Coombes, S., & Campbell, C. (1996). Efficient learning beyond saturation by single-layered neural networks.

Early version, also known as pre-print

Efficient learning beyond saturation by single-layered neural networks

S Coombes and C Campbell

Neural Computing Research Group, Department of Engineering Mathematics, Queens Building, University Walk, University of Bristol, Bristol, BS8 1TR, UK

Abstract.

We present two distinct learning rules that can be applied to the problem of constructing a dichotomy of an arbitrary pattern set. Both rules are given in closed form and are discussed within the context of a single layered network of N nodes storing P patterns. For $P \leq N$ both rules realize dichotomies of a pattern set and can guarantee perfect storage up to a maximal capacity of $P = N$. Beyond this maximal capacity the rules can be used to give efficient solutions with a small number of errors. Their efficiency for learning beyond saturation and the simple closed form of these solutions make them ideal for use in constructive algorithms. Generalization performance and the issue of pattern stability in the regime of perfect storage are also examined and theoretical predictions are compared with simulations.

1. Introduction

The Perceptron rule [1] is a fast and efficient algorithm but, since it is a rule for single-layered learning, it is unable to store pattern sets which are not linearly separable. However, a number of authors have recently proposed constructive algorithms in which single-layered learning rules are used during the generation of a multi-layered neural network capable of performing arbitrary mappings. Typically a rule such as Perceptron learning is used and the task at each step is to store all the patterns or maximize the number of patterns that can be stored and grow further hidden nodes to store the remaining patterns. Examples of such constructive algorithms are the Upstart algorithm [2], Perceptron Cascade [3], the Target Switch algorithms [4] and other algorithms for generating neural networks with a single hidden layer [5] or two hidden layers [6].

The performance of these constructive algorithms relies on the underlying rule for single-layered learning. A good rule should be able to perfectly store the pattern set if this is possible or, if trained beyond saturation, it should be able to store a maximal fraction of the patterns correctly. A good rule should also be fast and preferably in closed form.

The storage capacity of a single layer network that is able to correctly classify P patterns using a network of N nodes and one binary valued output is defined as $\alpha \equiv P/N$. The maximal storage capacity, α_c, of a single layer network with real valued weights is $\alpha_c = 2$ for randomly chosen real input patterns [7, 8]. This maximal storage capacity can actually be achieved using the Perceptron rule. Thus if ξ_j^μ are real inputs, with $\mu = 1 \ldots P, j = 1 \ldots N$, and η^μ are the corresponding target values ($\eta^\mu \in \{-1, +1\}$) then we find the weights using the following algorithm

1 Start with random weights $w_j^{(0)}$

2 Present all the patterns, $\mu = 1 \ldots P$, and for each pattern calculate

$$X^\mu = \eta^\mu \sum_j w_j \xi_j^\mu \tag{1}$$

if $X^\mu \leq 0$ then

$$w_j^{(t+1)} = w_j^{(t)} + \eta^\mu \xi_j^\mu \tag{2}$$

3 If all the patterns are stored correctly, $X^\mu > 0$, $\forall \mu = 1 \ldots P$, or some maximum number of iterations are completed then stop, else, goto step 2.

It has recently been shown that α_c can be less than 2 if similar patterns are mapped onto different outputs and vice-versa [9]. Furthermore, optimal basins of attraction are known to be small in the regime $1 < \alpha < 2$ [10]. Hence, in practice algorithms that can guarantee a maximal storage capacity of $\alpha_c = 1$ for patterns drawn from arbitrary distributions are of interest.

In this Letter we present two such rules for single-layered learning. The closed form solution of both learning strategies allows the use of efficient numerical techniques for their construction. However, more traditional iterative neural algorithms also exist for this purpose.

The task of mapping N input patterns to N outputs is easily accomplished with the aid of a set of N independent single output Perceptrons. Introducing the notation w_{ij} for the j^{th} weight of the i^{th} Perceptron the problem of association becomes equivalent

to demanding that the NP Gardner conditions [8] are satisfied:

$$\gamma_i^\mu \equiv \frac{\xi_i^\mu \sum_{j\neq i} w_{ij}\xi_j^\mu}{\parallel w_i \parallel} > \kappa \quad \parallel w_i \parallel = \sqrt{\sum_{j\neq i} w_{ij}w_{ij}} \tag{3}$$

if the real pattern vector ξ^μ with components ξ_j^μ is to be associated with the target value $\xi_i^\mu \in \{-1,+1\}, i \neq j$. κ is a positive stability parameter [8], that provides robustness to noise, although at the expense of a decreased maximal storage capacity (as will be discussed later).

