Complex Systems **8** (1994) 41–65

Hyperplane Dynamics as a Means to Understanding Back-Propagation Learning and Network Plasticity

Frank J. Śmieja*
German National Research Centre for Computer Science (GMD), Schloß Birlinghoven, 52357 St. Augustin, Germany

Abstract. The processing performed by a feed-forward neural network is often interpreted through use of decision hyperplanes at each layer. The *adaptation* process, however, is normally explained using the picture of gradient descent of an error landscape. In this paper the *dynamics* of the decision hyperplanes is used as the model of the adaptation process. An electro-mechanical analogy is drawn where the dynamics of hyperplanes is determined by interaction forces between hyperplanes and the particles that represent the patterns. Relaxation of the system is determined by increasing hyperplane inertia (mass). This picture is used to clarify the dynamics of learning, and goes some way toward explaining learning deadlocks and escaping from certain local minima. Furthermore, network plasticity is introduced as a dynamic property of the system, and reduction of plasticity as a necessary consequence of information storage. Hyperplane inertia is used to explain and avoid destructive relearning in trained networks.

1. Introduction

The back-propagation algorithm is normally explained and analyzed as a gradient descent of an error landscape in *weight* space. However, for feed-forward nets there is a degree of *structure* present in the organization of the weights, and we may expect it to be possible to reparameterize the process such that it may be represented as an optimization of a different, more meaningful set of variables. Using the constraints that the net is feed-forward and consists of layers of nodes that do not mutually interact, a common reparameterization is in the form of the *decision hyperplanes* that determine the input to each hidden and output unit of the network. This is generally employed only to explain the mappings learned by the network, and not the optimization process that led to them. In this paper we use the actual dynamics of the hyperplanes to shed more insight on the workings of the

*Electronic mail address: smieja@gmd.de

back-propagation algorithm in such networks. The optimization process is viewed as a minimization of the interaction forces that influence the movement of the hyperplanes in their respective spaces. The optimization process is then seen as a shifting and anchoring of the hyperplanes. The interactions are between learning-set patterns and hyperplanes and are directly related to the delta error learning rule of back-propagation. This way of viewing back-propagation allows better insight into the learning and reduction of plasticity of a network.

The paper is organized as follows. In section 2 we consider the single-unit case, building an analogy using ideas from a physical system in section 2.2. The analysis is extended in section 3 to networks having a hidden layer. In section 4 we discuss the picture of back-propagation learning from the viewpoint of the hyperplane reparameterization. The meaning of hyperplane mass is discussed in terms of network plasticity and information in section 5. We are then in a position in section 6 to annotate a learning scenario, and in section 7 to clarify descent into local minima and show how to escape from some of them. In section 8 we demonstrate how destructive relearning may be reduced when extending the capabilities of an optimized net.

2. Single-predicate case

Consider a single-predicate neural network with n input units, no hidden units, and a single output unit that is connected to the input units via the weights $\vec{w} = \{w_i\}$, $1 \leq i \leq n$. The network has a threshold weight w_0 associated with it. The output unit adopts states according to a monotonically increasing nonlinear *sigmoid* function $f(\vec{\xi} \cdot \vec{w} - w_0)$ of the input pattern $\vec{\xi}$, given by $f(\phi) = 1/(1+e^{-\beta\phi})$ where β represents the sharpness of the sigmoid function. $f(\phi)$ has the properties

$$f(\phi) \in [0,1] \tag{1}$$

$$f(\phi) + f(-\phi) = 1 \tag{2}$$

$$f(\phi) \begin{cases} \geq 0.5 & \text{if } \phi \geq 0 \\ < 0.5 & \text{otherwise.} \end{cases} \tag{3}$$

The output decision hyperplane determines the values of ϕ that cause the output to lie on the transition between two logical interpretations. Normally the interpretations are "`yes`" and "`no`," where

$$\texttt{yes} \equiv f(\phi) > 0.5 \tag{4}$$

$$\texttt{no} \equiv f(\phi) < 0.5, \tag{5}$$

the transition between the two thus being $f(\phi) = 0.5$, which is defined by the locus $\phi = 0$, or the points $\vec{r}$ on the hyperplane $\vec{w} \cdot \vec{r} = w_0$. The hyperplane may be written in its minimal (standard) form

$$\vec{n} \cdot \vec{r} = \theta, \tag{6}$$

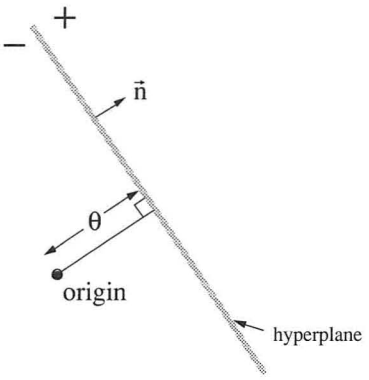


Figure 1: The parameters that define a decision hyperplane in pattern space.

where $\vec{n}$ is the unit vector parallel to $\vec{w}$ that determines the *orientation* of the hyperplane in the pattern space, and θ is the perpendicular distance of the hyperplane from the origin (see Figure 1).

$$\theta := w_0/\gamma \tag{7}$$
$$\vec{n} := \vec{w}/\gamma \tag{8}$$
$$\gamma := |\vec{w}|. \tag{9}$$

The parameter γ is positive and represents the *sharpness* of the logical output decision. It plays a vital role in determining the hyperplane dynamics.

In the back-propagation algorithm the weights $\vec{w}$ to the output unit are changed by the following amount at each time step (assuming *batch* update[1]):

$$\Delta\vec{w} = \eta \sum_p (t^p - o^p)\, f^{p\prime}\, \vec{\xi}^p, \tag{10}$$

where $f^{p\prime}$ is the differential of the output node function, t^p is the target value for pattern $\vec{\xi}^p$ at the output unit, and o^p is the actual network output value for this pattern. The threshold is changed as follows:

$$\Delta w_0 = \eta \sum_p (t^p - o^p)\, f^{p\prime}. \tag{11}$$

The hyperplane dynamics is determined by the changes in the parameters $\vec{n}$ and $\vec{\theta}$ per time step. In the following sections we will derive the dynamics equations resulting first from the back-propagation adaptation changes on the system, and second from considering the system as electro-mechanical in nature.

[1]When *online* (or *per-sample*) learning is performed in small steps it approximates to batch learning [13].

2.1 Back-propagation kinematics

From equations (7) and (8), at each time step the hyperplane parameters are changed according to

$$\Delta\vec{n} = \frac{1}{\gamma}\Delta\vec{w} - \frac{\Delta\gamma}{\gamma^2}\vec{w} \tag{12}$$

$$\Delta\theta = \frac{1}{\gamma}\Delta w_0 - \frac{\Delta\gamma}{\gamma^2}w_0. \tag{13}$$

Substituting from equations (7), (8), (10), and (11):,

$$\Delta\vec{n} = \frac{\eta}{\gamma}\sum_p (t^p - o^p)\, f^{p\prime}\,\vec{\xi}^p - \frac{\Delta\gamma}{\gamma}\vec{n} \tag{14}$$

$$\Delta\theta = \frac{\eta}{\gamma}\sum_p (t^p - o^p)\, f^{p\prime} - \frac{\Delta\gamma}{\gamma}\theta \tag{15}$$

Both the above equations consist of one term (the first) that is concerned with the change due to movement, and another (the second) concerned with change due to a change in the constant γ. We will see the meaning of this later.

