# Neural Networks for Classification of ARMA Models: An Experimental Study

Paul G. McKee*
Fifth Generation Systems, Inc.
10049 N. Reiger Road
Baton Rouge, LA 70809-4559

José M.F. Moura
LASIP, ECE
Carnegie Mellon University
Pittsburgh, PA 15213-3890

**Abstract.** This paper presents a set of extensive experiments with alternative neural network learning algorithms. These neural network configurations were tested on the problem of discriminating signals generated by autoregressive moving-average (ARMA) linear systems driven by white noise. These ARMA signals model a wide variety of signals arising in the ocean environment. We tested the various network models for their classification accuracy and speed of learning. The models investigated were back propagation, quickprop, Gaussian node networks, radial basis functions, the modified Kanerva method, and networks without hidden units. For comparison, nearest-neighbor classifiers were also tested. Classification performance and learning time results are presented.

## Introduction

In many problems in underwater acoustics, we are confronted with the task of detecting and classifying the signals present at the front end of the signal processor. These signals may arise from geophysical sources, marine life, or ambient background noise including wavebreaking and rain, or they may be man-made. Signals are often characterized by their frequency contents, e.g., the 20Hz signal bouts of finback whales, which are one-second-long frequency modulated signals centered around 20Hz, or the moans of bowhead whales, which are mostly tonal in the range of 25–900Hz.

We will model the wide variety of underwater acoustic signals as being the output of linear systems driven by a wideband noise signal—white noise. This is a common paradigm in many signal processing and communications problems. Signals from various origins will be modeled by systems with different parameters. Our parameters of choice are the poles and zeros of the system, which shape the spectrum of its output. Linear systems with both poles and zeros are usually termed ARMA (autoregressive moving-average) models.

This paper reports on an extensive study that compared eight different neural network learning techniques on the task of discriminating among ARMA models with very similar pole/zero configurations. Our major goals are (*i*) to evaluate the usefulness of neural networks for classifying ARMA models, (*ii*) to determine whether reflection coefficients are a useful set of features for classification purposes, and (*iii*) to compare the performance and training time of the neural networks tested.

## Signal Generation and Preprocessing

We tested two different configurations of linear systems [4]. Here we will report our findings for one of these configurations, illustrated by the pole-zero diagram in Figure 1.

*Formerly with LASIP, ECE, Carnegie Mellon University. This material is based upon work supported under a National Science Foundation Graduate Fellowship.

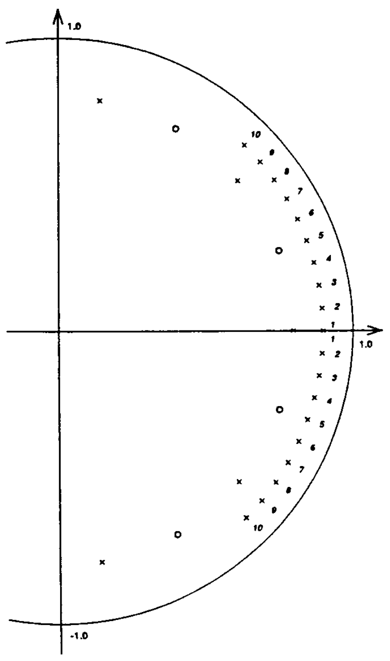


Figure 1: Pole-zero diagram for the systems.

