8-2010

Data Mining Using Surface and Deep Agents Based on Neural Networks

Subhash Kak
Oklahoma State University, subhash.kak@okstate.edu

Yuhua Chen
University of Houston, yuhua.chen@mail.uh.edu

Lei Wang
University of Houston, lwang21@uh.edu

Data Mining Using Surface and Deep Agents Based on Neural Networks

Subhash Kak
Oklahoma State University
subhash.kak@okstate.edu

Yuhua Chen
University of Houston
yuhua.chen@mail.uh.edu

Lei Wang
University of Houston
lwang21@uh.edu

ABSTRACT

This paper presents an approach to data mining based on an architecture that uses two kinds of neural network-based agents: (i) an instantaneously-trained surface learning agent that quickly adapts to new modes of operation; and, (ii) a deep learning agent that is very accurate within a specific regime of operation. The two agents perform complementary functions that improve the overall performance. The performance of the hybrid architecture has been compared with that of a back-propagation network for a variety of classification problems and found to be superior based on the RMS error criterion.

Keywords

Data mining, instantaneously trained networks, on-line learning

INTRODUCTION

Most current techniques of data mining use one-pass classification modeling of very large data sets (Salchenberger *et al.* 2007). These techniques are not very effective in the analysis and prediction of trends for data that is changing quickly (Aggarwal *et al*, 2006). Here we propose the use of instantaneously neural networks (Kak, 1994; Kak, 1999) for dynamic classification of data sets. This constitutes a classification system in which the training model can adapt quickly to the changes of the underlying data stream. Although instantaneously trained neural networks have been applied in a variety of applications (e.g. Kak, 1999) they have not been used for explicit data mining before as far as the authors are aware. Experimental results on benchmark problems indicate that the system maintains high classification accuracy in an evolving data stream, while providing an efficient solution to the classification task.

The proposed hybrid architecture employs a surface learning component which can adapt to quick changes and a deep learning component which is highly accurate to minimize errors within the same regime of operation. A high level cognitive agent monitors the outputs from the surface learning and deep learning components, and automatically generates desirable system outputs. The proposed model provides a framework which may lead to flexible intelligence that is able to obtain classification on the fly. It will be helpful for processing of information associated with complex "intelligent" tasks in the real world with dynamic and intentional phenomena in the presence of uncertainty about the environment. It will be able to explore novel and hybrid methods that extend the breadth of reasoning to include uncertain and dynamic environments.

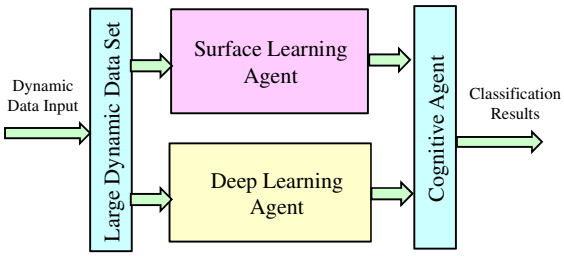

Figure 1. Hybrid System Architecture

Hybrid System Architecture for On-Line Learning

The hybrid system being proposed here represents an approach different from where the estimation is made based solely on signal theory (e.g. Wang *et al.* 2009). In addition to applications to financial or other time varying data, the system can also be used for cybersecurity applications.

The proposed hybrid system architecture is shown in Figure 1. It consists of a *surface learning agent*, a *deep learning agent*, and a *cognitive agent*. In this architecture, the surface learning agent and the deep learning agent work in parallel in responding to the changes of the environment. The surface learning agent picks up the short-term structure of the information signal, and performs quick analysis to respond to the stimulus. The deep learning agent seeks out long-term structure of the information, and abstracts longer correlations present within the system. The high level cognitive agent monitors the environment, as well as the decisions made by the surface and deep learning agents, and generates the most desirable system output.

The problem statement is as follows: the input $X(t)$ is generated by a random switching between several random processes $\{X_i(t), i = 1,...,N\}$; S is a dynamic system; and $Y(t)$ is the output random process. The objective is to predict the realization $Y(t)$ of the output process from its past values. The motivation for solving this problem is the consideration of real world processes where the input is time varying in such a manner that it is best represented as a series of different random processes.