Since $\vec{n}$ is orthogonal to $d\vec{n}$, by performing the operation "$\cdot\,\vec{n}$" (scalar product) on both sides of equation (14) we obtain an equation for the change in γ per unit time step:

$$\Delta\gamma = \eta\sum_p (t^p - o^p)\, f^{p\prime}\,\vec{\xi}^p \cdot \vec{n}, \tag{16}$$

which can also be obtained by differentiating equation (9).

From equation (15) the rate of change of θ is given by

$$\frac{d\theta}{dt} = \frac{\eta}{\gamma}\sum_p (t^p - o^p)\, f^{p\prime}(1 - \vec{\xi}^p \cdot \vec{n}\theta). \tag{17}$$

From equation (14) the rate of change of $\vec{n}$ is given by

$$\frac{d\vec{n}}{dt} = \frac{\eta}{\gamma}\sum_p (t^p - o^p)\, f^{p\prime}(\vec{\xi}_p - (\vec{\xi}_p \cdot \vec{n})\vec{n}). \tag{18}$$

In the three-dimensional case, because for small $d\vec{n}, d\vec{\psi} \approx d\vec{n} \wedge \vec{n}$ ($\wedge$ is the vector product),

$$\frac{d\vec{\psi}}{dt} = \frac{\eta}{\gamma}\sum_p (t^p - o^p)\, f^{p\prime}(\vec{\xi}_p \wedge \vec{n}). \tag{19}$$

This special case will be used for intuitive comparison with the mechanical analogue to be described in the next section.

During back-propagation learning the decision hyperplane in our one-node system is moved around: *translationally* as described in equation (17),

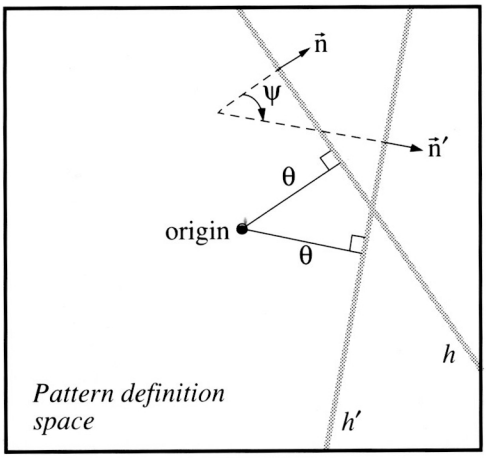


Figure 2: Rotation of a hyperplane from position h to position h' without change in θ. The rotation occurs about the origin, resulting in the angle ψ between $\vec{n}$ and $\vec{n}'$.

and *rotationally* as described in equation (18). It is moved around because "the system" tries to optimize the relative locations and orientations of pattern vectors and hyperplanes. In other words, the algorithm attempts to partition the patterns in such a way that the correct logical interpretations at the output unit are achieved for all of them. The translational and rotational equations express the process of the back-propagation algorithm in terms of the hyperplane reparameterization. There are clearly three important parameters associated with the hyperplane: $\vec{n}$, θ, and γ. The first two can be pictured (see Figure 1), but what the third might represent is not so obvious. From the equations it can be seen that increasing γ reduces both the translational and rotational velocity, all other parameters being held constant. Furthermore, at each time step γ is changed, and this removes part of the translational velocity (equation (15)) and rotational velocity (equation (14)) that would have been transmitted.

For this reason it seems reasonable to imagine γ as being an equivalent to inertia, or mass. But movement of a mass requires a force, and therefore if we are to develop a dynamical hyperplane model we will need to rewrite the movement equations in terms of an interaction between hyperplanes and patterns. To draw an analogy we will first need to derive some standard dynamics equations in an analogous mechanical system.

2.2 Electro-mechanical dynamics

Assume for the sake of picturing the system that all interactions are projected into two dimensions. We consider the hyperplane to be equivalent to a massive plate with uniform mass M and center of mass P. The center of

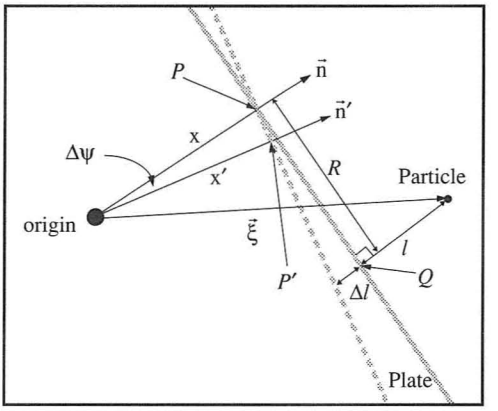


Figure 3: During the interaction time Δt the force between fixed particle p and plate h causes a rotation of the plate about the center of mass P, resulting in a new orientation $\vec{n}'$ of the plate.

mass for this system we define as that point on the plate nearest the origin, a distance x from the origin (see Figure 3).

A useful analogy for an interaction is electrostatics (see, for example, [9]). Let us distinguish the two plate surfaces using the vector $\vec{n}$, which points from the negative to the positive surface. Imagine the plate h to possess an overall uniform *charge* $+q_h$ on its positive surface and $-q_h$ on its negative surface (Figure 1). Consider a *fixed* particle p possessing a charge q_p, situated a perpendicular distance l_p from the plate (see Figure 4). The electrostatic force $\vec{F}$ is directed along the perpendicular and has the magnitude

$$F \propto q_p q_h g(l_p), \tag{20}$$

where $g(l_p)$ is a function representing the range of the interaction.

The rotational interaction results in both a new orientation $\vec{n}'$ and a new pivot distance x' for the plate. The interaction is therefore sufficient to explain the source of both rotational and (one) translational components of plate motion. The translational component is, however, of second-order magnitude and we may safely ignore it, and assume a stationary pivot for the duration of the turning interaction. The main component of translational motion comes from the linear component of the force acting at Q.

The form of the turning interaction is given by the torque $\vec{T}$ caused by the force $\vec{F}$ acting at the point Q about the center of mass P:

$$\vec{T} = \vec{F} \wedge \overrightarrow{QP}\,, \tag{21}$$

where the vector $\overrightarrow{QP}$ is given by

$$\overrightarrow{QP} = (\vec{\xi}_p \cdot \vec{n})\vec{n} - \vec{\xi}_p, \tag{22}$$

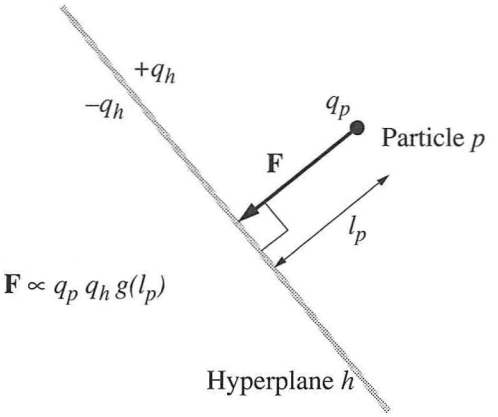


Figure 4: Interaction force between particle p and a hyperplane.

and the force is given by $\vec{F} = -F\,\vec{n}$. Substituting in equation (21) we obtain for the torque

$$\vec{T} = -F\,\vec{n} \wedge ((\vec{\xi_p} \cdot \vec{n})\vec{n} - \vec{\xi_p}) \tag{23}$$

$$= F\,\vec{n} \wedge \vec{\xi_p}. \tag{24}$$

The effect of this instantaneous torque is a change in the rate of angular momentum $\vec{L}$:

$$\frac{d\vec{L}}{dt} = I\vec{\omega} = F\,\vec{n} \wedge \vec{\xi_p}. \tag{25}$$

The moment of inertia I of a uniform plate is proportional to its mass M,

$$I = k\,M.$$

Thus we have for the rate of change of plate orientation $\vec{\psi}$, or rotational velocity $\vec{\omega}$, resulting from a fixed particle with position vector $\vec{\xi_p}$, charge q_p, and perpendicular distance l_p from the plate,

$$\vec{\omega} = G\frac{q_p q_h}{kM}\, g(l_p)\, (\vec{n} \wedge \vec{\xi_p}), \tag{26}$$

where G is the interaction constant.

