Networks with Learned Unit Response Functions

John Moody and Norman Yarvin
Yale Computer Science, 51 Prospect St.
P.O. Box 2158 Yale Station, New Haven, CT 06520-2158

Abstract

Feedforward networks composed of units which compute a sigmoidal function of a weighted sum of their inputs have been much investigated. We tested the approximation and estimation capabilities of networks using functions more complex than sigmoids. Three classes of functions were tested: polynomials, rational functions, and flexible Fourier series. Unlike sigmoids, these classes can fit non-monotonic functions. They were compared on three problems: prediction of Boston housing prices, the sunspot count, and robot arm inverse dynamics. The complex units attained clearly superior performance on the robot arm problem, which is a highly non-monotonic, pure approximation problem. On the noisy and only mildly nonlinear Boston housing and sunspot problems, differences among the complex units were revealed; polynomials did poorly, whereas rationals and flexible Fourier series were comparable to sigmoids.

1 Introduction

A commonly studied neural architecture is the feedforward network in which each unit of the network computes a nonlinear function $g(x)$ of a weighted sum of its inputs $x = w^t u$. Generally this function is a sigmoid, such as $g(x) = \tanh x$ or $g(x) = 1/(1 + e^{(x-\theta)})$. To these we compared units of a substantially different type: they also compute a nonlinear function of a weighted sum of their inputs, but the unit response function is able to fit a much higher degree of nonlinearity than can a sigmoid. The nonlinearities we considered were polynomials, rational functions (ratios of polynomials), and flexible Fourier series (sums of cosines.) Our comparisons were done in the context of two-layer networks consisting of one hidden layer of complex units and an output layer of a single linear unit.

This network architecture is similar to that built by projection pursuit regression (PPR) [1, 2], another technique for function approximation. The one difference is that in PPR the nonlinear function of the units of the hidden layer is a nonparametric smooth. This nonparametric smooth has two disadvantages for neural modeling: it has many parameters, and, as a smooth, it is easily trained only if desired output values are available for that particular unit. The latter property makes the use of smooths in multilayer networks inconvenient. If a parametrized function of a type suitable for one-dimensional function approximation is used instead of the nonparametric smooth, then these disadvantages do not apply. The functions we used are all suitable for one-dimensional function approximation.

2 Representation

A few details of the representation of the unit response functions are worth noting.

Polynomials: Each polynomial unit computed the function

$$g(x) = a_1 x + a_2 x^2 + ... + a_n x^n$$

with $x = w^T u$ being the weighted sum of the input. A zero'th order term was not included in the above formula, since it would have been redundant among all the units. The zero'th order term was dealt with separately and only stored in one location.

Rationals: A rational function representation was adopted which could not have zeros in the denominator. This representation used a sum of squares of polynomials, as follows:

$$g(x) = \frac{a_0 + a_1 x + ... + a_n x^n}{1 + (b_0 + b_1 x)^2 + (b_2 x + b_3 x^2)^2 + (b_4 x + b_5 x^2 + b_6 x^3 + b_7 x^4)^2 + ...}$$

This representation has the qualities that the denominator is never less than 1, and that n parameters are used to produce a denominator of degree n. If the above formula were continued the next terms in the denominator would be of degrees eight, sixteen, and thirty-two. This powers-of-two sequence was used for the following reason: of the $2(n-m)$ terms in the square of a polynomial $p = a_m x^m + ... + a_n x^n$, it is possible by manipulating $a_m ... a_n$ to determine the $n-m$ highest coefficients, with the exception that the very highest coefficient must be non-negative. Thus if we consider the coefficients of the polynomial that results from squaring and adding together the terms of the denominator of the above formula, the highest degree squared polynomial may be regarded as determining the highest half of the coefficients, the second highest degree polynomial may be regarded as determining the highest half of the rest of the coefficients, and so forth. This process cannot set all the coefficients arbitrarily; some must be non-negative.

Flexible Fourier series: The flexible Fourier series units computed

$$g(x) = \sum_{i=0}^{n} a_i \cos(b_i x + c_i)$$

where the amplitudes a_i, frequencies b_i and phases c_i were unconstrained and could assume any value.

Sigmoids: We used the standard logistic function:

$$g(x) = 1/(1 + e^{(x-\theta)})$$

3 Training Method

All the results presented here were trained with the Levenberg-Marquardt modification of the Gauss-Newton nonlinear least squares algorithm. Stochastic gradient descent was also tried at first, but on the problems where the two were compared, Levenberg-Marquardt was much superior both in convergence time and in quality of result. Levenberg-Marquardt required substantially fewer iterations than stochastic gradient descent to converge. However, it needs $O(p^2)$ space and $O(p^2 n)$ time per iteration in a network with p parameters and n input examples, as compared to $O(p)$ space and O(pn) time per epoch for stochastic gradient descent. Further details of the training method will be discussed in a longer paper.

With some data sets, a weight decay term was added to the energy function to be optimized. The added term was of the form $\lambda \sum_{i=1}^{n} w_i^2$. When weight decay was used, a range of values of λ was tried for every network trained.

Before training, all the data was normalized: each input variable was scaled so that its range was (-1,1), then scaled so that the maximum sum of squares of input variables for any example was 1. The output variable was scaled to have mean zero and mean absolute value 1. This helped the training algorithm, especially in the case of stochastic gradient descent.

4 Results

We present results of training our networks on three data sets: robot arm inverse dynamics, Boston housing data, and sunspot count prediction. The Boston and sunspot data sets are noisy, but have only mild nonlinearity. The robot arm inverse dynamics data has no noise, but a high degree of nonlinearity. Noise-free problems have low estimation error. Models for linear or mildly nonlinear problems typically have low approximation error. The robot arm inverse dynamics problem is thus a pure approximation problem, while performance on the noisy Boston and sunspots problems is limited more by estimation error than by approximation error.

Figure 1a is a graph, as those used in PPR, of the unit response function of a one-unit network trained on the Boston housing data. The x axis is a projection (a weighted sum of inputs $w^T u$) of the 13-dimensional input space onto 1 dimension, using those weights chosen by the unit in training. The y axis is the fit to data. The response function of the unit is a sum of three cosines. Figure 1b is the superposition of five graphs of the five unit response functions used in a five-unit rational function solution (RMS error less than 2%) of the robot arm inverse dynamics problem. The domain for each curve lies along a different direction in the six-dimensional input space. Four of the five fits along the projection directions are non-monotonic, and thus can be fit only poorly by a sigmoid.

