Proc. of the 2nd Joint Workshop on Application of AI in Civil Engineering
March 2000, Cottbus, Germany

Support Vector Machines: Review and Applications in Civil Engineering

Yonas B. Dibike, Slavco Velickov and Dimitri Solomatine
International Institute for Infrastructural, Hydraulic, and Environmental Engineering, P.O. Box 3015, 2601 DA Delft, The Netherlands, email-ybd@ihe.nl

Abstract: The subject of Support Vector Machines (SVM) covers emerging techniques which have proven successful in many traditionally neural network (NN)-dominated applications. An interesting property of this approach is that it is an approximate implementation of the Structural Risk Minimisation (SRM) induction principle that aims at minimising a bound on the generalisation error of a model, rather than minimising the mean square error over the data set. In this paper, the basic ideas underlying SVM are reviewed and the potential of this method for feature classification and multiple regression (modelling) problems is demonstrated using digital remote sensing data and data on the horizontal force exerted by dynamic waves on a vertical structure, respectively. The relative performance of the SVM is then analysed by comparing its results with the corresponding results obtained in previous studies where NNs have been applied on the same data sets.

1 Introduction

The problem of empirical data modelling is pertinent to many engineering applications. In empirical data modelling a process of induction is used to build up a model of the system, from which it is hoped to deduce responses of the system that have yet to be observed. Ultimately the quantity and quality of the observations govern the performance of this model. In most cases, data is finite and sampled and, typically, sampling is non-uniform and, due to the high-dimensional nature of the problem, the data forms only a sparse distribution in the input space.
The foundation of the subject of Support Vector Machines (SVM) has been developed principally by Vapnik (Vapnik [13] & [15]) and the corresponding SV devices are gaining popularity due to their many attractive features and promising empirical performance. They can generally be thought as constituting an alternative training technique for Multi-Layer Perceptron and Radial Basis Function classifiers, in which the weights of the network are found by solving a Quadratic Programming (QP) problem with linear inequality and equality constraints, rather than by solving a non-convex, unconstrained minimisation problem, as in standard neural network training techniques (Osuna, *et al*,[7]) Their formulation embodies the Structural Risk Minimisation (SRM) principle, which has been shown to be superior to the more traditional Empirical Risk Minimisation (ERM) principle employed by many of the other modelling techniques (Osuna, *et al*,[7], Gunn[3]). SRM minimises an upper bound on the expected risk, as opposed to the ERM, which minimises the error on the training data. It is this difference that provides SVM with a greater ability to generalise, which is the goal in statistical learning. SVMs were first developed to solve the classification problem, but recently they have been extended to the domain of regression problems and it has been shown that this approach can find more general solutions than the maximum likelihood-based least square method (Vapnik[15]).

2 Statistical Learning Theory

This section reviews a few concepts and results from the theory of statistical learning which has been developed by Vapnik and others (Vapnik [13] & [15]) and are necessary to appreciate the Support Vector (SV) learning algorithm.
For the case of two-class pattern recognition, the task of learning from examples can be formulated in the following way: we are given a set of functions

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

(where Λ is a set of parameters) and a set of examples, i.e. pairs of patterns $\mathbf{x}_i$ and labels y_i,

$$(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_l, y_l), \in \mathbf{R}^N \times \{-1, +1\}, \tag{2}$$

each one of them generated from an unknown probability distribution P($\mathbf{x}$, y) containing the underlying dependency. What is required is now to learn a function f_α which provides the smallest possible value for the average error committed on independent examples randomly drawn from the same distribution P, called the *risk*

$$R(\alpha) = \int \frac{1}{2} |f_\alpha(\mathbf{x}) - y| dP(\mathbf{x}, y) \tag{3}$$

The problem is that R(α) is unknown, since P($\mathbf{x}$, y) is unknown. Therefore an induction principle for risk minimisation is necessary.
The straightforward approach, which is to minimise the *empirical risk*

$$R_{emp}(\alpha) = \frac{1}{l} \sum_{i=1}^{l} \frac{1}{2} |f_\alpha(\mathbf{x}_\mathbf{i}) - y_i| \tag{4}$$

turns out not to guarantee a small actual risk if the number l of training examples is limited. In other words, a smaller error on the training set does not necessarily imply a higher *generalisation* ability (i.e. a smaller error on an independent *test set*). To make the most out of a limited amount of data, a novel statistical technique called *Structural Risk Minimisation* has been developed (Vapnik [15]). The theory of uniform convergence in probability, developed by Vapnik and Chervonenkis (VC), provides bounds on the deviation of the empirical risk from the expected risk. For $\alpha \in \Lambda$ and $l > h$, a typical uniform VC bound, which holds with probability 1-η, has the following form:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log\frac{2l}{h} + 1) - \log(\frac{\eta}{4})}{l}} \tag{5}$$

The parameter h is called the *VC (Vapnik-Chervonenkis)-dimension* of a set of functions and it describes the capacity of a set of functions to represent the data set. For binary classification, h is the maximal number of points, which can be separated in to two classes in all possible 2^h ways by using functions of the learning machines. The VC-dimension is a measure of the classifier's complexity and it is often proportional to the number of free parameters in the classifier f_α. The SVM algorithm achieves the twin goals of minimising a bound on the VC-dimension and the number of training errors at the same time.

2.1 Support Vector Classification: The Optimal Separating Hyperplane

The classification problem can be restricted to consideration of the two-class problem without loss of generality (Gunn, [3]). Consider the problem of separating the set of training vectors belonging to two separate classes as in eqn 2. In this problem the goal is to separate the two classes with a hyperplane

$$(\mathbf{w}\,.\,\mathbf{x}) + b = 0\,, \tag{6}$$

where **w** is a vector in R^n and b is a constant, which is induced from the available examples. The goal is to produce a classifier that will work well on *unseen* examples, i.e. that generalises well. Consider the example for a two-dimensional input space as shown in Figure 1; here there are many possible linear classifiers that can separate the data, but there is only one that maximises the 'margin' (i.e. maximise the distance between it and the nearest data point of each class). This linear classifier is termed the *optimal separating hyperplane*. Intuitively, this boundary is expected to generalise well as opposed to the other possible boundaries.

