Control of Inductive Bias in Supervised Learning using Evolutionary Computation: A Wrapper-Based Approach

William H. Hsu
Department of Computing and Information Sciences, Kansas State University, USA
Automated Learning Group, National Center for Supercomputing Applications (NCSA), USA

Abstract

In this chapter, I discuss the problem of *feature subset selection* for supervised inductive learning approaches to knowledge discovery in databases (KDD), and examine this and related problems in the context of controlling *inductive bias*. I survey several combinatorial search and optimization approaches to this problem, focusing on data-driven validation-based techniques. In particular, I present a *wrapper* approach that uses genetic algorithms for the search component, using a validation criterion based upon model accuracy and problem complexity, as the fitness measure. Next, I focus on design and configuration of high-level optimization systems (wrappers) for relevance determination and constructive induction, and on integrating these wrappers with elicited knowledge on attribute relevance and synthesis. I then discuss the relationship between this model selection criterion and those from minimum description length (MDL) family of learning criteria. I then present results on several synthetic problems on task-decomposable machine learning and on two large-scale commercial data mining and decision support projects: crop condition monitoring, and loss prediction for insurance pricing. Finally, I report experiments using the *Machine Learning in Java (MLJ)* and *Data to Knowledge (D2K)* Java-based visual programming systems for data mining and information visualization, and several commercial and research tools. Test set accuracy using a genetic wrapper is significantly higher than that of decision tree inducers alone and is comparable to that of the best extant search-space based wrappers.

Keywords: automatic relevance determination; constructive induction; decision support applications; evolutionary computation; feature subset selection; genetic algorithms; genetic programming; inductive bias; knowledge discovery (KD) problems, large-scale; machine learning problems, decomposable; model selection; programming systems for KDD; wrappers

INTRODUCTION

This chapter introduces the problems for change of representation (Benjamin, 1990) in supervised inductive learning. I address the focal problem of inductive learning in data mining and present a multistrategy framework for automatically improving the representation of learning problems. This framework incorporates methodological aspects of *feature subset selection and feature (attribute) partitioning*, *automated problem decomposition*, and *model selection*. The focus is on *wrapper-based* methods as studied in recent and continuing research.

As an example, I present a new metric-based model selection approach (composite learning) for decomposable learning tasks. The type of data for which this approach is best suited is heterogeneous time series data – that arising from multiple sources of data (as in sensor fusion or multimodal human-computer interaction tasks, for example). The rationale for applying multistrategy learning to such data is that, by systematic analysis and transformation of learning tasks, the efficiency and accuracy of classifier learning may both be improved for certain time series problems. Such problems are referred to in this chapter as *decomposable*; the methods addressed are: task decomposition and subproblem definition, quantitative model selection, and construction of hierarchical mixture models for data fusion. This chapter presents an integrated, multistrategy system for decomposition of time series classifier learning tasks.

A typical application for such a system is learning to predict and classify hazardous and potentially catastrophic conditions. This prediction task is also known as *crisis monitoring*, a form of pattern recognition that is useful in decision support or *recommender* systems (Resnick & Varian, 1997) for many time-critical applications. Examples of crisis monitoring problems in the industrial, military, agricultural and environmental sciences are numerous. They include: crisis control automation (Hsu *et al*, 1998), online medical diagnosis (Hayes-Roth *et al*, 1996), simulation-based training and critiquing for crisis management (Gaba *et al*, 1992; Grois *et al*, 1998), and intelligent data visualization for *real-time decision-making* (Horvitz & Barry, 1995).

Motivation: Control of Inductive Bias

The broad objectives of the approach I present here are to increase the robustness of inductive machine learning algorithms and develop learning systems that can be automatically tuned to meet the requirements of a knowledge discovery (KD) performance element. When developers of learning systems can map a KD application to a set of automatic higher-order parameter turning problems, the reuse of design and code embodied by this generalization over traditional learning can reduce development costs. When addressing KD problems in computational science and engineering, the time required to develop an effective representation and to tune these hyperparameters using training and validation data sets can be a significant fraction of the development time of the KD system, exceeding the time required to apply traditional learning algorithms with fixed hyperparameters and bias parameters. This setting introduces new flexibility and complexity into the learning problem and may extend the expected useful lifetime of the system. For example, if the learning component is made more adaptable through automated performance tuning, then the overall system, not merely the learning algorithms it uses, may last longer than one tailored to a specific data set or problem domain. Thus it becomes subject to traditional maintenance and evolution. On the other hand, performance tuning may reduce the development time of highly specialized KD systems as well, by identifying and constructing relevant variables. In this case reducing the cost of developing the more limited-use software can, in turn, significantly reduce that of solving the intended scientific or engineering problem. In many real-world KD applications, it is preferable to automatically tune some but not all of the available bias parameters to prevent overfitting of training data. This is because the computational time savings for the performance element (e.g., prediction, classification, or pattern detection function) and marginal gains in solution quality (e.g., utility or accuracy) do not make it worth while to fine-tune some bias parameters that are less significant for the learning problem. A significant component of development costs is related to reducing wasted development time and computation time by making the entire programming systems product (Brooks, 1995) responsive and adaptable to end user needs. Combinatorial search and statistical validation over representations, visualization of the models and their relation to quantitative inductive bias (Benjamin, 1990; Mitchell, 1997), and high-level user interfaces for KD can be applied to achieve these goals.

A major motivation for the automation of problem transformation is *transparency*. The end user of a KD system is often a specialist in scientific, engineering, or business-related technical fields other than intelligent systems and machine learning. He or she knows the requirements of the application in terms of the performance element: an analytical function that can predict the continuation of a historical time series; detect anomalous conditions in a time annotated episodic log; classify, diagnose, or explain set of database records; make a recommendation for a business or medical decision; or generate a plan, schedule, or design. These predictors, anomaly detectors, classifiers, diagnostic and recommender systems, policies, and other problem solvers have their own performance measures, perhaps including real-time requirements, which in turn dictate those of the learning system. This suggests that more robust KD may be achieved by letting the end user specify requirements pertaining to the performance element and automatically generating specifications for the desired representation and higher-order parameters to be tuned. In this way the improvement of problem representation by automated transformation can be driven by users' specified time and resource constraints.

The research covered in this chapter focuses on demonstrating, through development of a learning enhancement framework and through empirical evaluation, that these broad objectives can be achieved for a wide variety of real-world KD applications. This research thrust has two main objectives: assessing the breadth of applicability of automatic transformation of learning problems by training the resultant models and applying them to large-scale KD problems over real-world data sets, and developing information visualization techniques to help users understand this process of improving problem representations.

Attribute-Driven Problem Decomposition: Subset Selection and Partition Search

Many techniques have been studied for decomposing learning tasks, to obtain more tractable subproblems and to apply multiple models for reduced variance. This section examines *attribute-based* approaches for problem reformulation, especially *partitioning* of input attributes in order to define *intermediate concepts* (Fu & Buchanan, 1985) in problem decomposition. This mechanism produces multiple subproblems for which appropriate models must be selected; the trained models can then be combined using *classifier fusion* models adapted from bagging (Breiman, 1996), boosting (Freund & Schapire, 1996), stacking (Wolpert, 1992) and hierarchical mixture models (Jordan & Jacobs, 1994).

One of the approaches we shall examine in this chapter uses partitioning to *decompose* a learning task into parts that are individually useful (using *aggregation* as described in the background section of this chapter), rather than to *reduce* attributes to a single useful group. This permits new intermediate concepts to be formed by unsupervised learning methods such as conceptual clustering (Michalski & Stepp, 1983) or cluster formation using self-organizing algorithms (Kohonen, 1990; Hsu *et al*, 2002). The newly defined problem or problems can then be mapped to one or more appropriate hypothesis languages (model specifications). In our new system, the subproblem definitions obtained by partitioning of attributes also specify a mixture estimation problem (i.e., data fusion step occurs after training of the models for all the subproblems).

Subproblem Definition

This purpose of attribute partitioning is to define intermediate concepts and subtasks of decomposable time series learning tasks, which can be mapped to the appropriate submodels. In both attribute subset selection and partitioning, attributes are grouped into subsets that are relevant to a particular task: the overall learning task or a subtask. Each subtask for a partitioned attribute set has its own inputs (the attribute subset) and its own *intermediate concept*. This intermediate concept can be discovered using unsupervised learning methods, such as self-organizing feature maps (Kohonen, 1990; Hsu *et al*, 2002) and *k-means clustering* (Duda *et al*, 2000). Other methods, such as *competitive clustering* or *vector quantization* using radial basis functions (Haykin, 1999), *neural trees* (Li *et al*, 1993) and similar models (Ray & Hsu, 1998;

Duda *et al*, 2000), *principal components analysis* (Watanabe, 1985; Haykin, 1999), *Karhunen-Loève transforms* (Watanabe, 1985), or *factor analysis* (Watanabe, 1985), can also be used.

Attribute partitioning is used to control the formation of intermediate concepts in this system. Whereas attribute subset selection yields a *single*, reformulated learning problem (whose intermediate concept is neither necessarily nor intentionally different from the original concept), attribute partitioning yields *multiple learning subproblems* (whose intermediate concepts may or may not differ, but are simpler by design when they do). The goal of this approach is to find a natural and principled way to specify *how* intermediate concepts should be simpler than the overall concept.

