P1.1

GOING NONLINEAR: TOWARDS AUTOMATED PUFF INTERCEPT

George S. Young * and Sue Ellen Haupt,
The Pennsylvania State University, University Park, PA

1. INTRODUCTION

Both inadvertent and intentional releases of airborne contaminants pose a threat to society. Managing this threat requires realtime prediction of time evolving contaminant concentration fields from limited observations. These data assimilation algorithms perform best when provided with numerous observations within the advecting contaminant puff (Allen et al. 2006). Yet, the sensor density required to achieve this many "hits" early in the puff's lifetime can be economically impractical if only fixed sensors are used. Thus, solution of the contaminant puff prediction problem requires mobile sensors capable of intercepting and mapping a puff once it has been detected by a sparse array of fixed sensors.

Realtime contaminant puff interception becomes more feasible and less labor intensive if the unmanned aerial vehicles (UAV, eg., Valero et all 1996, Stephens et al. 2000) involved are also autonomous, i.e. able to make their own routing decisions. Designing the routing controller logic can be done either manually, e.g. as an expert system (Hall 1992) or by automated training. The latter offers the potential for finding optimal solutions that might not have been obvious to the designer. Thus, many recent studies on robot routing controllers have focused on automated training, often basing the controller architecture on neural networks (e.g. Kuperstein 1991, Walter and Schulten 1993, Lewis et al., 1995) or related methods (e.g. Selekwa et al., 2005).

This study examines the utility of the neural net approach for UAV routing in the puff sampling problem. Rather than directing an actual UAV in flight, this pilot study examines the suitability of various neural network training approaches in a synthetic, noise-free environment. The study focuses on the initial puff interception problem because, while it is nonlinear, it can be solved either analytically or by iterative optimization. In contrast, the subsequent puff mapping problem has no analytic solution. Thus, while a neural network approach can be applied to both the puff intercept and puff mapping problems, the first can be handled via supervised learning while the second requires unsupervised learning (Reed and Marks 1995). The puff intercept problem can also be solved by unsupervised learning of course. Therefore it will be used as the test problem for this study, allowing comparison of the results of both supervised and unsupervised learning algorithms to the analytic solution.

Neural network training via supervised learning usually makes use of some form of a back-propagation algorithm (Reed and Marks 1995) in which the gradient of error with respect to the network weights is computed and the trial solution is moved down the gradient to minimize the error. This method works well in most applications, although in some situations the trial solution can become trapped in a local minimum of error (Reed and Marks 1995). In contrast, neural net training via unsupervised learning requires that the traditional back-propagation approach be replaced by a method that does not require *a priori* knowledge of the true solution (i.e. desired network output) for each training case. One way to achieve this unsupervised training of a neural network is via a genetic algorithm (GA, Holland 1975, Reed and Marks 1995). Another is the Nelder-Mead simplex algorithm (Nelder and Mead 1965). Both of these algorithms use iterative improvement of a first guess solution. These two approaches differ in several ways although they are conceptually similar. Genetic algorithms work on a population of trial solutions, often updating most or all of them at each iteration (i.e. generation). In contrast, the simplex algorithm creates only one new trial solution each iteration. Another difference is in how the two approaches search the parameter space for lower-error solutions. Genetic algorithms do not explicitly compute the error gradient, but rather breed (i.e. exchange information between) the more successful solutions in the current population to create the next generation of trail solutions (Haupt and Haupt 2004). This approach allows a properly tuned GA to avoid becoming trapped in local minima (Goldberg 1989). It does not, however, make the most efficient use of the error gradient information inherent in the population of trial solutions. The Nelder-Mead simplex algorithm remedies this shortcoming by estimating the error gradient from the most recent trial solutions. It then creates a new trial solution in the down-gradient direction and repeats the error gradient estimation process. This approach is potentially faster at improving the solution, but can become trapped in a local minimum if started too far from the global minimum. Thus, hybrid techniques using a GA to roughly locate the global optimum and simplex to polish the answer have, for some problems, proven more efficient than either method working alone (Long et al. 2007).

There are many other artificial intelligence algorithms that could be applied to unsupervised learning problems of this sort, but the best choice is still an open question. This study will compare a genetic algorithm using various breeding mechanisms with the

* *Corresponding author address*: George S. Young, The Pennsylvania State University, Department of Meteorology, 503 Walker Building, University Park, PA 16802-5013; email: young@meteo.psu.edu

Nelder-Mead simplex. The utility of hybrids and the myriad of other techniques will be left for future work.

