The Effect of Hebbian Learning on Optimisation in Hopfield Networks

Richard A. Watson, C. L. Buckley, Rob Mills.

In neural networks, two specific dynamical behaviours are well known: 1) Networks naturally find patterns of activation that locally minimise constraints among interactions. This can be understood as the local minimisation of an energy or potential function, or the optimisation of an objective function. 2) In distinct scenarios, Hebbian learning can create new interactions that form associative memories of activation patterns. In this paper we show that these two behaviours have a surprising interaction – that learning of this type significantly improves the ability of a neural network to find configurations that satisfy constraints/perform effective optimisation. Specifically, the network develops a memory of the attractors that it has visited, but importantly, is able to generalise over previously visited attractors to increase the basin of attraction of superior attractors before they are visited. The network is ultimately transformed into a different network that has only one basin of attraction, but this attractor corresponds to a configuration that is very low energy in the original network. The new network thus finds optimised configurations that were unattainable (had exponentially small basins of attraction) in the original network dynamics.

1. Introduction

In this paper we investigate the interaction of two well-known properties of complex systems that have each been independently well-studied in neural networks: i) The energy-minimisation behaviour of dynamical systems (Hopfield 1982) which can be interpreted as a local optimisation of constraints (Hopfield & Tank 1985, 1986), and ii) Hebbian learning (Hebb 1949) with its capacity to implement associative memory (Hopfield 1982, Hinton & Sejnowski 1983). Specifically, we show that Hebbian learning significantly enhances the probability that a dynamical system arrives at low-energy attractors, and that the attractors thus found optimise constraints to otherwise unattainable levels. We view the effect as an extension of the 'emergent collective computational abilities' (Hopfield 1982) that come 'for free' in physical systems.

Our models employ the Hopfield network (Hopfield 1982) which is an abstract model of neural networks and a well-understood example of a simple dynamical system that has provided a vehicle for studying attractor dynamics across many disciplines.

Optimisation and model induction in Hopfield networks

Many natural dynamical systems have behaviours that can be understood as the local minimisation of an energy or potential function (Strogatz 1994). Hopfield networks, for example, are recurrent neural networks with symmetric weights and no positive self-recurrent connections; these conditions guarantee that the dynamics of the network can be described as the minimisation of an energy function and that the network exhibits only point attractors. Shortly after their initial introduction it was suggested that Hopfield networks can be used to solve optimisation problems (Hopfield & Tank 1985, 1986), Fig.1.a, and an extensive literature has developed on this, and similarly the optimisation behaviour of their stochastic counterpart, the Boltzmann machine (Hinton & Sejnowski 1985, Ackley et al 1985). The energy function simply corresponds to the degree to which internal network constraints remain unsatisfied – the more unsatisfied constraints, the higher the energy, and a state change that reduces energy resolves more constraints than it violates. Minima in this function thus correspond to attractors in the network dynamics that are locally optimal resolutions of these constraints. However, difficult optimisation problems, or networks with interactions that are difficult to resolve, have many local optima and this can create a Hopfield network that has a large number of local attractors. Running the network will obviously not result in optimal solutions in such cases (Tsirukis et al 1989).

A second well-known neural network behaviour, model induction, is indicated in Fig.1.b. Training a dynamical system to have a particular energy function may be interpreted as a model induction process which takes as input a set of points in configuration space, 'training patterns', (fig 1.b, left) and returns a model of those points (1.b, centre). The model may act as an associative or content addressable memory (right) (Hopfield 1982) which takes as input a (possibly partially specified) input pattern and 'recalls' the training pattern that is most representative of that pattern. Such a memory can be implemented with a dynamical system whose attractors correspond to the training patterns. Unlike the optimisation scenario, where one would ideally like to avoid local optima, this use of Hopfield networks exploits the fact that complex networks can have multiple attractors. A Hopfield network may be trained to implement such a dynamical system with Hebbian learning. In an associative memory, the intent may be to represent the original training patterns as accurately as possible, or the training patterns are sometimes interpreted as being a sample of some underlying distribution of points and the intent is to generalise from the training patterns to estimate the true distribution.

For example, in some cases, the learning process may afford some simple forms of generalisation such as the merging training patterns that are very similar into one class that becomes represented by an idealised exemplar (solid point in Fig.1.b, centre). An appropriately trained Hopfield network may thereby both classify patterns into different groups and generalise patterns within a group – an appropriate balance will produce a general model that is not over-fitted to the training set. Much is known about the capacity of such networks, i.e. the number of patterns they can store and their limitations with respect to storing very similar patterns (McEliece et al. 1987). In particular, the recall of 'spurious' patterns – patterns that are substantially different from all patterns in the training set – is naturally considered to be a problem and something to be avoided in associative memory (e.g. Gascuel et al 1994). A particularly successful approach for avoiding spurious attractors uses a combination of Hebbian and anti-Hebbian mechanisms in interleaved phases (Hopfield et al. 1983). The Hebbian learning captures associations in the training set and the anti-Hebbian learning is used to counteract the development of spurious attractors arising in the inherent dynamics of the network. This can enable the network to learn models of data where only a subset of the states is visible to the learning process, and consequently to learn models that are not a simple linear combination of pair-wise dependencies (Hinton & Sejnowski 1985).

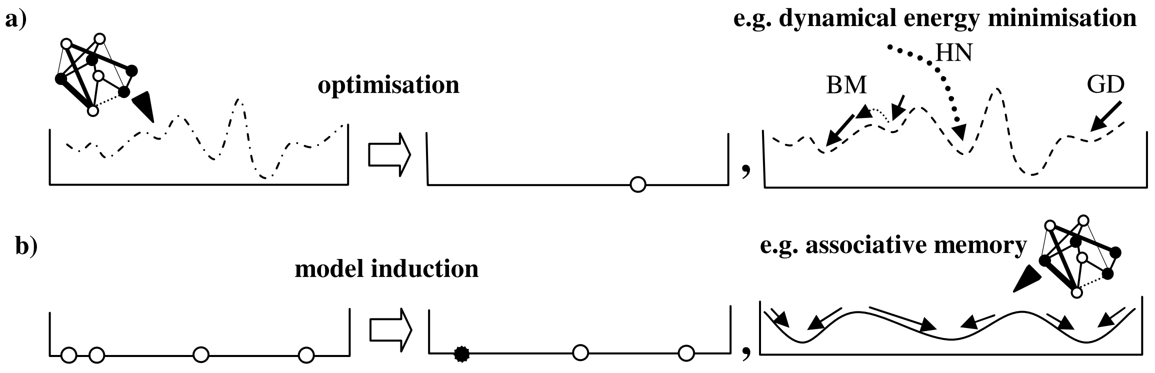


Fig 1. Optimisation and model induction. a) An optimisation process takes as input an implicitly defined function over a space of configurations (left), perhaps defined by a network of dependencies among a set of problem variables (above), and returns, in the ideal case, a single point in configuration space that corresponds to the minimum of that function (centre). Various stochastic local search processes (right) are imperfect methods for approximating this output: gradient descent (GD), Boltzmann machine (BM), Hopfield network optimisation (HN). All of these methods suffer the restrictions on energy minimisation imposed by local optima. b) Model induction is a process that takes as input a set of points in configuration space, 'training patterns', (left) and returns a model of those points (centre). The model may act as an associative or content addressable memory (right) which takes as input a (possibly partially specified) input pattern and 'recalls' the training pattern that is most representative of that input. Such a memory can be implemented with a dynamical system, such as a Hopfield network, trained by Hebbian learning to exhibit attractors that correspond to the training patterns (right) – see text.

Optimisation and model induction form complementary parts of a picture of organismic behaviour: For example, a neural network may be trained by Hebbian learning to represent a distribution of stimuli and the subsequent energy minimisation behaviour of this network accesses an associative memory that interprets a new, perhaps partial, stimulus by resolving constraints among competing 'hypotheses' about that stimulus and 'recalling' an exemplar pattern (Hinton & Sejnowski 1985). But the notion that energy minimisation (in the Hopfield network, Hopfield & Tank 1985, for example) performs effective optimisation is inconsistent with the notion that energy minimisation can recall local attractors in an associative memory (Hopfield 1982). Specifically, if recall works well it will find one of many local optima that represent each of the input patterns; but when optimisation works well it will not return a local optimum but the globally-minimum-energy optimum. If Hopfield networks were effective optimisers then in memory terms it would mean that all stimuli appeared to be the same pattern. In practise this is not a problem because, in fact, the optimisation afforded by dynamical energy minimisation is, put bluntly, not a very effective optimisation process.

