MULTI-SCALE KERNEL METHODS FOR CLASSIFICATION

Nick Kingsbury[*], *David B H Tay*[†], *M Palaniswami*[‡]

University of Cambridge[*], Dept. of Engineering, Cambridge, U.K.
LaTrobe University[†], Dept. of EE, Victoria, Australia
University of Melbourne[‡], Dept. of EEE, Victoria, Australia

ABSTRACT

We propose the enhancement of Support Vector Machines for classification, by the use of multi-scale kernel structures (based on wavelet philosophy) which can be linearly combined in a spatially varying way. This provides a good tradeoff between ability to generalize well in areas of sparse training vectors and ability to fit fine detail of the decision surface in areas where the training vector density is sufficient to provide this information. Our algorithm is a sequential machine learning method in that progressively finer kernel functions are incorporated in successive stages of the learning process. Its key advantage is the ability to find the appropriate kernel scale for every local region of the input space.

1. INTRODUCTION

The Support Vector Machine (SVM) is a machine learning technique that is receiving considerable attention in the scientific community for its superior ability to solve many types of classification problem, particularly in the non-linear form based on the *principle of kernels* [1, 2]. Much of the power of SVM algorithms lies in the implicit non-linear mapping of the input data to a higher-dimensional feature space by the kernel, and choice of the correct kernel for a given problem is very important.

The issue of the choice of kernel has received relatively little attention in the research community until recently. Amari and Wu [3] proposed a simple way of modifying a given kernel function by exploiting the structure of the Riemannian geometry induced by the kernel function. Based on the concept of Reproducing Kernel Hilbert Space from functional analysis Ong et. al. [4] propose the use of hyperkernels as a way to learn the optimal kernel given the data. The hybrid kernel method of Tan and Wang [5] is a way to construct a complex kernel by what is essentially a polynomial expansion of a basic kernel (eg. GRBF). The use of wavelet functions for kernel construction was considered by Zhang et. al. [6] where a basic wavelet, namely the Modulated Gaussian, was used in the simulations. The use of frames has been proposed in [7].

In this paper, we present a way to enhance the SVM through the use of multiple kernels. This approach is a significant departure from previous approaches described above and our philosophy is motivated by ideas and principles from multi-resolution and wavelet theory [8, 9], although a multiscale approach quite similar to ours has been presented briefly in [10]. With wavelets, one first obtains a coarse approximation to a signal using a globally smooth basis function. Successive enhancement to the approximation accuracy is then achieved by adding localized basis functions with increasingly finer scale until a desired level of accuracy is achieved. We show how this approach can be used to improve the ability of SVMs to learn complicated decision functions which contain both fine detail and large smooth regions. Note that this approach is quite different from the work in [6], in which some wavelet principles are employed but only with a single kernel.

This work was supported jointly by the Australian Research Council under the Research Network on Intelligent Sensor, Sensor Networks and Information Processing, and by the Universities of Cambridge and La-Trobe. First author email: ngk@eng.cam.ac.uk .

2. SINGLE KERNEL – KEY RESULTS

We start by summarizing the well-known key formulae arising from the use of conventional kernel-based support vector machines (SVMs) for binary classification. The reader is referred to tutorial papers [1, 11] for more details.

We consider training the SVM to interpret d-dimensional input vectors $\mathbf{x}$ into just two classes $y = \pm 1$. We assume that the N training vectors are of the form $\{\mathbf{x}_1 \ldots \mathbf{x}_N\}$ with associated decision variables $\{y_1 \ldots y_N\}$, where $\mathbf{x}_i \in \mathcal{R}^d$ and $y_i = \pm 1,\ \forall\, i$.

Kernel methods assume a non-linear mapping $\mathbf{\Phi}(\mathbf{x})$, generally to a much higher dimensional space than $\mathbf{x}$, known as the *feature space*, such that, for any pair of input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$, the scalar-valued kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ is defined by the following dot-product in the feature space

$$k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i) = \mathbf{\Phi}(\mathbf{x}_i) \cdot \mathbf{\Phi}(\mathbf{x}_j) \tag{1}$$

When designing a SVM, the aim is to find a function of the form

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{\Phi}(\mathbf{x}) + b \tag{2}$$

such that $y_i f(\mathbf{x}_i) \geq 1$ for all training samples. This corresponds to finding the projection vector $\mathbf{w}$, normal to a pair of parallel hyperplanes in the feature space, such that all points in class $+1$ are 'above' one hyperplane and all points in class -1 are 'below' the other hyperplane. The distance between the hyperplanes is maximized when $\mathbf{w} \cdot \mathbf{w} = ||\mathbf{w}||^2$ is minimized.

