Abstract: Multivariate data analysis techniques have the potential to improve data analysis. Support Vector Machines (SVMs) are a recent addition to the family of multivariate data analysis. A brief introduction to the SVM Vector Machines technique is followed by an outline of the practical application.

Key-Words: SVM vector machines, data analysis

1 Introduction

A common problem in various areas of science is that of classification. One approach is to develop an algorithm which finds complex patterns in input example data (labeled training data) to learn the solution to the problem. This is called supervised learning. Such an algorithm can map each training example onto two categories, more than two categories, or continuous, real-valued output. A potential problem with this approach is noise in training data. There may be no correct underlying classification function. Two similar training examples may be in different categories. Another problem may be that the resulting algorithm misclassifies unseen data because it has overfitted the training data. A better goal is to optimize “generalization” – the ability to correctly classify unseen data. Next section shows how the Support Vector Machine learning methodology addresses these problems. The description follows that of several references in the literature [4,5,21].

2 Problem Formulation

The SVM vector method was developed to construct separating hyperplanes for pattern recognition problems [4,5]. In the 1990s it was generalized for constructing nonlinear separating functions and for estimating real-valued functions. Applications of SVMs include text categorization, character recognition, and bioinformatics and face detection. Support Vector Machines (SVM) are learning machines that can perform classification and real valued function approximation. SVM creates functions from a set of labeled training data and operate by finding a hypersurface in the space of possible inputs. This hypersurface will attempt to split the positive examples from the negative examples. The split will be chosen to have the largest distance from the hypersurface to the nearest of the positive and negative examples. Intuitively, this makes the classification correct for testing data that is near, but not identical to the training data. In detail, during the training phase SVM takes a data matrix as input, and labels each sample as either belonging to a given class (positive) or not (negative). SVM treats each sample in the matrix as a point in a high-dimensional feature space, where the number of attributes determines the dimensionality of the space. SVM learning algorithm then identifies a hyperplane in this space that best separates the positive and negative training samples. The trained SVM can then be used to make predictions about a test sample’s membership in the class. In brief, SVM non-linearly maps their n-dimensional input space into a high dimensional feature space. In this high dimensional feature space a linear classifier is constructed.

3 SVM based algorithm

The main idea of the SVM approach is to map the training data into a high dimensional feature space in which a decision boundary is determined by constructing the optimal separating hyperplane. Computations in the feature space are avoided by using a kernel function. The formal goal is to estimate the function \( f : \mathbb{R}^N \rightarrow \{\pm 1\} \) using input output training data \((x_1, y_1), \ldots, (x_G, y_G) \in \mathbb{R}^N \times \{\pm 1\} \) such that \( f \) will correctly classify examples \((x, y)\), i.e. \( f(x) = y \cdot \ell \) is the number of training examples. For generalization we restrict the class of functions from which \( f \) is chosen. Simply minimizing the training error does not necessarily result in good generalization. SVM Vector classifiers are based on the class of hyperplanes \( \bar{w} \cdot \bar{x} + b = 0 \) with \( \bar{w} \in \mathbb{R}^N \), \( b \in \mathbb{R} \) and corresponding to the decision function \( f(\bar{x}) = \text{sign}[(\bar{w} \cdot \bar{x}) + b] \). \( \bar{w} \) is called the weight vector and \( b \) the threshold. \( \bar{w} \) and \( b \) is the parameters controlling the function and must be learned from the data. The unique hyperplane with maximal margin of separation between the two classes is called the optimal hyperplane. The optimization problem thus is to find the optimal hyperplane. Both the optimization problem and the final decision function depend only on dot products between input vectors. This is crucial for the successful generalization to the nonlinear case. If \( f(\bar{x}) \) is
a nonlinear function of $\bar{x}$ one possible approach is to use a neural network, which consists of a network of simple linear classifiers. Problems with this approach include many parameters and the existence of local minima. The SVM approach is to map the input data into a high, possibly infinite dimensional feature space, $F$ via a nonlinear map $\Phi : \mathbb{R}^n \rightarrow F$. Then the optimal hyperplane algorithm can be used in $F$. This high dimensionality may lead to a practical computational problem in feature space. Since the input vectors appear in the problem only inside dot products, however, we only need to use dot products in feature space. If we can find a kernel function, $K$, such that 

