Active Learning from Process Data

Gokaraju K. Raju and Charles L. Cooney
Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

Much of the prevailing connectionist machine learning research in chemical engineering assumes a one-way passive relationship between the learner and the application domain. This article investigates a two-way active relationship between learner and domain. An active relationship is useful and even necessary if the prevailing research is to be successfully applied to real-world problems involving sparse and strongly biased data. A process development case study is used to illustrate the impact of data quality and quantity and to compare the performance of active learning against conventional passive learning. This study highlights on the need to assess data quality and demonstrates the improvements in the rate of active learning.

Introduction

A number of computer-based approaches have been taken over the years to learn about unit operations and processes during both process development and commercial manufacturing. These can be classified into three groups: *first principle-based, expert knowledge-based* and *example-based* learning. *First principle-based* approaches require a detailed understanding of the underlying principles that govern the process and an ability to measure important process parameters. However, due to the complex, nonlinear, and time-varying nature of many chemical and biochemical applications, the development of such detailed first principle models is time-consuming and expensive. The time and money requirements make the use of a purely first principle-based approach intractable for many practical applications. Recent research has aimed at applying artificial intelligence techniques to such applications. *Expert knowledge-based* approaches involve capturing existing knowledge about a domain from an expert in the form of knowledge-based expert systems. Examples of this approach include the use of expert systems for fault diagnosis (Basila et al., 1990) and bioprocess control (O'Connor, 1989). Fuzzy logic-based systems follow a similar approach by defining membership functions to convert continuous process variables into symbolic variables that can be used within an expert-system type of framework (Konstantinov and Yoshida, 1992). Knowledge-based expert systems and fuzzy systems provide a framework to capture *already learned* knowledge in a consistent and understandable manner. This approach has two major limitations—the static nature of the knowledge and a knowledge-engineering bottleneck associated with eliciting and organizing knowledge from the expert. The *example-based* approach attempts to address these limitations by focusing on developing systems that can automatically acquire knowledge by a process of *induction* (Holland et al., 1989) or *machine learning* (Carbonell, 1990). The different models of learning that have been used include Bayesian networks, genetic algorithms (Rojas-Guzman, 1995), decision trees (Saraiva, 1992), and connectionist learning systems (Watanabe et al., 1994). One can identify four major approaches to machine learning: inductive, analytic, genetic, and connectionist (Carbonell, 1990). Of these, the connectionist model of learning, also called "neural network" or "parallel distributed system," has become particularly popular. Neural networks have been described to have advantages in applications that involve high dimensionality (Bishop, 1995).

Neural networks with different architectures (such as feedforward, recurrent, and multiple network architectures) and different basis functions (such as sigmoidal, radial, wavelet, and ellipsoidal basis functions) have been explored. These are summarized in Table 1a. Such neural networks have been used for several types of applications including fault diagnosis, dimensionality reduction, modeling, data rectification, dynamic optimization, experimental design, and estimation. These are summarized in Table 1b. The attractiveness of these models has been driven primarily by their ability to learn from examples and to model arbitrarily complex relationships. Figure 1 depicts the machine learning paradigm associated with this phase of research using connectionist networks. Data $(x_1, x_2, x_3, \ldots)$ are collected from the application and a set of labels $(y_1, y_2, y_3, \ldots)$ assigned to each of the data points.

Correspondence concerning this article should be addressed to C. L. Cooney.

Table 1a. Neural Network Architectures and Basis Functions

Network Attributes	Types	References
Architectures	Feedforward	Hoskins et al. (1991); Richard and Lippman (1991).
	Recurrent	Karjala and Himmelblau (1994).
	Multiple Network	Watanabe et al. (1994); Sridhar et al. (1996).
Basis Functions	Sigmoidal	Venkatsubramanian et al. (1990); Ungar et al. (1990).
	Radial	Yao and Zafiriou (1990); Leonard and Kramer (1991).
	Wavelet	Bhakshi and Stephanopoulos (1993).
	Ellipsoidal	Girosi (1992); Girosi et al. (1995).

Table 1b. Types of Recent Neural Network Applications in Chemical Engineering

Applications	References
Fault Diagnosis	Venkatsubramanian et al. (1990); Leonard (1991); Hoskins et al. (1991).
Dimensionality Reduction	Kramer (1992); Tan and Mavrovouniotes (1995).
Modeling	Thompson and Kramer (1994); Tholudur and Ramirez (1996).
Data Rectification	Karjala and Himmeblau (1994).
Dynamic Optimization	Chen and Weigand (1994).
Experimental Design	Glassey et al. (1994); Chen et al. (1998).
Estimation	Massimo et al. (1992); Jin et al. (1996).

These labels are obtained either directly from the application or from a teacher/expert familiar with the application. Each data point (x) and label (y) can either be a scalar or a vector. Each data-label pair ($x_1y_1, x_2x_2, x_3y_3, \ldots$) is an example. These examples are used to train a learner based on some performance criterion such as the mean-squared error or cross entropy. The learner is assumed to be a passive receptacle of training examples.

A major limitation of this approach is the assumption of a one-way passive interaction between the application and the learner. This assumption makes the quality and quantity of data available for learning completely dependent on the data generation characteristics of the application and the data collection characteristics of the teacher/expert. Much of the prevailing research avoids addressing this issue by either working on small applications and/or by making idealized assumptions about data quality and quantity. The inherent assumption is that data are abundantly available and are of appropriate quality. However, real world data are usually both sparse and biased. It is rare to have enough data. Moreover, the data that are available are often of low quality. It does not seem justifiable to assume that the data generation characteristics of the application and the data collection characteristics of the teacher/expert would be optimal for learning. As a result, we argue that only a few of the results will extend directly to practical applications because of the idealized assumptions about data quality and quantity. The problem can be severe enough to make many current applications intractable even if they are small. Using real world data may necessitate a paradigm shift away from conventional *passive learning*, where the neural network accepts and learns from all the examples provided, and towards *active learning* where the neural network determines which examples to learn from and when.

HBF Neural Network Learner

We choose the connectionist approach to learning because of its suitability in applications which are unstructured, continuous, and involve large amounts of data (Carbonell, 1990). Of the many connectionist approaches, the multilayer perceptron network (MLP) with sigmoidal nonlinearities has been the most popular architecture used in chemical engineering research. MLPs with sigmoidal transfer functions have been applied to a number of pattern recognition problems (Tables 1a and 1b) to demonstrate the ability to learn from examples; they have been found to often generalize successfully. Several authors have studied the use of radial basis functions (RBF) as substitutes for the sigmoidal basis function. Based on a training procedure developed by Moody and Darken (1989), RBF networks have been used for many chemical process applications (Tables 1a and 1b). While it has been shown that both MLP networks with sigmoids and RBF networks can fit any arbitrary function (Cybenko, 1989; Park and Sandberg, 1991) and that two kinds of networks are equivalent in many ways (Poggio and Girosi, 1990), RBF networks were viewed to be more attractive because it was possible to make more explicit computations of measures of extrapolation and local goodness of fit (Leonard, 1991). While RBFs have recently found applications in artificial intelligence, and, in particular, the problem of learning from examples, Girosi (1992) points out that the RBF method cannot be applied in a straightforward manner to many practical cases because it does not take into account some features which are typical of the problem of learning from examples. These drawbacks are overcome by computing the coefficients of the

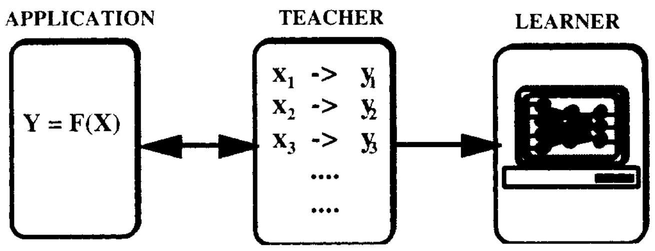


Figure 1. Conventional passive learning mode.

