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 explicitly identify the probabilistic model underlying LCS by linking it to a generalisation of the common Mixture-of-Experts model. Having an explicit representation of the model not only puts LCS on a strong statistical foundation and identifies the assumptions that the model makes about the data, but also allows us to use off-the-shelf training methods to train it. We show how to exploit this advantage by embedding the LCS model into a fully Bayesian framework that results in an objective function for a set of classifiers, effectively turning the LCS training into a principled optimisation task. A set of preliminary experiments demonstrate the feasibility of this approach.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Probabilistic Algorithms; G.1.2 [**Numerical Analysis**]: Approximation—*Nonlinear approximation*

General Terms

Theory

Keywords

Learning Classifier Systems, Mixtures-of-Experts, Bayesian Model Selection, Independently Trained Local Models, XCS

1. INTRODUCTION

While there are a variety of different Machine Learning methods, they all share something essential: Each of them has an underlying model that makes explicit the assumptions the method makes about the data that it models. In addition, the type of model determines how it can be trained, and its limitations.

LCS is often considered to be a Machine Learning method, but is usually described algorithmically rather than from a model-based perspective, despite its significance for understanding the assumptions and limitations of modern LCS. In this work we will demonstrate how LCS can be designed from a model-based perspective — by first identifying the model, and then using off-the-shelf training methods to train it. Through the model we also make explicit the assumptions that are made about the data-generating processes, and how the model links LCS to the common Mixtures-of-Experts model.

Due to space constraints we will mainly focus on the structure of the model itself and its relation to Mixture-of-Experts. As most of the LCS research is focused on model training, 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, and describe how this can be used to turn the search for a good set of classifiers into an optimisation problem. Finally, 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. 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[1] $\boldsymbol{\theta}$. 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

[1]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*.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

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 $\mathcal{X} = \mathcal{R}^{D_X}$ 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 $\hat{f} : \mathcal{X}_k \to \mathcal{Y}$, 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 $\hat{f} : \mathcal{X} \to \mathcal{Y}$, 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 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 over-fit) 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 Section 6. 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) [6, 7] 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.

3. MIXTURES OF EXPERTS

Mixture of Experts are 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 $\boldsymbol{z} = (z_1, \ldots, z_K)^T$ 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[2]; 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 $\boldsymbol{x}$ 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 computation of the posteriors $p(z_{nk} = 1|\boldsymbol{x}_n, y_n, \boldsymbol{v}_k)$ is 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 $\boldsymbol{Z}$. 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

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

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.

4. 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 ϑ is a 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 $m_k \equiv p(m_{nk}=1|\boldsymbol{x}_n)$, 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 both over 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 [5], current commonly used LCS — such as XCS [13] 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 (as the normalised fitness in XCS), but having a better probabilistic justification like the multinomial logit model is certainly an advantage.

5. FITTING THE MODEL

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 [3]. 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 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 so 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 [8], 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, mak-

ing it independent 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.

6. MODEL STRUCTURE SEARCH

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 over-fit the data arbitrarily.

We approach the problem of defining the quality of a set of classifiers by facilitating the link between LCS and MoE that allows us to use standard ML model selection methods. 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 method or Statistical Learning Theory.

Bayesian model selection is a method for model selection founded in Bayesian statistics which has fortunately already been applied to the standard MoE model [3, 10]. 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. 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 $p(\mathcal{M}|\mathcal{D})$.

To apply Bayesian model selection to LCS, we have embedded the generalised MoE model in a fully Bayesian framework similar to the one in [12, 11]. However, due to its complexity it does not allow for a closed-form solution to the posterior $p(\mathcal{D}|\mathcal{M})$. While sampling methods such as Markov Chain Monte Carlo (MCMC) can generate accurate solutions in such cases, they are slow and therefore not suitable for our task. Instead we have applied the variational Bayesian method [2] to provide a good approximation to the posterior, similar to its application to the standard MoE model in [12, 3, 10]. As its development is long-winded and complex, we will neither present the fully Bayesian model nor its variational approximation here, but postpone it to a future publication.

