

Synthesis of neural networks: the case of cascaded Hebbians

Csaba Szepesvári

August 1996

TR-96-102

Research Group on Artificial Intelligence
Hungarian Academy of Sciences,
“Attila József” University
Szeged, 6720, Aradi vrt tere 1.
HUNGARY

<http://www.inf.u-szeged.hu/~rgai>

Synthesis of neural networks: the case of cascaded Hebbians

Csaba Szepesvári

Bolyai Institute of Mathematics, “Attila József” University,

Research Group on Artificial Intelligence

Hungarian Academy of Sciences,

“Attila József” University

Szeged, 6720, Aradi vrt tere 1.

HUNGARY

Abstract

We show that cascading Hebbian learning with any other convergent algorithm (called the forward algorithm) results in the convergence of the Hebbian weights to a stationary point where the Hebbian algorithm would converge if the weights of the forward algorithm had already converged. Further, it is shown that the convergence rate of the composite algorithm does not deteriorate because of the cascading. This result is a consequence of a more general theorem which is also stated and proved here, the proofs being based on a global Lipschitzian assumption. The theory is illustrated by a composite PCA-Hebbian architecture introduced by Micheals (Michaels, 1995).

1 Introduction

Hybrid learning rules for training multi-layer neural networks have been considered by scores of researchers in the field. These include Hecht-Nielsen (Hecht-Nielsen, 1987), Ritter et al. (Ritter et al., 1988), Moody and Darken (Moody and Darken, 1989), Poggio and Girosi (Poggio and Girosi, 1990), Nowlan (Nowlan, 1990), Benaim and Tomasini (Benaim and Tomasini, 1991, Benaim and Tomasini, 1992), Martinetz and Schulten (Martinetz and Schulten, 1991) Fomin et al. (Fomin et al., 1994), Szepesvári et al. (Szepesvári et al., 1994), Joutsensalo et al. (Joutsensalo et al., 1995), and Michaels (Michaels, 1995) to name but a few.

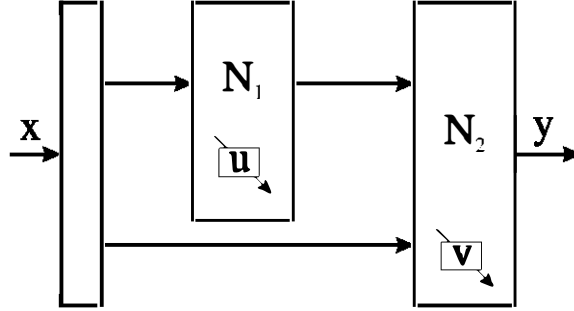


Figure 1: Single hidden layer cascaded network

The weight $\mathbf{u}$ of subnetwork N_1 is trained independently of the weight $\mathbf{v}$ of subnetwork N_2 . We investigate in this article if the convergence of the weight vector of N_2 follows from the convergence of the weight vector of N_1 . We impose conditions on the training rule of N_2 when training N_1 and N_2 simultaneously, or training N_2 after N_1 has converged, which make the two modes equivalent. Specifically, we prove that if N_2 is trained by the Hebbian law then the convergence speed of $\mathbf{v}$ is simply $\mathcal{O}(\max(1/\sqrt{t}, \|\mathbf{u}_t - \mathbf{u}\|))$.

Consider, for the sake of simplicity, a single hidden-layer network (see Figure 1). Formally, the output of the network is given by $\mathbf{y} = N_2(\mathbf{x}, N_1(\mathbf{x}, \mathbf{u}), \mathbf{v})$, where $\mathbf{w} = (\mathbf{u}, \mathbf{v}) \in \mathbf{R}^{m_1} \times \mathbf{R}^{m_2}$ are the weights of the network, $\mathbf{x} \in \mathbf{R}^n$ is the input, $\mathbf{y} \in \mathbf{R}^m$ is the output, $N_1 : \mathbf{R}^n \times \mathbf{R}^{m_1} \rightarrow \mathbf{R}^k$ and $N_2 : \mathbf{R}^n \times \mathbf{R}^k \times \mathbf{R}^{m_2} \rightarrow \mathbf{R}^m$ are the transfer functions of the hidden and the output layer, respectively.

The integer k is the number of hidden units. Here N_1 and N_2 describe the computation performed by the hidden layer neurons and output layer neurons, respectively.

Such a network can be viewed as the cascade of two subnetworks provided that the weights $\mathbf{v}$ do not influence the training of weights $\mathbf{u}$. Usually, the weights $\mathbf{u}$ are trained by an unsupervised learning rule (e.g. some form of competitive learning or a maximum likelihood algorithm) and the weights $\mathbf{v}$ are trained according to a supervised algorithm (Hebbian learning, competitive Hebbian learning, etc.). The training of weights is a special case of the stochastic approximation (SA) procedure introduced by Robbins and Monro (Robbins and Monro, 1951). Here we are interested in the convergence and rate of convergence of the weights of the cascaded part. Since, by assumption, the weights $\mathbf{v}$ do not influence the weights $\mathbf{u}$ we expect that the training of weights $\mathbf{u}_t$ (t denoting the training time) is such that the weights $\mathbf{u}_t$ converge to some limiting value $\mathbf{u}^*$. After that no further assumption is put on the training of N_1 . As for the other weights, we provide sufficient conditions under which the weights of the second layer, $\mathbf{v}_t$, converge to the value they should converge to when the weights $\mathbf{u}_t$ are bounded to their limiting value $\mathbf{u}^*$ during the training of N_2 . An estimate of the convergence rate is also provided in a simple but important case when N_2 is trained by supervised Hebbian learning.

