Charles Kingston,[1] *D.Crim.*

Neural Networks in Forensic Science

REFERENCE: Kingston, C., "**Neural Networks in Forensic Science,**" *Journal of Forensic Sciences*, JFSCA, Vol. 37, No. 1, Jan. 1992, pp. 252–264.

ABSTRACT: Neural networks were developed to study and mimic the functioning of the human brain. Humans are good at pattern recognition; the question is how good neural networks are at it, particularly with problems of forensic science interest. Simulation experiments with a type of neural network known as a Hopfield net indicate that it may have value for the storage of toolmark patterns (including bullet striation patterns) and for the subsequent retrieval of the matching pattern using another mark by the same tool for input. Another type of neural network, the back-propagation network (BPN), is useful for applications similar to those for which standard statistical methods of pattern classification can be used. This would be an appropriate approach to the matching of general component patterns, such as gas chromatograms of gasoline, or pyrolysis patterns from materials of forensic science interest, such as paint. The BPN may provide better results than statistical methods, but it is currently necessary to try both to determine which would be best for any given situation.

KEYWORDS: forensic science, neural networks, pattern recognition, classifications, discrimination analysis, statistical analysis, toolmarks, striations, bullets

General Background

Neural networks are basically machines (either hardware or software simulations) that are built to study the manner of operation, or to attempt to mimic the operation, of the human brain. The structure and operation of the various types of neural networks are loosely based on what is known about the structure and operation of the human brain. The basic structure of a neural network, at least those that will be discussed here, consists of a number of nodes (referred to as neurons) that are interconnected in a particular manner. Each interconnection will, in general, have a weight associated with it. The way in which the neurons are linked and the functions associated with the links, determine the type of network.

A book by Caudill and Butler [*1*] provides a good nonmathematical introduction to neural networks. A collection of a series of articles by Caudill [*2*] presents a simple discussion, including the mathematical operations, of several different types of neural networks. These two references provide a good general overall view of neural networks. A more detailed consideration of neural networks is provided in a book by Pao [*3*]. Müller and Reinhardt [*4*] look at the physics of neural networks in a detailed introduction to the subject that relies heavily on mathematical exposition (this one is not for light reading). Weiss and Kulikowski [*5*] discuss neural networks and how they fit in with other means of classification that have been, and are being, widely used. A discussion of the value of neural networks in comparison with other classification (and prediction)

Received for publication 22 March 1991; accepted for publication 19 June 1991.
[1]Professor of criminalistics, John Jay College of Criminal Justice, New York, NY.

methods is the primary focus of this book, and there is not much detail on the operation of neural nets. It does discuss various concepts that are of interest in forensic science.

The aspect of neural networks that makes them of interest in forensic science is their capability of pattern recognition and classification. Neural networks can be useful in complementing human capabilities in such operations. Humans are very good at recognizing and comparing patterns. It is not possible using current technology to develop a machine or write a computer program that can do better. Our superiority in this task is due not only to the structure of the brain but to the fact that we can bring a wealth of information to bear on the situation.

For instance, reading is based on the recognition of the patterns of the letters (as well as the interpretation of the meanings of groups of letters formed as words and the meaning of groups of words). Machines exist that are very capable of recognizing letters using optical character recognition methods. Both humans and machines are capable of adapting to a variety of letter forms (or fonts). Both can be slowed down by using letter forms that change in the same sentence or word, as can be the case with handwriting. We are better able than machines to make sense out of handwriting since we can recognize letters based on the letters adjacent to them in a word and perhaps on words adjacent to the word with the fuzzy letter or letters. In other words, we can determine what a letter must be by first interpreting the content of the written material based on recognizable letters and then determining what the fuzzy letter or letters must be. With some handwriting, it may even be necessary to evaluate the meaning of the written material by using knowledge of who wrote it and why. It is extremely unlikely that any machine or program could, or would, be designed to take all of this additional information into account.