The translational component of the force $\vec{F}$ is imparted as instantaneous change in linear momentum $\vec{K}$:

$$\frac{d\vec{K}}{dt} = M\vec{v} = \vec{F}. \tag{27}$$

Thus the translational velocity is

$$\vec{v} = \frac{d\vec{x}}{dt} = G\frac{q_p q_h}{M}\, g(l_p)\vec{n}, \tag{28}$$

which determines how much the center of mass of the plate moves in an infinitesimal time step, in the direction of the force $\vec{F}$.

Electrostatic interactions are additive; thus when more than one fixed particle is present, equations (26) and (28) change to (writing now in the form of orientation angle ψ)

$$\frac{d\vec{\psi}}{dt} = G \sum_p \frac{q_p q_h}{kM} g(l_p) (\vec{n} \wedge \vec{\xi_p}) \tag{29}$$

$$\frac{d\vec{x}}{dt} = G \sum_p \frac{q_p q_h}{M} g(l_p) \vec{n}, \tag{30}$$

If the particles are not fixed, they experience a *recoil momentum*, which simply means that the same linear momentum in equation (27) is received by the particle in the opposite direction.

2.3 Results of the analogy

There is an obvious similarity between the first and second set of dynamics equations. Specifically, comparing equations (17) and (19) (in the 3D case) with (30) and (29), we identify the following analogues between the hyperplane system and the mechanical system:

$$\text{mass } M \equiv \gamma \tag{31}$$

$$\text{separation } l \equiv \vec{\xi} \cdot \vec{n} - \theta \tag{32}$$

$$\text{plate charge } q_h \equiv 1 \tag{33}$$

$$\text{center of mass displacement } dx \equiv d\theta \tag{34}$$

$$\text{interaction range } g() \equiv f'() \tag{35}$$

$$\text{particle charge } q_p \equiv (t^p - o^p) \tag{36}$$

$$\text{force constant } G \equiv \eta \tag{37}$$

$$\text{moment of inertia constant } k \text{ set to } 1 \tag{38}$$

The range term $g(l_p)$ originates from the response function f of the nodes. It is illustrated in Figure 5 for the sigmoid case assumed here.

A complication arises in trying to understand the final terms in equations (17) and (18) for the general non-3D case. There is an additional component in the motion of a back-propagation hyperplane arising from the change in mass that occurs at each time step. The instantaneous increase of mass during the interaction (as encapsulated in equation (16)) removes a part of the linear momentum, and also causes a change of $\vec{n}$ in the direction $\vec{n}$ during the rotational interaction. The components of change that result from the presence of mass are the factors that cause the back-propagation hyperplane system to differ from a finitized physical system. The nearer the hyperplane reaches its optimal orientation, the more its mass increases.

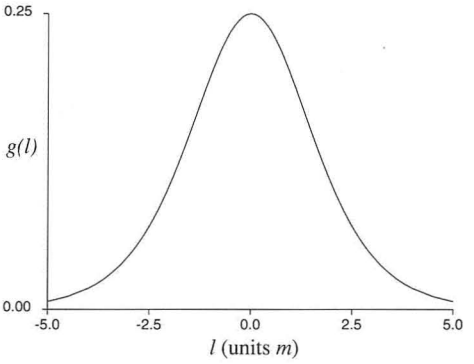


Figure 5: Form of the range function $g(l)$. Particles nearest the boundary have the most effect.

From now on we can write the back-propagation hyperplane equations in a form that uses the electro-mechanical analogy:

$$\frac{d\vec{n}}{dt} = \frac{G}{M}\sum_p g(l_p)\, q_p\, (\vec{\xi_p} - (\vec{\xi_p}\cdot\vec{n})\vec{n}), \tag{39}$$

$$\frac{dM}{dt} = G\sum_p g(l_p)\, q_p\, (\vec{\xi_p}\cdot\vec{n}). \tag{40}$$

$$\frac{d\vec{x}}{dt} = \frac{G}{M}\sum_p g(l_p)\, q_p (1 - (\vec{\xi_p}\cdot\vec{n})\, x)\vec{n}. \tag{41}$$

3. Nets having a hidden layer

We now insert a *hidden layer* of N_H nodes, fully connected to each input node via the weights w_{ij}, and fully connected to the output node via the weights w_j. The index j labels the hidden hyperplanes. Hidden-unit (HU) space [16] is cartesian and defined by the orthonormal set of vectors $\{\vec{e}_j\}$, $j = 1, \ldots, N_H$.

The situation is now more interesting because instead of a single hyperplane in a system that tries to optimize its position, there are now N_H hyperplanes optimizing their positions together in the input pattern space, and one hyperplane optimizing its position in the HU space.

The same hyperplane equations (39)–(41) are used for the hidden hyperplanes and the output hyperplane. The only quantity that changes is the form of the particle charge q_p, which also varies according to the hyperplane being considered.

From the back-propagation equations we have for the weight-change $\Delta\vec{w}_j$ per update for weights leading into hidden hyperplane j, when just one output hyperplane o is considered,

$$\Delta\vec{w}_j = \eta\sum_p f'_{pj}\,(t_p - o_p) f'_{po} w_{jo}\, f_{pj}. \tag{42}$$

This produces, after some manipulation (which it is not necessary to reproduce here), the following for the charge of a particle p in the input space with respect to the hidden hyperplane j:

$$q_p^j = M_o \, g_o(l_p) q_p^o (\vec{\zeta}_p \cdot \vec{n}_o) \, (\vec{n}_o \cdot \vec{e}^j). \tag{43}$$

$\vec{\zeta}^p$ is the position vector of the particle p in hidden space and o labels the output hyperplane. Notice the effect of the output hyperplane: One with a high mass induces a correspondingly large charge in the input particle with respect to the hidden hyperplane j. The induced charge is in the direction away from the output hyperplane and represents the recoil that would be effected in a real physical system. In the network system the particles themselves can only be moved indirectly, and this is in the form of a higher charge on the equivalent particles in the next higher layer that, by shifting hyperplanes in their space, change the location of the original particle (see Figure 6).

4. Remarks

The rewriting of the back-propagation equations in the form of hyperplane–particle interaction allows an original insight into the optimization process. In this section we list some characteristics of the optimization as seen from this viewpoint.

- The particles that have the most effect on the dynamics of the hyperplane are those nearest to it. This observation is also known as the "boundary-pattern" effect and has been empirically studied by other researchers [1, 14]. Furthermore, it is worth noting that, in Figure 5, l_p is measured in units of M. Thus, as the mass of the hyperplane increases, its range of influence decreases. This explains in part the reduction of network plasticity (section 5).

- From the form of the charge in equation (36), incorrectly sided particles will contribute more to the forces on the hyperplane than the correctly sided ones. This is because when a particle has a like charge it has a lower repulsive force than the attractive force of a particle with an unlike charge in the same position. This is how back-propagation places a greater priority on getting particles onto the right side of the hyperplane than on mapping them better once there.

