

Optimising Local Hebbian Learning: use the δ -rule

J.W.M. van Dam*, B.J.A. Kröse, F.C.A. Groen
Faculty of Mathematics and Computer Science
University of Amsterdam
Kruislaan 403
NL - 1098 SJ Amsterdam

1 Introduction

We consider problems where the *correlation* between signals $\mathbf{x} = (x_1, \dots, x_n)^T$ and $\mathbf{y}^* = (y_1^*, \dots, y_m^*)^T$ is to be learned supervisedly¹. This is useful in, e.g., sensor fusion applications, to correlate different representations of the same scene (see, e.g., [Gielen et al., 91]). Traditionally, *local Hebbian learning* is suggested to approach these problems. Here, it is proven, that if continuous-valued learning samples are available, the δ -rule gives better performance. For binary-valued learning samples, it is shown that the Hebb rule does not converge to the optimal weight values. Therefore, a modification to the Hebb rule is introduced and it is shown that the δ -rule is to be modified similarly.

Problems of the kind mentioned above can be approached with a single layer feed-forward neural network with input units x_1, \dots, x_n and output units y_1, \dots, y_m . Each unit y_i is connected to all input units x_j with weights w_{ij} . The optimal weight values w_{ij}^* are given by the correlations:

$$w_{ij}^* = E[y_i^* x_j] = \int \int y_i^* x_j f(y_i^*, x_j) dy_i^* dx_j \quad (1)$$

In this article, we compare two different learning algorithms to train such a network: the Hebb rule and the δ -rule.

2 Continuous-valued learning samples

We assume learning samples $(\mathbf{y}^*, \mathbf{x})$ are available where the inputs \mathbf{x} are uncorrelated, $E[x_j x_k] = 0$ for $k \neq j$, with zero mean, $E[x_j] = 0$, and with variance $\alpha_j = \int x_j^2 f(x_j) dx_j$. The outputs \mathbf{y}^* are characterized by:

$$E[\mathbf{y}^*] = \mathbf{H}\mathbf{x}$$

*The investigations were supported by the Foundation for Computer Science in the Netherlands (SION) with financial support from the Netherlands Organisation for Scientific Research (NWO).

¹We write \mathbf{y}^* for the "optimal" or "true" signal, whereas \mathbf{y} denotes an estimation of this signal.

where $\mathbf{H} = \mathbf{R}_{yx} \cdot (\mathbf{R}_{xx})^{-1}$, $\mathbf{R}_{yx} = E[\mathbf{y}^* \mathbf{x}^T]$, the correlation matrix, and $\mathbf{R}_{xx} = \alpha \mathbf{I}$, the autocorrelation matrix.

The Hebb rule initializes all weights to 0 and updates according to:

$$\Delta w_{ij} = \frac{1}{N} y_i^* x_j \quad (2)$$

with N the number of learning samples. Therefore, the network converges to

$$\lim_{N \rightarrow \infty} w_{ij} = \mathcal{E}[y_i^* x_j] = \int \int y_i^* x_j f(\mathbf{y}^*, \mathbf{x}) d\mathbf{y}^* d\mathbf{x} \quad (3)$$

The value $\mathcal{E}[y_i^* x_j]$ is an *estimation* of the true correlation $E[y_i^* x_j]$ and this is not necessarily a correct estimation since the Hebbian algorithm estimates all $m \times n$ correlations in parallel. We have

$$\mathcal{E}[y_i^* x_j] = \int \int y_i^* x_j f(\mathbf{y}^*, \mathbf{x}) d\mathbf{y}^* d\mathbf{x} = \dots = \int x_j f(x_j) \mathcal{E}[y_i^* | \mathbf{x}] dx_j$$

with

$$\mathcal{E}[y_i^* | \mathbf{x}] = \mathcal{E}\left[\frac{1}{\alpha_j} \sum_j w_{ij}^* x_j | \mathbf{x}\right] = \frac{1}{\alpha_j} w_{ij}^* x_j + \mathcal{E}\left[\sum_{k \neq j} \frac{1}{\alpha_k} w_{ik}^* x_k | \mathbf{x}\right] = \frac{1}{\alpha_j} w_{ij}^* x_j$$

which gives $\mathcal{E}[y_i^* x_j] = w_{ij}^*$; the algorithm converges to the optimal value.

