MULTI-SENSOR DATA FUSION USING NEURAL NETWORKS

D. Wade Fincher and Dwight F. Mix
Electrical Engineering Department
University of Arkansas
Fayetteville, AR 72701

ABSTRACT

In today's modern systems, multiple sensors are often used to acquire different types of data from the environment, which must then be fused to produce a single output. Fusion normally occurs at the output of each sensor, but early data fusion is better. A neural network provides a suboptimum but viable solution to this problem when other methods may be unknown.

We can optimize data fusion by Bayesian techniques at the output of each system if the necessary costs and probabilities are known (or assumed). Implementing Bayesian fusion earlier in systems is another matter, however. There are more signals or variables in earlier stages of a sensor, and we may not have sufficient information to implement this optimum fusion procedure except at the output. Thus it may be better to forego the optimum Bayes procedure at the system output in favor of using suboptimum neural networks earlier in the system. Our experimental results illustrate this point.

Introduction

In this paper we outline a general approach to the use of neural networks for data fusion. This approach applies to a variety of practical situations, including the fusion of multiple sensors in pattern recognition, robot navigation, and military environment assessment/evaluation.

We begin with examples of data fusion problems and follow up with a pattern recognition example. We discuss the differences between using post- and pre-detection signals and the advantages of using pre-detection signals. Finally we demonstrate how to apply a neural network to this problem and present experimental results for this example.

Our purpose is to provide insight into the data fusion problem, what it is, what post- and pre-detection fusion mean and to illustrate the use of a feed-forward neural network using the back-propagation training algorithm [1,2] in this application.

Multiple Sensors

There are many uses in modern systems for multiple sensor combination. One is in pattern recognition, where many sensors may be used to obtain different views or characteristics of the input pattern so a higher recognition rate is obtained. Another use for multiple sensors is in robotics. Here the information obtained from many inputs must be combined in a sensible manner to provide control information to the robot. Still another use is in military situation assessment. Here the various information systems on a military platform must be combined to present a coherent picture of the situation.

These are but a few of the applications of data fusion. Any system with multiple views of the same region of the world is a candidate for data fusion. It makes a great deal of difference in their performance where this fusion takes place, with earlier being better than late. It is better to fuse signals near the input rather than near the output. This can be shown using either an information theory argument or a probability of error viewpoint. The information in the source is better preserved for use in the decision making algorithm, and the probability of error is reduced when the signal nearest the input is used.

We will illustrate these concepts with a pattern recognition example. Any other multi-sensor system will have similar characteristics, and the extension of this example to these other systems should be straightforward. Suppose we wish to recognize zip codes as printed by hand or machine on an envelope. Since we are interested only in the pattern recognition segment of the overall system, we will assume that the numbers have been isolated, sized, and located in a 16 by 16 image for presentation to the recognition machine. Figure 1 illustrates a computer print out of four such numbers in a 16 by 16 grid. As you can see, there is a wide variety in the quality between these hand printed patterns and machine printed patterns. Thus any successful recognition scheme must necessarily look at several different characteristics of the input pattern. It is these different looks that give us the multi-sensor aspect of the problem.

In the following sections we will describe pattern recognition experiments which were conducted to show that it is not only possible, but advantageous to use neural networks for data fusion. First we describe the data used in the experiment, followed by a description of the sensors used, the neural network employed, and the results of the experiment.

Description of Data

One hundred samples of machine printed numerals in different fonts and one hundred samples of hand printed numerals constitute the data. Obtaining a 95-100 percent recognition rate on machine printed data is easily obtainable by various methods (template matching, etc.). For this reason a computer was utilized on the machine printed data to degrade its quality by various means (skewed character, missing pixels, etc.) so as to give data with as much difference as hand printed numerals. This data set is broken into a training set and a test set. The training set consists of the numbers from five different machine fonts (prestige, sans serif, etc.) plus the numbers from five different volunteers who provided us with the hand printed samples. Thus half of each data set is in the training set. The other half is in the test set. It consists of five more machine fonts plus the hand printed samples from five more volunteers. None of the data is duplicated, which means we have 200 different patterns in the total data set.

The purpose for breaking the data into a training set and a test set is to provide a check on a "real world" situation. The training set is used to train the neural network, a procedure which guarantees the system will correctly recognize all data in the training set. We then present data which the system has never seen from the test set, and check its performance on these patterns.

Now, let us identify and describe the three sensors used in this experiment. They are:

Quadrant Analysis

In this recognition algorithm the 16 by 16 character is divided up into 16 blocks, each 4 by 4. The number of "1's" or on bits in each block is then counted. This is a simple grey scale reader and returns 16 values, one for each block in the character. As an example let us look at the breakup and the data produced for the character "3" in Figure 1. Figure 2 shows the data for this character.

