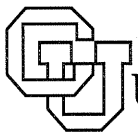
Nonlinear Gated Experts For Time Series: Discovering Regimes and Avoiding Overfitting

Andreas S. Weigend
Morgan Mangeas
Ashok N. Srivastava

CU-CS-798-95



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

University of Colorado Computer Science Technical Report CU-CS-798-95.
To appear in *International Journal of Neural Systems* Vol. 6 Nr. 4 (1995)
URL of this paper: ftp://ftp.cs.colorado.edu/pub/Time-Series/MyPapers/experts.ps.Z,
or http://www.cs.colorado.edu/~andreas/Time-Series/MyPapers/experts.ps.Z

Nonlinear gated experts for time series: discovering regimes and avoiding overfitting

Andreas S. Weigend
Department of Computer Science
and Institute of Cognitive Science
University of Colorado
Boulder, CO 80309-0430
andreas@cs.colorado.edu*

Morgan Mangeas
Electricité de France, Direction des Etudes et Recherches
1, av. du général de Gaulle, 92141 Clamart, France, and
Department of Computer Science
University of Colorado, Boulder, CO 80309-0430
morgan.mangeas@der.edf.fr

Ashok N. Srivastava
Department of Electrical and Computer Engineering
and Center for Space Construction
University of Colorado
Boulder, CO 80309-0529
srivasan@cs.colorado.edu

In the analysis and prediction of real-world systems, two of the key problems are nonstationarity (often in the form of switching between regimes), and overfitting (particularly serious for noisy processes). This article addresses these problems using *gated experts,* consisting of a (nonlinear) gating network, and several (also nonlinear) competing experts. Each expert learns to predict the conditional mean, and each expert adapts its width to match the noise level in its regime. The gating network learns to predict the probability of each expert, given the input. This article focuses on the case where the gating network bases its decision on information from the inputs. This can be contrasted to hidden Markov models where the decision is based on the previous state(s) (i.e., on the output of the gating network at the previous time step), as well as to averaging schemes over several predictors. In contrast, gated experts soft-partition the input space, only learning to model their region.

This article discusses the underlying statistical assumptions, derives the weight update rules, and compares the performance of gated experts to standard methods on three time series: (1) a computer-generated series, obtained by randomly switching between two nonlinear processes, (2) a time series from the Santa Fe Time Series Competition (the light intensity of a laser in chaotic state), and (3) the daily electricity demand of France, a real-world multivariate problem with structure on several time scales. The main results are (1) the gating network correctly discovers the different regimes of the process, (2) the widths associated with each expert are important for the segmentation task (and they can be used to characterize the sub-processes), and (3) there is less overfitting compared to single networks (homogeneous multi-layer perceptrons), since the experts learn to match their variances to the (local) noise levels. This can be viewed as matching the local complexity of the model to the local complexity of the data.

*http://www.cs.colorado.edu/~andreas/Home.html

1 Introduction

1.1 Different regimes with different noise levels: the need for gated experts

Conventional time series models are global models. They can be linear, assuming that the next value is a linear superposition of preceding values [Yule, 1927], or they can be nonlinear, conveniently described in the quite general language of neural networks with hidden units [Rumelhart et al., 1986, Lapedes and Farber, 1987]. Such single, global, and traditionally univariate models are well suited to problems with stationary dynamics.

However, the assumption of stationarity is violated on many real-world time series. An important sub-class of nonstationarity is *piece-wise stationarity* (also called stationarity by parts and multi-stationarity) where the series switches between different regimes. For example, regimes of electricity demand depend on the seasons, and regimes of financial forecasts depend on the economy (e.g., expansion and contraction, also called growth and recession) [Granger, 1994, Diebold and Rudebusch, 1996]. Although a single global model can in principle emulate any function, including regime switching, it is often very hard to extract such an unstructured, global model from the data. In particular, trying to learn regimes with different noise levels by a single network is a mismatch since the network will extract features that do not generalize well in some regime (local overfitting) before it has learned all it potentially could in another regime (local underfitting). A final motivation for different experts in different regions is that they can individually focus on that subset of input variables relevant for their specific region. This turns out to be particularly advantageous in modeling multivariate problems where different variables are important in different regimes.

Addressing these problems, we present a class of models for time series prediction that we call *gated experts*. They were introduced into the connectionist community as the *mixture of experts* [Jacobs et al., 1991], and are also called *society of experts* [Rumelhart et al., 1995]. We use the term gated experts for nonlinearly gated nonlinear experts. The input space can be split nonlinearly through the hidden units of the gating network, and the sub-processes can be nonlinear through the hidden units of the expert networks.

The basic idea behind gated experts is simple: Rather than using a global model, we try to learn from the data several local models, the experts. Simultaneously, we learn to split the input space. The problem is that the splitting of the input space is unknown—the only information available is the next value of the series. This requires blending supervised and unsupervised learning: the supervised component learns to predict the (observed) next value, and the unsupervised component discovers the (hidden) regimes. Since the only observable is the combination of the gate and the experts, many different ways of splitting the input space and fitting local models are possible. This trade-off between flexibility in the gates and flexibility in the experts is an important degree of freedom in this model class.

Summarizing, the key elements of gated experts are:

- nonlinear gate and experts,
- soft-partitioning the input space,
- adaptive noise levels (variances) of the experts.

In contrast to related work (e.g., [Hamilton, 1994, Jordan and Jacobs, 1994]) we allow the noise-level parameter associated with each individual expert to adapt separately to the data. In our experience, expert-specific variances are important for two reasons: to facilitate the segmentation (areas of different predictability are grouped together), and to prevent overfitting (different regimes are approximated with different accuracy, given by the granularity of the data). This is a new approach to the problem of overfitting.

1.2 Related Work

Gated experts have a solid statistical basis. This can be compared to prior connectionist work addressing segmentation of temporal sequences. [Elman, 1990] uses the size of the errors, and [Doutriaux and Zipser, 1990] use large changes

in the activations of the hidden units to indicate segmentation. [Levin, 1991] adds a set of auxiliary input units which encode a (discrete) state, set to fixed values in training (supervised) and estimated in testing. In this architecture, the single network has the difficult task of learning two potentially quite different mappings across the same set of units. Gated experts can also be compared and contrasted to connectionist architectures with local basis functions: whereas the architecture of radial basis functions [Broomhead and Lowe, 1988, Casdagli, 1989, Poggio and Girosi, 1990] does split up the input space into local regions (as opposed to global sigmoids), there is no incentive in the learning algorithm to find regions defined by similar structure, noise level, or dynamics.

In the time series community, the idea of splitting an input space into subspaces is not new. One of the first examples is the *threshold autoregressive* (TAR) model [Tong and Lim, 1980]. In contrast to gated experts, the splits there are very simple and ad hoc; there is no probabilistic interpretation. TAR models still are quite popular in economics and econometrics. Typically, a cut in one of the input variables is introduced, and in each subspace and separate hyperplane is fit. The constraint of continuity across the cut is introduced by hand (whereas it emerges naturally for gated experts). TAR models tend to be successful when there are relatively few data points available, $\mathcal{O}(100)$, too few to actually learn the splits. Splits are often made in an exogenous variable, such as the volatility [Engle, 1982, Bollerslev, 1986, Bollerslev et al., 1990]. A more flexible model of multivariate adaptive regression splines (MARS) [Friedman, 1991] has recently been applied to forecasting of financial data [Lewis et al., 1994].

Radial basis functions, TAR, and MARS models all define the state by proximity in the *observed* variables. What if the state is not directly observable? Such *latent* or *hidden* states were popularized in the econometrics community some five years ago [Hamilton, 1989, Hamilton, 1990]; they can be traced back to [Quandt, 1958]. However, expressed in connectionist language, these models do not have any hidden units: both the gate and the experts are linear. There are different sets of coefficients associated with each regime, but the functions are linear, and the Markov transition probabilities constant. To our knowledge, neither the double-nonlinear gated experts presented here, nor the flexible individual noise levels for the different regimes have been used in economics or econometrics [Granger and Teräsvirta, 1993, Hamilton, 1994].[1] However, the rigorous probabilistic interpretation of the linear regime switching models fully generalizes to the gated experts discussed here.

An important inspiration for our work has been the introduction of mixture models into the connectionist community by Jacobs, Jordan, Nowlan and Hinton (1991).[2] Subsequently, [Jordan and Jacobs, 1994] introduced the architecture of a *hierarchical* mixture of *linear* experts (with fixed widths). Recently, [Jordan and Xu, 1995] proved the convergence, and [Waterhouse and Robinson, 1995] applied the hierarchical mixture of linear experts to time series prediction of the sunspots [Weigend et al., 1990, Nowlan and Hinton, 1992] and to nonlinear regression on an example of noise heterogeneity [Weigend and Nix, 1994, Nix and Weigend, 1995]. [Xu, 1994] applied this architecture to two linear AR(2) processes (well suited for linear experts). Furthermore, gated experts can be compared to the approach developed independently by [Pawelzik et al., 1996]: (1) Gated experts automatically adjust the local noise levels of the experts in each regimes, whereas Pawelzik et al. externally adjust the global granularity of the approximation, and anneal it during learning. (2) Our gate is connected to the inputs, whereas their gate has no inputs: in their case, the partitioning of the input space can only happen indirectly. (3) Our segmentation is driven by local predictability (in the form of pattern-by-pattern local noise levels), whereas their segmentation is based on the assumption of switching after a certain number of steps (the errors are added for a certain number of steps, in the examples given in their paper with an acausal filter).

Finally, in the statistics literature, the use of mixture models goes back for more than a century. Pearson [Pearson, 1894] modeled the forehead size of crabs with a mixture of two Gaussians; he used the method of moments to estimate the parameters. The arrival of computers in the last half-century allowed the estimation of mixture models in a maximum likelihood framework. [Titterington et al., 1985] cite more than 130 applications of mixture models on real problems.

[1] The problem of estimating local noise levels is known in the statistics literature as *noise heterogeneity* [Seber and Wild, 1989]. They assume a specific, often rather stringent model for the expected value of the noise variance as a function of the input.

[2] Steve Nowlan suggested the application of Gaussian mixture models to time series analysis to us in 1991; the present article summarizes our work done since, and includes some of the results presented in invited talks at *IEEE Workshop on Neural Networks for Signal Processing* in 1994, at *Neural Networks in the Capital Markets (NNCM)* in 1994.

[Redner and Walker, 1984] review mixture density estimation as it was known a decade ago in the statistics community. One important difference with our work needs to be pointed out: all of the work discussed in that literature assumes that the parameters of the distribution of each component of the mixture (as well as the gating) are either independent of the inputs or–in the most flexible case–linear functions of the inputs. In contrast, we allow these relationships to be nonlinear.

1.3 Structure of this article

Section 2 discusses the assumptions gated experts make about the data-generating process and gives a mathematical and probabilistic interpretation of the architecture, the cost function and the search. Section 3 summarizes the performance of the architecture and learning algorithm on several time series problems. More specifically then, Section 4 discusses the instructive example of computer generated data with regime switches. Section 5 predicts and analyzes the laser data from the *Santa Fe Time Series Competition* [Weigend and Gershenfeld, 1994], and Section 6 applies gated experts to predicting the electricity demand of France. Section 7 analyzes why gated experts help avoid overfitting and compares them with a method developed to predict local error bars [Weigend and Nix, 1994, Nix and Weigend, 1995]. Section 8 summarizes the usefulness of gated experts for time series analysis.

2 Theory of Gated Experts

This section describes the ingredients needed to specify the model: the wiring (network architecture and activation functions), the cost function (reflecting the error model and the priors, both on the parameters and on outputs), and the search algorithm that adjusts the parameters such that the cost function is reduced.

2.1 Architecture

The goal is to create an architecture that allows (and encourages) a "carving up" of the input space between experts. Throughout this article we do not assume hard decisions, but we allow a given pattern to be assigned softly to several regimes. (We will see in Eq. 21 that the price to be paid for using more than one expert is proportional to the entropy of distributing the pattern across experts.) This goal of carving up the input space should be compared to the simple averaging of different "experts." In contrast to such an *additive model* where the weights of the individual predictors are fixed (independent of the input), the outputs of the gating network in our *mixture model* vary dynamically with the input. This allows the experts to specialize, i.e., to learn only the areas where they are responsible, whereas simple averaging requires all sub-models to be equally responsible over the entire space. As an additional benefit, this carving up lets us characterize the data in terms of the regime. In time series problems, where there is a linear ordering of patterns, this can be visualized particularly nicely by analyzing the output of the gating network as a function of the time of the series. Note that extracting regime information does not sacrifice prediction accuracy. In contrary, we can obtain better predictions since the experts can truly be experts in their region, as opposed to covering everything mediocerly.

The part of the model that does the carving up of the input space (i.e., assigns the patterns to the experts) is called the gate. Upon what information can the gate base its decision? The are two choices: *external* information, presented in the form of inputs to the gating network, and *internal* information, extracted from the network's estimate of the present state. This paper focuses on the first case where the gate bases its decision solely on the external inputs. In the daily energy prediction problem, for example, the gate is not only given information about external variables such as temperature and cloud coverage, but also information about the day of the week, and the proximity to a holiday. When applying gated experts to financial problems, inputs to the gating network include not only variables that are useful for determining the regime, such as empirical volatilities, interest rate differentials, but also quantities that capture market sentiment and trader behavior.[3]

[3]In applying gated experts to financial markets, we also found it useful to allow for some differentiation of the inputs to the experts, e.g., some experts are given primarily technical indicators whereas other are given primarily fundamental quantities, and some experts are given information important in a trending market, whereas others specialize in mean reversion.