Gradient descent (GD) (fig.1.a, right), the most basic form of local search, will necessarily find a local minimum in an energy function; The Boltzmann machine (BM) descends the energy surface but with a non-zero probability of admitting energy increases that may enable escape from local optima; The use of Hopfield networks (HN) for optimisation may find superior minima in some cases by allowing movements in a continuous space between the points of the original discrete configuration space (indicated by a trajectory which commences from a point that is not on the original energy surface) (Hopfield & Tank 1985). However, all of these methods suffer the restrictions on energy minimisation imposed by local optima.

Although the Boltzmann machine is proven to asymptotically approach the global optimum of an energy function if its ‘temperature’ (a parameter indirectly controlling the likelihood of escaping local minima) is appropriately annealed (Geoffrey & Sejnowski 1983, Kirkpatrick et al 1983), it nonetheless, is still a stochastic local search method. Low-energy local optima that distract from the globally-minimal optimum are still problematic. In short, there is an inevitable trade-off that any local search method must suffer; to the extent that local gradients are misleading they must be ignored (by allowing energy increases), and to the extent that local gradients are ignored, the time to find low energy states is increased. In the Boltzmann machine this trade-off is very obvious; low temperatures or quickly annealed temperatures find sub-optimal solutions quickly, slowly annealed temperatures can (in the limit) find optimal solutions but require time exponential in the size of the problem to do so. Many modifications and enhancements to the original Hopfield network have been proposed for optimisation purposes but in most cases the basic behaviour of the network remains – a relaxation to a *local* minimum.

The interaction of energy minimisation and associative memory

These two different uses of Hopfield networks (and Boltzmann machines) have an extensive and intertwined literature, but these two uses have apparently incommensurate objectives. It should be clear that model induction tasks, where Hebbian learning has been widespread, do not perform optimisation. Indeed, optimisation takes as input a network that represents a set of constraints or dependencies among problem variables, and if it works well, returns a single point in configuration space that minimises the conflicts or costs of unsatisfied dependencies. In contrast, model induction, takes as input a distribution of training samples, and if it works well, recreates that distribution, potentially capturing regularities and substructures inherent in the training set.

Note that *optimising a model* – optimising the goodness of fit between a model and the training data – is the problem of model induction, not optimisation: It is not the problem of finding the minimum energy state of a dynamical system. Hebbian learning is known to be effective at optimising a model (Ackley et al 1983), but the intent of this process is to output a network that has a specific (multi-attractor) energy function, not to output the minimum-energy configuration of an existing network (*sensu* Hopfield & Tank). Similarly, optimising a desired input-output mapping in a feed-forward network, e.g. with back-propagation or gradient descent (Rumelhart & McClelland 1986), may be assisted by Hebbian learning (e.g. the ‘Leabra’ algorithm O'Reilly & Munakata, 2000). But again this is a different objective from finding the minimum-energy state of a dynamical system, i.e. it aims to output a network that implements an input-output mapping, not a state configuration. In general, the use of Hebbian learning to identify and amplify the principle components of a training set (Linsker 1988) as a pre-processing stage for learning an input-output mapping is also common. The underlying reasons for the improvements in energy minimisation that we demonstrate below are related at a deep level to those demonstrated in optimising models and learning feed-forward networks. But in none of these prior works is an associative memory model (in the style of Hopfield 1982) developed within a Hopfield network that is simultaneously performing optimisation (in the style of Hopfield and Tank 1985, 1986). This is somewhat surprising, given how well-known each of these behaviours is – but perhaps understandable given their incommensurate objectives.

Despite their apparent incongruence, we find that bringing these two behaviours (optimisation via energy minimisation and induction of an associative memory model via Hebbian learning) together in the same network has surprising consequences that are very significant for the ability of dynamical systems to find low-energy attractors. The dynamical machinery involved is both that of model induction via Hebbian learning and of optimisation, but the outcome is an optimisation process because the 'model' that is induced is only a model of the lowest energy configurations. Specifically, a given energy function is transformed into a different energy function such that low-energy configurations, possibly the globally minimal energy configuration of the original system, are easily retrieved.

The basic protocol that we investigate (with variants) is as follows: A network is repeatedly run for some time from different arbitrary initial conditions. At all time steps, Hebbian learning is applied to the weights of the network. Accordingly, this alters the energy function of the network and potentially alters the dynamics of the network considerably. From an optimisation point of view, it might seem that altering the energy function away from something that represents the true objective function cannot be a good thing to do. But the application of Hebbian learning in this manner has systematic and predictable consequences on the energy minimisation behaviour of the network. Specifically, assuming that the duration of each run of the minimisation process is long compared to the time to find a local optimum, most learning occurs at local optima. This causes the system to develop a memory of the attractors that it visits. A learning network capable of inducing an associative memory, is thus 'turned upon itself' – augmenting its behaviour with an induced model of its own behaviour. In so doing the intrinsic behaviour of the network is modified, thus altering future behaviour and future learning, and so on. On the face of it, developing an associative memory of locally optimal attractors seems like it would be fruitless for optimisation. But, the generalisation ability of the learning process, which has arguably been under-appreciated, produces non-trivial effects.

The resultant transformation of the energy function is depicted in Fig 2. and proceeds as follows: i) the natural energy minimisation behaviour of a system repeatedly samples local optima in the energy function. These act as 'training samples' for the concurrent development of an associative memory that (imperfectly) models the original function. ii) Simple generalisation resultant from the training process may cause subsets of similar patterns to be represented by a single idealised exemplar in the associative memory (solid point). As local sampling continues in the energy function now augmented by the learned model, a slightly different distribution of local optima determines the training samples that further update the model. Occasionally, the training process may create 'spurious attractors' that do not correspond to any of the training points. iii) As sampling continues on this modified model, the distribution of local optima used for subsequent training becomes a more and more degenerate representation of the original function, including points that correspond to 'spurious' attractors (shaded). Because we are not using an anti-Hebbian phase or any other mechanism to deter them, the model increasingly amplifies spurious attractors that, through an (as yet unexplained) generalisation principle, come to correspond to the lowest energy attractors of the original function. Ultimately the modified system becomes a 'model' of the global optimum of the original function; That is, its only attractor corresponds to the global optimum of the original function.

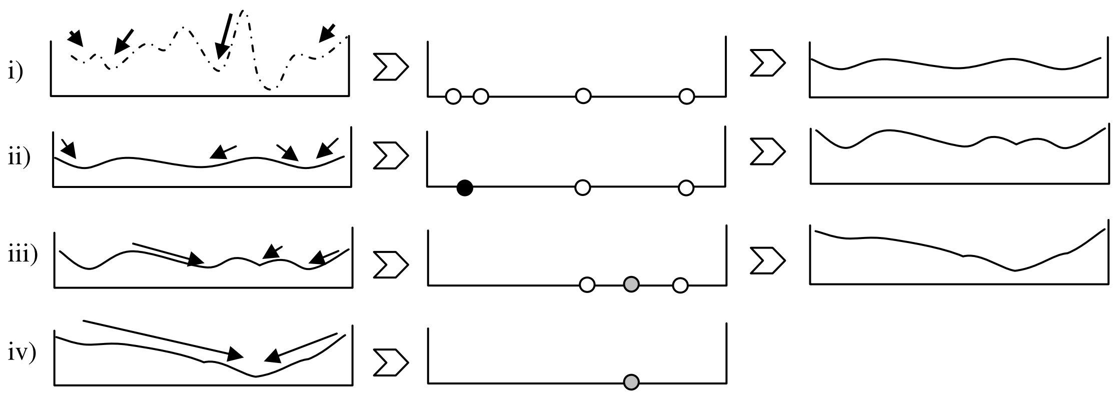


Fig 2. Overview of how Hebbian learning modifies the energy function of a dynamical system. We investigate the ability of associative memory to transform a complex function into a different function which is easier to optimise. This is achieved via a continuous process of sampling local optima and inducing a model of them that becomes an increasingly generalised representation of the original function (i-iii), see text. iv) Ultimately, (through some generalisation principle as yet unexplained), its only attractor corresponds to the global optimum of the original function – see text.