2. DATA

The puff intercept problem consists of determining the correct course for a UAV to steer in order to reach the centroid of a moving contaminant puff in the least possible time. For a puff drifting at a uniform speed this is the straight course shown in Figure 1. The inputs to the neural network are the x and y coordinates of the puff and the UAV, the course and speed of the puff, and the speed of the UAV. The output is the course-to-steer for the UAV.

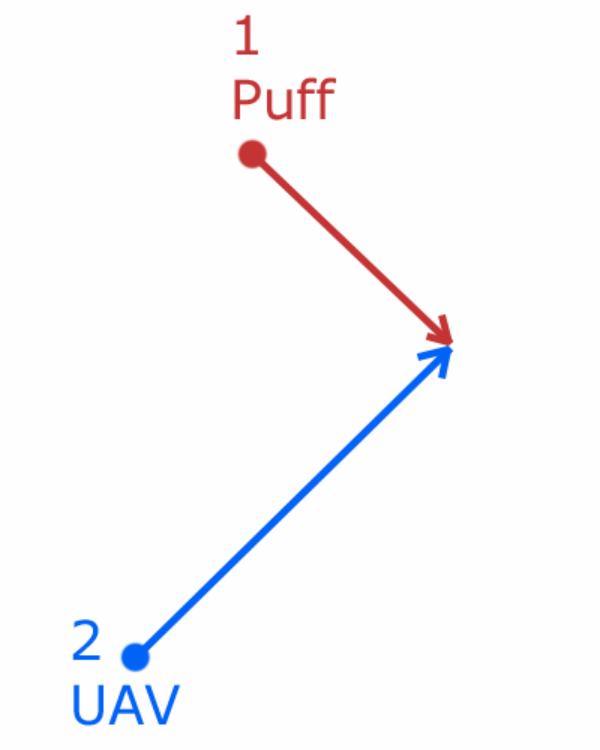

Figure 1. Schematic of the puff intercept problem.

The analytic solution to this problem can be obtained by equating the times required to eliminate the puff-UAV differences in the two components of position. The actual solution is greatly simplified if one first translates the coordinates to place the UAV at the origin and then rotates them so that the puff lies along one of the axes. The following equation for course-to-steer, a2, results.

$$a_2 = \tan^{-1}\left(\frac{u_1 \sin(a_1)}{u_2}, \frac{\left(\left(u_1 \sin(a_1)\right)^2 + \left(u_2\right)^2\right)^{1/2}}{u_2}\right) \qquad (1)$$

where a is the course-to-steer angle for intercept (0 being directly towards the puff's initial location), and u is the speed. Subscript 1 indicates the puff and subscript 2 indicates the UAV.

This analytic solution is used to create the data required to train the neural network. For this study, 1000 training cases were generated, enough to accurately describe the multi-dimensional form of (1). Each training case consisted of randomly generated input values with the UAV always being faster than the puff so that an interception was possible. An operational system would include a component to compute the puff-related inputs from the available sensor "hits" within the evolving three-dimensional concentration field. Long et al. (2006) demonstrate a GA approach to building this component. Future work will need to address the sensitivity of both the centroid tracking and intercept components to noise in the observations.

3. PROCEDURES

The standard feed-forward neural network architecture (Figure 2) is used here. In this architecture the value of each input parameter is fed to each node in the first processing layer. The output value for each of these processing nodes is then fed as input to all of the nodes in the next processing layer, and so on through any remaining processing layers. The output value from each node in the last processing layer is fed into an output node that produces the final answer.

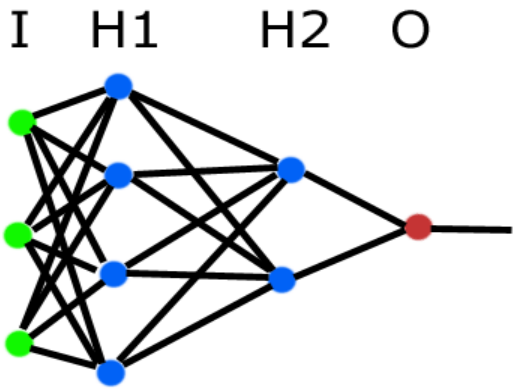

Figure 2. Feed-forward neural network architecture. The input parameters are shown as green circles labeled I, the two hidden layers of processing nodes are shown as blue circles labeled H1 and H2. The output node is shown as a red circle labeled O. Data flow follows the black lines from left to right.