Metric-Based Model Selection and Composite Learning

Model selection is the problem of choosing a hypothesis class that has the appropriate complexity for the given training data (Stone, 1977; Schuurmans, 1997). Quantitative methods for model selection have previously been used to learn using highly flexible *nonparametric* models with many degrees of freedom, but with no particular assumptions on the structure of decision surfaces.

The ability to decompose a learning task into simpler subproblems prefigures a need to map these subproblems to the appropriate models. The general mapping problem, broadly termed *model selection*, can be addressed at very minute to very coarse levels. This chapter examines quantitative, metric-based approaches for model selection at a coarse level. This approach is a direct extension of the *problem definition and technique selection* process (Engels *et al*, 1998). We will henceforth use the term *model selection* to refer to both traditional model selection and the metric-based methods for technique selection as presented here. We begin with background on the general framework of inductive bias control and then survey time series learning architectures, their *representation biases* (Witten and Frank, 2000), and methods for selecting them from a collection of model components.

BACKGROUND

Key Problem: Automated Control of Inductive Bias

We first focus on development of a new learning system for spatiotemporal KD. The KD performance element in this problem is not just analytical but includes decision support through model visualization and anomaly detection.

The problems we face are threefold and are surveyed in the above sections on current and related work. First, we must address *relevance determination* to determine what sensor data channels are useful for a particular KD objective and data set. This problem is related to the so-called *curse of dimensionality* wherein an overabundance of input variables makes it difficult to learn the prediction, classification, or pattern recognition element. Second, the task of identifying hyperparameters of the KD system is subject to deceptivity and instability because bias optimization in general introduces a new level of combinatorial complexity to the problem of inductive learning. Third, the very large spatiotemporal databases we are dealing with are highly heterogeneous, arising from many disparate sources such as global positioning systems (GPS), surveying, sensors such as radar, and historical databases; this heterogeneity presents the advantage of type information that can help constrain dimensionality but also aggravates the problem of relevance determination because including irrelevant sensors can lead to severe inefficiencies in data acquisition, interpretation of visual results, and the learning component.

In (Hsu *et al*, 2000), I address these problems through a framework called *composite learning* that is depicted in Figure 1. We define a composite learning system to be a committee machine (Haykin, 1999) designed to improve the performance of a collection of supervised inductive learning algorithms with specific parameters for representation and preference bias (Mitchell, 1997), over multivariate – possibly heterogeneous – data. The open research problem I discuss is how composite learning systems can be applied to automatically improve representations of complex learning problems.

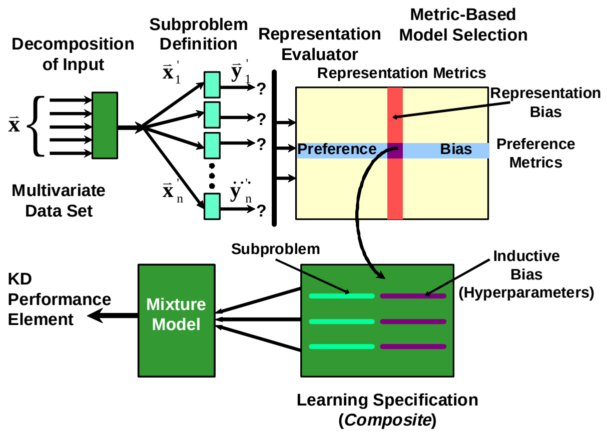


Figure 1. A Composite Learning Framework

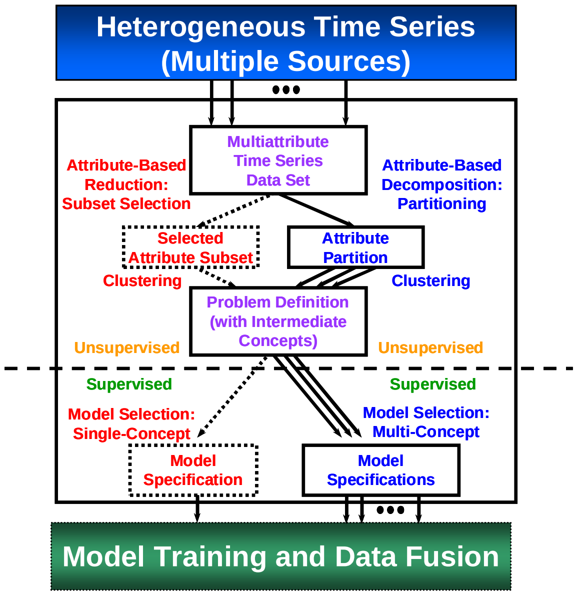

Heterogeneous Time Series (Multiple Sources)
Multiattribute Time Series Data Set
Attribute-Based Reduction: Subset Selection
Attribute-Based Decomposition: Partitioning
Selected Attribute Subset
Attribute Partition
Clustering
Clustering
Problem Definition (with Intermediate Concepts)
Unsupervised
Unsupervised
Supervised
Supervised
Model Selection: Single-Concept
Model Selection: Multi-Concept
Model Specification
Model Specifications
Model Training and Data Fusion

Figure 2. Systems for Attribute-Driven Unsupervised Learning and Model Selection

Composite learning provides a search-based and validation-based procedure for controlling the inductive bias, specifically the total problem specification, of systems for learning from decomposable, multi-attribute data sets. The central elements of this system are: *decomposition of input*, *metric-based model selection*, and a *mixture model* for integration of multiple submodels. In recent research, I applied composite learning to audio signal classification (Ray & Hsu, 1998) and crop condition prediction, the central component of a monitoring problem (Hsu, 1998; Hsu *et al*, 2000). Given a specification for decomposed – i.e., selected or partitioned – subsets of input variables, new intermediate concepts $\dot{\mathbf{y}}_i'$ can be formed by unsupervised learning. For this step we have used Gaussian *radial-basis functions* or RBFs (Ray & Hsu, 1998) and *self-organizing maps* (Kohonen, 1990). The newly defined problem or problems can then be mapped to one or more appropriate hypothesis languages (model specifications). We have developed a high-level algorithm for tuning explicit parameters that control representation and preference bias, to generate this specification of a composite. This algorithm is used by Hsu *et al* (2000) to select components for a hierarchical mixture network (specialist-moderator network) and train them for multistrategy learning. A data fusion step occurs after individual training of each model. The system incorporates attribute partitioning into constructive induction to obtain multiple problem definitions (decomposition of learning tasks); applies metric-based model selection over subtasks to *search for efficient hypothesis preferences*; and integrates these techniques in a data fusion (mixture estimation) framework. The metrics we have derived for controlling preference bias in hierarchical mixture models are positively correlated with learning performance by a particular learning method (for a learning problem defined on a particular partitioning of a time series). This makes them approximate indicators of the suitability of the corresponding mixture model and the assumption that the learning problem adheres to its characteristics (with respect to interaction among subproblems). Thus, preference bias metrics may be used for partial model selection.

Although this approach has yielded positive results from applying composite learning to KD, the breadth of domains for which this framework has been tested is still limited. A current research challenge and opportunity is the application of composite learning to learning problems in precision agriculture, specifically the illustrative example in and the problem of *soil fertility mapping*, which generates a map of quantitative fertility estimates from remotely sensed, hydrological, meteorological, wind erosion, and pedological data. One purpose of generating such maps is to control variable-rate fertilizer application to increase yield with minimal environmental impact. Test bed for heterogeneous data mining abound in the literature and are becoming freely available.

In past and current research, we have achieved empirical improvement in constructive induction in several ways in which we propose to further generalize and systematically validate. First, we found that decomposition of learning tasks using techniques such as attribute partitioning or ensemble selection can help reduce variance when computational resources are limited. We conjecture that this may be useful in domains such as real-time intelligent systems, where deadlines are imposed on training and inference time.

Current techniques such as *automated relevance determination*, *feature selection*, and *clustering* tend to address the problem of constructive induction in isolated stages rather than as an integrative mechanism for transforming the data – input and output schemata – into a more tractable and efficient form. As outlined in the previous section, we address this by combining search-based combinatorial optimization, statistical validation, and hierarchical abstraction into the coherent framework of composite learning.

Furthermore, many complex KDD problems can be decomposed on the basis of spatial, temporal, logical, and functional organization in their performance element. Techniques such as *model selection* and *ensemble learning* have been used to systematically identify and break down these problems, and, given a specification of a modular learning system, *hierarchical mixture estimation* techniques have been used to build pattern recognition models by parts and integrate them. The challenge is how to isolate prediction or classification models. The author has identified several low-order Box-Jenkins (autoregressive integrated moving average, *aka* ARMA or ARIMA) process models (Box *et al*, 1994) that can be isolated from heterogeneous historical data, based on quantitative metrics (Hsu, 1998). Composite learning can be applied to derive a complete committee machine specification from data to learn intermediate predictors (e.g., temporal artificial neural networks such as simple recurrent networks and time-delay neural networks). We believe that this approach can discover other hierarchical organization such as embedded clusters (Hsu *et al*, 2002), factorial structure (Ray & Hsu, 1998), and useful behavioral structure, which we shall outline in the next section on evolutionary computation for KD. The proposed research is therefore not specific to time series.