The hyperplane $(\mathbf{w}\,.\,\mathbf{x}) + b = 0$ satisfies the conditions:

$$(\mathbf{w}\,.\,\mathbf{x}_i) + b > 0 \text{ if } y_i = 1 \quad \text{and}$$
$$(\mathbf{w}\,.\,\mathbf{x}_i) + b < 0 \text{ if } y_i = -1 \tag{7}$$

By scaling **w** and b with the same factor, an equivalent decision surface can be formulated as:

$$(\mathbf{w}\,.\,\mathbf{x}_i) + b > 1 \text{ if } y_i = 1 \quad \text{and}$$
$$(\mathbf{w}\,.\,\mathbf{x}_i) + b < 1 \text{ if } y_i = -1 \tag{8}$$

which is equivalent to

$$y_i\,[(\mathbf{w}\,.\,\mathbf{x}_i) + b\,] \geq 1,\ \ i = 1, \ldots, l \tag{9}$$

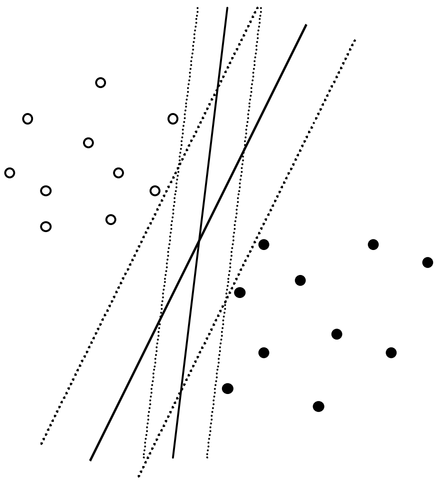


Figure 1 Optimal separating Hyper-plane (two-dimensional case)

The distance d(**w**,b;**x**) of a point **x** from the hyperplane (**w**,b) is,

$$d(\mathbf{w},b;\mathbf{x}) = |\,\mathbf{w}\,.\,\mathbf{x} + b\,|\,/\,\|\mathbf{w}\| \tag{10}$$

The optimal hyperplane is then given by maximising the margin, ρ(**w**, b), subject to the constraints of eqn (9). This margin is calculated as:

$$\rho(\mathbf{w},b) = \min_{\{\mathbf{x}_i:y_i=1\}} d(\mathbf{w},b;\mathbf{x}_i) + \min_{\{\mathbf{x}_j:y_j=-1\}} d(\mathbf{w},b;x_j) \tag{11}$$

Substituting eqn (10) in eqn (11) and further simplifying, the margin is given by:

$$\rho(\mathbf{w}, b) = 2\,/\,\|\mathbf{w}\| \tag{12}$$

Hence the hyperplane that optimally separates the data into two classes is the one that minimises

$$\Phi(\mathbf{w}) = \|\mathbf{w}\|^2/2 \tag{13}$$

The optimisation problem of eqn (13) under the constraints of eqn (9) can be reformulated into an equivalent non-constrained optimisation problem using Lagrangian multipliers and its solution is given by the saddle point of the Lagrange functional (Minoux [6]), as follows:

$$\mathbf{L}(\mathbf{w},b,\alpha) = \|\mathbf{w}\|^2/2 - \sum_{i=1}^{l}\alpha_i\{[(\mathbf{w}\cdot\mathbf{x}_i)+b]y_i - 1\} \tag{14}$$

where the α_i are the Lagrange multipliers. The Lagrangian has to be minimised with respect to $\mathbf{w}$ and b and maximised with respect to $\alpha_i \geq 0$. In the saddle point the solutions $\mathbf{w}_0$, b_0 and α^0 should satisfy the following conditions:

$$\partial\mathbf{L}(\mathbf{w}_0, b_0, \alpha^0) / \partial b = 0 \qquad \Rightarrow \qquad \sum \alpha^0_i\, y_i = 0 \tag{15}$$

$$\partial\mathbf{L}(\mathbf{w}_0, b_0, \alpha^0) / \partial\mathbf{w} = 0 \qquad \Rightarrow \qquad \mathbf{w}_0 = \sum \alpha^0_i\, \mathbf{x}_i\, y_i \tag{16}$$

Only the so-called *support vectors* can have non-zero coefficients α^0_i in the expansion of $\mathbf{w}_0$
The Kuhn-Tucker [4] theorem states that the necessary and sufficient conditions for the optimal hyperplane are that the separating hyperplane satisfies the conditions:

$$\alpha^0_i\,\{[(\mathbf{w}_0 \cdot \mathbf{x}_i) + b_0\,]\,y_i - 1\} = 0, \; i = 1, \ldots, l \tag{17}$$

as well as constituting a saddle point. Substituting these results into the Lagrangian, and after further simplifications, the following dual form of the function to be optimised is obtained:

$$\mathrm{W}(\alpha) = \sum_{i=1}^{l}\alpha_i - (1/2)\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i\cdot\mathbf{x}_j) \tag{18}$$

This function has to be maximised with constraints $\alpha_i \geq 0$ and $\sum \alpha_i\, y_i = 0$. Finding solutions for real-world problems will usually require application of quadratic programming optimisation techniques and numerical methods. Once the solution has been found in the form of a vector $\alpha^0 = (\alpha^0_1, \alpha^0_2, \ldots, \alpha^0_l)$,the optimal separating hyperplane is given by

$$\mathbf{w}_0 = \sum \alpha_i^0\, \mathbf{x}_i\, y_i \qquad \text{and} \qquad b_0 = -(1/2)\, \mathbf{w}_0 \,.\, [\mathbf{x}_r + \mathbf{x}_s]$$

where $\mathbf{x}_r$ and $\mathbf{x}_s$ are any support vectors from each class. The classifier can then be constructed as:

$$f(\mathbf{x}) = \mathrm{sign}(\mathbf{w}_0 \,.\, \mathbf{x} + b_0) = \mathrm{sign}(\sum y_i\alpha^0\,(\mathbf{x}_i \,.\, \mathbf{x}) + b_0) \tag{19}$$