RBF expansion using least squares instead of a system of linear equations and by defining a weighted norm that allows for nonradial (ellipsoidal) basis functions (Girosi, 1992). A further extension that allows for centers to also be movable leads to the definition of the hyper basis function (HBF) network model of the learner (Girosi et al., 1995). Like the sigmoidal MLP and RBF networks, HBF networks are attractive because of their ability to fit arbitrarily complex relationships and make probabilistic interpretations of the observed results. In addition, they are a more generalized version of the RBF network that is embedded in the framework of approximation theory. A common goal in training a HBF network is to minimize the sum of squared error between the network outputs and the desired outputs over the entire set of examples. That is, attempt to minimize H where

$$H = \sum_{i=1}^{\text{nexamples}} [F(\underline{X}_i) - \underline{Y}_i]^2 \tag{1}$$

where F represents the function encoded by the HBF network, $\underline{X}_i$ is the input vector corresponding to example i, and $\underline{Y}_i$ represents the desired vector output or label assigned to example i. The squared-error cost function has been used more frequently than any alternative and yields good performance with large databases on real-world problems involving prediction, input-output mapping, or classification (Richard and Lippmann, 1991). In addition, the relationship between minimizing a squared error function and estimating Bayesian probabilitie has been well established (Duda and Hart, 1973). The function F is implemented by the HBF network as

$$F(\underline{X}) = \sum_{k=1}^{\text{ncenters}} c_k e^{\frac{-(\| \underline{X} - \underline{T}_k \| \underline{W}_k)}{\sigma_k^2}} \tag{2}$$

In Eq. 2, the HBF network has four types of adjustable parameters: cluster centers (denoted by T_k), cluster sizes (denoted by σ_k), cluster shapes (determined by the weights which are denoted by W_k) and coefficients (denoted by C_k). The degree to which a HBF network has learned a relationship between the inputs and outputs is computed in terms of a mean-squared error (MSE), which is the sum of squared error H normalized by the number of examples. MSE represents the difference between the network's prediction and the desired outputs.

The initial guesses for the cluster centers are chosen to be the centroids determined by task-dependent clustering (Johnson and Wichern, 1988). The HBF network is trained using a gradient descent method that determines the appropriate cluster centers, sizes, shapes, and coefficients. Training involves three different data sets: *training* data set, *validation* data set and *testing* data set. Multiple networks are trained by minimization of the sum-of-squares error function defined with respect to the *training* data set. The performance of the networks is then compared by evaluating the error function using an independent *validation* set and the network with the lowest error with respect to the validation set is selected. The performance of the selected network is confirmed by measuring its performance on a third independent set of data called a *test* set. It is the performance on this test set that is described in terms of the generalization error.

Active Learning

Active learning is a form of learning that assumes that the learner has some control over what part of the input domain it receives information about. The quality of the data used for training has an impact on the rate of learning and generalization error (Cortes et al., 1994). The appropriateness of data depends on the function that is being learned. For example, more data are required in regions of rapid change and fewer in regions with little change in the value of the function as a function of the inputs. With this realization, there is an opportunity for a fundamental shift in learning mode. Instead of simply accepting all the examples that are presented to the network by the teacher or are available from a database, the network should only use examples that have new information. Rather than having the external environment drive the learning process, the learner can drive the learning process based on its needs. We can create a network that actively determines which examples have the needed information and use only the examples it needs. A number of recent articles describe the need for this transition from passive "learning from examples" to active learning in the case of the PAC (Kulkarni et al., 1993), Bayesian (Mackay, 1992a), neural network (Cohn et al., 1994), and statistical (Cohn et al., 1995) models of learning. These articles use theoretical bases to motivate the need for such a transition and use computer simulated data of simple mathematical functions such as the sine wave to compare the learning curves associated with passive learning with random sampling against active learning with selective sampling. In this work, we use an HBF network and focus on a practical chemical engineering application where the rate of learning can have significant economic implications. Here, the passive learning case does not involve truly random sampling but rather involves sampling that is biased by the application expert's own mental model (Senge, 1992) about the domain.

Information is relative to what is already known; in order to implement this approach, one needs a way to measure data quality relative to what the network already knows. Cohn et al. (1994) define a region of uncertainty to be an area in the domain where misclassification is possible and draw examples only from this region of uncertainty. However, as explained by them, this concept of a region of uncertainty only works for relatively simple concept classes. In this work, we define confidence limits associated with each of the network's predictions. We use the 95% confidence limit as defined by Leonard (1991) to evaluate the reliability of the network's predictions, because confidence limits capture information about both the accuracy of the model at a particular point and also about how much training data were available at that location. A local confidence interval is developed for each HBF unit and then the confidence limits for the model prediction at a given test point are formed by taking a weighted average of the confidence limits over all contributing units. The confidence limit for a test point is defined to be an average of the HBF unit confidence limits, weighted by the contribution of each hidden unit to the output for the current output as

$$CL_i(\underline{X}) = \frac{\sum_{h=1}^{H} a_h(\underline{X}) * CL_{hi}}{\sum_{h=1}^{H} a_h(\underline{X})} \tag{3}$$

where $CL_i(\underline{X})$ is the confidence limit for the output i at test point $\underline{X}$. $a_h(\underline{X})$ is a vector of the activations of hidden node h over all the data and CL_{hi}, the 95% confidence limit for the expected value of the residual associated with output i for unit h, is given by

$$CL_{hi} = \frac{t_{95} * s_{hi}}{\sqrt{n_h}} \tag{4}$$

where s_{hi} is the standard deviation of the model residuals for output i at hidden node h, n_h is the effective number of training points associated with unit h and t_{95} is the critical value of the Student t statistic for 95% confidence with $n_h - 1$ degrees of freedom. The local variance of the model residuals for output i at hidden node h, s_{hi}^2 is given by

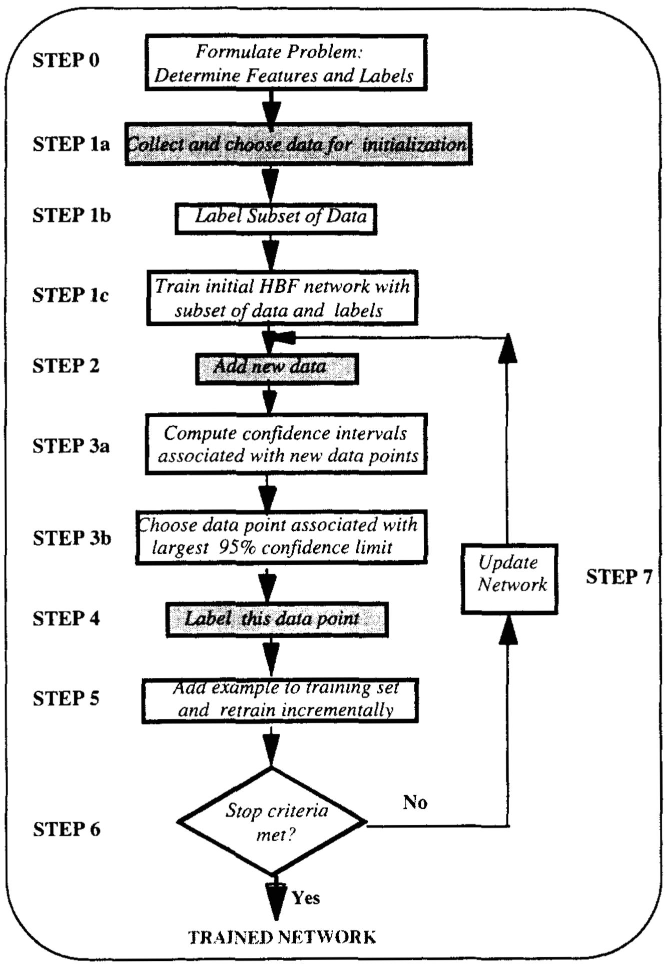
$$s_{hi}^2 = \frac{\sum_{k=1}^{K} a_h(\underline{X}_k) * E_{ik}^2}{n_h - 1} \tag{5}$$

where k is the total number of examples, $a_h(\underline{X}_k)$ is the activation of hidden unit h given training point $\underline{X}_k$, and E_{ik} is the cross validation error on output i for training example k. Each training point is allowed to contribute to the error estimate in proportion to its membership in the unit. n_h, the effective number of training points associated with unit h, is given by the sum of the activations of unit h over the training points

$$n_h = \sum_{k=1}^{K} a_h(\underline{X}_k) \tag{6}$$

This method assumes that the residuals of the model are independent, normally distributed with a mean of zero and constant variance over the neighborhood defined by each HBF unit by varying from unit to unit. This implies that the model form matches exactly the form of the true function. Since the true functional form is typically unknown and the model is empirical, this assumption of normally distributed errors may not always be accurate. For every example that is presented to a network, a confidence limit can be computed based on what the network has already learned. These confidence limits can become a basis on which the network can determine the next example to learn from allowing the network to focus on examples that it is least confident about.