In order to find the $\mathbf{w}$ with minimal magnitude which satisfies $y_i f(\mathbf{x}_i) \geq 1$, we define a Lagrangian function

$$L = \tfrac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{N} \lambda_i [y_i\, f(\mathbf{x}_i) - 1] \tag{3}$$

For a given set of Lagrange multipliers $\{\lambda_1 \ldots \lambda_N\}$ which must be non-negative, we differentiate L with respect to each component of $\mathbf{w}$ and also to b and set the results to zero to minimize L, which gives

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{\Phi}(\mathbf{x}_i) \quad \text{and} \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{4}$$

Substituting expressions (1), (2) and (4) into (3) and combining the $\frac{1}{2}\mathbf{w}\cdot\mathbf{w}$ term with the first summation term, produces the well-known dual Lagrangian formulation:

$$L = -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j\, k(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^{N}\lambda_i \qquad (5)$$

Now we solve for the λ_i which maximize this L while satisfying the constraints

$$\sum_{i=1}^{N}\lambda_i y_i = 0 \quad \text{and} \quad \lambda_i \geq 0 \qquad (6)$$

This produces the result which minimizes $\mathbf{w}\cdot\mathbf{w}$ while satisfying $y_i f(\mathbf{x}_i) \geq 1$. The decision function then becomes

$$f(\mathbf{x}) = \sum_{i=1}^{N}\lambda_i y_i\, k(\mathbf{x}_i,\mathbf{x}) + b \qquad (7)$$

The Karush-Kuhn-Tucker (KKT) conditions require that, at the optimum solution, each term of the summation on the right-hand side of (3) is zero, and this is achieved by λ_i only being non-zero for the terms where $y_i f(\mathbf{x}_i) = 1$ exactly. These non-zero λ_i then select the *support vectors* $\mathbf{x}_i$, used to produce $f(\mathbf{x})$ in (7).

If *slack variables*, $\xi_i \geq 0$, are introduced which allow some of the values of $y_i f(\mathbf{x}_i)$ to be $(1-\xi_i)$ instead of unity, in order to reduce the $\mathbf{w}\cdot\mathbf{w}$ term, then two straightforward solutions emerge, depending on the penalty function introduced into the Lagrangian function (3). If the penalty function is the $\mathcal{L}_1$ norm, $C\sum_{i=1}^{N}\xi_i$, then this simply causes an upper limit of C to be introduced to the λ_i, so that the constraint (6) now becomes

$$0 \leq \lambda_i \leq C \qquad (8)$$

The main expressions above can be simplified if vector and matrix notation is used. Hence the dual Lagrangian (5) becomes

$$L = -\tfrac{1}{2}\,\boldsymbol{\lambda}^T Y K Y \boldsymbol{\lambda} \;+\; \boldsymbol{\lambda}^T\mathbf{1} \qquad (9)$$

where $\boldsymbol{\lambda}$ is a column vector of all the λ_i terms, Y is a diagonal matrix of all the y_i terms, K is the kernel matrix with elements $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$, and $\mathbf{1}$ is a column vector of length N with unit elements.

Note that K is a symmetric $N\times N$ matrix, and, for a unique finite solution to be guaranteed, K must be positive definite. (This also implies that YKY is positive definite because Y is diagonal and full-rank.) This is equivalent to requiring that the kernel function satisfies Mercer's condition. If the kernel is shift-invariant (i.e. it is a function only of the vector difference, $\mathbf{x}_i - \mathbf{x}_j$), then the positive definite condition is equivalent to requiring that the Fourier transform of k is purely real and non-negative.

For example, a Gaussian radial basis function (GRBF) kernel is shift invariant, since it is defined by

$$k_{\mathrm{grbf}}(\mathbf{x}_i,\mathbf{x}_j) = \exp\left(\frac{-||\mathbf{x}_i-\mathbf{x}_j||^2}{2\sigma^2}\right) \qquad (10)$$

Its Fourier transform is another radial Gaussian function in d-dimensional frequency space and is therefore non-negative, as required.

3. MULTIPLE KERNEL METHOD

Returning to equation (7), we note that $f(\mathbf{x})$ is just a linear combination of kernel functions, located around the support vectors $\mathbf{x}_i$ for which the weights $\alpha_i = \lambda_i y_i$ are non-zero. For RBF kernels, these functions are radially symmetric and are centred upon each support vector, as shown in fig. 1(a) for a pair of 2-D support vectors (one of class $+1$ and the other of class -1). The polarities of the weights are given by the y_i values, which are the classes of the support vectors. The composite surface $f(\mathbf{x})$ for all 6 support vectors is shown in fig. 1(b) and the class decision surface is the boundary (black contour) corresponding to $f(\mathbf{x}) = 0$.

It is straightforward to see that if a GRBF kernel, such as the one defined by (10), is used and its standard deviation σ is large, then the surface $f(\mathbf{x})$ will be a smooth function of $\mathbf{x}$. Hence the decision surface, where $f(\mathbf{x}) = 0$, will also be smooth. This will be good for generalizing in regions of sparse training data, but it will be poor at fitting any rapidly fluctuating parts of the decision boundary.

On the other hand, a GRBF kernel with small σ will be good at fitting rapid fluctuations (as long as there is enough training data in such regions) but it will be poor at generalising in regions of sparse training data, since $f(\mathbf{x})$ will tend back to zero in between neighboring points of the same class which are significantly more than 2σ apart. Hence it is desirable to allow kernel functions of more than one scale to be used to define a given decision surface. Fig. 2 shows the same support vectors as fig. 1, but now different σ values are used for the kernels. Note how this modifies the decision boundary contour into a much sharper curve around the support vector with the small-σ kernel.

