

NEURAL NETWORKS REVISITED: A STATISTICAL VIEW ON OPTIMISATION AND GENERALISATION

Andreas Wendemuth

Otto-von-Guericke-University Magdeburg
Institute for Electronics, Signal Processing and Communications Technology
PO Box 4120, 39016 Magdeburg, Germany, wendemu@iesk.et.uni-magdeburg.de

Keywords: Neural Networks, robustness, capacity, generalisation, statistical analysis

ABSTRACT

Statistical methods can be applied to analysis of Neural Networks to come up with on-average results for robustness, capacity, and generalisation in the presence of certain network architectures and data distributions.

In particular, the dynamics of generalization and learning of the Adaline training algorithm are calculated for correlated patterns. Modified algorithms are derived which restore the characteristic times found for uncorrelated patterns. For the most complex of single-layer networks, the perceptron with threshold and biased pattern outputs, a maximally stable solution is algorithmically obtained. The investigation of its generalization ability shows that this network is superior to existing ones. The discussion is extended to unlearnable data, introducing refined algorithms and showing the training and generalization characteristics. In multilayer networks, upper bounds to the storage capacity are calculated. A lower bound is obtained as well which leads to modest expectations with regard to the performance of algorithms in these architectures.

1. INTRODUCTION

Artificial Neural Networks have become a popular tool in Computer Science. They serve as classifiers for a wide variety of data, from signal processing to stock forecasts or dna sequencing. Neural networks are typically trained in a supervised fashion from sample data in order to obtain their classification power. However, very little is known about the robustness of training algorithms (e.g. w.r.t. data outliers), about their capacity (i.e. number of data which can be correctly mapped in training) and, most importantly, about their generalisation ability, i.e. their performance when classifying unknown data. This holds even for the simplest of network architectures and/or data distributions. Some worst-case theorems exist for those questions.

The past witnesses four consecutive cycles of enthusiasm and scepticism in the field of neural network models. For the first time in the 40's, McCulloch and Pitts [18] described a model of neural information processing which was used by Hebb [12] who opened the field to theoretical treatment. They obtained initial but far-reaching results, triggering a wave of interest into this new scheme.

In the 60's, at the time of the proof of the famous *Perceptron* training algorithm by Rosenblatt [28], artificial intelligence was a much (over-)used word. First successes were being reported, until the equally famous denouement of Perceptron training by Minsky and Papert [20] brought that period to an untimely end. For a third time in the 80's, the theoretical window opened by Hopfield [13] as well as Rumelhart and McClelland's *backpropagation* algorithm [30] ushered the modern and still ongoing area of multilayer networks. The fourth step being ongoing, neural networks are about to overcome their serious lack of generalisation by introducing Vapnik's theory of Structural Risk Minimisation into the training algorithms [31].

How can we try to understand neural networks from a point of view of statistics, and what are the methods needed to achieve it?

In the case of analysing algorithms and predicting the possible features of a given architecture, one makes use of the analogy between statistical mechanics and neural networks. The closest mathematical analogy is found in solid state physics for spin-glasses [16]. In neural networks the "spins" are the predefined (fixed, quenched) patterns, and one investigates the dynamics of the couplings. Most efficiently, one may calculate the volume in the phase-space of couplings available for a given set of patterns under the constraint of learned systems. This method has been introduced by Gardner [11], its principle is illustrated for one example below in eq. (18). Statistical Mechanics has served to obtain storage capacities, learning times, internal potential distributions, basins of attraction and generalization abilities and it shines light on the role of noise and of diluting the synaptic connections.

How do we know that our algorithms really converge, and where they converge to? This is the realm of mathematical proofs, in particular the use of optimization theory. In sections 3 and 4, new training algorithms for particular input data and for maximally stabilised storage are derived and proven. We find that what is described in neural network terms is indeed just a special case of multi-dimensional cluster separation. Therefore algorithms can be geometrically understood, and they have much wider applications than the training of neural networks alone.

Can multilayer networks be trained and analysed by the same methods? Here we face a serious difficulty in that algorithms used for single-layer networks cannot be simply rewritten for multilayer nets. However, it is also known that training algorithms like *backpropagation* work

well in these architectures. Trying to combine these facts, one finds that the correct definition of *internal pattern representations* in the hidden layers of the network allows us in section 5 to calculate what amount of information could be stored in a multilayer network. This calculation can be performed without knowing how to bring about the required internal representations in an algorithmic way. On the other hand, we may use the same technique to give a lower bound on what to expect from algorithms irrespective of their design. The methods used here are again very different from the ones we used before. The results are obtained by way of counting the number of Boolean internal representations as well as the number of possible dichotomies of a single hidden neuron's input space. Both results are then combined in a probabilistic fashion. This shows the application of three different methods at the same time.

We shall see that our results are dependent on the pattern statistics, namely the correlations between patterns. Therefore, we will consider biased patterns in all subsequent cases, either imposed directly on the patterns or indirectly via a biased teacher network. The difference to analysis with uncorrelated patterns is a major outcome of the research presented here: correlations enforce modified training algorithms, produce higher generalization ability and govern storage capacities in multilayer networks.

2. NEURAL NETWORKS

We shall now look at the general model used throughout for the description of neural networks.

