A COMPARISON OF NEURAL NETS TO STATISTICAL CLASSIFIERS FOR STUBBORN CLASSIFICATION PROBLEMS

Patricia Davies and Brian R. Silverstein
Purdue University, West Lafayette, Indiana 47906 USA

ABSTRACT

This paper addresses two types of problems which prove difficult for tradtional classifiers: having very limited training data for at least one class, and having classes with a large amount of overlap. Issues discussed will include the 1) use of nearest neighbor methods and neural nets for classification of data which is completely inseparable by linear and quadratic classifiers, 2) dealing with training sets of unequal size from each class.

1. INTRODUCTION

This paper addresses the difficulties experienced in classifying groups with large overlap. Three groups of simulated 8-dimensional data are generated and examined with 3 standard statistical classification methods [1,2], a linear classifier, a quadratic classifier, and a nearest neighbor classifier. The results are compared to those of a backpropagation neural net.

1.1 The classifiers

Statistical classifiers fall into two categories, parametric and non-parametric. In the first type, classification rules are based on models of the probability density function of the data. The linear and quadratic classifiers are of this type. Each are based on the assumption that classes have Gaussian distributions, with mean M and Σ, and density functions of the form:

$$p(X) = \frac{1}{(2\pi)^{n/2}\,|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(X-M)^T\Sigma^{-1}(X-M) \right\} \quad (1)$$

The mean and covariance for the Gaussian distribution are defined from expected values as:

$$M = E\{X\} \quad (2) \qquad \Sigma = E\{(X-M)(X-M)^T\} \quad (3)$$

These are usually unknown, and unbiased estimates, $\hat{M}$ and $\hat{\Sigma}$, are used: given samples $X_1, \dots, X_N$,

$$\hat{M} = \frac{1}{N}\sum_{i=1}^{N} X_i \quad (4)$$

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N} (X_i - M)(X_i - M)^T \quad (5)$$

Fukunaga [1] shows that the quadratic decision rule for two Gaussian distributions, ω_1 determined by M_1 and Σ_1, and occurring with *a posteriori* probability P_1, and ω_2 determined by M_2 and Σ_2, having *a posteriori* probability P_2=1-P_1 is given by:

$$h(X) = \ln\frac{P_1\,|\Sigma_2|^{0.5}}{P_2\,|\Sigma_1|^{0.5}} + \sum_{i=1}^{2} (-1)^i\,\frac{1}{2}(X-M_i)^T\Sigma_i^{-1}(X-M_i) \quad (6)$$

For h(X) > 0 the sample is judged to be a member of ω_1, and for h(X) < 0, the sample judged to be in ω_2.

The linear classifier is a special case of the quadratic classifier. It assumes that the covariances of the two distributions are the same: $\Sigma_2 = \Sigma_1 = \Sigma$. Then

$$h(X) = (M_1-M_2)^T\Sigma^{-1}(2X-M_1-M_2) + \ln(P_1/P_2) \quad (7)$$

For h(X) > 0, $X \in \omega_1$, and for h(X) < 0, $X \in \omega_2$.

The parametric methods are limited by the assumption that distributions are Gaussian. When a density function is clearly not Gaussian and is dissimilar to all other easily quantified distributions, one can still directly estimate the density function of each class at each point to be examined. This is the basis for non-parametric classification methods. These methods assume only that there are enough points from each class such that in any small region within the decision space, that the number of points occurring in these regions indicate the true nature of each density function.

One of the most commonly used non-parametric methods is the k-nearest neighbor method. Here, one determines how many of the k neighbors nearest to each point are from each class. If neighbors are from different classes, then either a voting method may be used, or one may reject that point, refusing to classify it.

