ϵ-insensitive Unsupervised Learning

Colin Fyfe and Bogdan Gabrys,
Applied Computational Intelligence Research Unit,
Department of Computing and Information Systems,
The University of Paisley,
Scotland.

Abstract

One of the major paradigms for unsupervised learning in Artificial Neural Networks is Hebbian learning. The standard implementations of Hebbian learning are optimal under the assumptions of Gaussian noise in a data set. We derive ϵ-insensitive Hebbian learning based on minimising the least absolute error in a compressed data set and show that the learning rule is equivalent to the Principal Component Analysis (PCA) networks' learning rules under a variety of conditions.

1 Introduction

The basis of many unsupervised learning rules is Hebbian learning which is so called after Donald Hebb [9] who conjectured *"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."* This paper investigates a novel implementation of Hebbian learning

Neural nets which use Hebbian learning are characterised by making the activation of a unit depend on the sum of the weighted activations which feed into the unit. They use a learning rule for these weights which depends on the strength of the simultaneous activation of the sending and receiving neuron. These conditions are usually modelled as

$$y_i = \sum_j w_{ij} x_j \tag{1}$$

$$\text{and } \Delta w_{ij} = \eta x_j y_i \tag{2}$$

the latter being the Hebbian learning mechanism. Here y_i is the output from neuron i, x_j is the j^{th} input, and w_{ij} is the weight from x_j to y_i. η is known as the learning rate and is usually a small scalar which may change with time. We see that the learning mechanism says that if x_j and y_i fire simultaneously, then the weight of the connection between them will be strengthened in proportion to their strengths of firing.

Substituting (1) into (2), we can write the Hebb learning rule as

$$\Delta w_{ij} = \eta x_j \sum_k w_{ik} x_k = \eta \sum_k w_{ik} x_k x_j \tag{3}$$

It is this last equation which gives Hebbian learning its ability to identify the correlations in a data set.

Now it is well known that the simple Hebbian rule above is unstable in that repeated use causes the weights to increase without bounds. Thus weight change rules which have an inbuilt decay term in them have been developed. Many of the most significant of these have been those developed by Oja and colleagues

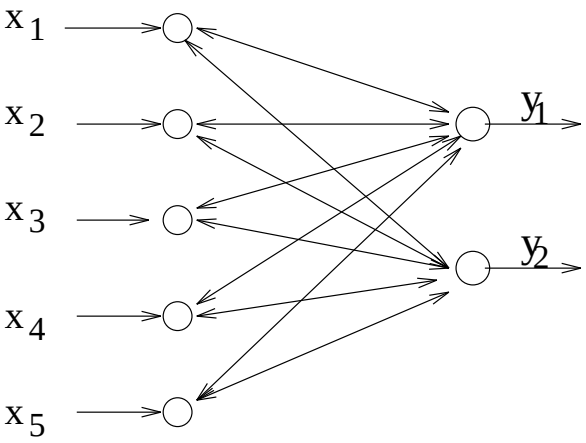

Figure 1: The negative feedback network. Activation transfer is fedforward and summed and returned as inhibition. Adaption is performed by simple Hebbian learning.

[13, 14, 15] which have been shown to not only cause the weights to converge but in particular, to converge so that the network is performing a Principal Component Analysis (PCA) of the data set. It is well known that PCA is the best linear compression of a data set in that it minimises the mean squared error between the compressed data and the original data set. An alternative definition is that PCA provides the linear basis of the data set that captures most variance. This basis is formed from the eigenvectors of the covariance matrix of the data set in order of largest eigenvalues.

In section 2, we describe a negative feedback implementation of Oja's rules and derive a new ϵ-insensitive form of Hebbian learning. In section 3, we experiment with the rule and show that it performs an approximation to Principal Component Analysis and that it is more robust in the presence of shot noise; we also illustrate the effectiveness of the method in an anti-Hebbian rule and in a topology preserving network.

