NISTIR 5696

The Effect of Training Dynamics on Neural Network Performance

Charles L. Wilson
James L. Blue
Omid M. Omidvar

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Advanced Systems Division
Gaithersburg, MD 20899

August 1995

The Effect of Training Dynamics on Neural Network Performance

Charles L. Wilson, Advanced Systems Division
James L. Blue, Applied and Computational Mathematics Division
National Institute of Standards and Technology, Gaithersburg, MD 20899
and Omid M. Omidvar, Computer Science Department
University of the District of Columbia

Abstract

In this paper, analysis of a simple model of recurrent network dynamics is used to gain qualitative insights into the training dynamics of multilayer perceptrons (MLPs). These insights allow the training methods used for MLPs to be modified to significantly improve network performance. In previous work [1], the Probabilistic Neural Network (PNN) [2], was shown to provide better zero-reject error performance on character and fingerprint classification problems than Radial Basis Function and MLP-based neural network methods. We will show that performance equal to or better than PNN can be achieved with a single three-layer MLP by making fundamental changes in the network optimization strategy. These changes are: 1) Neuron activation functions are used which reduce the probability of singular Jacobians; 2) Successive regularization is used to constrain the volume of the minimized weight space; 3) Boltzmann pruning [3] is used to constrain the dimension of the weight space; and 4) Prior class probabilities are used to normalize all error calculations so that statistically significant samples of rare but important classes can be included without distorting the error surface. All four of these changes are made in the inner loop of a conjugate gradient optimization iteration [4] and are intended to simplify the training dynamics of the optimization. On handprinted digits and fingerprint classification problems these modifications improve error-reject performance by factors between 2 and 4 and reduce network size by 40% to 60%.

1 Introduction

One of the basic difficulties in recognizing images using pattern recognition methods is that the set of patterns of interest is a small subset of all the possible patterns that can be represented in the image space. As an example, if characters in an OCR system are represented by 32 by 32 pixel binary images only a small fraction of the 2^{1024} possible images are characters. If the features used to represent these characters form a complete representation of the image down to some specified resolution, such as a discrete cosine transform, the feature set is a dense compact vector space representation of the image. The image training set is a fractal object in this feature space. This statement of the recognition problem poses it as the inverse of the fractal image compression problem [5] and shows that the neural network recognition problem for images retains the "geometry of nature" [6] seen in these images. This paper explores methods which allow neural networks to accurately classify image based objects which are sampled from fractal objects in the compact feature space used for recognition.

In previous work on character and fingerprint classification [1], PNN networks were shown to be superior to MLP networks in classification accuracy. In later work, [7], combinations of PNN and MLP networks were shown to be equal to PNN in accuracy and to have superior reject-accuracy performance. These results were achieved by using 45 PNN networks to make binary decisions between digit pairs and combining the 45 outputs with a single MLP. This procedure is much more expensive than conventional MLP training of a single network and uses much more memory space.

When the results of the binary decision network [7] were analyzed for digit recognition it was found that the feature space used in the recognition process had a topological structure which locally had an intrinsic dimension [8] of 10.5 but global Karhunen-Loeve (K-L) transform dimension of approximately 100. The number of features needed to make binary decisions machines discriminate between digits was larger than the intrinsic dimensionality. For binary decision machines, typical feature set sizes were 20 to 28 but never approached the number of features required by the global problem for MLPs, 48 to 52. Similar tests showed a comparable structure in the fingerprint feature data. This explains the difficulty of these problems; the MLP is being used to approximate a complex fractal object, the set of decision surfaces, which has a typical local dimension of 10 embedded in a space of dimension 100. Since the domain of each prototype in the PNN network is local, PNN can more easily approximate surfaces with this topology. Figure 1 shows a typical PNN decision surface and figure 2 shows a typical MLP decision surface. See figure 8 of [1] for additional examples of this type of local structure in PNN-based recognition.

The local nature of PNN decision surfaces also explains why MLPs have better error-reject performance. In [9], it was shown that the error-reject curve is most rapidly decreasing when binary choices are made between classes. The decision surfaces in figure 1 are such that, as the radius of a test region expands, multiple class regions are intersected. This will decrease the slope of the reject-accuracy curve. Simpler class decision surfaces result in better reject-accuracy performance, so that the shape of the reject curve can be used to assess the complexity of decision surfaces.

Neural networks have been proven to be a general nonlinear function approximator [10] so, in theory, they should be capable of approximating complex decision surfaces. An interesting, if somewhat simplified, conjecture is that since we are trying to learn a complex surface the complexity of the training process might provide some insight into the learning problem. To obtain these insights we study the dynamics of a simple weakly nonlinear recurrent network model. The model contains both linear and second-order rate-dependent terms which couple all of the nodal voltages. This model is solved in closed form but still shows several interesting ways in which the dynamics of feedback signals will influence network behavior.

The MLP networks used in practical recognition problems are too complex to be subject to direct analysis but the quantitative insights provided by the simple model can be used to develop better training methods.

In this paper we will show that four modifications to the conjugate gradient method discussed in [4] will allow a three-layer MLP to approximate the required decision surface with zero-reject error similar to PNN and k-nearest-neighbor (KNN) methods, and error-reject performance better than the best binary decision method discussed in [7], indicating that the resulting decision surfaces are both the most accurate and simplest approximations to the character and fingerprint classification problem yet found.

In section 2 a simple dynamical neural network model is presented which is complex enough to show many interesting dynamic effects but simple enough to be solved in closed form. In section 3 the relationship between structural stability and optimization is used to develop qualitative insights into network training. In section 4 we will discuss methods which the network dynamics suggest for changes in training. In section 5 we will discuss the changes in training method used to improve network accuracy while simplifying network structure. In section 6 we will discuss the results of these changes on classification of handprinted digits and fingerprints.

2 A Simple Dynamical Network Model

To understand the dynamics behind training, it is helpful to analyze a neural network model that has feedback between the nodes, as the MLP has during training, but is simple enough to be solved in closed form.

Consider a number of neuron nodes with nodal voltages, $\mathbf{U}$, which can be written either in matrix form:

$$\mathbf{U}_d = \begin{pmatrix} u_1(t) & \cdots & 0 \\ & \vdots & \\ 0 & 0 & u_n(t) \end{pmatrix} \tag{1}$$

or as a vector:

$$\mathbf{U} = \begin{pmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{pmatrix} \tag{2}$$

as is required by the linear or quadratic order of interaction.

These neurons are linear and the network is fully recurrent so that the interaction of the neurons is described by a non-stationary set of first order couplings:

$$\mathbf{G} = \begin{pmatrix} g_{1,1}(t) & \cdots & g_{1,n}(t) \\ & \vdots & \\ g_{n,1}(t) & \cdots & g_{n,n}(t) \end{pmatrix} \tag{3}$$

and a set of higher order couplings:

$$\mathbf{F} = \begin{pmatrix} f_{1,1}(t) & \cdots & f_{1,n}(t) \\ & \vdots & \\ f_{n,1}(t) & \cdots & f_{n,n}(t) \end{pmatrix} \tag{4}$$

The system dynamics is described by:

$$\frac{d\mathbf{U}}{dt} = (\mathbf{U}_d \mathbf{F} + \mathbf{G})\,\mathbf{U} \tag{5}$$

which allows the network to have steady states $\mathbf{U}_s$ given by:

$$\mathbf{U}_s = -\mathbf{F}^{-1}\,\mathbf{G} \tag{6}$$

The solution of (5) was postulated as (7) and checked using a symbolic mathematics program to validate the solution terms. Terms generated in this checking process are given in detail in the Appendix. See [11] page 99 or [12] page 60 for similar linear problems which were used as clues to the proposed solution. The solution derived and checked in this way is:

$$\mathbf{U} = (\mathbf{I} - \textstyle\int \mathbf{F} \exp(\int \mathbf{G}\, dt)\, dt)^{-1} \exp(\int \mathbf{G} dt))\, \mathbf{U}(\mathbf{0}) \tag{7}$$

where the matrix exponential is defined by:

$$\exp(\mathbf{A}) = \mathbf{I} + \mathbf{A} + \mathbf{A}^2/2! + \mathbf{A}^3/3! + \cdots \tag{8}$$

and where $\mathbf{A}$ can be expanded in terms of its eigenvalues and eigenvectors as:

$$\mathbf{A} = \mathbf{S}^{\mathrm{H}}\, \Lambda_A\, \mathbf{S} \tag{9}$$

with $\mathbf{S}$ unitary so that $\mathbf{S}^{\mathrm{H}} = \mathbf{S}^{-1}$, with Λ the matrix of eigenvalues of $\mathbf{A}$ taking the form:

$$\Lambda_A = \begin{pmatrix} \lambda_1(t) & * & * \\ & \vdots & * \\ 0 & 0 & \lambda_n(t) \end{pmatrix} \tag{10}$$

In general $\mathbf{G}$ will have complex eigenvalues; only in the symmetric case are all eigenvalues real and the upper triangle of $Lambda_A$ zero. This symmetry requirement makes the construction of stable associative memories with symmetric weights very difficult. These issues are discussed in [13] along with many other aspects of the general network stability problem. Any iteration carried out with finite precision on $\mathbf{G}$ will result in loss of symmetry and create oscillations in the dynamic system. These oscillations may decay in a stable system but intermediate oscillatory components will still exist.

The inclusion of driving factors which are assumed to be independent of time is relatively straightforward. The $\mathbf{G}$ matrix is augmented from a dimension of n to $n + q$ where q is the number of system independent driving factors. The solution to the corresponding augmented part of $\mathbf{U}_d$ matrix is just the augmented $\mathbf{U}(0)$ vector. This results in a stable driving term from equation (5) and a solution to this part of the equation of the form: $\mathbf{U}_d = \mathbf{U}(0)_d$. The appendix contains more details.

