A Principled Foundation for LCS

Jan Drugowitsch and Alwyn M. Barry

Department of Computer Science
University of Bath
Bath, BA2 7AY, UK
J.Drugowitsch@bath.ac.uk, A.M.Barry@bath.ac.uk

Abstract. In this paper we promote a new methodology for designing LCS that is based on first identifying their underlying model and then using standard machine learning methods to train this model. This leads to a clear identification of the LCS model and makes explicit the assumptions made about the data, as well as promises advances in the theoretical understanding of LCS through transferring the understanding of the applied machine learning methods to LCS. Additionally, it allows us, for the first time, to give a formal and general, that is, representation-independent, definition of the optimal set of classifiers that LCS aim at finding. To demonstrate the feasibility of the proposed methodology we design a Bayesian LCS model by borrowing concepts from the related Mixtures-of-Experts model. The quality of a set of classifiers and consequently also the optimal set of classifiers is defined by the application of Bayesian model selection, which turns finding this set into a principled optimisation task. Using a simple Pittsburgh-style LCS, a set of preliminary experiments demonstrate the feasibility of this approach.

1 Introduction

In this work we promote a model-based design approach for LCS that allows us to define formally and from first principles which set of classifiers LCS aim at learning, thus tackling the core question of LCS. The motivation behind acquiring the model-based approach is that, in essence, any machine learning method is based on some form of (sometimes implicit) model that determines its training and makes explicit the assumptions that are made about the data that it models. Thus, if the model underlying LCS is made explicit, we can use standard machine learning methods for its training and analysis, in addition to making explicit the assumptions about the data. The latter is important as it tells us how LCS differ from other approaches in machine learning, and in particular for which kind of data they might feature superior performance.

Up to now, the design of LCS is mostly handled in an ad-hoc way with an intuitive understanding of how the optimal set of classifiers might look like. This leads to algorithmic descriptions of their implementations and subsequent attempts of analysing their performance and inner working, such as, for example, the extensive analysis of XCS [1] by Butz et al. [2,3,4]. We propose a different approach: firstly, one should seek to make explicit the underlying model that

LCS uses for a set of classifiers to represent the data. Having such a model, the definition of the optimal set of classifiers, that is, the set that LCS training should seek, becomes simple, yet powerful: the optimal set of classifiers is the one that models the data best.

Such a definition of the optimal set of classifiers, and with it the whole methodology, seemingly only applies to regression and classification tasks, and only if all data is known before the LCS is trained (that is, for batch training). However, we claim that the approach can be used to derive both batch and incremental training methods, based on interpreting incremental methods as approximations to solutions that can equally be found by a batch learning approach[1]. Also, the approach can be extended to sequential decision tasks by the application of reinforcement learning, and using LCS to approximate the value function, which is an incremental non-stationary univariate regression task [5].

Naturally, there is no single model that describes all possible LCS variants. Similarly, there is not only a single machine learning method that can be applied to tell us which model represent the data best. Thus, we want to emphasise that the regression model that forms the core of this paper — which is one that closely resembles XCS(F) [1,6] — and the methods that we have applied to define and find the best set of classifiers are only meant to illustrate that the methodology we promote does indeed lead to feasible LCS implementations. The presented model can be easily reformulated to a classification model that leads to a different LCS specialised on classification tasks. Alternatively, it might be reformulated to make different assumptions about the data, which leads to a different LCS variants.

Due to space constraints we mainly focus on formulating the model, which we derive from a generalisation of the well-known Mixtures-of-Experts model. As most of the LCS research is focused on model training rather than the model itself, we will also give an overview of how to train the presented model, and show preliminary experimental results. While we understand that the brevity of the presentation might make the details of the approach not immediately accessible, we feel that it needs to be presented as a whole to at least initially underline this holistic approach. Consequently, the experiments that we present cannot be reproduced by the reader, but we will make all necessary details available in forthcoming publications.

We start by giving a general description of LCS as a model that combines localised models (that is, the classifiers) to a global model. We then link such a structure to a generalised Mixture-of-Experts model, followed by discussing how to keep its training computationally tractable. Furthermore, we exploit the model by introducing a principled approach to the identification of the quality of a set of classifiers given the data, leading to a formal and general definition of the optimal set of classifiers. As this approach requires a Bayesian model, we fit priors to the probabilistic LCS model that make explicit the usually implicit

[1] A simple example is the application of Recursive Least Squares as an incremental approach to finding a solution that can equally be found be any Least Squares batch method.

prior assumptions that each model makes about the data. The quality metric on a set of classifiers turns the search for a good set of classifiers into an optimisation problem that can be handled by a genetic algorithm, which allows for the creation of a simple Pittsburgh-style LCS. Using such an implementation, we present some experimental results that show the applicability of the previously introduced concepts and conclude by pointing out the achievements and implications of this work.

2 Assembling an LCS Model

To create an adequate model for LCS we will firstly discuss the general structure of LCS models, based on characterising them as a member of the family of parametric models. This reveals that we can facilitate their similarity to the well-known Mixtures-of-Experts model to provide a probabilistic formulation for their model structure, describing a fixed set of classifiers.

2.1 A Bird's Eye View of the LCS Model

A parametric model family in ML can be characterised by the model structure $\mathcal{M}$ and the model parameters[2] $\boldsymbol{\theta}$. While the model structure is usually chosen before the model is trained, the model is fitted to the data by adjusting its parameters. For example, given the family of feed-forward neural networks, the number of hidden layers and nodes in each of the layers determines the model structure, and the model parameters are the weights of the connections between these nodes. Accordingly, the model structure commonly determines the number of model parameters that need adjustment.

While the same concepts apply to classification and reinforcement learning tasks, let us for now consider only regression tasks where the observations are assumed to be sampled from a target function f that maps the input space $\mathcal{X} = \mathcal{R}^{D_\mathcal{X}}$ into the output space $\mathcal{Y} = \mathcal{R}$. In LCS, we have a set of K classifiers, each of which matches a subset of the input space. Considering classifier k, this classifier matches $\mathcal{X}_k \subseteq \mathcal{X}$ and provides a localised regression model $\hat{f} : \mathcal{X}_k \rightarrow \mathcal{Y}$, where the localisation is determined by $\mathcal{X}_k$ and is traditionally represented by the condition and action of a classifier. To provide a model of the full target function, the local models are *mixed* (that is, combined) in some way to provide the estimate $\hat{f} : \mathcal{X} \rightarrow \mathcal{Y}$, assuming that each element of $\mathcal{X}$ is matched by at least one classifier.

