Machine Learning, 38, 213–236, 2000.

A Multistrategy Approach to Classifier Learning from Time Series

WILLIAM H. HSU[*] bhsu@cis.ksu.edu
National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA

SYLVIAN R. RAY ray@cs.uiuc.edu
Department of Computer Science and Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

DAVID C. WILKINS dcw@uiuc.edu
Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Editors: Floriana Esposito, Ryszard Michalski, & Lorenza Saitta

Abstract. We present an approach to inductive concept learning using multiple models for time series. Our objective is to improve the efficiency and accuracy of concept learning by decomposing learning tasks that admit multiple types of learning architectures and mixture estimation methods. The decomposition method adapts attribute subset selection and constructive induction (cluster definition) to define new subproblems. To these problem definitions, we can apply metric-based model selection to select from a database of learning components, thereby producing a specification for supervised learning using a mixture model. We report positive learning results using temporal artificial neural networks (ANNs), on a synthetic, multiattribute learning problem and on a real-world time series monitoring application.

Keywords: multistrategy learning, time series, attribute partitioning, constructive induction, metric-based model selection, mixture estimation

1. Introduction

This paper discusses inductive concept learning from time series data. It presents a new approach that adapts attribute subset selection and constructive induction—especially *cluster definition* (Michalski, 1983; Stepp & Michalski, 1986; Donoho, 1996)—to decompose problems, then uses quantitative metrics to select techniques for each identifiable (and relevant) embedded subproblem. This approach is best suited for *heterogeneous* time series data—that arising from multiple sources of data (such as in sensor fusion or multimodal human-computer interaction). The multistrategy solution is compared to some hierarchical mixture models that recombinine specialized classifiers, for large-scale data sets. Experimental evaluation uses real and synthetic data that captures heterogeneity in time series and in general.

[*] Department of Computing and Information Sciences, Kansas State University, Manhattan, KS 66506.

The purpose of applying integrative, multistrategy learning to such data is to improve the accuracy and efficiency of classifier learning using a mixture model, through systematic transformation of learning tasks into a collection of subtasks. Problems that admit this transformation are referred to in this paper as *decomposable*, by means of task partitioning and subproblem definition, quantitative model selection, and construction of hierarchical mixture models for data fusion. Decomposition of time series learning tasks alleviates some aspects of heterogeneity, such as having multimodal inputs and diversity in scale and structure, that arise in monitoring problems. Equally important, it supports selection of the most appropriate learning architecture (Benjamin, 1990; Engels, Verdenius, & Aha, 1998) for each homogeneous component of a time series, and accounts for prior knowledge on subdivision of learning tasks.

The key novel contributions of the system are:

1. The explicit organization of learning components into recombinable and reusable classes
2. Metrics for properties of data sets that indicate an appropriate learning technique
3. A framework for decomposing learning tasks and combining classifiers learned using different techniques

A typical application for such a system is learning for crisis *monitoring*, the prediction and classification of anomalous, potentially catastrophic, conditions. This form of pattern recognition is useful in decision support or *recommender* (Resnick & Varian, 1997) systems 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 (Wilkins & Sniezek, 1997; Hsu et al., 1998), online medical diagnosis (Hayes-Roth et al., 1996), simulation-based training and critiquing for crisis management (Mengshoel & Wilkins, 1996; Grois et al., 1998), and intelligent data visualization (Horvitz & Barry, 1995).

2. Background

This section surveys background material on time series learning using stochastic process models, particularly temporal artificial neural networks (ANNs). It presents the concepts of *memory forms*, *convolutional codes*, the *autoregressive moving average (ARMA)* family of processes, and the linear models (corresponding to time-delay, recurrent, and gamma memories) that can be used to represent ARMA processes. It then describes a framework for integrated, multistrategy learning of time series that adapts one of several constructive induction techniques to decompose a learning task, derives a learning specification by selecting among supervised inductive learning architectures and algorithms, and synthesizes the resultant predictions using a mixture model.

2.1. Heterogeneous time series: learning techniques

A key assumption made in this paper is that predictive capability is a good indicator of performance (classification accuracy) for a time series learning architecture, such as a recurrent ANN. Although the merit of this assumption varies among time series classification problems (Gershenfeld & Weigend, 1994; Mozer, 1994), the authors have found it to be reliable for a variety of problems studied. The design rationale that follows from this assumption defines metrics for evaluating problem definitions. Each metric estimates an intrinsic statistical property: namely, how closely a particular type of stochastic process fits (i.e., can generate) observed data. Our objective is to identify the predominant *process type* to select an appropriate learning architecture. The *memory form*, as defined by Mozer (1994), is a property of a time series learning architecture that characterizes how it represents a temporal sequence. Memory forms include limited-depth buffers, exponential traces, gamma memories (Principé & deVries, 1992; Principé & Lefebvre, 1998), and state transition models. In the ideal case, learning subtasks can be isolated that each exhibit exactly one process type (i.e., each is *homogeneous*), and these can be matched to known memory forms in the system's catalogue.

For temporal ANNs, a memory form can be represented using a functional descriptor called a *convolutional code*. 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 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, 1998). The interested reader is referred to Mozer (1994) and Hsu (1998) for definitions of the convolutional codes for temporal ANNs discussed in this paper. These are tapped delay-line memories, also called time-delay neural networks, or TDNNs (Lang, Waibel, & Hinton, 1990); exponential trace memories, also called input recurrent networks (Ray & Hsu, 1998); and gamma memories (Principé & deVries, 1992; Principé & Lefebvre, 1998). The latter express both resolution and depth, at a cost of more degrees of freedom, convergence time, and kernel function complexity.

To evaluate the degree to which a time series exhibits known memory forms, the convolutional code for each one is applied to the time series data, and the transformed data sets are compared to choose the most effective one. The criterion for this metric-based model selection step is the change in *conditional entropy* (Cover & Thomas, 1991), with respect to each convolutional code, 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 mode being evaluated.

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. Following the parameter estimation literature (Duda & Hart, 1973), 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 (Dempster, Laird, & Rubin, 1977). 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 (Kantz & Schreiber, 1997), some of the most common stochastic models used in time series learning are parametric, linear models for generating processes called *autoregressive (AR)*, *moving average (MA)*, and *autoregressive moving average (ARMA)* processes. We refer the reader to Hsu et al. (1998) and Hsu (1998) for the mathematical definition of these models. In *heterogeneous* time series, the embedded temporal patterns belong to different categories of statistical models, such as $MA(q)$ and $AR(p)$, where q and p denote the polynomial order of the process. 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 homogeneous 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). Complex attributes may be *synthesized* explicitly by constructive induction, as in causal discovery of latent (hidden) variables (He96); or implicitly by preprocessing transforms (Haykin, 1994; Mozer, 1994; Ray & Hsu, 1998).

For a stochastic process (time-indexed sequence of random variables) $X(t)$, we are interested in the conditional entropy of the next value given earlier ones. This can be written as: $H_d =_{\text{def}} H(X(t) \mid X_1(t), \ldots, X_d(t))$. To measure the improvement due to convolution with a kernel function with d components, we further define: $\hat{H}_d =_{\text{def}} H(X(t) \mid \hat{X}_i(t), 1 \le i \le d)$ where $\hat{X}_i(t)$ is as defined above. Similarly, we define H_d^s and $\hat{H}_d^s$, for the restriction $X^s(t)$ of $X(t)$ to the *subset* of attributes s. Our refinement permits specific subsets of input data to be evaluated to determine the predominant process type. Given a kernel function for a candidate learning architecture, we define a metric: $M_R = H_d^s / \hat{H}_d^s$ for a recurrent ANN of type $R \in \{\mathit{TDNN}, \mathit{SRN}, \mathit{GAMMA}\}$, which denotes the degree of match between a known memory form and observed time series data.

