# Construction of Recurrent Mixture Models for Time Series Classification

William H. Hsu[1] and Sylvian R. Ray[2]
[1]bhsu@ncsa.uiuc.edu http://www.ncsa.uiuc.edu/People/bhsu
*Automated Learning Group, National Center for Supercomputing Applications, UIUC*
[2]ray@cs.uiuc.edu http://www.cs.uiuc.edu/contacts/faculty/ray.html
*Department of Computer Science, University of Illinois at Urbana-Champaign*

## Abstract

*We present a new hierarchical network architecture that integrates the outputs of recurrent ANNs. The purpose of this architecture is to apply decomposition of time-series learning tasks (using self-organization on multi-channel input). Our approach yields the variance-reducing benefits of techniques such as stacked generalization, but exploits the ability of abstract targets to be factored based upon preprocessing, feature extraction, or multimodal sensor constraints. This research demonstrates how prior information can be applied to learn factorial structure from time series, to build a mixture of recurrent ANNs.*

***Keywords: mixture models, recurrent networks, clustering, data fusion, time series learning***

## Introduction

This paper presents a new hierarchical network architecture that integrates the outputs of multiple classifiers (recurrent ANNs). The purpose of this architecture is to apply decomposition of learning tasks to achieve higher classification accuracy over time series. Decomposition is achieved using self-organization on multi-channel input, to produce multiple, intermediate training targets for *specialist* subnetworks in the hierarchy. This approach yields the variance-reducing benefits of techniques such as stacked generalization, but facilitates the use of *factorial structure* (the ability of abstract targets to be factored). Accounting for the modular grouping of inputs is critical to learning for time series classification, because factorial structure occurs frequently due to time series preprocessing, feature extraction, or multimodal sensors. This research demonstrates the feasibility of learning such structure from data (using prior information to partition inputs) and applying it in time series to build a mixture model composed of recurrent ANNs.

## Background

### Recurrent Mixture Models

Mixture estimation has been extensively studied in artificial neural network research [1, 2]. Hierarchical mixture models such as HME [1] have been shown to greatly reduce convergence time by decomposing learning tasks through self-organization. Recent research has also shown how inductive learning algorithms can be augmented by *aggregation mixtures* such as bootstrap aggregation (or bagging) [3] and stacked generalization [4], and by *partitioning mixtures* such as *boosting* [5] and *hierarchical mixture of experts* (or *HME*) [1].

More recently, hierarchical mixture estimation using temporal models such as multi-time models [6] has been applied to robot planning. Meanwhile, structural partitioning through control of mixture hyperparameters (such as the number of trainable weights in an ANN or HMM [7]) has recently been applied to time series learning. [8] describes such approach as applied to continuous speech recognition using expectation-maximization (EM) and mixtures of HMMs.

This paper describes a hierarchical mixture model called a *specialist-moderator* network [9], which combines recurrent ANN classifiers in a bottom-up fashion, and derive an algorithm to construct and train the network. The primary novel contribution is the model's ability to self-organize intermediate training targets (concepts) based upon a partition of the input channels (attributes presented to each mixture component). A time series learning experiment demonstrates how this partitioning may be known in advance based upon prior processing specifications (transforms applied to analog data) or sensor specifications. The construction of this network and its learning properties are a function of the factorial structure of the data and the unsupervised learning techniques used to define the intermediate targets [10]. The benefits of this mixture model as a supervised learning architecture are higher classification accuracy, and a faster rate of convergence to this accuracy, compared to non-modular mixtures.

Factorial structure in time series is a strongly exhibited property of multimodal sensor integration problems [11, 12]. The classification accuracy boost yielded by a hierarchical mixture typically means reduced localization error, such as in the *what* and *where* vision tasks [13, 11]. We demonstrate, for a synthetic audio sequence

recognition task, that recognition accuracy can also be improved using a *partial* speficiation of factorial structure (input partitioning).

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 [14, 15], 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 [15], 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 [19], 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.