Hence, if a set of weights w_{ij} satisfies the Gardner conditions then a solution to a reduced single layer network problem classifying the pairs (ξ^μ, ξ_1) is given by:

$$w_i = w_{i1} \tag{4}$$

We now discuss two recent learning rules that achieve the NP Gardner conditions and consider their reduction to single layer learning, with one output node.

1.1. The Inverse-Hebb Rule

The Inverse-Hebb rule may be used to classify N arbitrary patterns in a net of N nodes [11, 12]. To classify less than N patterns requires the introduction of some arbitrary pattern vectors to increase the total number in the pattern set to N. The rule does not provide the network with any stability in the sense that the Gardner parameters, γ_i^μ, are zero $\forall\, i, \mu$. The Inverse-Hebb rule is so called because it takes the form:

$$w_{ij} = -(C^{-1})_{ij} \quad C_{ij} = \frac{1}{M}\sum_{\mu=1}^{M} \xi_i^\mu \xi_j^\mu \quad M = N \tag{5}$$

The stabilization algorithm [12, 13] is able to increase the Gardner parameters for a subset of patterns stored using the Inverse-Hebb rule. The analogue of the Inverse-Hebb rule for the single layer network with one output node is obtained via (4) and the corresponding stabilization algorithm takes the form:

$$\delta w_i = \frac{\epsilon}{P}\sum_{\mu=1}^{P}\left\{e^{-\lambda\eta^\mu h^\mu}\eta^\mu(\xi_i^\mu - h^\mu w_i)\right\} \quad h^\mu = \sum_{j\neq 1} w_j\xi_j^\mu \quad \epsilon,\ \lambda > 0 \tag{6}$$

where $\eta^\mu = \xi_1^\mu$ and ϵ and λ are learning rates (example values are given in section 2). The first term performs gradient descent on the convex function $\exp(-\lambda\eta^\mu h^\mu)$ which is bounded from below ensuring that a minimum exists. Since this function is a continuous decreasing function of $-\eta^\mu h^\mu$, the minimum is to be found for as large a value of $\eta^\mu h^\mu$ as possible. The second term is a form of weight decay, such that at the fixed point $\parallel w \parallel = 1$. Hence, the stability parameter $\eta^\mu h^\mu \parallel w \parallel^{-1}$ is also maximized. The stabilization algorithm is local, does not require the choice of a stability parameter prior to learning and actually has learning times that decrease with increasing capacity [12]. These properties contrast favourably with more traditional Perceptron learning algorithms based upon the Gardner algorithm [8].

1.2. The Feature Matrix

The Feature Matrix also achieves a maximal storage capacity of $\alpha_c = 1$ for patterns drawn from an arbitrary pattern distribution [14]. In common with the closed form

of the Pseudo-Inverse rule [15, 16] it is a projection operator on the pattern subspace with components given by:

$$w_{ij} = \sum_{a=1}^{P} f_i^a f_j^a \quad \sum_j C_{ij} f_j^a = \lambda^a f_i^a \quad C_{ij} = \frac{1}{P} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu \tag{7}$$

Where the label a orders the eigenvalues λ^a of the correlation matrix by decreasing value. It has the advantage over the closed form of the Pseudo-Inverse rule in that it is not restricted to the case in which the patterns must be linearly independent. It is worth mentioning that the Inverse-Hebb matrix is also a projector, explicitly onto the subspace spanned by the training patterns. The Feature Matrix is so called since it is built from the *features* or principal components of a pattern set. The rule ensures that the local field, $h_i^\mu = \sum_{j \neq i} w_j \xi_j^\mu$, of a pattern is the optimal (in the least squares sense) linear reconstruction of the target vector ξ_i^μ, for a given pattern index μ. The Gardner conditions are satisfied since the local field of the pattern points in the same direction as the target vector [14]. A consequence of self-averaging in the thermodynamic limit is that Feature Matrix yields the same distribution $\rho(\gamma)$ for the pattern stabilities as the Pseudo-Inverse rule, namely $\rho(\gamma) = \delta(\gamma - \gamma_0)$, where $\gamma_0 = \sqrt{(1-\alpha)/\alpha}$, for patterns drawn from a random unbiased distribution. Again, the analogue of the Feature Matrix storage rule for the single layer network with one output node is obtained via (4).