1.2 The Back-propagation neural net

The back-propagation neural net (Rumelhart, *et al.* [3], and Lippmeann [4]) consists of 3 layers of processing units, referred to as "nodes" or "neurons", which pass a linear combination of inputs through a nonlinearity:

$$y = f\left(\sum_{i=0}^{N} w_i\,x_i - \theta\right) \quad (8)$$

where y is the node output, θ is the node offset, w_i are linear coefficients or "weights", x_i are the scalar inputs, and f denotes a nonlinear function. The nonlinear function f is chosen here to be a sigmoid function:

$$f(a) = (1 + e^{-a})^{-1} \quad (9)$$

This function is popular for two reasons. First, its derivative is easily calculated:

$$\partial f(a)/\partial a = -e^{-a}[1+e^{-a}]^{-2} = -f(a)^2 e^{-a} \quad (10)$$

Second, the output is a number between zero and one, with negative inputs producing outputs below 0.5 and positive inputs producing outputs above 0.5. This allows all nodes above the first layer to have outputs in the range [0,1], which simplifies some calculations. It also allows the linear combinations of inputs to be treated either as a geometric or a logical decision boundary. The output of each node is the input to nodes in the next layer. Figure 1 contains a representation of the connections for a 3-layer neural net with 5 inputs, 6 middle layer (hidden layer) nodes, and 4 outputs.

0-7803-2431-5/95 $4.00 © 1995 IEEE

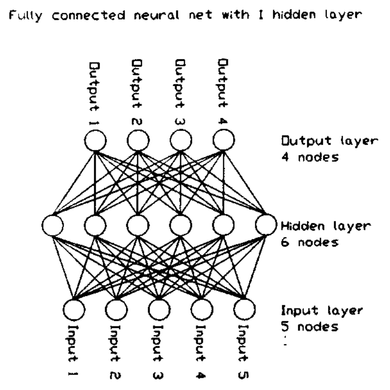

Figure 1 Neural net connections

The neural net may either have analytically predetermined weights, be trained on a selected training set, or adjusted on-line with incoming data. Training the net consists of adjusting the weights, w_i, and offsets, θ, for each node, by a minimization process discussed in detail by Rumelhart [3]. Lippmann [4] also describes the algorithm for implementing the back-propagation program. In updating the weights, an estimate of the total net error is needed. The total error, e_t, is a function of the difference between the neural net's outputs, O, and the desired outputs, E, taken over n net outputs and N members of the training set:

$$e_t = \sum_{i=1}^{N} \sum_{j=1}^{n} g(O_{ij} - E_{ij}) \qquad (11)$$

where typically $g(x) = |x|$ or $g(x) = x^2$. The back-propagation method, however, uses only the local estimate of the error, e_i, at each of the N members of the training set:

$$e_i = \sum_{j=1}^{n} g(O_{ij} - E_{ij}). \qquad (12)$$

The use of local error instead of global error can be the cause of difficulties in training the net when there are large overlaps in the training data.

2. SIMULATION DATA

Three classes of 8-dimensional data were created. Classes 1 and 2 were generated from Gaussian distributions having slightly different means and covariances. Class 3 was non-Gaussian, having bimodal distribution in each of the 8 dimensions.

- Class 1 consists of the identity distribution. Its mean was $M_1 = [0\ 0\ \ldots\ 0]^T$ and it had a diagonal covariance $\Sigma_1 = \text{diag}\ [1\ 1\ \ldots\ 1]$. Thus, each element of the 8-dimensional vector comes independently from a 0-mean Gaussian distribution with standard deviation of 1.
- Class 2 was a Gaussian distribution with $M_2 = [.1\ .1\ \ldots\ .1]^T$ and diagonal covariance $\Sigma_2 = \text{diag}\ [1.1\ 1.1\ \ldots\ 1.1]$.
- Class 3 was a bimodal distribution, where each element had a 50% chance of being from each of two Gaussian distributions, having a standard deviation of 0.2 and a mean of either 0.9 or -0.9. The spread of the modes was chosen so that the overall mean and covariance closely match those of class 1.

Histograms were generated from the first elements of 2500 samples from each of the 3 classes, shown in Figure 2. Classes were divided into 2 identically distributed subclasses of 2500 points each or 100 points each, referred to as 1a, 1b, 2a, 2b, 3a, and 3b.