Much the same can be said with respect to the comparison of many types of patterns. It is unlikely that a machine or program could be designed that would be superior to (or as good as) a human in comparing fingerprints (particularly trace prints), toolmarks, broken edges, and other similar types of patterns that occur in physical evidence, or in determining whether or not they came from a common origin. Common characteristics of such patterns are that they are generally complex, that individual patterns tend to be far apart in some sort of pattern space, and that variations among the patterns from the same source tend to be close together in this same pattern space. In general, only morphological patterns tend to fit this criterion.

On the other hand, machines can operate significantly faster for many tasks than can humans, and they have better memories. Computerized latent fingerprint systems have been developed that use this superior capability to search for potential matching prints, leaving the final comparison and decision to humans. Once the information from a file of prints has been encoded for the system, it remembers it all. Given the specific algorithm for comparing the questioned print with the file prints, the system can search through the file using this algorithm orders of magnitude faster than could a human.

When such morphological patterns occur, we must generally deal with incomplete or noisy samples of the patterns. The pattern from a fingerprint does not vary, although it may show spatial distortion due to the flexibility of the skin and some fuzziness of specific pattern characteristics. A trace fingerprint will generally only present a limited part of the entire finger pattern. Patterns such as those formed by the striations on a fired bullet do show changes on different bullets fired from the same weapon. An intriguing claim about neural networks is that some configurations can find a stored pattern based on an incomplete or noisy example of that pattern. This suggests that they may be useful for developing and searching files of the types of patterns mentioned here.

The next section of this paper discusses a neural network configuration known as the "Hopfield net" and some experiments done with it using simulated striation-type patterns, such as those found on bullets and toolmarks.

The Hopfield Net and Toolmarks

A Hopfield net provides an associative memory which allows recall of its contents based on partial knowledge of those contents. It consists of a number of nodes that are all connected to each other (see Fig. 1). The physical arrangement of the nodes for display purposes is not material to the operation of the net. A weight is assigned to each connection (or link) in each direction. These weights may be thought of in terms of an n by n matrix for a net with n nodes. The link between a node and itself may or may not be used.

A brief technical description of the binary Hopfield net is given next for those who are interested, and may be skipped without loss of continuity.

Technical Description of the Hopfield Net

The u^{th} input to a binary Hopfield net can be represented by a vector $\mathbf{x}^u = (x_1^u, x_2^u, \ldots x_n^u)$ where there are n nodes in the net. Each component of $\mathbf{x}$ (for example, x_i) can be 0 or 1. The net has n^2 weights associated with the links connecting the nodes. The weight associated with the link going from node i to node j will be referred to as w_{ij}. The weights may be symmetrical so that $w_{ij} = w_{ji}$, but that is not absolutely required. The weight associating a node with itself (w_{ii}) may be set to zero and thus essentially ignored.

The network learns a set of patterns by setting the weights according to a specific procedure. The simplest procedure is one referred to as Hebb's rule

$$w_{ij} = \frac{1}{N}\sum_{u=1}^{N}(2x_i^u - 1)(2x_j^u - 1)$$

where N is the number of patterns. Note that $(2x - 1)$ sets the value to -1 if $x = 0$, and to 1 if $x = 1$.

A pattern $\mathbf{z} = (z_1, z_2, \ldots z_n)$ can be input to the net in order to recall the associated stored pattern. This is initiated by evaluating

$$z_i = \sum_{j=1}^{n} w_{ij}(2z_j - 1)$$

for $i = 1$ to n.

Then z_i is updated as follows

$$z_i \Rightarrow 1 \quad \text{if}\left(\sum_{j=1}^{n}(2z_j - 1)\right) \geq T_i$$

$$z_i \Rightarrow 0 \quad \text{if}\left(\sum_{j=1}^{n}(2z_j - 1)\right) < T_i$$

where T_i is a threshold value which is usually set to 0. This process is repeated until the z_i values do not change.

If $\mathbf{z}$ is close enough to one of the stored patterns, say $\mathbf{x}^u$, then $\mathbf{z}$ will equal $\mathbf{x}^u$ after the recall sequence is finished. If $\mathbf{z}$ is too distorted, it may end up in a recall pattern that is not the same as any stored pattern. We may then select the stored pattern that is closest to the recall pattern for evaluation.