The convergence of cascaded algorithms have been studied in some detail by Benaim (Benaim, 1993, Benaim, 1996). First, he considered continuous-time learning under the adiabatic assumption (i.e., that the cascaded part works on a substantially slower time-scale). Recently, he extended his method to discrete time SA algorithms and was able to drop the adiabatic assumption as well (Benaim, 1996), but did not prove any convergence rates. Benaim's recent convergence results are, in a way, both more and less general than ours. He considers hybrid algorithms with the cascaded part trained by an arbitrary gradient rule (in this respect his result is very general), but he also presupposes

the input layer is trained by another gradient rule which we do not require here. Actually, he does not prove the convergence of the cascaded weights (except for the linear case elaborated upon later by us), but just says that if those weights converge they would converge to the “right” limit point. The advantage of our approach is not only its simplicity but that it allows one to estimate the convergence rate of the compound process directly.

The article is organized as follows: in the next section the key results are cited and discussed. Then in Section 3 the results obtained are used to derive the convergence and speed of convergence of a hybrid algorithm introduced in a different context by Michaels, an algorithm which is an alternative to the classical LMS algorithm. Finally, all the proofs needed are presented in the Appendix section of the paper.

2 Results

The following theorem forms the basis of our results:

THEOREM 2.1 *Let X and Y be Banach spaces, $U_n : X \times Y \rightarrow X$ ($n = 0, 1, 2, \dots$) be a sequence of mappings, let $\theta_n \in Y$ be an arbitrary series, let $\theta_\infty \in Y$ and $x_\infty \in X$. Let*

$$x_{n+1} = U_n(x_n, \theta_\infty), \quad (1)$$

and

$$y_{n+1} = U_n(y_n, \theta_n), \quad (2)$$

with $x_0 = y_0$.¹ Then

$$\|y_n - x_\infty\| \leq \|x_n - x_\infty\| + \sum_{s=0}^n \|\theta_s - \theta_\infty\| \prod_{t=s+1}^n I_t^x, \quad (3)$$

¹This condition, i.e., that $x_0 = y_0$ is not necessary at all. However, it is a useful assumption for simplifying the proof.

where² L_k^θ is the uniform Lipschitz index of $U_k(x, \theta)$ w.r.t. θ at θ_∞ and, similarly, L_k^x is the uniform Lipschitz index of $U_k(x, \theta_\infty)$ w.r.t. x .³

Now suppose that x_n and θ_n converge to x_∞ and θ_∞ respectively, in the norm of the appropriate spaces. If the Lipschitz constants L_n^x and L_n^θ satisfy the relations

$$L_n^\theta \leq C(1 - L_n^x), \quad (4)$$

and

$$\prod_{m=n}^{\infty} L_m^x = 0, \quad (5)$$

where $n = 0, 1, 2, \dots$ then also $\lim_{n \rightarrow \infty} \|y_n - x_\infty\| = 0$.

If, $L_n^\theta = c/n$ for some $c > 0$, $L_n^x \leq 1 - 1/n$ and $\|\theta_n - \theta_\infty\| = \mathcal{O}(1/n^\alpha)$, where $0 < \alpha < 1$ then

$$\|y_n - x_\infty\| = \mathcal{O}(\max(1/n^\alpha, \|x_n - x_\infty\|)).$$

It is clear that the given convergence rate must be a strict one hence we cannot expect y_n converge faster than either x_n or θ_n .

Theorem 2.1 could have many applications. For example, if $U_n(x, \theta) = U(x, \theta)$ is a contraction with contraction index γ in x ($L_n^x = \gamma$) and $L_n^\theta < B$ for some $B > 0$ then the above theorem yields the result that $y_{n+1} = U_n(y_n, \theta_n)$ converges to the fixed point of $U(\cdot, \theta_\infty)$. One simple application of this is in reinforcement learning, where θ_n corresponds to the estimated parameters of a Markovian decision process and $U_n(\cdot, \theta)$ is the optimal value operator corresponding to a discounted Markovian decision problem with parameter vector θ . This is just the synchronous version of the convergence theorem of Gullapalli and Barto (Gullapalli and Barto, 1994).

In this article the above theorem is applied to the simplest supervised training algorithm, to that of Hebbian learning (Haykin, 1994). That is, we assume

²Here we admit the convention that $\prod_{t=m}^n a_t = 1$ if $m > n$.

³That is, for all $x \in X$ and $\theta \in Y$ $\|U_k(x, \theta) - U_k(x, \theta_\infty)\| \leq L_k^\theta \|\theta - \theta_\infty\|$ and for all $x, y \in X$ $\|U_k(x, \theta_\infty) - U_k(y, \theta_\infty)\| \leq L_k^x \|x - y\|$.

that the output neurons are trained by the rule

$$\mathbf{v}_{t+1} = (1 - \alpha_t)\mathbf{v}_t + \alpha_t F_t, \quad (6)$$

where $\alpha_t \geq 0$, $t = 0, 1, 2, \dots$ satisfy the usual criteria, i.e., they are square summable but not linear summable and $F_t \in \mathbf{R}^{m_2}$ may depend on the desired output value $\mathbf{y}_t$, the actual weights of the hidden layer $\mathbf{u}_t$ and input of the net $\mathbf{x}_t$, but not depend on $\mathbf{v}_t$:

$$F_t = F(\mathbf{y}_t, \mathbf{x}_t, \mathbf{u}_t).$$

(Usually, the dependence of F_t on $\mathbf{x}_t$ and $\mathbf{u}_t$ goes through the output of the hidden layer neurons.) Note that almost all the hybrid algorithms cited above use Hebbian learning for the training of the “hidden layer” weights.

