Identification of support vector machines for runoff modelling
Michaela Bray and Dawei Han

ABSTRACT
This paper describes an exploration in using SVM (Support Vector Machine) models, which were initially developed in the Machine Learning community, in flood forecasting, with the focus on the identification of a suitable model structure and its relevant parameters for rainfall runoff modelling. SVM has been applied in many fields and has a high success rate in classification tasks such as pattern recognition, OCR, etc. The applications of SVM in regression of time series are relatively new and they are more problematic in comparison with classifications. This study found that exhaustive search of an optimum model structure and its parameter space is prohibitive due to their sheer size and unknown characteristics. Some parameters are very sensitive and can increase the CPU load tremendously (and hence result in very long computation times). All these make it very difficult to efficiently identify SVM models, which has been carried out by manual operations in all study cases so far. The paper further explored the relationships among various model structures ($\ell_i$-SV or $\nu$-SV regression), kernel functions (linear, polynomial, radial basis and sigmoid), scaling factor, model parameters (cost C, epsilon) and composition of input vectors. These relationships should be able to provide useful information for more effective model identification in the future. The unit response curve from SVM was compared with a transfer function model and it is found that a TF model outperforms SVM in short-range predictions. It is still unclear how the unit response curve could be utilised for model identification processes and future exploration in this area is needed.

Key words | flood forecasting, model identification, parameter optimisation, rainfall runoff, support vector machines

INTRODUCTION
Over the past few years, Neural Networks, one of the branches in Artificial Intelligence technology, have gained popularity among the hydrological and hydraulic engineering community (Dibike et al. 1999; Bernd et al. 1999; Campolo et al. 1999) 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 (Cristianini et al. 1999). It has been applied successfully to classification tasks such as pattern recognition, OCR and also, more recently, 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 developed by Vapnik (1995). In recent years, a number of non-linear classification and regression SVMs have been developed and these have been benchmarked against artificial neural networks (ANNs). It has been found that the empirical performance of SVMs is generally as good as the best ANN solutions (Hearst et al. 1998). 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 (Brown et al. 1999). Compared with traditional artificial neural networks, learning in SVMs is very robust from the point
of view of the precision of the computations (Anguita et al. 1999). A major distinction between the two approaches is the training algorithm. 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 (Cherkassky & Mulier 1998). It should be stressed that most SVM success stories are in classification tasks such as pattern recognition, OCR, etc. The applications of SVM in regression of time series are relatively new and they are more problematic in comparison with classifications. It is useful to note that SVM is finding its way into the water sector (Sivapragasam et al. 2001; Dibike et al. 2001; Han & Yang 2001) and a new variant of SVM called RVM has also been attempted recently (Han et al. 2002). Despite all of these application cases published so far, users still found that it was an art to identify an optimum SVM model and the automation in model development is almost impossible. All the studies carried out so far are based on a manual process, which is tedious, slow and unlikely to reach a global optimum. There is an urgent need to establish a guideline for model identification processes, which may lead to an automated process using techniques like Genetic Programming. The study reported here has been set to explore the relationships between various model structures (ξ-SV or ν-SV regression), kernel functions (linear, polynomial, radial basis and sigmoid), scaling factor, model parameters (cost C, epsilon) and composition of input vectors. The results should be a useful contribution to solving the model identification problem mentioned above.

**BASIC THEORY OF SUPPORT VECTOR ALGORITHMS**

There are many papers and books which provide a detailed description of the theory of SVM (Vapnik 1998; Cristianini & Shawe-Taylor 2000), and hence only a brief description of SVM is given here. Like ANNs, SVM 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 the statistical learning process is

\[ y = f(x) = \sum_{i=1}^{M} \alpha_i \varphi_i(x) = w\varphi(x) \]  

where the output is a linearly weighted sum of \( M \). The nonlinear transformation is carried out by \( \varphi(\cdot) \).

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

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

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 maximising their objective function.

In SVM, all input data are organised as vectors (i.e. one-dimensional arrays) and some of these vectors...
are used in the modelling process (as demonstrated in Equation (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 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 it 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 is chosen). The unique structures of the kernel functions used for the nonlinear transformation of input vectors enable SVM to get rid of most training vectors, so that the resulting model is much smaller. The reduced support vectors also improve the model's generalisation ability and decrease the computational load. Since SVM theory was originally created from the machine learning community, this type of model is termed a Support Vector Machine.

