborhood system, is a discrete Markov one if its probability mass function satisfies the following three properties:

1. Positivity $P(X = x) > 0$ for all $x \in \Omega$.
2. Markov property $P(X_t = x_t | X_{S|t} = x_{S|t}) = P(X_t = x_t | X_{\mathrm{d}t} = x_{\mathrm{d}t})$.
3. Homogeneity $P(X_t = x_t | X_{\mathrm{d}t} = x_{\mathrm{d}t})$ is the same for all sites t.

The notation $S|t$ refers to the set of all M sites, excluding site t itself. The notation dt refers to all sites in the neighborhood of site t, excluding site t itself.

Besag suggested a pairwise interaction model of the form

$$U(x) = \sum_{t=1}^{M} F(x_t) + \sum_{t=1}^{M} \sum_{r=1}^{s} H(x_t, x_{t:+r}), \tag{20}$$

where $H(a,b) = H(b,a)$, $H(a,a) = 0$ and s depends on the size of the neighborhood around each site. Function $F(x)$ is the potential function for single pixel cliques and $H(x,y)$ is the potential function for all cliques of size two.

The Derin–Elliott model can be expressed by using a scalar α and a vector θ:

$$F(x_t) = \alpha x_t, \quad H(x_t, x_{t:+r}) = \theta_r I(x_t, x_{t:+r}), \quad r = 1, 2, \ldots, s, \tag{21}$$

where $I(a,b) = -1$ if $a = b$, and 1 otherwise.

4.2. Random field classification

We generated two classes of random fields using (18), (20) and (21), one class generated according to the parameters α_1, θ_1 and the other according to α_2, θ_2. Each random field was represented by a black and white 32×32 pixel matrix, where the pixel values are ± 1. The TSS contained 100 random field samples from each class, and the test set 200 different random field samples from each of the two classes.

The following results show the percentage of support vectors out of the training sample sets and the error rate of the support vector classifiers on the test sets. We also display a field sample from each class of random fields.

4.2.1. Random field classification test number 1

See the fields samples in Fig. 2, and the test results in Table 2.

We see that for simple cases of very different classes, horizontal and vertical lines, all the kernel functions perform well, and the error rate on the test set is about zero for them all.

4.2.2. Random field classification test number 2

See the fields samples in Fig. 3, and the test results in Table 3.

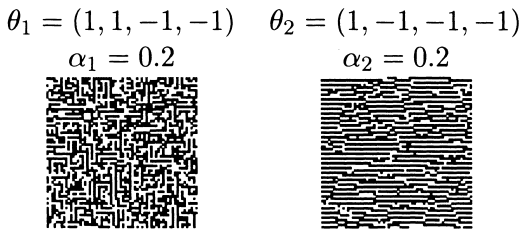

Fig. 2. Samples of random field classification test 1.

Table 2
Random field classification test 1 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d=1$	14%	0%
Polynomial SVM $d=2$	17%	0%
Polynomial SVM $d=3$	20%	0%
Polynomial SVM $d=4$	29%	0%
Polynomial SVM $d=5$	39%	0%
Radial SVM $\gamma=30$	15%	0%
Neural network SVM $w=1$, $c=2$	10%	1%
Neighborhood SVM $d=2$, $s=4$	18%	0%
Neighborhood SVM $d=3$, $s=4$	20%	0%

In this case of diagonal lines which is not much more difficult than the previous case from a human point of view, the regular kernel functions fail to perform any reasonable classification while the neighborhood kernel function demonstrates excellent performance.

4.2.3. Random field classification test number 3

See the fields samples in Fig. 4, and the test results in Table 4.

This is another case where the regular kernel functions cannot classify the random fields, while the neighborhood kernel function performs well.

4.3. Detection of inhomogeneity in random fields

In the following tests the two classes of patterns contain random fields generated according to the same parameters α and θ. Again each random field was represented by a black and white 32×32 pixel matrix, with pixel values ± 1. A central part of the images in class 2 was replaced by an area with

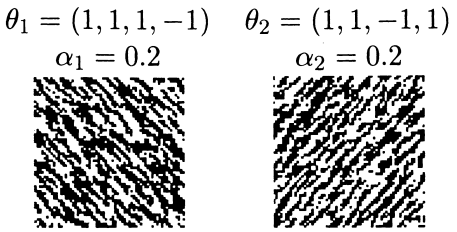

Fig. 3. Samples of random field classification test 2.

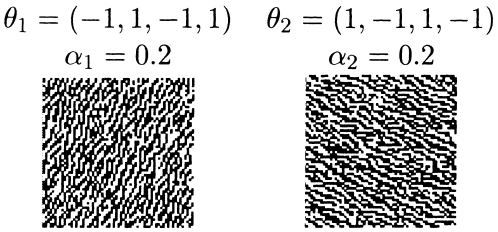

Fig. 4. Samples of random field classification test 3.