The term "neural network" as well as the term "neuron" is obviously taken from the biological equivalent. The simplest model of a neuron which neurophysiology provides is that of a *linear threshold unit* which works as follows [15]. Over a certain interval of time, the neuron integrates (adds) all pulses $\xi_j(t)$ sent from previous neurons. The pulses are transmitted in an electrochemical way via so-called synapses which link different neurons. These synapses are more or less effective: their pulses are transmitted at a certain speed and with a certain loss of amplitude. In a crude fashion, this effectivity can be described by a synaptic strength J_{ij} which is a factor multiplying the total pulse sent from neuron j in a certain time interval, the product then is the total pulse arriving at neuron i . In general, $J_{ij} \neq J_{ji}$ since the synapses are directed bonds.

The sum of the pulses from previous neurons j , graded by the synaptic strengths J_{ij} , arriving at neuron i in a certain time interval is called *activation*. Due to this activation, the postsynaptic neuron i can now flip into a certain state $\tau_i(t)$. If the activation exceeds a certain threshold value T_i , the neuron i will be active or firing, otherwise it will remain quiescent, which can mathematically be modelled by a threshold function, e.g. *sign*. After some time delay, this neuron state will be transmitted on to other neurons which react in the same fashion:

$$\tau_i(t+1) = \text{sign} \left(-T_i + \frac{1}{\sqrt{N}} \sum_{j=1}^N J_{ij} \xi_j(t) \right) \quad (1)$$

With this neuron model, we may describe autoassociative and feedforward networks. In autoassociative networks, all neurons are connected to each other, forming a grid. The function of the connections is to embed certain stable configurations ("maps") within this grid. Due to neural activity, perturbed versions of those configurations will relax into the closest embedded map. This is very useful e.g. for image reconstruction. In contrast, feedforward networks are classifiers. The information at the neural input layer is "fed forward" through zero or more intermediate layers into outputs which are specified according to some classification scheme. This is very useful in decision making of any kind, e.g. control applications or options assessments.

In feedforward networks, it is helpful to have some further notations in order to keep layers separate. A general model with N neurons in the input layer may then look as in figure 1, where the "dots" represent the neuron's information transfer given by eq. (1).

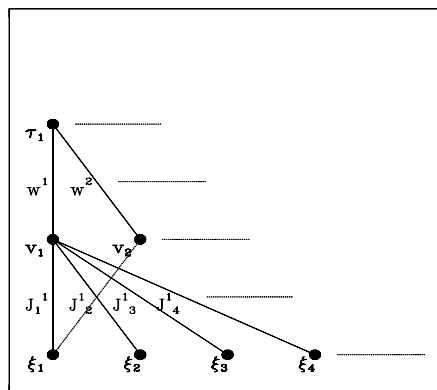


Figure 1: Feedforward Neural network with input layer (ξ_j), hidden layer (v_k) and output layer (τ_i). The synaptic strengths are given by J_j^k in the first layer and by w^k in the output layer.

We will consider here the particular mode of *supervised learning*. A *teacher* provides the student with p input configurations or patterns $\underline{\xi}^\mu$ and the corresponding desired outputs or targets τ^μ . The student network is to learn these mappings by adjusting its interior synaptic strengths or weights. Mathematically, this can happen through an explicit computation of the weights, see e.g. section 3. This calculation is rather intricate. It is however desirable from a biological point of view to have local, mathematically easy and iterative procedures of learning. This is achieved by repeated presentation of the set of patterns and outputs to the student net which adapts its weights in an algorithmic fashion.

One can impose further requirements on the student network. It can be asked to learn the patterns with maximum *margin*, i.e. maximum distance of the activation from the threshold. If the teacher's outputs are drawn according to a certain *rule*, the student may as well be asked to learn with maximum *generalization ability*, this is with maximum probability of obtaining the output to an arbitrarily selected pattern correctly. And finally, the student

may be asked to find a threshold independently which satisfies these requirements optimally. This most complex task in single-layer learning is investigated in section 4.

An additional complexity arises in multilayer networks. Here the states of the neurons in the hidden layers are not given by the teacher. We shall call these states *internal representations* $\{v_k^\mu\}$, use of that concept will be made extensively in section 5. Obviously, the student has to isolate proper internal representations in order to store the teachers example patterns.

3. ADALINE LEARNING

The eqns. 1 are solved for nodes without threshold for a weight vector $\underline{\mathbf{J}}$ and a given set of N -dimensional patterns $\{\underline{\xi}_1, \dots, \underline{\xi}_p\}$ and corresponding outputs ("targets"), if

$$\tau_\mu \underline{\mathbf{J}} \cdot \underline{\xi}_\mu = 1 \quad ; \quad (\mu = 1 \dots p) \quad , \quad (2)$$

i.e. activation $\underline{\mathbf{J}} \cdot \underline{\xi}_\mu$ and desired output τ_μ must have the same sign for all patterns according to eq. 1. This is sufficient but not necessary; necessary conditions will be given in section 4. The set of p linear equations (2) can be solved exactly if $p \leq N$. One therefore defines the *capacity* of a network by $\alpha = p/N$, i.e. the number of patterns per input dimension which can be correctly stored. For $\alpha < 1$, eqns. (2) are underdetermined. One may then choose the solution with minimum norm $|\underline{\mathbf{J}}|$ which is known as the *pseudoinverse* and given as follows [25]: Denote the pattern matrix $\underline{\mathbf{S}}$ by $(\underline{\mathbf{S}})_{\mu j} = (\tau_\mu \xi_{\mu j})_j$, and a p -dimensional vector $\underline{\mathbf{1}}$ with all elements 1, then satisfying (2) is performed with an ansatz (3), yielding:

$$\underline{\mathbf{J}} = \underline{\mathbf{S}} \cdot \underline{\mathbf{x}} \quad (3)$$

$$\underline{\mathbf{1}} = \underline{\mathbf{S}}^T \cdot \underline{\mathbf{J}} = \underline{\mathbf{S}}^T \cdot \underline{\mathbf{S}} \cdot \underline{\mathbf{x}} \quad , \quad (4)$$

$$\underline{\mathbf{J}} = \underline{\mathbf{S}} \cdot (\underline{\mathbf{S}}^T \cdot \underline{\mathbf{S}})^{-1} \cdot \underline{\mathbf{1}} \quad . \quad (5)$$

The pseudoinverse solution has the advantage that it lives exclusively in the space spanned by the patterns themselves, hence any "noise" in the null space perpendicular to the patterns will be ignored. This makes the neural network robust against noise.