Recurrent components studied in this research include: simple recurrent networks (SRNs), time-delay neural networks (TDNNs) [16], and gamma memories [17]. SRNs used in preliminary experiments are of the Elman [18], Jordan [15], and input recurrent [19, 10] varieties. [15] presents the convolutional code interpretation of this family of *autoregressive moving average* (ARMA) process models, and the functional descriptor (kernel function) for each memory form.

## Decomposition of Learning Tasks

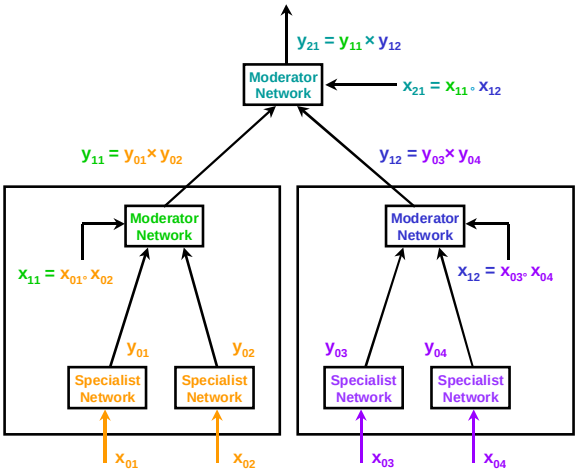


**Figure 1. A specialist-moderator network**

Figure 1 depicts a specialist-moderator (SM) network. An SM network is constructed from a specification of input and output attributes for each of several modules (the leaves of the network). Training data and test input will be presented to these "specialists" according to this specification. The construction algorithm simply generates new input-output specifications for *moderator* networks. The target output classes of each parent are the Cartesian product (denoted $\times$) of its childrens', and the childrens' outputs *and* the concatenation of their inputs (denoted $\circ$) are given as input to the parent. This construction employs *attribute partitioning* [20] and a clustering (vector quantization) step [10] to identify intermediate training targets ($y_{11}$ and $y_{12}$ in Figure 1).

The following is a brief introduction to learning task decomposition by attribute partitioning [9]. *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 [21, 22]. 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: to formulate a new input specification from the original one [23]. It uses subset partitioning to *decompose* a learning task into parts that are individually useful, using *aggregation* of attributes (which, for this time series learning framework, denote the input channels). By contrast, attribute subset selection attempts to *reduce* attributes to a single useful group.

Partitioning permits new intermediate concepts to be formed by unsupervised learning (e.g., conceptual clustering [SM86] or cluster formation using self-organizing algorithms [24, 10]). 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). [9] describes a metric-based model selection algorithm for this architecture.

One significant benefit of this abstraction approach is that it exploits factorial structure in abstract (decomposable) learning tasks. This results in a reduction in network complexity compared to non-modular or non-hierarchical methods, *whenever this structure can be identified* (using prior knowledge, or more interestingly, through vector quantization methods). In addition, the bottom-up construction supports natural grouping of input attributes based on *modalities* of perception (e.g., the data

*channels* or observable attributes available to each "specialist" via a particular sensor). Finally, experiments demonstrate that the achievable test error on decomposable time series learned using a specialist-moderator network is lower than that for non-modular feedforward or temporal ANN, when both are trained to convergence.

It is important to note that prior knowledge of a partition is *not* essential to attribute-based decomposition of learning tasks. Experiments using synthetic (modular variants on parity, or cumulative XOR, functions) showed that simple heuristic search (using A/A* and *the modular mutual information* criterion [20, 9]) can locate the optimal partition for small (5-20) attribute sets. This technique is applicable even when there is absolutely no prior information regarding the factorial structure of the data set. [20] documents these experiments fully.

### Neural Clustering in Constructive Induction

A self-organizing algorithm first presented in [10] is used to organize each of the training sets into self-organized equivalence classes (SOECs). This results in each input vector having an assigned target class for training the relevant expert, or specialist network (each one being a recurrent component: SRN, TDNN, or Gamma memory).