These values are then compared with a template of values obtained from a standard set of numbers. This produces a set of 10 numbers representing how close the character presented is to each template. The algorithm then picks the number which produced the closest match to the character analyzed.

Experimental results show that this recognition scheme produced 49% recognition on the test set of data.

Outside in Analysis

This algorithm goes around the outside of the character and determines how many pixels into the character the scanner must go before it encounters a "1", or on, pixel. The scan goes from the right side of the character to the left, the top to the bottom, the left side to the right side and the bottom to the top. This produces 64 values for each character. For example, the number "3" from Figure 1 would produce the following set of data shown in Fig. 3.

These 64 values are then compared to a template in much the same manner as the previous algorithm. The final decision of the algorithm is the value which most closely matched the presented value.

Experimental results showed that this algorithm alone would correctly recognize 44% of the test data.

Run Length Encoding

This final algorithm scans the character from left to right and from top to bottom and counts the number of transitions from "0" to "1", or off to on transitions for each row or column of the character. This algorithm produces 32 values, 16 horizontal counts and 16 vertical counts. For the same character "3" from Figure 1 the output would be as shown in Figure 4.

The 32 values produced are then matched to templates in the same manner as the previous two methods. The final decision is again made depending on the best match out of the template characters.

It was experimentally determined that this algorithm could correctly recognize 52% of the test data.

Rank Data Fusion

Different recognition schemes work better on different characters. With this in mind it is obvious that by combining the outputs of the three schemes, the system should be better able to accomplish the task of recognizing the character presented. The simplest way of combining the data is to rank the outputs of each algorithm. Each algorithm produces 10 output numbers which correspond to how well the character present at the input correlates with each of the ten templates. Figure 5 shows a block diagram of the three recognition schemes in this configuration.

These values can then be ranked in order of how well the match is for each template character. For example, the worst match would be assigned a rank value of 0, the second worst a value of 2, the third a value of 4, etc. These match values are then added up for each character (i.e. add all 3 values for 0, all 3 values for 1, etc.) and the final decision is the character with the highest total ranking.

This combination scheme was applied to the test data with the results being a recognition rate of 59%. This is an increase of 7% to 15% over the individual algorithms. This value is still low and must be improved if the recognition system is to be viable. Let us now examine how recognition rates can be improved by use of a neural network.

Declaration and Detection

In order to understand the different ways we use the neural networks in this example, it will be necessary to clarify two ideas: declaration and detection. The final decision made by any algorithm or the entire system is said to be a declaration. The comparison of the input signals with the stored templates is called detection. With these two ideas in mind, we define the following. The ten signals that each algorithm produces from the template comparison to allow the decision to be made are called the pre-declaration, or post-detection, signals. The data that the algorithm compares to the template (for example the number of on bits in each quadrant) is called the pre-detection signals. The use of both of these signals as inputs for the neural network will be covered in the following sections.

Neural Networks

One solution to the data fusion problem is to fuse data with neural networks. Since the training algorithms for these networks generally converge to the first workable solution with no effort to optimize, this provides a sub-optimum but practical solution. Neural networks have as their input a set of numbers, and have as there output a decision. Thus we can select the pre-declaration signals, or the pre-detection signals, or any combination of the two as input signals to a multi-layer feed-forward network. The output of this network is then a declaration. In essence, the neural network replaces the declaration units and/or the detector units of all sensors.

Let us now discuss those pertinent aspects of neural networks that are unique to our problem. It is not our purpose in this section to describe or introduce neural networks, but only to point out some features of our problem that are unique to the proper use of these networks. Figure 6 shows a three-layer feed-forward network which is suitable for our application. For illustrative purposes, suppose we wish to fuse the 10 post-detection values from each of the above sensors. These values are shown as x_1 through x_{30} in Figure 6.

Each node, shown as a small circle in Figure 6, consists of two parts, a summer and a non-linearity. Thus inside each node the sum of all input signals is supplied to a non-linearity whose purpose it is to keep the node output between -1 and +1. (Some networks use non-linearities with output values between 0 and 1, and some use the output range from -1 to +1. It seems to make no difference, so we have arbitrarily chosen the range from -1 to +1.) Thus it makes sense for two reasons to change all input data to first have a range from -1 to +1. The first reason is to make the input data compatible with all other data at level i and j, but the most important reason is to make each input sample equally important. If some data has values on the order of 1000, while other data has values on the order of 10, they assume importance values out of proportion unless their values are adjusted.

We use a three-level network, labeled k, i, and j in Figure 6. There are 30 nodes at level k, 10 nodes in the hidden layer at level i, and 10 output nodes at level j, one for each class. The output class (the network decision) is determined by the largest output value. The network is trained to have its largest output on the y-value corresponding to the known input class.

