LETTER Communicated by Jo-Anne Ting

Online Learning of Single and Multivalued Functions with an Infinite Mixture of Linear Experts

Bruno Damas
bdamas@isr.ist.utl.pt
Instituto de Sistemas e Robótica IST, 1049-001 Lisboa, Portugal,
and Escola Superior de Tecnologia IPS, 2910-761 Setúbal, Portugal

José Santos-Victor
jasv@isr.ist.utl.pt
Instituto de Sistemas e Robótica IST, 1049-001 Lisboa, Portugal

We present a supervised learning algorithm for estimation of generic input-output relations in a real-time, online fashion. The proposed method is based on a generalized expectation-maximization approach to fit an infinite mixture of linear experts (IMLE) to an online stream of data samples. This probabilistic model, while not fully Bayesian, can efficiently choose the number of experts that are allocated to the mixture, this way effectively controlling the complexity of the resulting model. The result is an incremental, online, and localized learning algorithm that performs nonlinear, multivariate regression on multivariate outputs by approximating the target function by a linear relation within each expert input domain and that can allocate new experts as needed. A distinctive feature of the proposed method is the ability to learn multivalued functions: one-to-many mappings that naturally arise in some robotic and computer vision learning domains, using an approach based on a Bayesian generative model for the predictions provided by each of the mixture experts. As a consequence, it is able to directly provide forward and inverse relations from the same learned mixture model. We conduct an extensive set of experiments to evaluate the proposed algorithm performance, and the results show that it can outperform state-of-the-art online function approximation algorithms in single-valued regression, while demonstrating good estimation capabilities in a multivalued function approximation context.

1 Introduction

Online and incremental learning algorithms that can efficiently deal with high-dimensional streams of data have become increasingly required in the robotics community. As humanoid robots evolve and become more complex, their sensorimotor relations become more difficult to understand,

Neural Computation **25, 1–48** (2013)
doi:10.1162/NECO_a_00510

leading to situations where analytical models fail to provide accurate approximations of these sensorimotor maps. These situations range from lack of knowledge of certain hard-to-measure physical parameters (e.g., friction) to highly nonlinear physical interactions, such as actuator nonlinearities and unmodeled mass distributions (Peters & Schaal, 2006; Nguyen-Tuong & Peters, 2008). Resorting to modern learning techniques is, in these cases, the only way to provide these systems with the necessary representation capability. Approaches such as neural networks (Bishop, 1995), memory-based learning (Hastie & Loader, 1993; Atkeson, Moore, & Schaal, 1997), support vector regression (SVR) (Smola & Schölkopf, 2004) and gaussian processes regression (GPR) (Rasmussen & Williams, 2006), among many others, have been successfully applied to robotic learning tasks in the past few decades.

Many of these learning techniques, however, are not suitable when we desire to achieve a continuous and autonomous operation, such that in a biologically plausible way, the robot continuously learns and adapts as it interacts with the surrounding environment. In this setup, an offline learning phase does not exist. As a consequence, online learning algorithms are needed that can keep their computational demands low in the presence of potentially very large sensorimotor spaces to explore. This rules out many state-of-the-art algorithms like SVR and GPR: these methods fit nonlinear functions globally and require, in their original form, the presence of all training points in memory and a computational effort that grows very fast with the increasing number of samples to process. Other methods, like neural networks, can suffer from slow convergence and destructive interference when learning new data, a problem reported, for example, by Schaal and Atkeson (1998). To overcome such computational burden, one can identify two major trends: methods based on global, sparse approximations (Csató & Opper, 2002; Ma, Theiler, & Perkins, 2003; Quiñonero-Candela & Rasmussen, 2005) try to keep a representative reduced set of training samples to represent the function to approximate, while a second group of methods is based on local approximations of the function to learn (Meeds & Osindero, 2006; Nguyen-Tuong & Peters, 2008). The former class of methods may become inadequate in the presence of large streams of online training data, as it tries to represent the function to learn in a global manner and is particularly sensitive to shifts in the input data distribution.

Local methods have some strong support in neurobiology (see the references in Atkeson et al., 1997, for instance), and their ability to perform localized learning seems more suitable for an online operation: they often can be seen as probabilistic mixtures of models, where each model describes the function to be learned in a particular region of the input space. They have their origins in the mixture of experts (ME) concept (Jacobs, Jordan, Nowlan, & Hinton, 1991; Jordan & Jacobs, 1994), where competing experts, assigned to different zones of the input space, are responsible for generating a corresponding output, being the final prediction produced by a

gating network that combines their outputs. Particularly suited for real-time operation, mixtures of local linear models avoid the need to keep all training points in memory, getting a dramatic increase in the speed of operation while possibly sacrificing the better approximation errors typical of state-of-the-art GP methods. Xu, Jordan, and Hinton (1995) provided one of the earliest examples of such mixture of linear experts architecture, trained using the expectation-maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977); in its simplest form it is an offline algorithm, for which the number of components—number of experts—must be set beforehand. Sato and Ishii (2000) suggest an online version for the ME, based on the introduction of a discount factor in the update of the sufficient statistics vector of the mixture. Training, again, is done using the EM algorithm, although the regularization and allocation of new experts are performed in a somewhat heuristic way. LWPR (Vijayakumar, D'Souza, & Schaal, 2005) and XCSF (Wilson, 2002) are two other popular nonprobabilistic algorithms based on this mixture of linear experts concept. LWPR has been widely used for online, real-time learning of robotic tasks; it uses a gradient descent on the prediction error, based on a stochastic leave-one-out cross-validation algorithm, to adapt the distance metrics of the receptive fields that partition the input space. Within each receptive field, a linear relation from input to output is obtained using an incremental partial least squares algorithm that efficiently deals with redundant and high-dimensional input spaces. XCSF updates the input distance metric of each model resorting to a steady-state genetic algorithm, while each linear model is fitted using a recursive least squares algorithm.

Unfortunately, almost all of the current state-of-the-art learning algorithms fail to deal with multivalued functions; these relations, also known as multifunctions or multimaps, are one-to-many maps where a single input can be associated with one or more different outputs and can be thought of as inverse relations for noninjective functions. They naturally arise in some robotic domains, such as perceptual aliasing in reinforcement learning (Whitehead & Ballard, 1991) or robotic learning from demonstration (Chernova & Veloso, 2008; Grollman & Jenkins, 2010). However, the most evident example of multivalued functions comes from serial and parallel robot kinematics that relate actuated joint space variables to the task space configuration vector of the robot. The forward kinematics for serial robots is straightforward to obtain and consists of a proper, single-valued function of the joint variables, while the inverse kinematics for such type of manipulators usually exhibits multiple joint space solutions for the same task space variable, even for nonredundant robots, where the task and joint space have the same dimension (Craig, 1989). The opposite situation occurs with parallel robots. Now, the same value of joint space variables may correspond to different values of the task space vector. Inverse kinematics, however, is usually easy to obtain and provides only a single solution for every task space configuration (Merlet, 2006).

For many supervised learning algorithms, this lack of ability to obtain different multivalued predictions for the same input query point is a result of assuming the single-valued model during the training phase, either explicitly or by allowing some sort of interference across multiple solution branches. As a consequence, many of the learning methods described can provide only single-valued predictions, and no inverse predictions are directly available from the learned forward model in its original form. D'Souza, Vijayakumar, and Schaal (2001), for instance, used LWPR to learn inverse kinematics for serial robots, but this approach, like other inverse kinematics learning methods, obtains only a single, local inverse solution by application of some sort of constraining or optimization criterion.

Three major approaches to the multivalued prediction problem that do not rely on the single-valued model can be devised. The first approach explicitly takes into account the multivalued model and mostly relies on artificial neural networks, using recurrent neural networks (Tomikawa & Nakayama, 1998), feedforward neural networks (Lee & Lee, 2001; Brouwer, 2004), or regularization networks that provided a direct algebraic representation of the multivalued function to learn (Shizawa, 1994). These implementations, however, were tested only in very low-dimensional toy problems, leaving their performance in more demanding problems an open question.

Alternatively, as Ghahramani and Jordan (1994) suggested, unsupervised learning algorithms can be applied to supervised learning tasks by means of the conditional densities obtained from the learned joint density function over both inputs and outputs. In principle, this makes it possible to obtain multivalued predictions for both forward and inverse relations (Lopes & Damas, 2007). Different techniques exist for learning the joint density function of the data; the mixture model is a popular approach, for which specializations exist such as the mixture of gaussians (Ghahramani & Jordan, 1994) or the mixture of factor analyzers (Ghahramani & Hinton, 1997). One of the major problems with this type of model is the choice of the number of components that constitute the mixture; several extensions to the mixture model exist that are capable of automatically selecting an appropriate number of components. The infinite mixture concept presented by Rasmussen (2000), for instance, is a Bayesian method that assigns a Dirichlet process prior on the mixing proportions of the mixture, responsible for the automatic generation of the correct number of components. This, however, usually requires expensive offline computational training based on Markov chain Monte Carlo sampling methods, not suitable for real-time learning, although some variational techniques can be used to accelerate the training process. Other extensions rely on the EM algorithm, either using a greedy approach to grow the mixture to an appropriate number of components (Vlassis & Likas, 2002) or, using the opposite idea, starting with a large number of components and, using carefully chosen priors, automatically shrinking the number of components of the mixture in a Bayesian fashion

(Figueiredo & Jain, 2002). Unsupervised learning, however, is a more difficult problem than its supervised counterpart and usually conducts to worse results, as it ignores that joint data, apart from noise corruption, lie in a lower-dimensional manifold. This is the main drawback of the unsupervised approach to multivalued learning and prediction, as it typically results in convergence to suboptimal solutions that do not take the problem structure into account.

The last, and perhaps the most promising, approach to multivalued learning is to consider an ME framework, where different experts are allowed to share the same input space regions, while allocated to different solutions in the output space. This is done, for instance, in Rasmussen and Ghahramani (2002) and Meeds and Osindero (2006), using a mixture of gaussian processes (GP) experts, or in Bishop (1994) and Qin and Carreira-Perpinán (2008), resorting to the mixture density network (MDN). These examples, however, fail to deal with two aspects of the learning process that we would like to achieve. First, they are not originally designed for online operation, assuming the training to be done in a batch, offline way. The work of Grollman and Jenkins (2010) is based on the mixture of GP experts of Meeds and Osindero (2006) and changes it to allow for an online operation of the algorithm, using a sparse GP model for each of the experts; however, a compromise between accuracy and speed reduces its flexibility, since training requires performing both a computationally expensive Monte Carlo integration over particles and an equally expensive sparse GP online training, which severely reduces the algorithm performance in medium- to high-dimensional learning problems, as discussed in section 5. MDN does not have such computational cost, but it requires the number of experts to be set beforehand. Second, no explicit inversion of the relation to be learned is available; this means that two distinct models, for inverse and forward prediction, must be separately trained, for instance, when trying to simultaneously learn direct and inverse kinematics.

The infinite mixture of linear experts (IMLE) proposed in this letter builds on the mixture of linear models concept to obtain an online, real-time supervised learning algorithm that is able to learn multivalued functions while allocating experts to the mixture as needed, achieving, for single-valued regression, performance at least comparable to state-of-the-art online learning methods like LWPR. It is, at its core, an infinite version of the model for mixture of linear experts given by Xu et al. (1995) and Sato and Ishii (2000), with a careful choice of priors for some of its parameters and the adoption of a generalized EM approach that allows an online operation based on the concepts present in Neal and Hinton (1999) and Cappé and Moulines (2009) and the automatic allocation of experts to the mixture in order to adapt to the complexity of the function to be learned. The training procedure automatically rejects interference among different multivalued function branches, and by considering simple linear local relations from input to output, it is possible to easily obtain the inverse prediction from each

local expert. The probabilistic model underlying the proposed algorithm is described in section 2, and the training procedure is presented in section 3. This is the first major contribution of this letter.

After the probabilistic model has been trained, an important question that remains is how to generate a possibly multivalued set of predictions for a given query data point. Qin and Carreira-Perpinán (2008), for instance, use an MDN and obtain the set of modes for the resulting conditional distribution. We instead suggest a different approach, based on clustering the individual expert predictions with a predictive probabilistic model that relates multivalued solutions to individual expert predictions. This is the topic of section 4, where we describe how to obtain forward and inverse multivalued predictions for any query, given the current state of the IMLE model. This topic is the second main contribution of this letter. Finally, section 5 provides a large set of experimental evaluations of the proposed algorithm, testing it in both single-valued and multivalued prediction situations. Section 6 provides the concluding remarks.