2 The Negative Feedback Network and Cost Functions

Figure 1 shows the network which we have shown to perform a PCA [5] : the data is fed forward from the input neurons (the x-values) to the output neurons. Here the weighted summation of the activations is performed and this is fed back via the same weights and used in the simple Hebbian learning procedure. Consider a network with N dimensional input data and having M output neurons. Then the i^{th} output neuron's activation is given by

$$y_i = \mathrm{act}_i = \sum_{j=1}^{N} w_{ij} x_j \tag{4}$$

with the same notation as before. This firing is fed back through the same weights as inhibition to give

$$x_j(t+1) \leftarrow x_j(t) - \sum_{k=1}^{M} w_{kj} y_k \tag{5}$$

where we have used (t) and $(t+1)$ to differentiate between activation at times t and $t+1$. Now simple Hebbian learning between input and output neurons gives

$$\Delta w_{ij} \quad = \quad \eta_t y_i x_j(t+1) = \eta_t y_i \{x_j(t) - \sum_{l=1}^{M} w_{lj} y_l\} \tag{6}$$

where η_t is the learning rate at time t. This network has previously been shown in [18, 19, 5] to perform a Principal Component Analysis (PCA). In [5], we showed that we may have different weights for feeding back activation from those used for feeding forward activation which accords better with biological neurons in that synapses are one-directional. This network actually only finds the subspace spanned by the Principal Components; we can find the actual Principal Components by introducing some asymmetry into the network [6].

We have previously [8] introduced nonlinearity after the feedback operation in the learning rule to give

$$\Delta w_{ij} \quad = \quad \eta_t f(y_i) x_j(t+1) = \eta_t f(y_i)\{x_j(t) - \sum_{l=1}^{M} w_{lj}\, y_l\}$$

and related the resulting network to Exploratory Projection Pursuit. We will in section 4 be more interested in the addition of nonlinearity before the feedback,

$$y_i = f(\mathrm{act}_i) = f(\sum_{j=1}^{N} w_{ij} x_j) \tag{7}$$

followed by (5) and (6). [10] have shown that learning rule (6) may be derived as an approximation to gradient descent on the mean squared residuals (5) after feedback.

We can use the residuals (5) after feedback to define a general cost function associated with this network as

$$J = f_1(\mathbf{e}) = f_1(\mathbf{x} - W\mathbf{y}) \tag{8}$$

where in the above $f_1 = ||.||^2$, the (squared) Euclidean norm. It is well known (e.g. [17, 1]) that with this choice of $f_1()$ the cost function is minimised with respect to any set of samples from the data set on the assumption of Gaussian noise on the samples.

It can be shown that, in general (e.g. [17]), the mimimisation of J is equivalent to minimising the negative log probability of the error or residual, $\mathbf{e}$, which may be thought of as the noise in the data set. Thus if we know the probability density function of the residuals, we may use this knowledge to determine the optimal cost function.

It is well known that e.g. speech signals give a data set with kurtotic statistics. An approximation to these density functions is the (one-dimensional) function

$$p(e) = \frac{1}{2+\epsilon} \exp(-|e|_\epsilon) \tag{9}$$

where

$$|e|_\epsilon = \{ \begin{array}{cc} 0 & \forall |e| < \epsilon \\ |e| - \epsilon | & \text{otherwise} \end{array} \tag{10}$$

with ϵ being a small scalar ≥ 0. Using this model of the noise, the optimal $f_1()$ function (to minimise the negative log probability of the error) is the ϵ - insensitive cost function

$$f_1(e) = |e|_\epsilon \tag{11}$$

Therefore when we use this function in the (nonlinear) negative feedback network we get the learning rule

$$\Delta W \quad \propto \quad -\frac{\partial J}{\partial W} = -\frac{\partial f_1(e)}{\partial e}\frac{\partial e}{\partial W}$$

which gives us the learning rules

$$\Delta w_{ij} = \{ \begin{array}{cc} 0 & \text{if } |x_j - \sum_k w_{kj} y_k| < \epsilon \\ \eta.y_i.sign(x_j - \sum_k w_{kj} y_k) = \eta.y.sign(e) & \text{otherwise} \end{array} \tag{12}$$

where $sign(t) = 1$ if $t > 0$ and $sign(t) = -1$ if $t < 0$.