If one were interested in predicting $Y(t)$ based on its past, it is essential to learn which of the N X_is is at the input, where it is assumed that the statistical characteristics of the X_is are known so that its prediction can be made relatively easily. The problem arises from the fact that the input signal switches between different processes and therefore no single statistical technique will work at the output all the time.

Deep learning operates within the regime of a specific index i of X_i and provides the best performance with respect to some criterion (such as minimization of RMS error). One would expect that it would take the deep learning agent some amount of time to adapt to the statistical characteristics of the signal at the input and, therefore, it would be assumed that the switching between the modes takes place slower than this characteristic time. Surface learning operates best during the transition regime between, let us say, the indices i and j.

The terminology of the surface and deep learning agents comes from linguistics. In problems of speech or text understanding, it is essential to know the context, without which a narrative may appear meaningless. The syntax is provided by the surface structure, whereas the semantics come from the deep structure. The proposed structure of Figure 1 may be taken as having an analogy with syntax and semantics.

The cognitive agent in the hybrid architecture monitors the errors produced by the surface learning agent and the deep learning agent, and automatically switches the system output according to the system conditions. Its role can be as simple as making a binary decision of selecting the system output from either the surface learning agent or the deep learning agent, or as complex as synthesizing the outputs of the two learning agents into a high level of intelligence. When the system function changes suddenly, the surface learning agent is able to catch up with the changes quickly while the deep learning agent requires more time to produce results with acceptable outputs. In this case, the cognitive agent will use the outputs produced by the surface learning agent. When the system stabilizes in a certain operational region, the deep learning agent will eventually catch up and produces outputs with more accuracy. When this happens, the cognitive agent will select the outputs from the deep learning agent instead. Thus, the hybrid architecture can quickly respond to the changing system.

Components of Hybrid Architecture

The components of the hybrid system can be chosen in a variety of ways. When signal prediction is involved, they must perform some kind of signal analysis. In text mining, they could do semantic analysis. The deep learning agent in the hybrid system must have the property of disregarding quick changes or transients in the input. In other words, it must learn such exemplars that are typical of an established mode. The deep learning agent is, therefore, not very good at determining the transition between modes and this is the reason why it takes more time than an agent that is focused on learning quick changes in the signal. In the biological domain, the deep learning agent is the collection of long-term learning mechanisms.

The surface learning agent needs to be able to learn the changed system function instantaneously or near-instantaneously. The performance of the surface learning agent directly affects the performance of the hybrid system. We describe two candidates for the surface learning agent in detail as follows.

Surface Learning Network

We propose the use of *corner classification* (CC) networks for surface learning because they have the ability of instantaneous learning (Kak, 1996; Kak, 1999). There are four versions of the CC technique, represented by CC1 through CC4. Each node in the network acts as a filter for the training sample. The filter is realized by making it act as a hyperplane to separate the corner of the *n*-dimensional cube represented by the training vector and hence the name corner-classification technique. The CC4 is shown to be better than the other networks in the CC category (Kak, 2002), and is described in detail as follows. CC4 uses a feedforward network architecture consisting of three layers of neurons. The number of input neurons is equal to the length of input patterns or vectors plus one, the additional neuron being the bias neuron, which has a constant input of 1. The number of hidden neurons is equal to the number of training samples, each hidden neuron correspond to one training example. The last node of the input layer is set to one to act as a bias to the hidden layer. The binary step function is used as the activation function for both the hidden and output neurons. The output of the activation function is 1 if summation is positive and zero otherwise.

For each training vector presented to the network, if an input neuron receives a 1, its weight to the hidden neuron corresponding to this training vector is set to I. Otherwise, it is set to -1. The bias neuron is treated differently. If *s* is the number of l's in the training vector, excluding the bias input, and the desired radius of generalization is *r*, then the weight between the bias neuron and the hidden neuron corresponding to this training vector is $r - s + 1$.

The weights in the output layer are equal to 1 if the output value is 1 and –1 if the output value is 0. This amounts to learning both the input class and its complement and thus instantaneous. The radius of generalization, *r* can be seen by considering the all-zero input vectors for which $w_{n+1} = r + 1$. The choice of *r* will depend on the nature of generalization sought. Since the weights are 1, -1, or 0, it is clear that actual computations are minimal. In the general case, the only weight that can be greater in magnitude than 1 is the one associated with the bias neuron. When real data is represented in binary, that should be done using *unary* coding.