In summary, we can identify a good set of classifiers by maximising the model structure posterior $p(\mathcal{M}|\mathcal{D})$, which we can find by embedding the generalised MoE in a Bayesian framework and use variational methods to find the posterior $p(\mathcal{D}|\mathcal{M})$. This maximisation problem can be solved by applying a GA to a population of classifier sets with $p(\mathcal{M}|\mathcal{D})$ as the fitness measure of an individual — in the spirit of Pittburgh-style LCS. Alternatively, we can derive an incremental version of the update equations which is simplified through the Bayesian formulation of the LCS model, and use the single objective function of the Bayesian model selection approach to derive the fitness function in Michigan-style LCS.

7. BUT..., DOES IT WORK?

As a proof-of-concept, we have tested our approach on two simple but fairly noisy non-linear regression tasks. Rather than using a GA to optimise the model structure, we have applied an MCMC method to sample from the model structure posterior Eq. (7). As such a sampling strategy spends more time in high-probability regions of the posterior, it has a tendency to maximise this posterior, but requires storing the best solution found so far due to its stochastic search nature. In particular, we have used the Metropolis-Hastings algorithm similar to the one used in [4] that on each sampling step either adds a classifier to the population, removes one, or changes the matching function of a classifier.

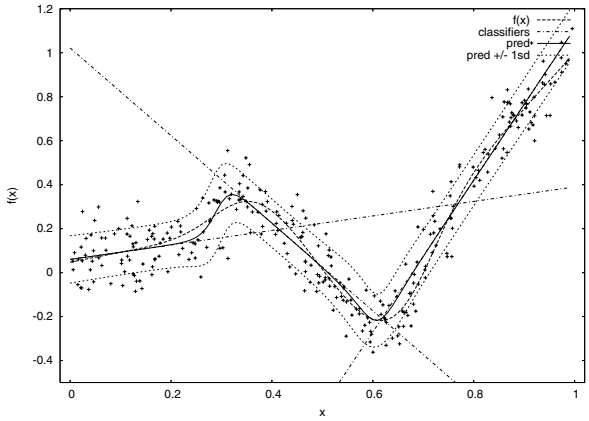

Figure 1: The original function $f_1(x)$ generated by a localised mixture of 3 straight lines, and the noisy training data that was available to the LCS. As can be seen, the method has identified all 3 classifiers correctly. The error bars show one standard deviation from the model prediction.

While LCS usually use binary matching, that is, matching functions that either return 0 or 1, we have chosen to use continuous matching functions to demonstrate that our method extends naturally to such functions. In particular, we have used Gaussian radial basis matching functions that are in a 1D setting fully specified by their mean μ and standard deviation σ^2, and are defined by

$$m_k(x) = \exp\left(-(2\sigma^2)^{-1}(x-\mu)^2\right). \tag{8}$$

To test the method's ability to correctly identify the number of classifiers to use, we have generated the 1D function

f_1 as shown in Figure 1 with added Gaussian noise from 3 straight lines as if it were generated by 3 localised classifiers. As can be seen, the method correctly identifies both the number of classifiers and their location.

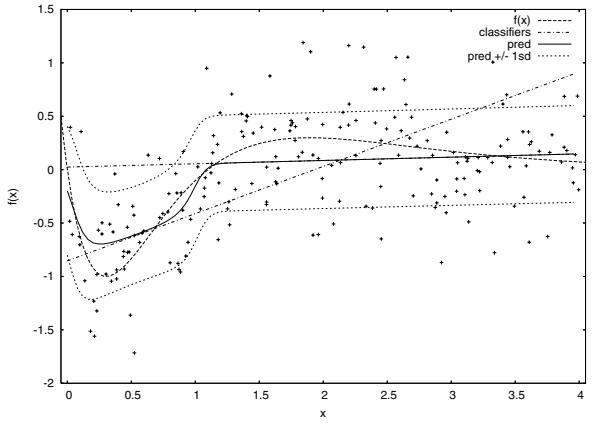

Figure 2: The original function $f_2(x)$, the training data with added noise, the prediction of the separate classifiers and the mixed prediction. The error bars show a single standard deviation to both sides of the model prediction