2. Simulations

In this section we explore the performance of the two algorithms presented in section 1. As a benchmark we make comparisons with the performance of a single layer network possessing a theoretically optimal architecture rather than with other existing training algorithms. We have simulated networks with 100 inputs and averaged over a 100 typical implementations. Explicitly, we consider random unbiased binary input-output mappings from 100 dimensions to 1. The critical storage capacity of $\alpha_c = 1$ is still reached by both algorithms when considering biased patterns.

The symmetry and positive definite properties of the pattern correlation matrix allows one to use algorithms such as Cholesky decomposition in the construction of the Inverse-Hebb rule requiring $O(N^3/6)$ operations (see [17] for appropriate numerical routines).

The Feature Matrix is easily obtainable with traditional numerical routines that extract eigenvectors, although many neural learning algorithms also exist to obtain principal components [18, 19]. Again, one may exploit the properties of the pattern correlation matrix and perform a Householder reduction on it (requiring $O(4N^3/3)$ operations). An algorithm such as QL may be used to obtain the eigenvectors and eigenvalues of such a real symmetric, tri-diagonal matrix (requiring $O(3N^3)$ operations).

A discussion of alternative neural mechanisms for the construction of both rules may be found in [12] and [14].

In figure 1 we show the value of κ for the Feature Matrix and also for the fixed point of the stabilization algorithm (primed with the Inverse-Hebb rule). Also shown is the optimal Gardner result $\alpha_G(\kappa)$ and the Feature Matrix result $\alpha_{FM}(\kappa) = 1/(1+\kappa^2)$ for random unbiased patterns. The stabilization of pattern subsets was catered for using (6) with $\epsilon = 2.0$ and $\lambda = 0.1$ and the results demonstrate the ability of this algorithm

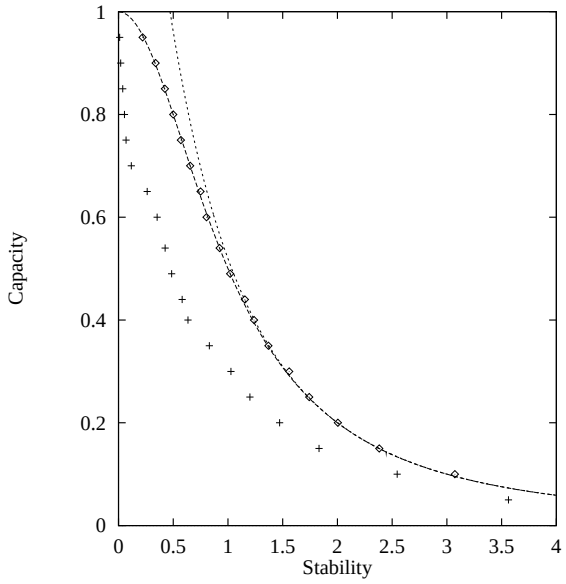

Figure 1. Storage capacity as a function of stability for the combination Inverse-Hebb and stabilisation algorithm (+) and Feature Matrix rule ($\diamond$). The lower dashed line is the theoretical curve describing the stability of the Feature Matrix rule whilst the upper is the optimal Gardner result.

to enhance pattern stabilities away from zero towards the upper limit obtained by Gardner. The close agreement between theory and simulation for the stability of the Feature Matrix, combined with the proximity of results to the Gardner optimum for $\alpha < 0.8$ demonstrates the effectiveness of this closed form solution. Although still achieving the same level of storage as the Feature Matrix, the combination of Inverse-Hebb and stabilization algorithm does not, in practice, yield comparable stabilities. However, as we shall demonstrate, the Inverse-Hebb rule has favourable performance gains in the regime beyond saturation.

