Extended Nonlinear Hebbian Learning for Developing Sparse-Distributed Representation

Bai-ling Zhang and T.D. Gedeon

Department of Information Engineering,
School of Computer Science and Engineering,
University of New South Wales,
Sydney 2052, Australia
{bailingz,tom}@cse.unsw.edu.au

Abstract. *Recently, Hebbian learning has been extended to nonlinear* units with a number of interesting properties and potential applications, e.g., blind signal separation. However, when generalizing these nonlinear Hebbian learning algorithms to a network with multiple units, all the existing methods assume orthonormality constraints, which is too strict in many occasions. In this paper, we propose two alternative approaches to generalize nonlinear Hebbian learning to a network with M neurons, based on the mixture-of-experts paradigm. Preliminary simulation shows interesting results.

1 Introduction

In statistics, many established techniques can be considered as generalizations of principal components analysis (PCA). For example, factor analysis, projection pursuit (PP), principal curve analysis, etc. Originating from Oja's work [1] on extracting the first principal component by a linear neuron, the issue of neural learning PCA has arisen great interest in recent years. Among a number of properties of the largest principal component, the maximum variance property, or dimensionality-reduction property has often been applied.

To overcome the limitations of PCA, *i.e.*, its reliance on second-order statistics and linear projection, some nonlinear extensions have been proposed in recent years. Oja et al extended the Hebbian learning rule to nonlinear units [2], which has drawn increased attention. It was pointed out in [5] that the utilization of units with nonlinear activation functions which employ Hebbian learning may lead to robust principal component analysis. In [4] a nonlinear Hebbian learning (NHL) was derived from a variance maximization objective, which can provide a nonlinear nonparametric approach to dimension reduction. Under appropriate conditions, the projection obtained from the NHL performs a statistical operation equivalent to projection pursuit. Another closely related nonlinear Hebbian learning has also been shown to have a similar characteristics and applied to primary vision research [7].

As the task of capturing nonlinear statistical structure within the data is ubiquitous in various engineering problems, it is significant to further study NHL. However, the extension of NHL to a network of multiple units in [4] is trival in the sense that an orthonormality constraint is introduced among the units. In practice, multiple non-orthogonal non-Gaussian projections are more interesting. In this paper, we propose two alternative extensions of NHL to a network with M neurons. In the first method, competition is introduced among the units, each of which is adapted by the nonlinear Hebbian learning. Instead of the simple winner-take-all, which has some disadvantages such as underutilization, we apply a soft-competition mechanism inspired by the 'neural gas' algorithm in [9]. In the second method, we treat each nonlinear neuron as a local expert and explicitly use a gating network to coordinate the adaptation of each neuron. Both methods employ a deterministic simulated annealing scheme to avoid the local minimum problem and both schemes can develop a sparse-distributed representation in the network. This has recently been proposed as the aim of sensory coding, especially in respect to the coding of natural visual scenes [13]

2 Nonlinear Hebbian learning rule for a single neuron

We consider a nonlinear neuron model with connections w_i, $i = 1, \cdots, L$. For an input vector $\mathbf{x} \in \mathrm{R}^{\mathrm{L}}$, the neuron's output is a nonlinear transform via $\mathbf{w} = [w_1, w_2, \cdots, w_{\mathrm{L}}]^{\mathrm{T}}$:

$$\begin{aligned} z &= f(y) \\ &= f(\mathbf{w}^{\mathrm{T}}\mathbf{x}) \end{aligned} \tag{1}$$

where f is a nonlinear activation function, $y = \mathbf{w}^{\mathrm{T}}\mathbf{x}$. In this paper, we consider the simple sigmoid $f(t) = \frac{1}{1+e^{-t}}$.

