Feature Selection Using a Multilayer Perceptron*

Dennis W. Ruck Steven K. Rogers Matthew Kabrisky

Department of Electrical and Computer Engineering
Air Force Institute of Technology
AFIT/ENG, Wright-Patterson AFB, Ohio 45433-6503

7 November 1989

Published in *Journal of Neural Network Computing*, Volume 2, Number 2, 1990, pp 40-48.

Abstract

The problem of selecting the best set of features for target recognition using a multilayer perceptron is addressed in this paper. A technique has been developed which analyzes the weights in a multilayer perceptron to determine which features the network finds important and which are unimportant. A brief introduction to the use of multilayer perceptrons for classification and the training rules available is followed by the mathematical development of the saliency measure for multilayer perceptrons. The technique is applied to two different image databases and is found to be consistent with statistical techniques and independent of the network initial conditions. The saliency measure is then used to compare the results of two different training rules on a target recognition problem.

1 Introduction

Recently there has been a great deal of interest in the use of multilayer perceptrons as classifiers in pattern recognition problems (see, for example, [11]). Unfortunately, little work has been accomplished to properly characterize their performance in such an application. As an example consider vehicle recognition. Initially a large set of features is computed for each of the vehicles in the database. The features might include length-to-width ratio, average temperature, contrast ratio, *etc*. Some of these features will provide a great deal of discrimination for this vehicle recognition problem while others will be essentially useless providing no discrimination. The features providing no discrimination are useless inputs to the classifier. One question of interest is how does a multilayer perceptron handle such useless inputs? Another question to be addressed is how do solutions arrived at by different training rules compare?

*This work was supported in part by the Rome Air Development Center (RADC/COTC) and the Wright Research and Development Center (WRDC/AARA).

Requests for reprints should be directed to the above address.

The question of what the multilayer perceptron does with useless inputs leads directly to the larger question of what features should be used in a classification problem. It is generally desirable to reduce the number of features that are used while still maintaining adequate classification accuracy. The problem then is one of selecting the *best* subset of features to use for classification. For example, suppose one wants to discriminate targets from non-targets where targets are tanks, trucks, and armored personnel carriers (APCs). Depending on what sensor is used to view the scene a variety of features could be used such as length-to-width ratio, brightness, number of pixels on the border to total pixels, and contrast ratio. Which of these features will be useful for discriminating targets from non-targets? The technique described in this paper answers these questions.

In the next section, a brief review of the classification problem is provided as well as a review of how a multilayer perceptron can be applied to classification. Also, the training rules available are described. In Section 3, the technique for feature selection is developed while Section 4 provides details of the tests performed to verify the consistency and usefulness of the approach. The following section applies this technique to the comparison of two training rules. The final section summarizes the benefits of this technique.

2 Background

In this section, the classification problem will be introduced followed by a discussion of how multilayer perceptrons can be applied to it. Also, two training rules for multilayer perceptrons will be briefly discussed. The general classification problem consists of extracting a set of features from the object which is to be classified and then processing this set of features to determine the class of the object. In image recognition, the features might be related to the shape of the object such as length-to-width ratio, absolute size, etc. The set of features for a given object can be considered a vector. Hence, the feature vectors are processed to determine the class of each object. Typically, a set of feature vectors whose classification is already known is used to design and test the classifier. As stated previously, it is not always known which features will provide the best recognition. Also, it is desirable to minimize the number of features to make the classifier faster and also more accurate because as the number of features increases the size of the design set must also be increased [4, 6]. Like the traveling salesman problem, the selection of the optimal subset of features from a larger set for best recognition suffers from a combinatoric explosion and is in general NP-complete [3].

A multilayer perceptron can be used to perform classification. Consider the network shown in Figure 1. Each node in the network performs a simple function as shown in Figure 2. The input to this network is the feature vector extracted from the object to be classified, and the output is typically a block code where one output is high indicating the class of the object and all other outputs are low. The weights connecting the nodes are determined using some training rule with a set of feature vectors, the training set. The network shown uses two hidden layers. Cybenko has shown that at most one hidden layer is required for approximating functions [1, 2]; however, experience has shown that a two hidden layer network will train more quickly than a one hidden layer network on some classification problems. The number of layers used is problem dependent, as is the number of nodes in each hidden layer.