Leaving incremental training methods aside for now, this perspective reveals that the model structure $\mathcal{M}$ in LCS is in fact the number of classifiers in the population, and the parts of $\mathcal{X}$ that each of them matches. On the other hand, the model parameters $\boldsymbol{\theta}$ are the combined parameters of the regression model of

[2] While a *parameter* in LCS often refers to a constant that is set before training LCS and remains unchanged during training, we use it when referring to a variable of the model that is modified during training, and call a parameter in the LCS sense a *system parameter*.

each of the classifiers and the parameters of the model used to mix the classifier predictions. It also shows that LCS do not only aim at training a model $\mathcal{M}$ by adjusting the parameters of classifiers and mixing, but also tries to find an adequate model structure that fits (but does not overfit) the target function. While the second task is the more challenging one, let us for now concentrate on the first one, that is, how to adjust the model parameters for a given model structure, and come back to improving the model structure in Sect. 3. To do so, we need to define exactly the regression models underlying the classifiers and the model used for mixing their prediction.

Fortunately, Mixtures-of-Experts (MoE) [7,8] feature a similar model structure to LCS, and we can use this similarity to generalise MoE such that it corresponds to the model that underlies LCS. While we introduce the standard MoE model in the next section, we present the generalisations that make it similar to LCS in the section thereafter.

2.2 Mixtures of Experts

Mixture of Experts are probably most intuitively explained from the generative point-of-view: Let $\boldsymbol{X} = \{\boldsymbol{x}_n \in \mathcal{X}\}$ be the set of N inputs, and $\boldsymbol{Y} = \{y_n \in \mathcal{Y}\}$ the corresponding set of outputs, together giving the data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$. For a set of K experts, each observation $\{\boldsymbol{x}, y\}$ is assumed to be generated by one and only one expert. We can model this by introducing the latent random vector $\boldsymbol{z} = (z_1, \dots, z_K)^T$ of binary random variables, each of which corresponds to an expert. Given that expert $\bar{k}$ generated the observation, then $z_{\bar{k}} = 1$ and $z_k = 0$ for all $k \neq \bar{k}$. Hence, $\boldsymbol{z}$ has a 1-of-K structure, that is, it always contains one and only one element that is 1, with all other elements being 0.

Concentrating again on regression tasks, let the model of expert k be given by the conditional probability

$$p(y|\boldsymbol{x}, \boldsymbol{w}_k, \tau_k) = \mathcal{N}(y|\boldsymbol{w}_k^T \boldsymbol{x}, \tau_k^{-1}), \tag{1}$$

that is, by a univariate Gaussian with mean $\boldsymbol{w}_k^T \boldsymbol{x}$ and precision (that is, inverse variance) τ_k, where $\boldsymbol{w}_k$ is the weight parameter and τ_k is the precision parameter of expert k. This is a standard model for linear regression assuming constant noise variance over all observations and can easily be fitted by maximum likelihood, resulting in a linear least-squares problem.

As each observation is generated by one and only one expert, we can facilitate the 1-of-K structure of $\boldsymbol{z}$ to get the probability of y given $\boldsymbol{x}$ and all experts by

$$p(y|\boldsymbol{x}, \boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{z}) = \prod_{k=1}^{K} p(y|\boldsymbol{x}, \boldsymbol{w}_k, \tau_k)^{z_k}, \tag{2}$$

where $\boldsymbol{W} = \{\boldsymbol{w}_k\}$, $\boldsymbol{\tau} = \{\tau_k\}$, and z_k are the elements of the latent variable $\boldsymbol{z}$ that corresponds to the observation $\{\boldsymbol{x}, y\}$.

If we know the values of $\boldsymbol{Z} = \{\boldsymbol{z}_n\}$, where $\boldsymbol{z}_n$ stands for the latent variable corresponding to observation $\{\boldsymbol{x}_n, y_n\}$, then we can train each expert independently to fit only the observations that it generated. This can be seen by expanding the expression of the log-likelihood over the whole data

$$\begin{aligned}\ln p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{Z}) &= \ln \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{z}_n) \\ &= \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \ln p(y_n|\boldsymbol{x}_n, \boldsymbol{w}_k, \tau_k),\end{aligned}$$

where z_{nk} assigns the observations to the different experts. However, as $\boldsymbol{Z}$ is usually not known beforehand, we need to learn a model for it at the same time as training the experts. For that task MoE uses the multinomial logit model[3]; this is a standard model for categorical data and in the MoE context is known as the gating network, as it is responsible for associating observations and experts. It is defined by introducing another parameter vector $\boldsymbol{v}_k$ per expert that determines the probability of expert k having generated observation $\{\boldsymbol{x}_n, y_n\}$ by

$$g_k(\boldsymbol{x}_n) \equiv p(z_{nk} = 1|\boldsymbol{x}_n, \boldsymbol{v}_k) = \frac{\exp(\boldsymbol{v}_k^T \boldsymbol{x}_n)}{\sum_{j=1}^{K} \exp(\boldsymbol{v}_j^T \boldsymbol{x}_n)}. \tag{3}$$

This function is also known as the softmax function, and defines a soft linear partitioning over $\mathcal{X}$. The model emerges from the assumption that the relation between the probability of an expert k generating an observation $\{\boldsymbol{x}, y\}$ is log-linear in $\boldsymbol{x}$, that is $p(z_k = 1|\boldsymbol{x}, \boldsymbol{v}_k) \propto \exp(\boldsymbol{v}_k^T \boldsymbol{x})$.

Given the model structure $\mathcal{M}$ of MoE by the number of experts K, and having defined both the model for the experts and the gating network, the model parameters $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{V}\}$ can be found by the EM-algorithm: in the E-step, the posteriors $p(z_{nk} = 1|\boldsymbol{x}_n, y_n, \boldsymbol{v}_k)$ are computed based on the current goodness-of-fit of the experts. The M-step uses these posteriors to adjust the expert and gating network parameters in order to maximise the likelihood of the data $\mathcal{D}$ and the latent variables $\boldsymbol{Z}$ [8]. This update has the effect that the gating network is adjusted according to the goodness-of-fit of the different experts, and the experts are trained according to how the gating network assigns the observations to the experts. In combination, this causes the input space to be partitioned by a soft linear partition, and each expert models the observations that fall in one of these partitions. Hence, the experts form localised models, where the localisation is determined by the gating network.

At this point the relation to LCS should be clear: The classifiers in LCS correspond to the experts in MoE, and the gating network has the same task as the mixing model in LCS. However, while the localisation of the classifiers in LCS is part of the model structure, the experts in MoE are localised by the interaction between the gating network and the experts. In the next section we show how an additional localisation layer in the MoE model can act as a generalisation to both the MoE model and the LCS model, and as such provides a strong probabilistic foundation for the LCS model.

[3] For details about the multinomial logit model and other generalised linear models see [9].