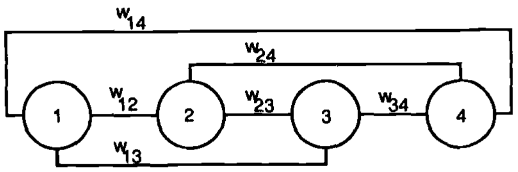

FIG. 1—*Hopfield network with four nodes showing the links and weights from the low to high nodes.*

When using binary data, closeness (or distance) is typically measured in terms of the number of nonmatching associations between the patterns. This is called the Hamming distance metric. The Hamming distance, H, between **x** and **z** can be calculated as

$$H = \sum_{i=1}^{n} (x_i - z_i)^2$$

where n is the number of values in the patterns. A value of zero is obtained for a perfect match (that is, the distance between the two patterns is 0).

One way of looking at a neural network and its recall process is to consider the blank net as a surface generated as a function of the weights. Each pattern that is stored in the net produces an indentation in the surface, with the bottom of the indentation being set by the functional value of the weights. As patterns are stored, the network will appear to be a surface with a lot of depressions, with the bottom of each depression ideally representing one of the patterns. Not so ideally, there may also be other depressions that are not associated with any of the stored patterns.

When an input pattern is presented to the net, the net will try to equilibrate so that it is in one of the depressions. If the starting position is somewhere within a depression, it will tend to go to the bottom of the depression. If the starting pattern is too noisy, the starting position may not be within any depression, or in the wrong depression. If the starting position is somewhere in a depression for one of the stored patterns, it is said to lie in the basin of attraction of that pattern. The result of a search will be the pattern represented by the weight function after equilibrium has been reached.

The simulations of a toolmark (ballistics) file using a Hopfield net will be discussed next.

Toolmark Simulations Using a Hopfield Net

Müller and Reinhardt [*4*] discuss the structure and operation of associative memory and the Hopfield network. A disk is provided with their book which contains several programs designed for use on computers using the MS-DOS or an equivalent operating system, along with their source code (in C), which provide simple examples of various neural networks. The source code "asso.c" (for program "asso.exe"), which is provided on the disk, was modified and used for evaluating the operation of the Hopfield network for storing and retrieving toolmarks.

The original program displayed the network input and output in a 12 by 13 matrix. The patterns used were letters and numbers formed within the matrix. This was rearranged to a 3 by 52 matrix in which toolmarks could be formed. This allowed 52 positions within which striations could be placed, with each striation using 3 equally valued nodes. The overall appearance is shown in Fig. 2. This is considered to be a view looking down on the pattern, where each dark line is a striation. Adjacent lines were considered to be individual striations; they could also have been viewed as single wide striation marks.

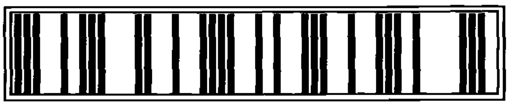

FIG. 2—*Node representation showing the top view of a toolmark.*

Figure 3 shows a bullet comparison photograph that was scanned into a computer. The photo was scanned as a halftone image, and a network striation pattern was superimposed on the image to illustrate the relationship of the network patterns used here with actual bullet markings. The striations are slightly slanted in the image; the pattern lines were aligned at the top.

Fifty patterns were randomly generated and saved in a file. This file was used for all experiments. The random process was set up so that each of the 52 positions had a probability of 0.5 of being a striation. This probability was fixed at 0.5 in the original program. A modification was made to allow the value to be changed as a startup parameter to allow more flexibility in future tests.