Under a neural training paradigm, we would like iterative solutions as stated above. There are (at least) three other reasons for iteration:

1. Direct computation as in eq.(5) requires $\mathcal{O}(N^2)$ operations
2. For $\alpha > 1$, the pseudoinverse solution does not exist.
3. There is no intermediate "approximate" solution before all calculations are finished.

We would therefore like to have one iteration procedure for any α , which for $\alpha > 1$ should reach a measurable accuracy of eqns. (2), for $\alpha \leq 1$ converge to the pseudoinverse solution, and which in all cases should require $\mathcal{O}(N)$ operations regardless of the pattern statistics. The latter point is crucial in case of correlated patterns, as we shall see shortly. This iteration procedure is called the "*Adaline Algorithm*" [38].

The idea is to minimize the quadratic error E of the mappings E_μ of all patterns:

$$E_\mu = 1 - \tau_\mu \underline{\mathbf{J}} \cdot \underline{\xi}_\mu \quad ; \quad E = \sum_\mu E_\mu^2 \quad (6)$$

by gradient descent $\delta \underline{\mathbf{J}} = -\frac{\gamma}{N} \text{grad}_{\underline{\mathbf{J}}} E$ with positive gain parameter γ . This gives

$$\delta \underline{\mathbf{J}}(t) = \frac{\gamma}{N} \sum_\mu E_\mu(t) \tau_\mu \underline{\xi}_\mu \quad (7)$$

Starting with $\underline{\mathbf{J}}(t=0) = \underline{\mathbf{0}}$, after T iteration steps the weight vector is given by

$$\underline{\mathbf{J}}(T) = \sum_\mu x_\mu(T) \tau_\mu \underline{\xi}_\mu \quad \text{with} \quad x_\mu(T) = \frac{\gamma}{N} \sum_{t=1}^T E_\mu(t)$$

which is of the same structure as eq. (3). The $x_\mu(T)$ are called *embedding strengths*, since they describe the contribution of pattern $\underline{\xi}_\mu$ to the Adaline solution $\underline{\mathbf{J}}(T)$. It has been shown ([7], [33]) that for capacities $\alpha \leq 1$ Adaline converges to the Pseudoinverse solution, for $\alpha > 1$ it reaches the global minimum of eq. (6).

This leaves us with showing that Adaline is "fast", as required. It has been shown in [33] that the error E of the mappings, eq. (6), decays as

$$E(t) = E(\infty) + E(0) \exp(-t/\tau) \quad , \quad (8)$$

where $E(\infty)$ is the best possible error ($= 0$ for $\alpha \leq 1$) and τ is the characteristic error decay time or, shorter, learning time. It can be minimized by choosing a suitable gain factor γ which depends on the pattern correlations. Using methods which are equivalent to minimizing the Free Energy in Statistical Mechanics, it has been shown in [7] that for unbiased distribution of patterns, the optimal choice is

$$\gamma_{opt} = (1 + \alpha)^{-1}; \quad \tau_{min} = \frac{1}{2 \ln \left(\frac{1+\alpha}{2\sqrt{\alpha}} \right)} \quad (9)$$

For $\alpha \rightarrow 0$ or $\alpha \rightarrow \infty$, $\tau \rightarrow 0$ since either few patterns have to be learnt ($\alpha \rightarrow 0$) or so many ($\alpha \rightarrow \infty$) that patterns quasi-homogeneously fill the entire space, hence $E(\infty)$ is extremely high and has almost the same value everywhere, which is found quickly by Adaline training. Eq. (9) shows that learning times are of order $\tau = \mathcal{O}(1)$, hence a required high accuracy $E(t) - E(\infty)$ of order $\mathcal{O}(\exp(-N \cdot \text{const.}))$ can be reached with $\mathcal{O}(N)$ many steps.

However, these encouraging results do not hold any more for correlated patterns [33] since then these correlations sum up over training, resulting in $\tau = \mathcal{O}(N)$. Let us model correlations through a set of training patterns each of which has the bias

$$m_\mu = \frac{1}{N} \sum_{j=1}^N \xi_j^\mu \quad . \quad (10)$$

The average correlation between patterns is then $C = \frac{1}{p^2} \sum_{\mu=1}^p \sum_{\nu=1}^p m_\mu m_\nu$. It was shown in [34] that two modifications to the Adaline algorithms will restore the unbiased results. Define a vector $\underline{\mathbf{1}}$ with all components equal to 1, then

1. Let $\delta\tilde{\mathbf{J}}(t) = \frac{\gamma}{N} \sum_{\mu} E_{\mu}(t) \tau_{\mu} (\underline{\xi}_{\mu} - m_{\mu} \underline{\mathbf{1}})$

2. Let $\tilde{\gamma} = \gamma^{unbiased} / (1 - C)$

In other words, the updates of \mathbf{J} are taken from artificial patterns with their bias removed, and the gain factor has to be corrected for the correlations. With these modifications, the results eq (9) are restored. Derivations of convergence are given in [34], and the optimality of learning time was again derived using methods from Statistical Mechanics in [33].