2.3 LCS as Generalised Mixtures of Experts

As a generalisation to the standard MoE model, we want to restrict the possibility of experts to generate observations to the inputs that the expert matches. Such matching is easily introduced by an additional binary random variable m_{nk} that is 1 if expert k matches input $\boldsymbol{x}_n$, and 0 otherwise. In contrast to the latent variables z_{nk}, m_{nk} does not need to obey the 1-of-K as several experts can match the same input. To enforce this matching, we define the probability of expert k generating the observation $\{\boldsymbol{x}_n, y_n\}$ to be

$$p(z_{nk} = 1 | m_{nk}, \boldsymbol{x}_n, \boldsymbol{v}_k) \propto \begin{cases} \exp(\boldsymbol{v}_k^T \vartheta(\boldsymbol{x}_n)) & \text{if } m_{nk} = 1, \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $\vartheta : \mathcal{X} \to \mathcal{R}^{D_V}$ is a transfer function over the input vectors, resulting in an additional generalisation over the MoE model, which uses $\vartheta(\boldsymbol{x}) = \boldsymbol{x}$. Therefore, if expert k matches input $\boldsymbol{x}_n$ then the probability of it generating the observation $\{\boldsymbol{x}_n, y_n\}$ is determined by a log-linear model as for the standard MoE model. If it does not match, however, then it could not have produced the observation either (that is, with probability 0). If we marginalise that probability over m_{nk}, we get

$$p(z_{nk} = 1 | \boldsymbol{x}_n, \boldsymbol{v}_k) \propto m_k(\boldsymbol{x}_n) \exp(\boldsymbol{v}_k^T \vartheta(\boldsymbol{x}_n)), \tag{5}$$

where we have defined the matching function $m_k(\boldsymbol{x}_n) \equiv p(m_{nk} = 1 | \boldsymbol{x}_n)$, giving the probability of expert k matching input $\boldsymbol{x}_n$. Adding the normalisation constant, we get the redefined gating network

$$g_k(\boldsymbol{x}_n) \equiv p(z_{nk} = 1 | \boldsymbol{x}_n, \boldsymbol{v}_k) = \frac{m_k(\boldsymbol{x}_n) \exp(\boldsymbol{v}_k^T \vartheta(\boldsymbol{x}_n)}{\sum_{j=1}^K m_j(\boldsymbol{x}_n) \exp(\boldsymbol{v}_j^T \vartheta(\boldsymbol{x}_n))}. \tag{6}$$

Comparing Eq. (6) to the gating network Eq. (3) of the standard MoE model, we can see the additional mediation by the matching functions. As matching is unchanged during the model fitting process, it is part of the model structure which is hence given by $\mathcal{M} = \{K, \boldsymbol{M}\}$, where $\boldsymbol{M} = \{m_k\}$ is the set of the expert's matching functions.

We do not need to modify the expert models, as by Eq. (4) an expert can only generate observations for a particular input if it matches that input, that is, $p(z_{nk} = 1 | \boldsymbol{x}_n, \boldsymbol{v}_k) > 0$ only if $m_k(x_n) > 0$. Thus, Eq. (2) still remains valid in the generalised MoE model.

To demonstrate that the model generalises over both MoE and LCS, we show how each of them can be recovered by fixing parts of the model structure: To get the standard MoE from our generalisation we simply need to assume a model structure where each expert matches all inputs. This structure is for some K given by the matching functions $m_k(\boldsymbol{x}_n) = 1$ for all n and k. Additionally, we have $\vartheta(\boldsymbol{x}) = \boldsymbol{x}$ as the gating network relies on the same inputs as the experts. LCS are not (yet?) as well defined as MoE and thus we could already claim that the generalised MoE by itself describes an LCS: A classifier corresponds to an expert with its matching function being specified by its condition/action pair, that is, $m_k(\boldsymbol{x}) = 1$ if classifier k matches input $\boldsymbol{x}$, and $m_k(\boldsymbol{x}) = 0$ otherwise. Naturally, if the function ϑ is defined as something other than $\vartheta(\boldsymbol{x}) = 1$, then training the generalised MoE would cause the classifiers to be localised in regions of overlap beyond what is determined by their condition/action pair. While we have experimented with such a setup in [10], current commonly used LCS — such as XCS [1] and its derivates — perform mixing/gating based on an input-independent scalar, which can be modelled by setting $\vartheta(\boldsymbol{x}) = 1$ for all $\boldsymbol{x}$. Additionally, mixing is usually performed by heuristics (such as the normalised fitness in XCS), but having a better probabilistic justification like the multinomial logit model is certainly an advantage.

2.4 Training the Classifiers Independently

While the generalised MoE can be trained just like the standard MoE by using the EM-algorithm, its training comes with the same disadvantages: As the objective function for MoE is highly multi-modal, we will easily get stuck in local optima [11]. This problem is usually addressed by random restarts when training MoE, which still does not guarantee finding the optimal solution. In LCS, fitting a model to the data (that is, tuning its model parameters) is necessary to evaluate a certain model structure, but that needs to be performed many-fold when searching the space of possible model structures to find a good set of classifiers. As this space is potentially huge and very complex, we need to quickly be able to evaluate a single model structure, which is certainly not possible when utilising a random restart strategy.

Fortunately we do not need to look very far to solve this problem: XCS addresses it implicitly by not considering the interaction between classifiers and mixing. In fact, the multitude of local optima in the MoE model stem from the interdependence of expert and gating network training. Note that this interdependence is required to perform the necessary localisation of the experts. However, in our generalisation of the MoE model there is a second layer of localisation that is defined by the matching functions. Hence, for training the classifiers we can assume that the localisation of the different classifiers is fully defined by the matching function, which makes it independent of how their predictions are mixed. This has the advantages that i) classifiers can be trained by a single pass over the data; and ii) classifiers with the same associated condition/action always have the same parameter values, independent of the other classifiers in the population. The mixing parameters can now be either determined heuristically or, alternatively, trained in a single pass based on the goodness-of-fit of the different classifiers. On the downside, removing the link between classifier training and how they are mixed reduces the goodness-of-fit of the whole model, which needs to be counterbalanced by a more powerful model structure search.

Note that the modified training schema moves the model away from MoE towards ensemble learning where independently trained models are combined to form a single model. While this link has also been established independently in [12], it is clearly beyond the scope of this paper to elaborate on its implication. Let us just point out that while interdependent classifier/mixing training assumes that each observation is generated by one and only one classifier, training the classifiers independently changes the assumptions about the data such that each observation is assumed to be a mixture of different generative processes, each of which is modelled by a different classifier.

3 Finding a Good Set of Classifiers

Probably the most important part of LCS is to find a good set of classifiers that fit the data. But what is a good quality metric when we want to evaluate the "goodness" of a set of classifiers? Its error when modelling the data is certainly an important component. However, given a set of observations, the model error is minimal if each observation is modelled by a single classifier — a not very satisfactory solution, given that it does not provide any more information about the data than the data itself. An alternative is to seek for the smallest set of classifiers that still results in an error-free fit of the data. Although better than one classifier per observation, this method would not fare well in the light of noisy data. XCS handles this problem by considering classifiers as accurate up to a user-defined error threshold and thus provides some form of accuracy/generalisation tradeoff. However, the solution does not give guidelines on how to set the error threshold and thus can overfit the data arbitrarily.