Table 3
Random field classification test 2 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d=1$	81%	51%
Polynomial SVM $d=2$	95%	38%
Polynomial SVM $d=3$	99%	35%
Polynomial SVM $d=4$	99%	33%
Polynomial SVM $d=5$	100%	36%
Radial SVM $\gamma=30$	85%	49%
Neural network SVM $w=1$, $c=2$	93%	42%
Neighborhood SVM $d=2$, $s=4$	65%	0%
Neighborhood SVM $d=3$, $s=4$	96%	0%

Table 4
Random field classification test 3 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d=1$	75%	52%
Polynomial SVM $d=2$	91%	42%
Polynomial SVM $d=3$	99%	36%
Polynomial SVM $d=4$	99%	39%
Polynomial SVM $d=5$	100%	41%
Radial SVM $\gamma=30$	82%	53%
Neural network SVM $w=1$, $c=2$	93%	46%
Neighborhood SVM $d=2$, $s=4$	78%	0%
Neighborhood SVM $d=3$, $s=4$	97%	5%

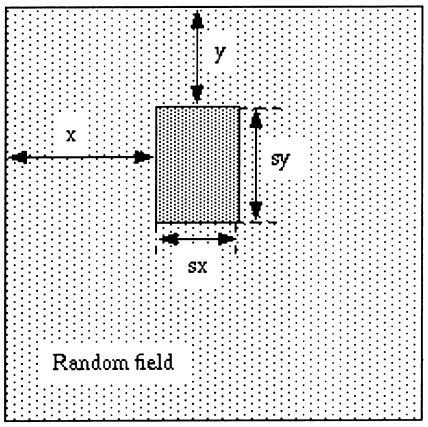

Fig. 5. Inhomogeneity changes for class 2.

random pixel colors. This area had the form of a random box (see Fig. 5).

Its top left corner was placed at coordinates $(x, y) = (12 + \eta_x, 12 + \eta_y)$ of the fields where η_x, η_y were randomly chosen from $\{-3, -2, \ldots, +3\}$. The box width and length were $(sx, sy) = (8 + \delta_x, 8 + \delta_y)$ where δ_x, δ_y were randomly chosen from $\{-2, -1, \ldots, +2\}$. The values of the random variables $\delta_x, \delta_y, \eta_x, \eta_y$ and the pixel colors within the box were randomly chosen for each pattern.

The TSS contained 100 random field samples from each class, and the test set 200 different random field samples from each of the two classes. The percentage of support vectors out of the TSS, the error rate of the SVM on the test sets, and a field sample from each class are given below.

4.3.1. Inhomogeneity detection test number 1

See the fields samples in Fig. 6, and the test results in Table 5.

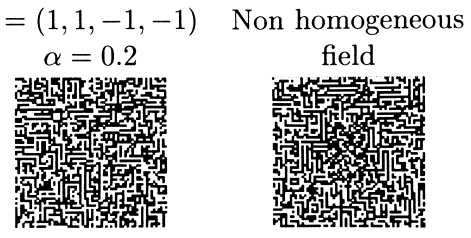

Fig. 6. Samples of inhomogeneity detection test 1.

Table 5
Inhomogeneity detection test 1 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d = 1$	94%	53%
Polynomial SVM $d = 2$	95%	53%
Polynomial SVM $d = 3$	98%	48%
Polynomial SVM $d = 4$	100%	50%
Radial SVM $\gamma = 30$	100%	50%
Neural network SVM $w = 1$, $c = 2$	83%	44%
Neighborhood SVM $d = 2$, $s = 4$	46%	7%

We see that the standard kernel functions fail to perform detection, while the neighborhood kernel function still performs very well.

4.3.2. Inhomogeneity detection test number 2

See the fields samples in Fig. 7, and the test results in Table 6.

The polynomial, radial and neural network kernel functions classify very poorly in comparison to the neighborhood kernel function.

4.3.3. Inhomogeneity detction test number 3

See the fields samples in Fig. 8, and the test results in Table 7.

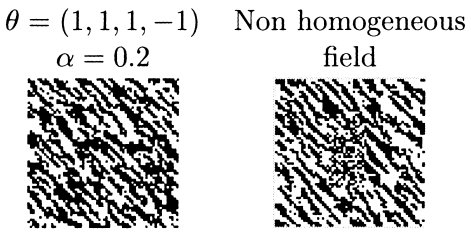

Fig. 7. Samples of inhomogeneity detection test 2.

Table 6
Inhomogeneity detection test 2 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d = 1$	85%	37%
Polynomial SVM $d = 2$	90%	35%
Polynomial SVM $d = 3$	94%	35%
Polynomial SVM $d = 4$	100%	35%
Radial SVM $\gamma = 30$	100%	46%
Neural network SVM $w = 1$, $c = 2$	80%	33%
Neighborhood SVM $d = 2$, $s = 4$	51%	4%

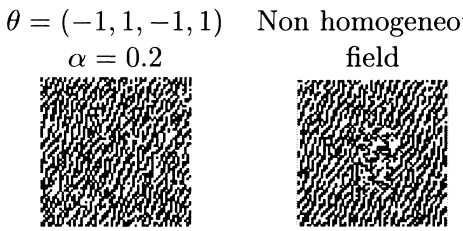

Fig. 8. Samples of inhomogeneity detection test 3.