In the second case of no connections between input and gate, the gate bases its decision on its estimate of the previous state(s). If this mapping is linear (i.e., described by a transition matrix between these hidden states), this architecture is identical to a hidden Markov model (HMM) [Poritz, 1988, Rabiner, 1989, Baldi and Chauvin, 1994, Fraser and Dimitriadis, 1994, Ivanova et al., 1994, Nadas and Mercer, 1996]. The conceptual generalization to HMMs with nonlinear transition functions is straightforward: The transition matrix is replaced by a network with hidden units. Note, however, that HMMs split the input space only indirectly, whereas the gate here is directly connected to the inputs. This direct connection can facilitate the splitting. The information from external inputs and from the previous internal state of the gating network can also be combined: [Cacciatore and Nowlan, 1994], in discussing control problems, feed the output of the gating network back to the input of the gating network at the next time step, providing the gating network with the information of the last state, and recently, [Bengio and Frasconi, 1995] and [Predoviciu and Jordan, 1995] presented a framework for an "input-output hidden Markov model" that allows the gate to access both external information (from the inputs) and its own past outputs.

Figure 1 shows the architecture of gated experts, consisting of K experts and one gating network. Both the experts and the gating network have access to the inputs; they can either share the same inputs, or be given different sets of inputs. The task of each expert is to approximate a function over a region of its input space. The task of the gating network is to assign one expert to each input vector. (Ultimately, the goal is to have only one expert for each pattern. However, this is a soft constraint; by paying a price, the entropy of distribution over the experts of that pattern, more than one expert can be gated in.) In time series prediction, the only teacher signal directly available is the next value of the series—the splitting of the input space is not known. To solve the problem, we need to blend supervised and unsupervised learning: The supervised component learns to predict the (observed) next value, and the unsupervised component discovers the (hidden) regimes.

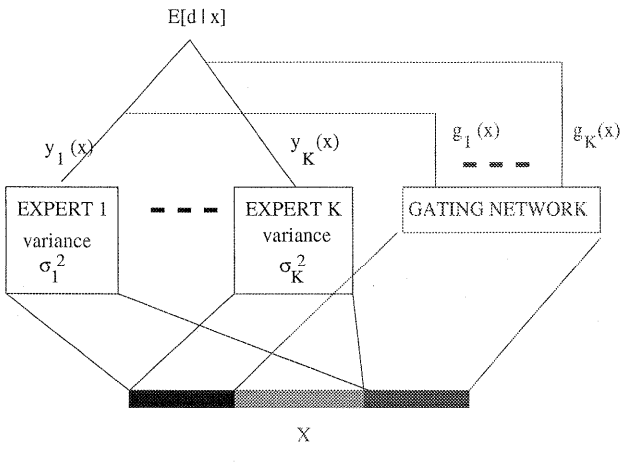


Figure 1: Architecture of gated experts. $\mathbf{x}$ denotes the inputs, drawn at bottom of the figure. Some of the inputs are connected only to the experts, some only to the gate, and some to both. Each box is a nonlinear neural network with tanh hidden units. The gating outputs, $g_j(\mathbf{x})$, weight the expert outputs, $y_j(\mathbf{x})$; the overall expected value is $\sum_{j=1}^{K} g_j(\mathbf{x}) y_j(\mathbf{x})$. This product of two network outputs represents a different function class with a different representational bias than a single network.

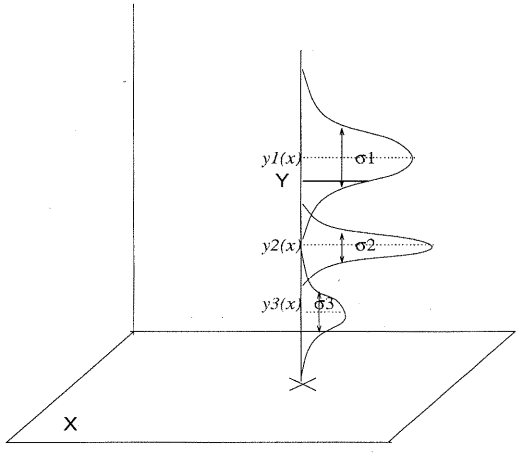


Figure 2: Probability density function of the output variable Y given by a mixture of Gaussians, plotted for an input $\mathbf{x}$. The outputs of the experts, $y_j(\mathbf{x})$, determine the centers of the Gaussians and vary with the location in input space. The widths, σ_j, are independent of the input. The three Gaussian densities sketched are weighted with their corresponding gating outputs, g_j. This is a snapshot early in training, before a clean segmentation has emerged.

In more detail, expert j learns a function $y_j(\mathbf{x})$, implemented as standard neural network with a linear output unit and tanh hidden units. With x_i denoting the ith input, the activation ξ_h of the hth hidden unit is given by

$$\xi_h = \tanh\left(\sum_i w_{hi} x_i + c_h\right) \quad ; \tag{1}$$

the weights w_{hi} and offsets (also called biases) c_h are the model parameters to be estimated. The specific choice of tanh nonlinearities as internal building blocks is less important than the choice of the architecture and, in particular, the choice of the cost function that reflects the underlying assumptions about the switching, the error distributions, and the priors about activations and weight values.

The output y_j can be interpreted as a parameter of a conditional target distribution. For example, if we use Gaussians as building blocks for the distribution, y_j corresponds to the mean of the Gaussian.[4] The other parameter of a Gaussian, its width σ_j, is a property of the expert; it does not depend on the specific input vector, but it adapts during learning to the noise level in the regime of the expert (see Eq. 23). Fig. 2 sketches the mixture of three Gaussians whose centers y_j depend on the input; at a different location in input space, the centers of the Gaussians will be at different locations, and they will be weighted differently, but their widths will be the same.

The gating network has one output for each expert; $g_j(\mathbf{x})$ denotes its value. Its goal is to estimate the probability that a given input $\mathbf{x}$ was generated by expert j. The hidden units of the gating network have tanh activation functions as building blocks for the nonlinearities (Eq. 1). The outputs of the gating network are *normalized exponentials* (also called "softmax"-units) [McCullagh and Nelder, 1989, Bridle, 1989]. This choice incorporates into the architecture the constraints that the outputs should be positive and sum to unity. Normalized exponentials combine the hidden unit activations ξ with a weight vector $\mathbf{w}_j$ for each output $(j = 1, \cdots, K)$ into an intermediate activation

$$s_j = \sum_h w_{jh}\xi_h + c_j \quad , \tag{2}$$

where the sum extends over the hidden units. The s_j are then exponentiated and normalized by their sum, giving the final output

$$g_j = \frac{\exp(s_j)}{\sum_{k=1}^{K} \exp(s_k)} \quad . \tag{3}$$

The gating network generates K mutually completing probabilities as a function of the input $\mathbf{x}$. The built-in constraint that the gating outputs sum to unity implements the competition between the experts. This competition is soft: The outputs can take any value between 0 and 1, and, for points close to the boundary between two regimes, the entropy of having several experts active for a given input is traded off against the reduction in error, as we will see in Eq. 21 below.

2.2 Cost function

Recall that we do not know the regimes when we start out—they are hidden variables. This implies that we cannot use simple supervised learning, but need to employ a more general framework to derive a cost function that incorporates our beliefs. Using the statistical framework of maximum likelihood allows us to obtain a cost function.

We begin by defining the variables we use:

- $\mathbf{x}$ is the input vector
- d is the target (or "desired output value")
- $y_j(\mathbf{x})$ is the output of expert j (corresponding to the mean of the Gaussian)[5]
- σ_j is the width of the Gaussian represented by expert j

[4] If there is only a single expert and we assume a Gaussian error model with constant noise level (variance), then this is equivalent to minimizing the squared error between the output and the target value (as can be seen by taking the negative logarithm of the Gaussian) [Rumelhart et al., 1995]. If we allow the width of the Gaussian to become a function of the inputs (e.g., by adding a second output unit to the network to predict the local error bar), we obtain a model for estimating the local noise level [Nix and Weigend, 1995].

[5] In this article, we assume the output to be a scalar. The generalization to multivariate outputs with identity covariance matrix (times σ_j^2, the noise level for expert j) is trivial, both in terms of the update rules and in terms of the interpretation of each expert having a certain degree of predictability associated with it. A diagonal covariance matrix with different diagonal elements can still be done in this framework by pre-scaling the outputs. A full covariance matrix, however, is hard to interpret as a single predictability measure of that expert.

- $P(Y = d \mid \mathbf{x}, j)$ is the probability associated with the jth expert that the stochastic variable Y takes on the value d
- $g_j(\mathbf{x})$ is the output of the gating network, denoting the probability $P(j \mid \mathbf{x})$ that, given the input $\mathbf{x}$, the pattern is generated by the jth expert
- $h_j(\mathbf{x}, d, y_j)$ is the posterior probability of the jth expert, given the output y_j and the pattern
- j denotes the event that the pattern is generated by the jth expert ($1 \leq j \leq K$)
- t is the pattern index
- i is the iteration (in the search) index
- θ_j and θ_g denote the set of parameters for expert j and the gate, respectively.

Recall the goal of "carving up the input space," expressed at the beginning of Section 2.1: We now explicitly assume that *one and only one expert is responsible for each pattern*, i.e., we assume that the experts are mutually exclusive: the probability of two different experts generating a data point y given the input x is zero.[6] We can view the K experts as K ways of observing $Y = d$, given $\mathbf{x}$:

$$P(Y = d \mid \mathbf{x}) = P\left((Y = d \cap \text{expert} = 1) \cup (Y = d \cap \text{expert} = 2) \cup \cdots \cup (Y = d \cap \text{expert} = K) \mid \mathbf{x}\right) \quad (4)$$

where 'expert $= 1$' denotes the event that expert 1 is chosen, etc. This notation can be shortened by replacing 'expert $= j$' by 'j', and '$\cap$' by a comma. Then, assuming mutually exclusive probabilities and using Bayes' rule, we can write the overall probability as

$$P(d \mid \mathbf{x}) = \sum_{j=1}^{K} P(d, j \mid \mathbf{x}) = \sum_{j=1}^{K} P(j \mid \mathbf{x}) P(d \mid \mathbf{x}, j) = \sum_{j=1}^{K} g_j(\mathbf{x}) P(d \mid \mathbf{x}, j) \quad . \quad (5)$$

At first sight, Eq. 5 seems to contain a serious identifiability problem—there are infinite ways of writing one function as a product of two functions! However, some constraints are already imposed by the class of functions; e.g., the gating outputs g_j only take values between 0 and 1. Further constraints enter indirectly through the search and the specific architectures, e.g., different complexities or, in the case of recurrent networks, time constants in the gate and the experts. There is significantly more room for modeling and need for assumptions than in the case of a single feed-forward network. The "product model" (Eq. 5) represents a broader functional class than single networks.

So far we have dealt with probability distributions. If the goal is to obtain a single number as "the prediction," we take the expected value of the probability density. It is given by the linear combination of the expected values of the individual experts, $y_j(\mathbf{x}) = E\left[y \mid \mathbf{x}, j\right]$, weighted by the g_j's,

$$\text{weighted mean of the experts} = \sum_{j=1}^{K} g_j(\mathbf{x})\, y_j(\mathbf{x}) \quad . \quad (6)$$

The expected value is only a useful statistic if the overall distribution is more or less unimodal. If the assumption that each pattern was generated by a single expert is correct, the g's will become binary during learning. In that case, only one expert remains active for every pattern, and the goal of predicting a single point is well justified.

A good check is to observe the distribution of the g values. If they remain at intermediate levels, and if the corresponding y_j's are not close to each other, the model may be mis-specified, indicating a violation of the underlying assumptions. In that case, it can still be interesting to read off the entire conditional density, and compare it to a simple density estimation with Gaussians [Bishop, 1994], or to nonparametric density estimation using "fractional binning" [Weigend and Srivastava, 1995].

[6] If K events (A_i), $i \in \{1, 2, ..., K\}$, are mutually exclusive, then $P(A_i \cap A_j) = 0$ if $i \neq j$, and $P(A_1 \cup A_2 \cup \cdots \cup A_K) = \sum_{i=1}^{K} P(A_i)$. This case assumes that $P\left((Y = d \cap \text{expert} = i) \cap (Y = d \cap \text{expert} = j) \mid \mathbf{x}\right) = 0$ for $i \neq j$; $i, j \in \{1, 2, ..., K\}$.

In order to evaluate the likelihood of the data given the model—how well the model predicts the observed data—we need to assume a specific distribution for the measurement errors. Since this paper focuses on regression (continuous, unbounded output), a Gaussian error model is a reasonable choice. A Gaussian is defined by two parameters, its mean and its variance (and all higher moments vanish; it is the distribution with the highest entropy given the first two moments). Each expert represent a Gaussian density whose mean is given by the output of the expert; it varies nonlinearly with the input. Each expert also has its own variance, independent of the location in input space, but usually different from the other experts—we will see later that the variances can play a crucial role in organizing the experts along areas of similar predictability.[7] The probability of generating a value d by expert j is then proportional to

$$P(d \mid \mathbf{x}, \theta_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d - y_j(\mathbf{x}, \theta_j)\right)^2}{2\,\sigma_j^2}\right) \quad . \tag{7}$$

The parameters θ_j and the variance σ_j^2 characterize expert j. $y_j(\mathbf{x})$ depends on the input $\mathbf{x}$, whereas σ_j^2 does not. P is the probability density associated with the observation $Y = d$.

