Neural Network Classifier with Analytic Translation and Scaling Capabilities for Optimal Signal Viewing

by

Rick B. Vilim
and
Stephan W. Wegerich

Reactor Analysis
Argonne National Laboratory
9700 S. Cass Avenue
Argonne, Illinois 60439

1Work supported by the U.S. Department of Energy, Nuclear Energy Programs under Contract W-31-109-ENG-38.
A neural network originally proposed by Szu for performing pattern recognition has been modified for use in a noisy manufacturing environment. Signals from the factory floor are frequently affine transformed and, as a consequence, a signal may not be properly aligned with respect to the input node that corresponds to the signal leading edge or with respect to the number of nodes representing the time varying part. Rather than translate and scale the presented signal, an operation which because of noise can be prone to numerical error since the signal is not smoothly varying, the network in this paper has the capability to analytically translate and scale its internal representation of the signal so that it overlays the presented signal. A response surface in the neighborhood of the stored reference signal is built during training and covers the range of translate and scale parameter values expected. A genetic algorithm is used to search over this hilly terrain to find the optimal values of these parameters so that the reference signal overlays the presented signal. The procedure is repeated over all hypothesized pattern classes with the best fit identifying the class.
A Neural Network Classifier with Analytic Translation and Scaling Capabilities for Optimal Signal Viewing

Rick B. Vilim and Stephan W. Wegerich
Reactor Analysis Division, ANL

I. Introduction

Manufacturing processes give rise to a potentially rich collection of signals that can provide valuable information for diagnosing conditions and for scheduling maintenance operations. The extent to which new methods are successfully introduced on the factory floor is determined by several factors. An important one is how performance fares when the method is introduced on the production line after simulation trials. One needs to be sure that significant non-ideal behavior on the production line has been modeled and taken into account. In this respect, there is a class of data transformations that can occur on the factory floor that need to be considered when designing a neural network classifier.

It is easy to imagine how signal data, to be processed by a pattern recognition technique, might be inadvertently transformed before they are sampled by the digital data acquisition system. Consider a stand-alone piece of equipment that is run repeatedly through a characteristic cycle of operation. The equipment might be located on a production line with operation automatically monitored to provide early detection and diagnosis of improper operation. The speed of the equipment is likely a variable that can change under external load or is operator settable causing a scaling in time of the signal. The leading edge of the signal may be translated in time relative to the data acquisition cycle depending on what triggers the equipment operation, or the presence of noise on a signal makes it difficult to determine the leading edge. Clearly there is the potential for misalignment: the network was trained expecting a specific node to correspond to the leading edge of the signal and the time varying part to a specific number of nodes. One can attempt to correct for these effects by numerically translating and scaling the measured signal so...
as to optimize the fit with stored reference signals. Since the signal is not smoothly varying, however, the presence of noise can make these two operations prone to numerical error. Another approach would be to intentionally translate and scale the training signals so that they span the expected range of the measured data. Then when a measured signal is presented to the network it will fall within the training envelope. There are two disadvantages, however. First, the crispness of the network identification is degraded and, second, there is the error introduced by curve fitting noisy signals. The problem considered in this paper is how signal pattern recognition can be performed by a neural net in a way that is invariant to the affine transformation.

The solution proposed here is to build into the network the capability to translate and scale analytically the network's internal model of each signal class. The network attempts to align itself so that it best overlays the signal presented at the input. This can be viewed as a tuning exercise that makes up for the inherent difficulty in designing an on-line algorithm that can accurately identify the start and length of a single noisy signal when it is presented to the network. The representation is chosen so that scaling and translating the reference signal can be done without having to numerically interpolate. The reference signal is stored in a neural network in analytic form using wavelet functions. The affine representation of time in the wavelet functions allows the reference signal to be translated and scaled without numerical approximation. With the neural net assembled and trained as a classifier, the reference signals corresponding to the classes are then each best fit to the presented signal by adjusting the parameters that control the affine representation. The output node with the greatest activation identifies the pattern class to which the presented signal belongs.

One might note, however, that the need to align the presented signal and the network input nodes at some point in the training-operation cycle has not gone away. Instead of aligning at the time the signal is presented for classification, as is standard, it still needs to be done prior to training to ensure that the network has a crisp stored image. So one must still translate and scale. Yes, this is true, but presumably this can be done with greater intelligence and precision under the hand of a human prior to training, as opposed to the alternative case, where after training and
The network is based on the connectionist model proposed by Szu for classifying speech signals [1]. In his paper wavelet functions are used to represent a time dependent signal. By adding weights and output nodes a network architecture with classification capabilities is then obtained.