2.2. Integrated multistrategy learning systems

Figure 1 depicts a learning system for decomposable, multi-attribute data sets. The central elements of this system are: *attribute partitioning, metric-based model selection*, and a *data fusion* mechanism for integration of multiple models. Given a specification for reformulated (reduced or partitioned) input, new intermediate concepts $\vec{y}_i'$ can be formed by unsupervised learning—e.g., conceptual clustering cf. Stepp and Michalski (1986); the newly defined problem or problems can then be mapped to one or more appropriate hypothesis languages (model specifications). The next section presents **Select-Net**, a high-level algorithm for generating this specification, which we shall refer to as a *composite*. This algorithm also configures and trains subnetworks in a hierarchical system for multistrategy learning, whose components are selected by **Select-Net**; a data fusion step occurs after individual training of each model. The system incorporates attribute partitioning into constructive induction to

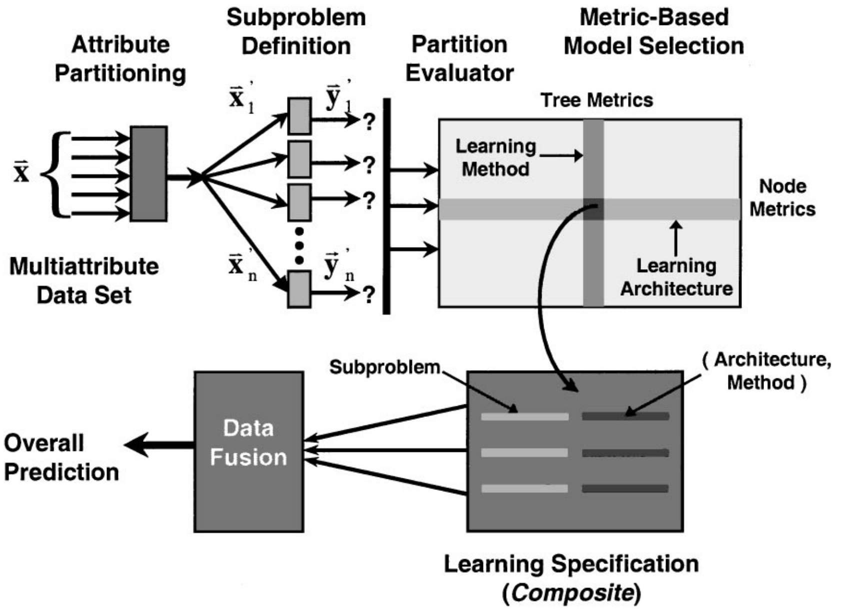

Figure 1. Overview of the integrated, multistrategy learning system.

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.

3. Learning task decomposition and model selection

This section introduces *attribute partitioning* for problem decomposition in multiattribute inductive learning and a new metric-based model selection approach (composite learning) for decomposable learning tasks.

3.1. 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).

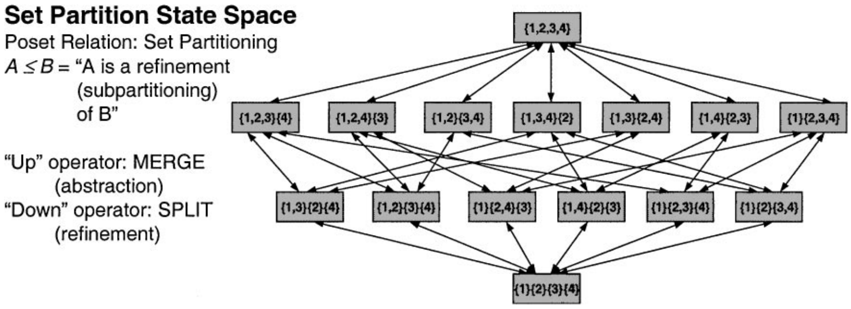
Set Partition State Space
Poset Relation: Set Partitioning
$A \leq B$ = "A is a refinement (subpartitioning) of B"
"Up" operator: MERGE (abstraction)
"Down" operator: SPLIT (refinement)
{1,2,3,4}
{1,2,3}{4}
{1,2,4}{3}
{1,2}{3,4}
{1,3,4}{2}
{1,3}{2,4}
{1,4}{2,3}
{1}{2,3,4}
{1,3}{2}{4}
{1,2}{3}{4}
{1}{2,4}{3}
{1,4}{2}{3}
{1}{2,3}{4}
{1}{2}{3,4}
{1}{2}{3}{4}

Figure 2. State space formulation of the attribute partitioning problem.

Attribute subset selection is the task of focusing a learning algorithm's attention on some subset of the given input attributes, while ignoring the rest (Kira & Rendell, 1992; Kohavi & John, 1997). In this research, it 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. Whereas subset selection presumes a single learning model by default, partitioning is designed specifically for multiple-model learning. This new approach adopts the role of feature construction in constructive induction: to formulate a new input specification from the original one (Donoho, 1996). It uses partitioning to *decompose* a learning task into parts that are individually useful (using *aggregation* as described in Section 2.1), 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 (Stepp & Michalski, 1996) or cluster formation using self-organizing algorithms (Kohonen, 1990; Hsu et al., 1999). 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).

Figure 2 depicts the state space of all partitions of a set of 4 attributes. The size of the state space for n attributes is B_n, the nth Bell number, defined as follows:

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

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

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 algorithms such as hill climbing, best-first search, beam search, and A/A* (Barr & Feigenbaum, 1981; Russell & Norvig, 1995) may be applied. This evaluation function is the *modular mutual information score* (Hsu, 1998), which measures mutual information across subsets of a partition (Jordan, 1997b). 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). This quantity, *modular mutual information*, is denoted I_i for each subset of input attributes $\mathbf{X}_i$. The score is inversely proportional to the difference between joint and total conditional mutual information (i.e., shared information among all subsets). The *modular common information* is denoted I_∇ for an entire partition $\mathbf{X}$:

$$I_i =_{\text{def}} I(\mathbf{X}_i; \mathbf{Y} \mid \mathbf{X}_{\neq i}) =_{\text{def}} H(\mathbf{X}; \mathbf{Y}) - H(\mathbf{X}_i \mid \mathbf{Y}, \mathbf{X}_1, \ldots, \mathbf{X}_{i-1}, \mathbf{X}_{i+1}, \ldots, \mathbf{X}_k)$$

$$I_\nabla =_{\text{def}} I(\mathbf{X}_1; \mathbf{X}_2; \ldots; \mathbf{X}_k; \mathbf{Y}) =_{\text{def}} I(\mathbf{X}; \mathbf{Y}) - \sum_{i=1}^{k} I_i$$

$$M_{\mathit{MS\text{-}HME}} = \left(\sum_{i=1}^{k} I_i\right) - I_\nabla = 2\left(\sum_{i=1}^{k} I_i\right) - I(\mathbf{X}; \mathbf{Y})$$

The purpose of the score, $M_{\mathit{MS\text{-}HME}}$, is to reward high conditional mutual information between an attribute subset and the desired output given other subsets (i.e., *each learning component will be alloted a large share of the work through the subproblem defined on that subset*). It should also penalize high common information (i.e., the gating network is alloted more work relative to the experts). Note that while the partition separates inputs into groups of channels $\mathbf{X}_i$, it does not affect the intrinsic cross-information among these groups.