We can now formulate a methodology for active learning (Figure 2). Active learning begins with problem formulation. The learning task is formulated by determining the appropriate features and labels. The choice of appropriate features can often be a nontrivial task in many real-world applications. Too many unnecessary features can make the learning computationally expensive and difficult, while leaving out relevant features can make the task "unlearnable." Once the appropriate features and labels are determined, the first step is to initialize the learner which in this case is a network. Initialization involves collecting a subset of data, extracting the appropriate features from the data, assigning labels to the data, and training an initial HBF network on this subset of examples. This is shown as Steps 1a to 1c. The labels may be obtained either directly from the application or from an expert/teacher. The values of the features are typically scaled between 0 and 1 (value of x). The label is the corresponding value of y. Once an initialized HBF network is available, new data are added (Step 2) and confidence intervals are computed for the new data using Eq. 3 (Step 3a). The data point associated with the highest 95% confidence limit (most uncertainty) is chosen (Step 3b) and labeled (Step 4). This example is added to the training database and the network is trained incrementally (Step 5). In this way, the network is trained as examples are added one by one into the training set. This process continues until the stop criteria are met (Step 6). Multiple stop criteria can be used. Some of them include a pre-specified generalization error level based on the requirements of the application or the reaching of a minimum generalization level. The level of accuracy to which the desired relationship needs to be learned depends on the quan-



STEP 0
Formulate Problem: Determine Features and Labels
STEP 1a
Collect and choose data for initialization
STEP 1b
Label Subset of Data
STEP 1c
Train initial HBF network with subset of data and labels
STEP 2
Add new data
STEP 3a
Compute confidence intervals associated with new data points
STEP 3b
Choose data point associated with largest 95% confidence limit
STEP 4
Label this data point
STEP 5
Add example to training set and retrain incrementally
STEP 6
Stop criteria met?
No
Yes
Update Network
STEP 7
TRAINED NETWORK

Figure 2. Methodology for active learning.

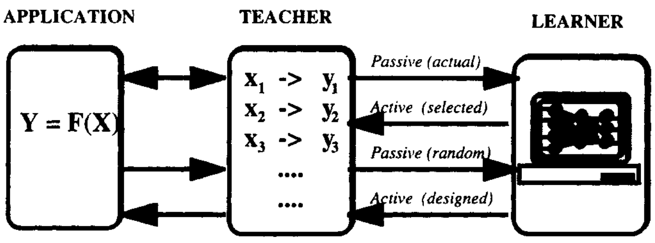


Figure 3. Modes of information exchange: passive and active.

tity and quality of examples available to learn from. Using this methodology, we are now in a position to explore the impact of both data quantity and data quality on learning.

Active Learning: Selected and Designed

Figure 3 contrasts different models of information exchange between the learner and the application domain. Four modes of information exchange are shown: two passive modes and two active. Passive learning (actual) refers to the mode in which data were actually collected from the application by the expert and collected in a database. These data are then used to train the learner. This was already described in Figure 1. Such a data collection approach is dependent on the data gathering approach of the expert and the data available in the database is representative of what might actually be available in many industrial environments. Passive learning (random) refers to a mode in which data is collected randomly from the application. No expert biases are involved in the data gathering process. However, the learner does not play any role in determining which data to collect.

Selected active learning and designed active learning refer to two different ways in which active learning can be performed. The shaded boxes of Step 1a, Step 2, and Step 4 of Figure 2 highlight steps in the active learning methodology that are performed differently for selected active learning and designed active learning. In the case of selected active learning the data have already been generated by the application. The learner has played no role in the data gathering process. The data have already been gathered and active learning involves choosing the next example to learn from within an existing database. That is, in Figure 2, Step 1a involves choosing a subset of data from the database containing all the data that have already been collected. Step 2 involves searching the database for data that have not already been used by the learner and Step 4 involves adding a label to the chosen data point. The interaction of the learner is primarily with the database/teacher and not with the application. The process of data gathering from the application depends only on the interaction between the application and the teacher/expert and not on the machine learner. However, in many applications, it is the actual data collection itself that is most expensive. Rather than collecting data and then selecting examples that have the highest quality from an existing database, there is significant benefit from collecting only high quality data in the first place. This way, only the necessary data are generated and collected in the first place. This extension of the selected active learning paradigm is designed active learning.

In this case of designed active learning the initial examples are drawn from a uniform initialization grid (Step 1a of Figure 2). A uniform initialization grid is the best alternative when there is no *a priori* knowledge about the underlying function that is being learned. If three equally spaced sample points are taken for each dimension of the data vector, this would be equivalent to using a three-level factorial design (Hogg and Ledolter, 1992) for initialization. In Step 2, designed active learning requires the network to collect new data directly from the application. It is desirable to explore the entire experimental space and choose the next data point from the region of greatest uncertainty. This is equivalent to searching an infinitely large database of all possible examples. Instead of actually performing an experiment on the actual process, we can use a uniformly spaced experimental grid of the possible database. This larger database is used in a manner similar to the case of selected active learning. In effect, the HBF network is searching the experimental space of possible examples and then choosing a data point from regions where the confidence limits are the largest. Only after this data point is chosen is an experiment performed and the data point labeled (Step 4). This way, only the necessary experiments are performed. The machine learner now has an active role in determining the data gathering process as well.

Case Study: Process Development of an *S. cerevisiae* Fermentation

In order to study "learning from examples" and active learning in a practical process application at the unit operation level we need to choose a representative unit operation. We seek to study a unit operation that is associated with significant variability in performance. Pharmaceutical and biochemical processes involve significant variability in performance. To a large extent, this is because these processes are poorly understood. It can be argued that an important reason for this poor understanding is an inadequate level of learning at the process level. While this variability is observed in many different unit operations along the process flow (Raju, 1994), we will focus on the fermentation unit operation in particular. The motivation for choosing this unit operation is summarized in Table 2a and described in more detail by Raju (1998). Fermentation processes involve a complex interaction between microorganisms and their environment in a process vessel and are used to make billions of dollars worth of commercial products. These processes are highly complex and nonlinear, and there is a serious limitation in our ability to both measure performance and to use conventional models to relate observed parameters to performance. As a result, there has been significant recent research aimed at applying artificial intelligence techniques such as knowledge-based expert systems (Konstantinov and Yoshida, 1992) and neural networks to fermentations (Jin et al., 1996). Of the many possible model systems, we choose the *S. cerevisiae* model system because of its extensive use in industry and its suitability for recombinant DNA technology (Romanos et al., 1992). This basis for this choice of model system is summarized in Table 2b and described in more detail by Raju (1998).

Bioprocess development involves taking a process from initial conception to final commercial production. During this time, a large number of experiments are performed and data

Table 2a. Characteristics of the Fermentation Unit Operation

Features	Dimensions	Description
Use of Unit Operation	Present Use Future use	Used to make billions of dollars worth of products. Future facilitated by recombinant DNA technology.
Availability of Data	Quantity of data Diversity of data	Large amounts of data have been collected. Laboratory, pilot plant and manufacturing scale.
Complexity	Nonlinear, time-varying High dimensionality	Usual linear time invariant models difficult to apply. Representative of many real-world problems.
Impact on Performance	Product cost	Often is one of the most expensive unit operations. Determines downstream cost to a large extent.
	Process cycle time	Fermentations take days to weeks to complete. Often, this is one of the longest steps in mfg. process.
	Product quality	Product quality determined in fermentation step. Many impurities are created in fermentation step.
Potential to Improve	Variance in performance Value of Learning	Large variability in yields, productivities and titers. Large impact of rate of learning on performance.
Lack of Measurement	Yield, productivity, titer	Unable to measure performance directly during run.