The connections $w_1, w_2, \cdots, w_{\mathrm{L}}$ can be adapted via the maximum variance principle. Specifically, the cost function for $\mathbf{w}$ is the following objective

$$\begin{aligned} &\text{maximize} \quad J = E[z^2] \\ &\text{subject to} \quad \|\mathbf{w}\| = 1 \end{aligned} \tag{2}$$

which is a nonlinear extension of Oja's rule [1] and has been studied in several recent papers [2,4-5,7-8]. As can be seen, utilizing a Taylor series expansion of the sigmoidal nonlinearity, the objective function (2) implicitly maximizes higher order moments [5].

The gradient ascent of J leads to the following learning rule [4]

$$\mathbf{w}_{\mathrm{k+1}} = \mathbf{w}_{\mathrm{k}} + \mu_{\mathrm{k}} z(\mathbf{x} - \mathbf{w}_{\mathrm{k}}\mathbf{w}_{\mathrm{k}}^{\mathrm{T}}\mathbf{x})g \tag{3}$$

where g is the derivative of f, $g = f'(\mathbf{w}_{\mathrm{k}}^{\mathrm{T}}\mathbf{x})$ and μ_k is the learning parameter.

In the case of a linear unit, the objective function (2) seeks projections of input patterns in the direction of maximum variance, *i.e.*, the direction of the principal component. When applying the same objective function to a nonlinear neuron model, the result will be usually different due to the restriction of output nonlinearity. In the case of $tanh(\cdot)$ nonlinearity which bound the output within $[-1, +1]$, the variance maximization objective will seek a weight vector $\mathbf{w}$ such that the output is distributed near -1 and $+1$, *i.e.*, a U-shape output distribution. This implies that the nonlinear Hebbian learning rule eqn (3) performs a statistical operation to seek higher order statistical projections in the input space, and are closely related with exploratory projection pursuit techniques in the sense that they deviate from the Gaussian.

3 Extension of NHL to a network of M nonlinear units

In [4], the nonlinear Hebbian learning has been extended to a network with multiple units under the orthonormality constraint. As non-orthogonal projections are more interesting in many engineering problems, we study alternative extensions of NHL to a network in this section.

For a network with M neurons, the mth neuron has output $z_m = f(\mathbf{x}^{\mathrm{T}}\mathbf{w}(\mathrm{m}))$, $\mathbf{w}(\mathrm{m}) = [w_1(\mathrm{m}), w_2(\mathrm{m}), \cdots, w_{\mathrm{L}}(\mathrm{m})]^{\mathrm{T}}$.

Suppose each neuron is still adapted under the variance maximum objective (2). Each time an input $\mathbf{x}$ is presented, we first make an ordering of the output activations $z_1, z_2, \cdots, z_M$ and then determine the adjustment of $\mathbf{w}(\mathrm{m})$. Specifically, we determine an activation-ranking $z_{m_0}, z_{m_1}, \cdots, z_{m_{M-1}}$ of the activations, with z_{m_0} being the largest, z_{m_1} being second largest, z_{m_k}, $k = 0, \cdots, M-1$ being the activation for which there are k outputs z_j with $z_j > z_{m_k}$. At the same time, each neuron adjusts its own weight via a dynamical learning rate which depends on the ranking of its activation. Denote the number k associated with each neural unit m by k_m. The following learning rule is a natural extension of NHL:

$$\begin{aligned} \Delta\mathbf{w}_{\mathrm{k}}(\mathrm{m}) &= \mu_k h_\lambda(\mathrm{k_m}) z_{\mathrm{m}}(\mathbf{x} - \mathbf{w}_{\mathrm{k}}(\mathrm{m})\mathbf{w}_{\mathrm{k}}(\mathrm{m})^{\mathrm{T}}\mathbf{x}) g_{\mathrm{m}} \\ & m = 1, \cdots, M \end{aligned} \tag{4}$$

where $h_\lambda(k_m)$ is 1 for $k_m = 0$ and decays to zero for increasing k_m. In the simulation we choose $h_\lambda(k_m) = exp(-k_m/\lambda)$, with λ being a decay constant. $g_m = f'(\mathbf{w}_{\mathrm{k}}(\mathrm{m})^{\mathrm{T}}\mathbf{x})$.

