J. Phys. A: Math. Gen. 28 (1995) 5423–5436. Printed in the UK

Learning the unlearnable

Andreas Wendemuth†

Department of Physics, Theoretical Physics, Oxford University, 1 Keble Road, Oxford, OX1 3NP, UK

Received 2 March 1995, in final form 6 June 1995

Abstract. Three neural network training algorithms are presented which are robust to non-learnable problems. The first algorithm converges to the Gardner stability limit if the learning problem is linearly separable, and otherwise finds a locally maximally stable solution.

The second algorithm is a robust version of Rosenblatt's perceptron learning algorithm which will converge to a solution of the learning problem if one exists, and otherwise will converge locally to a solution with a certain fraction of wrongly mapped patterns.

The third algorithm is suited most favourably to unlearnable problems: it will always find a solution if the problem is learnable and otherwise it locally maximizes the number of patterns which are stored correctly. The error rate of this algorithm and other known algorithms for unlearnable problems are compared for two benchmark problems.

Proofs of the existence of solutions are given. Convergence is proven as well to be global in the case of learnable and local in unlearnable cases.

1. Introduction

By now standard training algorithms in perceptrons like Rosenblatt's algorithm [11], or refined versions like Krauth and Mezard's minover algorithm [8], Anlauf and Biehl's AdaTron [2] and the generalization of these algorithms to problems with adjustable threshold [14] only converge if the problem is learnable, i.e. linearly separable. The same holds for improved example selection [7] or Rujan's algebraic methods that use active sets [12]. In order to overcome this problem, various approaches for training unlearnable problems have been used. Early methods use cost functions like least mean square (LMS) [16] or cross-entropy [6], however these cost functions are not designed to maximize the number of correctly stored patterns. Annealing is utilized in the works of Rujan [13] and Frean [4]. Nabutovsky and Domany [10] have presented a method that *detects* unlearnable problems. Gallant [5] stores a faithful set of weights until they are replaced by a better one in further training, this procedure can be shown to maximize *in probability* the number of stored examples but can take a very long time to converge.

It would be desirable to have modified algorithms that manage to exhibit the properties of their predecessors but converge as well, even if the problem is *unlearnable*. In the latter case, a suitable measure for the learning error must be found and minimized. These algorithms should be local and should not include computationally or time expensive schemes like complicated algebraic calculations, annealing procedures or the storage of an intermediate set of weights.

† E-mail address: wendemu@thphys.ox.ac.uk

0305-4470/95/185423+14$19.50

Such algorithms will have the advantage of guaranteed convergence and therefore applicability to any kind of problems without prior or online inspection of the feasibility of a solution.

The algorithms presented here satisfy these demands.

2. The problem

The given problem then is to find an N-dimensional perceptron vector J such that for a given set of N-dimensional patterns $\{\xi_1, \ldots, \xi_p\}$ and corresponding outputs ('targets') $\{\tau_1, \ldots, \tau_p\}$ $(\xi_\mu \in R^N; |\xi_\mu|^2 = N \ \forall \mu \ ; \tau_\mu \in \{\pm 1\})$ the equations

$$\tau_\mu = \operatorname{sign}(J \cdot \xi_\mu) \qquad \mu = 1 \ldots p \tag{1}$$

are either, if a solution exists, satisfied, or if no solution exists, a solution is found which, though not satisfying equation (1), is locally stable. The algorithms presented here are designed in such a way that without inspection of the feasibility of the solution of equation (1), the convergence is either global in the feasible or local in the unfeasible case. Therefore the algorithms can simply be run on the given problem, their result after learning will then show whether the given problem was feasible or not. This feature of the algorithms will be called *robustness*.

3. Making the updates robust

Making the algorithms robust means modifying them such that they will converge even if the training problem is not solvable. The idea is derived from the spirit of Rosenblatt's perceptron algorithm or Krauth and Mezard's minover algorithm [8]. In these algorithms, at every step an unstabilized pattern is added to the perceptron vector J. The algorithms can be interpreted geometrically: at every iteration step, the perceptron vector J is *turned into* the direction of the unstabilized pattern. By feasibility of the solution, this simultaneously *increases* the norm of J. The effect of this increase of the norm is a decrease of the turning angle and therefore a self-adjustment of the stepsize in the geometric positioning of J, or in other words, a dynamic decrease of the gain parameter. In the minimal-overlap algorithm of Krauth and Mezard, this leads to guaranteed convergence to the maximally stable solution after an infinite number of time steps. In Rosenblatt's perceptron algorithm, after a finite number of steps a solution will be found.

However this convergence can only be proven for feasible problems, in the case of unfeasibility these algorithms fail completely. In the following, algorithms will therefore be modified in a certain fashion such that the two features of turning J and increasing the norm are maintained even in the unfeasible case. It will be shown that this suffices to establish convergence of the algorithms.

The main idea is as follows. Let $\sigma_\mu := \tau_\mu \xi_\mu$. Instead of adding the pattern σ^*/N used for updating, one adds to the perceptron vector J a vector r/N with $r = aJ + b\sigma^*$ $(a, b \geqslant 0)$ but such that *the overlap $J \cdot r$ is non-negative* and such that r has a bounded norm. The norm of J will then always be increased by this procedure, as demanded.

These conditions are met by

$$r = \sigma^* + x(\sigma^*; J)\frac{\sqrt{N}}{|J|}J \tag{2}$$

with the following conditions: $0 \leqslant x(\sigma^*; J) \leqslant x^*$, $J \cdot r \geqslant 0$, and for large $|J|$, $J \cdot r < |\Delta||J|$ where Δ is the maximum stability introduced in the following section. It is

noted that this condition suffices for the formal convergence proof for learnable problems which is presented later. However, Δ is normally not known *a priori* and in this case the stronger condition $\forall \Delta_\mu \leqslant 0 : (J \cdot r_\mu/|J|) \to 0$ may be imposed. Further, x must be such that for $|\sigma^* \cdot J| \neq |\sigma^*||J|$, $J \cdot r > 0$ since otherwise the algorithm may be trapped into a halt.