We see that this is a simplification of the usual Hebb rule using only the sign of the residual rather than the residual itself in the learning rule. We will find that in the linear case (section 3) allowing ϵ to be zero gives generally as accurate results as non-zero values but the data sets in section 4 require non-zero ϵ because of the nature of their innate noise.

2.1 Is this a Hebbian Rule?

The immediate question to be answered is "does this learning rule qualify as a Hebbian learning rule given that the term has a specific connotation in the Artificial Neural Networks literature?". We may consider e.g. covariance learning [12] to be a different form of Hebbian learning but at least it still has the familiar product of inputs and outputs as a central learning term.

The first answer to the question is to compare what Hebb wrote with the equations above. We see that Hebb is quite open about whether the pre-synaptic or the post-synaptic neuron would change (or both) and how the mechanism would work. Indeed it appears that Hebb considered it unlikely that his conjecture could ever be verified (or falsified) since it was so indefinite[4]. Secondly, it is not intended that this method replaces traditional Hebbian learning but that it coexists as a second form of Hebbian learning. This is biologically plausible -*"Just as there are many ways of implementing a Hebbian learning algorithm theoretically, nature may have more than one way of designing a Hebbian synapse"*[2]. Indeed the suggestion that Long Term Potentiation *"seems to involve a heterogeneous family of synaptic changes with dissociable time courses"* seems to favour the coexistance of multiple Hebbian learning mechanisms which learn at different speeds.

We now demonstrate that this new simplified Hebbian rule performs an approximation to Principal Component Analysis in the linear case and finds independent components when we have nonlinear activation functions.

3 ϵ-insensitive Hebbian Learning

In this section we will use the linear neural network

$$
\begin{aligned}
y_i &= \sum_{j=1}^{N} w_{ij} x_j \\
x_j(t+1) &\leftarrow x_j(t) - \sum_{k=1}^{M} w_{kj} y_k \\
\Delta w_{ij} &= \begin{cases} 0 & \text{if } |x_j(t+1)| < \epsilon \\ \eta . y_i . sign(x_j(t+1)) & \text{otherwise} \end{cases}
\end{aligned}
$$

and show that the network converges to the same values to which the more common PCA rules [14, 16] converge.

3.1 Principal Component Analysis

To demonstrate PCA, we use the ϵ-insensitive rules on artificial data. When we use the above learning rules on Gaussian data, we find an approximation to a PCA being performed. The weights shown in Table 1 are from an experiment in which the input data was chosen from zero mean Gaussians in which the first input has the smallest variance, the second the next smallest and so on. Therefore the first Principal Component direction is a vector with zeros everywhere except in the last position which will be a 1 so identifying the filter which minimises the mean square error (which is equivalent to maximising the variance in the projection of the data onto this filter). In our experiment, we have three outputs and five inputs; the weight vector has

2.0749643e-003	6.1149565e-002	**-3.9888122e-001**	**3.6858096e-001**	**-8.3912233e-001**
2.8296234e-002	7.7505112e-003	**8.5505936e-001**	**4.7678573e-001**	**-1.9534208e-001**
5.0981820e-003	-2.5554618e-002	**3.2719504e-001**	**-7.9844131e-001**	**-5.0683308e-001**

Table 1: The subspace spanned by first three principal components is captured after only 5000 iterations. ϵ =0.1

-1.7574163e-002	3.3526187e-002	2.6317708e-002	4.9533961e-002	**1.0068893e+000**
-1.7960927e-002	-2.0583177e-002	2.1191622e-002	**-1.0037621e+000**	6.8503537e-003
-4.1927978e-003	3.6645443e-002	**-9.9189108e-001**	-3.5904231e-002	6.6094374e-002

Table 2: The actual principal components are captured after only 5000 iterations. ϵ =0.5

converged to an orthonormal basis of the principal subspace spanned by the first three principal components: all weights to the inputs with least variance are an order of magnitude smaller than those to the three inputs with most variance. This experiment used 5000 presentations of samples from the data set, $\epsilon = 0.1$ and the learning rate was initially 0.01 and was annealed to 0 during these 5000 iterations.