The following theorem is a consequence of Theorem 2.1.

THEOREM 2.2 *Let $\|\cdot\|$ denote an arbitrary norm defined for random vectors. Assume that for any random vector $\mathbf{v}$, the norm of $\mathbf{v}$ depends only on the distribution of $\mathbf{v}$. Let $\mathbf{u}_t$ be a stochastic process which converges to some vector $\mathbf{u}^*$ in the chosen norm: $\lim_{t \rightarrow \infty} \|\mathbf{u}_t - \mathbf{u}^*\| = 0$. Assume that $(\mathbf{x}_t, \mathbf{y}_t)$ is a stochastic process of identically-distributed random variables. Assume further that for all vector $\mathbf{u}$ the following inequality holds:*

$$\|F(\mathbf{y}_0, \mathbf{x}_0, \mathbf{u}) - F(\mathbf{y}_0, \mathbf{x}_0, \mathbf{u}^*)\| \leq K \|\mathbf{u} - \mathbf{u}^*\|,$$

where $K > 0$ is a fixed real number. Then the following things hold:

1. The output weights $\mathbf{v}_t$ trained by rule (6) converge iff they converge when trained by the rule

$$\hat{\mathbf{v}}_{t+1} = (1 - \alpha_t)\hat{\mathbf{v}}_t + \alpha_t F(\mathbf{y}_t, \mathbf{x}_t, \mathbf{u}^*). \quad (7)$$

Further, in case of convergence the limiting values generated by the two rules are equal.

2. Let this limit point be $\mathbf{v}^*$ and assume that $\alpha_t = c/t$, where $c > 0$. Then the convergence rate of $\|\mathbf{v}_t - \mathbf{v}^*\|$ is no worse than the maximum of the convergence rates of $\hat{\mathbf{v}}_t$ and $\mathbf{u}_t$.

Note that the theorem is easily extended to the case when the condition that $\mathbf{u}_t$ converges to some limit point $\mathbf{u}^*$ is relaxed to the assumption that $\mathbf{u}_t$ converges to one element of a set U^* containing isolated vectors.

In order to obtain a practical convergence rate let us assume that $\mathbf{u}_t$ is computed by some SA method and that $\|\cdot\|_2$ is the L^2 norm $E[\|\cdot\|^2]^{1/2}$, where $E[\cdot]$ stands for the expectation value operator. Now under usual conditions (Wasan, 1969, Benveniste et al., 1990)

$$\|\mathbf{u}_t - \mathbf{u}^*\|_2, \|\hat{\mathbf{v}}_t - \mathbf{v}^*\|_2 = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right),$$

so by Theorem 2.2 also

$$\|\mathbf{v}_t - \mathbf{v}^*\|_2 = \mathcal{O}\left(\frac{1}{\sqrt{t}}\right).$$

It is easy to prove a theorem similar to Theorem 2.2 but for probability-one convergence cases and when the norm $\|\cdot\|$ is an ordinary vector norm; all we have to do is to consider pointwise convergence over the probability space underlying the sampling process. The precise restriction on F is that F should be Lipschitzian in $\mathbf{u}$, independently of its first two variables. This normally only holds for bounded $\mathbf{x}$ and $\mathbf{y}$, so we must assume that $\mathbf{x}_t$ and $\mathbf{y}_t$ are bounded with probability one (w.p.1): If it is further assumed that the process $(\mathbf{x}_t, \mathbf{y}_t)$ consists of independent, identically distributed (i.i.d.) random variables and $\alpha_t = 1/t$ then under some mild conditions on the forward algorithm the iterated logarithm theorem of P. Major (Major, 1973) gives that

$$\|\mathbf{u}_t - \mathbf{u}^*\|, \|\hat{\mathbf{v}}_t - \mathbf{v}^*\| = o\left(\sqrt{\frac{\log \log t}{t}}\right)$$

and eventually allows one to prove that

$$\|\mathbf{v}_t - \mathbf{v}^*\| = o\left(\sqrt{\frac{\log \log t}{t}}\right) \quad (8)$$

holds with probability one.

We note that Theorem 2.1 enables us to extend Theorem 2.2 to the case where F depends on $\mathbf{v}$ and just F should be a non-expansion w.r.t. $\mathbf{v}$, independently of the other parameters of F (it is sufficient if it holds for vectors in the tail of $\mathbf{v}_t$). The one-dimensional example $F(u, v) = uv^2$ shows that if F does not satisfy this condition then the conclusion of Theorem 2.2 no longer holds. In this case the limit of the process $\hat{\mathbf{v}}_t$ converges to a fixed point or to infinity depending on $\hat{\mathbf{v}}_0$, so the initial values of the process $\mathbf{u}_t$ can determine the final behaviour.