The line of research that we have described in this section shall lead to the development of techniques for making inductive learning more robust by controlling inductive bias to increase generalization accuracy. We propose to use my framework, composite learning, for specifying high-level characteristics of the problem representation to be tuned in a systematic way. The next section presents a specific combinatorial optimization technique for tuning these hyperparameters using validation and other criteria.

Evolutionary Computation Wrappers for Enhanced KD

Over the past three years we have been engaged in the development of a novel system for combinatorial optimization in KD from complex domains, which uses evolutionary computation – genetic algorithms (GA) and genetic programming (GP) – to enhance the machine learning process. Mechanisms for KD enhancement that use the empirical performance of a learning function as feedback are referred to in the intelligent systems literature as *wrappers* (Kohavi & John, 1997). Our objective at this stage of the research is to relax assumptions we have previously made regarding two aspects of automatically improving the representation of learning problems. First, we seek to generalize the *structure* of the mapping between the original and improved representations, not restricting it merely to feature selection or construction. Second, we seek to design a framework for automatic discovery of hierarchical structure in learning problems, both from data and from reinforcements in problem solving environments. The key contribution of this component of the research is to make the automatic search for representations more systematic and reproducible by putting it into an engineering framework.

Problems: Attribute Subset Selection and Partitioning

This section introduces the *attribute partitioning* problem and a method for subproblem definition in multiattribute inductive learning.

Attribute-Driven Problem Decomposition for Composite Learning

Many techniques have been studied for decomposing learning tasks, to obtain more tractable subproblems and to apply multiple models for reduced variance. This section examines *attribute-based* approaches for problem reformulation, which start with restriction of the set of input attributes on which the supervised learning algorithms will focus. First, this chapter presents a new approach to problem decomposition that is based on finding a good *partitioning* of input attributes. Previous research on attribute subset selection (Kohavi & John, 1997), though directed toward a different goal for problem reformulation, is highly relevant; this section outlines differences between subset selection and partitioning and how partitioning may be applied to task decomposition. Second, this chapter compares top-down, bottom-up, and hybrid approaches for attribute partitioning, and considers the role of partitioning in feature extraction from heterogeneous time series. Third, it discusses how grouping of input attributes leads naturally to the problem of forming *intermediate concepts* in problem decomposition. This mechanism defines different subproblems for which appropriate models must be selected; the trained models can then be combined using *classifier fusion* models adapted from bagging (Breiman, 1996), boosting (Freund & Schapire, 1996), stacking (Wolpert, 1992), and hierarchical mixture models (Jordan & Jacobs, 1994).

Overview of Attribute-Driven Decomposition

Figure 2 depicts two alternative systems for attribute-driven reformulation of learning tasks (Benjamin, 1990; Donoho, 1996). The left-hand side, shown with dotted lines, is based on the traditional method of attribute *subset selection* (Kohavi & John, 1997). The right-hand side, shown with solid lines, is based on attribute *partitioning*, which is adapted in this chapter to decomposition of time series learning tasks. Given a specification for reformulated (reduced or partitioned) input, new intermediate concepts can be formed by unsupervised learning (e.g., conceptual clustering); the newly defined problem or problems can then be mapped to one or more appropriate hypothesis languages (model specifications). The new models are selected for a reduced problem or for multiple subproblems obtained by partitioning of attributes; in the latter case, a data fusion step occurs after individual training of each model.

Subset Selection and Partitioning

Attribute subset selection, also called *feature subset selection*, is the task of focusing a learning algorithm's attention on some subset of the given input attributes, while ignoring the rest (Kohavi & John, 1997). In this research, subset selection is adapted to the systematic decomposition of learning problems over heterogeneous time series. Instead of focusing a single algorithm on a single subset, the set of all input attributes is partitioned, and a specialized algorithm is focused on *each* subset. While subset selection is designed for refinement of attribute sets for single-model learning, attribute partitioning is designed specifically for multiple-model learning. This new approach adopts the role of feature construction in constructive induction (Michalski, 1983; Donoho, 1996), as depicted in Figure 2. It uses subset partitioning to *decompose* a learning task into parts that are individually useful, rather than to *reduce* attributes to a single useful group. This permits multiple-model methods such as *bagging* (Breiman, 1996), *boosting* (Freund & Schapire, 1996), and *hierarchical mixture models* (Jordan & Jacobs, 1994) to be adapted to multistrategy learning.

Partition Search

For clarity, I review the basic combinatorial problem of *attribute partitioning*. First, consider that the state space for attribute subset selection grows exponentially in the number of attributes n: its size is simply 2^n. The size of the state space for n attributes is B_n, the nth Bell number, defined as follows[1]:

$$B_n = \sum_{k=0}^{n} S(n,k)$$

$$S(n,k) = \begin{cases} 0 & if\ n < k\ or\ k = 0, n \neq 0 \\ 1 & if\ n = k \\ S(n-1,k-1) + kS(n-1,k) & otherwise \end{cases}$$

Thus, it is impractical to search the space exhaustively, even for moderate values of n. The function B_n is $\omega(2^n)$ and $o(n!)$, i.e., its asymptotic growth is strictly *faster* than that of 2^n and strictly *slower* than that of $n!$. It thus results in a highly intractable evaluation problem if all partitions are considered. Instead, a heuristic evaluation function is used so that informed search methods (Russell & Norvig, 1995) may be applied. This evaluation function is identical to the one used to prescribe the *multistrategy hierarchical mixture of experts* (*MS-HME*) model; therefore, its definition is deferred until the next section.

The state space for of a set of 5 attributes consists of 52 possible partitions. We shall examine a simple synthetic problem learning problem, *modular parity*, can be used to test search algorithms for the optimum partition. As the parity problem, a generalization of *XOR* to many variables, demonstrates the expressiveness of a representation for models or hypotheses in inductive learning (and was thus used to illustrate the limitations of the perceptron), the modular parity problem tests the expressiveness and flexibility of a learning system when dealing with heterogeneous data.

Subproblem Definition

This section summarizes the role of attribute partitioning in defining intermediate concepts and subtasks of decomposable time series learning tasks, which can be mapped to the appropriate submodels.

Intermediate Concepts and Attribute-Driven Decomposition

In both attribute subset selection and partitioning, attributes are grouped into subsets that are relevant to a particular task: the overall learning task or a subtask. Each subtask for a partitioned attribute set has its own inputs (the attribute subset) and its own *intermediate concept*. This intermediate concept can be discovered using unsupervised learning algorithms, such as *k-means clustering*. Other methods, such as competitive clustering or vector quantization (using radial basis functions (Lowe, 1995; Hassoun, 1995; Haykin, 1999), neural trees (Li *et al*, 1993), and similar models (Duda *et al*, 2000; Ray & Hsu; 1998), principal components analysis (Watanabe, 1985; Hassoun, 1995; Haykin, 1999), Karhunen-Loève transforms (Watanabe, 1985, Hassoun, 1995), or factor analysis (Watanabe, 1985; Duda *et al*, 2000), can also be used.

Attribute partitioning is used to control the formation of intermediate concepts in this system. Attribute subset selection yields a single, reformulated learning problem (whose intermediate concept is neither necessarily different from the original concept, nor intended to differ). By contrast, attribute partitioning yields multiple learning *subproblems* (whose intermediate concepts may or may not differ, but are simpler by design when they do differ).

The goal of this approach is to find a natural and principled way to specify *how* intermediate concepts should be simpler than the overall concept. In the next section, two mixture models are presented: the ***H****ierarchical* ***M****ixture of* ***E****xperts* (HME) of Jordan and Jacobs (1994), and the ***S****pecialist*-***M****oderator* (SM) network of Ray and Hsu (Ray & Hsu, 1998; Hsu *et al*, 2000). The following sections explain and illustrate why this design choice is a critically important consideration in how a hierarchical learning model is built, and how it affects the performance of multistrategy approaches to learning from heterogeneous time series. The mechanisms by which HME and SM networks perform data fusion, and how this process is affected by attribute partitioning, are examined in both theoretical and experimental terms in this chapter. Finally, a survey of experiments by the author investigates the empirical effects of attribute partitioning on learning performance, including its indirect effects through intermediate concept formation.

Role of Attribute Partitioning in Model Selection

Model selection, the process of choosing a hypothesis class that has the appropriate complexity for the given training data (Geman *et al*, 1992; Schuurmans, 1997), is a consequent of attribute-driven problem decomposition. It is also one of the original directives for performing decomposition (i.e., to apply the appropriate learning algorithm to each homogeneous subtask). Attribute partitioning is a determinant of subtasks, because it specifies new (restricted) views of the input and new target outputs for each model. Thus, it also determines, indirectly, what models are called for. This system organization may be described as a *wrapper* system *cf.* (Kohavi & John, 1997) whose primary adjustable parameter is the attribute partition. A second parameter is a high-level model descriptor (the architecture and type of hierarchical *classifier fusion* model).

Machine Learning Methodologies: Models and Algorithms

Recurrent Neural Networks and Statistical Time Series Models

SRNs, TDNNs, and gamma networks (Mehrotra *et al*, 1997) are all temporal varieties of artificial neural networks (ANNs). A *temporal naïve Bayesian network* is a restricted type of Bayesian network called a *global knowledge map* as defined by Heckerman (1991), which has two stipulations. The first is that some random variables may be temporal (e.g., they may denote the durations or rates of change of original variables). The second is that the topological structure of the Bayesian network is learned by naïve Bayes. A hidden Markov model (HMM) is a stochastic state transition diagram whose transitions are also annotated with probability distributions over output symbols (Lee, 1989).