The network defined in [1] can be used to recognize those features in a signal that make it a member of a class of signals that are separate from other classes of signals also to be recognized. The approach is to find a set of basis functions that span the space defined by the input signals. Each signal is regarded as a vector in Cartesian space where each of the sequential sample points lies on its own coordinate axis. The idea is that a small number of basis functions if properly chosen can represent many different and complex input signal features. The network operates by projecting the input signal onto the basis function coordinate axes. The coordinate values are then passed to a classifier which associates coordinate values with signal classes.

The value before scaling of the $p^{th}$ output of the network when the $l^{th}$ signal is presented at the input is given by

$$u_{p,l} = \sum_{k=1}^{K} w_{kp} \sum_{t=1}^{T} i_l(t) c_k h \left( \frac{t-b_k}{a_k} \right)$$

(1)

where

$i_l(t) =$ value of input signal $l$ at time $t$ and

$w_{kp} =$ $k^{th}$ weight into output node $p$. 

II. Wavelet-Based Neural Network
The sigmoid function

$$o_{p1} = \frac{1}{1 + e^{-u_{p1}}}$$

scales the classifier output so that it lies between 0 and 1. A schematic of the network is shown in Fig. 1.

The network operates so that when a particular signal pattern is presented at the input, a predetermined pattern appears at the output. The specific mapping is learnt during a "training" session. Paired input-output patterns are presented and the network weights are adjusted so that the network reproduces these patterns. The adjustment is made by minimizing the error equation.
where \( t_{p,l} \) = target value at \( p^\text{th} \) output node when input signal \( l \) is presented to network. Since this equation is non-linear, it must be minimized through an iterative procedure.

Because each input node represents a point in time, there is a correspondence between the point at which a time varying signal begins to vary and the node with which this leading edge is coincident. This alignment is established during a training session. After the network has been trained, the same alignment needs to be followed when a pattern is presented for recognition.

III. The Method

This section describes the automatic viewing feature that has been developed to produce a classifier that is tolerant to differences in alignment between the network input nodes and the presented signal. The algorithm is composed of two parts: the training part where network weights are computed, and the signal viewing part where the already trained network is adjusted for an optimal fit to the signal presented for classification.

A. Calculation of Network Weights

The network has been successfully trained if the error equation given by Eq. (3) is minimized when the training data is inserted. Standard neural net practice would be to use a gradient descent method to find the corresponding values of \( w_{kp} \), \( a_k \), \( b_k \) and \( c_k \). In this work, however, linearity characteristics permit solving directly for \( w_{kp} \) for given values of \( a_k \), \( b_k \) and \( c_k \). This reduces the dimension of the search space thereby accelerating the training process. Inserting the target values in Eq. (1) and combining with Eq. (2) gives an equation that is linear in \( w_{kp} \).
\[
\ln\left(\frac{1}{t_{pl}} - 1\right) = -\sum_{k=1}^{K} w_{kp} \sum_{t=1}^{T} i_{t}(t) c_{k} h\left(\frac{t-b_{k}}{a_{k}}\right).
\]

which in matrix notation is

\[
A \ W = B
\]

where

\[
[A]_{ik} = -\sum_{t=1}^{T} i_{t}(t) c_{k} h\left(\frac{t-b_{k}}{a_{k}}\right) \quad 1 = 1, \ldots, L \quad k = 1, \ldots, K
\]

\[
[W]_{kp} = w_{kp}
\]

\[
[B]_{lp} = \ln\left(\frac{1}{t_{pl}} - 1\right).
\]

The solution to the above equation, if one exists, is one that minimizes the right side of Eq. (3).

If no solution exists when searching on the \(w_{kp}\) alone, which will almost always be the case if there are more signals than wavelets, the \(a_{k}\), \(b_{k}\) and \(c_{k}\) are introduced to provide additional degrees of freedom. First, a best set of \(w_{kp}\) is found by least squares. Then a gradient descent method is used while holding the \(w_{kp}\) constant to find values for the \(a_{k}\), \(b_{k}\) and \(c_{k}\) that minimize Eq. (3). These values are inserted back into Eq. (5) and the process repeated until Eq. (3) has been minimized with respect to all the \(w_{kp}\), \(a_{k}\), \(b_{k}\) and \(c_{k}\).

