DYNAMIC PATTERN RECOGNITION FOR TEMPORAL DATA[1]

Sameer Singh

School of Computing
University of Plymouth
Plymouth PL4 8AA
United Kingdom

tel: +44-1752-232 612
fax: +44-1752-232 540
e-mail: s1singh@plym.ac.uk

ABSTRACT

The main aim of this paper is to demonstrate the performance of a possibility based classifier which implements dynamic pattern recognition. The objectives of the paper are: 1) to detail the working of a Massively Parallel Fuzzy System (MPFS) which can be used to classify two spiral data; 2) to develop and detail the dynamic pattern recognition approach; and 3) to discuss the results obtained. The results suggest that dynamic pattern recognition will perform high quality decision making for tasks involving uncertain, noisy and small amounts of data. The two spiral task can be solved with dynamic pattern recognition with more than 90% recognition success and this performance remains stable with increasingly noisy test sets.

INTRODUCTION

Spiral data has several interesting characteristics for validating novel neural and other pattern recognition algorithms as for example: 1) it is highly non-linear and with linear techniques a poor recognition rate is achieved; 2) it is temporal in the sense that x and y coordinates of successive points on the spiral are a function of time, thus purely gaussian techniques for classification achieve limited success; 3) validating results on spiral data can be difficult since validation procedures such as cross-validation [18] are not well developed for temporal data [1]; and 4) it is possible to test the reliability of a classifier working on such data by exhaustively testing it on spirals of varying radii σ. The two spiral task is one of many in a class of spirals in n dimension. Learning to tell two spirals apart is important both for purely academic reasons [16] and for industrial applications [2, 6]. In this problem, two different class data lies on two distinct spirals that coils around each other and the origin. Neural networks using backpropagation and its relatives encounter significant problems with the spiral problem [16]. Neural solutions have been found to be too slow and are only suited to off-line training [4, 5, 9]. For this reason, they are not well suited for real-time spiral and other temporal data analysis. In this vein, recognising the importance of training and classifying data in real-time, previous attempts on the spiral task include minimal configuration neural nets [2], input data encoding methods [3, 7], fast learning ANNs [15], hypercube separation algorithm [17] and neuro-fuzzy methods [14]. These attempts have achieved a limited amount of success although it is difficult to comment how reliable these methods are in relation to each other since their comparison is not well documented.

In this paper, the two spiral classification task is approached with the dynamic pattern recognition approach. In order to implement this approach, only classifiers training in real-time qualify as computationally cheap and viable. The classifier used here is a fuzzy possibility based system. The classifier is a parallel processing architecture implementing an algorithm for testing new data on the basis of available training data. The possibility calculations rely on gaussian methods [10, 19] or through the determination of nearest neighbours of test data in training data. The latter technique is well suited for the spiral problem and it will be described later in this paper. First, the concept of dynamic pattern recognition is explained.

---

[1] Singh, S. "Dynamic Pattern Recognition for Temporal Data", Proc. 5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97), Aachen, Germany. Vol 3 pp.1993-1997 (8-12 September , 1997)

