Journal of Computer Science 3 (2): 84-87, 2007
ISSN 1549-3636

Lizard Learning Algorithm for Invariant Pattern Recognition

[1]G. Mallikarjuna Rao, [2]G. R. Babu and [3]G. Vijaya Kumari
[1]G. Narayanamma Institute of Technology and Science, Shaipet, Hyderabad-8, India
[2]Jawaharlal Nehru Technological University, Hyderabad, India
[3]Computer Science Department, Jawaharlal Nehru Technological University, Hyderabad, India

Abstract: Researches are keen to know astonishing and intricate details of the nature. Each creature has its own admiring abilities and performs their routine task in more efficient manner. The bug navigation system has drawn keen attention among research community to know how they are able to perform their routine task in utmost skillful manner. The lizard is capable of identifying slowly varying features and able to trap the insects with more admiring skill set. The Lizard Learning Algorithm (LLA) was proposed for tracking invariant features which uses modified slow feature analysis. The article covers mathematical treatment for the slow feature analysis, proposed modification, higher order neural network training and ORL database for experimentation purpose. The results are most pleasing compared to conventional classifiers for the invariant features.

Key words: Slow feature analysis, higher order neural network, eye crapping, unsupervised feature extraction, invariant pattern recognition

INTRODUCTION

The bug-navigation system reveals most effective techniques carried out by the small living creatures to meet their bread and butter. It was shocking to note that the small paddle bug is able to choose exact straight-line path to reach their nest from unknown/new-location. The ant column is triggered the development of more efficient tracking algorithms. Further the small spider is capable of choosing an optimal shortest path in a widely spread environment. On the same line in this paper it is proposed a lizard learning algorithm capable of tracking invariant features. The study reveals that the effortless techniques used by lizards while capturing the insects are so effective. They capture the slowly varying features of the insect while tracking and detecting its exact location. These techniques are suitable for implementing the invariant pattern recognition.

Slow feature analysis (SFA)[1,2] is a new unsupervised algorithm to learn nonlinear functions that extract slowly varying signals from time series[1]. SFA was originally conceived as a way to learn salient features of time series in a way invariant to frequent transformations[2]. Such a representation would of course be ideal to perform classification in pattern recognition problems. Most such problems, however, do not have a temporal structure and it is thus necessary to reformulate the algorithm. The basic idea is to construct a large set of small time series with only two elements chosen from patterns that belong to the same class. In order to be slowly varying, the functions learned by SFA will need to respond similarly to both elements of the time series and therefore ignore the transformation between the individual patterns. As a consequence, patterns corresponding to the same class will cluster in the feature space formed by the output signals of the slowest functions, making it suitable to perform classification with simple techniques such as Gaussian classifiers.

The higher order neural network require one pattern should be presented only once during the learning stage. This ability makes them faster compared to single order multi-layered networks. However their computational complexity exponentially grows with respect to the size of the image. In this study some modifications are suggested so that the network can be trained for OCR database.

The SFA algorithm: We can now formulate the Slow Feature Analysis (SFA) algorithm[2]

Mathematical constraints: Given a multidimensional time series $x(t) = (x_1(t),... ,x_N(T))^T$, $t \in [t_0,t_1]$, find a set of real-valued functions $g_1(x),..., g_M(X)$ lying in a function space F such that for the output signals $y_j(t) := g_j(x(t))$

$$\Delta(y_j) := (y_j^2)_t \text{ is minimal} \quad (1)$$

Under the constraints

$$(y_j)_t = 0 \text{ (zero mean)} \quad (2)$$

$$(y_j^2)_t = 1 \text{ (unit variance)} \quad (3)$$

$$\forall i < j, (y_i y_j)_t = 0$$

(decorrelation and order), (4)

with $(.)_t$ and y indicating time averaging and the time derivative of y, respectively.

Corresponding Author: G. Mallikarjuna Rao, G. Narayanamma Institute of Technology and Science, Shaipet, Hyderabad-8, India, Tel: 9440755716

Equation (1) introduces a measure of the temporal variation of a signal (the *Δ-value* of a signal) equal to the mean of the squared derivative of the signal. This quantity is large for quickly-varying signals and zero for constant signals. The zero-mean constraint (2) is present for convenience only, so that (3) and (4) take a simple form. Constraint (3) means that each signal should carry some information and avoids the trivial solution gj(x) = 0. Alternatively, one could drop this constraint and divide the right side of (1) by the variance *(yj)t*- Constraint (4) forces different signals to be uncorrelated and thus to code for different aspects of the input. It also induces an order, the first output signal being the slowest one, the second being the second slowest, etc. .