2 Probabilistic Model

The IMLE assumes the following generative model for a sample point $(\boldsymbol{z}_i, \boldsymbol{x}_i)$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ is the input vector and $\boldsymbol{x}_i \in \mathbb{R}^D$ is the corresponding output response:

$$\boldsymbol{x}_i | \boldsymbol{z}_i, w_{ij}; \boldsymbol{\Theta} \sim \mathcal{N}(\boldsymbol{\mu}_j + \boldsymbol{\Lambda}_j(\boldsymbol{z}_i - \boldsymbol{\nu}_j), \boldsymbol{\Psi}_j), \tag{2.1a}$$

$$\boldsymbol{z}_i | w_{ij}; \boldsymbol{\Theta} \sim \mathcal{N}(\boldsymbol{\nu}_j, \boldsymbol{\Sigma}_j), \tag{2.1b}$$

$$p(w_{ij}; \boldsymbol{\Theta}) = \frac{m_j}{M}, \quad \text{with} \quad M = \sum_{k=1}^{\infty} m_k. \tag{2.1c}$$

Here w_{ij} denotes a latent or hidden indicator variable that equals 1 if data point i was generated by linear model j and 0 otherwise, with $\sum_j w_{ij} = 1$ (sometimes we will use the shorthand notation w_{ij} to denote the event $w_{ij} = 1$, as in the above equations). These variables can be gathered, for each i, in a binary vector $\boldsymbol{w}_i$ whose jth component is equal to w_{ij}. The parameter m_j indicates if expert j is activated, effectively contributing to the mixture. It is equal to 1 if expert j is active and 0 otherwise. Each of the latent indicator variables w_{ij} consequently has an associated probability of $1/M$ if linear model j contributes to the mixture, where M is the total number of active experts. This uniform probability distribution on the latent variables is a more natural approach, in our opinion, to the online regression problem, since fully learned mixture coefficients depend heavily on the input training data distribution. This can vary greatly in an online data acquisition setting. Although we use a probabilistic mixture representation, the ultimate goal of

this model is to describe a mapping from inputs to outputs. In this context, assigning the same importance to different parts of this mapping seems to make more sense.

Given w_{ij}, input $\boldsymbol{z}_i$ follows a normal distribution with parameters $\boldsymbol{\nu}_j$ and $\boldsymbol{\Sigma}_j$, while output $\boldsymbol{x}_i$ follows a linear relation from $\boldsymbol{z}_i$, with mean $\boldsymbol{\mu}_j$, design matrix $\boldsymbol{\Lambda}_j$, and diagonal covariance matrix $\boldsymbol{\Psi}_j$, corresponding to uncorrelated gaussian noise in $\boldsymbol{x}$. This model, apart the uniform distribution for w_{ij}, is similar to the one presented in Xu et al. (1995) and Sato and Ishii (2000), where each expert j models a linear relation from input $\boldsymbol{z}$ to output $\boldsymbol{x}$ in some region of the input domain, defined by input center $\boldsymbol{\nu}_j$ and covariance $\boldsymbol{\Sigma}_j$, this way softly partitioning the input space among the experts.

Unlike this previous work however, we define the following priors for the parameters of the active experts in the mixture, for which $m_j = 1$:

$$\boldsymbol{\nu}_j|\boldsymbol{\Sigma}_j \sim \mathcal{N}\left(\boldsymbol{\nu}_{0j}, \frac{1}{n_\nu}\boldsymbol{\Sigma}_j\right), \tag{2.2a}$$

$$\boldsymbol{\Sigma}_j|\bar{\boldsymbol{\Sigma}} \sim W^{-1}(n_\Sigma \bar{\boldsymbol{\Sigma}}, n_\Sigma), \tag{2.2b}$$

$$\boldsymbol{\Lambda}_j(k)|\boldsymbol{\Psi}_j(k) \sim \mathcal{N}\left(\boldsymbol{\Lambda}_0 = \mathbf{0}, \frac{\boldsymbol{\Psi}_j(k)}{n_\Lambda}\mathbf{I}\right), \tag{2.2c}$$

$$\boldsymbol{\mu}_j|\boldsymbol{\Psi}_j \sim \mathcal{N}\left(\boldsymbol{\mu}_{0j}, \frac{1}{n_\mu}\boldsymbol{\Psi}_j\right), \tag{2.2d}$$

$$\boldsymbol{\Psi}_j(k)|\bar{\boldsymbol{\Psi}} \sim \mathcal{G}^{-1}\left(\frac{n_\Psi}{2}, \frac{n_\Psi}{2}\psi_k\right), \tag{2.2e}$$

and also define the following Bernoulli prior for m_j:

$$m_j \sim Bern\left(\frac{1}{j+1}\right). \tag{2.2f}$$

The prior on the activations m_j imposes an increasing penalty on the number of linear experts the learning phase tries to allocate. We detail this in section 3.2. As for the remaining priors, W^{-1} and $\mathcal{G}^{-1}$ denote multivariate inverse-wishart and univariate inverse-gamma distributions, respectively. $\boldsymbol{\Psi}_j(k)$ is the kth element of the diagonal of $\boldsymbol{\Psi}_j$, while $\boldsymbol{\Lambda}_j(k)$ corresponds to the kth row of $\boldsymbol{\Lambda}_j$ and $\boldsymbol{\mu}_j(k)$ is the kth element of $\boldsymbol{\mu}_j$. Constants n_ν, n_Σ, n_μ, n_Λ, and n_Ψ determine the strength of the respective priors, $\boldsymbol{\nu}_{0j}$, $\bar{\boldsymbol{\Sigma}}$, $\boldsymbol{\mu}_{0j}$, $\boldsymbol{\Lambda}_0$, and $\bar{\boldsymbol{\Psi}}$, expressed as an equivalent number of "fake" data points. We choose these distributions for convenience, since they are conjugate priors for the observed data distribution. The purpose of $\bar{\boldsymbol{\Sigma}}$, the common elliptical prior

on $\mathbf{\Sigma}_j$, is threefold: it introduces some regularization, so that $\mathbf{\Sigma}_j$ always has an inverse; it ensures that the experts' input region shapes do not differ too much from each other; and it prevents nonneighboring experts from competing for the same data in the initial phase of the learning process of each expert—a serious problem that occurs in ME models, referred to, for instance, in Schaal and Atkeson (1998) and Vijayakumar, D'Souza, and Schaal (2005), thus enforcing the principle of localized learning. The normal prior on $\boldsymbol{\nu}_j$, on the other hand, controls the degree of mobility of this parameter: $n_\nu = 0$ makes it dependent solely on the training data, while $n_\nu = \infty$ leads to a fixed center, as typically occurs in radial basis networks or in LWPR; the prior on $\boldsymbol{\mu}_j$ has the same purpose. Finally, the inverse gamma prior on $\mathbf{\Psi}_j$ defines a kind of average noise all experts share, while the prior $\mathbf{\Lambda}_0 = \mathbf{0}$ for the rows of $\mathbf{\Lambda}_j$ performs a coefficient shrinkage similar to ridge regression. Its main purpose, however, is to impose a regularization mechanism in order to make the matrix inversion required for the estimation of $\mathbf{\Lambda}$; full rank. Such prior introduces some bias in the expert prediction, and consequently n_Λ should be kept to a low value in order to make this undesired effect negligible.

Diagonal matrices $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$, with diagonal elements σ_k and ψ_k, respectively, represent the prior knowledge for the common structure of $\mathbf{\Sigma}_j$ and $\mathbf{\Psi}_j$. These values strongly depend on the specific map to learn, particularly its characteristic input length scale and the output noise. We define some vague hyperpriors on these matrices, here represented by scaled inverse chi-squared distributions, to avoid relying on such problem-specific information:

$$\sigma_k \sim \text{Scale-Inv-}\chi^2(n_\sigma, \sigma_{0k}), \tag{2.3a}$$

$$\psi_k \sim \text{Scale-Inv-}\chi^2(n_\psi, \psi_{0k}), \tag{2.3b}$$

with σ_{0k} and ψ_{0k} standing, respectively, for the kth diagonal elements of $\mathbf{\Sigma}_0$ and $\mathbf{\Psi}_0$, diagonal matrices representing the initial guesses for $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$. Free parameters n_σ and n_ψ control these hyperpriors' strength: setting $n_\sigma = 0$ and $n_\psi = 0$, we get the uninformative priors $p(\sigma_k) \propto 1/\sigma_k$ and $p(\psi_k) \propto 1/\psi_k$, respectively. The (infinite) parameter vector $\mathbf{\Theta}$ that defines this mixture, to be learned from the data, is consequently given by $\mathbf{\Theta} = \{\bar{\mathbf{\Sigma}}, \bar{\mathbf{\Psi}}\} \cup \{\boldsymbol{\nu}_j, \mathbf{\Sigma}_j, \boldsymbol{\mu}_j, \mathbf{\Lambda}_j, \mathbf{\Psi}_j, m_j\}_{(1 \le j \le \infty)}$, and the graphical model corresponding to the probabilistic model is shown in Figure 1.

3 Training

Given $\boldsymbol{Z}$, a collection of input training data $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\}$, and $\boldsymbol{X}$, the corresponding output, we would like to obtain the posterior distribution $p(\mathbf{\Theta}|\boldsymbol{Z}, \boldsymbol{X})$ and use it to obtain $p(\boldsymbol{x}|\boldsymbol{z}_q, \boldsymbol{Z}, \boldsymbol{X})$ and $p(\boldsymbol{z}|\boldsymbol{x}_q, \boldsymbol{Z}, \boldsymbol{X})$, respectively,

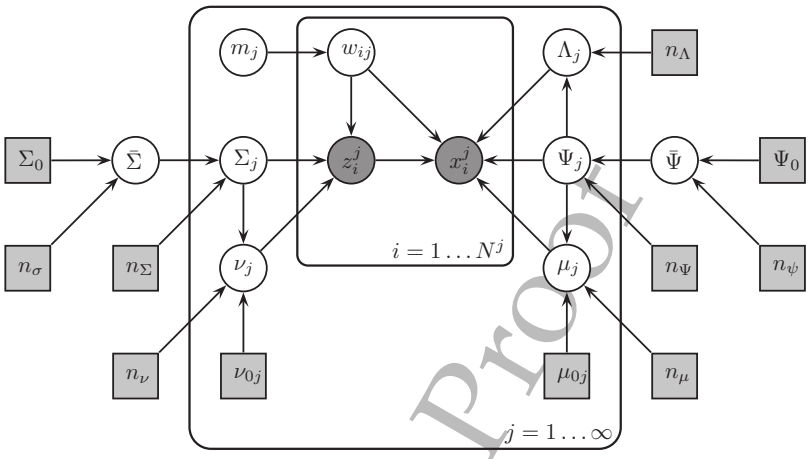


Figure 1: Graphical model representing the infinite mixture of linear experts. Lightly shaded rectangular boxes represent fixed parameters. Observed data points are grouped according to their label: $\boldsymbol{z}_i^j$ and $\boldsymbol{x}_i^j$ represent the ith data point generated by expert j.

the forward and inverse posterior predictive distributions. This is, however, intractable without also considering the latent variables $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N\}$. Although several variational or Monte Carlo Bayesian methods exist to approximate $p(\boldsymbol{\Theta}|\boldsymbol{Z}, \boldsymbol{X})$, in this letter, we use the EM algorithm (Dempster et al., 1977) to find a maximum a posteriori (MAP) estimate for the unknown $\boldsymbol{\Theta}$ due to its easy adaptation to online learning schemes.

The log likelihood of parameter vector $\boldsymbol{\Theta}$, given the complete training data $\{\boldsymbol{Z}, \boldsymbol{X}, \boldsymbol{W}\}$, is given by

$$l(\boldsymbol{\Theta}; \boldsymbol{Z}, \boldsymbol{X}, \boldsymbol{W}) = \log \left[p(\boldsymbol{\Theta}) \prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{w}_i; \boldsymbol{\Theta}) p(\boldsymbol{z}_i|\boldsymbol{w}_i; \boldsymbol{\Theta}) p(\boldsymbol{w}_i; \boldsymbol{\Theta}) \right], \tag{3.1}$$

where $p(\boldsymbol{\Theta})$ encompasses the priors defined in equations 2.2 and 2.3. The probabilities appearing in equation 3.1 are given by $p(\boldsymbol{x}_i|\boldsymbol{z}_i, \boldsymbol{w}_i; \boldsymbol{\Theta}) = \prod_{j=1}^{\infty} p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}; \boldsymbol{\Theta})^{w_{ij}}$, $p(\boldsymbol{z}_i|\boldsymbol{w}_i; \boldsymbol{\Theta}) = \prod_{j=1}^{\infty} p(\boldsymbol{z}_i|w_{ij}; \boldsymbol{\Theta})^{w_{ij}}$, and $p(\boldsymbol{w}_i; \boldsymbol{\Theta}) = \prod_{j=1}^{\infty} p(w_{ij}; \boldsymbol{\Theta})^{w_{ij}}$. The application of the EM algorithm to this log likelihood produces a sequence of estimates $\hat{\boldsymbol{\Theta}}^t$ by alternating between the expectation step (E-step), which calculates the Q-function $Q(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}^t)$, the conditional expectation of $l(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W})$ with respect to the latent variables $\boldsymbol{W}$, for the

current value of $\hat{\boldsymbol{\Theta}}^t$, and the maximization step (M-step), that finds the new value of $\hat{\boldsymbol{\Theta}}^{t+1}$ given the previous expectation.