Let us now look at the generalization ability $G(t)$ if the *Adaline* algorithm is used to train the neuron with $p = \alpha N$ many examples $\underline{\xi}^{\mu}$ whose outputs τ_{μ} are given by a *teacher network* $\underline{\mathbf{B}}$ according to the rule

$$\tau^{\mu} = \frac{1}{\sqrt{N}} \sum_{j=1}^N B_j \xi_j^{\mu} \quad (11)$$

The generalization ability G is defined as the probability that, after learning, the student's answer S_{μ}^o to any question $\underline{\mathbf{S}}_{\mu}$ is given correctly ($S_{\mu}^o = \tau_{\mu}$), i.e. in accordance with the teacher's rule. Defining a threshold function $\Theta(x) = \{0(x < 0); 1(x \geq 0)\}$, we can write:

$$G(U, \alpha) = \frac{1}{2^p N} \sum_{\{S_j^{\mu} = \pm 1\}} \Theta(\tau_{\mu} S_{\mu}^o) \quad (12)$$

With the normalization $\underline{\mathbf{B}}^2 = N$, and the *bias* of $\underline{\mathbf{B}}$ (the "teacher bias") $\tilde{B} = \frac{1}{N} \sum_i B_i$, one obtains [33] with methods again derived from Statistical Mechanics, for all pattern distributions:

$$G \Big|_{m=0, t=\infty} = 1 - (1 - \alpha) \cdot \Theta(1 - \alpha) \quad (13)$$

$$G \Big|_{m \neq 0, t=\infty} = 1 - [1 - \tilde{B}^2] (1 - \alpha) \cdot \Theta(1 - \alpha) \quad (14)$$

Note that generalization is better in the case of teacher bias. However, this can only be achieved if the patterns themselves are biased ($m \neq 0$). The explanation is that the student network has to positively correlate patterns with the weight vector, this is only possible if the patterns are biased or (!) if the student network has an adjustable threshold. We will study the latter in section 4. For $\alpha > 1$, generalisation is perfect. This is a consequence of the teacher network having the same structure as the student network. This structure being linear, it can be fully "detected" by the student after $p = N$ lin. indep. examples.

As for dynamics, choosing γ as in eq. 9 is optimal for generalisation as well, and gives exactly the same decay times for the generalization as for the training error [33].

In summary, we have been able to provide fast and accurate algorithms for any pattern distribution, using statistical methods. We optimally exploit bias in our formulae for fast training and generalisation, and for high generalisation ability. Altogether, we have given new tools for optimal application of the Adaline algorithm, which in numerical mathematics is equivalent to a Gauss-Jordan procedure and therefore still is very popular.

Adaline learning as described above has the nice feature that it is a linear procedure, and closed solutions are available hence.

However, one might ask for solutions which maximize the margin between different output clusters. This is especially interesting since it has been shown that margin maximization is required for a much more powerful class of networks called support vector machines [31].

In this case, one might resort to optimization-theoretic methods [10]. In this context, learning might as well be defined as a linear or quadratic programme. However, these calculations are nonlocal, complex and algebraic. Algorithmic learning is possible as well. A first step was the famous *Perceptron* training algorithm [28] which will always find a solution if one exists.

Learning procedures for the perceptron of optimal margin have been studied by a number of authors (e.g. [1], [17]). Recently, generalized procedures have been presented which solve the optimal stability problem even if an adjustable threshold is included as introduced in [29]. This general problem can be formulated as follows: find a perceptron vector $\underline{\mathbf{J}}$ and a threshold T such that for all patterns:

$$|\underline{\mathbf{J}}|^2 = \min ; \left(\frac{\underline{\mathbf{J}} \tau_{\mu} \underline{\xi}_{\mu}}{\sqrt{N}} - T \tau_{\mu} \right) \geq c \equiv \Delta |\underline{\mathbf{J}}| = 1 \quad (15)$$

Eqns. (15) can be understood as follows: all patterns are mapped to their correct output, hence $\geq c$. In contrast to eq. (2) for Adaline, here we have *inequalities* which make the conditions (15) necessary and sufficient: patterns "far away" from c are mapped correctly without equality to c being forced. With c fixed (arbitrarily) to 1, and $|\underline{\mathbf{J}}|^2$ being minimized, the margin 2Δ between pos. and neg. mapped patterns is maximized, giving good robustness against perturbations. This of course only holds if the problem is *linearly separable*, i.e. if a hyperplane with normal vector $\underline{\mathbf{J}}$ and distance from the origin T exists which separates the patterns with pos. and neg. output mappings. This situation is very unsatisfying, since linear separability is not known a priori. Algorithms have been devised [22] which *detect* non-linear-separable cases but do not provide any approximate solution then. Our aim therefore is to find an algorithm which, without prior inspection of linear separability, will automatically either find a maximum margin solution (15) if one exists, or a robust solution which ignores misclassified patterns far away from the separating hyperplane.

In order to motivate the operating principle of such an algorithm, we shall have a look at eqns. (15) again from the point of view of optimization theory [10]. There, it is known that the solution to this problem is derived from an optimal *subset* of patterns for which eqns. (15) hold with equality, whereas for the remaining patterns strict inequality holds. There exist *optimality conditions* $\underline{\lambda} > \underline{\mathbf{0}}$ which guarantee the uniqueness of the optimal solution. The optimal subset can be found by regarding intermediate solutions drawn from *active sets*. Details of these procedures can be found in [10].