To clarify, it is worth emphasising what exactly is shown in this effect. We start with a dynamical system with complex constrained interactions that produces many local optima. In general, relaxation of the network results in a configuration that is locally optimal but possibly far from the minimum energy attractor that is possible in this network. The globally minimal configuration of this network is rarely visited and in a finite sample of initial conditions it may remain unvisited with high probability. Let us, so to speak, take a copy of that original network and put it to one side. Now we apply Hebbian learning to the network, as it repeatedly visits different local attractors, as described above. The result of this is that the network is modified into a new network. This network no longer exhibits the full range of behaviours that the original network did, and in the limit has only one attractor. The globally minimal energy configuration of this new network is easy to find – the network finds this configuration by relaxation from any initial condition. We then compare the one attractor of this new system to the attractors of the original system we saved earlier. We find that the configuration found at this one attractor is a configuration that has very low energy in the original system, and under certain conditions, is actually the configuration that was the globally minimal energy configuration of the original system. Hebbian learning does not merely modify the network into a 'simpler' network with fewer attractors, but the attractors of the new system have a special relationship to the attractors of the original system in that they are especially low-energy configurations.

If we understand that actually, the lowest energy attractors of a network like this are in fact the largest attractors of the network (Fontanari 1990) – then this seems less mysterious. The more an attractor is visited, the more it is learned, and if visitation is proportional to the size of the basin of attraction and large attractors are correlated with low energy, then the learning learns the low-energy attractors. The effect would be interesting even if all it showed was that as a dynamical system repeatedly sampled its local attractors, the attractors that are visited most often (having the largest basins of attraction and on average tending to have the lowest energy) become 'over-learned' such that they become the only attractors of the system. This alone would indicate that Hebbian learning can be used not just to model an energy function or distribution of samples but to modify that energy function such that the lowest energy patterns are found more quickly and reliably, and that the basin of attraction for these attractors is enlarged making these configurations more robust to perturbations. But the effect we illustrate is not just this.

The more surprising aspect of the effect is that the point attractors in the trained system correspond to point attractors of the original system that would not have been sampled on this timescale without the learning process. That is, although these point attractors existed in the original system, and they had larger basins than other attractors, their basins of attraction were actually very small – so small that, on average, they would not have been visited. If this were not the case, they would not be difficult optimisation problems. That means that the attractors for these optima are enlarged by the learning process *before* they are visited for the first time. This enlargement occurs by the same mechanism as the creation of what would be called spurious attractors in associative memory models, meaning that they are learned patterns that were not in the training samples used for the model induction. But rather than view them as erroneous distractions we show that under general circumstances they are in fact reasonable 'predictions' about the location of low energy attractors that have not yet been visited.

Indeed the difference between these two scenarios, a recall of good configurations (that have been visited) and a prediction of good configurations (that have not been visited), is the difference between a memory and an optimisation process. A process that can only recall good solutions that have already been seen would not make a very good optimisation process. That is, enumerating the local optima of the function and inducing a model from them will not aid optimisation – if the local optima are enumerable, then finding the global optimum is trivial/already done. But a process that builds a model of a function so as to predict the location of its optima facilitates a non-trivial optimisation process, and is a much more interesting behaviour for a distributed learning process to exhibit. In this scenario we can see that memory overcomes the conflict between following or ignoring local gradients that is inescapable for stochastic local search. That is, a stochastic local search process without memory will lose the information inherent in one local optimum when it accepts an energy increase that enables it to escape that optimum and visit another. But the process we show here retains useful information from each optimum visited which informs future search trajectories.

Note that the process of using local attractors, found from arbitrary starting vectors, as training patterns for the Hopfield network has been utilised in other work (Robins 1995, Robins & McCallum 1998) – but in that context this process is used to avoid forgetting previously learned training patterns rather than for optimisation purposes. Some prior work using Hopfield networks for optimisation also uses local optima to modify weights which in turn modify the future behaviour of the network. Serpen (2008) modifies the weights of the network via an error function that seeks to avoid locally optimal solutions that are invalid. Similarly, Tang et al. (2001) modify the weights of the network to reduce the likelihood of revisiting local optima. In fact, this is the opposite intuition from the work presented here – these authors fail to recognise that every local optimum, rather than being an obstacle, in fact, has useful information about features of the global optimum. Chen (1998) modifies a set of weights to make future trajectories more likely to reach the same optima – correctly recognising that local optima share features in common with the global optimum. But these are not the pair-wise weights of the original network; they are simply a vector of univariate probabilities that alter the initial conditions of future runs. Thus none of these works using the Hopfield network recognise that simple Hebbian learning on local optima can be used to amplify global optima.

However, the idea of building a model from a sample of optima from a local optimisation process, specifically a hill-climber in the form of a '(1+1)EA', is seen in (Iclanzan & Dumitrescu 2007, 2008a, 2008b). Also, the notion of using a neural network to model the dependencies between problem variables is discussed in (Iclanzan & Dumitrescu 2008b). Although this work is not described as a Hopfield network, the learning rule is more complex than Hebbian learning, and the model building is not described as an associative memory or a modification of dynamical attractors, some underlying ideas are similar. Specifically, Iclanzan and Dumitrescu exploit the fact that local optima contain useful information about the location of better optima/the global optimum. Like earlier work of our own that attempts to learn pair-wise dependencies among variables (Watson & Pollack 2002, Watson 2006, Mills and Watson 2007), this work is derived from the evolutionary algorithms literature and is influenced by ideas of linkage learning (Harik & Goldberg 1996) and 'estimation of distribution algorithms' (Pelikan et al. 1999) that attempts to identify problem structure. The present work identifies problem structure in a simple but subtle manner via model induction, and this model is used to modify the optimisation process (network relaxation) in a similarly simple and direct manner. Bringing the two together in the same network provides a distributed process that exploits problem structure to accelerate optimisation. Mills & Watson (in prep) provides a probabilistic model building process based on these principles and proves its superior scalability with respect to ordinary evolutionary processes. Importantly, this work (unlike the present work) includes the explicit encapsulation of variables that have been identified as strongly interdependent into composite variables.

In Sections 2 and 3 we detail our models and investigate the affordances and limitations of the effect.

2. Models

Hopfield networks, energy functions and Hebbian learning

The state of a Hopfield network consisting of N discrete states $s_i = \pm 1$where $i=1,2,\ldots,N$ can be written as $S=(s_1,\ldots,s_N)$. The dynamics of a recurrent neural network used by Hopfield can be described by updates to individual states:

$$s_{i\,(t+1)} = \theta\left[\sum_{j}^{N} \omega_{ij} S_{i\,(t)}\right] \tag{1}$$

where ω_{ij} are elements of the connection matrix Ω, and θ is the Heaviside threshold function (taking-values -1 and +1 for negative and positive arguments respectively). The Hopfield network is run by repeatedly choosing a unit, i, uniformly at random and setting its state according to the above formula. Hopfield showed that if the connection matrix is symmetric $\omega_{ij}=\omega_{ji}$, and under suitable constraint on the self-weights (here $\omega_{ii}=1$), all trajectories described by Eq. 1 converge on point attractors which are minima of the energy function given by

$$E_S = H(S,\Omega) \equiv -\sum_{ij}^{N} \omega_{ij} s_i s_j \tag{2}$$

Consequently one can describe the asymptotic behaviour of such a network in terms of a process that minimises this function. The Boltzmann machine is a discrete stochastic counterpart of the Hopfield network where a single state change is accepted probabilistically according to the change in energy it produces. We can describe such a dynamical process more generally via a probability of accepting a stochastic change to the system state:

$$\mathrm{P}[S_{(t+1)} \leftarrow f(S_{(t)})]=\sigma(T,\Delta E) \tag{3}$$

where the operator f is, in the Boltzmann or discrete Hopfield cases, a 'bit flip' operator defined as $f(S) = (s_1,\ldots,-s_X,\ldots,s_N)$ where X is a uniform random variable on $[1,\ldots,N]$, $\Delta E=E_{S'}-E_S$ is the change in energy implied by the new state $S'=f(S)$, and $\sigma(T, x) = \dfrac{1}{1+\exp(-\frac{1}{T}x)}$ is a sigmoid function of x where T is the *temperature* of the system; a parameter that indirectly controls the probability of an increase in energy. When the temperature is reduced gradually this describes a simulated annealing process. But for a deterministic system, as T=0, we can simply write:

$$\mathrm{P}[S_{(t+1)} \leftarrow f(S_{(t)})]=\theta'(\Delta E). \tag{4}$$

Where θ' is a threshold function taking values 0 and 1 for negative and non-negative arguments respectively. Thus, the discrete Hopfield network and the Boltzmann machine with T=0 are both equivalent to a bit-flip gradient descent algorithm. Although the continuous state version of the Hopfield network and the non-zero temperature Boltzman machine are, in many circumstances, better optimisers than the bit-flip gradient descent algorithm, these enhancements are not required for the effect we investigate in this paper and the discrete deterministic process is sufficient. This type of 'perturbation' model, updating the network by accepting a stochastic change in state if and only if it decreases system energy, is conventional in stochastic local search, whereas the Hopfield model, which deterministically modifies system states in the direction that minimises energy, is conventional in neural networks. However, for our purposes these frameworks are interchangeable; all that concerns us here is the property that these models use an energy function defined by a set of pair-wise connections or weights, Ω. The stochastic framework also permits an interesting extension to allow the incorporation of macro state changes (Watson et al 2009c).

Since these deterministic dynamics will quickly find a local optimum in the energy function, it is useful to consider a 'random restart' version of the model where the state of the system takes a random state configuration, $R=\{-1|1\}^N$, every τ time steps[1]. We refer to each inter-reset duration of τ time steps as a *relaxation*, and assess the minimum energy and average energy of states visited by the system over many relaxations. Note that the energy of configurations visited within a relaxation will be minimal at the end of that period, so it is often useful to monitor the energy of the system at that time. If each relaxation is long enough for the system to reach a local optimum, this will thus monitor the local optima visited by the system.

Thus far we have assumed that the connection matrix, Ω, remains constant but we can now combine the restart Hopfield network above with a Hebbian learning mechanism such that for all ω_{ij}, $i \neq j$:

$$\omega_{ij}(t+1) = \theta[\omega_{ij}(t) + \frac{\delta}{\tau} s_i(t) s_j(t)]$$

where (δ/τ) is a learning rate. All weights are capped at a magnitude of 1 by the θ function. If we assume that $\tau >> t^*$, where t^* is the time for the system to reach a local optimum then the cumulative effect of learning over a relaxation will be approximately equal to a single application of learning at the end of each relaxation:

$$\omega_{ij}(\tau+1) = \theta[\omega_{ij}(\tau) + \delta s_i(\tau) s_j(\tau)] \qquad (5)$$

This end-of-relaxation learning is computationally less expensive to implement so we use this in our experiments, but a continuous learning model may be more natural in biological systems.

Note that for any two weight matrices x and y, $H(S,x+y)=H(S,x)+H(S,y)$. Accordingly, we can separate the effect of the original weight values, Ω^0, from the effect of the accumulated weight changes that result from learning, Ω^L; i.e. $\Omega=\Omega^0+\Omega^L$, such that $E=H(S,\Omega)=H(S,\Omega^0)+H(S,\Omega^L)$. In fact, in principle, the original behaviour of the system need not be governed by a weight matrix but by some arbitrary energy function, $e(S)$, which is just a function of the state. The behaviour of the modified/learning system is thus governed by $P[S_{(t+1)} \leftarrow f(S_{(t)})]=\theta'(\Delta E)$, where either

$$E=H(S,\Omega) \text{ and } \omega_{ij}(\tau+1) = \theta[\omega_{ij}(\tau) + \delta s_i(\tau) s_j(\tau)] \qquad (6)$$

or

$$E=e(s)+H(S,\Omega^L) \text{ and } \omega^L_{ij}(\tau+1)=\theta[\omega^L_{ij}(\tau)+\delta s_i(\tau)s_j(\tau)]. \qquad (7)$$

In the second version (Eq. 7) the original energy function, $e(S)$, may thus be a function of unknown internal structure that we wish to optimise, an external objective function, (and may or may not be representable as a sum of pair-wise dependencies), and Ω^L, initially $\omega^L_{ij}=0$, is the accumulated changes to weights that result from learning. Even in the case where $e(S)$ *is* representable as a sum of pairwise dependencies, i.e. $e(S) =H(S,\Omega^0)$, it is useful to keep the original components of the weights conceptually separate from the learned components of the weights i.e. $E=H(S,\Omega^0)+H(S,\Omega^L)$ and $\omega^L_{ij}(\tau+1)=\theta[\omega^L_{ij}(\tau)+\delta s_i(\tau)s_j(\tau)]$.

We can thus view the learning process either as a mechanism that modifies the behaviour of a dynamical system by modifying the weights of the system directly (Eq.6), or even in the case where this separation is not implemented explicitly, we can equivalently view it as a process that augments the energy function of a dynamical system, or a local optimisation process, with a set of learned weights, Ω^L (Eq.7). From the latter point of view we can see that Ω^L will become an associative memory model of (the local optima of) the original energy function/objective function. But importantly, since the learned weights model the local optima of the original system (not its full Boltzmann distribution), Ω^L becomes a memory for predicting the dynamical outcome of $e(S)$ or the behaviour defined by the original weights, shortcutting transient dynamics where a greater number of

[1] Use of the more general Boltzmann conditions, $T>0$, may alleviate the need for this, but this framework suits the current purposes.

dependencies are in tension. Moreover, since the learning attends to the dynamical outcome of its own influence there will necessarily be positive feedback in the learned weights values (be they embedded in a modified Ω or separate in Ω^L). And, in a system that uses no normalisation or any other means to prevent the learned weights from growing in magnitude as is the case here, the learned weights will come to dominate the behaviour of the system. This will eventually canalise the dynamical behaviour into a single attractor, overpowering all other attractors that were present in the original behaviour of the network. However, our results will show that the resultant attractor is systematically lower in energy than the average attractor of the original system (measured using $e(S)$ or the original weights, Ω^0) and under some conditions is the globally-minimal energy configuration of the original system.

To summarise,

- Our non-adaptive dynamical model, *M0*, is a Hopfield network that uses the standard (bit-flip) perturbation operator, *f*, and discrete state dynamics, *Eq.* 4, with multiple relaxations restarted randomly after each τ time steps.
- Our learning model, *M1*, is the same as *M0* but with the modified energy function resulting from the learned weights (Eq. 6 for the intrinsic modification of a Hopfield network, or Eq. 7 for an external objective function).

Accordingly, *M1* uses learning to modify the energy function. Note that when δ=0, *M1* is identical to the original non-adaptive Hopfield network, *M0*. Similarly, the initial behaviour of *M1* before significant learning has taken place is qualitatively identical to *M0*.

Illustrative Energy Functions

To assess the limitations and affordances of the Hebbian model (*M1*) we examine its ability to find optimal state configurations compared to that of the non-adaptive behaviour of the Hopfield network (i.e. *M0*). The energy functions are represented by different types of weight matrices, Ω^0, as below. We examine two different classes of system: a spatial connectivity matrix (S1) and an explicitly modular matrix (S2). These choices facilitate supporting analysis and illustration. We start with the spatial system since this provides the most easily interpretable behaviour, enabling us to build intuition for the behaviours we observe.

S1. **Spatial connectivity**: w_{ij}=(0, 1), a symmetric Gaussian Toeplitz matrix defines range of weight values, i.e.

