Journal of Manufacturing Systems
Vol. 14/No. 1

A Methodology for Manufacturing Process Signature Analysis

Steven D. Eppinger, Christopher D. Huber, and **Van H. Pham,** Massachusetts Institute of Technology, Cambridge, Massachusetts

Abstract

Improvement of control systems entails collection of more information about the process and/or more effective use of that information. We present manufacturing process signature analysis to construct a relationship between collected information (process signatures) and the quality of process output, which can be used for on-line monitoring and control. The general procedure consists of feature extraction, feature selection, and classification.

Extraction of large sets of features from signatures is straightforward, and several classification schemes are available, with neural networks being the most general and powerful. Feature selection, however, is generally quite difficult for complex data structures. We present several feature extraction methods and show that neural networks can be useful in choosing different feature sets. Using a data set from an automated solder joint inspection system, we demonstrate the unique capabilities of neural networks for both feature selection and classification, using more traditional statistical classification techniques as a benchmark.

Keywords: *Manufacturing Process Monitoring, Feature Selection, Manufacturing Process Signature Analysis, Neural Network Applications*

Introduction

Motivation

Most manufacturing processes are monitored for output quality, either continuously or by sampling. Process monitoring assesses whether the process is performing its function adequately so that appropriate corrective actions can be taken if necessary. Inputs to the process monitor include one or more measured attributes of the product or process that reflect the product's ability to perform its intended function.

To improve the quality of manufacturing processes, we seek better monitoring and control systems. One approach is to use more sophisticated measurements. At every stage of even the simplest manufacturing process, there are many opportunities for quality measurements involving process inputs (for example, material thickness or temperature), process attributes (such as feed rate, spindle speed, or tool vibration), and process outputs (such as critical dimensions or surface finish). To monitor a larger process or series of process steps, we could inspect the finished product or assess ultimate customer satisfaction through questionnaires or warranty information. Integration of computers into the manufacturing environment (CIM) facilitates collection of these different process measurements, but information alone is not enough; without an appropriate procedure for analysis and evaluation of this information, the resultant flood of data can confound process improvement efforts.

Our approach to process monitoring is called manufacturing process signature analysis. During each cycle, we measure one or more process signals and/or parameters over the duration of the process. We then analyze these measurements to determine the quality of the process iteration that just took place. Ideally, this quality classification is then fed back through an appropriate controller to close the process control loop.

Related Signature Analysis Work

Although the focus of this paper is on manufacturing applications, we draw on research from a broad range of disciplines. From analysis of human electrocardiogram (ECG) signals (the ECG signature), information about the health of a patient is obtained.[1] The ECG signals are broken down into a series of basic waveform elements called complexes and segments. These are analyzed through syntactic pattern recognition techniques where grammars or rules of syntax classify the ECG signatures.

Computer-access security systems employ a signature consisting of the length of time between keystrokes in the entry of a password.[2] A Bayesian classifier is used to authenticate a given password by comparing the current entry against a known signature.

In manufacturing research, success has been achieved using acoustical signals from machining and forming operations.[3-5] In work involving punch stretching and deep drawing of aluminum sheetmetal,

an acoustical sensor is placed in direct contact with the sheetmetal during the forming process.[6] Energy content, spectral characteristics, and time-series behavior of the resulting signature are then used to identify critical transitions in the forming process.

From this literature, we recognize and adopt a general principle for process signature analysis: parameterize the original signature through extraction of key features and then find an appropriate classification for the signature based on these extracted features. For the ECG signature, the shape parameters of each component complex (features) are used to determine the patient's health (classification). For password security, keystroke timing is used to authenticate the user's identity. For forming, the energy, spectral characteristics, and time-series behavior are used to characterize the forming process. Success of this principle is critically dependent on the ability to locate and extract appropriate features of the original signature.

In this paper, we extend the feature-extraction strategy for signature analysis to include neural network based schemes for feature selection and quality classification. The most important contribution of this paper is the development of a neural network based feature selection scheme that can identify the most useful features for a simplified on-line monitoring system.

To develop our signature analysis approach, we first explain a progression of signature classification tools that serve to highlight the specific utility of neural networks and the important role of feature selection in the process. Next we describe a data set from an automated solder joint inspection project, which was the primary vehicle of research. We then show how the signature analysis methodology is applied to solder joint inspection data. We conclude with a discussion of the strengths and weaknesses of our approach.

Signature Analysis Tools

There are many signature analysis utilities available, from simple statistical process control to advanced pattern recognition techniques using artificial neural networks. Selecting the combination of signature analysis tools most effective for a given problem requires an understanding of the relationship between signature and quality as well as the capabilities and limitations of the analysis tools employed. A progression of classification tools is described below in order of increasing sophistication. Our discussion presents the capabilities and limitations of each tool; successive tools overcome some limitations at the cost of implementation complexity.

Statistical Process Control

One of the most common process monitoring tools in industry today is statistical process control (SPC). Traditionally, SPC entails sampling the process output at given intervals and then measuring some critical attribute of the sampled parts. Control limits are established based on the mean and deviation of the "normal" process, and the process is said to be "out of control" when any data points from process samples fall outside these control limits.

When multiple discrete measurements are required to gain an adequate description of a process, we say that the signature is multidimensional. The basic practice of SPC can be extended to accommodate multidimensional signatures by expanding the one-dimensional control limits into a multidimensional control limit mask. *Figure 1* illustrates an example of a multidimensional signature in the measurement of bias force of a computer's hard disk drive assembly.

With multidimensional SPC, each dimension of the signature has a corresponding pair of control limits determined based on the mean and variance of that dimension, measured for several good-quality drives. The set of control limits then constitutes a control limit mask. The accept or reject decision is based on whether a given signature falls entirely within the bounds of the control limit mask. A commercial system using this basic technique was developed at General Motors Corp. and is now publicly available through Assurance Technologies Inc.[7,8]

While this scheme may be suitable for a variety of process signatures, it is not difficult to imagine cir-

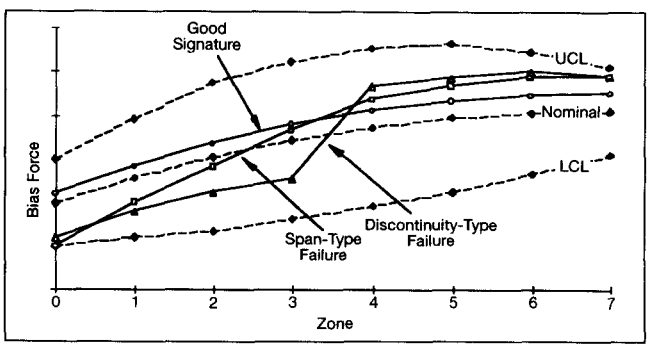

Figure 1
Multidimensional Control Limit Mask

cumstances where application of control limits would produce misleading results. Consider the scenario illustrated in *Figure 1*, where the desired bias force signature follows the form of the nominal signature, and departures from the nominal form constitute failure. The control limit mask does not capture these failures.