Since the Inverse-Hebb matrix is built from exactly N patterns one first needs to decide how to extend its definition in the regime beyond saturation before any generalization properties can be investigated. To this end we extend the definition of the rule (5) in the regime beyond saturation (ie $P > N$) by setting $M = P$. Figure 2 demonstrates the graceful degradation achieved when using the Inverse-Hebb rule in the regime beyond saturation by plotting the fraction of errors developed in a single layer network trying to store more than N random unbiased patterns. It is apparent that the Feature Matrix suffers far more serious performance losses in this regime.

In this figure we also show the theoretical result obtained in [8, 20] for $\kappa = 0$. This is an optimal curve for Perceptron performance and in practice numerical simulations of Perceptron learning algorithms (including Optimal Perceptron learning) fall well short of this result.

The Inverse-Hebb rule performs more favourably in the regime beyond saturation than the Feature Matrix. For example, at $\alpha = 2$, the Feature Matrix shows a drastic performance depreciation, but the Inverse-Hebb rule shows only a relatively small

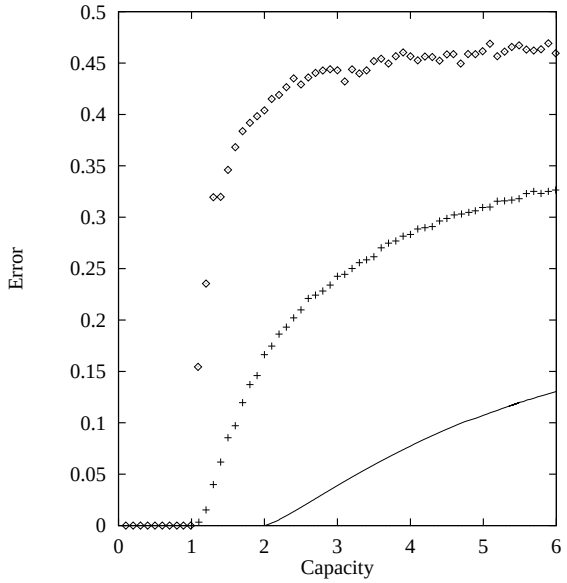

Figure 2. Fraction of errors generated by the Inverse-Hebb rule (+), and Feature Matrix ($\diamond$) beyond saturation. Solid line corresponds to the theoretical optimal Gardner scenario with $\kappa = 0$.

degradation in performance. However, in comparison to the Perceptron rule (and indeed variants on the rule such as Minover [21] and Adatron [22]) extra criteria for obtaining local optima of negative stability do not have to be introduced in the regime beyond saturation. Recently Imhoff has introduced the RECOMI algorithm which is a polynomially fast algorithm for searching optimally stable solutions of the Perceptron learning problem [23]. Although simulations have shown that it can operate in the regime beyond saturation, there is as yet no proof of convergence. Furthermore, it requires $O(N^4)$ operations which is an order of magnitude larger than required for the construction of either the Inverse-Hebb rule or the Feature Matrix.

Figure 3 shows the training errors for the Feature Matrix and Inverse-Hebb rules learning a realizable rule. A realizable rule is one for which training pairs can be generated by a *teacher* network of the same architecture as the *student* single layer architecture. In the simulations, teacher weights are generated at random from an unbiased distribution and all results are averaged over a hundred instances of the teacher. The Inverse-Hebb rule is able to increase its storage capacity beyond one, to about 1.5, in contrast to the Feature Matrix which retains a storage capacity of one.