Though the proofs presented below are quite simple this is not counted as a drawback but rather as an advantage. The proofs are based on the classical approach to SA, i.e., the Robbins-Monro formalism which reduces the convergence of SA algorithms to the convergence of difference equations. Here we prefer this formalism to the method of ordinary differential equations (ODEs) introduced by Ljung (Ljung, 1977) and Kushner and Clark (Kushner and Clark, 1978) because of its simplicity.

3 Application: LMS minimisation by hybrid learning

Michaels proposed a two-layer, hybrid neural network that performs principal component analysis (i.e., decorrelation of the inputs) in its first layer and learns correlations (i.e., performs Hebbian-learning) in its second layer (Michaels, 1995). He has shown that this network solves the association of *finite* samples composed of input-output pairs with the minimum LMS error. Here we extend his analysis to *infinite random* samples under the assumption that the training samples are drawn independently from a common probability distribution having a finite second moment. It is demonstrated that the proposed learning algorithm solves the LMS (linear) approximation problem and also give an estimate of its

convergence speed.

3.1 Preliminaries

Let $\xi_t \in \mathbf{R}^n$, $t = 0, 1, 2, \dots$ be i.i.d. random variables with zero mean and finite second moment, and let $f : \mathbf{R}^n \rightarrow \mathbf{R}^m$ be an arbitrary bounded function. The task is to find the best linear approximation of f , that is the matrix $W \in \mathbf{R}^{m \times n}$ such that

$$E(W) = \frac{1}{2} E[||W\xi - f(\xi)||^2]$$

is minimal (here and in the following ξ denotes a random variable with the same distribution as ξ_t .) The classical incremental solution to this problem is to perform gradient-descent on the error surface $E(W)$ (Widrow and Hoff, 1960). Gradient-descent is guaranteed to converge to the optimal solution, since $E(W)$ has only one minimum which satisfies the equation

$$W^* = R_{f(\xi),\xi} R_{\xi,\xi}^{-1}, \quad (9)$$

where $R_{f(\xi),\xi} = E[f(\xi)\xi^T]$ is the cross-correlation matrix of the input and output data and $R_{\xi,\xi} = E[\xi\xi^T]$ is the autocorrelation matrix of the input data. Therefore the Widrow-Hoff LMS algorithm

$$W_{t+1} = W_t + \alpha_t (f(\xi_t) - W_t \xi_t) \xi_t^T$$

incrementally computes the product of $R_{f(\xi),\xi}$ and inverse of matrix $R_{\xi,\xi}$.

3.2 LMS algorithm based on decorrelation and correlation

Assume that all the eigenvalues of $R_{\xi,\xi}$ are positive and that they are distinct as well. Let U be the matrix composed of the orthonormalised eigenvectors of $R_{\xi,\xi}$ (that is $UU^T = E$, where E is the unit matrix) and let Λ be the diagonal matrix whose i th diagonal element corresponds to the i th eigenvector in U , that is

$$R_{\xi,\xi} = U\Lambda U^T. \quad (10)$$

Rewriting (9) in terms of (10) yields

$$W^* = R_{f(\xi),\xi}(U\Lambda U^T)^{-1} = R_{f(\xi),\xi}U\Lambda^{-1}U^T,$$

that is, if we let

$$Q = R_{f(\xi),\xi}U\Lambda^{-1} \quad (11)$$

then $W^* = QU^T$. Now let us introduce the variable

$$\eta = \eta(\xi) = U^T\xi. \quad (12)$$

Then the optimal LMS approximation of f is given by

$$y(\xi) = Q\eta(\xi). \quad (13)$$

Besides this Q can be written in terms of η since

$$\begin{aligned} Q &= R_{f(\xi),\xi}U\Lambda^{-1} = E[f(\xi)\xi^T U\Lambda^{-1}] = \\ &= E[f(\xi)(\Lambda^{-1}U^T\xi)^T] = E[f(\xi)(\Lambda^{-1}\eta)^T] = R_{f(\xi),\Lambda^{-1}\eta}. \end{aligned} \quad (14)$$

Equations (12) and (13) can be interpreted as the output computation performed by a two-layer (linear) neural network. This idea was developed further by Michaels (Michaels, 1995), who proposed that the learning of Q and U^T should be performed by neural algorithms (though he considered the finite sample case which requires a different treatment). There are several learning algorithms that are known to converge to U^T (see e.g. (Oja and Karhunen, 1985, Rubner and Tavan, 1989, Baldi and Hornik, 1989, Sanger, 1989, Xu, 1993, Xu and Yuille, 1995, Palmieri and Zhu, 1995, Chen and Chang, 1995)) and Hebbian learning can be used to learn the correlation matrix and the eigenvalues Λ . One possible algorithm is shown in Table 1. The equation of Step 2 may be replaced by any one of the many learning algorithms that learn the principal components of $R_{\xi,\xi}$. In Step 5 $\text{diag}(A)$ is defined by $\text{diag}(A)_{ij} = 0$ if $i \neq j$ and $\text{diag}(A)_{ii} = a_{ii}$. Thus $\text{diag}(A)^{-1}$ is easy to compute: $\text{diag}(A)_{ij}^{-1} = 0$ if $i \neq j$ and $\text{diag}(A)_{ii}^{-1} = 1/a_{ii}$. It is possible to avoid the inversion of the diagonal elements of Λ_t by learning

Algorithm 1: LMS algorithm based on decorrelation and correlation.

repeat forever{

1. Generate the training data: (ξ_t, f_t) , where $f_t = f(\xi_t)$.
2. Learn the principal components:

$$U_{t+1}^x = P_t(\xi_t, U_t^x).$$

3. Compute output of the first layer:

$$\eta_t = U_t^T \xi_t.$$

4. Learn the eigenvalue matrix:

$$\Lambda_{t+1} = (1 - \alpha_t)\Lambda_t + \alpha_t \eta_t \eta_t^T.$$

5. Learn the cross-correlation matrix:

$$W_{t+1} = (1 - \beta_t)W_t + \beta_t f_t (\text{diag}[\Lambda_{t+1}]^{-1} \eta_t)^T.$$

6. $t := t + 1$.