Feature Extraction

We can avoid the above difficulties through feature extraction. Feature extraction reduces the amount of data we must process without discarding useful information. Consider the bias force signature shown in *Figure 2*; in this case, two features are extracted from the original signature—the slope of the best-fit line and the range of the signature's extreme points. By plotting slope versus range for many samples of hard disk drive assemblies, we arrive at a feature-space scatter plot as shown in *Figure 3*. Three signature classes become apparent by the clustering of data points. Class 1 in the figure corresponds to good signatures, which follow the form of the nominal curve (both range and slope are low), Class 2 (higher range) corresponds to signatures with a discontinuity, and Class 3 (higher slope and range) corresponds to signatures that span the control limit range without a discontinuity.

In the bias force example we extracted two features, and the resulting feature-space plot is two-dimensional. There are also two one-dimensional feature spaces associated with each individual feature. These one-dimensional spaces, shown opposite the axes in *Figure 3*, are simpler to interpret; however, the clustering of classes that we observed in the two-dimensional space may not be so apparent. As the number of features grows, the number of feature spaces increases tremendously. For a signature described by 23 features, as with the data set used for this research, there are more than 8 million distinct feature spaces. Before we address the question of which of these 8 million feature spaces is most effective for the given problem, a formulation for the classification decision function within the context of feature space must be examined.

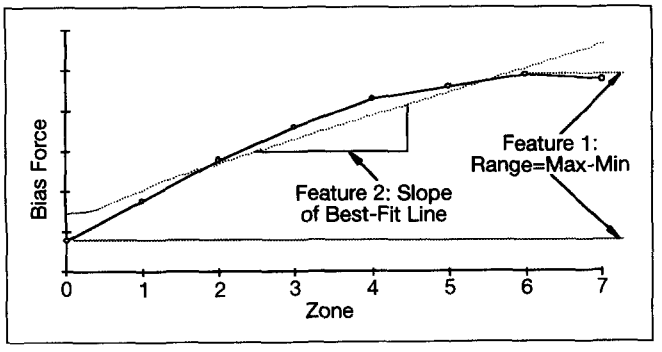

Figure 2
Feature Extraction

Linear Statistical Classification

Classification within the context of feature space requires a mathematical formulation fundamentally different from the control limits of SPC. The objective here is to formulate a function whose output indicates the signature class. Unlike the accept/reject decision of SPC, we can now consider multiple classes of process output. In this section we describe the traditional linear statistical classifier.

Bayes Classifier

In geometric terms, a linear statistical classifier parameterizes each class of data with its mean and covariance. (The geometric interpretation given here cannot describe all of the details of statistical classification; rather the purpose is to provide an intuitive understanding of the essence of statistical classification.) Based on these parameters, an ellipse is constructed about each class of data, as shown in *Figure 4*. The common secant between two ellipses then

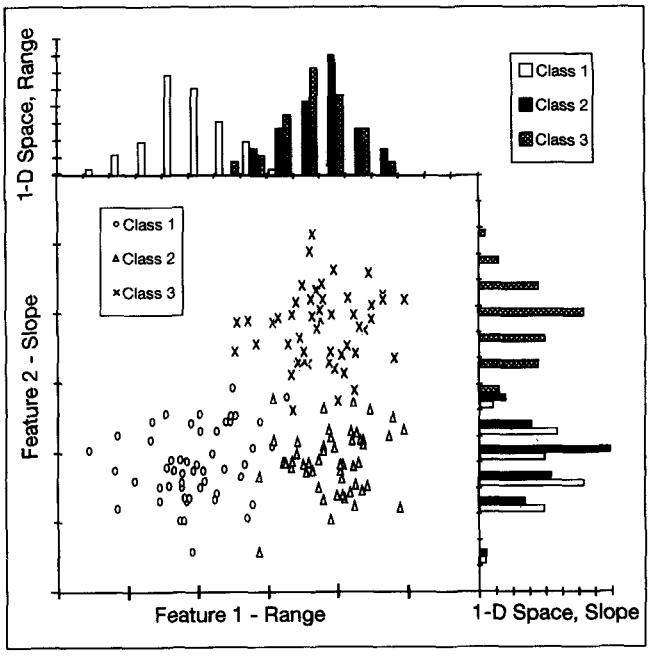

Figure 3
Feature Space

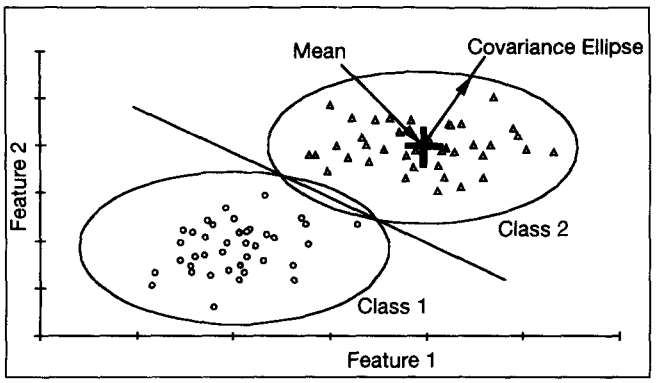

Figure 4
Construction of Linear Statistical Classifier (Bayes)

describes the linear decision function separating the two classes of data.

In mathematical terms, construction of a linear statistical classifier (Bayesian) begins with the assumption that each class of data can be described by a normal probability density function.[9] Given this assumption, the likelihood that a given signature comes from a given class, ω_i, is as follows:

$$p(x|\omega_i) = \frac{1}{(2\pi)^{n/2}|C_i|^{1/2}} \exp \left[-\frac{1}{2}(x-m_i)^T C_i^{-1}(x-m_i) \right] i = 1...M$$

where:

x = feature vector for a given signature
ω_i = ith class of data
n = number of features
C_i = covariance matrix for class i
m = mean vector for class i
M = number of classes

For a one-dimensional feature vector, the above expression reduces as follows:

$$p(x|\omega_i) = \frac{1}{\sqrt{2\pi}\,\sigma_i} \exp \left[-\frac{1}{2}\left(\frac{x-m_i}{\sigma_i} \right)^2 \right] i = 1...M$$

where:

σ_i = standard deviation of class i

The classification decision for a signature characterized by a feature vector involves calculating the likelihood function for each possible class and then selecting the class with the largest corresponding likelihood: classify signature, x, as class ω_i, if $p(x|\omega_i) > p(x|\omega_j)$ for $i \neq j$). The application of a linear statistical classifier is restricted to those problems that conform to its underlying assumptions, as follows:

1. Normality: Each feature in the feature vector is normally distributed about some mean.
2. Linear Separability: Each class of output is linearly separable from all other classes.

Prediction of Error

We would like to quickly evaluate how much class-discerning information a set of signatures carries, without constructing and testing a complete linear statistical classifier. In predicting the success rate of the linear classifier we can also estimate the utility of each feature in the overall classifier performance. To do this efficiently, we do not examine every individual signature, but we evaluate the data set as a whole and predict the expected contribution of each feature.