Post Detection Neural Network Data Fusion

For a recognition problem such as this, rank data fusion is not desired in that much information is lost in the ranking of the template matches. For example, if the input character is an eight, the template comparisons for eight and three will probably be very close. However, when the ranking is completed, these two values will be separated by as much distance as all other values since it is not the value but its rank which determines its importance. To improve upon ranking, let us use a neural network for the fusion of the post-detection signals.

When post-detection signals are used, the neural network will simply replace the rank combination scheme used above. The 30 outputs produced by the recognition schemes are used as the input signals and the network bases its decision on all aspects of the applied data such as how close one value is to another, the importance of certain values, etc.

The neural network to which these values are supplied has 30 input nodes, 10 hidden nodes, and 10 output nodes. Training occurs on the training set until all weights in the network stabilize and testing then is performed on all 200 patterns, with the following results.

Training set: no errors (as expected).
Test set: 73% correct.
Overall result: 86% correct.

These results show a vast improvement over the rank combination technique. The question arises about how much improvement is possible, and the answer is provided by the optimum Bayes fusion technique [3].

Bayesian Data Fusion

This fusion technique requires that much information be known about the system, specifically about the probabilities of error, the class transition probabilities, and the decision costs. We can obtain this information by using the 200 character test set as all possible input characters. This results in a 1000 by 10 apriori probability matrix which can be used as outlined by Nahin and Pokoski [3] to implement the Bayesian fusion. This fusion process was implemented on the post detection values and produced a recognition rate of 75% on the test set. The 73% recognition rate of the neural network for the same data compares surprisingly well with this 75% rate for the optimum procedure. But surprising or not, these rates can be improved dramatically by moving further back in the system and fusing the pre-detection signals.

Pre-detection Neural Network Data Fusion

Much information is lost when the pre-detection data is compared to the stored templates. If the character is skewed or is not placed correctly on the grid, the template match may return a value that does not truly represent the character. To avoid this problem, the neural network can be used as the entire recognition device rather than just as a way to combine the decisions of the three algorithms. In this way, all possible information from the character is used in the decision process and the overall results should improve.

The neural network used in this part of the experiment has 112 inputs (16 from quadrant analysis, 64 from the outside-in procedures, and 32 from run length encoding), 10 hidden nodes and 10 output nodes. The network is trained the same as before and the same 200 characters are input as tests, with the following results.

Training set: no errors (as expected).
Test set: 86% correct.
Overall result: 93% correct.

Thus overall improvement in performance has been demonstrated with neural network fusion.

Direct Neural Network Input Recognition

As a demonstration of the advantage of combining the outputs of multiple sensors, a neural network was trained to recognize the characters directly from the 16 by 16 input grid. The network was the same as the ones used above but had 256 inputs. Training was implemented using half the sample characters and testing was done on the other half. The result was a recognition rate of 64% on the test set. This demonstrates that obtaining data from further back in the system does not always override the advantage of gathering more information with multiple sensors.

A comparison of the efficiency of all of the above character recognition schemes is shown in Figure 7.

Conclusion

Data fusion applies to any system that uses more than one sensor. How to best fuse this data can be determined if the pertinent statistics are known, i.e., prior probabilities, transitional probabilities for each sensor, etc. Since this information is difficult to measure, a sub-optimum scheme using neural networks provides a workable solution.

In this report we have described the use of neural networks for data fusion and provided experimental results from a pattern recognition example. We have described declaration and detection as the processing of signals for the purpose of making decisions. The detector reduces the input data to the form compatible with the declaration unit input, and the declaration unit then selects the appropriate output decision for this data.

It has been shown elsewhere [4] that it is best to fuse data as early as possible in the system. In this paper we have shown how to fuse both pre-detection and pre-declaration data from several sensors using a neural network. Although our example was for pattern recognition, the same general principles apply to any multi-sensor system. The neural network can be trained with typical data and then applied to the real world.

References

[1] P. Werfbos, "Beyond regression: New tool for prediction and analysis in the behavior sciences," Ph.D. dissertation, Harvard Univ., Cambridge, MA. 1974.

[2] D. Rumelhart, G. Hinton, and G. Williams, "Learning internal representations by error propagation," in Parallel Distributed Processing, Vol. 1 (D. Rumelhart and J. McCleland, Eds. Cambridge, MA: MIT Press, 1986.

[3] P. J. Nahin and J. L. Pokoski, "NCTR plus sensor fusion equals IFFN or can two plus two equal five?", IEEE Transactions on Areospace and Electronic Systems, Vol. AES-16. May,1980, pp.320-337

[4] K. C. Overman and D. F. Mix, "Vector decision making," IEEE Pattern Recognition and Image Processing Conference PRIP-82, June 1892, pp 8-10.