A variety of training rules have been developed for setting the weights. The most popular rule to date is back propagation which was originally developed by Paul Werbos [17], rediscovered by

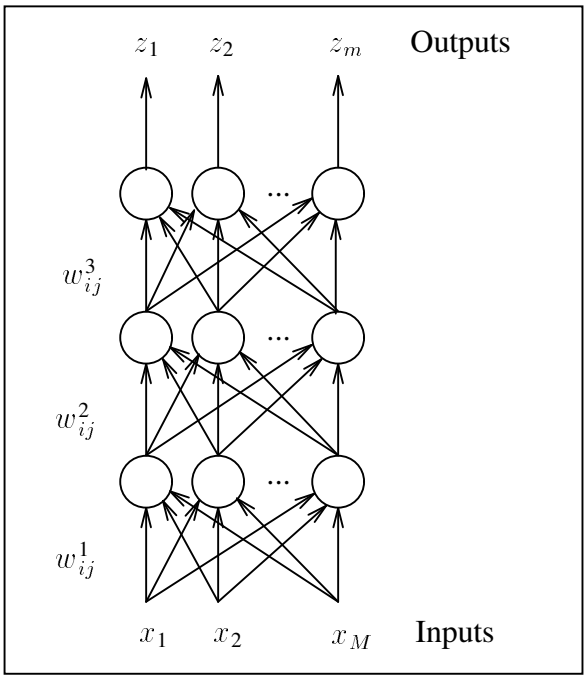

Figure 1: Multilayer Perceptron. Superscripts are used to denote layers. Thus, x_i^j is the output of node i in layer j, and w_{ij}^k is the weight connecting node i in layer $k-1$ to node j in layer k. Layer 1 is the first hidden layer, and the inputs can be considered Layer 0.

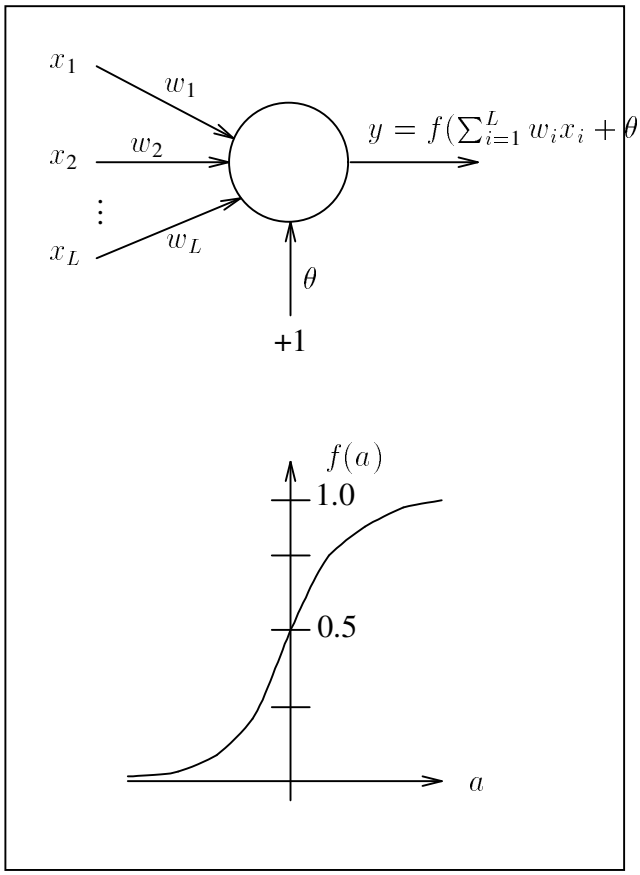

Figure 2: Function of Network Node

David Parker [7], and popularized by David Rumelhart [14]. The technique is based on gradient descent in the weight space over an error surface created by the sum of the squared error at each output node over the entire training set. Another rule for adjusting the weights is the extended Kalman filtering algorithm initially proposed by Singhal and Wu [16]. The Kalman filtering approach considers finding the weight values in the network to be an estimation problem. The goal of a Kalman filter is to estimate the state of a system based on observations taken over time. In the multilayer perceptron case, the state of the system being estimated is the weights in the network. The observations are the outputs of the network as training vectors are presented. The Kalman filter is based on a linear model for the system and observations which are linearly related to the state of the system. Since the multilayer perceptron is a nonlinear network, an extended Kalman filter is used. The extended Kalman filter uses a linearized model of the nonlinear system, and the model is relinearized after each update in the estimated state of the system (that is, whenever the weights are updated). This algorithm can find good weight values with a smaller number of training vector presentations but suffers from high computational loading compared to back propagation. There are many other methods for setting the weights which will not be discussed here (see, for example, [15]). However, a question often raised when a new training method is proposed is how does it differ from the others? The authors have previously shown that back propagation can be considered a degenerate form of extended Kalman filtering [13]. That is, when a few simplifying assumptions are made in the extended Kalman filtering algorithm, the resultant algorithm is back propagation. The consequences of these assumptions is that back propagation discards a great deal of information which can be effectively used to update the weights during training. As a result, the extended Kalman filtering algorithm can achieve higher classification accuracy than back propagation in the same number of epochs (an epoch is one pass through the training set). The question of interest here is does the extended Kalman filtering training method use the input features differently than back propagation?