Our prediction of error begins with a measure of feature quality that is based on a set of features' ability to separate the classes. The measure is called the Mahalanobis distance, given as follows:[9]

$$FQ_{ij} = (m_i - m_j)^T C_{ij}^{-1} (m_i - m_j)$$

$$C_{ij} = \frac{1}{2}(C_i + C_j)$$

where:

FQ_{ij} = Mahalanobis distance between class i and class j (feature quality with respect to class i and class j)
m_i = mean vector for class i
m_j = mean vector for class j
C_{ij} = effective covariance matrix for class i and class j
C_i = covariance matrix for class i
C_j = covariance matrix for class j

In one dimension (for one feature), the above expressions can be reduced to the following:

$$FQ_{ij} = \frac{(m_i - m_j)^2}{\sigma_{ij}^2}$$

$$\sigma_{ij} = \frac{1}{2}(\sigma_i + \sigma_j)$$

where:

σ_{ij} = effective standard deviation for class i and class j
σ_i = standard deviation of class i
σ_j = standard deviation of class j

In essence, the Mahalanobis distance is a feature quality score based on the signal-to-noise ratio between two classes of data. The "signal" is the distance between the classes in feature space, and the "noise" is the variation or spread of each class. From this feature quality score, a prediction of error can be calculated as follows:

$$P(e) = 1 - \text{erf}\left(\sqrt{FQ/4}\right)$$

where:

$$\text{erf}(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}y^2\right) dy$$

Besides the assumptions of a linear statistical classifier (normality and linear separability), the prediction of error as defined above is subject to a third assumption: covariance matrices for each class of data are approximately equal. In calculating the feature quality score between two classes of data, we use an effective covariance matrix, which is a simple averaging of the covariance matrices for each individual class. If the covariance matrices are not similar, then the effective covariance matrix does not accurately describe the spread of the class' respective distributions.

Neural Network Pattern Classification

All of the classification tools presented thus far have their basis in normal, linear statistical theory. As such, any application to which these tools can be reliably applied is restricted to normal and linear domains; however, many process measurements are not so well behaved. While these ill-behaved signatures may have significant content (class-discerning information), traditional classification tools are not able to extract this content. We now turn to artificial neural networks as an alternative computational classification technique. Artificial neural networks make classification decisions by a fundamentally different mechanism, so they are not encumbered by the rules of normal, linear statistics. While we provide a brief introduction to artificial neural networks here, there are several sources of more detailed explanations.[10,11]

Like the Bayes classifier, a neural network produces a mathematical function whose input is the process signature (feature vector $\boldsymbol{x}$) and whose output is the classification decision. The way we arrive at the mathematical function is fundamentally different, though. Formulation of a neural network also begins with a training set of signatures with known classifications. Each signature is applied to the input nodes of a network of simple processing elements. Numerical values propagate through the network via nonlinear weighted connections and eventually result in a classification decision at the output nodes. In all likelihood, this decision will be incorrect (output values will not correspond to the known classifications) because the weights within the network structure are initially randomized.

At this point, a learning algorithm compares the actual network output with the desired output. Then based on the error, weights within the network structure are modified. After many iterations, the network eventually "learns" to distinguish among signatures from various classes. In formulating a decision function with a neural network, no assumptions are made regarding the underlying distribution of the data or linear separability. Instead, the motivating force behind the learning algorithm is the minimization of error. (Neural networks can be confused by the presence of local minima. One solution to this problem is to simply train multiple networks with different initial starting weights. Alternatively, Gaussian noise can be added to the neural network inputs, which will serve to "bump" the solution out of local minima.[12]) As such, a neural network will attempt to find the line or surface that most effectively separates the classes.

After the network has been trained, each new signature for classification, characterized by a feature vector $\boldsymbol{x}$, is simply applied to the input nodes of the network. Output values determine which class the signature best matches. The classification decision function, therefore, is the nonlinear mapping found in the network training stage.

The Perceptron

The fundamental processing element in the neural network classifier used for this study is the perceptron, introduced by Rosenblatt.[13] A perceptron, shown

in *Figure 5*, calculates the weighted sum of its inputs and passes the result through a nonlinear thresholding function. The thresholding function shown in the figure is a simple signum function. Some other common threshold functions used in neural networks include the hyperbolic tangent and the sigmoid. The nonlinear threshold function allows a neural network to extend the reach of pattern classification into the domain of generalized nonlinear functions.

Multilayer Feed-Forward Neural Network

The neural network structure used for this study is a multilayer feed-forward neural network that uses the back-propagation learning algorithm. A general schematic of this network is shown in *Figure 6*. The input layer has one node for each feature extracted from the raw signature. Succeeding layers of the network consist of one or more perceptron nodes. The output layer, where the classification decision emerges, also consists of one or more perceptron nodes. The actual number of output nodes depends on the number of possible classes in the data set as well as on the way we code the different classes. For instance, only one output node is needed for a two-class problem where an output of +1 corresponds to the first class and an output of −1 corresponds to the second class. In problems that involve a larger number of classes, we could assign one output node to each possible class or we could encode each class as a binary number, thereby reducing the number of output nodes needed to identify each distinct class. Layers between the input and output layers are called hidden layers. Lines between various nodes in the network represent weighted connections through which processing elements communicate. (The bias node in *Figure 6* acts like an input layer node except that it is held at a constant value. The bias node allows the neural network to shift a decision function away from the origin in feature space.)

The neural network structure described above, in essence, represents a complex nonlinear function. The learning algorithm adjusts the parameters of the nonlinear function until the classification error is minimized. Given a sufficiently complex topology, a neural network could eventually learn to correctly classify with zero errors. In general, this is not a desirable result because the neural network is functioning as a lookup table (memorizing) instead of a generalized classifier. Excessive complexity in the network structure prevents the network from making generalized decisions. Several authors offer guidelines on selection of a network structure.[11,14,15]

Neural Networks in the Presence of Nonlinearity and Nonnormality

There are two important situations in which a neural network is particularly useful. The first case is when a nonlinear decision function is required to separate two classes of data. The second case is for nonnormal data distributions, as illustrated by *Figure 7*. The outliers in the distributions invalidate the assumption of normality. A statistical classifier constructs a decision function at the intersection of the normal density functions for the two classes. In contrast, a neural network avoids these difficulties by ignoring the assumptions and instead moves the decision function with each iteration of the learning process until classification error is minimized.