Assuming statistical independence of the measurement errors of each pattern (the superscript t enumerates the patterns, their total number is N) allows us to obtain of the full likelihood by taking the product over the likelihoods of the individual patterns:

$$\mathcal{L} = \prod_{t=1}^{N} P(Y = d^{(t)} \mid \mathbf{x}^{(t)}) = \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g)\, P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j) \tag{8}$$

$$= \prod_{t=1}^{N} \sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right) \quad . \tag{9}$$

The cost function $\mathcal{C}$ is the negative logarithm of the likelihood function,

$$\mathcal{C} = -\ln \mathcal{L} = \sum_{t=1}^{N} -\ln\left[\sum_{j=1}^{K} g_j(\mathbf{x}^{(t)}, \theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\,\sigma_j^2}\right)\right] \quad . \tag{10}$$

Having described the global probability model, we now need to estimate the parameters θ_g, θ_1, θ_2, $\cdots$, θ_K, σ_1, σ_2, $\cdots$, σ_K. This is done by minimizing the cost function $\mathcal{C}$ with respect to the parameters. In principle, this should be possible with standard gradient descent. However, it turns out to be quite hard to learn at the same time both the individual maps of the experts and the splits of the input space through the gating network. Furthermore, the sum inside the logarithm makes the cost function significantly more complicated than in the case of a single network.

2.3 Search

The cost function derived in the previous section (Eq. 10) is difficult to minimize with gradient descent. However, we can reformulate the problem such that it allows us to apply the *Expectation-Maximization* algorithm (EM, [Dempster et al., 1977, Nowlan, 1991, Tanner, 1993, Hamilton, 1994, Jordan and Xu, 1995]). This algorithm is based on the assumption that some variables are missing (or "hidden"). To map the problem onto EM, we need to identify the missing variables: We choose the probabilities that a given pattern t was generated by expert j. We use the trick of introducing an "indicator variable"

$$I_j^{(t)} = \begin{cases} 1 & \text{if pattern } t \text{ is generated by the } j\text{th expert} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

[7] For classification problems, the usual choice is a binomial distribution [McCullagh and Nelder, 1989, Rumelhart et al., 1995]. A binomial distribution is described by a single parameter (the mean and the variance are related). The lack of an independent order-parameter such as the variance of the Gaussian independent of the mean makes it difficult for a model with binomial error distribution to carve up the input space based on the degree of predictability. A more appropriate choice for a general classification model is a Beta distribution, since its mean and width can be chosen independently.

The set of random indicator variables $\mathcal{Z} = \{I_j^{(t)}; j = 1, ..., K; t = 1, ..., N\}$ constitute the missing data. Since the indicator variables are binary, they filter out all but the "true" term. Note that this assumption of one expert being responsible for each pattern is identical to the assumption that allowed us to write Eq. 5 above.

Summarizing, there are two reasons for using the indicator variables: They embed the assumption of mutually exclusive experts, and they allow us to replace the awkward *sum* over experts inside the product in Eq. 9 by a *product*. After taking the logarithm, we end up with a simple double-sum, yielding a much more tractable log-likelihood function.

Now, the problem is that we do not know the values of I_j. This is where the two-step EM algorithm comes in. In the first step (the E-Step), we compute the expected values for I_j (assuming that all the network parameters are known). And in the second step (the M-Step), we update the parameters of the model (assuming that the I_j's are known: we just take the values computed in the E-step).

2.3.1 The E-step

We assume that the likelihood of the "complete data" is given by

$$P(\mathcal{D}, \mathcal{Z} \mid \mathcal{X}, \Theta) = \prod_{t=1}^{N} \prod_{j=1}^{K} \left[g_j(\mathbf{x}^{(t)}, \theta_g)\ P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j) \right]^{I_j^{(t)}} , \tag{12}$$

where $\mathcal{X}$ and $\mathcal{D}$ denote the input and target training data (constituting the observed data), $\mathcal{Z}$ the unobserved or missing data, and Θ the ensemble of parameters, $\Theta = (\theta_g, \theta_1, \theta_2, \cdots, \theta_K, \sigma_1, \sigma_2, \cdots, \sigma_K)$.

Now the key feature of the EM algorithm enters: it allows us to replace the missing variables $I_j^{(t)}$ by their expected values. The computation of these expected values $h_j^{(t)}$ is called the **E-step**:

$$h_j^{(t)(i)} := E\left[I_j^{(t)} \mid \mathcal{X}, \mathcal{D}, \Theta^{(i)} \right] = P(j \mid \mathbf{x}^{(t)}, d^{(t)}) \tag{13}$$

$$= \frac{P(j, d^{(t)} \mid \mathbf{x}^{(t)})}{P(d^{(t)} \mid \mathbf{x}^{(t)})} = \frac{P(j \mid \mathbf{x}^{(t)})\ P(d^{(t)} \mid \mathbf{x}^{(t)}, j)}{P(d^{(t)} \mid \mathbf{x}^{(t)})} \tag{14}$$

$$= \frac{g_j(\mathbf{x}^{(t)}, \theta_g^{(i)})\ P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_j^{(i)})}{\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)}, \theta_g^{(i)})\ P(d^{(t)} \mid \mathbf{x}^{(t)}, \theta_k^{(i)})} . \tag{15}$$

The superscript i denotes the iteration number; we iterate back and forth between the E-step and the M-step. (We here include the iteration number i explicitly, but suppress it whenever possible.)

Note that there are two probabilities, g and h. The gating network output $g_j(\mathbf{x}^{(t)})$ is based solely on the knowledge of $\mathbf{x}$, not on knowledge of y or d. In contrast, the posterior probability $h_j(\mathbf{x}^{(t)}, d^{(t)}, y_j(\mathbf{x}^{(t)}, \theta_j))$ includes knowledge about how close expert j's output $y_j(\mathbf{x}^{(t)}, \theta_j)$ is to the observed value $d^{(t)}$.

So far, the E-step has not assumed any specific distribution. In order to actually compute h, we need to use a specific measure of "closeness" between y and d, i.e., specify the error model. Consistent with Eq. 7, we assume that the experts output Gaussian densities. This allows us to express h through g, d, and y_j (as well as through the parameters σ and θ):

$$h_j^{(t)} = h_j\left(\mathbf{x}^{(t)}, d^{(t)}, y_j(\mathbf{x}^{(t)}, \theta_j), g_j(\mathbf{x}^{(t)}, \theta_g)\right) \tag{16}$$

$$= \frac{g_j(\mathbf{x}^{(t)}, \theta_g)\ \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j)\right)^2}{2\sigma_j^2} \right)}{\sum_{k=1}^{K} g_k(\mathbf{x}^{(t)}, \theta_g)\ \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\left(d^{(t)} - y_k(\mathbf{x}^{(t)}, \theta_k)\right)^2}{2\sigma_k^2} \right)} . \tag{17}$$

These h's are the result of the E-step; they will be used in the M-step.

2.3.2 The M-step

Taking the expectation of the negative logarithm of Eq. 12, and replacing each I_j by its expected value h_j yields the cost function that includes the assumption of missing values (hence the subscript M):

$$\mathcal{C}_{\mathrm{M}} = -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \ln\left[g_j(\mathbf{x}^{(t)},\theta_g)\, P(d^{(t)} \mid \mathbf{x}^{(t)},\theta_j)\right] \tag{18}$$

$$= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \ln\left[g_j(\mathbf{x}^{(t)},\theta_g)\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{2\sigma_j^2}\right)\right] \tag{19}$$

$$= -\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \left[\ln\left(g_j(\mathbf{x}^{(t)},\theta_g)\right) - \frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{2\sigma_j^2} - \frac{1}{2}\ln 2\pi\sigma_j^2\right] \tag{20}$$

$$= \sum_{t=1}^{N}\sum_{j=1}^{K} -h_j^{(t)} \ln\left(g_j(\mathbf{x}^{(t)},\theta_g)\right) + \frac{1}{2}\sum_{t=1}^{N}\sum_{j=1}^{K} h_j^{(t)} \left[\frac{\left(d^{(t)} - y_j(\mathbf{x}^{(t)},\theta_j)\right)^2}{\sigma_j^2} + \ln\sigma_j^2 + \ln 2\pi\right] . \tag{21}$$

The first term in Eq. 21 describes the cross-entropy between g and h. It can be interpreted from two angles. In a 1-of-K classification framework, it is the standard cost function [Rumelhart et al., 1995]. However, if we assume that the g learns to approximate h, it can also be viewed as $(-\sum_j p_j \ln p_j)$ (with $p = g$ or h), i.e. as the entropy of distributing a pattern across the experts. This cost is zero if one p_j is unity and the others are all zero. When experts "share" a pattern, they have to pay an entropy price for it. This will happen if they jointly manage to reduce the other terms in the cost function by more than they pay for the entropy.

The second term in Eq. 21 is the (h_j-weighted) *actual* squared error, $(d-y_j)^2$, expressed in units of the *average* squared error of that expert, σ_j^2, plus the cost associated with the average error. (The larger the log of the average squared error, or the log of the width of the Gaussian, the more expensive.) The weighting with the posterior probability h_j implies that if another expert is doing a lot better than expert j on a given pattern, expert j's large error does not matter, since it will be suppressed by a very small h_j. Only when the h_j's are significantly different from zero, will the performance of expert j enter into the cost. This fact drives the specialization of the experts (or the carving up of the input space). Note that this is different from simple averaging of different predictors, where we can only hope for reduction of statistical error to the degree that the individual predictors are uncorrelated [Perrone, 1994, Jacobs, 1995]. The cost function Eq. 21 is central for the gated experts model.

The **M-step** minimizes this cost function by adjusting the parameters. The parameters under consideration are the variances of the experts, as well as the weights of all the experts and the gating network.

The **updates for the variances** can be computed directly by setting $\partial C_{\mathrm{M}}/\partial\sigma_j^2$ to zero. The corresponding value of $\sigma_j^{2(i)}$ at iteration i that minimizes ∂C_{M} is then given by

$$\sigma_j^{2(i)}(\text{no prior}) = \frac{\sum_{t=1}^{N} h_j^{(t)(i)} \left(d^{(t)} - y_j(\mathbf{x}^{(t)},\theta_j^{(i)})\right)^2}{\sum_{t=1}^{N} h_j^{(t)(i)}} . \tag{22}$$

The sum extends over the patterns t. The variance of the jth expert is the weighted average of the squared errors; the weight is given by $h_j^{(t)}$, the posterior probability that expert j generated pattern t. The denominator normalizes the weightings for that expert.

Consider the case that a certain expert j happens to only win a few patterns, and that it fits those really well (e.g., if the expert is too flexible and overfits the training patterns and consequently underestimates the noise level). [Nix and Weigend, 1995] use a cross-validation scheme where a subset of the data is used for the estimation of $\sigma^2(\mathbf{x})$ different from the subset used for the estimation of the network parameters. For gated experts with potentially only

a few patterns assigned to some expert, this scheme becomes statistically very unreliable; it strongly depends on the specific splits of the data into the different subsets [Weigend and LeBaron, 1994]. Instead of using the maximum likelihood update (Eq. 22) that does not incorporate a prior, we include our belief for the size of the variance into the update rule for σ_j^2:

$$\sigma_j^{2(i)} = \frac{\sum_{t=1}^{N} h_j^{(t)(i)} \left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j^{(i)})\right)^2 + \lambda \, \sigma_{0\,j}^2}{\sum_{t=1}^{N} h_j^{(t)} + \lambda} \quad . \tag{23}$$

$\sigma_{0\,j}^2$ is a prior variance whose value depends on the problem. For high-noise financial data, for example, we set it to the value that corresponds to the random walk hypothesis. For the low-noise laser data from the Santa Fe competition, we set it to the quantization error of the analog-to-digital converter.[8] λ reflects our belief in that prior value $\sigma_{0\,j}^2$. A value of $\lambda = 0$ reproduces the maximum likelihood case of Eq. 22. For very large λ, we can neglect the first term both in the numerator and in the denominator, and the variance becomes $\sigma_{0\,j}^2$, independent of the data. We determine the value for λ through a validation set. In this paper, we present gated experts in a maximum likelihood framework. The need to introduce terms such as $\lambda\sigma_{0\,j}^2$ shows the limitations of that framework. Gated experts can also be expressed in a MAP (maximum a posterior)-framework. Eq. 23 is an approximation to the general case of a Gamma prior [Waterhouse et al., 1996]. [9]

Since we use nonlinear hidden units, the weights of the networks cannot be computed directly but require iterative techniques. The **weight changes in the expert networks** are proportional to the difference between the desired value d and the expert output y_j:

$$\frac{\partial \mathcal{C}_{\mathrm{M}}^{(t)(i)}}{\partial y_j} = -h_j^{(t)(i)} \frac{1}{\sigma_j^2} \left(d^{(t)} - y_j(\mathbf{x}^{(t)}, \theta_j^{(i)})\right) \quad . \tag{24}$$

This learning rule adjusts the parameters such that the expert output y_j moves towards the desired value d. However, note the two factors in front of the usual difference between desired and predicted value:

- The first factor, $h_j^{(t)}$, modulates the weight change proportional to the importance of expert j for pattern t.
- The second factor, $1/\sigma_j^2$, modulates the learning according to the general noise level in the regime of expert j. If the average squared error in the regime is large, the influence of the error on the weight update is scaled down. If the regime is believed to have only little noise, small differences in $(d - y_j)$ are exaggerated by dividing by a small number. This can be interpreted as a form of "weighted regression," increasing the effective learning rate in low-noise regions and reducing it in high-noise regions. As a result, the network emphasizes small errors in low-noise regions (low σ^2), and discounts learning patterns for which the expected error is going to be large anyway (large σ^2).
- The third factor is the usual difference between the desired value and the target; all other things being equal, the weight change is proportional to that difference.