- From equations (39) and (41) it can be seen that the hyperplane stops turning and shifting as $M \to \infty$ and/or $g(l_p) \to 0$ and/or $q_p \to 0 \ \forall p$. The latter option is only possible in this perceptron-like network for linearly separable problems [8], which indicates "problem solved." The second condition is satisfied when the first condition is already being satisfied, because the forces all become shorter range as M increases (Figure 5).

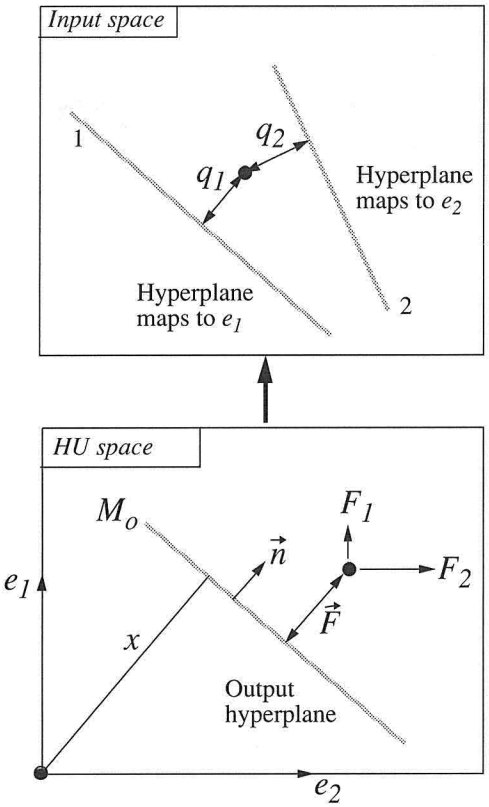


Figure 6: A hyperplane in the HU space (output hyperplane) imparts a recoil to the particle that interacts with it, through translating the components of the force F_1 and F_2 into charges between the input space representation of the particle and the hyperplanes that are responsible for the respective dimensions of the HU space.

- One can predict a "deadlocking" situation for the output hyperplane whereby two particles with opposite charges cause the following for a given hyperplane configuration:

$$g(l_1) = g(l_2) \tag{44}$$

$$q_1(\vec{\xi_1} \wedge \vec{n}) = -q_2(\vec{\xi_2} \wedge \vec{n}) \tag{45}$$

  The result of this is that, from equations (39) and (41), the hyperplane will not be moved to produce the required mappings, and from equation (40), the hyperplane will just grow in mass.

- Deadlocking can be overcome when a hidden layer is present by utilizing the fact that two wrongly mapped particles on opposite sides with

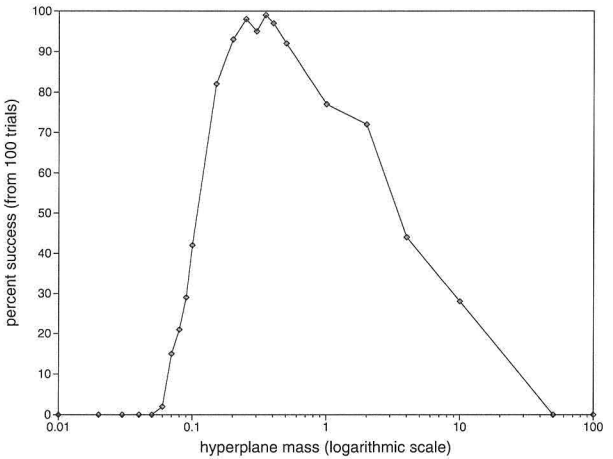


Figure 7: Success rate on the XOR problem as a function of the mass of the hyperplanes used to initialize the network (2 hidden hyperplanes, all hyperplanes have same initial mass).

orthogonal position vectors will pass back opposite charges, and (thanks to the $\vec{\zeta}_p \cdot \vec{n}_o$ term in equation (43)) will encourage their representations in the HU space to be reversed. Deadlock then will not occur when the two particles are on the same side of the *hidden* hyperplane: The particles themselves can be moved instead of the output hyperplane! The sense of the movement in the input space is to separate them by pulling one closer to the hidden hyperplane and the other further away, such as to swap their current representations in the HU space. (Deadlock of this form can still occur, but it is more unlikely because the particles would need to be placed equally on opposite sides of the hidden hyperplane, and this would need to be the case for about half the population of hidden hyperplanes.)

- If the hyperplane is *initialized* to have too large a mass, it may be very difficult for it to be moved to an optimal position and orientation because the force ranges are too small. This is demonstrated in Figure 7 where the success rate of finding a solution for the XOR problem is plotted as a function of the initial mass used for the hyperplanes. If an attempt is made to counteract the drawback of a large mass by increasing the strength of the forces in the system, given by G (in back-propagation parlance "step size"), then equation (40) sees to it that the mass changes *more quickly*, either in a positive direction, or unstably in large steps about its present large value. This explains why forced escape from local minima through temporary increase in step size is not terribly successful. If the hyperplane is initialized to

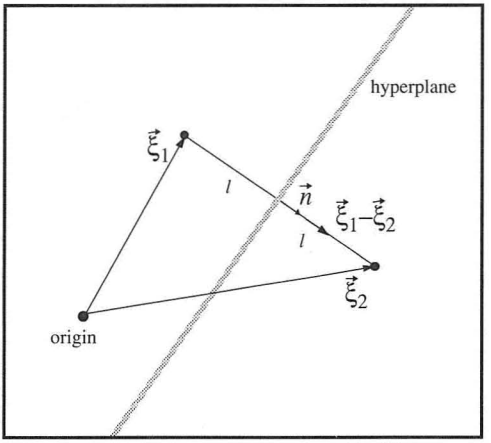


Figure 8: Final optimal state of the hyperplane when two particles with position vectors $\vec{\xi}_1$ and $\vec{\xi}_2$ are mapped to opposite sides. The back-propagation algorithm looks for the orientation that gives it the maximum perpendicular distance from each particle.

have too small a mass, a solution will also be hard to find, because the interaction potential is too flat.

- The change in hyperplane orientation angle is such as to minimize the acute angle between the hyperplane perpendicular and the weighted vector addition of the particle vectors. Consider two particles $\vec{\xi}_1$ and $\vec{\xi}_2$. Assume they have opposite charges, and that they are both on the correct sides of the hyperplane in question (set q_1 to be negative). At dynamical equilibrium they are the same perpendicular distance from the hyperplane (because equation (41) says the translational movement is stable when $g(l_1) = g(l_2)$). Also, it can be seen that at equilibrium the hyperplane adopts the orientation (from equation (39)), which says the hyperplane perpendicular $\vec{n}$ should be parallel to the vector $\vec{\xi}_2 - \vec{\xi}_1$, when the perpendicular distances $\{|l_p|\}$ of the particles from the hyperplane are maximized (see Figure 8). In other words, the hyperplane looks for the position that would be found by a *least-squares approximation* through the points specified by the particle positions. This is precisely the claim made in the perceptron convergence theorem [12], and could be shown here in an intuitive way using the hyperplane interpretation. This important part of the back-propagation iteration goal has been shown in general to have this form [3].

- Once the orientation of the hyperplane has been optimized as described above, one sees that the dot product in equation (40) has reached its maximum value, thus increasing the mass of the (now) stationary hyperplane until the ranges $\{g(l_p)\}$ decay to zero.