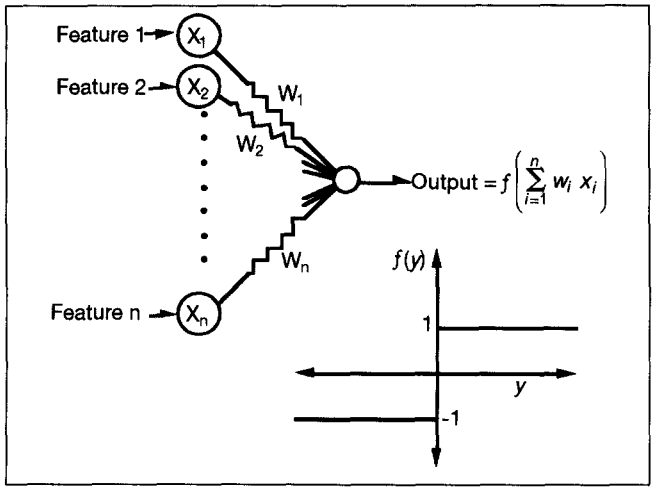

Figure 5
The Perceptron

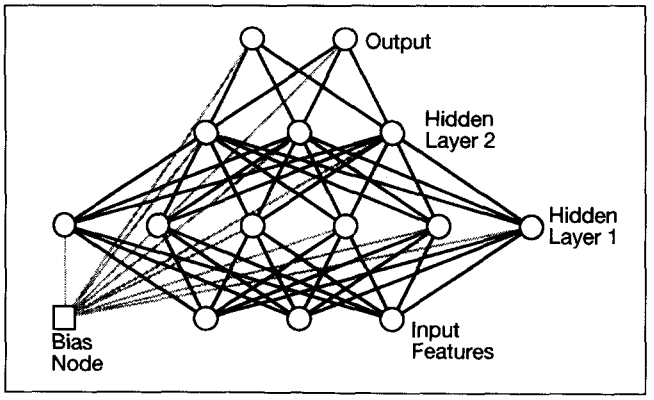

Figure 6
Multilayer Feed-Forward Neural Network

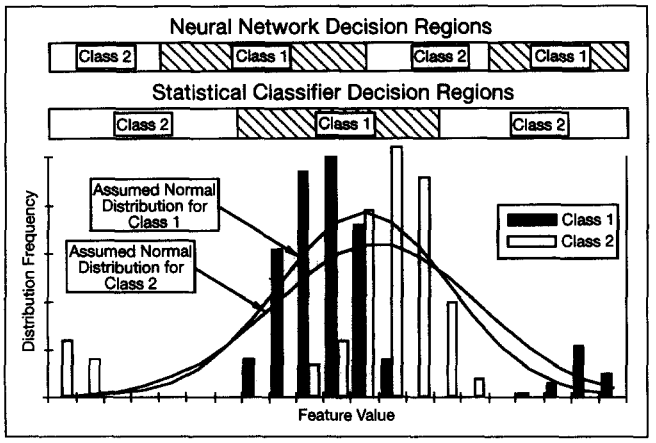

Figure 7
Classification Decision Regions for Nonnormal Distributions

Feature Selection

Given a perfect understanding of a process and its corresponding signatures, we could theoretically parameterize the signature with a finite set of descriptive features without discarding useful signature content. In reality, feature extraction may involve a significant amount of guesswork in proposing features because we must contend with an imperfect understanding of the process. It is generally necessary to parameterize the signature with an abundance of features in the hope that a subset will yield useful information. The difficulty associated with this approach is that the remaining features that do not add useful information do add complexity to the classification problem. Clearly, if the goal is development of an on-line process control tool, useless features should be eliminated.

Feature selection is the process of eliminating those features that do not contribute toward the goal of discriminating among different classes of output. Four different heuristic algorithms were investigated during this research:

1. One-dimensional prediction of error
2. One-dimensional statistical classification
3. One-dimensional neural network classification
4. Multidimensional neural network first-layer weights

A description of each selection algorithm follows, along with respective advantages and drawbacks.

One-Dimensional Prediction of Error

The one-dimensional prediction of error scheme, unlike the other schemes, is limited to two-class problems. (Prediction of error, as defined earlier, is based on the Mahalanobis distance between two classes of data.) In a two-class problem, the utility of each feature is examined individually through normal, linear statistics according to the following algorithm:

1. Calculate the mean and standard deviation for each feature within each class.
2. Calculate the one-dimensional Mahalanobis distance between classes for each feature.
3. Calculate a prediction of classification error for each feature.
4. Rank the features based on the prediction of classification error.

Predictions that result from the algorithm above represent the classification error that would be incurred if the given feature was the only information available (interaction among features is ignored). If the data set conforms to the three assumptions associated with a prediction of error—normality, linear separability, and similar covariance among classes—this scheme provides the least costly (in terms of computation time) evaluation of the feature set.

One-Dimensional Statistical Classification

With one-dimensional statistical classification, like one-dimensional prediction of error, each feature is examined individually, but the data are applied to a statistical classifier. Steps in the one-dimensional statistical classification algorithm are as follows:

1. Calculate the mean and standard deviation for each feature within each class.
2. Apply each signature in the data set to the one-dimensional likelihood function for each class.
3. Classify each signature according to magnitude of the likelihood function for each class.
4. Determine the classification error by comparing the computed classifications against the known signature classifications.
5. Rank the features based on the classification error for each feature.

The one-dimensional statistical classification scheme has the following advantages over the one-dimensional prediction of error scheme:

1. Statistical classification is not restricted to two-class problems.

2. Assumption of similar covariance among classes is eliminated.
3. Actual data are applied to the classifier, helping to expose deviation from normal, linear statistics.

The first two advantages are self-explanatory, but the third deserves closer examination. The prediction of error scheme employs the assumptions of normal, linear statistics and then decides the classification error based on what the data would look like if it conformed to these assumptions. The statistical classifier scheme employs the same assumptions, but instead of relying on those assumptions, they are tested against the actual data. Deviation from the assumptions of normal, linear statistics can then be exposed in the form of higher misclassification rates. The prediction of error scheme, though less costly in terms of time and computing resources, is more easily fooled because it makes only indirect use of the data in arriving at its conclusion.

One-Dimensional Neural Network Classification

With the one-dimensional neural network scheme, like the preceding schemes, the utility of each feature is examined in isolation from the other features. This scheme is similar to the classification scheme except that a neural network classifier is used in place of normal, linear statistics. The algorithm is as follows:

1. Train a one-input neural network for each feature.
2. Determine the classification error of each one-input neural network by comparing actual output against desired output.
3. Rank the features based on the classification error for each feature.

The following neural network structure was used for this purpose:

1. Back-propagation learning algorithm
2. Hyperbolic-tangent thresholding function
3. One input node for the feature being evaluated
4. Four perceptron nodes in the first hidden layer
5. Two perceptron nodes in the second hidden layer
6. Number of output nodes depends on how many classes of output exist. For a two-class problem, one output node is sufficient.

The primary advantage of a neural network classifier is its independence from normality and linearity issues. Minimization of error underlies the iterative learning algorithm of neural networks, with no assumption of normality. Furthermore, the thresholding function in each processing element allows the neural network to twist and contort the decision function in a manner which is not available through traditional statistics.

Multidimensional Neural Network First-Layer Weights

Each selection scheme presented above, including one-dimensional neural network classification, evaluates the utility of each feature in isolation from the other features. Interactions between features are ignored. In contrast, the first-layer weight scheme takes a global view of the full feature set. The first-layer weight algorithm is as follows:

1. Train a multidimensional neural network using the full feature set as input to the network.
2. Determine the sum of the absolute values of the first-layer weights associated with each feature.
3. Rank the features based on the sum of first-layer weights.

The following neural network structure was used for the first-layer weight scheme:

1. Back-propagation learning algorithm
2. Hyperbolic-tangent thresholding function
3. N input nodes (one for each feature)
4. $2N$ perceptron nodes in the first hidden layer
5. N perceptron nodes in the second hidden layer
6. Number of output nodes depends on how many classes of output exist. For a two-class problem, one output node is sufficient.