It is demonstrated in [35] that both the minimization eq. 15 and the search for active sets can be performed in one single algorithm, which also works if the two output classes are not linearly separable! This can be achieved by taking into account two effects:

1. The updates of the weight vector as in eq. 7 add portions of the training vectors to the weight vector. These portions get smaller as the training error decreases. The additions amount to *turning* the weight vector in space towards erroneous training vectors.
2. For inseparable sets of training vectors, these methods fail. It is known that the reason for failing is that the magnitude of the abovementioned turning does not converge, since there are always training vectors which are not correctly mapped. This destroys previously learned vectors.

The (mathematically proven in [35]) solution of this dilemma is twofold:

1. to force the magnitude of turning to decrease even if the mappings are not correct (yet). However, if a separable solution exists, it has to be found. Both is achieved simultaneously by turning and rescaling, the latter being possible by upscaling of \mathbf{J} .
2. Only update with the training vector which is "just" incorrectly mapped. Mathematically, this is the vector which least violates eq. 15.

This vector being ξ_{μ} , updates then look as follows (compare eq. 7)

$$\delta \mathbf{J}(t) = \frac{\gamma}{N} \tau_{\mu} \xi_{\mu} + \eta \mathbf{J}(t-1) \quad (16)$$

where γ, η do not have to be adjusted, and are given explicitly in [35].

Let us now look again from a statistical point of view at the generalisation ability of such networks with thresholds. In order to shine more light on the effect of thresholds, we will again assume a *teacher network* with weight vector \mathbf{B} and threshold U , providing the rule:

$$\tau_{\mu} = \text{sign} \left(\frac{1}{\sqrt{N}} \mathbf{B} \cdot \xi_{\mu} - U \right) . \quad (17)$$

The student network eq. (1) *learns* $p = \alpha N$ examples of this rule by selecting his network weights \mathbf{J} and his threshold T such that the examples are stored with maximum margin $\Delta = \Delta_{\max}$ according to eq. (15). Obviously, due to identical structure of the teacher and student network, the problem is linearly separable.

A calculation [11] of the volume V available for a solution to eq. (15) in the phase-space (\mathbf{J}, T) with margin $\Delta > 0$ will show the desired generalisation (eq. 12). At this point, we give some details of the technicalities of these calculations since all "Statistical Mechanics" style results in this paper follow these lines. Readers who do not find this instructive to see may directly consult fig. 2 below for the results.

$$V = \int_{-\infty}^{+\infty} dT \prod_{j=1}^N dJ_j \delta(|\mathbf{J}|^2 - N) \times \quad (18)$$

$$\prod_{\mu} \Theta \left[-\Delta + \left(\frac{\mathbf{J} \cdot \xi_{\mu}}{\sqrt{N}} - T \right) \text{sign} \left(\frac{\mathbf{B} \cdot \xi_{\mu}}{\sqrt{N}} - U \right) \right]$$

The integration is performed over a \mathbf{J} -sphere in order to have a bounded measure. The argument of the Θ -function indicates that teacher and student mappings match, and that a minimum margin of 2Δ between the two output classes is achieved. Eq. (18) is then averaged over $p = \alpha N$ many, randomly selected examples which bound the space for possible (\mathbf{J}, T) . At fixed α , when increasing the margin Δ the available volume of this space shrinks further until at zero volume, maximum margin $\Delta_{\max}(\alpha, U)$ is obtained. The generalisation ability eq. (12) can then be given as a function $G = G(\Delta_{\max}) = G(\alpha, U)$. Details of this approach are given in [36], the results are as follows:

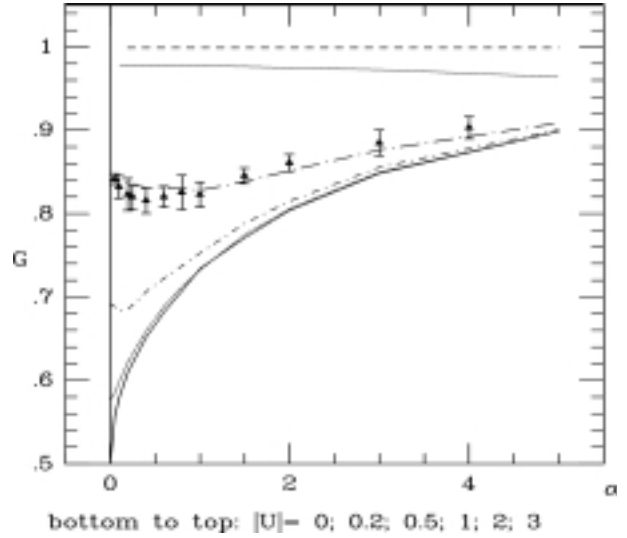


Figure 2: Generalization ability G as a function of capacity α for various teacher thresholds U . Theoretical results are given as solid lines. Generalization converges to 1 for $U \rightarrow \infty$. For $U = 1$, simulation results (triangles) and standard deviations (bars) are given as well for a network with 100 neurons, averaged over 40 runs. For any α, U , this network was trained to optimal stability. Then 500 question vectors and outputs were selected randomly, and G was obtained as the fraction of correct assignments by the network.

It is first of all striking that in networks with threshold, the generalization ability initially *decreases* with α . This effect was not observed in networks without thresholds [23]. Furthermore, generalization is improved by an optimally chosen threshold throughout. In particular, the generalization ability at $\alpha = 0$ is already higher than the "random" result $G = 0.5$ and increases with the teacher's threshold. These effects demand some detailed interpretation.