The program provides several parameter options that can be used. The basic initial parameters used were the following:

Total number of patterns = 50
Number of patterns to be read from the file = 50
Sign constraint = 0 (that is, data are represented by -1 or 1)
Learning rule = 5 (an extension of Hebb's rule)

The secondary parameters used were the following:

How to iterate? = 1 (sequential)
Temperature = 0.0
Threshold potential = 0.0
Start with noisy pattern (1) or incomplete pattern (2) = 1

To make a search for a pattern stored in the net, one of the patterns is first selected. Then a noise factor is selected in order to produce a noisy version of the pattern. This is done by considering each value in the pattern and making a decision to alter that value based on the noise factor. If the noise factor is 0.6, the decision to alter the value would be made with a probability of 0.6. If that decision is made, the value is set to 1 with a probability of 0.5 and to -1 with a probability of $1 - 0.5$, or 0.5. The alternative of searching with an incomplete pattern (a selected number of nodes starting at the end being set to -1) was not used.

The resulting noisy pattern is then presented to the net, and the recall process mentioned above is initiated. The original program is set up so that this is a manual process, which can quickly get tedious. Therefore, the program was modified to select each pattern

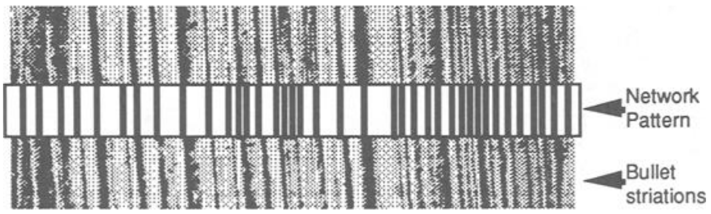

FIG. 3—*Scanned bullet striation pattern (from photo) with an overlay of a network pattern.*

automatically in sequence, apply a specified noise factor, go through the recall process, and save and report the results.

Bullet striations can be stored in two different ways. Each land and groove could be stored separately. When this is done, it is reasonable to assume that the lateral orientation of the markings can be fixed with respect to the start and end of a land or groove. When the full circumference is coded and stored, or when using marks made with a crowbar, screwdriver, or some other type of tool, the lateral orientation cannot necessarily be fixed. In such cases, we would want to somehow find a way of representing the patterns that would be invariant to translation, or some equivalent of this. If the patterns are two dimensional, such as would be the case with fingerprints, we would also want rotational invariance. Scale invariance would probably not be required, although distortion invariance might be.

Müller and Reinhardt [*4*] review some work on invariant pattern recognition in Section 8.3 of their book. In discussing the preprocessing of the input patterns, they point out (page 84 of Ref *4*) that a translation of a two-dimensional input pattern in a plane corresponds to a multiplication of its Fourier transform by a constant phase. A short program was written to examine the Fourier transforms of the simulated toolmarks to see if this might offer a way of obtaining translational invariance. It became clear that this was not going to provide a "quick fix" for the problem, and that approach was set aside for future exploration. Müller and Reinhardt pointed out in a footnote (page 84 of Ref *4*) that "a trivial but not very elegant preprocessor for translational invariant pattern recognition would simply shift the input pattern around until it 'locks in' with one of the stored patterns."

That is the approach that was selected for use, and the network program (now called "toolmark.exe") was again modified to allow shifting the input pattern a selected number of times for searching. If a shift count is specified, the search is first made using the input pattern directly. The pattern is then shifted one position to the left (with the leftmost value rotating around to the right end), and the search is made again. This is repeated until the pattern is shifted the specified number of times. The best match from all searches made with the pattern is selected as the final result. The best match is selected as the stored pattern that has the greatest overlap with the recall pattern. If the full circumference of bullets is being stored, the shifts would be made to bring each land-groove sequence into a specific alignment position.

The overlap between two patterns, $\mathbf{x}$ and $\mathbf{y}$, is calculated as

$$\text{overlap} = \frac{1}{n} \sum_{i=1}^{n} (2x_i - 1)(2y_i - 1)$$

where n is the number of values for each pattern. Note that the overlap is related to the Hamming distance, and will be 1 when the recall pattern is exactly the same as the selected stored pattern, and less than 1 when it is not

The above process will generally pick the pattern that is most similar to the input pattern in terms of the number of matching lines. This is not necessarily the criterion that provides the best results. In a study of bullet striations, Biasotti [*6*] found that the average percentage of matching lines (striations) for bullets fired from the same weapon was low (36 to 38% for lead bullets and 21 to 24% for metal-cased bullets) and that for bullets fired from different weapons was only somewhat lower (about 15 to 20%). He concluded that the presence of sequential matching lines was a stronger indication of common origin than the percentage of total lines that matched. His data indicate that the probability of finding a sequence of four (or more) consecutive matching lines is much higher between bullets fired from the same weapon than for bullets fired from different weapons.