$w_{ij}=X$, where X is a random value drawn uniformly in the range (0, e^{-d}), and d is the distance between i and j in 1D torriodal space (i.e. a ring), i.e. d=mod(|i-j|,N).

S2. **Modular connectivity with parameterised strength of inter-module connections**: $w_{ij}=\{1, 0<p<1\}$. Specifically, all intra-module weights=1, all inter-module weights=p.

$w_{ij}=1$, if $\left\lfloor \frac{i}{k} \right\rfloor = \left\lfloor \frac{j}{k} \right\rfloor$, $w_{ij}=p$, otherwise.

where k is the size of modules. N=100, k=5.

The modular connectivity matrix, *S2*, (Mills & Watson 2007, Watson & Jansen 2007) provides a scenario where the global optimum is known and the probability of finding the global optimum without this effect is known to be small. In particular, for a large range of $0<p<1$, all configurations described by the regular language $(-1^k|1^k)^Z$, where k is the size of modules and $Z=N/k$ is the number of modules, are local optima. Here we use N=100, k=5 (number of blocks, Z=20). For these parameters, the following observations about the resultant landscape are notable (see Fig 3):

a. p=1 (unstructured). Only two local optima (global minima), which are easy to find. *'easy'*.
b. $\varepsilon<p<q$ (midrange structure). A balance of inter and intra module dependencies creates a structure that is neither undifferentiated nor fully separated. Global optima have small basins of attraction, but confer a significant difference in energy when found. $\rightarrow$ *'intermediate difficulty (intermediate reward)'*. (q is a threshold below which all

configurations described by the regular language $(-1^k|1^k)^Z$ are local optima. For these parameters, $0.01<q<0.1$ – see Fig.5).

c. p=ε (almost separable modules). Two global optima of 2^{20} local optima with almost equal sized basins of attraction. i.e. very weak inter-module strength makes global optima very difficult to find (each basin ≈ 2^{-20} of configuration space). But all local optima have nearly the same fitness as the global optima. *Very difficult to find the global optimum (but not much energy benefit in doing so).*

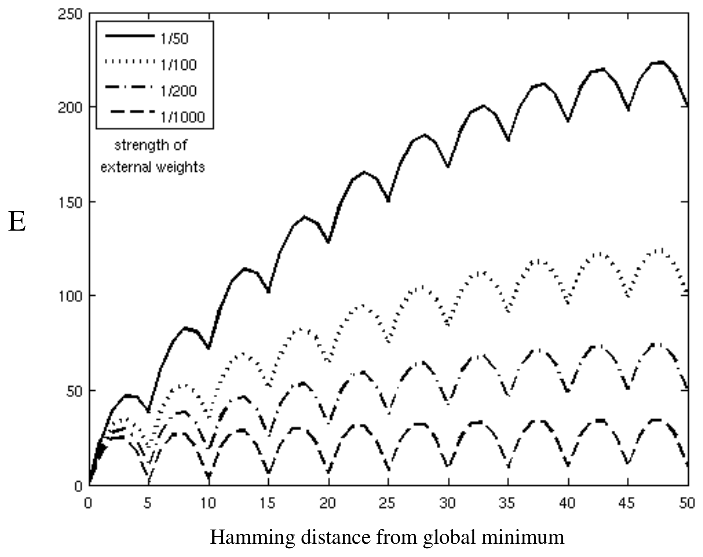


Figure 3: Energy landscapes for extreme and intermediate values of *p* in S2.

Note that when p=ε the basins of the two global optima (-1^N and 1^N) are each only $\approx 2^{-Z}$ of the total configuration space. Accordingly, these functions cannot be solved by a stochastic local search process that allows only single-bit changes, such as the non-adaptive Hopfield network, in less than time on average exponential in Z, the number of modules (Watson & Jansen 2007). In principle, a random mutation hill climber that allows several simultaneous (uncorrelated) bit changes can escape local optima but the expected time to flip the bits that need changing without changing the bits that are already correct is an exponential function of k (Watson & Jansen 2007). Thus no stochastic local search process can find the global optimum of these functions in polynomial time when p is sufficiently close to 0 and the number of blocks and size of blocks are both a function of N (Watson & Jansen 2007).

3. Results

Experiment 1. Illustrations using spatial structure problem.

This experiment uses *M1* on the spatial model *S1*. Figure 4 shows three illustrative runs of *M1* with different learning rates. Because it uses a one dimensional spatial structure it is informative to depict state configurations using a 1D bit pattern (horizontal) changing over time in the vertical dimension.

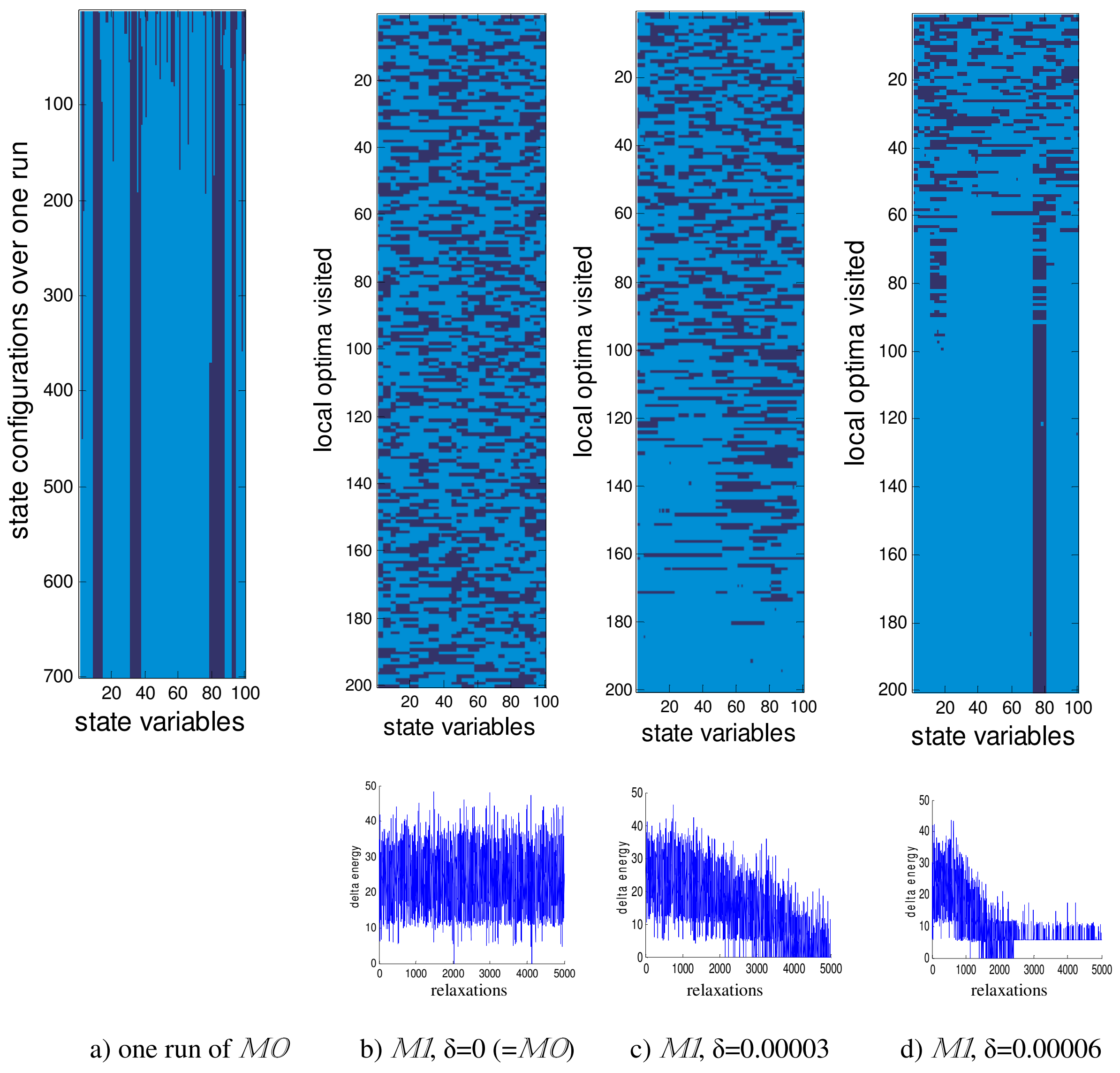
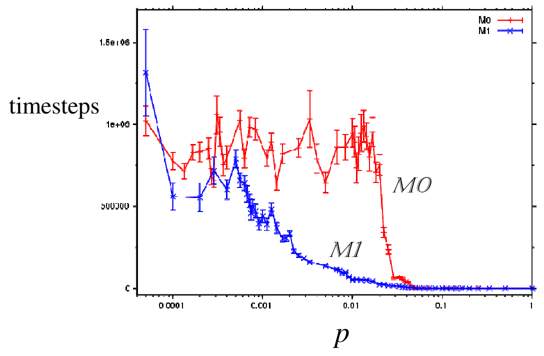