Linear expansion: For the linear case the function $g_j(x)$ becomes

$$g_j(x) = w_j^T x \quad (5)$$

Where x is the input vector and w is the weight vector.

On solving the above function with the constraints (1), (2), (3)

$$\Delta(y_j) = \lambda_j \quad (6)$$

The eigenvectors are stored in the ascending order of eigenvalues to provide slowly varying signals with smaller indices[3].

Nonlinear expansion: Expand the input data and compute the mean over time $h_0 := (h(x))_t$ to obtain the expanded signal

$$z := h(x) - h_0 \quad (7)$$

$$= (h_1(x),...,h_M(x)) - h_0 \quad (8)$$

Slow feature extraction: Solve the generalized eigenvalue problem

$$AW = BWA \quad (9)$$

$$A := (\dot{z}\dot{z}^T)_t \quad (10)$$

$$B := (zz^T)_t \quad (11)$$

The *K* eigenvectors $w_1,...,w_K$ *(K ≤ M)* corresponding to the smallest generalized eigenvalues $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_k$[4] define the nonlinear input-output functions $g_1(x),..., g_k(x) \in F$:

$$g_j(x) = w_j^T (h(x) - h_0) \quad (12)$$

which satisfy Constraints (2)-(4) and minimize (1).

SFA for pattern recognition: The pattern recognition problem can be summarized as follow. Given *C* distinct classes $c_1,..., c_C$ and for each class c_m a set of P_m patterns $p_1^{(m)}.., p^{(m)}_{Pm}$ we are requested to learn the mapping c(.) between a pattern $p_j^{(m)}$ and its class $c(p_j^{(m)}) = c_m$. We define $P := \Sigma^c_{m=1} P_m$ to be the total number of patterns.

In general in a pattern recognition problem the input data does not have a temporal structure and it is

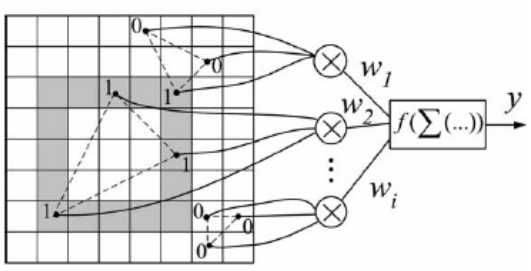


Fig. 1: Schematic description of a third-order network

thus necessary to reformulate the definition of the SFA algorithm. Intuitively, we want to obtain a set of functions that respond similarly to patterns belonging to the same class. The basic idea is to consider time series of just two patterns ($p_k^{(m)}$, $p_l^{(m)}$), where *k* and l are two distinct indices in a class c_m.

Rewriting Equation (1) using the mean over all possible pairs we obtain

$$\Delta(y_j) = a.\sum_{m=1}^{c}\sum_{\substack{k,l=1 \\ k<1}}^{p_m}\left(g_j(p_k^{(m)}) - g_j(p_l^{(m)})\right)^2 \quad (13)$$

where the normalization constant *a* equals one over the number of all possible pairs, i.e.

$$a = \frac{1}{\sum_{m=1}^{c} P_{m_2}} \quad (14)$$

We reformulate Constraints (2)-(4) by substituting the average over time with the average over all patterns, such that the learned functions are going to have zero mean, unit variance and be de-correlated when applied to the whole training data. This reduces to an optimization problem.

$$\frac{1}{p}\sum_{m=1}^{c}\sum_{k=1}^{p_m} g_j(p_k^{(m)}) = 0 \text{ (zero mean)} \quad (15)$$

$$\frac{1}{p}\sum_{m=1}^{c}\sum_{k=1}^{p_m} g_j(p_k^{(m)})^2 = 1 \text{ (unit variance)} \quad (16)$$

$$\forall i < j, \frac{1}{p}\sum_{m=1}^{c}\sum_{k=1}^{p_m} g_j(p_k^{(m)}) g_j(p_k^{(m)}) = 0$$

(de-correlation and order) (17)

* Additional material is available at http://yann.lecun.com/exdb/mnist/. More information about SFA is available[5]. Several authors[4-6] proposed variant solutions for slow feature analysis.

