

Neurodynamics of Learning and Its Effect on 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

Using a simple dynamic model, we show that performance equal to or better than the Probabilistic Neural Network (PNN) can be achieved with a single three-layer Multilayer Perceptron (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 weight space being minimized; 3) Boltzmann pruning 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 distortion of the error surface. All four of these changes are made in the inner loop of a conjugate gradient optimization iteration 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

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, [3], 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 [3] were analyzed it was found that the feature space used in the recognition process has a topological structure which locally had an intrinsic dimension [6] of 10.5 but global Karhunen-Loeve (K-L) transform dimension of approximately 100. The number of features needed to make binary decision machines discriminate between digits was larger than the intrinsic dimensionality. For binary decision

machines, typical feature set sizes are 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 the problem: the MLP is being used to approximate a very 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 reject accuracy performance. In [7], it was shown that the slope of the reject-accuracy 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 general nonlinear function approximators [8] so, in theory, they should be capable of approximating complex decision surfaces. In this paper we show that four modifications to the conjugate gradient method discussed in [5] will allow a three layer MLP to approximate the required decision surface with zero-reject accuracy similar to PNN and k nearest neighbor (KNN) methods and reject-accuracy performance better than the best binary decision method discussed in [3], 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 we discuss the simplified dynamical model of the network. In section 3 we discuss the changes in training method used to improve network accuracy while simplifying network structure. In section 4 we discuss the dynamics which suggested these changes in network training. In section 5 we 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:

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

or as a vector:

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

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:

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

and a set of higher order couplings:

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

The system dynamics is described by:

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

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

$$\mathbf{U}_s = -\mathbf{F}^{-1} \mathbf{G} \quad (6)$$

The solution of (5) was postulated as (7) and checked using a symbolic mathematics program to validate the solution terms. See [9] page 99 or [10] page 60 for similar linear problems which were used as clues to the proposed solution. The solution derived and checked in this way is:

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

where the matrix exponential is defined by:

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

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

$$\mathbf{A} = \mathbf{S}^H \Lambda_A \mathbf{S} \quad (9)$$

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

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

In general \mathbf{G} will have complex eigenvalues; only in the symmetric case are all eigenvalues real and the upper triangle of Λ_A zero. This symmetry requirement makes the construction of stable associative memories with symmetric weights very difficult. These issues are discussed in [11] 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: $\mathbf{U}_d = \mathbf{U}(0)_d$.

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 Lorenz attractor [12], 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 kind is treated in [13, 12, 10]. 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, the change in these terms can be neglected and we write the solution as:

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

where for \mathbf{G} symmetric:

$$a_{ij} = \frac{|\mathbf{H}_{ij}|^* \sum_{k=1}^n s_{ki} e^{\lambda_k t} s_{kj}}{|\mathbf{H}|} \quad (12)$$

where $|\mathbf{H}_{ij}|^*$ is the matrix of ij th 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 symmetric, and the elements of \mathbf{H} are given by:

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

where:

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

The full solution of equation (6) can be expanded using equations (11)–(14). 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 $O(n^2)$ terms for an n th order system. Equations (11)–(14) involve calculation of $O(n^8)$ terms for an n th 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:

$$f_{ik} \frac{s_{lj} s_{lk} e^{\lambda_l t}}{\lambda_l} s_{pq} e^{\lambda_p t} s_{pq} \quad (15)$$

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 [10] p. 115 and [12] 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 [14]. 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), 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 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 [5]. 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. 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.

3.1 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.

3.2 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 sets of conditions where the Jacobian of the dynamical system being optimized is effectively singular [15]. This results in large numbers of near-zero eigenvalues for the optimization process and forces the optimization to be dominated by center manifold dynamics [14, 16]. 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 [17].

3.3 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 a feature is no greater than the number of significant digits 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 [4].

In [4], 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.

3.4 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 [18]. 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:

$$Error = \frac{\sum Class\ Weights \times Raw\ Error}{\sum Class\ Weights} \quad (16)$$

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.

4 Neurodynamics of Learning

The design and implementation of most neural network architectures is based on the result of analysis of the size and content of the network training data. These direct forms of analysis

are 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 rank of the feature data are very different. The complex structure of the training data requires that large networks, $O(10^4)$ weights and $O(10^2)$ 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 [4] 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 complex, low accuracy system we looked directly at the qualitative properties expected in systems of this kind [14, 9, 12] 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:

$$\frac{du_i}{d\tau_i} = -\frac{u_i}{\tau_i} + f_j(u_j) + I_i \quad (17)$$

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

$$f_j(u_j) = \frac{1}{2} + \frac{1}{2} \sin\left(\sum_j w_{ij} u_j\right) \quad (18)$$

and I_i is the initial input. We effectively reduce the dimension of the problem using the center manifold approach [16]. This approach is similar to the Lyapunov-Schmidt technique [20] 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 scaled conjugate gradient (SCG) learning algorithm.