Table 2b. Characteristics of *S. cerevisiae* Model System

Features	Dimensions	Description
Use of Model System	Present Use Future use	One of the most widely used model systems. Very suitable for recombinant DNA technology.
Metabolism	Carbon Source Products Oxygen	Can use multiple carbon sources. Can produce multiple products—cell mass, ethanol, and so on Both aerobic and anearobic growth possible.
Measurement	Mass Spectrometer	Gas data facilitates indirect monitoring.
Prior Knowledge	Models, Expert System	Facilitates evaluation of different learning approaches.
Potential to Improve	Variance in performance Value of Learning	Large variability in yields, productivities and titers. Large impact of rate of learning on performance.

collected (Stephanopoulos et al., 1997) with a view to better understanding the process. This process understanding is important, because it facilitates the optimization of process performance (often formulated in terms of yield, productivity, and product titer). A typical process development scenario for an *S. cerevisiae* fermentation is depicted in Figure 4. The operational variables such as air-flow rate, stirring rate, glucose feed rate, and ammonia feed rates are manipulated during the course of the fermentation; the effect is observed in terms of variables such as the dissolved oxygen level, O_2 uptake rate, CO_2 evolution rate, pH, temperature, and cell density. These observations taken together reflect the aggregate metabolic state of individual microorganisms within the bioreactor. These metabolic states over time determine the final yield, productivity, and titer levels associated with each run. During the course of bioprocess development, a large amount of data are generated relating to the operational and observed variables over time together with the overall yield,

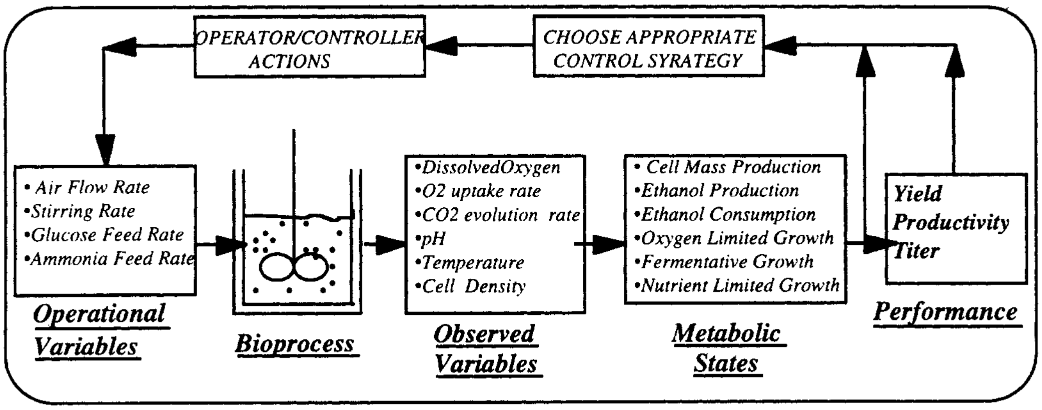


Figure 4. Typical bioprocess development scenario.

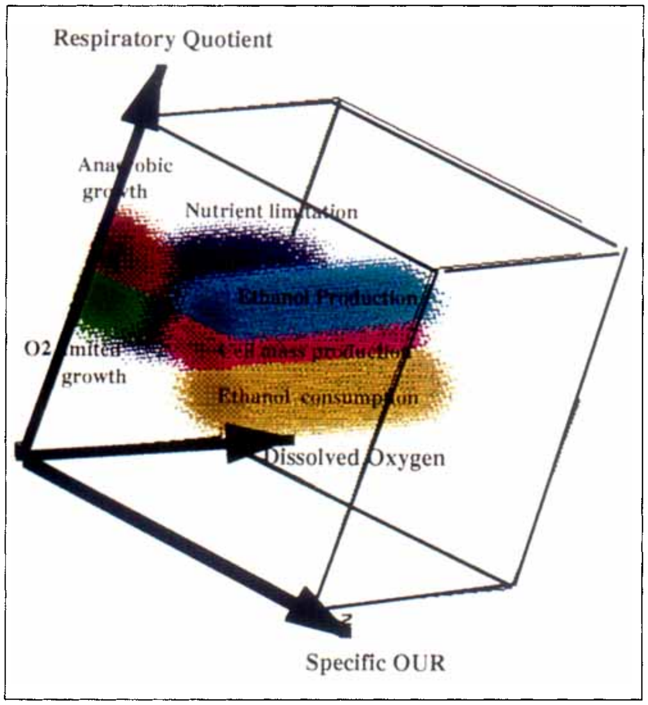


Figure 5. Simplified pattern recognition problem.

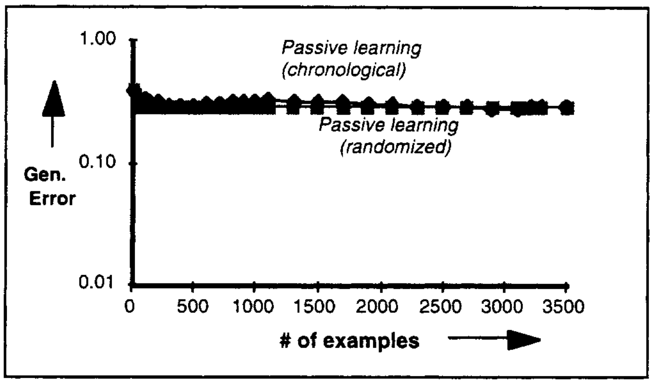


Figure 6. Generalization error vs. data quantity.

productivity, and titer levels. Since the final yields, productivities, and titers depend primarily on the progression of metabolic states over time, an important pattern recognition task in bioprocess development is to relate the observed variables to the metabolic states (Raju and Cooney, 1992).

In order to apply active learning to a process application, we consider a bioprocess development scenario involving the growth of *S. cerevisiae* on glucose. The available data are gathered from 12 different experiments (O'Connor, 1989). Each run involved growing *S. cerevisiae* in a fed-batch reactor; on-line data were collected every two minutes and the number of data points from each run varied between 100 and 500 time points. In order to simplify the case study, we only consider three observed variables—a measurement of the dissolved oxygen level using a dissolved oxygen probe and calculation of the values of the specific oxygen uptake rate (sOUR) and the respiratory quotient (RQ) (defined by the ratio of the carbon dioxide evolution rate to the oxygen uptake rate) based on mass spectrometer readings of exhaust oxygen and carbon dioxide. Such a choice of variables can be made either using *a priori* domain knowledge or directly from the data by using statistical techniques to identify the variables that capture the desired information. We choose these variables because of the relatively orthogonal nature of the information each of them provide about metabolic state. These choices were also confirmed by using a separate neural-network based analysis-of-variance analysis (Raju, 1998). The dissolved oxygen (DO) level describes the relative balance between oxygen supply and oxygen consumption. The specific oxygen uptake rate (sOUR) reflects the quantity of oxygen being consumed by each cell, while the respiratory quotient (RQ) reflects the quality of the cell respiration (that is, the balance between aerobic and anaerobic metabolism). At each time point, we record values of DO, sOUR, and RQ. We assume that the process development of an *S. cerevisiae* fermentation involves learning a mapping between three measured or measurement derived variables: respiratory quotient (RQ), dissolved oxygen level (DO), and specific oxygen uptake rate (sOUR) and six metabolic states (labels) corresponding to cell mass production, ethanol production, ethanol consumption, nutrient limited growth, anaerobic growth, and oxygen limited growth (O'Connor et al., 1992). The basic premise is that if a bioprocess can be operated to maintain the microorganisms in preferred metabolic states during a fermentation, that would optimize overall performance. Process development involves learning a mapping between the features and the metabolic states. Here, we assume that there is a continuous one-to-one mapping between the three features and the six metabolic states. This mapping is depicted in Figure 5. This can be considered to be the mental map or mental model of the domain expert (Senge, 1992).