$$K(\bar{x}_1, \bar{x}_2) = \Phi(\bar{x}_1) \cdot \Phi(\bar{x}_2)$$

then we don’t need to know $\Phi$ explicitly. Mercer’s Theorem tells us that a function $K(\bar{x}, \bar{y})$ is a kernel, i.e. there exists a mapping $\Phi$ such that

$$K(\bar{x}_1, \bar{x}_2) = \Phi(\bar{x}_1) \cdot \Phi(\bar{x}_2)$$

We can choose from known kernel functions: polynomial of degree, Gaussian Radial Basis Function or sigmoid. We propose a new SVM-based method (BSVM), which uses the dynamic programming algorithm as a kernel function. A detailed description of experiments can be located in [19]. The result of computational experiment show that the BSVM method outperforms existing algorithms we tested.

Over the past a few years, Neural Networks, one of the branches in Artificial Intelligence technology, have gained popularity among the hydrological and hydraulic engineering community and some encouraging results have been achieved. Recently, a new tool from the Artificial Intelligence field called a Support Vector Machine (SVM) has gained popularity in the Machine Learning community. It has been applied successfully to classification tasks such as pattern recognition, OCR and more recently also to regression and time series. Mathematically, SVMs are a range of classification and regression algorithms that have been formulated from the principles of statistical learning theory. So far, these SVMs have been benchmarked against artificial neural networks (ANNs) and outperformed ANN in many application areas. It has been hypothesised that this is because there are fewer model parameters to optimise in the SVM approach, reducing the possibility of over-fitting the training data and thus increasing the actual performance. Compared with traditional artificial neural networks, training in SVMs is very robust due to their quadratic objective functions. It is useful to explore this new technology in river flow modeling area, with the hope that it could overcome some of the problems in ANN and may perform much better than the traditional linear models.

Both SVMs and ANNs can be represented as two-layer networks (where the weights are non-linear in the first layer and linear in the second layer). However, while ANNs generally adapt all the parameters (using gradient or clustering-based approaches) SVMs choose the parameters for the first layer to be the training input vectors because this minimises the VC-dimension as indicated in Figure 1.

Mathematically, a basic function for statistical learning process is

$$f(x) = \sum_{i=1}^{M} \alpha_i \phi_i(x) = w \phi(x)$$

(1)

Where the output is a linearly-weighted sum of M. The nonlinear transformation is carried by $\phi$.

The range of models represented by Equation 1 is extremely broad. SVM is a special form of them and its decision function is represented as

$$f(x) = \left\{ \sum_{i=1}^{N} \alpha_i K(x_i, x) \right\} - b$$

(2)

where $K$ is the kernel function, $\alpha_i$ and $b$ are parameters, $N$ is the number of training data, $x_i$ are vectors used in training process and $x$ is the independent vector. The parameters $\alpha_i$ and $b$ are derived by maximize their objective function.

In SVM, all input data are organised as vectors (i.e., one dimensional array) and some of these vectors are used in the modelling process (as demonstrated in Eq 2). This is quite different compared with other models like ANN and Linear TF models, which are global models. In these models, model parameters are derived from the training data set and then only the derived parameters are used in future simulations. The data for training would play no part in the prediction process. SVM is quite different. It
uses the training data for model calibration so as to estimate the model parameters, but also keeps the most important part of the input vectors in its model. These vectors are called support vectors (only a small number of training vectors are chosen). The unique structures of the kernel functions used for nonlinear transformation of input vectors enable SVM to get rid of most training vectors, so that the resulted model is much smaller. The reduced support vectors also improve the model’s generalisation ability and decrease the computation load. Since SVM theory was originally created from the machine learning community, this type of models is coined as Support Vector Machines.

SVM has a strong nonlinear ability and this is analogous to the nonlinear treatment for the traditional linear models. As we know, it is possible to transform the input variables with a certain nonlinear functions so that linear models can be used to model nonlinear processes (Generalized linear system framework). For example, an input vector \( \mathbf{x} = (x_1, x_2) \) can be transformed into a higher dimension input vector \( z = (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \), which can then be treated as a linear system. In a similar fashion, SVM uses some specific kernel functions which transform the input vector as an inner product of nonlinear functions in the model. The selection of suitable kernel function for a specific problem is a very complicated process at the moment and much more research work is still needed.

A major problem in any model training is the decision about the complexity of the model’s structure. More complicated models tend to do well in training but do badly in prediction. For example, a common problem in ANN’s applications is overfitting, sometimes the model’s weights are even less than the training data points.