- One can view the optimization process as *focusing the sensitivity* of the hyperplane: at the start, all the particles in the unit hypercube (we assume they are all defined here) contribute to the movement of the hyperplane; but as the hyperplane starts to increase in mass it loses sensitivity to distant particles (force becomes more short range) and is controlled to a greater extent by the closer particles. It appears that back-propagation has an *automatic* form of annealing that allows it to relax its sensitivity, or "temperature," from high to low. We equate this in the next section with its adaptation plasticity.

- From equation (43), the heavier the output hyperplane the greater the induced charge on the particles in the input space. This is the *recoil force* on a particle from a hyperplane that is itself not able to move further. This effect is known as the "moving-targets problem" in back-propagation learning [4]. The output hyperplanes try to solve as much as possible through their own positioning, but a few particles are left on the wrong sides because the problem is not linearly separable. When the majority of particles have been correctly sided, the mass of the output hyperplanes will increase anyway, leaving the incorrectly sided patterns no chance to move them. However, a larger output hyperplane mass means a greater magnitude of charge than the input particles would otherwise have, which allows the particles to produce more force on the *hidden* hyperplanes. This makes it easier for the hyperplanes to change their configurations and thus separate the input particles.

- The procedures can be extended layer-wise upwards through the hidden layers of the network, presenting the picture of a learning process where there are fewer particles with high charges the higher up (i.e., the nearer the input layer) one is. This is because more and more particles are mapped correctly as one propagates up through the network. This is the reverse of the way most constructive algorithms [4, 10, 5] go about the task of mapping: They start at the top, mapping more and more patterns as the networks grow in the direction nearer the output [15].

- To extend to the N_O output-unit case, the only change needed is to perform a summation in equation (43) over the output hyperplanes k for the mass and potential terms:

$$q_p^j = \sum_k^{N_O} M_k \, g_k(l_p) q_p^k(\vec{\zeta}_p \cdot \vec{n}_k) \, (\vec{n}_k \cdot \vec{e}^j). \tag{46}$$

5. The significance of hyperplane mass

5.1 Network plasticity

The receptivity of the network to adaptation is known as its *plasticity*. The plasticity is directly related to the degree to which the hyperplanes are freely

moveable. This freedom is hindered by a high hyperplane mass. Equation (43) reveals that the degree of the induced charge is proportional to the mass of the hyperplane j from which the particle is recoiling. When substituted in the dynamics equations (39)–(41) the induced charge serves to provide the tempering factor M_o/M_j. This allows free movement of the hidden hyperplanes despite their high masses, when the output hyperplane has a similar mass.

There is, however, an additional factor in the dynamics equations, and this is the interaction range: Higher masses reduce the range of the interaction (Figure 5), and thus the charge of the particles. Thus the free motion of the hidden hyperplanes is limited to the particles in input space that lie near the hidden hyperplanes, and to the particles in the HU space that lie near the output hyperplane.

This would provoke the notion that an overall high hyperplane mass is an indication of low network plasticity. More exactly, the plasticity has an inverse relation to the lowest mass of the remaining usable hidden hyperplanes. A "usable" hyperplane is one that is positioned so that it can respond to particle charges.

5.2 Information

The increase of a hyperplane mass is very significant for the system. The anchoring of a hyperplane in this way is equivalent to the long-term storage of information. In a similar way to the Shannon information measure, the probability of a particle being on one or the other side of a hyperplane can be taken as its resultant representation in the successive space, which becomes more boolean as the mass of the mapping hyperplane increases. Thus the information content of the system increases as the order in the system increases and as the plasticity decreases.

A lower mass indicates plasticity and is typified by more "fuzzy" output decisions. Long-term memories are only held in the network by anchoring the hyperplanes using high masses. The relative orientations of the hyperplanes provides the pattern correlation information, which, together with the relative hyperplane masses, allows weak or strong generalizations to be made on unseen patterns. Lower hyperplane masses result in more fuzzy mappings and consequently lower Shannon information content of the system. Trained networks are generally very good at gracefully degrading with respect to extra noisy or uncorrelated learning, precisely because of the resistance offered by the heavy hyperplanes that have been evolved.

6. A successful learning scenario

To illustrate the dynamics of hyperplanes we consider a successful learning scenario. The problem used is XOR using a network with two hidden units. This simple problem was chosen for ease of illustration. First we consider how best to initialize the network.

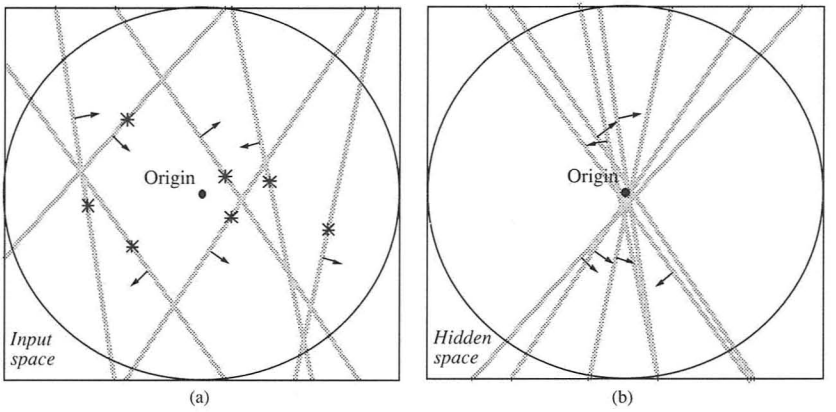


Figure 9: Example correct initial distribution of hyperplanes in (a) input space, (b) HU space.

6.1 Optimal network initialization

An untrained network has an optimal starting point when the hyperplane vectors are chosen randomly from a spherical distribution, and the origin distances are chosen around the center of the pattern distribution. This can best be explained with a picture. Figure 9a shows a set of hyperplanes initialized in the input space. The point at which the perpendicular from the origin meets the hyperplane is shown by a star. These stars should remain within the largest hypersphere that can be drawn completely within the input space, which gives a good range of hyperplanes that slice the distribution of input patterns. The orientations are randomly initialized. Figure 9b shows a set of hyperplanes initialized in the hidden space. The distances from the origin are small, and the orientations chosen randomly.

The importance of initial weight configurations has been often noted (see, for example, [6]). With our representation of the system it is easier to see what kind of a difference they can make. The above suggestion ensures that the hidden hyperplanes have a range of orientations and are positioned anywhere within the largest hypersphere that can be drawn in the space that is contained by the set of possible input patterns. The output hyperplanes, on the other, hand start off somewhere near the middle of the HU space and have a range of orientations. It makes sense to start off near the middle of HU space because the low initial masses of the hidden hyperplanes mean that the particles in the HU space will all have representations around the origin.

The masses of the hidden hyperplanes should be set at initial values such that the influence of each hyperplane, as indicated by the range function shown in Figure 5, covers the whole input space. For a $[0,1]^2$ input space a good value would appear to be about 0.25. The output hyperplane, on the other hand, can start with a higher value because the patterns are initially very close. We set it to 2.