The primary criterion used to characterize a stochastic process in my multistrategy time series learning system is its *memory form*. To determine the memory form for temporal ANNs, two properties of statistical time series models are exploited. The first property is that the temporal pattern represented by a memory form can be described as a *convolutional code*. That is, past values of a time series are stored by a particular type of recurrent ANN, which transforms the original data into its internal representation. This transformation can be formally defined in terms of a *kernel function* that is convolved over the time series. This convolutional or functional definition is important because it yields a general mathematical characterization for individually weighted "windows" of past values (time delay or *resolution*) and nonlinear memories that "fade" smoothly (attenuated decay, or *depth*) (Principé & deVries, 1992; Mozer, 1994; Principé & Lefebvre, 2001). It is also important to metric-based model selection, because it concretely describes the transformed time series that we should evaluate, in order to compare memory forms and choose the most effective one. The second property is that a transformed time series can be evaluated by measuring the change in *conditional entropy* (Cover & Thomas, 1991) for the stochastic process of which the training data is a sample. The entropy of the next value conditioned on past values of the *original* data should, in general, be higher than that of the next value conditioned on past values of the *transformed* data. This indicates that the memory form yields an improvement in predictive capability, which is ideally proportional to the expected performance of the model being evaluated.

Given an input sequence $\mathbf{x}(t)$ with components $\{\hat{\mathbf{x}}_i(t), 1 \le i \le n\}$, its convolution $\hat{\mathbf{x}}_i(t)$ with a kernel function $c_i(t)$ (specific to the i^{th} component of the model) is defined as follows:

$$\hat{\mathbf{x}}_i(t) = \sum_{k=0}^{t} c_i(t-k)\,\mathbf{x}(k)$$

(Each $\mathbf{x}$ or $\mathbf{x}_i$ value contains all the attributes in *one subset* of a partition.)

Kernel functions for simple recurrent networks, Gamma memories, and are presented in the context of convolutional codes and time series learning by Mozer (1994), Mehrotra *et al* (1997), and Hsu (1998). The interested reader may also refer to data sets such as the Santa Fe corpus (Gershenfeld & Weigend, 1994) and ANN simulation software for additional information, readers new to this family of learning models are encouraged to experiment with such test corpora and codes in order to gain basic experience.

Evolutionary Computation: Genetic Algorithms and Genetic Programming

The notion of using evolutionary computation to improve the representation of learning problems in KD draws from foundational work on controlling genetic algorithms and finds applications in evolutionary control and data mining using genetic algorithms as inducers.

In the field of evolutionary computation, many aspects of the genetic coding and evolutionary system can be tuned automatically. Much of the recent research has focuses on this meta-optimization problem and has led to both theoretical and empirical results on population sizing (Horn, 1997), probability of selection, crossover, and mutation (Goldberg, 1998), and parallel, distributed load balancing in genetic algorithms (Cantu-Paz, 1999). Genetic algorithms that tune some of their own hyperparameters are referred to in the literature as *parameterless* (Harik & Lobo, 1997). This idea has also been used to develop genetic wrappers for performance enhancement in KD, an innovation dating back to the first applications of genetic algorithms to inductive learning (Booker *et al*, 1989; Goldberg, 1989; Dejong *et al*, 1993).

We seek to optimize the representation and preference biases of a learning system for KD. Therefore, we are interested in four kinds of hyperparameter: input descriptors, output descriptors, specifications for what kind of committee machine or ensemble architecture to use, and control variables for the search algorithm (the choice of search algorithm itself, heuristic coefficients, and hyperparameters in various learning frameworks). The first three kinds of hyperparameter control representation bias, the fourth, preference bias. (Witten and Frank, 2000) This distinction is important in our study of evolutionary computation because it generates requirements for coding and fitness evaluation in our specification of combinatorial optimization problems. For example, finding intermediate learning targets can be formulated as an unsupervised learning problem, and the gene expression of an evolved selector, partition, or construction rule or program for describing these target outputs shall differ from that for inputs.

Koza (1992) defines five specification components for a GP system: determining the terminal set, function set, fitness cases or evaluation function, termination conditions, and result. The process of determining these drives the design of a GP-based wrapper. In data mining with evolutionary algorithms, many direct approaches have been made toward constructive induction: selecting and extracting features is very natural with a genetic algorithm because the hyperparameters (e.g., feature subsets) can be encoded as bit strings and, provided the proper parallel and distributed computing system is used, the task of evaluating fitness based upon model criteria and statistical validation data is trivially parallelizable. Similarly, with the proper encoding of synthetic variables as symbolic (e.g., logical or arithmetic) expressions over the original ground variables, GP is well suited to performing feature construction by combinatorial optimization.

There is a extensive but diffuse literature on hierarchical learning, especially in areas of biologically inspired computing where it is studied in contexts of: neural modularity and hierarchy; niching, speciation, and demes; and artificial societies. In contrast, the concept of divide-and-conquer algorithms is pervasively and thoroughly studied. This line of research aims toward raising the understanding of layered learning in soft computing to such a level, particularly for evolutionary computation in KD and reinforcement learning over large spatial and temporal databases.

METHODLOGIES

Metric-Based Model Selection in Time Series Learning

For time series, we are interested in actually identifying a stochastic process from the training data (i.e., a process that generates the observations). The performance element, time series classification, will then apply a model of this process to a continuation of the input (i.e., "test" data) to generate predictions. The question addressed in this section is: "To what degree does the training data (or a restriction of that data to a subset of attributes) probabilistically match a prototype of some known stochastic process?" This is the purpose of metric-based model selection: to estimate the degree of match between a subset of the observed data and a known prototype. Prototypes, in this framework, are memory forms (Mozer, 1994), and manifest as embedded patterns generated by the stochastic process that the memory form describes. For example, an exponential trace memory form can express certain types of *MA*(1) processes. The kernel function for this process is given in (Hsu, 1998). The more precisely a time series can be described in terms of exponential processes (wherein future values depend on exponential growth or decay of previous values), the more strongly it will match this memory form. The stronger this match, the better the expected performance of an *MA*(1) learning model, such as an input recurrent (IR) network. Therefore, a metric that measures this degree of match on an arbitrary time series is a useful predictor of IR network performance.

Control of Representation Bias: A Time-Series Learning Example

Table 1 lists five learning representations, each exemplifying a type of *representation* or *restriction bias* for inductive learning from time series, and the metrics corresponding to their strengths. These are referred to as representation metrics because, as documented in the first section (see Figure 1), the choice of representation is local to each node (subnetwork) in the hierarchy, corresponding to a single set within an attribute partition. The choice of hierarchical model is global over the partition and the corresponding metrics are therefore called *representation metrics*. Note that this set may be an abstraction, or "merge", of the lowest-level partition used, and is likely to be a refinement, or "split" of the top-level (unpartitioned) set. The metrics are called *prescriptive* because each one provides evidence in favor of a particular architecture.

(Time Series) Representation Bias	Representation Metric
Simple recurrent network (SRN)	**Exponential trace (AR) score**
Time delay neural network (TDNN)	**Moving average (MA) score**
Gamma network	**Autoregressive moving average (ARMA) score**
Temporal naïve Bayesian network	**Relevance score**
Hidden Markov model (HMM)	**Test set perplexity**

Table 1. Five time series representations and their prescriptive metrics

The design rationale is that each metric is based on an attribute chosen to *correlate positively* (and, to the extent feasible, *uniquely*) with the *characteristic memory form* of a time series. A *memory form* as defined by Mozer (1994) is the representation of some specific temporal pattern, such as a limited-depth buffer, exponential trace, gamma memory (Principé & Lefebvre, 2001), or state transition model.

To model a time series as a stochastic process, one assumes that there is some mechanism that generates a random variable at each point in time. The random variables *X(t)* can be univariate or multivariate (corresponding to single and multiple attributes or *channels* of input per exemplar) and can take discrete or continuous values, and time can be either discrete or continuous. For clarity of exposition, the experiments focus on discrete classification problems with discrete time. The classification model is *generalized linear regression* (Neal, 1996), also known as a *1-of-C coding* (Sarle, 2002) or *local coding* (Kohavi & John, 1997).

Following the parameter estimation literature (Duda *et al*, 2000), time series learning can be defined as finding the parameters $\Theta = \{\theta_1, \ldots, \theta_n\}$ that describe the stochastic mechanism, typically by maximizing the likelihood that a set of realized or *observable* values, $\{x(t_1), x(t_2), \ldots, x(t_k)\}$, were actually generated by that mechanism. This corresponds to the backward, or maximization, step in the *expectation-maximization (EM)* algorithm (Duda *et al*, 2000). Forecasting with time series is accomplished by calculating the conditional density $P(X(t) \mid \{\Theta, \{X(t-1), \ldots, X(t-m)\}\})$, when the stochastic mechanism and the parameters have been identified by the observable values *{x(t)}*. The order *m* of the stochastic mechanism can, in some cases, be infinite; in this case, one can only approximate the conditional density.