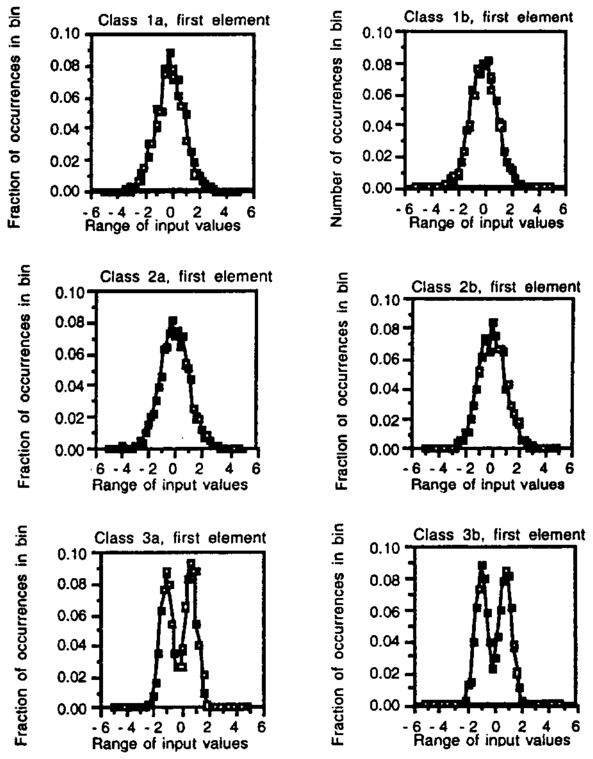

Figure 2 Histograms of simulated data.

3. CLASSIFIER RESULTS

Each classifier is judged by the percentage of classification errors it makes. For each classifier, and each pair of classes, this is tested by

1) training and testing the classifier on the same sub-class. For each pair of classes, there are 4 classifications of this type: *e.g.*, for the separating class 1 from class 2, there are 1a-2a, 1a-2b, 1b-2a, and 1b-2b.
2) training the classifier on one of the pair of sub-classes, and testing on the other. This produces a larger error. For the nearest neighbor classifier, the test point is already excluded from the set of neighbors tested.

Table 1: Classifier results for classes 1 and 2: error percentages

sub-classes	Same or Other class for testing	linear classifier error percentage			quadratic classifier error percentage			11 nearst-neighbor classifier error percentage			neural net classifier error percentage		
		l	l/s	s	l	l/s	s	l	l/s	s	l	l/s	s
1a-2a	Same	43.5	49.8	37.0	41.7	33.3	28.0	47.2	3.8	46.5	44.7	33.0	38.5
	Other	44.9	40.7	46.5	44.5	57.3	43.5				44.3	44.6	44.0
1a-2b	Same	44.1	33.3	37.0	42.5	33.2	29.0	47.9	3.8	49.0	44.8	30.4	38.0
	Other	44.6	58.3	51.5	44.4	59.5	48.0				44.4	46.1	46.0
1b-2a	Same	43.6	48.6	40.5	41.9	28.3	34.0	45.3	3.8	49.0	43.2	38.7	38.0
	Other	44.9	40.5	52.0	44.2	59.8	49.0				44.8	47.8	41.0
1b-2b	Same	44.0	39.0	41.5	41.4	39.4	33.0	45.7	3.8	41.0	43.4	33.6	33.0
	Other	44.2	51.3	47.0	44.1	55.2	46.5				44.6	46.4	44.0
average	Same	43.8	42.7	39.0	41.9	33.6	30.8	46.5	3.8	46.4	44.0	33.9	36.9
	Other	44.7	47.7	49.3	44.3	58.0	46.8				44.5	46.2	43.8