For small α , there is very few information at hand for the student about the direction of \mathbf{B} . However, since $U \neq 0$, the student will detect already from a few teacher's answers that the output is biased. Thus a *statistical* best choice is to set *all* outputs to one value according to the teacher's bias. One can evaluate G at small α . With the parameter $m = 2 \text{erf}(|U|) - 1$, which is identical to the

absolute output bias, one obtains

$$G = \frac{1+m}{2} - \frac{1}{\sqrt{2\pi}} \sqrt{\alpha} \sqrt{1-m^2} \exp \left\{ -\frac{m^2}{2\alpha(1-m^2)} \right\}$$

which shows the behaviour as in fig. 2. Similarly, for high $\alpha \gg e^{U^2/2}$ one obtains in leading order

$$G \simeq 1 - \frac{0.501}{\alpha}, \quad (19)$$

where the value 0.501 is obtained analytically (see [36]). The derivation of this number is valid for networks and rules with or without threshold. Note that G does not depend on U to leading order in $1/\alpha$ which clearly shows the *deterministic* character of generalization: for large α , the examples fill the input space quasi-homogenously. Furthermore, note that the generalization ability almost reaches the Bayes optimal result of $G = 1 - 0.44/\alpha$ which was obtained numerically in [24]. Thus Bayes-optimal learning will improve generalization only very slightly, but at the cost of sampling a (high) number of possible solutions in the full (\mathbf{J}, T) -space allowed for by eq. (15) [32]. Note that in contrast to the Adaline algorithm eqns. 13,14, we cannot achieve perfect generalization despite identical structure of teacher and student networks. The reason is that the networks are now nonlinear.

The theoretical results for G can be verified by simulations, see fig. 2.

In summary, we have provided algorithmic tools which overcome one of the most serious drawbacks of perceptron learning, the requirement of linear separable problems. This drawback had almost stopped neural network research altogether in 1969, through the famous publication by Minsky and Papert [20] mentioned already above. We have seen that our algorithms provide the generalisation ability predicted by theory. That is, we can assess through the output bias of our samples (via U) which generalisation we can expect after presenting p samples to the network. Altogether, this enables us to take full advantage of perceptron learning which is superior to the previously considered Adaline learning.

5. MULTILAYER NETWORKS

For multilayer networks, the problem of finding internal representations is most crucial. The corresponding mathematical task has been shown to be non-polynomial-complete [14], which makes it very unlikely that an algorithm will be devised which solves the problem. However, numerous approaches have been made to tackle the learning task. The most prominent one is the *backpropagation* algorithm [30] which applies gradient descent procedures and alternates between forward and backward processing of the learning error. Several alterations to backpropagation exists which make it faster, e.g. second order learning [9] or the inclusion of momentum terms [26]. The *moving targets* algorithm [27] directly aims at finding internal representations. If the architecture is flexible, *tiling* methods [19] have been devised which will always solve the learning problem, however the number of hidden units may become excessively large. In this spirit, a *Boolean* network may always be used.

Let us consider a multilayer perceptron as in figure 1. We shall be interested in its learning capacities. Denote the *internal representation* of pattern μ at node k by v_k^μ . Let F be the “output function” which maps $\{v_k^\mu = \pm 1\} \rightarrow \pm 1$. In particular, F can for example be a neural architecture of one additional layer with fixed weights w_k and a fixed threshold T in the second layer, i.e. $F = \text{sign}(T + \sum_k w_k v_k)$. Then the network equations for the multilayer network under consideration can be written:

$$\begin{aligned} \tau_\mu &= F(\{v_k^\mu\}) \\ \text{where } v_k^\mu &= \text{sign} \sum_{i=1}^N J_i^k \xi_i^\mu ; \forall \mu. \end{aligned} \quad (20)$$

N is the number of input neurons accessible to one node in the first hidden layer. For a tree-like architecture, N therefore has to be replaced by N/K , which is not necessary if the first hidden layer is fully connected (FC) to the input layer.

Several approaches have been made to apply statistical analysis to this type of network. In Computational Learning Theory [2], the generalization probability P is inspected, that any learning algorithm matches unknown patterns with error $< \varepsilon$. It is found that, if $P > 1 - \delta$ is required, $p \geq \frac{N}{\varepsilon} (\ln 2 - \frac{\ln \delta}{N})$ many examples are needed to achieve this accuracy, regardless of distribution of mappings and patterns [4]. In other words, for high generalization the number of required examples of the underlying pattern distribution scales with N , but diverges as $1/\varepsilon$. This is a worst case result, since it holds for any pattern distribution. It is, however, the basis for support vector machines [31].

We instead investigate capacities (rather than generalization) of multilayer networks. This will show to be a problem which is easier to handle. The capacity α is always taken as p/N , where N is the full number of neurons in the input layer. The following analysis provides results in the large- N limit for the fully connected architecture. Upper and lower bounds to storage capacities are known for some particular output functions ([5], [21]). Here, we would like to extend these results to networks with arbitrary (but fixed) output statistics and output functions. Of course the known results must hold as special cases. As we shall see shortly, the networks we consider can be fully characterised by few parameters which allow for general formulae for upper and lower capacity bounds which we define as $\underline{\alpha}$ and $\bar{\alpha}$.