The significance of first-layer weights in a multidimensional neural network can best be explained by reexamining the general structure of a neural network as shown in *Figure 6*. Consider all of the lines representing the weighted connections between a single input node and the first hidden layer. With the first-layer weight scheme, the importance of a given feature is measured by the sum of the absolute values of this feature's first-layer weights. In the case of all of the weights being zero or nearly zero, the feature is effectively filtered from the classification decision. Conversely, if the weights are large, the feature will have a greater effect on the classification decision, thus the rationale behind the first-layer weight

scheme. This scheme can recognize nonnormality and nonlinearity with the added advantage of recognizing interaction among features.

The first-layer weight scheme does not explicitly address the presence of redundant features, although we would expect the first-layer weights associated with redundant features to be similar. While redundant features will not hinder the accuracy of the network, elimination of these features would further reduce computational complexity. To further simplify the network, we must examine the weights associated with each feature and identify any similarities. To verify redundancy, we must then eliminate all but one of the redundant features and retrain the network. If the classification error is unchanged, redundant features have been successfully identified and removed.

Selecting the Right Features

Each feature selection scheme described above simply ranks the utility of the features. The question remains of how many features to employ in an actual classifier. One solution to this problem is to test a succession of classifiers, adding additional features with each test. When performance of the classifier does not improve significantly with the addition of new features, feature selection is complete. Alternatively, feature selection scores can be examined and the feature cutoff can be placed where there is a significant drop in the feature selection score. For this research, we directly compared the four feature selection schemes' abilities to find the best features, so we used an arbitrary cutoff of the best five features for each scheme.

Choosing a Classifier

Once an appropriate subset of features has been selected, the task of constructing a classifier remains. If a given data set conforms to the assumptions of normality and linear separability, there is no need to expend additional time and energy developing a neural network classifier. A neural network and a statistical classifier will arrive at the same decision function for well-behaved data sets. If a given data set does not conform to the assumptions of normality and linear separability, a neural network can be a highly effective solution. While the amount of time required to develop a neural network is significantly greater than a statistical classifier, cycle time of the actual classifier is not necessarily significantly greater. If there is any question regarding the integrity of the data with respect to the assumptions of normal, linear statistics, then a neural network is worth investigating.

Automated Solder Joint Inspection

The data set that was the primary vehicle of research for this paper came from an automated solder joint inspection project at Digital Equipment Corp. The goal of automation was replacement of manual visual inspection methods for surface-mount electronic components (SMD technology). With surface-mount technology, printed circuit boards are screened with solder paste and then the components are placed onto the boards using automated equipment. The boards are then heated to reflow the solder, creating the lead bonds. Although surface-mount technology allows the electronics industry to continue reducing the size of components, new problems have been introduced. In particular, very high lead density causes manual visual inspection methods to become less and less effective (as measured by higher classification error). Inefficiency drives the need for redundant inspection and testing, which in turn provides the impetus for automated inspection methods.

Modes of Solder Joint Failure

Among the common problems associated with the SMD assembly process is the application of incorrect solder paste volume. Insufficient or excess solder conditions have been correlated with premature failure of the solder bond due to thermal cycling of the circuit. Electronic testing of circuit boards can detect more severe classes of defects, such as short and open circuits, but cannot guarantee that a board will remain defect-free over its life.

The automated solder joint inspection system at Digital[16] is illustrated in *Figure 8*. A laser is directed at a single component lead while two infrared camera sensors monitor the temperature of the solder joint as it is heated and as it cools. These signals are digitized and stored for off-line analysis. Eventually these data would be utilized on-line with the results of our signature analysis scheme.

These data represent the time derivative of the lead temperatures as observed by the two different cameras. Each camera extracts energy in a different infrared wavelength band. The result is a pair of thermal signatures for the solder joint, with each signature consisting of a discrete time series of 500

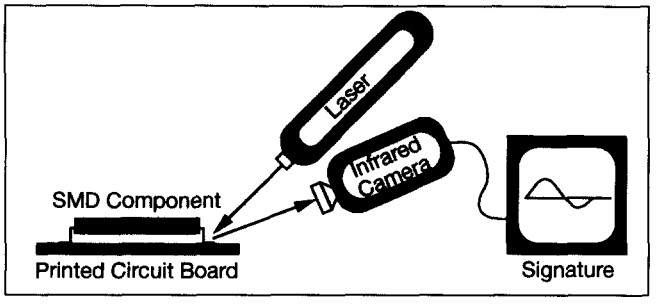

Figure 8
Automated Solder Joint Inspection

data points (*Figure 9*). The physics underlying the success of this inspection method involves the heat transfer characteristics of the solder joints. The hypothesis is as follows: If the lead is well bonded, the board will conduct heat quickly away from the lead. If the connection is insufficient, heat transfer will be slower and the lead will heat excessively. If the solder mass is excessive, the joint will also retain more heat and will heat up slower. Accurate physical models of this phenomenon have not been proven.

Data Collection

The data for this study were collected by specially fabricating a series of circuit boards under the three solder paste conditions: excess, normal, and insufficient. (These data were part of a larger experiment at Digital comparing the performance of several automated solder joint inspection schemes and also other defect types that are not included in our data set.) SMD components were placed onto the solder pads and then the boards were heated in the conventional reflow process. Finally, each of the solder joints was inspected using the laser/infrared system described above. The solder joint data set consisted of 3510 signatures—1510 from solder joints with excess solder, 1688 from solder joints with insufficient solder, and 312 from solder joints with normal solder. Given three classes of output, the signature analysis methodology can be applied to four distinct classification problems—three associated with class pairs and one associated with all three classes. From a manufacturing standpoint, a classifier that can distinguish between insufficient and good solder conditions but ignores the existence of excess solder conditions is of marginal utility. From a research standpoint, however, each of these four problems can be employed in our exploration of the principles of signature analysis methodology. As such, all four classi-

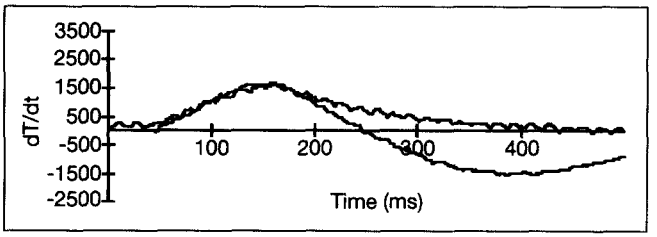

Figure 9
Typical Thermal Signature

fication decisions (excess-good, excess-insufficient, insufficient-good, and excess-insufficient-good) were investigated during this research.

Feature Extraction

For each signature, 23 numerical features were extracted from the raw data by analyzing the two signal curves. In developing our list of candidate features, we drew first from our physical understanding of the process to define appropriate features. As an example, heat transfer theory suggests that the area under the temperature derivative curve up to the first zero crossing (see *Figure 9*) should be proportional to the total amount of heat absorbed by the solder joint during the measurement (features 14 and 15). Other physical features included peak times and peak values (features 1, 2, 4, and 5), minimum times and values (features 16-19), zero crossing times (features 13 and 14), and the ratio of the peak values (feature 3). The remaining features were statistical quantities computed for each signature, such as mean, variance, and RMS values (features 6-11). We also computed three features based on the "difference curve" obtained by subtracting one signal curve from the other (features 20-22). The final feature represented the RMS value of an "error curve" obtained by subtracting the signal from the average of all good signal curves (feature 23).