Conventional knowledge-based expert systems, fuzzy systems, and neural networks use different means to represent such a mental model. A conventional knowledge-based expert system (O'Connor, 1989) would represent this mental model in terms of hyperrectangles by defining IF-THEN rules. This form of pattern recognition, however, has two types of limitations. First, it requires the operator to formulate an explicit reasoning process associated with the pattern recognition in the form of a rule. Second, it involves an arbitrary discretization in both the antecedent and consequent (Negoita, 1985) portions of the rule. Fuzzy systems attempt to overcome this second limitation by defining linguistic variables such as "low," "medium" and "high" to enable a smooth mapping between the continuous process variables and the symbolic language of the operator. This allows the mental map to be described as a set of continuous fuzzy relations between the three features and the metabolic states. The qualitative characteristics of the antecedent portions of such a mapping are shown in Table 3. However, the development of such a fuzzy system requires an explicit definition of membership functions for the antecedents and consequents of each fuzzy rule and an explicit formulation of fuzzy expert rules necessary to build the system. Here, we avoid the subjective definition of membership functions (Magdalena and Monasterio-Huelin, 1997) and the explicit formulation of fuzzy rules by using the expert only to "recognize" the metabolic state in

Table 3. Metabolic States and their Qualitative Characteristics

DO Level	RQ	sOUR	Metabolic State
Low to very high	Medium	Medium	Cell mass production
Low to very high	High to very high	Medium	Ethanol production
Low to very high	Low	Medium	Ethanol consumption
Very low	Medium	Very low to low	O_2 limited growth
Very low	High to very high	Very low	Anaerobic growth
Medium to very high	Medium	Very low	Nutrient limitation

terms of a label with some level of precision and accuracy. The HBF network can be used to learn the detailed membership functions and rules from the data. We assume that for each vector of DO, sOUR, and RQ values, an expert is able to assign a label corresponding to the metabolic state that this vector reflects with some degree of accuracy and precision. Here, we assume that the domain expert is able to label each of these metabolic states in terms of five levels (that is, in terms of linguistic variables such as "very low," "low," "medium," "high," and "very high"). Random inaccuracy is modeled by adding 2% random Gaussian noise to the labels. A detailed description of the labeling process and an analysis of the impact of labeling accuracy and precision on learning is described by Raju (1998).

Learning and Generalization

A HBF network is used to learn the mapping between the data (features) and the labels (metabolic states) using the method of cross-validation. Initial guesses for the cluster centers are determined by task-dependent clustering. Learning involves the adjustment of the HBF network cluster centers, sizes and shapes, and the coefficients in order to minimize the mean-squared error as defined in Eq. 1. Training the HBF network involves using different initial guesses for the network parameters and choosing the final network configuration that results in the lowest cross-validation error. The gradient descent method is used to determine the cluster centers, sizes, shapes and coefficients. Given the definition of these six metabolic states, 8,000 examples of each metabolic state are generated using a $20 \times 20 \times 20$ grid of each metabolic state. This provides us with a database of approximately 48,000 examples. This is our oracle of examples or benchmark grid. A benchmark grid of these 48,000 examples is used to compute the generalization error. This generalization error is used to measure of the extent of learning.

Results

Passive learning

The HBF network is initially trained using only the first 25 data points collected during the bioprocess development program. This network is trained incrementally by adding batches of data one at a time to the training database. Here, we add batches of 50–100 experimental data points at a time over the course of the process development program. Each generalization error value corresponds to a measure of how well the process is understood. If we assume that for a process to be optimized it must be first understood, it can be argued that the generalization error is a measure of nonperformance. Traditionally, learning curves describe the relationship between cost per unit and cumulative number of units produced (Hayes and Wheelwright, 1984). They also have been used to follow the relationship between the number of defects and the cumulative number of units produced (Strata, 1989). Here, the learning curve depicts the relationship between the generalization error (which is analogous to cost) and the cumulative amount of data generated during the process development program.

Figure 6 shows two different learning curves associated with passive learning using the data generated during bioprocess development. In the first case, examples are chosen chronologically from the database based on when the data were generated in the real process. The generalization error at any point indicates the extent of learning by the machine learner at any point during the course of the actual bioprocess development. At any point of time, the learner only knows about the data that have already been collected. The observed learning curve is different from the continuously decreasing generalization error that would have been expected by analogy with learning curve literature (Hayes and Wheelright,

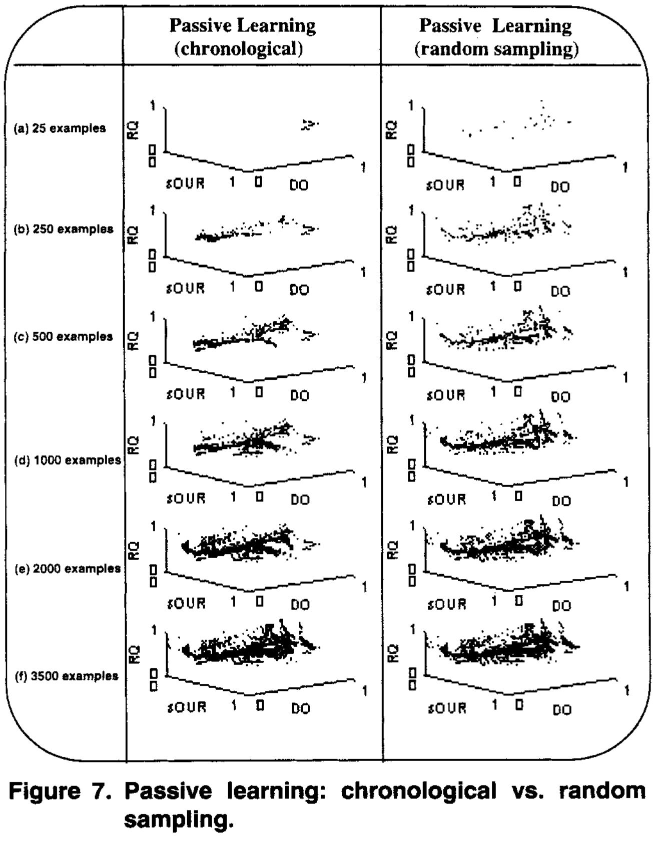


Figure 7. Passive learning: chronological vs. random sampling.

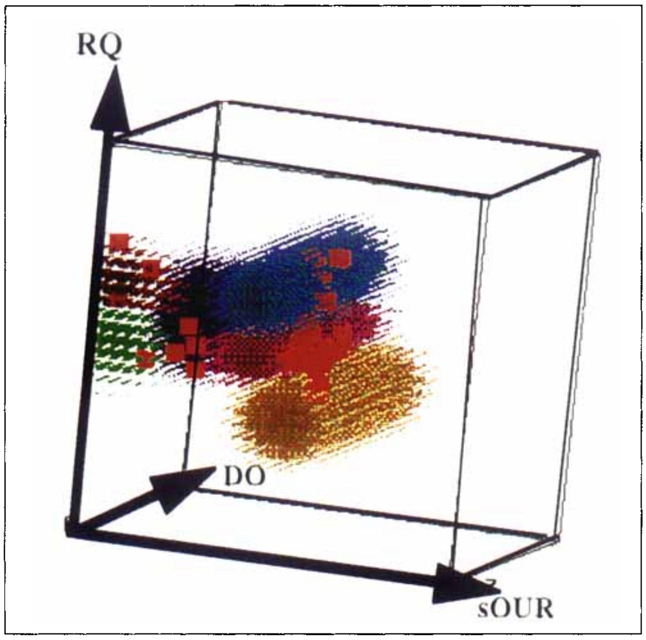


Figure 8. Bioprocess pattern recognition problem: data quantity and quality issues.

1984) in that very little learning takes place. Most of the learning seems to have taken place using the first few examples after which there is little or no incremental learning. That is, with passive learning, where data are learned from as they are generated, additional examples do not provide significant additional learning after the first few examples. The large number of additional examples in the database do not help to learn the relationship further. In the second case, examples are drawn randomly from an existing database after all the data have been collected. Each randomly selected batch of data reflects the overall characteristics of the entire database. This results in a continuously decreasing generalization error. Again, very little learning takes place beyond the first few examples.