A suitable choice for x will be given. Obviously, the bounds on x and $J \cdot r = J \cdot \sigma^* + x\sqrt{N}|J| = |J|\sqrt{N}(\cos(J, \sigma^*) + x) \geqslant 0$ immediately require

$$x(\sigma^*; J) \geqslant \max\{0, -\cos(J, \sigma^*)\}. \tag{3}$$

Further, $|r|$ is bounded by $|r| \leqslant R = \sqrt{N(1 + x^{*2})}$ which for $x^* = \mathcal{O}(1)$ has the same scaling as the conventional update σ^*.

Under these conditions, one can first show that the desired algorithmic features are actually obtained. Let us suppose that learning is started with J^0 where γ_0 is the angle $(J_0; \sigma)$. J^0 will be updated by vectors r which derive from σ. If $\gamma_0 = \pi$, r is not defined which can always be restored by an arbitrarily small distortion of J_0. It is desired that after M learning steps, the angle $(J; \sigma) \leqslant \gamma^* > 0$ where γ^* is any predefined angle and σ can be a pattern or a combination of patterns. The required total turning angle will always be less than π. We are going to show that these conditions can always be matched after a finite number of M steps.

First, observe that after t steps have been taken, $|J|$ has not decreased since always $J \cdot r \geqslant 0$. It has grown at most as $|J| \leqslant |J^0| + t|r|_{\max}$. Suppose the angle γ^* cannot be reached. Then we have $0 < D_{\min} \leqslant (J \cdot r)/|J| \leqslant D_{\max} < |r|$. These bounds establish that there exists a vector s_t which is the component of r_t perpendicular to J_t, pointing into the direction of σ. Further, the length of s_t will be bounded from below by $s_{\min} > 0$ for all t. Then it follows also that $|J^t|$ will grow infinitely at least as $|J^t|^2 \geqslant |J^0|^2 + ts_{\min}^2$.

The angle γ_t by which J_t is turned at step t into the direction of σ is then $\gamma_t \geqslant \arctan\big(s_{\min}/(|J^0| + t|r|_{\max})\big)$. After M steps, we obtain

$$\begin{aligned}\gamma_M = \sum_{t=1}^{M} \gamma_t &\geqslant \sum_{t=1}^{M} \arctan\big(s_{\min}/(|J^0| + t|r|_{\max})\big) \\ &\geqslant \int_1^M dt \arctan\big(s_{\min}/(|J^0| + t|r|_{\max})\big) \overset{M \text{ large}}{\to} \propto \ln(M) \overset{M\to\infty}{\to} \infty .\end{aligned}$$

This shows that γ_M is unbounded, therefore J will be turned by any required angle. This contradicts the assumption that the total turning angle $\gamma_0 - \gamma^*$ cannot be reached and establishes that M is finite which completes the proof.

Additionally, since $|r|$ is bounded, we also have $|s_t| \leqslant |r|_{\max}$. Then $\gamma_t \leqslant \arctan(|r|_{\max}/|J_t|)$. With increasing $|J_t|$, γ_t is then going to be decreasing in the course of learning. This allows for arbitrarily small update angles if this should become necessary in the course of the algorithm. We shall make use of this property later.

In summary, we have shown so far that the presented updates will get J arbitrarily close to any desired direction. We shall now indicate directions which are desirable and define algorithms which converge to these directions. Before this is done, a choice of suitable x for practical problems will be given.

4. A suitable choice for x

A basic guess for x is a constant, $x_0 = 1$, which satisfies equation (3) but does *not* satisfy for large $|J|$, $J \cdot r < |\Delta||J|$ for any Δ. Therefore a much finer tuned choice for x is

required which is favourably matched by the following:

$$|J^{(0)}| > \sqrt{N}$$
$$x^{(t)} = \frac{N - J^{(t)} \cdot \sigma^{(t)}}{\left(J^{(t)}\right)^2 - J^{(t)} \cdot \sigma^{(t)}} \frac{|J^{(t)}|}{\sqrt{N}} \Theta\left(-J^{(t)} \cdot \sigma^{(t)}\right) \tag{4}$$

i.e.

$$r^{(t)} = \sigma^{(t)} + \frac{N - J^{(t)} \cdot \sigma^{(t)}}{\left(J^{(t)}\right)^2 - J^{(t)} \cdot \sigma^{(t)}} J^{(t)} \Theta\left(-J^{(t)} \cdot \sigma^{(t)}\right).$$

Note that this does not impose further computational effort: in any algorithm, the quantity $J^{(t)} \cdot \sigma^{(t)}$ would have to be computed anyway to find out whether the pattern must be used for updating or not, and J_t^2 must be computed in both formulations (2) and (4). The latter is in fact the only quantity that has to be computed additionally if compared to the non-robust algorithms, this is just one dot product in addition to the $\mathcal{O}(p)$ ones needed to locate a yet unstabilized pattern.

Let us give further results using the following notation:

$$\frac{J \cdot \sigma_\mu}{|J|} = \sqrt{N} \cos\left(J; \sigma_\mu\right) = \Delta_\mu \tag{5}$$
$$\frac{J \cdot r_\mu}{|J|} = |r_\mu| \cos\left(J; r_\mu\right) = D_\mu \tag{6}$$
$$|J| = aN. \tag{7}$$