3.1 E-Step. The log likelihood is clearly linear with respect to the latent variables w_{ij}, and hence it suffices, to obtain $Q(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}^t)$, to calculate $h_{ij}^t = E[w_{ij}|\boldsymbol{X}, \boldsymbol{Z}; \hat{\boldsymbol{\Theta}}^t]$, the estimate of the posterior probability that data point i was effectively generated by expert j, also called the responsibility that expert j has generated data point i. Since w_{ij} depends on only $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$, we have, using Bayes' theorem,

$$\begin{aligned} h_{ij}^t &\equiv E[w_{ij}|\boldsymbol{X}, \boldsymbol{Z}; \hat{\boldsymbol{\Theta}}^t] = E[w_{ij}|\boldsymbol{x}_i, \boldsymbol{z}_i; \hat{\boldsymbol{\Theta}}^t] = p(w_{ij}|\boldsymbol{x}_i, \boldsymbol{z}_i; \hat{\boldsymbol{\Theta}}^t) \\ &= \frac{p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}; \hat{\boldsymbol{\Theta}}^t) p(\boldsymbol{z}_i|w_{ij}; \hat{\boldsymbol{\Theta}}^t)\ \hat{m}_j^t}{\sum_{k=1}^{\infty} p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ik}; \hat{\boldsymbol{\Theta}}^t) p(\boldsymbol{z}_i|w_{ik}; \hat{\boldsymbol{\Theta}}^t)\ \hat{m}_k^t}. \end{aligned} \tag{3.2}$$

As a result, the Q-function becomes

$$\begin{aligned} Q(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}^t) \equiv E_W[l(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W}); \hat{\boldsymbol{\Theta}}^t] &= \log p(\boldsymbol{\Theta}) + \\ &+ \sum_{i=1}^{N} \sum_{j=1}^{\infty} h_{ij}^t [\log p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}; \boldsymbol{\Theta}) \\ &+ \log p(\boldsymbol{z}_i|w_{ij}; \boldsymbol{\Theta}) + \log p(w_{ij}; \boldsymbol{\Theta})]. \end{aligned} \tag{3.3}$$

The complete data log likelihood in equation 3.1 belongs to the exponential family, and thus the Q-function depends on only training data through $\boldsymbol{S}^t$, the expected value of the sufficient statistics vector, given the observed data and the current value of the parameter vector $\hat{\boldsymbol{\Theta}}^t$. For the mixture model, equations 2.1 and 2.2, it comprises the terms S_{hj}^t, S_{hzj}^t, S_{hxj}^t, S_{hzzj}^t, S_{hxxj}^t, and S_{hxzj}^t, for $1 \leq j \leq \infty$, defined as follows:

$$\begin{aligned} S_{hj}^t &= \textstyle\sum_{i=1}^N s_{hj}^t(i) \qquad s_{hj}^t(i) = h_{ij}^t, \\ \boldsymbol{S}_{hzj}^t &= \textstyle\sum_{i=1}^N \boldsymbol{s}_{hzj}^t(i) \quad \boldsymbol{s}_{hzj}^t(i) = h_{ij}^t \boldsymbol{z}_i, \\ \boldsymbol{S}_{hxj}^t &= \textstyle\sum_{i=1}^N \boldsymbol{s}_{hxj}^t(i) \quad \boldsymbol{s}_{hxj}^t(i) = h_{ij}^t \boldsymbol{x}_i, \\ \boldsymbol{S}_{hzzj}^t &= \textstyle\sum_{i=1}^N \boldsymbol{s}_{hzzj}^t(i) \quad \boldsymbol{s}_{hzzj}^t(i) = h_{ij}^t \boldsymbol{z}_i \boldsymbol{z}_i^T, \\ \boldsymbol{S}_{hxzj}^t &= \textstyle\sum_{i=1}^N \boldsymbol{s}_{hxzj}^t(i) \quad \boldsymbol{s}_{hxzj}^t(i) = h_{ij}^t \boldsymbol{x}_i \boldsymbol{z}_i^T, \\ \boldsymbol{S}_{hxxj}^t &= \textstyle\sum_{i=1}^N \boldsymbol{s}_{hxxj}^t(i) \quad \boldsymbol{s}_{hxxj}^t(i) = h_{ij}^t \boldsymbol{x}_i \boldsymbol{x}_i^T. \end{aligned} \tag{3.4}$$

We also define $s_i^t = \{s_{hj}^t(i), s_{hzj}^t(i), s_{hxj}^t(i), s_{hzzj}^t(i), s_{hxzj}^t(i), s_{hxxj}^t(i)\}_{(1\le j\le\infty)}$ for convenience, so that $S^t = \sum_{i=1}^N s_i^t$.

Performing the above E-step requires the availability of all observed data, which of course is not admissible during online training. Neal and Hinton (1999) present a view of the standard EM algorithm that allows for partial E-steps to be implemented, resulting in an incremental version of EM. It consists, at iteration $t+1$, of performing an update of the sufficient statistics using solely some data point i, according to $S^{t+1} = S^t + s_i^{t+1} - s_i^t$, instead of the whole data set as in equations 3.4. For a continuous stream of data, each point is visited and used only once, and thus its index i can be associated with corresponding iteration number t. The partial E-step can be written in this case as

$$S^{t+1} = S^t + s^{t+1}, \tag{3.5}$$

where $s^{t+1} \equiv s_i^{t+1}$ for data point $i = t+1$. We follow a more recent result by Cappé and Moulines (2009), where more general conditions for convergence of online EM algorithms are provided and where the following E-step is suggested:

$$\bar{S}^{t+1} = \bar{S}^t + \gamma_{t+1}(s^{t+1} - \bar{S}^t), \tag{3.6}$$

where γ_t is a step size. Setting $\gamma_t = t^{-\alpha}$, for $\alpha \in (0.5, 1]$, guarantees the algorithm convergence under some mild assumptions, while introducing a time decay in the sufficient statistics that may be beneficial when slowly time-varying data are presented to the algorithm. Such a situation may typically occur within the context of robotic applications. Equation 3.6 can be reformulated if we use an equivalent set of sufficient statistics, $S^t = \bar{S}^t/\gamma_t$, then becoming

$$S^{t+1} = \lambda_{t+1} S^t + s^{t+1}, \quad \text{where} \quad \lambda_{t+1} = \gamma_t\left(\gamma_{t+1}^{-1} - 1\right).$$

This is the decaying statistics formulation presented in Sato and Ishii (2000). Setting $\lambda_t = 1$ corresponds to having $\gamma_t = t^{-1}$, an accumulation of the sufficient statistics with no forgetting over the time, equivalent to equation 3.5.

3.2 M-Step. The M-step picks the parameter vector $\boldsymbol{\Theta}$ that maximizes the current value of the Q-function. We can use the fact that most of the priors are conjugate to the data likelihood to arrive at the following new estimates (see the appendix for details) for $j \in \mathcal{M}^t$, where $\mathcal{M}^t = \{j \in \mathbb{N} : \hat{m}_j^t = 1\}$ is the set containing the experts effectively contributing to the mixture at

iteration t:

$$\hat{\boldsymbol{v}}_j^{t+1} = \frac{S_{hzj}^t + n_v \boldsymbol{v}_{0j}}{S_{hj}^t + n_v}, \tag{3.7a}$$

$$\hat{\boldsymbol{\Sigma}}_j^{t+1} = \frac{S_{hzzj}^t - (S_{hj}^t + n_v)\hat{\boldsymbol{v}}_j^{t+1}\hat{\boldsymbol{v}}_j^{t+1^T} + n_\Sigma \hat{\bar{\boldsymbol{\Sigma}}}^{t+1} + n_v \boldsymbol{v}_{0j}\boldsymbol{v}_{0j}^T}{S_{hj}^t + n_\Sigma + d + 2}, \tag{3.7b}$$

$$\hat{\boldsymbol{\Lambda}}_j^{t+1} = \left(S_{hxzj}^t - \frac{S_{hxj}^t + n_\mu \boldsymbol{\mu}_{0j}}{S_{hj}^t + n_\mu}(S_{hzj}^t)^T\right)\left(n_\Lambda \mathbf{I} + S_{hzzj}^t - \frac{S_{hzj}^t (S_{hzj}^t)^T}{S_{hj}^t + n_\mu}\right)^{-1}, \tag{3.7c}$$

$$\hat{\boldsymbol{\mu}}_j^{t+1} = \frac{S_{hxj}^t + n_\mu \boldsymbol{\mu}_{0j}}{S_{hj}^t + n_\mu} + \hat{\boldsymbol{\Lambda}}_j^{t+1}\left(\hat{\boldsymbol{v}}_j^{t+1} - \frac{S_{hzj}^t}{S_{hj}^t + n_\mu}\right) \quad \text{and} \tag{3.7d}$$

$$\hat{\boldsymbol{\Psi}}_j^{t+1} = \frac{n_\Psi \hat{\bar{\boldsymbol{\Psi}}}^{t+1} + \mathrm{diag}\{S_{hxxj}^t - \hat{\boldsymbol{\Lambda}}_j^{t+1}(S_{hxzj}^t)^T - (\hat{\boldsymbol{\mu}}_j^{t+1} - \hat{\boldsymbol{\Lambda}}_j^{t+1}\hat{\boldsymbol{v}}_j^{t+1})(S_{hxj}^t + n_\mu \boldsymbol{\mu}_{0j})^T\}}{n_\Psi + S_{hj}^t + 2}. \tag{3.7e}$$

Diag{·} denotes a diagonal matrix equal to the diagonal of its argument. For the common input variance parameter $\bar{\boldsymbol{\Sigma}}$, however, we must obtain the partial derivatives of $Q(\boldsymbol{\Theta}, \hat{\boldsymbol{\Theta}}^t)$ with respect to each σ_k and equate them to zero, getting

$$\hat{\sigma}_k^{t+1} = \frac{\left(\frac{M^{t+1}}{2} - \frac{n_\sigma + 1}{n_\Sigma}\right) + \sqrt{\left(\frac{M^{t+1}}{2} - \frac{n_\sigma + 1}{n_\Sigma}\right)^2 + 2\frac{n_\sigma}{n_\Sigma}\sigma_{0k}\sum_{j\in\mathcal{M}^{t+1}} \hat{\boldsymbol{\Sigma}}_j^{-1}(k)^{t+1}}}{\sum_{j\in\mathcal{M}^{t+1}} \hat{\boldsymbol{\Sigma}}_j^{-1}(k)^{t+1}}, \tag{3.7f}$$

where $\hat{\boldsymbol{\Sigma}}_j^{-1}(k)^{t+1}$ denotes the kth element of the diagonal of the inverse of $\hat{\boldsymbol{\Sigma}}_j^{t+1}$ and M^t corresponds to the effective number of experts in the mixture at iteration t, that is, $M^t = \sum_{j=1}^{\infty} \hat{m}_j^t$. When we use the same procedure, a similar result holds for $\bar{\boldsymbol{\Psi}}$:

$$\hat{\psi}_k^{t+1} = \frac{\left(\frac{M^{t+1}}{2} - \frac{n_\psi + 1}{n_\Psi}\right) + \sqrt{\left(\frac{M^{t+1}}{2} - \frac{n_\psi + 1}{n_\Psi}\right)^2 + 2\frac{n_\psi}{n_\Psi}\psi_{0k}\sum_{j\in\mathcal{M}^{t+1}} \hat{\boldsymbol{\Psi}}_j^{-1}(k)^{t+1}}}{\sum_{j\in\mathcal{M}^{t+1}} \hat{\boldsymbol{\Psi}}_j^{-1}(k)^{t+1}}. \tag{3.7g}$$

Equations 3.7b and 3.7f, on one hand, and 3.7e and 3.7g, on the other, are coupled, without an explicit closed-form solution for the parameters being estimated. To deal with this issue, the maximization step is relaxed, and each of these parameters is maximized individually, conditionally on the others remaining fixed. This corresponds to the expectation conditional maximization algorithm , a particular case of the generalized variant of the EM algorithm, where the M-step is modified to an update that improves the Q-function without necessarily maximizing it (Dempster et al., 1977). Solving for the values of $\hat{m}_j^{t+1}$ that maximize the likelihood is, however intractable, as it requires evaluating all the infinite combinations of values for m_j and picking the one that maximizes the Q-function.

3.2.1 Growing the Mixture. Activations m_j define the number of experts constituting the current mixture and play a key role in defining the complexity of the global probabilistic model. Choosing the appropriate number of components for a mixture is a difficult problem, and several methods have been proposed to deal with it. Among them, Bayesian methods provide an elegant framework that automatically generates a trade-off between the fitness of the data to the model and the complexity of the same model. Moreover, the infinite mixture models based on the Dirichlet process nonparametric prior for the mixing coefficients allow for generative models where the number of components of the mixture is not defined a priori (Antoniak, 1974; Rasmussen & Ghahramani, 2002; Meeds & Osindero, 2006).

Unfortunately, training these infinite mixtures usually requires either computationally expensive Markov chain Monte Carlo sampling methods or variational approaches that typically rely on some sort of truncation that imposes a bound on the admissible number of components for the mixture. Furthermore, the online operation we are trying to achieve in this work imposes some additional difficulties. In an offline setting, Bayesian methods in principle can efficiently grow or annihilate mixture components, but when it is operating online, the full set of training points is no longer available, and decisions concerning the allocation or removal of components of the mixture must be made, resorting only to the most recently available training points and the current mixture state, a far more demanding learning challenge. Sato (2001) derives an online variational Bayesian algorithm for learning mixture models, but it requires the maintenance of a parallel hypothesis about the number of components that can easily become too computationally expensive. Some recent work also views the variational Bayesian learning model under an online perspective but is based on nondeterministic approaches based on Gibbs sampling (Wang & Blei, 2012) or require processing the training points in smaller batches of data (Gomes, Welling, & Perona, 2008). For all these methods, an adequate computational speed that allows the processing of hundreds of samples per second, as required for online learning schemes for robotic applications, is yet to be shown.

In contrast to Bayesian methods, it is more difficult to assess an optimal value for the number of components of a mixture when using an EM algorithm for training. A well-known drawback of EM-based techniques is the fact that the maximized observed data likelihood will never decrease when new components are added to the mixture. This allows the derivation of a broad class of criteria to decide when to add new components, ranging from only allowing the existence of a single component, equivalent to performing a global linear regression on the data, to the activation of a new one for each data point processed, which would correspond to a memory-based learning approach, where predictions would be made resorting to all available training data. As a consequence, most EM deterministic methods impose some kind of penalty over the number of mixture components during optimization, such as Akaike's information criterion (AIC), the Bayesian inference criterion (BIC), or the minimum message length criterion (MML), to name just a few. A comprehensive comparison and review of these kind of penalty methods can be found in McLachlan and Peel (2000).