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 certain nonlinear functions so that linear models can be used to model nonlinear processes (the generalised linear system framework). For example, an input vector \( \mathbf{x} = (x_1, x_2) \) can be transformed into a higher dimension input vector, \( \mathbf{z} = (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \), 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 functions for a specific problem is a very complicated process at the moment and all application cases in the water sector so far use the standard kernel functions developed from the classification field, which is very different to the regression problems. Hence the full potential of SVM has not been utilised and, as a result, 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 applications is overfitting: in some application cases, it has been found that the model's weights were even less than the training data points. As indicated by Figure 2, to choose a suitable model structure which achieves the best test result is very important. In this aspect, SVM has an advantage over ANN that it can automatically minimise the number of support vectors, thus improving its generalisation ability.

**IDENTIFICATION OF SVM MODELS**

Unlike former learning machines, the hypothesis space of SVM is limited to linear functions in a high-dimensional feature space. These hypotheses are trained by a learning algorithm, which is based on optimisation theory. These algorithms implement a learning bias derived from statistical learning theory. By fine-tuning the learning machine in this way the aim of optimising the machine's ability to generalise is achieved. The problem of linear regression is in finding a linear function \( y = f(x) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b \) that best interpolates a set of training points. The least-squares approach prescribes choosing the parameters \((\mathbf{w}, b)\) to minimise the sum of the squared deviations of the data,

\[
\sum_{i=1}^{l} (y_i - \langle \mathbf{w} \cdot \mathbf{x}_i \rangle - b)^2 \quad \text{(Cristianini & Shawe-Taylor 2000)}.
\]

To allow for some deviation \(\epsilon\), between the eventual targets \(y_i\) and the function \(f(x) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b\) modelling the
data, the following constraints are applied: \( y_i - w \cdot x - b \leq \epsilon \) and \( y_i - w \cdot x + b \leq \epsilon \). This can be visualised as a band or a tube around the hypothesis function \( f(x) \) with points outside the tube regarded as training errors, otherwise called slack variables \( \xi_i \). These slack variables are zero for points inside the tube and increase progressively for points outside the tube. This approach to regression is called \( \epsilon \)-SV regression (Vapnik 1998). It is the most common approach, although it is not the only one. The task is now to minimise \( ||w||^2 + C \sum_{i=1}^{m} (\xi_i + \xi_i^*) \) subject to: \( y_i - w \cdot x - b \leq \epsilon + \xi_i \) and \( (w \cdot x + b) - y_i \leq \epsilon + \xi_i^* \) (Cristianini et al. 1999). An alternative form of SVM is called \( \nu \)-SV regression (Smola & Schölkopf 1998). This model uses \( \nu \) to control the number of support vectors. Given a set of data points \( \{(x_1, z_1), \ldots (x_l, z_l)\} \), such that \( x_i \in \mathbb{R}^n \) is an input vector and \( z_i \in \mathbb{R}^l \) is the corresponding target, the form is

\[
\min_{w, b, \xi, \xi^*} \frac{1}{2} w^T w + C \left( \nu \epsilon + \frac{1}{\lambda} \sum_{i=1}^{l} \xi_i + \sum_{i=1}^{l} \xi_i^* \right)
\]

subject to \( w^T \phi(x_i) + b - z_i \leq \epsilon + \xi_i \) and \( z_i - w^T \phi(x_i) - b \leq \epsilon + \xi_i^* \) where \( \xi \) is the upper training bound and \( \xi^* \) is the lower training bound.

The role of the kernel function simplifies the learning process by changing the representation of the data in the input space to a linear representation in a higher-dimensional space called a feature space. A suitable choice of kernel allows the data to become separable in the feature space despite being nonseparable in the original input space. This allows us to obtain nonlinear algorithms from algorithms previously restricted to handling linearly separable data sets. The kernel is defined to be a function \( K(x, z) \), which computes the inner product \( \phi(x) \cdot \phi(z) \), directly from the input points. Four standard kernels are usually used in classification problems and are also used in regression cases:

Linear: \( u^T v \)
Polynomial: \( (y \times u^T v + \text{coef})^{\text{degree}} \)
Radial basis: \( e^{-r \times |u-v|^2} \)
Sigmoid: \( \tanh(y \times u^T v + \text{coef}) \)