We have $|\Delta_\mu| \leqslant \sqrt{N}$ and $0 \leqslant D_\mu \leqslant \sqrt{N\left(1 + x^{*2}\right)}$. It is convenient to start updates with $|J^0| = \mathcal{O}(N)$ since then the quantities a, Δ_μ and D_μ will usually be of order $\mathcal{O}(1)$. The features required earlier can now be verified. We obtain for $\Delta_\mu < 0$:

$$0 \leqslant x = \frac{1}{\sqrt{N}} \frac{1 - \Delta_\mu a}{a - \Delta_\mu / N} \simeq \mathcal{O}\left(1/\sqrt{N}\right) \leqslant x^* = 1$$
$$0 \leqslant D_\mu = \frac{1 - \Delta_\mu^2 / N}{a - \Delta_\mu / N} \leqslant\simeq \frac{1}{a}\left(1 + \frac{1}{4a^2 N}\right) \tag{8}$$
$$0 \leqslant |r|^2 = N - \Delta_\mu^2 + D_\mu^2 \leqslant 2N\left(1 + \frac{1}{a\sqrt{N} - 1}\right)^2$$

where $|r| = D_\mu = 0$ only for $|\sigma^*| |J| = -\sigma^* \cdot J$. For $\Delta_\mu \geqslant 0$ one obtains $x = 0$, $D_\mu = \Delta_\mu$, $|r|^2 = N$. This together with equations (8) meet the requirements set earlier in equation (2). The quantities D_μ and $|r|$ are of the same order as their counterparts Δ_μ and $|\sigma|$.

Let us now turn to the definitions and convergence proofs of algorithms.

5. Maximum stability

One possible objective for a convergence point of an algorithm is a direction of maximum stability, the definition of which can be given by equation (1).

If the problem is learnable, several algorithms exist which solve it [2, 8, 12, 14]. In a geometric context, equations (1) ask for a plane through the origin separating the points ξ_μ into two classes with positive and negative output. The distances of the points from the plane are given by

$$\Delta_\mu = |J|^{-1} \left(J \tau_\mu \xi_\mu\right)^* \qquad (\mu = 1 \ldots p). \tag{9}$$

The sign of Δ_μ is positive if ξ_μ lies on its class's side of the plane. For a correct classification, $\Delta_\mu > 0(\forall\mu)$, implying that equations (1) are satisfied. If this is the case, there will be a gap between the two classes of outputs. Using the two minimal distances in the respective classes of patterns, the size of this gap is $\min_{\tau_\mu=1}\Delta_\mu + \min_{\tau_\mu=-1}\Delta_\mu$.

Optimal separation is achieved if the gap size between the two classes of patterns is maximized. Note that this definition holds as well for the case of unlearnable problems. Then the gap size is negative, and the optimal solution will still be defined as the solution with maximal gap size.

The minimum distance of any ξ_μ to the plane is the measure of stability:

$$\Delta =: \min_\mu \Delta_\mu = \min_\mu \left(\frac{J \cdot \tau_\mu \xi_\mu}{|J|} \right). \tag{10}$$

Thus the perceptron of optimal stability is (J^*) such that

$$\Delta_{\text{opt}} = \min_\mu \left(\frac{J^* \cdot \tau_\mu \xi_\mu}{|J^*|} \right) = \max_{\{J\}} \min_\mu \left(\frac{J \cdot \tau_\mu \xi_\mu}{|J|} \right). \tag{11}$$

The following *minimal overlap algorithm* aims at maximizing the gap size between the two output clusters, even if the gap is negative. It is immediately noted that in the negative stability regime, the *number* of patterns stabilized under this concept is far from maximal. If a high number of stabilized patterns is desired, the third algorithm presented in this paper will be useful.

The algorithm starts in the space of vectors $\sigma_\mu := \tau_\mu \xi_\mu$ with $J^{(0)} = J^{\text{Hopfield}} = \sum_{\mu=1}^{p} \sigma_\mu$. This is the only and best choice one can make with no prior knowledge about the pattern distribution. First of all, the Hopfield vector is normalized to length N. Then at any iteration step t the algorithm proceeds as follows.

Algorithm 1. Minimal overlap.
Let the quantity $\sigma^{(t)}$ be given by $J^{(t)} \cdot \sigma^{(t)} = \min_\mu \{J^{(t)} \cdot \sigma_\mu\}$.
If $|J^{(t)}| \leqslant c$ (c = some fixed positive number)

then

update $J^{(t+1)} = J^{(t)} + \dfrac{1}{N} r^{(t)}$

else

stop (after $t = M$ steps).

r is taken from equation (2), or (4) as a special choice.

The stability obtained is then $\Delta_c = J^{(M)} \cdot \sigma^{(M)}/|J^{(M)}|$. For $c \to \infty$, algorithm 1 will approach a local maximum of Δ with any required accuracy, in the case of $\Delta > 0$, this maximum is global.

The proof of this algorithm is deferred to the appendix.

In summary, for $\Delta > 0$ it will always converge to the only (global) stability maximum, otherwise it will converge to a local maximum. The advantage is that previous knowledge about positiveness of Δ is not necessary but can be read off after termination of the algorithm.

5.1. Robust perceptron algorithm

Here we will present a robust version of Rosenblatt's perceptron algorithm [11] which can be stated immediately for any given desired stability value κ.

Again, the algorithm starts in the space of vectors $\sigma_\mu := \tau_\mu \xi_\mu$ with $J^{(0)} = J^{\text{Hopfield}} = \sum_{\mu=1}^{p} \sigma_\mu$, normalized to length N. Then at any iteration step t the algorithm proceeds as follows.

Algorithm 2. Robust perception.
If ($|J^{(t)}| \leqslant c$) (c = some fixed positive number) and if ($\Delta^{(t)} = (J^{(t)} \cdot \sigma^{(t)})/|J^{(t)}| < \kappa$) can be satisfied for some pattern σ_μ

then

choose randomly any $\sigma^{(t)}$ with $\Delta^{(t)} < \kappa$ and

$$\text{update } J^{(t+1)} = J^{(t)} + \frac{1}{N} r^{(t)} \tag{12}$$

else

stop (after $t = M$ steps).

r is taken from equation (2), or (4) as a special choice.

As in the previous case, the stop condition can always be met since either the algorithm will terminate for $\Delta_\mu > \kappa$ ($\forall \mu$) and $\Delta \geqslant 0$, or $|J^{(t)}|$ will be infinitely growing, as shown before†.