To answer the questions of which features are useful for classification, can a multilayer perceptron learn to ignore useless input features, and how does back propagation compare to extended Kalman filtering in its use of features for classification, the following network analysis technique was developed.

3 Network Analysis

The network analysis technique developed to answer these questions examines the responsiveness of the network's outputs. The sensitivity of the network outputs to its inputs is used to rank the input features' usefulness. First, an expression for the derivative of an output with respect to a given input will be derived; then it will be shown how this can be used to create a measure of the sensitivity of a trained multilayer perceptron to each input feature. The following section will examine this measure for consistency and utility.

The notation to be used throughout is as follows. Superscripts always represent a layer index and not a quantity raised to a power. The layers are counted from the first layer of nodes which compute the sigmoid of a weighted sum. Thus, Layer 1 is the first hidden layer nodes and not the inputs. The output of node i in Layer j is denoted by x_i^j. Input i is represented by x_i with no superscript, and output i is represented by z_i. For the weights, the first subscript denotes the source node, and the second denotes the destination. The superscript on weights represents the layer of

the destination node. Hence, w_{ij}^k is the weight connecting node i in layer $k-1$ to node j in layer k.

Consider again the network in Figure 1. It is desired to calculate the derivative of output z_i with respect to input x_j. Suppose each of the nodes in the network performs a weighted sum of its inputs plus a threshold term and puts the result through a sigmoid as shown in Figure 2. To compute the desired derivative, all that is required is some simple calculus and the chain rule. Using the fact that the derivative of the sigmoid is the output times one minus the output yields,

$$\frac{\partial z_i}{\partial x_j} = z_i(1-z_i)\frac{\partial}{\partial x_j}(a_i^3) \tag{1}$$

where a_i^3 is the activation of node i in layer 3. Activation is the weighted sum of the inputs plus the node threshold. Substituting the expression for the activation gives

$$\frac{\partial z_i}{\partial x_j} = z_i(1-z_i)\frac{\partial}{\partial x_j}(\sum_m w_{mi}^3 x_m^2 + \theta_i^3) \tag{2}$$

where w_{mi}^3 is the weight connecting node m in the second hidden layer to node i in layer 3, x_m^2 is the output of node m in layer 2, and θ_i^3 is the threshold associated with node i in layer 3. Hence, the summation is over all nodes in layer 2. Applying the derivative to this expression for the activation produces

$$\frac{\partial z_i}{\partial x_j} = z_i(1-z_i)\sum_m w_{mi}^3 x_m^2(1-x_m^2)\frac{\partial}{\partial x_j}(a_m^2) \tag{3}$$

where a_m^2 is the activation of node m in layer 2. Let $\delta_i^3 = z_i(1-z_i)$. Then

$$\frac{\partial z_i}{\partial x_j} = \delta_i^3 \sum_m w_{mi}^3 x_m^2(1-x_m^2)\frac{\partial}{\partial x_j}(a_m^2) \tag{4}$$

Continuing the process through two more layers yields

$$\frac{\partial z_i}{\partial x_j} = \delta_i^3 \sum_m w_{mi}^3 \delta_m^2 \sum_n w_{nm}^2 \delta_n^1 w_{jn}^1 \tag{5}$$

where

$$\delta_m^2 = x_m^2(1-x_m^2) \tag{6}$$
$$\delta_n^1 = x_n^1(1-x_n^1) \tag{7}$$

and x_n^1 is the output of node n in layer 1. Note that the derivative of the output with respect to the input (Equation 5) depends not only on the weights in the network, the w_{ij}, but also on the current outputs of the nodes in the network, the x_l^k and the network outputs, z_i. Thus, the derivative depends on the current input to the network as well as the network weights.

The dependence of the derivative of the outputs with respect to the inputs on the current input to the network forces the evaluation of the derivative at a set of points in the input space. Ideally, each input would be independently sampled over its expected range of values. Suppose R points were used for each input. Then the total number of points the derivatives would be evaluated at would be R^m where m is the number of inputs. As an example, suppose 22 features were being evaluated, and 20 points in the feature space were sampled over the expected range of each feature. In this

case, $R = 20$ and $m = 22$, so the number of points at which the derivative must be evaluated is approximately 10^{28}. If 10^6 derivative evaluations could be performed each second, it would take about 10^{15} years to evaluate all the derivatives. Even for this small example the number of computations is intractable. Obviously, as the number of inputs grows the number of derivative evaluations increases tremendously. In fact, this is an NP-complete problem. Since the input space cannot be sampled uniformly, it is desired to sample it at the *important* points. The points of greatest importance in the input space are those where training data exists; hence, the training vectors are used as starting points to sample the input space. For every training vector, each input is sampled over its range while the other inputs are determined by the training vector currently being evaluated. Suppose there are p training vectors, then the number of derivative evaluations is pmR. Continuing the above example, suppose 500 training vectors are available. The number of derivative evaluations is 220,000 which would take about one-fifth of a second to compute. Now the number of evaluations increases linearly with the number of inputs which is tractable.