Clearly, our algorithm (4) utilizes a soft-competition scheme called "neural gas" proposed in [9]. By ranking of the activations, the nonlinear neurons are driven into competition, with a main feature of "competitive annealing". A large value of λ implies that the nonlinear neurons almost equally share the same data

for training. Decreasing λ enforces the competition, thereby driving the nonlinear neurons to specialize in different subsets of the data.

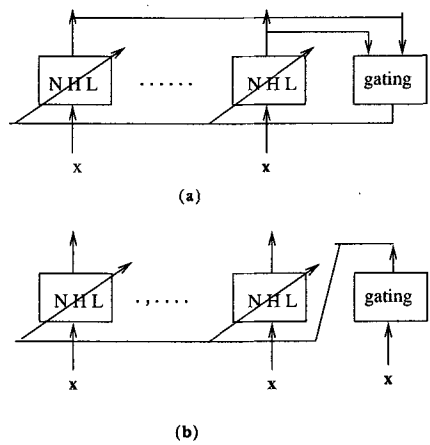

Fig. 1. Illustration of the two schemes for extension of the nonlinear Hebbian learning (NHL) to a network with M neurons.

Above method of introducing competition in learning among the neurons can be schematically illustrated in Fig.1(a), in which NHL stands for the nonlinear Hebbian learning in a nonlinear neuron. This is closely related to the well-known "mixture of experts" architecture [3], in which a gating network is usually not decided by a local expert's performance but depends on input. A more explicit application of the mixture of experts paradigm can proceed as follows. Suppose that an input-dependent gating network provides a coordinating parameter π_m as a dynamical learning rate for the mth nonlinear neuron, then the adaptation can proceed as follows:

$$\begin{aligned} \Delta \mathbf{w_k}(\mathrm{m}) &= \mu_k \pi_m z_\mathrm{m}(\mathbf{x} - \mathbf{w_k}(\mathrm{m})\mathbf{w_k}(\mathrm{m})^\mathrm{T}\mathbf{x}) g_\mathrm{m} \\ m &= 1, \cdots, M \end{aligned} \tag{5}$$

There are many ways to adapt a gating network. In the the following, we apply a normalized parametric function which was proposed in [11]

$$\pi_m = \frac{exp(-\frac{\|\mathbf{x}-\mathbf{w}(\mathrm{m})\|^2}{2T^2})}{\sum_{n=1}^{M} exp(-\frac{\|\mathbf{x}-\mathbf{w}(\mathrm{m})\|^2}{2T^2})} \tag{6}$$

where T is a parameter called temperature for controlling the covariance in the respective Gaussian distribution.

Following the deterministic annealing procedures in [10], the annealing process can be introduced by starting from a high T where a data point equally influences all the neurons, and all the neurons equally share their responsibilities in representing an input. Progressively, as T decreases, the influence of each data point is gradually localized.

The deterministic annealing algorithm can be outlined in steps as follows:

1. Set $T = T_{max}$;
2. Choose an arbitrary set of initial weights;
3. Calculate π_m by (6) and iterate according to (5) for a fixed number of samples;
4. Decrease T;
5. If $T > T_{min}$, go to 3. Otherwise end

The annealing schedule, or the rate at which T is decreased, is problem dependent. In our experiment, we have tried different schemes. For example, if we denote K as the maximum iteration number for executing annealing, a simple way is linearly decreasing T:

$$T_k = \frac{kT_{min} + (K-k)T_{max}}{K}, \; k = 1, \cdots, K \tag{7}$$

There are many ways to adapt a gating network. In the the following, we apply a normalized parametric function which was proposed in [11]

$$\pi_m = \frac{exp(-\frac{\|\mathbf{x}-\mathbf{w}(\mathrm{m})\|^2}{2T^2})}{\sum_{n=1}^{M} exp(-\frac{\|\mathbf{x}-\mathbf{w}(\mathrm{m})\|^2}{2T^2})} \tag{8}$$

where T is a parameter called temperature for controlling the covariance in the respective Gaussian distribution.
Following the deterministic annealing procedures in [10], the annealing process can be introduced by starting from a high T where a data point equally influences all the neurons, and all the neurons equally share their responsibilities in representing an input. Progressively, as T decreases, the influence of each data point is gradually localized.