In the probabilistic model given in equations 2.1 and 2.2, the prior distribution on m_j plays the role of such penalty, making a high number of experts increasingly less probable. Changing, at iteration t, the value of a particular parameter $\hat{m}_j$ from 0 to 1 will affect only the Q-function in equation 3.3 through the term $\log p(\boldsymbol{\Theta})$, as the values h_{ij}^t will still be equal to 0 for all i and this particular value of j, following equation 3.2. This results in a decrease of the Q-function value every time a new expert is activated in the M-step. In the absence of the prior, equation 2.2f, the Q-function value would not change.

Although momentarily decreasing the Q-function, activating a new expert can nevertheless increase the observed data likelihood in the subsequent iteration. Of course, under the online paradigm followed in this letter, it is not possible to calculate the likelihood of the entire observed data, as each training point is discarded after the corresponding update of the mixture sufficient statistics. However, at the end of M-step, we can evaluate the log likelihood of the next training point $(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1})$ under the new parameter vector $\hat{\boldsymbol{\Theta}}^{t+1}$. This log likelihood is given, considering also the priors on $\boldsymbol{\Theta}$, by $L^{t+1}(\hat{\boldsymbol{\Theta}}^{t+1}) = \log[p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}^{t+1})p(\hat{\boldsymbol{\Theta}}^{t+1})]$, where $p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}^{t+1})$ is obtained from the complete data likelihood by marginalizing out the latent variables w_{ij}:

$$p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}^{t+1}) = \sum_{j=1}^{\infty} p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|w_{(t+1)j}, \hat{\boldsymbol{\Theta}}^{t+1}) p(w_{(t+1)j}|\hat{\boldsymbol{\Theta}}^{t+1})$$

$$= \frac{1}{M^t} \sum_{j=1}^{M^t} p(\boldsymbol{x}_{t+1}|\boldsymbol{z}_{t+1}, \hat{\boldsymbol{\Theta}}_j^{t+1}) p(\boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}_j^{t+1}).$$

In this expression, $\hat{\boldsymbol{\Theta}}_j^{t+1}$ refers to the current estimates of the parameters associated with expert j. If, alternatively, expert $j^+ = M^t + 1$ is activated at the end of the M-step, by making $\hat{m}_{j+} = 1$ and initializing $\boldsymbol{\mu}_{0j+}$ and $\boldsymbol{v}_{0j+}$ to $\boldsymbol{x}_{t+1}$ and z_{t+1}, respectively, we obtain the alternative log likelihood $L^{t+1}(\hat{\boldsymbol{\Theta}}_+^{t+1}) = \log[p(\boldsymbol{x}_{t+1}, z_{t+1}|\hat{\boldsymbol{\Theta}}_+^{t+1})p(\hat{\boldsymbol{\Theta}}_+^{t+1})]$, where $\hat{\boldsymbol{\Theta}}_+^{t+1}$ is a changed version of parameter vector $\hat{\boldsymbol{\Theta}}^{t+1}$, with expert j^+ activated and where the next training point likelihood is now given by

$$\begin{aligned} & p(\boldsymbol{x}_{t+1}, z_{t+1}|\hat{\boldsymbol{\Theta}}_+^{t+1}) \\ &= \sum_{j=1}^{\infty} p(\boldsymbol{x}_{t+1}, z_{t+1}|w_{(t+1)j}, \hat{\boldsymbol{\Theta}}_+^{t+1})p(w_{(t+1)j}|\hat{\boldsymbol{\Theta}}_+^{t+1}) \\ &= \frac{M^t}{M^t+1} p(\boldsymbol{x}_{t+1}, z_{t+1}|\hat{\boldsymbol{\Theta}}^{t+1}) + \frac{1}{M^t+1} p(\boldsymbol{x}_{t+1}, z_{t+1}|\hat{\boldsymbol{\Theta}}_{j+}^{t+1}). \end{aligned}$$

Activating expert j^+ will increase the log likelihood of the next training point if $L^{t+1}(\hat{\boldsymbol{\Theta}}_+^{t+1}) > L^{t+1}(\hat{\boldsymbol{\Theta}}^{t+1})$, and this can be used as a criterion for deciding when to activate a new expert. This approach, however, will often lead to too many local models being allocated. Instead we take a statistical approach of activating a new expert only when strong evidence supports the alternative parameter vector $\hat{\boldsymbol{\Theta}}_+^{t+1}$ against the null hypothesis $\hat{\boldsymbol{\Theta}}^{t+1}$. This later parameter vector can be seen as a special case of $\hat{\boldsymbol{\Theta}}_+^{t+1}$, with one less mixture component. This suggests using a likelihood ratio test to compare them, where the test statistic $T = 2L^{t+1}(\hat{\boldsymbol{\Theta}}_+^{t+1}) - 2L^{t+1}(\hat{\boldsymbol{\Theta}}^{t+1})$ approximately follows a chi-squared distribution with degrees of freedom equal to the difference of free number of parameters between $\hat{\boldsymbol{\Theta}}_+^{t+1}$ and $\hat{\boldsymbol{\Theta}}^{t+1}$ (Kendall, Stuart, Ord, Arnold, & O'Hagan, 1998). At the time of activation of a new expert j^+, only $\boldsymbol{\mu}_{0j+}$ and $\boldsymbol{v}_{0j+}$ are effectively defined, so the change in the number of free parameters is equal to $d + D$. Let $\mathcal{X}^2(p_0, d + D)$ be the critical value of a chi-squared distribution with $d + D$ degrees of freedom, corresponding to the number of free parameters introduced in the mixture: a new expert should be activated, according to the likelihood ratio, if, for a significance value p_0,

$$2L^{t+1}(\hat{\boldsymbol{\Theta}}_+^{t+1}) - 2L^{t+1}(\hat{\boldsymbol{\Theta}}^{t+1}) > \mathcal{X}^2(p_0, d + D).$$

Since changing m_{j+} affects only $p(\hat{\boldsymbol{\Theta}}_+^{t+1})$ through the term $p(m_{j+})$, given by equation 2.2f, we have $p(\hat{\boldsymbol{\Theta}}_+^{t+1})/p(\hat{\boldsymbol{\Theta}}^{t+1}) = 1/(M^t + 1)$, and the above

expression results in activating a new expert j^+ when

$$\sum_{j=1}^{M^t} p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}_j^{t+1}) \leq \frac{1}{\frac{M^t+1}{M^t} - \frac{e^{-0.5\mathcal{X}^2(p_0,d+D)}}{M^t+1}} \cdot \frac{e^{-0.5\mathcal{X}^2(p_0,d+D)}}{M^t+1} \cdot p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}_{j^+}^{t+1}), \tag{3.8}$$

where expert j^+ parameters are equal to they prior values (expert j^+ has not yet accumulated any sufficient statistics). Since $\boldsymbol{v}_{0j^+} = \boldsymbol{z}_{t+1}$ and $\boldsymbol{\mu}_{0j^+} = \boldsymbol{x}_{t+1}$, this results in

$$p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}_{j^+}^{t+1}) = \frac{1}{\sqrt{(2\pi)^D|\hat{\tilde{\boldsymbol{\Psi}}}^{t+1}|}} \cdot \frac{1}{\sqrt{(2\pi)^d|\hat{\tilde{\boldsymbol{\Sigma}}}^{t+1}|}}.$$

The right side of equation 3.8 introduces an increasingly penalty on the activation of new experts as the value of M increases. Adjustable parameter p_0 can regulate the propensity to activate new experts: the lower its value, the higher the critical value of the chi-squared distribution will be, making the experts' activation criterion harder to be met, resulting in fewer components in the mixture. In a general way, equation 3.8 tells us that a new expert should be activated when the next acquired training point is poorly explained by the current probabilistic model. This is a sensible approach to mixture grow in online algorithms. LWPR, for instance, creates a new linear model each time an input training point $\boldsymbol{z}_i$ fails to activate the nearest receptive field by more than a given threshold. Of course, when learning multivalued functions, an activation scheme must take into account both the input and output part of the training point, as occurs in equation 3.8.

3.2.2 Handling Outliers. Equation 3.8 can be viewed under a different and equivalent perspective, if a special class w_0 is considered, corresponding to outliers that are not generated by any of the currently activated experts, with an improper constant distribution

$$p(\boldsymbol{x}, \boldsymbol{z}|w_0, \boldsymbol{\Theta}) = \frac{1}{\sqrt{(2\pi)^D|\hat{\tilde{\boldsymbol{\Psi}}}^{t+1}|}} \cdot \frac{1}{\sqrt{(2\pi)^d|\hat{\tilde{\boldsymbol{\Sigma}}}^{t+1}|}} e^{-0.5\mathcal{X}^2(p_0,d+D)} \tag{3.9}$$

and a prior distribution $p(w_0|\boldsymbol{\Theta}) = 1/(M+1)^2$. Then the probability that a training point is an outlier generated by w_0, given the current mixture parameters, follows from Bayes' rule:

$$p(w_0|\boldsymbol{x}, \boldsymbol{z}, \hat{\boldsymbol{\Theta}}) = \frac{p(\boldsymbol{x}, \boldsymbol{z}|w_0, \hat{\boldsymbol{\Theta}})p(w_0|\hat{\boldsymbol{\Theta}})}{\sum_{j=1}^{M^t} p(\boldsymbol{x}, \boldsymbol{z}|\hat{\boldsymbol{\Theta}}_j)p(w_{ij}|\hat{\boldsymbol{\Theta}}) + p(\boldsymbol{x}, \boldsymbol{z}|w_0, \hat{\boldsymbol{\Theta}})p(w_0|\hat{\boldsymbol{\Theta}})}.$$

This posterior probability is dominant over the posterior probabilities for the ME experts if $p(w_0|\boldsymbol{x}_i, \boldsymbol{z}_i, \hat{\boldsymbol{\Theta}}) > 0.5$. This happens only at the end of iteration t and for a new point $(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1})$ if

$$\sum_{j=1}^{M^t} p(\boldsymbol{x}_{t+1}, \boldsymbol{z}_{t+1}|\hat{\boldsymbol{\Theta}}_j^{t+1}) \leq \frac{1}{M^t + 2} \cdot \frac{1}{\sqrt{(2\pi)^D |\hat{\bar{\boldsymbol{\Psi}}}^{t+1}|}} \cdot \frac{1}{\sqrt{(2\pi)^d |\hat{\bar{\boldsymbol{\Sigma}}}^{t+1}|}} e^{-0.5\mathcal{X}^2(p_0, d+D)}.$$

This result is very similar to equation 3.8, since the first factor on right side of equation 3.8 quickly approaches 1 as the number of active experts M increases.

The probabilistic outlier model $p(\boldsymbol{x}, \boldsymbol{z}|w_0, \boldsymbol{\Theta})$ in equation 3.9 depends on the current estimates for output noise $\bar{\boldsymbol{\Psi}}$ and input length-scale $\bar{\bar{\boldsymbol{\Sigma}}}$. This is a better approach than considering a fixed threshold, as tuning this latter parameter would require some problem domain-specific knowledge. If we consider a newly activated, not-yet-trained expert (for which $\hat{\boldsymbol{\Psi}}_j = \hat{\bar{\boldsymbol{\Psi}}}$, $\hat{\boldsymbol{\Lambda}}_j = \mathbf{0}$ and $\hat{\boldsymbol{\Sigma}}_j = \hat{\bar{\boldsymbol{\Sigma}}}$), $p(\boldsymbol{x}, \boldsymbol{z}|w_0, \hat{\boldsymbol{\Theta}})$ can be seen as the evaluation of the probability density of the expert at a point $(\boldsymbol{x}, \boldsymbol{z})$ that lies over the equidensity contour that encircles the region, centered at $(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{v}}_j)$, corresponding to a $1 - p_0$ probability.

The results above show us that the mechanisms for activating a new expert or recognizing an outlier are essentially the same and correspond to identifying training points poorly explained by the current mixture. How do we know, then, if a training point satisfying equation 3.8 is an outlier or, alternatively, an indication that a new expert is needed in the corresponding region of the input-output space? This is challenging problem. For single-valued regression, outliers can be detected as training points that have enough support from the current model in the input space, while presenting a large deviation from it in the output space. The same, however, does not happen in multivalued regression, where such a situation may simply correspond to a yet unseen branch of the multivalued function being learned. Furthermore, the online assumption does not allow us to look at the whole training set, where identifying outliers in principle would be made easier by searching for isolated points.

This question can be answered under the assumption that the training data are temporally correlated, as often is the case in online learning. In this situation, a training point will have a high probability of being poorly explained by the current mixture if the same occurred with the previous point. In contrast, the same does not happen with statistically independent outliers. Observing two consecutive training points satisfying equation 3.8

is then more likely to be caused by a lack of fit of the points to the current model than the occurrence of two consecutive outliers. In such an event, we decide to activate a new expert.