TABLE 1—*Results of simulation runs with the toolmark.exe program.*[a]

Noise, P	Times Shifted	% Found	% Found/Sequence
0.2	0	100	100
0.4	0	100	100
0.6	0	90	92
0.8	0	44	56
0.2	6	100	100
0.4	6	96	98
0.6	6	58	72
0.8	6	32	40
0.2	10	100	100
0.4	10	98	98
0.6	10	62	78
0.8	10	18	30

[a]The sequence count is four matching lines.

The program (toolmark.exe) was again modified to incorporate this into the search process. A new parameter was added for indicating the number of sequential lines that would remain undisturbed when the input pattern was altered to add noise. One area near the center of the pattern was selected to be the fixed sequence of matching lines that would remain undisturbed. A secondary selection process was then built in to the program. If the match normally selected by the network contained a matching sequence of at least the indicated number of lines (anywhere in the pattern), the secondary process is not initiated. If it does not, the secondary process selects the pattern that meets the joint condition of the highest overlap and a matching sequence. This match is then reported, in addition to the original one, but is tallied separately.

A number of runs were made with the noise and number of shifts varied. The results are shown in Table 1 as the percentage of searches resulting in the correct selection, both without and with sequence matching.

Some runs were made to evaluate the effect of the sequence count. The results are shown in Table 2.

Comparing striations as a set of undifferentiated lines (except possibly for width), as was done in the above simulations, is a simplification of the process used in practice. In actuality, the depth or contour characteristics of the lines are also important when declaring a match. Therefore, the program was again modified to allow the inclusion of some limited contour information. This was done by considering the node layout as representing a cross-sectional view of the mark rather than a top view. Each mark position is graded into three depth levels.[2] Adjacent marks are considered to be part of one wide striation. The top of the display is considered to be the surface of the marked object. A mark only in the first row represents a shallow indentation. An additional mark in the second row for the same column represents a medium indentation, and an additional third mark represents a deep indentation. The overall view is of a crude contour diagram of a striation pattern. This is shown in Fig. 4. The requirements for a sequence match were modified to include a "contour" match of the specified number of consecutive striations.

A set of 50 contour marks was generated and some runs were made to evaluate the effect of using the contour information. Some results are shown in Table 3.

[2]The number of depth levels can, of course, be increased by adding more nodes in the form of more rows to obtain better resolution of the depth information. It is also possible to set up an analog Hopfield network in which the contour measurements can be directly entered.

TABLE 2—*Results of runs with the toolmark.exe program, showing the effect of changing the sequence count.*

Noise, *P*	Times Shifted	Sequence Length (No. of Sequences)	% Found	% Found/Sequence
0.8	6	3	18	28
0.8	6	4	32	40
0.8	6	5	28	48
0.8	6	6	44	72

Discussion of the Hopfield Net

The results of the above simulations indicate that a Hopfield network, or one similar to it, has promise for storing and recalling bullet striation patterns or other types of toolmarks. There are some factors that need to be kept in mind when considering such usage.

Neural networks in general require considerable computing power, particularly for the learning phase. Learning for the small network used here with 50 patterns takes around 40 min using a 16-MHz 386 computer. It is not clear at this time whether this process would have to be repeated in full for each new pattern stored in the network. Recall time for a single search can take from less than 1 s to more than 1 min. This depends upon how quickly the networi· locks into a pattern that matches the search pattern. A limit can be set to the number of iterations that are permitted to set the maximum time allowed. The network default value is 200 iterations, but the run can be aborted by pressing a key, which might be done after a given number of iterations produced no change. The limit was set to 12 for the runs discussed here, as this appeared to be generally adequate for stabilization of the output pattern.