The deterministic annealing algorithm can be outlined in steps as follows:

1. Set $T = T_{max}$;
2. Choose an arbitrary set of initial weights;
3. Calculate π_m by (6) and iterate according to (5) for a fixed number of samples;
4. Decrease T;
5. If $T > T_{min}$, go to 3. Otherwise end

The annealing schedule, or the rate at which T is decreased, is problem dependent. In our experiment, we have tried different schemes. For example, if

we denote K as the maximum iteration number for executing annealing, a simple way is linearly decreasing T:

$$T_k = \frac{kT_{min} + (K-k)T_{max}}{K}, \; k = 1, \cdots, K \tag{9}$$

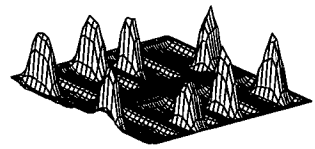

Fig. 2. Converged weight vectors in equalized images in a single layer bidirectional network, trained with random horizontal and vertical bars by learning algorithm (5).

4 Simulations

The problem of how a network responds to randomly placed Gaussian-shaped spots is often used to demonstrate a network's capability in partitioning the input space. We take this example to illustrate our learning algorithm. The data generation scheme and training were the same as that used in [12]. In the experiments, 100 input units in a 10×10 square array were tested. Each input vector was a random located Gaussian spot, with its center at arbitrary position except that there must be two input units away from the nearest edge in the input array. The average brightness of 1000 Gaussian spots was calculated beforehand and then subtracted from each random Gaussian spot during training. Initial weights were set to small random values. Typically, we tested 5000-10000 training samples. Fig. 2 illustrated a typical result of 8 nodes via algorithm (5). In this example, we take the sigmoidal nonlinearity $f(t) = \frac{1}{1+e^{-t}}$. We can find that different nodes have developed strong responses in overlapped different regions of the input space. The localized masks have their descriptive scopes that are narrowed to only certain regions of the full data space. The receptive fields of distinct units share their responsibilities in accounting for each observed data. After the learning is complete, we demonstrate the response of 100 units to a random input spot as shown in Fig.3 from which we can see that an input is represented by several units.

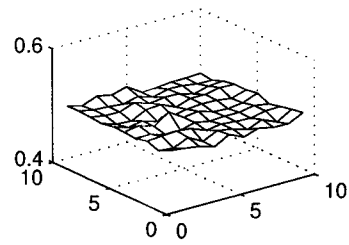

Fig. 3. The activations of 100 units in response to a randomly generated Gaussian-shaped spot.

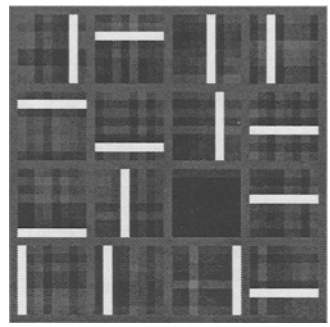

Fig. 4. Converged weight vectors in equalized images trained with random horizontal and vertical bars by learning algorithm (5).

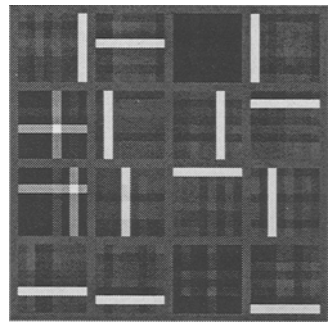

Fig. 5. Converged weight vectors in equalized images trained with random horizontal and vertical bars by learning algorithm (4).

