ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Pattern Recognition using Chaotic Transients

Wee Jin Goh and Nigel Crook

School of Technology - Department of Computing
Oxford Brookes University, Wheatley Campus, Oxford - United Kingdom

Abstract. This paper proposes a novel nonlinear transient computation device described as the LTCM that uses the chaotic attractor provided by the Lorenz system of equations to perform pattern recognition. Previous work on nonlinear transient computation has demonstrated that such devices can process time varying input signals. This paper investigates the ability of the LTCM to correctly classify static, linearly inseperable data sets commonly used as benchmarks in the pattern recognition research community. The results from the LTCM are compared with those from support vector machines and multi-layer perceptrons on the same data sets.

1 Introduction

This article extends the recently proposed Nonlinear Transient Computation Machine (NTCM)[1]. At the heart of the NTCM is the nonlinear attractor N_T that provides a nonlinear transformation from input space to phase space. The N_T relies on the property of a nonlinear attractor called sensitivity to initial conditions [2]. This property states that two slightly different starting points of a chaotic system will result in an very different trajectories over time. In the NTCM, inputs are encoded as perturbations of the system away from the attractor, proportional input will result in a proportional response as the system wanders back onto the attractor. Nevertheless, regardless of how chaotic a system is, the trajectories will be similar for a short period of time before diverging. This means that observing transients near the point of perturbation will yield a high level of noise robustness, while observing transients further along the time series from the perturbation will increase the sensitivity of the classifier [1].

The Liquid State Machine (LSM) is a method of computation that has gained a considerable interest [3]. It was formulated to overcome the inherent difficulties of training a recurrent network, which are notorious for their long execution times and high memory requirements. The LSM is composed of a large recurrent layer of neurons (called the "liquid layer"), with many random short lateral connections and few long. The liquid layer serves as a nonlinear projection of inputs into a higher dimensional space, facilitating the classification of said input by boosting the power of linear readout units in classification and regression tasks. It is claimed that this is similar to how a support vector machine (SVM) functions.

Previously, work has been done to demonstrate how the NTCM functions in a comparable way to the LSM [4]. It was demonstrated that the N_T which was modeled by an NDS neuron[5], had similar properties to the LSM modeled by a layer of recurrent neurons. This paper develops this idea further, replacing the NDS neuron in the N_T with the Lorenz attractor. The motivation for this change is to further study the contribution of chaos, specifically the property that separates nearby trajectories. While the NDS neuron is a chaotic system, it is a system that isn't as well understood as the Lorenz. In addition to that, it was observed that the NDS neuron was difficult to control, mainly because of the mapping from the 3-dimensional chaotic internal state to the 1-dimensional spike output.

The transients from the N_T are presented to a set of observers, called the N_R. In this model, the N_R is made up of a single perceptron with a sigmoidal transfer function for *each* class of input. Training of the observer is via theδ rule.

2 Experimental Setup

This article presents a variant on the previously published NTCM, called the Lorenz Transient Computation Machine (LTCM) which uses the well known Lorenz attractor [6] as the N_T, with perceptrons as the N_R. The equations for the Lorenz attractor are given in Equation 1. The Lorenz is used as the N_T because it is a widely understood and extensively studied chaotic attractor.

$$\begin{aligned} \dot{x} &= ay - ax \\ \dot{y} &= xb - xz - y \\ \dot{z} &= xy - cz \end{aligned} \tag{1}$$

The N_T consists of a chaotic Lorenz attractor. Each input attribute is represented by a separate trajectory on the attractor. Each attribute of the input is multiplied by a set of weights and the resulting value is added to the initial starting point of the respective trajectory, the triplet $\{x_1, y_1, z_1\}$. Thus, for a pattern set that consisted of N input attributes N trajectories are required for the N_T. The initial conditions for the N trajectories are given by Equation 2. P corresponds to an instance of input, and w represent the weights to the attractor, x_{0i}, y_{0i}, and z_{0i} are the initial conditions of the respective trajectories, while x_{1i}, y_{1i}, and z_{1i} are the perturbed points.

$$\begin{aligned} x_{1i} &= x_{0i} + \sum_{i=1}^{N} P_i * w_i \\ y_{1i} &= y_{0i} + \sum_{i=1}^{N} P_i * w_i \\ z_{1i} &= z_{0i} + \sum_{i=1}^{N} P_i * w_i \end{aligned} \tag{2}$$

The resulting transients generated by the attractor are sampled at τ intervals. The input I that is presented to the observer, is the column sum of the transients.