The maximum number of patterns that can be stored in a Hopfield network is dependent on the number of nodes in the network. The number is limited to no more than twice the number of nodes. The network used for the above simulations could theoretically hold 312 patterns (which might require 2 to 3 h or more to be learned). On the other hand, the network seems to settle into the correct pattern from a noisy input pattern more frequently when the number of patterns is much less than that. This suggests that it is probably a good idea to limit the number of base entries in a network to considerably less than the maximum.

These considerations suggest that neural network implementations for striation pattern storage and recall would be restricted in the number of patterns that could be practically stored, and would therefore be more suitable for local use than for large central systems. This limitation may change as more powerful and faster computers (particularly parallel computers) become readily available and affordable. It may be possible to use a system consisting of several different nets and develop a preliminary classification for toolmark patterns that could be used to determine which net to use for the initial search. This would allow more patterns to be stored in a single system.

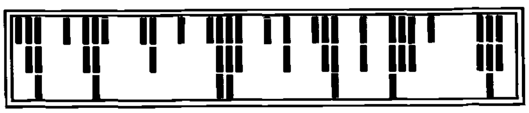

FIG. 4—*Node representation showing the contour view of a toolmark.*

TABLE 3—*Results of runs with the toolmark.exe program, using contour information for sequence matching.*

Noise, P	Times Shifted	Sequence Length (No. of Structures)	% Found	% Found/Sequence
0.6	6	4	44	98
0.8	6	4	12	86
0.8	6	3	12	56

The system aspects of using a neural net need to be considered. Since the Hopfield net will generally select the base pattern that is closest in the sense described above to the search pattern, it would be possible to generate a program that is not based on a neural network model and which would provide a similar filing and searching capability. The decision on the way to go should be based in part on the relative balance of the filing and searching operations. The neural net requires a lot of time to learn the base patterns (that is, to file the patterns), but can perform a search relatively quickly. A database type of operation to do this would allow relatively fast filing but would require considerable time for searching since the closeness calculation must be done with every base pattern for each search. If we decide to provide some sort of preclassification to the patterns to cut down the number of base patterns used for calculation during a search, we then increase the filing time. Another consideration is that a database type of approach would probably allow, at least at the present time, much larger systems to be used, provided the searching times were reasonable.

The benefits of creating a toolmark file, such as one for bullet striations or perhaps case markings, also need to be considered. Because of the recidivistic nature of criminal behavior, it is reasonable to expect that a weapon used in one criminal incident will have a high probability of being used, or of having been used, in other criminal incidents.A file for recording bullet or case markings for those items found at an incident location would provide a means of associating these materials with those found in previous or subsequent incidents. This would undoubtedly have beneficial results for the investigation or prosecution, or both, of such cases. Whether this would be worth the cost and effort to develop and maintain such a system must be carefully weighed. The storage and retrieval of the pattern data are only part of the overall system. A practical means of getting the data from the bullet, case, or other toolmark must be found, and a way to translate those data into a form acceptable to the data system must be developed. A small system might use photographs, together with a digital input tablet or a scanner, and a human-assisted computer program to code the striations. A large system would probably require an automated way of encoding the markings. The technical capability is undoubtedly available to do this; the problem will be in putting it all together into a practical system.

Another type of neural net that may have potential use in forensic science is the back-propagation network, which will be considered in the next section of this paper.

The Back-Propagation Network

A back-propagation network (BPN) is a multilayer network that is trained by a modification of the least mean square procedure called back-propagation. The network is structured in layers, where each node of a layer is connected to the nodes of the next layer. This is illustrated in Fig. 5. The first (top) layer is the input layer and the last (bottom) layer is the output layer. The layers in between these are called hidden layers (only one hidden layer is shown in Fig. 5).