From the Kuhn-Tucker[4] condition, only the points $\mathbf{x}_i$ which satisfy $\{y_i\,(\mathbf{w}_0 \,.\, \mathbf{x} + b_0) = 1\}$, will have non-zero Lagrangian multipliers. These points are termed *Support Vectors* (SV). If the data is linearly separable, all the SV will lie on the margin and hence the number of SV can be very small. Subsequently the hyperplane is determined by a small subset of the training data; the other points could be removed from the training set and recalculating the hyperplane would produce the same answer. Hence SVMs can be used to summarise the information contained in a data set by the SV produced in this way.
The above solution only holds for separable data, and still has to be slightly modified for non-separable data by introducing a new set of variables $\{\xi_i\}$ that measure the amount by which the constraints are violated. Then the margin is maximised, paying a penalty proportional to the amount of constraint violation. Formally one solves the following problem:

$$\text{Minimise} \qquad \Phi(\mathbf{w}) = \|\mathbf{w}\|^2/2 + C\,(\,\sum\xi_i\,) \tag{20}$$

subject to $y_i\,[(\mathbf{w}\,.\,\mathbf{x}_i) + b\,] \geq 1\text{-}\xi_i$, and $\xi_i \geq 0 \qquad i = 1, \ldots, l$ (21)

where C is a parameter chosen a priori and defining the cost of constraint violation. The first term in eqn (20) provides a minimisation of the VC-dimension of the learning machine, thereby minimising the second term in the bound of eqn (5). On the other hand, minimising the second term in eqn (20) controls the empirical risk, which is the first term in the bound of eqn (5). This approach, therefore, constitutes a practical implementation of Structural Risk Minimisation on the given set of functions. In order to solve this problem, the Lagrangian is constructed as follows:

$$\mathbf{L}(\mathbf{w}, b, \alpha) = \|\mathbf{w}\|^2/2 + C\left(\sum_{i=1}^{l}\xi_i\right) - \sum_{i=1}^{l}\alpha_i\{[(\mathbf{w}\cdot\mathbf{x}_i) + b]y_i - 1 + \xi_i\} - \sum_{i=1}^{l}\gamma_i\xi_i \qquad (22)$$

where α_i and γ_i are associated with the constraints in eqn (21) and the values of α_i have to be bounded as $0 \leq \alpha_i \leq C$. The solution of this problem is determined by the saddle point of this Lagrangian in a way similar to that described in eqns (14)-(18).
In the case where a linear boundary is inappropriate (or when the decision surface is non linear), the SVM can map the input vector **x**, into a high dimensional feature space **z**, by choosing a non-linear mapping a priori. Then the SVM constructs an optimal separating hyperplane in this higher dimensional space as shown in Figure 2. Polynomial, radial basis functions and certain sigmoid functions are among others providing acceptable mappings (see the Appendix). In this case, the optimisation problems of eqn (18) becomes:

$$W(\alpha) = \sum_{i=1}^{l}\alpha_i - (1/2)\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \mathbf{K}(\mathbf{x}_i \cdot \mathbf{x}_j) \qquad (23)$$

where $\mathbf{K}(x,y)$ is the kernel function performing the non linear mapping into the feature space, and the constraints are unchanged. Solving the above equation determines the Lagrange multipliers, and a classifier implementing the optimal separating hyperplane in the feature space is given by,

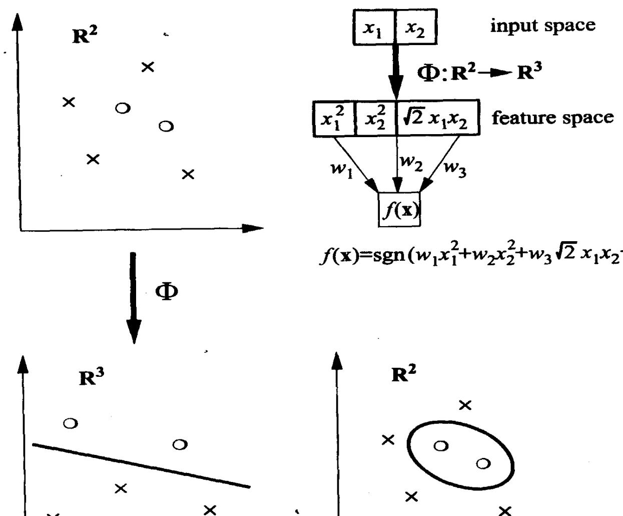
$$f(\mathbf{x}) = \text{sign}(\textstyle\sum y_i\alpha^0\, \mathbf{K}(\mathbf{x}_i\,.\,\mathbf{x}) + b_0) \qquad (24)$$

The other case arises when the data lies in multiple classes. In order to obtain *k-class* classification, a set of binary classifiers $f^1, \ldots, f^k$ are constructed, each trained to separate one class from the rest, and combining them by doing the multi-class classification (applying a voting scheme) according to the maximal output before applying the sign function (Scholkopf [10]), i.e taking $\text{argmax}_{j\,=\,1,\,\ldots,\,k}\; g^j(\mathbf{x})$, where

$$g^j(\mathbf{x}) = \textstyle\sum y_i\alpha^j\, \mathbf{K}(\mathbf{x}_i\,.\,\mathbf{x}) + b_j$$

and

$$f^j(\mathbf{x}) = \text{sign}(g^j(\mathbf{x})) \qquad (25)$$



R²
R³
R²
Φ
input space
Φ:R²→R³
feature space
x_1 x_2
x_1^2 x_2^2 $\sqrt{2}\,x_1x_2$
w_1 w_2 w_3
f(x)
$f(\mathbf{x})=\text{sgn}(w_1x_1^2+w_2x_2^2+w_3\sqrt{2}\,x_1x_2+b)$

Figure 2 By mapping the input data nonlinearly into a higher dimensional feature space and constructing a separating hyperplane there, a SV corresponds to a non-linear decision surface in input space (adopted from Scholkopf [10]).