DYNAMIC PATTERN RECOGNITION

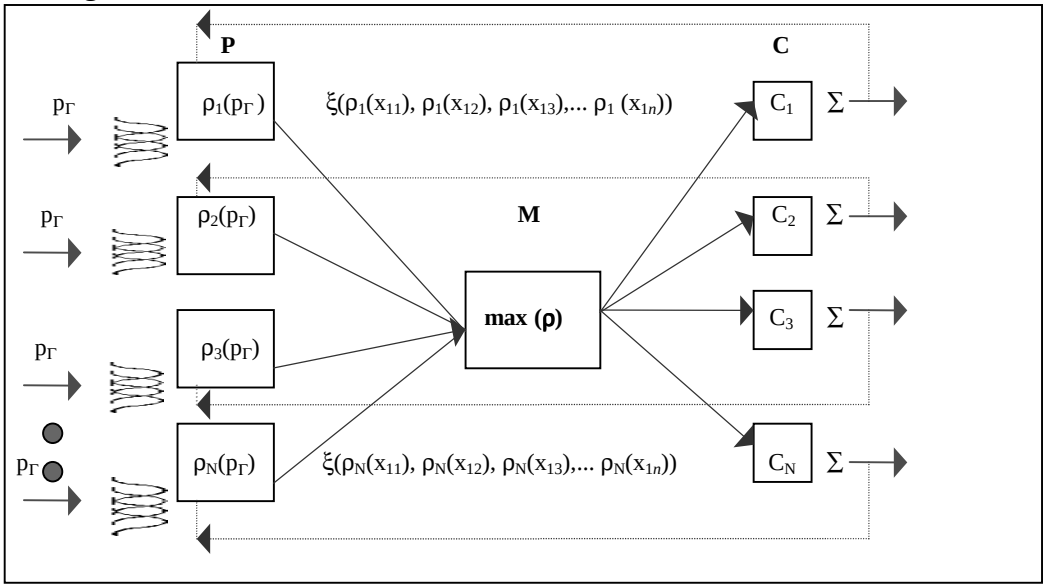
The traditional approach to pattern classification consists of three stages: training, validation and testing [1, 8]. The validation of classifier performance ensures that the test results are based on the estimate of classifier performance on data population than the sample only [18]. This approach is particularly useful for neural networks and statistical classifiers such as the discriminant analysis. However, using traditional validation approaches on temporal data may give misleading results and further work needs to be done [1]. Traditionally the training set is a non-changing entity over the duration of train-test cycle. The logic is simple: *make decisions on what you know at the start of the test trial*. However, in reality, our state of knowledge is dynamically increasing as we test new patterns. Logically speaking, better decisions could be made on what information is available at time *t* rather than what was available at the start of the trial at *t*=0. In other words, correct test classifications may be added dynamically at time *t* to the training set for making decisions at time *t*+1. This implies a new dynamic training procedure: at time t if the test pattern is correctly classified with reasonable confidence in class C, then add this test pattern to the training data of C for training at time t + 1 and testing the next pattern; if a wrong prediction is made, the training set remains unchanged.

The above approach can be useful in situations where training data is limited. Temporal test data in several manufacturing and intelligent control applications can be dynamically used to increment the training set. Users must be however cautious on one account. In many circumstances where the test target is unknown or if there is limited belief in the quality of decision making, including test misclassifications in the training set may impair future decision making. There is however a counter argument that if the majority of the training patterns produce accurate decision boundaries, then the system may remain fault tolerant to the inclusion of a few poor patterns. In practice if there exist mechanisms to ensure that only correct classifications are added to the training set at time t, then the dynamic approach has three important benefits: 1) decision making with a larger input space; 2) dynamic updating of information which is particularly useful for recognising temporal data; and 3) suitability to on-line applications where the starting training set T_0 at time t =0 contains a small number of patterns. Implementing this pattern recognition approach implies that the size of the training set is an implicit function of time, i.e. T = *f*(t). Neural iterative approaches would be computationally very expensive for this purpose since for a total of N test patterns, the training must take place as many times as correct test classifications. For the proposed approach, a dynamic classifier is needed which trains in real-time and is computationally cheap. One such system is detailed below.

MPFS - A POSSIBILITY BASED DYNAMIC CLASSIFIER

A Massively Parallel Fuzzy System is a parallel classifier operating in real-time. It is non-iterative, computationally cheap and well suited for both temporal and non-temporal data. The system is shown in Fig. 1.

Fig. 1. The Massively Parallel Fuzzy System (MPFS) architecture for dynamic pattern recognition.



The overall system consists of three major components: a set of processors P which are responsible for calculating the possibility that a particular test pattern belongs to class k (*k* = 1... N), a decision making module M which

takes the output from individual processors and assigns the test pattern to one of the k classes, and a set of counters C which monitor the total number of correctly predicted patterns. The input data is presented to the processors sequentially and at any time *t*, each processor receives the same test pattern p_Γ for analysis. The processor P_k computes the possibility that the test pattern p_Γ belongs to class k. This possibility is the combination of possibilities that test values in the pattern p_Γ belong to a given class. The MPFS system is restricted to use the same possibility calculation method across all processors during input presentation. For parametric data, possibilities may be calculated using a quadratic function [10, 19]. For temporal data, other procedures may be used such as the nearest neighbour algorithm. In Fig. 1, feedback loops from outputs to processors represent the process of dynamic recognition, i.e. correct test classifications are added to the training set held by processors for making decisions at the next time instant. The overall system is: parallel, since different processors work with their own training data partition, i.e. processor P_k contains training data for only class *k*, and fuzzy, since test data proximity to training data of different classes may be estimated using possibilistic methods. With a large number of classes, the number of processors grows linearly but since different processors hold different partitions of the training data, the system is fast and computationally cheap. The MPFS system has been previously tested successfully on applications involving manufacturing assembly data [11] and electronic nose data [12-13].