3.1 Applying Bayesian Model Selection

As already alluded to in the introduction, we approach the problem of defining the quality of a set of classifiers by assessing how well the model structure it represents explains the given data. Fortunately, this problem is well-known in machine learning and is handled by the field of *model selection*. The essential problem that model selection deals with is to find a model structure that does, on one hand, correctly identify all pattern within the data (within the realm of the model family) but avoids modelling randomness, essentially identifying a good tradeoff between generalisation and goodness-of-fit of a model. This is a difficult problem and different philosophical assumptions about the nature of randomness leads to different results, such as the Minimal Description Length [13] method or Statistical Learning Theory [14].

Bayesian model selection is a model selection method founded in Bayesian statistics which has fortunately already been applied to the standard MoE model [11,15]. It is based on the idea that the probability of a model structure given the data can be evaluated by

$$p(\mathcal{M}|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M})p(\mathcal{M}), \tag{7}$$

where $p(\mathcal{D}|\mathcal{M})$ is the goodness-of-fit of the data given a certain model structure, and $p(\mathcal{M})$ is the prior probability for that model structure[4]. Hence, given that we have a fully Bayesian model, the problem of finding a good model structure becomes as simple as finding one that maximises the model structure posterior $p(\mathcal{M}|\mathcal{D})$.

3.2 A Bayesian LCS Model

To apply Bayesian model selection we require a Bayesian LCS model, which we introduce by extending the probabilistic LCS model by adequate priors on its model parameters. The priors that we introduce are similar to the ones used by Waterhouse et al. for the Bayesian MoE model [17,18]. They are conjugate priors[5] where possible, except for the gating network parameters, where Laplace approximation (for example, [19]) is required to keep the evaluation of the parameter posteriors analytically tractable.

We also give recommendations on the prior parameters that cause them to be very uninformative; that is, in the light of some evidence the influence of the prior on the posterior is negligible. This makes the model selection criterion and subsequently also our definition of the optimal set of classifiers almost completely independent of the choice of priors.

Classifier Model Priors. For the univariate linear classifier model Eq. (1) we acquire the conjugate normal inverse-gamma prior

$$\begin{aligned} p(\boldsymbol{w}_k, \tau_k|\alpha_k) &= \mathcal{N}(\boldsymbol{w}_k|\boldsymbol{0}, (\alpha_k\tau_k)^{-1}\boldsymbol{I})\mathrm{Gam}(\tau_k|a_\tau, b_\tau) \\ &= \left(\frac{\alpha_k\tau_k}{2\pi}\right)^{D_\mathcal{X}/2} \frac{b_\tau^{a_\tau}\tau_k^{(a_\tau-1)}}{\Gamma(a_\tau)} \exp\left(-\frac{\alpha_k\tau_k}{2}\boldsymbol{w}_k^T\boldsymbol{w}_k - a_\tau\tau_k\right), \quad (8) \end{aligned}$$

where $\mathrm{Gam}(\cdot|a_\tau, b_\tau)$ denotes the gamma distribution with scale a_τ and shape b_τ, and $\Gamma(\cdot)$ is the gamma function. This prior expresses that we expect the elements of the weight vector $\boldsymbol{w}_k$ to be independently distributed by a zero-mean Gaussian with precision (that is, inverse variance) $\alpha_k\tau_k$. In other words, we expect the weight elements to be small, which is a realistic assumption, given that the target function that a classifier aims at modelling is expected to be smooth. The noise precision τ_k is distributed according to a gamma distribution which we will parameterise as in [11] by $a_\tau = 10^{-2}$ and $b_\tau = 10^{-4}$ to keep the prior sufficiently broad an uninformative.

[4] Bayesian model selection differs from the approach of maximum likelihood in that, assuming a uniform model structure prior $p(\mathcal{M})$, Bayesian model selection aims at finding the $\mathcal{M}$ that maximises $p(\mathcal{D}|\mathcal{M})$, whereas maximum likelihood aims at finding the parameters $\boldsymbol{\theta}$ that maximise $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})$. The $p(\mathcal{D}|\mathcal{M})$ is found by marginalising over $p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M})$ (see Eq. (14)) which implicitly penalises the complexity of the parameter space $\{\boldsymbol{\theta}\}$ and hence protects from overfitting [16].

[5] Conjugate priors are priors that, when conditioned on the data likelihood, result in a posterior of the same distribution family as the prior.

Even though we could specify α_k directly as an additional prior parameter, we rather assign it a hyperprior as in [11] to allow it to automatically adjust to the data. We specify this hyperprior by the gamma distribution

$$p(\alpha_k|a_\alpha, b_\alpha) = \mathrm{Gam}(\alpha_k|a_\alpha, b_\alpha) = \frac{b_\alpha^{a_\alpha}\alpha_k^{(a_\alpha-1)}}{\Gamma(a_\alpha)}\exp(-a_\alpha\alpha_k), \tag{9}$$

with parameters $a_\alpha = 10^{-2}$ and $b_\alpha = 10^{-4}$ to keep it uninformative.

Gating Network Priors. In the prospect of applying Laplace approximation to the gating network model Eq. (6), resulting in a Gaussian, we apply conjugate Gaussian priors to the gating weights, given by

$$\begin{aligned} p(\boldsymbol{v}_k|\beta_k) &= \mathcal{N}(\boldsymbol{v}_k|\boldsymbol{0}, \beta_k^{-1}\boldsymbol{I}) \\ &= \left(\frac{\beta_k}{2\pi}\right)^{D_V/2} \exp\left(-\frac{\beta_k}{2}\boldsymbol{v}_k^T\boldsymbol{v}_k\right). \end{aligned} \tag{10}$$

This again corresponds to the assumption of having gating weight vectors with small and independent elements. β_k is again modelled by a hyperprior, given by the gamma distribution

$$p(\beta_k|a_\beta, b_\beta) = \mathrm{Gam}(\beta_k|a_\beta, b_\beta) = \frac{b_\beta^{a_\beta}\beta_k^{(a_\beta-1)}}{\Gamma(a_\beta)}\exp(-a_\beta\beta_k), \tag{11}$$

which parameter values $a_\beta = 10^{-2}$ and $b_\beta = 10^{-4}$ to keep the hyperprior uninformative.

Model Structure Prior. Recalling that $\mathcal{M} = \{K, \boldsymbol{M}\}$ is the number of classifiers K and their matching functions $\boldsymbol{M}$, the model structure prior $p(\mathcal{M})$ in Eq. (7) lets us specify any prior belief we have about the number of classifiers and their matching functions, such as that the number of classifiers is certainly not infinite. At first thought it might seem reasonable to specify it to express that every possible unique model structure is equally likely to represent the data. One should note, however, that the number of possible unique model structures for a fixed number of classifiers K grows exponentially with K. Thus, putting a uniform prior on the model structure space will put an implicit bias on model structures with a large number of classifiers.