2.2 Support Vector Regression

SVMs can also be applied to regression problems by the introduction of an alternative loss function that is modified to include a distance measure (Smola[11]). Using SVM can overcome certain problems (in case of non-Gaussian noise) of Fisher's maximum likelihood principle leading to the least squares approach to regression traditionally used in statistics (Vapnik[15]).

Considering the problem of approximating the set of data, $\{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_l\ y_l), \quad \mathbf{x} \in \mathbf{R}^N, y \in \mathbf{R}\}$ with a linear function, $f(\mathbf{x}, \alpha) = (\mathbf{w} \cdot \mathbf{x}) + b$, the optimal regression function is given by minimising the empirical risk:

$$R_{emp}(\mathbf{w}, b) = 1/l \sum |y_i - f(\mathbf{x}_i, \alpha)|_\varepsilon \tag{26}$$

With the most general loss function with ε-insensitive zone described as

$$|y - f(\mathbf{x}, \alpha)|_\varepsilon = \{\varepsilon \text{ if } |y - f(\mathbf{x}, \alpha)| \le \varepsilon;\ |y - f(\mathbf{x}, \alpha)| \text{ otherwise }\} \tag{27}$$

the goal is to find a function $f(\mathbf{x}, \alpha)$ that has at most a deviation of ε from the actual observed targets y_i for all the training data, and at the same time is as flat as possible. This is equivalent to minimising the functional,

$$\Phi(\mathbf{w}, \xi^*, \xi) = \|\mathbf{w}\|/2 + C(\sum\xi_i^* + \sum\xi_i) \tag{28}$$

where C is a pre-specified value and ξ^*, ξ (see Figure 3) are slack variables representing upper and lower constraints on the outputs of the system, as follows:

$$\begin{array}{lll} y_i - ((\mathbf{w} \cdot \mathbf{x}_i) + b) \le \varepsilon + \xi_i & & i = 1,2, \ldots, l \\ ((\mathbf{w} \cdot \mathbf{x}_i) + b) - y_i \le \varepsilon + \xi_i^* & & i = 1,2, \ldots, l \\ \xi_i^* \ge 0 \quad \text{and} & \xi_i \ge 0 & i = 1,2, \ldots, l \end{array} \tag{29}$$

Now the Lagrange function is constructed from both the objective function and the corresponding constraints by introducing a dual set of variables as follows:

$$\begin{aligned} \mathbf{L} = {} & \|\mathbf{w}\|^2/2 + C\sum(\xi_i + \xi_i^*) \\ & - \sum \alpha_i[\varepsilon + \xi_i - y_i + (\mathbf{w} \cdot \mathbf{x}_i) + b] \\ & - \sum (\alpha_i^*[\varepsilon + \xi_i^* + y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b] \\ & - \sum(\eta_i \xi_i + \eta_i^* \xi_i^*) \end{aligned} \tag{30}$$

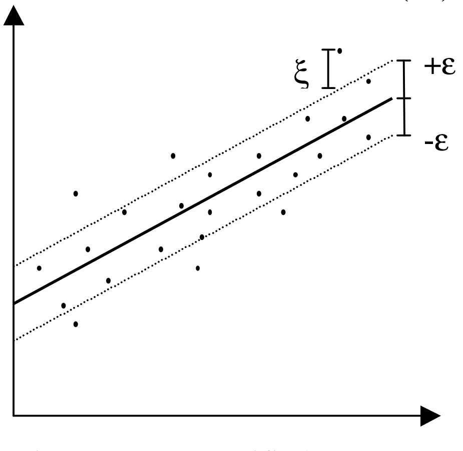


Figure 3 Pre-specified accuracy ε and a slack variable ξ in SV regression.

It follows from the saddle point condition that the partial derivatives of **L** with respect to the primary variables ($\mathbf{w}, b, \xi_i, \xi_i^*$) have to vanish for optimality. Substituting the results of this derivation in to eqn (30) yields the dual optimisation problem:

$$W(\alpha^*, \alpha) = -\varepsilon\sum(\alpha_i^* + \alpha_i) + \sum y_i(\alpha_i^* - \alpha_i) - (1/2)\sum\sum(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(\mathbf{x}_i \cdot \mathbf{x}_j) \tag{31}$$

that has to be maximised subject to the constraints:

$\sum \alpha_i^* = \sum \alpha_i$; $\quad 0 \le \alpha_i^* \le C \quad$ and $\quad 0 \le \alpha_i \le C \qquad$ for $i = 1,2, \ldots, l$

Once the coefficients α_i^* and α_i are determined from eqn (31), the desired vectors can now be found as:

$$\mathbf{w}_0 = \sum (\alpha_i^* - \alpha_i)\, \mathbf{x}_i \quad \text{and therefore} \quad f(\mathbf{x}) = \sum (\alpha_i^* - \alpha_i)\, (\mathbf{x}_i \,.\, \mathbf{x}) + b_0 \tag{32}$$

where $b_0 = -(1/2)\, \mathbf{w}_0 \,.\, [\mathbf{x}_r + \mathbf{x}_s]$

Similarly to the non-linear support vector classification approach, a non-linear mapping kernel can be used to map the data into a higher-dimensional feature space where linear regression is performed. The quadratic form to be maximised can then be rewritten as:

$$W(\alpha^*, \alpha) = -\varepsilon \sum (\alpha_i^* + \alpha_i) + \sum y_i (\alpha_i^* - \alpha_i) - (1/2) \sum \sum (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) \tag{33}$$

and the regression function is given by

$$f(\mathbf{x}) = \mathbf{w}_0 \,.\, \mathbf{x} + b_0 \tag{34}$$

where $\mathbf{w}_0 .\, \mathbf{x} = \sum (\alpha_i^0 - \alpha_i^{0*})\, \mathbf{K}(\mathbf{x}_i, \mathbf{x})$ and $b_0 = -(1/2) \sum (\alpha_i^0 - \alpha_i^{0*})[\mathrm{K}(\mathbf{x}_r, \mathbf{x}_i) + \mathbf{K}(\mathbf{x}_s, \mathbf{x}_i)]])$