EXPERIMENTAL DETAILS

The spiral data has three important parameters: density of the spiral φ, radius σ and offset δ used to displace the validation and test sets. The original benchmark was proposed with $\phi = 1$, $\sigma = 6.5$ and $\delta = 0.1$ (ref: Carnegie Mellon AI repository). A variety of spirals may be created by varying these parameters. The original benchmark consists of 194 training patterns, 97 each for class C_1 and C_2. The validation and the test set represent a noisy spiral, i.e. if training data is {**x**, **y**} then the validation set may be represented as {**x**+δ, **y**} and the test set as {**x**, **y**+δ} where δ is a pre-defined scalar representing constant noise. For a given radius, an increase in δ makes data recognition more difficult and it is important for robust classifiers to demonstrate a smooth declination in performance than an irregular one with increasing offsets. The proposed classifier consists of a Massively Parallel Fuzzy System whose processors use the following algorithm for calculating the possibility that a given test pattern belongs to different classes. The main function of the algorithm is to take a given test pattern $p_\Gamma = (X_\Gamma, Y_\Gamma)$ and determine the possibility that X_Γ belongs to class C_1 and to do the same for Y_Γ. The overall possibility that p_Γ belongs to C_1 is the combination of possibilities that X_Γ and Y_Γ belong to C_1. The same applies for C_2. The test pattern will be assigned the class for which the combined possibility is the highest. The possibility calculation will be based on the proximity between a given test value and its nearest upper or lower bound in the training data. The procedure may be formally described as:

❶ Label the data as for training set **T** and test set **Γ**. The benchmark consists of 194 patterns in each of these sets.
❷ Separate class C_1 data and class C_2 training data in two training files f1 and f2 respectively.
❸ For every pattern $p_\Gamma = (X_\Gamma, Y_\Gamma)$ in the test set Γ, perform the following steps:
❹ Find the lower and upper bounds of X_Γ for class C_1 from f1. These may be represented as $X_{T(lb1)}$ and $X_{T(ub1)}$. The upper bound of a test value is the next largest value in the training set and the lower bound is the next smallest value in the training set. If $X_\Gamma > X_T$ for all $X_T \in$ f1, then X_Γ has only a lower bound and if $X_\Gamma < X_T$ for all $X_T \in$ f1, then X_Γ has only an upper bound. Similarly we can determine $Y_{T(lb1)}$ and $Y_{T(ub1)}$.
❺ For f1, calculate the possibility that the given test pattern belongs to class C_1.

$\rho_1(X_\Gamma) = 1.0/\sqrt{1.0 + \eta_1}$ if $\eta_1 < \eta_2$, else $\rho_1(X_\Gamma) = 1.0/\sqrt{1.0 + \eta_2}$
where $\eta_1 = |X_\Gamma - X_{T(lb1)}|$ and $\eta_2 = |X_{T(ub1)} - X_\Gamma|$

$\rho_1(Y_\Gamma) = 1.0/\sqrt{1.0 + \eta_3}$ if $\eta_3 < \eta_4$, else $\rho_1(Y_\Gamma) = 1.0/\sqrt{1.0 + \eta_4}$ if $\eta_4 < \eta_3$
where $\eta_3 = |Y_\Gamma - Y_{T(lb1)}|$ and $\eta_4 = |Y_{T(ub1)} - Y_\Gamma|$

❻ Perform steps 4 & 5 on f2 to calculate $\rho_2(X_\Gamma)$ and $\rho_2(Y_\Gamma)$.
❼ Allocate target class based on an optimal function ξ for the following:
If $\xi(\rho_1(X_\Gamma), \rho_1(Y_\Gamma)) > \xi(\rho_2(X_\Gamma), \rho_2(Y_\Gamma))$ then $p_\Gamma \in C_1$, else $p_\Gamma \in C_2$.
❽ Determine the recognition rate through correctly classified test patterns.