Similarly, we can also calculate the *variance* $var[y_i^* x_j]$. This value gives a measure of how close to our optimum we may expect our algorithm to converge. Naturally, this measure can also be calculated if instead of the Hebb-rule, the δ -rule is applied. If linear activation values are used, the δ -rule takes

$$w_{ij}^{\text{new}} = w_{ij} + (y_i^* - y_i) x_j$$

and the variance in the δ -rule is given by (see [Hammersley, Handscomb (1965)]):

$$var[w_{ij} + y_i^* x_j - y_i x_j] = var[y_i^* x_j] + var[y_i x_j] - 2cov[y_i^* x_j, y_i x_j] \quad (4)$$

So, if we can prove that $2cov[y_i^* x_j, y_i x_j] > var[y_i x_j]$, we may expect the δ -rule to converge closer to the optimal weight values than the Hebbian learning rule. We take for the activation function $y_i = \sum_j \frac{w_{ij}}{\alpha_j} x_j$. The calculation of the terms in (4) is straightforward, and differs little from the calculation of $\mathcal{E}[y_i^* x_j]$. The results are, that if $\forall i, j \quad 2|w_{ij}^*| > |w_{ij}|$, we have $2cov[y_i^* x_j, y_i x_j] > var[y_i x_j]$. And thus, if we are careful with the initialisation of the weights w_{ij} , the δ -rule can be expected to give better performance than the Hebb-rule.

3 Binary-valued Learning samples

In some cases, we may only have binary-valued, non-linear learning samples (see [Van Dam et al., (1993)] for an example). In this case, the correlation between y_i^* and x_j is interpreted as:

$$w_{ij}^* = E[y_i x_j] = P(y_i = 1 | x_j = 1)$$

We assume that all possible patterns are equally distributed, i.e., $P(x_j = 1) = \frac{1}{2}$.

For the Hebb rule, we take $\forall i, j \ y_i^*, x_j \in \{-1, +1\}$. The algorithm converges to:

$$\mathcal{E}[y_i^* x_j] = \sum_{a=+1,-1} \sum_{b=+1,-1} a \cdot b \cdot P(x_j = a \ \& \ y_i^* = b)$$

Using elementary probability theory, we obtain:

$$\mathcal{E}[y_i^* x_j] = w_{ij}^* u_{ij}^*$$

where

$$u_{ij}^* = P(y_i^* \text{ not set to } +1 \text{ by any } x_k, k \neq j) = \prod_{k \neq j} (1 - w_{ik}^* \tilde{x}_k)$$

with $\tilde{x}_k = \frac{x_k + 1}{2}$ such that $\tilde{x}_k \in \{0, 1\}$. This follows directly from the definition of the learning samples.

In other words, the Hebb rule does not converge to the optimal weight value. We can, however, formulate a modified Hebbian Learning rule, which is applicable to binary-valued learning samples, by simply modulating (2) with $1/u_{ij}^*$. We can calculate estimations of u_{ij}^* by using the current weight values w_{ij} . However, the Hebb rule is very sensitive to bad estimations.

If we want to train the network with the δ -rule, we take $\forall i, j \ y_i^*, x_j \in \{0, 1\}$. The network activation function must be chosen to suit the definition of the learning samples:

$$y_i = P(y_i^* \text{ set to } +1 \text{ by at least one } x_j)$$

From combinatorial probability theory this probability is given by:

$$y_i = \sum_{k=1}^n (-1)^{k-1} \sum_{0 \leq j_1 < \dots < j_k \leq n} w_{ij_1} x_{j_1} \cdot \dots \cdot w_{ij_k} x_{j_k} = 1 - \prod_j (1 - w_{ij} x_j) \quad (5)$$

Now that we have a different activation function, the δ -rule gives:

$$\Delta w_{ij} = -\gamma \frac{\partial}{\partial w_{ij}} (y_i^* - y_i)^2 = 2\gamma (y_i^* - y_i) x_j \prod_{k \neq j} (1 - w_{ik} x_k) \quad (6)$$

Not surprisingly, the modified δ -rule introduces a modification which is similar to the proposed modification of the Hebb-rule.