For this example application, feature extraction was conducted on an intuitive basis, although more interesting alternatives are available. Of particular interest is a method of feature extraction based on wavelet decomposition.[17]

Feature Selection

Each of the four feature selection schemes described above was investigated to determine the success of each scheme and to test our understanding of the behavior of each scheme. The one-dimen-

sional prediction of error scheme was applied to only the two-class problems associated with distinct class pairs (excess-good, excess-insufficient, and insufficient-good), whereas the remaining schemes were applied to all four classification problems. Each feature selection scheme produced a feature ranking for each applicable classification problem. An evaluation of these rankings was carried out by selecting the top five features in each ranking and then calculating a prediction of error, statistical classification error, and neural network classification error for each five-feature subset.

Signature Analysis Results

Results of the signature analysis computations discussed in preceding sections of this paper are presented in *Table 1*. The four column groups in this table correspond to the four distinct classification problems, while the rows give the classification errors for individual features, feature subsets, and the full feature set. For example, results from the excess-insufficient problem are in the first column group: The prediction of error for feature Sig. $Peak_1$, acting alone, was 32%, while the statistical and neural network classifiers had classification errors of 20% and 18%, respectively. For the same problem, the first-layer weight scheme produced a subset of five features that resulted in a prediction of 17% error and statistical and neural network classification errors of 17% and 14%, respectively. The next few sections of this paper are devoted to an explanation of the characteristics and significance of the results in *Table 1*.

Generation of Data Sets for Analysis

In the formulation and training of the various classification tools, random subsets of the full data set were used. These subsets consisted of 500 excess

Table 1
Signature Analysis Classification Results

Individual Features	Excess-Insufficient			Insufficient-Good			Excess-Good			Excess-Good-Insufficient	
	P(e)	Class	NNet	P(e)	Class	NNet	P(e)	Class	NNet	Class	NNet
1 Sig. Peak 1	32%	20%	18%	26%	18%	17%	50%	67%	DNC	51%	32%
2 Sig. Peak 2	37%	21%	17%	26%	17%	DNC	48%	66%	DNC	52%	30%
3 Peak Ratio	48%	44%	DNC	49%	61%	DNC	49%	64%	DNC	76%	DNC
4 Peak Time 1	47%	48%	DNC	30%	29%	26%	43%	62%	DNC	58%	DNC
5 Peak Time 2	46%	49%	DNC	30%	31%	DNC	42%	58%	DNC	59%	DNC
6 Mean 1	42%	48%	42%	43%	65%	DNC	45%	53%	DNC	71%	53%
7 Mean 2	28%	21%	20%	28%	20%	16%	49%	47%	DNC	39%	35%
8 Variance 1	19%	16%	16%	20%	18%	15%	48%	43%	DNC	36%	31%
9 Variance 2	43%	50%	DNC	39%	55%	DNC	48%	66%	DNC	80%	DNC
10 Sig. RMS 1	20%	16%	16%	21%	17%	13%	48%	50%	DNC	39%	29%
11 Sig. RMS 2	30%	19%	18%	27%	19%	DNC	49%	62%	DNC	52%	31%
12 Zero Xing 1	44%	50%	DNC	45%	62%	DNC	47%	53%	DNC	73%	DNC
13 Zero Xing 2	43%	41%	41%	29%	37%	DNC	37%	50%	DNC	64%	50%
14 Area Below 1	27%	17%	18%	27%	16%	15%	50%	40%	DNC	34%	31%
15 Area Below 2	27%	22%	21%	27%	20%	18%	50%	55%	DNC	48%	35%
16 Sig. Min 1	39%	30%	18%	24%	23%	15%	49%	67%	DNC	56%	DNC
17 Sig. Min 2	49%	50%	DNC	44%	48%	DNC	46%	60%	DNC	69%	DNC
18 Min Time 1	49%	45%	DNC	45%	35%	DNC	48%	65%	DNC	60%	DNC
19 Min Time 2	38%	39%	39%	41%	28%	DNC	47%	56%	DNC	49%	48%
20 Peak of Diff	50%	52%	DNC	43%	46%	24%	46%	69%	DNC	70%	DNC
21 Min of Diff	39%	32%	DNC	26%	26%	DNC	49%	67%	DNC	58%	DNC
22 Zero Xing Diff	50%	49%	DNC	50%	66%	DNC	50%	67%	DNC	79%	DNC
23 ERMS 2	45%	45%	DNC	47%	71%	DNC	46%	67%	DNC	81%	DNC
Best 5 Features											
1-D Prediction of Error	17%	22%	17%	19%	38%	15%	34%	54%	27%	N/A	N/A
1-D Statistical Classification	17%	18%	14%	16%	25%	12%	46%	57%	DNC	59%	28%
1-D Neural Net Classification	17%	18%	14%	18%	33%	12%	N/A	N/A	N/A	63%	29%
First-Layer Weights	17%	17%	14%	15%	26%	9%	33%	44%	26%	57%	26%
Full Feature Set	14%	21%	13%	9%	24%	9%	24%	56%	22%	52%	27%

signatures, 500 insufficient signatures, and 200 good signatures. For the statistical tools (prediction of error and statistical classification), one subset was used for training and testing. For neural network classification, two subsets were drawn from the full data set, one for training and one for testing. The reason for using two different subsets for training and testing was suggested earlier: Improper selection of the neural network structure can cause the neural network to act as a lookup table instead of a generalized decision function. By training the neural network with one data set and testing it with another, we can verify formulation of a generalized decision function.

Explanation of Anomalies in Results Table

In several instances, neural network classification was unsuccessful because the neural network was not able to converge on a decision function. DNC entries in *Table 1* are instances where the neural network "did not converge." Whenever inputs to a neural network do not contain sufficient content (class-discerning information), a neural network learns to select the class that it saw most frequently during training, regardless of input. For the excess-good classification problem, the neural networks were trained with 500 excess signatures and 200 good signatures. In this case, a neural network that did not converge learned to classify every signature as excess because it was presented with more excess signatures during training. In actuality, this corresponds to a classification error rate of 29% (200/700); however, we enter DNC in the results table. In almost every case where the neural network could not converge, such as feature Sig. $Peak_1$ in the excess-good problem, the statistical prediction of error (50%) and statistical classification error (67%) were even less useful.

N/A entries in *Table 1* denote two situations where classification error could not be calculated. In the first situation, the individual neural net feature selection scheme failed for the excess-good classification problem because none of the individual neural networks could converge on a decision function. In the second instance, the prediction of error scheme failed for the excess-insufficient-good classification problem because prediction of error was not defined for the three-class problem. As a result, these schemes could not offer a ranking of features.

Feature Content and Signature Content

For the excess-insufficient problem, feature Sig. Min_2 produced prediction, statistical, and neural network classification errors of 49%, 50%, and DNC, respectively. Clearly, this feature cannot offer any more than a random guess as to the appropriate classification. This is only one of many examples where the various classification tools were not able to perform any better than the toss of a coin. These high classification errors occur due to lack of signature content. The greatest challenge in the development of a process-monitoring system involves what in this paper we call signature content, feature quality, or simply observability. In the extreme case, if we choose an inappropriate sensor measurement that does not contain the information needed to predict quality, then an entirely different measurement scheme must be chosen. Techniques presented in this paper cannot remedy this situation, as exemplified by the excess-good classification problem.