The contrary, which is to put a uniform prior on the number of classifiers rather than the unique model structures, leads to a bias *against* particular model structures with higher numbers of classifiers, as for those there exist more possible model structures. Thus, how to appropriately define $p(\mathcal{M})$ is a topic of further investigation, but for now we have chosen to specify it by

$$p(\mathcal{M}) \propto \frac{1}{K!}, \tag{12}$$

to capture that in most LCS implementations the same model structure with K classifiers can be specified in $K!$ different ways by simply reordering the classifiers.

A summary of the full Bayesian model including its priors and hyperpriors is given in App. A. The variable dependency structure that shows the different random variables depend on each other is shown in Fig. 1.

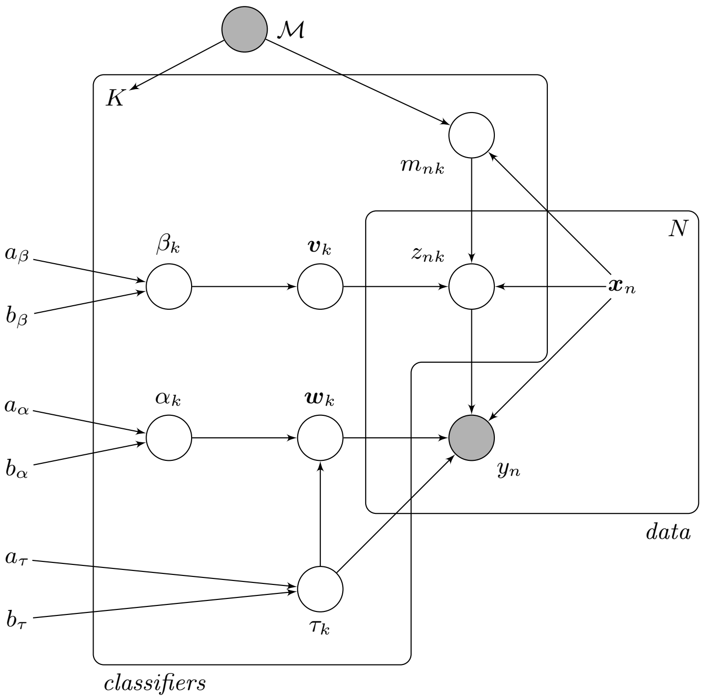

Fig. 1. Directed graphical model of the Bayesian LCS model. The circular nodes are random variables, which are observed when shaded. Labels without nodes are either constants or adjustable parameters. The boxes are "plates", comprising replicas of the entities inside them. Note that to train the model we assume the data $\mathcal{D}$ and the model structure $\mathcal{M}$ to be given. Hence, the y_n's and $\mathcal{M}$ are observed random variables, and the $\boldsymbol{x}_n$'s are constants.

3.3 Evaluating Posterior and Model Evidence

In order to evaluate the model structure posterior $p(\mathcal{M}|\mathcal{D})$ by Eq. (7), one needs to know $p(\mathcal{M})$ and $p(\mathcal{D}|\mathcal{M})$. In fact, as the posterior density is only used to compare different model structures, we can equally use the unnormalised log-density, that, by using Eq. (12), is given by

$$\ln p(\mathcal{M}|\mathcal{D}) = \ln p(\mathcal{D}|\mathcal{M}) - \ln K! + \text{const.}, \tag{13}$$

where the constant term represents the logarithm of the normalisation constant. To get the *model evidence* $p(\mathcal{D}|\mathcal{M})$ we need to evaluate

$$p(\mathcal{D}|\mathcal{M}) = \int_\theta p(\mathcal{D}|\boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M}) \mathrm{d}\boldsymbol{\theta}, \tag{14}$$

where $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{V}, \boldsymbol{\alpha}, \boldsymbol{\beta}\}$ denotes the parameters of a model with structure $\mathcal{M}$, and $p(\boldsymbol{\theta}|\mathcal{M})$ represents their priors, as we have specified above. Unfortunately, there is no closed-form solution to Eq. (14) and so we have to either apply sampling techniques to sample from the posterior or resort to approximating it. We have decided for the latter, as sampling techniques are slower and would prohibit the quick evaluation of a large number of model structure, which is essential for LCS.

Variation Bayesian Inference with Factorial Distributions. Our goal is, on one hand, to find a variational distribution $p(\boldsymbol{U})$ that approximates the true parameter posterior $p(\boldsymbol{U}|\mathcal{D}, \mathcal{M})$ and, on the other hand, to get the model evidence $p(\mathcal{D}|\mathcal{M}) \equiv p(\boldsymbol{Y}|\boldsymbol{X}, \mathcal{M})$, where $\boldsymbol{U} = \boldsymbol{\theta} \cup \{\boldsymbol{Z}\}$ denotes all hidden variables. Variational Bayesian inference is based on the decomposition [20,19]

$$\ln p(\boldsymbol{Y}|\boldsymbol{X}, \mathcal{M}) = \mathcal{L}(q) + \mathrm{KL}(q\|p), \tag{15}$$

$$\mathcal{L}(q) = \int q(\boldsymbol{U}) \ln \frac{p(\boldsymbol{U}, \boldsymbol{Y}|\boldsymbol{X}, \mathcal{M})}{q(\boldsymbol{U})} \mathrm{d}\boldsymbol{U}, \tag{16}$$

$$\mathrm{KL}(q\|p) = -\int q(\boldsymbol{U}) \ln \frac{p(\boldsymbol{U}|\boldsymbol{X}, \boldsymbol{Y}, \mathcal{M})}{q(\boldsymbol{U})} \mathrm{d}\boldsymbol{U}, \tag{17}$$

which holds for any choice of the variational distribution q. As the Kullback-Leibler divergence $\mathrm{KL}(q\|p)$ is always non-negative, and zero if and only if $q(\boldsymbol{U}) = p(\boldsymbol{U}|\boldsymbol{X}, \boldsymbol{Y}, \mathcal{M})$, the variational bound $\mathcal{L}(q)$ is a lower bound on $\ln p(\boldsymbol{Y}|\boldsymbol{X}, \mathcal{M})$ and only equivalent to the latter if $q(\boldsymbol{U})$ is the true posterior $p(\boldsymbol{U}|\boldsymbol{X}, \boldsymbol{Y}, \mathcal{M})$. Hence, we can approximate the posterior by maximising the lower bound $\mathcal{L}(q)$, which brings the variational distribution closer to the true posterior and at the same time gives us an approximation of the model evidence by $\mathcal{L}(q) \leq \ln p(\boldsymbol{Y}|\boldsymbol{X}, \mathcal{M})$.