3.3 Computational Complexity. Learning with IMLE is very fast. For a new observation $(\boldsymbol{z}_t, \boldsymbol{x}_t)$, a complete update of the mixture parameters consists of (1) deciding whether a new expert should be activated in equation 3.8; (2) assigning responsibilities h_{ij} to active experts using equation 3.2 (E-step); (3) updating the sufficient statistics (E-step, equation 3.6); and (4) obtaining the new value for $\hat{\boldsymbol{\Theta}}$ (M-step, equations 3.7). All these calculations have a computational complexity of $\mathcal{O}(Md(d+D))$, since the matrix inversions required in this process can be efficiently performed using the Sherman-Morrison formula to perform a rank 1 update for these quantities. The computational complexity of a complete update of IMLE parameters is consequently linear in D and M, the number of experts, and quadratic in d, the number of input dimensions, making it directly comparable to the state-of-the-art LWPR in terms of computational complexity per training point. Like LWPR, this complexity can be made linear in d if the input distance metrics $\boldsymbol{\Sigma}_j$ are constrained to be diagonal.

4 Prediction

Under a full Bayesian paradigm, a forward prediction for an input query $\boldsymbol{z}_q$ is represented by a distribution $p(\boldsymbol{x}|\boldsymbol{z}_q, \boldsymbol{X}, \boldsymbol{Z})$, where the dependence on learned mixture parameters and latent variables is marginalized out. The same occurs in inverse prediction, where now $p(\boldsymbol{z}|\boldsymbol{x}_q, \boldsymbol{X}, \boldsymbol{Z})$ is considered. However, these posterior distributions cannot be analytically calculated for most probabilistic models. Instead, EM-based learning algorithms will normally provide predictions based on $\hat{\boldsymbol{\Theta}}$, the point estimate for the parameter vector being learned. For IMLE forward prediction, this results in

$$p(\boldsymbol{x}|\boldsymbol{z}_q, \hat{\boldsymbol{\Theta}}) = \sum_{j \in \mathcal{M}} w_j^x(\boldsymbol{z}_q) p(\boldsymbol{x}|\boldsymbol{z}_q, w_j, \hat{\boldsymbol{\Theta}}), \tag{4.1}$$

where

$$w_j^x(\boldsymbol{z}_q) = p(w_j|\boldsymbol{z}_q, \hat{\boldsymbol{\Theta}}) = \frac{p(\boldsymbol{z}_q|w_j, \hat{\boldsymbol{\Theta}})}{\sum_{k \in \mathcal{M}} p(\boldsymbol{z}_q|w_k, \hat{\boldsymbol{\Theta}})},$$

and where w_j and w_k are a shorthand for $w_{qj} = 1$ and $w_{kj} = 1$; $p(\boldsymbol{x}|\boldsymbol{z}_q, w_j, \hat{\boldsymbol{\Theta}})$ and $p(\boldsymbol{z}_q|w_j, \hat{\boldsymbol{\Theta}})$ follow from equation 2.1, with $\boldsymbol{\Theta}$ replaced by its estimate.

Inverse prediction, on the other hand, leads to

$$p(\boldsymbol{z}|\boldsymbol{x}_q, \hat{\boldsymbol{\Theta}}) = \sum_{j \in \mathcal{M}} w_j^z(\boldsymbol{x}_q) p(\boldsymbol{z}|\boldsymbol{x}_q, w_j, \hat{\boldsymbol{\Theta}}), \tag{4.2}$$

where now we have

$$w_j^z(\boldsymbol{x}_q) = p(w_j|\boldsymbol{x}_q, \hat{\boldsymbol{\Theta}}) = \frac{p(\boldsymbol{x}_q|w_j, \hat{\boldsymbol{\Theta}})}{\sum_{k \in \mathcal{M}} p(\boldsymbol{x}_q|w_k, \hat{\boldsymbol{\Theta}})}.$$

Distributions for $p(\boldsymbol{z}|\boldsymbol{x}_q, w_j, \hat{\boldsymbol{\Theta}})$ and $p(\boldsymbol{x}_q|w_j, \hat{\boldsymbol{\Theta}})$ can be obtained from the joint input-output distribution $p(\boldsymbol{y}|w_j, \hat{\boldsymbol{\Theta}})$, where $\boldsymbol{y}^T = [\boldsymbol{z}^T \ \boldsymbol{x}^T]$. The joint distribution given w_j is normal, with mean $\boldsymbol{y}_j^T = [\boldsymbol{v}_j^T, \boldsymbol{\mu}_j^T]$ and covariance matrix

$$\boldsymbol{R}_j^y = \begin{bmatrix} \boldsymbol{\Sigma}_j & \boldsymbol{\Sigma}_j \boldsymbol{\Lambda}_j^T \\ \boldsymbol{\Lambda}_j \boldsymbol{\Sigma}_j & \boldsymbol{\Psi}_j + \boldsymbol{\Lambda}_j \boldsymbol{\Sigma}_j \boldsymbol{\Lambda}_j^T \end{bmatrix}.$$

From this result, it immediately follows that

$$\boldsymbol{z}|\boldsymbol{x}_q, w_j; \hat{\boldsymbol{\Theta}} \sim \mathcal{N}(\hat{\boldsymbol{z}}_j(\boldsymbol{x}_q), \boldsymbol{R}_j^z),$$
$$\boldsymbol{x}_q|w_j; \hat{\boldsymbol{\Theta}} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Psi}}_j + \hat{\boldsymbol{\Lambda}}_j \hat{\boldsymbol{\Sigma}}_j \hat{\boldsymbol{\Lambda}}_j^T),$$

where $\hat{\boldsymbol{z}}_j(\boldsymbol{x}_q)$ and $\boldsymbol{R}_j^z$ are given, respectively, by

$$\hat{\boldsymbol{z}}_j(\boldsymbol{x}_q) = \hat{\boldsymbol{v}}_j + \boldsymbol{R}_j^z \hat{\boldsymbol{\Lambda}}_j^T \hat{\boldsymbol{\Psi}}_j^{-1} (\boldsymbol{x}_q - \hat{\boldsymbol{\mu}}_j) \quad \text{and}$$
$$\boldsymbol{R}_j^z = (\hat{\boldsymbol{\Sigma}}_j^{-1} + \hat{\boldsymbol{\Lambda}}_j^T \hat{\boldsymbol{\Psi}}_j^{-1} \hat{\boldsymbol{\Lambda}}_j)^{-1}.$$

The same reasoning for obtaining an inverse probability distribution from the IMLE model can also be used to produce more general predictions. Assuming the joint vector $\boldsymbol{y}$ to comprise a query part $\boldsymbol{y}_q$ and an answer part $\boldsymbol{y}_a$, such that $\boldsymbol{y} = \boldsymbol{y}_q \cup \boldsymbol{y}_a$, we can easily derive $p(\boldsymbol{y}_a|\boldsymbol{y}_q, w_j, \hat{\boldsymbol{\Theta}})$, for each expert j, by conditioning the joint distribution with respect to $\boldsymbol{y}_q$.

Although not being able to provide a complete Bayesian forward estimate, IMLE probabilistic model conjugacy relations make it possible to marginalize out some of the learned parameters. This fact allows us to improve equation 4.1, by incorporating the uncertainty in these parameters in the final prediction. Given the training data, experts' activations

$\hat{m} = \{\hat{m}_j\}_{(1 \le j \le \infty)}$, and current predictions for $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$, we now have

$$p(\boldsymbol{x}|\boldsymbol{z}_q, \boldsymbol{X}, \boldsymbol{Z}, \hat{\bar{\mathbf{\Sigma}}}, \hat{\bar{\mathbf{\Psi}}}, \hat{m}) = \sum_{j \in \mathcal{M}} w_j^x(\boldsymbol{z}_q)\, p(\boldsymbol{x}|\boldsymbol{z}_q, w_j, S^t, \hat{\bar{\mathbf{\Psi}}}) \tag{4.3}$$

and

$$w_j^x(\boldsymbol{z}_q) = \frac{p(\boldsymbol{z}_q|w_j, S^t, \hat{\bar{\mathbf{\Sigma}}})\, p(w_j|\hat{m})}{\sum_{k=1}^{\infty} p(\boldsymbol{z}_q|w_k, S^t, \hat{\bar{\mathbf{\Sigma}}})\, p(w_k|\hat{m})} = \frac{p(\boldsymbol{z}_q|w_j, S^t, \hat{\bar{\mathbf{\Sigma}}})}{\sum_{k \in \mathcal{M}} p(\boldsymbol{z}_q|w_k, S^t, \hat{\bar{\mathbf{\Sigma}}})}, \tag{4.4}$$

where we summarize the dependence on training data $(\boldsymbol{X}, \boldsymbol{Z})$ using the sufficient statistics S^t; distributions for $p(\boldsymbol{x}|\boldsymbol{z}_q, w_j, S^t, \hat{\bar{\mathbf{\Psi}}})$ and $p(\boldsymbol{z}_q|w_j, S^t, \hat{\bar{\mathbf{\Sigma}}})$ are derived in the appendix and are given by equations A.5 and A.1, respectively. To keep the notation simple we will use $\hat{\mathbf{\Theta}}^*$ to refer to either $\hat{\mathbf{\Theta}}$ or $(S^t, \hat{\bar{\mathbf{\Sigma}}}, \hat{\bar{\mathbf{\Psi}}}, \hat{m})$ in the rest of this section, depending on whether forward or inverse prediction is considered.

The conditional densities, equations 4.3 and 4.2, can be understood as a weighted mixture of M normal densities, each corresponding to a point estimate provided by a different expert, together with an uncertainty value, and where the mixture weights are given by the posterior probabilities that the query point was generated by each expert. A single-valued forward prediction, together with an associated uncertainty, can be obtained from equation 4.3 by taking its mean and variance,

$$\hat{\boldsymbol{x}} = E[\boldsymbol{x}|\boldsymbol{z}_q, \hat{\mathbf{\Theta}}^*] = \sum_j w_j^x \hat{\boldsymbol{x}}_j \quad \text{and} \tag{4.5}$$

$$\hat{\boldsymbol{R}} = V[\boldsymbol{x}|\boldsymbol{z}_q, \hat{\mathbf{\Theta}}^*] = \sum_j w_j^x \boldsymbol{R}_j^x + \sum_j w_j^x (\hat{\boldsymbol{x}}_j - \hat{\boldsymbol{x}})(\hat{\boldsymbol{x}}_j - \hat{\boldsymbol{x}})^T, \tag{4.6}$$

where $\hat{\boldsymbol{x}}_j$ and $\boldsymbol{R}_j^x$ are, respectively, the mean and variance of $p(\boldsymbol{x}|\boldsymbol{z}_q, w_j, \hat{\mathbf{\Theta}}^*)$, and where from now on, we drop the dependence on $\boldsymbol{z}_q$ for notational convenience. This is the approach followed in LWPR, and it works reasonably well under the single-valued hypothesis, although it tends to overestimate the true variance of the data due to the cross-variance between expert estimates, given by the last term in equation 4.6. However, when multivalued functions are considered, this approach will be able to generate only a single estimate, together with a large value of the associated uncertainty. As reported, for instance, by Ghahramani and Jordan (1994), merging together the distinct solutions provided by each expert might result in a poor overall estimate for a nonconvex solution space, where the weighted mean of different experts' predictions might itself be far from the true value to estimate.

This is clearly unacceptable. Searching for all the modes of the underlying distribution (Carreira-Perpiñán, 2000), on the other hand, can introduce many low-weight spurious modes, corresponding to the contributions of distant experts. In this case, some kind of filtering must be done to remove them. Even after removing low weighted components, the topography of the mixture can be complex in prediction spaces with more than one dimension, as analyzed by Ray and Lindsay (2005), where, counterintuitively, there may exist more modes than mixture components. Consequently, in order to obtain multivalued forward predictions, we must deal with two major issues: (1) how to identify the correct number of solutions for a given query $\boldsymbol{z}_q$ and (2) how to group and merge the experts' predictions accordingly. These are the topics of the following sections. Although exemplified by forward prediction, the concepts and techniques are exactly the same for inverse, forward, or more general prediction.