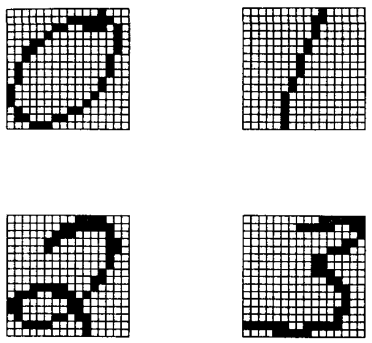


Figure 1

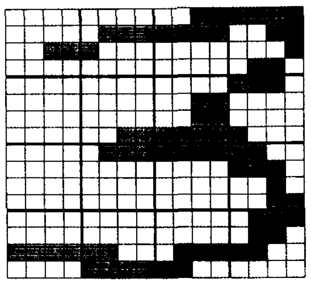


Figure 2a

2	4	6	9
0	2	8	4
0	3	4	8
4	6	5	7

Figure 2b

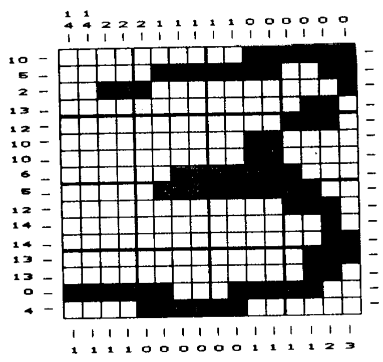


Figure 3

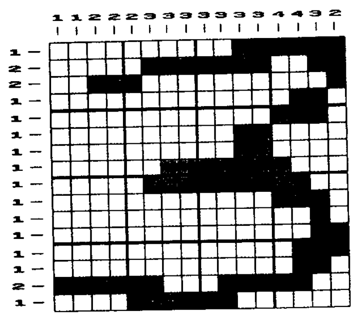
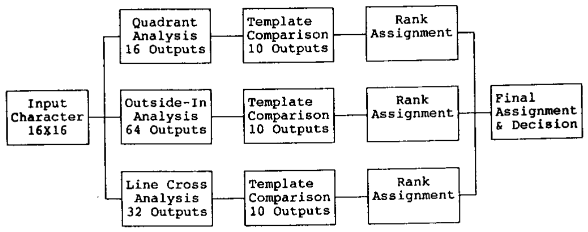


Figure 4



Input Character 16X16
Quadrant Analysis 16 Outputs
Outside-In Analysis 64 Outputs
Line Cross Analysis 32 Outputs
Template Comparison 10 Outputs
Template Comparison 10 Outputs
Template Comparison 10 Outputs
Rank Assignment
Rank Assignment
Rank Assignment
Final Assignment & Decision

Figure 5

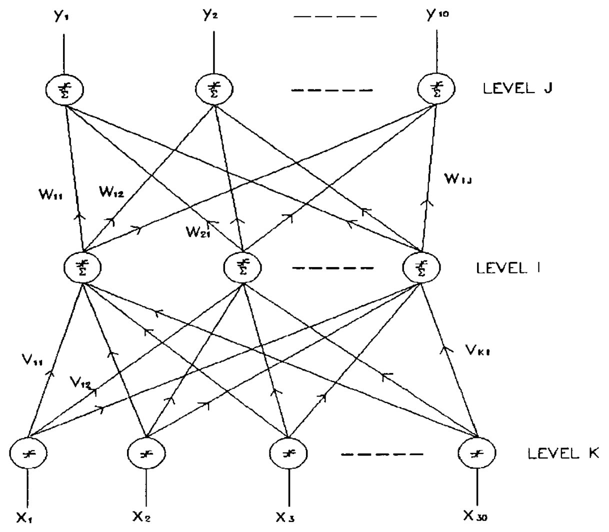


Figure 6

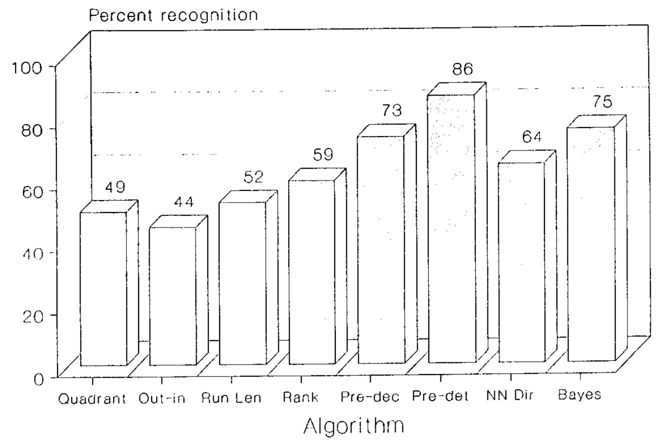


Figure 7 Comparison of All Rates