We have compared the δ -rule to the Hebb rule in a similar way to section 2, but a fair comparison is difficult to make (see [Van Dam et al., (1994)]). Nevertheless, the δ -rule can be expected to give better performance since it is less sensitive to bad estimations of the modification term.

4 Implementation and results

We tested the algorithms presented in the previous sections on two grids \mathbf{y}^* and \mathbf{x} of 16×16 square cells each, which are rotated and translated with respect

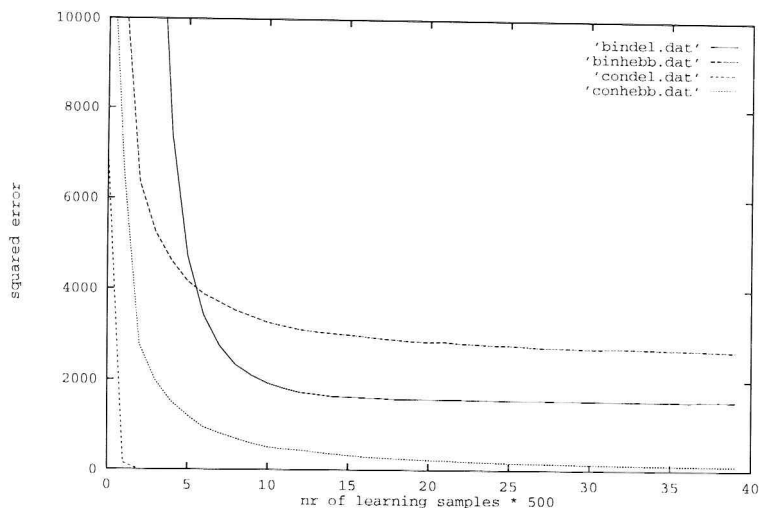


Figure 1: Comparing the δ -rule to the Hebbian learning rule. Plotted are the squared errors summed over all 256 output units and over 1000 continuous-valued test samples against the number of learning samples presented.

to each other. We define the correlation w_{ij}^* as the *intersection* of cell y_i^* with cell x_j . Continuous valued learning samples are constructed by taking random values for each x_j and by taking $y_i^* = \sum_j w_{ij}^* x_j$, where weight values w_{ij}^* are given by calculating the intersections of cells based on a-priori knowledge. To create binary patterns, we take a number of random points in the grid. We set $x_j = 1$ if a point is present somewhere in its cell. The same is done for values y_i^* . All values not set to 1 are set to -1 for the modified Hebb-rule and to 0 for the modified δ -rule. For the Hebb rule, we calculated u_{ij}^* using w_{ij}^* (which makes the Hebb rule look good). Note that to create binary patterns, we don't need a-priori knowledge on the values of w_{ij}^* . Results for all four learning rules are given in figure 1. Both if continuous valued learning samples are available and if only binary samples are available, the δ -rule gives better performance.

References

- [Gielen et al., 91] C.Gielen, K.Krommenhoek, J. van Gisbergen (1991), "A procedure for self-organised sensor-fusion in topologically ordered maps". Proceedings of the Second International Conference on Autonomous systems, 417-423.
- [Hammersley, Handscomb (1965)] J.M. Hammersley, D.C. Handscomb (1965), "Monte Carlo Methods", 50-76.
- [Van Dam et al., (1993)] J.W.M. van Dam. B.J.A. Kröse, F.C.A. Groen (1993), "Transforming Occupancy Grids Under Robot Motion", ICANN93 Amsterdam, the Netherlands, 318.
- [Van Dam et al., (1994)] J.W.M. van Dam. B.J.A. Kröse, F.C.A. Groen (1994), "On Local Hebbian Learning", internal report.