This column sum is described by Equation 3, where ρ is the number of samples taken. In effect, this process collapses a 3-dimensional time series into a 1-dimensional time series. This is done in order to reduce the number of inputs presented to the observer N_R. In the experiments presented in this paper, the value of τ was set to 10, and the number of samples taken was 25. The system was allowed to evolve for 250 time steps after the point of perturbation before sampling began in order to allow for more separation of transients. The observers N_R used in this setup are perceptrons with no hidden units, trained using the δ rule.

$$I_t = x_t + y_t + z_t : t = 1...\rho \tag{3}$$

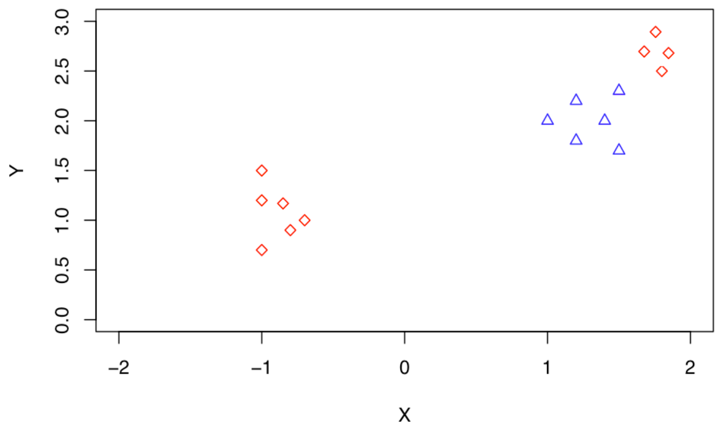

Fig. 1: The two classes of input, denoted by the diamonds and triangles are nonlinearly separable. One of the diamonds, lies on the far side of the cluster of triangles.

A contrived demonstration of how the model is able to classify nonlinearly separable input follows. Consider the two classes of inputs denoted by diamonds and triangles presented in Figure 2. The data-set is made up of 2 attributes X and Y, and is nonlinearly separable, because some of the diamonds lies on the far side of the cluster of triangles. The LTCM is able to correctly classify both the clusters of diamonds, because the observer is able to associate the points in the transients belonging to the diamond class that are linearly separable from transients of the triangle input class. A snapshot of the transients which are presented to the observer is given in Figure 2. Points in the transients which are close together, are assigned a stronger observer weight which are denoted in by (a), and points which are slightly further apart but still linearly separable are assigned a weaker observer weight, denoted by (b).

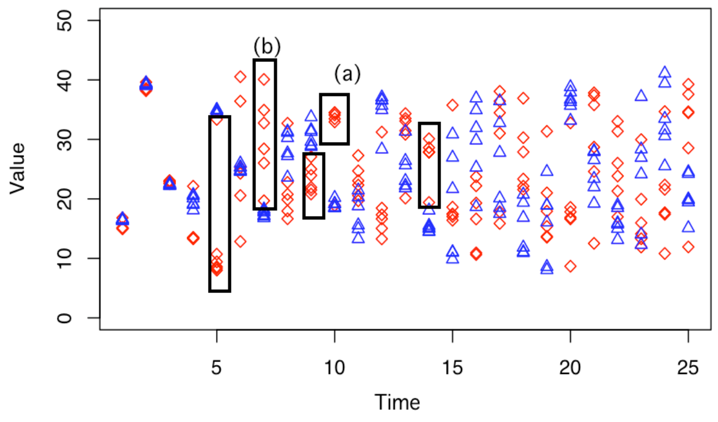

Fig. 2: Inputs to the observer neuron. The observer picks out points in the transients that are linearly separable, denoted by the black boxes. Points that are closely clustered (a) are assigned larger weights while points that are more widespread (b) are assigned weaker weights.

This demonstrates that the observer is able to pick out regions within attractor space, where transients of a particular class are most likely to pass through. In effect, these regions or hot spots in attractor space can be likened to the characteristics of that particular class of input. Hence, the observer is classifying based on the emergent characteristics of that particular class of input, facilitated by the N_T.