The partial derivatives used in a gradient decent method are given by
\[ \frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \sum_{i=1}^{I} (t_{pi} - y_{pi})^2 \alpha_i c^{-u_{pi}} \cdot \frac{\partial u_{pi}}{\partial \alpha}, \quad \alpha = w_{ri}, a_{ri}, b_{ri}, c_{ri} . \]  

where the derivatives on the right hand side are given by

\[ \frac{\partial u_{pi}}{\partial w_{rs}} = \sum_{t=1}^{T} i_{s}(t) h\left(\frac{t-b_{rs}}{a_{rs}}\right), \quad s=p \]

\[ = 0, \quad s \neq p \]

\[ \frac{\partial u_{pi}}{\partial a_{ri}} = \frac{-w_{rp}}{a_{ri}} \sum_{t=1}^{T} (t-b_{ri}) \frac{\partial h(\beta)}{\partial \beta} i_{s}(t) \]  

\[ \frac{\partial u_{pi}}{\partial b_{ri}} = -\frac{w_{rp}}{a_{ri}} \sum_{t=1}^{T} \frac{\partial h(\beta)}{\partial \beta} i_{s}(t) \]

\[ \frac{\partial u_{pi}}{\partial c_{ri}} = \sum_{k=1}^{K} w_{kp} \sum_{t=1}^{T} i_{s}(t) h\left(\frac{t-b_{ki}}{a_{ki}}\right) \]  

The first of the above four equations is not used if one solves for the \(w's\) directly by least squares according to the above method.

**B. Optimization of Signal Viewing**

It would seem that in order to recognize a signal as an affine transformed version of another signal, the network need be only trained on the latter signal. Then when the former signal is
presented to the network for classification, the network will converge to values for \(s\) and \(r\) such that

\[
\mathbf{u}_{p,t} = \sum_{k=1}^{K} w_k \sum_{i=1}^{r} i_i(t) c_k h\left(\frac{t'-b_k}{a_k}\right)
\]

where

\[
t' = \frac{(t-s)}{r}
\]

equals the learnt output for the latter signal. The means for converging to the values of \(s\) and \(r\) present a problem, however. Any search strategy will necessarily involve computing the network output at other values of \(s\) and \(r\). Since the corresponding signals are not in the training set, the network output will be indeterminate.

The solution is to not only train on that one signal that represents the class, but also train on translated and scaled versions in the neighborhood of that signal. The target output values are set lower so that a gradient is established. A cartesian coordinate system is set up with the original signal at the point \((0,1)\). The first axis corresponds to the translate dimension and the second axis to the scale dimension. Enough grid points are included in the training set so that a well shaped response surface is created. If the presented signal is a member of the class of signals corresponding to this response surface, then a search along the two dimensions will lead to the peak.

Now consider the case where the presented signal is a member of a second class for which the output has been encoded differently. Hypothesize that the signal presented belongs to the first class. Then when \(r\) and \(s\) are searched over to drive the network output toward the target for the first class, the final difference (between the actual network output and the class target) will be larger than if the network had been driven toward the second class target. A numerical arbiter would detect this difference and assign the signal to the second class. We therefore have
two assumptions built into the network. The first is that the presented signal must belong to one of the classes. The second is that the presented signal belongs to the class for which the error between class target and actual output is least.

The response surface typically has many local extremum that make the search for a global extremum using a gradient search method unreliable. Many local extremum exist because the width of a basis function is typically much smaller than the length of the signal. Many basis functions are then needed to represent the signal. To avoid this difficulty a genetic search technique is used. Such methods maintain an image of the overall surface topography and are less likely to become stuck at a local extremum.

IV. Results

A. Case Description

Proprietary considerations prevent showing results for manufacturing data, so results obtained with human heart signals are presented. Heart signals exhibit the same characteristics as are found in manufacturing data that are important from the standpoint of this network. These include a one-shot signal (a single heart beat) that may be affine transformed. A normal, lead 2 heart beat is shown at the top of Fig. 2. A heart beat exhibiting ventricular bigeminy is shown at the bottom of Fig. 2. The vertical dashed lines delineate a single pattern. The degree of reproducibility within a class is shown in Fig. 3 with superimposed signals.
The pattern recognition problem is to determine whether the signal presented for recognition, which may be affine transformed an unknown amount, is the normal sinus rhythm (class I) or the abnormal rhythm (class II). The signals presented for identification have been translated up to 16 sample intervals (about six percent) to either side of the leading edge of the same class signal stored in the network and have been compressed or dilated up to twenty percent with the left edge of the signal anchored. The degree of variability of signals within a class is small as seen in Fig. 3 and is much smaller than the degree of translation and scaling being admitted. Since the class patterns are significantly different, the optimal viewing feature should in principle return a crisp indication of class and affine parameter values.
B. Training