We also consider another benchmark example of extracting a number of independent horizontal and vertical bars on an input pixel grid [12]. In a training data, each of the 16 possible lines in an 8×8 grid (horizontal or vertical bars) are drawn with a fixed probability, for example, $\frac{1}{8}$, independently from all the others. Pixels that are part of a drawn line have value 0, all others are 1. In this example, hidden statistical structures corresponding to the horizontal and vertical bars interact such that data pixels occurring at the intersection of bars remain black. The network has 16 representation units. An extra node is introduced to account for the average brightness. In this simulation, we tested with a generating probability $\frac{1}{8}$. Other generating probabilities, e.g., $\frac{2}{8}$ and $\frac{3}{8}$, usually yield similar results. In the experiment, we applied the algorithm (5), together with deterministic annealing, with results shown in Fig. 4 (equalized images of the weight vector arranged in an 8×8 square). 16 bars have been extracted. As a comparison, we give the result from our first extended nonlinear Hebbian learning rule (4). The two algorithems are qualitatively similar, while rule (5) shows a better result on extracting the independent horizontal and veryical bars .

5 Conclusion

Our proposed learning paradigm is essentially a mixture model of local projection pursuit, which can be considered as an extension of the mixture of experts from supervised learning to unsupervised learning. This mixture model concurrently performs two activities: partitioning data manifold into a number of regions and taking projection pursuit (or local nonlinear PCA) within each region. As an alternative to global PCA, it is expected to better describe some complex data generative mechanisms.

References

1. E.Oja, "Neural Networks, principal components, and subspaces," *Int. J. Neural Systems,* vol.1, no.1, pp. 61-68, 1989.
2. E.Oja, H.Ogawa, and J.Wangviwattana, "Learning in nonlinear constrained Hebbian networks," *Artificial Neural Networks (Proc.ICANN-91, Espoo, Finland)* , Eds. T.Kohonen, etc., North-Holland, pp.385-390, 1991
3. M.I.Jordan and R.A.Jacobs, "Adaptive mixtures of local experts," *Neural Computa.,* vol.3, pp.79-87, 1991.
4. A.Sudjianto, M.H.Hassoun, "Nonlinear Hebbian rule: A statistical interpretation," In: *Proc. IEEE Intl. Conf. Neural Networks,* pp. 1247-1252, 1994
5. J.Karhunen, J.Joutsensalo, "Representation and separation of signals using nonlinear PCA type learning," *Neural Networks,* vol.7, pp.113-127, 1994.
6. J.Karhunen, P.Rajunen and E.Oja, "The nonlinear PCA criterion in blind source separation: relations with other approaches," *Technical Report,* Helsinki University of Technology, 1998.
7. C.Fyfe, R.Baddeley, "Finding compact and sparse-distributed representations of visual images," *Network: Computation in Neural Systems,* vol.6, pp.333-344, 1995.
8. C.Fyfe, R.Baddeley, "Nonlinear data structure extraction using simple Hebbian networks," *Biol. Cybern.,* vol.72, pp.533-541, 1995.
9. T.Martinetz, K.Schulten, " 'Neural gas' network learns topologies," in Kohonen et al. (Eds.),*Artificial neural networks,* (vol.I, pp.397-402). Amsterdam: North Holland, 1991.
10. K.Rose, F.Gurewitz and G.Fox, "Statistical mechanics and phase transitions in clustering," *Physical Rev. Lett.,* vol.65, pp. 945-948, 1990.
11. L.Xu, "A modified gating network for the mixtures of experts architecture," in *World Congress on Neural Networks,* San Diego, pp.II.405-410, 1994.
12. B.L.Zhang, L.Xu and M.Y.Fu, "Learning multiple causes by competition enhanced least mean square error reconstruction," *Intl. J. Neural System,* vol. 7, pp. 223-236, 1996.
13. C.J.S. Webber, "Emergent componential coding of a handwritten image database by neural self-organization," *Network: Computation in Neural System,* vol.9, pp. 433-447, 1998.