We choose to start by defining the form that $f(\mathbf{x})$ should take for just two kernel functions k_1 and k_2, which is:

$$f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x}) \qquad (11)$$

$$\text{where} \quad f_1(\mathbf{x}) = \sum_{i=1}^{N}\alpha_i\, k_1(\mathbf{x}_i,\mathbf{x}) + b_1 \qquad (12)$$

$$\text{and} \quad f_2(\mathbf{x}) = \sum_{i=1}^{N}\beta_i\, k_2(\mathbf{x}_i,\mathbf{x}) + b_2 \qquad (13)$$

Our aim is that k_1 should be a coarse-scale kernel (e.g. a GRBF with large σ) and the α_i should select support vectors in smooth regions of $f(\mathbf{x})$, while k_2 should be a finer-scale kernel (smaller σ) and the β_i should select support vectors in regions where $f(\mathbf{x})$ needs to vary more rapidly.

We can define separate feature space mappings $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ so that, for any $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\boldsymbol{\Phi}_1(\mathbf{x}_i)\cdot\boldsymbol{\Phi}_1(\mathbf{x}_j) = k_1(\mathbf{x}_i,\mathbf{x}_j) \qquad (14)$$

$$\boldsymbol{\Phi}_2(\mathbf{x}_i)\cdot\boldsymbol{\Phi}_2(\mathbf{x}_j) = k_2(\mathbf{x}_i,\mathbf{x}_j) \qquad (15)$$

$$\text{and} \quad \boldsymbol{\Phi}_1(\mathbf{x}_i)\cdot\boldsymbol{\Phi}_2(\mathbf{x}_j) = 0 \qquad (16)$$

The orthogonal relation between $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ in (16) is useful, and can be easily achieved (notionally) by increasing the dimensionality of the feature space and allowing the two functions to use separate sets of dimensions in that space, such that no dimension component is ever non-zero in both functions. If we let

$$\mathbf{w}_1 = \sum_{i=1}^{N}\alpha_i\,\boldsymbol{\Phi}_1(\mathbf{x}_i) \quad \text{and} \quad \mathbf{w}_2 = \sum_{i=1}^{N}\beta_i\,\boldsymbol{\Phi}_2(\mathbf{x}_i) \qquad (17)$$

then we can rewrite $f(\mathbf{x})$ from (11) in the form of equation (2) as

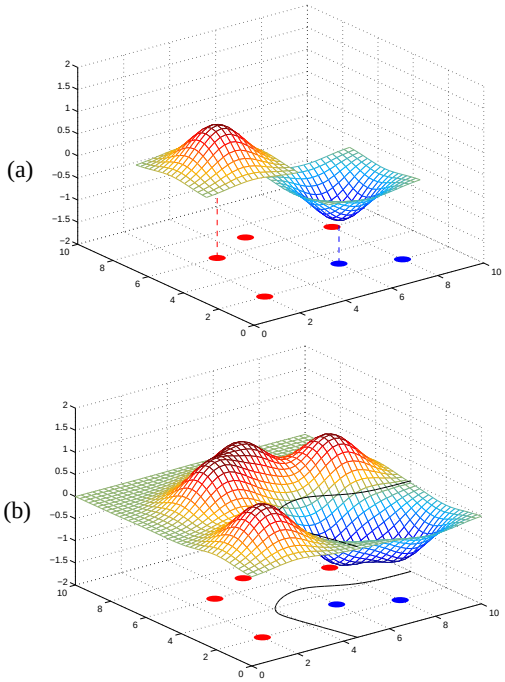


Fig. 1. Example of the formation of a decision surface $f(\mathbf{x})$ with a GRBF kernel of single scale ($\sigma = 1$). Support vector locations are shown by the large dots on the base plane.

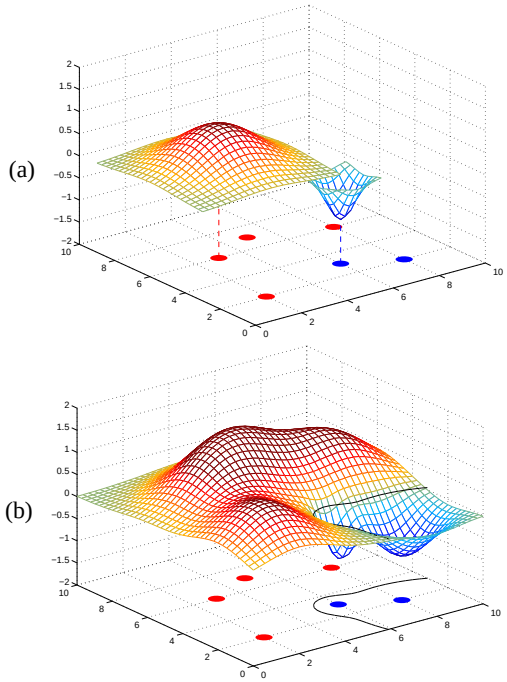


Fig. 2. Example of the formation of a decision surface $f(\mathbf{x})$ with multi-scale GRBF kernels ($\sigma = 0.5, 1, 1.5$).

