A Generalized Hebb (GH) rule based on a cross-entropy error function for deep belief recursive learning

Mark J. Embrechts and Bernhard Sick

***Abstract*—** The purpose of this article is to show a novel learning rule derived from the Hebb rule, for the training of recursive deep belief networks. The derivation is de novo, elegant, and similar to Hinton's Contrastive Divergence CD-N, where N is the number of recursions (e.g., 1 in a traditional neural multi-layered perceptron). This rule is introduced in this article as "*Generalized Hebb rule*" (GH-N) for an entropic cost function for deep belief recursive learning. This rule is important because: (i) It is easy to apply, (ii) it applies to a stacked deep belief network, (iii) Hinton's Contrastive Divergence rule CD-N for continuous units is a special case of this rule, and (iv) preliminary experimental results show that – for binary patterns – a deep belief auto-associator trained with a recursive neural network often shows a clearer separation of classes in the bottleneck layer than trained with backpropagation or compared to principal component analysis.

***Keywords*—**deep belief network, Generalized Hebb rule, contrastive divergence, auto-associator.

I. MOTIVATION

DEEP belief networks (DBN) [1-2] have rejuvenated interest in artificial neural networks, but are still hard to grasp for novices in artificial neural networks. DBN are basically a stacking of layers of neurons and can be trained layer by layer using Restricted Boltzmann Machines (RBM) [3-6]. The purpose of this article is to introduce and derive a novel training rule, the *Generalized Hebb rule (GH-N)*, for recursive deep belief stacked auto-associators, which can also be applied to deep belief networks. The resulting training rule has a strong similarity with Hinton's Contrastive Divergence rule (CD-N) [6-8] but applies directly to continuous units [16] as well, and does not need the Boltzmann type of "stochasticity" to interpret the firing of a neuron. This rule is derived de novo, starting from the Hebb rule, and applies to a recursive single layer of a stacked auto-associator. Preliminary tests using the Italian olive oil data [11-12] show that the bottleneck neuron outputs of a deep belief recursive stacked auto-associator for binary and multi-class classification patterns show a clearer separation on the test data than a deep belief network trained with backpropagation based on LeCun's Efficient BackProp [9-10].

This work was supported in part by the German Federal Ministry of Education and Research under Grant 03EK3536A (PrIME project).

M. J. Embrechts is with CardioMag Imaging, Inc., 13 British American Boulevard, Latham, NY 12110, U.S.A. (mark.embrechts@gmail.com).

B. Sick is with the University of Kassel, Germany, Electrical Engineering and Computer Science Department, Intelligent Embedded Systems Lab, Wilhelmshoeher Allee 71-73, 34121 Kassel, Germany (bsick@uni-kassel.de).

This article is organized as follows: Section II shows an intuitive derivation of the GH-N algorithm; Section III addresses how this rule can be implemented for a stacked auto-associator; Section IV discusses preliminary experiments; and Section V summarizes the key findings and gives an outlook to future work.

II. DEEP BELIEF AUTO-ASSOCIATIVE NEURAL NETWORKS

A deep belief auto-associative neural network is an auto-associator [13-15] with many layers, usually with symmetric weights, and trained with a deep belief method. An auto-associator can be regarded as an artificial neural network, where the output values (i.e., the target values) are exactly the same as the input values. Such an auto-associator has many layers of neurons and a bottleneck layer.

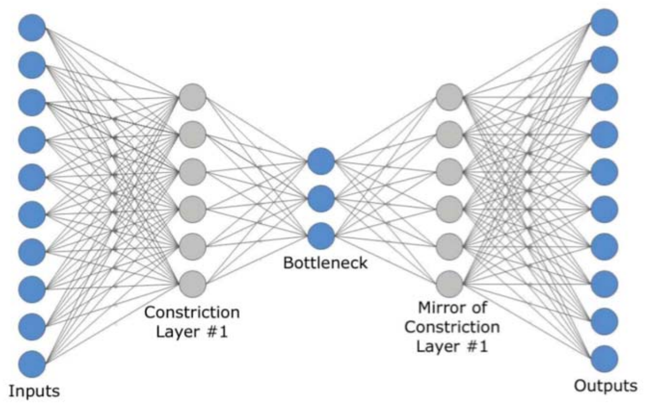


Figure 1. Example of a deep auto-associator where the target outputs are the same as the inputs. Except for the input layer, all other elements represented by a circle are artificial neurons. It can be shown that the weights are symmetric. [21]

The auto-associator depicted in Fig. 1 has a symmetric structure and in this case, also symmetric weights. Usually the bottleneck layer is often of special interest and can be used in a similar manner as principal components and/or independent components. Fig. 1 represents the scheme of a deep auto-associator, where the outputs are the same as the inputs. Except for the input layer, all other elements represented by a circle are artificial neurons (i.e., first computing a weighted sum of the inputs, and then applying a nonlinear activation function: typically a sigmoid or a hyperbolic tangent function).