As indicated by Figure 2, to chose a suitable model structure which achieve the best test result is very important. In this aspect, SVM has an advantage over ANN that it can automatically minimize the number Support Vectors, thus to improve its generalisation ability.

In the modeling process, rainfall data series \( (x_n, x_{n+1}, \ldots) \) and flow data series \( (y_n, y_{n+1}, \ldots) \) are used to construct vectors for the training and testing. At each time step \( t \), \( y_{t+1} \) is the target value and some fixed moving windows for rainfall and flow data are selected for input vectors. An input vector can have a mixture of various variables (e.g., rain, flow, temperature, date, etc). At each computation step, we sequentially add the newly acquired data and remove the earlier ones, to predict the flow in the future. Before the training, several key parameters have to be selected by manual operations. They are

a) Three parameters to control the SVM training: Cost of error \( C \), Slackness tube \( \varepsilon \) and kernel function
b) Window sizes for rainfall and flow data;
c) Scale factors for rainfall and flow data

In the process above, \( C \) is useful for controlling the smoothness of the function. Large \( C \) values penalise the errors, hence the resulted SVMs have small number of SV. Slackness tube with \( \varepsilon \) is a new concept (in traditional least square method, \( \varepsilon \) is always zero ) and the input data which fall in the tube are not penalised. Three popular kernel functions are tested: \( D \)th degree polynomial (only 2 is used in this project), radial basis and Sigmoid functions. Various window sizes were tested (3 rain, 3 flow; 1 rain, 5 flow; 0 rain, 10 flow; 10 rain, 0 flow; 20 rain, 0 flow; …). Scale factors are used to transform rainfall and flow data into a similar range otherwise the data of high values (i.e., small unit) would dominate the training process. Most of work in this part is manual, hence a tedious process due to the huge number of combinations.

In the model calibration stage, we find that SVM can perform very well in many cases. With the data used in the training (Birdcreek), Polynomial kernel function performed much better than radial basis and Sigmoid kernel functions (see Figure 3). Comparison with linear TF model clearly demonstrated the nonlinear effect of SVMs. The TF model’s overestimation of small peaks were removed by polynomial kernel functions.

![Figure 2 The influence of model complexity](image-url)
Despite the success of SVM training, in testing stage, we found that SVMs were usually less stable than linear TF models and tended to perform poorly in comparison. However, there were some interesting features from SVMs that could make them useful for modeling high flows. For example, in Figure 4, although SVM simulated flow is not as close to the measured flow as TF, its predicted peak flow is much closer to the real peak than the TF model’s one.

4 Practical Experiments
In this section a data set is used to test BSVM. We consider is the Swiss roll data. It is a three dimensional data that looks like Figure 1.

Fig 1. Swiss Roll Data
The distance between the samples is the geodesic distance of the surface of two samples. The BSVM method is applied and the average testing misclassification error equal to 3.9%. It shows that the BSVM method is good at this case.

5 Conclusion
BSVM provide nonlinear function approximations by mapping input vectors into a high dimensional feature space where a hyperplane is constructed to separate classes in the data. Computationally intensive calculations in the feature space are avoided through the use of kernel functions. BSVM correspond to a linear method in feature space which makes them theoretically easier to analyze.

Over the past a few years, Neural Networks, one of the branches in Artificial Intelligence technology, have gained popularity among the hydrological and hydraulic engineering community and some encouraging results have been achieved. Recently, a new tool from the Artificial Intelligence field called a Support Vector Machine (SVM) has gained popularity in the Machine Learning community. It has been applied successfully to classification tasks such as pattern recognition, OCR and more recently also to regression and time series. Mathematically, SVMs are a range of classification and regression algorithms that have been formulated from the principles of statistical learning theory. So far, these SVMs have been benchmarked against artificial neural networks (ANNs) and outperformed ANN in many application areas. It has been hypothesised that this is because there are fewer model parameters to optimise in the SVM approach, reducing the possibility of over fitting the training data and thus increasing the actual performance. Compared with traditional artificial neural networks, training in SVMs is very robust due to their quadratic objective functions. It is useful to explore this new technology in river flow modeling area, with the hope that it could overcome some of the problems in ANN and may perform much better than the traditional linear models.

References:


[22] Zhang L. and B. Zhang, “A