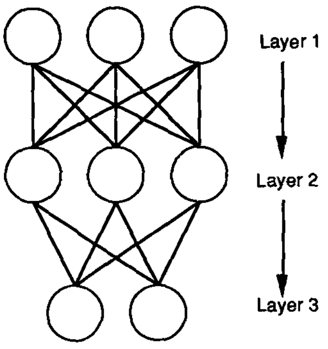

FIG. 5—*Back-propagation neural network with three layers.*

The BPN has probably found more practical application than other neural network. It is competitive with statistical classification procedures such as the use of linear discriminants and of nearest neighbor methods. Weiss and Kulikowski [5] compare the relative performance of BPNs, statistical procedures, and decision rule procedures.

Technical Description of the Back-Propagation Network

As in the Hopfield net, each node will have a value, and a weight is associated with each forward connection. The nodes in the first layer are set to the values that are input to the net. In a binary net, there will be either zero or one. In an analog net, they will usually be scaled so that they fall between zero and one. Only the analog net is considered here. The output of the first layer is simply the input value.

The values are propagated through the net from layer to layer in a similar fashion and will be described only for the first and second layers. If $\mathbf{x}$ is the input pattern vector $(x_1, x_2, x_3, \ldots x_n)$, and $\mathbf{w}_k$ is the weight vector $(w_{1k}, w_{2k}, w_{3k}, \ldots w_{nk})$ from the input layer to Node k in the second layer, then the value for that node (s_k) is

$$s_k = f\left(\sum_{i=1}^{n} (w_{ik} \times x_i)\right)$$

where a possible choice for the activation function, $f(\ )$, is

$$f(h) = \frac{1}{e^{-(h+T)}}$$

where T is a threshold value, which is generally set to zero. The activation function ensures that the value lies between zero and one. This same process continues through the layers until we reach the final or output layer.

During this training process, we know what the final output should be for each input pattern. We can therefore calculate the error vector, $\mathbf{E}$, which is the vector of differences between the values that were calculated and the desired values. This is used to adjust the weight values between the next to last and last (output) nodes ($\mathbf{W}$) using what is called the Delta rule

$$\mathbf{W}_{t+1} = \mathbf{W}_t + \frac{\mu \mathbf{E} \mathbf{V}}{|\mathbf{V}|^2}$$

where $\mathbf{W}_{t+1}$ is the new weight vector (at time $t + 1$), $\mathbf{W}_t$ is the old weight vector (at time t), μ is a constant between zero and one, $\mathbf{V}$ is the vector of values from the nodes of the next-to-last layer, and $|\mathbf{V}|$ is the magnitude of the $\mathbf{V}$ vector.

The errors for the next-to-last layer (a hidden layer) must now be calculated and used to adjust the weights coming from the layer previous to that. These errors are not known, but usable values can be obtained by calculating the error for each of the hidden layer nodes as

$$e_k = f'\sum_{i=1}^{n} (w_{ik} \times x_i) \times \sum_{j=1}^{m} (w_{kj} \times E_j)$$

where e_k is the error for hidden node k, w_{kj} is the weight between node k and output node j, E_j is the error for output node j, n is the number of hidden layer nodes, and m is the number of output layer nodes. An explanation of this is given by Caudill (Ref *2*, Part 3), and the reader is referred there, or to the other references, for more detail.

This forward and back propagation cycle is repeated as many times as necessary to reduce the error to the maximum that we wish to allow in the output for a given set of input patterns. This learning process can take a long time when done on a general purpose computer. Special boards have been developed that decrease the time by factors up to 100. This can mean learning times of several minutes rather than of several hours.

Discussion of the Back-Propagation Network

The difference in use between the Hopfield net and the BPN as described here should be noted. For the Hopfield net, we input one example of a given pattern for the learning process. We then hope that a noisy search pattern will enter the net in the basin of attraction for the correct base pattern. This approach is most useful for patterns for which samples from a common source are far apart in some pattern space. For the BPN, several noisy examples of each pattern are used for training the net. The goal is to have the training samples for each pattern represent all of the variations of that pattern, although this may not be possible in reality. The output for a search pattern will be the base pattern that is closest in some sense to the search pattern. In this way, a BPN is useful for patterns for which samples from a common source may not be widely separated in some pattern space.