III. DERIVATION OF THE GENERALIZED HEBB RULE

It is in principle possible to train an auto-associative network via the backpropagation algorithm [17], and a deep auto-associative network via Efficient BackProp [9-10]. However, a stepwise building up of a deep belief auto-associative network [10] can easier avoid that neurons get into saturation and possibly reduce the training time, too. A stepwise deep belief auto-associator can be trained – layer by layer – by Hinton's Contrastive Divergence rule CD-N [7-8], or by our new Generalized Hebb rule, GH-N.

The derivation of the Generalized Hebb rule will proceed as follows: (i) First we will derive the delta rule as an extension of the Hebb rule; (ii) using this rule we will derive the training rule for a single layer of a symmetric recursive auto-associator; (iii) then we will derive the training rule of a stacked symmetric auto-associator. A comparison of the GH-N rule with Hinton's CD-N will then be made.

A. A de novo derivation of the Widrow-Hoff Delta rule from Hebb's rule

The Widrow-Hoff Delta rule can be considered as an extension of Hebb's rule, which states: "*When an axon of a 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.*" [10, 18].

The above rule is a "carrot rule": i.e., good behavior gets rewarded. We can extend this to "a carrot and a stick" rule: i.e., good behavior gets rewarded and bad behavior gets punished.

Rosenblatt's single-layer Perceptron rule for binary patterns can be derived from that principle and written more elegantly for polar (i.e., [-1,1]) units. In this case, the output of a neural network is the weighted sum of the inputs (modified by a bias *b*) according to:

$$output = \sum_{i=1}^{m} w_i x_i + b.$$

Patterns are shown one-at-a-time. If a pattern is classified correctly, the weights are not modified. If a pattern is misclassified, one of the following rules is applied:

$$\vec{w}^{(N+1)} = \vec{w}^{(N)} - \eta^{(N)} \vec{x}^{(N)}$$
$$\vec{w}^{(N+1)} = \vec{w}^{(N)} + \eta^{(N)} \vec{x}^{(N)},$$

depending on whether x belongs to the negative or the positive class [10]. *N* indicates the update iteration level.
For continuous units and supervised learning both rules can be combined into the Widrow-Hoff delta rule

$$\vec{w}_{ji}^{(N+1)} = w_{ji}^{(N)} - \eta \delta_j \vec{x}_i,$$

where delta is the error. Here, we use the following notation: w_{ji} is a weight for the connection from neuron *i* on the input side to neuron *j* on the output side.

In the backpropagation algorithm δ' is used rather than δ, where $\delta' = (y_j - x_j).f'(x_j)$. In case just δ is used, this is also equivalent to a backpropagation rule where a different cost function than the least-squares error is applied, the bi-level entropic error function described by Baum [19,20]:

$$C(x,y) = -\sum_{\mu} \left[x^{\mu} \log\left(y^{\mu}\right) + \left(1 - x^{\mu}\right) \log\left(1 - y^{\mu}\right) \right].$$

B. Stacked (recursive) symmetric auto-associator

A stacked auto-associator with symmetric weights is shown in the left hand side of the figure below. In the unfolded recursive auto-encoder with shared weights the weights are not shown. For symmetric weights and three recursions in the auto-encoder the Widrow-Hoff delta rule with a bi-level entropic error function and three recursions the learning rule is shown below.

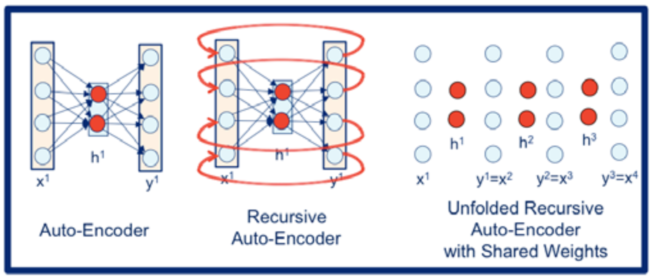


Figure 2. Recursive auto-associator with cross-entropic cost function, symmetric weights, weight sharing and delta rule through time approach with a 3-step example.