To make this approach tractable, we need to choose a family of distributions $q(\boldsymbol{U})$ that gives an analytical solution. A frequently used approach (for example, [17,11]) that is sufficiently flexible to give a good approximation to the true posterior is to use the set of distributions that factorises with respect to disjoint groups $\boldsymbol{U}_i$ of variables $q(\boldsymbol{U}) = \prod_i q_i(\boldsymbol{U}_i)$, which allows us to maximise $\mathcal{L}(q)$ with respect to each group of hidden variables while keeping the other ones fixed. This results in

$$\ln q_i^*(\boldsymbol{U}_i) = \mathbb{E}_{i \neq j}(\ln p(\boldsymbol{U}, \boldsymbol{Y}|\boldsymbol{X}, \mathcal{M})) + \text{const..}, \tag{18}$$

when maximising with respect to $\boldsymbol{U}_i$, where the expectation is taken with respect to all hidden variables except for $\boldsymbol{U}_i$, and the constant term is the logarithm of the normalisation constant of q_i^* [19].

Applying variational Bayesian inference to the Bayesian LCS model is a long-winded and complex process that we will not illustrate here, but is described in more detail in [21]. An analytical solution is only acquired if the model has a conjugate-exponential structure, which is violated by the gating network. Thus, we have applied Laplace approximation to its distribution, of which the details can also be found in [21]. Overall, the procedure results in a closed-form expression for the variational bound $\mathcal{L}(q)$ that can be improved incrementally and can replace the model evidence $p(\mathcal{D}|\mathcal{M})$ in Eq. (7) to approximate the model structure posterior.

3.4 Summarising the Approach

Our starting point was to apply model selection to find the set of classifiers that model the data best, that is, without overfitting and underfitting. Approaching the problem by Bayesian model selection, the problem becomes the one of finding a set of classifiers $\mathcal{M}$ that maximises $p(\mathcal{M}|\mathcal{D})$. Given an efficient method to evaluate $p(\mathcal{M}|\mathcal{D})$ for any $\mathcal{M}$, any global optimisation method can be applied to search the space of possible sets of classifiers to maximise $p(\mathcal{M}|\mathcal{D})$.

To get an expression for $p(\mathcal{M}|\mathcal{D})$, or more specifically for the unnormalised $\ln p(\mathcal{M}|\mathcal{D})$ that can equally be used to compare different $\mathcal{M}$, we have applied variational Bayesian inference. This results in an approximation $\mathcal{L}(q)$ to $\ln p(\mathcal{D}|\mathcal{M})$ which can be used in Eq. (13) to approximate $\ln p(\mathcal{M}|\mathcal{D})$. Overall, this is sufficient to implement simple algorithms that search for the optimal set of classifiers for some data, with respect to the previously defined LCS model.

4 But..., Does it Work?

To illustrate that the introduced LCS design methodology leads to useful LCS implementations, we demonstrate the performance of the introduced LCS model and its training on two simple one-dimensional regression tasks. We understand that the brevity of the presentation does not allow the results to be replicated, but a forthcoming publication will, on one hand, give all the details that are required for replication and, on the other hand, present further results.

In all experiments we have used linear classifiers with input vectors $\boldsymbol{x} = (1, x)^T$ for input x, causing the model to by represented by a straight line.

4.1 Model Structure Search

To search the space of possible sets of classifiers we have applied a genetic algorithm (GA) to create a Pittsburgh-style LCS. The individuals represent model structures, and their fitness is the approximated model structure posterior, evaluated by Eq. (13). Thus, by searching for the fittest individual, the GA aims at finding the set of classifiers that maximises $p(\mathcal{M}|\mathcal{D})$.

We have used a rather small population of 20 individuals, and tournament selection with a tournament size of 5 individuals. The individuals are of variable length, and uniform crossover is performed by random sampling without replacement from the combined set of matching functions of two selected individuals. The mutation operator is dependent on the representation of the matching function and will not be detailed. In both experiments, the best individual of any of the 500 training epochs is reported.

To demonstrate that any global optimisation method can be used to find adequate model structures we have performed similar experiments using Monte Carlo Markov Chain (MCMC) methods, analogous to how it was applied by Chipman et al. in [22]. The found model structures were about the same as when applying the GA for model structure search, but we expect MCMC to perform worse in more complex problems where the GA can be expected to exploit building blocks in the solution structure. More details on applying MCMC to model structure search are given in [21].

4.2 Approximating a Generated Function

To test if the method correctly identifies the model structure when the data was generated in conformation to the LCS model assumptions, we have generated such data by combining the models of three localised classifiers with added Gaussian noise. To demonstrate the LCS model's ability to perform matching by degree, we use radial basis matching function, given by

$$m_k(x) = \exp\left(-(2\sigma_k^2)^{-1}(x - \mu_k)^2\right), \tag{19}$$

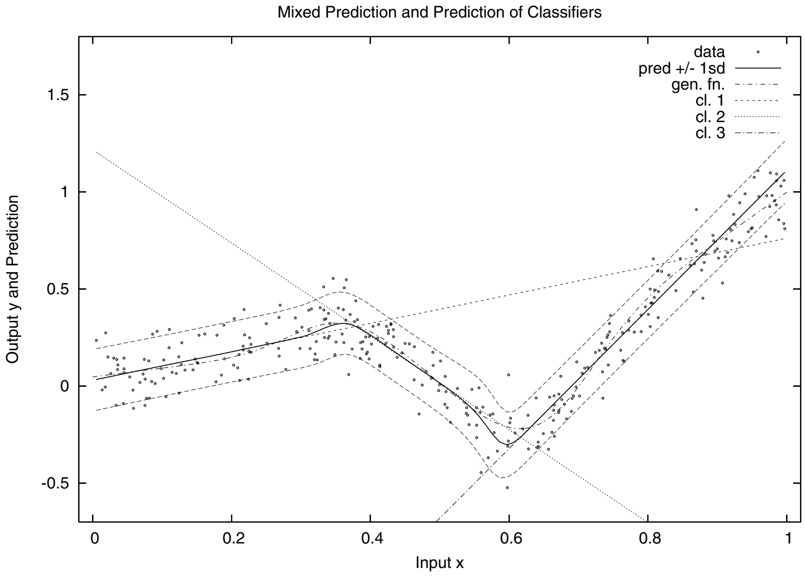

Fig. 2. The plot shows the mean of the data-generating function and the training data itself. Additionally, it shows the linear models of the 3 identified classifiers and the prediction of the full model with confidence intervals that represent one standard deviation to either side of the prediction. As can be seen, the method identified a model structure that fits the data well and is close in its parameter to the model that generated the data.

which is an unnormalised Gaussian that is centred on μ_k and has variance σ_k^2. Thus the classifier matches input $x = \mu_k$ with probability 1, and all other inputs with a probability that decreases in proportion to the distance from μ_k.

The best model structure found in a single run of the GA is shown in Fig. 2, illustrating the data, the classifiers in the model structure, and the mean prediction with confidence intervals. The latter is an additional feature of the model-based approach: due to the probabilistic model, we can make clear statements about the confidence of the prediction. As can be seen, the found model structure represents the data well. Additionally, the model structure parameters are fairly close to the ones that were used to generate the data, and we do not expect the search to find a perfect match as the model structure space is infinite.