In the ideal case, this metric yields a speedup that reduces partition search to an NP-complete problem—that is, finding the optimum partition of size 2, from among 2^{n-1}, then repeating this for the resulting refinements (or subpartitions). Empirical experiments described in (Hsu, 1998) demonstrate a speedup (up to 40 times for a synthetic 8-attribute problem) that effectively doubles the number of attributes that can be partitioned using algorithm A. Members of the schema for the optimum partition (Goldberg, 1989) are also shown to have high evaluation function scores.

3.2. 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. 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., 1999) and *k-means clustering* (Russell & Norvig, 1995). Other methods, such as competitive clustering or vector quantization using radial basis functions (Haykin, 1994), neural trees (Li, Fang, & Li, 1993), and similar models (Duda & Hart, 1973; Ray & Hsu, 1998), principal components analysis (Watanabe, 1985; Haykin, 1994), 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.

3.3. 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, 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, Bienenstock, & Doursat, 1992). Learning without this characterization is known in the statistics literature as *model-free estimation or nonparametric statistical inference*.

For time series, we seek to *identify* a stochastic process type from the training data (i.e., a process that generates the observations, as documented in Section 2.1). The performance element, time series classification, will then apply a *model* of this process (represented by exactly one memory form) to a continuation of the input (i.e., "test" data) to generate predictions. For example, an exponential trace memory form (Mozer, 1994; Ray & Hsu, 1998; Hsu, 1998) can express certain types of MA(1) processes (Box, Jenkins, & Reinsel, 1994; Kantz & Schreiber, 1997). The more precisely a time series can be described in terms of exponential processes, 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. A metric that measures this degree of match on a time series is therefore a useful predictor of IR network performance.

Table 1 lists three learning architectures (rows of the "lookup table" in figure 1) and metrics corresponding to their strengths. These are referred to as *node metrics* because the choice of architecture is local to each node (subnetwork) in a hierarchy, corresponding to a single learning subtask. The choice of hierarchical model is global over all subtasks, so the corresponding metrics are called *tree metrics*. The metrics are called *prescriptive* because each one provides evidence in favor of a particular architecture.

Table 1. Learning architectures and their prescriptive metrics.

Learning architecture	Node metric
Simple recurrent network (SRN)	Exponential trace (MA) score
Time delay neural network (TDNN)	Autoregressive (AR) score
Gamma network	Autoregressive moving average (ARMA) score

Table 2. Hierarchical models for classifier fusion and their prescriptive metrics.

Hierarchical model type	Tree metric
Specialist-Moderator (SM) Network	Factorization score
Multistrategy Hierarchical Mixture of Experts (MS-HME) Network	Modular mutual information score

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 paper 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, Verdenius, & Aha, 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.) The time series learning architectures that populate part of a collection of model *components* (Smyth, 1998), along with their prescriptive (node) metrics, are documented in Section 2.1.

Two mixture models (columns of the "lookup table" in figure 1, listed in Table 2), are presented in this paper. These are the *Hierarchical Mixture of Experts* (HME) of Jordan et al. (Jordan, Jacobs, & Barto, 1991; Jacobs et al., 1991; Jordan & Jacobs, 1994) and the *Specialist-Moderator* (SM) network of Ray and Hsu (Ray & Hsu, 1998; Hsu & Ray, 1998). This design choice is a critically important consideration in how a hierarchical learning model is built, and thereby affects the performance of multistrategy approaches to learning from heterogeneous time series. The learning methods being evaluated define the hierarchical model used to perform multistrategy learning in the integrated, or composite, learning system. The expected performance of this 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 the subproblem definitions into account. As a convention, the choice of *partition* (and intermediate training targets) is committed first; next, the hierarchical model type; then, the learning architectures for each subset. Each selection is made subject to the previous choices.

The tree metric for specialist-moderator networks is the *factorization score*. This is an empirical measure of how *evenly* the learning problem is modularized (Ray & Hsu, 1998); it is not specific to time series data. In (Hsu, 1998), a factorization is defined for an intermediate target that is formed through cluster definition using a subset $\boldsymbol{a}_i$ of the partition; that is, the set of distinguishable classes depends on the *restricted view* through a subset of the original attributes. We characterize this restricted view in terms of the number of distinguishable output classes o_i for each subset $\boldsymbol{a}_i$, $1 \leq i \leq k$. If the product of all o_i is N, then the score is defined:

$$M_{SM} = -\sum_{i=1}^{k} \left| \lg \left(\frac{o_i}{\sqrt[k]{N}} \right) \right|$$

The tree metric $M_{MS\text{-}HME}$ for HME-type networks (Jacobs, Jordan, & Barto, 1991; Jacobs et al., 1991; Jordan & Jacobs, 1994) is given in Section 3.1.

Definition. A *composite* is a set of tuples $\mathbf{L} = ((A_1, B_1, \theta_1, \gamma_1, S_1), \ldots, (A_k, B_k, \theta_k, \gamma_k, S_k))$, where A_i and B_i are sets of input and output attributes, θ_i and γ_i are names of network parameters and hyperparameters cf. (Neal, 1996) (i.e., the learning architecture), and S_i is the name of a learning method (a training algorithm and a mixture model).

A composite is depicted in figure 1 in the box labeled “learning specification”. Ideally, a composite would specify the partitioning of input attributes, synthetic attributes, and all high-level model descriptors. These include trainable weights and biases; the specification of model structure (e.g., number, size, and connectivity of ANN hidden layers); the initial conditions for learning (e.g., prior distributions of parameter values); and most important for time series learning, the process type. Composites are generated using the following algorithm.

Given:

1. A (multiattribute) time series data set $D = ((\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, \boldsymbol{y}^{(n)}))$ with input attributes $\boldsymbol{A} = (a_1, \ldots, a_I)$ such that $\boldsymbol{x}^{(i)} = (\boldsymbol{x}_1^{(i)}, \ldots, \boldsymbol{x}_I^{(i)})$ and output attributes $\boldsymbol{B} = (b_1, \ldots, b_O)$ such that $\boldsymbol{y}^{(i)} = (y_1^{(i)}, \ldots, y_O^{(i)})$
2. A constructive induction function F (as described in Sections 3.1 and 3.2) such that $F(A, B, D) = \{(A', B')\}$, where A' is an attribute partition and B' is the set of intermediate concepts for each subset of A'.

```
Algorithm **Select-Net** $(D, \boldsymbol{A}, \boldsymbol{B}, F)$
  **repeat**
    Generate a candidate representation $(A', B') \in F(A, B, D)$
    **for** each learning architecture $\tau^a$
      **for** each subset $A'_i$ of $A'$
            Compute *node* metrics $\mathbf{x}^a_{i\tau} = m^a_\tau(A'_i, B'_i)$ that evaluate $\tau^a$ with respect to
                  $(A'_i, B'_i)$.
      **for** each learning method $\tau^d$
            Compute *tree* metrics $\mathbf{x}^d_\tau = m^d_\tau(A', B')$ that evaluate $\tau^d$ with respect to
                  $(A', B')$.
    Normalize the metrics $\boldsymbol{x}_\tau$ using a precalibrated function $\boldsymbol{G}_\tau$—see Eq. (1).
    Select the most strongly prescribed architecture $(\theta, \gamma)$ and learning method $S$ for
      $(A', B')$, i.e., the table entry (row and column) with the highest metrics.
    **if** the fitness (strength of prescription) of the selected model meets a
      predetermined threshold
    **then** accept the proposed representation and learning technique $(A', B', \theta, \gamma, S)$
  **until** the set of plausible representations is exhausted
  Compile and train a *composite*, **L**, from the selected complex attributes and
      techniques.
  Compose the classifiers learned by each component of **L** using data fusion.
```

$$
\begin{aligned}
t_\tau &: \text{shape parameter} \\
\lambda_\tau &: \text{shape parameter} \\
G_\tau(x_\tau) &= \int_0^{x_\tau} f_\tau(x)\,dx \\
f_\tau(x) &= \frac{\lambda_\tau e^{-\lambda_\tau x}(\lambda_\tau x)^{t_\tau - 1}}{\Gamma(t_\tau)} \\
\Gamma(t_\tau) &= \int_0^{\infty} e^{-y} y^{t_\tau - 1}\,dy
\end{aligned}
$$