The convergence proof is in two parts, one for the case of $\Delta > \kappa$ and $\Delta \geqslant 0$, the other one for $\Delta \leqslant \kappa$ or $\Delta < 0$. It is presented in the appendix.

In summary, for the cases where convergence is global, there will be one global minimum which the algorithm converges to. Otherwise, local fixed points exist which the algorithm will converge to. These are however just marginally stable, and macroscopic jumps of J away from them will occur.

5.2. *High number of stabilized patterns*

The following algorithm aims at maximizing the number of stabilized patterns. This is associated with a Gardner–Derrida cost function [9]. A high number of stabilized patterns will for example be useful if one is 'only' interested in learning, or if the patterns are noisy and therefore unreliable. Since this is the most interesting case for unrealizable problems, we will compare the performance of this algorithm and other known algorithms for unlearnable problems in numerical simulations for two benchmark problems introduced by Frean [4].

Finding the maximum number of stabilized patterns in the unfeasible case has been shown [1] to be NP-complete. Therefore we are facing a very hard problem here, the algorithm will however be able to maximize this number locally. For feasible problems, all patterns will be stored but the results will be otherwise suboptimal. It is therefore advisable to start running the first algorithm to find out whether the problem can be solved. If not, one continues with the iteration presented now.

A pattern is regarded as stabilized if $(J^{(t)} \cdot \sigma_\mu)/|J^{(t)}| > \kappa$. For unlearnable problems, maximizing the number of stabilized patterns means that a subset of patterns will be allowed to have very low (negative) stabilities. One does not aim at using these patterns for training. Instead one uses at any iteration step the one incorrectly mapped pattern with stability *closest to* κ. In this fashion, one arrives at the following algorithm.

As before, the algorithm starts in the space of vectors $\sigma_\mu := \tau_\mu \xi_\mu$ with $J^{(0)} = J^{\text{Hopfield}} = \sum_{\mu=1}^{p} \sigma_\mu$, normalized to length N. Then at any iteration step t the algorithm proceeds as follows.

Algorithm 3. Gardner–Derrida.

† In passing we note that in the case of $\Delta > \kappa$ and $\Delta \geqslant 0$, it can be shown that σ might be taken instead of r.

If ($|J^{(t)}| \leqslant c$) (c = some fixed positive number) and if ($\Delta^{(t)} = (J^{(t)} \cdot \sigma^{(t)})/|J^{(t)}| < \kappa$) can be satisfied for some pattern σ_μ

then

let the quantity $\sigma^{(t)}$ be given by
$J^{(t)} \cdot \sigma^{(t)} = \max_\mu \{J^{(t)} \cdot \sigma_\mu | (J^{(t)} \cdot \sigma_\mu)/|J^{(t)}| \leqslant \kappa\}$ and
update $J^{(t+1)} = J^{(t)} + \frac{1}{N} r^{(t)}$

else

stop (after $t = M$ steps).

r is taken from equation (2), or (4) as a special choice.

With increasing c, the algorithm converges to a fixed *direction* $J^*/|J^*|$.

The convergence proof is again deferred to the appendix.

The algorithm will stabilize all patterns for problems with $\Delta > \kappa$ and $\Delta > 0$. Otherwise, it will locally converge to a fixed point which locally maximizes the number of stabilized patterns. The distribution of stabilities will then have a gap below κ and a δ-function of stabilized patterns at κ.

Since a high number of stabilized patterns is usually regarded as the key quality figure if the problem is unlearnable, we will in the following investigate two benchmark problems introduced by Frean [4]. Further results for the full stability distributions of all three algorithms presented here are published elsewhere (see outlook).

The first problem we consider is to maximize the number of correctly assigned binary patterns which are randomly targeted. As in [4], we take all 1024 binary patterns in 10 dimensions and assign to each of them a binary target with equal probability. This is clearly a highly non-separable problem. It is attacked with the algorithm in this chapter, and compared to five of the above-mentioned methods: the least mean square (LMS) algorithm [16], the cross-entropy algorithm [6], Frean's thermal perceptron [4] and Gallant's [5] pocket algorithm as well as his pocket and ratchet algorithm. Data for the performance of these algorithms is taken from [4]. The result one obtains is as shown in table 1

Table 1. The randomly targeted binary patterns problem for 1024 patterns in 10 dimensions. Data other than for this algorithm after [4]. For all algorithms, the data is the mean of 1000 independent trials, and standard deviations are of the order of 0.02.

Algorithm	Proportion of correctly mapped patterns
Pocket algorithm	0.515
LMS	0.542
Cross-entropy	0.542
Pocket and ratchet	0.546
Thermal perceptron (no annealing, best starting temperature)	0.548
Algorithm 3 (Gardner–Derrida)	0.569
Thermal perceptron (α and T annealed, best starting temperature)	0.572

Clearly, algorithm 3 presented in this section is just slightly worse than the best of the other algorithms, the optimal thermal perceptron. However, that one requires a great number of trials to obtain the best starting temperature, as well as a delicately tuned update scheme for the annealing of two parameters. Therefore, with a small loss in accuracy, the algorithm presented in this section is conceptually much simpler and requires much less training time.