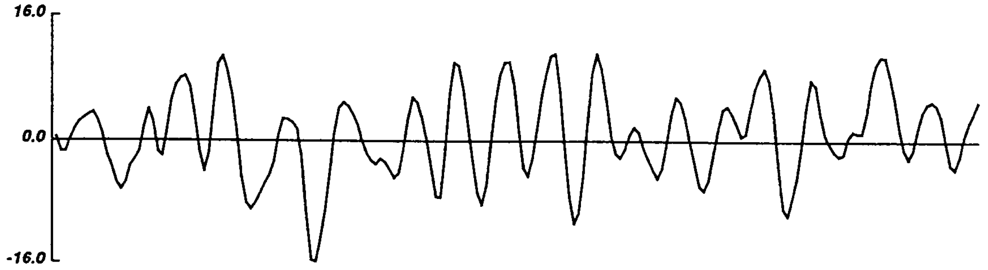
Each of the ten systems in this configuration has eight poles and four zeros. Six of the poles, and all of the zeros, are the same for all ten systems; all of these poles and zeros are located in conjugate pairs on a circle of radius 0.8. The remaining conjugate pairs of poles lie at a radius of 0.9 at intervals of five degrees from the real axis. System 1 has a pair of poles at $0.9\angle 0^\circ$, system 10 has poles at $0.9\angle \pm 45^\circ$, and the remaining systems are spread out in between. These poles, being closer to the unit circle, tend to dominate the response of the systems. In the figure, the variable poles are labeled with the number of the system to which they belong.

Since the input to each linear system is noise, the output is a random process whose power spectral density has been shaped by the response of the system. Figure 2 shows 200 samples of the signals produced by systems 8 and 9 with the same input noise supplied to each. The classifier must be able to distinguish similar signals such as these.

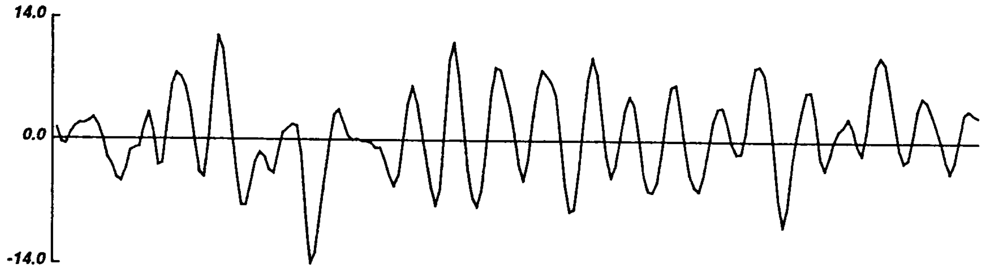
Preliminary experiments were conducted using the raw signal samples directly as inputs to the neural network classifiers. Extremely poor performance in classifying signals not in the training set was found. Therefore, all subsequent experiments used a preprocessing step to extract an invariant property of the signal.

This preprocessing consisted of two steps: feature extraction and scaling. The features employed in this study are the first twelve reflection coefficients, which were computed by the Burg algorithm [1] for sequences of 256 and 1024 signal samples. The quality of the estimates provided by the Burg technique increases monotonically with the number of samples used, and for the 256-sample sequences the estimated reflection coefficients are quite noisy. However, such short sequences often must be used in processing time-varying signals.

The reflection coefficients are intrinsically limited to the range $[-1, 1]$ and thus might seem suitable for input directly into the network, but in some cases the coefficients take on values in a very narrow range, e.g., $(-0.159, -0.018)$. We found that learning to respond to such inputs



(*a*) Output of system 8.



(*b*) Output of system 9.

Figure 2: Typical output signals from two representative systems

was extremely slow. Therefore, we found the minimum and maximum values of each reflection coefficient for our data and used these limits to scale each coefficient independently into the range [0, 1].

The preprocessed inputs are illustrated in Figure 3. In these diagrams, each row of twelve squares represents the average input vector derived from signals produced by the system indicated to the left; the area of each white square is proportional to the average value of the corresponding element of the vector.

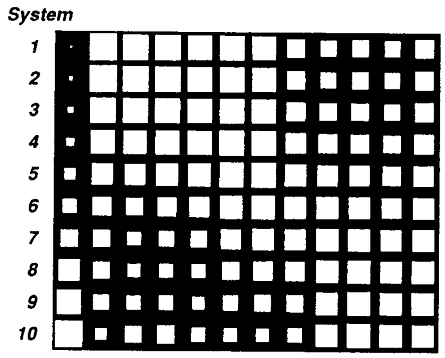


Figure 3: Average preprocessed input vectors.