Instantaneous and Near-Instantaneous Learning of Non-binary Data

The fast classification (FC) network (Tang and Kak, 2002) is a generalization of the CC networks. If the CC networks perform nearest-neighbor generalization of data as binary vectors, the FC network does the same by considering analog-valued vectors. The FC network uses nearest neighbor generalization. The input data is normalized and presented as input vector $\boldsymbol{x}$**.** The hidden neuron is represented by the weight vector $\boldsymbol{w}_i$ and its elements are represented by $w_{i,j}$,$i = (1,2,\ldots,S)$ and $j = (1,2,\ldots,R)$, where *R* is the number of components of the input vector and *S* is the number of hidden neurons (the number of training samples). The output is the dot product of the vectors $\boldsymbol{\mu}$ and $\mathbf{u}$, where $\boldsymbol{\mu}$ is the vector at the output of the Rule Base and $\mathbf{u}$ is the vector of weights in the output layer as shown in Figure 2**.** This network can be trained with just two passes of the samples, the first pass assigns the synaptic weights and the second pass determines the radius of generalization for each training sample by fuzzification of the location of the each training sample and by assigning fuzzy membership functions of output classes to new input vectors.

The network behaves as a 1NN classifier and a kNN classifier (1- or k- nearest neighbor classifier) according to whether the input vector falls within the radius of generalization of a training vector or not. The radius of generalization acts as a switch between the 1NN classifier and the kNN classifier. The FC network meets the specifications set by traditional function approximation that every data point is covered in the given training sample. In the practical case, the *k* values are determined by the sample size and be a fraction of the sample size. If k=S then the FC network operating as a kNN classifier can be viewed as a RBF network provided the membership function is chosen to be a Gaussian distributed. On the other hand, if the weighting function is chosen to be the membership function, the FC classifier can be considered as kernel regression (Looney, 1997). As in the CC networks, this network requires as many hidden neurons as the number of training samples (although the number of hidden neurons could be trimmed to a certain extent).

Although the CC networks were postulated for binary data, they can also be used for continuous data if quantization of that data is performed using *unary* coding. FC networks require a comparison with stored information in one pass, and, therefore, they are not quite instantaneous, but the calculation can be done in time that could be smaller than the time instant at which the next data comes in.