Just as Oja's Subspace Rule may be transformed into a PCA rule by using deflationary techniques [16] we may find the actual Principal Components by using a deflationary rule with this Hebbian learning. Thus the feedforward rule is as before but feedback and learning occur for each output neuron in turn.

$$\begin{aligned} x_j(t+1) &\leftarrow x_j(t) - w_{kj} y_k \qquad \forall j \\ \Delta w_{kj} &= \eta_t y_k x_j(t+1) \qquad \forall j \end{aligned}$$

for $k = 1, 2....$

Table 2 shows the results when 5 dimensional data of the same type as before was used as input data, the learning rate was 0.1 decreasing to 0 and ϵ was 0.1. These results were taken after only 5000 iterations. The convergence is very fast: a typical set of results from the same data are shown in Table 3 where the simulation was run over only 1000 presentations of the data (We have also used ϵ=0 in this case to demonstrate that the particular value of ϵ is not crucial). So, as might have been expected, minimising the mean square error and minimising the mean absolute error give the same results in the Gaussian case.

We may expect that since the learning rule is insensitive to the magnitude of the input vectors $\mathbf{x}$, the rule is less sensitive to outliers than the usual rule based on mean square error. To test this, we add noise from a uniform distribution in [-10,10] to the last input (that with smallest variance) in 30% of the presentations of the input data. Table 4 shows that the PCA properties of the ϵ-insensitive deflationary network are unaffected by the noise. In comparison, Table 5 shows that the Sanger [16] network responds to this noise (as one would expect).

We note that this need not be a good thing, however in the context of real biological neurons we may wish each individual neuron to ignore high intensity shot noise and so the ϵ-insensitive rule may be optimal. Finally the insertion of a differentiated θ_i term in the calculation of the residual (as in [15]) also causes convergence to the actual Principal Components but was found to be two orders of magnitude slower than the deflationary technique described above.

-5.1776166e-003	4.5376865e-002	-2.3840385e-003	1.2658529e-002	**1.0019231e+000**
-3.6847661e-002	1.0566274e-002	-1.3372074e-001	**9.9144360e-001**	1.6136625e-003
-1.1309247e-002	-8.1029262e-003	**9.9036232e-001**	1.4140185e-001	-1.4256455e-003

Table 3: The actual principal components are almost found after only 1000 iterations. ϵ =0.

2.5339642e-002	-3.9976042e-002	4.7171348e-002	3.3109733e-002	**9.9560379e-001**
1.5587732e-002	1.4625496e-002	9.7532991e-003	**-1.0013772e+000**	3.2922597e-002
-4.8078854e-002	-6.4830431e-002	**9.9738211e-001**	2.1815735e-002	-4.5000940e-002

Table 4: The 30% outliers are ignored by the ϵ-insensitive rule.

6.5436571e-002	-4.2781770e-002	-3.5680891e-002	5.9717521e-002	**-9.9664319e-001**
6.9542470e-002	-1.4830770e-002	1.7929844e-002	**-9.9627047e-001**	-7.1398709e-002
-9.5947374e-001	-1.7583321e-003	-2.1653710e-001	-9.4726964e-002	-1.1765086e-001

Table 5: The standard Sanger rule finds the noise irresistable.

3.2 Anti-Hebbian Learning

Now the ϵ-insensitive rule was derived in the context of the minimisation of a specific function of the residual. It is perhaps of interest to enquire whether similar rules may be used in other forms of Hebbian learning. We investigate this using anti-Hebbian learning.

Földiák [3] has suggested a neural net model which has anti-Hebbian connections between the output neurons.

The equations which define its dynamical behaviour are

$$y_i = x_i + \sum_{j=1}^{N} w_{ij} y_j$$

In matrix terms, we have

$$\begin{aligned} \mathbf{y} &= \mathbf{x} + \mathbf{W}\mathbf{y} \\ \text{And so, } \mathbf{y} &= (\mathbf{I} - \mathbf{W})^{-1}\mathbf{x} \end{aligned}$$

He shows that, after training with the familiar anti-Hebbian rule,

$$\Delta w_{ij} = -\eta y_i y_j \text{ for } i \neq j$$

the outputs, $\mathbf{y}$ are decorrelated.