Despite recent developments with nonlinear models, some of the most common stochastic models used in time series learning are parametric linear models called *autoregressive (AR)*, *moving average (MA)*, and *autoregressive moving average (ARMA)* processes.

MA or moving average processes are the most straightforward to understand. First, let *{Z(t)}* be some fixed zero-mean, unit-variance "white noise" or "purely random" process (i.e., one for which $Cov[Z(t_i), Z(t_j)] = 1 \ \ iff \ \ t_i = t_j, 0$ otherwise). *X(t)* is an *MA(q)* process, or "moving average process of order *q*", if $X(t) = \sum_{\tau=0}^{q} \beta_\tau Z(t-\tau)$, where the β_τ are constants. It follows that $E[X(t)] = 0$ and $Var[X(t)] = \sum_{\tau=0}^{q} \beta_\tau$. Moving average processes are used to capture "exponential traces" in a time series (Mozer, 1994; Mehrotra *et al*, 1997; Principé & Lefebvre, 2001). For example, input recurrent neural networks (Ray & Hsu, 1998) are a restricted form of nonlinear *MA* process model.

AR or autoregressive processes are processes in which the values at time *t* depend linearly on the values at previous times. With *{Z(t)}* as defined above, *X(t)* is an *AR(p)* process, or "autoregressive process of order *p*", if $\sum_{\upsilon=0}^{p} \alpha_\upsilon X(t-\upsilon) = Z(t)$, where the α_υ are constants. In this case, $E[X(t)] = 0$, but the calculation of $Var[X(t)]$ depends upon the relationship among the α_υ; in general, if $|\alpha_\upsilon| \geq 1$, then *X(t)* will quickly diverge. Autoregressive processes are often used to describe stochastic mechanisms that have a finite, short-term, linear "memory"; they are equivalent to infinite-length *MA* processes constants. Both *Jordan recurrent neural networks* (Mozer, 1994) and *time-delay neural networks* (Lang *et al*, 1990), also known as *tapped delay-line neural networks* or *TDNNs*, are a restricted form of nonlinear *AR* process model (Mehrotra *et al*, 1997, Principé & Lefebvre, 2001).

ARMA is a straightforward combination of *AR* and *MA* processes. With the above definitions, an *ARMA(p, q)* process is a stochastic process *X(t)* in which $\sum_{\upsilon=0}^{p} \alpha_\upsilon X(t-\upsilon) = \sum_{\tau=0}^{q} \beta_\tau Z(t-\tau)$, where the $\{\alpha_\upsilon, \beta_\tau\}$ are constants (Mozer, 1994). Because it can be shown that *AR* and *MA* are of equal expressive power, that is, because they can both represent the same linear stochastic processes, possibly with infinite *p* or *q* (Box *et al*, 1994), *ARMA* model selection and parameter fitting should be done with specific criteria in mind. For example, it is typically appropriate to balance the roles of the *AR(p)* and *MA(q)*, and to limit *p* and *q* to small constant values (typically 4 or 5) for tractability (Box *et al*, 1994; Principé & Lefebvre, 2001). The Gamma memory (Principé & deVries, 1992; Principé & Lefebvre, 2001) is a restricted, nonlinear *ARMA* process model with a neural network architecture and learning rules.

In *heterogeneous* time series, the embedded temporal patterns belong to different categories of statistical models, such as *MA(1)* and *AR(1)*. Examples of such embedded processes are presented in the discussion of the experimental test beds. A multichannel time series learning problem can be decomposed into homegeneous subtasks by aggregation or synthesis of attributes. *Aggregation* occurs in multimodal sensor fusion (e.g., for medical, industrial, and military monitoring), where each group of input attributes represents the bands of information available to a sensor (Stein & Meredith, 1993). In geospatial data mining, these groupings may be topographic. Complex attributes may be *synthesized* explicitly by constructive induction, as in causal discovery of latent (hidden) variables (Heckerman, 1996); or implicitly by preprocessing transforms (Mehrotra *et al*, 1997; Ray & Hsu, 1998; Haykin, 1999).

Control of Preference Bias: A Data Fusion Example

Preference Bias (Combiner Type)	Preference Metric
Specialist-Moderator (SM) Network	**Factorization score**
Multistrategy Hierarchical Mixture of Experts (MS-HME) Network	**Modular mutual information score**

Table 2. Hierarchical committee machines (combiners) and their prescriptive metrics

The learning methods being evaluated define the hierarchical model used to perform multistrategy learning in the integrated, or composite, learning system. Examples of these are listed in Table 2. Continuing research (Hsu, 1998) also considers the training algorithm to use, but is beyond the scope of this chapter. This section presents the metrics for preference bias (the *combiner* type) and presents hierarchical models for classifier fusion in greater detail.

The expected performance of a hierarchical model is a *holistic* measurement; that is, it involves all of the subproblem definitions, the learning architecture used for each one, and even the training algorithm used. It must therefore take into account at least the subproblem definitions. Hence, the metrics used to select a hierarchical model are referred to as *preference metrics*. Preference metrics in this case are designed to evaluate only the subproblem definitions. This criterion has three benefits: first, it is consistent with the holistic function of hierarchical models; second, it is minimally complex, in that it omits less relevant issues such as the learning architecture for each subproblem from consideration; and third, it measures the quality of an attribute partition. The third property is very useful in heuristic search over attribute partitions: the tree metric can thus serve double duty as an evaluation function for a partition (given a hierarchical model to be used) and for mixture model (given a partitioned data set). As a convention, the choice of *partition* is committed first; next, the hierarchical model type; then, the learning architectures for each subset, with each selection being made subject to the previous choices.

The preference metric for specialist-moderator networks is the *factorization score*. The interested reader is referred to (Hsu, 1998; Hsu *et al*, 2000).

Multistrategy Hierarchical Mixture of Experts (MS-HME) Network

The tree metric for HME-type networks is the *modular mutual information score*. This score measures mutual information across subsets of a partition[2]. It is directly proportional to the conditional mutual information of the desired output given each subset *by itself* (i.e., the mutual information between one subset and the target class, *given* all other subsets). It is inversely proportional to the difference between joint and total conditional mutual information (i.e., shared information among all subsets). Let the first quantity be denoted I_i for each subset $\boldsymbol{a}_i$, and the second quantity as I_∇ for an entire partition.

The mutual information between discrete random variables X and Y is defined (Cover & Thomas, 1991) as the Kullback-Leibler distance between joint and product distributions:

$$\begin{aligned} I(X;Y) &=_{def} D\big(p(x,y) \,\|\, p(x)p(y)\big) \\ &= H(X) - H(X \mid Y) \\ &= H(X) + H(Y) - H(X,Y) && \text{(chain rule)} \\ &= H(Y) - H(Y \mid X) \end{aligned}$$

The conditional mutual information of X and Y given Z is defined (Cover & Thomas, 1991) as the change in conditional entropy when the value of Z is known:

$$\begin{aligned} I(X;Y|Z) &=_{def} H(X \mid Z) - H(X \mid Y,Z) \\ &= H(Y \mid Z) - H(Y \mid X,Z) \end{aligned}$$

The *common information* of X, Y, and Z (the analogue of k-way intersection in set theory, except that it can have negative value) can now be defined:

$$\begin{aligned} I(X;Y;Z) &=_{def} I(X;Y) - I(X;Y \mid Z) \\ &= I(X;Z) - I(X;Z \mid Y) \\ &= I(Y;Z) - I(Y;Z \mid Y) \\ &= I(X;Z) - I(X;Z \mid Y) \\ &= I(Y;Z) - I(Y;Z \mid Y) \end{aligned}$$

The idea behind the modular mutual information score is that it should reward high conditional mutual information between an attribute subset and the desired output given other subsets (i.e., *each expert subnetwork will be alloted a large share of the work*). It should also penalize high common information (i.e., the gating network is alloted more work relative to the experts). Given these dicta, we can define the modular mutual information for a partition as follows:

$$
\begin{aligned}
I(\mathbf{X};\mathbf{Y}) &=_{def} D\left(p(x_1, x_2, \ldots, x_n, y) \,\|\, p(x_1)p(x_2)\ldots p(x_n)p(y)\right) \\
\mathbf{X} &= \{\mathbf{X}_1, \ldots, \mathbf{X}_k\} \\
\bigcap_{i=1}^{k} \mathbf{X}_i &= \varnothing \\
\bigcup_{i=1}^{k} \mathbf{X}_i &= \{X_1, X_2, \ldots, X_n\}
\end{aligned}
$$

which leads to the definition of I_i (modular mutual information) and I_∇ (modular common information):

$$
\begin{aligned}
I_i &=_{def} I(\mathbf{X}_i;\mathbf{Y} \mid \mathbf{X}_{\neq i}) \\
&=_{def} H(\mathbf{X};\mathbf{Y}) - H(\mathbf{X}_i \mid \mathbf{Y}, \mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_k) \\
I_\nabla &=_{def} I(\mathbf{X}_1;\mathbf{X}_2;\ldots;\mathbf{X}_k;\mathbf{Y}) \\
&=_{def} I(\mathbf{X};\mathbf{Y}) - \sum_{i=1}^{k} I_i
\end{aligned}
$$

Because the desired metric rewards high I_i and penalizes high I_∇, we can define:

$$
\begin{aligned}
M_{MSHME} &= \left[\sum_{i=1}^{k} I_i\right] - I_\nabla \\
&= \left[\sum_{i=1}^{k} I_i\right] - I(\mathbf{X};\mathbf{Y}) - \sum_{i=1}^{k} I_i \\
&= 2\left[\sum_{i=1}^{k} I_i\right] - I(\mathbf{X};\mathbf{Y})
\end{aligned}
$$

Model Selection and Composite Learning

As explained in the introduction, being able to decompose a learning problem into simpler subproblems still leaves the task of mapping each to its appropriate model – the hypothesis language or *representation bias* (Mitchell, 1997; Witten and Frank, 2000). In the above methodology section, we have just formulated a rationale for using quantitative metrics to accumulate evidence in favor of particular models. This leads to the design presented here, a metric-based selection system for *time series learning architectures* and *general learning methods*. Next, we have studied specific time series learning architectures that populate part of a collection of model components, along with the metrics that correspond to each. We then addressed the problem of determining a preference bias (data fusion algorithm) for multistrategy learning by examining two hierarchical mixture models to see how they can be converted into classifier fusion models that also populate this collection. Finally, we surveyed metrics that correspond to each.