As a second problem introduced in [4], we investigate a problem with outliers. Again,

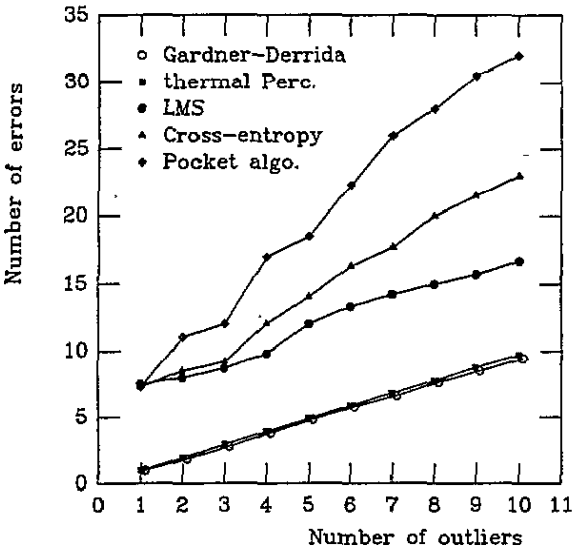

Figure 1. *Training set with outliers.* Number of errors after 100 epochs against number of outliers for various algorithms. 1024 linearly separable binary targets in 10 dimensions have to be stabilized as well. The algorithm presented here achieves less errors than outliers throughout, and outperforms the other algorithms.

all 1024 binary patterns in 10 dimensions are used. They are assigned a binary target given by a randomly generated perceptron such that the output bias is 0. Therefore the problem is clearly linearly separable. Now a number of n outliers is produced by randomly selecting n patterns and reversing their output. A good learning rule should then produce *not more than* n errors: the number of errors can actually be *less than* n for a very good algorithm that manages to stabilize even a fraction of the outliers.

The result of this problem is shown in figure 1. It is again compared to the other above-mentioned algorithms. Data for the performance of these algorithms is taken from [4]. Consistently, algorithm 3, presented in this section (Gardner–Derrida), outperforms all the other methods on much less computational and conceptual effort. It also actually manages to produce less than n errors.

In summary, we have presented an algorithm that achieves a high number of stabilized patterns. Comparisons to other much more complex algorithms on benchmark problems have verified that the new method is of equal or superior performance.

6. Outlook

Three training algorithms have been presented which are robust to infeasible problems. No prior inspection of feasibility is necessary. The algorithms converge to (local) maximum stability, to a robust Rosenblatt solution and to a solution with a locally maximized number of stabilized patterns.

It is desirable to quantify the stability distributions reached by these three algorithms especially in the case of infeasible problems. In this context, in a recent paper [9], calculations have been performed which for random unbiased distributions of a large number of patterns in many dimensions give results for the stability distributions, using cost functions that correspond to the performance of the three algorithms presented here. In a forthcoming paper [15], results of large-scale simulations with these three algorithms will be presented and compared to the results in [9].

Acknowledgments

I would like to to thank O Winther, M Biehl and W Whyte for inspiring discussions. Also, I would like to acknowledge support by the Friedrich–Naumann–Stiftung and the European Community under contract no ERB4001GT922302.

It is a pleasure to acknowledge support by CONNECT/The Niels Bohr Institute, Copenhagen, where part of this work was performed.

Appendix A. Maximum stability

We prove here the convergence of the maximum-stability algorithm. This will be done largely using results from optimization theory, the proofs of which can be found, for example, in [3].

The optimization problem is stated in equation (11) but can equivalently be stated as

$$\min_{\{J\}}(\gamma) \qquad \text{with angle } (J;\sigma_\mu) \leqslant \gamma(\forall\mu) \tag{A1}$$

or as

$$\min_{\{J\}}(-\Delta) \qquad \text{with } J\cdot\sigma_\mu \geqslant \Delta(\forall\mu);\ J^2 = 1\,. \tag{A2}$$

The latter is a linear optimization problem in N dimensions with one quadratic equality and p linear inequality constraints. For $p \leqslant N$, there will always be a solution. The quadratic constraint establishes a nonlinearity from which it already follows that the number of *active* constraints, i.e. those obeyed with equality, cannot be larger than $\min(p, N)$. To see the structure of the solutions more clearly, equation (A2) can be rewritten into the following quadratic optimization problems in the *standard form* of optimization theory:

$$\text{for } \Delta > 0 : \min_{\{J\}}\left(J^2\right) \text{ with } J\cdot\sigma_\mu \geqslant 1\ (\forall\mu) \tag{A3}$$

$$\text{for } \Delta \leqslant 0 : \min_{\{J\}}\left(-J^2\right) \text{ with } J\cdot\sigma_\mu \geqslant -1\ (\forall\mu)\,. \tag{A4}$$

For the last two equations, the following theorems from optimization theory can be stated. The optimal vector J^* is going to be obtained as a pseudoinverse solution (see e.g. [3, 14]) from an active set with a elements, $2 \leqslant a \leqslant \min(p, N)$. The other constraints are satisfied not with equality. The optimal vector J^* has equal overlap to all patterns in the active set. Regarding the two parts of the solution, in equation (A3) the cost function is strictly convex which guarantees that a solution exists which is unique and which is a global minimum. In equation (A4), the cost function is strictly concave but the area defined by the inequality constraints is closed. Then more than one solution exists which minimizes the cost function locally. The global minimum can only be found upon inspection of all the local minima.

In general, we therefore have to identify the active set which generates the solution. Well known *active-set search techniques* (cf [3]) can be applied for that purpose. We can start anywhere in J-space since equation (A2) will always be obeyed with $\Delta = -\sqrt{N}$. We will find a local minimum which in the case of $\Delta > 0$ will automatically be the global minimum. This is advantageous since we do not know beforehand whether $\Delta > 0$ or not, thus an algorithm designed according to the active-set search techniques will not require us to have this knowledge.

In the following, it will be shown that the routes in J-space taken by the algorithm and by the active-set search technique are identical. As an illustration, refer to figure A1.

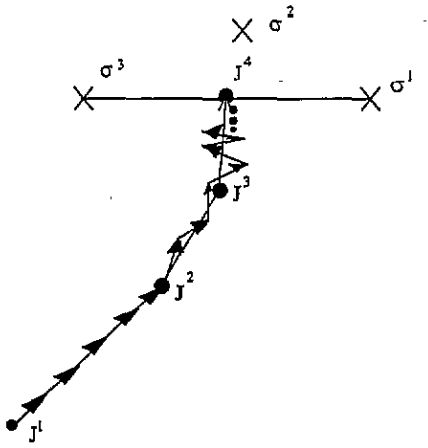

Figure A1. Paths of active set method ($J^1 - J^4$), and minover algorithm (arrows). Minover steps are drawn schematically. At J^1, the active set is $A = \{1\}$, at J^2, it is $A = \{1, 2\}$, at J^3, it is $A = \{1, 2, 3\}$. J^3 is also the pseudoinverse solution (centre of the circle given by σ^1, σ^2, σ^3.) J^3 is unstable, therefore pattern 2 is removed from the active set, and at J^4, *it is* $A = \{1, 3\}$. J^4 *is again the pseudoinverse solution (centre* of the line given by σ^1, σ^3). Being stable, J^4 is the optimal solution. Note that minover flips between directions given by the active patterns. However, since the contribution of pattern 2 to J^3 cannot be removed, J^4 is approached on a zig–zagging path.