Equation 1. Normalization formulas for metrics x_τ ($\tau =$ metric type)

The normalization formulas for metrics, given in Eq. (1), simply describe how to fit a multivariate gamma distribution $\boldsymbol{f}_\tau$, based on a *corpus of homogeneous data sets* (cf. (Hsu & Zwarico, 1995), for a related model selection application). In this calibration phase, each data set is a “training point” for the metric normalization function, $\boldsymbol{G}_\tau$ (i.e., the shape and scale parameters of $\boldsymbol{f}_\tau$). By applying **Select-Net** with $\boldsymbol{G}_\tau$ thus calibrated, we can generate a learning composite—a *specification* for supervised, multistrategy learning on a decomposed time series. A composite is implemented from a database of model components. For discussions on populating this database, the reader is referred to (Hsu, 1998).

4. Adapting hierarchical models to multistrategy time series learning

Decomposition of supervised learning tasks, as presented in this paper, entails three stages: subproblem definition, model selection for subproblems, and reintegration of trained models. This section examines the third and final stage, reintegration, by means of *hierarchical mixture models*. It presents the problem of *data fusion* in composite learning, and a generic, hierarchical approach using probabilistic networks. It then surveys the *hierarchical mixture of experts* (HME) of Jordan et al. (Jacobs, Jordan, & Barto, 1991; Jacobs et al., 1991; Jordan & Jacobs, 1994), and the *specialist-moderator (SM) network*, an architecture that was specifically designed for data fusion in decomposition of learning tasks.

4.1. Data fusion and probabilistic network composites

This section presents *specialist-moderator (SM)* networks and *hierarchical mixtures of experts (HME)* for reintegration of composite time series models. The system overview in Figure 3 depicts a learning system for decomposable time series. The central element of this system is a hierarchical *mixture model*—a general architecture for combining predictions from submodels. In this research, the submodels are recurrent ANNs. Attribute partitioning, described in Section 3.1, produces the subdivided inputs, x_{0n}, to these specialist, or expert, subnetworks (henceforth called *specialists* or *experts*). Unsupervised learning methods, such as self-organizing feature maps (SOMs) (Kohonen, 1990; Hsu et al., 1999) and competitive clustering (Haykin, 1994), are applied to form intermediate targets y_{0n}, as

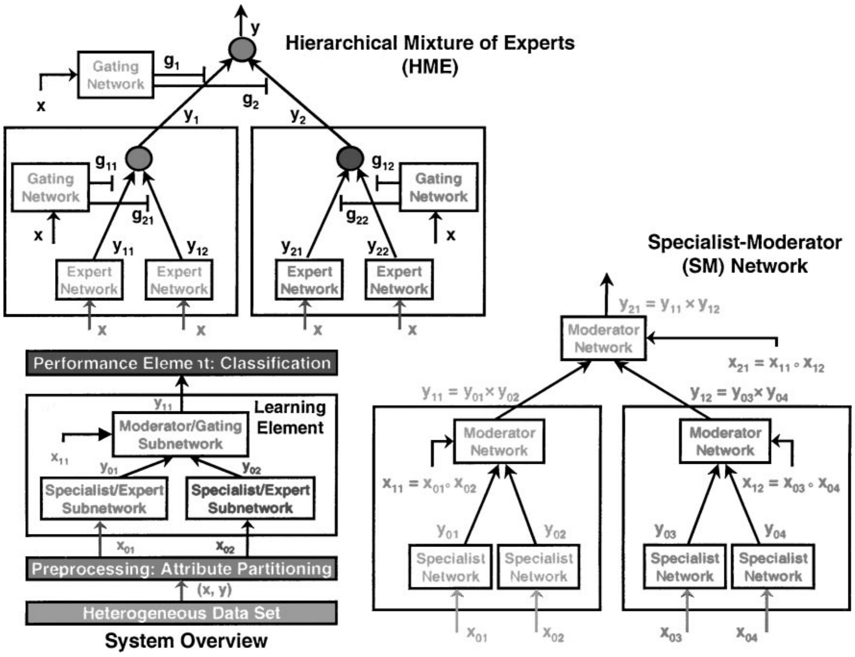

Figure 3. Role of hierarchical mixtures and two mixture models (HME and SM networks).

described in Section 3.2. Selection of subnetwork types is documented in Section 3.3. The overall concept (y_{11}) is the learning target for the top-level moderator.

A *mixture model* is one that combines the outputs of a finite set of subordinate models by weighted averaging (Haykin, 1994). The weights are referred to as *mixing proportions* (Haykin, 1994), *mixing coefficients*, *gating coefficients* (Jordan & Jacobs, 1994), or simply "weights". Traditionally, a mixture model is formally defined as a probability density function (pdf), f, that is the sum of weighted contributions from subordinate models:

$$f(\mathbf{y}; \boldsymbol{\theta}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \pi_n f_n(\mathbf{y}; \boldsymbol{\theta})$$

$$\text{where} \sum_{n=1}^{N} \pi_n = 1 \quad \text{and} \quad \pi_n \geq 0 \qquad \text{for all } n$$

f_n are the individual pdfs for mixture components, drawn from populations $S_n 1 \leq n \leq N$, and f is a pdf over samples $\mathbf{y}$ drawn uniformly from the population S. That is, f_n denotes the likelihood that S_n contributes $\mathbf{y}$ to the mixture S. π_i denotes the *normalized weight* for this likelihood (Haykin, 1994). The parameters θ include all unknowns in the subordinate models upon which the distributions f_n are to be conditioned. This generalizes over all parameters of the learning architecture, such as network weights and biases. The hyperparameters π are simply the mixing coefficients. The mixture estimation problem is to fit π, given training data $(\mathbf{y}_1, \ldots, \mathbf{y}_n, \mathbf{y})$.

An alternative definition (Jordan & Jacobs, 1994) that is more familiar to the nomenclature of connectionist (probabilistic network) learning is to estimate the distribution of **y** as a weighted sum of predictions. The mixing coefficients still denote the normalized weight for a likelihood function over samples from a population, but we have now specified that the *estimator* for the likelihood function is the output of an *expert*. As Jordan and Jacobs (Jordan & Jacobs, 1994) and Haykin (1994) note, experts may be arbitrary learning components. For example, Haykin specifically considers experts that are *rule generators* or arbitrary probabilistic network regression models, with real-valued, discrete, or 1-of-C ("locally") coded targets (Kohavi & John, 1997; Sarle, 1999). In this paper, only discrete (including binary) and 1-of-C-coded classification targets are considered.