Now the matrix W must be symmetric and has only non-zero non-diagonal terms i.e. if we consider only a two input, two output net,

$$W = \begin{pmatrix} 0 & w \\ w & 0 \end{pmatrix} \tag{13}$$

However the ϵ-insensitive anti-Hebbian rule is non-symmetrical since, if w_{ij} is the weight from y_i to y_j, we have

$$\begin{aligned} \Delta w_{ij} &= -\eta y_j sign(y_i) \text{ if } |y_i| > \epsilon \\ \Delta w_{ij} &= 0 \qquad \text{otherwise} \end{aligned} \tag{14}$$

To test the method, we generated 2 dimensional input vectors, $\mathbf{x}$, where each element is drawn independently from N(0,1) and then added another independently drawn sample from N(0,1) to both elements. This gives a data set with sample covariance matrix (10000 samples) of

$$\begin{pmatrix} 1.9747 & 0.9948 \\ 0.9948 & 1.9806 \end{pmatrix} \tag{15}$$

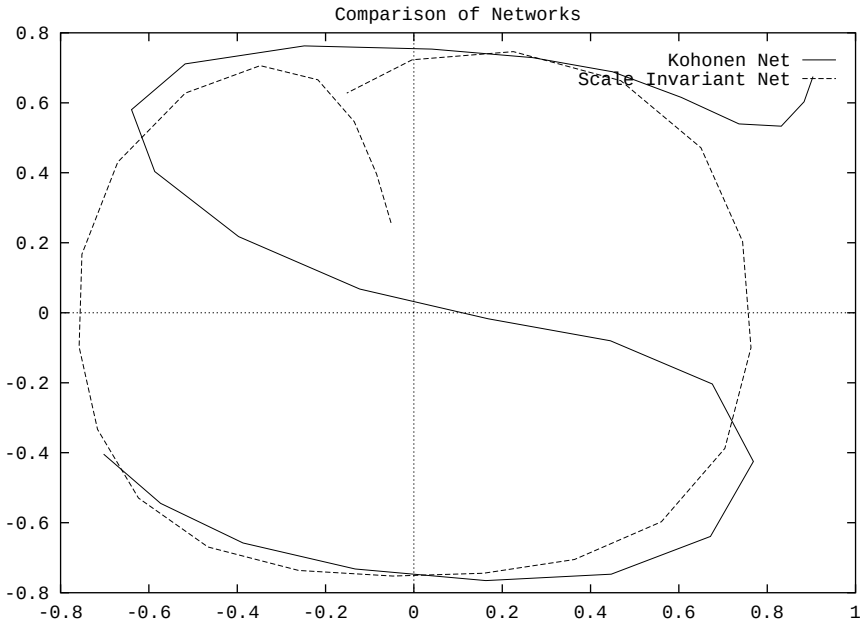

Figure 2: The Kohonen mapping and the Scale Invariant mapping on two dimensional uniform distribution.

The covariance matrix of the outputs, $\mathbf{y}$, (over the 10000 samples) from the network trained using the ϵ-insensitive learning rule is

$$\begin{pmatrix} 1.8881 & -0.0079 \\ -0.0079 & 1.1798 \end{pmatrix} \tag{16}$$

We see that the outputs are almost decorrelated. It is interesting to note that

- the asymmetrical learning rules have resulted in non-equal variances on the outputs.
- but the covariance (off-diagonal) terms are equal.

It is our finding that the outputs are always decorrelated but the final values on the diagonals (the variances) are impossible to predict and seem to depend on the actual values seen in training, the initial conditions etc..

A feedforward decorrelating network, $\mathbf{y} = (I + W)\mathbf{x}$, may also be created with the ϵ-insensitive anti-Hebbian rule with similar resuts.