Why can't we just integrate equation (6) and use the answer for our model? First, for n nodes there are $2n^2$ functions which must be evaluated and integrated to fully expand equation (12). Second, equation (7) is a generalization of the dynamic system which generates the Lorentz attractor [14], which is the source of the "butterfly wing effect" in weather forecasting. This system has no attainable stable state and has very high sensitivity to initial conditions. The theory of the solution to systems of this kinds is treated in [15, 14, 12]. The simplified answer is that small changes in the values of the initial conditions or in the integration of the coefficient matrices will result in large changes in the functional form of equation (6) when the matrix exponential and inverse are expanded.

For values $\mathbf{F}$ and $\mathbf{G}$ that are locally stationary, as during a training iteration, change in these terms can be neglected and we write the solution as:

$$\mathbf{U}(t) = \mathbf{A}(t)\, \mathbf{U}(0) \tag{11}$$

where for $\mathbf{G}$ symmetric:

$$a_{ij} = \frac{|\mathrm{H}_{ij}|^{\star} \sum_{k=1}^{n} s_{ki} e^{\lambda_k t} s_{kj}}{|\mathbf{H}|}. \tag{12}$$

where $|\mathrm{H}_{ij}|^*$ is the matrix of ijth cofactors of $\mathbf{H}$, $|\mathbf{H}|$ is the determinant of $\mathbf{H}$, s_{ij} are the elements of the similarity transform of $\mathbf{G}$ and λ_i are its eigenvalues as given by equation (8) assuming that $\mathbf{G}$ is Hermitian, and the elements of $\mathbf{H}$ are given by:

$$\mathbf{H} = \mathbf{I} - (h_{ij}) = (\delta_{ij} - h_{ij}) \tag{13}$$

where:

$$h_{ij} = \sum_{k=1}^{n} f_{ik} \sum_{l=1}^{n} \frac{s_{lj} s_{lk}\, e^{\lambda_l t}}{\lambda_l} \tag{14}$$

For the case of $n = 3$ appropriate terms are given in the appendix.

The full solution of equation (6) can be expanded using equations (11)–(14), as shown in the appendix. These equations are much less complex than the approach used in our training method but show that complex dynamics will evolve even in a network that has only a quadratic nonlinearity. Even for a symmetric network, obtaining the local linear solution involves calculating matrices with $\mathrm{O}(n^2)$ terms for an nth order system. Equations (11)–(14) involve calculation of $\mathrm{O}(n^8)$ terms for an nth order system. In addition, each term in (7) is a product of the form $s_{ki}e^{\lambda_k t}s_{kj}$, where each term in the non-linear solution is a product at least as complex as:

$$f_{ik} \frac{s_{lj} s_{lk}\, e^{\lambda_l t}}{\lambda_l} s_{pq} e^{\lambda_p t} s_{pq}. \tag{15}$$

Each term involves one non-linear factor, components of four eigenvectors (which in general are complex numbers) and the product of two exponential terms. This increase in complexity is a direct consequence of the $\mathbf{F}$ term in equation (6) being non-zero.

The center manifold theorem, given more concisely in [12] p. 115 and [14] p. 127, states that the flow of solution to nonlinear system can be divided into three parts based on the eigenvalue spectrum of the Jacobian of the right hand side of the equation. These three parts are the stable manifold associated with eigenvalues with negative real part, the unstable manifold associated with eigenvalues with positive real part, and the center manifold associated with eigenvalues with zero real part. The stable and unstable manifolds are unique but the center manifold need not be. The center manifold is tangent to the stable and unstable manifolds. If any of the eigenvalues in equation (15) has zero real part, the center manifold theorem must be used to transform equation (5) into stable, center, and unstable manifold parts, and the solution expansion of equation (7) must be revised accordingly [16]. In numerical calculations, rounding error and lack of precision in the input data can cause some solution components of the stable and unstable solutions to approach the center manifold within the rounding error of the calculation.

The complexity of even small models, $n = 3$ with 9 terms in the primary matrix, each of which has 729 terms similar in form to equation (15) (see Appendix), exceeds the level of complexity which can be locally analyzed by direct linearization about equilibrium points to study weight stability during training. We could return to equation (5) and build a numerical model but this would cause all of the noise present in the training data to be present in a system of equations which would couple this noise directly to the sensitive initial conditions and result in various numerical significance problems. The resulting terms are multiplied to form n^8 terms which are summed to form the solution. The noise seen in the resulting solution is not a numerical artifact but **an inherent part of the system.**

The situation we face in analyzing even this over-simplified model is one in which the mechanics of the process are clear and are soluble in the closed form given by equation

(11)–(14), but no direct comparison with real training data is feasible because the expanded nonlinear solution is too complex to allow the components to be individually analyzed.

3 Dynamics of Optimization and Structural Stability

The dynamics of neural network training effects the network on two time scales. The time scales are associated with the calculation of feedback signals within an iteration and a longer time scale associated with the sequence of optimization iterations. The dynamics can be used to analyze the sequence of dynamic systems generated as the optimization iteration proceeds. In any given iteration the structure of the network is fixed and the dynamics involves the application of the feature vectors as driving inputs. Since the feature vectors provide a forcing function this is not an equilibrium but a problem where the training data drives the network, with error correcting feedback, to a steady state. This is the short time scale.

The second time scale is the time scale of the optimization iteration process. At each time step of the iteration the structure of the network is changed by weight updates and Boltzmann pruning. On this time scale structural stability is dominant. The goal of the iteration process is to approach a steady state where the effect of network changes minimizes the feedback error over the training set and thus a steady state is achieved.

The link between the study of time dependent network performance and nonlinear network optimization is provided by the analysis of structural stability. The mathematical correspondence between the structural stability analysis and the linearization used in optimization is a result of both methods being based on a Taylor series expansion of the nonlinear system about the point of interest. This expansion yields a local linear approximation to the nonlinear system. The usual analysis of structural stability [12] for:

$$\frac{d\mathbf{U}}{dt} = f(\mathbf{U}, \mathbf{W}) \tag{16}$$

is performed by a local analysis of the eigenspectrum of $Df(\mathbf{U_0}, \mathbf{W}_0)$ at the point $\mathbf{U}_0, \mathbf{W}_0$. The optimization by gradient-based methods of the nonlinear network is performed by linearization of the nonlinear system:

$$\frac{d(\mathbf{U} - \mathbf{U_t})}{dt} = f(\mathbf{U}, \mathbf{W}) - f(\mathbf{U_t}, \mathbf{W}) \tag{17}$$

where the $\mathbf{U_t}$ are a set of training examples. The expansion of terms in the right hand side of (16) is used to compute the dynamic change in the error over the training set. The optimization is performed over the sum of all training examples using the second order term in the expansion, the Hessian matrix, because near the desired minimum point the first order, Jacobian, term approaches zero. Similar additional terms are also required for nonlinear stability analysis when the Jacobian is too small. Both the analysis of local dynamic stability and the analysis of structural stability are based on the expansion of the system of equations in a Taylor series. When dynamic stability is involved the series is used to provide linear approximations to the right hand side of equations such as (5). When structural stability is involved the solution is expanded in terms of the critical parameters. For equation (7) this requires expansion of (7) in terms of the $\mathbf{F}$ and $\mathbf{G}$ matrix elements. Details of this type of analysis are given in [14], chapter 5. This yields insights which can be applied to the structural stability of the learning process.

We have found that three of the changes in training dynamics are directly related to insights obtained from dynamics. Regularization has been shown to act as a smoother when the neural network training process is treated as an approximation problem [17]. We can qualitatively evaluate the effect of a quadratic regularization term on the dynamics since such a term acts to constrain the optimization to a region near the origin. In our highly fractal feature spaces [7], the smaller the dynamic volume the more effective the features will be in spanning the training space, as is demonstrated from the approximation point of view in [17]. The local feature dimension of features is about 11. The large networks used here have 96 K-L features. The undersampling increases roughly as the power of the difference in the global and the local dimension. For the OCR problem this is $96 - 11$ so that the undersampling increases as the 85th power of feature space radius. This indicates that additional training examples which expand the dynamic volume but do not alter the local or global dimensionality of the feature set will not produce any significant improvement in network performance.

The second insight we can obtain from dynamics is that as we expand the Jacobian of (17), any nonlinear terms which are very small will vanish when compared to the noise in real training data. The calculation of the K-L transform involves, for the OCR problem, calculation of a mean image value for every location in the image. Since the values used to calculate the mean are binary, the mean for an N example training set can only take on N values and is only known to $\log_2 N$ bits. Even with perfect training data the K-L features have this finite precision. Imperfect or ambiguous characters add additional uncertainty that results in uncertainty throughout the dynamic calculation. Small signal changes in network dynamics caused by training data noise are indistinguishable from small changes caused by real but small dynamic terms. For MLPs with sigmoidal activation functions, these small terms occur for both large positive and large negative values of the sum of the weighted inputs. For the data used here the K-L transform normalizes the input signals to a uniform dynamic range; the neural network is not needed to perform this task. This means that very large or very small signals to the nodes are usually the result of very large or very small weight values. Using an alternative form of the activation function, a sine function as in (19), solves this problem by providing hidden nodal signals which have significant even or odd order derivatives for all signals.

The third insight provided by dynamic stability considerations involves the dynamics of those small weights near the origin in weight space. In weight space, the stability analysis of these variables will involve small real eigenvalues which will be dominated by weight oscillations. The dynamics of these weights are associated with dynamic processes that have small real parts and are therefore near the center manifold. Since the training data has a significant noise component for characters that are nearly well formed, these characters are readable but have numerous obscuring edge pixels. For example, the dynamics near the center manifold is in part the result of an attempt by the optimization process to make fine character distinctions on large signals well above the noise level and in part the result of small weights interacting with the noise. Since these two effects are indistinguishable and create very complex dynamics, it is better to set the weights involved to zero and force these dynamic processes out of the network.