3 Applications of SVMs

3.1 Features Classification of Digital Remote Sensing Data.

Remote sensing images (aerial and satellite) and information extracted from such images using one of the image analysis tools has become, nowadays, the major data source for GISs. Classification is one of the important processes of image analysis, being used to categorise automatically all the pixels in the image into such land cover classes as woods, water, roads, etc based on pixel spectral and spatial information. This information can be used further, for example, for coupling GIS maps with hydrological modelling results.
For the present experiment, the results of a previous study (Velickov *et al.*, [16]) on a classification of remote sensing data are used. In that study, digital airborne photographs (47 in total) of the Athens region were considered. In each source image, there are 4387*2786 pixels and they are stored in grey-scale TIFF format. The classification experiment needed two modules, namely: (1) Feature Extractor and (2) Self-Organising Feature Map (SOFM) classifier. Extraction of feature from the pattern space is important for classification or pattern recognition. By doing this, the amount of data is reduced (*pattern space* is transformed in to *feature space*) while the features can still keep the information in the input pattern (see Figure 4).

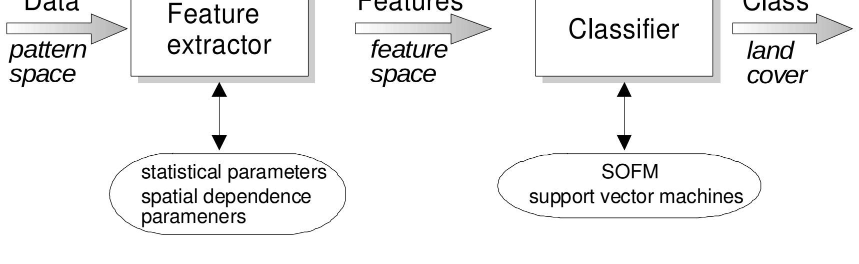


Figure 4. Design of the image recognition system

Statistical parameters, such as *mean* and *variance* and spatial relation parameters (texture) such as *Energy*, *Momentum, Entropy*, *Correlation, Cluster Shade* and *Cluster Prominence* (which are based on the co-occurrence of its grey level with values between 0 and 255) are important when carrying out an image analysis. Since each image of the above size is too big to process on a PC, all images were cut into several pieces of the size 1024*1024. From these sub-images, small windows (segments) with the size of 16*16 pixels and assumed to be characterised by homogeneous texture and statistical parameters, were chosen randomly. For each chosen window, basic statistical and spatial relation parameters were calculated (see the Appendix) using the feature extractor. This gives 1024 sample patterns each containing 8 features. Finally, the SOFM is used to classify these patterns and then the four discovered classes (namely Agriculture, Meadow, Woods and Scrub and Urban area) were labelled based on the land cover specification provided by the Soil Conservation Service (SCS [12]) and statistical and texture parameters extracted from the well-known image segments.
The main objective of this case study is then to demonstrate the potential of SVM for classification (multi-class in this case) of non-linear systems. In order to realise this, the 1024 patterns with extracted features (with 8 dimensions) and their corresponding classes obtained from the SOFM were used to train SVM. The SVM software developed by the Royal Holloway University of London and the AT&T Research Lab (Saunders *et al* [9]) was used in this study. Different types of kernel functions (presented in the Appendix) were used during the experiment and Table 1 summarises results obtained at the training stage.

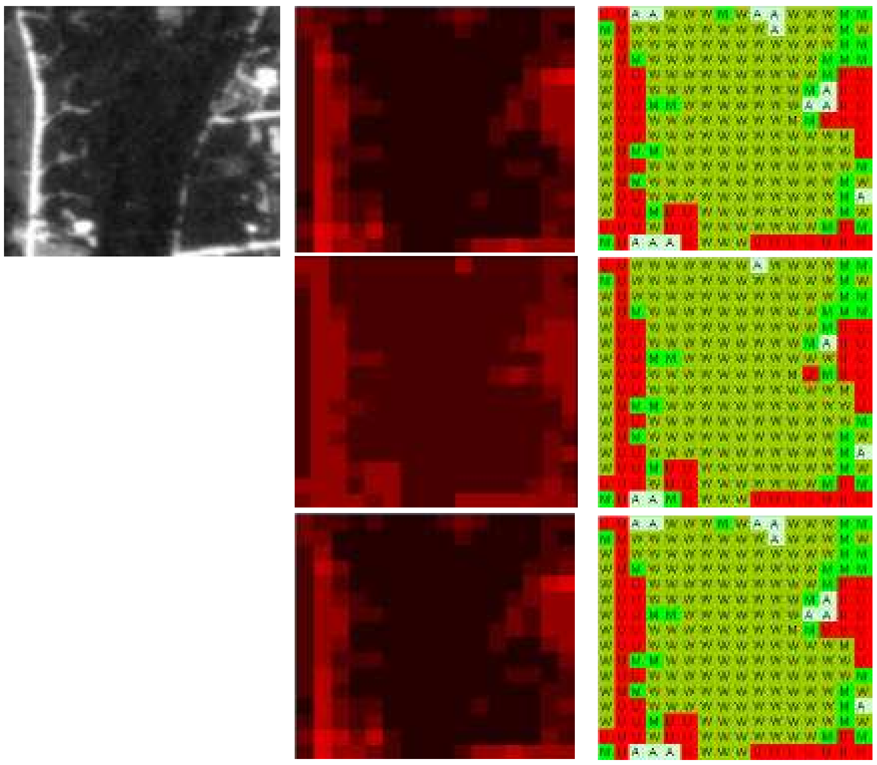
Table 1. Training results of the SVMs

Kernel function	Absolute error [%]	Total number of support vectors positive SV	negative SV
simple dot product	24.3	9	8
simple polynomial kernel	0.195	31	22
real polynomial kernel	0.097	32	23
radial basis function	0.097	106	133
full polynomial kernel	0.097	29	24

Results obtained by training the SVM using a simple dot product (which performs well only on linear classification problems) suggested a high non-linearity of the feature space. Best results during the training stage were obtained using radial basis and full polynomial kernels. However the large number of support vectors found using the radial basis function kernel, which define the shape of the optimal hyperplane has resulted in a longer computational time needed to train the SVMs.
Validation of the SVMs was performed using a test remote sensing image with a size of 256*256 pixels already labelled with land cover classes (Figure 5a). As the computational uniform unit chosen in this study is a window of 16*16 pixels size, the average gray level of each window for the original image is calculated, as shown in Figure 5b. SOFM and SVMs classifiers were used to predict the land cover classes for the test image. The performances for both classifiers are summarised in Table 2 and represented in Figure 5.