Neural Network Classification vs. Statistical Tools

An interesting scenario arises when a neural network classifier can perform with reasonable accuracy, while the statistical tools fail. In these cases, the neural network has found a better nonlinear or non-normal decision function that was invisible to traditional classification tools. As an example, in the insufficient-good problem the first-layer weight scheme produced a subset of features that generated a 9% error rate with the neural network and a 26% error rate with the statistical classifier. Given the same information, the neural network performed significantly better.

Throughout *Table 1*, the neural network classification error is lower than both the statistical prediction of error and the statistical classification error except in a few instances where the various error rates are similar to within a 1-2% noise factor. (Recall that random subsets of data were drawn from the full data set for the development of each tool. The noise factor is the result of the slight differences in each random subset of data.) This is a direct result of the iterative learning process and nonlinear decision functions of neural networks. While statistical tools are confounded by ill-behaved data, neural networks exploit these differences. Conversely, when the assumptions of normality and linear separability are fairly accurate, performance of statistical tools

and neural networks is similar. If the assumptions of normal, linear statistics hold true, neural networks and statistical tools will arrive at the same decision function and there is no need to implement a neural network classifier.

Full Feature Set Outperforms Feature Subsets

Neural networks using the full feature set invariably performed better than any individual feature and better than any subset of features. This suggests that neural networks can employ interactions among features, thereby generating a decision that is better than the sum of its parts yet is still able to effectively filter those features that do not add useful information to the problem. As an example, consider the insufficient-good problem. In this case, the best individual feature (Sig. RMS_1) produced a 13% classification error, whereas neural network classification with the full feature set produced a 9% classification error. This was accomplished although more than half the features had no signature content at all (DNC). If the feature extraction process entails a significant amount of guesswork, we can be sure that a neural network will filter poor guesses. This is not the case with the statistical tools. In the excess-insufficient problem, the best individual feature (Sig. RMS_1) produced a statistical classification error of 16%, whereas the full feature set could do no better than 21%. Those features that lack content, while filtered out of the neural network decision function, serve to confuse statistical decision functions.

Evaluation of Feature Selection Schemes

In evaluating various feature selection schemes, we are most interested in those schemes that selected features with the most content. In all four classification problems, the first-layer weight scheme produced the lowest neural network classification error. In fact, the first-layer weight scheme produced a subset of features that performed nearly as well as the neural network classifiers that used the full feature set. Consider the insufficient-good problem; the first-layer weight scheme produced a subset of features with a neural network classification error of 9%, which was the same error produced by the full feature set. The remaining tools and schemes produced classification errors ranging from 12-38%. By eliminating useless features, we gain significant computational advantages without sacrificing classification accuracy.

Normality Issues

To evaluate the effect of normality on various classification tools, normality scores were calculated for each feature in each class. These scores are presented in *Table 2* along with results of the excess-insufficient classification problem. The normality score calculation consists of the Pearson product-moment correlation between the given variable (a particular feature from a particular class) and the corresponding normal scores (standardized z-score).

In the event that a given data set conforms to the assumptions of normal, linear statistics, we would expect that the prediction of error, statistical classification error, and neural network classification error would reflect similar results.[18] Results shown in *Table 2* for the features $Mean_2$, Sig. RMS_1, Zero $Xing_2$, and Area $Below_2$ (features 7, 10, 13, and 15, respectively) tend to support this hypothesis. Conversely, we might expect that poor normality scores would produce results where statistical errors were significantly

Table 2
Excess-Insufficient Classification and Normality

		Excess-Insufficient			Normality	
Individual Features		**P(e)**	**Class**	**NNet**	**E**	**I**
1	Sig. Peak 1	32%	20%	18%	0.70	0.90
2	Sig. Peak 2	37%	21%	17%	0.60	0.55
3	Peak Ratio	48%	44%	DNC	0.89	0.87
4	Peak Time 1	47%	48%	DNC	0.40	0.91
5	Peak Time 2	46%	49%	DNC	0.43	0.54
6	Mean 1	42%	48%	42%	0.67	0.60
7	Mean 2	28%	21%	20%	0.95	0.95
8	Variance 1	19%	16%	16%	0.95	0.86
9	Variance 2	43%	50%	DNC	0.32	0.25
10	Sig. RMS 1	20%	16%	16%	0.97	0.91
11	Sig. RMS 2	30%	19%	18%	0.82	0.68
12	Zero Xing 1	44%	50%	DNC	0.72	0.70
13	Zero Xing 2	43%	41%	41%	0.95	0.98
14	Area Below 1	27%	17%	18%	0.99	0.89
15	Area Below 2	27%	22%	21%	0.98	0.96
16	Sig. Min 1	39%	30%	18%	0.51	0.78
17	Sig. Min 2	49%	50%	DNC	0.30	0.26
18	Min Time 1	49%	45%	DNC	0.79	0.90
19	Min Time 2	38%	39%	39%	0.81	0.60
20	Peak of Diff	50%	52%	DNC	0.33	0.50
21	Min of Diff	39%	32%	DNC	0.46	0.59
22	Zero Xing Diff	50%	49%	DNC	0.75	0.71
23	ERMS 2	45%	45%	DNC	0.59	0.48
Best 5 Features						
1-D Prediction of Error		17%	22%	17%		
1-D Statistical Classification		17%	18%	14%		
1-D Neural Net Classification		17%	18%	14%		
First-Layer Weights		17%	17%	14%		
Full Feature Set		14%	21%	13%		

higher than neural network errors; however, results do not corroborate this hypothesis. Consider the feature Sig. RMS_2 (feature 11); here the normality scores for the excess and insufficient classes are .82 and .68, respectively, yet the statistical and neural network classification errors are 19% and 18%, respectively. This can be explained by noting that poor normality implies that the normal probability model does not accurately represent the data and any subsequent decision function may or may not separate the data. Low statistical classification error under these circumstances cannot be taken as reliable.

Conclusion

Improvement of process control systems is of critical importance to manufacturing firms today. In pursuit of rapid improvement, we must make effective use of the wealth of process information now made available by on-line sensors and computation. The signature analysis methodology presented in this paper is offered as a formal approach to this challenge. The three-step analysis method (feature extraction, feature selection, and classification) accepts complex process signatures and their corresponding quality classifications as input and produces a relationship between the signatures and the quality of process output.

In applying this method to a data set from an automated solder joint inspection system, artificial neural networks were found to offer significant performance advantages over traditional classification tools, yet this performance was achieved at the expense of significant computational resources. To reduce computational cost of classification, we must reduce the complexity of the classification problem by employing only the most useful features. We call this the feature selection problem. Again we turn to the unique abilities of artificial neural networks. In particular, first-layer weights in the neural network topology were found to be effective in identifying which components of the input signature contain the most useful information. Using first-layer weights, we can identify a subset of features that offers classification rates comparable to the full feature set. This result also suggests that a neural network employing the full feature set can effectively filter those features that lack content.