A measure of the *saliency* of an input can now be formulated as follows. Let Λ_j represent the saliency of input j.

$$\Lambda_j = \sum_{\mathbf{x} \in \mathbf{S}} \sum_i \sum_{x_j \in D_j} \left| \frac{\partial z_i}{\partial x_j}(\mathbf{x}, \mathbf{w}) \right| \tag{8}$$

where **x** indicates the m network inputs, **S** is the set of p training vectors, **w** represents the weights in the network, D_j represents the set of R points for input x_j which will be sampled. Normally, D_j is a set of uniformly spaced points covering the expected range of input x_j. Note that the dependence of the derivative of the output, z_i, with respect to the input, x_j, on the weights, **w**, and the input to the network, **x** is explicitly shown in Equation 8. Two questions now need to be asked. Is this measure consistent with existing statistical techniques and independent of network initial conditions? Specifically, does the ranking of the of the Λ_j actually correspond to the importance of the inputs as determined by the single feature minimum probability of error method? Secondly, will the ranking of the Λ_j be independent of the initial network conditions? The next section will address these questions.

4 Test of the Feature Selection Rule

In this section, the consistency of the saliency measure developed in the previous section will be examined followed by a test of the actual utility of the measure. The first test consisted of training 100 networks on a set of image recognition data. The classification problem consisted of using a set of 22 moment invariants computed from objects segmented from doppler imagery to determine the type of object. These features were chosen for this recognition task because they have previously been shown to be effective for aircraft recognition using conventional techniques [5]. There were four object classes: tank, jeep, 2.5-ton truck, and oil tanker truck. The pq ordinary moment of an object is an integral over the object of the factor $x^p y^q$. That is,

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) x^p y^q dx dy \tag{9}$$

where M_{pq} is the pq ordinary moment of the function $g(x, y)$. In this case, the function $g(x, y)$ simply represents the silhouette of the segmented objects. These moments are then scaled to provide invariance to changes in either the position or size of the object. Moment invariants

are created by combining individual scaled moments so that the resulting feature is invariant to changes in rotation of the object in the viewing plane. Thus, the moment invariants are unchanged by differences in position, scale, or in-plane rotation of an object. See [12] for more information on this recognition problem.

Each network was started with a different set of random weights. The order of presentation of the training data was different for each training session. After each network was trained, the saliency values for each input, Λ_j, were calculated. For each network, the rank of every input was computed from the sorted ordering of the saliency measure. The input with the lowest Λ_j, indicating the least sensitivity of the output to that input, was ranked zero; and the input with the highest was ranked number 21. Next a histogram was computed for each input using the 100 different networks indicating the number of times the input had a given rank. Ideally, the rank of a given input would be independent of the network's initial weights and presentation of the training data. In that case, the histogram for an input would be a single spike at the rank of the input. Figure 3 shows the histograms for three of the 22 input moment invariants. The three features selected are representative of the type of distributions observed for all the features. The distribution about rank 21 represents a feature that was consistently determined to be important using Equation 8. The distribution spread across the middle rank values represents a feature which does not provide significantly more information than approximately 10 of the other features because the median of the distribution is near rank 11. Finally, the distribution that is dominated by low values represents a feature that is consistently found to be unimportant independent of the network initial conditions. It is possible from the histograms to produce an overall ranking of the saliency of the inputs. Alternatively, the saliency measure can be averaged for a given feature over all networks trained. The average saliencies can then be ranked to yield an overall ranking for the input features. This approach yields nearly identical results to the histogram method. As a preliminary test of the utility of the measure, the top three features and their associated weights were eliminated from a trained network and the classification accuracy dropped from 85 percent to below 50 percent. Also, the bottom three features and their weights were eliminated from a trained network which resulted in no degradation of classification accuracy.

The next question to be answered is whether or not this saliency measure is useful. As previously stated, a preliminary test indicated that it would be; hence, a more rigorous test was devised. In this test, the problem was to identify targets from non-targets in forward looking infrared (FLIR) images where targets consisted of tanks, trucks and armored personnel carriers (APCs). A set of nine features were extracted for this purpose. The features chosen have previously been shown to be effective using conventional techniques [8, 9, 10]. The features are listed in Table 1. These features were ranked using a probability of error criterion. That is, each feature was used individually to determine whether the object was a target or not and the probability of error was calculated. The feature with the lowest probability of error was given the highest rank and *vice versa*. This ranking provided the baseline for evaluation of the utility of the saliency measure developed.