For any model development, the most important components are the input data, called input vectors in SVM. An input vector can have a mixture of various variables (e.g. rain, flow, temperature, date, etc). For example, in this study, rainfall data series \( (x_t, x_{t-1}, \ldots) \) and flow data series \( (y_t, y_{t-1}, \ldots) \) are used to construct vectors for 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. At each computation step, we sequentially add the newly acquired data and remove the earlier ones in order to predict the flow in the future. A crucial question to ask during the model identification is what the input vectors are composed of. The combination of various rainfall and flow data in each vector could be huge (e.g. rainfall data from 0–9 and flow from 0–9 will generate 100 combinations that should be tried for optimisation purposes). Since rainfall and flow have different units and magnitudes, scale factors should be applied to make sure that the distance measurement will not be biased to the variable which has the higher values. The choices of rainfall and flow scale factors will further increase the input vectors’ complexity. Two model training parameters, cost of error \( C \) and Slackness tube \( \epsilon \) could influence the training quality and time. \( C \) is useful for controlling the smoothness of the function. Large \( C \) values penalise the errors, and hence the resulting SVMs have a small number of SV. A Slackness tube with \( \epsilon \) is a new concept (in the traditional least-squares method, \( \epsilon \) is always zero) and the input data which fall in the tube are not penalised.

It is clear that the selection of optimum SVM models is quite tedious and there is no established methodology to efficiently guide the process. A flowchart on the model selection is shown in Figure 5. It is possible to calculate the options available when optimising the model selection. For example, if the choice of rainfall window is 10 (from past one step to ten steps); flow data window, ten; rainfall scale factor, 5; flow scale factor, 5; cost \( C \), 5; \( \epsilon \), 5; kernel functions, 4, and the parameters for the kernel functions 2, the selection process would need to be carried out 2 million times! Some of the factors are very sensitive and a wrong setting could result in an extremely long training time. Hence it is very difficult to automate the selection process. Due to the limit of resources and time,
most studies have tried to minimise the work involved and hence many decisions made would not be globally optimum. Much more work is needed in this area to provide SVM users with more efficient optimisation knowledge. This study has not been intended to solve this global optimisation problem, but to explore the relationships among different factors which influence the SVM model development under a specific catchment and the results will enrich our knowledge in the identification of SVM models.

THE CATCHMENT

The data used for this paper were collected in a region called Bird Creek in the USA. The data formed part of a real-time hydrological model intercomparison exercise conducted in Vancouver, Canada in 1987 and reported by WMO (1999). The data set is divided into two parts: a calibration (training) period and a verification (testing) period. The daily rainfall values were derived from 12 rain gauges situated in or near the catchment area. The river flow values were obtained from a continuous stage recorder. The period used for model calibration spanned some eight years from October 1955 to September 1963 and the verification period ranged from November 1972 to November 1974. During the calibration period the discharge at the basin outlet ranged from 0 to 2540 m$^3$/s and the rainfall was up to 153.8 mm/d. The highest recorded discharge during the verification period was 1506 m$^3$/s (Hajjam 1997).

The Bird Creek catchment covers an area of 2344 km$^2$ and is located in Oklahoma, close to the northern state border with Kansas. The outlet of the basin is near Sperry, about ten km north of Tulsa. The catchment is relatively low lying, with altitudes ranging from 175 up to 390 m above the mean sea level. There are no mountains or large water surfaces to influence local climatic conditions. Some 20% of the catchment surface is covered by forest while the main vegetative cover is grassland. The storage capacity of the soil is very high (Georgakakos & Smith 1990).
The catchment receives significant rainfall in most years and the catchment can be classified as humid, although extended periods with very low rainfall can occur. Well defined rainy seasons occur in the spring and summer, with rain in the form of showers and thunder-showers of convective origin. Snowfall remains on the ground for only a very short period. From the latter part of July to September air temperatures are high (38°C is common) and, as a result, significant evapotranspiration occurs during this time. At the same time, the relative humidity is low and southerly breezes are common (Georgakakos et al. 1988). The river basin and the stream network are shown in Figure 4 and the training data are depicted in Figure 5.