4.1 Grouping Experts Predictions. If, for a query $\boldsymbol{z}_q$, the true number of solutions N_{sol} is assumed to be known, estimating these solutions reduces to the problem of clustering M observations $\hat{\boldsymbol{x}}_j$ into N_{sol} classes, where each observation comes with an associated variance $\boldsymbol{R}_j^x$ and weight w_j^x. To provide a Bayesian probabilistic model that relates experts' predictions $\hat{\boldsymbol{x}}_j$ to the unknown multivalued solutions $\bar{\boldsymbol{x}}_k$, for $1 \leq k \leq N_{sol}$, and that additionally takes $\boldsymbol{R}_j^x$ and w_j^x into consideration, we propose the generative model

$$\boldsymbol{x}_j | s_{jk}, \boldsymbol{\Theta} \sim \mathcal{N}(\bar{\boldsymbol{x}}_k, \boldsymbol{\Psi}_j / w_j^x), \tag{4.7}$$

where s_{jk} is a latent indicator variable that signals if $\boldsymbol{x}_j$, expert j conditional mean given $\boldsymbol{z}_q$, was produced by solution k. Expert j true conditional mean $\boldsymbol{x}_j$ is also unobserved: it relates to the point estimate $\hat{\boldsymbol{x}}_j$ given by equation A.3 according to equation A.7, which takes uncertainty on current expert parameters into account. The rationale for the variance in the previous model follows from the traditional probabilistic view of weighted least squares and best linear unbiased estimators (Gelman, Carlin, Stern, & Rubin, 2004; Vijayakumar et al., 2005), where we incorporate each expert weight w_j^x in the respective predictor variance. Using equation A.7 and compounding the distribution, equation 4.7 with the posterior distribution for $\boldsymbol{\Psi}_j$ given $\hat{\boldsymbol{\Theta}}^*$ results in an approximate normal distribution with variance $\hat{\boldsymbol{\Psi}}_j / w_j^x$. Putting these facts together, we get

$$\hat{\boldsymbol{x}}_j | s_{jk}, \hat{\boldsymbol{\Theta}}^* \sim \mathcal{N}(\bar{\boldsymbol{x}}_k, \boldsymbol{R}_j), \quad \text{with} \quad \boldsymbol{R}_j \equiv (1/w_j^x + \gamma_j)\hat{\boldsymbol{\Psi}}_j. \tag{4.8}$$

Estimates $\hat{\bar{\boldsymbol{x}}}_k$ for solutions $\bar{\boldsymbol{x}}_k$ can be provided by a simple EM procedure. The likelihood for the complete data is equal to $Q = \prod_j \prod_k p(\hat{\boldsymbol{x}}_j | s_{jk}; \hat{\boldsymbol{\Theta}}^*)^{s_{jk}}$,

and from this expression, the following EM iterations are easily obtained (we dropped the dependence on $\hat{\boldsymbol{\Theta}}^*$ for simplicity), for $1 \leq k \leq N_{sol}$:

$$h_{jk}^t = E[s_{jk}|\hat{\boldsymbol{x}}_j, \hat{\tilde{\boldsymbol{x}}}_k^t] = p(s_{jk}|\hat{\boldsymbol{x}}_j, \hat{\tilde{\boldsymbol{x}}}_k^t) = \frac{p(\hat{\boldsymbol{x}}_j|s_{jk}, \hat{\tilde{\boldsymbol{x}}}_k^t)}{\sum_l p(\hat{\boldsymbol{x}}_j|s_{jl}, \hat{\tilde{\boldsymbol{x}}}_l^t)}, \quad \text{(E-step)} \quad (4.9a)$$

$$\hat{\tilde{\boldsymbol{x}}}_k^{t+1} = \left(\sum_j h_{jk}^t \boldsymbol{R}_j^{-1}\right)^{-1} \left(\sum_j h_{jk}^t \boldsymbol{R}_j^{-1} \hat{\boldsymbol{x}}_j\right). \quad \text{(M-step)} \quad (4.9b)$$

We found that it takes only a few iterations for the algorithm to converge. After that, predictions are hard-assigned to solutions according to the final value of h_{jk}, resulting in the following estimate for each solution k, where the sums are over experts assigned to each particular solution:

$$\hat{\tilde{\boldsymbol{x}}}_k = \hat{\boldsymbol{R}}_k \sum_j \boldsymbol{R}_j^{-1} \hat{\boldsymbol{x}}_j, \quad \text{with} \quad \hat{\boldsymbol{R}}_k \equiv \left(\sum_j \boldsymbol{R}_j^{-1}\right)^{-1}.$$

The uncertainty for prediction $\hat{\tilde{\boldsymbol{x}}}_k$ is given by $\hat{\boldsymbol{R}}_k$. When $w_j^x = 1$ for a particular expert j, meaning that only that expert contributes to the prediction, this uncertainty becomes equal to $\boldsymbol{R}_j = (1 + \gamma_j)\hat{\boldsymbol{\Psi}}_j$, which agrees with the predictive uncertainty of the ordinary least squares solution for multiple linear regression.

4.2 A Test for Multivalued Prediction Goodness of Fit. For each $\hat{\boldsymbol{x}}_j$, the quantity $(\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_k)^T \boldsymbol{R}_j^{-1} (\hat{\boldsymbol{x}}_j - \bar{\boldsymbol{x}}_k)$ follows a chi-squared distribution with D degrees of freedom, given that $\hat{\boldsymbol{x}}_j$ was indeed generated by solution k using model 4.8. Under the null hypothesis that (1) generative model 4.8 corresponds to the true distribution for observed $\hat{\boldsymbol{x}}_j$, (2) N_{sol} is the true number of multivalued solutions for query $\boldsymbol{z}_q$, and (3) the previous EM algorithm correctly grouped the experts' predictions into N_{sol} different solutions, the statistic T_k follows a chi-squared distribution for every solution k,

$$T_k = \sum_j (\hat{\boldsymbol{x}}_j - \hat{\tilde{\boldsymbol{x}}}_k)^T \boldsymbol{R}_j^{-1} (\hat{\boldsymbol{x}}_j - \hat{\tilde{\boldsymbol{x}}}_k) \quad \sim \chi^2_{(M_k - 1)D}, \quad (4.10)$$

where again the sums are over experts assigned to solution k and M_k is the number of experts belonging to that solution. A low value for this statistic indicates a good fit of observations $\hat{\boldsymbol{x}}_j$ to the estimated solutions $\hat{\tilde{\boldsymbol{x}}}_k$. On the other hand, if the p-value for any solution k is lower than a given significance

level α_{multi}, the current set of solutions $\hat{\hat{\mathbf{x}}}_k$ is considered to be badly explained by the data.

A practical detail of the above test is that $\hat{\mathbf{x}}_j$ will not have arbitrary large values for corresponding small values of w_j^x, as predicted by model 4.8. As a consequence T_k will assume much lower values than the ones expected under the null hypothesis distribution, since in a typical scenario, most of the experts will have a negligible contribution to the prediction. In this situation, we found that replacing $M_k - 1$ in equation 4.10 with $1/\sum(w_j^x)^2 - 1$, the effective degrees of freedom of the mixture, provides a better fit of the statistic to its corresponding distribution, under the null hypothesis.

4.3 Obtaining a Valid Set of Multivalued Solutions. Multivalued prediction starts with the single-valued estimation, $N_{sol} = 1$. If the test described in the previous section rejects this single-valued solution, N_{sol} is increased to 2 and the EM iterations in equation 4.9 are performed. If the test for goodness of fit described in the previous section rejects at least one of the solutions, N_{sol} is incremented, and this procedure is repeated until a value of N_{sol} is found for which the test fails to reject the null hypothesis for any of the solutions thus obtained. $\hat{\hat{\mathbf{x}}}_k$, for $1 \leq k \leq N_{sol}$, is then the set of multivalued forward solutions predicted by IMLE, while $\hat{\mathbf{R}}_k$ are the respective uncertainties. To speed up the prediction process, each EM procedure initializes $\hat{\hat{\mathbf{x}}}_k$ with the values found in the previous run of the algorithm, while the extra solution starts near the solution k that produced the smallest p-value in the previous goodness-of-fit test, this way dividing in two the solution responsible for the null hypothesis rejection.

During this process, the significance level α_{multi} controls the number of solutions found. The lower its value, the harder it is to reject the null hypothesis, and fewer solutions are likely to be found. Increasing α_{multi} helps to separate different solutions, but as an unwanted consequence, predictions $\hat{\mathbf{x}}_j$ for neighbor experts may stop being merged together due to the function curvature around $\mathbf{z}_q$.

5 Experimental Results

In this section we evaluate IMLE in several different experimental settings, especially focusing on large training sets arising from continuous stream of data that particularly suit online learning. We compare our results with different online learning algorithms, namely LWPR (Vijayakumar et al., 2005), probably one of the most widely used state-of-the-art online learning methods for robotic applications, SOGP; a sparse online approximation for gaussian process regression (Csató & Opper, 2002), and ROGER, an online infinite mixture of SOGP experts (Grollman & Jenkins, 2010). All of these algorithms have their C++ implementation code available, and their most

recent version to this date is used in our comparisons.[1] A C++ implementation of the IMLE algorithm is also freely available for download.[2] We also compare IMLE to standard GPR. This is not an online algorithm in its standard formulation, but it can give some insights into the performance loss we expect when going to an online operation setting. In the following experiments, we specify a gaussian likelihood to be used with an isotropic squared exponential covariance function, using exact inference for training and prediction. Optimal values for input length scale of the kernels and output noise, the free hyperparameters of the model, are obtained using standard optimization techniques over the training set, using the GPML Matlab code.[3]

Parameters of interest for tuning the LWPR algorithm comprehend $\boldsymbol{D}_{init}$, α_{init}, w_{gen}, and penalty γ (for details on these parameters' meaning, consult the related documentation). Additionally, in all experiments we set *diagOnly* to *false* and *useMeta* and *updateD* to *true*. We use SOGP with a gaussian kernel. The remaining tuning parameters for this algorithm are σ_k^2, the kernel width, σ_0^2, the expected output noise for the function to learn, and β, the maximum number of training basis points to be kept by the algorithm. SOGP behavior is supposedly similar to standard GPR if no upper limit is set to this number of basis points. Besides the (common) parameters for each of the SOGP experts, ROGER also needs same parameters to be defined: the most important quantities, according to its authors, are P; the number of particles, α; the Chinese Restaurant Process concentration parameter that drives the propensity of new SOGP experts to be created within each particle; and the common parameters for each of its SOGP experts. In general, the higher the number of particles and SOGP capacity, the slower ROGER will run.

As for IMLE, there are 11 parameters that can be tuned to change the resulting behavior of the algorithm. Some of them typically do not need any tweaking, and the following experiences, unless otherwise noted, will keep them with their default values, namely, $n_\Lambda = 0.1$ (a small value is needed for regularization), $n_\nu = 0$, and $n_\mu = 0$ (experts locations fully learned). As the input space dimension increases, a stronger prior on input covariance matrices $\boldsymbol{\Sigma}_j$ and output noise $\boldsymbol{\Psi}_j$ is needed to make the learning process relatively invariant with respect to the trajectory nature of the training data acquisition process. A good rule of thumb is to set $n_\Sigma = n_\Psi = n_\sigma = 2^d$ and then choose n_ψ based on the confidence on the value of $\boldsymbol{\Psi}_0$, with smaller values corresponding to a larger uncertainty on this parameter. A typical value for the forgetting factor lies in the range $\alpha = 0.99 \sim 0.999$.

[1] LWPR 1.2.3: http://wcms.inf.ed.ac.uk/ipab/slmc/research/software-lwpr. SOGP 2.0, ROGER 1.5: http://cs.brown.edu/people/dang/code.shtml.

[2] IMLE 1.1: http://users.isr.ist.utl.pt/∼bdamas/IMLE.

[3] GPML 3.1: http://www.gaussianprocess.org/gpml/code/matlab/doc/.

The remaining parameters, $\mathbf{\Psi}_0$, $\mathbf{\Sigma}_0$, and p_0, have a strong influence in the experts' activation process, equation 3.8, and ultimately on the number of local experts created during the training phase. $\mathbf{\Psi}_0$ represents the expected output noise variance, while $\mathbf{\Sigma}_0$ corresponds to the input activation region for which the function to be learned can be approximately represented by a linear relation.

While setting and tuning such apparently high number of free parameters may appear to be challenging at first, the convergence of the probabilistic model is not very sensitive to specific values of these parameters. Perhaps the most sensible issue when considering the tuning of IMLE free parameters is to ensure a correct convergence of $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$, the input length scale and output noise estimates; this problem is discussed thoroughly next.

5.1 Single-Valued Function Aproximation. In this section we evaluate IMLE ability to perform single-valued function approximation, applying the algorithm to three different learning problems with different input dimensions and comparing its performance to LWPR, SOGP, and GPR. Some care must be taken when confronting these different learning schemes. In general, increasing the model complexity for each of the algorithms will produce smaller approximation errors, while incurring some heavier computational cost. The number of local linear models activated by IMLE and LWPR is a good measure of model complexity for these online algorithms. Its final value, after the training process, is a consequence of the choices for their tunable parameters and the training data themselves. Since IMLE and LWPR have the same computational complexity per training point, the final number of activated models provides a fair comparison ground for IMLE and LWPR in terms of the approximation error/computational complexity trade-off. GRP and SOGP model complexity, on the other hand, is measured by the number of stored training points used for posterior prediction over the test data. For GPR, this number is set beforehand, while SOGP learns a sparse subset of the training data to be used for prediction, possibly limiting the maximum number of these inducing points to a value of β. IMLE and LWPR computational demands are linear in the number of local models, while GPR and SOGP are much more penalized by the increase in the number in stored training points or inducing points, respectively.

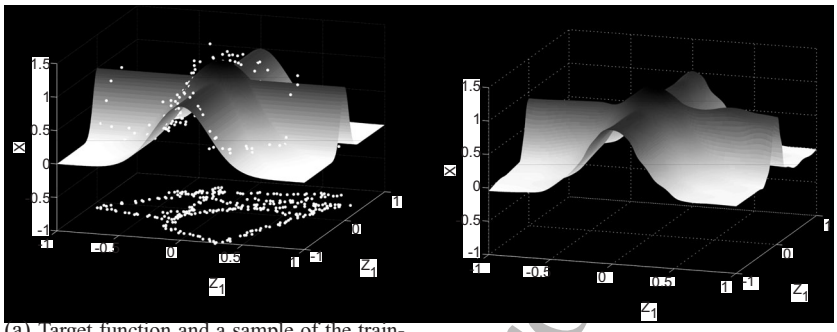
The amount of information implicitly available to the algorithms is another important issue concerning a fair comparison between them: for regression, the input length-scale of the data and the noise level present in the output are two critical properties of the function to be learned. GPR learns them offline by optimizing the likelihood of the training data with respect to these hyperparameters. Since GPR is an offline algorithm, its prediction performance strongly depends on the stored training points' input locations. If they efficiently cover all the input space, we expect GPR to outperform methods based on local linear approximations in terms of

prediction error. Yet such highly desirable informative training set may be unavailable or can be difficult to generate, as in typical robotic applications, where visiting all the input space can be very time-consuming.