In figure 4 we show the generalization performance for this task. The Feature Matrix shows the unusual property of over-fitting the data in the storage regime causing a decrease in the generalization performance as $P \to N$. This is also true for the Pseudo-Inverse rule and has been investigated theoretically for the case of a linear Perceptron by several authors [24, 25, 26]. The generalization properties of single layer networks learning realizable rules with the stability distribution $\rho(\gamma)$ of the Pseudo-Inverse rule have been studied in [27]. Since the Feature Matrix also possesses this distribution of stabilities we plot the corresponding theoretical generalization

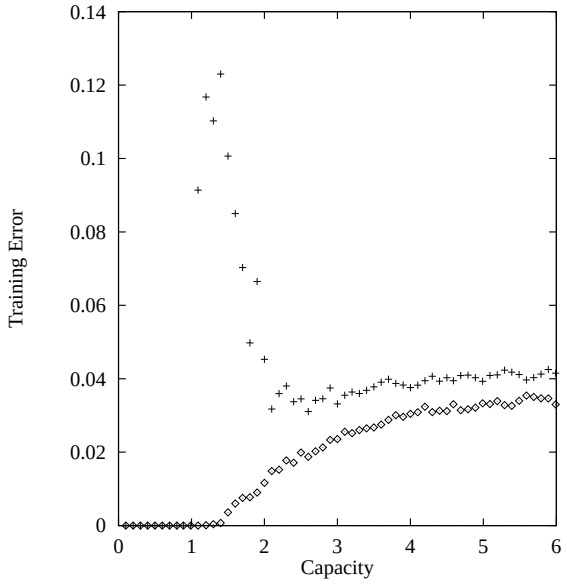

Figure 3. Training error for the Inverse-Hebb rule ($\diamond$) and Feature Matrix ($+$) learning a realizable rule.

curve for comparison. There is reasonable agreement between theory and experiment, particularly in the regime of over-fitting where $\alpha \leq 1$. Also plotted is the theoretical prediction for the Optimal Perceptron learning rule [27]. (A numerical error [28] in the original plot of this curve in [27] has been corrected for). A comparison of results from the Inverse-Hebb rule and this curve is apt since the rule was originally derived by examining the properties of a Perceptron in the limit of high capacity. Although not fitting the Inverse-Hebb data it does bound both the simulation results for the two rules as expected.

3. Discussion

Unlike the Feature Matrix, the combination of the Inverse-Hebb rule and the stabilization algorithm does not produce the same value for all the P stability parameters. The inability of the stabilization algorithm to increase the stability parameters at the same rate is the reason for this. Although the Feature Matrix achieves a higher level of stability in the storage regime it does not perform as well as the Inverse-Hebb rule beyond saturation. This is reflected in the rapid rise of the network error level. However, it is the generalization properties of these rules that are of more consequence than their storage properties. Although both rules perform well at learning a realizable rule it is again the Inverse-Hebb rule that performs best. Also noteworthy is the fact that it almost doubles its storage capacity for this case, coming closer to the theoretical maximum. The generalization error for the Inverse-Hebb rule is currently being investigated by us using replica techniques.

Single layer learning rules are limited in the sense that they can only realism dichotomies for pattern sets which are linearly separable. However, a number of

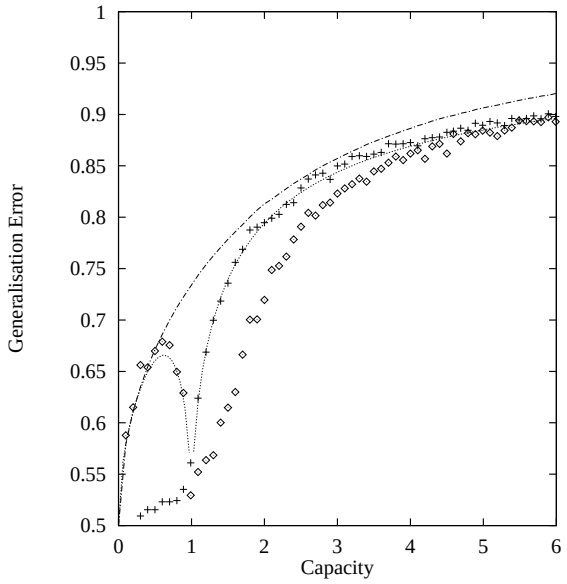

Figure 4. Generalization error for the Inverse-Hebb rule (+) and Feature Matrix ($\diamond$) learning a realizable rule. The theoretical generalization curve for the Feature Matrix (- - - -) and the optimal Perceptron (— · —) are also shown.

authors have proposed constructive algorithms that generate the architecture of a feed-forward network in addition to determining the number of weights required for a given pattern classification task. For example, the Target Switch algorithm [4] can be used to generate feed-forward architectures by composing sets of single layer elements into a larger structure. Each of the single layer elements can be trained with any of the existing single layer algorithms. Hence, both the rules discussed in this paper find ready application in constructive algorithms. The performance gains that are available when using these rules is being investigated for a class of dichotomy based constructive learning algorithms and will be presented elsewhere.