Note that we used the notation as explained in the right hand-side of Fig. 2, where the auto-encoder with three recursions is unfolded. We now use symmetric weights (i.e., only the weights of the second layer in the auto-encoder are trained, the weights in the first layer are just copied, using the weight symmetry property). Note also that we use weight sharing, which explains why the respective weight updates contain the factor 1/3.

$$\Delta w_{ji} = \frac{\eta}{3} \left\langle \left[h_i^{(1)} \left(x_j^{(1)} - y_j^{(1)} \right) \right] + \left[h_i^{(2)} \left(x_j^{(2)} - y_j^{(2)} \right) \right] + \left[h_i^{(3)} \left(x_j^{(3)} - y_j^{(3)} \right) \right] \right\rangle$$
$$\Delta w_{ji} = \frac{\eta}{3} \left\langle \left[h_i^{(1)} \left(x_j^{(1)} - x_j^{(2)} \right) \right] + \left[h_i^{(2)} \left(x_j^{(2)} - x_j^{(3)} \right) \right] + \left[h_i^{(3)} \left(x_j^{(3)} - y_j^{(3)} \right) \right] \right\rangle$$
$$\Delta w_{ji} = \frac{\eta}{3} \left\langle h_i^{(1)} x_j^{(1)} + \left[x_j^{(2)} \left(h_j^{(2)} - h_j^{(1)} \right) \right] + \left[x_j^{(3)} \left(h_j^{(3)} - h_j^{(2)} \right) \right] - h_i^{(3)} y_j^{(3)} \right\rangle$$
$$\Delta w_{ji} \cong \eta' \left\langle \left[x_j^{(1)} h_i^{(1)} - y_j^{(3)} h_i^{(3)} \right] \right\rangle$$

C. *Stacking several recursive symmetric auto-associators*

Fig. 3 expands to concept of Fig. 2 to an indefinite level of *K* recursions.

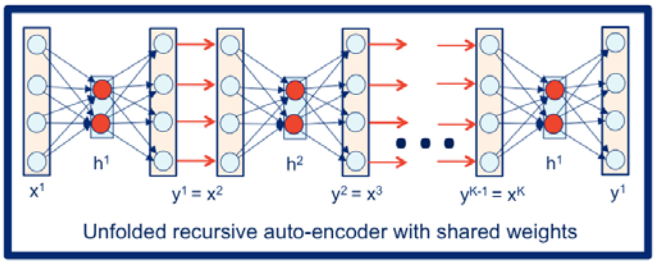


Figure 3. Recursive auto-associator with cross-entropy cost function, symmetric weights, weight sharing and delta rule through time approach (general ➔ *K* steps).

The learning formula can now be approximated as shown below. An explanation of the bracket notation is now in order. The bracket notation shows that all the patterns need to be applied. Note that we assumed in the notation a more or less convergent behavior from one recursive auto-associator to the next layer.

$$\Delta w_{ji} = \frac{\eta}{K}\left\langle\left[h_i^{(1)}\left(x_j^{(1)} - y_j^{(1)}\right)\right] + \left[h_i^{(2)}\left(x_j^{(2)} - y_j^{(2)}\right)\right] + \ldots + \left[h_i^{(K)}\left(x_j^{(K)} - y_j^{(K)}\right)\right]\right\rangle$$

$$\Delta w_{ji} = \frac{\eta}{K}\left\langle\left[h_i^{(1)}\left(x_j^{(1)} - x_j^{(2)}\right)\right] + \left[h_i^{(2)}\left(x_j^{(2)} - x_j^{(3)}\right)\right] + \ldots + \left[h_i^{(3)}\left(x_j^{(K)} - y_j^{(K)}\right)\right]\right\rangle$$

$$\Delta w_{ji} = \frac{\eta}{K}\left\langle h_i^{(1)} x_j^{(1)} + \left[x_j^{(2)}\left(h_j^{(2)} - h_j^{(1)}\right)\right] + \ldots\left[x_j^{(K)}\left(h_j^{(K)} - h_j^{(K-1)}\right)\right] - h_i^{(K)} y_j^{(K)}\right\rangle$$

$$\Delta w_{ji} \cong \eta'\left\langle\left[x_j^{(1)} h_i^{(1)} - y_j^{(K)} h_i^{(K)}\right]\right\rangle$$

$$\Delta w_{ji} \cong \eta'\left[\left\langle x_j^{(1)} h_i^{(1)}\right\rangle - \left\langle y_j^{(K)} h_i^{(K)}\right\rangle\right]$$