## Neural Network Models

We considered eight feedforward networks having fixed architectures that are trained using supervised learning procedures; these are summarized in Table 1. In all networks tested, the

Table 1: Summary of network types tested.

| Symbol | Hidden Units | Learning Algorithm | Optimizations |
|---|---|---|---|
| SL-M | None | Gradient descent | Momentum |
| SL-A | None | Gradient descent | Learning rate adaptation |
| BP-M | Sigmoidal | Back propagation | Momentum |
| BP-A | Sigmoidal | Back propagation | Learning rate adaptation |
| QP | Sigmoidal | Quickprop | (Inherent in quickprop) |
| MKM | Gaussian | Gradient descent | Learning rate adaptation |
| RBF | Gaussian | Gradient descent | Learning rate adaptation |
| Gauss | Gaussian | Generalized error propagation | Learning rate adaptation |
| $k$-NN | Not applicable | | |

Note: MKM is the modified Kanerva method, RBF is the method of radial basis functions, and $k$-NN is the $k$-nearest-neighbor classifier.

output units had the familiar sigmoidal activation function. Several types also had sigmoidal hidden units, while three had Gaussian ellipsoidal hidden units, that is, hidden units exhibiting a Gaussian response centered around a point in the input space. Of these, the modified Kanerva method [5] and the method of radial basis functions [5] have fixed Gaussian parameters; they do not change during training, so simple gradient descent can be used for training the output layer weights. On the other hand, the third type uses *generalized error propagation* [5], a generalization of back propagation, to adjust the parameters of the Gaussian hidden units.

We explored two optimizations to the basic learning algorithms. *Momentum* is the addition of a smoothing term to the weight updating equation in order to reinforce weight changes that occur in a consistent direction through weight space. The other is *learning rate adaptation* of the style utilized in [5]. If the total squared error for all patterns in the training set decreased from one epoch to the next, then the learning rate is increased by a fixed factor (here 0.1%). Otherwise, the learning rate is reduced by a different factor (here 20%).

In addition, we tested Fahlman's quickprop [3], a heuristic second-order optimization technique somewhat similar to Newton's method. It is based on two assumptions about weight space: (1) that the weights can be adjusted independently, and (2) that if the error of the outputs is plotted as a function of any given weight, it will form an upward-opening parabola. At each epoch, quickprop attempts to adjust every weight to the value that would put it at the minimum of the error curve. Of course, the assumptions do not hold in typical networks, so quickprop remains an iterative technique.

As a benchmark for neural network performance, we also tested $k$-nearest-neighbor classifiers [2]. These are not neural networks, but rather a statistical classification technique that can be applied here. The classifier explicitly stores example patterns (i.e., the training set) and groups each new pattern into the category of the pattern in the set of examples that is closest to the new pattern according to some distance metric. A $k$-nearest-neighbor classifier chooses the category most commonly represented among the $k$ closest example patterns. Our nearest-neighbor classifiers used the Euclidean distance metric and stored as examples the same training set used in training the neural networks.

Table 2: Parameters used with each network type.

| Type | Hidden Units | Parameters |
|---|---|---|
| BP-A | 24 | $\eta = 0.01$ |
| BP-M | 24 | First 40 epochs: $\epsilon = 0.02$, $\alpha = 0$; thereafter: $\epsilon = 0.1$, $\alpha = 0.9$ |
| Gauss | 48 | $\eta = 0.05$ |
| QP | 24 | $r = 1.0$, $\epsilon = 0.001$ |
| MKM | 48 | $\eta = 0.1$ |
| RBF | 48 | $\eta = 0.1$ |
| SL-A | None | $\eta = 0.02$ |
| SL-M | None | First 40 epochs: $\epsilon = 0.02$, $\alpha = 0$; thereafter: $\epsilon = 0.5$, $\alpha = 0.9$ |