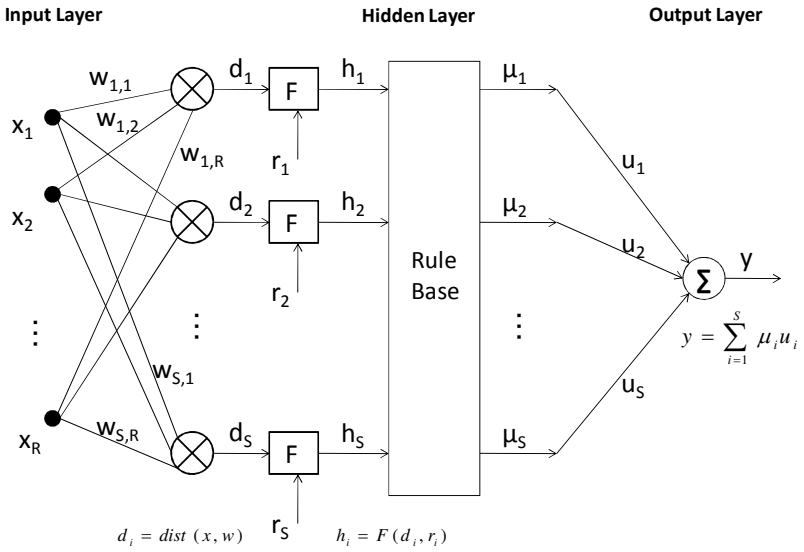

Figure 2. FC network architecture

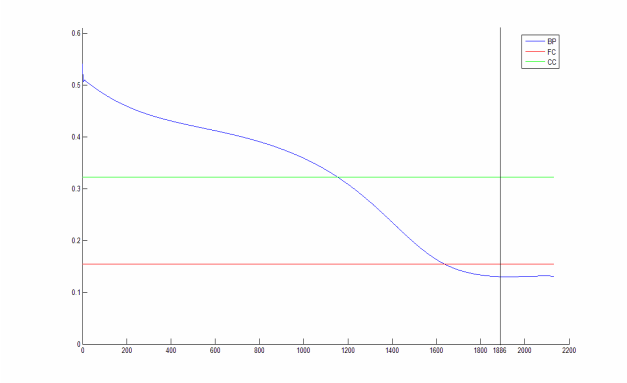

Figure 3. RMS Errors vs. Adaptation cycles

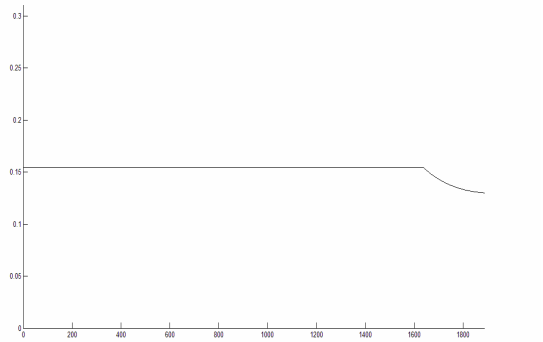

Figure 4. RMS error of the hybrid system

Performance Comparison

We evaluate the performance of the hybrid system for time-varying data, which could be a financial time series, using several benchmarks. The network is trained by historical data set with time index and the network predicts the future values based on past values.

Mackey-Glass Time Series Prediction

Mackey-Glass (MG) time series is a chaotic time series based on Mackey-Class differential equation,

$$\frac{dx(t)}{dt} = -B * x(t) + a * \frac{x(t-D)}{1 + x(t-D)^C},$$

where the choice of *D* has important influence on its chaotic behavior; 17 < D <30. The raw data is from the Working Group on Data Modeling Benchmarks (under IEEE Neural Networks Council Standards Committee). The first 3500 points in the initialization stage are not used for either training or testing. The next 1000 points are used for training and validation, and another 500 points are used for testing. CC and FC are used for the surface learning agent, and back-propagation (BP) network is used for the deep learning agent.

The RMS error is the performance measure of the system response to the relearning process. In Figure 3, we observe that as the candidates for the surface learning agent, CC and FC has reduced RMS error almost instantaneously. Conversely, it takes some time for the deep learning agent BP to converge to new parameters. However, in a long run, the RMS error produced by the deep learning agent BP is smaller than the surface learning agent CC or FC. The vertical line marks the stopping point (at 1886-th adaptation cycle) which is determined by the RMS error validation set so as to avoid system over-fitting. Since FC has smaller RMS error we choose it as the surface learning agent.

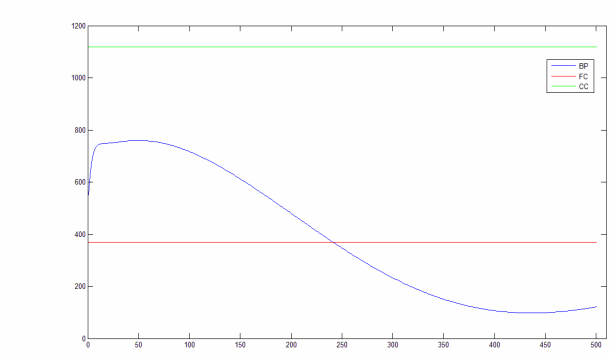

Figure 5. E_1 Errors during the relearning process

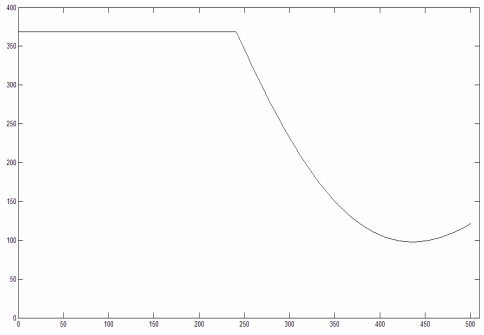

Figure 6. $\mathbf{E_1}$ Error produced by the hybrid system

The cognitive agent selects the output from the surface learning agent FC once it detects that the system function changes, and switches to the deep learning agent BP after it catches up. The RMS error of the hybrid system is shown in Figure 4. It is clear that the hybrid system has superior performance.

CATS Benchmark Test

The background to the CATS benchmark is the Time Series Prediction Competition held in 2004, in which participants were invited to predict 100 missing points from a 5000-point artificial time series. This 100 data points were divided into 5 groups, each located in the position range 981-1000, 1981-2000, 2981-3000, 3981-4000 and 4981-5000, respectively. Participants proposed several prediction methods, linear as well as nonlinear and the winner's strategy indicated that division of the problem into short-term prediction and long-term prediction lead to superior result. Since we are interested in the system response of the hybrid system, we skip the input selection stage and use only short-term prediction. The short-term prediction is done over a window of size W.