Figure 7 compares the data gathering profiles associated with passive learning when done chronologically and when data are chosen randomly from the existing database. As seen, in the case of random sampling, the learner gets a picture of the overall database with a few randomly chosen examples and most of the learning is completed with a few examples. Additional examples provide the same information (Figures 7b–7f) and so there is little or no incremental learning. In the case of chronological passive learning different areas of the feature space are explored at different times during process development. This relates to both the data generation characteristics of the application and the data collection approaches of the expert/teacher. As an illustration, data generated from one experimental run superimposed on the decision surfaces that the network is attempting to learn is depicted in Figure 8. From this, it can be argued that the data gathered during the experimental campaign may not provide a means to adequately sample the feature space. Rather, data density is very high in certain areas of the feature space and is sparse and even virtually nonexistent in other areas. While this is partly because portions of the feature space are difficult to reach, this also indicates that experimental runs conducted during a bioprocess development program are usually performed with *a priori* expectation about which portions of the feature space are likely to result in desirable process performance in terms of yields, productivities, and titers. While such *a priori* knowledge may provide a means to quickly reach a satisfactory performance level, large portions of the experimental space are consistently left unexplored. This is depicted in the passive learning curve where there is little incremental learning (that is, the generalization error does not decrease) after the first few runs.

Active learning: selected and designed

Rather than train the network sequentially, as in conventional passive learning with all the data generated during the experimental runs, we can actively select the examples to learn from based on the network's current confidence levels associated with each of the examples in the database. Even with selected active learning, the learning curve is still quite flat. Instead of using active learning to choose a set of examples from an existing database, let us assume that we are not restricted to using only existing data and that the learner can ask for new data. The learner can design experiments based on its confidence limits in feature space. With this protocol, an experiment is performed to collect data in the region of least confidence and the data are added to the training set and so on. The designed active learning network is initialized with an initialization grid of 3-D equally spaced examples with a spacing of 0.5 in the unit cube feature space. This provides a $3\times3\times3$ grid of 27 points to initialize the networks. The experimental design aspect is implemented by using a 0.05 spacing benchmark grid comprising equally spaced examples. Using this grid as the basis to draw examples from is the equivalent of actually searching the entire feature space. Figure 9 compares the learning curves with passive learning (actual), selected active learning, passive learning (random), and designed active learning. The difference due to designed active learning is quite large. Designed active learning required fewer examples and resulted in significantly lower generalization error. Figure 10 compares the data gathering profiles of selected active learning and designed active learning and provides a basis to understand the observed learning curves. As

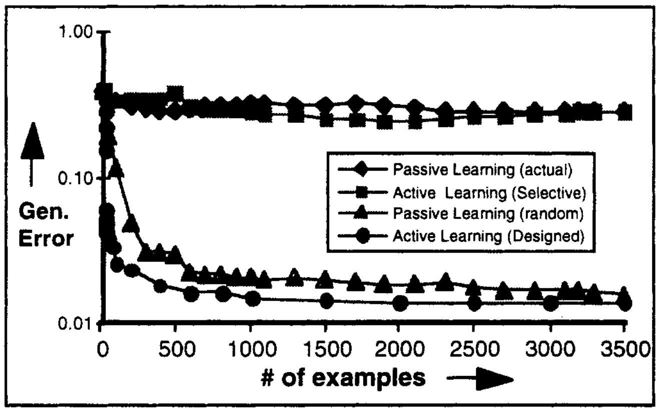


Figure 9. Passive and active learning: learning curve consequences.

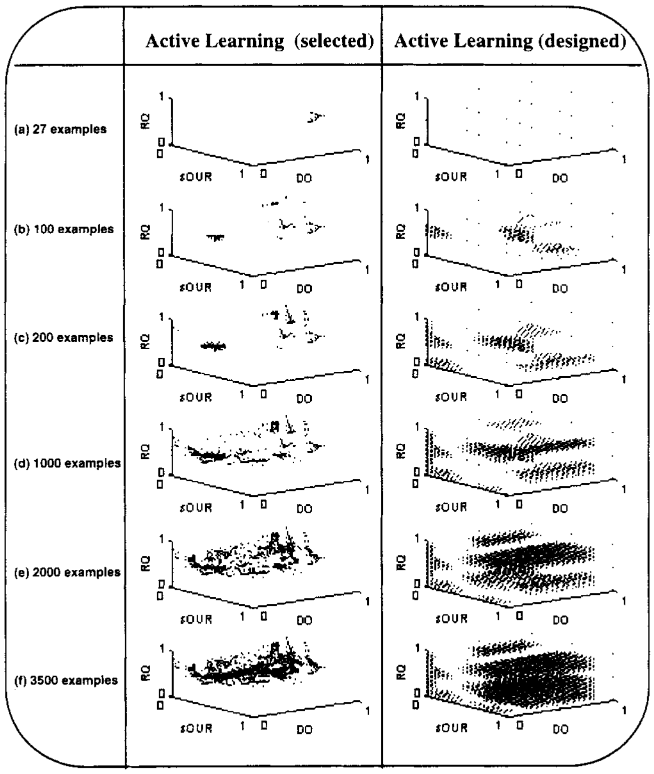


Figure 10. Active selected vs. active designed learning.

shown, active designed learning is initialized with a $3 \times 3 \times 3$ grid of 27 examples that sample the entire feature space. This is in contrast to the selected active learning case that is initialized with the first 25 examples from an existing database (Figure 10a). This results in initializing the selected active learning process with multiple examples from a small portion of the overall feature space. Following Figures 10b, 10c, and 10d indicates that the data for designed active learning covers the feature space more broadly than selected active learning. Beyond 1,000 examples, there is little or no incremental learning in the case of designed active learning. This can be explained from the figure by the observation that no significant additional areas of the feature space are explored beyond 1,000 examples (Figures 10e and 10f). The dramatic improvement in the average slope of the learning curve in the case of designed active learning indicates that careful choice of experiments (or data gathering) was more important than careful choice of which examples to learn from within an existing database.

A comparison of the actual passive learning curve against the random passive learning curve shown in Figure 9 demonstrates the impact of the expert biases on learning. Significantly enhanced learning could have been obtained by simply exploring the entire feature space randomly rather than using the expert's *a priori* knowledge to constrain the data gathering to certain portions of the feature space.

Discussion

Learning and process development

Process development involves learning the relationship between the operational variables that can be manipulated and the overall process performance (Figure 4). For a fermentation process, this requires the development of an understanding of the relationship between process variables, such as dissolved oxygen (DO), respiratory quotient (RQ), and specific oxygen uptake rate (sOUR), and different metabolic states. This involves making choices about data generation and analysis. The case study discussed previously uses a machine learning approach to understand the consequences of choices made in data generation and analysis. A study of twelve fermentation experiments is used to compare the passive learning, selected active learning, and designed active learning curves. The mode of learning is described as passive or active with respect to the machine learner. There is also the human learner who has performed the experiments and collected the data during the course of developing the process. There is a learning curve associated with the human learner as well. The machine learner should not be expected to eliminate the need for a human learner. Rather, it should be viewed as a potentially important decision support system. The learning curve of the human learner can be viewed as being similar to the passive learning (actual) case of the machine learner. The passive learning mode indicates that data collected resulted in very little learning beyond the first few experiments. The number of experiments performed is relevant, because it can have significant impact on the time and cost for a process to be commercialized. Also, an incompletely learned relationship between process variables and metabolic states could have a detrimental impact on process performance in terms of yields, productivities, and titers. The impact of such learning on time-on-market and process performance is discussed in more detail by Raju (1998). Here, we argue that this limited learning is due to the fact that data are typically generated in certain portions of the feature space while other portions are left unexplored. As a result, much of the data gathered from the experimental runs did not provide any additional information. This may be due to the *a priori* expectation that certain regions would lead to desirable process performance. A more active mode of learning involving the selection of interesting data to learn from and designing the appropriate experimental data to generate can have a significant impact on learning and the amount of experimental data required to learn the relationship.

Active learning, experimental design and Bayes theory

Active learning has strong analogues in the statistical design of experiments (Montgomery, 1984) literature. In contrast to full factorial or fractional factorial designs (Hogg and Ledolter, 1992) such as the Taguchi (Schmidt and Launsby, 1992) or Plackett-Burman designs that assume linear models, and response surface designs (Box and Draper, 1987) such as Box-Behnken or Central Composite designs that assume polynomial models, active learning using a neural network is analogous to sequential or stagewise designs such as the D-optimal designs (Federov, 1972) that assume nonlinear models. Similarly, active learning using neural networks has strong analogues in the Bayes theory (Mackay, 1992a) literature. Richard and Lippmann (1991) explain that many neural network classifiers including multilayer perceptron (MLP) networks trained with backpropagation and radial basis function (RBF) networks provide outputs that estimate Bayesian *a* *posteriori* probabilities. A quantitative practical Bayesian framework for backpropagation networks (Mackay, 1992b) allows for quantified estimates of error bars on network parameters and outputs. Within a Bayesian learning framework, different information-based objective functions for active data selection (Mackay, 1992c) have been derived showing similarities with the D-optimal and Q-optimal designs in the optimal experimental design literature (Federov, 1972).