HONN architecture: The HONN (Higher Order Neural Network)[7-9] out performs single order multi layered networks. The HONN requires application of learning pattern only once. Hence its response is faster. They are able to exploit the inter-relations among the input data while providing invariant pattern recognition. The third order network is shown below.

The output of a third-order network can be described by the following equation:

$$y_i = f\left(\sum_a\sum_b\sum_c w_{iabc} x_a x_b x_c\right) \quad (18)$$

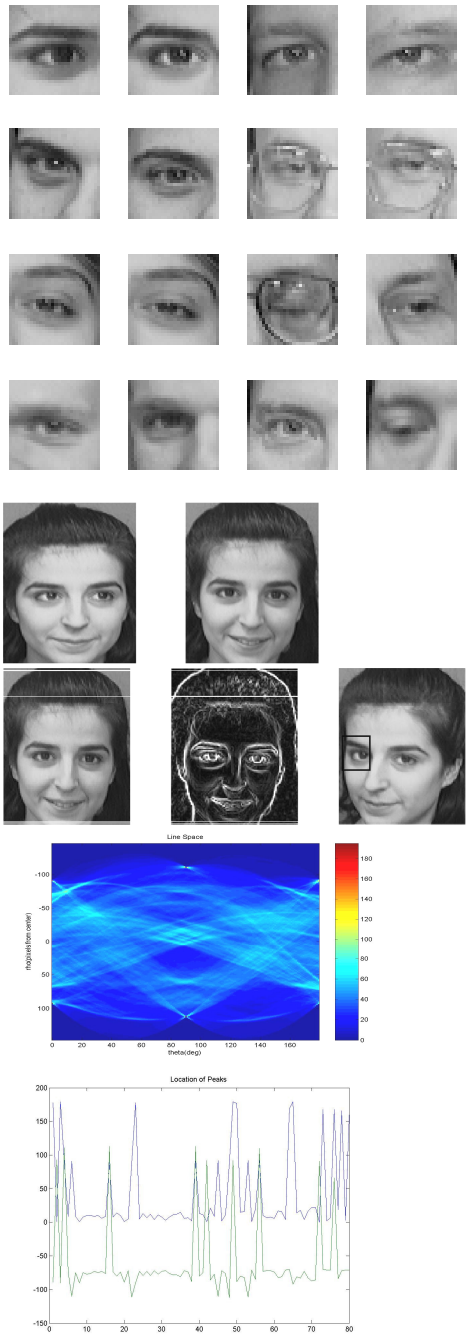


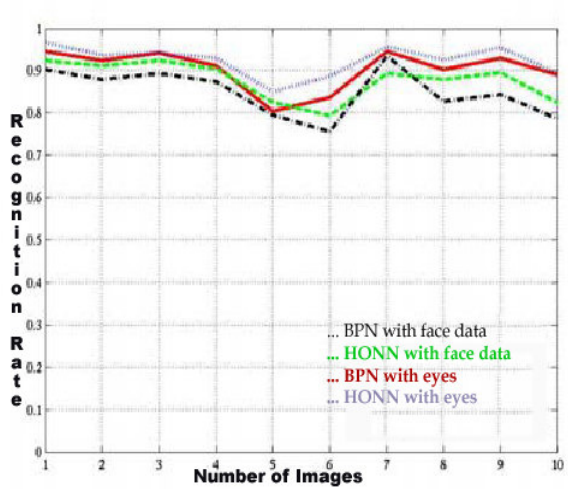
Fig. 2: Training with facial features ORL Database

where *i* is the output index, *w* is the weight associated with a particular triangle, *y* is the actual output, *x* is a binary input and *a, b and c* are the indices of the inputs. A schematic description of this network is shown in Fig. 1.