3.3 Topology Preserving Maps

We have also used the negative feedback network to create topology preserving maps [7] with somewhat different properties from Kohonen's SOM [11]. The feedforward stage is as before but now we invoke a competition between the output neurons and the neuron with greatest activation wins. The winning neuron, the p^{th}, is deemed to be maximally firing (=1) and all other output neurons are suppressed(=0). Its firing is then fed back through the same weights to the input neurons as inhibition.

$$x_j(t+1) \leftarrow x_j(t) - w_{pj}.1 \text{ for all } j \tag{17}$$

where p is the winning neuron. Now the winning neuron excites those neurons close to it i.e. we have a neighbourhood function $\Lambda(p,j)$ which satisfies $\Lambda(p,j) \leq \Lambda(p,k)$ for all $j,k : \| p-j \| \geq \| p-k \|$ where $\| . \|$ is the Euclidean norm. Typically, we use a Gaussian whose radius is decreased during the course of the simulation. Then simple Hebbian learning gives

$$\begin{aligned} \Delta w_{ij} &= \eta_t \Lambda(p,i).x_j(t+1) & (18) \\ &= \eta_t \Lambda(p,i).(x_j(t) - w_{pj}) & (19) \end{aligned}$$

The somewhat different properties of this mapping from the Kohonen mapping may be seen in Figure 2: the Kohonen SOM spreads out evenly across the data set while with the scale invariant mapping, each output neuron captures a slice of the inputs of approximately equal magnitude angle. We may now report that if we use the ϵ-insensitive learning rule

$$\Delta w_{ij} = \begin{cases} 0 & \text{if } |x_j(t+1)| < \epsilon \\ \eta.\Lambda(p,i).sign(x_j(t+1)) & \text{otherwise} \end{cases} \quad (20)$$

convergence to the same type of mapping is also achieved. As before, the resultant mapping is found more quickly and is more robust against shot noise.

4 Conclusion

We have derived a slightly different form of Hebbian learning which we have shown capable of performing PCA type learning in a linear network. We have shown that the method may also be used for anti-Hebbian learning and for the creation of topology preserving maps.

Future work will concentrate on analysing the form of noise to be expected in real situations and creating cost functions which are optimal for these situations.

References

[1] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford:Clarendon Press, 1995.

[2] T. H. Brown and S. Chatterji. *The Handbook of Brain Theory and Neural Networks*, chapter Hebbian Synaptic Plasticity. MIT Press, 1995.

[3] P. Földiák. *Models of Sensory Coding.* PhD thesis, University of Cambridge, 1992.

[4] Y. Fregnac. *The Handbook of Brain Theory and Neural Networks*, chapter Hebbian Synaptic Plasticity: Comparative and Developmental Aspects. MIT Press, 1995.

[5] C. Fyfe. Pca properties of interneurons. In *From Neurobiology to Real World Computing, ICANN 93*, pages 183–188, 1993.

[6] C. Fyfe. Introducing asymmetry into interneuron learning. *Neural Computation*, 7(6):1167–1181, 1995.

[7] C. Fyfe. A scale invariant feature map. *Network: Computation in Neural Systems*, 7:269–275, 1996.

[8] C. Fyfe and R. Baddeley. Non-linear data structure extraction using simple hebbian networks. *Biological Cybernetics*, 72(6):533–541, 1995.

[9] D. O. Hebb. *The Organisation of Behaviour.* Wiley, 1949.

[10] Juha Karhunen and Jyrki Joutsensalo. Representation and separation of signals using nonlinear pca type learning. *Neural Networks*, 7(1):113–127, 1994.

[11] Tuevo Kohonen. *Self-Organising Maps*. Springer, 1995.

[12] R. Linsker. From basic network principles to neural architecture. In *Proceedings of National Academy of Sciences*, 1986.

[13] E. Oja. A simplified neuron model as a principal component analyser. *Journal of Mathematical Biology*, 16:267–273, 1982.

[14] E. Oja. Neural networks, principal components and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[15] E. Oja, H. Ogawa, and J. Wangviwattana. Pca in fully parallel neural networks. In Aleksander & Taylor, editor, *Artificial Neural Networks,2*, 1992.

[16] T.D. Sanger. Analysis of the two-dimensional receptive fields learned by the generalized hebbian algorithm in response to random input. *Biological Cybernetics*, 1990.

[17] A. J. Smola and B. Scholkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2 Technical Report Series, October 1998.

[18] R. J. Williams. Feature discovery through error-correcting learning. Technical Report 8501, Institute for Cognitive Science, University of California, San Diego, 1985.

[19] Lei Xu. Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Networks*, 6(5):627 – 648, 1993.