*k*-means clustering or Gaussian clustering algorithms would be quite sufficient for this task, as is Kohonen's Self-Organizing Feature Map (SOFM or SOM). [25] documents a "neural tree" algorithm that implements a form of competitive clustering. However, an even simpler algorithm was used in the study presented here. This algorithm intentionally does not apply any "high-powered" clustering technique. On the contrary, it was chosen to produce only a crude measure of statistical proximity.

The algorithm for SOEC formation is as follows:

**Define:**

D = maximum diameter of the training set.
R = the common radius of classes, a fraction of D
$\vec{\mu}_j$ = exemplar of class j
$\vec{d}_{ij}$ = distance between $\vec{x}_i$ and $\vec{\mu}_i$.

**Algorithm Find-SOEC:**

1. Mark all training vectors *uncommitted.*
2. Choose *R* as any fraction of *D*.
3. j := 1
4. Select any uncommitted training vector, $\vec{x}*$. (If none, stop.)
5. Assign it to $\vec{\mu}_j \leftarrow \vec{x}*$ and mark it committed.
6. Assign all uncommitted $x_i$ satisfying $\vec{d}_{ij} < R$ to class *j*, and mark them committed.
7. Increment *j* and return to step 4.

The number of SOECs will be j-1 when the algorithm terminates, a condition that will, of course, depend on R. A similar algorithm was recently published in [26] as the *APC-III* algorithm. [20] reports another similar union-find-based algorithm for detection of SOECs.

## Methodology

### Time Series Classification Problems

Time series learning tasks are represented as discrete classification (concept learning) problems over continuous-valued input. They can be systematically decomposed by partitioning the input channels (or attributes) based on prior information such as sensor modality. State space search may be applied to automatically search for partitions even if no such information is available [20].

### Attribute Partitioning

Figure 2 depicts the factorial structure of a learning problem defined over a musical tune classification database. The input data was generated as follows. First, digital audio was recorded of the tunes being played by one of the authors. These samples were preprocessed using a simple autocorrelation technique to find a coarse estimate of the *fundamental frequency* [27]. 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.

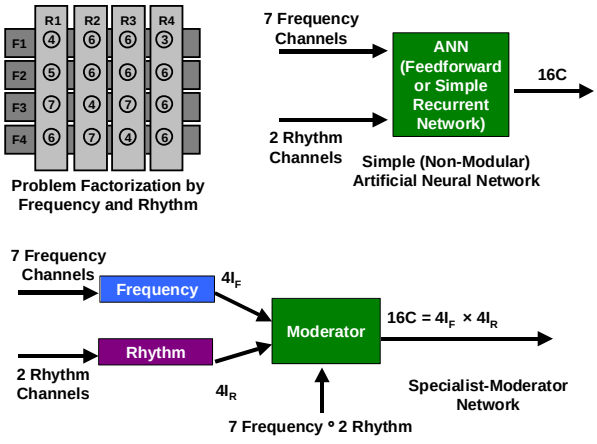


**Figure 2. The musical tune classification problem**

Non-modular and specialist-moderator architectures for this problem, whose performance is reported in the next section, are shown in Figure 2. Each contains approximately 1200 trainable weights.

### Construction of Recurrent Mixtures

Clustering, or vector quantization, is then performed on the *partitioned* training data – i.e., the training data is restricted to one *subset of channels* in the partition on each application of clustering algorithm. This produces new intermediate training targets and defines new learning *subtasks* (mappings from a subset of the input channels to an intermediate target, or codebook, defined by clustering). Preliminary experiments used two clustering algorithm: Kohonen's Self-Organizing Feature Map (SOFM or SOM) [24], and **Find-SOEC**. For these experiments, simple recurrent networks (SRNs) of the Elman [18], Jordan [15], and *input recurrent* (exponential trace memory) [19, 10] variety are used on the resultant learning subtasks. The input recurrent variety was found to yield the highest mixture estimation accuracy on cross-validation data.