RAINFALL RUNOFF MODELLING WITH SVM

A number of support vector machine software programmes are now available. The software used in this project is LIBSVM, a freeware programme, developed by Chih-Chung Chang and Chih-Jen (Chang & Lin 2001). The basic algorithm is a simplification of both SMO by Platt and SVMLight by Joachims. LIBSVM is capable of C-SVM classification, one-class classification, ν-SV classification, ν-SV regression and ε-SV regression. A flowchart for adopting LIBSVM for this study is illustrated in Figure 6. As mentioned above, a global optimum solution is almost impossible to find by manual operation and this study is set to explore some aspects of SVM model development to gain some behavioural knowledge of SVM with regard to rainfall and runoff modelling. Figure 7 illustrates the flowchart describing the processes carried out in this study. It is clear that only a small proportion of possible combinations are explored. In stage 1, rainfall and flow windows are all set to three and four kernels are used. To narrow down the kernel functions, the parameters of each kernel were kept at their default values and consequently a fair comparison of each SVM could be made. Obviously this posed a problem in processing the data, since the optimum values of each parameter had not yet been established and the input list consisted of over 2000 vectors. To overcome this, the original list was divided into smaller
data sets. In addition to reducing the list of input vectors, each model received a further two lists of scaled input vectors. The scale factors considered were \( S = \frac{1}{\text{max \ value}} \) and \( S = \frac{1}{2 \times \text{max \ value}} \). In all, three differently scaled lists were applied to each of the eight support vector machines, giving twenty-four possible models.

Initial results showed the importance of scaling the input vectors (Figure 8). Three kernels (radial basis, polynomial and sigmoid) could not find a hypothesis to fit the unscaled list of vectors. In the case where the input vectors were factorised, all eight support vector machines managed to predict the time of peak flow with some accuracy. However, the accuracy of the magnitude of the peak flow varied quite considerably. In all but one case, \( \nu \)-SVR with kernel radial basis normalised input vectors produced better results than those which were scaled by twice the maximum values. The results from two support vector machines stood out considerably from the remaining six. These were \( \nu \)-SVR with linear kernel and \( \epsilon \)-SVR with linear kernel. At this point, the \( \nu \)-SVR with linear kernel appeared to outperform the \( \epsilon \)-SVR with linear kernel. However, since neither machine was calibrated it was unclear whether this performance was due to the regression algorithm implemented or whether optimising the parameters within each algorithm would enhance the performance of one support vector machine over the other.

The next phase of work concentrated on the training of the two support vector machines, \( \nu \)-SVR with linear kernel and \( \epsilon \)-SVR with linear kernel (Figure 9). The deviation between the target value and the function describing the hypothesis found by the support vector machine is controlled by the \( \epsilon \) parameter. \( \epsilon \) values were varied between \( \epsilon = 1 \) to \( \epsilon = 0.00001 \) (the default value is \( \epsilon = 0.01 \)) whilst keeping all other parameters fixed at their default values (Figure 10). This was done for both normalised input vectors and vectors scaled by \( S = \frac{1}{2 \times \text{max \ value}} \). The results were analysed by calculating the root mean square error of the actual flow and the flow the machine produced. This showed that the least error occurred when \( \epsilon = 0.01 \) in the case of \( \nu \)-SVR and using normalised data. In all other cases the least error occurred when \( \epsilon = 0.1 \). For \( \epsilon \) less than 0.1 the difference in the magnitude of the error is negligible, although it increases rapidly for \( \epsilon \) greater than 0.1. This suggests that \( \epsilon \) should be...
Figure 8 | Selection of kernel functions (Linear is chosen).
set to 0.1 for both accurate results and economical computer processing. The decision to concentrate all future research solely on the \( \nu \)-SVR regression with linear kernel model was taken after comparing the performance of both regression algorithms with varying \( \varepsilon \) values. The \( \nu \)-SVR regression algorithm showed slightly enhanced learning capabilities in comparison to the \( \varepsilon \) algorithm. However, that is not to say that the \( \varepsilon \) algorithm is not worth further investigation at some later date.

The cost of error assigns a penalty for the number of vectors falling between the two hyperplanes. If the data is good quality the distance between the two hyperplanes is narrowed down. If the data is noisy it is preferable to have a smaller value of \( C \) which will not penalise the vectors. To ascertain the optimum cost value, the support vector machine made from the \( \nu \)-SVR regression algorithm and linear kernel was run several times with differing values of \( C \). This was done for two sets of input vectors: one set was normalised and the second set was scaled by twice the maximum value in the data set. The performance of the

---

**Figure 9** Selection of regression models (\( \nu \) is chosen), scale factor (normalised).

**Figure 10** Selection of \( \varepsilon \) factor (0.1 is chosen).
models was compared by calculating the root mean square error of the output flow given by the hypothesis with that of the actual flow. The model produced similar results for both sets of scaled input vectors. The cost value was chosen to be 2 since it produced the least error, while being the most economical on computing processing (Figure 11).