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}_1 \cdot \mathbf{\Phi}_1(\mathbf{x}) + b_1 + \mathbf{w}_2 \cdot \mathbf{\Phi}_2(\mathbf{x}) + b_2 \\ &= (\mathbf{w}_1 + \mathbf{w}_2) \cdot (\mathbf{\Phi}_1(\mathbf{x}) + \mathbf{\Phi}_2(\mathbf{x})) + b \end{aligned} \qquad (18)$$

where $b = b_1 + b_2$, since the cross terms, $\mathbf{w}_1 \cdot \mathbf{\Phi}_2(\mathbf{x})$ and $\mathbf{w}_2 \cdot \mathbf{\Phi}_1(\mathbf{x})$, are zero because of the orthogonality of $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$.

Given the equivalence between equations (2) and (18) if $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$ and $\mathbf{\Phi}(\mathbf{x}) = \mathbf{\Phi}_1(\mathbf{x}) + \mathbf{\Phi}_2(\mathbf{x})$, it seems natural to try to find the α_i and β_i which solve the same Lagrangian optimization as before in (3). We will call this the parallel method of solution. Solving for zero gradient of the primary Lagrangian function as in (4) requires that

$$\mathbf{w}_1 = \sum_{i=1}^{N} \lambda_i y_i \mathbf{\Phi}_1(\mathbf{x}_i) \;\text{ and }\; \mathbf{w}_2 = \sum_{i=1}^{N} \lambda_i y_i \mathbf{\Phi}_2(\mathbf{x}_i) \qquad (19)$$

Comparing equations (17) and (19), we see it is necessary that $\alpha_i = \lambda_i y_i$ and $\beta_i = \lambda_i y_i$, and hence $\alpha_i = \beta_i$, for all i. This is *not* what we were hoping for as it implies that $f(\mathbf{x})$ is comprised of equal amounts of each kernel, k_1 and k_2, at every support vector location. It conflicts with our aim of using k_1 predominantly for locations in smooth parts of the decision function and k_2 where it is less smooth.

To overcome the problems of undesirable linkage between the α and β terms in the parallel method, we now propose a sequential method of solution. It is based on the observation that, when slack variables are introduced into the SVM model (as in equation (8)), the magnitudes of the α_i terms are constrained so that $f(\mathbf{x})$ has less than unit magnitude at the locations where the slack variables ξ_i are non-zero. If the kernel function is too smooth, then the ξ_i will tend to be non-zero at the locations where this smoothness is a problem.

Our sequential strategy is first to solve for the α_i using a fairly low value for the slack parameter C, in order to create $f_1(\mathbf{x})$ from a single smooth kernel k_1, so that it fits the training data as well as it can in smooth areas but leaves significant errors elsewhere. Then we use a less smooth (finer scale) kernel k_2 to create $f_2(\mathbf{x})$ such that the combined function $f_1(\mathbf{x}) + f_2(\mathbf{x})$ is a better fit to the data than just $f_1(\mathbf{x})$. Slack variables must be used for the first step, to prevent the smooth kernel from 'trying too hard' to fit the data everywhere and thus leaving nothing to be done in the second step. Similarly they can also be used in the second step, and if there are still regions of poor fit, then additional steps may be used with kernel functions of progressively finer scales. We want the distribution of slack variables to be relatively sparse (to encourage the best fit in areas where kernel smoothness is not a problem), so the $\mathcal{L}_1$ error norm is chosen in preference to the $\mathcal{L}_2$ norm as the most appropriate minimization criterion in this application.

The first step of the sequential algorithm is basically identical to the single-kernel case, outlined in section 2. Hence we obtain $f_1(\mathbf{x})$ from (12) where $\boldsymbol{\alpha} = Y\boldsymbol{\lambda}_1$ and, as in (9), $\boldsymbol{\lambda}_1$ is the set of Lagrange multipliers from step 1 which maximize the dual Lagrangian

$$L_1 = -\tfrac{1}{2}\,\boldsymbol{\lambda}_1^T Y K_1 Y \boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_1^T \mathbf{1} \qquad (20)$$

subject to

$$\boldsymbol{\lambda}_1^T Y \mathbf{1} = 0 \;\text{ and }\; 0 \le \lambda_{1,i} \le C_1,\ \forall\, i \qquad (21)$$

The kernel matrix K_1 has elements $k_1(\mathbf{x}_i, \mathbf{x}_j)$.

In the second step of the algorithm, $\boldsymbol{\alpha}$, and hence $\mathbf{w}_1$ and f_1, are fixed, while we solve for $\boldsymbol{\beta}$ and f_2. The primary Lagrange function is derived from (3) using (18), but excludes the $\mathbf{w}_1 \cdot \mathbf{w}_1$ term as it is constant and the $\mathbf{w}_1 \cdot \mathbf{w}_2$ terms are zero. Hence

$$L_2 = \tfrac{1}{2}\,\mathbf{w}_2 \cdot \mathbf{w}_2 - \sum_{i=1}^{N} \lambda_{2,i}[y_i f(\mathbf{x}_i) - 1] \qquad (22)$$

We now substitute for $f(\mathbf{x})$ from (18) and differentiate only with respect to $\mathbf{w}_2$, as $\mathbf{w}_1$ is fixed, to obtain

$$\mathbf{w}_2 = \sum_{i=1}^{N} \lambda_{2,i} y_i \mathbf{\Phi}_2(\mathbf{x}_i) \quad \text{and so} \quad \boldsymbol{\beta} = Y\boldsymbol{\lambda}_2 \tag{23}$$