The **weight changes in the gating network** are proportional to the gradient of the cost function with respect to the intermediate variable s_j (prior to exponentiation and normalization in the "softmax" part, see Eq. 2. As usual, the activation function is chosen such that the update rules become simple [McCullagh and Nelder, 1989]):

$$\frac{\partial \mathcal{C}_{\mathrm{M}}^{(t)}}{\partial s_j} = -\left(h_j^{(t)} - g_j(\mathbf{x}^{(t)}, \theta_g)\right) \tag{25}$$

[8] An alternative that also worked for the laser data is to introduce a lower bound for σ^2, set to the experimental resolution of the analog-to-digital converter or of 1 bit out of 8 bits. This hard limit corresponds to the assumption of a prior distribution for the variance that is flat above the cut-off and zero below the cut-off. Choosing an appropriate prior is an important part of modeling, particularly for short and noisy data sets.

[9] In our case of nonlinear experts, we do not subtract in the denominator the effective number of well-determined parameters of expert j, as [Waterhouse et al., 1996] do in their treatment of linear experts. We thank Steve Waterhouse for pointing this out.

The parameters in the gating network are adjusted such that $P(j \mid \mathbf{x}) = g_j(\mathbf{x}^{(t)}, \theta_g)$ gets pulled toward $P(j \mid \mathbf{x}, d) = h_j^{(t)}$. Recall the discussion of g and h following Eq. 15. h_j is the *posterior probability* of using the jth expert—its computation uses both input and output information. g_j is only a function of the input; it tries to approximate h_j without knowing the target value. In learning the g's move toward the h's; a scatter plot of g_j vs. h_j is a good diagnostic.

We close this section with three remarks: on priors for the weights and activations, on a modification that improves the segmentation for noisy data, and on the search algorithm employed.

2.3.3 Improving *generalization* through priors on the variances, the gating outputs, and the weights

In the context of Eq. 23, we discussed explicitly how to incorporate prior assumptions about the **variances of the experts**. The need for including priors instead of the direct maximum likelihood estimate (Eq. 22) arose from the problem that too few in-sample-patterns for one expert can be fitted too well by that expert, and the maximum likelihood estimate would be misleading. The need for a prior is at the heart of the general problem of induction: we want to build a model with the data we have, but that model should perform well on new data. The simplest case restricts σ^2 to a fixed range, appropriate for the hard limit of quantization noise in the laser example. Eq. 23 is a soft version, penalizing the deviation from $\sigma_{0\,j}^2$. Formally, this can be related to assuming $\ln(\sigma^2)$ to be drawn from a Gamma distribution [Waterhouse et al., 1996].

Beliefs about the **outputs of the gate**, g_j, are incorporated in a similar way. The simplest case uses a hard threshold for s_j (Eq. 2): If ($|s_j| \geq s_j \max$), the weights are not changed; this prevents them from growing too large. (Too large values for s_j are assumed to indicate overfitting: The gate should not be *that* sure about one expert being responsible for any pattern.) A soft version assumes that the g_j's are drawn from a Dirichlet distribution [Johnson and Kotz, 1972]. Expressed as a prior on s, this implies $\exp(s)$ to be Gamma distributed. For two experts, the Dirichlet distribution for g reduces to a Beta distribution (see Footnote 7).

The beliefs and hopes about the gating output enter as additional terms to the cost function or to the update rule (Eq. 25). E.g., we can discourage segmentation by making it cheap for the g_j to be close to their initial value of $1/K$, or we can encourage segmentation further by making large values of s cheap and effectively pushing the g's away from $1/K$, towards zero or one.

A third way of expressing prior beliefs is through the **weight values**. Treating all the weights independently, this again enters as an additional term into the update rules. In the simplest case, known as ridge regression in the statistics community, and introduced as *weight decay* to the connectionist community by Hinton and Le Cun in 1987, a term proportional to the present size of the weight is subtracted from the standard backpropagation weight change. Integrating this gives a cost proportional to w^2, reflecting the belief that all the weights are drawn from a Gaussian centered at zero. Although usually better than no penalty at all, the problem with weight decay for nonlinear networks is that it makes it hard for the network to develop significant nonlinearities since these can only be achieved through large weights.

Trying to avoid the strong bias towards linear models, we use the method of *weight-elimination*. It counts the significantly nonzero weights; the weights are assumed to be drawn from a distribution that is the sum of a uniform distribution (for the weights that should be present) and a Gaussian (for the weights that should be absent) [Weigend et al., 1990].

2.3.4 Improving *segmentation* through annealing

A further modification to the learning improves the segmentation implied by the outputs of the gating network. In both Eq. 22 and Eq. 23, the contributions of the individual patterns to σ_j^2 are weighted with the corresponding $h_j^{(t)}$'s. For noisy data, these h's will remain noisy throughout training. Consider the case that one data point lies in the center of regime j but happens to have a large observational error. Although it should have been assigned to regime j, the noise pushes down its h_j (and correspondingly, due to the competition, raises up some of the other h's). Thus other experts are given the blame for the bad fit and try to learn that point, although it is only due to the random output error. The

idea is to replace h_j in the maximum likelihood update rule for σ_j^2 by h'_j, a convex combination between h_j and g_j:

$$h_j'^{(t)} = (1 - \kappa_j)\, h_j^{(t)} + \kappa_j\, g_j^{(t)} \quad , \tag{26}$$

where $h_j^{(t)}$ is given in general by Eq. 15, and, assuming Gaussian noise, by Eq. 17.

What value should the new parameter κ_j take? We want it to depend on how well g approximates h. With $\Delta(g, h)$ expressing the proximity between g and h (e.g., as squared error or cross-entropy), we set

$$\kappa_j = e^{-\beta\, \Delta(g_j, h_j)} \quad . \tag{27}$$

β can be viewed as a knob that sets the scale for Δ; as usual, we can interpret β as inversely proportional to a temperature that is annealed during the learning process. It reflects a level of granularity for the segmentation, similarly to a level of granularity in clustering [Rose et al., 1990]. The annealing of β is discussed in [Srivastava and Weigend, 1995].

With β and Δ positive, κ_j lies between 0 and 1. (κ_j corresponds to the probability that the "observed" h_j's were generated by the g_j's, the outputs of the gating network.) At the beginning of learning, κ_j starts at a very small value since Δ is large; the g's have not learned anything yet, it would not make sense to consult them for the regime estimation. As training progresses, κ increases towards unity since Δ decreases. With larger κ, more weight is put on the smooth regime estimation from g, and less on the noisy one by h. This leads to more stable segmentation, particularly in high-noise problems where only a few experts are required.

2.3.5 Improving *learning time* through second-order methods

The nonlinear optimization in the M-step (of the weights in the gating network and the experts) can in principle be done through first-order gradient descent. In our experience, the learning of gated experts is significantly slower than in the case of a single network, due to the combined tasks of learning both the (unsupervised) segmentation and the (supervised) individual mappings. In the simulations reported here, we use a second order method, the Broyden-Fletcher-Goldfarb-Shanno algorithm (BFGS) as described in [Press et al., 1992]. It is a batch method: Once the entire learning set has been presented, it computes a descent direction as function of the first and second derivatives, and chooses the best step in this direction. One question remains, how many inner loop iterations within an M-step should be done before going back to the outer loop and re-estimating the h's. Given that if we go too far in the M-step, the last E-step estimate of the h's is no longer valid, and we learn the wrong things, and given that an E-step is cheap in computer time compared to an M-iteration, we tend to use only one or a few BFGS iterations in each M-step.

2.4 Comparison to other cost functions

We close this section by interpreting the cost function for gated experts by comparing it to related cost functions. For clarity, we suppress implicit dependencies on parameters and drop the sum over patterns (this could be called the "per-pattern cost function"). Assuming mutually completing experts, we started out with a mixture of Gaussians:

$$\mathcal{C}_{\text{Incomplete Data}} = -\ln \left[\sum_{j=1}^{K} g_j(\mathbf{x})\, \frac{1}{\sqrt{2\pi\sigma_j^2}} \exp\left(-\frac{\left(d - y_j(\mathbf{x})\right)^2}{2\,\sigma_j^2} \right) \right] \quad . \tag{28}$$

We then introduced missing variables, and replaced their distributions by their expected values $h_j = h_j(\mathbf{x}, d, y_j(\mathbf{x}), g_j(\mathbf{x}))$. We showed that h_j combines information from the input, the target, and the output y_j of expert j. In that sense it can be viewed as a posterior probability that a pattern (**x**, y) was generated by expert j. With these assumptions, we obtain the expected value of the complete data likelihood as the cost function for the M-step:

$$\mathcal{C}_{\text{M}} = \sum_j -h_j \ln g_j(\mathbf{x}) + \frac{1}{2} \sum_j h_j \left[\frac{\left(d - y_j(\mathbf{x})\right)^2}{\sigma_j^2} + \ln \sigma_j^2 + \ln 2\pi \right] \quad , \tag{29}$$

where M stands for M-step. The first term here can be viewed as cross-entropy between network output g and "target" h, as it would be in supervised classification (but here h is only the estimated target value for the hidden variables).

The first term can also be viewed as entropy of the experts, i.e., as a measure of "disorder" of the experts: It is cheapest if there is most order, when only one expert is fully responsible for the pattern, and all the others are off. The cost increases if more than one expert is gated in, and reaches is maximum if all K experts are evenly gated in with $1/K$, i.e., if an average over all experts is taken.

The expression in square brackets, weighting expert j by its relevance $h_j^{(t)}$ for pattern t, is identical to the cost function derived and discussed in [Nix and Weigend, 1995] for the case of predicting "local error bars" with a single network with two output units, one for the conditional mean, the other one for the conditional variance:

$$\mathcal{C}_{\mathrm{LEB}} = \frac{1}{2}\left[\frac{\left(d - y(\mathbf{x})\right)^2}{\sigma^2(\mathbf{x})} + \ln\sigma^2(\mathbf{x}) + \ln 2\pi\right] \quad , \tag{30}$$

where LEB stands for *local error bars*. This architecture is more complicated in that the variance $\sigma^2(\mathbf{x})$ is an explicit function of the input, and it is more simple in that there is no gating network. Eq. 30, and the square bracket in Eq. 29 share the trade-off between the two terms containing σ^2. The squared-error term could be made small by a large value of σ^2, but the cost increases logarithmically with σ^2.

Finally, simplifying the cost function further by assuming σ^2 to be constant reduces it to standard *least mean squares*:

$$\mathcal{C}_{\mathrm{LMS}} = \frac{1}{2}\left[\frac{\left(d - y(\mathbf{x})\right)^2}{\sigma^2} + \ln\sigma^2 + \ln 2\pi\right] \quad . \tag{31}$$

If we are only interested in finding the minimum, dropping all constants from Eq. 31 is equivalent to minimizing the squared error, $\left(d - y(\mathbf{x})\right)^2$.

All these cost functions are derived in a maximum likelihood framework. In all cases, we incorporate prior assumptions (about the outputs, the variances, and the weights) by adding appropriate terms to the cost functions, as discussed specifically in Section 2.3.3, see also [Weigend et al., 1990, Buntine and Weigend, 1991, Waterhouse et al., 1996].

3 Overview of the Experiments

The remainder of this article compares the performance of gated experts to standard methods on three time series: a computer-generated series, obtained by randomly switching between two nonlinear processes (Section 4), a time series from the Santa Fe Competition (the light intensity of a laser in chaotic state (Section 5), and the daily electricity demand of France, a real-world multivariate problem with structure on several time scales (Section 6). These three data sets try to cover time series problems along a few axes:

- from artificial (computer generated), through laboratory (laser), to real-world (electricity);
- from one time-scale (computer generated), through two time scales (laser individual oscillations and collapses), to multi-scale (days, weeks, years);
- from deterministic chaos (laser), through mix of chaos and stochastic behavior (computer generated), to non-chaotic but noisy behavior;
- from low-noise (laser), through mixed dynamics (computer generated), to high noise (electricity);
- from a clearly defined number of regimes (computer generated), to no prior knowledge of regimes (laser and electricity).

On these quite different problems, we consistently obtain the following results:

- gated experts yield significantly better results than single networks;
- gated experts discover the regimes correctly when there is a true segmentation to compare it to, and they come up with plausible segmentation when there is no true segmentation;

- gated experts allow us to characterize the sub-processes through their variances;
- gated experts show less overfitting than single networks (homogeneous multi-layer perceptrons) due to the correct matching of the noise level.