The three ideas combine to yield a training method which is smoothed on the exterior, large weight, region by regularization. At the same time the weights are smoothed in the interior, small weight, region by Boltzmann pruning. The combination of these two methods greatly restricts the active region of weight space. Weight removal simplifies the problem by reducing the number of degrees of freedom. Restricting the range of weight values to a spherical shell near but excluding the origin matches the significance of the network to the significance of the training data. In the region between constraints, dynamic stability is enhanced by choosing activation functions with nonzero even or odd derivatives through out the space.

After this extended discussion, one may wonder if this type of recognition problem is unique to OCR and fingerprint problems. The authors would argue that it is not. The characters use in this work are normalized to 32 by 32 binary images with 1024 pixels each. Characters form a very small subset of all 2^{1024} the possible 32 by 32 images. The point of regularization is to confine the training dynamics to a region of weight space where the images are near the subset of images which are meaningful characters. At the other extreme, we argue that the subset of character images is not dense in feature space and that many noisy images (that are near character images but are not easily recognizable characters) are found in large training sets. The complex but inconclusive dynamics associated with the noisy images are pruned away by the Boltzmann process. These properties are not unique to character recognition. They are even more pronounced for fingerprints because the set of meaningful images is a much smaller fraction of all 512 by 480 8-bit gray images. This should not be surprising since patterns generated or identified by an image algebra [18] are a small subset of the patterns which could be generated with arbitrary generators and links in any image algebra.

4 Neurodynamics of Learning

The design and implementation of most neural network architectures is based on an analysis of the size and content of the network training data. The direct form of analysis is suitable when the size of the training data is small, the class distribution is uniform, and the local and global dimension of the feature set are approximately equal. In fingerprint and character classification applications, the training sets are large and the local and global ranks of the feature data are very different. The complex structure of the training data requires that large networks, 10^4 weights and 10^2 nodes, be used. If the training process is treated as a dynamical system with the weights as the independent variables this would result in a Jacobian with 10^8 terms. Previous pruning studies have shown that these networks contain at least 50% redundant weights [3] and have confirmed that no more than 12 bits of these weights are significant. This makes direct analysis of the Jacobian numerically intractable.

To avoid the difficulties in analyzing this type of complex low accuracy system we looked directly at the qualitative properties expected in systems of this kind [16, 11, 14] and altered the training procedure to take the expected dynamic behavior into account. This analysis used the dynamical systems approach to provide us with qualitative information about the phase portrait of the system during training rather than a statistical representation of the weight space of the MLP network. For this approach we considered the training process as an n-dimensional dynamical system [19] where for a given neuron:

$$\frac{du_i}{d\tau_i} = -\frac{u_i}{\tau_i} + f_j(u_j) + I_i \tag{18}$$

where τ_i is the decay time for the unit and f_j is the input-output transfer function, a sinusoidal function driven through the w_{ij} interconnection weights,

$$f_j(u_j) = \frac{1}{2} + \frac{1}{2}\sin(\sum_j w_{ij}u_j), \tag{19}$$

and I_i is the initial input. We effectively reduce the dimension of the problem using the center manifold approach [20]. This approach is similar to the Lyapunov-Schmidt technique [21] which reduces the dimension of the system from n to the dimension of the center manifold, which in numerical calculations is equal to the number of calculable eigenvalues. Since the number of weights in the typical network is approximately 10^4 and the number of bits in the feature data is approximately 12, direct numerical methods for calculation of the eigenvalues from the linearized dynamics are very poorly conditioned. The center manifold method has the advantage over the Lyapunov method in that the reduced problem still is a dynamical system with the same dynamic properties as the original system. This reduction in dimension is implemented using the Boltzmann machine for a scaled conjugate gradient (SCG) learning algorithm.

The reduced problem after application of the center manifold method is still an SCG system. The SCG requires that at any given point, the performance of the dynamical system be assessable through a certain error function, E. Then the system parameters are iteratively adjusted in the opposite direction of error. The reduction on the size of error can be approximated as follows:

$$\frac{dw_{ij}}{d\tau} = -\eta\frac{\partial E}{\partial w_{ij}}, \tag{20}$$

where η is the learning rate or time constant for parameter dynamics, w_{ij} are the weights, and E is the error.

This approach, unlike most training methods, can reduce the error independent of the content of the particular sample distribution and the size of training data. This results in a saving in training time and improvement in performance without analysis of those network components which make minimal contributions to the learning process.

5 Optimization Constraints

The level of improvement in network performance which is achieved here requires four modifications in the optimization, each of which must be incorporated in the weight and error calculations of the scaled conjugate iteration [4]. Each of these constraints alters the dynamics of the training process in a way that simplifies the form of the decision surfaces, which globally have a dimension of about 100 with a local dimensionality of 10.5. Understanding the topology of this space is useful for developing improved training methods based on dynamics.

The four modifications all modify the error surface being optimized by changing the shape or dimension of the error function. All of the modifications take place in the inner loop of the optimization.

5.1 Regularization

Regularization decreases the volume of weight space used in the optimization process. This is achieved by adding an error term which is proportional to the sum of the squares of the weights. The effect is to create a parabolic term in the error function that is centered on the origin. This reduces the average magnitude of the weights. A scheduled sequence of regularization values is used which starts with high regularization and decreases until no further change in the form of the error-reject curve is detected. Constraining the network weights causes a simplification in network structure by reducing the number of bits in the weights and therefore the amount of information contained in the network.

5.2 Sine Activation

The usual form for the activation function for neural networks is a sigmoidal or logistic function. This function has small changes in all derivatives for large or small value of the input signal. This results in conditions where the Jacobian of the dynamical system being optimized is effectively singular [22]. This results in large numbers of near zero eigenvalues for the optimization process and forces the optimization to be dominated by center manifold dynamics [16, 20]. Changing the activation function to a sinusoidal function creates a significant change in the dynamics of the training since even and odd higher derivatives of the dynamical system are never both small. This improves network training dynamics and results in better reject-accuracy performance and simpler networks [23].

5.3 Boltzmann Pruning

Boltzmann pruning has two effects on the training process. First, it takes small dynamic components which have small real eigenvalues, and are therefore near the center manifold, and places them on the center manifold. This simplifies training dynamics by reducing weight space dimension. Second, Boltzmann pruning keeps the information content of the weights bounded at values which are equal to or less than the information content of the feature set. For example, when K-L features are derived from binary images, the significance of the feature is no greater than the number of significant bits in the mean image value used in the calculation, $\log_2 N$ bits for N training examples. Boltzmann pruning forces this constraint on the weights [3].

In [3], when Boltzmann pruning was used, detailed annealing schedules were used to insure convergence of the training process. When regularization is combined with pruning, the need for annealing schedules is removed and pruning can proceed concurrently with the regularization process. This reduces the cost of pruning to a small computational cost associated with the weight removal.

5.4 Class Based Error Weights

In problems with widely variant prior class probabilities, such as fingerprint classification, it may be necessary to provide large samples of rare classes so that class statistics are accurately represented, but it is important to train the classifier with the correct prior class probabilities. This is discussed in chapter 7 of [24]. In the conjugate gradient method used here, both the network errors and error signals used in the control of the iteration must be calculated using class weights thus:

$$Error = \frac{\sum Class\ Weights \times Raw\ Error}{\sum Class\ Weights}. \qquad (21)$$

This insures that the optimization is performed in a way that produces the best solution to the global problem but allows reasonable sampling of less common classes. In the digit classification used here, uniformly distributed classes were available but in a sample of alphabetic text the classes are not uniformly distributed and class based error weights should be used.

6 Results

The training method was used on samples of handprinted digits and fingerprints. The digit sample contained 7480 training and 23140 testing examples equally distributed for classes "0" to "9". The fingerprint data contained 2000 training and 2000 testing samples from NIST database SD4 [25]. These training and test samples are identical to those used in [1]. The MLPs used here were three-layer networks with 96-96-10 structure and 10282 weights for the OCR problem, and 112-112-5 structure and 13221 weights for fingerprints.

Training was carried out by selecting a Boltzmann temperature and successively training the network at decreasing values of the regularization factor, R_f. Typically the first training sequence used $R_f = 2.0$. After each training pass R_f was reduced in a 2, 1, 0.5 sequence until the regularization had no significant effect on the reject-error performance of the test sample. As the regularization is decreased the number of weights in the network increases, as does the size of the average weight. The smallest net will occur at high temperature and regularization and the largest at low temperature and regularization. The goal of the study is to find the smallest network that gives the steepest fall in error for a given reject level.

6.1 Distribution of Weights

We have developed various methods for evaluation of network dynamics based on the statistical properties of the weights. This is necessary both because the number of computationally identically fully connected nets is very large, 96! and 112! and because there are potentially more than 10,000 interconnecting weights. To simplify the discussion we will show the weight analysis for the OCR problem only. Identical arguments would apply to the fingerprint case.

6.1.1 Network Topology

Figure 3 and 4 show the distribution of interconnecting weights for a OCR network trained at a temperature of $T = 0.001$ at high and low regularization, $R_f = 2.0$ and $R_f = 0.1$. The original network had 10282 weights, the $R_f = 2.0$ network has 2365 weights (23%), and the $R_f = 0.1$ network has 3139 weights (31%). The vertical axis in figures 3 and 4 is the input node number. Since K-L features are used, the input nodes with the smallest numbers have the greatest statistical significance. These nodes are highly connected; the degree of connectively decreases with increasing node number and decreasing statistical significance. The hidden layer nodes are always fully connected to the output nodes in these experiments. This connectivity is not forced but the pruning process does not select these weights for removal.