The processing nodes contain linear regression equations which have been squashed by a sigmoid function so that the output tends asymptotically to the limits of some range (typically 0 to 1 or -1 to 1) as the inputs tend to plus or minus infinity. Equation 2 shows the formula for a processing node using hyperbolic tangent as the squashing function.

$$y = \tanh\left(\sum_{i=1}^{M} w_i x_i + c\right) \quad (2)$$

where y is the course-to-steer for intercept, x_i are the M input parameters, w_i are the corresponding weights, and c is a threshold or offset. The output node is similar, but does not include the squashing function if the output is a real number as is the case in the puff interception problem.

Training of a feed-forward neural network consists of finding a set of values for these weights and thresholds that minimizes the error in the output. This task can be approached either as a supervised learning problem in which values of the correct output are available for each training case or as an unsupervised learning problem in which these values are not available but the quality of the answer can be evaluated by the network's performance in achieving an intercept.

The standard back-propagation method of neural net training is a supervised learning technique (Reed and Marks 1995). The network is initialized with random values for the weights and thresholds, and then applied to the training data. The errors are computed by comparing the network output to the known solution values for each case, and the impact of each weight and threshold on that error is calculated. The weights and thresholds are then changed in a direction designed to decrease the error, making this a classic down-gradient optimization technique. The run, test, tune, cycle is repeated until convergence occurs or network performance reaches a satisfactory level. Implementation details and refinements can be found in Reed and Marks (1995) and Witten and Frank (2005). The back-propagation implementation used here is part of the freely available data mining software package Weka (http://www.cs.waikato.ac.nz/~ml/weka/). In all of the runs described below, 500 epochs (i.e. iterations) were used.

In contrast, genetic algorithms offer a means of implementing either supervised or unsupervised training of a neural network. As with back-propagation, many variants exist, but all follow roughly the approach used here:

1. A population of trial solutions is created
2. Each solution is evaluated on the training cases
3. The better solutions are bred to create new members for the population
4. Less successful solutions are eliminated from the population
5. Remaining solutions are mutated to explore more of the solution space
6. Steps 2 through 5 are repeated until convergence or until satisfactory network performance is achieved.

For the puff interception problem, each trial solution consists of a list of real-numbered values, one for each of the weight and threshold parameters in the network. The GA used here uses rank-proportional breeding (Haupt and Haupt 2004) where a trial solution's chances of breeding depend on its rank in the sorted list of mean squared error in output, i.e. the UAV course-to-steer. Breeding is by one of three standard techniques:

- Blending (Eshelman and Schaffer 1993), in which offspring are a weighted average of the parents, with the weight randomly selected so that the offspring can lie anywhere along the line between the two parents or the extension of that line in either direction by one-half the distance between the parents. This extension prevents the breeding mechanism from reducing the variance of the solutions and thus forcing premature convergence of the population of trial solutions.
- Uniform crossover (Haupt and Haupt 2004) in which parameter values are taken at random from one or the other of the parents.
- Single-point crossover (Haupt and Haupt 2004) in which all parameters before a randomly selected element come from one parent and the remainder comes from the other.

The mutation mechanism used here is simply the addition of a small amount of Gaussian noise to each parameter value. Many other mechanisms are discussed in the literature (Haupt and Haupt 2004). For this implementation, the noise amplitude decays to zero over the course of the iterations to prevent jostling of the converged solution.

The number of trial solutions and generations (i.e. iterations) can be varied depending on the needs of the problem. For the results presented here 100 trial solutions and 200 generations were used. Thus, each GA run evaluated 20,000 trial networks compared with 500 for the back-propagation runs.

The success of each neural network training approach will be evaluated by the number of network evaluations required, the correlation between the true course-to-steer and the resulting neural network output, and the Mean Absolute Error (MAE) of this course-to-steer output. Linear regression will serve as a baseline for these comparisons, not because it is expected to work well on this non-linear problem, but because its failure demonstrates just how non-linear this problem is.