In the above description, $\rho_1(X)$ denotes the possibility that the test datum X belongs to class C_1, ξ is a possibility combination function which is either a min-max or product operator in most cases, and $|X|$ represents the positive magnitude of X. It may be seen in the above procedure that the overall possibility of a test pattern belonging to class C_i is the combination of possibilities that feature measurements in that pattern belong to C_i. For a *n* feature test pattern, $p_\Gamma = (x_{1\Gamma}, x_{2\Gamma} \ldots x_{n\Gamma})$ and N classes $C_1 \ldots C_N$, the possibility that p_Γ belongs to class C_i is a combination of the possibilities that individual feature measurements $x_{1\Gamma}, x_{2\Gamma} \ldots x_{n\Gamma}$ belong to C_i. The possibility that $x_{j\Gamma}$ belongs to class C_i is based on the

proximity of $x_{j\Gamma}$ and its nearest bound in the training data for C_i; the nearest bound is the upper or the lower bound which ever is the nearest. The nearest bound of different $x_{j\Gamma}$ in the same test pattern are independent of each other, i.e. not all $x_{j\Gamma}$ in a test pattern will find their nearest bounds which come from the same training pattern. In most cases however all $x_{j\Gamma}$ in a test pattern will find their nearest bounds come from the same training pattern and in such cases we have found a single nearest neighbour training pattern of the test pattern. In either case the algorithm can continue to predict correctly even though we are more likely to be confident of our decision if we do manage to find a single nearest neighbour training pattern. In practice the desirability of finding nearest neighbour training patterns for making decisions is application dependent; for the spiral data it is a desirable property of the classifier.

RESULTS

The MPFS system was validated with the spiral validation set using the above procedure and good results were obtained. In the following section only test results are discussed. The dynamic pattern recognition approach may be followed in two ways: to start with a minimal training set and gradually increment it with test data, or to start with a reasonable amount of initial training data and then increment it with correctly classified test data. The second approach is reasonable with sufficient training data at time t=0. The experiment was undertaken with the initial training data of the original benchmark (density $\varphi = 1$; radius $\sigma = 6.5$). Here the radius of the spiral represents the maximum radius, which defines the two dimensional envelope of the spiral. Spiral density determines the total number of points generated. The test data for this spiral was dynamically tested, and then simultaneous test sets at time t+1 with increasing offsets were tested with training data in its state at time t. The results are shown in Table 1. Here a total of eight test sets with varying offsets have been tested. The offset represents constant noise in the test set. $Base_1$ and $base_2$ represent the number of patterns in the training set at the beginning of the train→test→ modify-training-set procedure for a particular offset δ test data, i.e. the size of the training files f1 and f2. Here, c_1 and c_2 represent the total number of correct classifications for class C_1 and C_2 respectively at the end of testing set Γ. The success of the classification process is represented by the recognition rate in percentage, $\Re$, which shows the overall proportion of correctly classified patterns.

Table 1. Recognition rate as a function of the dynamically increasing training set (radius $\sigma = 6.5$; offset $.1 \leq \delta \leq 1.5$).

Offset δ	$base_1$	$base_2$	c_1	c_2	$\Re$
*0.1**	97	97	92	87	92.2
0.3	189	184	88	90	91.7
0.5	277	274	87	82	87.1
0.7	364	356	90	75	85.0
0.9	448	431	84	79	84.0
1.1	532	510	89	80	87.1
1.3	619	590	87	80	88.6
1.5	706	670	88	81	89.1

In Table 1, $\Re$ represents the recognition rate achieved with the dynamic recognition approach. It may be seen that the recognition rate begins to fall at around $\delta = .9$ and then starts to rise again. This is because of the nature of the spiral data and this change can be understood by plotting a test set with $\delta = .9$ and overlapping it on the training data. It was found that at an offset of $\delta = 1.0$ the test points are most vulnerable to misclassification because the distance between two adjacent spirals of different classes is also nearly 1.0. For an offset of 1.0 a low recognition rate of 76.8% was achieved. This is still a good recognition rate because in the test set only the y dimension is affected and most misclassifications are localised in the upper and lower halves of the spiral; the left and right halves are not affected by the same amount.

So does the proposed classifier detect a single nearest training neighbour of a test pattern for class allocation ? Since the **x** values of the test set are not different from training data, determination of the upper and lower bounds is straight forward. If the test pattern is $p_\Gamma = (X_\Gamma, Y_\Gamma)$ then it is possible to find a training pattern $\{x_j, y_j\}$ where $X_\Gamma = x_j$ for at least one of the classes. In this case the nearest bound of X is X itself. However it is not necessary that the nearest bound of Y_Γ to be y_j. In case y_j is found to be the nearest bound then we have detected the training pattern $\{x_j, y_j\}$ to be the nearest neighbour of p_Γ . In the above results, on average 90% of the correct classifications with the described procedure are actually based on detecting a single nearest neighbour training pattern for a given test pattern. In the remaining cases correct classifications are made simply because Y_Γ is found to be more likely to belong to its target class. This weakness can be minimised by calculating the possibility of Y_Γ belongs to a given class by determining the distance between Y_Γ and y_j rather than allowing Y_Γ to find its own nearest bound. This modification may give superior results for the given

spiral data but will be weaker for other data sets and spiral data when both **x** and **y** test values are contaminated. Hence the proposed technique is recommended for use in its present form.