Table 2. Validation results for SOFT and SVMs classifiers

	SOFM	Support Vector Machine				
		simple dot product	*simple polynomial kernel*	*real polynomial kernel*	*radial basis function*	*full polynomial kernel*
Absolute error[%]	3.12	24.5	0.39	0.39	0.00	0.00



- *original image (256*256 pixels)*
- *gray scale of window average (16*16)*
- *class labels*

- *reproduced image by the SOFM classifier, gray scale of window average (16*16)*
- *predicted class labels*

- *reproduced image by the SVM classifier (using radial basis and full polynomial kernels), gray scale of window average (16*16)*
- *predicted class label*

Figure 5. Original test image and reconstructed image by the SOFM and SVM classifiers

For this particular experiment, validation of the SVM constructed using radial basis function and full polynomial kernels has shown its superior performance in comparison to SOFM.

3.2 Prediction of Horizontal Force on Vertical Break Water

Vertical breakwaters are upright concrete structures built in the sea in order to protect an area from wave attack. They are specifically used in harbour areas with relatively high water depths, where traditional rubble-mound (or rock slope) breakwaters become a rather expensive alternative. The stability analysis of such vertical breakwaters (as in Figure 6) relies on an estimation of the total force and overturning moments caused by dynamic wave-pressure action (Meer and Franco, [5]).
The total horizontal force on a vertical breakwater is the result of the interaction between the wave field, the foreshore and the structure itself (Gent and Boogaard [1]). The wave field in the offshore region can be described using two standard parameters, namely, the significant wave height in the offshore region (H_s) and the corresponding peak wave period (T_p). The effect of the foreshore is described by the slope of the foreshore (tan θ) and the water depth in

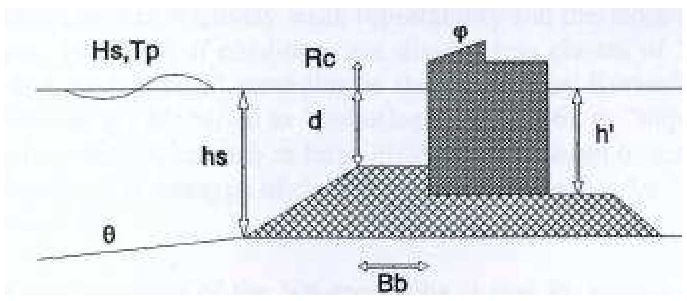


Figure 6 Vertical breakwater layout and the different parameters involved (adopted from Gent and Boogaard, 1998)

front of the structure (h_s). The structure is characterised by the height of the vertical wall below and above the water level (h' and R_c), the water depth above the rubble-mound foundation (d) and the shape of the superstructure (φ). The berm-width is also added as an additional parameter (B_b).

Present-day design practice of vertical breakwater mainly depends on physical model tests and empirical formulae. The data used for this study are collected from physical-model tests performed at several laboratories. Based on the published data, the parameter corresponding to the total horizontal force that is exceeded by 99.6% of the wave has been derived (Gent and Boogaard, [1]). In total 612 data-patterns obtained from various hydraulic research institutes in different European countries have been used in this study.

One of the best known sets of formulae to predict total horizontal forces on vertical structures is provided by the Goda method (Goda [2]). Recently, Artificial Neural Networks (ANNs) were also applied for the same problem successfully (Gent and Boogaard [1]). However they followed a slightly modified approach, including explicit physical knowledge adopted from Froude's law, in preparing the data for training the network. The input/output patterns were then re-scaled to a form where the H_{s0} (equivalent deep-water wave height) become 0.1m (this choice of 0.1 being rather arbitrary). As a result, every input output pattern was scaled with a different factor λ. In this way a 'new' data set still consisting of 612 input/output patterns was produced but the input dimension was reduced from 9 to 8 by removing H_{s0}. Gent and Boogaard reported that, as a result of this reduction of the input dimension, in addition to the important physical knowledge included in the data, training could be performed with NN of reduced size and complexity.

In this study, however, the possibility of using Support Vector Regression (SVR) as yet another data-driven modelling approach for prediction of the horizontal force on vertical breakwater has been investigated. The same data set described above is used in order to make comparison with the previous NN modelling results. Five types of kernel functions, namely the simple dot product, simple polynomial, radial basis function, semi-local and linear splines(see the Appendix) were used for the SVM training. Using the simple dot product kernel amounts to approximating with a linear SVM, while the rest of the kernel results in non linear SVMs. Table 3 shows the RMSE of best performing SVMs for each kernel type with the corresponding numbers of support vectors, while Figure 7a shows the changes in the training and validation RMSE for radial basis function kernel which resulted from different combination of parameters (such as *C*, ε and γ). It shows that, while it is possible to reduce the training error by increasing the capacity of the machine by fine tuning the different parameters, the generalisation ability will eventually deteriorate. The other observation is that an increase in the performance of the machine on the training data usually corresponds to an increase in the number of corresponding support vectors. This, in its turn, results in a relatively larger computational time and memory requirement. There should, therefore, always be a trade off between these factors, and this also holds in the case of ANN applications. In the present application, the SVMs were found to generalise well by setting the capacity factor *C* between 10 and 100. Similarly, in most cases, ε values in the range of 0.01 to 0.025 resulted in the best performing SVM. For radial basis function and semi-local kernels, γ values in the range of 3 to 10 were found to be appropriate. The choice of even larger values of the capacity factor *C* or smaller value for the error insensitive region ε could reduce the training error still further. However, this usually leads to an *overfitting* on the training data and a lesser generalisation ability of the resulting SVM. Instead, the performances of the SVMs could still be improved by exploring even more combinations in the above parameter ranges during the training stage.