The optimal experimental design (OED) based approach to active learning involves choosing the next example so as to minimize the expected variance of the network. The standard optimal experimental design approach assumes normality and linearity and has the advantage of being optimal given the assumptions (Cohn et al., 1995). However, for a neural network, these assumptions hold only approximately and computing the variance requires the inversion of a large Hessian matrix for each new example. In contrast, the computation of confidence limits for an HBF network can be done efficiently.

Case study simplifications

In comparison to previous work in this area that has focused on using computer simulated data of mathematical functions such as the sine wave to study the impact of active learning, we have used a real example in terms of the source of data (actual process rather than simulation), purpose of data gathering (to actually develop a real bioprocess rather than to study active learning), and the model system (actual fermentation process of significant commercial importance and complexity rather than a sine-wave type model system). However, this case study also involved important simplifications. First, we assumed that the labels associated with different metabolic states were known to some level of precision and accuracy. This is reasonable for the purposes of this article because the focus of this work is to examine the data gathering process rather than the labeling process. In reality, this knowledge is usually implicit, ill-coded, inconsistent, approximate and even inaccurate in many real-world applications. Understanding the impact of imperfect labeling on learning is described in more detail by Raju (1988). Second, in performing active designed learning we assumed that desired levels of DO, RQ, and sOUR levels could be reached experimentally. In reality, as shown in Figure 4, these variables cannot always be directly manipulated during a fermentation and there is a nonobvious mapping between the actual manipulated variables such as air-flow rate, stirring rate, temperature, pressure, pH, and these three variables. Some areas of the feature space may be difficult to reach. Third, we have chosen a fermentation model system that is fairly well understood (O'Connor et al., 1992) compared to many industrial fermentation systems. A relatively well-defined system and a university laboratory setting were chosen so that the approach and results of the research could be evaluated on some known basis. It would be useful to extend this research to industrial case studies that are less defined. Using mass spectrometer readings to measure oxygen consumption and carbon dioxide production to follow the progress of fermentations is quite prevalent. There are many industrial fermentations where mass spectrometer readings are either not available or not useful based on the nature of the product. While this may change the nature of the measurements, the issues of data gathering, selecting and design, which are the focus of this article, are likely to be the same.

Computational costs

While active learning proved to be superior in terms of the rate of learning, there are tradeoffs. One such tradeoff involves computer time. Passive learning involves using batches of data as examples to learn from. In contrast, active learning involves adding new examples into the database one by one based on confidence limits. For each new example, this involves retraining the entire network. In the case study used above, passive learning of 3,550 examples taking approximately 50 examples at a time took a total of 1.1 h of computer time on Gateway 2000 PentiumPro PC using Matlab. Selected active learning of the same 3,550 examples one by one took 319 h of computer time. In this case study, we assume that the extra benefit associated with having to label fewer examples and achieving a lower generalization error is worth the extra cost associated with computer time. Designed active learning required less than 65 h of computer time to achieve a much lower generalization error, because many fewer examples were used. More importantly, this mode of learning could potentially save experimentation time as fewer experiments are required.

In applications where data collection and labeling is inexpensive, it is possible that the additional computational cost associated with active learning may be significant. For such applications, it is possible to speed up the active learning process by adding in the high confidence limit examples in batches rather than one by one. This can result in a significant reduction in computational time. The computation time for selected active learning was reduced from 319 h to 4.5 h by increasing the batch size to 50 with little or no impact on the learning curve. Similarly, the computational time for active designed learning was reduced from 65 h to 1 h by increasing the batch size to 50. This puts active learning on a par with the passive learning case in terms of computational time. In general, however, the rate of learning decreases as the batch size increases. While the reduction in computational time is expected, the tradeoff between computational time and generalization error may be different for different types of applications. One might expect that increasing the batch size beyond a certain level would slow down the rate of learning significantly.

Limitations of active learning method

Active learning as implemented here is based on choosing the next example based on its confidence limit. The confidence limit computation is based on the assumption that the current neural network model is correct. We know that at any given time the current neural network model is only an approximation. Since HBF networks are primarily interpolation models, it may be possible to use the current model for interpolation. However, it is difficult to justify using this approximation for extrapolation. To a large extent, this limitation is avoided by initializing the network with a set of equally spaced examples distributed over the grid as done in the case of designed active learning (Figure 10). Also, since confidence limits are usually largest beyond the most extreme points where data have been gathered, this usually leads to the active learning algorithm calling for more data near an edge. This may be beyond the region of our interest or beyond the area of feasibility. This issue can be addressed by the appropriate choice of feature space and use of constraints.

Conclusions

In this article, we argued that data quantity and quality issues associated with practical process applications may justify a change in the mode of interaction between learner and domain assumed in recent chemical engineering connectionist learning research.

We explored the consequences of a shift from passive learning to a more active mode of connectionist learning. Results from an actual process development case study indicated that:

- The use of data collected from the process during the process development program resulted in little learning beyond the first few examples.
- Selected active learning from an existing database of process development data did not lead to a significant enhancement in the rate of learning.
- Random passive learning from the application led to a significantly increased rate of learning.
- Designed active learning resulted in an enhanced rate of learning.

Rather than viewing the learner as a passive receptacle of examples, the learner should have a significantly more active role in data selection and design. This can have a dramatic impact on the rate of learning. The rate of learning, in turn, can have a direct impact on process development. This is important, because process development can have a significant impact on time-to-market and process performance.

Acknowledgments

We would like to acknowledge the help of Prof. Michael Jordan for his encouragement and helpful comments. In addition, G. K. Raju would like to acknowledge the financial support provided by the National Research Council through the Integrated Manufacturing Fellowship, the MIT Biotechnology Process Engineering Center, and the Alfred P. Sloan Foundation through the MIT Program on the Pharmaceutical Industry.

Literature Cited

Basila, M. R., G. Stefanek, and A. Cinar, "A Model-Object Based Supervisory Expert System for Fault Tolerant Chemical Reactor Control," *Comput. and Chem. Eng.*, **14**, 551 (1990).

Bhakshi, B., and G. Stephanopoulos, "Wavenet: a Multiresolution Hierarchical Neural Network with Localized Learning," *AIChE J.*, **39**, 57 (1993).

Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford Univ. Press, New York (1995).

Box, G. E. P., and N. R. Draper, *Empirical Model-Building and Response Surfaces*, Wiley, New York (1987).

Carbonell, J., ed., *Machine Learning*, MIT Press, Cambridge, MA (1990).

Chen, Q., and W. A. Weigand, "Dynamic Optimization of Nonlinear Processes by Combining Neural Net Model with UDMC," *AIChE J.*, **40**, 1488 (1994).

Chen, J., D. S. H. Wong, and S.-S. Jang, "Product and Process Development Using Artificial Neural-Network Model and Information Analysis," *AIChE J.*, **44**, (1998).

Cohn, D., L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning*, **15**, 201 (1994).

Cohn, D. A., Z. Ghahramani, and M. I. Jordan, "Active Learning with Statistical Models," *Advances in Neural Information Processing Systems 7*, Morgan Kaufmann, San Francisco, CA (1995).

Cortes, C., L. D. Jackel, and W.-P. Chiang, "Limits on Learning Machine Accuracy Imposed by Data Quality," *Advances in Neural Information Processing Systems*, **6**, 239, Morgan Kaufmann, San Francisco (1994).

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. Control Signals Systems*, **2**, 303 (1989).

Duda, R. O., and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York (1973).

Federov, V. V., *Theory of Optimal Experiments*, Academic Press, New York (1972).

Girosi, F., "Some Extensions of Radial Basis Functions and Their Applications in Artificial Intelligence," *Comput. Math. Applic.*, **24**, 61 (1992).