The significant topological process seen in the interconnection pattern in going from high to low regularization is the increase in connectivity of high number K-L features. The 96th input node (top line of figure 3) is not connected at $R_f = 2.0$ and has two connections at $R_f = 0.1$ (top line of figure 4). The number of connections to input node 30 is 23 at $R_f = 2.0$ and increases to 37 at $R_f = 0.1$. The number of non-zero of weights increases by 35% and the number of connections to input node 30 increases by 61%. Nodes higher than number 30 get a disproportionate part of the increased connectivity.

6.1.2 Weight Distribution

In addition to altering the topology of the network, the training process changes the distribution of the weights. The distribution of weight on intervals of 0.05 is shown in figures 5 and 6 for temperatures of 10^{-3} and 10^{-5}. The change in the magnitude of the weights is more sensitive to regularization than to temperature. While the number of weights changes by a factor of 2.76, 8653 to 3134, the distribution peak changes by a factor of 6.0 from 3369 to 565. Reducing the regularization doubles the mean weight size at both temperatures. At the lower temperature, regularization created large numbers of small weights on low order K-L features. These weights are no more effective in the classification process than the reduced numbers of larger weights produced at higher temperature.

6.1.3 Reject Error Performance

Figures 7 and 8 show the reject error performance of ten different networks at two temperatures, $T = 10^{-3}$ and and $T = 10^{-5}$ and at five values of R_f. This study involved eight temperatures and an average of six regularization values, 48 networks, for OCR. These values are sufficient to illustrate the process. The significant factor here is that the best error-reject performances for the two temperatures are very similar. Both $R_f = 0.1$ networks reach 0.1% error after rejecting about 17% of the characters despite having different topology, mean weight sizes different by a factor of 2.17, and weight counts different by a factor of 2.76. Since the total information content of the network is proportional to the log base 2 of the product of the weight size and the number of weights, the information content decreases by only 23%, which is fully compensated for by improved network topology.

6.2 Digit Recognition

Figure 9 compares the results of digit recognition using MLP networks with sinusoidal and sigmoidal activation functions and a PNN network. Both MLPs were trained using successive regularization and Boltzmann pruning. The zero reject error rates are 3.34% for the sigmoidal MLP, 2.54% for the PNN, and 2.45% for the sinusoidal MLP. The sinusoidal result is the best yet achieved on this data and is comparable to human performance [26]. The slope of the curves is initially proportional to their accuracy, but at higher reject rates the sinusoidal MLP has substantially better performance, indicating simpler decision surfaces.

6.3 Fingerprint Classification

Figure 10 compares the results of fingerprint classification using MLP networks with sinusoidal and sigmoidal activation functions and a PNN network. Both MLPs were trained using successive regularization and Boltzmann pruning. The zero reject error rates are 9.2% for the sigmoidal MLP, 7.2% for the PNN, and 7.8% for the sinusoidal MLP. The sinusoidal result is not as low as PNN at zero rejection, indicating that the decision surfaces required for this problem are more complex than the less difficult digit recognition problem. The slope of the reject-error curves is not proportional to the accuracy initially nor at any other point.

However, at higher reject rates the sinusoidal MLP has substantially better performance, indicating simpler decision surfaces are providing better confidence estimates used to generate the error-reject curve. At 10% reject the error rates have changed to 5.45% for the sigmoidal MLP, 4.96% for the PNN, and 3.43% for the sinusoidal MLP. This again demonstrates that the decision surfaces generated by the dynamically optimized MLP are simpler than those of the other networks.

7 Conclusions

In this paper we have shown that some relatively low cost modifications to the MLP training process based on training dynamics can result in lower error and better error-reject performance on difficult classification problems. The changes in training strategy are motivated by an analysis of a simplified recurrent model which illustrates the complexity of a network with feedback signals. These improvements yield less complex decision surfaces. The digit recognition problem was solved with consistently better performance at all reject rates. The more difficult fingerprint classification problem was solved in a way which still showed some advantage for complex PNN decision surfaces at zero reject, but which yielded better performance than PNN after a small percentage of the low confidence classifications were rejected. In all cases the sigmodial MLPs had more error at all levels of rejection than MLPs with sinusoidal hidden-nodes.

Appendix

Given a network containing n neurons with nodal voltages $\mathbf{U}$,

$$\frac{d\mathbf{U}}{dt} = (\mathbf{U}_d\,\mathbf{F} + \mathbf{G})\,\mathbf{U} \qquad (A-1)$$

with equilibria $\mathbf{U}_s$ given by:

$$\mathbf{U}_s = -\mathbf{F}^{-1}\,\mathbf{G} \qquad (A-2)$$

the system has solutions of the form:

$$\mathbf{U} = (\mathbf{I} - \textstyle\int \mathbf{F}\exp(\int \mathbf{G}\,dt)\,dt)^{-1}\,\exp(\int \mathbf{G}\,\mathbf{U}(0)dt). \qquad (A-3)$$

If we write the solution in the form:

$$\mathbf{U} = \mathbf{A}\,\mathbf{U}(0) \qquad (A-4)$$

then for a general 3 neuron network with no driving signal, time independent internal interconnections and $\mathbf{G}$ with real eigenvalues, a symmetric network, a similarity transform $\mathbf{S}$ and eigenvalues λ_i:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \qquad (A-5)$$

$$a_{11} = \frac{(h_{12}\,h_{23} - h_{13}\,h_{22} + h_{13})\,\left(s_{31}\,s_{33}\,e^{\lambda_3\,t} + s_{21}\,s_{23}\,e^{\lambda_2\,t} + s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{\left(h_{12}\,h_{33}-h_{13}\,h_{32}-h_{12}\right)\,\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{22}-1\right)\,h_{33}-h_{23}\,h_{32}-h_{22}+1\right)\,\left(s_{31}^2\,e^{\lambda_3\,t}+s_{21}^2\,e^{\lambda_2\,t}+s_{11}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-6)$$

$$a_{12}=\frac{\left(h_{12}\,h_{23}-h_{13}\,h_{22}+h_{13}\right)\,\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{\left(h_{12}\,h_{33}-h_{13}\,h_{32}-h_{12}\right)\,\left(s_{32}^2\,e^{\lambda_3\,t}+s_{22}^2\,e^{\lambda_2\,t}+s_{12}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{22}-1\right)\,h_{33}-h_{23}\,h_{32}-h_{22}+1\right)\,\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-7)$$

$$a_{13}=\frac{\left(h_{12}\,h_{23}-h_{13}\,h_{22}+h_{13}\right)\,\left(s_{33}^2\,e^{\lambda_3\,t}+s_{23}^2\,e^{\lambda_2\,t}+s_{13}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{\left(h_{12}\,h_{33}-h_{13}\,h_{32}-h_{12}\right)\,\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{22}-1\right)\,h_{33}-h_{23}\,h_{32}-h_{22}+1\right)\,\left(s_{31}\,s_{33}\,e^{\lambda_3\,t}+s_{21}\,s_{23}\,e^{\lambda_2\,t}+s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-8)$$

$$a_{21}=-\frac{\left(\left(h_{11}-1\right)\,h_{23}-h_{13}\,h_{21}\right)\,\left(s_{31}\,s_{33}\,e^{\lambda_3\,t}+s_{21}\,s_{23}\,e^{\lambda_2\,t}+s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{11}-1\right)\,h_{33}-h_{13}\,h_{31}-h_{11}+1\right)\,\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{\left(h_{21}\,h_{33}-h_{23}\,h_{31}-h_{21}\right)\,\left(s_{31}^2\,e^{\lambda_3\,t}+s_{21}^2\,e^{\lambda_2\,t}+s_{11}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-9)$$

$$a_{22}=-\frac{\left(\left(h_{11}-1\right)\,h_{23}-h_{13}\,h_{21}\right)\,\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{11}-1\right)\,h_{33}-h_{13}\,h_{31}-h_{11}+1\right)\,\left(s_{32}^2\,e^{\lambda_3\,t}+s_{22}^2\,e^{\lambda_2\,t}+s_{12}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{\left(h_{21}\,h_{33}-h_{23}\,h_{31}-h_{21}\right)\,\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-10)$$

$$a_{23}=-\frac{\left(\left(h_{11}-1\right)\,h_{23}-h_{13}\,h_{21}\right)\,\left(s_{33}^2\,e^{\lambda_3\,t}+s_{23}^2\,e^{\lambda_2\,t}+s_{13}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{\left(\left(h_{11}-1\right)\,h_{33}-h_{13}\,h_{31}-h_{11}+1\right)\,\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{(h_{21}\,h_{33}-h_{23}\,h_{31}-h_{21})\left(s_{31}\,s_{33}\,e^{\lambda_3\,t}+s_{21}\,s_{23}\,e^{\lambda_2\,t}+s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-11)$$

$$a_{31}=\frac{((h_{11}-1)\,h_{22}-h_{12}\,h_{21}-h_{11}+1)\left(s_{31}\,s_{33}\,e^{\lambda_3\,t}+s_{21}\,s_{23}\,e^{\lambda_2\,t}+s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{((h_{11}-1)\,h_{32}-h_{12}\,h_{31})\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{(h_{21}\,h_{32}+(1-h_{22})\,h_{31})\left(s_{31}^2\,e^{\lambda_3\,t}+s_{21}^2\,e^{\lambda_2\,t}+s_{11}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-12)$$