4.3 Variable Measurement Noise

XCS seeks for classifiers that feature a mean absolute error close to a preset minimum error ϵ_0, leading to classifier models with approximately equal variances. The introduced LCS model is more flexible by allowing the classifier models to have different variances, depending on the given data. To test if this feature can be exploited we generate data where the level of noise varies over the input range. More specifically, the target function is for $-1 \leq x \leq 1$ given by $f(x) = -1-2x+\mathcal{N}(0, 0.6)$ if $x < 0$, and is $f(x) = -1+2x+\mathcal{N}(0, 0.1)$ otherwise, resulting in a V-shaped function with two different noise levels.

To let the classifiers match distinct areas of the input space we use interval matching with soft boundaries to indicate that in the light of finite data we can never be certain about where the interval boundaries lie. Given that classifier k matches the interval $[l_k, u_k]$, its matching function is given by

$$m_k(x) = \begin{cases} \exp\left(-\frac{1}{2\sigma_k^2}(x - l'_k)^2\right) & \text{if } x < l'_k, \\ \exp\left(-\frac{1}{2\sigma_k^2}(x - u'_k)^2\right) & \text{if } x > u'_k, \\ 1 & \text{otherwise,} \end{cases} \tag{20}$$

where $l'_k \approx l_k+0.0566b_k$, $u'_k \approx u_k+0.0566b_k$, $\sigma_k \approx 0.0662b_k$, and b_k is the interval width $b_k = u_k - l_k$. This causes the classifier to match the interval $[l'_k, u'_k]$ with probability 1, with unnormalised Gaussian boundaries that are with one standard deviation inside $[l_k, u_k]$ and otherwise outside of the interval. Additionally, 95% of the area underneath $m_k(x)$ is inside $[l_k, u_k]$.

The training data and best model structure found in a single training run with 500 epochs is shown in Fig. 3 and clearly shows by the width of its confidence intervals that the identified model features different noise levels in different areas of the input space. This illustrates that the method does not only protect from overfitting at the model structure level, but also at the classifier level by correctly separating the underlying pattern from the data.

The results of both experiments suggest that the method we have derived works as expected. Still, we want to emphasise that the developed method is only an example that show that the proposed model-based design methodology

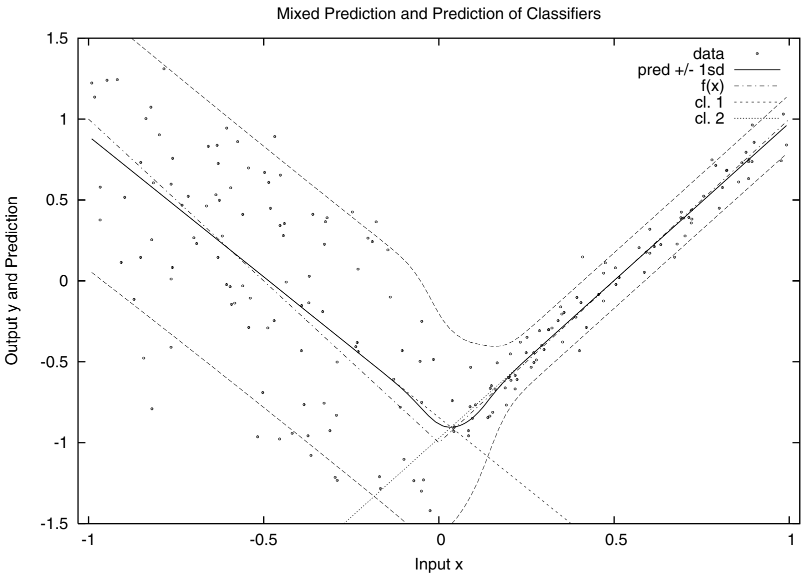

Fig. 3. The plot shows the mean of the data-generating function and the training data itself. In addition, it shows the linear models of the 2 identified classifiers and the prediction of the full model with confidence intervals that represent one standard deviation to either side of the prediction. These confidence intervals clearly show that the model is able to handle and model data whose noise level varies over the input space.

not only provides us with a good understanding of the model underlying different LCS variants, but also leads to useful and well-understood implementations.

5 Summary and Conclusions

We have proposed a new methodology for the design of LCS that is based on first specifying the model underlying a set of classifiers and then applying standard machine learning methods to train this model and identify a good set of classifiers. The advantages of this approach are manifold, such as i) having a formal definition of what it means for a set of classifiers to be the optimal set with respect to some data; ii) conceptually separating the LCS model from its training; iii) making explicit the assumptions the LCS model makes about the data; iv) establishing strong links to other machine learning methods through the application of standard machine learning methods for LCS training; and v) advancing the theoretical understanding of LCS by transferring the understanding of the applied machine learning methods to LCS. Furthermore, the methodology promises a high degree of flexibility in designing new LCS through changing the structure of the LCS model, or applying different machine learning methods for their training.

To demonstrate that the methodology is feasible, we have introduced a model for LCS and have described how it can be trained. In this work we have particularly focused on the design of the model that is — closely related to XCS — specified as combining a set of independently trained localised models to form a global prediction over the whole input space. This characterisation allows it to be linked to a generalisation of the well-known Mixtures-of-Experts model, which puts it on a strong probabilistic foundation. For the identification of good model structures, that is, good sets of classifiers, we have used Bayesian model selection that results in the aim of maximising the probability of the model structure given the data. For training we have use the variational Bayesian method to find the model structure posterior, and have used a GA to search the space of possible model structures, resulting in a Pittsburgh-style LCS. To illustrate that the sets of classifiers identified that way indeed represent the data well, we have shown the methods performance on two simple regression tasks.

The work has wide implications and it is opening up significant future research directions, amongst which are i) to create a suitable LCS model specialised on classification by changing the classifier models from regression to classification models; ii) to compare and contrast LCS that train their classifiers independently to those that do not; iii) to design suitable methods, eventually using the additional probabilistic information that is available through the model, to apply the same methodology to design Michigan-style LCS; iv) analysing the suitability of the regression model for approximating the value function of reinforcement learning tasks.

Acknowledgements

We would like to thank Christopher M. Bishop, Markus Svensén and Matthew Beal for useful comments to our questions regarding the variational Bayesian approach.

References

1. Wilson, S.W.: Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2), 149–175 (1995)
2. Butz, M.V., Pelikan, M.: Analyzing the Evolutionary Pressures in XCS. In: [23], pp. 935–942.
3. Butz, M.V., Goldberg, D.E., Tharakunnel, K.: Analysis and Improvement of Fitness Exploitation in XCS: Bounding Models, Tournament Selection and Bilateral Accuracy. Evolutionary Computation 11, 239–277 (2003)
4. Butz, M.V., Kovacs, T., Lanzi, P.L., Wilson, S.: Toward a Theory of Generalization and Learning in XCS. IEEE Transaction on Evolutionary Computation 8, 28–46 (2004)
5. Drugowitsch, J., Barry, A.M.: A Formal Framework for Reinforcement Learning with Function Approximation in Learning Classifier Systems. Technical Report 2006–02, University of Bath, U.K (January 2006)
6. Wilson, S.W.: Function Approximation with a Classifier System. In: [23], pp. 974–981.