Girosi, F., M. Jones, and T. Poggio, "Regularization Theory and Neural Networks Architectures," *Neural Computation*, **7**, 219 (1995).

Glassey, J., G. A. Montague, A. C. Ward, and B. V. Kara, "Artificial Neural Network Based Experimental Design Procedures for Enhancing Fermentation Development," *Biotechnol. and Bioeng.*, **44**, 397 (Aug. 1994).

Hayes, R., and S. Wheelwright, "The Experience Curve," *Restoring Our Competitive Edge*, Wiley, New York (1984).

Hogg, R. V., and J. Ledolter, *Applied Statistics for Engineers and Physical Scientists*, 2nd ed., Macmillan, New York (1992).

Holland, J. H., K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, *Induction: Processes of Inference, Learning and Discovery*, MIT Press, Cambridge, MA (1989).

Hoskins, J. C., K. M. Kaliyur, and D. M. Himmelblau, "Fault Diagnosis in Complex Chemical Plants Using Artificial Neural Networks," *AIChE J.*, **37**, 137 (1991).

Jin, S., K. Ye, K. Shimizu, and J. Nikawa, "Application of Artificial Neural Network and Fuzzy Control for Fed-Batch Cultivation of Recombinant *Saccharomyces cerevisiae*," *J. of Ferment. and Bioeng.*, **81**, 412 (1996).

Johnson, R. A., and D. W. Wichern, *Applied Multivariate Statistical Analysis*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ (1988).

Karjala, T. W., and D. M. Himmelblau, "Dynamic Data Rectification by Recurrent Neural Networks vs. Traditional Methods," *AIChE J.*, **40**, 1865 (1994).

Konstantinov, K. B., and T. Yoshida, "Knowledge-Based Control of Fermentation Processes," *Biotechnol. and Bioeng.*, **39**, 479 (1992).

Kramer, M. A., "Autoassociative Neural Networks," *Computers and Chem. Eng.*, **16**, 299 (Apr. 1992),

Kulkarni, S. R., S. K. Mitter, and J. N. Tsitsiklis, "Active Learning Using Arbitrary Binary Valued Queries," *Machine Learning*, **11**, 23 (1993).

Leonard, J. A., "A Reliable Neural Network Architecture for Fault Diagnosis and Modeling of Chemical Processes," PhD Thesis, Mass. Inst. of Technol., Cambridge (Dec., 1991).

Leonard, J. A., and M. A. Kramer, "Radial Basis Function Networks for Classifying Process Faults," *IEEE Control. Sys. Mag.*, **11**, 31 (1991).

MacKay, D. J. C., "Bayesian Interpolation," *Neural Computation*, **4**, 415 (1992a).

MacKay, D. J. C., "A Practical Bayesian Framework for Backpropagation Networks," *Neural Computation*, **4**, 448 (1992b).

MacKay, D. J. C., "Information-Based Objective Functions for Active Data Selection," *Neural Computation*, **4**, 590 (1992c).

Magdalena, L., and F. Monasterio-Huelin, "A Fuzzy Logic Controller with Learning Through the Evolution of its Knowledge Base," *Int. J. Approximate Reasoning*, **16**, 335 (1997).

Massimo, C. D., G. A. Montague, M. J. Willis, M. T. Tham, and A. J. Morris, "Towards Improved Penicillin Fermentation via Artificial Neural Networks," *Comput. and Chem. Eng.*, **16**, 283 (Apr., 1992).

Montgomery, D. C., *Design and Analysis of Experiments*, 2nd ed., Wiley, New York (1984).

Moody, J., and C. J. Darken, "Fast Learning in Networks of Locally Tuned Processing Units," *Neural Computation*, **1**, 281 (1989).

Negoita, C. V., *Expert Systems and Fuzzy Systems*, Benjamin/Cummings Publishing Co., Plenlo Park, CA (1985).

O'Connor, G. M., "Development of an Intelligent Fermentation Control System," PhD Thesis, Mass. Inst. of Technol., Cambridge (Sept., 1989).

O'Connor, G. M., F. Sanchez-Riera, and C. L. Cooney, "Design and Evaluation of Control Strategies for High Cell Density Fermentations," *Biotechnol. and Bioeng.*, **39**, 293 (1992).

Park, J., and I. W. Sandberg, "Universal Approximation Using Radial Basis Function Networks," *Neural Computation*, **3**, 246 (1991).

Pierce, J. R., *An Introduction to Information Theory: Symbols, Signals and Noise*, 2nd ed., Dover Publications, New York (1980).

Poggio, T., and F. Girosi, "Regularization Algorithms for Learning that are Equivalent to Multilayer Networks," *Sci.*, **247**, 978 (1990).

Raju, G. K., and C. L. Cooney, "Fermentation Monitoring," *Current Opinion in Biotechnol.*, **3**, 40 (1992).

Raju, G. K., "Pharmaceutical Manufacturing: Structuring Organizational Learning through Benchmarking," Masters Thesis, Mass. Inst. of Technol., Cambridge (June, 1994).

Raju, G. K., "Automatic Learning From Process Data Using Neural Network Methodologies," PhD Thesis, Mass. Inst. of Technol., Cambridge (June, 1998).

Richard, M. D., and R. P. Lippmann, "Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities, *Neural Computation*, **3**, 461 (1991).

Rojas-Guzman, C., "An Evolutionary Programming Approach to Probabilistic Model-based Fault Diagnosis of Chemical Processes," PhD Thesis, Mass. Inst. of Technol., Cambridge (1995).

Romanos, M. A., C. A. Scorer, and J. J. Clare, "Foreign Gene Expression in Yeast: a Review," *Yeast*, **8**, 423 (1992).

Saraiva, P., and G. Stephanopoulos, "Continuous Process Improvement Through Inductive and Analogical Learning," *AIChE J.*, **38**, 161 (1992).

Schaal, S., and C. G. Atkeson, "Assessing the Quality of Learned Local Models," *Advances in Neural Information Processing Systems*, Morgan Kaufmann, San Francisco (1994).

Schmidt, S. R., and R. G. Launsby, *Understanding Industrial Designed Experiments*, 3rd ed., Academy Press, Colorado Springs, CO (1992).

Senge, Peter M., "Mental Models," *Planning Rev.*, (Mar-Apr., 1992).

Sridhar, D. V., R. C. Seagrave, and E. B. Bartlett, "Process Modeling Using Stacked Neural Networks," *AIChE J.*, **42**, 2529 (1996).

Stephanopoulos, G., G. Locher, M. J. Duff, and R. Kamimura, "Fermentation Data Mining by Pattern Recognition," *Biotechnol. and Bioeng.*, **53**, 443 (1997).

Strata, Ray, "Organizational Learning—The Key to Management Innovation," *Sloan Mgmt. Rev.* (Spring, 1989).

Tan, S., and M. L. Mavrovouniotes, "Reducing Data Dimensionality through Optimizing Neural Network Inputs," *AIChE J.*, **41**, 1471 (1995).

Tholudur, A., and W. F. Ramirez, "Optimization of Fed-Batch Bioreactors Using Neural Network Parameter Function Models," *Biotechnol. Prog.*, **12**, 302 (1996).

Thompson, M. L., and M. A. Kramer, "Modeling Chemical Processing Using Prior Knowledge and Neural Networks," *AIChE J.*, **40**, 1328 (1994).

Ungar, L. H., B. A. Powell, and S. N. Kamens, "Adaptive Networks for Fault Diagnosis and Process Control," *Comput. and Chem. Eng.*, **14**, 461 (1990).

Venkatasubramanian, V., R. Vaidyanathan, and Y. Yamamoto, "An Analysis of the Learning, Recall, and Generalization Characteristics of Neural Networks for Process Fault Diagnosis," *Comput. and Chem. Eng.*, **14**, 699 (1990).

Watanabe, K., S. Hirota, L. Hou, and D. M. Himmelblau, "Diagnosis of Multiple Simultaneous Faults via Hierarchical Artificial Neural Networks," *AIChE J.*, **40**, 839 (1994).

Yao, S. C., and E. Zafiriou, "Control System Sensor Failure Detection via Networks of Localized Receptive Fields," *Proc. Amer. Control Conf.*, San Diego (1990).

Manuscript received Sept. 15, 1997, and revision received June 17, 1998.