## Results

Each network we tested had ten output units, one for each possible classification of the input vector. The networks were trained using target vectors in which one element (corresponding to the correct classification) was 0.9, while the rest were 0.1. Likewise, we interpret the network's output vector according to a "best guess" criterion: the input falls into the category corresponding to the largest output value. This paper reports the percentage of input cases correctly classified according to the best guess criterion. We will refer to this figure as *classification performance.*

Our experiments used three data sets computed from independent sequences of signal samples. The network was trained on a *training set* which contained 50 examples for each system (500 in all). Periodically during training, the network's classification performance on a *testing set* of 50 examples per system was measured. When reporting the performance of the network, we give its classification performance on a third set, the *reporting set* (200 examples per system), for the epoch at which the testing set performance first reached its peak value. Training was continued after this peak only to ascertain that it was unlikely that any higher figure would be achieved.

This three-set method has the advantage that it reports a good estimate of the performance that could be expected on any set of inputs, because the decision to halt training is independent of the data set actually used in measuring the performance. A disadvantage of this method is that more labeled training data is required, which may pose a problem in situations where such data is difficult to obtain.

Table 2 details the parameters used for the eight types of networks studied. The hidden units, if any, were organized in a single layer of the indicated size. The network weights were initialized to random values in the range $(-r, r)$, where $r = 0.3$, except in the case of quickprop, where a value of $r = 1.0$ was used. In the cases where learning rate adaptation was used, the $\eta$ value (learning rate) listed is the initial value; $\eta$ will change as training proceeds. All networks that employed learning rate adaptation multiplied the learning rate by 1.001 if total error went down or by 0.8 if total error went up.

The other quickprop parameters not listed in the table were sigmoid prime offset = 0.1, momentum = 0, mode threshold = 0, $\mu = 1.75$, and weight decay factor = $-0.0001$. The hyperbolic error function was always used, as was "split epsilon" scaling of the learning rate (i.e., for each weight, $\epsilon$ was divided by the fan-in of the unit receiving an input via that weight).

The parameters used were found after extensive experimentation to be those that placed the

Table 3: Classification Performance results.

| Net | | Classification Performance (%) for $N_{\text{samp}} = 256$ | | | | Classification Performance (%) for $N_{\text{samp}} = 1024$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| Type | Runs | Average | S.D. | Min. | Max. | Average | S.D. | Min. | Max. |
| QP | 5 | 87.260 | 0.6804 | 86.450 | 88.050 | 98.980 | 0.1718 | 98.750 | 99.150 |
| Gauss | 1 | 85.900 | — | — | — | 96.850 | — | — | — |
| BP-M | 5 | 85.560 | 0.5237 | 85.150 | 86.250 | 97.080 | 0.2361 | 96.850 | 97.400 |
| BP-A | 1 | 81.750 | — | — | — | 89.650 | — | — | — |
| 41-NN* | | | | | | 73.000 | — | — | — |
| 40-NN* | | | | | | 72.950 | — | — | — |
| 67-NN* | — | 51.200 | — | — | — | | | | |
| 68-NN* | — | 51.100 | — | — | — | | | | |
| SL-M | 5 | 50.730 | 0.0274 | 50.700 | 50.750 | 55.430 | 0.5686 | 54.950 | 56.050 |
| SL-A | 5 | 44.680 | 0.1396 | 44.550 | 44.850 | 53.290 | 0.2535 | 53.050 | 53.650 |
| 1-NN* | — | 44.350 | — | — | — | 66.800 | — | — | — |
| MKM | 2 | 27.375 | 1.1667 | 26.550 | 28.200 | 37.975 | 1.0960 | 37.200 | 38.750 |
| RBF | 3 | 26.100 | 0.1803 | 25.950 | 26.300 | 47.917 | 2.0207 | 45.750 | 49.750 |

* The results for the $k$-nearest-neighbor classifier are reported for $k = 1$ and for the two values of $k$ for which the performance was highest.

networks on a relatively equal footing.