}

the inverse of Λ directly. This can be done by replacing the learning equation of Step 4 by a special LMS rule:

$$\Lambda_{t+1}^{-1} = \Lambda_t^{-1} + \alpha_t (E - \Lambda_t^{-1} \eta \eta^T).$$

3.3 Convergence properties

By Theorem 2.2 and the argument that followed it, we have that the proposed algorithm converges in the mean-square error norm with probability one since it uses only Hebbian learning in the cascaded part (Step 4 and 5). Assuming the L^2 norm of random vectors, the rate of convergence of the cascaded part is $\mathcal{O}(1/\sqrt{t})$. In general, the convergence rate of the whole algorithm will depend on the rate of convergence of the forward algorithm. Computer simulations indicate that this convergence rate is also $\mathcal{O}(1/\sqrt{t})$ (see e.g. the experiments of Xu and Yuille (Xu and Yuille, 1995)), so by Theorem 2.2 the convergence rate of the whole algorithm is $\mathcal{O}(1/\sqrt{t})$, which is just the same as the convergence rate of the LMS algorithm. Note that the actual convergence rate of both algorithms

depend on the eigenstructure of the correlation matrix $\mathbf{R}_{\xi, \xi}$.

Appendix

First we prove Theorem 2.1 through a series of simple lemmas and then consider the proof of Theorem 2.2. We start with a basic lemma concerning iterated linear inequalities.

LEMMA 3.1 *Let $\{k_n\}, \{a_n\}, \{b_n\}$ be sequences of real numbers and assume that the numbers k_n all satisfy the same inequality*

$$k_{n+1} \leq a_n k_n + b_n, \quad n = 0, 1, 2, \dots$$

Then

$$k_n \leq k_0 \prod_{i=0}^n a_i + \sum_{i=0}^n b_i \prod_{j=i+1}^n a_j. \quad (15)$$

Proof. By assumption $k_1 \leq a_0 k_0 + b_0$ and $k_2 \leq a_1 k_1 + b_1$. Substituting the estimate of k_1 into the latter inequality yields $k_2 \leq a_1 a_0 k_0 + a_1 b_0 + b_1$. Continuing in this way we get (15). $\square$

LEMMA 3.2 *Let X and Y be Banach spaces, $U_n : X \times Y \rightarrow X$ be a sequence of arbitrary mappings ($n = 0, 1, 2, \dots$), and let $\theta_n \in Y$ be a convergent series with limit θ_∞ (that is $\|\theta_n - \theta_\infty\|$ converges to zero). Further, let*

$$x_{n+1} = U_n(x_n, \theta_\infty), \quad (16)$$

and let

$$y_{n+1} = U_n(y_n, \theta_n), \quad (17)$$

with $x_0 = y_0$. Then

$$\|y_n - x_n\| \leq \sum_{s=0}^n \|\theta_s - \theta_\infty\| L_s^\theta \prod_{t=s+1}^n L_t^x, \quad (18)$$

where L_k^θ is the uniform Lipschitz index of $U_k(x, \theta)$ w.r.t. θ at θ_∞ and, similarly, L_k^x is the uniform Lipschitz index of $U_k(x, \theta_\infty)$ w.r.t. x .

Proof. The proof is the application of Lemma 3.1 to the sequence $\delta_n = \|x_n - y_n\|$. Indeed, by the triangle inequality we have that

$$\begin{aligned} \delta_{n+1} &= \|U_n(x_n, \theta_\infty) - U_n(y_n, \theta_n)\| \\ &\leq \|U_n(x_n, \theta_\infty) - U_n(y_n, \theta_\infty)\| + \|U_n(y_n, \theta_\infty) - U_n(y_n, \theta_n)\| \\ &\leq L_n^x \delta_n + L_n^\theta \|\theta_n - \theta_\infty\|. \end{aligned}$$

Thus δ_n satisfies the conditions of Lemma 3.1 with $a_n = L_n^x$ and $b_n = L_n^\theta \|\theta_n - \theta_\infty\|$. $\square$

Part 1 of Theorem 2.1 follows immediately from this lemma since $\|y_n - x_\infty\| \leq \|y_n - x_n\| + \|x_n - x_\infty\|$. Now let us consider the second part.

Firstly, some extra conditions are required to make $\|y_n - x_n\|$ converge to zero. The r.h.s. of (18) can be taken as the “triangle” transformation of the sequence $\|\theta_n - \theta_\infty\|$. If Δ_n is a sequence of numbers, and $a_{n,m}$, $0 \leq m \leq n$ is a triangular matrix then the transformed sequence

$$b_n = \sum_{m=0}^n a_{n,m} \Delta_m$$

is called the triangle transform of Δ_n with transformation sequence $a_{n,m}$. Triangle transforms of infinite series for which $\sum_{m=0}^n a_{n,m}$ is a constant independent of n have been analysed in (Pólya and Szegő, 1972). Here we consider a broader class. A transformation sequence $a_{n,m}$ is called *regular* if the following two conditions hold:

1. $\lim_{n \rightarrow \infty} a_{n,m} = 0$ for all m and
2. $\lim_{n \rightarrow \infty} (\sum_{m=0}^n a_{n,m}) = c$ with some $c > 0$.