I pause to justify this coarse-grained approach to model selection. As earlier defined, *model selection* is the problem of choosing a hypothesis class that has the appropriate complexity for the given training data (Stone, 1977; Hjorth, 1994; Schuurmans, 1997). Quantitative, or *metric-based*, methods for model selection have previously been used to learn using highly flexible models with many degrees of freedom (Schuurmans, 1997), but with no particular assumptions on the structure of decision surfaces, e.g., that they are linear or quadratic (Geman *et al*, 1992). Learning without this characterization is known in the statistics literature as *model-free estimation* or *nonparametric statistical inference*. A premise of this chapter is that, for learning from heterogeneous time series, indiscriminate use of such models is too unmanageable. This is especially true in diagnostic monitoring applications such as crisis monitoring, because decision surfaces are more sensitive to error when the target concept is a catastrophic event (Hsu *et al*, 1998).

The purpose of using model selection in *decomposable* learning problems is to *fit* a suitable hypothesis language (model) to each subproblem. (Engels *et al*, 1998) A subproblem is defined in terms of a subset of the input and an intermediate concept, formed by unsupervised learning from that subset. Selecting a model entails three tasks. The first is *finding partitions* that are consistent enough to admit at most one “suitable” model per subset. The second is *building a collection of models* that is flexible enough so that some partition can have at least one model matched to each of its subsets. The third is to *derive a principled quantitative system for model evaluation* so that exactly one model can be correctly chosen per subset of the acceptable partition or partitions. These tasks indicate that a model selection system *at the level of subproblem definition* is desirable, because this corresponds to the granularity of problem decomposition, the design choices for the collection of models, and the evaluation function. This is a more comprehensive optimization problem than traditional model selection typically adopts (Geman *et al*, 1992; Hjorth, 1994), but it is also approached from a less precise perspective; hence the term *coarse-grained*.

RESULTS

Synthetic and Small-Scale Data Mining Problems

This section presents experimental results with comparisons to existing inductive learning systems (Kohavi *et al*, 1996): decision trees, traditional regression-based methods as adapted to time series prediction, and non-modular probabilistic networks (both atemporal and ARIMA-type ANNs).

The Modular Parity Problem

Figure 3 shows the classification accuracy in percent for specialist and moderator output for the concept:

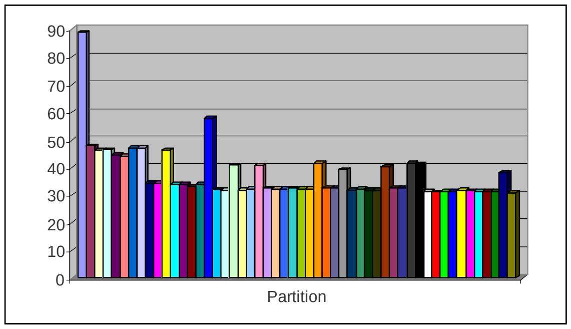
$$\begin{aligned}\mathbf{Y} &= \prod_{i=1}^{k} Y_i \\ &= Y_1 \times Y_2 \times \ldots \times Y_k \\ Y_i &= X_{i1} \oplus X_{i2} \oplus \ldots \oplus X_{in_i} \\ X_{ij} &\in \mathbf{H} \equiv \{0, 1\}\end{aligned}$$

All mixture models are trained using 24 hidden units, distributed across all specialists and moderators. When used as a heuristic evaluation function for partition search, the HME metric documented in the previous section finds the best partition for the 5-attribute problem (shown below) as well as 6, 7, and 8, with no backtracking, and indicates that an MS-HME model should be used.

This section documents improvements in classification accuracy as achieved by attribute partitioning. Figure 3 shows how the optimum partition {{1,2,3}{4,5}} for the concept:

$$parity(x_1, x_2, x_3) \times parity(x_4, x_5)$$

achieves the best specialist performance for any size-2 partition.



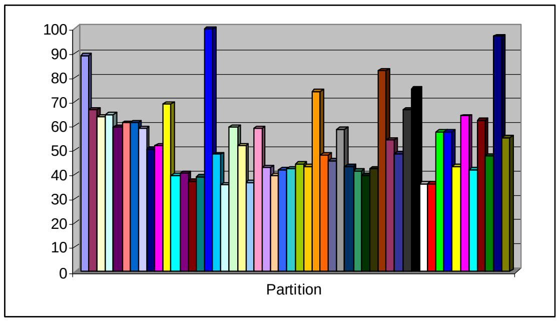


Figure 3. Mean Classification Accuracy of Specialists vs. Moderators for all (52) Partitions of 5-Attribute Modular Parity Problem

Figure 3 shows how this allows it to achieve the best moderator performance overall. Empirically, "good splits" – especially descendants and ancestors of the optimal one, i.e., members of its schema (Goldberg, 1989) – tend to perform well.

As documented in the background section, partition search is able to find Partition #16, {{1,2,3}{4,5}} (the optimum partition) after expanding all of the 2-subset partitions. This reduces B_n evaluations to $\Theta(2^n)$; attribute partitioning therefore remains an intractable problem, but is more feasible for small to moderate numbers of attributes (30-40 can be handled by high-performance computers, instead of 15-20 using exhaustive search). Approximation algorithms for polynomial-time evaluation (Cormen *et al*, 2001) are currently being investigated by the author.

For experiments using specialist-moderator networks on a musical tune classification problem – synthetic data quantized from real-world audio recordings – the interested reader is referred to (Hsu *et al*, 2000).

Application: Crop Condition Monitoring

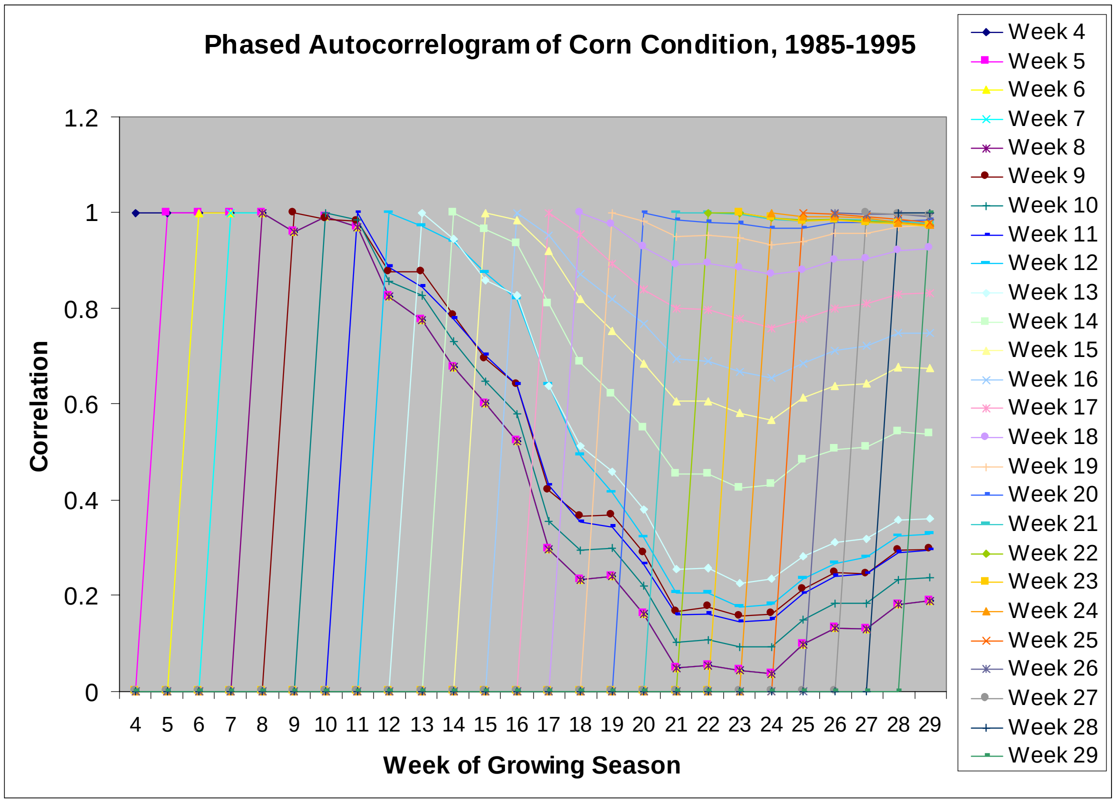


Figure 4. Phased autocorrelogram (plot of autocorrelation shifted over time) for crop condition (average quantized estimates)