4. RESULTS

Table 1 shows the results obtained without the coordinate transformation described in Section 2. Linear regression is clearly inadequate for the task of selecting a course-to-steer for puff interception, with a correlation coefficient of less then 0.65 and an MAE of more than a radian. Considering that the maximum possible error is π radians, this is not good. The simplest neural network, with a single hidden layer of seven processing nodes yields results that are only slightly better than that for linear regression. This suggests that more complex neural network architectures are required to solve this problem. Indeed, the results improve when a second hidden layer of three processing nodes is added. The improvement is more modest, however, when the network complexity is improved still further to include three hidden layers with seven, five, and three processing nodes. Thus, the

7,3 architecture appears to reach the skill plateau for this problem.

Table 1. Success statistics for linear regression and three neural network architectures trained by back-propagation. All four methods are tested with 10-fold cross validation.

Method	Correlation Coefficient	MAE
Linear Regression	0.6468	1.0796
Neural Network 7	0.6998	1.0502
Neural Network 7,3	0.7974	0.6744
Neural Network 7,5,3	0.8053	0.6631

While it is clear from the results above that tuning network complexity can improve the results, even the best network design did not perform particularly well. The challenge, as is so often the case in artificial intelligence problems involving angles, is the discontinuity that occurs in the definition of course-to-steer as the solution changes from just above 0 to just below 2π. Transforming the coordinate system as described in Section 2 eliminates this discontinuity from the problem because an intercept never takes place in the half-circle away from the UAV-to-puff vector so long as the UAV is faster than the puff.

The results shown in Table 2 illustrate the impact of this coordinate transformation on the success of the neural network.

Table 2. Success statistics for linear regression and three neural network architectures trained by back-propagation. All four methods are tested with 10-fold cross-validation. Raw correlations (CC) are taken from Table 1, while those labeled as rotated were obtained using training data that had undergone the coordinate transformation. The final column shows the percent improvement in the correlation coefficient that resulted from this coordinate transformation.

Method	Raw CC	Rotated CC	%
Linear Regression	0.6468	0.6697	3.5
Neural Network 7	0.6998	0.9679	37.5
Neural Network 7,3	0.7974	0.9917	24.4
Neural Network 7,5,3	0.8053	0.9908	23.0

The coordinate transformation yielded only a modest improvement in the linear regression results, as expected given the non-linearity of the problem to be solved. In contrast it resulted in near perfect solutions for the two best neural network architectures. Thus, the 7,3 network is sufficient for high accuracy emulation of the analytic solution to the puff interception problem, but only if the coordinate system is shifted to take the 0-2π discontinuity out of play.

Having demonstrated that a 7,3 neural network is sufficient to solve the puff interception problem, it will be used as benchmark architecture to explore the ability of GA and Nelder-Mead simplex methods in unsupervised training. Table 3 presents the results for a range of neural network training techniques. The correlation coefficient between neural net output and true course-to-steer is computed here on the training data rather than using 10-fold cross-validation as was done for Table 2. The correlation values for these two tests are very similar for the GA trained via back-propagation, 0.9943 versus 0.9917. The same comparison for linear regression was 0.6717 versus 0.6697, suggesting that the similarity of skill between developmental and independent data is not a result of near perfection of the neural network results, but rather of having sufficient training data available to avoid overfitting the neural network. Thus, the true skill on independent data is probably only slightly less than the values shown in Table 3.

Each GA or Nelder-Mead simplex result is the best of a three-run set. The Nelder-Mead simplex algorithm is run from a randomly selected trial solution, rather than from a GA-produced best-guess solution. For the GA-trained neural networks the skill varied little from run to run, probably because the large population of trial solutions made the final result independent of the randomly selected starting points. In contrast, the Nelder-Mead-simplex-trained neural-networks exhibited large run-to-run variations in performance. Therefore these three-run experiments were themselves repeated four times and the winners from the best and worst three-run sets presented in Table 3. The run-to-run variability in the Nelder-Mead simplex results is probably due to its use of a single first guess rather than a 100-member population as in the GA.

Table 3. Success statistics for various methods of training the 7,3 neural network on the coordinate transformed puff interception problem. All methods are tested on the developmental data.