Although we can show the advantage that neural networks have over traditional classification tools, the classification error remains unacceptably high for the example data set. This difficulty cannot be resolved due to the lack of quality content in the particular signatures. Our analysis suggests that either more or different information is needed to more accurately monitor the quality of the solder joint.

The signature analysis methodology employed in this paper, along with the specific feature selection and classification tools, represents only one component of a larger signature-monitoring procedure, defined as follows:

Step 1: Establishment of quality metrics
Step 2: Selection of process measurement sensors
Step 3: Signature analysis (feature extraction, feature selection, and classification)
Step 4: Implementation of on-line monitoring/control

Steps 1 and 2 represent inputs to the signature analysis methodology described in this paper. We must first establish quality metrics so that we can definitively classify each instance of the process output. Without accurate classifications, we cannot build a detailed relationship between the process output and the corresponding signatures. Second, we must decide how to install sensors for process measurements (type, quantity, and location of sensors). As demonstrated in this paper, signatures that lack content cannot contribute to an accurate classification decision. In general, effective sensor selection is dependent on an understanding of the process. Given that such understanding is often limited, we would expect sensor selection to be modified after signature analysis is begun. The third step of this procedure involves application of the signature analysis tools presented in this paper—feature extraction, feature selection, and classification. If, given the application of these tools, we discover that the signatures lack content, we must reevaluate sensor selection and configuration. Finally, in Step 4, we apply the results of the preceding steps in the form of a signature-monitoring system. We can also make the quality classification data available for on-line control, operator intervention, or both.

Future Research Directions

The first research question to address in developing a process monitoring system is how to definitively quantify the quality of the process output. To

answer this question, we must develop the means to translate product specifications into quality metrics to identify the quality of the process output without ambiguity. In essence, measurements of quality metrics serve as calibration standards by which we tune our classification tools. If the calibration standards are not accurate, the classification tools will not be accurate. Alternatively, we might try to build a relationship between process signatures and the most significant process variables. This would entail a design-of-experiments approach where we vary the pertinent process variables (for example, dull tool to sharp tool) under a variety of sensor configurations so that we can guarantee a complete yet concise view of the process variables in the data set used for training. Under this scheme, multiple classes of process output would be defined according to the variables used in the experiment. In correlating process signatures to process variables, we are assuming a direct relationship between process variables and the quality of the process output.

Once the training data are available we must still formulate a relationship between process signatures and their corresponding classifications. As demonstrated in this paper, artificial neural networks offer many advantages over traditional statistical classification tools in this capacity. Further research in this area could result in more efficient network training algorithms, better feature extraction and selection methods, or more accurate decision functions. There is a tremendous volume of research conducted in the field of neural networks that can be applied to the manufacturing signature analysis problem.

Finally, consideration must be given to application of these new technologies in the manufacturing environment. This includes both development of software tools, which allow manufacturing engineers to exploit the technology, and implementation of neural network hardware that could facilitate real-time control of complex processes. The long-term vision of our research consists of a signature analysis toolbox that includes a variety of sensors, signature analysis tools, and the ability to export the "blueprint" of a signature-monitoring system. Ideally, a manufacturing engineer could approach a new process with this toolbox, collect a number of signatures, and then subject these signatures to a selection of analysis tools. The result of this analysis would be the blueprint for an on-line process monitoring system.

Acknowledgment

Funding for this research was provided by Digital Equipment Corp. and by the Massachusetts Institute of Technology Leaders for Manufacturing Program.

References

1. P. Trahanias and E. Skordalakis, "Syntactic Pattern Recognition of the ECG," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (v12, n7, July 1990), pp648-657.

2. S. Bleha, C. Slivinsky, and B. Hussien, "Computer-Access Security Systems Using Keystroke Dynamics," *IEEE Transactions on Pattern Analysis and Machine Intelligence* (v12, n12, Dec. 1990), pp1217-1222.

3. G. Chryssolouris, P. Sheng, and F. von Alvensleben, "Process Control of Laser Grooving Using Acoustic Sensing," *Journal of Engineering for Industry* (v113, n3, Aug. 1991).

4. J. Rotberg, E. Lenz, and S. Braun, "Vibration-Based Drill Wear Monitoring," *Manufacturing Review* (v3, n1, Mar. 1990), pp60-65.

5. R. Teti and D. Dornfeld, "Modeling and Experimental Analysis of Acoustic Emission from Metal Cutting," *Journal of Engineering for Industry* (v111, n3, Aug. 1989), pp229-237.

6. S.Y. Liang and D.A. Dornfeld, "Characterization of Sheet Metal Forming Using Acoustic Emission," *Journal of Engineering Materials and Technology* (v112, n1, 1990), pp44-51.

7. R.N. Boggs, "System Analyzes Processes to Eliminate Defective Parts," *Design News* (May 18, 1992), pp91-92.

8. Assurance Technologies Inc., 503D Highway 70 East, Garner, NC 27529.

9. J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles* (Reading, MA: Addison-Wesley Publishing Co., 1974).

10. R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine* (April 1987), pp4-22.

11. R. Schalkoff, *Pattern Recognition: Statistical, Structural and Neural Approaches* (New York: John Wiley & Sons, Inc., 1992).

12. K. Matsuoka, "Noise Injection into Inputs in Back-Propagation Learning," *IEEE Transactions on Systems, Man, and Cybernetics* (v22, n3, June 1992), pp436-440.

13. F. Rosenblatt, *Principles of NeuroDynamics* (Washington, DC: Spartan Books, 1951).

14. E.D. Karnin, "A Simple Procedure for Pruning Back-Propagation Trained Neural Networks," *IEEE Transactions on Neural Networks* (v1, n2, June 1990), pp239-242.

15. K.G. Mehrotra, C.K. Mohan, and S. Ranka, "Bounds on the Number of Samples Needed for Neural Learning," *IEEE Transactions on Neural Networks* (v2, n6, Nov. 1991), pp548-558.

16. J.E. Goldsberry and C. Burke, "Characterization of the Two Color Infrared Bond Inspection Method" (Digital Equipment Corp., Technical Resource Group, May 1990).

17. O. Rioul and M. Vetterli, "Wavelets and Signal Processing," *IEEE Signal Processing Magazine* (Oct. 1991), pp14-38.

18. F. Kanaya and S. Miyake, "Bayes Statistical Behavior and Valid Generalization of Pattern Classifying Neural Networks," *IEEE Transactions on Neural Networks* (v2, n4, July 1991), pp471-475.

Authors' Biographies

Steven D. Eppinger is an associate professor of management science at Massachusetts Institute of Technology. He received his SB, SM, and ScD degrees from MIT in mechanical engineering. His research is in the field of manufacturing, specializing in product development and manufacturing operations.

Christopher D. Huber received his BS degree in mechanical engineering from the University of Minnesota and his SM degree in mechanical engineering from Massachusetts Institute of Technology. He is currently manufacturing systems engineering manager at Saint Jude Medical Inc.

Van H. Pham received his SB and SM degrees from Massachusetts Institute of Technology in mechanical engineering. He is currently a doctoral student in economics at Cornell University.