Figure 4 visualizes a heterogeneous time series. The lines shown are phased *autocorrelograms*, or plots of autocorrelation shifted in time, for (subjective) weekly *crop condition* estimates, averaged from 1985-1995 for the state of Illinois. Each *point* represents the correlation between one week's mean estimate and the mean estimate for a subsequent week. Each *line* contains the correlation between values for a particular week and all subsequent weeks. The data is heterogeneous because it contains both an autoregressive pattern (the linear increments in autocorrelation for the first 10 weeks) and a moving average pattern (the larger, unevenly spaced increments from 0.4 to about 0.95 in the rightmost column). The autoregressive process, which can be represented by a time-delay model, expresses weather "memory" (correlating early and late drought); the moving average process, which can be represented by an exponential trace model, physiological damage from drought. Task decomposition can improve performance here, by isolating the AR and MA components for identification and application of the correct specialized architecture – a time delay neural network (Lang *et al*, 1990; Haykin, 1999) or simple recurrent network (Principé & Lefebvre, 2001), respectively.

We applied a simple mixture model to reduce variance in ANN-based classifiers. A paired *t*-test with 10 degrees of freedom (for 11-*year* cross-validation over the weekly predictions) indicates significance at the level of $p < 0.004$ for the moderator versus TDNN and at the level of $p < 0.0002$ for the moderator versus IR. The null hypothesis is rejected at the 95% level of confidence for TDNN outperforming IR ($p < 0.09$), which is consistent with the hypothesis that an MS-HME network yields a performance boost over either network type alone. This result, however, is based on relatively few samples (in terms of weeks per year) and very coarse spatial granularity (statewide averages).

	Classification Accuracy, Crop Condition Monitoring (%)							
	Training				Cross Validation			
Inducer	**Min**	**Mean**	**Max**	**StdDev**	**Min**	**Mean**	**Max**	**StdDev**
ID3	100.0	100.0	100.0	0.00	33.3	55.6	82.4	17.51
ID3, bagged	99.7	99.9	100.0	0.15	30.3	58.2	88.2	18.30
ID3, boosted	100.0	100.0	100.0	0.00	33.3	55.6	82.4	17.51
C5.0	90.7	91.7	93.2	0.75	38.7	58.7	81.8	14.30
C5.0, boosted	98.8	99.7	100.0	0.40	38.7	60.9	79.4	13.06
IBL	93.4	94.7	96.7	0.80	33.3	59.2	73.5	11.91
Discrete Naïve-Bayes	74.0	77.4	81.8	2.16	38.7	68.4	96.7	22.85
DNB, bagged	73.4	76.8	80.9	2.35	38.7	70.8	93.9	19.63
DNB, boosted	76.7	78.7	81.5	1.83	38.7	69.7	96.7	21.92
PEBLS	91.6	94.2	96.4	1.68	27.3	58.1	76.5	14.24
IR Expert	*91.0*	*93.7*	*97.2*	*1.67*	*41.9*	*72.8*	*94.1*	*20.45*
TDNN Expert	*91.9*	*96.8*	*99.7*	*2.02*	*48.4*	*74.8*	*93.8*	*14.40*
Pseudo-HME	*98.2*	*98.9*	*100.0*	*0.54*	*52.9*	*79.0*	*96.9*	*14.99*

Table 3. Performance of a HME-type mixture model compared with compared with that of other inducers on the crop condition monitoring problem

Table 3 summarizes the performance of an MS-HME network versus that of other induction algorithms from *MLC++* (Kohavi *et al*, 1996) on the crop condition monitoring problem. This experiment illustrates the usefulness of learning task decomposition over heterogeneous time series. The improved learning results due to application of multiple models (TDNN and IR specialists) and a mixture model (the Gamma network moderator). Reports from the literature on common statistical models for time series (Box *et al*, 1994; Gershenfeld & Weigend, 1994; Neal, 1996) and experience with the (highly heterogeneous) test bed domains documented here bears out the idea that “fitting the right tool to each job” is critical.

Application: Loss Ratio Prediction in Automobile Insurance Pricing

Table 4 summarizes the performance of the *ID3* decision tree induction algorithm and the state-space search-based feature subset selection (FSS) wrapper in *MLC++* (Kohavi *et al*, 1996) compared to that of a *genetic wrapper* for feature selection. This system is documented in detail in (Hsu *et al*, 2002). We used a version of *ALLVAR-2*, a data set for decision support in automobile insurance policy pricing. This data set was used for clustering and classification and initially contained 471-attribute record for each of over 300000 automobile insurance policies, with 5 bins of *loss ratio* as a prediction target. Wall clock time for the *Jenesis* and *FSS-ID3* wrappers was comparable. As the table shows, both the *Jenesis* wrapper and the *MLC++* wrapper (using *ID3* as the wrapped inducer) produce significant improvements over unwrapped *ID3* in classification accuracy and very large reductions in the number of attributes used. The test set accuracy, and the number of selected attributes, are averaged over 5 cross validation folds (70 aggregate test cases each). Results for data sets from the Irvine database repository that are known to contain irrelevant attributes are also positive.able 10 presents more descriptive statistics on the 5-way cross-validated performance of ID3, FSS-ID3 (the *MLC++* implementation of *ID3* with its feature subset selection wrapper), and *Jenesis*. Severe overfitting is quite evident for *ID3*, based on the difference between training and test set error (perfect purity is achieved in all 5 folds) and the larger number of attributes actually used compared to the wrappers. *Jenesis* and *FSS-ID3* perform comparably in terms of test set error, though *FSS-ID3* has less difference between training and test set error and *Jenesis* is less likely to overprune the attribute subset. Note that *FSS-ID3* consistently selects the fewest attributes, but still overfits (*Jenesis* achieves lower test set error in 3 of 5 cross validation cases). The test set errors of *Jenesis* and *FSS-ID3* are not significantly different, so generalization quality is not conclusively distinguishable in this case. We note, however, that excessively shrinking the subset indicates a significant tradeoff regarding generalization quality. The classification model was used to audit an existing rule-based classification system over the same instance space, and to calibrate an underwriting model (to guide pricing decisions for policies) for an experimental market.

		Cross Validation Segment						
		0	**1**	**2**	**3**	**4**	**Mean**	**Stdev**
Training Set Accuracy (%)	**ID3**	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	*100.0*	0.00
	FSS-ID3	55.00	54.29	67.86	50.36	60.71	57.64	6.08
	Jenesis	65.71	67.14	71.43	71.43	55.71	66.29	5.76
Test Set Accuracy (%)	**ID3**	41.43	*42.86*	28.57	41.43	44.29	39.71	5.67
	FSS-ID3	*48.57*	35.71	*34.29*	47.14	54.29	44.00	7.74
	Jenesis	41.43	*42.86*	31.43	*52.86*	*55.71*	***44.86***	8.69
Attributes Selected	**ID3**	35	35	37	40	35	36.40	1.96
	FSS-ID3	7	8	7	13	18	10.60	4.32
	Jenesis	20	19	22	20	23	20.80	1.47

Table 4. Results from *Jenesis* for One Company (5-way cross validation), representative data sets

We have observed that the aggregation method scales well across lines of business (the indemnity and non-indemnity companies) and states. This was demonstrated using many of our decision tree experiments and visualizations using *ALLVAR-2* samples and subsamples by state.

ACKNOWLEDGEMENTS

Support for this research was provided in part by the Army Research Lab under grant ARL-PET-IMT-KSU-07, by the Office of Naval Research under grant N00014-01-1-0519, and by the Naval Research Laboratory under grant N00014-97-C-2061. The authors thank Nathan D. Gettings for helpful discussions on data fusion and time series analysis and the anonymous reviewer for comments on background material. Thanks also to David Clutter, Matt A. Durst, Nathan D. Gettings, James A. Louis, Yuching Ni, Yu Pan, Mike Perry, James W. Plummer, Victoria E. Lease, Tom Redman, Cecil P. Schmidt, and Kris Wuollett for implementations of software components of the system described in this chapter.

References

Benjamin, D. P. editor (1990). *Change of Representation and Inductive Bias*. Norwell, MA: Kluwer Academic Publishers.

Bogart, K. P. (1990). *Introductory Combinatorics, 2nd Edition*. Orlando, FL: Harcourt.

Booker, L. B., Goldberg, D. E., & Holland, J. H. (1989). Classifier Systems and Genetic Algorithms. *Artificial Intelligence*, *40*, 235-282.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time Series Analysis, Forecasting, and Control (3rd edition)*. San Francisco, CA: Holden-Day.

Breiman, L. (1996) Bagging Predictors. *Machine Learning*, *24*, 123-140.

Brooks, F. P. (1995). *The Mythical-Man Month, Anniversary Edition: Essays on Software Engineering*. Reading, MA: Addison-Wesley.