Finally, an even more flexible formulation of mixture models is as a *hierarchical mixture network* (Hsu, 1998), whose vertices all represent subnetworks. The leaves are experts or specialist networks; the internal vertices, gating or moderator subnetworks. The target distribution $f(\mathbf{y})$ is thus described as a parameter estimation problem, where the submodel parameters θ belong to probabilistic networks such as feedforward or recurrent ANNs. For multilayer perceptrons, or MLPs (a type of feedforward ANN), the mixture is $\mathbf{y} = f(\mathbf{y}^n)$, $1 \leq n \leq N$, where the vector-valued function f is defined over output channels f_k (of which each is a mixture) in the output layer of the MLP (Neal, 1996; Hsu, 1998). f_k are the mixture estimation targets.

Data fusion, in the context of composite learning, can naturally be interpreted as a mixture estimation problem. Each expert is an inducer trained on some intermediate target concept, resulting in a classifier that maps a subset of the input, or of a continuation of the input (Gershenfeld & Weigend, 1994) for time series, to intermediate predictions that are combined using the mixture model. The rest of this section defines *partitioning* and *aggregation* mixtures, two general types of mixture models that are exemplified by HME and SM networks.

4.2. Multistrategy hierarchical mixture of experts (MS-HME)

This section presents the HME architecture, one of two mixture models that may be selected in our system, and discuses its adaptation to multistrategy learning as an *integrative* method. For a review of existing learning procedures for HME, such as algorithms for Bayesian learning of ANN parameters (Neal, 1996; Jordan, 1997a), and a discussion of how they may be incorporated into a "repertoire" of techniques, the interested reader is referred to (Hsu, 1998). Figure 3 shows an HME network of height 2, with 4 expert networks at its leaves. Note that the expert and gating networks all receive the same input x. The target output values y_{lj}, for level l and (gating or expert) network j, are also identical. Traditional HME uses a tree-structured network of *generalized linear models* (GLIMs), or fixed, continuous, nonlinear functions with linear parameters (McCullagh & Nelder, 1983). GLIMs include single layer perceptrons with linear, sigmoidal, and piecewise linear transfer (activation) functions, which implement regression, binary classification, and hazard models for survival analysis, respectively (Jordan & Jacobs, 1994; Neal, 1996). The mixing is implemented by *gating*

GLIMs that combine outputs from the expert GLIMs. HME networks are trained using an *interleaved* updvate algorithm that computes the error function at the topmost gating network, then propagates credit down through the hierarchy on every pass (a single training *epoch*). This generic procedure can be specialized to expectation-maximization (EM) (Jordan & Jacobs, 1994), gradient (Hsu, 1998), and Markov chain Monte Carlo (MCMC) learning algorithms (Neal, 1996). HME thus supports a type of self-organization over submodels (which are identical in the original formulation); (Jordan & Jacobs, 1994) explains this property and how the "learning load" is distributed over experts by the multi-pass algorithm.

We adapt HME to multistrategy learning (*MS-HME*) by replacing GLIMs with feedforward and recurrent ANNs, with nonlinear (sigmoidal or hyperbolic tangent) or piecewise linear input-to-hidden layer transfer functions and linear hidden-to-output layer transfer functions. The purpose of this modification is to permit an arbitrary *mixture function*, which is implemented by all of the interior (moderator) subnetworks as a whole, to be learned. As Kohavi, Sommerfield, & Dougherty (1996) point out, however, mixture functions that are *not* linear combinations of the input (i.e., those that do not have the same mixing coefficients for any input data) are semantically obscure. Furthermore, the real issue is not the ability to fit a mixture perfectly, because (just as in general concept learning) it is always possible to learn by rote if there are sufficient model resources. The true criterion is *generalization* quality. In general concept learning as well as mixture modeling, we can evaluate generalization by means of cross validation methods (Wolpert, 1992). While such empirical results may not be as conclusive as a computational learning theoretic model (Vapnik, 1995), it is more feasible to collect them than to develop a formal generalization model for recurrent ANNs. Some discretion, therefore, is essential when undertaking to use a general mixture function instead of a linear gating or fusion model.

Finally, in order to adapt HME to *decomposition* of learning problems, it is necessary to make inputs to experts (at the leaves of the tree-structured network) *nonidentical*. Section 3.1 describes attribute partitioning algorithms that split the input data along "columns". Each expert receives the data restricted to one subset of input attributes (i.e., the columns or channels specified by that subset), and each gating network receives as inputs the concatenation of inputs to each expert and the normalized output from each expert. The target outputs at every expert and gating level are identical to one another and to the overall target. Thus, a training exemplar is a set of subnetwork outputs, concatenated with the total input to all experts in the *domain* of the moderator (i.e., the subtree rooted at that moderator). Training a moderator means revising its internal weights to approximate a mixture function.

4.3. Specialist-moderator (SM) networks

This section presents the SM network architecture and its construction, and discuses its adaptation to multistrategy learning. The SM network, which was developed by Ray and Hsu (Ray & Hsu, 1998; Hsu & Ray, 1998), is one of two mixture types that may be selected in our composite learning system. Figure 3 shows an SM network with two layers of moderators. The construction of SM networks allows arbitrary real inputs to the expert (specialist) networks at the leaves of the mixture tree, but constructs higher level attributes based upon x_{0j} (see figure 3). The target output classes of each parent are the Cartesian product (denoted ×) of its children's, and the children's outputs and the concatenation of their input (denoted o) are given as input to the parent. This is one of the two main differences between SM networks and HME. The other is that a specialist-moderator network is trained in a single bottom-up pass, while HME networks are trained iteratively, in a top-down fashion, during *each* M step of EM (Jordan & Jacobs, 1994). Interested readers are referred to (Hsu, 1998) for details of the construction algorithm **SM-net**.

Gradient learning in SM networks was introduced in (Ray & Hsu, 1998) and (Hsu & Ray, 1998). As does HME, SM networks also admit EM (Dempster, Laird, & Rubin, 1977) and MCMC (Neal, 1996) learning for certain specialist architectures (Hsu, 1998).

The primary novel contribution is the model's synergy with attribute-based learning task decomposition:

1. *Reduced variance*. On decomposable time series learning probems, SM networks exhibit lower classification error than non-modular networks of comparable complexity. Section 5 reports results that demonstrate this in a manner very similar to that of Reuckl, Cave and Kosslyn (1989) and Jacobs, Jordan and Barto (1991).
2. *Improved learning efficiency*. Compared to non-modular networks, SM networks require fewer trainable weights or fewer training cycles to achieve convergence on decomposable problems (the reader is referred to (Hsu, 1998) and (Hsu & Ray, 1999) for experimental details).
3. *Facility for multistrategy learning*. Our experimental results, reported in Sections 5 and 6, show improvements using time series specialists of *different types* within the SM network, selected from Table 1.

The main practical distinction between SM networks and HME are the ways in which each one achieves reduced variance and reduced computational complexity. **SM-net** produces moderator networks whose worst-case complexity is the product of that of their children. This growth is limited, however, because the tree height and maximum branch factor are typically (very) small constants (Hsu, 1998). Thus, SM networks trade more rapid growth in complexity and susceptibility to overtraining (compared to HME, which computes no cross-product targets) for increased *resolution capability* and reduction of *localization error*. By exploiting differences among the problem definitions for each subnetwork, an SM network can distinguish among more concepts than its components, and achieve higher classification accuracy than a comparable non-modular network. In time series learning applications such as multimodal sensor integration, this localization error may be reduced in space or time (Jacobs, Jordan, & Barto, 1991; Stein & Meredith, 1993).

5. Experimental results

This section presents experimental results with comparisons to existing inductive learning systems (Kohavi, Sommerfield, & Dougherty, 1996), traditional regression-based methods as adapted to time series prediction, and non-modular probabilistic networks (both atemporal and ARMA-type ANNs).