**Signal Classification Results** Table 3 presents the performance results we obtained. Results are reported for two values of $N_{\text{samp}}$, the number of signal samples in each input case. The amount of noise in the inputs to the network increases with decreasing $N_{\text{samp}}$. This is reflected in the results, which show that all the classifiers achieve better results for $N_{\text{samp}} = 1024$ than for $N_{\text{samp}} = 256$. The noisier cases are better tests of generalization because the testing data bears less resemblance to the training data.

These results show that the networks containing hidden units (except MKM and RBF) yield the best classification performance overall. Neither networks with weighted-sum hidden units nor networks with Gaussian ellipsoidal units seem to have a decisive advantage over the other, given that the right learning algorithm is applied. The poor performance of MKM and RBF networks could be due to a poor choice of the number of hidden units, since an exhaustive search to find the optimum number could not be undertaken.

One surprise here is that in the noisier cases the values of $k$ found to yield the best results from $k$-nearest-neighbor classifiers were quite high, and the performance for $k = 1$ was always significantly lower. In case of $N_{\text{samp}} = 256$, the best values of $k$ exceeded the number of training cases present for each linear system. The fact that a large $k$ may be needed for best performance further increases the high computational expense of nearest-neighbor classifiers.

**Learning Time Comparisons** Table 4 presents learning time results. We find that the learning rate adaptation scheme we used gave disappointing results. Among networks that achieved good performance, the longest training times were for networks using learning rate adaptation. Back

Table 4: Learning time results.

| Net | | Training Time (epochs) for $N_{\text{samp}} = 256$ | | | | Training Time (epochs) for $N_{\text{samp}} = 1024$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| Type | Runs | Average | S.D. | Min. | Max. | Average | S.D. | Min. | Max. |
| SL-A | 5 | 1450.0 | 157.32 | 1360 | 1730 | 964.0 | 190.87 | 720 | 1250 |
| QP | 5 | 2332.0 | 632.31 | 1390 | 3090 | 4994.0 | 1334.46 | 3780 | 6880 |
| BP-M | 5 | 5800.0 | 1903.54 | 3700 | 7880 | 5310.0 | 1484.17 | 3810 | 7770 |
| RBF | 3 | 9876.7 | 1778.32 | 7860 | 11220 | 10306.7 | 6161.30 | 4500 | 16770 |
| SL-M | 5 | 13242.0 | 16.43 | 13220 | 13260 | 92.4 | 15.71 | 76 | 114 |
| BP-A | 1 | 22970.0 | — | — | — | 9290.0 | — | — | — |
| MKM | 2 | 24220.0 | 1909.19 | 22870 | 25570 | 15570.0 | 5260.87 | 11850 | 19290 |
| Gauss | 1 | 45760.0 | — | — | — | 52530.0 | — | — | — |

propagation with momentum beat back propagation with learning rate adaptation. Apparently, the adaptation scheme's reduction of the learning rate the whenever the total squared error of the outputs increases often causes a very small average learning rate, leading to long learning times. However, we were unable to achieve better results using a fixed learning rate unless momentum was used.

We must caution that we found learning time to be highly dependent on the specific classification problem at hand. Results of an investigation of a different configuration of linear systems [4] show Gaussian node networks, here the slowest model, to be the fastest. Furthermore, here quickprop is faster than any other training method for networks with hidden units; for the other configuration, it was slower than either form of back propagation. Further study is needed to clarify what aspects of a classification problem make it difficult or easy for a network model to learn.