The constant c is called the asymptotic value of the sequence $a_{n,m}$. Regular transformations actually keep the limit of the transformed sequence, as can be shown by the following lemma:

LEMMA 3.3 Let $\mathbf{a} = (a_{n,m})$, $a_{n,m} \geq 0$ ($n, m \in \mathbb{N}$, $0 \leq m \leq n$) be an arbitrary regular sequence with asymptotic value c . Let Δ_n be an arbitrary convergent series with limit Δ . Let b_n be the “triangle transform” of Δ_n with transformation sequence $a_{n,m}$:

$$b_n = \sum_{m=0}^n a_{n,m} \Delta_m.$$

Then b_n converges to $c\Delta$, and the following estimate holds as well:

$$|b_n - c\Delta| \leq |\epsilon_n \Delta| + 2cH_K + H_0 \sum_{m=0}^K a_{n,m}, \quad (19)$$

where $0 \leq K \leq n$ is arbitrary (but it may depend on n), $c_n = c - \sum_{m=0}^n a_{n,m}$, and

$$H_K = \sup_{n \geq K} |\Delta_n - \Delta|.$$

Proof. In the following sequence of estimates

$$\begin{aligned} |b_n - c\Delta| &= \left| \sum_{m=0}^n a_{n,m} \Delta_m - \left(\epsilon_n + \sum_{m=0}^n a_{n,m} \right) \Delta \right| \\ &\leq \sum_{m=0}^n a_{n,m} |\Delta_m - \Delta| + |\epsilon_n \Delta| \\ &\leq \sum_{m=0}^K a_{n,m} |\Delta_m - \Delta| + \left(\sum_{m=K+1}^n a_{n,m} \right) \sup_{i \geq K} |\Delta_i - \Delta| + |\epsilon_n \Delta| \\ &\leq H_0 \sum_{m=0}^K a_{n,m} + 2cH_K + |\epsilon_n \Delta|, \end{aligned} \quad (20)$$

where $0 \leq K \leq n$ is arbitrary, we have exploited the fact that $a_{n,m} \geq 0$. Hence (19) follows.

Now, let us consider the convergence of b_n . Pick an arbitrary $\epsilon > 0$. Then let us choose the number K so that $2cH_K < \epsilon/2$, and choose $N \geq K$ such that $|\epsilon_n \Delta| + H_0 \sum_{m=0}^K a_{n,m} < \epsilon/2$ is satisfied whenever $n \geq N$. Such numbers exist because H_K and ϵ_n converge to zero and also $a_{n,m}$ converges to zero as n tends to infinity for any fixed m . Then by (21) if $n \geq N$ then $|b_n - c\Delta| < \epsilon$. Since ϵ was arbitrary b_n converges to $c\Delta$. $\square$

The inequality relation (19) enables us to estimate the convergence speed of b_n . However, since the above inequality holds for *any* regular triangle transformation the resulting estimate will in general be rather crude. We note too that inequality (19) is strict in the *worst case sense*. This becomes evident when considering $\Delta_n = \Delta = 0$ for all n . If the convergence of $a_{n,m}$ towards zero is independent of m we can do much better in the derivation of the convergence speed as shown by the next lemma:

LEMMA 3.4 *Assume that $a_{n,m}$ decreases uniformly in m , i.e., that there exists a function $h(n)$ such that*

$$a_{n,m} \leq h(n) \quad (21)$$

holds for all n . Let $g : \mathbf{R}^+ \rightarrow \mathbf{R}$ be a monotonically decreasing, positive-valued integrable function which satisfies

$$\Delta_n \leq g(n). \quad (22)$$

Then the triangle transform of Δ_n , b_n , satisfies the inequality

$$b_n \leq h(n)G(n) + \mathcal{O}(h(n)), \quad (23)$$

where $G(x) = \int_0^x g(s)ds$.

Proof. From the mean-value theorem of integrals

$$0 \leq \int_0^n g(s)ds - \sum_{i=0}^{n-1} g(i) \leq g(0) - g(n) \leq g(0),$$

while from equations (21) and (22) we have that

$$b_n \leq h(n) \sum_{m=0}^n g(m).$$

Combining the above two inequalities yields the desired result. $\square$

Note that if $G(x)$ converges to zero then we get the result that $b_n = \mathcal{O}(h(n))$, while if $G(n)$ does not converge to zero then $b_n = \mathcal{O}(h(n)G(n))$. For example,

if $a_{n,m} = 1/(n+1)$ (which corresponds to averaging) and $\Delta_n \leq 1/(n+1)^s$ with $0 < s < 1$ then we get that $b_n = \mathcal{O}(1/(n+1)^s)$, i.e., averaging does not change the speed of convergence. Note that although the primitive function of $g(x) = \sqrt{\log \log x/x}$ cannot be written in a closed form, clearly

$$\int_6^T \sqrt{\log \log x/x} dx \leq \sqrt{T \log \log T}$$

which can be easily verified by differentiation. From these follow our extension of the iterated logarithm theorem of P. Major to cascaded Hebbian learning, touched upon in Section 2. In order to derive this and the convergence rate estimate of Theorem 2.1, simply observe that if $L_n^\theta \leq c/(n+1)$ and $L_n^x \leq 1 - 1/(n+1)$, then after some computation equation (18) will reduce to

$$\|x_n - y_n\| \leq \frac{c}{n+1} \sum_{s=0}^n \|\theta_s - \theta_n\|.$$

The following lemma completes the proof of Theorem 2.1:

LEMMA 3.5 *Let $x_n, y_n, L_n^\theta, L_n^x, \theta_n$ be the series as defined in Lemma 3.2. Assume that there exists a constant $C > 0$ such that*

$$L_n^\theta \leq C(1 - L_n^x) \tag{24}$$

holds⁴ for all $n \geq 0$ and also for all $k \geq 0$

$$\prod_{i=k}^{\infty} L_i^x = 0. \tag{25}$$

Then $\lim_{n \rightarrow \infty} \|x_n - y_n\| = 0$.

Proof. Let $\Delta_n = \|\theta_n - \theta_\infty\|$ and $a_{n,m} = L_m^\theta \prod_{j=m+1}^n L_j^x$ where we have used the convention again that an empty product (i.e., when $m \geq n$) is equal to 1. Observe that by Lemma 3.2 $\|x_n - y_n\|$ can be estimated from above by the triangle transform of Δ_n with coefficients $a_{n,m}$. Since $L_m^\theta \leq C(1 - L_m^x)$ then $a_{n,m} \leq$

⁴Note that this condition requires that $L_n^x \leq 1$ holds for all n since $L_n^\theta \geq 0$.

$b_{n,m}$, where $b_{n,m} = C(1 - L_m^x) \prod_{j=m+1}^n L_j^x$. Therefore, since $\sum_{m=0}^n a_{n,m} \Delta_n \leq \sum_{m=0}^n b_{n,m} \Delta_n$, it is sufficient to estimate the sum $\sum_{m=0}^n b_{n,m} \Delta_n$ from the above to obtain an upper bound on $\|x_n - y_n\|$. Now, observe that $(b_{n,m})$ satisfies the assumptions of Lemma 3.3: $b_{n,m} \geq 0$,

$$\lim_{n \rightarrow \infty} b_{n,m} = C(1 - L_m^x) \prod_{j=m+1}^{\infty} L_j^x = 0$$

and

$$\sum_{m=0}^n b_{n,m} = C \sum_{m=0}^n (1 - L_m^x) \prod_{j=m+1}^n L_j^x$$

which is a telescopic sum that reduces to $C(1 - \prod_{j=0}^n L_j^x)$ after expansion. Therefore

$$\lim_{n \rightarrow \infty} \sum_{m=0}^n b_{n,m} = C(1 - \prod_{j=0}^{\infty} L_j^x) = C > 0.$$

Thus Lemma 3.2 can be applied and we see that $\|x_n - y_n\|$ converges to zero.

□

Now let us consider a cascaded Hebbian algorithm of the form

$$\mathbf{v}_{t+1} = (1 - \alpha_t) \mathbf{v}_t + \alpha_t F(\mathbf{y}_t, \mathbf{x}_t, \mathbf{u}_t),$$

where the involved quantities satisfy the conditions of Theorem 2.2. First, observe that this corresponds to a diagonal process y_n of form (2), with $y_n = \mathbf{v}_n$, $\theta_n = \mathbf{u}_n$, and $U_n(y_n, \theta_n, \omega) = F(\mathbf{y}_n, \mathbf{x}_n, \mathbf{u}_n)$. (In order to make the differentiation easier between the symbols of the two contexts, symbols related to the cascaded Hebbian algorithm have a boldface typeset, while the symbols of the Banach-space context have a normal typeset.) U_n does not depend on θ_n so the Lipschitz constant of U_n w.r.t. θ_n is zero. Therefore the possible Lipschitz constants are $L_n^x = 1 - \alpha_n$ and $L_n^\theta = K\alpha_n$, where K is the Lipschitz constant of F w.r.t. $\mathbf{u}$ defined in Theorem 2.2. Since we assumed that $\sum_{t=0}^{\infty} \alpha_t = \infty$ the Lipschitz constants satisfy the conditions of Lemma 3.5 (inequality bound (24) and equation (25)) thus $\|\mathbf{v}_t - \mathbf{u}_t\|$ converges to zero, and hence by Lemma 3.2 $\|\mathbf{v}_t - \mathbf{v}^*\|$ also converges to zero (here $\mathbf{v}^*$ denotes the limit

of process $\hat{\mathbf{v}}_{t+1} = (1 - \alpha_t)\hat{\mathbf{v}}_t + \alpha_t F(\mathbf{y}_t, \mathbf{x}_t, \mathbf{u}^*)$ where $\lim_{t \rightarrow \infty} \|\mathbf{u}^* - \mathbf{u}_t\| = 0$). The convergence rate estimate follows from the properties of the Lipschitz constants.