## Experimental Results

### Musical Tune Classification

The non-modular network shown in Figure 2 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. The targets are intermediate attributes $I_F = \{ F_1, F_2, F_3, F_4 \}$ and $I_R = \{ R_1, R_2, R_3, R_4 \}$. This 4-by-4 factorization was discovered using the clustering algorithm **Find-SOEC** presented in the second section. In this experiment, the frequency and rhythm partitioning of *input* is self-evident in the signal processing construction, so the *subdivision of input* is known (note, however, that the intermediate targets are *not* known in advance). When the input subdivision is also unknown, attribute partitioning methods can be used to automatically determine which inputs are *relevant* to a particular specialist [20].

### Boosting Prediction Accuracy

Table 1 shows the performance of the non-modular (simple feedforward and input recurrent) ANNs compared to their specialist-moderator counterparts. 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. Significant overtraining was detected only in the frequency specialists. This did not, however, affect classification accuracy for our data set. The results illustrate that input recurrent networks (simple, specialist, and moderator) are more capable of generalizing over the temporally coded music data than are feedforward ANNs. Table 2 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 are all part of the MLC++ package [28].

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

**Table 1. Performance of non-modular versus SM networks.**

| | Classification Accuracy (%), Musical Tune Classification | | | |
|---|---|---|---|---|
| | Cross Validation | | | |
| Inducer | Min | Mean | StdDev | Max |
| ID3 | 46.6 | 63.4 | 5.67 | 73.2 |
| ID3, bagged | 48.6 | 63.4 | 5.55 | 74.0 |
| ID3, boosted | 53.4 | 66.6 | 4.85 | 83.6 |
| C5.0 | 67.1 | 77.1 | 3.41 | 84.9 |
| C5.0, boosted | 57.5 | 77.5 | 5.57 | 89 |
| IBL | 41.1 | 52.7 | 4.88 | 62.3 |
| Discrete Naïve-Bayes | 41.1 | 59.6 | 4.79 | 67.1 |
| DNB, bagged | 47.9 | 60.8 | 4.19 | 67.1 |
| DNB, boosted | 45.2 | 58.3 | 5.34 | 69.2 |
| PEBLS | 30.8 | 42.5 | 4.71 | 56.8 |
| *SM net, FF* | – | – | – | *60.2* |
| *SM net, IR* | – | – | – | *81.3* |

**Table 2. SM network versus MLC++ inducers (CV)**

### Improving Learning Speed

Time to convergence, based on a stopping criterion (i.e., stop training when cross-validation MSE increases), in at most 4000 epochs for both SM networks and non-modular

networks. When run to convergence under a threshold criterion (i.e., stop training when training MSE changes by less than 0.0001), the SM and non-modular networks achieve the same accuracy as reported above.

## Applications

### Multisensor Integration

The novel contribution of this architecture is that it uses newly-formulated intermediate training targets, which are discovered using input partitioning and clustering, in all mixture components. It extends traditional HME by making subtask specialization an explicit, unsupervised learning step, rather than an effect of interleaved training of gating (mixture estimation) and expert subnetworks. The experiments above focus upon factorial structure in time series and recurrent network mixtures. The new architecture can reduce variance more efficiently than mixtures of atemporal and non-modular inducers, including feedforward ANNs (using the same input channels to each mixture component and the same target concepts). We hypothesize that this is true when the supervised learning task can be factored: additional experiments reported in [20] support this hypothesis.

### Supervised Learning using Multiple Models

[9] presents a multistrategy learning framework that employs the recurrent mixture (SM networks) presented here. Information theoretic criterion can be used to empirically determine when it is more appropriate to use SM networks versus recurrent HME-type mixtures.