As with the consistency test, 100 networks were trained on the same data with different initial weights and a different order of presentation of the training data. Histograms of the ranking of three input features were constructed and are shown in Figure 4. The three features plotted are representative distributions. The ranking of the features by the probability of error criterion and the saliency measure is shown in Table 2. Note that the two methods of ranking the data are in agreement for the top two and bottom two features. As a further test of the validity of the saliency measure, the accuracy of classifiers designed using the top three features from each ranking were

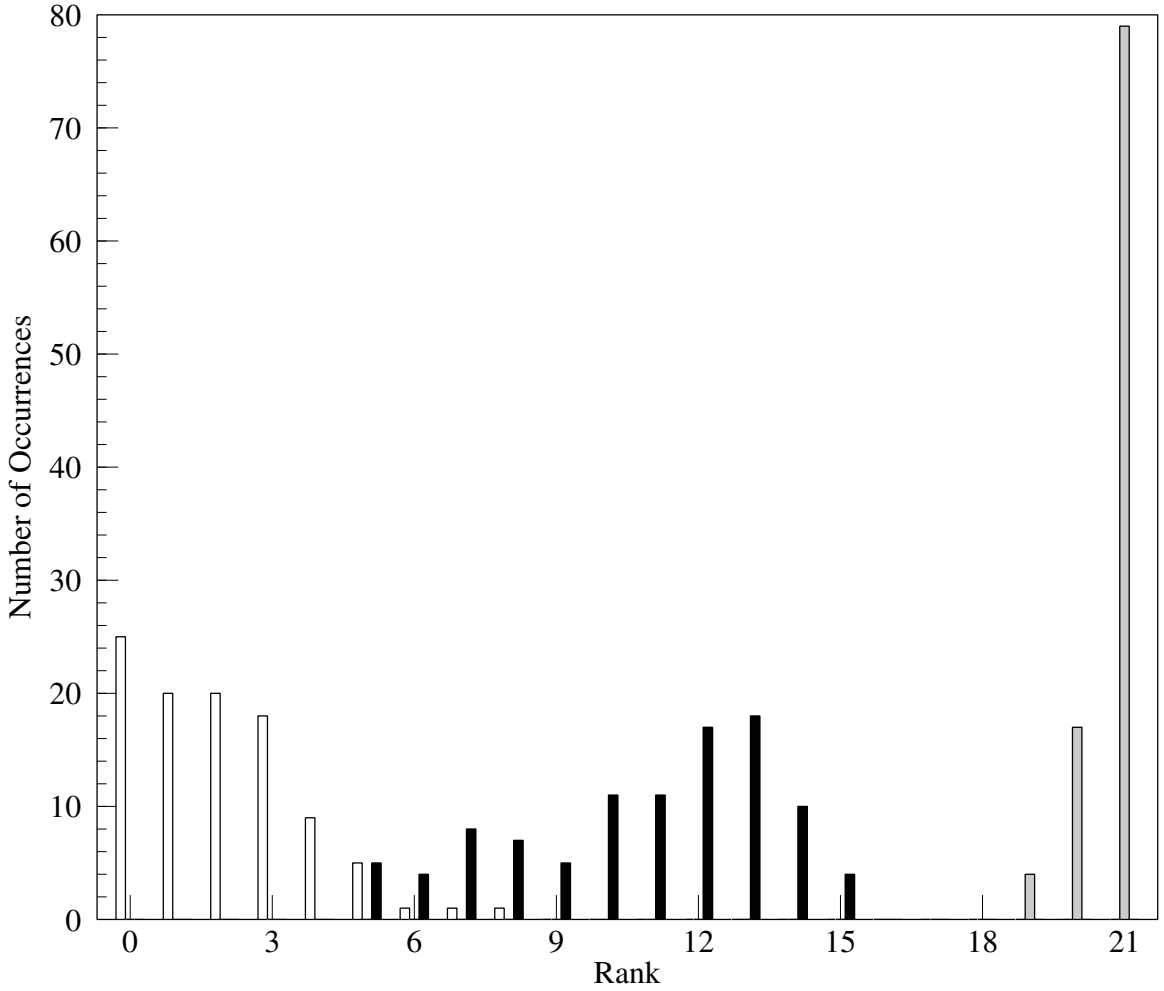

Figure 3: Rank Histograms for Three Moment Invariants. Each feature is displayed using a different fill pattern. The moment invariant feature represented by the lightly shaded bars ranging in rank from 0 to 8 shows very little saliency, meaning that this feature provided little information in the pattern classification process. The darkly shaded bars ranging in rank from 5 to 15 represent a moment invariant feature which in most cases provided less information for classification than approximately ten other input features. The feature represented by the medium shaded bars ranging from rank 19 to 21 has a great deal of saliency and provided much information in classifying the input patterns.