Two different error measures are used in the following. The first is the RMS error, normalized so that error of 1 corresponds to no training. The second measure is the

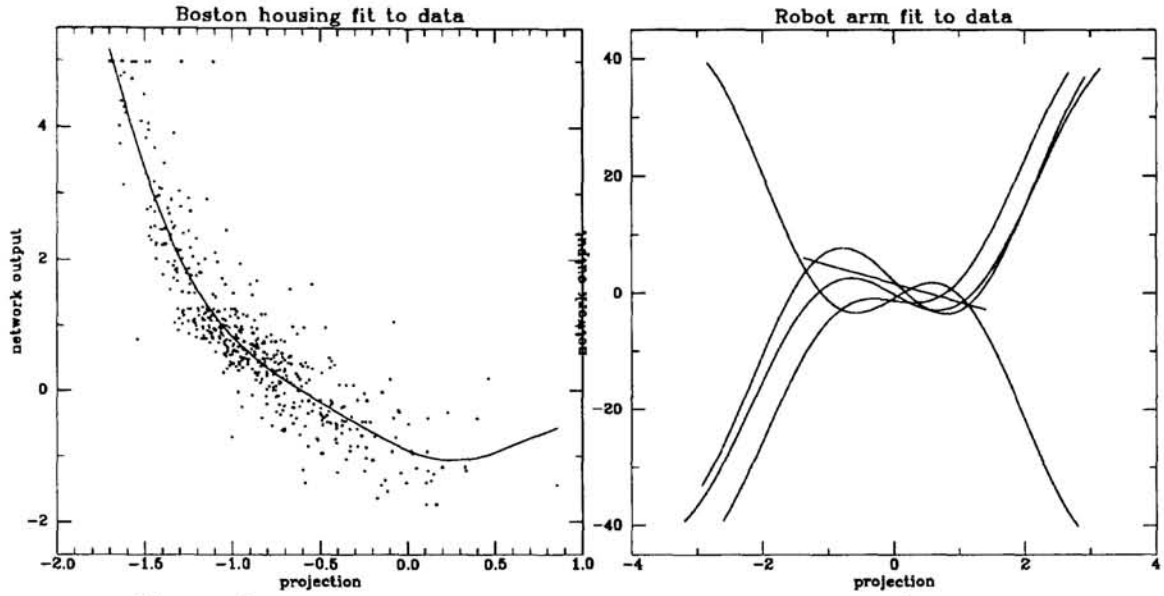

Figure 1: a b

square of the normalized RMS error, otherwise known as the fraction of explained variance. We used whichever error measure was used in earlier work on that data set.

4.1 Robot arm inverse dynamics

This problem is the determination of the torque necessary at the joints of a two-joint robot arm required to achieve a given acceleration of each segment of the arm, given each segment's velocity and position. There are six input variables to the network, and two output variables. This problem was treated as two separate estimation problems, one for the shoulder torque and one for the elbow torque. The shoulder torque was a slightly more difficult problem, for almost all networks. The 1000 points in the training set covered the input space relatively thoroughly. This, together with the fact that the problem had no noise, meant that there was little difference between training set error and test set error.

Polynomial networks of limited degree are not universal approximators, and that is quite evident on this data set; polynomial networks of low degree reached their minimum error after a few units. Figure 2a shows this. If polynomial, cosine, rational, and sigmoid networks are compared as in Figure 2b, leaving out low degree polynomials, the sigmoids have relatively high approximation error even for networks with 20 units. As shown in the following table, the complex units have more parameters each, but still get better performance with fewer parameters total.

Type	Units	Parameters	Error
degree 7 polynomial	5	65	.024
degree 6 rational	5	95	.027
2 term cosine	6	73	.020
sigmoid	10	81	.139
sigmoid	20	161	.119

Since the training set is noise-free, these errors represent pure approximation error.

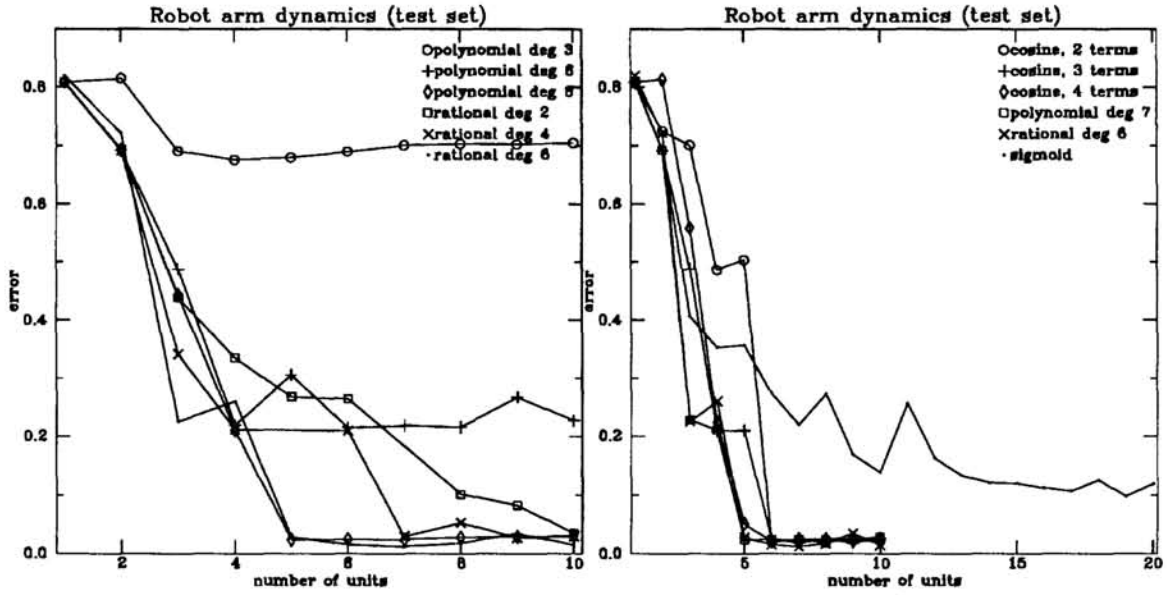

Figure 2: a b

The superior performance of the complex units on this problem is probably due to their ability to approximate non-monotonic functions.