Hence $\boldsymbol{\lambda}_2$ is chosen to maximize the dual Lagrangian:

$$\begin{aligned} L_2 &= -\tfrac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_{2,i}\lambda_{2,j} y_i y_j \, k_2(\mathbf{x}_i, \mathbf{x}_j) \\ &\qquad + \sum_{i=1}^{N} \lambda_{2,i}(1 - y_i f_1(\mathbf{x}_i)) \\ &= -\tfrac{1}{2}\,\boldsymbol{\lambda}_2^T Y K_2 Y \boldsymbol{\lambda}_2 \;+\; \boldsymbol{\lambda}_2^T[\mathbf{1} - Y\,\mathbf{f}_1(X)] \end{aligned} \tag{24}$$

subject to

$$\boldsymbol{\lambda}_2^T Y \mathbf{1} = 0 \;\text{ and }\; 0 \le \lambda_{2,i} \le C_2, \;\forall\, i \tag{25}$$

where $\mathbf{f}_1(X)$ is a column vector comprising the values of $f_1(\mathbf{x}_i) \;\forall\, i$.

Comparing (24) with (20) shows that the only effect that step 1 of the sequential algorithm has on step 2, is the contribution $-Y\,\mathbf{f}_1(X)$ to the final term. This represents the amount that the $f_1(\mathbf{x}_i)$ terms reduce the desired minimum values for $f_2(\mathbf{x}_i)$, so that $y_i[f_1(\mathbf{x}_i) + f_2(\mathbf{x}_i)] \ge (1 - \xi_i)$ at each location $\mathbf{x}_i$.

As indicated above, if there are still regions of poor fit to the training data (and this is judged *not* to be due to measurement noise), then further iterations may be employed, using progressively finer-scale kernels until an acceptable level of fit is achieved. The only change in later iterations to the above algorithm is the inclusion of all previous functions in the final term of (24). Hence for iteration p the dual Lagrangian becomes

$$L_p = -\tfrac{1}{2}\,\boldsymbol{\lambda}_p^T Y K_p Y \boldsymbol{\lambda}_p \;+\; \boldsymbol{\lambda}_p^T[\mathbf{1} - Y\sum_{r=1}^{p-1}\mathbf{f}_r(X)] \tag{26}$$

and the vectors of support coefficients are

$$\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta} \ldots\} = Y\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \boldsymbol{\lambda}_3, \boldsymbol{\lambda}_4 \ldots\} \tag{27}$$

We now turn to the choice of function type for kernels $k_2 \ldots k_p$ which may well depend on the application area. We can make some progress however by assuming that k_1 is a d-dimensional GRBF of the form of (10). This will have a lowpass frequency response that is also a radially symmetric Gaussian function in d-dimensional frequency space, with a bandwidth that is inversely proportional to the standard deviation, σ_1, of k_1.

In accordance with multi-scale wavelet philosophy, let k_2 have half the standard deviation of k_1, so that $\sigma_2 = \sigma_1/2$ and its radial bandwidth will be twice that of k_1. Wavelets tend to be designed to produce low correlation between the basis functions at adjacent scales. This is achieved by starting with a lowpass scaling function for the lowest frequencies of interest, and then defining wavelets to be bandpass functions with octave bandwidths to cover the higher frequency components of the signal. A key feature of a bandpass wavelet function is that it has zero gain at zero frequency (all admissible wavelets must have this property). A potentially conflicting requirement for SVMs is that all kernels should satisfy Mercer's condition, which for shift-invariant RBFs means that their Fourier transforms must be non-negative. A good way to create bandpass RBFs (BRBFs) in d-dimensional space, which satisfy this condition, is to take the difference of two lowpass GRBFs:

$$\begin{aligned} k_{\text{brbf}}(\mathbf{x}_i, \mathbf{x}_j) &= g_2 \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_2^2}\right) \\ &\qquad - g_1 \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma_1^2}\right) \end{aligned} \tag{28}$$

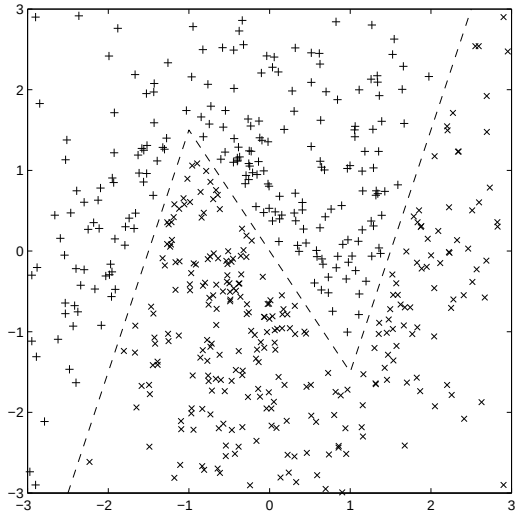


Fig. 3. Training data (500 points) for the 'zig-zag' test example, shown as '+' and '×' for points above and below the decision boundary (dashed line) respectively.

where $g_2 = \sigma_1^d/(\sigma_1^d - \sigma_2^d)$ and $g_1 = \sigma_2^d/(\sigma_1^d - \sigma_2^d)$ in order to obtain zero gain at zero frequency and unit amplitude at the centre of the BRBF. In 2-D this is the familiar 'mexican hat' function.