References

- (Baldi and Hornik, 1989) Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:52–58.
- (Benaim, 1993) Benaim, M. (1993). The “off line learning approximation” in continuous time neural networks: an adiabatic theorem. *Neural Networks*, 6(5):655–665.
- (Benaim, 1996) Benaim, M. (1996). A dynamical system approach to stochastic approximations. *SIAM J. Control and Optimization*, 34(2):437–472.
- (Benaim and Tomasini, 1991) Benaim, M. and Tomasini, L. (1991). Competitive and self-organizing algorithms based on the minimisation of an information criterion. In *Artificial Neural Networks I*, volume 1, pages 391–396. North-Holland, Amsterdam.
- (Benaim and Tomasini, 1992) Benaim, M. and Tomasini, L. (1992). Approximating function and predicting time series with multisigmoidal basis functions. In *Artificial Neural Networks II*, volume 1, pages 407–411. Elsevier, Amsterdam.
- (Benveniste et al., 1990) Benveniste, A., Métivier, M., and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximations*. Springer Verlag, New York.
- (Chen and Chang, 1995) Chen, L. and Chang, S. (1995). An adaptive learning algorithm for principal component analysis. *IEEE Tran. on Neural Networks*, 6(5):1255–1263.

- (Fomin et al., 1994) Fomin, T., Szepesvári, C., and Lörincz, A. (1994). Self-organizing neurocontrol. In *Proc. of IEEE WCCI ICNN'94*, volume 5, pages 2777–2780, Orlando, Florida. IEEE Inc.
- (Gullapalli and Barto, 1994) Gullapalli, V. and Barto, A. (1994). Convergence of indirect adaptive asynchronous value iteration algorithms. In *Advances in Neural Information Processing Systems 6*, pages 695–702. San Francisco: Morgan Kaufmann.
- (Haykin, 1994) Haykin, S. (1994). *Neural networks: a comprehensive foundation*. MacMillen College Publ. Co., New York.
- (Hecht-Nielsen, 1987) Hecht-Nielsen, R. (1987). Nearest matched filter classification of spatiotemporal patterns. *Applied Optics*, 26(10).
- (Joutsensalo et al., 1995) Joutsensalo, J., Miettinen, A., and Zeindl, M. (1995). Nonlinear dimension reduction by combining competitive and distributed learning. In *Proc. of ICANN'95*, volume 2, pages 395–400, Paris.
- (Kushner and Clark, 1978) Kushner, H. and Clark, D. (1978). *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, Berlin, Heidelberg, New York.
- (Ljung, 1977) Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE Tran. Automat. Control*, 22:551–575.
- (Major, 1973) Major, P. (1973). A law of the iterated logarithm for the Robbins-Monro method. *Studia Scientiarum Mathematicarum Hungarica*, 8:95–102.
- (Martinetz and Schulten, 1991) Martinetz, T. and Schulten, K. (1991). A “neural-gas” network learns topologies. In *Proc. of ICANN*, volume 1, pages 397–402. Elsevier, Amsterdam.
- (Michaels, 1995) Michaels, R. (1995). Associative memory with uncorrelated inputs. *Neural Computation*, 8(2):256–259.

- (Moody and Darken, 1989) Moody, J. and Darken, C. (1989). Fast learning in neural networks of locally tuned processing units. *Neural Computation*, 1:281–582.
- (Nowlan, 1990) Nowlan, S. (1990). Maximum likelihood competitive learning. In *Proc. of Neural Information Processing Systems*, pages 574–582.
- (Oja and Karhunen, 1985) Oja, E. and Karhunen, J. (1985). On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *J. of Math. Anal. and Applications*, 106:69–84.
- (Palmieri and Zhu, 1995) Palmieri, F. and Zhu, J. (1995). Self-association and Hebbian learning in linear neural networks. *IEEE Tran. on Neural Networks*, 6(5):1165–1184.
- (Poggio and Girosi, 1990) Poggio, T. and Girosi, F. (1990). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:979–982.
- (Pólya and Szegő, 1972) Pólya, G. and Szegő, G. (1972). *Problems and Theorems in Analysis I*. Springer-Verlag Berlin.
- (Ritter et al., 1988) Ritter, H., Martinetz, T., and Schulten, K. (1988). Topology conserving maps for learning visuo-motor coordination. *Neural Networks*, 2:159–168.
- (Robbins and Monro, 1951) Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Ann. Mat. Stat.*, 22:400–407.
- (Rubner and Tavan, 1989) Rubner, J. and Tavan, P. (1989). A self-organizing network for principal component analysis. *Europhys. Lett.*, 10:639–689.
- (Sanger, 1989) Sanger, T. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 12:459–473.

- (Szepesvári et al., 1994) Szepesvári, C., Balázs, L., and Lőrincz, A. (1994). Topology learning solved by extended objects: a neural network model. *Neural Computation*, 6(3):441–458.
- (Wasan, 1969) Wasan, T. (1969). *Stochastic Approximation*. Cambridge University Press, London.
- (Widrow and Hoff, 1960) Widrow, B. and Hoff, M. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, pages 96–104, New York. Reprinted in: *Neurocomputing*, J.A. Anderson and E. Rosenfeld (ed.), The MIT Press, Cambridge, Massachusetts, 1988, pp. 126–134.
- (Xu, 1993) Xu, L. (1993). Least mean square error reconstruction principle for self-organizing neural-nets. *Neural Networks*, 6(5):627–648.
- (Xu and Yuille, 1995) Xu, L. and Yuille, A. (1995). Robust principal component analysis by self-organizing rules based on statistical physics approach. *IEEE Tran. on Neural Networks*, 6(1):131–143.