The active-set search technique [3] is described as follows: at every step, a feasible movement is taken which aims at minimizing the cost function. This can be demonstrated for the original equation (A1), which is also the way our algorithm proceeds. At the start, some pattern σ_ν will have the maximum angle $(J^0; \sigma_\nu)$ and therefore constitute the cost function. Turning J towards this pattern will minimize the cost function further until another pattern ρ becomes active, such that the optimization problem in the following is constituted by patterns ν and ρ. This is dealt with by turning J into the direction of the bisection or pseudoinverse of patterns ν and ρ which again will minimize the cost function further until yet another pattern becomes active. The process is iterated, moving towards the direction of pseudoinverses of the assembled patterns in the active set as long as the process will not terminate in the last pseudoinverse found.

Convergence of the minimal overlap algorithm can now be proven on four observations.

(i) The turning directions that the algorithm takes follow exactly the prescriptions of the active-set search technique. Movements into the prescribed directions are mimicked iteratively by alternately turning towards the patterns of the current active set. Reaching a candidate for a termination point, the algorithm takes small movements away from this point into any of the directions of patterns in the active set and thereby checks stability as prescribed. Should one of the patterns have to be removed from the active set, the algorithm will continue moving towards the patterns remaining in the active set. It then continues searching into a direction which has equal overlap to all of these patterns which again is a mimicry of the direction of the prescribed reduced pseudoinverse.

(ii) As previously shown, any required turning angle can be achieved.

(iii) The mimicry described in (i) will be performed to any required accuracy in the course of the algorithm. This is due to the previously shown result that update angles will become arbitrarily small. Should in early stages of the algorithm the updates be too coarse, this may at worst lead to a new path selecting a different local minimum than previously. However, since the accuracy is constantly increasing, this possibility will be ruled out as well in the course of the algorithm.

(iv) The algorithm will terminate with some accuracy in the vicinity of the minimum J^* which is a pseudoinverse of, at most, $N^* = \min(p, N)$ patterns. The worst deviation from this solution that the algorithm can perform without minimizing the cost function again is $N^*|r|_{\max}$ in the perpendicular direction from the solution. Then the angle $(J^t; J^*) \leqslant \arctan\left(N^*|r|_{\max}/|J^t|\right) \stackrel{t\to\infty}{\to} 0$ since $|J^t|$ is infinitely growing. Therefore the

solution will be approximated with any required accuracy after a finite number of steps.

This completes the proof of the minimal overlap algorithm.

Appendix B. Robust perceptron algorithm

The convergence proof for the robust perceptron algorithm can be started in the case that $\Delta > \kappa$ and $\Delta \geqslant 0$. In this case convergence is global. The case $0 > \Delta > \kappa$ converges globally only in the case where the starting vector happens to be in the basin of attraction of a local minimum J^l with $\Delta(J^l) > \kappa$. Therefore in general, convergence in this case can be local.

The proof is in parts motivated by the method introduced in [11]. We note that by feasibility there exists a vector J^* which solves the problem with maximal stability Δ and observe that

$$\begin{aligned} J^* \cdot J^{(t+1)} &= J^* \cdot J^{(t)} \left(1 + \frac{x^{(t)}}{\sqrt{N}|J^{(t)}|}\right) + \frac{1}{N} J^* \cdot \sigma^{(t)} \\ &\geqslant J^* \cdot J^{(t)} \left(1 + \frac{x^{(t)}}{\sqrt{N}|J^{(t)}|}\right) + \frac{1}{N}|J^*|\Delta . \end{aligned} \tag{B1}$$

This equation can be iterated to give

$$\begin{aligned} |J^{(t)}| &\geqslant \frac{J^* \cdot J^{(t)}}{|J^*|} \\ &= \frac{\Delta}{N} + \left(1 + \frac{x^{(t)}}{\sqrt{N}|J^{(t)}|}\right)\left[\frac{\Delta}{N} + \left(1 + \frac{x^{(t-1)}}{\sqrt{N}|J^{(t-1)}|}\right)\right. \\ &\quad \left. \times \left[\ldots \left[\frac{\Delta}{N} + \left(1 + \frac{x^{(s)}}{\sqrt{N}|J^{(0)}|}\right)\frac{J^* \cdot J^{(0)}}{|J^*|}\right]\ldots\right]\right] \\ &\geqslant t\frac{\Delta}{N} . \end{aligned} \tag{B2}$$

The last inequality is true since $x \geqslant 0$ and we start the algorithm with any vector $J^{(0)} = \sum a_\mu \sigma_\mu$ with non-negative a_μ which gives $J^* \cdot J^{(0)} > 0$.