Table 3 General performance of SVMs with different kernel functions

Kernel function	RMSE		Total number of support vectors	
	Training	Testing	positive SV	Negative SV
simple dot product	0.311	0.288	143	146
simple polynomial	0.218	0.219	129	138
radial basis function	0.208	0.199	213	222
semi-local	0.174	0.210	142	131
linear splines	0.211	0.191	119	133
ANN(with sigmoid tran. fun.)	0.203	0.186		

As can be seen from Table 3, the overall performance of the SVM (especially with radial basis and linear spline kernels) is quite comparable with that of ANN's with mean absolute errors (both in training and testing) of less than 10% of the data range. However one can also see that the SVM predicted the output quite well in the lower and middle range of the data set while ANN seemed to predict the higher range of the output values relatively better. For both cases, a large scatter is observed, which is to a large extent caused by the quality of the measured data (see Gent and Boogaard [1]). The overall results of this investigation, however, show that SVMs are still able to provide realistic predictions of forces on vertical structures.

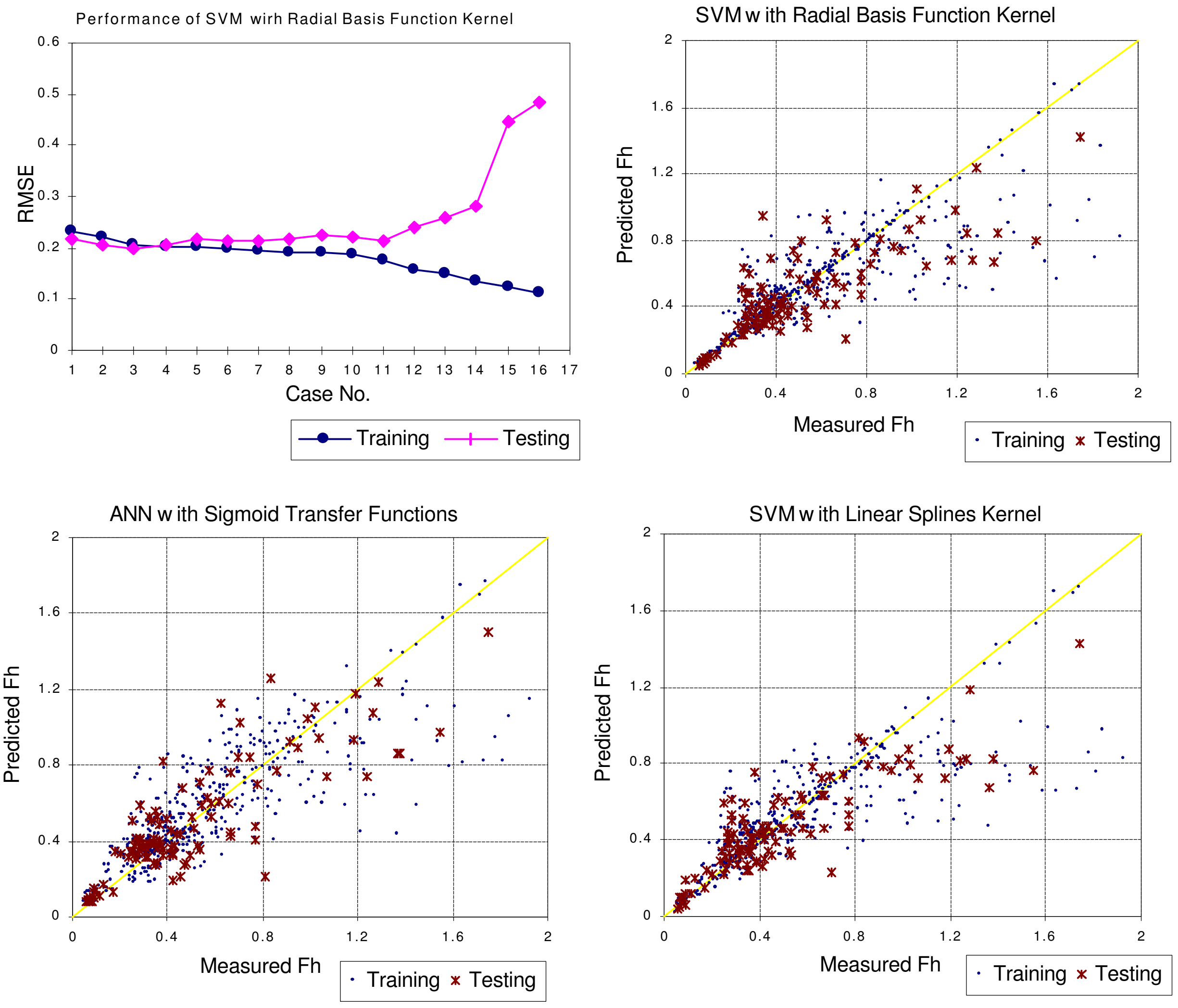


Figure 7 Scattered plot of predicted verses measured output from SVMs and ANN

4. Conclusions and Recommendations

In this paper the main principles of SVM are reviewed and it has been shown that they provide a new approach for feature classification and multiple regression problems with clear connections to the underlying statistical learning theory. In both cases, SVM training consists of solving a – uniquely solvable – quadratic optimisation problem, unlike the ANN training, which requires non-linear optimisation with the possibility of converging only in local minima. An SVM is largely characterised by the choice of its kernel; as a result it is required to choose the appropriate kernel for each application cases in order to get satisfactory results. For digital remote sensing data, the SVM classifies the four features distinctively, with a better performance than with the SOFM. The SVM has also shown a satisfactory performance for the prediction of forces on vertical structures due to dynamic waves. In this case the performance of the SVM was comparable to that of ANNs trained with sigmoid transfer functions.
In general, SVMs provide an attractive approach to data modelling and have started to enjoy increasing popularity in the machine-learning and computer-vision research communities. This paper shows the potential of SVMs for applications in civil engineering, and specially hydroinformatics, problems. However, several aspects remain to be addressed. For example, determining the proper parameters C and ε is still a heuristic process and almost surely sub-optimal. Automation of this process could be beneficial. The other limitations are that of speed and the maximum size of the training set relative to the available memory resource, and these still need further research.