Notice first that since F is a function of the K binary variables v_k , and since F takes binary values only, it can always be written in its Boolean conjunctive normal form, i.e.

$$F = \sum_{j=1}^R \left[\prod_{k=1}^K \overline{v_k} \right]_j, \quad (21)$$

where the variables v_k appear either non-negated or negated (overlined) in each product j , and no two products equal. R then is the number of Boolean K -products that lead to output one. In order to produce the required output values, it is legitimately possible to store p patterns through

$$W =: R^{p(1+m_{out})/2} \cdot (2^K - R)^{p(1-m_{out})/2} \quad (22)$$

many *sets* of internal representations $\{v_k^\mu\}$. One of these sets thus contains one possible internal representation of all patterns at all hidden layer nodes, i.e. $K \cdot p$ elements. However, not every such set can be learnt by adjusting the weights of the input layer. We will therefore look at the *learning probability* $P = P(p, N, m_{out}, F, \text{architecture})$, i.e. the probability that p random patterns with output bias m_{out} can be learnt by a network of given architecture (tree, FC) with N input units and a function F relating the K hidden nodes to the output.

Cover [6] has shown that a single neuron can produce $C(p, N)$ many *dichotomies*, i.e. number of possible sets of output assignments $\{v^\mu\}$. As a lower bound, we may further use [37] that the number of dichotomies which are accessible *with certainty* is $C = \exp(2p \ln 2/\alpha)$. We can make use of these quantities as follows: The learning probability P is given as the probability that *at least one* of the W correct sets of internal representations is reached by the C^K many sets of internal representations accessible by the input layer, where these sets are located anywhere within the 2^{pK} many possible sets of internal representations. The storage capacity is then found at $P = 0.5$. This translates our problem into a combinatorial one. The solution is given implicitly [37], with $r = R/2^K$:

$$1 \geq \frac{(1 + m_{out}) \log_2(r) + (1 - m_{out}) \log_2(1 - r)}{-2K}$$

$$= \frac{\log_2(\bar{\alpha})}{\bar{\alpha}} - \left(1 - \frac{1}{\bar{\alpha}}\right) \log_2\left(1 - \frac{1}{\bar{\alpha}}\right) = \frac{2}{\underline{\alpha}} \quad (23)$$

Note that dependency on the output layer function F has been reduced to dep. on r . Let us now look at two prominent examples which have been studied by a number of authors ([8], [21]). The parity machine is defined as $F = \prod_k (v_k)$ and therefore

$$r^{(P)} = 2^{-K} \sum_{j=0}^{(K-1)/2} \binom{K}{2j} = \frac{1}{2} \quad (24)$$

The committee machine is defined as $F = \text{sign}(\sum_k v_k)$:

$$r^{(C)} = 2^{-K} \sum_{j=0}^{(K-1)/2} \binom{K}{j} = \frac{1}{2} \quad (25)$$

As the names suggest, for the parity machine the product of all signs provided by the hidden layer neurons must be 1. The committee machine emulates a perfect committee which decides in favor (1) of a candidate (pattern) if more than half of their members (hidden nodes) are in favor (1).

Evaluating eqns. 23 gives the following results which are compared with other calculations and experiments. Here the terms "replica symmetric" and "1 step replica symmetry breaking" refer to technical procedures in statistical calculations of Statistical Mechanics style, the latter being more accurate. Details are provided in [3] and [8].

Table 1 clearly shows that replica-symmetric calculations violate the upper capacity bound for $K \geq 7$. However, the large K behaviour shows that our bounds derived from combinatorial approaches are tight only for the parity machine. This can be understood in the light of considerations given e.g. in [8]: there is no "clique building" for the

K	$\bar{\alpha}$	α_{RS}	α_{1RSB}	$\alpha_{Sim.}$	$\underline{\alpha}$
1	2	2	-	2	2
$3^{(P)}$	16.3	30.9	15.0	-	6
$3^{(C)}$	16.3	12.1	9.0	6.5	6
5	32.2	28.9	-	10.9	10
7	49.5	51.1	-	15.3	14
9	67.8	78.3	-	20.0	18
large	$\rightarrow K \ln(K)$	$(P) \propto K \ln(K)$	-	-	$2K$
		$(C) \propto K \sqrt{\ln(K)}$			

Table 1: Storage capacity bounds $\bar{\alpha}$, $\underline{\alpha}$, replica symmetric calculation (RS), one step replica symmetry breaking (1RSB) and simulation results (Sim) for various K in the fully connected architecture [8]. The data for the parity machine is taken from Barkai et al. [3].

hidden nodes in the parity machine: any sign flip in one node will change the result. In contrast, if in a committee machine a majority of the hidden nodes already "vote" erroneously for a pattern, the other nodes are not able to change that decision. Hence errors can be less easily corrected, and consequently the storage capacity is lower.

Furthermore, simulation results are just slightly higher than $\underline{\alpha}$. The reason is that gradient descent or similarly constructed algorithms can only be expected to give results in the vicinity of $\underline{\alpha}$. One has to use very elaborate search techniques in the space of internal representations to improve the storage capacity. For example, the moving targets algorithm of Rohwer [27] works along these lines.

The strength of the capacity derivation given in this section are therefore not good bounds for particular, well investigated output functions. Table 1 showed that in these cases our bounds are at least fair. The strength of the bounds provided in eq. (23) is rather that it is uniquely applicable to any type of output function and output bias. This makes it a powerful tool for network design. In particular in using the conservative lower bound, one can assess for a required Boolean output characteristic the number of hidden nodes necessary for storage of patterns with particular output bias with high confidence. These results have again been obtained by statistical and combinatorial methods.

6. SUMMARY

This paper takes a new look at neural networks from a statistical point of view. Many illustrative features have been derived under this concept both for single-layer and multilayer networks. This produces results both for optimal design of training algorithms, and for analysis of the powers such networks have. These results could not have been brought about by algebraical methods, hence we hope to have shone a new light on the treatment of neural networks.