3 Results

The LTCM is put through some standard pattern recognition tasks, with the datasets obtained from the UCI Machine Learning repository [7]. The datasets used were Iris (IR), Breast Cancer (BC) and Sonar (SN). The IR dataset consists of 3 classes with 4 numeric attributes. Of the 3 classes 1 is linearly separable from the other 2, while the remaining 2 are not linearly separable from each other. The BC dataset is split into 2 classes, benign and malignant. It consists of 9 integer valued attributes and is also nonlinearly separable. Finally, the SN dataset is split into 2 classes, rock or mine and consists of 61 attributes.

The LTCM is compared with two other well known pattern classifiers, the multilayer perceptron with 6 hidden units (MLP) and the support vector machine with a 3rd order polynomial kernel (SVM). The results for the MLP and SVM come from the WEKA test suite [8]. The datasets are trained and tested using 10-fold cross validation, where each dataset is split into 10 equal parts with the classifiers trained on 9 parts and tested on 1 part. This process is repeated

	$LTCM$	SVM	MLP
BC	95.42%	96.28%	95.27%
IR	92.22%	92.87%	95.73%
SN	77.87%	84.82%	83.11%

Table 1: Comparison of pattern classifiers. The table lists the percentage correct each classifier scored when doing 10-fold cross validation.

10 times. The results of classification are presented in Table 3. The results show that the model presented in this paper compares favorably with other well established methods.

4 Discussion

The results from the 10-fold cross validation experiments presented in table 1 show that the LTCM offers comparable performance to SVMs and MLPs on the three benchmark pattern recognition tasks. As with all transient computation devices (including LSMs), the LTCM reduces the computational overheads often required for training classifiers on linearly inseparable problems. This is achieved by projecting the input space on to the phase space of the Lorenz chaotic attractor. This projection is modelled by representing each input pattern as a vector (or a set of vectors) that emerges from a fixed starting point that is within the basin of attraction of the Lorenz system. The end point of this vector defines the starting point of the chaotic trajectory to be associated with that input. Since the vector in phase space is characteristic of the input pattern that gave rise to it, the trajectory that evolves from the end point of this vector will also characteristic of the input pattern (this is guaranteed by the chaotic nature of the attractor). As the trajectory evolves around the chaotic attractor it will (eventually) come very near to every point on the attractor surface (another common property of chaotic systems). Therefore, two different trajectories on the same attractor will repeatedly converge and diverge as they progress around the attractor, even if they have evolved from very different starting points. Consequently, two input patterns from the same class that are somewhat separated in input space (as with the two clusters of the diamond class in Figure 1), will result in two trajectories that repeatedly converge and diverge as they evolve in time (illustrated in Figure 2). It then becomes a trivial task for a simple linear readout device observing these trajectories to recognise the points in time when they are close together. Any input that is similar to either of these patterns will result in a trajectory that, in the short term, will come close to the points in phase space where the trajectories caused by the original two patterns converged. Clearly, there are issues concerning the length of the trajectory that is observed by the readout devices. These have been addressed in detail elsewhere [4, 1].

5 Conclusion

The results presented in this paper show that it is possible to create a pattern classifier by using the property of a chaotic attractor, that the trajectory of nearby points will diverge over time. The divergence of the trajectories can be likened to the transformation in a nonlinear kernel of a support vector machine. The resulting transformed input can be classified by a simple perceptron.

This paper has focussed on the classification of static patterns. We have also applied the LTCM to classifying time-dependent input signals [9]. Future work on transient computation will investigate the use of other chaotic attractors within the NTCM framework.

References

[1] N. T. Crook. Nonlinear transient computation. *Neurocomputing*, In press.

[2] Robert C. Hilborn. *Chaos and Nonlinear Dynamics, 2nd Edition.* Oxford University Press, 2000.

[3] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[4] N. T. Crook. Nonlinear transient computation and variable noise tolerance. In *Proceedings of 14th European Symposium on Artificial Neural Networks (ESANN'2006)*, pages 509–514, 2006.

[5] N. T. Crook, W. J. Goh, and M. Hawarat. The nonlinear dynamic state neuron. In M. Verleysen, editor, *Proceedings of 13th European Symposium on Artificial Neural Networks (ESANN'2005)*, 2005.

[6] E.N. Lorenz. Deterministic non-periodic flow. *Journal of Atmospheric Science*, 20(2):130–141, 1963.

[7] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

[8] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques 2nd Edition.* Morgan Kaufmann, 2005.

[9] N. T. Crook and W. J. Goh. Human motion recognition using nonlinear transient computation. In *Submitted to European Symposium of Artificial Neural Networks*, 2007.