4.2 Boston housing

The second data set is a benchmark for statistical algorithms: the prediction of Boston housing prices from 13 factors [3]. This data set contains 506 exemplars and is relatively simple; it can be approximated well with only a single unit. Networks of between one and six units were trained on this problem. Figure 3a is a graph of training set performance from networks trained on the entire data set; the error measure used was the fraction of explained variance. From this graph it is apparent

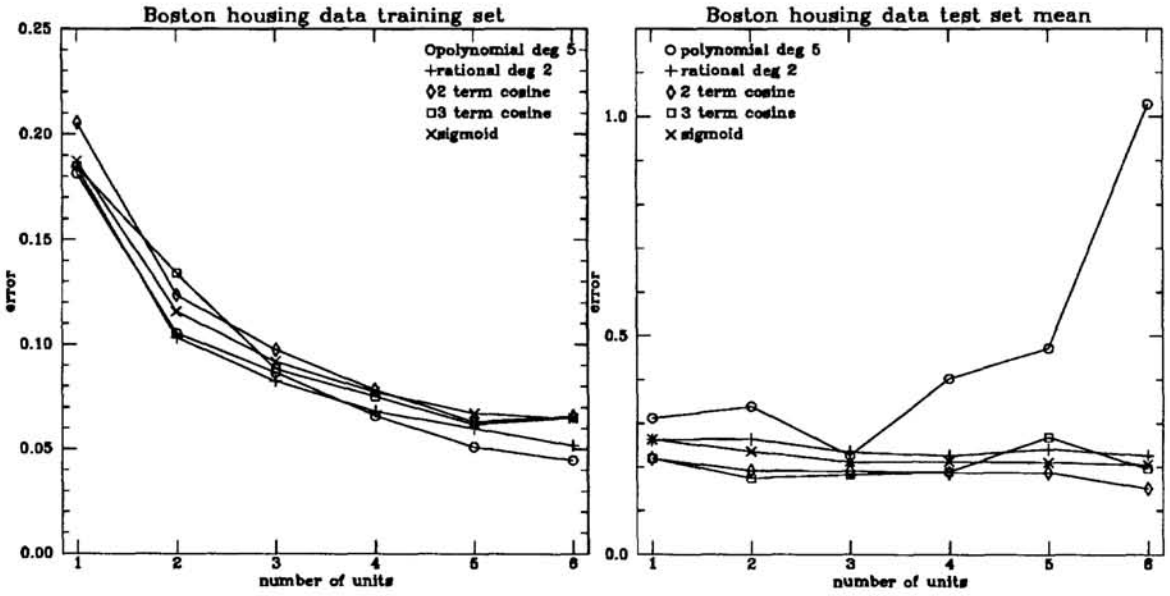

Figure 3: a b

that training set performance does not vary greatly between different types of units, though networks with more units do better.

On the test set there is a large difference. This is shown in Figure 3b. Each point on the graph is the average performance of ten networks of that type. Each network was trained using a different permutation of the data into test and training sets, the test set being 1/3 of the examples and the training set 2/3. It can be seen that the cosine nets perform the best, the sigmoid nets a close second, the rationals third, and the polynomials worst (with the error increasing quite a bit with increasing polynomial degree.)

It should be noted that the distribution of errors is far from a normal distribution, and that the training set error gives little clue as to the test set error. The following table of errors, for nine networks of four units using a degree 5 polynomial, is somewhat typical:

Set	Error								
training	0.095	0.062	0.060	0.090	0.076	0.065	0.068	0.066	0.091
test	0.085	1.677	0.171	0.197	0.143	0.546	0.250	0.158	0.395

Our speculation on the cause of these extremely high errors is that polynomial approximations do not extrapolate well; if the prediction of some data point results in a polynomial being evaluated slightly outside the region on which the polynomial was trained, the error may be extremely high. Rational functions where the numerator and denominator have equal degree have less of a problem with this, since asymptotically they are constant. However, over small intervals they can have the extrapolation characteristics of polynomials. Cosines are bounded, and so, though they may not extrapolate well if the function is not somewhat periodic, at least do not reach large values like polynomials.

4.3 Sunspots

The third problem was the prediction of the average monthly sunspot count in a given year from the values of the previous twelve years. We followed previous work in using as our error measure the fraction of variance explained, and in using as the training set the years 1700 through 1920 and as the test set the years 1921 through 1955. This was a relatively easy test set – every network of one unit which we trained (whether sigmoid, polynomial, rational, or cosine) had, in each of ten runs, a training set error between .147 and .153 and a test set error between .105 and .111. For comparison, the best test set error achieved by us or previous testers was about .085. A similar set of runs was done as those for the Boston housing data, but using at most four units; similar results were obtained. Figure 4a shows training set error and Figure 4b shows test set error on this problem.

4.4 Weight Decay

The performance of almost all networks was improved by some amount of weight decay. Figure 5 contains graphs of test set error for sigmoidal and polynomial units,

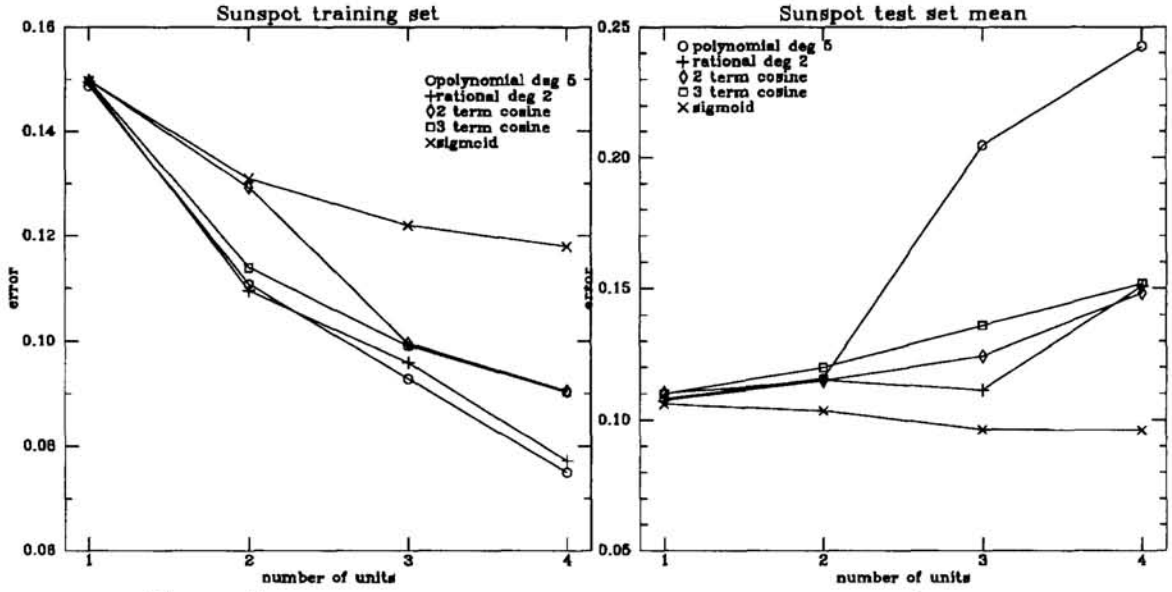

Figure 4: a b

using various values of the weight decay parameter λ. For the sigmoids, very little weight decay seems to be needed to give good results, and there is an order of magnitude range (between .001 and .01) which produces close to optimal results. For polynomials of degree 5, more weight decay seems to be necessary for good results; in fact, the highest value of weight decay is the best. Since very high values of weight decay are needed, and at those values there is little improvement over using a single unit, it may be supposed that using those values of weight decay restricts the multiple units to producing a very similar solution to the one-unit solution. Figure 6 contains the corresponding graphs for sunspots. Weight decay seems to help less here for the sigmoids, but for the polynomials, moderate amounts of weight decay produce an improvement over the one-unit solution.