5.1. A time series learning problem: musical tune classification

This section documents a sensor fusion experiment on *musical tune classification*, illustrated in figure 4. It gives a design rationale for the test bed used to evaluate the SM network. In experiments using hierarchical classifier fusion models, our focus is primarily on *classification* of time series. The architecture addresses one of the key shortcomings of many current approaches to time series learning: the need for an *explicit, formal model of inputs from different modalities*. For example, the specialists at each leaf in the SM network might represent audio and infrared sensors in an industrial or military monitoring system (Stein & Meredith, 1993). The SM network model and learning algorithm, described in Section 4.3, capture this property by allocating different channels of input (collected in each complex input attribute) to every specialist. Other models that can be represented by SM architecture are hierarchies of decision-making committees (Bishop, 1995).

The input data was generated from digitized audio recordings of musical tunes, preprocessed using a simple autocorrelation technique to find a coarse estimate of the *fundamental frequency* (Beauchamp, Maher, & Brown, 1993). This signal was used to produce the *frequency component*, an exponential trace of a tune over 7 input channels (essentially, a 7-note scale). The other group of input attributes is the *rhythm component*, containing 2 channels: the position in the tune (i.e., a time parameter ranging from 1 to 11) and a binary sound-gap indicator. Figure 4 also depicts non-modular and specialist-moderator architectures for learning the musical tune classification database. The non-modular network

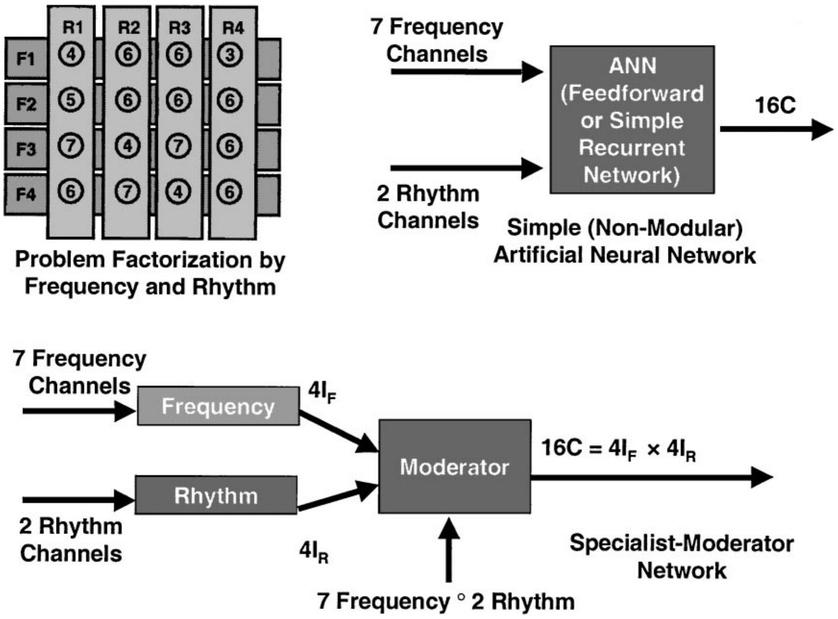

Figure 4. Organization of the musical tune classification experiment.

Table 3. Performance of an SM network versus that of other inducers on the tune classification problem.

	Classification accuracy, musical tune classification (%)							
	Training				Cross validation			
Inducer	Min	Mean	StdDev	Max	Min	Mean	StdDev	Max
ID3	99.4	99.4	0.09	99.6	46.6	63.4	5.67	73.2
ID3, bagged	99.4	99.4	0.09	99.6	48.6	63.4	5.55	74.0
ID3, boosted	99.4	99.4	0.09	99.6	53.4	66.6	4.85	83.6
C5.0	95.0	95.8	0.64	96.3	67.1	77.1	3.41	84.9
C5.0, boosted	94.4	98.9	1.11	99.6	57.5	77.5	5.57	89
IBL	92.7	94.0	1.02	95.6	41.1	52.7	4.88	62.3
Discrete Naïve-Bayes	93.8	95.6	0.78	96.3	41.1	59.6	4.79	67.1
DNB, bagged	93.4	94.6	0.79	96.3	47.9	60.8	4.19	67.1
DNB, boosted	93.8	94.4	0.47	96.5	45.2	58.3	5.34	69.2
PEBLS	72.6	76.8	1.67	84.2	30.8	42.5	4.71	56.8
SM net, FF	–	–	–	*74.9*	–	–	–	*60.2*
SM net, IR	–	–	–	*100.0*	–	–	–	*81.3*

receives all 9 channels of input and is trained using the overall concept class. The first-level (leaf) networks in the specialist-moderator network receive *specialized* inputs: the frequency component only or the rhythm component only. The concatenation of frequency and rhythm components (i.e., the entire input) is given as input to the moderator network, and the target of the moderator network is the Cartesian product of its children's targets (Hsu, 1998). The intermediate targets are equivalence classes $I_F = \{F_1, F_2, F_3, F_4\}$ and $I_R = \{R_1, R_2, R_3, R_4\}$. Experiments using feedforward networks and Elman, Jordan, and input recurrent varieties of simple recurrent networks (Elman, 1990; Principé & Lefebvre, 1998) showed input recurrent networks to achieve higher performance (accuracy and convergence rate) for exponentially coded time series, alone and as part of the specialist-moderator networks (Ray & Hsu, 1998).

Table 3 lists performance statistics (mean, extrema, and standard deviations of classification accuracy) using atemporal inducers such as *ID3*, *C5.0*, Naïve Bayes, *IBL*, and *PEBLS* on the the musical tune classification problem (S4 data set) described in this section. The non-ANN inducers tested are all part of the *MLC++* package (Kohavi, Sommerfield, & Dougherty, 1996). Table 4 shows the performance of the non-modular (simple feedforward and input recurrent) ANNs compared to their specialist-moderator counterparts—each network has approximately 1200 weights each. Each tune is coded using between 5 and 11 exemplars, for a total of 589 training and 128 cross validation exemplars (73 training and 16 cross validation tunes). The italicized networks have 16 targets; the specialists, 4 each. Prediction accuracy is measured by the number of individual exemplars classified correctly in a 1-of-4 or 1-of-16 coding (Sarle, 1999). Significant overtraining was detected only in the frequency specialists and did not affect classification accuracy for this data set. The results illustrate that input recurrent networks (simple, specialist, and moderator) are more capable

Table 4. Performance of non-modular and specialist-moderator networks.

Design	Network type	Training MSE	Training accuracy	CV MSE	CV accuracy
Feedfwd.	*Simple*	*0.0575*	*344/589 (58.40%)*	*0.0728*	*67/128 (52.44%)*
Feedfwd.	Rhythm	0.0716	534/589 (90.66%)	0.1530	104/128 (81.25%)
Feedfwd.	Frequency	0.0001	589/589 (100.0%)	0.0033	128/128 (100.0%)
Feedfwd.	*Moderator*	*0.0323*	*441/589 (74.87%)*	*0.0554*	*77/128 (60.16%)*
Input rec.	*Simple*	*0.0167*	*566/589 (96.10%)*	*0.0717*	*83/128 (64.84%)*
Input rec.	Rhythm	0.0653	565/589 (95.93%)	0.1912	107/128 (83.59%)
Input rec.	Frequency	0.0015	589/589 (100.0%)	0.0031	128/128 (100.0%)
Input rec.	*Moderator*	*0.0013*	*589/589 (100.0%)*	*0.0425*	*104/128 (81.25%)*

Table 5. Performance of HME and specialist-moderator networks.