$$a_{32}=\frac{((h_{11}-1)\,h_{22}-h_{12}\,h_{21}-h_{11}+1)\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{((h_{11}-1)\,h_{32}-h_{12}\,h_{31})\left(s_{32}^2\,e^{\lambda_3\,t}+s_{22}^2\,e^{\lambda_2\,t}+s_{12}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{(h_{21}\,h_{32}+(1-h_{22})\,h_{31})\left(s_{31}\,s_{32}\,e^{\lambda_3\,t}+s_{21}\,s_{22}\,e^{\lambda_2\,t}+s_{11}\,s_{12}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-13)$$

$$a_{33}=\frac{((h_{11}-1)\,h_{22}-h_{12}\,h_{21}-h_{11}+1)\left(s_{33}^2\,e^{\lambda_3\,t}+s_{23}^2\,e^{\lambda_2\,t}+s_{13}^2\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$-\frac{((h_{11}-1)\,h_{32}-h_{12}\,h_{31})\left(s_{32}\,s_{33}\,e^{\lambda_3\,t}+s_{22}\,s_{23}\,e^{\lambda_2\,t}+s_{12}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|}$$

$$+\frac{(h_{21}\,h_{32}+(1-h_{22})\,h_{31})\left(s_{31}\,s_{33}\,e^{\lambda_3\,t}+s_{21}\,s_{23}\,e^{\lambda_2\,t}+s_{11}\,s_{13}\,e^{\lambda_1\,t}\right)}{|\mathbf{H}|} \qquad (A-14)$$

where:

$$h_{11}=f_{13}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{11}\,s_{13}\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{12}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{11}\left(\frac{s_{31}^2\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}^2\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{11}^2\,e^{\lambda_1\,t}}{\lambda_1}\right) \qquad (A-15)$$

$$h_{12}=f_{13}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{12}\left(\frac{s_{32}^2\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{22}^2\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{12}^2\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{11}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1\,t}}{\lambda_1}\right) \qquad (A-16)$$

$$h_{13} = f_{13}\left(\frac{s_{33}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{23}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{13}^2\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{12}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{11}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{1,1}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right) \qquad (A-17)$$
$$h_{21} = f_{23}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{22}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{21}\left(\frac{s_{31}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}^2\,e^{\lambda_1 t}}{\lambda_1}\right) \qquad (A-18)$$
$$h_{22} = f_{23}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{22}\left(\frac{s_{32}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}^2\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{21}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1 t}}{\lambda_1}\right) \qquad (A-19)$$
$$h_{23} = f_{23}\left(\frac{s_{33}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{23}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{13}^2\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{22}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{21}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{1,1}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right) \qquad (A-20)$$
$$h_{31} = f_{33}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{32}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{31}\left(\frac{s_{31}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{21}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{11}^2\,e^{\lambda_1 t}}{\lambda_1}\right) \qquad (A-21)$$
$$h_{32} = f_{33}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1 t}}{\lambda_1}\right)$$
$$+f_{32}\left(\frac{s_{32}^2\,e^{\lambda_3 t}}{\lambda_3}+\frac{s_{22}^2\,e^{\lambda_2 t}}{\lambda_2}+\frac{s_{12}^2\,e^{\lambda_1 t}}{\lambda_1}\right)$$

$$+f_{31}\left(\frac{s_{31}\,s_{32}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}\,s_{22}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{11}\,s_{12}\,e^{\lambda_1\,t}}{\lambda_1}\right) \qquad (A-22)$$

$$h_{33}=f_{33}\left(\frac{s_{33}^2\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{23}^2\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{13}^2\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{32}\left(\frac{s_{32}\,s_{33}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{22}\,s_{23}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{12}\,s_{13}\,e^{\lambda_1\,t}}{\lambda_1}\right)$$

$$+f_{31}\left(\frac{s_{31}\,s_{33}\,e^{\lambda_3\,t}}{\lambda_3}+\frac{s_{21}\,s_{23}\,e^{\lambda_2\,t}}{\lambda_2}+\frac{s_{1,1}\,s_{13}\,e^{\lambda_1\,t}}{\lambda_1}\right) \qquad (A-23)$$

and

$$|\mathbf{H}|=(1-h_{11})\,((1-h_{22})\,(1-h_{3,3})-h_{23}\,h_{32})$$
$$+h_{12}\,(-h_{21}\,(1-h_{33})-h_{23}\,h_{31})-h_{13}\,(h_{21}\,h_{32}+(1-h_{22})\,h_{31}) \qquad (A-24)$$

To add driving terms the **G** matrix must be augmented with n additional terms, $\mathbf{U}_{in}$, which are coupled to the system through n new terms, **G**, where each term is a n by n matrix.:

$$\mathbf{G}=\begin{pmatrix}\mathbf{O} & \mathbf{G}_{in}\\ \mathbf{O} & \mathbf{U}_{in}\end{pmatrix} \qquad (A-25)$$

For an uncoupled n = 3 case, such as a conventional isolated neuron MLP, this takes the form:

$$\mathbf{G}=\begin{pmatrix}0&0&0&\mathrm{g1}&0&0\\0&0&0&0&\mathrm{g2}&0\\0&0&0&0&0&\mathrm{g3}\\0&0&0&\mathrm{u1}&0&0\\0&0&0&0&\mathrm{u2}&0\\0&0&0&0&0&\mathrm{u3}\end{pmatrix} \qquad (A-26)$$

$$\Lambda=\begin{pmatrix}\mathrm{u3}&0&0&0&0&0\\0&\mathrm{u2}&0&0&0&0\\0&0&\mathrm{u1}&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{pmatrix} \qquad (A-27)$$

$$\mathbf{S}_r=\begin{pmatrix}0&0&\frac{|\mathrm{g1}|}{\sqrt{\mathrm{u1}^2+\mathrm{g1}^2}}&1&0&0\\0&\frac{|\mathrm{g2}|}{\sqrt{\mathrm{u2}^2+\mathrm{g2}^2}}&0&0&1&0\\\frac{|\mathrm{g3}|}{\sqrt{\mathrm{u3}^2+\mathrm{g3}^2}}&0&0&0&0&1\\0&0&\frac{|\mathrm{g1}|\,\mathrm{u1}}{\mathrm{g1}\sqrt{\mathrm{u1}^2+\mathrm{g1}^2}}&0&0&0\\0&\frac{|\mathrm{g2}|\,\mathrm{u2}}{\mathrm{g2}\sqrt{\mathrm{u2}^2+\mathrm{g2}^2}}&0&0&0&0\\\frac{|\mathrm{g3}|\,\mathrm{u3}}{\mathrm{g3}\sqrt{\mathrm{u3}^2+\mathrm{g3}^2}}&0&0&0&0&0\end{pmatrix} \qquad (A-28)$$

$$\mathbf{S}_l = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \frac{g3\sqrt{u3^2+g3^2}}{|g3|\,u3} \\ 0 & 0 & 0 & 0 & \frac{g2\sqrt{u2^2+g2^2}}{|g2|\,u2} & 0 \\ 0 & 0 & 0 & \frac{g1\sqrt{u1^2+g1^2}}{|g1|\,u1} & 0 & 0 \\ 1 & 0 & 0 & -\frac{g1}{u1} & 0 & 0 \\ 0 & 1 & 0 & 0 & -\frac{g2}{u2} & 0 \\ 0 & 0 & 1 & 0 & 0 & -\frac{g3}{u3} \end{pmatrix} \qquad (A-30)$$

$$\exp(\Lambda t) = \begin{pmatrix} e^{t\,u3} & 0 & 0 & 0 & 0 & 0 \\ 0 & e^{t\,u2} & 0 & 0 & 0 & 0 \\ 0 & 0 & e^{t\,u1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad (A-31)$$

$$\exp(\mathbf{G}t) = \mathbf{S}_r\,\exp(\Lambda t)\,\mathbf{S}_l = \begin{pmatrix} 1 & 0 & 0 & \frac{g1\left(e^{t\,u1}-1\right)}{u1} & 0 & 0 \\ 0 & 1 & 0 & 0 & \frac{g2\left(e^{t\,u2}-1\right)}{u2} & 0 \\ 0 & 0 & 1 & 0 & 0 & \frac{g3\left(e^{t\,u3}-1\right)}{u3} \\ 0 & 0 & 0 & e^{t\,u1} & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{t\,u2} & 0 \\ 0 & 0 & 0 & 0 & 0 & e^{t\,u3} \end{pmatrix} \qquad (A-32)$$

For general **G** the eigenvalues will be complex so that the full inverse of a complex **S** will be required and the system will be still more complex.

Acknowledgement

The authors would like to acknowledge Patrick Grother and Jerry Candela for helpful discussions of the digit and fingerprint classification data and for providing the PNN calculations in figures 3 and 4.

References

[1] J. L. Blue, G. T. Candela, P. J. Grother, R. Chellappa, and C. L. Wilson. Evaluation of Pattern Classifiers for Fingerprint and OCR Applications. *Pattern Recognition*, 27(4):485–501, 1994.

[2] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, 1990.

[3] O. M. Omidvar and C. L. Wilson. Information Content in Neural Net Optimization. *Journal of Connection Science*, 6:91–103, 1993.

[4] J. L. Blue and P. J. Grother. Training Feed Forward Networks Using Conjugate Gradients. In *Conference on Character Recognition and Digitizer Technologies*, volume 1661, pages 179–190, San Jose California, February 1992. SPIE.

[5] Michael F. Barnsley and Lyman P. Hurd. *Fractal Image Compression.* AK Peters, Wellesley MA, 1993.

[6] B. Mandelbrot. *The Fractal Geometery of Nature.* W. H. Freeman and Co., San Francisco, 1982.