Online methods alleviate this dependence on an initial representative training set by learning their models on the fly, adapting them as new training data arrive. This is achieved in SOGP by maintaining a representative subsample of the data. Parameters σ_k^2 and σ_0^2, however, are not adapted during the learning process, and thus SOGP must rely on a good initialization of its input length scale and output noise parameters. Adequate values can be obtained, for instance, from an initial offline optimization, similar to GPR. Both IMLE and LWPR learn the input length scale and output noise for each of their local linear models—in fact, they learn a full input distance metric, represented by covariance matrices $\mathbf{\Sigma}_j$ and $\boldsymbol{D}_j^{-1}$, respectively. They differ in the way they initialize these quantities: LWPR initializes the input distance metrics to a constant value $\boldsymbol{D}_{j_{init}}$; IMLE, on the other hand, puts a common prior on $\mathbf{\Sigma}_j$, defining then a vague hyperprior for $\bar{\mathbf{\Sigma}}$. While LWPR initialization strongly influences the number of receptive fields created during learning, in IMLE the information conveyed in hyperpriors' parameters $\mathbf{\Sigma}_0$ and $\mathbf{\Psi}_0$ can quickly lose importance if n_σ and n_ψ are small. This capability to learn, in an online fashion, the characteristic input length scale and output noise makes IMLE more robust to poor parameter initialization and less dependent on problem-specific knowledge; this complex probabilistic model structure, however, can make the learning convergence depend more on the training data input distribution, since hyperparameters $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$ strongly influence the behavior of newly activated experts, which also contribute to the estimation of $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$. Ultimately this can lead to convergence to poor local maxima of the likelihood function.

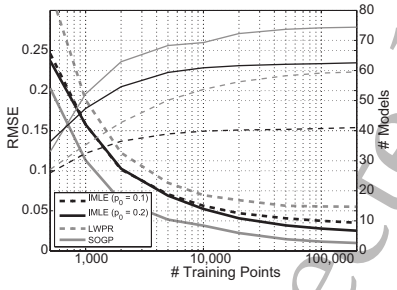
5.1.1 Cross Function. We first ran IMLE on a sequential stream of data taken from the cross function suggested in Vijayakumar et al. (2005), a two-dimensional input, univariate output function displayed in Figure 2a. The training set consisted of points sampled from a random trajectory performed in the input space and corresponding output data, for which we added gaussian noise with 0.1 standard deviation. A small sample of such training data can be seen in Figure 2a, superimposed on the target function.

We adopted all the suggested values for LWPR parameters presented in the cross-2D example given in LWPR source code, namely, $\boldsymbol{D}_{init} = 50\mathbf{I}$. SOGP was left with its default parameters of $\sigma_k^2 = \sigma_0^2 = 0.1$, with no limit on the number of inducing points. As for IMLE, we chose $\mathbf{\Sigma}_0 = 0.02\mathbf{I}$ to match the LWPR initial input covariance matrix, while defining $\mathbf{\Psi}_0 = 0.1^2$ and making $n_\Sigma = n_\Psi = n_\sigma = n_\psi = 2^d = 4$. For comparison purposes, we varied the parameter p_0, considering two values, $p_0 = 0.1$ and $p_0 = 0.2$. Since the output was known to be single-valued, we did not group experts' predictions into different solutions when estimating the output for the test input



(a) Target function and a sample of the training data (also projected on the z-plane to enhance the trajectory nature of data acquisition).

(b) Reconstructed function using IMLE.



(c) Learning curves for the Cross 2D function: RMSE and number of models created. For better visualization, we represent the number of stored induced points of SOGP scaled by 10.

	RMSE	# Models	CPU time (s)
$IMLE_{0.1}$	0.0351	41.04	$6.3 + 6.6$
$IMLE_{0.2}$	0.0252	62.58	$9.5 + 9.9$
LWPR	0.0550	59.55	$2.3 + 3.1$
SOGP	0.0098	744	$2600.9 + 593.9$
GPR_{1000}	0.0563	—	—
GPR_{5000}	0.0261	—	—

(d) Final RMSE, number of models and CPU time spent (training + testing).

Figure 2: The cross 2D data set.

data, setting $\alpha_{multi} = 1.0$ for the hypothesis testing described in section 4. Finally, we also trained a standard GP model, using a random, nonsequential training set with two different sizes (M = 1000 and M = 5000). All of the online algorithms were trained on a set of 200,000 sequential points coming from a random trajectory in the input space, and the prediction root mean square error (RMSE) was evaluated on a noiseless test grid of 200×200 equally spaced input points and corresponding output values. For accuracy, the IMLE and LWPR presented results are averages over 100 randomly trials.

Figure 2b shows a typical reconstruction of the original target function after learning ($p_0 = 0.2$). Figure 2c shows, for the online methods, the evolution of RMSE and number of created models as a function of the processed training points, and Figure 2d presents the final results after training. We

can see that SOGP achieves the best RMSE, but at a high computational cost, compared to the other online methods. LWPR, on the other hand, is the fastest algorithm but has a worse function approximation error when compared to IMLE, even when IMLE resorts to less local models ($p_0 = 0.1$). Note that increasing p_0 results in a better function approximation, at a penalty on the number of linear experts activated and consequent increase in computation time. As for offline GPR, low error rates can be obtained if the training set is large enough, but this comes at a prohibitive cost in terms of offline computation time and memory required to perform the necessary matrix inversions. We also used a random set for training the GP. As stated before, this may not be easy to generate in many practical real-time applications. This may also explain why SOGP has a better RMSE than GPR while using significantly fewer training points for prediction, since SOGP keeps only the most informative points taken out of the full training set.

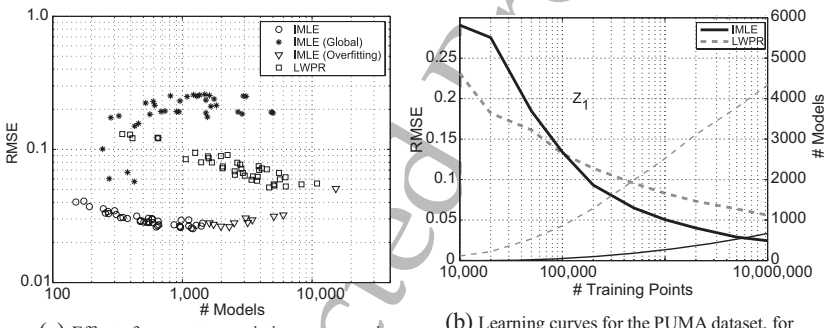
5.1.2 The PUMA 560 Serial Robot. The Unimation PUMA 560 is a well-known six degrees of freedom industrial robotic arm. Its forward kinematics function is described, for instance, in Craig (1989). To evaluate the single-valued prediction capabilities of IMLE, we simulated the PUMA 560 robot kinematics, defining a 10 cm tool extending along the z-axis of the frame associated with the sixth joint, and generated random trajectories over the joint space of the robot, calculating the corresponding 3D position of the tool tip. The kinematic function to be learned was thus a map from an input space of dimension 6 to an output space of dimension 3, even if the last joint was irrelevant as it changed only the tool orientation. Note that the fully stretched arm plus tool measured more than 90 cm, which made the range of each of the output variables to be almost 2 m.

We trained both LWPR and IMLE with a set of 10 million training points, evaluating the final achieved RMSE on a different test set comprising 100,000 points. Output training values were corrupted with gaussian noise, with standard deviation equal to 2 cm. This corresponds approximately to 1/100 of the output range and can model, for instance, moderate noise in a vision-based end-effector tracking process. SOGP unfortunately was left out of the comparisons. Its parameters turned out to be difficult to tune and its behavior unstable and very slow in the face of a large stream of highly correlated input data. We also considered standard GPR, generating a random set of training data for hyperparameter optimization and testing on the independent test set. Once again, the results thus obtained do not compare fairly to the other online algorithms, as one of the most challenging difficulties that arise with this data set is the massive, sequential, and correlated nature of the training data.

Experimental results are shown in Table 1, where the RMSE for two different instances of the GPR algorithm, trained with 1000 and 5000 data points, are presented. We set $\alpha = 0.999$, $p_0 = 0.1$ and the recommended values $n_\sigma = n_\Sigma = n_\Psi = 2^d = 64$, and since we did not wish to give IMLE any

Table 1: Results on the PUMA Data Set for IMLE, LWPR, and GPR Learning Algorithms: Final RMSE, Number of Models, and CPU Time (Training + Testing).

Method	RMSE	Number of Models	CPU Time (s)
IMLE	0.0245	668	4181 + 141
LWPR	0.0560	4338	11,974 + 725
GPR_{1000}	0.0513	—	—
GPR_{5000}	0.0144	—	—



(a) Effect of parameters variation on approximation error and number of local models created.

(b) Learning curves for the PUMA dataset, for a typical parameter configuration.

Figure 3: Learning the PUMA 560 Forward Kinematics Model.

information about the input length scale and output noise characteristics of the function to be learned, we defined $\mathbf{\Psi}_0 = \mathbf{I}$ and $\mathbf{\Sigma}_0 = \mathbf{I}$, and set n_ψ to a low value of 8 in order to quickly decay the influence of $\mathbf{\Psi}_0$ in the estimate for $\bar{\mathbf{\Psi}}$. As for LWPR, we picked the combination of parameters that made LWPR achieve the lowest error while not letting the number of created receptive fields grow to inadmissible values: $\gamma = 10^{-8}$, $w_{gen} = 0.1$, $\boldsymbol{D}_{init} = 10\mathbf{I}$, and $\alpha_{init} = 50$. Tuning these parameters to have roughly the same number of allocated models as IMLE never produced an RMSE of less than 10 cm on each output dimension. Figure 3b shows the corresponding learning curves for IMLE and LWPR. Due to the conservative large parameter values for $\mathbf{\Psi}_0$ and $\mathbf{\Sigma}_0$, IMLE converged slowly in the initial learning phase but quickly recovered after convergence of $\bar{\mathbf{\Psi}}$ and $\bar{\mathbf{\Sigma}}$ to better output noise and input length-scale estimates, respectively.

The LWPR algorithm was tested over an exhaustive combination of parameters: $\gamma \in \{10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}\}$, $w_{gen} \in \{0.1, 0.2\}$, $\boldsymbol{D}_{init} \in \{10, 20\}\mathbf{I}$, and $\alpha_{init} \in \{10, 20, 50, 150, 250\}$. These parameters affected the final error through the number of receptive fields they tended to create:

the larger this number, the lower the error, as expected. The same happened to IMLE when varying its tunable parameters, with $\mathbf{\Psi}_0 \in \{1.0, 0.0016, 0.0001\}\mathbf{I}$, $\mathbf{\Sigma}_0 \in \{4.0, 1.0, 0.25\}\mathbf{I}$, $n_\psi \in \{64, 32, 8\}$, $n_\sigma \in \{64, 32, 8\}$, and $p_0 \in \{0.03, 0.1, 0.3\}$. In general, a lower RMSE would be achieved at a cost of an increasing number of local linear models. It is very illustrative to depict the final RMSE as a function of the number of models created, for each parameter configuration of IMLE or LWPR, as in Figure 3a, since the number of local models affects the computational training and testing time in a similar way for these two algorithms. In the figure we can identify, for IMLE, three convergence behaviors: the first one, marked IMLE (Global), corresponds to a low value of $\mathbf{\Psi}_0 = 0.0001\mathbf{I}$. Since this value of $\mathbf{\Psi}_0$ is lower than the actual output noise, IMLE activates a large number of linear experts in the initial learning phase, each one covering all the input space (due to a high $\mathbf{\Sigma}_0$) and a particular region of the output space. This is of course an undesirable behavior that goes against the principle of localized learning. The second case, identified as IMLE (overfitting), is a consequence of choosing a low value of $\mathbf{\Sigma}_0$ in combination with a low or medium value of n_σ, allowing the individual $\mathbf{\Sigma}_j$ to shrink to a point where each new training point has a high probability of activating a new expert. This may result in a snowball effect, where more experts contribute to a decrease of $\bar{\mathbf{\Sigma}}$, which will lead to a smaller $\mathbf{\Sigma}_j$ and, consequently, more experts being activated. In Figure 3a, this does not result in a reduction of RMSE in the test set, a good indicator that IMLE is overfitting in that situation. These are, however, two extreme situations, caused by setting $\mathbf{\Psi}_0$ or $\mathbf{\Sigma}_0$ to smaller values than the output noise and input length scale, respectively. All other combinations of parameters exhibit good robustness regarding the training convergence. Note that in this case, IMLE has a much better performance than LWPR, achieving a much smaller RMSE while activating considerably fewer local models.

5.1.3 The SARCOS Inverse Dynamics Data Set. The SARCOS anthropomorphic robotic arm is a 7 degrees of freedom manipulator that has been used to test several function approximation algorithms. The learning task considered here is the estimation of its inverse dynamics from training examples. This is a nonlinear map from a 21-dimensional input space, consisting of positions, velocities, and accelerations of each joint to a 7-dimensional output space comprising the corresponding joint torques. The learned model can then be used to estimate the torques that achieve a desired trajectory in the joint space. The data set consists of 48,933 training points and 4449 test points, taken from trajectories performed with the real robot. Output values are normalized by the variance of the outputs of the test set to make the results presented here directly comparable to the ones in Rasmussen and Williams (2006) and Vijayakumar, D'Souza, Shibata, Conradt, and Schaal (2002). We compare IMLE results on this data set to the following models:

Linear regression (LR): A linear regression model is fitted to the data to provide a baseline for comparison.