Table 2: Classifier results for classes 1 and 3: error percentages

sub-classes	Same or Other class for testing	linear classifier error percentage			quadratic classifier error percentage			11 nearst-neighbor classifier error percentage			neural net classifier error percentage		
		l	l/s	s	l	l/s	s	l	l/s	s	l	l/s	s
1a-3a	Same	47.9	41.0	36.5	45.1	39.3	27.5	32.0	3.8	61.5	50.0	43.7	47.5
	Other	50.1	61.5	56.0	48.0	56.8	65.0				50.8	50.0	51.5
1a-3b	Same	48.0	23.0	41.0	44.8	17.4	29.5	31.1	3.8	51.5	49.1	50.0	48.0
	Other	50.9	74.6	51.5	48.2	79.0	45.0				49.7	49.9	49.5
1b-3a	Same	44.0	55.7	36.5	47.4	45.1	30.5	30.4	3.8	46.5	50.0	50.0	48.5
	Other	47.9	45.8	50.5	50.3	51.2	48.0				50.3	50.0	50.0
1b-3b	Same	44.4	17.8	40.5	47.0	24.3	31.0	29.7	3.8	51.5	50.0	47.9	46.5
	Other	47.5	77.6	53.5	50.9	71.5	51.5				50.0	49.9	49.5
average	Same	46.1	34.4	38.6	46.1	31.5	29.6	30.8	3.8	52.8	49.8	47.9	47.6
	Other	49.1	64.9	52.9	47.7	64.6	52.4				50.2	50.0	50.1

Table 3: Classifier results for classes 2 and 3: error percentages

sub-classes	Same or Other class for testing	linear classifier error percentage			quadratic classifier error percentage			11 nearst-neighbor classifier error percentage			neural net classifier error percentage		
		l	l/s	s	l	l/s	s	l	l/s	s	l	l/s	s
2a-3a	Same	44.1	43.8	37.5	43.0	35.3	29.5	31.2	3.8	50.5	43.3	37.1	43.0
	Other	45.1	46.0	50.0	46.6	56.4	50.0				43.7	44.9	43.5
2a-3b	Same	44.0	45.9	37.0	44.0	56.3	28.0	31.1	3.8	50.5	42.6	33.7	42.0
	Other	45.7	43.3	53.0	44.9	35.5	51.0				44.5	42.2	44.5
2b-3a	Same	44.5	47.5	35.0	43.1	37.8	28.5	32.7	3.8	49.0	45.2	37.1	36.5
	Other	44.9	40.6	49.5	45.5	55.2	52.0				43.2	46.1	46.5
2b-3b	Same	43.9	23.0	37.0	43.4	29.3	33.5	31.7	3.8	58.0	43.1	40.1	40.0
	Other	45.6	68.2	49.5	45.0	63.4	49.0				43.6	44.2	46.5
average	Same	44.1	40.1	36.6	43.4	39.7	29.9	31.7	3.8	52.0	43.6	37.0	40.4
	Other	45.3	49.5	50.5	45.5	52.6	50.5				43.8	44.4	45.3

The small variation in the error percentages from 4 repetitions of classification provides a consistancy check for the results. Results from each of the classifications are contained in Tables 1-3, where each classifier is tested on both large sets (l), 2500 points, small data sets (s), 100 points, and combination sets where one class is large and the other is small (l/s). Small data sets generally give worse generalization information than large ones [1,5].

When both classes are Gaussian, as in comparison of classes 1 and 2, the best classifier is the quadratic. Gaussian distributions are the main assumption made in designing quadratic classifiers. The best classifier for comparison of class 1 with class 3 or class 2 with class 3 was the nearest neighbor classifier. However, there was a clear trend with the nearest neighbor method, of improved error as the number of neighbors was increased from 1 to 11, though the improvement was never more than 4%. With the nearest neighbor method, the best separation is between class 3 and either of the other two, while the linear and quadratic methods have poor separation among any combination of classes. This is the case, of course, since, the means and covariances are intentionally kept similar.