The network output is a single node with the target values binary encoded. The target output is 0.9 for class I and 0.1 for class II. Because the network contains exponentials at the output, it is standard to represent the logical 0 and 1 states at the output by 0.1 and 0.9 respectively. The values 0 and 1 can be achieved only by having an essentially infinite value at the argument of the exponential, which gives rise to numerical difficulties.
The basis functions, $h(t)$, are given by

$$h(t) = c \ e^{-\frac{t-b}{a}}$$

The initial values for the parameters $a$, $b$ and $c$ were chosen so that each peak of the signals shown in Fig. 3 was overlaid with the above distribution. This gave a total of five basis functions. The initial values are shown in Table I.

<table>
<thead>
<tr>
<th>Basis Function</th>
<th>Initial $a_0$</th>
<th>Initial $b_0$</th>
<th>Initial $c_0$</th>
<th>Final $a$</th>
<th>Final $b$</th>
<th>Final $c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h_1$</td>
<td>5</td>
<td>50</td>
<td>0.700</td>
<td>5.212</td>
<td>75.613</td>
<td>-0.340</td>
</tr>
<tr>
<td>$h_2$</td>
<td>15</td>
<td>125</td>
<td>0.200</td>
<td>15.855</td>
<td>125.626</td>
<td>0.219</td>
</tr>
<tr>
<td>$h_3$</td>
<td>5</td>
<td>45</td>
<td>0.550</td>
<td>2.108</td>
<td>49.087</td>
<td>0.721</td>
</tr>
<tr>
<td>$h_4$</td>
<td>13</td>
<td>75</td>
<td>-0.200</td>
<td>3.219</td>
<td>78.183</td>
<td>-0.678</td>
</tr>
<tr>
<td>$h_5$</td>
<td>7</td>
<td>200</td>
<td>0.250</td>
<td>24.588</td>
<td>182.831</td>
<td>0.089</td>
</tr>
</tbody>
</table>

Table I. Initial and Final Parameter Values

The network was trained in a two step process. In the first step, the network weights and the values of the parameters in Table I that produce the target values were calculated. The resulting values are shown in Table I. The purpose of the second step was to build a response surface in the neighborhood of each of the two training points in step 1. The response surface gives the value of the network output as a function of the amount the signal presented in step one is translated and scaled when presented in step two. The global extremum is at $(0.11, 0.089)$ and has the same value as in step 1. In other words, if the unity scaling and zero translation signal from
class I were presented to the network, the output would be 0.9. Scaled and translated versions of this same signal give network outputs less extreme in value. The key in building the response surface is to ensure that the extremum occurs at unity scaling and zero translation. The presence of local extremum is not important if a genetic algorithm is used in the search for the global extremum. The response surface was built during the second training session by including the new training signals shown in Table II.

<table>
<thead>
<tr>
<th>Class I Input</th>
<th>Translation</th>
<th>Scaling Factor</th>
<th>Target Output</th>
<th>Actual Output</th>
<th>Class II Input</th>
<th>Translation</th>
<th>Scaling Factor</th>
<th>Target Output</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>i₁</td>
<td>0</td>
<td>1.00</td>
<td>0.900</td>
<td>0.896</td>
<td>i₁₀</td>
<td>0</td>
<td>1.00</td>
<td>0.100</td>
<td>0.150</td>
</tr>
<tr>
<td>i₂</td>
<td>-16</td>
<td>1.00</td>
<td>0.750</td>
<td>0.758</td>
<td>i₁₁</td>
<td>-16</td>
<td>1.00</td>
<td>0.250</td>
<td>0.255</td>
</tr>
<tr>
<td>i₃</td>
<td>16</td>
<td>1.00</td>
<td>0.750</td>
<td>0.752</td>
<td>i₁₁</td>
<td>16</td>
<td>1.00</td>
<td>0.250</td>
<td>0.281</td>
</tr>
<tr>
<td>i₄</td>
<td>0</td>
<td>0.90</td>
<td>0.750</td>
<td>0.744</td>
<td>i₁₂</td>
<td>0</td>
<td>0.90</td>
<td>0.250</td>
<td>0.219</td>
</tr>
<tr>
<td>i₅</td>
<td>0</td>
<td>1.10</td>
<td>0.750</td>
<td>0.782</td>
<td>i₁₄</td>
<td>0</td>
<td>1.10</td>
<td>0.250</td>
<td>0.253</td>
</tr>
<tr>
<td>i₆</td>
<td>16</td>
<td>0.90</td>
<td>0.600</td>
<td>0.587</td>
<td>i₁₅</td>
<td>16</td>
<td>0.90</td>
<td>0.400</td>
<td>0.378</td>
</tr>
<tr>
<td>i₇</td>
<td>16</td>
<td>1.10</td>
<td>0.600</td>
<td>0.608</td>
<td>i₁₆</td>
<td>16</td>
<td>1.10</td>
<td>0.400</td>
<td>0.407</td>
</tr>
<tr>
<td>i₈</td>
<td>-16</td>
<td>0.90</td>
<td>0.600</td>
<td>0.585</td>
<td>i₁₇</td>
<td>-16</td>
<td>0.90</td>
<td>0.400</td>
<td>0.388</td>
</tr>
<tr>
<td>i₉</td>
<td>-16</td>
<td>0.90</td>
<td>0.600</td>
<td>0.623</td>
<td>i₁₈</td>
<td>-16</td>
<td>0.90</td>
<td>0.400</td>
<td>0.404</td>
</tr>
</tbody>
</table>