[7] C. L. Wilson, P. J. Grother, and C. S. Barnes. Binary Decision Clustering for Neural Network Based Optical Character Recognition. Technical Report NISTIR 5542, National Institute of Standards and Technology, December 1994.

[8] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* New York: Academic Press, second edition, 1990.

[9] L. K. Hansen, C. Liisberg, and P. Salamon. The error-reject tradeoff. computer reprint, 1995.

[10] E. J. Hartman, J. D. Keeler, and J. M. Kowalski. Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 2:210–215, 1990.

[11] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, New York, NY, 1974.

[12] Lawrence Perko. *Differential Eqiations and Dynamical Systems.* Springer-Verlag, New York, NY, 1991.

[13] M. A. Cohen. The construction of arbitrary stable dynamics in nonlinear neural networks. *Neural Networks*, 5(1):83–103, 1992.

[14] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, NY, 1983.

[15] M. W. Hirsch, C. Pugh, and M. Shub. *Invarient Manifolds*, volume 583 of *Lecture Notes in Mathematics.* Springer-Verlag, New York, NY, 1977.

[16] J. Carr. *Applications of Centre Manifold Theory.* Springer-Verlag, New York, NY, 1981.

[17] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[18] Ulf Grenander. *General Pattern Theory.* Oxford University Press, Oxford, 1993.

[19] M. A. Cohen and S. Grossberg. Stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Trans. on Systems, Man and Cybernetics*, 13:815–826, 1983.

[20] J. Sijbrand. Properties of center manifolds. *Transactions of the American Mathematical Society*, 289(2):431–469, June 1985.

[21] S. N. Chow and J. K. Hale. *Methods of Bifurcation Theory.* Springer-Verlag, New York, Heildberg, Berlin, 1982.

[22] S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *SIAM J. Sci. Comput.*, 14(3):693–714, 1993.

[23] James L. Blue. Sine activation in neural networks. NIST IR.

[24] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, second edition, 1988.

[25] C. I. Watson and C. L. Wilson. Fingerprint database. *National Institute of Standards and Technology*, Special Database 4, **FPDB**, April 18, 1992.

[26] J. Geist, R. A. Wilkinson, S. Janet, P. J. Grother, B. Hammond, N. W. Larsen, R. M. Klear, M. J. Matsko, C. J. C. Burges, R. Creecy, J. J. Hull, T. P. Vogl, and C. L. Wilson. The Second Census Optical Character Recognition Systems Conference. Technical Report NISTIR 5452, National Institute of Standards and Technology, May 1994.

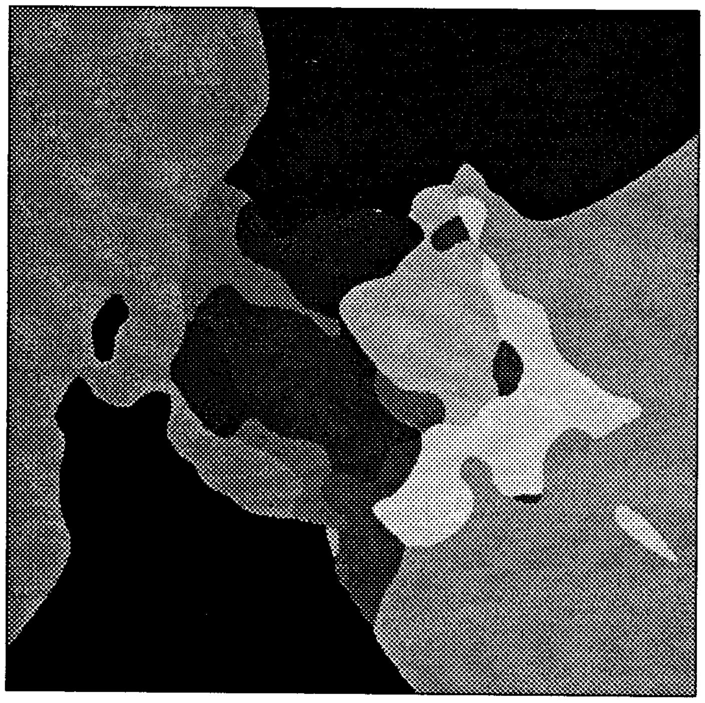

Figure 1: The diagram shows digit classifications generated by a PNN classifier using the first two K-L components in a region centered on $(0, 0)$ with an extent large enough to contain the feature vectors.

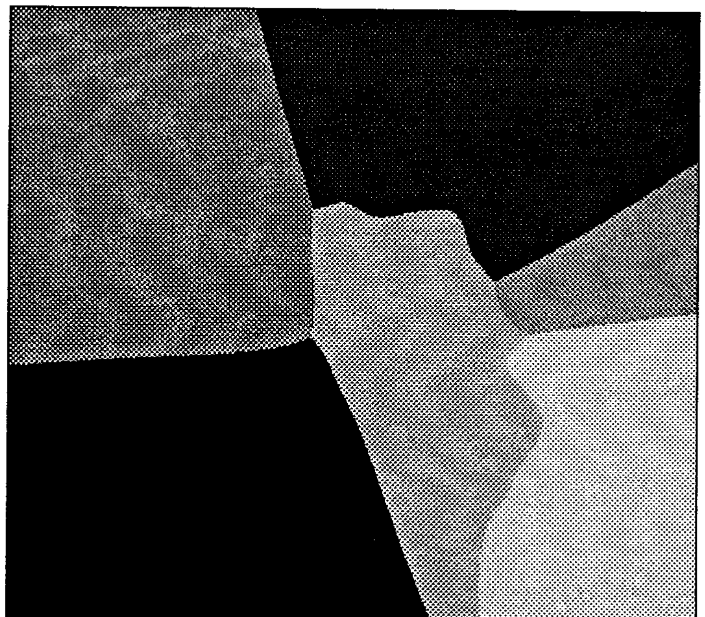

Figure 2: The diagram shows digit classifications generated by a MLP classifier using the first two K-L compenents in a region centered on $(0, 0)$ with an extent large enough to contain the feature vectors.

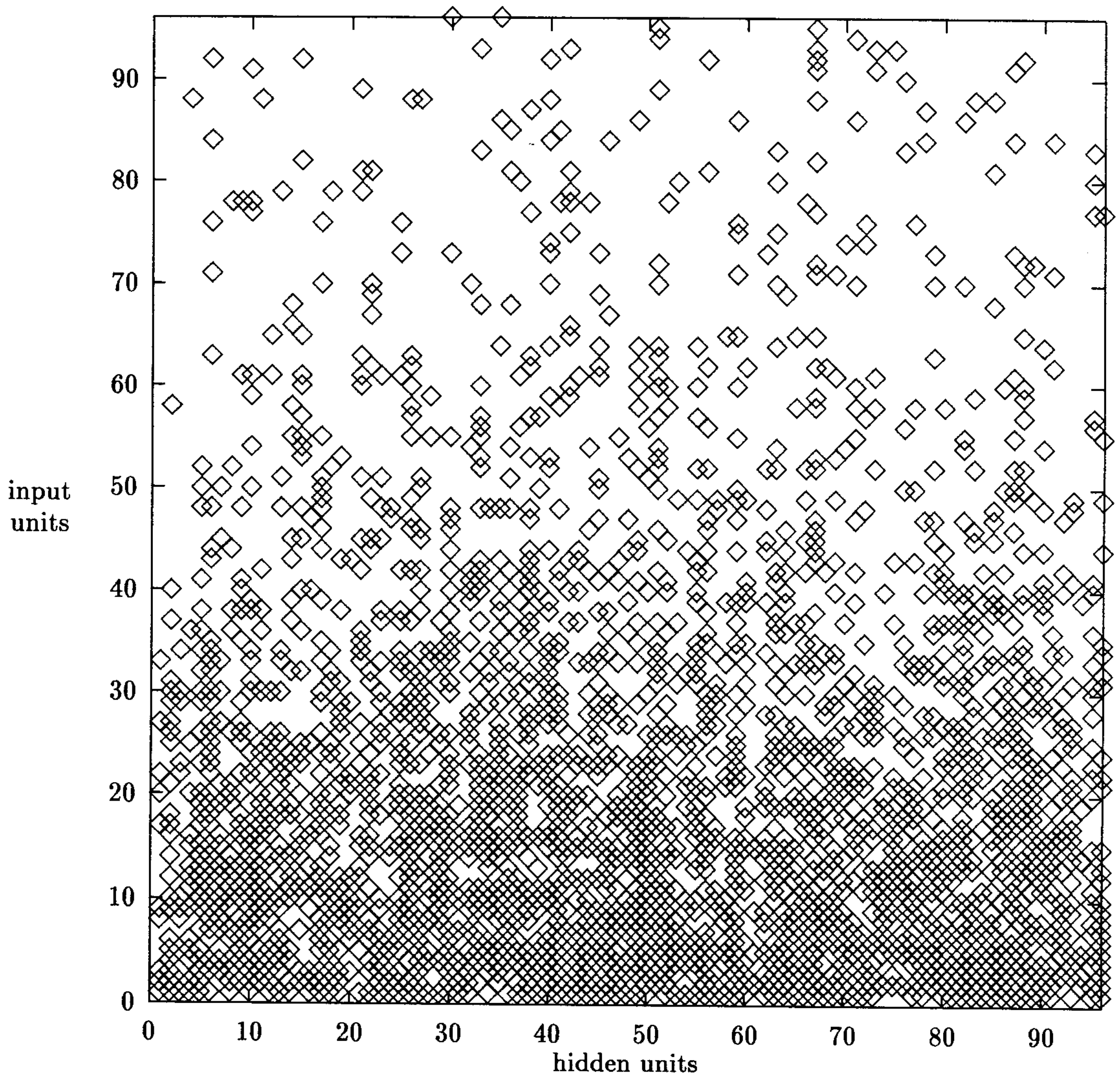

Figure 3: Nonzero weight input layer connection pattern for $R_f = 0.1$ and $T = 10^{-3}$.