The poorest performance is from the neural net classifier. In this research, the neural net had 8 input nodes, 10 hidden layer nodes, and 1 output node, and trained on 1500 cycles through the data, for a total of up to (5000 points) * (1500 cycles) = 7.5 million updates to each of the weights (300,000 updates for the small classes). In trials with classes of no overlap, the net was trained perfectly in 50,000 updates or fewer, so we can assume that the number of training cycles is not the problem. This differs from typical classification problems, where the net is trained to near zero error [6,7].

Before praising the nearest neighbor classifier too much, we considered the thought that the apparent separation between classes may not be entirely due to the structure of the distributions: the classifiers work on a finite number of points, which may bias the results. To evaluate this effect, we compare the classification error for each method using the "a" and "b" subclasses of the same class, in Table 4. For no bias, we expect a result of 50%.

Table 4: Classifier results for subclasses: error percentages

sub-classes	linear classifier error	quadratic classifier error	11-neighbor classifier error	neural net classifier error
1a-1b	48.3	46.1	51.2	50.0
2a-2b	48.0	46.2	49.3	49.1
3a-3b	47.4	46.4	49.8	50.0
average	47.9	46.2	50.1	49.7

The nearest neighbor and neural net methods are completely unable to separate classes generated from identical density functions. For the statistical methods, the error was similar to the results from the class 1-class 3 pairs, a few points lower than 50%. We can see that using the effect of a finite sample set with the linear classifier is a roughly 2% negative bias on the estimate of actual error, while the quadratic classifier produces a roughly 3% - 5% negative bias on the error. Neither the nearest neighbor nor the neural net methods show any of this bias.

One further observation should be made, regarding the large/small classification by the nearest neighbor classifier. This algorithm cannot adjust for different set sizes. For the l/s evaluation, error percentage is measured as the total number of incorrect classifications compared to the total number of points, rather than the average of the percentages from each class. 3.8% error corresponds to 100% misclassification of the small class and 100% correct classifcation of the large class.

4. CONCLUSIONS

- When both classes are Gaussian (classes 1 and 2), the best classifier is the quadratic.
- The best method for comparison of either Gaussian class with the bimodal class was the nearest neighbor classifier.
- The poorest performance with large classes is from the neural net classifier. In these tests, the net was trained until no change in classification was observed, so we can assume that the number of training cycles is not the problem: it is characteristic of classification of highly overlapped classes.
- For small classes, all classifiers showed less error when testing on the same data, and more error when testing on the independent data, in comparison with large classes. The decrease in generalization ability is most pronounced for the quadratic classifier, where estimation of the inverses of two covariance matrices is needed. The best generalization was by the neural network.
- The consistent 3.8% error for the nearest neighbor classifier with the large-small data sets corresponds to 0% error in the large class, and 100% error in the small class. Unlike the other methods, there is no way to compensate for unequal class sizes.

5. REFERENCES

[1] Fukunaga, Keinosuke, *Introduction to Statistical Pattern Recognition*, 2nd Ed., Academic Press, NY, 1990.
[2] Patrick, Edward A., *Fundamentals of Pattern Recognition*, Prentice-Hall,Englewood Cliffs, NJ, 1972.
[3] Rumelhart, D.E., Hinton, G.E., Williams, R.J., "Learning Internal Representations By Error Propagation," in Rumelhart D.E. and McClelland, J.L. (Eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol.1, 1986.
[4] Lippmann Richard P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp4-22, April, 1987.
[5] Fukunaga, Keinosuke and Hayes, Raymond R., "Effects of Sample Size in Classifier Design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 8, pp 873-885, Augusr, 1989.
[6] Chang, Tsu-Shuan, and Abdel-Ghaffar, Khaled A.S., "A Universal Neural Net with Guaranteed Convergence to Zero System Error," *IEEE Trans. on Signal Processing*, Vol 40, No. 12, pp 3022-3031, December, 1992.
[7] Willis, M.J., Montague, G.A., Morris, A.J., and Tham, M.T., "Artificial Neural Nets: A Panacea to Modeling Problems," *1991 American Control Conference*, Vol. 2, pp2337-2342, 1991.