Design	Training MSE	Training acc.	CV MSE	CV accuracy
HME, 4 leaves	0.0576	387/589 (65.71%)	0.0771	58/128 (45.31%)
HME, 8 leaves	0.0395	468/589 (79.46%)	0.0610	77/128 (60.16%)
SM net, FF	0.0323	441/589 (74.87%)	0.0554	77/128 (60.16%)
SM net, IR	0.0013	589/589 (100.0%)	0.0425	104/128 (81.25%)

of generalizing over the temporally coded music data than are feedforward ANNs. The advantage of the specialist-moderator architecture is demonstrated by the higher accuracy of the moderator test predictions (100% on the training set and 81.25% or 15 of 16 tunes on the cross validation set, the highest among the inducers tested). As Table 5 shows, our implementation of a non-recurrent HME network (trained using gradient learning) with 8 leaves outperforms the version with 4 leaves and is comparable to the specialist-moderator network of feedforward networks. It is, however, outperformed by the specialist- moderator network of input recurrent networks.

5.2. An application: crop condition monitoring

The real-world test bed *for model selection* in our multistrategy learning system is a prediction and monitoring problem using weekly crop condition estimates (corn condition in Illinois farms) collected over 11 years. Hsu (1998) describes the problem in detail and presents several visualizations of the time series data. The data is shown to admit two embedded process types: an exponential trace (MA) process and an autoregressive (AR) process. 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, Waibel, & Hinton, 1990; Haykin, 1994) or simple recurrent network (Elman, 1990; Principé & Lefebvre, 1998), respectively). The training target is quantized to nominal values: {*very poor, poor, fair, good, very good*}, thereby defining a *predictive evaluation*, or *simulation*, model. (Hsu, 1998) reports how recurrent ANNs outperform linear prediction methods (and certainly outperform naïve linear or quadratic regression, which invariably predict no change in condition from one week to the next) in the "middle to distant future". This is important because the utility of near-term predictions tends to be lower for decision support systems (Russell & Norvig, 1995).

To demonstrate the decomposability of the *crop condition-monitoring* problem, an experiment was first conducted an experiment using Elman, Jordan, and input recurrent networks as well as TDNNs and MLPs (Hsu, 1998). A gamma network in an MS-HME configuration (as defined in Section 4.2) was used to select the correct classifier (if any) for each exemplar. This context-sensitive fusion step combined predictions from the two best overall networks (input recurrent, or IR, with momentum of 0.9 and time-delay neural networks, or TDNNs, with momentum of 0.7). It reduced the error by almost half, indicating that even with identical inputs and targets, a simple mixture model could reduce variance. These results are reported in (Hsu, 1998). 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).

Table 6 summarizes the performance of an MS-HME network versus that of other induction algorithms from *MLC*++ (Kohavi, Sommerfield, & Dougherty, 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, Jenkins, & Reinsel, 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. Research that is related to this paper (Hsu et al., 1999) applies this methodology to specific problems in diagnostic monitoring for decision support (or *recommender*) systems (Resnick & Varian, 1997).

6. Conclusions and future work

This section analyzes the experimental results reported in the previous sections, especially Section 5. It begins with a discussion of the design choices and properties of interest, continues with an account of the the main findings and their ramifications, and concludes with a brief synopsis of current and future work.

Table 6. Accuracy of MS-HME versus that of other inducers on the crop condition monitoring problem.

	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*
MS-HME	*98.2*	*98.9*	*100.0*	*0.54*	*52.9*	*79.0*	*96.9*	*14.99*

Traditionally, domain knowledge about the sources of data is used in their decomposition (Hsu & Ray, 1998; Hsu et al., 1998). This is typical of the time series learning problems surveyed in Section 1; examples of heterogeneous time series with multiple data sources include multimodal sensor integration (sensor fusion) and multimodal HCI. Section 3.1 describes a knowledge-free approach (attribute partitioning) that can be applied when such information is not available, but the learning problem is decomposable. As explained in Section 1, this paper focuses on decomposable learning problems defined over heterogeneous time series. To briefly recap, a heterogeneous time series is one containing data from multiple sources (Stein & Meredith, 1993), and typically contains different embedded temporal patterns, which can be formally characterized in terms of different memory forms (Mozer, 1994). These sources can therefore be thought to correspond to different "pattern-generating" stochastic processes. A decomposable learning problem is one for which multiple subproblems can be defined by systematic means, possibly based on heuristic search (Barr & Feigenbaum, 1981; Russell & Norvig, 1995; Kohavi & John, 1997) or other approximation algorithms. Some specific properties that characterize most kinds of heterogeneous and decomposable time series, and are typically of interest for real-world data, are as follows:

1. *Heterogeneity*: multiple processes for which a stochastic model is known or can be hypothesized and tested
2. *Decomposability*: a known or hypothesized method for isolating one or more of these processes (often part of the application domain knowledge)

3. *Feasibility*: evidence that that (ideally) all of identifiable embedded processes are homogeneous

These properties are present to some degree in the musical tune classification and crop condition monitoring test beds, and can be simulated in purely synthetic data (Hsu, 1998).

An important topic of continued research is the process of automating task decomposition for model selection. This paper has shown how recurrent neural networks and hierarchical mixture models can be organized for multistrategy learning. Some of the findings reported here indicate that the most appropriate learning architecture, mixture model, and training algorithm can be selected for each subproblem in a modular task decomposition. For example, a boost in classifier accuracy was achieved on the crop condition monitoring problem by using multistrategy (i.e., multiple process model) learning. This shows how the quality of generalization achieved by a mixture of classifiers can benefit from the ability to identify the "right tool" for each job. The findings reported here, however, only demonstrate the improvement for a very limited set of real-world problems, and a (relatively) small range of stochastic process models. This needs to be greatly expanded (through collection of much more extensive corpora) to form any definitive conclusions regarding the efficacy of the coarse-grained model selection approach. The relation of model selection to attribute formation and data fusion in time series is an area of continuing research (Hsu & Ray, 1998; Hsu & Ray, 1999). A key question that the authors continue to investigate is: how does attribute partitioning-based decomposition support *relevance determination* (Kohavi & John, 1997) in a modular learning architecture?

Another very important issue that is beyond the scope of this paper is the role of prior background knowledge (e.g., about time series preprocessing, the sources of data, etc). In the musical tune classification problem, for example, the 4-by-4 factorization was discovered using competitive clustering by Gaussian radial-basis functions (RBFs) (Haykin, 1994; Ray & Hsu, 1998). In this experiment, the frequency and rhythm partitioning of *input* is self-evident in the signal processing construction, so the *subdivision of input* through attribute partitioning could have been constrained or guided by prior knowledge from sensor specifications (Stein & Meredith, 1993). (Note, however, that the intermediate targets are *not* known in advance, and the same knowledge is not necessarily useful for cluster definition.)

Acknowledgments

Support for this research was provided in part by the Office of Naval Research under grant N00014-95-1-0749 and by the Naval Research Laboratory under grant N00014-97-C-2061. The authors thank Nathan Dale Gettings for helpful discussions on data fusion and time series analysis. Thanks also to Nathan Dale Gettings, Yu Pan, and Victoria Lease for implementations of several key components of the system described in this paper.

References

Barr, A. & Feigenbaum, E. A. (1981). Search. *The handbook of artificial intelligence* (Vol. 1, pp. 19–139). Reading, MA: Addison-Wesley.