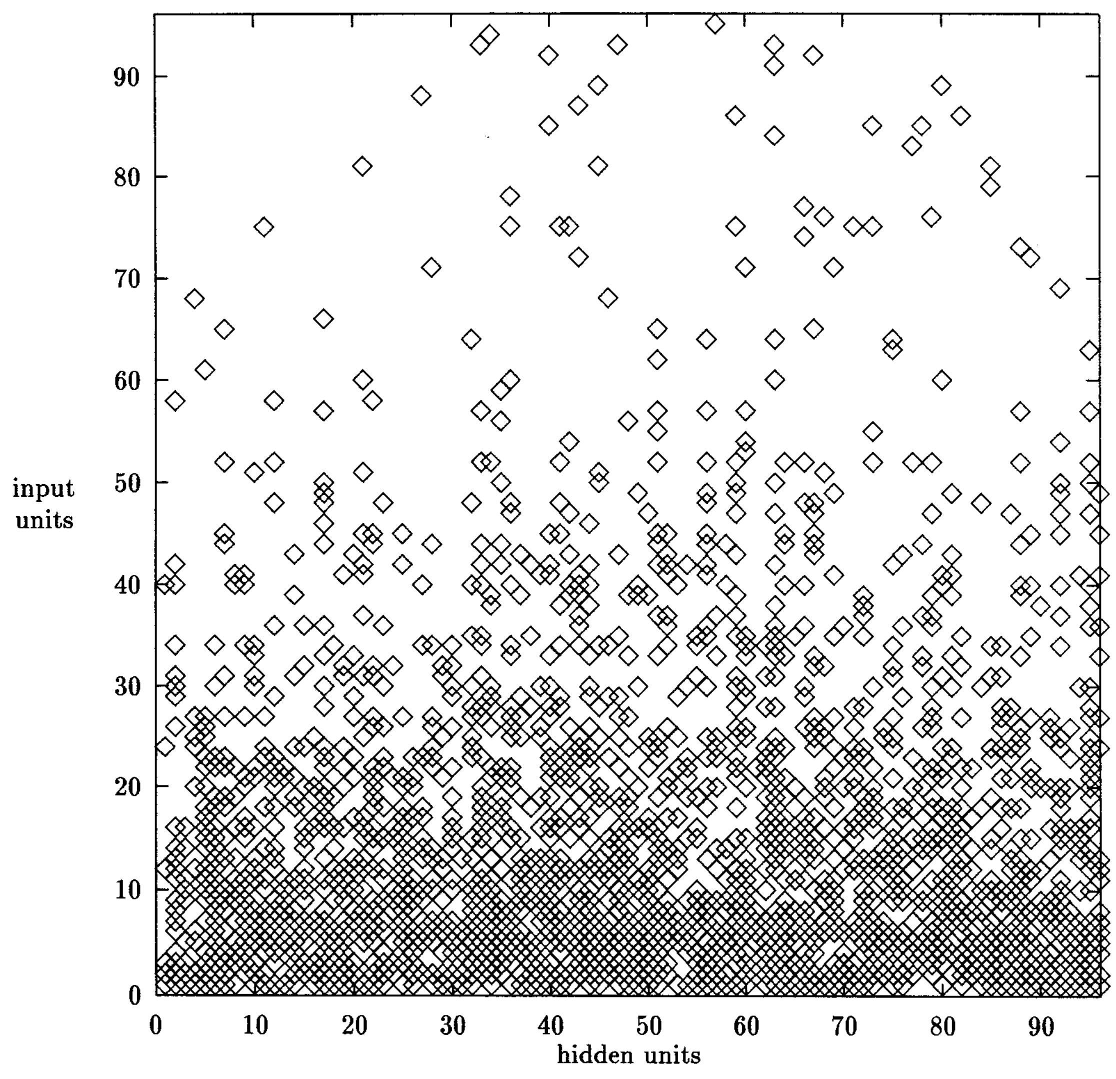

Figure 4: Nonzero weight input layer connection pattern for $R_f = 2.0$ and $T = 10^{-3}$.

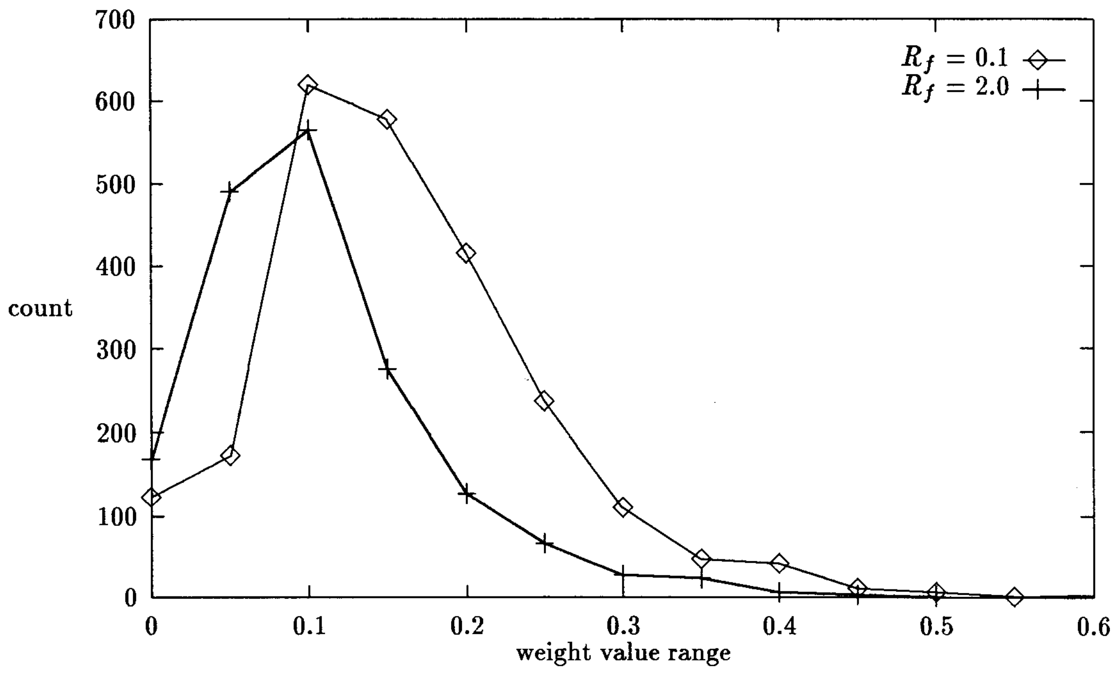

Figure 5: Distribution of weight magnitudes for sinusoidal MLP as a function of the regularization factor, R_f, for classification of digits at $T = 10^{-3}$.

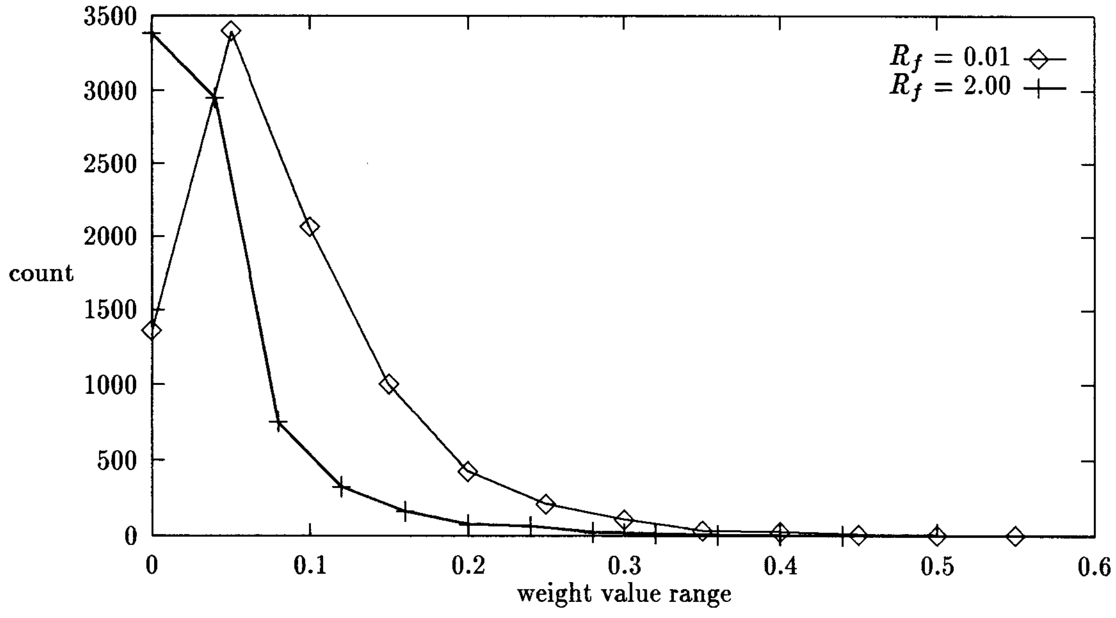

Figure 6: Distribution of weight magnitudes for sinusoidal MLP as a function of the regularization factor, R_f, for classification of digits at $T = 10^{-5}$.