Figure 4: The behaviour of (*M0* and) *M1* on an energy function with a spatial dependency structure. a) The behaviour of *M0* over one relaxation. The attractor state at t=700 shows multiple internally satisfied domains (contiguous regions of variables whose internal constraints are satisfied – i.e. same colour). (Transients are short compared to the time spent at the attractor state so, in the implementation of *M1* learning can be applied once at the end of each relaxation rather than on every intermediate state.) b) Attractor states visited by *M1* with no learning applied (identical therefore to *M0*). Each row of this figure is a final state from a relaxation like that shown in (a). The lower frame shows the energy of the attractors visited in each relaxation. This has high variance but shows no trend. c) *M1* with an appropriate learning rate. The figure shows 200 attractor states sampled at regular intervals over a total of 5000 relaxations. The average size of domains increases in later samples, until eventually the domain size equals the whole system – i.e. the globally minimal energy attractor is found. (Since each attractor state in a Hopfield network has a complementary attractor, the colouring uses the lighter shade for the majority state of each attractor). d) This example shows what happens when the learning rate is too high, a single state configuration becomes the global attractor but this is not the globally minimal attractor of the original system (i.e. it contains two domains not one).

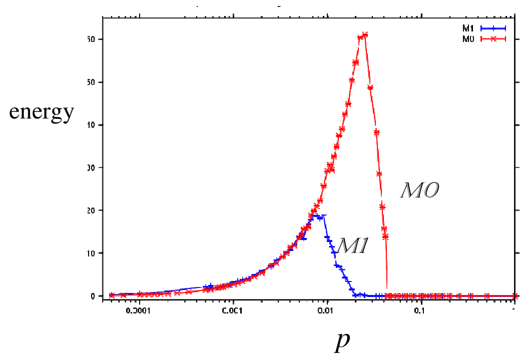
We can understand how the optimisation process works as follows. For this illustration we have used only positive weight values because in this case states whose mutual constraints are satisfied have the same value, and can be shown as the same colour – making it easy to visualise how well constraints are being satisfied. In the beginning of the single run shown in Fig. 4.a. the size of domains (contiguous regions where all internal dependencies among variables are satisfied – i.e. one colour) is small. We see that in general any two variables may or may not satisfy their mutual constraints (they agree or disagree with almost equal probability). But as the run continues the energy minimisation process stabilises small domains such that variables that are close are likely to agree. This is simply due to the fact that nearby variables have the strong direct dependencies. States that co-occur at local optima tend to satisfy the constraints between variables better than arbitrary (e.g. initial) configurations – but a local optimum is soon reached and domain sizes cannot increase further. Examining a few different local optima from a few different relaxations at the beginning of Fig 4.b. we see that on average local optima provide some information about the resolution of constraints among variables that are close, but still very little information about the resolution of constraints among variables that are far apart. Over multiple relaxations, Hebbian learning (Fig.4.c) will reinforce the weights between all variables whose states agree and weaken the connections between all variables whose states disagree – so on average it strengthens local connections, and on average long-range connections are strengthened and weakened almost equally often. But as local domains are reinforced this helps the system to find larger domains of agreement in future, and hence more correlations are learned, and so on. The incremental increases in the size of domains at local attractors (at end of relaxations) as learning progresses indicate that the learning incrementally reduces the effective degrees of freedom in the system. That is, as indirect short-range connections are reinforced the average satisfied domain size increases and this makes it possible to learn longer-range connections, and so on. This shows that the learning process is 'canalising' the subsets of variables that co-vary most reliably, hence producing additional correlations, which then become canalised by subsequent learning, and so on.

Experiment 2. Examining sensitivity to strength of inter-module connections

This experiment uses *S2* to examine the magnitude of the effect as the strength of the modular structure is varied. In *S2* we know how many global optima there are, how far apart they are, what their basins of attraction are, and what their energies are. This helps us to understand the limitations of *M1*.



a) time to find global minimum



b) mean minimum energy after 54 relaxations

Figure 5: The behaviour of *MO* and *M1* on an energy function built from an explicitly modular dependency structure. *p* (the strength of inter-module connections) is varied. For large *p* there is no significant difference between intra- and inter-module connections and no local optima are created, *M1*=*MO*. a) Shows that the problem is difficult for *MO* so long as *p* is sufficiently small to create multiple optima. When *p* is intermediate, creating modular interdependency or nearly decomposable structure, the distinction between the mechanisms is greatest (speed of *M1* >*MO*), but as *p* becomes very small *M1* finds it very difficult to resolve inter-module dependencies correctly and is almost as slow as *MO*. b) Shows the energy of optima found by the three methods after 50,000 relaxations. The region where *M1* >*MO* is easily seen. However, we can also see that for very small *p*, the difference in energy between the global minima (energy 0) and the local optima found by *M1* and *MO* becomes insignificant. Thus it is the intermediate range of *p* that is most interesting; here inter-module dependencies are weak enough to be difficult to resolve correctly, but strong enough to make a significant difference to the energies found.

Fig. 5.a. shows the time taken to find the global optimum for a range of *p*. For high values of *p* not all local optima are present and there is no distinction between the models. In the mid range of *p*, all optima are present. Note that the basin of the global optimum is 10^{-6} of the total search space as *p* approaches zero, so the expected number of timesteps for *M0* to find the global optimum is large, and Fig. 5.a. shows a poor performance level of *M0* extends for most of the range of *p*, i.e. the time for *M0* to reach the global optimum decreases only for large *p*. This experiment reinforces the result that low energy attractors are 'predicted' not merely 'recalled': i.e. the average time to find the global optimum with *M1* is less than the time to first hit the global optimum with *M0*. More specifically, *M0* is equally (un)likely to find the global optimum on every relaxation, and since it takes on average about 100 times longer than *M1* to visit it for the first time, the probability of visiting the global optimum on this timescale (i.e. in the time that *M1* takes to find the global optimum) is approximately 0.01. So clearly, *M1* cannot be working by reinforcing visits to the global optimum, and must be enlarging the basin of attraction for the global optimum before it is visited for the first time.

M1 takes longer to find the global optimum as *p* decreases. Progress for *M1* requires that it learn the inter-module dependencies correctly, and this is difficult to do when the inter-module dependencies are weak. Specifically, since the probability of variables in different modules agreeing is only very slightly greater than 0.5 for small *p*, *M1* must sample a very large number of local optima in order to find the correct sign for inter-module weights. Thus as *p* approaches zero the fastest learning rate that enables *M1* to succeed reliably approaches zero and the time to find the global optimum approaches that taken by *M0*.

Note however, that for very small *p*, the difference in energy of the local and global optima is small (see Fig. 4). For intermediate values of *p* inter-module dependencies are strong and if solved provide a significant decrease in energy. It is therefore these intermediate values of *p* that show the greatest differences in the energies of optima found (fig 5.b).

Together the results of these experiments show:

- The effect of applying Hebbian learning to the weights of a network that is repeatedly sampling its own local optima is to enlarge the basin of attraction of low-energy optima/highly-optimised configurations.
- *M1* finds the highly-optimised configurations that it finds significantly faster than the first hit of these attractors by *M0*. Thus it is 'predicting' the position of low energy attractors rather than merely 'recalling' the best attractors visited.
- We can understand how *M1* works as a gradual canalisation of degrees of freedom (see spatial example). Variables that are only weakly correlated in the original system (because they are either weakly connected or only indirectly connected in the original system) become strongly connected by learning. But reducing the degrees of freedom in the correct manner facilitates discovery of better optima.