The reduced problem after application of the center manifold method is still a 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:

$$\frac{dw}{d\tau} = -\eta \frac{\partial E}{\partial w}, \quad (19)$$

where η is the learning rate or time constant for parameter dynamics, w is the weight, 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 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 [21]. These training and test samples are identical to those used in [1].

5.1 Digit Recognition

Figure 3 compares the results of digit recognition using MLP networks with sinusoidal and sigmoidal activation functions to 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 [22]. The slope of the curves is initially proportional to their accuracy initially, but at higher reject rates the sinusoidal MLP has substantially better performance, indicating simpler decision surfaces.

5.2 Fingerprint Classification

Figure 4 compares the results of fingerprint classification using MLP networks with sinusoidal and sigmoidal activation functions to 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, 7.18% 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 for 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.

6 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. These improvements follow directly from 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.

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] 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.
- [4] O. M. Omidvar and C. L. Wilson. Information Content in Neural Net Optimization. *Journal of Connection Science*, 6:91–103, 1993.
- [5] 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.
- [6] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. New York: Academic Press, second edition, 1990.
- [7] L. K. Hansen, C. Liisberg, and P. Salamon. The error-reject tradeoff. computer reprint, 1995.
- [8] 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.
- [9] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, New York, NY, 1974.
- [10] Lawrence Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, New York, NY, 1991.
- [11] M. A. Cohen. The construction of arbitrary stable dynamics in nonlinear neural networks. *Neural Networks*, 5(1):83–103, 1992.
- [12] J. Guckenheimer and P. Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, New York, NY, 1983.
- [13] M. W. Hirsch, C. Pugh, and M. Shub. *Invariant Manifolds*, volume 583 of *Lecture Notes in Mathematics*. Springer-Verlag, New York, NY, 1977.
- [14] J. Carr. *Applications of Centre Manifold Theory*. Springer-Verlag, New York, NY, 1981.
- [15] S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *SIAM J. Sci. Comput.*, 14(3):693–714, 1993.
- [16] J. Sijbrand. Properties of center manifolds. *Transactions of the American Mathematical Society*, 289(2):431–469, June 1985.
- [17] James L. Blue. Sine activation in neural networks. NIST IR.

- [18] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, second edition, 1988.
- [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] S. N. Chow and J. K. Hale. *Methods of Bifurcation Theory*. Springer-Verlag, New York, Heidelberg, Berlin, 1982.
- [21] C. I. Watson and C. L. Wilson. Fingerprint database. *National Institute of Standards and Technology*, Special Database 4, **FPDB**, April 18, 1992.
- [22] 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.

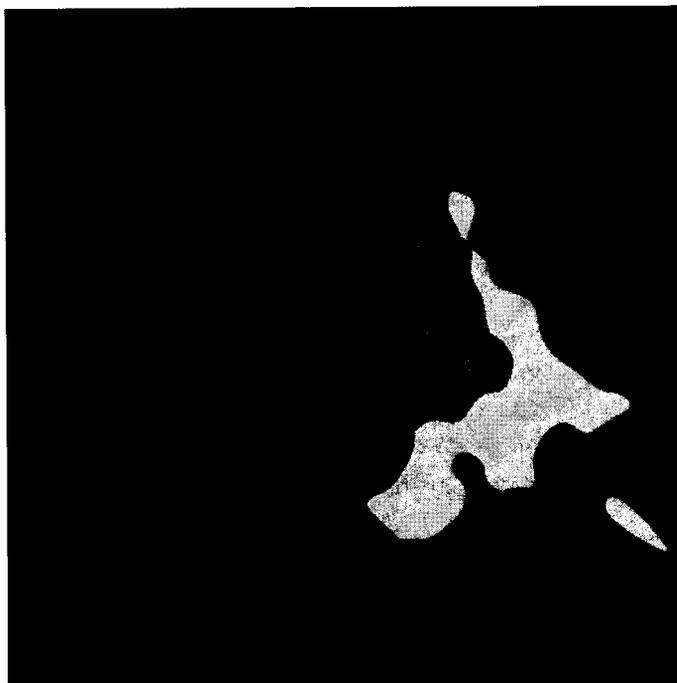


Figure 1: The diagram shows digit classifications generated by a PNN classifier using the first two K-L components in a region centered on $(0,0)$ with an extent large enough to contain the feature vectors.