Cantu-Paz, E. (1999). *Designing Efficient and Accurate Parallel Genetic Algorithms*. Ph.D. thesis, University of Illinois at Urbana-Champaign. Technical report, Illinois Genetic Algorithms Laboratory (IlliGAL).

Cover, T. M. & Thomas, J. A. (1991). *Elements of Information Theory*. New York, NY: John Wiley and Sons.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stern, C. (2001). *Introduction to Algorithms, Second Edition*. Cambridge, MA: MIT Press.

DeJong, K. A., Spears, W. M., & Gordon, D. F. (1993). Using genetic algorithms for concept learning. *Machine Learning*, *13*, 161-188.

Donoho, S. K. (1996). *Knowledge-Guided Constructive Induction*. Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.

Duda, R. O., Hart, P. E., & Stork, D. (2000). *Pattern Classification, Second Edition*. New York, NY: John Wiley and Sons.

Engels, R., Verdenius, F., & Aha, D. (1998). *Proceedings of the 1998 Joint AAAI-ICML Workshop on the Methodology of Applying Machine Learning (Technical Report WS-98-16)*, AAAI Press, Menlo Park, CA.

Freund, T. & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of ICML-96*.

Fu, L.-M. & Buchanan, B. G. (1985). Learning Intermediate Concepts in Constructing a Hierarchical Knowledge Base. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 659-666, Los Angeles, CA.

Gaba, D. M., Fish, K. J., & Howard, S. K. (1994). *Crisis Management in Anesthesiology*. New York, NY: Churchill Livingstone.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural Networks and the Bias/Variance Dilemna. *Neural Computation*, *4*, 1-58.

Gershenfeld, N. A. & Weigend, A. S., editors. (1994). The Future of Time Series: Learning and Understanding. In *Time Series Prediction: Forecasting the Future and Understanding the Past (Santa Fe Institute Studies in the Sciences of Complexity XV)*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Goldberg, D. E. (1998). *The Race, The Hurdle, and The Sweet Spot: Lessons from Genetic Algorithms for the Automation of Design Innovation and Creativity*. Technical report, Illinois Genetic Algorithms Laboratory (IlliGAL).

Grois, E., Hsu, W. H., Voloshin, M., & Wilkins, D. C. (1998). Bayesian Network Models for Generation of Crisis Management Training Scenarios. In *Proceedings of IAAI-98*. Menlo Park, CA: AAAI Press.

Harik, G. & Lobo, F. (1997). *A parameter-less genetic algorithm*. Technical report, Illinois Genetic Algorithms Laboratory (IlliGAL).

Hassoun, M. H. (1995). *Fundamentals of Artificial Neural Networks*. Cambridge, MA: MIT Press.

Hayes-Roth, B., Larsson, J. E., Brownston, L., Gaba, D., & Flanagan, B. (1996). *Guardian Project Home Page*, URL: http://www-ksl.stanford.edu/projects/guardian/

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation, Second Edition*. Englewood Cliffs, NJ: Prentice Hall.

Heckerman, D. A. (1991). *Probabilistic Similarity Networks*. Cambridge, MA: MIT Press.

Heckerman, D. A. (1996). *A Tutorial on Learning With Bayesian Networks*. Microsoft Research Technical Report 95-06, revised June 1996.

Hjorth, J. S. U. (1994). *Computer Intensive Statistical Methods: Validation, Model Selection and Bootstrap*. London, UK: Chapman and Hall.

Horn, J. (1997). *The Nature of Niching: Genetic Algorithms and The Evolution of Optimal, Cooperative Populations*. Ph.D. thesis, University of Illinois at Urbana-Champaign. Technical report, Illinois Genetic Algorithms Laboratory (IlliGAL).

Horvitz, E. & Barry, M. (1995). Display of Information for Time-Critical Decision Making. In *Proceedings of the Eleventh International Conference on Uncertainty in Artificial Intelligence (UAI-95)*. San Mateo, CA: Morgan-Kaufmann.

Hsu, W. H. (1998). *Time Series Learning With Probabilistic Network Composites*. Ph.D. thesis, University of Illinois at Urbana-Champaign. Technical Report UIUC-DCS-R2063. URL: http://www.kddresearch.org/Publications/Theses/PhD/Hsu, 1998.

Hsu, W. H., Gettings, N. D., Lease, V. E., Pan, Y., & Wilkins, D. C. (1998). A New Approach to Multistrategy Learning from Heterogeneous Time Series. In *Proceedings of the International Workshop on Multistrategy Learning*.

Hsu, W. H., Ray, S. R., & Wilkins, D. C. (2000). A Multistrategy Approach to Classifier Learning from Time Series. *Machine Learning*, *38*, 213-236.

Hsu, W. H., Welge, M., Redman, T., & Clutter, D. (2002). Constructive Induction Wrappers in High-Performance Commercial Data Mining and Decision Support Systems. *Data Mining and Knowledge Discovery*, to appear. Preprint URL: http://www.kddresearch.org/Publications/Journal/KDDM-D2K-07052000.zip

Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, *6*, 181-214.

Kohavi, R. & John, G. H. (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence, Special Issue on Relevance, 97(1-2)*, 273-324.

Kohavi, R., Sommerfield, D., & Dougherty, J. Data Mining Using *MLC++*: A Machine Learning Library in C++. In *Tools with Artificial Intelligence*, p. 234-245, IEEE Computer Society Press, Rockville, MD, 1996. URL: http://www.sgi.com/Technology/mlc.

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, *78*:1464-1480.

Koza, J. R. (1992). *Genetic Programming*. Cambridge, MA: MIT Press.

Lang, K. J., Waibel, A. H., & Hinton, G. E. (1990). A Time-Delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks*, *3*, 23-43.

Lee, K.-F. (1989). *Automatic Speech Recognition: The Development of the SPHINX System*. Norwell, MA: Kluwer Academic Publishers.

Li, T., Fang. L., & Li, K. Q-Q. (1993). Hierarchical Classification and Vector Quantization With Neural Trees. *Neurocomputing*, *5*, 119-139.

Lowe, D. (1995). Radial Basis Function Networks. In Arbib, M. A., editor,*The Handbook of Brain Theory and Neural Networks*, 779-782.

Mehrotra, K., Mohan, C. K., & Ranka, S. (1997). *Elements of Artificial Neural Networks*. Cambridge, MA: MIT Press.

Michalski, R. S. (1993). A Theory and Methodology of Inductive Learning. *Artificial Intelligence*, *20(2)*, 111-161, reprinted in Buchanan, B. G. & Wilkins, D. C., editors, *Readings in Knowledge Acquisition and Learning*,. San Mateo, CA: Morgan-Kaufmann.

Michalski, R. S., and Stepp, R. E. (1983). Learning from observation: Conceptual clustering. In Michalski, R. S.; Carbonell, J. G.; and Mitchell, T. M., eds., *Machine Learning: An Artificial Intelligence Approach*. San Mateo, CA: Morgan Kaufmann.

Mitchell, T. M. (1997). *Machine Learning*. New York, NY: McGraw-Hill.

Mozer, M. C. (1994). Neural Net Architectures for Temporal Sequence Processing. In Weigend, A. S. & Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past (Santa Fe Institute Studies in the Sciences of Complexity XV)*. Reading, MA: Addison-Wesley.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. New York, NY: Springer-Verlag.

Palmer, W. C. (1965). *Meteorological Drought*. Research Paper Number 45, Office of Climatology, United States Weather Bureau.

Principé, J. & deVries. (1992). The Gamma Model – A New Neural Net Model for Temporal Processing. *Neural Networks*, *5*, 565-576.

Principé, J. & Lefebvre, C. (2001). *NeuroSolutions v4.0*, Gainesville, FL: NeuroDimension. URL: http://www.nd.com.

Ray, S. R. & Hsu, W. H. (1998). Self-Organized-Expert Modular Network for Classification of Spatiotemporal Sequences. *Journal of Intelligent Data Analysis*, *2(4)*.

Resnick, P. & Varian, H. R. (1997). Recommender Systems. *Communications of the ACM*, *40(3)*:56-58.

Russell, S. & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: Prentice Hall.

Sarle, W. S., editor. (2002). *Neural Network FAQ*, periodic posting to the Usenet newsgroup *comp.ai.neural-nets*, URL: ftp://ftp.sas.com/pub/neural/FAQ.html

Schuurmans, D. (1997). A New Metric-Based Approach to Model Selection. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, Providence, RI, 552-558. Menlo Park, CA: AAAI Press.

Stein, B. & Meredith, M. A. (1993). *The Merging of the Senses*. Cambridge, MA: MIT Press.

Stone, M. (1997). An Asymptotic Equivalence of Choice of Models by Cross-Validation and Akaike's Criterion. *Journal of the Royal Statistical Society Series B*, *39*, 44-47.

Watanabe, S. (1985). *Pattern Recognition: Human and Mechanical*. New York, NY: John Wiley and Sons.

Witten, I. H. and Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Mateo, CA: Morgan-Kaufmann.

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks*, *5*, 241-259.

[1] *S* is a recurrence known as the Stirling Triangle of the Second Kind. It counts the number of partitions of an *n*-set into *k* classes (Bogart, 1990).

[2] This idea is based upon suggestions by Michael I. Jordan.