Method	Correlation Coefficient
Back-propagation	0.9943
GA / blending	0.7013
GA / uniform crossover	0.6674
GA / single-point crossover	0.6618
Nelder-Mead simplex (best of 3)	0.8623
Nelder-Mead simplex (worst of 3)	0.7989

The wide difference in correlation values shown in Table 3 demonstrates that the choice of training method matters greatly in the success of a neural network at solving the puff interception problem. Back-propagation achieves near perfect results with only 500 trial solutions while the various genetic algorithms yield results that are not much better than linear regression. Of the three breeding techniques, blending appears to be the best for this problem. Nelder-Mead simplex training yields somewhat better skill than was achieved with genetic algorithm training, but the results varied greatly from run to run, depending on the randomly selected starting point.

The obvious result from Table 3 is that those neural network training methods suitable for unsupervised

learning are doing worse on the puff interception problem than is back-propagation. This raises the question of whether similarly poor performance can be expected when applying unsupervised learning to other non-linear problems, such as routing a UAV during the mapping of a previously intercepted puff. Table 4 presents the results of a preliminary examination of this issue. The training data for the analyses above is used to compute two new non-linear functions to serve as predictands, thereby replacing the puff interception problem with two different ad hoc non-linear problems taking the same inputs.

$$a_2 = x_1 + \left(u_1 u_2\right) + \left(a_1\right)^{0.3} + 3.2 \qquad (2)$$

$$a_2 = x_1 + \frac{\left(u_1 u_2\right)}{\left(a_1\right)^{0.3}} + 3.2 \qquad (3)$$

where x is the initial puff position in the rotated coordinate system where both the UAF and the Puff lie on the x axis, and the other variables are defined as in equation 1.

Table 4. Success statistics for various methods of unsupervised training the 7,3 neural network on two alternative non-linear problems All methods are tested on the training data. As in Table 3, correlation coefficient (CC) is used to measure success.

Method	Eqn (2) - CC	Eqn (3) - CC
GA / blending	0.9760	0.9967
Nelder-Mead Simplex	0.9969	0.9978

The results shown in Table 4 indicate that the ability of the GA and Nelder-Mead simplex is highly problem dependent. While both did poorly on the puff interception problem, both did well on (2) and (3) with the GA and with the Nelder-Mead simplex method. Thus, there must be something about equation 1 that makes it much more challenging for these two training methods than are equations 2 and 3. Given the wide range of results obtained in training a neural network to solve these three non-linear problems, it is apparent that the puff mapping problem could lie anywhere in the range from easy to challenging for the methods tested here.

If the puff mapping problem turns out to be challenging, as indicated by our initial calculations, there are two possible routes to its solution. First, reformulation of the cost function to be minimized by the GA and Nelder-Mead simplex may solve this problem (Long et al. 2007). Second, one could try any of several other unsupervised learning techniques that have been developed for network-based robot controllers (e.g. Guadiano et al., 1996).

5. CONCLUSIONS

A simple neural network with two hidden layers is shown to be able to emulate the analytic solution for the contaminant puff interception problem. This network architecture performs equally well on two other non-linear problems taking the same set of inputs. Supervised learning via back-propagation works well on all three problems. In contrast, unsupervised learning via genetic algorithm or Nelder-Mead simplex works well on the two alternative problems but not on the puff interception problem. Further work will be required to determine if training a neural network UAV controller for the post-interception puff mapping problem can be handled by one of these unsupervised learning methods or will require different techniques.

Success and efficiency vary considerably between methods. Both of the unsupervised learning techniques require at least an order of magnitude more trial solutions to train the neural network than does supervised back-propagation. On the "hard" puff interception problem discussed at the beginning of section 4, Nelder-Mead simplex yields results midway between those for back-propagation and the genetic algorithm. On the "easier" alternative problems ((2) and (3)) the Nelder-Mead simplex method performs quite well with the genetic algorithm close behind. For the "hard problem," genetic algorithm results vary somewhat with the breeding techniques used. Blending yields the best results. For all of the neural network training methods tested, results on the puff interception problem were markedly worse until a coordinate transformation was used to remove the discontinuity in the function to be emulated.

Future work should focus on development of a network-based UAV controller for the post-interception puff mapping problem. Questions to be answered include:

- Is the mapping problem GA-hard or GA-easy?
- Is the Nelder-Mead simplex method a viable alternative?
- If neither of these methods perform well, do others taken from the robot navigation literature perform better?