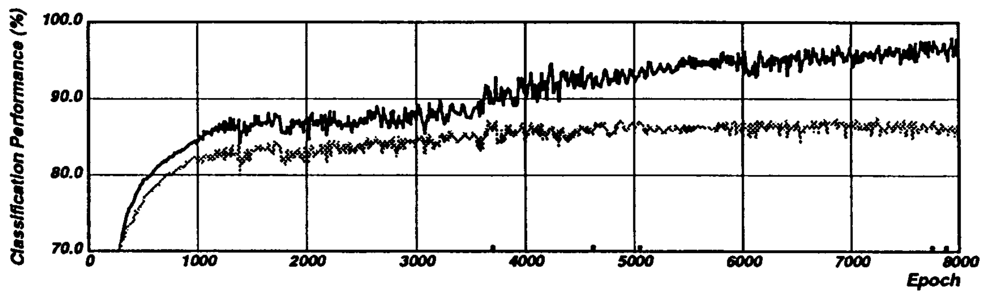
**Performance during Training** This section presents two examples of the performance of the networks during the course of training. Figure 4 shows performance during training for back propagation with momentum and the method of radial basis functions with $N_{\text{samp}} = 256$. Note that the scales of the graphs are different in order to show as much detail as possible.

Each curve is the average of several trials. It is reasonable to average the curves because the trends in performance are similar from one trial to the next, and the average curves are easier to understand than multiple curves or multiple graphs. In each graph, the black curve represents training set classification performance, while the gray curve shows testing set classification performance. Each dot along the horizontal axis indicates the epoch at which one trial reached peak performance on the testing set.

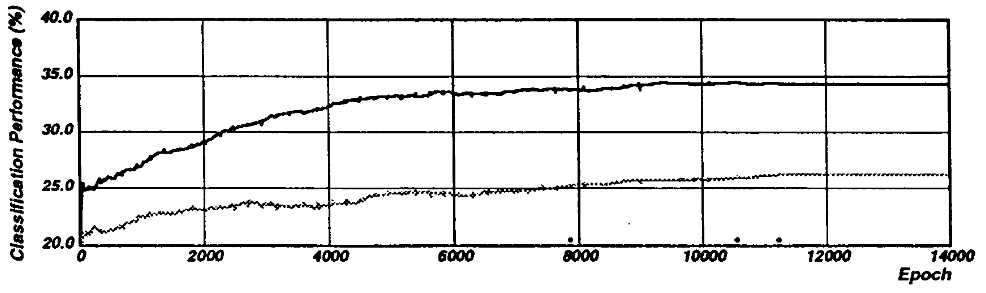
## Conclusions

This project has demonstrated that neural networks can be used successfully for signal classification. By modeling the signals to be classified as the output of ARMA linear systems driven by white noise, a wide range of underwater acoustic signals is included.

Reflection coefficients were found to be a useful set of features for the signals studied. The best overall performance was exhibited by networks with weighted-sum hidden units, trained by back propagation with momentum or by quickprop, and by networks with Gaussian ellipsoidal units,



(a) BP-M (Average of 5 trials)



(b) RBF (Average of 3 trials)

Figure 4: Performance during training for two typical network models.

trained by generalized error propagation. Their performance was generally better than that of networks with no hidden units and of $k$-nearest-neighbor classifiers.

## References

[1] J. P. Burg. *Maximum Entropy Spectral Analysis.* PhD thesis, Stanford University, May 1975.

[2] T. M. Cover and P. E. Hart, Nearest neighbor pattern classification. *IEEE Transactions on Information Theory,* IT-13(1):21–27, January 1967.

[3] Scott E. Fahlman. Faster-learning variations on back-propagation: an empirical study. In *Proceedings, 1988 Connectionist Models Summer School,* pages 38–51. Morgan Kaufmann, 1988.

[4] P. G. McKee, *Neural Networks for Linear Signal Classification.* LASIP Technical Report, Electrical and Computer Engineering, Carnegie Mellon University, December 1989.

[5] A. J. Robinson, M. Niranjan, and F. Fallside. *Generalising the Nodes of the Error Propagation Network.* Technical Report CUED/F-INFENG/TR.25, Cambridge University Engineering Department, Cambridge, England, November 1988.

[6] Robert G. Simpson. A decision-theoretic performance benchmark for neural networks trained to discriminate two autoregressive processes. In *Proceedings, 1988 IEEE International Conference of Acoustics, Speech, and Signal Processing,* pages 2148–2151, 1988.