CONCLUSION

In this paper, initial test results of the dynamic approach to spiral pattern classification have been reported. The proposed classifier works in real-time and is able to test all 194 test patterns in less than 2 seconds on a Pentium machine in the static mode; the time taken in the dynamic mode depends on the size of the training set. The study confirms that strategies for pattern recognition which do not determine data density prior to testing can be viable for processing highly non-linear data in real-time. Their role is particularly important for dynamic pattern recognition which may in practice give good results. Dynamic pattern recognition is relevant to on-line data processing where a reasonable amount of belief may be placed on the correctness of test predictions. It is hoped that this pilot study on an artificial benchmark will prompt further work in applications with other types of data.

REFERENCES

[1] C. M. Bishop, *Neural networks for pattern recognition*, Clarendon Press, Oxford, 1995.

[2] T. Chin-Chi and B. W. Wah, "An automated design system for finding the minimal configuration of a feed-forward neural network," *Proceedings of the IEEE International conference on neural networks*, Florida, vol. 3, pp. 1295-1300, 1994.

[3] H. C. Chua, J. Jia, L. Chen and Y. Gong, "Solving the two-spiral problem through input data encoding," *Electronics letters*, vol. 31, issue 10, 813-14, 1995.

[4] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," *Proceedings of the 1988 Connectionist models summer school*, Morgan Kaufmann, 1988.

[5] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation learning architecture," In Touretzky (Ed.) *Advances in neural information processing systems 2*, Morgan Kaufmann, 1990.

[6] M. Brown and C. A. Harris, "Fuzzy logic expert system for iron ore processing," *Proceedings of the IEEE industry applications conference*, Canada, vol. 3, pp. 2190-9, 1993.

[7] J. Jia and H. Chua, "Solving two-spiral problem through input data representation," *Proceedings of the IEEE International conference on neural networks*, vol. 1, pp. 132-135, 1995.

[8] B. Kosko, *Neural networks and fuzzy systems - A dynamical systems approach to machine intelligence*, Prentice Hall International edition, 1992.

[9] K. J. Lang and M. J. Witbrock, "Learning to tell two spirals apart," *Proceedings of the 1988 Connectionist models summer school*, Morgan Kaufmann, 1988.

[10] *Fuzzy reasoning and its applications*, E. H. Mamdani and B. R. Gaines (eds.), Academic Press, 1981.

[11] S. Singh and M. Steinl, "Fuzzy pattern recognition for knowledge-based systems," *Proceedings of the International conference on knowledge based computer systems*, Bombay, 1996, pp. 383-394.

[12] S. Singh, E. L. Hines and J. Gardner, "Fuzzy neural computing of coffee and tainted water data on electronic nose," *Sensors and Actuators B*, vol. 30, issue 3, pp. 190-195, 1996.

[13] S. Singh, E. L. Hines and J. Gardner, "Classifier systems based on possibility distributions: A comparative study," *Proceedings of the 3rd International conference on neural networks and genetic algorithms (ICANNGA97)*, Norwich, Wien: Springer-Verlag, 1997.

[14] C. T. Sun and J. S. Jang, "A neuro-fuzzy classifier and its applications," *Proceedings of the IEEE International conference on fuzzy systems*, vol. 1, pp. 94-98, 1993.

[15] L. P. Tay and D. J. Evans, "Fast learning artificial neural network (FLANN II) using the nearest neighbour recall," *Neural, parallel and scientific computations*, vol. 2, issue 1, pp.17-27, 1994.

[16] D. S. Touretzky and D. A. Pomerleau, "What's hidden in the hidden layers ?" *Byte* (August issue), pp. 227-233, 1989.

[17] F. Ulgen, N. Akamatsu and T. Iwasa, "The hypercube separation algorithm: a fast and efficient algorithm for on-line handwritten character recognition," *Applied Intelligence*, vol. 6, issue 2, 101-116, 1996.

[18] S. M. Weiss, and C. A. Kulikowski, *Computer systems that learn*, Morgan Kaufmann, San Mateo, CA, 1991.

[19] L. A. Zadeh, *Fuzzy logic and its applications*, Academic press, New York, 1965.