4. Discussion

Learning bias, generalisation and prediction

The above results show that Hebbian learning is able to convert a difficult problem into a new problem that has the same low-energy minima but is easier to solve; i.e. to convert a difficult problem into an easy version of the same problem. When the learning process works well it enlarges the largest basin of attraction (which is correlated with the lowest energy attractor; Fontanari 1990). But it is important to realise this attractor is not enlarged because it is the attractor that is most often sampled since, as we have shown, it is not sampled at all on these timescales. The really surprising aspect of the results is thus that the network has been trained to retrieve the global optimum on a problem like *S2* despite the fact that on the timescale that it takes *M1* to solve this problem we would not expect *M0* to have sampled the global optimum even once, on average.

In short, this is possible because the generalisation afforded by the learning process is well-matched to the class of systems the learning is applied to; i.e. the learning process can identify pair-wise correlations and the problems we are examining are built from the superposition of many pair-wise dependencies. Although there is no guarantee of such a match when a network is trained to model an arbitrary external function (as in Fig 2.a.ii. and Eq. 7), such a match necessarily occurs when a dynamical system built from pair-wise dependencies is trained to model its own attractors (as in Fig 2.a.i. and Eq. 6). More generally, Hebbian learning is able to identify and amplify the common components of local optima and the superposition of these is a good heuristic for finding configurations that satisfy more constraints than any one local optimum sampled. This process thus exploits correlations in the training patterns provided by the local optima: Whereas correlations are inconvenient for building an associative memory (Gascuel et al. 1994), causing different attractors to merge together, it is exactly the correlated aspects of local optima that enable the process to enhance optimisation and convert a multi-optima energy function into a single-optimum energy function as described.

Exploiting correlations in sub-optimal solutions is a very natural and general inductive bias to exploit – i.e. treating all variables as independent is often overly naïve but a common and useful heuristic in many scenarios, and exploiting pair-wise correlations between variables is the obvious next step in model sophistication – i.e. it depends on only very weak assumptions about the problem domain (see the simplest types of 'estimation distribution algorithms' Baluja 1994, Yu et al. 2003). Note also that the recursive 'compounding' of the learning process may be important in some problems – i.e. the learning in the later stages of the process models the *modified* behaviour of the system rather than the *original* behaviour of the system. This provides the potential to use strong correlations to amplify weaker correlations and make them easier to learn, as is exploited to find the global optimum in the spatial model, *S1*.

It is also important to note that generalisation can occur in two qualitatively different ways: by 'idealisation' and via 'spurious' attractors. The former is simply a process of finding the centroid of the training patterns, whereas the latter can create or amplify attractors that are distant from all training patterns. These two types of generalisation occur via the same learning bias and merely reflect the difference between the learning behaviour in unstructured problems and those have internal structure or modularity, respectively. The simple idealisation kind of generalisation, often referred to as noise reduction (Krebs & Theumann 1993, Branchtein & Arenzon 1992), is limited algorithmically since it exploits only the kind of biases that might be exploited by a more general stochastic local search process (e.g. the Boltzmann Machine with non-zero temperature). In contrast, modular problems (e.g. *S2*), exhibiting large semi-independent sub-systems of variables, create local optima that are separated by wide fitness valleys (energy barriers) and cannot be solved by such simplistic methods. A function like *S2* (with small *p*) cannot be solved by *uncorrelated* multi-variable perturbations (Watson 2006, Watson & Jansen 2007). Nonetheless, in this type of modular problem, spurious attractors canalise the principle components of the training samples whilst allowing these components to vary semi-independently. This can produce the 'recall' of new combinations of subsystem-attractors that are distant from all training patterns in Hamming space but are nonetheless very specific state configurations (Jang et al. 1992). Specifically, new attractors are new combinations of module-solutions – new combinations of the principle components of co-variation found in the local optima.

Even in a case that is provably difficult for $\mathcal{MO}$ (any stochastic local search process takes exponential time) the Hebbian learning protocol can provide high quality solutions in polynomial time (Watson et al., in prep). That is, for large modular systems, the Hebbian learning mechanism can find low-energy configurations that cannot be found by a non-adaptive model in reasonable time. This emphasises the finding that it can be significantly quicker to induce a model of the problem structure from limited samples and use this model to solve the problem than it is to 'brute force' the solution, i.e. using multiple restarts of a stochastic local search process, if the inductive bias of the model building is appropriate. This is, of course, entirely dependent on the problem structure: It in no way implies that this is a general purpose optimisation method (Wolpert & Macready 1997), only that there exist systems, namely modular ones, where the effect is algorithmically significant.

However, it is important to note that the biological significance of our results does not depend on the performance or ability of these models to provide effective optimisation compared to other purpose-built optimisation methods. Biological significance is ensured simply by virtue of the fact that these effects enhance constraint satisfaction ability/optimisation ability compared to ordinary dynamical system relaxation or incremental improvement – even if the enhanced ability is not comparable to a state of the art optimisation method.

Implications for neural systems

Our emphasis in this paper has been on the Hopfield network as an abstract model of a complex system and on optimisation in the abstract. However, Hopfield and Tank (1986) make clear the many ways in which an optimisation process facilitates neurobiological functions from vision (3D scene perception, edge detection, stereopsis, motion detection) and speech understanding to behavioural choice and motor control. Our results show that a neural network can, via the application of simple distributed Hebbian mechanisms, find low energy configurations of neural dynamical systems that cannot be found by local relaxation alone. Hebbian learning mechanisms are ubiquitous in models of brain function and the mechanisms utilised by our results are well within the capabilities of the cortex. The enhanced optimisation process we have demonstrated thus has the potential to enhance a broad class of neurobiological functions. For example, in perception, improved optimisation corresponds to an ability to recognise the general class of an object, rather than recall its immediate identity, or to find a more satisfactory interpretation of an unusual scene, rather than a superficial interpretation. In action selection, the ability to find lower-energy optima corresponds to the identification of a more appropriate (or more general) response given an 'unseen' stimulus, or an appropriate response in a more unusual (difficult to interpret/conflicted) situation.

A more direct 'problem solving' interpretation in higher cognitive function applies when a problem domain is mentally internalised – e.g. where connections represent constraints or dependencies among features of a problem domain as via model induction (Hinton & Sejnowski 1985). Given such a mental model, the repeated relaxation of the network thus corresponds to imagined trajectories through the configuration space of the domain, and locally-minimal configurations may either be interpreted as expected states or predictions about the observed domain, or as locally optimal resolutions of the constraints in the problem domain. Repeated relaxation of the network can thus be interpreted as repeated 'reflecting' on experiences or mentally 'rehearsing' behaviours. And the application of Hebbian learning (using short term potentiation so as not to permanently alter the domain model) results in better predictions or better resolution of problem constraints.

Fernando et al. (in prep) describe an alternative implementation of *M1* in neurally plausible systems and discuss the implications for cognitive function.

Conclusions

In this paper we showed that forming memories of activation patterns with Hebbian learning has a surprising interaction with the ability of a network to find patterns of activation that locally minimise constraints. Learning of this type significantly improves the ability of the network to find configurations that satisfy constraints/perform effective optimisation. The novelty of the contribution here is not so much in the details of the learning process, which is well-understood in neural networks, but in the notion of turning a learning dynamical system to the task of modelling its own energy minimisation behaviour. The surprising outcome of this is that a network does not merely recall the lowest energy attractors that it has visited, but predicts the location of lower energy attractors that it has not previously visited.