Once a suitable unsupervised training method has been developed, the resulting network should be coupled to a puff-centroid tracking algorithm to yield a control system capable of using the UAV's own observations to guide its routing decisions.

ACKNOWLEDGEMENTS

This work was supported by DTRA project number W911NF-06-C-0162.

REFERENCES

Allen, C. T., S. E. Haupt, and G. S. Young, 2006: Source characterization with a receptor-dispersion model coupled with a genetic algorithm, In press, *J. Appl. Meteor. Climate.*

Eshelman, L. J. & J.D. Schaffer, 1993: Real-coded Genetic Algorithms and Interval-Schemata, in *Proc. FOGA '93,* 187-202.

Gaudiano, P., E. Zalama, and J. L. Coronado, 1996: An unsupervised neural network for low-level control of a wheeled mobile robot: noise resistance, stability, and hardware implementation. *IEEE Trans. Systems, Man, Cybernetics*, 26, 485-496.

Goldberg, D.E., 1989: *Genetic Algorithms in Search, Optimization, and Machine Learning*, New York: Addison-Wesley.

Hall, D. L., 1992: *Mathematical Techniques in Multisensor Data Fusion*, Artech House, Norwood, MA, 301 pp.

Haupt, R. L. and S. E. Haupt, 2004: *Practical Genetic Algorithms, 2nd edition with CD.* John Wiley & Sons, New York, NY, 255 pp.

Haupt, S. E., G. S. Young, and C. T. Allen, 2006: Validation of a receptor/dispersion model coupled with a genetic algorithm using synthetic data, *J. Appl. Meteor., 45, 476-490.*

Holland, J.H., 1975: *Adaptation in Natural and Artificial Systems,* Ann Arbor: The University of Michigan Press.

Kuperstein, M, 1991: INFANT neural controller for adaptive sensory-motor coordination, *Neural Networks*, **4**, 131–145.

Long, K. J., C. T. Allen, S. E. Haupt, and G. S. Young, 2007: Characterizing contaminant source and meteorological forcing using data assimilation with a genetic algorithm. Extended Abstract, *Fifth Conference on Artificial Intelligence and its Applications to Environmental Sciences*, San Antonio, TX, Amer. Meteor. Soc.

Nelder, J. A., and R. Mead, 1965: A simplex method for function minimization, *Computer Journal*, **7**, 308-313.

Reed, R. D. and R. J. Marks, 1995: *Neural Smithing – Supervised Learning in Feed Forward Artificial Neural Networks*, MIT Press, Cambridge, MA. 346 pp.

Selekwa, M. F., D. D. Dunlap, and E. G. Collins, Jr., 2005: Implementation of multi-valued fuzzy behavior control for robot navigation in cluttered environments, in Proc. IEEE Conf. on Robotics and Automation, Barcelona, Spain, 3699-3706.

Stephens, G. L., R. G. Ellingson, J. Vitko Jr., W. Bolton, T. P. Tooman, F. P. J. Valero, P. Minnis, P. Pilewskie, G. S. Phipps, S. Sekelsky, J. R. Carswell, S. D. Miller, A. Benedetti, R. B. McCoy, R. F. McCoy Jr., A. Lederbuhr, and R. Bambha, 2000: The Department of Energy's Atmospheric Radiation Measurement (ARM) Unmanned Aerospace Vehicle (UAV) Program, *Bull. Amer. Meteor. Soc.*, **81**, 2915–2938.

Valero, F. P. J., S. K. Pope, R. G. Ellingson, A. W. Strawa, and J. Vitko Jr., 1996: Determination of clear-sky radiative flux profiles, heating rates, and optical depths using unmanned aerospace vehicles as a platform, *J. Atmos. and Ocean. Tech*., 13, 1024-1030.

Witten, I. H. and E. Frank, 2005: *Data Mining: Practical machine learning tools and techniques, 2nd Edition*, Morgan Kaufmann, San Francisco, CA, 525 pp.

Walter, J. A., and K. Schulten, 1993: Implementation of a self-organizing neural network for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks*, **4**, 86–95, 1993.

Lewis, E. L., K. Liu. and A. Yesildirek, 1995: Neural Net Robot Controller with Guaranteed Tracking Performance, *IEEE Trans. on Neural Networks*, **6**, 703-715.