4. RESULTS

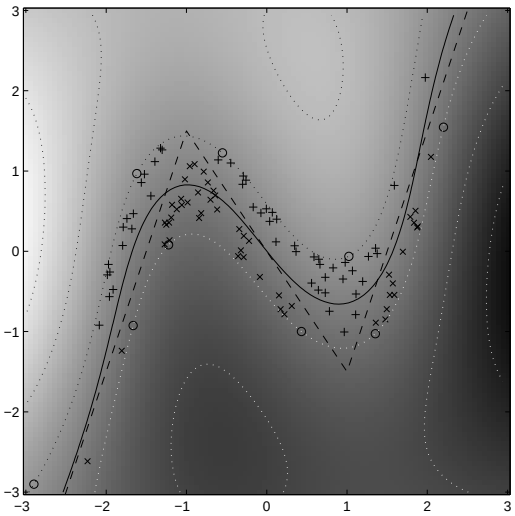
To demonstrate our proposed multi-scale techniques, we took a 2-D example of part of a decision boundary which is a zig-zag shape, as shown by the dashed line in figure 3. This shape contains features at multiple scales and is representative of the sort of boundary that conventional single-scale kernel functions would have difficulty in matching.

Training samples were generated randomly from a 2-D Gaussian circular distribution, of standard deviation $= 1.5$, truncated to lie between ± 3. Samples which were within a small distance ϵ (chosen to be 0.1 in this case) of the decision boundary were removed, as such samples would tend to lead to unreliable decisions in practice. Our set of $N = 500$ training samples are shown in figure 3. For testing we used 10,000 samples, $\mathbf{x}_t$, arranged on a uniform 100×100 grid covering the whole region shown. Our algorithm used a GRBF with $\sigma = 2$ for the first iteration, and then BRBFs with $\{\sigma_1, \sigma_2\} = \{2, 1\}, \{1, 0.5\}$ and $\{0.5, 0.25\}$ for the second, third and fourth iterations respectively. We set $C_p = 5 \,\forall\, p$.

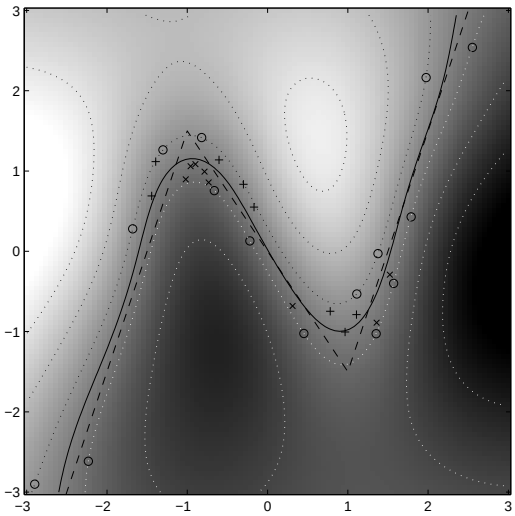
The results after each iteration are shown in figure 4, (a) to (d). The level-zero decision contour is a solid line, and the dashed line shows the 'ideal' decision contour of the model from which the training data was generated and to which we hope the solid line will converge. Note how the wide (large σ) GRBF of the first iteration results in a smooth decision contour, which does not follow the sharp transitions of the zig-zag, but is good at estimating the correct boundary near the outer edges of the region despite the relative low density of training samples in these regions (see figure 3). Beside each figure we quote the hard and soft probabilities of error, measured using the $T = 10,000$ uniformly spaced test samples $\mathbf{x}_t$ and defined (in Matlab-style notation) as follows:

$$P_{\text{hard}} = \tfrac{1}{T}\textstyle\sum_{t=1}^{T}[y_t f(\mathbf{x}_t) < 0] \tag{29}$$

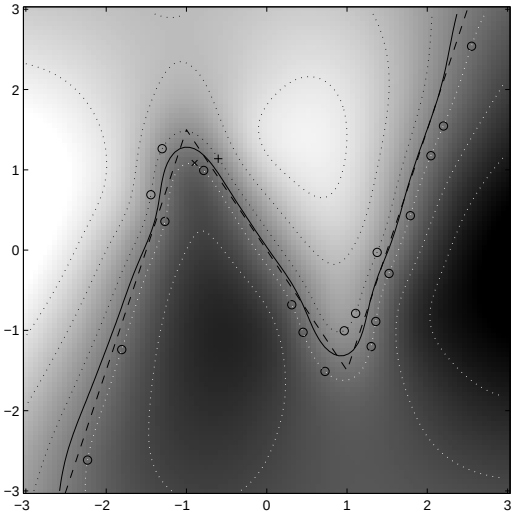
$$P_{\text{soft}} = \tfrac{1}{T}\textstyle\sum_{t=1}^{T}\max[1 - y_t f(\mathbf{x}_t), 0] \tag{30}$$