Furthermore, as long as updates have not finished, $J \cdot \sigma < \kappa|J|$. Since $\kappa < \Delta$, this guarantees† for a large number of time steps $J \cdot r < \Delta|J|$. Together with $|r| \leqslant R$ and equation (B2) we obtain

$$\begin{aligned} J_{t+1}^2 &= J_t^2 + \frac{2}{N} J_t \cdot r + \frac{1}{N^2} r_t^2 \leqslant J_t^2 + \frac{2D_t|J_t|}{N} + \frac{R^2}{N^2} \\ &= J_t^2 \left(1 + \frac{2D_t}{N|J^{(t)}|} + \frac{R^2}{N^2|J^{(t)}|^2}\right) \leqslant J_t^2 \left(1 + \frac{2D_t}{t\Delta} + \frac{R^2}{t^2\Delta^2}\right) . \end{aligned} \tag{B3}$$

We iterate this equation:

$$|J_t| \leqslant |J_1| \prod_{s=1}^{t} \sqrt{1 + \frac{2D_s}{s\Delta} + \frac{R^2}{s^2\Delta^2}} . \tag{B4}$$

Combining equations (B2) and (B4) we obtain

$$\ln\left(\frac{\Delta}{N}\right) + \ln(t) \leqslant \frac{1}{2}\sum_{s=1}^{t} \ln\left(1 + \frac{2D_s}{s\Delta} + \frac{R^2}{s^2\Delta^2}\right) + \ln|J_1| . \tag{B5}$$

† If it were $\kappa > \Delta$, the algorithm would pick up patterns with $0 < \Delta_t < \kappa$ in which case $x_t = 0$ and the limit for a large number of time steps would be $J \cdot r_t = J \cdot \sigma_t < \kappa|J|$, failing the convergence proof method presented here.

We show now that this condition cannot be satisfied, hence learning must stop. To this end, we evaluate the convergence behaviour of the series for large t. Due to Cauchy's criterion, this can be done by replacing the sum by an integral, keeping only the dominant terms:

$$\ln(t) \leqslant \frac{1}{2}\sum_{s=1}^{t} \ln\left(1+\frac{2D_s}{s\Delta}+\frac{R^2}{s^2\Delta^2}\right) \leqslant \frac{\max\{D_s\}}{\Delta}\int_{t_0}^{t} \mathrm{d}s\,\frac{1}{s}+\mathcal{O}\left(\frac{1}{s^2}\right) \to \frac{\max\{D_s\}}{\Delta}\ln(t)\,. \tag{B6}$$

Since $\max\{D_s\}/\Delta < 1$, equation (B6) cannot be observed, which completes the proof for $\Delta > \kappa$ and $\Delta > 0$.

For $\Delta \leqslant \kappa$ or $\Delta < 0$ the algorithm will stop only since $|J^{(t)}| > c$. Let us prove local convergence for sufficiently large c. First of all, a fixed point J_f of the algorithm is derived from equation (12). Since any one of the patterns used for update is picked randomly with the same probability, we obtain (d = constant)

$$dJ_f = J_f + \sum_{\mu}\left[\left(\sigma_\mu + x_\mu\frac{\sqrt{N}}{|J_f|}J_f\right)\middle|\frac{J_f\cdot\sigma_\mu}{|J_f|}\leqslant\kappa\right]$$

$$\Longrightarrow J_f = -\sum_{\mu}\left[\sigma_\mu\middle|\mathrm{angle}(J_f;\sigma_\mu)\geqslant\gamma=\arccos\left(\frac{\kappa}{\sqrt{N}}\right)\right]. \tag{B7}$$

Let us first of all show that such a fixed point exists. The proof is by iterative construction. We will regard the equation

$$J_a = J_a(J) = -\sum_{\mu}\left[\sigma_\mu\middle|\mathrm{angle}(J;\sigma_\mu)\geqslant\gamma\right] \tag{B8}$$

and show that there exists a vector J such that $J_a = dJ$ which satisfies equation (B7).

Begin by picking any J^0 and compute $J_a^0(J^0)$ from equation (B8). If $J_a^0 = dJ^0$ then the fixed point is found. If not, then define a circle about the origin in the $(J_a^0; J^0)$-plane and a forward direction on the circle such that turning J^0 towards J_a^0 is a forward turn, and define the angle $(J_a^0; J^0) = \beta$.

Now perform on this circle a small forward turn $\delta\beta$ of J^0. This will also turn the boundaries defined at an angle γ from J^0. If during that turn the boundaries do not pass through a pattern, J_a^0 remains at its position which decreases β to $\beta-\delta\beta$. Since the patterns occupy discrete positions, this is always possible other than for discrete locations on the circle.

Further, perform on the circle a 360° forward turn of J, starting at J^0. Any time the boundaries pass through a pattern, J_a will be shifted by $-\sigma_\mu$ if the pattern enters the γ-area, and by σ_μ if the pattern leaves the γ-area. However, after a full circle, every pattern will have entered and left the γ-area exactly once such the total shift induced on J_a^0 by the patterns is zero. Having returned to the starting position, β is recovered as well. The total change of β being zero, we observe that since we initially decreased β in the forward movement there must be other parts in the turn with no patterns passing through the boundaries where β was increased. This, however, is only possible if the projection of J_a on the circle plane was in a *backward* position of J in these parts. Therefore there must be a position J_1 on the circle and a corresponding J_a^1 where the projection of J_a^1 on the circle plane is passing from the forward to backward position of J_1. If J_a^1 at this position has no component perpendicular to the circle plane, the desired fixed point is found. Of course this is always true if $N = 2$.

Note that this may happen while a pattern passes through a boundary as well. In this case, we must allow for this pattern to be with one fraction on one side of the boundary

and with the remaining fraction on the other. In this fashion, we will be able to move 'continuously' through a pattern.

Let us now look at three dimensions and define a new path on a three-dimensional shell. We start at J_1 and move an infinitesimal step in a direction such that the angle $(J_1; J_a^1)$ is reduced. This tilts the initial $(J_a^0; J^0)$-plane slightly. On this new plane, perform the same procedure as described above. Since the move is continuous in the above sense, the position of J_1 will also move continuously. This defines an initial gradient for the position of J_1. Follow that gradient, keeping perpendicular to it the grand circles which define the positions of J_1. In this fashion, a closed path J_2 on the three-dimensional shell is defined where at every position the projection of J_a^2 on the circle plane equals J_2.

Consider the start of the J_2-movement as moving forward. By the very same arguments as above, there must then be a point on this path where the projection of J_2^a on the *shell* changes from forward to backward—it is on the shell since by construction on the path J_2 as well as on the underlying grand circles, the projection changes from forward to backward. This procedure can be used again as a subroutine for dimension four, etc. If the full dimensionality of the problem is reached, a fixed point is found.