Table II. Target and Actual Network Outputs After Training

Ten additional basis functions were included to provide more degrees of freedom.

The response surface created for signal class I is shown in Fig. 4. The surface extremum is a ridge that contains the point (0.1). This should be interpreted as follows. Since the signals
along the ridge have very nearly the same output value as the point (0,1). Each of these signals when affine transformed backwards according to their coordinates will yield very nearly the signal at (0,1). Thus there are many ordered pairs of translation and scaling of the signal at (0,1) that produce signals that are close. For example, if a signal is stretched while anchored at the left edge, shifting to the left will tend to restore the stretched signal back to the original.

The direction of the ridge is correct to only the first order, however. This is the conclusion after backward shifting and translating those signals that lie along the ridge and then superimposing the result on the (0,1) signal. The degree of coincidence was not as good as it might be. The possibility of improving ridge definition by refining the selection of training points in Table II or by changing the set of basis functions is to be investigated.

Figure 4. Response Surface for Class I
C. Classification with Optimal Viewing

To identify the pattern class the presented signal is tested for membership in each of the classes stored in the network. If there are \( n \) classes, then the presented signal must be tested \( n \) times. Each time, the signal is hypothesized to belong to a different class than those already tested. An error function that is the squared difference of the target output for the hypothesized class and the actual network output for the signal is minimized. The minimization is done by searching over \( r \) and \( s \) where the network output is given by Eq. (2) and (10). When all \( n \) classes have been tested, the class corresponding to the least error is the class where the signal belongs.

One can observe qualitatively in Fig. 4 how successful the network will be in determining the affine parameter values for a class 1 signal presented for viewing. If the parameter values lie off the main ridge, the network will determine values that are inaccurate to the degree to which the signals comprising the main ridge are different. That is, the genetic search will terminate on the ridge, but its final position is uncertain, because the peak of the ridge is more nearly a line rather than a point. To improve viewing accuracy, one needs to introduce more curvature, possibly by using a different set of basis functions and more degrees of freedom.

Despite the fact that in this example there is a limit to how well the affine parameters for the presented signal were resolved, the network quite reliably determines to which class the affine transformed input signal belongs. This is seen in Fig. 5. The vertical axis shows the network output after the genetic search was performed for each point in the horizontal plane. Each point on this plane represents an affine transformed copy of a class I training signal. The hypothesis in the upper plot was that the signal was class I and as seen in the figure the network output is almost uniformly 0.9. The lower plot was obtained by hypothesizing the signal was from class II. The network output remains far from 0.1. The top plot is closer to 0.9 than the bottom plot is to 0.1 at all points in the horizontal plane, indicating that every transformed version of the signal is correctly classified. The degree of rejection of a hypothesis is proportional to the difference between the actual network output, after the genetic search has converged, and the target value. This difference should be greatest when the hypothesis is false. The parallel of Fig. 5 for a presented class II signal is shown in Fig. 6. The results presented in Fig. 5 and 6
show that the network is 100 percent effective in classifying the affine transformed heart signals.

Figure 5. Network Classification
V. Conclusions

The ability of the proposed network to classify heart signals was examined. When presented with heart signals that were affine transformed versions of those in the training data, the network
correctly classified them. The results show, however, that the degree to which the automatic viewing feature accurately determines the translation and scaling parameter values is dependent on the quality of the response surfaces built up during training. These surfaces are a function of the set of basis functions. How to select a best set of basis functions is an issue that has not been investigated and will be the subject of future work.

VI. References