### Scalable Data Mining

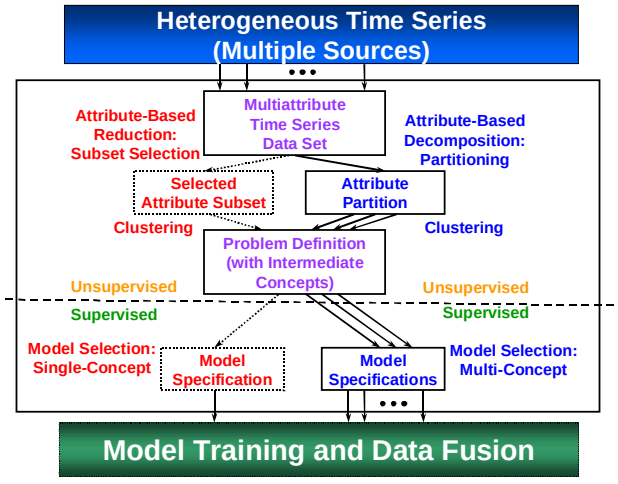


**Figure 3. Role of recurrent mixtures in data mining**

The synthesis of a new group of attributes (also known as the *feature construction* problem) in inductive concept learning is an optimization problem. Its control parameters include the attributes selected as relevant [22, 20], how they are grouped (with respect to multiple targets), and how new attributes are defined in terms of *ground* (original) attributes. This synthesis and selection problem is a key initial step in *constructive induction* [23] – the reformulation of a learning problem in terms of its inputs (attributes) and outputs (concept class descriptors).

Figure 3 illustrates the role of attribute selection and partitioning in a generic data mining process. In this framework, the input consists of *heterogeneous* data (that originating from multiple sources). Supervised learning is to acquire the performance element (time series classification [20] and other forms of pattern recognition that are important for decision support). Our applications include insurance risk valuation, precision agriculture, and record and document clustering for information retrieval.

## Conclusions

### Novel Contributions

The novel contribution of this architecture is that it uses newly-formulated intermediate training targets, which are discovered using input partitioning and clustering, in all mixture components. It extends traditional HME by making subtask specialization an explicit, unsupervised learning step, rather than an effect of interleaved training of gating (mixture estimation) and expert subnetworks.

### Current and Future Work

The experiment described in this paper illustrates the "best case scenario" in terms of learning task factorization. It is important, however, to note that no knowledge about problem structure (i.e., the intermediate targets) has been hardwired into the modular network. This structure is discovered using constructive induction, based on the input specification. Examples of knowledge-free algorithms that have been used to generate intermediate attributes in this fashion are Fu and Buchanan's hierarchical approach [29] and self-organizing feature maps [24]. Clustering and vector quantization algorithms such as *k*-means clustering [30], learning vector quantization (LVQ), radial basis functions (RBFs), and principal components analysis (PCA) are also applicable [30, 31]. Continuing research compares the performance of SM networks to that of other mixture models, especially EM-based ones. Future experiments are planned using variants of HME [1] and mixtures of factor analyzers [32, 33] and Gaussian processes [34]. Statistical learning theoretic properties (particularly generalization quality) of SM networks are also under investigation.

## References

[1] M. I. Jordan and R. A. Jacobs. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, 6:181-214, 1994.