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

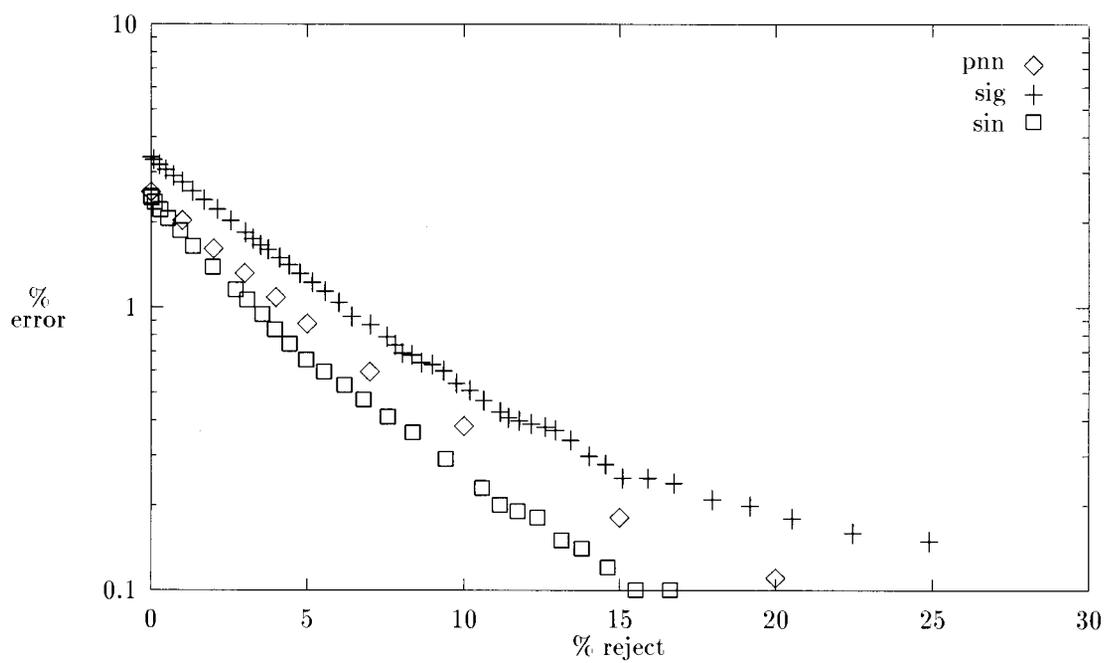


Figure 3: Error reject graph for sigmoidal MLP, sinusoidal MLP, and PNN for classification of digits.

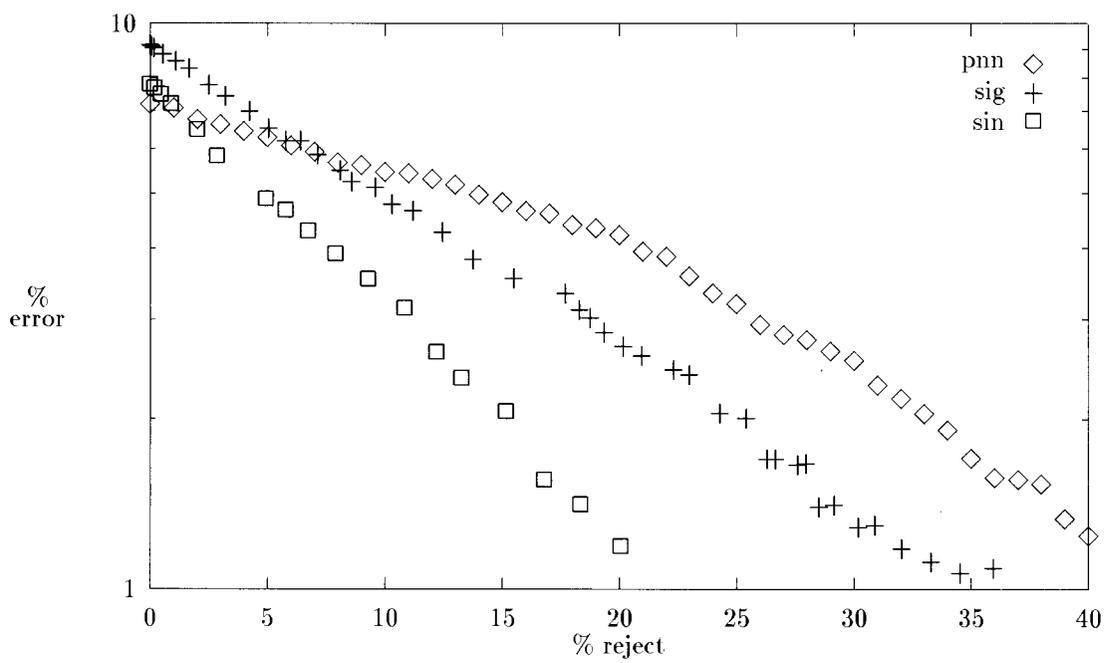


Figure 4: Error reject graph for sigmoidal MLP, sinusoidal MLP, and PNN for classification of fingerprints.