At each stage of the research all the models have had both a normalised input vector list and an input vector list which has been scaled by twice the maximum value in the data set. In all cases the support vector machines have performed better with the normalised input list. For the chosen support vector machine using the $\nu$-SVR algorithm and linear kernel, the difference in results produced by the two input lists is very slight. Nevertheless it was decided that all further work would be carried out using a normalised list of input vectors. In the end, the support vector machine used was the $\nu$-SVR algorithm with linear kernel, $\varepsilon$ set to 0.01, the cost set to 2 and the input list normalised. Since optimum values (more likely local optima) of $\varepsilon$ and cost had been found, it was not longer necessary to split the data up into smaller sets since the support vector machine could cope with the complete set of calibration data. Further work is undertaken to vary the number of flow data and rainfall data (from 0–6). The input lists were also normalised and the complete data set was used. It was found that the support vector machine performed very well with all rainfall input lists. It managed to learn the time, period and magnitude of elevated flow, although the magnitude of the peak flow was slightly overestimated. As in training, optimum performance was achieved with one rain observation in each vector. The error in the test output was slightly higher overall than that of the trained output. The support vector machine also performed very well with all flow input lists. It managed to learn the time, period and magnitude of elevated flow, although the magnitude of the peak flow was slightly overestimated. As in training, optimum performance was achieved with three flow observations in each vector. As expected, the error in the test output is higher than that of the trained output. However, optimum performance in terms of accuracy and computer processing is achieved with a combination of one rain observation and three flow observations (Figure 12).

To predict flow for more than one step, multi-step SVM models are used. For three time step analysis, the input list of vectors was constructed so that the machine would have to find a mapping to the target from data observed three time steps (18 h) in advance. It was assumed that rainfall could be predicted perfectly so input values for rainfall were taken one time step away from the target. Training results show that the support vector machine learns well with all rain and flow input combinations although there is a slight underestimation of elevated results indicate that the support vector machine does not rely heavily on observed rainfall when searching for a hypothesis. Best results were produced with one rainfall and three flow observations in the input vectors. The improvement in the performance of the model with increased flow observations is more significant than the improvement due to increased rainfall observations. This suggests that the machine relies more on the runoff observations than rainfall observations.

The hypotheses generated whilst investigating the different quantities of rain and flow observations in the input vector list were verified using the corresponding input vectors built from the test data set. As before, rain and flow observations were fixed at three while the respective flow and rain observations ranged between 0 and 6. The input lists were also normalised and the complete data set was used. It was found that the support vector machine performed very well with all rainfall input lists. It managed to learn the time, period and magnitude of elevated flow, although the magnitude of the peak flow was slightly overestimated. As in training, optimum performance was achieved with one rain observation in each vector. The error in the test output was slightly higher overall than that of the trained output. The support vector machine also performed very well with all flow input lists. It managed to learn the time, period and magnitude of elevated flow, although the magnitude of the peak flow was slightly overestimated. As in training, optimum performance was achieved with three flow observations in each vector. As expected, the error in the test output is higher than that of the trained output. However, optimum performance in terms of accuracy and computer processing is achieved with a combination of one rain observation and three flow observations (Figure 12).
Figure 12 | Test data for one step lead time.
flow and its duration in all cases. Optimum results are achieved with three rain observations and four flow observations in each input vector. The hypotheses also generalise well with all rain and flow input combinations. Again there is some slight underestimation of flow and its duration. This time optimum results are achieved with three rain and three flow observations in each input vector list. The best overall combination of input data is three rain observations and three flow observations (Figure 13).

For six time step analysis, the input list of vectors was constructed so that the machine would have to find a mapping to the target from data observed six time steps (36 h) in advance. As with three time step analysis, it was assumed that rainfall could be predicted perfectly and therefore input values for rainfall were taken just one time step away from the target. Initial training results shows a marked depreciation in the support vector machine’s ability to learn. It is still capable of accurately learning when peak runoff occurs. However, the magnitude and duration is underestimated by a factor of 2. As the number of rain inputs is increased the less accurate the found hypotheses become. Conversely, the hypotheses become less accurate as the number of flow inputs is increased. Analyses of the test results show the same characteristics as the training data results. However, no further decline in the accuracy of the predicted time and magnitude of peak flow is observed. This suggests that, although the learning ability of the support vector machine has been reduced, its ability to generalise has not. Increased rain observations improve the accuracy of peak runoff predictions but at the expense of the accuracy of the predicted time and duration. The optimum input vector consists of three rain and three flow observations, although generally speaking the machine has run out of steam (Figure 14).