In an additional experiment, we have tested the performance on an artificial data set used in [12] by sampling from $f_2(x) = 4.26(e^{-1} - 4e^{-2x} + 3e^{-3x}) + \mathcal{N}(0, 2)$ over the range [0, 4]. Surprisingly, as can be seen in Figure 2, the curve was fitted more compactly with one classifier interleaving the prediction of another global expert, where would have expected the use of 3 classifiers that perform a piecewise linear fit.

These results suggest that the method we have derived works as expected. Additionally we have shown that the model structure search can be performed by any stochastic optimisation method, one of which is a GA. While the MCMC search worked for these simple examples, experiments in high-dimensional input spaces have failed to yield satisfying results within reasonable time. For such cases, the GA still might be the best choice, but further experiments are pending.

8. SUMMARY AND CONCLUSIONS

We have developed LCS from a principled foundation by characterising them as a Machine Learning method that searches an adequate model structure for a model that combines a set of localised models to form a global prediction over the whole input space. This characterisation allowed them to be linked to a generalisation of the well-known Mixture-of-Experts model, which puts the LCS model on a strong probabilistic foundation. To identify good model structures that adequately capture the pattern in the data but does not model the random noise, we have used Bayesian model selection together with variational Bayesian methods to keep the method tractable. We have demonstrated that the method works by testing it on a set of simple proof-of-concept regression tasks.

The immediate achievement are: i) the development of a probabilistic basis for LCS; ii) clear identification of what assumptions are made about the data that is modelled; iii) strong links to other Machine Learning methods; iv) identification of the task of finding a good set of classifiers as a model selection task; v) solving that task by providing a Bayesian formulation of an LCS population; vi) making this formulation computationally feasible by assuming variational Bayesian methods; vii) allowing for non-binary matching that has a clear probabilistic interpretation; viii) demonstrating that the GA in LCS can be replaced by any other stochastic search methods.

The work has wide implications, as well as is opening up significant future research, amongst which is i) the design of a suitable GA that is efficient in searching the model structure space by using the probabilistic model information that is available – effectively creating a Pittsburgh-style LCS; ii) creating an incremental implementation from the Bayesian update rules and extracting classifier fitness information from the model structure posterior to create a Michigan-style LCS; iii) replacing the current univariate regression model in classifiers with other models, such as multivariate regression models, or classification models; 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.

9. REFERENCES

[1] *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM Press, 2007.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[3] C. M. Bishop and M. Svensén. Bayesian Hierarchical Mixtures of Experts. In *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 57–64, San Francisco, CA, 2003. Morgan Kaufmann.

[4] H. A. Chipman, E. I. George, and R. E. McCulloch. Bayesian CART Model Search. *Journal of the American Statistical Association*, 93(443):935–948, sep 1998.

[5] J. Drugowitsch and A. M. Barry. Mixing Independent Classifiers. [1].

[6] R. A. Jacobs, M. I. Jordan, S. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:1–12, 1991.

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

[8] J. A. R. Marshall, G. Brown, and T. Kovacs. UCSpv: Principled Voting in UCS Rule Populations. [1].

[9] P. McCullach and J. A. Nelder. *Generalized Linear Models*. Monographs on Statistics and Applied Probability. Chapman and Hall, 1983.

[10] N. Ueda and Z. Ghahramani. Bayesian model search for mixture models based on optimizing variational bounds. *Neural Networks*, 15:1223–1241, 2002.

[11] S. Waterhouse. *Classification and Regression using Mixtures of Experts*. PhD thesis, Department of Engineering, University of Cambridge, 1997.

[12] S. Waterhouse, D. MacKay, and T. Robinson. Bayesian Methods for Mixtures of Experts. In D. S. T. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 351–357. MIT Press, 1996.

[13] S. W. Wilson. Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2):149–175, 1995. http://prediction-dynamics.com/.