Acknowledgements

The authors would like to acknowledge support from ONR grant N00014-89-J-1228, AFOSR grant 89-0478, and a fellowship from the John and Fannie Hertz Foundation. The robot arm data set was provided by Chris Atkeson.

References

[1] J. H. Friedman, W. Stuetzle, "Projection Pursuit Regression", *Journal of the American Statistical Association*, December 1981, Volume 76, Number 376, 817-823

[2] P. J. Huber, "Projection Pursuit", *The Annals of Statistics*, 1985 Vol. 13 No. 2, 435-475

[3] L. Breiman et al, *Classification and Regression Trees*, Wadsworth and Brooks, 1984, pp217-220

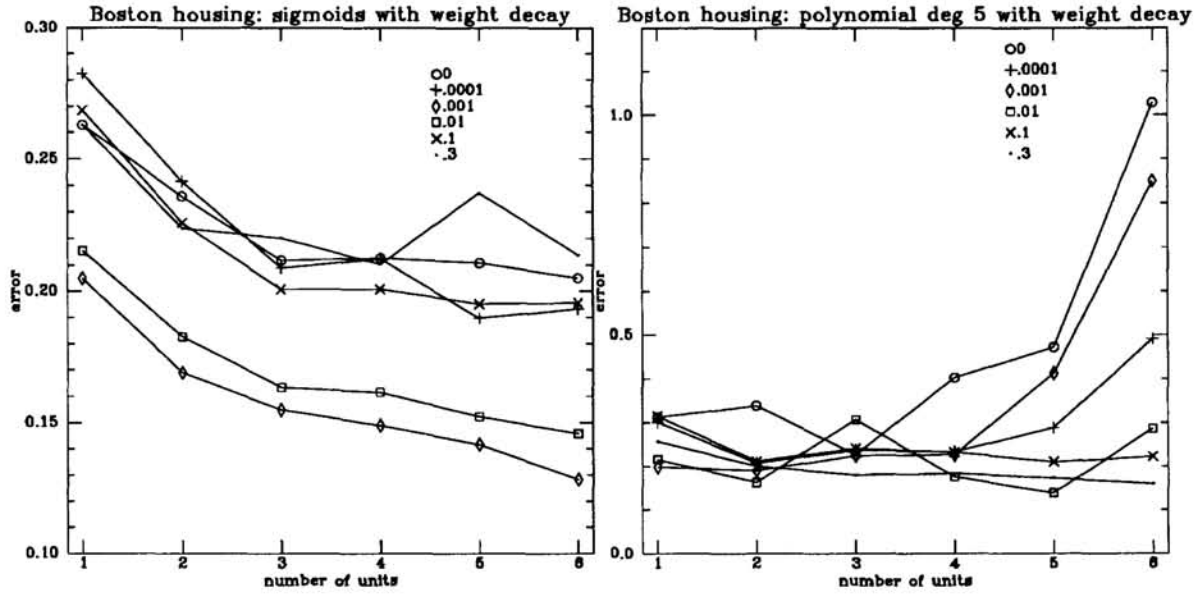

Figure 5: Boston housing test error with various amounts of weight decay

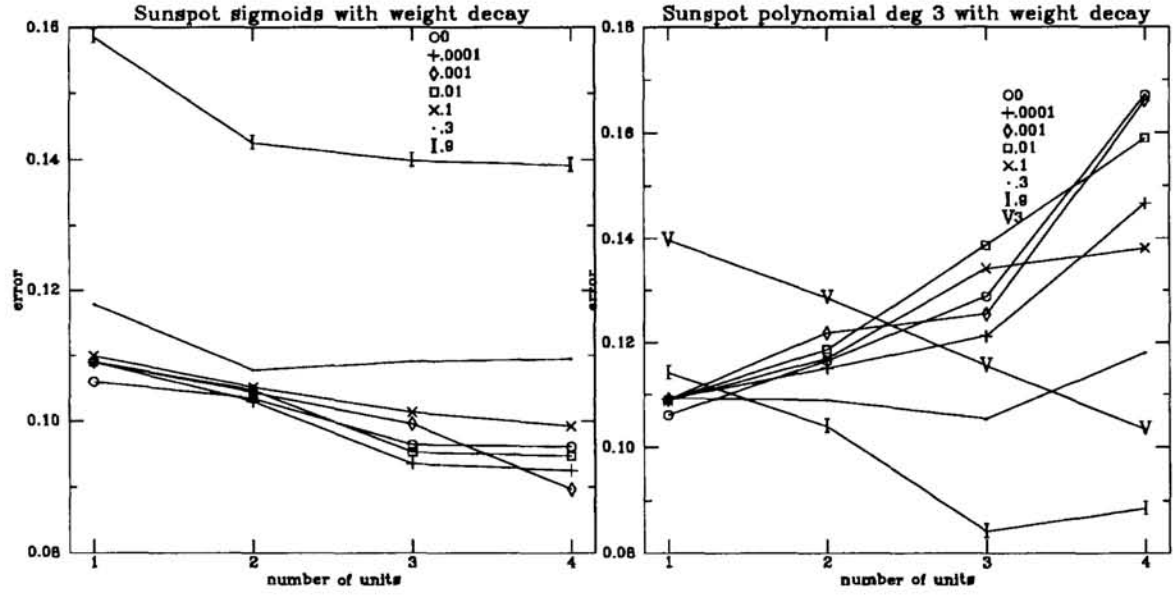

Figure 6: Sunspot test error with various amounts of weight decay

Perturbing Hebbian Rules

Peter Dayan
CNL, The Salk Institute
PO Box 85800
San Diego CA 92186-5800, USA
dayan@helmholtz.sdsc.edu

Geoffrey Goodhill
COGS
University of Sussex, Falmer
Brighton BN1 9QN, UK
geoffg@cogs.susx.ac.uk

Abstract

Recently Linsker [2] and MacKay and Miller [3, 4] have analysed Hebbian correlational rules for synaptic development in the visual system, and Miller [5, 8] has studied such rules in the case of two populations of fibres (particularly two eyes). Miller's analysis has so far assumed that each of the two populations has exactly the same correlational structure. Relaxing this constraint by considering the effects of small perturbative correlations within and between eyes permits study of the stability of the solutions. We predict circumstances in which qualitative changes are seen, including the production of binocularly rather than monocularly driven units.

1 INTRODUCTION

Linsker [2] studied how a Hebbian correlational rule could predict the development of certain receptive field structures seen in the visual system. MacKay and Miller [3, 4] pointed out that the form of this learning rule meant that it could be analysed in terms of the eigenvectors of the matrix of time-averaged presynaptic correlations. Miller [5, 8, 7] independently studied a similar correlational rule for the case of two eyes (or more generally two populations), explaining how cells develop in V1 that are ultimately responsive to only one eye, despite starting off as responsive to both. This process is again driven by the eigenvectors and eigenvalues of the developmental equation, and Miller [7] relates Linsker's model to the two population case.