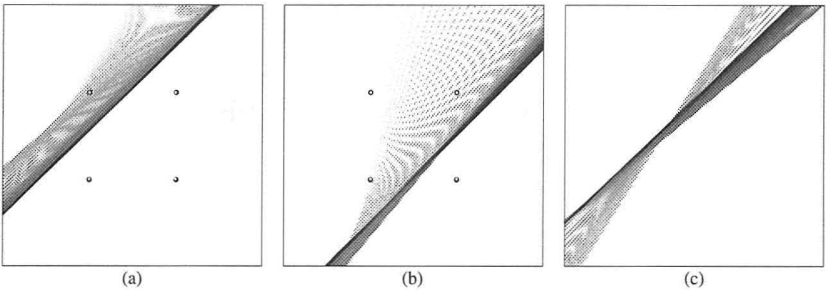


Figure 10: Hyperplane movements during the iteration to a successful solution to the XOR problem. The hyperplanes start with a light shade and become darker with iteration time. (a) hidden hyperplane 1, (b) hidden hyperplane 2, (c) output hyperplane. The four input points are shown in the HU spaces.

6.2 The learning process

Initially the potential is broad, encompassing all the patterns, and the light hidden representations feel little recoil force. Figures 10 and 11 show the pictorial and graphical scenarios for the successful case.

The first stage of optimization is the output hyperplane orienting such as to get as many patterns as possible on the correct side. It begins settling to an optimal position, determined by *all* the patterns (because they all have about the same importance at this stage, and are all about the same distance from the hyperplane) and an unweighted average. It then begins to get more massive due to the forces from correctly sided patterns in range giving repulsion (equation (40)), the variation in ϕ becoming small, and consequently the incorrectly sided patterns experiencing larger recoil forces. These negative and positive (but now smaller) forces from correctly sided patterns then control the dynamics of the hidden-layer hyperplanes. Patterns still on the wrong side of the output hyperplane when it is becoming quite massive will receive stronger forces that demand reorientation of the *hidden* hyperplanes. This is more favorable because there is now less chance of the output hyperplane being reoriented.

The recoil forces on the HU particles cause the hidden hyperplanes to seek optimal positions and orientations, defined by minimizing the sum of forces from all directions against the hyperplane (to end with little turn and shift). This can be seen in Figures 11a, 11c, 11d, and 11f. After this process has been optimized the hidden hyperplanes may begin to grow in mass. It can be seen from Figure 11 how this triggers the simultaneous mass increase for all hyperplanes. As they get more massive, the patterns that are correctly sided (all of them) decay in terms of charge and importance (equation (43)), thus slowing down the hyperplane rotational and lateral motion.

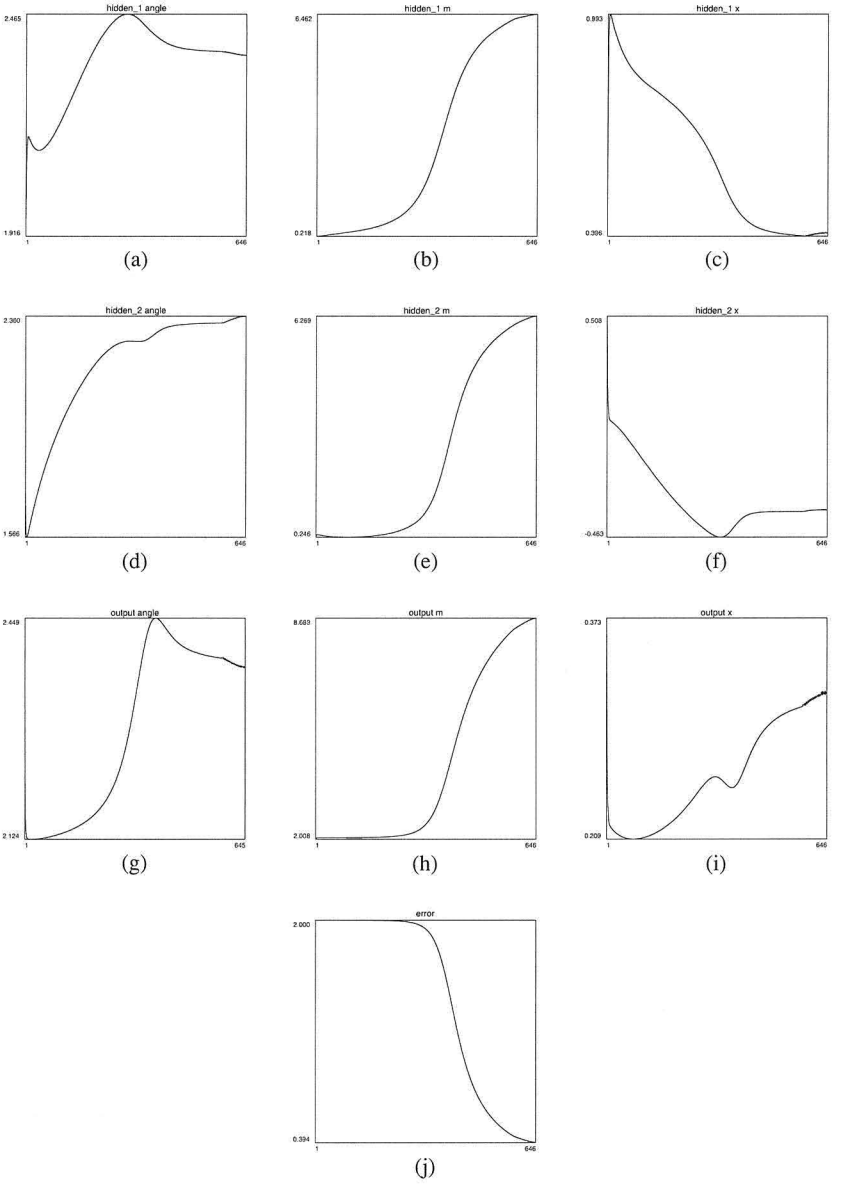
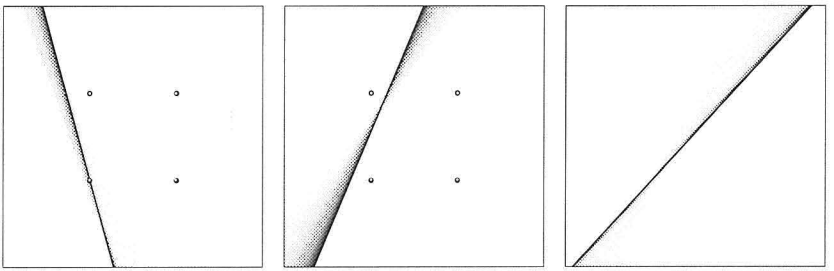
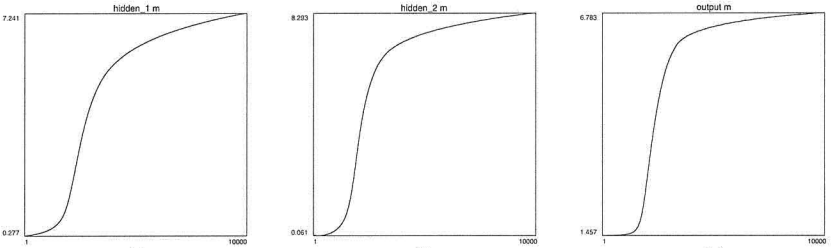


Figure 11: Progress of the three hyperplane parameters during iteration to a successful solution. (a)–(c), hidden hyperplane 1; (d)–(f), hidden hyperplane 2; (g)–(i), output hyperplane; (j), learning error.



(a) (b) (c)

Figure 12: Descent into a local minimum of type 1. The hyperplanes start with a light shade and become darker with iteration time. (a) hidden hyperplane 1; (b) hidden hyperplane 2; (c) output hyperplane. The four input points are shown in the HU spaces. The culprit is hyperplane 1, which hits a deadlock.