D. *The Generalized Hebb rule GH-N for deep belief auto-encoders*

In short, the Generalized Hebb rule for updating the weight w_{ji} can be written as:

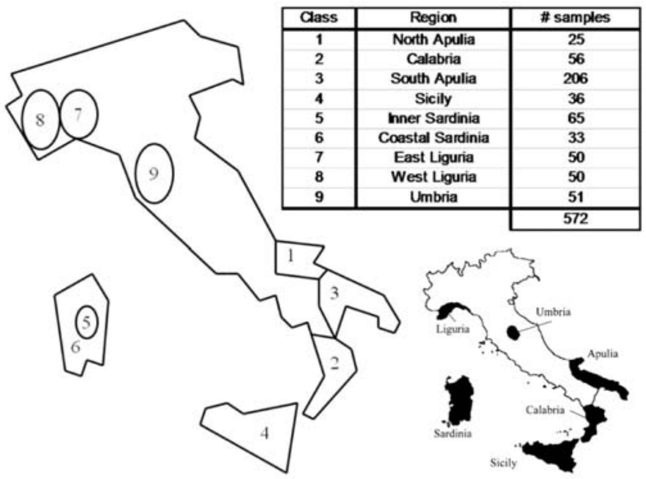
$$\Delta w_{ji}^{GH-K} = \eta \cdot$$
$$\left[\left\langle x_j^{(1)} h_i^{(1)}\right\rangle + \sum_{k=2}^{K-1}\left\langle x_j^{(k)}\left(h_j^{(k)} - h_j^{(k-1)}\right)\right\rangle - \left\langle y_j^{(K)} h_i^{(K)}\right\rangle\right]$$

Assuming that we are near a converging behavior for the output of the hidden layer, this rule can be approximated by the well-known Contrastive Divergence rule for continuous (non-stochastic) units:

$$\Delta w_{ji}^{CD-K} = \eta \cdot \left[\left\langle x_j^{(1)} h_i^{(1)}\right\rangle - \left\langle y_j^{(K)} h_i^{(K)}\right\rangle\right].$$

IV. PRELIMINARY RESULTS

We will illustrate this procedure on the Italian olive oil data [11-12]. In this case there are 572 olive oils (Fig. 4) from different regions in Italy, described by 8 different fatty acids.



Class	Region	# samples
1	North Apulia	25
2	Calabria	56
3	South Apulia	206
4	Sicily	36
5	Inner Sardinia	65
6	Coastal Sardinia	33
7	East Liguria	50
8	West Liguria	50
9	Umbria	51
		572

Figure 4. Presentation of 572 Italian olive oils. The olive oils are described by measures for 8 different fatty acids. Not that the 9 classes of olive oils are not balanced.

Fig. 5 shows several deep belief network results for the Italian olive oil data. Fig. 5a is equivalent to a principal component projection on the first two principal components, while 5b – 5c are results from a backpropagation algorithm for deep belief networks for different network structures. Even though in this particular case the results were obtained from applying the backpropagation algorithm without recursion, the GH-1 results are of a similar nature and also show a much clearer separation than the principal components. Note that the northern and southern Italian olive oils become more clearly separated the deeper the network is.

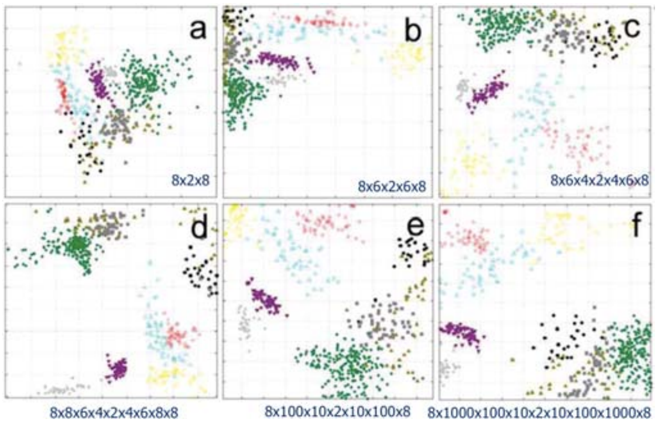


Figure 5. Projections in the bottleneck layer of 572 Italian olive oil data for various deep belief neural network structures. The olive oils become the more clearly separated, the deeper the network structure is (cf. [21]).