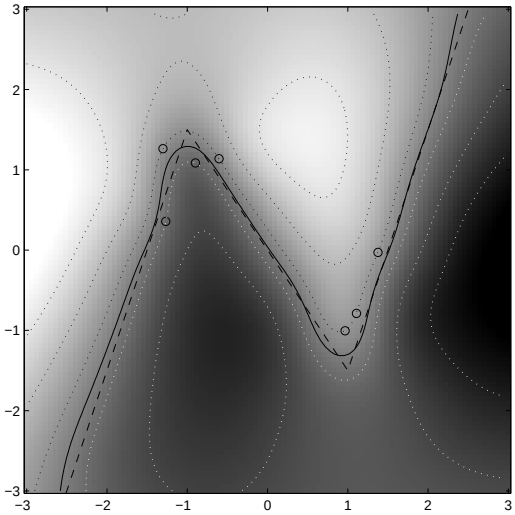
(a)
Iteration 1:
$\sigma = 2$
$P_{\text{soft}} = 0.125$
$P_{\text{hard}} = 0.034$



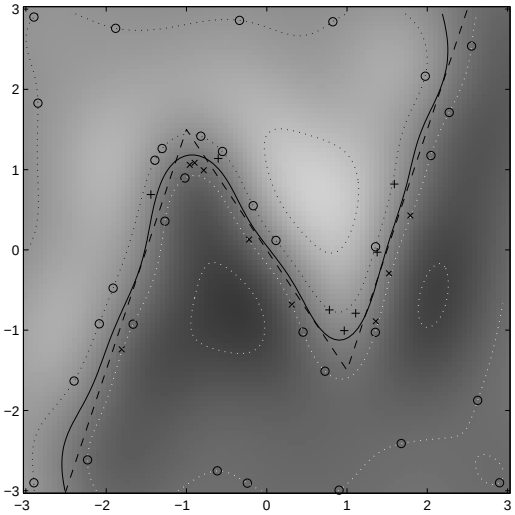
(b)
Iteration 2:
$\sigma_1 = 2, \sigma_2 = 1$
$P_{\text{soft}} = 0.078$
$P_{\text{hard}} = 0.021$



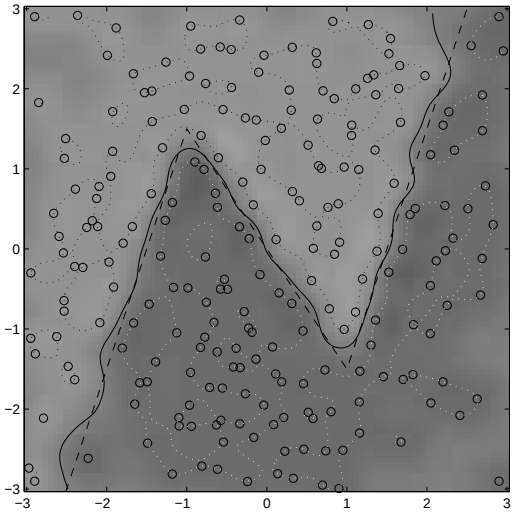
(c)
Iteration 3:
$\sigma_1 = 1, \sigma_2 = 0.5$
$P_{\text{soft}} = 0.065$
$P_{\text{hard}} = 0.013$



(d)
Iteration 4:
$\sigma_1 = 0.5, \sigma_2 = 0.25$
$P_{\text{soft}} = 0.065$
$P_{\text{hard}} = 0.013$



(e)
Single kernel:
$\sigma = 0.7$
$P_{\text{soft}} = 0.089$
$P_{\text{hard}} = 0.018$



(f)
Single kernel:
$\sigma = 0.25$
$P_{\text{soft}} = 0.190$
$P_{\text{hard}} = 0.018$

Fig. 4. Evolution of the decision surface and support vectors for four iterations of the multi-scale algorithm (a to d) and two single-kernel examples (e and f). The gray-scale image is of the surface $f(\mathbf{x})$ with dotted contours at levels $\{\pm 1, \pm 3, \pm 5\}$. Support vectors whose λ_i are below the upper bound, $C = 5$, are shown as circles, while support vectors with λ_i at the upper bound are shown as crosses ('$+$' for class $+1$ and '$\times$' for class -1).

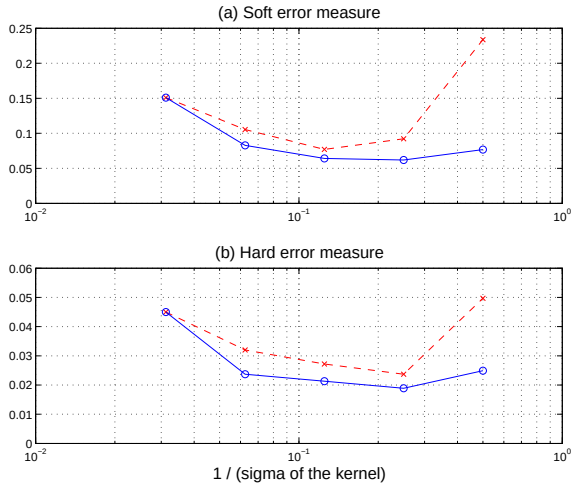


Fig. 5. Results of tests with the Wisconsin breast cancer database: the dashed line with '$\times$' shows the single kernel results and the solid line with 'o' shows the multiple kernel results.