Table 1: FLIR Features Evaluated

Feature	Description
Complexity	Ratio of border pixels to total object pixels
Length/Width	Ratio of object length to width
Mean Contrast	Contrast ratio of object's mean to local background mean
Maximum Brightness	Maximum brightness on object
Contrast Ratio	Contrast ratio of object's highest pixel to its lowest
Difference of Means	Difference of object and local background means
Standard Deviation	Standard deviation of pixel values on object
Ratio Bright Pixels/Total Pixels	Ratio of number of pixels on object within 10% of maximum brightness to total object pixels
Compactness	Ratio of number of pixels on object to number of pixels in rectangle which bounds object

compared. Table 3 shows these results. Note that the differences between the top three of either ranking for all three types of classifiers are negligible. Hence, the saliency measure does provide a useful measure of the significance of input features.

5 Application to Training Rule Evaluation

In this section, the saliency measure will be used to compare two training rules for multilayer perceptrons. As stated in Section 2, there are several training rules available for determining the weights in a multilayer perceptron. Two will be compared here. The traditional back propagation and the extended Kalman filtering approaches will be considered. The question to be considered is whether or not the Kalman approach would use the features in a different way than the back propagation approach. That is, would the importance of the inputs for back propagation differ significantly from those for extended Kalman filtering?

The doppler image data with the four classes, which was used to test the consistency of the saliency measure, provided the test data for this question. One hundred networks were trained with the extended Kalman filtering approach using the same data, initial weight values, and presentation order of training vectors as the 100 which were trained using back propagation. Thus, the only difference between the two sets of 100 networks was the training method employed. Figure 5 shows the histograms of the ranking of the same three input features used as examples in Figure 3. Compare these with the histograms in Figure 3. Note that while there are some differences between the two, the ranking of the features is mostly the same between the two training methods. Hence, the two training rules place equal relative emphases on the input features.