Acknowledgements

The author was recipient of Fellowship ERB CH1 BCT93 0500. His MCFA Member Number is 1767.

7. REFERENCES

- [1] J.K. Anlauf and M. Biehl: *Europhys.Lett.* **10** (1989), 687.
- [2] M. Anthony and N. Biggs: *Computational Learning Theory* Cambridge 1992
- [3] E. Barkai, D. Hansel and I. Kanter: *Statistical Mechanics of a Multilayered Neural Network* , *Phys. Rev. Let.* **65** **18** (1990), 2312.
- [4] E. Baum and D. Haussler: *What size net gives valid generalization?* *Neural Computation* **1** (1989), 151.
- [5] M Budinich and E. Milotti: *Properties of feedforward neural networks* *J. Phys. A* **25** (1992), 1903.
- [6] T.M. Cover: *Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition* , *IEEE Trans. Elec. Comp.* **EC-14** (1965), 326.
- [7] S. Diederich and M. Opper: *Phys.Rev.Lett.* **58** (1987), 949, and M. Opper: *Phys.Rev.A* **38** (1988), 707.
- [8] A. Engel, D. Malzahn, I. Kanter: *Storage properties of correlated perceptrons* *Minerva Workshop on Mesoscopics, Fractals and Neural Networks* **Eilat** (1997), 25.
- [9] S. E. Fahlmann, in: *Advances in Neural Information Processing Systems II* D. S. Touretzky (ed.), Morgan Kaufmann, San Mateo 1990.
- [10] R. Fletcher: *Practical methods of optimization* , John Wiley, New York 1987.
- [11] E. Gardner: *The space of interactions in neural network models* , *J. Phys. A* **21** (1988), 257.
- [12] D.O. Hebb: *The Organization of Behavior* Wiley, New York 1949.
- [13] J. J. Hopfield: *Neural Networks and physical systems ...* *Proc. Natl. Acad. Sci. USA* **79** (1982), 2554.
- [14] J.J. Judd: *Neural Network Design and the Complexity of Learning* , MIT Press, Cambridge, Massachusetts, 1990.
- [15] B. Katz: *Nerve, muscle and synapse* McCraw-Hill, New York 1966.
- [16] S. Kirkpatrick and D. Sherrington: *Infinite ranged models of spin-glasses* *Phys.Rev. B* **17** (1978), 4384.
- [17] W. Krauth and M. Mezard: *J. Phys A: Math. Gen.* **20** (1987), L745.
- [18] W.S. McCulloch and W.A. Pitts: *A logical calculus of the ideas immanent in neural nets* *Bull.Math.Biophysics* **5** (1943), 115.
- [19] M. Mézard and J.P. Nadal: *Learning in Feedforward Neural Networks: The Tiling Algorithm* *J.Phys.A* **22** (1989), 2194–2204.
- [20] M. Minsky and S. Papert: *Perceptrons* Cambridge, Mass. 1969.
- [21] G.J. Mitchinson and R.M. Durbin: *Bounds on the Learning Capacity of Some Multi-Layer Networks* , *Biological Cybernetics* **60** (1989), 345.
- [22] D. Nabutovsky and E. Domany: *Learning the unlearnable* *Neural Computation* **3** (1991), 604.
- [23] M. Opper et al.: *On the ability of the optimal perceptron to generalise* , *J. Phys A* **23** (1990), L581–L586.
- [24] M. Opper and D. Haussler: *Phys. Rev. Let.* **66** (1991), 2677.
- [25] R. Penrose: *Proc. Cambridge Phil. Soc.* **51** (1955), 406 – 413.
- [26] Plaut, D. C., Nowlen, S. J., Hinton, G. E.: *Experiments on learning by Backpropagation* Technical Report CMU-CS-86-126, Carnegie Mellon University, Science Dept., Pittsburgh 1986.
- [27] R. Rohwer: *The moving targets training algorithm* in: L. B. Almeida and C. J. Wellehens (eds.): *Lecture Notes in Computer Science 412 “Neural Networks”* EURASIP workshop, 15.2. – 17.2.1990
- [28] F. Rosenblatt: *Principles of Neurodynamics – Perceptrons and the theory of brain* Spartan Books, Washington D. C., 1961.
- [29] P. Rujan: *A fast method for calculating the perceptron with maximal stability* , *J. Physique I* **3** (1993), 277–290.
- [30] D. E. Rumelhart, G. E. Hinton: *Parallel Distributed Processing* MIT Press, Cambridge, Mass. 1986.
- [31] V. Vapnik: *The nature of statistical learning theory* Springer, 1995.
- [32] T. Watkin, A. Rau and M. Biehl: *The statistical physics of generalization* , *Rev. Mod. Phys.* **65** (1993), 499.
- [33] A. Wendemuth, M. Opper and W. Kinzel: *The effect of correlations in neural networks* , *J. Phys. A* **26** (1993), 3165.
- [34] A. Wendemuth and D. Sherrington: *Fast Learning of Biased Patterns in Neural Networks* , *Int. Jou. of Neural Systems* **4** (1993), .
- [35] A. Wendemuth: *Learning the Unlearnable* , *J. Phys. A* **28** (1995), 5423.
- [36] A. Wendemuth: *Performance of Robust Training Algorithms for Neural Networks* , *J. Phys. A* **28** (1995), 5485.
- [37] A. Wendemuth: *Storage capacity bounds in multilayer neural networks* , *Int. Jou. of Neural Systems* **5** (1994), 217-228.
- [38] B. Widrow and M.E. Hoff: *Adaptive Switching Circuits* , IRE WESCON Convention Report **4** (1960), 4 - 96.