Miller's analysis so far assumes that the correlations of activity within each population are identical. This special case simplifies the analysis enabling the projections from the two eyes to be separated out into sum and difference variables. In general,

one would expect the correlations to differ slightly, and for correlations between the eyes to be not exactly zero. We analyse how such perturbations affect the eigenvectors and eigenvalues of the developmental equation, and are able to explain some of the results found empirically by Miller [6].

Further details on this analysis and on the relationship between Hebbian and non-Hebbian models of the development of ocular dominance and orientation selectivity can be found in Goodhill (1991).

2 THE EQUATION

MacKay and Miller [3, 4] study Linsker's [2] developmental equation in the form:

$$\dot{w} = (Q + k_2 J)w + k_1 n$$

where $w = [w_i], i \in [1, n]$ are the weights from the units in one layer $\mathcal{R}$ to a particular unit in the next layer $\mathcal{S}$, Q is the covariance matrix of the activities of the units in layer $\mathcal{R}$, J is the matrix $J_{ij} = 1, \forall i, j$, and n is the 'DC' vector $n_i = 1, \forall i$.

The equivalent for two populations of cells is:

$$\begin{pmatrix} \dot{w}_1 \\ \dot{w}_2 \end{pmatrix} = \begin{pmatrix} Q_1 + k_2 J & Q_c + k_2 J \\ Q_c + k_2 J & Q_2 + k_2 J \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} + k_1 \begin{pmatrix} n \\ n \end{pmatrix}$$

where Q_1 gives the covariance between cells within the first population, Q_2 gives that between cells within the second, and Q_c (assumed symmetric) gives the covariance between cells in the two populations. Define Q_* as this full, two population, development matrix.

Miller studies the case in which $Q_1 = Q_2 = Q$ and Q_c is generally zero or slightly negative. Then the development of $w_1 - w_2$ (which Miller calls S^D) and $w_1 + w_2$ (S^S) separate; for $Q_c = 0$, these go like:

$$\frac{\delta S^D}{\delta t} = QS^D \text{ and } \frac{\delta S^S}{\delta t} = (Q + 2k_2 J)S^S + 2k_1 n.$$

and, up to various forms of normalisation and/or weight saturation, the patterns of dominance between the two populations are determined by the initial value and the fastest growing components of S^D. If upper and lower weight saturation limits are reached at roughly the same time (Berns, personal communication), the conventional assumption that the fastest growing eigenvectors of S^D dominate the terminal state is borne out.

The starting condition Miller adopts has $w_1 - w_2 = \epsilon' a$ and $w_1 + w_2 = b$, where ϵ' is small, and a and b are $\mathcal{O}(1)$. Weights are constrained to be positive, and saturate at some upper limit. Also, additive normalisation is applied throughout development, which affects the growth of the S^S (but not the S^D) modes. As discussed by MacKay and Miller [3, 4], this is approximately accommodated in the $k_2 J$ component.

Mackay and Miller analyse the eigendecomposition of $Q + k_2 J$ for general and radially symmetric covariance matrices Q and all values of k_2. It turns out that the eigendecomposition of Q_* for the case $Q_1 = Q_2 = Q$ and $Q_c = 0$ (that studied by Miller) is given in table form by:

E-vector	E-value	Conditions	
(x_i, x_i)	λ_i	$Qx_i = \lambda_i x_i$	$n.x_i = 0$
$(x_i, -x_i)$	λ_i	$Qx_i = \lambda_i x_i$	$n.x_i = 0$
$(y_i, -y_i)$	μ_i	$Qy_i = \mu_i y_i$	$n.y_i \neq 0$
(z_i, z_i)	ν_i	$(Q + 2k_2 J)z_i = \nu_i z_i$	$n.z_i \neq 0$

Figure 1 shows the matrix and the two key $(y, -y)$ and $(x, -x)$ eigenvectors.

The details of the decomposition of Q_* in this table are slightly obscured by degeneracy in the eigendecomposition of $Q + k_2 J$. Also, for clarity, we write (x_i, x_i) for $(x_i, x_i)^T$. A consequence of the first two rows in the table is that $(\eta x_i, \theta x_i)$ is an eigenvector for any η and θ; this becomes important later.

That the development of S^D and S^S separates can be seen in the (u, u) and $(u, -u)$ forms of the eigenvectors. In Miller's terms the onset of dominance of one of the two populations is seen in the $(u, -u)$ eigenvectors – dominance requires that μ_j for the eigenvector whose elements are all of the same sign (one such exists for Miller's Q) is larger than the μ_i and the λ_i for all the other such eigenvectors. In particular, on pages 296-300 of [6], he shows various cases for which this does and one in which it does not happen. To understand how this comes about, we can treat the latter as a perturbed version of the former.

3 PERTURBATIONS

Consider the case in which there are small correlations between the projections and/or small differences between the correlations within each projection. For instance, one of Miller's examples indicates that small within-eye anti-correlations can prevent the onset of dominance. This can be perturbatively analysed by setting $Q_1 = Q + \epsilon E_1$, $Q_2 = Q + \epsilon E_2$ and $Q_c = \epsilon E_c$. Call the resulting matrix Q^ϵ_*.

Two questions are relevant. Firstly, are the eigenvectors stable to this perturbation, *ie* are there vectors a_1 and a_2 such that $(u_1 + \epsilon a_1, u_2 + \epsilon a_2)$ is an eigenvector of Q^ϵ_* if (u_1, u_2) is an eigenvector of Q_* with eigenvalue ϕ? Secondly, how do the eigenvalues change?

One way to calculate this is to consider the equation the perturbed eigenvector must satisfy:[1]

$$Q^\epsilon_* \begin{pmatrix} u_1 + \epsilon a_1 \\ u_2 + \epsilon a_2 \end{pmatrix} = (\phi + \epsilon\psi) \begin{pmatrix} u_1 + \epsilon a_1 \\ u_2 + \epsilon a_2 \end{pmatrix}$$

and look for conditions on u_1 and u_2 and the values of a_1, a_2 and ψ by equating the $\mathcal{O}(\epsilon)$ terms. We now consider a specific example. Using the notation of the table above, $(y_i + \epsilon a_1, -y_i + \epsilon a_2)$ is an eigenvector with eigenvalue $\mu_i + \epsilon\psi_i$ if

$$\begin{aligned} (Q - \mu_i I)\, a_1 + k_2 J\, (a_1 + a_2) &= -(E_1 - E_c - \psi_i I)\, y_i, \quad \text{and} \\ (Q - \mu_i I)\, a_2 + k_2 J\, (a_1 + a_2) &= -(E_c - E_2 + \psi_i I)\, y_i. \end{aligned}$$

Subtracting these two implies that

$$(Q - \mu_i I)\, (a_1 - a_2) = -(E_1 - 2E_c + E_2 - 2\psi_i I)\, y_i.$$

[1] This is a standard method for such linear systems, *eg* in quantum mechanics.