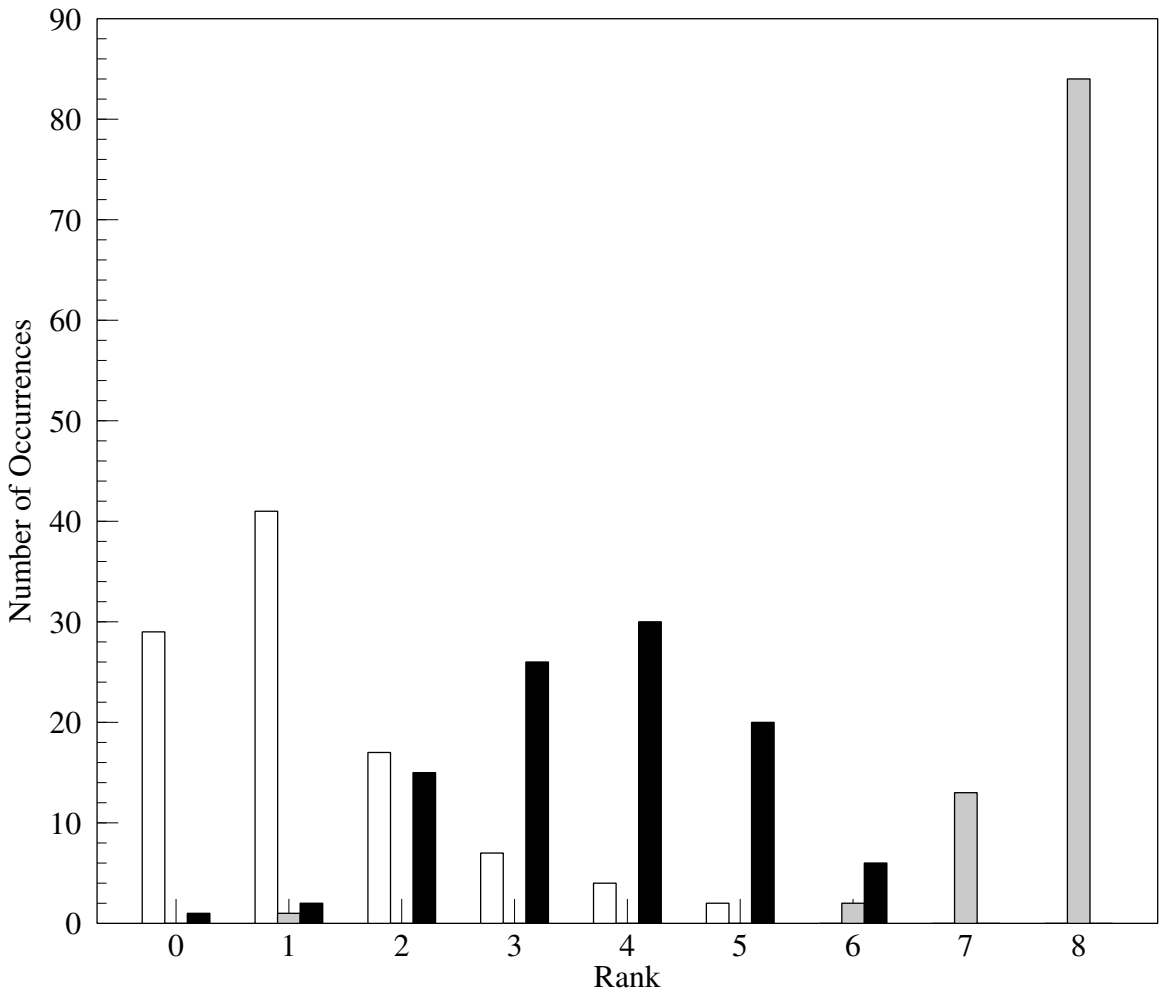

Figure 4: Rank Histograms for Three FLIR Imagery Features. Each feature is plotted using a different fill pattern. The FLIR feature represented by the unfilled bars shows little saliency and, thus, did not provide much information for classifying the input patterns. The darkly shaded bars represent a FLIR feature with a moderate degree of saliency providing some information for classification. The feature shown with the lightly shaded bars has a large saliency and provided much information for classifying the FLIR objects.

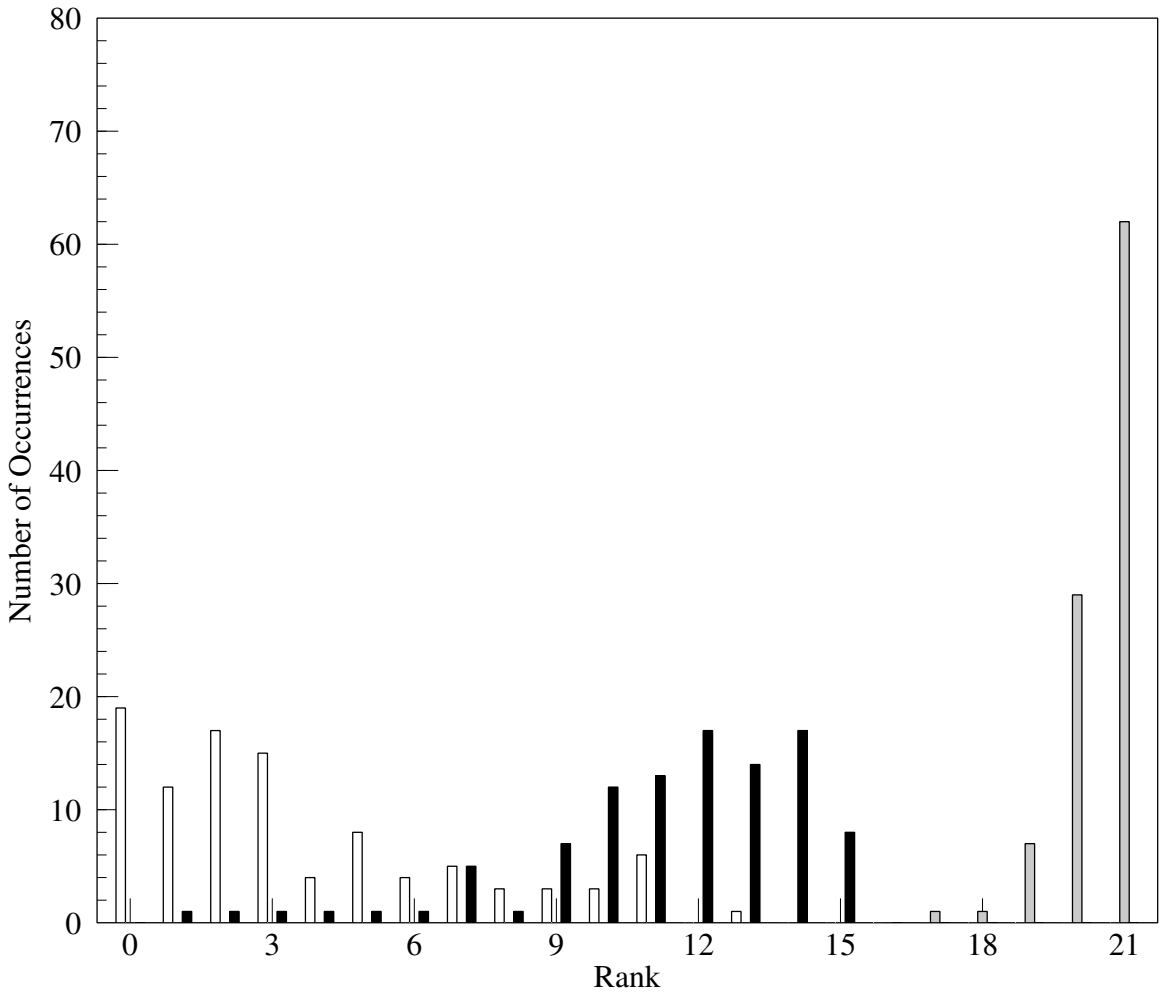

Figure 5: Rank Histograms for Three Moment Invariants Using Kalman Approach. The same three features are displayed as those in Figure 3 using the same fill patterns. Comparing each of the three histograms with those in Figure 3 shows that the Kalman approach and back propagation place equal relative emphases on the input features.