Acknowledgement

The authors would like to thank Dr. Henk van den Boogaard from Delft Hydraulics for providing the data on forces on vertical breakwaters. They are also very grateful to Prof. M.B. Abbott for his comments on the manuscript.

References

1. Gent, M. R.A. and Boogaard, H. v.d., *Neural Network and Numerical Modelling of Forces on Vertical Structures*, MAST-PROVERBS report, Delft Hydraulics, 1998.
2. Goda, Y., *Random Seas and Design of Maritime Structures*. University of Tokyo Press, 1985.
3. Gunn, S., Support Vector Machines for Classification and Regression, *ISIS Technical Report*, 1998.
4. Kuhn, H. W., and Tucker, A. W., Nonlinear programming, *In Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*, pp. 481-492, Berkeley, 1951.
5. Meer, J.W. v.d and Franco, L., *Vertical breakwaters*, Delft hydraulics publications No. 487, 1995.
6. Minoux, M., *Mathematical Programing: Theory and Algorithms*, John Wiley and Sons, 1986.
7. Osuna, E., Freund, R. and Girosi, F., An improved training algorithm for support vector machines, *In Proc. of the IEEE Workshop on Neural Networks for Signal Processing VII*, pp. 276-285, New York, 1997.
8. Platt, J., *Sequential Minimal Optimization: A fast algorithm for training support vector*

machines, Technical report MSR-TR-98-14, Microsoft Research, 1998.
9. Saunders C., Stitson M.O., Weston J., Bottou L., Schölkopf B.and Smola A., Support Vector Machine - Reference Manual, *Royal Holloway Technical Report CSD-TR-98-03*, published by Royal Holloway, 1998.
10. Scholkopf, B., *Support Vector Learning*, R. Oldenbourg, Munich, 1997.
11. Smola, A., *Regression Estimation with Support Vector Learning Machines*, Technische Universitat Munchen, 1996.
12. USDA, Soil Conservation Service (SCS), *TR-20 project formulation hydrology*, Central Technical Service, Portland USA, 1965.
13. Vapnik, V., *Statistical Learning Theory*, Wiley, New York, 1998.
14. Vapnik, V., and Chervonenkis, *Theory of Pattern Recognition [in Russian],* Nauka, Moscow, 1974.
15. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer, New York, 1995.
16. Velickov S., Solomatine D.P., Yu X. and Price R.K., Application of data mining technologies for remote sensing image analysis, *to be published on the Proc. of the 3rd International Conference on Hydroinformatics (2000),* Iowa City, USA.

APPENDIX

A. SV kernel functions

1. The simple dot product: $K(\mathbf{x},\mathbf{y}) = \mathbf{x} \,.\, \mathbf{y}$

2. The simple polynomial kernel: $K(\mathbf{x},\mathbf{y}) = ((\mathbf{x} \,.\, \mathbf{y}) + 1)^d$
Where the degree of polynomial *d* is user defined.

3. Vovk's real polynomial: $K(\boldsymbol{x},\boldsymbol{y}) = \dfrac{1-(\boldsymbol{x}.\boldsymbol{y})^d}{1-(\boldsymbol{x}.\boldsymbol{y})}$

4. Radial Basis Function: $K(\mathbf{x},\mathbf{y}) = \exp(-\gamma|\mathbf{x} - \mathbf{y}|^2)$
Where γ is user defined

5. Two layer neural network: $K(\mathbf{x},\mathbf{y}) = \tanh(b(\mathbf{x} \,.\, \mathbf{y}) - c)$
Where *b* and *c* are user defined

6. Linear splines: $K(x,y) = \Pi_{k=1}^{n}(\mathbf{x}^k, \mathbf{y}^k)$

7. Semi local kernel: $K(\mathbf{x},\mathbf{y}) = [(\mathbf{x}.\,\mathbf{y}) + 1]^d \exp(-\|\mathbf{x} - \mathbf{y}\|^2\, \sigma^2)$
Where *d* and σ are user defined

B. Selected Features for image classification

Descriptor	Formula
Mean	$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$

Descriptor	Formula
Variance	$s^2 = \frac{\mathrm{n}}{\mathrm{n-1}} \sum_{i=1}^{n} \left(X_i - \overline{X} \right)^2$
Energy	$\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} p_{ij}^2$
Entropy	$-\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} p_{ij} \log p_{ij}$ (for $p_{ij} \neq 0$)
Momentum	$-\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} (Gi - Gj)^2 p_{ij}$
Cluster Shade	$\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} ((Gi - \mu_i)(Gj - \mu_j))^3 p_{ij}$
Cluster prominence	$\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} ((Gi - \mu_i)(Gj - \mu_j))^4 p_{ij}$
Correlation	$\sum_{j=1}^{Ng} \sum_{i=1}^{Ng} \frac{(Gi - \mu_i)(Gj - \mu_j)}{\sigma_i \sigma_j} p_{ij}$
Where : Gi is the the gray scale with the rank of i P $_{ij}$ is the normalized co-occurrence matrix $\mu_i = \sum_{i=1}^{Ng} Gi \sum_{j=1}^{Ng} p_{ij}$ $\quad \sigma_i = \sum_{i=1}^{Ng} (Gi - \mu_i)^2 \sum_{j=1}^{Ng} p_{ij}$	