After these considerations, the fixed point is accurately given as

$$J_f = -\sum_{\mu}\{\sigma_\mu | \text{ angle } (J_f; \sigma_\mu) > \gamma\} - \sum_{\mu}\{\lambda_\mu \sigma_\mu | \text{ angle } (J_f; \sigma_\mu) = \gamma; 0 < \lambda_\mu < 1\}.$$

Let us now look at the stability of such fixed points.

(i) If at the fixed point no pattern is situated on the boundary it follows immediately from equations (12) and (B7) that after a small distortion, the algorithm will move away from the fixed point and is therefore unstable.

(ii) If a pattern is situated on the boundary, we can consider a distortion without change of the position of J which adds or subtracts a fraction of the weight of this pattern in the γ-area. Both will induce a macroscopic move of J_a, inducing J to move back to the original position. Since distortions of type (i) perpendicular to this boundary pattern will not be stabilized, the γ-area has to be bounded by N boundary patterns, in such a fashion that all directions are stabilized, to guarantee stability.

It is important to notice that in the actual algorithm, patterns can obviously not be split. Therefore J_a will always perform macroscopic jumps away from boundary patterns. This may move J away from the locally stable fixed point altogether. Further, stability relies on a very large $|J|$ to ensure quasi-continuity of the movements taken by J and a quasi-homogenous frequency of non-stabilized patterns used for updating.

Appendix C. High number of stabilized patterns

The convergence proof in the case of feasible problems is identical to the robust perceptron proof and therefore need not be considered.

If the problem is infeasible, we have $\Delta < \kappa$. We must again inspect the existence and stability of a fixed point. This can be given by construction. Since $|J|$ is infinitely growing, we can regard arbitrarily small updates.

Updates are performed with pattern $\alpha | J^{(t)} \cdot \sigma_\alpha = \max_\mu \{J^{(t)} \cdot \sigma_\mu | (J^{(t)} \cdot \sigma_\mu)/|J^{(t)}|\} \leqslant \kappa$. The perceptron vector J will be turned towards σ_α. As long as no other patterns cross the stability boundary $\Delta_\mu = \kappa$, updates will continue with this same pattern σ_α until it is stabilized. The process continues then with the next pattern. In this fashion, one unstabilized pattern after another will be stabilized, which locally *maximizes* the number of stabilized patterns, as required.

However, this process cannot continue forever since the problem is infeasible. In the course of turning J towards the respective patterns, another pattern β which was formerly stabilized will eventually cross the stability boundary in the opposite direction and become unstabilized. As soon as this happens, this pattern now satisfies the condition $J^{(t)} \cdot \sigma_\beta = \max_\mu \{J^{(t)} \cdot \sigma_\mu | (J^{(t)} \cdot \sigma_\mu)/|J^{(t)}| \leqslant \kappa\}$ and is immediately stabilized again. Taking discrete steps, the algorithm will therefore alternate between patterns α and β. This amounts to a *turning* of the hypercone $J \cdot \sigma = \kappa$ around the direction σ_β. In other words, the movement of the hypercone is fixed in one direction.

Continuing with the updates, the movements hypercone will be fixed in an increasing number of directions. This process converges to a fixed point where the hypercone is completely fixed, i.e. any small deviation from this fixed point leads to one pattern becoming unstabilized again. Therefore this fixed point is a pseudoinverse with $J \cdot \sigma = \kappa$ which is locally stable.

For this process, it is unimportant where the patterns σ_μ with $J \cdot \sigma_\mu < J \cdot \sigma_\alpha$ are located. The ignorance of these patterns and the stabilization of a limited set of patterns, which at termination at the boundary has increased stability, leads to a *gap* between the sets of stabilized and unstabilized patterns.

This completes the convergence proof of the algorithm for a high number of stabilized patterns.

References

[1] Amaldi E 1991 On the complexity of training perceptrons *Artificial Neural Networks* ed T Kohonen, K Mäkisara, O Simula and J Kangas (Amsterdam: North-Holland)

[2] Anlauf J K and Biehl M 1989 The AdaTron algorithm *Europhys. Lett.* **10** 687

[3] Fletcher R 1987 *Practical Methods of Optimization* (New York: Wiley)

[4] Frean M A 1992 Thermal perceptron learning rule *Neural Comput.* **4** 946

[5] Gallant S I 1990 Perceptron-based learning algorithms *IEEE Trans. Neural Networks* **NN-1** 179

[6] Hinton G E *1989* Connectionist learning procedures *Artif. Intell.* **40** *185*

[7] Kinzel W and Rujan P 1990 Improving a network generalization ability by selecting examples *Europhys. Lett.* **13** 473

[8] Krauth W and Mezard M 1987 *J. Phys. A: Math. Gen.* **20** L745

[9] Majer P, Engel A and Zippelius A 1993 Perceptrons beyond saturation *J. Phys. A: Math. Gen.* **26** 7405

[10] Nabutowsky D and Domany E *1991* Learning the unlearnable *Neural Comput.* **3** 604

[11] Rosenblatt F 1961 *Principles of Neurodynamics—Perceptrons and the Theory of Brain* (Washington, DC: Spartan Books)

[12] Rujan P 1993 A *fast method for calculating the perceptron with maximal stability J. Physique I* **3** 277

[13] Rujan P 1988 Searching for optimal configurations by simulated tunneling *Z. Phys.* B **73** 391

[14] Wendemuth A 1994 Training of optimal cluster separation networks *J. Phys. A: Math. Gen.* **27** L387

[15] Wendemuth A 1995 *Performance of Robust Training Algorithms for Neural Networks* (Oxford: Oxford University Press) *Preprint*

[16] Widrow B and Hoff M E 1960 *Adaptive Switching Circuits (IRE WESCON)* Convention Report, vol 4, pp 4–96