Table 2: Ranking of FLIR Features

Feature	P_e Criterion Rank	Saliency Measure Rank
Complexity	8	8
Length/Width	7	7
Mean Contrast	6	3
Maximum Brightness	5	2
Contrast Ratio	4	4
Difference of Means	3	6
Standard Deviation	2	5
Ratio Bright Pixels/Total Pixels	1	0
Compactness	0	1

Table 3: Classification Accuracies of Two Feature Subsets

Classifier	Best 3 P_e Features	Best 3 Saliency Measure Features
Bayesian w/ Parzen Windows	92.5%	93.4%
Nearest Neighbor	92.7%	92.5%
Multilayer Perceptron	90.7%	90.2%

6 Conclusion

In this paper, a new technique for ranking the importance of features for a multilayer perceptron has been developed. The saliency measure has been shown empirically to be both consistent and useful. When compared with the traditional method of ranking input features by the probability of error criterion, it performed as well and resulted in a similar ranking for the input features. Also, back propagation was compared to extended Kalman filtering using the saliency measure. It was found that both methods for setting the weights in a multilayer perceptron place equal relative emphases on the features.

One extension to the saliency measure is its application to the hidden nodes in a multilayer perceptron. The outputs of the hidden nodes are inputs to the following layer; hence, the saliency of the hidden nodes can also be computed. On-going research into this area will, hopefully, provide a method for automatically sizing the hidden layers in a multilayer perceptron.

References

[1] BARRON, A. R. Statistical properties of artificial neural networks. In *Proceedings of Conference on Decision and Control* (1989).

[2] CYBENKO, G. Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems* (1989).

[3] DEVIJVER, P. A., AND KITTLER, J. *Pattern Recognition: A Statistical Approach*. Prentice Hall International, London, 1982.

[4] DUDA, R. O., AND HART, P. E. *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York, 1973.

[5] DUDANI, S., BREEDING, K., AND MCGHEE, R. Aircraft identification by moment invariants. *IEEE Transactions on Computers C-26* (1977), 39–45.

[6] FOLEY, D. H. Considerations of sample and feature size. *IEEE Transactions on Information Theory IT-18* (September 1972), 618–626.

[7] PARKER, D. B. Learning-logic. Invention Report 581-64, Stanford University, Stanford, CA, October 1982.

[8] ROGGEMANN, M. C. *Multiple Sensor Fusion for Detecting Targets in FLIR and Range Images*. PhD thesis, Air Force Institute of Technology, June 1989.

[9] ROGGEMANN, M. C., MILLS, J., KABRISKY, M., ROGERS, S., AND TATMAN, J. An approach to multiple sensor target detection. In *Proceedings of SPIE Conference on Sensor Fusion* (1989), vol. 1100.

[10] ROGGEMANN, M. C., MILLS, J., KABRISKY, M., ROGERS, S., AND TATMAN, J. Multiple sensor tactical target detection in FLIR and range images. In *Proceedings of Tri-Service Data Fusion Symposium* (Johns Hopkins University, May 1989).

[11] RUCK, D. W., ROGERS, S. K., AND KABRISKY, M. Tactical target recognition: Conventional and neural network approaches. In *Proceedings of 5th Annual Aerospace Applications of Artificial Intelligence Conference* (October 1989), pp. 247–253.

[12] RUCK, D. W., ROGERS, S. K., KABRISKY, M., AND MILLS, J. P. Multisensor fusion target classification. In *Proceedings of SPIE Symposium on Optics, Electro-Optics, and Sensors* (1988), pp. 14–21.

[13] RUCK, D. W., ROGERS, S. K., MAYBECK, P. S., AND KABRISKY, M. Back propagation: A degenerate Kalman filter? *IEEE Transactions on Pattern Analysis and Machine Intelligence* (June 1989). Submitted for publication.

[14] RUMELHART, D. E., MCCLELLAND, J. L., AND THE PDP RESEARCH GROUP. *Parallel Distributed Processing*, vol. 1: Foundations. MIT Press, Cambridge, Mass., 1986.

[15] SIMPSON, P. K. *Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations*. Pergammon Press, Elmsford, New York, 1989.

[16] SINGHAL, S., AND WU, L. Training multilayer perceptrons with the extended Kalman algorithm. In *Advances in Neural Information Processing Systems 1* (1989), D. S. Touretzky, Ed., Morgan Kaufmann, pp. 133–140.

[17] WERBOS, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, Mass., 1974.