After each prediction is made, the oldest point in the window is dropped and the newly predicted point is appended on the other side, becoming the current value to predict the next point. We use validation sets of 10 points (from the known points) to find the best window size, with which we train the final network. We first train the networks to learn points 1-980 and 1001-1980 to predict points 981-1000. Thereafter, based on the trained network, relearning is launched to learn another two segments 3001-3980 and 4001-4980. The system response during the relearning process is thus evaluated. In both cases, 10-point validation sets are utilized, and they are selected to be points 971-980 and 3971-3980, which are adjacent to the to-be-predicted points 981-1000 and 3981-4000, respectively. CC precision is fixed at 256 and window size is selected to be 30 based on the results of validation sets.

The E_1 errors for points 3981-4000 ($\frac{\sum_{t=3981}^{4000}(y_t - \hat{y}_t)^2}{100}$) produced by BP, CC and FC during the relearning process are plotted in Figure 5. The BP result is obtained by averaging the results of 10 different trials. Note that the average stopping point for BP is at the 502th adaptation cycle determined by the validation set. As expected, FC produces good results almost instantly.

BP converges slowly but achieves better results eventually. The large error produced by CC is due to the quantization that has been used in it. Since FC is consistently better than CC, we use FC as the surface learning agent and BP as the deep learning agent for the hybrid system. The hybrid system switches automatically between the surface learning agent FC and the deep learning agent BP in the relearning process based on whose error is lower for the previously predicted values. The E_1 error of the hybrid system is shown in Figure 6.

Smooth Function Approximation

In smooth function approximation we need to implement a function, $F(x)$, which approximates an unknown smooth function, $f(x)$, with the input-output pairs defined in a Euclidian sense.

We use the CC and FC as the potential candidates for the surface learning agent, and BP as the deep learning agent. The hybrid system is first trained to approximate a smooth function with parameter set P1, and then relearns the function as the parameter set changes to P2. The objective smooth function is selected to be the pdf of the Multivariate Normal Distribution, $f_X(x_1, x_2) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))$. The configurations of the two parameter sets are in shown Table 1. The training set is constructed by uniformly sampling 900 points in the 2-D input space, and another 1600 points make up the testing set.

P1	P2
μ={0,0}	μ={-0.2,0.7}
∑={0.8 0.2;0.2 0.1}	∑={0.25 0.3;0.3 1}

Table 1. Smooth function parameter configuration

To understand how the deep learning agent and the surface learning agent in the hybrid system respond to the changed function, we show transitional plots. Figure 7 captures the relearning processes for the deep learning agent BP, and. Figures 8 and 9 show the approximation of the surface learning agent candidates CC and FC after instantaneous training, respectively.

The RMS errors produced by the BP, CC and FC during the relearning processes are plotted in Figure 10. As we can see from this figure, it takes a considerable number of adaptation cycles for the deep learning agent BP to catch up with the surface learning agent CC or FC. Since FC produces smaller RMS errors, we choose FC as the surface learning agent for smooth function approximation.

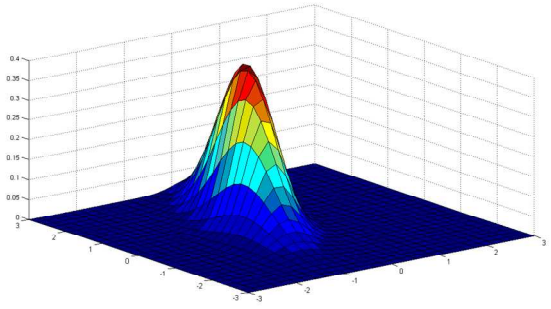

Figure 7. BP shape after 300003 relearning adaptation cycles

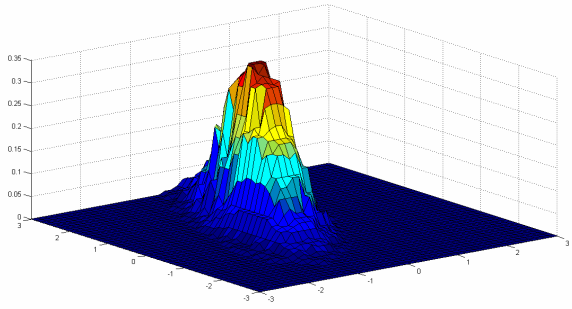

Figure 8. CC Approximation of P2 (instantaneous)