7. Jacobs, R.A., Jordan, M.I., Nowlan, S., Hinton, G.E.: Adaptive mixtures of local experts. Neural Computation 3, 1–12 (1991)
8. Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Computation 6, 181–214 (1994)
9. McCullach, P., Nelder, J.A.: Generalized Linear Models. Monographs on Statistics and Applied Probability. Chapman and Hall, Boca Raton (1983)
10. Drugowitsch, J., Barry, A.M.: Mixing Independent Classifiers. In: [24]
11. Bishop, C.M., Svensén, M.: Bayesian Hierarchical Mixtures of Experts. In: Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI 2003), pp. 57–64. Morgan Kaufmann, San Francisco (2003)
12. Brown, G., Kovacs, T., Marshall, J.: UCSPv: Principled Voting in UCS Rule Populations. In: [24], pp. 1774–1782.
13. Grünwald, P.D.: A tutorial introduction to the minimum description length. In: Grünwald, P., Myung, J., Pitt, M.A. (eds.) Advances in Minimum Description Length Theory and Applications. Information Processing Series, pp. 3–79. MIT Press, Cambridge (2005)
14. Vapnik, V.N.: An Overview of Statistical Learning Theory. IEEE Transactions on Neural Networks 10(5), 988–999 (1999)
15. Ueda, N., Ghahramani, Z.: Bayesian model search for mixture models based on optimizing variational bounds. Neural Networks 15, 1223–1241 (2002)
16. MacKay, D.J.C.: Bayesian interpolation. Neural Computation 4(3), 415–447 (1992)
17. Waterhouse, S., MacKay, D., Robinson, T.: Bayesian Methods for Mixtures of Experts. In: Touretzky, D.S.T., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems 8, pp. 351–357. MIT Press, Cambridge (1996)
18. Waterhouse, S.: Classification and Regression using Mixtures of Experts. PhD thesis, Department of Engineering, University of Cambridge (1997)
19. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, Heidelberg (2006)
20. Jaakkola, T.S.: Tutorial on variational approximation methods. In: Opper, M., Saad, D. (eds.) Advanced Mean Field Methods, pp. 129–160. MIT Press, Cambridge (2001)
21. Drugowitsch, J., Barry, A.M.: Generalised Mixtures of Experts, Independent Expert Training, and Learning Classifier Systems. Technical Report 2007–02, University of Bath, U.K (2007)
22. Chipman, H.A., George, E.I., McCulloch, R.E.: Bayesian CART Model Search. Journal of the American Statistical Association 93(443), 935–948 (1998)
23. Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.): GECCO-2001: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco (2001)
24. Thierens, D., Beyer, H.G., Birattari, M., Bongard, J., Branke, J., Clark, J.A., Cliff, D., Congdon, C.B., Deb, K., Doerr, B., Kovacs, T., Kumar, S., Miller, J.F., Moore, J., Neumann, F., Pelikan, M., Poli, R., Sastry, K., Stanley, K.O., Stützle, T., Watson, R.A., Wegener, I. (eds.): GECCO-2007: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation Conference 2007, vol. 2. ACM Press, New York (2007)

A The Bayesian LCS Model

The following table gives an overview over the Bayesian LCS model with all its components.

Data, Model Structure, and Likelihood	
	N observations $\{(\boldsymbol{x}_n, y_n)\}$, $\boldsymbol{x}_n \in \mathcal{X} = \mathcal{R}^{D_\mathcal{X}}$, $y_n \in \mathcal{Y} = \mathcal{R}$
	Model structure $\mathcal{M} = \{K, \boldsymbol{M}\}, k = 1, \dots, K$
	K classifiers
	Matching functions $\boldsymbol{M} = \{m_k : \mathcal{X} \to [0,1]\}$
Likelihood	$p(\boldsymbol{Y}\|\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\tau}, \boldsymbol{Z}) = \prod_{n=1}^N \prod_{k=1}^K p(y_n\|\boldsymbol{x}_n, \boldsymbol{w}_k, \tau_k)^{z_{nk}}$
Classifiers	
Variables	Weight matrices $\boldsymbol{W} = \{\boldsymbol{w}_k\}, \boldsymbol{w}_k \in \mathcal{R}^{D_\mathcal{X}}$
	Noise precisions $\boldsymbol{\tau} = \{\tau_k\}$
	Weight shrinkage priors $\boldsymbol{\alpha} = \{\alpha_k\}$
	Noise precision prior parameters a_τ, b_τ
	α-hyperprior parameters a_α, b_α
Model	$p(y\|\boldsymbol{x}, \boldsymbol{w}_k, \tau_k) = \mathcal{N}(y\|\boldsymbol{w}_k^T \boldsymbol{x}, \tau_k^{-1})$
Priors	$p(\boldsymbol{w}_k, \tau_k\|\alpha_k) = \mathcal{N}(\boldsymbol{w}_k\|\boldsymbol{0}, (\alpha_k \tau_k)^{-1}\boldsymbol{I})\mathrm{Gam}(\tau_k\|a_\tau, b_\tau)$
	$p(\alpha_k) = \mathrm{Gam}(\alpha_k\|a_\alpha, b_\alpha)$
Gating	
Variables	Latent variables $\boldsymbol{Z} = \{\boldsymbol{z}_n\}$, $\boldsymbol{z}_n = (z_{n1}, \dots, z_{nK})^T \in \{0,1\}^K$, 1-of-$K$
	Gating weight vectors $\boldsymbol{V} = \{\boldsymbol{v}_k\}$, $\boldsymbol{v}_k \in \mathcal{R}^{D_\mathcal{V}}$
	Gating weight shrinkage priors $\boldsymbol{\beta} = \{\beta_k\}$
	β-hyperprior parameters a_β, b_β
Model	$p(\boldsymbol{Z}\|\boldsymbol{X}, \boldsymbol{V}, \boldsymbol{M}) = \prod_{n=1}^N \prod_{k=1}^K g_k(\boldsymbol{x}_n)^{z_{nk}}$
	$g_k(\boldsymbol{x}) \equiv p(z_k = 1\|\boldsymbol{x}, \boldsymbol{v}_k, m_k) = \frac{m_k(\boldsymbol{x}) \exp(\boldsymbol{v}_k^T \vartheta(\boldsymbol{x}))}{\sum_{j=1}^K m_j(\boldsymbol{x}) \exp(\boldsymbol{v}_j^T \vartheta(\boldsymbol{x}))}$
Priors	$p(\boldsymbol{v}_k\|\beta_k) = \mathcal{N}(\boldsymbol{v}_k\|\boldsymbol{0}, \beta_k^{-1}\boldsymbol{I})$
	$p(\beta_k) = \mathrm{Gam}(\beta_k\|a_\beta, b_\beta)$