P_{hard} measures the (quantized) area between the solid and dashed lines in figures 4(a) to 4(f) as a proportion of the total area of each figure. P_{soft} measures the mean amount by which $y_t f(\mathbf{x}_t)$ is less than unity, and is a more sensitive measure of lack of fit between $f(\mathbf{x})$ and the desired model.

In figure 4 we see that P_{hard} improves from 0.034 to 0.013, while P_{soft} improves from 0.125 to 0.065 over the four iterations.

To provide a comparison with single-kernel methods, figures 4(a), (e) and (f) show corresponding results for coarse ($\sigma = 2$), medium ($\sigma = 0.7$) and fine scale ($\sigma = 0.25$) GRBF single kernels, trained on identical data. We can see that the proposed sequential multi-scale method achieves a good fit to the model (dashed line) by iteration 3 (figure 4(c)), and that this is significantly better than that achieved by the single-kernel examples, which demonstrate either a poor ability to fit the finer features (corners) of the model (figures 4(a) and (e)) or over-fitting of the data with a poor ability to generalize in between test samples (figure 4(f)). Figure 4(d) shows that the algorithm converges to a sensible final solution that is appropriate to the density of training samples provided.

The above examples are limited to 2-dimensional data. To show that the multi-scale concept works for real data of much higher dimensionality, we have applied it to the well-known Wisconsin breast cancer dataset [12], which is 30-dimensional and comprises 569 samples, of which 357 are classed as benign and 212 as malignant. To test our systems, we randomly chose 400 samples as a training dataset and used the remaining 169 as test data, and repeated this for 5 random selections in total. The only preprocessing performed on the data was to scale each of the 30 components to have unit variance over the 569 samples. Figure 5 shows the results of these tests.

The plots show P_{soft} and P_{hard} respectively as a function of $1/\sigma$ on a log scale; from left to right $\sigma = \{32, 16, 8, 4, 2\}$. For the multiple-kernel system, these are the consecutive values of σ or σ_2 on each iteration, 1 to 5, and the results are shown by the solid line with circles at the data points. The dashed line with crosses plots the results with a single GRBF kernel whose σ takes the 5 values in turn. As before, the multiple-kernel system uses a GRBF on the first iteration and BRBFs on iterations 2 to 5. C_p is again set to 5 for all p. The multiple-kernel system consistently outperforms the single-kernel cases and achieves a hard error probability of 1.9% (i.e. 98.1% probability of correct classification), compared with a best result of 2.4% for single kernels under the same input conditions. The plots of P_{soft} show a similar trend.

Note that the multiple-kernel systems are much less critical of the precise choice and range of values for σ since their curves display a broader minimum than the single-kernel curves.

5. CONCLUSION

We have presented a generalization of the SVM by introducing multiple kernels and multi-scale learning, motivated by wavelet concepts of coarse to fine scale approximations. We have shown how different kernels can learn different parts of a complicated decision boundary comprising both smooth and sharp parts. Computation tends to rise only linearly (or less) with the number of kernel scales and the dimensionality of the input space. This work is an initial step in multi-scale SVM research, with many aspects still to be investigated such as the choice of parameters like C and the range of σ. Alternative kernel types, computational efficiency aspects, and linkage with Bayesian ideas (e.g. Relevance Vector Machines) are also key areas for future work.

6. REFERENCES

[1] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An Introduction to Kernel Based Learning Algorithms," *IEEE Trans. on Neural Networks*, vol. 12, no. 2, pp. 181–202, March 2001.

[2] F. Perez-Cruz and O. Bousquet, "Kernel Methods and Their Potential use in Signal Processing," *IEEE Signal Processing Magazine*, pp. 57–65, May 2004.

[3] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Networks*, vol. 12, pp. 783–789, 1999.

[4] C. S. Ong, A. J. Smola, and R. C. Williamson, "Learning with Hyperkernels," *Journal of Machine Learning Research, submitted*, 2003.

[5] Y. Tan and J. Wang, "A Support Vector Machine with a Hybrid Kernel and Minimal Vapnik-Chervonenkis Dimension," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 4, p. 385, April 2004.

[6] L. Zhang, W. Zhou, and L. Jiao, "Wavelet Support Vector Machine," *IEEE Trans. on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 34, no. 1, pp. 34–39, February 2004.

[7] J. B. Gao, C. J. Harris, and S. R. Gunn, "Support Vector Kernels based on Frames in Function Hilbert Spaces," *Neural Computation*, vol. 13, pp. 1975–1994, 2001.

[8] I. Daubechies, *Ten Lectures on Wavelets*. Society for Industrial and Applied Maths., 1992.

[9] M. Vetterli and J. Kovacevic, *Wavelets and Subband Coding*. Prentice-Hall, 1995.

[10] F. Steinke, B. Schlkopf, and V. Blanz, "Support Vector Machines for 3D Shape Processing," *Eurographics*, vol. 24, no. 3, 2005.

[11] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Knowledge Discovery and Data Mining*, vol. 2, pp. 121–167, 1998.

[12] O. Mangasarian, W. Street, and W. Wolberg, "Breast cancer diagnosis and prognosis via linear programming," *Operations Research*, vol. 43, no. 4, pp. 570–577, July 1995.