One of the appealing features of Hebbian learning is that it provides a simple and completely distributed learning mechanism. These same properties that make Hebbian learning plausible in neural networks also makes it plausible for other types of evolved biological networks (Fernando et al. 2008). The current work utilises Hebbian learning to provide an optimisation process (rather than a model induction process) that is provably superior to local search, and also enjoys the property of being a fully-decentralised mechanism. Accordingly, it is plausible that other systems, aside from neural networks, can exhibit this type of optimisation behaviour. Watson et al. (in prep) shows that, in fact, we should expect Hebbian mechanisms, and this sort of distributed optimisation, in a large class of complex adaptive systems (not just networks evolved for the purpose of performing learning). For example, Watson et al. (2009a) models the development of associations between species in a dynamical ecosystem that follows the same principles (Watson et al. 2009b).

Acknowledgements
Many thanks to Jason Noble and Seth Bullock.

5. References

Ackley, D.H., Hinton, G.E., Sejnowski, T.J. (1985) A Learning Algorithm for Boltzmann Machines, *Cognitive Science*, 9: 147-169.

Baluja, S. (1994) *Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning* (Tech. Rep. No. CMU-CS-94-163). Pittsburgh, PA: Carnegie Mellon University.

Branchtein, M.C., Arenzon, J.J. (1992) Categorization and Generalization in the Hopfield Model. *J. Phys. I* (France) 22019.

Chen, K. (1998) A general learning algorithm for solving optimization problems and its application to spin glass problems. *Europhys Letters* 43:(6)635-640.

Fernando, C., Goldstein, R., Szathmáry, E., (in prep) Copying and Evolution of Neuronal Activity.

Fernando, C. Liekens, A.M.L., Bingle, L.E.H., Beck, C., Lenser, T., Stekel, D.J. ,Rowe, J.E. (2008) Molecular circuits for associative learning in single-celled organisms. *Journal of the Royal Society Interface.* 6(34):463-9.

Fontanari, J.F. (1990) Generalization in a Hopfield network. *Journal de Physique*, 51, 2421-2430.

Gascuel, J.-D., Moobed, B. Weinfeld, M. (1994) An Internal Mechanism for Detecting Parasite Attractors in a Hopfield Network, *Neural Computation* 6, 902-915.

Harik, G.R. Goldberg, D.E. (1996) Learning Linkage. *Foundations of Genetic Algorithms - 1996*: 247-262.

Hebb, D.O. (1949) *The organization of behaviour*. New York: Wiley.

Hinton, G.E., Sejnowski, T.J. (1983) Analyzing Cooperative Computation. In *Proceedings of the 5th Annual Congress of the Cognitive Science Society*, Rochester, NY, May 1983.

Hinton, G.E., Sejnowski, T.J. (1983b) Optimal Perceptual Inference. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 448-453.

Hinton, G.E., Sejnowski, T.J. (1985) Learning in Boltzmann Machines, Cognitiva, 85, Paris, France.

Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities, *PNAS USA*, 79 (8) 2554-2558.

Hopfield, J.J., Tank, D.W. (1985) 'Neural' computation of decisions in optimization problems. *Biol Cybern* 52:141-152.

Hopfield, J.J. Tank, D.W. (1986) Computing with neural circuits: A model. *Science* 233: 625-633

Hopfield, J.J., Feinstein, D. Palmer, R. (1983) 'Unlearning' has a Stabilizing Effect in Collective Memories. *Nature*, 304. 158-159.

Iclanzan, D., Dumitrescu, D. (2007) Overcoming hierarchical difficulty by hill-climbing the building block structure. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007).* 1256-1263.

Iclanzan, D., Dumitrescu, D. (2008a) Large-Scale Optimization of Non-separable Building-Block Problems. In *Proceedings of PPSN 2008*. 899-908.

Iclanzan, D., Dumitrescu, D. (2008b) How can Artificial Neural Networks help making the intractable search spaces tractable. *IEEE Congress on Evolutionary Computation 2008.* 4015-4022.

Jang, J.-S., Kim, M.W., Lee, Y. (1992) A Conceptual Interpretation of Spurious Memories in the Hopfield-type Neural Network. *Neural Networks, 1992. IJCNN., International Joint Conference on.* 1:21-26.

Kirkpatrick, S., Gelatt, C.D. Vecchi M.P. (1983) Optimization by Simulated Annealing. *Science*. New Series 220 (4598): 671-680.

Krebs, P.R., Theumann, W.K. (1993) Generalization in a Hopfield network with noise. *J. Phys. A*. 26 - 3983.

Linsker, R. (1988) Self-Organization In A Perceptual Network. *IEEE Computer*, 21:105-117

McEliece, R.J., Posner, E.C., Rodemich, E.R., Venkatesh, S.S. (1987) The capacity of the Hopfield associative memory. *IEEE Transactions on Information Theory* 33, pp. 461–483.

Mills, R., Watson, R.A. (in prep) Dynamic problem decomposition via evolved symbiotic associations.

Mills, R., Watson, R.A. (2007) Variable Discrimination of Crossover Versus Mutation Using Parameterized Modular Structure. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007).* pp. 1312-1319.

Mills, R., Watson, R.A. (2007) Symbiosis, Synergy and Modularity: Introducing the Reciprocal Synergy Symbiosis Algorithm. *Proceedings of 9th European Conference on Artificial Life (ECAL 2007).* pp. 1192-1201.

O'Reilly, R.C., Munakata, Y. (2000) *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*, Cambridge, MA: MIT Press.

Pelikan, M., Goldberg, D.E., Lobo, F. (1999), *A Survey of Optimization by Building and Using Probabilistic Models*, Illinois Genetic Algorithms Laboratory (IlliGAL) Technical Report, University of Illinois at Urbana-Champaign.

Robins, A. (1995) Catastrophic Forgetting, Rehearsal, and Pseudorehearsal. *Connection Science*, 7, 123 - 146.

Robins, A., McCallum, S. (1998) Pseudorehearsal and the Catastrophic Forgetting Solution in Hopfield Type Networks. *Connection Science*, 7 : 121 - 135.

Rumelhart, D.E., McClelland, J.M. (1986) *Parallel distributed processing- Explorations in the Microstructure of Cognition .Volume 1: Foundations* , MIT Press.

Serpen, G. (2008) Hopfield network as static optimizer: Learning the weights and eliminating the guesswork. *Neural Processing Letters*. 27:(1)1-15.

Strogatz, S.H. (1994) *Nonlinear Dynamics and Chaos*. Addison-Wesley, Reading, MA.

Tang, Z., Jin, H.H., Murao, K., Ishizuka, O., Tanno, K. (2001) A Hill-climbing learning method for Hopfield networks. *Electronics And Communications In Japan Part Iii-Fundamental Electronic Science* 84:(7)28-40.

Tsirukis, A.G., Reklaitis, G.V., Tenorio, M.F. (1989) Nonlinear Optimization Using Generalized Hopfield Networks. *Neural Computation* 1, 511-521.

Watson, R. A. (2006) *Compositional Evolution: The impact of Sex, Symbiosis and Modularity on the Gradualist Framework of Evolution*. Vienna series in theoretical biology, MIT Press.

Watson, R.A., Buckley C. L., Mills, R., (in prep) Hebbian Learning and Distributed Optimisation in Complex Adaptive Systems.

Watson, R.A., Jansen, T. (2007) A Building-Block Royal Road Where Crossover is Provably Essential. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007).* pp. 1452-1459.

Watson, R.A., Mills, R., Buckley, C. L., Palmius, N., Powers, S.T., Penn, A.S. (2009b) Hebbian Learning in Ecosystems: Species that fire together wire together (ABSTRACT). Submitted *European Conference on Artificial Life 2009.*

Watson, R.A., Palmius, N., Mills, R., Powers, S.T., Penn, A.S. (2009a) Can Selfish Symbioses Effect Higher-level Selection? Submitted *European Conference on Artificial Life 2009.*

Watson, R.A., Pollack, J.B. (2002) A Computational Model of Symbiotic Composition in Evolutionary Transitions. *Biosystems*, 69 (2-3). pp. 187-209.

Wolpert, D.H., Macready, W.G. (1997), No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation* 1, 67.

Yu, T.-L., Goldberg, D. E., Yassine, A. & Chen, Y.-p. (2003) Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm, *Proceedings of Artificial Neural Networks in Engineering 2003 (ANNIE 2003)*, 327–332, St. Louis, Missouri, USA, Nov. 2003.