(a) (b) (c)

Figure 13: The corresponding hyperplane mass change during the scenario of Figure 12.

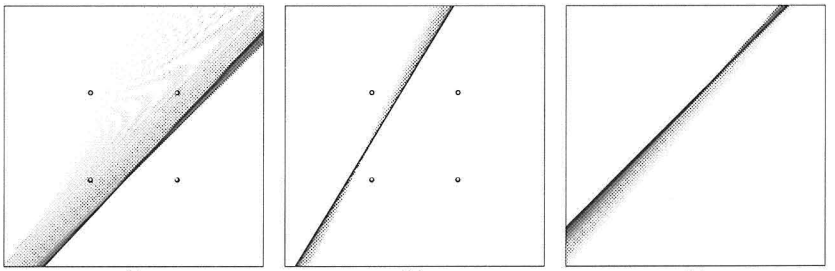
7. Unsuccessful scenarios: local minima and escaping from them

We identify two criteria that lead to the system getting stuck in local minima. They both involve the hyperplanes losing "contact" with incorrectly mapped patterns: (1) premature increase of hyperplane mass causing shorter range interactions, (2) too low hyperplane mass causing disappearance of interaction gradient. Both can be understood from equations (39)–(41).

7.1 Local minimum of type 1

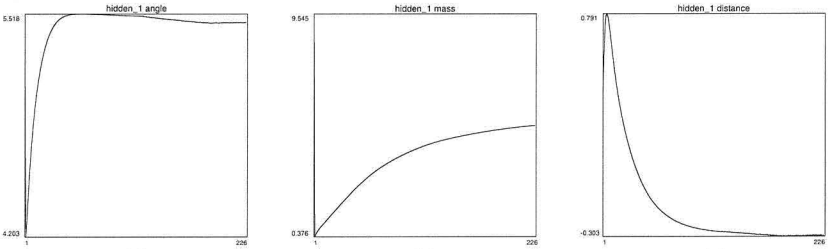
If, for the set of patterns $p \in \mathcal{S}$, symmetry conditions develop such that $d\vec{\psi} \approx 0$ and $d\vec{x}/dt \approx 0$, then the situation is a deadlock. It will often be the case that $dm/dt \neq 0$ in such situations, because it requires solely that the hyperplane lie off the center of the pattern cluster.

An example of such a local minimum is illustrated in Figure 12, and the progress of hyperplane mass in Figure 13. It can be seen that hyperplane 1 (Figure 12a) gets stuck in a deadlock situation where its angle and distance (x) changes are zero, and its mass increases regardless. It is no longer



(a) (b) (c)

Figure 14: Escape from the local minimum of Figure 12.



(a) (b) (c)

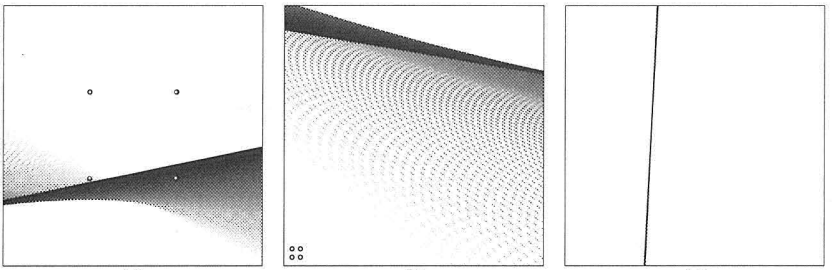
Figure 15: The corresponding parameter change of hyperplane 1 during the escape scenario of Figure 12.

moveable. Hyperplane 2 (Figure 12b), on the other hand, has successfully separated a pattern. At deadlock the mass increase stops.

Such a local minimum can be escaped from simply by reducing the mass of the deadlocked hyperplane. Thus we set the mass of hyperplane 1 in Figure 12a to 0.25 and continued the learning. The ensuing scenario is depicted in Figure 14. It can be seen that hyperplane 1 immediately escapes from the deadlock situation. How? The symmetry leading to deadlock was only present at the beginning, but long enough for hyperplane 1 to over-increase its mass so that it could not come out again. Meanwhile hyperplane 2 had separated some particles, thus changing the symmetry condition. On reduction of its mass hyperplane 1 was able to finish off the particle separation. Figure 15 shows the continued development of hyperplane 1's parameters.

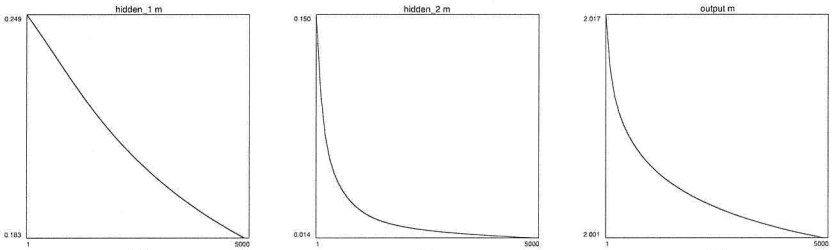
7.2 Local minimum of type 2

Figure 16 shows the pictorial scenario for another type of unsuccessful iteration. This time the hidden hyperplane masses (Figure 17) drop to very low values, and hyperplane 2 flies right out of the pattern space. The reason why hidden hyperplanes fly out of the area (and this is a very common occurrence when many hyperplanes are available) is that the probability of having angle and distance rapidly changed increases at the beginning of the adaptation,



(a) (b) (c)

Figure 16: Descent into a local minimum type 2. The hyperplanes start with a light shade and become darker with iteration time. (a) hidden hyperplane 1, (b) hidden hyperplane 2, (c) output hyperplane. The four input points are shown in the HU spaces. Hidden hyperplane 2 leaves the pattern definition area early on (picture is zoomed out to show this).



(a) (b) (c)

Figure 17: The corresponding hyperplane mass change during the scenario of Figure 16.

and once (by chance) thrown out of the pattern cluster it is possible that an overall repulsion from the patterns will be built up. This happens when the term $(\vec{\xi} \cdot \vec{n})$ is negative (see equation (41)). This pushes the hyperplane further out, which also reduces the mass (equation (40)). As soon as mass becomes low, the lateral motion increases in size (thanks to the $1/M$ factor).

The only way of escaping from such a minimum is to start again. The lost hyperplane indicates either that nothing or everything is correctly mapped (because there is no net bias pulling it). That is why when many hyperplanes are used to solve an easy problem many are thrown out early on. The solution to this obvious drain on time and resources is to add properly initialized hyperplanes successively, as described in [2].

7.3 Difficult problems

Some problems may be of such a type that they may be represented by superposition of a much easier problem and a small (in terms of input patterns)

also easy "perturbing" problem. For example, one oddly mapped pattern may be hidden in a large region of regularly mapped patterns. Then the forces of the larger problem will be acted on first, and those from the perturbing problem may be so small that they will be ignored until mass increase has reduced the repulsive forces from the correctly sided patterns. Then it may be too late. One such example is the "= 1" problem [11]. All patterns with one bit on must be mapped to 1, and the rest to 0. The main problem (in terms of interaction strength) here is the separation of all the $>$ 1-bit patterns from the 1-bit patterns. After this is done successfully, there remains a single pattern $(0, 0, 0, \ldots, 0)$ that has to be mapped. This is the perturbation. What is seen is a rapid learning of all the other patterns (steep drop in learning curve) followed by a long, slightly sloping portion as the nearest free hyperplane is dragged over to perform the final separation. It takes a long time because the hyperplane is already quite massive due to the many other correctly mapped patterns.