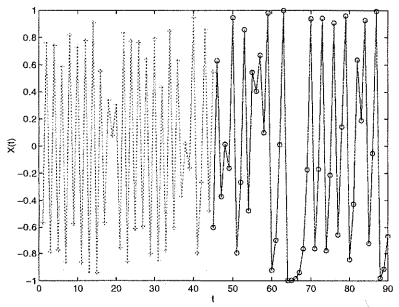


Figure 3: Segment of the computer generated time series. Left half: noisy tanh map (Eq. 33). Right half: quadratic map (Eq. 32).

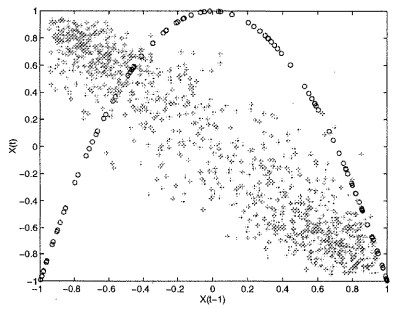


Figure 4: Two-dimensional return plot. '+' denotes data from the noisy tanh process, 'o' from the quadratic map.

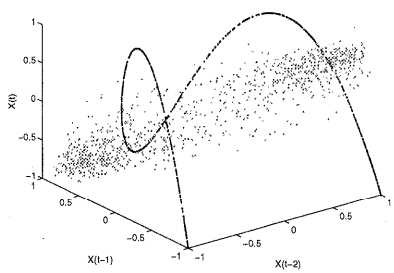


Figure 5: Three-dimensional return plot of the computer generated time series.

4 Computer-Generated Data

4.1 Data: Mixture of Two Processes

The first example is a computer generated toy problem where the data generation process matches the assumption of the architecture. There are two processes:

$$x_{t+1} = 2\,(1 - x_t^2) - 1 \quad \text{if switch} = 1 \tag{32}$$

$$x_{t+1} = \tanh(-1.2x_t + \varepsilon_{t+1})\ ;\ \varepsilon \sim \mathcal{N}(\text{mean} = 0, \text{var} = 0.1) \quad \text{if switch} = 0 \tag{33}$$

The first is a deterministic chaotic process; it is the quadratic map on the interval [-1,1]. The second is a noisy non-chaotic process; it is the composition of an autoregressive process of order one, with Gaussian noise of variance 0.1 (a relatively high noise level, standard deviation = 0.32), squashed though a hyperbolic tangent, to confine it to the same interval as the quadratic map. After this squashing, the effective noise level has an empirical variance of 0.071. Our error model assumes that it can still be modeled with a Gaussian. We chose the parameters to give a similar appearance in the time domain (Fig. 3), although their behavior in lag-space is very different, as evidenced by two- and three-dimensional return plots (Fig. 4 and Fig. 5, respectively).

These are the individual processes. The dynamics between the processes is governed by a first-order Markov process. The transition matrix of the probabilities from one process to the next is given by the values of 0.98 on the diagonal (i.e., the probability of staying in the same state is set to 0.98), and 0.02 off-diagonal (i.e., the probability of switching to the other process is set to 0.02). This corresponds to an average time between switches of 50 time steps. Only the expected value is known; the exact time when the switching occurs is random. In our experiments, we used 1,000 points for training and 1,000 for testing.[10]

[10] `http://www.cs.colorado.edu/~andreas/Time-Series/MyPapers/experts-demo.html` allows the reader to develop some intuitions about the learning behavior of the gated experts and the stability of the segmentation by running some of the experiments.

4.2 Architecture and Learning

The architecture consists of three experts[11] and one gating network. The goal is to forecast the next point, x_{t+1} (single-step prediction). Each expert has access to the past two values of the series as inputs, $\{x_{t-1}, x_t\}$. The gate has access to the last four values. (This is an example of thinning out the input space of the experts; the single networks used below for the comparison have access to all four past values.) Each expert has 10 tanh hidden units, the gating network has 20 hidden units. All output units are linear.

The 50 runs started with different initial weights. All of them converged to two experts surviving, and one expert not being used (i.e., its $g_j = 0$ for all patterns, see Fig. 6). All of them learned the splittings very well (Fig. 6), and the predictions for the switch were very similar to the real switches (except for the exact switching points which are randomly generated and thus unpredictable). Finally, the variance associated with the expert that found the quadratic process (Fig. 7) is very small ($\sim$0.001), limited by the prior we introduced (Eq. 23), and the variance associated with the expert that emulates the noisy process is close to the noise level of that process.

We compared the performance of the gated experts with that of a single network of varying numbers of hidden units (10, 20, 30, 40, and 50). In all cases, the gated experts give significantly better performance than any of the single networks. We express the performance in terms of *normalized mean squared error*,

$$E_{\text{NMS}} = \frac{\sum_{t \in \mathcal{T}} (\text{observation}_t - \text{prediction}_t)^2}{\sum_{t \in \mathcal{T}} (\text{observation}_t - \text{mean}_{\mathcal{T}})^2} \quad . \tag{34}$$

E_{NMS} compares the performance of the model on set $\mathcal{T}$ to simply predicting the mean on that set. For $\mathcal{T}$ as test set, we obtain the following results, expressed as gated experts / single network with 50 hidden units:

- Ratio of overall E_{NMS}: $0.026/0.031 = 83\%$,
- Excluding errors of the first two steps after each switch, ratio of E_{NMS}: $0.012/0.018 = 67\%$.

This is a significant improvement in performance. We also tried a single network with *two* layers of 10 hidden units each (about the same number of weights as the gated experts). Fig. 10 shows that the two-layer results are better than the one-layer results (since two hidden layers are better suited to the switching task than a single layer) but still not as good as the gated experts model. To guide the eye, the minimum of the gated experts is entered as a thin horizontal line at the same value in all three figures. Compared to the horizontal line, the minima on the test set of the two single networks are not as good as the minimum of the gated experts model.

Gated experts show only weak overfitting, for two reasons: the carving up of the input space, and the different values for the variances for different regimes. The evolution of the individual variances is shown in Fig. 7. In comparison to the weak overfitting the gated experts exhibit in Fig. 8, Fig. 9 shows a single network with one layer of 50 hidden units, and Fig. 10 shows a single network with two layers of 10 hidden units each, both trained by minimizing squared errors (Eq. 31) without allowing for different noise levels. This yields stronger overfitting: Trying to minimize the errors in the low-noise regime to the same degree as the errors in the high-noise regime constitutes a mismatch. Whereas the single network already overfits in the high-noise regime, it still underfits in the low-noise regime. These effects hold for single networks with numbers of hidden units as small as those of an individual expert as well as networks with numbers of hidden units as large as the entire gated expert architecture.[12]

[11] Although two experts would suffice for this series, we start with a larger number of experts since we do not want inject the knowledge of the number of experts.

[12] Furthermore, adaptive variances make the final model more interpretable. Running gated experts with the variances clamped to equal and fixed values (i.e., without updating them), gave comparably good performance, but the regimes were no longer discovered reliably.

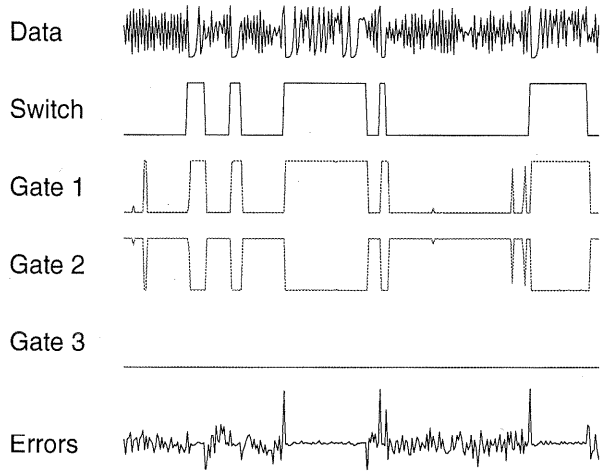


Figure 6: Gating outputs on the test set. Compare the true switch to the outputs of the gating network. Most of the time the outputs are binary. Note that Gate 1 discovers the hidden switch and Gate 3 is always zero.

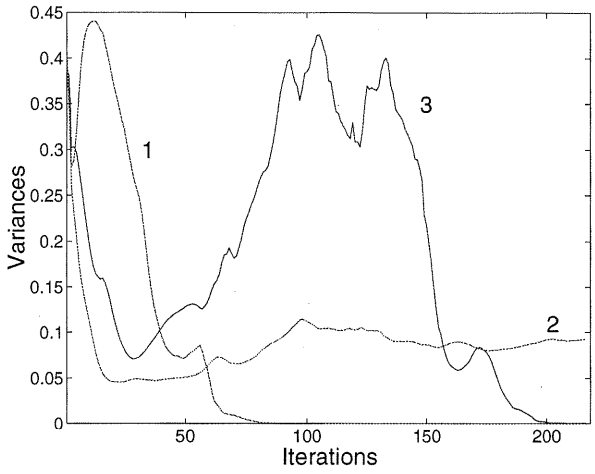


Figure 7: Evolution of the variances during training. Expert 1 learns the quadratic map; its variance reaches the minimum at iteration 70. Expert 2 learns the noisy tanh process and estimates its variance just below 0.1.

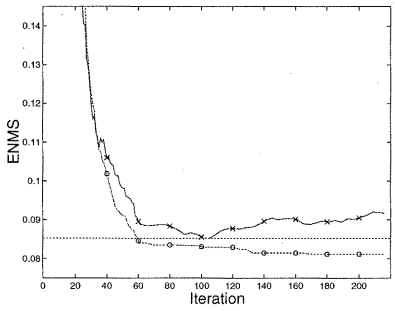


Figure 8: Learning curves for gated experts on the computer generated data. 'o' = training, '×' = test. The minimum of the test error is 0.086.

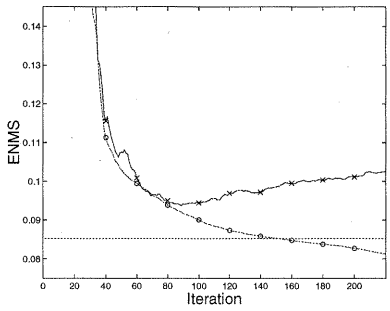


Figure 9: Learning curves for the single network with one layer of 50 hidden units. The minimum of the test error is 0.095.

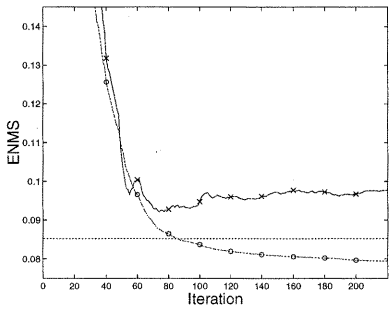


Figure 10: Learning curves for a single network with two layers of 10 hidden units each. The minimum of the test error is 0.092.

5 Laboratory Data

5.1 Data: Laser (Deterministic Chaos)

The second example applies gated experts to the laser time series from the *Santa Fe Time Series Prediction and Analysis Competition* [Weigend and Gershenfeld, 1994].[13] The laser is a stationary system on the time scale of the observations of the time series used here. Its behavior can be approximated reasonably well by a set of three coupled, nonlinear differential equations, the Lorenz equations. These equations are invariant under time shift. The noise level is very low; the main source of noise in the observed data is the quantization error of the analog-to-digital converter. Its dynamic range is 8 bits, corresponding to a signal-to-noise ratio of about 250:1. The collapses, interrupting areas of steadily growing envelopes, are shown in the top panel of Fig. 11. They are not caused by outside shocks but are part of the

[13]The volume by Weigend and Gershenfeld (1994) describes this data set as well as several attempts to predict and characterize it. The data is available by anonymous ftp at `ftp.cs.colorado.edu` in `/pub/Time-Series/SantaFe` as file `A.full`. More than thirty papers that use this data set have been found on the Internet to date; they can be accessed through `http://www.cs.colorado.edu/~andreas/TSWelcome.html`.

internal dynamics of this nonlinear system. The long-time behavior cannot be predicted due to the divergence of nearby trajectories, the hallmark of chaotic dynamics. We used 10,000 patterns for training (10 times the Santa Fe competition size), and 1,250 patterns each for two out-of-sample data sets, Test I and Test II.

5.2 Architecture and Learning

The goal is to forecast the next point, x_{t+1} (single-step prediction). The inputs to each expert are the 10 most recent values of the time series, $\{x_{t-9}, x_{t-8}, , ..., x_{t-1}, x_t\}$. (Given that the dimension of the laser data is around 2.1, fewer than ten inputs should suffice in principle. In our experience, however, it is easier to obtain good predictions by using more than the minimal number of inputs.) Each expert has a single layer of 5 tanh hidden units and a linear output unit. The gating network sees the same 10 inputs as the experts, and is equipped with one layer of 10 tanh hidden units, followed by the softmax outputs.

We do not know the optimal number of experts. Based on the size of the data set and our experience, we typically begin with 8 to 15 experts. In about half of the runs, we end up with 5 experts, in the other half, with 6 experts; the gating outputs for the remaining experts are zero for every pattern. Fig. 11 shows a run where 6 experts survive. The single-step E_{NMS} is equal to 0.0035 on the test set. In comparison, an architecture of eight gated *linear* experts gave 0.0045. We have not tried a mixture of linear experts.