[1] F Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

[2] M Frean. The upstart algorithm: A method for constructing and training neural networks. *Neural Computation*, 2:198–209, 1990.

[3] N Burgess. A constructive algorithm that converges for real-valued inputs. *International Journal of Neural Systems*, 5:59–66, 1994.

[4] C Campbell and C P Vicente. The target switch algorithm: A constructive learning procedure for feedforward neural networks. *Neural Computation*, 7:1221–1240, 1995.

[5] M Marchand, M Golea, and P Rujan. A convergence theorem for sequential learning in two-layer perceptrons. *Europhysics Letters*, 11:487–492, 1990.

[6] D Martinez and S Esteve. The offset algorithm: Building and learning method for multilayer neural networks. *Europhysics Letters*, 18:95–100, 1992.

[7] T Cover. Geometrical and statistical properties of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computing*, 14:326, 1965.

[8] E Gardner. The space of interactions in neural network models. *J. Phys. A*, 21:257–270, 1988.

[9] B Lopez, M Schroder, and M Opper. Storage of correlated patterns in a perceptron. *Submitted to J. Phys. A Letters*, 1995.

[10] G Kohring. Neural networks with many-neuron interactions. *Le Journal de Physique*, 51:145–155, 1990.

[11] S Coombes and J G Taylor. Using generalized principal component analysis to achieve associative memory in a hopfield net. *Network*, 5:75–88, 1993.

[12] S Coombes and J G Taylor. The storage and stabilisation of patterns in a hopfield net. *Neural Network World*, 5(2):133–150, 1995.

[13] P Peretto. On the dynamics of memorization processes. *Neural Networks*, 1:309–322, 1988.

[14] S Coombes and J G Taylor. Using features for the storage of patterns in a fully connected net. *To appear in Neural Networks*, 1996.

[15] I Kanter and H Sompolinsky. Associative recall of memory without errors. *Phys. Rev. A*, 35:380–392, 1987.

[16] L Personnaz, I Guyon, and G Dreyfus. Information storage and retrieval in spin glass like neural networks. *Journal de Physique Lettres*, 46:L359–L365, 1985.

[17] W H Press, S A Teukolsky, W T Vetterling, and B P Flannery. *Numerical Recipes in C*. Cambridge University Press, 1994.

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

[19] T D Sanger. Optimal unsupervised learning in a single layer linear feedforward neural network. *Neural Networks*, 2(7):459–473, 1989.

[20] R Erichsen and W K Theumann. Optimal storage of a neural network model: a replica symmetry breaking solution. *J. Phys. A*, (26):L61–L67, 1993.

[21] W Krauth and M Mezard. Learning algorithms with optimal stability in neural networks. *J. Phys. A*, 20:L745–L752, 1987.

[22] J K Anlauf and M Biehl. Properties of an adaptive perceptron algorithm. In R Eckmiller, G Hartmann, and G Hauske, editors, *Parallel Processing in Neural Systems and Computers*, pages 153–156. North Holland: Amsterdam, 1990.

[23] J Imhoff. A polynomial training algorithm for calculating perceptrons of optimal stability. *J. Phys. A*, 28:2173–2181, 1995.

[24] S Amari. Overtraining in simple perceptrons. *To appear in Annals of Mathematics in Artificial Intelligence*, Baltzer, 1996.

[25] D Barber, D Saad, and P Sollich. Finite size effects and optimal test set size in linear perceptrons. *J. Phys. A*, 28:1325–1334, 1995.

[26] S Bos, W Kinzel, and M Opper. Generalization ability of perceptrons with continuous outputs. *Phys. Rev. E*, 47(2):1384–1391, 1993.

[27] M Opper, W Kinzel, J Kleinz, and R Nehl. On the ability of the optimal perceptron to generalise. *J. Phys. A*, 23:L581–L586, 1990.

[28] M Opper. Private communication. 1995.