However, $\mathbf{y}_i^T (Q - \mu_i I) = 0$, since Q is symmetric and $\mathbf{y}_i$ is an eigenvector with eigenvalue μ_i, so multiplying on the left by $\mathbf{y}_i^T$, we require that

$$2\psi_i \mathbf{y}_i^T \mathbf{y}_i = \mathbf{y}_i^T (E_1 - 2E_c + E_2)\, \mathbf{y}_i$$

which sets the value of ψ_i. Therefore $(\mathbf{y}_i, -\mathbf{y}_i)$ *is* stable in the required manner.

Similarly $(\mathbf{z}_i, \mathbf{z}_i)$ is stable too, with an equivalent perturbation to its eigenvalue. However the pair $(\mathbf{x}_i, \mathbf{x}_i)$ and $(\mathbf{x}_i, -\mathbf{x}_i)$ are not stable – the degeneracy from their having the same eigenvalue is broken, and two specific eigenvectors, $(\alpha_i \mathbf{x}_i, \beta_i \mathbf{x}_i)$ and $(-\beta_i \mathbf{x}_i, \alpha_i \mathbf{x}_i)$ are stable, for particular values α_i and β_i. This means that to first order, $\mathbf{S}^D$ and $\mathbf{S}^S$ no longer separate, and the full, two-population, matrix must be solved.

To model Miller's results, call $Q_*^{\epsilon,m}$ the special case of Q_*^{ϵ} for which $E_1 = E_2 = E$ and $E_c = 0$. Also, assume that the $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ are normalised, let $e_1(\mathbf{u}) = \mathbf{u}^T E_1 \mathbf{u}$, *etc*, and define $\gamma(\mathbf{u}) = (e_1(\mathbf{u}) - e_2(\mathbf{u}))/2e_c(\mathbf{u})$, for $e_c(\mathbf{u}) \neq 0$, and $\gamma_i = \gamma(\mathbf{x}_i)$. Then we have

$$\beta_i/\alpha_i = -\gamma_i \pm \sqrt{1 + \gamma_i^2} \tag{1}$$

and the eigenvalues are:

E-vector	Eigenvalue for case:		
	Q_*	$Q_*^{\epsilon,m}$	Q_*^{ϵ}
$(\alpha_i \mathbf{x}_i, \beta_i \mathbf{x}_i)$	λ_i	$\lambda_i + \epsilon e_1(\mathbf{x}_i)$	$\lambda_i + \epsilon[e_1(\mathbf{x}_i) + e_2(\mathbf{x}_i) + \Xi_i]/2$
$(-\beta_i \mathbf{x}_i, \alpha_i \mathbf{x}_i)$	λ_i	$\lambda_i + \epsilon e_1(\mathbf{x}_i)$	$\lambda_i - \epsilon[e_1(\mathbf{x}_i) + e_2(\mathbf{x}_i) + \Xi_i]/2$
$(\mathbf{y}_i, -\mathbf{y}_i)$	μ_i	$\mu_i + \epsilon e_1(\mathbf{y}_i)$	$\mu_i + \epsilon[e_1(\mathbf{y}_i) + e_2(\mathbf{y}_i) - 2e_c(\mathbf{y}_i)]/2$
$(\mathbf{z}_i, \mathbf{z}_i)$	ν_i	$\nu_i + \epsilon e_1(\mathbf{z}_i)$	$\nu_i + \epsilon[e_1(\mathbf{z}_i) + e_2(\mathbf{z}_i) + 2e_c(\mathbf{z}_i)]/2$

where $\Xi_i = \sqrt{[e_1(\mathbf{x}_i) - e_2(\mathbf{x}_i)]^2 + 4e_c(\mathbf{x}_i)^2}$. For the case Miller treats, since $E_1 = E_2$, the degeneracy in the original solution is preserved, *ie* the perturbed versions of $(\mathbf{x}_i, \mathbf{x}_i)$ and $(\mathbf{x}_i, -\mathbf{x}_i)$ have the same eigenvalues. Therefore the $\mathbf{S}^D$ and $\mathbf{S}^S$ modes still separate.

This perturbed eigendecomposition suffices to show how small additional correlations affect the solutions. We will give three examples. The case mentioned above on page 299 of [6], shows how small same-eye anti-correlations within the radius of the arbor function cause a particular $(\mathbf{y}_i, -\mathbf{y}_i)$ eigenvector (i.e. one for which all the components of $\mathbf{y}_i$ have the same sign) to change from growing faster than a $(\mathbf{x}_i, -\mathbf{x}_i)$ (for which some components of $\mathbf{x}_i$ are positive and some negative to ensure that $\mathbf{n}.\mathbf{x}_i = 0$) to growing slower than it, converting a monocular solution to a binocular one.

In our terms, this is the $Q_*^{\epsilon,m}$ case, with E_1 a negative matrix. Given the conditions on signs of their components, $e_1(\mathbf{y}_i)$ is more negative than $e_1(\mathbf{x}_i)$, and so the eigenvalue for the perturbed $(\mathbf{y}_i, -\mathbf{y}_i)$ would be expected to decrease more than that for the perturbed $(\mathbf{x}_i, -\mathbf{x}_i)$. This is exactly what is found. Different binocular eigensolutions are affected by different amounts, and it is typically a delicate issue as to which will ultimately prevail. Figure 2 shows a sample perturbed matrix for which dominance will not develop. If the change in the correlations is large ($\mathcal{O}(1)$), then the eigenfunctions can change shape (eg 1s becomes 2s in the notation of [4]). We do not address this here, since we are considering only changes of $\mathcal{O}(\epsilon)$.