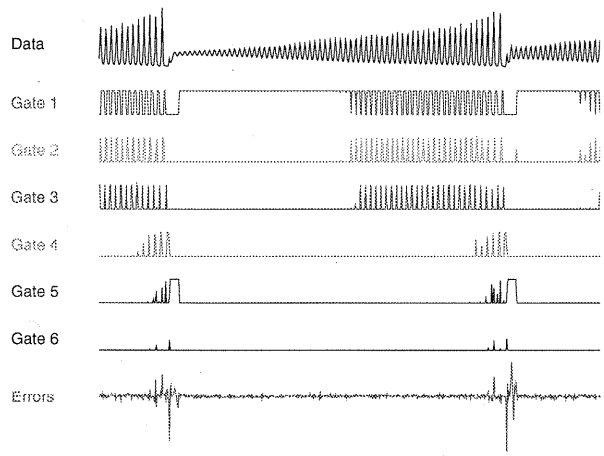


Figure 11: Outputs of the gating network on part of the test set. Note the binary behavior of the gates. Just after the collapses, the first expert remains gated in for almost half the time, until the amplitude exceeds a certain level.

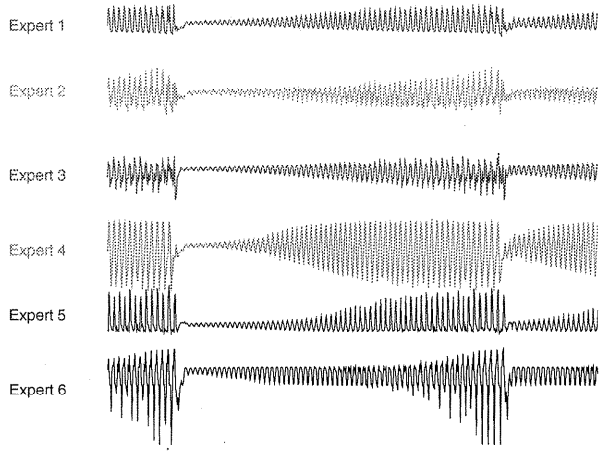


Figure 12: "Un-gated" outputs of the individual experts. Note the different behavior of experts 4 through 6, the experts for the collapses, in comparison to experts 1 through 3 which deal with the rest of the series.

5.3 Interpretation

Figure 11 shows that the gating network allocates experts 4, 5 and 6 for predicting and mapping the collapse. The others (1, 2 and 3) fit the rest of the series. Within these three experts, 1 by itself takes care of the small oscillations at the beginning of the growth period. When the oscillations get larger, expert 1 deals with the valleys and gets help from expert 2 and 3 for the peaks. Expert 4 comes in right before the collapse, and expert 5 is the post-collapse expert, taking care of the dynamics right after the re-injection. Expert 6, sometimes present, sometimes absent, basically takes the blame for a big error right at the collapse.

There are two types of segmentation: according to shape (peaks, valleys), and according to regimes (pre-collapse, collapse, post-collapse). Fig. 12 shows that expert 5 generates good forecasts for the collapse, expert 1 good forecasts for the post-collapse.

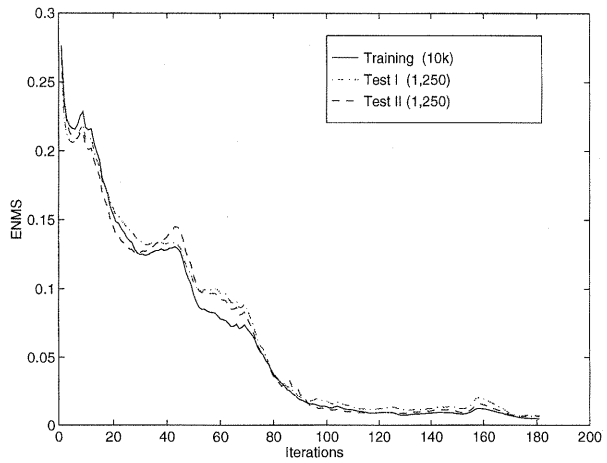


Figure 13: The squared error, E_{NMS}, as a function of iterations. There is no overfitting. The local increases in error indicate the phase separations (see text).

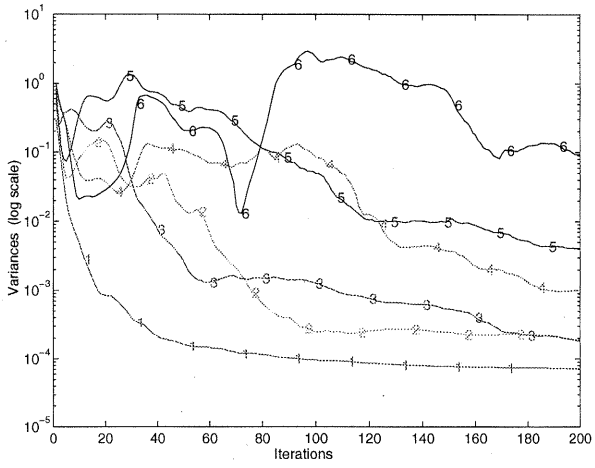


Figure 14: Evolution of the variances. The numbers correspond to the numbering given to the experts in Fig. 11 and Fig. 12.

Figure 13 shows the normalized mean squared error as a function of epochs. Note that there are several areas of increase in E_{NMS}. These can be viewed as **phase separations**, borrowing the term from metallurgy for processes where different regimes emerge during cooling. Such increases in E_{NMS} are possible: the M-step descends on the *entire* cost function (Eq. 29),

$$\mathcal{C}_{\text{M}} = \sum_j -h_j(\mathbf{x}, d, y_j(\mathbf{x})) \ln g_j(\mathbf{x}) + \frac{1}{2} \sum_j h_j(\mathbf{x}, d, y_j(\mathbf{x})) \left[\frac{(d - y_j(\mathbf{x}))^2}{\sigma_j^2} + \ln \sigma_j^2 + \ln 2\pi \right] \quad ,$$

whereas the squared error E_{NMS} analyzed here corresponds to the middle term only.

In our experience, the gate drives the variances: An increase in squared error implies a decrease in entropy, corresponding to the creation of a new phase which allows the experts to carve up the space better, and to become more binary. Continuing the analogy of phase separations, the system evolves from less ordered, mixed phases to more ordered, separated phases. These phase separations and the corresponding new roles of the experts imply changes in the variances, which can be clearly observed in Fig. 14 around iterations 10, 40 and 80. Note also that the variances at the end of the training span three orders of magnitude. The large values correspond to the experts that try to fit the collapse. This second example showed again that adaptive variances are crucial.

6 Real-World Data

6.1 Data: Electricity Demand of France

The third example is a real-world task: to predict the daily electricity demand of France. From a time series perspective, this series exhibits three interesting features:

- *multi-variate:* there are many inputs, encompassing both an endogenous variable (e.g., past electricity demand) and exogenous variables (e.g., temperature, cloud coverage, weather, etc.);
- *multi-scale:* there is structure on several time scales (e.g., daily patterns, weekly patterns, yearly patterns);
- *multi-stationary:* there are different regimes (e.g., holidays vs. workdays, summer vs. winter, etc.).

The data, task and performance are described in more detail in [Cottrell et al., 1995]; the gated experts approach is presented in [Mangeas et al., 1995]. To illustrate the gated experts approach, Section 6.2 focuses on the two-stage model that we developed to remove most of the non-stationarity present in the series, and Section 6.3 analyzes the splitting of the input space generated by the gated experts and shows how information about the process can be obtained through the task-directed clustering performed by the gated experts.

6.2 The Two-Stage Model

We split the task into two parts. The first model predicts the electricity demand from exogenous variables alone. We here use gated experts, i.e., build a conditional Gaussian mixture model whose parameters vary nonlinearly with the exogenous inputs. The second model is trained to predict the residual error of the first. The point of this two-stage architecture is that the residuals of the first model by effectively removing trends and drifts are more stationary than the raw data. We use gated experts for the first stage. It turns out that a single network suffices for the second stage.

X_t is the electricity demand at day t, $t = 1 \cdots N$ denotes the day, and $Z_t^1, Z_t^2, \cdots, Z_t^m$ the m exogenous variables available as inputs for Stage 1. p is the number of lags of the residual error of the first model used as inputs to the second model. We have

$$X_t = f_1(Z_t^1, Z_t^2, \cdots, Z_t^m) + \varepsilon_t \tag{35}$$

$$\varepsilon_t = f_2(\varepsilon_{t-1}, \varepsilon_{t-2}, \cdots, \varepsilon_{t-p}) + \eta_t \tag{36}$$

where ε_t and η_t are the residual errors for the first and second model, respectively. The prediction of the global model is given by

$$\widehat{X}_t = f_1(Z_t^1, Z_t^2, \cdots, Z_t^m) + f_2(\varepsilon_{t-1}, \varepsilon_{t-2}, ..., \varepsilon_{t-p}) \quad . \tag{37}$$

This is the expected value of the energy demand, conditioned on the inputs. It is obtained by setting η_t to its expected value of zero.

Stage 1 corresponds to regression onto the exogenous variable. The experts each had 1 layer of 5 tanh hidden units, and the gating network one hidden layer with 10 tanh units. The experts and gating network were connected to the same 51 exogenous inputs:

1–11: *Proximity to holidays.* If the current day is within 5 days of a holiday, the corresponding input of these 11 binary inputs is set to one. (Input 1 corresponds to a day 5 days before a holiday, input 2 to a day 4 days before a holiday, etc.).

12–27: *Task-independent, unsupervised clustering (Kohonen).* These inputs, based on a self-organizing map, are included since they are standard inputs into the model currently used.

28–34: *Day of the week.* 7 binary inputs denote the day of the week.

35–41: *Proximity to special tariff days.* EDF has certain days where special tariffs apply. If the current day is within 3 days of a such a day, the corresponding input of these 7 binary inputs is set to one. (Input 35 corresponds to Monday, etc.).

42–43: *Annual cycle.* 2 continuous [-1,1] inputs determine the position of the present day during the year (one sine, one cosine, with a period of $365/\pi$).

44-51: *Weather.* The remaining 8 inputs give the temperature as well as the degree of cloud coverage for the present day, the day before, the 1-day, and the 5-day differences.

This choice is based on the models currently used in France. No attempt was made to improve on this set of inputs. The role of the inputs for the different regimes will be considered in the context of Fig. 17.

We split the available data into two sets:

- Training set: January 1, 1988 through December 31, 1992 (1826 days)

- Test set: January 1, 1993 through March 1, 1994 (424 days).

We performed 10 runs with different initial sets of weights, always starting with 8 experts. The final number of "surviving" experts varied from 2 (three runs), 3 (five runs), to 4 (two runs). This variation may be due to the relatively small range of final variances of the different regimes (less than a factor of three between the low-noise and the high-noise experts).

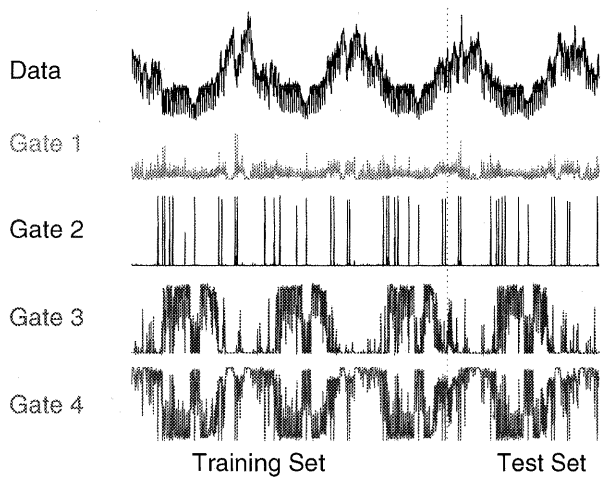


Figure 15: The data during the second half of the training period and the entire test period. The annual cycle, peaking in the cold winter months, is evident—and so is the summer vacation. Gate 2 picks out the holidays, gate 1 the days around holidays, gate 3 the warmer weather, and gate 4 the colder seasons.

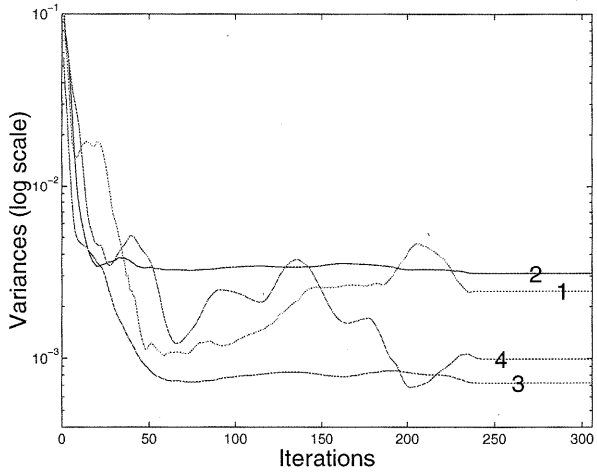


Figure 16: Evolution of the variances of the experts. At the end of training, expert 3 (summers) has the lowest variance, whereas expert 2 (holidays) has the largest variance. Experts 4 and 1 converge later than 2 and 3, and show some trade-off with each other before the final phase separation.