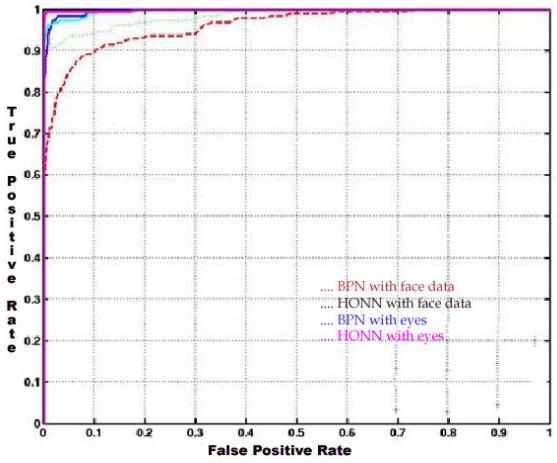
In the training phase, a perceptron-like rule is used:

$$\Delta w_{iabc} = \eta(t_i - y_i)x_a x_b x_c \quad (19)$$

where *t* is the expected training output, *y* is the actual output, *g* is the learning rate and *x* is a binary input. The exponential increase of the triangles with the input image is the major problem in higher order neural



(i)



(ii)

Fig. 3: Recognition with eye data using (i) BPF & (ii) HONN

networks. By restricting to the contour cells (active pixels) and similarities among the triangles are used to reduce the number of triangles and weight classes during the process of training. The following tabular form gives the complexity problem of HONN with image size[10,11].

Table 1: The number of triangles as a function of input image size

Input image size	Number of triangles
4x4	560
8x8	41664
12x12	487 344
20x20	10586800
40x40	681387200
100 x100	1.6662x10"
256 x 256	4.6910 xIO[13]

Number of weight classes is equal to the number of triangles which is equal to IN_{C_3}

$$NoT = \frac{IN!}{(IN-3)!3!} \quad (20)$$

where IN is the number of input nodes. The number of possible triangles for different input sizes is given in Table 1.

HONN training:

1. The coordinates of active pixels on the contour are to be stored in separate X,Y arrays for every pattern.
2. Compute the active triangles for every pattern using the coordinates $(x_a,y_a),(x_b,y_b),(x_c,y_c)$ and identify weight classes(W). Similar triangles are to be placed in the same class.
3. Initialize the weights with the number of triangles contained by them (Nk).
4. Compute the output $y_i = f\left(\sum_j w_{ij} N_{kj}\right)$
5. Update the concerned weight if $N_{kj} > 0$ $\Delta w_{ij} = \eta(t_i - y_i)$

Example application: We illustrate our method by its application to a person identification using eyes and faces for invariant recognition. We consider the CBCL and ORL face database, which contains of a 20 people approximately 200 images for the person. From the ORL database 10 eyes inclusive of left eye and right eye are considered during the process of training (Fig. 2). They are able to recognize the person in-spite of variations due to the expressions.

The features obtained from SFA are trained using both conventional back-propagation algorithm and higher order neural network. The higher order neural network with eyes features given better recognition rate.

CONCLUSION

Our experimental results show the higher order neural network training gives 15% higher performance conventional backpropagation (Fig. 3). We have tried with double backpropagation training the results are not uniformly encouraging. The work can be extended by adding self similarity probabilities while capturing slow features. Further this work can be extended for seen/expression analysis.

REFERENCES

1. Wiskott, L., 1998. Learning Invariance Manifolds. In: Niklasson, L., Boden, M., Ziemke, T. (Eds.), Proc. Intl. Conf. on Artificial Neural Networks, ICANN'98, Skovde. Perspectives in Neural Computing. Springer, pp: 555-560.
2. Wiskott, L. and T. Sejnowski, 2002. Slow feature analysis: Unsupervised learning of invariances. Neural Computation, 14: 715-770.
3. Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition.
4. Gantmacher, F.R., 1959. Matrix Theory. Vol. 1. AMS Chelsea Publishing.
5. Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.
6. Bray, A. and D. Martinez, 2002. Kernel-based extraction of Slow Features: Complex cells learn disparity and translation invariance from natural images. In: NIPS 2002 Proceedings.
7. Barnard, E. and D. Casasent, 1991. Invariance and neural nets. IEEE Trans. Neural Networks, 2: 498-507.
8. Chaudhuri, B.B. and U. Bhattacharya, 2000. Efficient training and improved performance of multilayer perceptron in pattern classification. Neurocomputing, 34: 11-27.
9. He, Z., 1999. Invariant pattern recognition with higher-order neural networks. M.Sc. Thesis, School of Electrical and Computer Engineering, Nanyang Technological University, Singapore.
10. Mahmoud, I.K. and M.B. Mohamed, 2000. Invariant 2D object recognition using the wavelet modulus maxima. Pattern Recogn. Lett., 21: 863-872.
11. Spirkovska, L. and M.B. Reid, 1993. Coarse-coded higher-order neural networks for PSRI object recognition. IEEE Trans. Neural Networks, 4: 276-283.