UNIT RESPONSE OF SVM

Unit response is an important indicator in transfer function models. It is possible to work out the characteristics of a catchment from unit response curves, e.g. concentration time, time delay, percentage runoff, etc. A TF model was also built for the study catchment and the comparison for 1 to 6 lead step test data between the two is illustrated in Figure 15. The unit response curves for these two types of models are described in Figure 16. It can be found that, for short-range predictions, the TF model performs better than SVM, but for longer range it is the other way round. It would be interesting to see if this pattern exists in other catchments. The unit response curves are quite different between SVM and TF, albeit both showed stability. It is known that the unit response of a TF model could provide useful knowledge about its suitable model structure, so it is possible that the unit response of SVM may also have some extra information to offer for model development. (Here, several unit (or double or triple units) response curves may be necessary to explore the full potential of nonlinear features in SVM.) It is interesting to note that there is a nonlinear effect illustrated in Figure 17 when the unit rainfall is doubled. However, when two consecutive rainfalls are fed into the SVM model there is a huge discrepancy between the actual response and the linear one (Figure 18). It is unclear why the SVM model behaves like this. Further study is needed in this area to explore the information embedded in the unit response curves.

DISCUSSIONS

This paper illustrates that optimisation of SVM is a very complex process and the current practise using manual operation is unlikely to yield global optima. However, automation by techniques like Genetic Programming is too premature until more knowledge is gained about the relationships between the SVM’s components. For example, to use GP, it is necessary to specify the parameter space and it has been found that some SVM parameters are very sensitive and GP could easily be trapped in some CPU-intensive dip so the search time would be too long to be practical. In this study, some interesting observations have also been found.

Positive points

1. SVM’s nonlinear features are potentially useful for modelling high flows, albeit only linear kernels are explored here.
Figure 13: Test data for three step lead time.
Figure 14 | Test data for six step lead time.
2. SVM is able to select the key vectors in the training process as its support vectors and remove the nonsupport vectors automatically from the model. This makes the model cope well with noisy conditions.

3. With some key actual training vectors embedded in the models as support vectors, SVM has the potential to trace back historical events so that future predictions can be improved with the lessons learnt from the past.

4. SVM’s input vectors are quite flexible: hence it is very easy to incorporate other influential factors into the model (such as temperature, evaporation, date, etc).

5. The parameter optimisation in SVM is a convex problem: hence there is only one optimum point.

This is advantageous over ANN, which has many local optima.

Negative points

1. Because of the embedding of past historical data in the model, SVMs are huge in size and will demand large amounts of RAM to carry out the computations.

2. Scale factors are needed to unify variables with different units (e.g. rain and flow). Unlike ANN, there is no need to scale everything down to below one. However, it is convenient to scale the values to a certain range so that the parameters for $\varepsilon$ and $C$ could be easily tuned.

3. The training process is usually much slower than for linear models.
4. It is a semi-black-box model (albeit much better than ANN in this aspect): hence unexpected results could still happen in applications.
5. Correct kernel function is crucial to the success of SVM. However, it is a tedious and complicated task to find a suitable kernel function for specific problems.
6. Model extrapolation can be poor, since the model depends on the past records as support vectors.
7. SVM produces only point prediction and is not designed for probabilistic forecasts.
8. Much of the training work is still a manual process, as described above.

CONCLUSIONS

Despite its success in the machine leaning field (mainly in classification problems), SVM still has many problems in its application in rainfall and runoff modelling. This paper illustrates the difficulties in searching for global optima for a SVM model. The accumulated experience and knowledge in using SVM should gradually improve our understanding of the unique features and inter-relationships among SVM components. Ultimately, automation (or partial automation) of model identification should be possible to relieve a large part of the tedious manual operations currently practised in the research community.

REFERENCES

Hajam, S. 1997 Real time flood forecasting model intercomparison and parameter updating using rain gauge and weather radar data. PhD thesis, Telford Research Institute, University of Salford, UK.
Han, D. & Yang, Z. 2001 River flow modelling using Support Vector Machines. 29th IAHR Congress, Beijing, China, September 17–21, 2001.