Beauchamp, J. W., Maher, R. C., & Brown, R. (1993). Detection of musical pitch from recorded solo performances. *Proceedings of the 94th Convention of the Audio Engineering Society*, Berlin, Germany.

Benjamin, D. P. (Ed.) (1990). *Change of representation and inductive bias*. Boston: Kluwer Academic Publishers.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford, UK: Clarendon Press.

Box, G. E. P., Jenkins, G. M., & Reinsel, G. C. (1994). *Time series analysis, forecasting, and control*, 3rd ed. San Fransisco, CA: Holden-Day.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

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

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, 39* (Series B), 1–38.

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. (1973). *Pattern classification and scene analysis*. New York, NY: John Wiley and Sons.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179–211.

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)*. Menlo Park, CA: AAAI Press.

Freund, T. & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference on (ICML-96)*.

Fu, L.-M. & Buchanan, B. G. (1985). Learning intermediate concepts in constructing a hierarchical knowledge base. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-85)*, Los Angeles, CA (pp. 659–666).

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. (1994). The Future of Time Series: Learning and Understanding. In A. S. Weigend & N. A. Gershenfeld (Eds.), *Time series prediction: forecasting the future and understanding the past, Santa Fe institute studies in the sciences of complexity* (Vol. XV). Reading, MA: Addison-Wesley.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Grois, E., Hsu, W. H., Wilkins, D. C., & Voloshin, M. (1998). Bayesian network models for automatic generation of crisis management training scenarios. *Proceedings of the National Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, Madison, WI (pp. 1113–1120). Menlo Park, CA: AAAI 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/index.html.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. New York, NY: Macmillan College Publishing.

Horvitz, E. & Barry, M. (1995). Display of information for time-critical decision making. *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo, CA: Morgan-Kaufmann.

Heckerman, D. A. (1996). *A tutorial on learning with bayesian networks*. Microsoft Research Technical Report 95-06, revised June 1996.

Hsu, W. H. (1998). *Time series learning with probabilistic network composites*. Ph.D. Thesis, University of Illinois at Urbana-Champaign (UIUC-DCS-R2063). URL: http://www.ncsa.uiuc.edu/People/bhsu/thesis.html.

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. *Proceedings of the International Workshop on Multistrategy Learning*, Milan, Italy.

Hsu, W. H., Auvil, L. S., Pottenger, W. M., Teheng, D., & Welge, M. (1999). Self-organizing systems for knowledge discovery in databases. *Proceedings of the International Joint Conference on Neural Networks (IJCNN-99)*, Washington, DC.

Hsu, W. H. & Ray, S. R. (1998). A new mixture model for concept learning from time series. *Proceedings of the 1998 Joint AAAI-ICML Workshop on AI Approaches to Time Series Problems (Technical Report WS-98-07)*, Madison, WI (pp. 42–43). Menlo Park, CA: AAAI Press.

Hsu, W. H. & Ray, S. R. (1999). A recurrent mixture model for time series classification. *Proceedings of the International Joint Conference on Neural Networks (IJCNN-99)*, Washington, DC.

Hsu, W. H. & Zwarico, A. E. (1995). Automatic synthesis of compression techniques for heterogeneous files. *Software: Practice and Experience, 25*(10), 1097–1116.

Jacobs, R. A., Jordan, M. I., & Barto, A. G. (1991). Task decomposition through competition in a modular connectionist architecture: the what and where vision tasks. *Cognitive Science, 15*, 219–250.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79–87.

Jordan, M. I. (1997a). Approximate inference via variational techniques. *International Conference on Uncertainty in Artificial Intelligence*, Providence, RI, invited talk.

Jordan, M. I. (1997b). Personal communication.

Jordan, M. I. & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation, 6*, 181–214.

Kantz, H. & Schreiber, T. (1997). *Nonlinear time series analysis*. Cambridge, UK: Cambridge University Press.

Kira, K. & Rendell, L. A. (1992). The feature selection problem: traditional methods and a new algorithm. *Proceedings of the National Conference on Artificial Intelligence (AAAI-92)*, San Jose, CA (pp. 129–134). Cambridge, MA: MIT Press.

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. (1996). Data mining using *MLC*++: A machine learning library in C++. *Tools with artificial intelligence* (pp. 234–245). Rockville, MD: IEEE Computer Society Press.

Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE, 78*, 1464–1480.

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.

Li, T., Fang, L. & Li, K. Q-Q. (1993). Hierarchical classification and vector quantization with neural trees. *Neurocomputing, 5*, 119–139.

McCullagh, P. & Nelder, J. A. (1983). *Generalized linear models*. London, UK: Chapman and Hall.

Mengshoel, O. J. & Wilkins, D. C. (1996). Recognition and critiquing of erroneous student actions. *Proceedings of the AAAI Workshop on Agent Modeling* (pp. 61–68). Menlo Park, CA: AAAI Press.

Michalski, R. S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence, 20*(2), 111–161. Reprinted in; *Readings in knowledge acquisition and learning*, B. G. Buchanan & D. C. Wilkins (Eds.) (1993). San Mateo, CA: Morgan-Kaufmann.

Mozer, M. C. (1994). Neural net architectures for temporal sequence processing. In A. S. Weigend & N. A. Gershenfeld (Eds.), *Time series prediction: forecasting the future and understanding the past, Santa Fe institute studies in the sciences of complexity* (Vol. XV). Reading, MA: Addison-Wesley.

Neal, R. M. (1996). *Bayesian learning for neural networks*. New York, NY: Springer-Verlag.

Principé, J. & deVries. (1992). The Gamma Model—A new neural net model for temporal processing. *Neural Networks, 5*, 565–576.

Principé, J. & Lefebvre, C. (1998). *NeuroSolutions v3.02*. 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). URL: http://www-east.elsevier.com/ida/browse/0204/ida00039/ida00039.htm.

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

Rueckl, J. G., Cave, K. R., & Kosslyn, S. M. (1989). Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A computational investigation. *Journal of Cognitive Neuroscience, 1*, 171–186.

Russell, S. & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Englewood Cliffs, NJ: Prentice Hall.

Sarle, W. S. (Ed.) (1999). *Neural network FAQ*, periodic posting to the USENET newsgroup *comp.ai.neural-nets*.

Schuurmans, D. (1997). A new metric-based approach to model selection. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, Providence, RI (pp. 552–558).

Smyth, P. (1998). Challlenges for the application of machine learning problems. *1998 Joint AAAI-ICML Workshop on the Methodology of Applying Machine, Madison*, WI, Invited talk. Menlo Park, CA: AAAI Press.

Stein, B. & Meredith, M. A. (1993). The merging of the senses. Cambridge, MA: MIT Press.

Stepp, R. E. & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial Intelligence Approach*. San Mateo, CA: Morgan-Kaufmann.

Stone, M. (1977). An asymptoticl equivalence of choice of models by cross-validation and akaike's criterion. *Journal of the Royal Statistical Society Series B, 39*, 44–47.

Vapnik, V. N. (1996). *The nature of statistical learning theory*. New York, NY: Springer-Verlag.

Watanabe, S. (1985). *Pattern recognition: human and mechanical*. New York, NY: John Wiley and Sons.

Wilkins, D. C. & Sniezek, J. A. (1997). *DC-ARM: Automation for reduced manning*. Knowledge Based Systems Laboratory, Technical Report UIUC-BI-KBS-97-012. Beckman Institute, University of Illinois at Urbana-Champaign.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks, 5*, 241–259.

Received December 22, 1998

Accepted June 18, 1999

Final manuscript June 18, 1999