This is essentially the same general procedure that is used with statistical classification (or discrimination) procedures. For these, we want to use several examples of each pattern so that the statistical distributions of the values for each pattern can be estimated or incorporated into the calculations as well as the distributions of the locations of the different patterns. Statistical discriminant methods generally do not work well if the distributions do not have a single mode. On the other hand, back propagation networks can work well for patterns that have multiple modes.

The example generally used to illustrate this last statement is the XOR (or parity) problem. Consider an input with two values. If these values are equal (both zeros or both ones), the pattern output is assigned a value of 1. Otherwise it is assigned a value of 0. This is shown as a two-dimensional mapping in Fig. 6. Although this is a binary problem, it can be thought of as an analog situation with each distribution having two modes if we allow the inputs to vary by some small value around 0 or 1. Linear discriminant models cannot provide proper classification for this type of input since the two sets of output values are not linearly separable (that is, a line cannot be drawn that separates the two output values). A properly configured BPN has no problem with the XOR example or an analog variation of it.

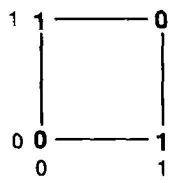

FIG. 6—*The XOR pattern.*

The type of situation in which a BPN might be used is one in which investigators are trying to associate some evidential material with its source and are evaluating several variable parameters of the material. For instance, a chip of paint may be pyrolyzed to obtain a pyrolysis pattern for comparison with such patterns from suspected sources. The locations and areas of the pyrolysis peaks, or values generated from these, would be used as input patterns to the BPN. The comparison of gasolines by means of the patterns obtained by capillary gas chromatography is another example. The results obtained in such a project with gasoline, along with a comparison of the performance of a BPN with statistical methods in analyzing data, will be discussed in a separate paper.

Whether a BPN provides better performance than standard statistical or other methods depends upon the particular problem. As mentioned above, the book by Weiss and Kulikowski [5] presents comparative information on the relative performance of BPNs and other methods. These authors conclude that it is necessary to try the various approaches to select the one that performs the best. Statistical methods are generally quicker and therefore preferable unless it is found that a neural network can provide better performance.

When a decision is made by using a statistical method, one can readily ascertain why that choice was made. This is not so easily done when using a neural net. When a neural net learns a group of patterns, it is not readily apparant how the net has categorized the patterns, even though we know exactly what algorithm it is using for the learning process. This could be a problem if a neural net is used for evidence evaluation when testimony is likely to be required on that evidence.

Summary

Neural networks were developed to understand and mimic the performance of the human brain. The Hopfield net structure has been described and has been used to simulate a file for storing toolmark patterns (such as bullet striations) and recovering the correct pattern from a different example of a pattern (which may vary somewhat from the base pattern) from the same source. The results of the simulations suggest that the technique could be useful for this purpose, particularly if contour information is included in the pattern description. The back-propagation network has also been described, and the author has pointed out that it could be useful for comparing pyrolysis patterns of paint or component patterns produced by gas chromatography of mixtures such as gasoline. Such networks generally require long learning times, which makes it prudent to try competing statistical methods initially to see if they can provide adequate discrimination.

References

[1] Caudill, M. and Butler, C., *Naturally Intelligent Systems*, MIT Press, Cambridge, MA, 1990.
[2] Caudill, M., *Neural Networks Primer*, Miller Freeman Publications, San Francisco, CA, 1989 (a collection of eight articles reprinted from issues of *AI EXPERT* magazine).

[3] Pao, Y.-H., *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, Reading, MA, 1989.
[4] Müller, B. and Reinhardt, J., *Neural Networks: An Introduction*, Springer-Verlag, Berlin, Germany, 1990.
[5] Weiss, S. M. and Kulikowski, C. A., *Computer Systems That Learn*, Morgan Kaufmann Publishers, San Mateo, CA, 1991.
[6] Biasotti, A. A., "A Statistical Study of the Individual Characteristics of Fired Bullets," *Journal of Forensic Sciences*, Vol. 4, No. 1, Jan. 1959.

Address requests for reprints or additional information to
Charles Kingston, D.Crim.
John Jay College of Criminal Justice
445 W. 59th St.
New York, NY 10019