Stage 2. The second network has as input the residuals from the first stage, and its task is to predict the daily energy demand. (This is the same target as stage 1. However, stage 1 did not have any lagged energy-demand variables as inputs.) For the autoregression on the residual errors from the gated experts (stage 2), we use a single network with one hidden layer of eight tanh units. (It turned out that this is as good for the second stage as more complicated architectures.) Its input consisted of ten lags of the residual errors from Stage 1. For the final performance, we used standard pruning [Finnoff et al., 1993, Cottrell et al., 1995] to remove irrelevant weights. The basic idea is to compare the size of the weight with the size of its fluctuations (i.e., the standard deviation of the weight changes in response to the input patterns). If the size of the weight is small compared to these fluctuations, it will be removed. For the electricity prediction task, we typically removed some 35% of the weights. The resulting network squared error was 4% smaller on the training set than on the test set, indicating that there was essentially no overfitting after pruning.

6.3 Performance and analysis

We compare the performance of the combined stage 1 + stage 2 model to two benchmarks: a linear autoregressive model using exogenous variables (ARX), and a single neural network with two hidden layers [Cottrell et al., 1995]. In terms of squared error, the two-stage model gives an 11% improvement over the ARX model presently used. The performance of the single neural network falls between the two; it gives a 7% improvement over the ARX model.

We give here some examples of the interpretation of the gated experts. Fig. 15 shows part of the time series and the corresponding activations of the gates. Fig. 16 shows the evolution of the variances. The final ordering is consistent with the knowledge of the engineers that it is easier to predict the energy demand for standard summer days than for standard winter days, and it is particularly difficult to predict the energy demand on holidays (expert 2) and the days

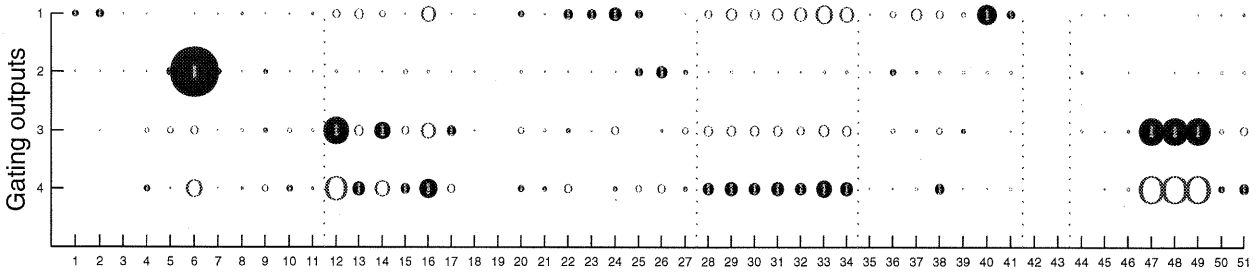


Figure 17: Linear correlations between the inputs and the outputs of the gating network. The numbers on the x-axis denote the input number; they are described in the text in Section 6.2. Filled circles denote positive correlation, open circles negative correlation. The size of the circles indicates the absolute value of the correlation. To give the scale, the correlation between input 6 and output 1 is 0.98; the correlation between input 47 and output 3 is 0.54.

around the holidays (expert 1), Fig. 17 visualizes the linear correlation coefficients between the outputs of the gating network and the 51 inputs. The largest correlation is between gate 2 and the input indicating a holiday (input 6). Inputs 28 through 34, the proximity to special tariff days, are positively correlated with gate 4, and negatively with gates 1 and 3. This is consistent with the fact that special tariffs exist only in the winter, and never in the vicinity of a holiday. A general feature is that gates 3 and 4 tend to have correlation coefficients with the inputs of opposite signs. Their values are particularly large for the last set of inputs that describes the weather. In summary, in addition to better performance than a single network, the gated experts used in stage 1 also allow us to gain some insight into the dynamics of the system.

7 Adaptive Variances: Less Overfitting and Better Segmentation

7.1 Problem and Conventional Approaches

One of the most serious problems in applying flexible machine learning techniques to noisy real-world data is the problem of overfitting. Section 2.3.3 discussed one approach to combat overfitting—we modified the cost function and the update rules by incorporating prior beliefs about the variances and about output and weight values. Furthermore, in Section 6.2 we discussed briefly the technique for pruning non-significant weights that we used in the energy prediction example.

To study the effect of the gated experts architecture and cost function on the overfitting problem, we do not use any such techniques in this section but only monitor training and test errors. One manifestation of overfitting is when the performance on out-of-sample data, plotted as a function of training time, starts deteriorating after having reached an optimal point. This indicates that the network is learning more features at that stage in training that do not generalize to new data than features that do. One technique against overfitting then simply consists of stopping training early and taking the network at the minimum of a validation set [Weigend et al., 1990].

As training proceeds, the network tends to shift its resources towards the high noise regions: the more noisy a data point, the bigger its error and thus the bigger its effect in error backpropagation. We assume that there are some regions in input space that are more noisy than others ("noise heterogeneity"). If every data point is weighted equally, the noisy regions tend to attract the resources of the network, mistaking the noise as signal and trying to model it (i.e., overfitting). At the same time, the resources are moved away from the less noisy regions, resulting in underfitting there. Using large networks and early stopping, although usually faring much better than "training until convergence," cannot deal with this problem, since differences in local structure cannot enter into such a global error model.

This article considers two alternatives to a standard network trained with global least mean squared errors. Section 7.2 reviews a technique, introduced in [Weigend and Nix, 1994, Nix and Weigend, 1995], originally developed to obtain

local error bars for regression and prediction problems. Section 7.3 compares the dynamics of overfitting on the three architectures: gated experts, a network with local error bars performing weighted regression, and a standard single network.

7.2 Estimating Local Error Bars and Weighted Regression

The goal of the technique presented in [Nix and Weigend, 1995] is to estimate explicitly the local noise level (in addition to the value of the prediction itself). "Local" means that, just like the prediction, the noise level is a function of the inputs.

Figure 18 illustrates the architecture. The σ^2-unit has a hidden layer of its own that receives connections from both the y-unit's hidden layer and the input pattern itself. This allows great flexibility in learning $\sigma^2(\mathbf{x})$. In contrast, if the σ^2-unit had no hidden layer of its own, the network would be constrained to approximate $\sigma^2(\mathbf{x})$ by linear combination (and exponentiation) of the features useful for $y(\mathbf{x})$.

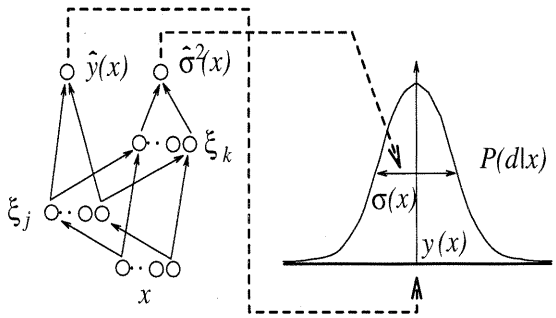


Figure 18: Network architecture for estimating local error bars using an auxiliary output unit. This architecture allows the unit for the conditional variance $\sigma^2(\mathbf{x})$ access to both information in the input pattern itself and in the hidden unit representation formed while learning the conditional mean, $y(\mathbf{x})$. Reprinted from [Weigend and Nix, 1994].

The cost function was discussed in Section 2.4 as Eq. 30:

$$\mathcal{C}_{\mathrm{LEB}} = \frac{1}{2}\left[\frac{\left(d - y(\mathbf{x})\right)^2}{\sigma^2(\mathbf{x})} + \ln \sigma(\mathbf{x})^2 + \ln 2\pi\right] \quad . \tag{38}$$

Its derivation in a maximum likelihood framework assumes Gaussian distributed errors on the outputs.

In order to be able to write the weight-update equations explicitly, we need to specify the activations functions. We choose a linear y output, tanh hidden units, and an exponential function for the σ^2-unit, incorporating the constraint that the variance should be positive. Taking derivatives of the cost $\mathcal{C}_{\mathrm{LEB}}$ with respect to the network weights, we obtain:

$$\frac{\partial \mathcal{C}^{(t)}_{\mathrm{LEB}}}{\partial w_{yj}} = -\frac{1}{\sigma^2(\mathbf{x}^{(t)})}\left[d^{(t)} - y(\mathbf{x}^{(t)})\right]\xi_j(\mathbf{x}^{(t)}) \tag{39}$$

$$\frac{\partial \mathcal{C}^{(t)}_{\mathrm{LEB}}}{\partial w_{\sigma^2 k}} = -\frac{1}{2\sigma^2(\mathbf{x}^{(t)})}\left\{\left[d^{(t)} - y(\mathbf{x}^{(t)})\right]^2 - \sigma^2(\mathbf{x}^{(t)})\right\}\xi_k(\mathbf{x}^{(t)}) \tag{40}$$

where ξ denotes a hidden unit activation. For weights not connected to the output, the weight-update equations are derived using the chain rule in the same way as in standard backpropagation. Note that Eq. 40 is equivalent to training a separate function-approximation network for $\sigma^2(\mathbf{x})$ where the targets are the squared errors $[d^{(t)} - y(\mathbf{x}^{(t)})]^2$. We use a three-step procedure for the search, first learning the weights to the y-unit alone, then freezing those and learning the weights to the σ^2-unit, and only in the last stage, we update all weights by gradient descent in $\mathcal{C}_{\mathrm{LEB}}$.

Although different in architecture, this method of estimating local error bars shares with the gated experts method the ability to model local noise levels. This is done explicitly in the local error bar model and implicitly in the gated experts

by partitioning the input space into sub-regions characterized by similar noise levels. The interpretation in terms of weighted regression given below Eq. 24 in the context of gated experts holds here as well. The next section compares and contrasts the different approach for the example of predicting the daily energy demand of France.

7.3 Evolution of Learning: Learning Curves

To investigate the learning dynamics, we compare three architectures for the prediction problem introduced in Section 6. The three architectures are: gated experts (Fig. 19), learning local variances (Fig. 20), and a single neural network (Fig. 21). The measure of comparison is E_{NMS}, the normalized mean squared error, defined in Eq. 34.

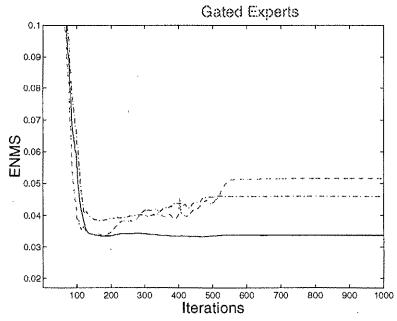


Figure 19: E_{NMS} vs. iterations for gated experts. The solid line is the in-sample error (training set), the two broken lines are out-of-sample errors on two test sets.

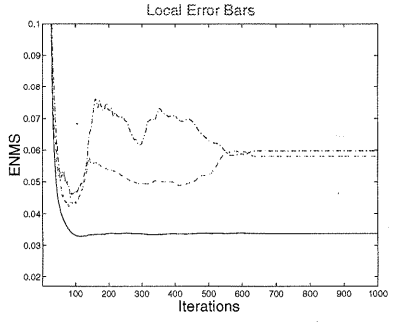


Figure 20: E_{NMS} learning curve for the local error bar model that learns to predict both the next value of the series and its (local) variance. The test-minima are not as low as for the gated experts.

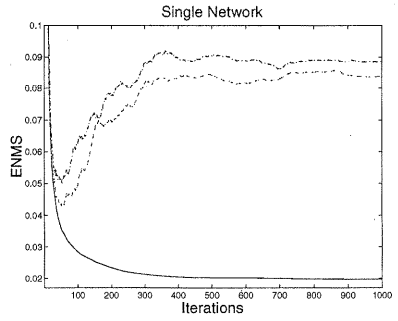


Figure 21: E_{NMS} learning curves for a single network trained by back-propagating squared errors. Since it overfits strongly, the exact stopping point is important.

Figures 19–21 reveal significantly different degrees of overfitting. Whereas the learning of the gated expert is stable and does not overfit much, the local error bar network is somewhat worse, and the single network a lot worse. Note that the in-sample error (solid line) is significantly lower for the single network than for the other two architectures: the single network is trained to minimize precisely this E_{NMS} *performance* measure, whereas in the other cases, more complex costs are minimized.[14]

It is thus important to distinguish between the full cost (which might include penalty terms, robust errors, etc.) and the performance term we are ultimately, out-of-sample, interested in (which we take here to be squared error, but it could be anything). In order to study the dynamics of overfitting of the different architectures, we switched off any penalty terms, priors, pruning, etc., and only performed descent on the maximum likelihood cost functions. We found that the weighted regression implied by the local variance terms in the local error bars model helps, and we found that the additional segmentation of the gated experts helps further. In this sense, both can be viewed as additions to the anti-overfitting arsenal.

8 Summary

This article describes nonlinearly gated nonlinear experts with adaptive variances and applies them to the prediction and analysis of time series. Gated experts can be viewed as a new method for data analysis, particularly well suited for discovering processes with changes in conditions, ranging from the importance of vacations in France to market sentiment in trading. The blend of supervised and unsupervised learning, embedded in the architecture and cost function, gives results in three areas:

[14] We also analyzed the learning curves of the *costs* (as opposed to E_{NMS}) of the other two examples and found that the *costs* also overfit significantly.