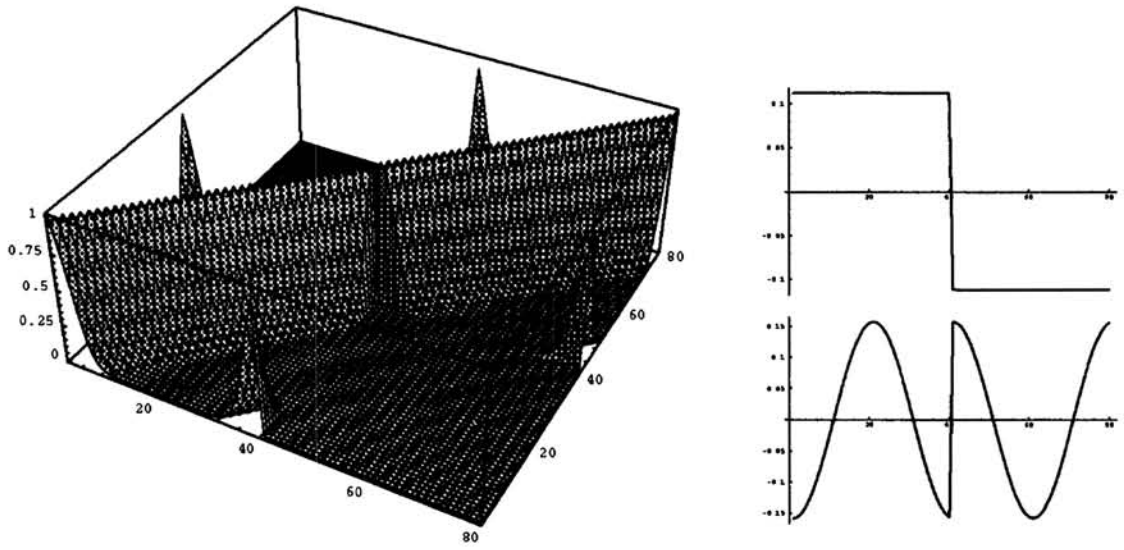

Figure 1: Unperturbed two-eye correlation matrix and $(y, -y)$, $(x, -x)$ eigenvectors. Eigenvalues are 7.1 and 6.4 respectively.

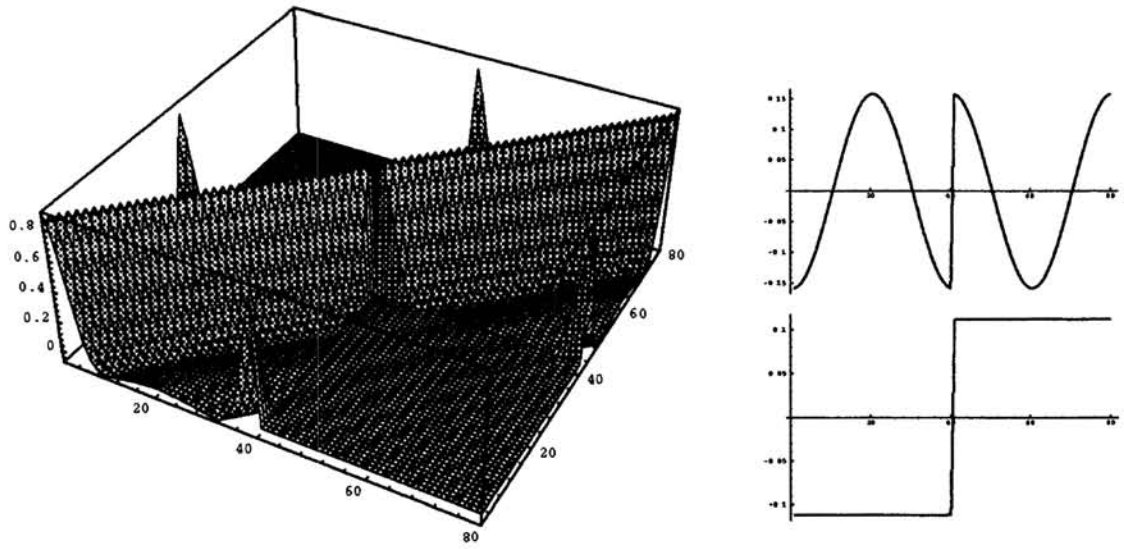

Figure 2: Same-eye anti-correlation matrix and eigenvectors. $(y, -y)$, $(x, -x)$ eigenvalues are 4.8 and 5.4 respectively, and so the order has swapped.

Positive opposite-eye correlations can have exactly the same effect. This time $e_c(\mathbf{y}_i)$ is greater than $e_c(\mathbf{x}_i)$, and so, again, the eigenvalue for the perturbed $(\mathbf{y}_i, -\mathbf{y}_i)$ would be expected to decrease more than that for the perturbed $(\mathbf{x}_i, -\mathbf{x}_i)$. Figure 3 shows an example which is infelicitous for dominance.

The third case is for general perturbations in Q_*^ϵ. Now the mere signs of the components of the eigenvectors are not enough to predict which will be affected more. Figure 4 gives an example for which ocular dominance will still occur. Note that the $(\mathbf{x}_i, -\mathbf{x}_i)$ eigenvector is no longer stable, and has been replaced by one of the form $(\alpha_i \mathbf{x}_i, \beta_i, \mathbf{x}_i)$.

If general perturbations of the same order of magnitude as the difference between w_1 and w_2 (*ie* $\epsilon' \simeq \epsilon$) are applied, the α_i and β_i terms complicate Miller's S^D analysis to first order. Let $w_1(0) - w_2(0) = \epsilon \mathbf{a}$ and apply Q_*^ϵ as an iteration matrix. $w_1(n) - w_2(n)$, the difference between the projections after n iterations has no $\mathcal{O}(1)$ component, but two sets of $\mathcal{O}(\epsilon)$ components; $\{2\mu_i^n (\mathbf{a}.\mathbf{y}_i)\, \mathbf{y}_i\}$, and

$$\begin{aligned}\{ \quad & \lambda_i^n[1 + \epsilon(\Upsilon_i + \Xi_i)/2\lambda_i]^n (\alpha_i \mathbf{x}_i.w_1(0) + \beta_i \mathbf{x}_i.w_2(0))(\alpha_i - \beta_i)\mathbf{x}_i - \\ & \lambda_i^n[1 + \epsilon(\Upsilon_i - \Xi_i)/2\lambda_i]^n (\alpha_i \mathbf{x}_i.w_2(0) - \beta_i \mathbf{x}_i.w_1(0))(\alpha_i + \beta_i)\mathbf{x}_i \quad \}\end{aligned}$$

where $\Upsilon_i = e_1(\mathbf{x}_i) + e_2(\mathbf{x}_i)$. Collecting the terms in this expression, and using equation 1, we derive

$$\left\{ \lambda_i^n \left[(\alpha_i^2 + \beta_i^2)\mathbf{x}_i.\mathbf{a} + 2n\frac{\Xi_i}{\lambda_i}\gamma_i \alpha_i \beta_i \mathbf{x}_i.\mathbf{b} \right] \mathbf{x}_i \right\}$$

where $\mathbf{b} = w_1(0) + w_2(0)$. The second part of this expression depends on n, and is substantial because $w_1(0) + w_2(0)$ is $\mathcal{O}(1)$. Such a term does not appear in the unperturbed system, and can bias the competition between the $\mathbf{y}_i$ and the $\mathbf{x}_i$ eigenvectors, in particular towards the binocular solutions. Again, its precise effects will be sensitive to the unperturbed eigenvalues.