Table 7
Inhomogeneity detection test 3 results

Kernel function	Support vectors	Test set error rate
Polynomial SVM $d=1$	96%	49%
Polynomial SVM $d=2$	99%	46%
Polynomial SVM $d=3$	99%	42%
Polynomial SVM $d=4$	100%	42%
Radial SVM $\gamma=30$	100%	55%
Neural network SVM $w=1$, $c=2$	92%	42%
Neighborhood SVM $d=2$ $s=4$	50%	6%

Again only the neighborhood kernel function manages to perform inhomogeneity detection.

4.4. Observations

As mentioned at the beginning, an SVM with kernels (5)–(7) produces very good results in many applications. Here we considered some examples of texture recognition problems where for a given size of TSS the regular kernel functions defined by Vapnik, produce poor classification results.

To understand these failures we need to examine the number of parameters which are used to build an SVM. The number of free parameters α_i equals the size of the TSS, meaning that in these tests the SVM had 200 parameters to perform feature selection, in which it had to detect 1024 (32×32 pixels) relevant neighborhoods out of the $O(1024^d)$ total features generated by the kernel (5).

The neighborhood kernel function (15) was defined to feed the SVM with only the significant features, thus avoiding most of the feature selection process. This explains its success in classifying the random fields. Furthermore the neighborhood kernel function produced good classification results with nearly no errors on the test sets, even when the training sample set was as small as 40 vectors.

According to a lemma by Vapnik (1995), the percentage of support vectors of the TSS is an upper bound for the error rate using the SV classifiers. The above results show that this bound may be a poor estimate for the error rate. The neighborhood kernel functions produce many support vectors, but the error rate of the classifiers on the test sets was zero. These results were confirmed using the leave-one-out method. The reason for the difference between the bound and the error rate, is that the same classification rule can be built with all or some of the support vectors. Since the SVM building process usually chooses all possible support vectors, and no procedure of minimizing the number of support vectors is used, their number is not always a good estimate for the error rate.

Texture recognition using an SVM demonstrates the importance of choosing a good kernel function for good classification results. It also shows how one can include domain knowledge about the classes in the SV classifier.

5. Conclusions

We consider the support vector machine to be an excellent universal learning machine which has demonstrated outstanding classification abilities. It is easy to implement (once you solve the quadratic programming problem) and easy to use. Building an SVM does not require guessing parameters like the number of layers in a neural network or the branches of a decision tree. Solution of the quadratic programming problem automatically delivers the global maximum of the problem (1).

At the same time from our experiments we conclude that the feature selection is still needed while using an SVM. In Section 2 we demonstrate the need in feature selection in the input space R_d. In Sections 3 and 4 we demonstrate the need in feature selection in the feature space which may be reduced to finding a specific form of kernel that reflect a domain knowledge for a given type of

problems. In the latter case we demonstrate that choosing an appropriate kernel may lead to a drastic improvement of the result from complete failure to errorless recognition.

The problem of connection between feature selection and SVM has different aspects (see e.g. Bradley and Mangasarian, 1998), but these problems are beyond the scope of this paper.

Another result of our experiments demonstrate that the error rate estimate based on the number of support vectors, is an unduly pessimistic bound, while using the neighborhood kernel for TR problem. In the future we should find out if this phenomenon takes place for other applications of SVM.

Acknowledgements

The second author had some helpful discussions with V. Vapnik (see also (Vapnik, 1982, 1995)) on the nature of the SVM approach, for which we thank him. We also express appreciation to P. Spellucci for letting us use his Donlp2 package for solving the quadratic programming problem defined in (1).

References

Anderson, C., 1958. An Introduction to Multivariate Statistical Analysis, Chapter 3. Wiley, New York.

Bradley, P., Mangasarian O., 1998. Feature selection via concave minimization and support vector machines. In: Machine Learning: Proceeding of the 15th International Conference (ICML'98), Morgan Kaufmann, San Francisco, pp. 82–90.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Machine Learning 20, 273–297.

Dubes, R.C., Jain, A.K., 1989. Random field models in image analysis. Journal of Applied Statistics 16 (2).

Duda, R.O., Hart, P.E., 1973. Pattern Classification and Scene Analysis.

Friedman, J.H., 1989. Regularization discriminant analysis. Journal of American Statistical Association 84, 165–175.

Guyon, I., Boser, B.E., Vapnik, V., 1992. A training algorithm for optimal margin classifiers. In: Proceedings Fifth Annual Workshop of Computational Learning Theory. ACM, New York, pp. 144–152 .

Scholkoph, B., Simard, P., Smola, A., Vapnik, V., 1997. Prior knowledge in support vector kernels, NIPS'97; see also http://svm.first.gmd.de/.

Vapnik, V., 1982. Estimation of Dependencies Based on Empirical Data. Springer, New York.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer, Berlin.