In such exceptional circumstances, the minimal solution (defined in [11]) consists of one hyperplane that separates this single pattern from the other patterns, in the input layer. The best way to avoid the slow learning is either to realize before learning that the single pattern is an exception, or to discover through the learning itself that the pattern is an exception, in that it is the only pattern that has not been mapped correctly. Then this pattern should be isolated and learned using an extra hyperplane after all the others have been learned. In the simulations we carried out we were frequently successful by letting the system solve all the patterns apart from the exception (which it does automatically), and then, when it was clear that mass increase had set in, choosing any hidden hyperplane and reducing its mass by about a factor 10. Further learning had the effect of using this suddenly mobile hyperplane to separate the remaining pattern. All the other patterns were reasonably well accounted for by the other hyperplanes, which had merely shifted slightly to accommodate the loss of the suddenly lighter hyperplane. The effect of such controlled "interfering" was startling in such experiments: long plateaux of steady error suddenly dropped to zero in a matter of a few cycles—a very clear demonstration of the effect of hyperplane mass.

What happens if we then wish to perform *additional* learning using the weights of a trained network? Is this possible without destroying previous knowledge?

8. Relearning in trained networks

Consider a new pattern that is to be learned in a trained network consisting of heavy hyperplanes. How can destructive relearning be avoided? There are three cases to consider.

1. The new pattern represents an example of the rule already self-contained in previous learned patterns, and adds no new information other than perhaps a more exact placement of decision regions.

2. The new pattern provides new information that can nevertheless be learned when the hyperplanes already placed are softly rearranged (by this is meant that the entire problem does not have to be relearned).

3. The new pattern can only be learned when a completely new configuration of the hyperplanes is created, or cannot be learned together with the current patterns in the current network.

The third case can only be handled by either complete resetting of the network, or as a special case. Neither option is pleasant, but drastic action cannot be avoided when the information really is new and important.

More promising are the first two cases. To minimize the small destructive effects of the extra learning only manipulation in the hidden layer should be required, and the output hyperplane should be held fixed. Increasing the output hyperplane mass, however, will not damage the other pattern mappings, assuming they were all properly mapped before. We suggest the following method.

The hidden hyperplanes can be shifted singly for the particular pattern. The one that is chosen is that with respect to which the hidden particle in question has the greatest charge (obtainable from equation (43)). The others are held fixed. After every movement the progress of the pattern is monitored. This continues until the charge for this hyperplane is no longer the greatest, at which point the new hyperplane whose charge is the highest is taken and moved singly. This method utilizes the *favored directions* of change in the system in order to reduce the damage to the information currently in the system.

Of course, it may not always be possible to learn a new pattern without damaging the network's past information, and the above merely outlines heuristic ideas of how to ease the effects of a new pattern.

From the hyperplane perspective, a new pattern will generally have a destructive effect on a network when the movements it wishes to enforce on the hyperplanes are large, but unable to be communicated because the ranges are too low and masses too high. The correlation of a great attraction (charge) for a hyperplane, and a low range to enforce it (related to the high hyperplane mass) together with a heavy hyperplane to move, signifies probable destructive behavior. Destructive learning means large alterations in hyperplane structures, which is seen as large movements of heavy hyperplanes. A heavy hyperplane holds more information than a light one, and its movement thus more easily destroys information.

9. Conclusion

In this paper it has been demonstrated how multilayer perceptron learning using sigmoid response functions and the back-propagation learning algorithm can be interpreted usefully in terms of hyperplane dynamics and mass increase of the system. The parameters describing rotation, translation, and mass of the hyperplane were used for this interpretation. We believe the idea

of a hyperplane mass has not been employed explicitly before in any description of back-propagation. Normally it is mentioned that "weights just get ever larger," without really understanding how important this is. We are aware of only one other technique [7] that indirectly makes use of a quantity similar to mass.

The interpretation was used to explain pictorially the learning dynamics of back-propagation with hidden layers, and how the information is stored in the network in the form of hyperplane mass. The importance of weight initialization was made intuitively obvious. The descent into local minima was also described and explained using the interpretation. It was shown how escape from certain types of local minima may be enacted painlessly. Plasticity of a network was defined, and a method of gentler relearning in non-plastic networks was suggested. It is useful to know whether, in a trained, non-plastic network, further learning of a new pattern will be very destructive. This was viewed in terms of easy and hard rearrangements of the hyperplane system.

This type of detailed network analysis is worthwhile because not only can the problems during learning and generalization be better understood and dealt with, and performance improved, but it also allows a number of useful network state and learning parameters, such as plasticity, pattern charge, interaction range, to be defined and used advantageously in the systematic construction of specific neural networks.

10. Acknowledgments

The author would like to thank Jörg Kindermann, Alexander Linden, and Christoph Tietz for reading the original manuscript, and fellow researchers in the GMD Adaptive Systems groups for generally thought-provoking (and time-consuming!) discussions. This work is part of the GMD Adaptive Systems REFLEX project, supported by BMFT grant number 01IN111A/4.

References

[1] S. Ahmad and G. Tesauro. "A Study of Scaling and Generalization in Neural Networks," in *Abstracts of the First Annual Meeting of the INNS, supplement to Neural Networks, Vol. 1*, Boston, MA (September 1988).

[2] T. Ash, "Dynamic Node Creation in Backpropagation Networks," *Connection Science*, **1**(4) (1989) 365–375.

[3] D. S. Broomhead and D. Lowe. "Multivariable Functional Interpolation and Adaptive Networks," *Complex Systems*, **2** (1988) 321–355.

[4] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," in *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky (San Mateo, CA: Morgen Kaufmann, 1990).

[5] M. Frean, "The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks," *Neural Computation*, **1** (1990) 198–209.

[6] J. F. Kolen and J. B. Pollack, "Back Propagation is Sensitive to Initial Conditions," *Complex Systems*, **4** (1990) 269–280.

[7] J. K. Kruschke and J. R. Movellan, "Benefits of Gain: Speeded Learning and Minimal Hidden Layers in Back-Propagation Networks," *IEEE Transactions on Systems, Man and Cybernetics*, **21**(1) (1991) 273–280.

[8] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, (April 1987) 4–21.

[9] P. Lorraine and D. Corson, *Electromagnetic Fields and Waves* (New York: W. H. Freeman, 1970).

[10] M. Mézard and J-P. Nadal, "Learning in Feedforward Layered Networks: The Tiling Algorithm," *Journal of Physics A*, **22**(12) (1989) 2191–2203.

[11] H. Mühlenbein, "Limitations of Multilayer Perceptrons: Steps Towards Genetic Neural Networks," *Parallel Computing*, **14**(3) (1990) 249–260.

[12] F. Rosenblatt, *Principles of Neurodynamics* (New York: Spartan Books, 1959).

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams," "Learning Internal Representations by Error Propagation," *Nature*, **323** (1986) 533.

[14] A. Schütte, "How Can Multi-layer Perceptrons Classify," in *Proceedings of the Workshop on Distributed Adaptive Neural Information Processing*, GMD, St Augustin, April 1989 (Oldenbourg Verlag, 1989).

[15] F. J. Śmieja, "Neural Network Constructive Algorithms: Trading Generalization for Learning Efficiency?" *Circuits, Systems and Signal Processing*, **12**(2) (1993) 331–374.

[16] F. J. Śmieja and H. Mühlenbein, "The Geometry of Multilayer Perceptron Solutions," *Parallel Computing*, **14** (1990) 261–275.