V. CONCLUSION

This paper introduced a novel Generalized Hebb rule (GH-N) as an alternate to Hinton's Contrastive Divergence rule (CD-N) for training deep belief networks. While both rules have many similarities, the emphasis of this paper is on a simple derivation from basic principles.

In our future work we will investigate the theoretical behavior and the actual performance of the novel GH-N technique in much more detail.

REFERENCES

[1] Hugo Larochelle, Yoshua Bengio, Jérôme Louradour, and Pascal Lamblin [2009] Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, Vol. 1, pp. 1-40.

[2] Yoshua Bengio [2012] *Learning Deep Architectures for AI*. Technical Report 1312, University of Monreal, Canada.

[3] Geoffrey E. Hinton, and Terrence J. Tejnowski [1986] Learning and relearning in Boltzmann machines. In Parallel Distributed Processing: *Explorations in the Microstructure of Cognition*. MIT Press. Cambridge, MA, Vol. 1, pp. 283-317.

[4] Hsin Chen and Alan. F. Murray [2003] Continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings of Visual Image and Signal Processing*, Vol. 150(3), pp. 153-158.

[5] Benjamin M. Marlin, Kevin Swersky, Bo Chen, and Nondo de Freitas [2010] Inductive principles for restrictive Boltzmann Machine Learning. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9, pp. 509-516.

[6] Geoffrey Hinton [2010] A practical guide to training Restricted Boltzmann Machines, Version 1. *University of Toronto Technical Report*, UTML TR 2010-003 [Augst2, 2010].

[7] Ilya Sutskever and Tijmen Tieleman [2010] On the convergence properties of Contrastive Divergence. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS). May 13-15, Sardinia, Italy.

[8] Geoffrey Hinton [2002] Training products of experts by minimizing contrastive divergence. *Neural Computation*, Vol. 14(8), pp. 1771-1800.

[9] Yan LeCun, L. Bottou, G. Orr and K. Muller [1988] Efficient BackProp. In Orr, G. and Muller, K. (Eds.) *Neural Networks: Tricks of the Trade*. Springer

[10] Simon Haykin [2009] *Neural Networks and Learning Machines, Third Edition*, Pearson.

[11] Michele Forina and Carla Armanino [1981] Eigenvector projection and simplified nonlinear mapping of fatty acid content of Italian olive oils. *Ann. Chem.*, Vol. 72, pp. 125-127.

[12] Jure Zapan and Johann Gasteiger [1999] *Neural Networks in Chemistry and Drug Design (2nd Edition)*. Wiley –VCH.

[13] Hervé Bourlard and Yves Kamp, Auto-association by multilayer perceptrons and singular value decomposition, *Biological Cybernetics*, Vol. 59, pp. 291-294, 1988.

[14] Nathalie Japkowicz, Stephen J. Hanson, and Mark A. Gluck, Nonlinear autoassociation is not equivalent to PCA, *Neural Computation*, Vol. 12, pp. 531-545, MIT, 2000.

[15] M. A. Kramer, Autoassociative neural networks, *Computers and Chemical Engineering*, Vol.16, pp. 313-328, Pergamon Press, 1992.

[16] Hugo LaRochelle, Benjamin Bengio, Jerôme Lourdour, and Pascal Lamblin [2007] Exploring strategies for training deep belief networks. *Journal of Machine Learning Research*, Vol. 1, pp. 1-40.

[17] Paul J. Werbos [1994] *The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting*. New York, NY. John Wiley & Sons, Inc.

[18] Donald O. Hebb [1949] *The Organization of Behavior*. New York, Wiley and Sons.

[19] Eric B. Baum, and Frank Wilzek [1988] Supervised learning of probability distributions by neural networks. In Neural Information Processing Systems. Denver 1987. D. Z. Anderson, Editor, pp. 52-56.

[20] Pascal Vincent, Hugo LaRochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoinne Manzagol [2010] Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research, Vol. 11, pp. 3371-3408.

[21] Mark J. Embrechts, Blake Hargis, and Jonathon D. Linton, "Augmented Efficient BackProp for Backpropagation Learning in Deep Autoassociative Neural Networks." *Proceedings of the 2010 IEEE International Joint Conference on Neural Networks (IJCNN 2010)* as part of The World Congress on Computational Intelligence (WCCI 2010), pp. 1012-1020, Barcelona, Spain, July 18-23, 2010.