4 CONCLUSIONS

Perturbation analysis applied to simple Hebbian correlational learning rules reveals the following:

- Introducing small anti-correlations within each eye causes a tendency toward binocularity. This agrees with the results of Miller.
- Introducing small positive correlations between the eyes (as will inevitably occur once they experience a natural environment) has the same effect.
- The overall eigensolution is not stable to small perturbations that make the correlational structure of the two eyes unequal. This also produces interesting effects on the growth rates of the eigenvectors concerned, given the initial conditions of approximately equivalent projections from both eyes.

Acknowledgements

We are very grateful to Ken Miller for helpful discussions, and to Christopher Longuet-Higgins for pointing us in the direction of perturbation analysis. Support

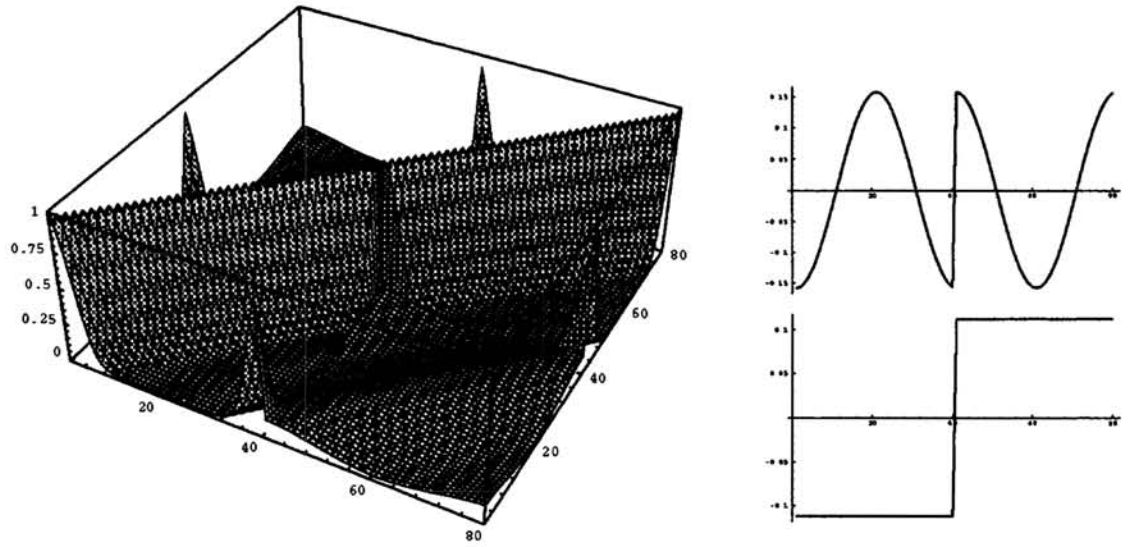

Figure 3: Opposite-eye positive correlation matrix and eigenvectors. Eigenvalues of $(y, -y)$, $(x, -x)$ are 4.8 and 5.4, so ocular dominance is again inhibited.

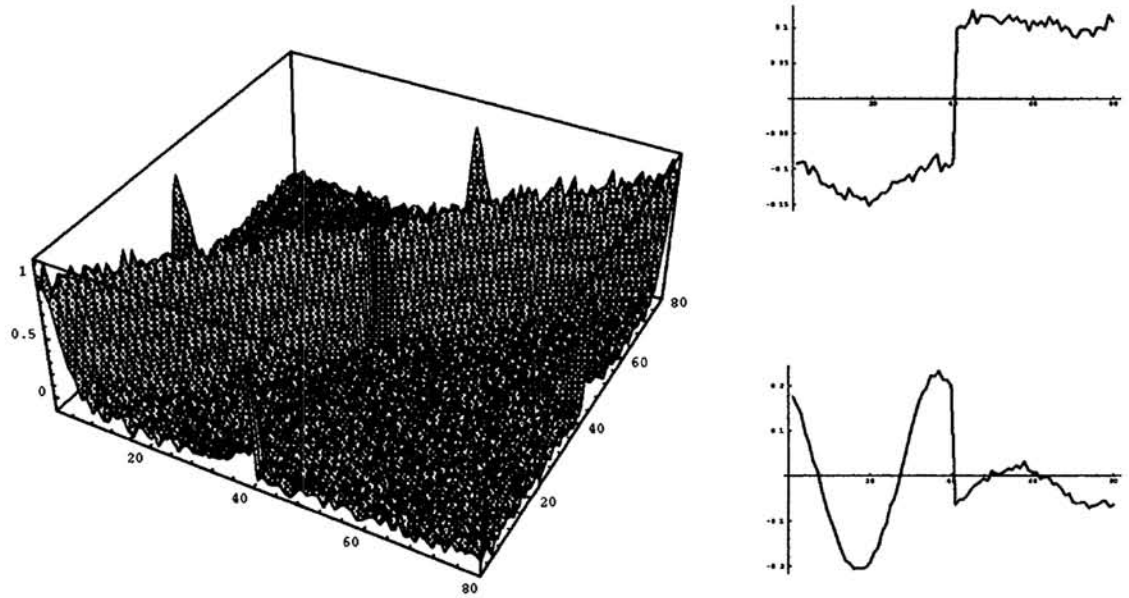

Figure 4: The effect of random perturbations to the matrix. Although the order is restored (eigenvalues are 7.1 and 6.4), note the $(\alpha x, \beta x)$ eigenvector.

was from the SERC and a Nuffield Foundation Science travel grant to GG. GG is grateful to David Willshaw and the Centre for Cognitive Science for their hospitality. GG's current address is The Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, and correspondence should be directed to him there.

References

[1] Goodhill, GJ (1991). *Correlations, Competition and Optimality: Modelling the Development of Topography and Ocular Dominance.* PhD Thesis, Sussex University.

[2] Linsker, R (1986). From basic network principles to neural architecture (series). *Proc. Nat. Acad. Sci., USA*, **83**, pp 7508-7512, 8390-8394, 8779-8783.

[3] MacKay, DJC & Miller, KD (1990). Analysis of Linsker's simulations of Hebbian rules. *Neural Computation*, **2**, pp 169-182.

[4] MacKay, DJC & Miller, KD (1990). Analysis of Linsker's application of Hebbian rules to linear networks. *Network*, **1**, pp 257-297.

[5] Miller, KD (1989). *Correlation-based Mechanisms in Visual Cortex: Theoretical and Empirical Studies.* PhD Thesis, Stanford University Medical School.

[6] Miller, KD (1990). Correlation-based mechanisms of neural development. In MA Gluck & DE Rumelhart, editors, *Neuroscience and Connectionist Theory.* Hillsborough, NJ: Lawrence Erlbaum.

[7] Miller, KD (1990). Derivation of linear Hebbian equations from a nonlinear Hebbian model of synaptic plasticity. *Neural Computation*, **2**, pp 321-333.

[8] Miller, KD, Keller, JB & Stryker, MP (1989). Ocular dominance column development: Analysis and simulation. *Science*, **245**, pp 605-615.