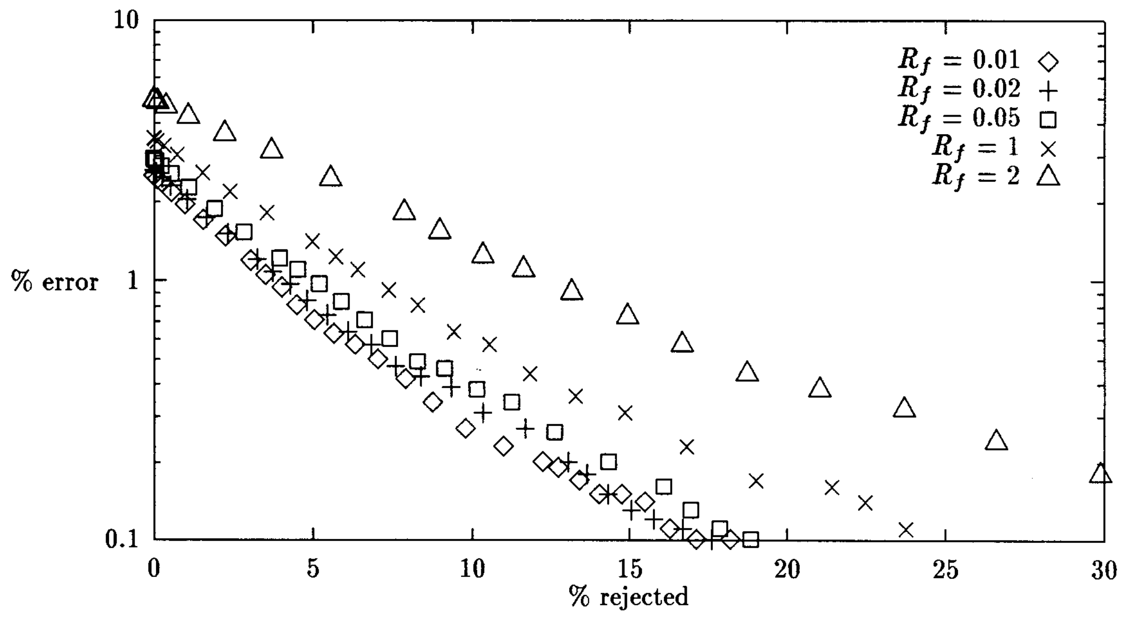

Figure 7: Error reject graph for sinusoidal MLP as a function of the regularization factor, R_f, for classification of digits at $T = 10^{-3}$.

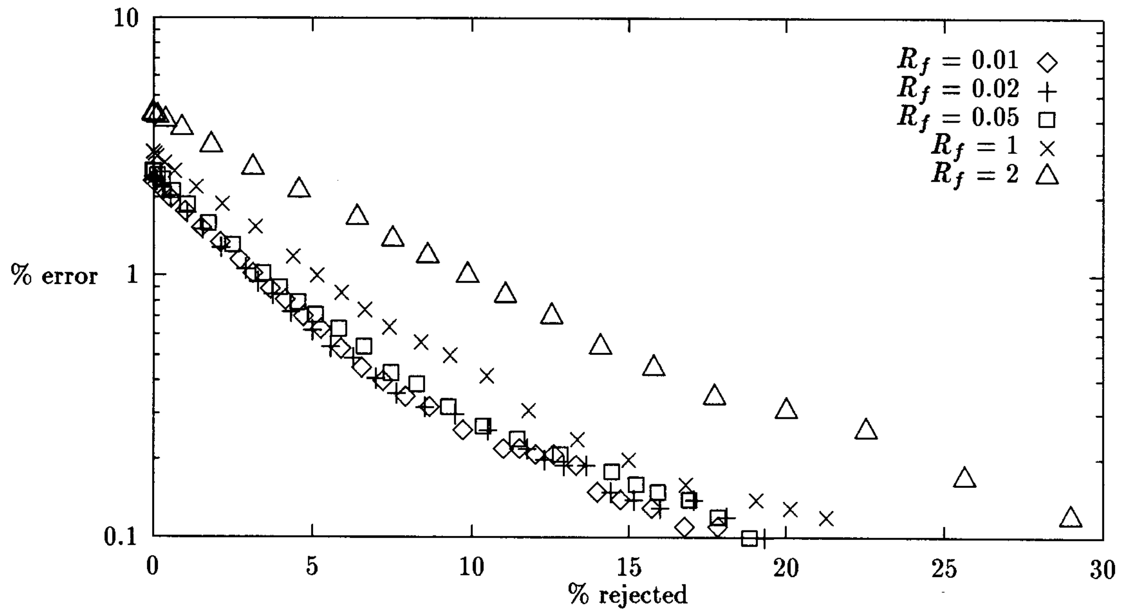

Figure 8: Error reject graph for sinusoidal MLP as a function of the regularization factor, R_f, for classification of digits at $T = 10^{-5}$.

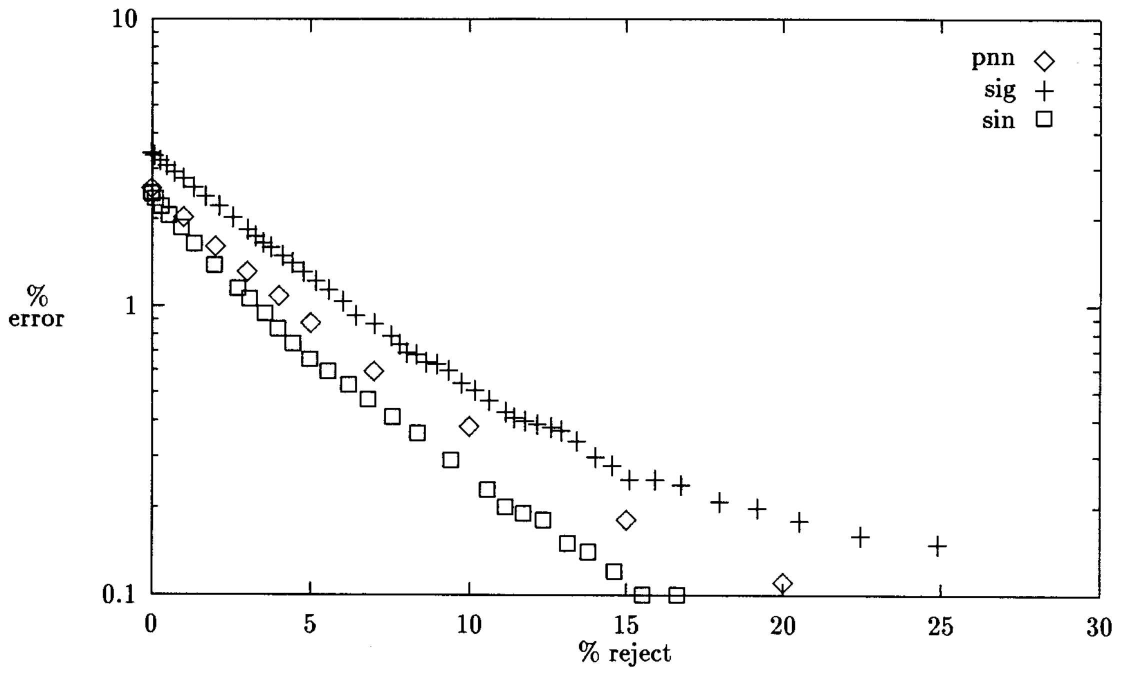

Figure 9: Error reject graph for sigmoidal MLP, sinusoidal MLP, and PNN for classification of digits.

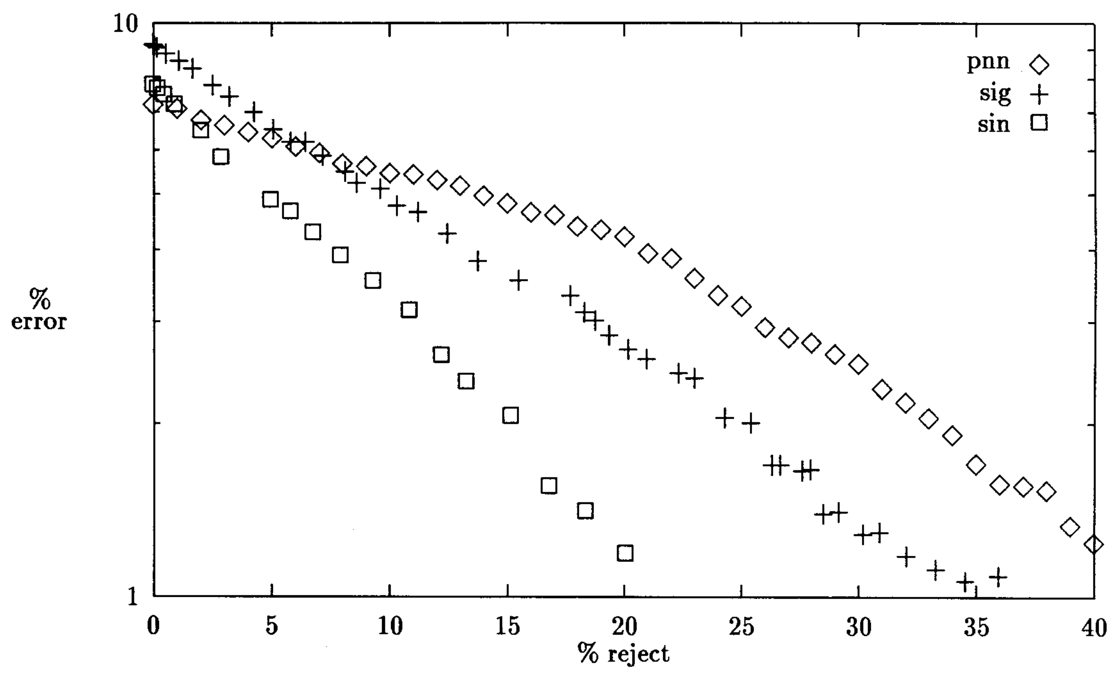

Figure 10: Error reject graph for sigmoidal MLP, sinusoidal MLP, and PNN for classification of fingerprints.