[2] J. Fritsch, M. Finke, and A. Waibel. Adaptively Growing Hierarchical Mixtures of Experts. In *Advances in Neural Information Processing Systems 9: Proceedings of the 1996 Conference*, p. 459-465. MIT Press, Cambridge, MA, 1997.
[3] L. Breiman. Bagging Predictors. *Machine Learning*, 24:123-140, 1996.
[4] D. H. Wolpert. Stacked Generalization. *Neural Networks*, 5:241-259, 1992.
[5] T. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In *Proceedings of ICML-96*.
[6] D. Precup and R. S. Sutton. Multi-time Models for Temporally Abstract Planning. In *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, p. 1050-1056. MIT Press, Cambridge, MA, 1998.
[7] Y. le Cun, J. Denker, S. Solla, R. E. Howard, and L.D. Jackel. Optimal Brain Damage. In *Advances in Neural Information Processing Systems 2: Proceedings of the 1989 Conference*. Morgan-Kaufmann, San Mateo, CA, 1990.
[8] C. Neukirchen and G. Rigoll. Controlling the Complexity of HMM Systems by Regularization. In *Advances in Neural Information Processing Systems 11: Proceedings of the 1998 Conference*. MIT Press, Cambridge, MA, 1999.
[9] W. H. Hsu, S. R. Ray, and D. C. Wilkins. A Multistrategy Approach to Classifier Learning from Time Series. *Machine Learning, Special Issue on Multistrategy Learning*, to appear.
[10] S. R. Ray and W. H. Hsu. 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. October, 1998.
[11] R. A. Jacobs, M. I. Jordan, and A. G. Barto. Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks. *Cognitive Science*, 15:219-250, 1991.
[12] B. Stein and M. A. Meredith. *The Merging of the Senses*. MIT Press, Cambridge, MA, 1993.
[13] J. G. Rueckl, K. R. Cave, and S. M. Kosslyn. Why are "What" and "Where" Processed by Separate Cortical Visual Systems? A Computational Investigation. *Journal of Cognitive Neuroscience*, 1:171-186.
[14] N. A. Gershenfeld and A. S. Weigend. 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)*, A. S. Weigend and N. A. Gershenfeld, editors. Addison-Wesley, Reading, MA, 1994.
[15] M. C. Mozer. Neural Net Architectures for Temporal Sequence Processing. In *Time Series Prediction: Forecasting the Future and Understanding the Past (Santa Fe Institute Studies in the Sciences of Complexity XV)*, A. S. Weigend and N. A. Gershenfeld, editors. Addison-Wesley, Reading, MA, 1994.
[16] K. J. Lang, A. H. Waibel, and G. E. Hinton. A Time-Delay Neural Network Architecture for Isolated Word Recognition. *Neural Networks* 3:23-43, 1990.
[17] J. Principé and deVries. The Gamma Model – A New Neural Net Model for Temporal Processing. *Neural Networks*, 5:565-576, 1992.
[18] J. L. Elman. Finding Structure in Time. *Cognitive Science*, 14:179-211, 1990.
[19] J. Principé, C. Lefebvre. *NeuroSolutions v3.02*, NeuroDimension, Gainesville, FL, 1998. URL: http://www.nd.com.
[20] W. H. Hsu. *Time Series Learning With Probabilistic Network Composites*. Ph.D. thesis, UIUC. URL: http://www.ncsa.uiuc.edu/People/bhsu/thesis.html, August, 1998.
[21] K. Kira and L. A. Rendell. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-92)*, p. 129-134, San Jose, CA. MIT Press, Cambridge, MA, 1992.
[22] R. Kohavi and G. H. John. Wrappers for Feature Subset Selection. *Artificial Intelligence, Special Issue on Relevance, 97*(1-2):273-324, 1997.
[23] S. K. Donoho and L. A. Rendell. Rerepresenting and restructuring domain theories: A constructive induction approach. *Journal of Artificial Intelligence Research*, 2:411-446, 1995.
[24] T. Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, 78:1464-1480, 1990.
[25] T. Li, L. Fang, and K. Q-Q. Li. Hierarchical Classification and Vector Quantization With Neural Trees. *Neurocomputing* 5:119-139, 1993.
[26] Y-S. Hwang and S-Y. Bang, An Efficient Method to Construct a Radial Basis Function Neural Network Classifier. *Neural Networks*, 10(8): 1495-1503, 1997.
[27] J. W. Beauchamp, R. C. Maher, and R. Brown. Detection of Musical Pitch from Recorded Solo Performances. *In Proceedings of the 94th Convention of the Audio Engineering Society*, Berlin, Germany, 1993.
[28] R. Kohavi, D. Sommerfield, and J. Dougherty. 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.
[29] L.-M. Fu and B. G. Buchanan. 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, 1985.
[30] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, NY, 1973.
[31] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan College Publishing, New York, NY, 1994.
[32] G. Hinton. Towards Neurally Plausible Bayesian Networks. Plenary talk, *International Conference on Neural Networks (ICNN)*, Houston, TX, 1997.
[33] B. Frey. Personal communication. Beckman Institute, UIUC, unpublished, 1998.
[34] D. J. C. MacKay. Gaussian Processes: A Replacement For Supervised Neural Networks? Invited talk, *UAI-97*.