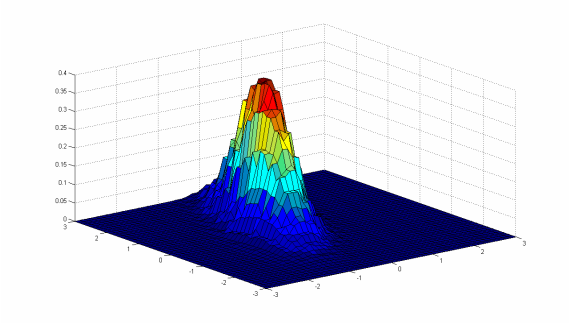

Figure 9. FC approximation of P2 (one cycle)

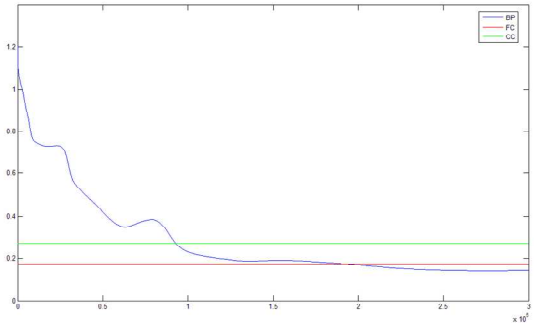

Figure 10. RMS errors vs. Adaptation cycles

CONCLUDING REMARKS

This paper has proposed a hybrid data mining architecture that uses two kinds of neural networks simultaneously: (i) a surface learning network that can quickly adapt to new modes of operation; and, (ii) a deep learning, accurate network for use within a specific regime of operation. This may be viewed as the development of a bicameral learning scheme that is inspired by the dual nature of biological memory, in which one agent learns quickly and the other more slowly, corresponding to short-term and long-term memory.

Such a hybrid architecture can be based on different classes of learning systems although in this paper we have considered CC and FC fast learning networks on the one hand and back-propagation networks on the other. It has been shown that the hybrid architecture provides a superior performance based on the RMS error criterion. In specific data mining application, training samples would be trained with respect to the patterns being sought and then the larger body of data will be checked to see if the same patterns exist in it. The proposed system will allow for the discovery of patterns that change with time.

We expect the applications of our hybrid neural network architecture to problems in finance, control, time-series prediction, economic forecasting, and cybersecurity. This general architecture can also be applied to data mining of text and images.

REFERENCES

1 Aggarwal, C.C., Han, J., Wang, J., Yu, P.S. (2006) A framework for on-demand classification of evolving data streams. *IEEE Trans. on Knowledge and Data Engineering*. 18, 5, 577-589.

2 Kak, S. (1994) New algorithms for training feedforward neural networks. *Pattern Recognition Letters*, 15, 295-298.

3 Kak, S. (1996) Three languages of the brain: quantum, reorganizational, and associative. In *Learning as Self-Organization,* K. Pribram and J. King, eds., Lawrence Erlbaum, Mahwah, N.J., 185--219.

4 Kak, S. (1999) Faster web search and prediction using instantaneously trained neural networks. *IEEE Intelligent Systems*. 14, 79-82, November/December.

5 Kak, S. (2002) A class of instantaneously trained neural networks. *Information Sciences*. 148, 97-102.

6 Looney, C.G. (1997) *Pattern Recognition Using Neural Networks*. Oxford University Press, New York.

7 Salchenberger, L.M., Cinar, E.M., Lash, N.A. (2007) Neural networks: a new tool for predicting thrift failures. *Decision Sciences*. 23, 899-916.

8 Tang, K.W. and Kak, S. (1998) A new corner classification approach to neural network training." *Circuits, Systems, Signal Processing*. 17, 459-469.

9 Tang, K.W. and Kak, S. (2002) Fast classification networks for signal processing. *Circuits, Systems, Signal Processing*. 21, 207-224.

10 Wang, Z., Liu, Y., Liu, X. (2009) State estimation for jumping recurrent neural networks with discrete and distributed delays. *Neural Networks.* 22, 41-48.