- **Prediction.** The performance of gated experts is significantly better than that of single networks for piece-wise stationary processes that switch between regimes with different dynamics and noise levels.
- **Analysis.** Gated experts discover hidden regimes. Analyzing the individual experts (e.g., in terms of average predictability in that regime, or sensitivity analysis of the importance of its inputs), and analyzing the gate (e.g., the correlations between its outputs and auxiliary variables) yields a deeper understanding of the underlying process.[15]
- **Overfitting.** Matching the complexity of the model to that of the data is a central goal of machine learning. Mismatches manifest themselves as overfitting. This article focuses on one aspect: matching the local noise level and compares gated experts (where the local matching is done through the locality of each expert) with the architecture of predicting local error bars. Both architectures implement weighted regression and show less overfitting than standard backpropagation that assumes a global noise scale.

The gated experts described in this paper serve as a starting point into several directions. The idea of annealing to achieve stable segmentation, briefly mentioned in Section 2.3.4, is discussed in detail in [Srivastava and Weigend, 1995]. The present article assumes that the switching occurs at random. To capture the dynamics of the switching process, we introduce recurrence into the gating network. This enables us to develop a special architecture for the early detection of regime switches, and allows us to extract temporal processes on more than one time scale. [Bjorn and Weigend, 1995] address the multi-scale problem explicitly by putting the experts into the wavelet domain: each expert learns the structure within a single octave, and the experts compete for predictability on the different time scales.

Acknowledgments

Applying mixture models to time series analysis was first suggested to us by Steve Nowlan in 1991. Andreas Weigend would like to thank Steve Waterhouse for discussions about the priors, Klaus Pawelzik and Klaus Müller for discussions on similarities and differences to their work, Jong-Hoon Oh for discussions about the phase segmentation, Shanming Shi for the implementation in MATLAB, and Tom McCabe and Ralph Slutz for helping to eliminate most of the typos in this article. Morgan Mangeas thanks Mike Jordan for discussions during the summer school of Electricité de France (EDF) in 1994 and acknowledges support by EDF while visiting the Computer Science Department of the University of Colorado at Boulder. This material is based upon work supported by the National Science Foundation under Grant No. RIA ECS-9309786.

References

[Baldi and Chauvin, 1994] Baldi, P. and Chauvin, Y. (1994). Smooth online learning algorithms for hidden Markov models. *Neural Computation*, 6:307–318.

[Bengio and Frasconi, 1995] Bengio, Y. and Frasconi, P. (1995). An input output HMM architecture. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*, pages 427–434. MIT Press, Cambridge, MA.

[Bishop, 1994] Bishop, C. M. (1994). Mixture density networks. Technical report, Aston University.

[Bjorn and Weigend, 1995] Bjorn, V. and Weigend, A. S. (1995). Measuring predictability using multi-scale embedding. Technical Report CU-CS-797-95, University of Colorado at Boulder, Computer Science Department.

[Bollerslev, 1986] Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 21:307–328.

[Bollerslev et al., 1990] Bollerslev, T., Chou, R. Y., Jayaraman, N., and Kroner, K. F. (1990). ARCH modeling in finance: A review of the theory and empirical evidence. *Journal of Econometrics*, 52(1):5–60.

[15] If segmentation is the sole goal, a variation of the present architecture uses a two-sided, acausal filter, by presenting data from both before and after the present point as input.

[Bridle, 1989] Bridle, J. S. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In Fogelman-Soulie, F. and Heŕault, J., editors, *Neuro-computing: Algorithms, Architectures and Applicatations*, pages 227–236. Springer-Verlag.

[Broomhead and Lowe, 1988] Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355.

[Buntine and Weigend, 1991] Buntine, W. L. and Weigend, A. S. (1991). Bayesian back-propagation. *Complex Systems*, 5:603–643.

[Cacciatore and Nowlan, 1994] Cacciatore, T. W. and Nowlan, S. J. (1994). Mixtures of controllers for jump linear and nonlinear plants. In Cowen, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6 (NIPS*93)*, pages 719–726, San Francisco, CA. Morgan Kaufmann.

[Casdagli, 1989] Casdagli, M. (1989). Nonlinear prediction of chaotic time series. *Physica D*, 35:335–356.

[Cottrell et al., 1995] Cottrell, M., Girard, B., Girard, Y., Mangeas, M., and Muller, C. (1995). Neural modeling for time series: a statistical stepwise method for weight elimination. *IEEE Transaction on Neural Networks*, 6:1355–1364.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Stat. Soc. B*, 39:1–38.

[Diebold and Rudebusch, 1996] Diebold, F. X. and Rudebusch, G. D. (1996). Measuring business cycles: A modern perspective. *Review of Economics and Statistics*.

[Doutriaux and Zipser, 1990] Doutriaux, A. and Zipser, D. (1990). Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J., and Hinton, G. E., editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 303–309, San Fransisco, CA. Morgan Kaufmann.

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

[Engle, 1982] Engle, R. F. (1982). Autoregressive conditional heteroskedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50:987–1007.

[Finnoff et al., 1993] Finnoff, W., Hergert, F., and Zimmermann, H. G. (1993). Improving generalization performance by nonconvergent model selection methods. *Neural Networks*, 6:771–783.

[Fraser and Dimitriadis, 1994] Fraser, A. M. and Dimitriadis, A. (1994). Forecasting probability densities by using hidden Markov models. In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 265–282, Reading, MA. Addison-Wesley.

[Friedman, 1991] Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19:1–142.

[Granger, 1994] Granger, C. W. J. (1994). Forecasting in economics. In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 529–538, Reading, MA. Addison-Wesley.

[Granger and Teräsvirta, 1993] Granger, C. W. J. and Teräsvirta, T. (1993). *Modelling Nonlinear Economic Relationships*. Oxford University Press, Oxford, UK.

[Hamilton, 1989] Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57:357–384.

[Hamilton, 1990] Hamilton, J. D. (1990). Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45:39–70.

[Hamilton, 1994] Hamilton, J. D. (1994). *Time Series Analysis*. Princeton University Press, Princeton.

[Ivanova et al., 1994] Ivanova, T. O., Mottle, V. V., and Muchnik, I. B. (1994). Estimation of the parameters of hidden Markov models of noise-like signals with abruptly changing probabilistic properties. *Automation and Remote Control*, 55:1299–1315,1428–1445.

[Jacobs, 1995] Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888.

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

[Johnson and Kotz, 1972] Johnson, N. L. and Kotz, S. (1972). *Distributions in Statistics. Continuous Multivariate Distributions*. Wiley, New York.

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

[Jordan and Xu, 1995] Jordan, M. I. and Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, (in press).

[Lapedes and Farber, 1987] Lapedes, A. and Farber, R. (1987). Nonlinear signal processing using neural networks. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM.

[Levin, 1991] Levin, E. (1991). Modeling time varying systems using hidden control neural architecture. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3 (NIPS*90)*, pages 147–154. Morgan Kaufmann.

[Lewis et al., 1994] Lewis, P. A. W., Ray, B. K., and Stevens, J. G. (1994). Modeling time series using multivariate adaptive regression splines (MARS). In Weigend, A. S. and Gershenfeld, N. A., editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 296–318, Reading, MA. Addison-Wesley.

[Mangeas et al., 1995] Mangeas, M., Muller, C., and Weigend, A. S. (1995). Forecasting electricity demand using a mixture of nonlinear experts. In *World Congress on Neural Networks (WCNN'95)*, pages II–48–53.

[McCullagh and Nelder, 1989] McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models*. Chapman and Hall, London.

[Nadas and Mercer, 1996] Nadas, A. and Mercer, R. L. (1996). Hidden Markov models and some connections with artificial neural networks. In Smolensky, P., Mozer, M. C., and Rumelhart, D. E., editors, *Mathematical Perspectives on Neural Networks*. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Nix and Weigend, 1995] Nix, D. A. and Weigend, A. S. (1995). Learning local error bars for nonlinear regression. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*, pages 488–496. MIT Press, Cambridge, MA.

[Nowlan, 1991] Nowlan, S. J. (1991). *Soft Competitive Adaptation: Neural Network Learning Algorithms based on Fitting Statistical Mixtures*. PhD thesis, School of Computer Science, Carnegie Mellon University. Technical Report CMU-CS-91-126.

[Nowlan and Hinton, 1992] Nowlan, S. J. and Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

[Pawelzik et al., 1996] Pawelzik, K., Kohlmorgen, J., and Müller, K.-R. (1996). Annealed competition of experts for a segmentation and classification of switching dynamics. *Neural Computation*, 8.

[Pearson, 1894] Pearson, K. (1894). Contributions to the mathematical theory of evolution. *Phil. Trans. Royal Soc.*, 185:71–110. See also V. 185A, p. 195.

[Perrone, 1994] Perrone, M. P. (1994). General averaging results for complex optimization. In Mozer, M. C., Smolensky, P., Touretzky, D. S., Elman, J. L., and Weigend, A. S., editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 364–371, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Poggio and Girosi, 1990] Poggio, T. and Girosi, F. (1990). Networks for Approximation and Learning. *Proceedings of the IEEE*, 78(9):1481–1497.

[Poritz, 1988] Poritz, A. B. (1988). Hidden Markov models: A guided tour. *IEEE*, pages 7–13.

[Predoviciu and Jordan, 1995] Predoviciu, M. M. and Jordan, M. I. (1995). Learning the parameters of HMMs with auxilliary input. Technical report, MIT.

[Press et al., 1992] Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge.

[Quandt, 1958] Quandt, R. E. (1958). The estimation of parameters of linear regression system obeying two separate regimes. *Journal of the American Statistical Association*, 55:873–880.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Redner and Walker, 1984] Redner, R. A. and Walker, H. F. (1984). Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26:195–237.

[Rose et al., 1990] Rose, K., Gurewitz, E., and Fox, G. C. (1990). Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65:945–948.

[Rumelhart et al., 1995] Rumelhart, D. E., Durbin, R., Golden, R., and Chauvin, Y. (1995). Backpropagation: The basic theory. In Chauvin, Y. and Rumelhart, D. E., editors, *Backpropagation: Theory, Architectures, and Applications*, pages 1–34, Hillsdale, NJ. Lawrence Erlbaum Associates.

[Rumelhart et al., 1986] Rumelhart, D. E., McClelland, J. L., and group, P. R. (1986). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition. Volume 1: Cognition*. MIT Press, Cambridge, MA.

[Seber and Wild, 1989] Seber, G. A. F. and Wild, C. J. (1989). *Nonlinear Regression*. Wiley, New York.

[Srivastava and Weigend, 1995] Srivastava, A. N. and Weigend, A. S. (1995). Improving time series segmentation with gated experts through annealing. Technical Report CU-CS-795-95, University of Colorado at Boulder, Computer Science Department.

[Tanner, 1993] Tanner, M. A. (1993). *Tools for Statistical Inference*. Springer-Verlag, 2nd edition.

[Titterington et al., 1985] Titterington, D., Smith, A. F. M., and Makov, U. E. (1985). *Statistical Analysis of Finite Mixture Distributions*. John Wiley, New York.

[Tong and Lim, 1980] Tong, H. and Lim, K. S. (1980). Threshold autoregression, limit cycles and cyclical data. *J. Roy. Stat. Soc. B*, 42:245–292.

[Waterhouse et al., 1996] Waterhouse, S. R., MacKay, D. J. C., and Robinson, A. J. (1996). Bayesian modesl for mixtures of epxerts. In *Advances in Neural Information Processing Systems 8 (NIPS*95)*. MIT Press.

[Waterhouse and Robinson, 1995] Waterhouse, S. R. and Robinson, A. J. (1995). Non-linear prediction of acoustic vectors using hierarchical mixture of epxerts. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7 (NIPS*94)*, pages 835–842. MIT Press, Cambridge, MA.

[Weigend and Gershenfeld, 1994] Weigend, A. S. and Gershenfeld, N. A., editors (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison-Wesley, Reading, MA.

[Weigend et al., 1990] Weigend, A. S., Huberman, B. A., and Rumelhart, D. E. (1990). Predicting the future: A connectionist approach. *International Journal of Neural Systems*, 1:193–209.

[Weigend and LeBaron, 1994] Weigend, A. S. and LeBaron, B. (1994). Evaluating neural network predictors by bootstrapping. In *Proceedings of International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212. Technical Report CU-CS-725-94, Computer Science Department, University of Colorado at Boulder, ftp://ftp.cs.colorado.edu/pub/Time-Series/MyPapers/bootstrap.ps.

[Weigend and Nix, 1994] Weigend, A. S. and Nix, D. A. (1994). Predictions with confidence intervals (local error bars). In *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, pages 1207–1212, Seoul, Korea.

[Weigend and Srivastava, 1995] Weigend, A. S. and Srivastava, A. N. (1995). Predicting conditional probability distributions: A connectionist approach. *International Journal of Neural Systems*, 6:109–118.

[Xu, 1994] Xu, L. (1994). Signal segmentation by finite mixture model and EM algorithm. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN'94)*, pages 453–458, Tainan, Taiwan.

[Yule, 1927] Yule, G. (1927). On a method of investigating periodicity in disturbed series with special reference to wolfer's sunspot numbers. *Phil. Trans. Roy. Soc. London*, A 226:267–298.