Rigid body dynamics (RBD): This is a parametric physics-based model for the inverse dynamics function that is estimated using a least-squares approach with the available training data.

LWPR: This model was trained using diagonal distance metrics $\boldsymbol{D}_j$, cycling through the training data over 6 million iterations. This roughly corresponds to 123 passes over the full training data.

GPR: Due to the computational infeasibility of using the full training data for optimization of hyperparameters and prediction over test data, a subset of regressors method was employed, with size 4096. A squared exponential covariance function was used, and its hyperparameters were optimized accordingly, using a subset of the training data.

As for IMLE, we used out-of-the-box default parameters, setting conservative large values for its noise and length-scale parameters, $\boldsymbol{\Psi}_0 = \mathbf{I}$ and $\boldsymbol{\Sigma}_0 = \mathbf{I}$, while setting the usual value of $p_0 = 0.1$. Using the input dimension to set up the value of n_Σ, n_Ψ, and n_σ according to the rule of thumb we presented before would result in overly large prior strengths. We resort instead to the notion that for actual robot movements, the generated data tend to be low-dimensional, with around four to six effective dimensions (Schaal, Vijayakumar, & Atkeson, 1998). This is a standard assumption in robotic applications in order to circumvent the curse of dimensionality. Learning a sensorimotor map would require a full exploration of the input space if this condition did not hold. In such case, as the input dimension increased, the time required for exploration and learning would grow exponentially, making the learning task infeasible from a practical point of view. In our experiments, we found that a value of seven effective dimensions could provide satisfactory learning behavior and thus set $n_\Sigma = n_\Psi = n_\sigma = 2^7$. We also set n_ψ to a low value of 8 due to the high uncertainty on $\boldsymbol{\Psi}_0$. Varying n_ψ over a considerably wide interval around this value did not change IMLE convergence behavior significantly.

Unlike GPR and LWPR, the IMLE algorithm can directly provide multivariate output predictions without a need to train different models for each of the output dimensions. If the maps from inputs to each output variable have the same length-scale properties, as frequently happens in robotic sensorimotor maps, a huge economy of computational resources can be attained, as each local linear model can describe the interaction between the inputs and the full output vector. We show in Table 2 the prediction error when the output consists of only the first joint torque, comparing it to the results presented in Rasmussen and Williams (2006) for LWPR, GPR, RBD, and LR. We also used IMLE to learn the full output torque vector. For both situations, we present the MSE and number of activated experts after (1) a full pass over the training data and (2) 10 consecutive passes over the

Table 2: Results on the SARCOS Data Set for LR, RBD, LWPR, GPR, and IMLE Methods.

Method	MSE	Number of Models	Method	MSE	Number of Models
LR	0.075	—	IMLE_{1D} (1 epoch)	0.019	313
RBD	0.104	—	IMLE_{1D} (10 epochs)	0.010	563
LWPR	0.040	260	IMLE_{7D} (1 epoch)	0.018	271
GPR	0.011	—	IMLE_{7D} (10 epochs)	0.010	550

Note: IMLE_{1D} is the model corresponding to a map from $\mathbb{R}^{21}$ to $\mathbb{R}^1$, while IMLE_{7D} is the model obtained from the full map from $\mathbb{R}^{21}$ to $\mathbb{R}^7$.

same data. For confirmation purposes, we also trained IMLE with $p_0 = 0.0$, resulting in a model with a single expert that provided a global linear approximation to the training data. As expected, an MSE equal to that of the LR model was then obtained.

The extremely good convergence of the IMLE algorithm is noteworthy. After only a single pass through the training data, IMLE achieves a better approximation error than LWPR (after more than 100 passes through the same data) while activating a comparable number of linear models. If more points are presented to IMLE, cycling through the data ten times, an MSE comparable to state-of-the-art GPR is achieved. It is also worth noticing that IMLE performance did not changed much when a full 7-dimensional output vector was considered for the learning task: the MSE remained the same, while, perhaps surprisingly, the number of experts even dropped a bit.[4] We did not notice any increased computational time in this situation, as the slight increase in computation due to this higher-output dimension was balanced by a smaller number of activated models. In this aspect, IMLE compares favorably to GPR and LWPR, which would require seven times more computational power to learn the full inverse dynamics map.

5.2 Multivalued Function Aproximation. We now proceed to evaluating IMLE learning capabilities under a multivalued target function scenario. As stated in section 3.2, learning multivalued functions in an online fashion poses several additional problems. Without further information, it is difficult to distinguish noise or outliers from new multivalued function branches. In addition, it is no longer possible to set large values for $\mathbf{\Sigma}_0$ and $\mathbf{\Psi}_0$, hoping that the learning process finds adequate values for the input length scale and output noise estimate, as multivalued relations may then be interpreted as single-valued functions with large output noise. There is also the problem of time-varying functions. This issue can easily be addressed in online single-valued algorithms by introducing forgetting mechanisms

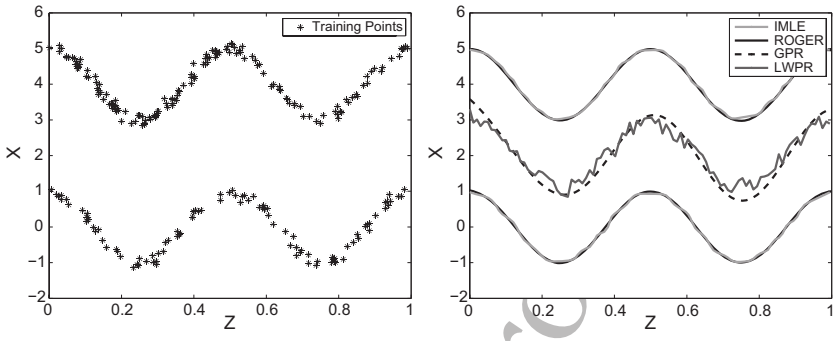
[4]This probably is explained by a lesser tendency for overfitting when the output dimension increases.

to allow for a quick adaptation of the internal model to the time-varying data. However, things become more complicated with multivalued function approximation, where there is no clear border between fast time-changing training data and multivalued functions.

We tested IMLE against ROGER whenever possible. This later algorithm, however, consists of an infinite mixture of SOGPs, thus inheriting the same limitations mentioned in the previous section: the difficulty in getting a good parameter configuration and the slow operation for medium- or high-dimensional input spaces. Also, ROGER implementation code did not provide a set of multivalued solutions for the function being approximated, instead sampling a single solution from the infinite mixture. This feature imposes severe limitations to real applications, as ROGER predictions will permanently switch between different multivalued branches. To provide a fair comparison to IMLE, we adapted ROGER code to provide a finite set of prediction solutions. For each input query, ROGER was used to predict a solution over 100 different trials, and the distinct solutions thus obtained were gathered to become the set of predicted solutions.

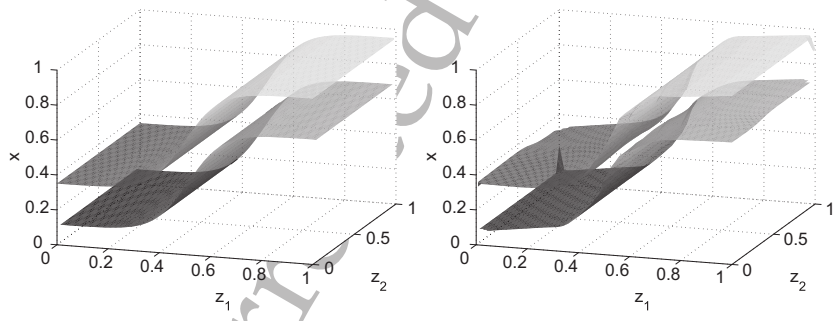
The approximation error was measured by taking, for each input query, the multivalued prediction closest to the true output. Note that we do not address here the important problem of choosing a solution among the set of multivalued predictions produced by the algorithm. This can be seen as a context estimation, and it is a requirement when the multivalued model is to be used for control purposes. In the following experiments, we set $\alpha_{multi} = 0.9$. The lower this value was, the higher the number of solutions found for the same query on average. However, when the branches of the multivalued function to learn are well separated in the output space, we found that the value of this parameter did not influence the final prediction much, and that setting it on a range of, say, $\alpha_{multi} \in [0.8, 0.99]$ resulted in similar prediction results.

5.2.1 Synthetic Data Sets. To illustrate the fundamental differences between multi- and single-valued function approximation algorithms, we start with a simple toy example consisting of a multivalued target sinusoidal function. Standard function approximation learning methods, like LWPR or GPR, typically behave poorly in this setting. Training data were generated by alternating sequential sweeps over each of the two branches of the multivalued function, given, respectively, by $f_1(x) = \cos(z)$ and $f_2(x) = \cos(z) + 4$, and each output x_i was corrupted with gaussian noise with standard deviation equal to 0.1. Figure 4a represents a sample of the training data, while Figure 4b depicts estimates from IMLE, ROGER, LWPR, and GPR taken after training. As expected, LWPR and GPR tend to average the two branches of the multivalued function, while IMLE and ROGER correctly identify the two solutions. In particular, ROGER achieves an almost perfect approximation to the true function (RMSE = 0.014), when compared to IMLE (RMSE = 0.039).



(a) Training data. (b) Reconstructing the relation.

Figure 4: Synthetic multivalued sinusoidal relation.



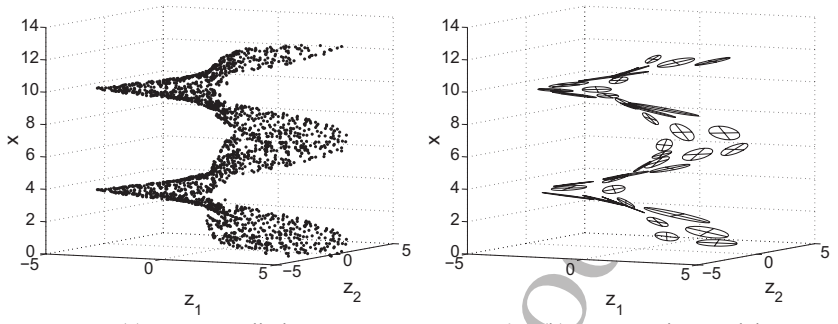
(a) Target function. (b) IMLE prediction.

Figure 5: Toy example suggested by Shizawa (1996).

Shizawa (1996) suggests a synthetic data set that consists of a multivalued function from $\mathbb{R}^2$ to $\mathbb{R}^1$, with two distinct branches, described by equations

$$f_1(z_1,z_2)=\frac{0.6}{1+e^{-15(z_1-0.5)}}+0.1 \quad \text{and} \quad f_2(z_1,z_2)=\frac{0.6}{1+e^{-15(z_1-0.5)}}+0.35.$$

The target function is depicted in Figure 5a, while IMLE multivalued prediction, after conducting a 10,000-point training phase, is represented in Figure 5b. This toy example is particularly challenging for multivalued prediction. The two function branches are very close to each other and can easily become merged during learning. IMLE achieves an RMSE of 0.017 in a grid-like independent, noise-free test set. ROGER was not able to generate a good estimate. It systematically produced, for every tested parameter configuration, more than 20 SOGP experts. This led to a severe output



(a) IMLE prediction. (b) IMLE active models.

Figure 6: Prediction using the toy example suggested by Lee and Lee (2001).

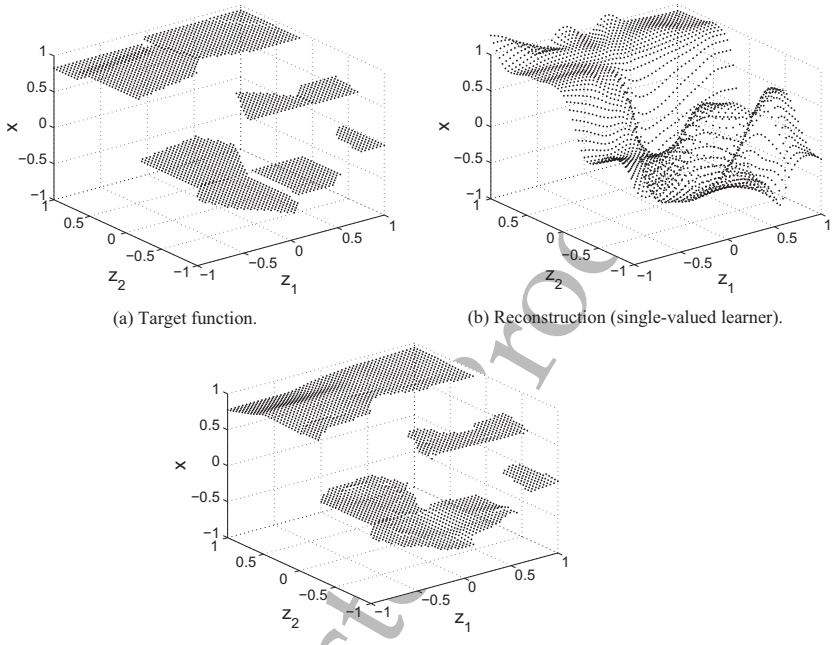
prediction interference, with typically more than 10 solutions generated for each input query.[5] Another toy example, presented now by Lee and Lee (2001), consists of a cylindrical spiral surface, again a multivalued function from $\mathbb{R}^2$ to $\mathbb{R}^1$, described by

$$x = \tan^{-1}\left(\frac{z_2}{z_1}\right), \quad \text{where } x \in [0, 4\pi), \quad \text{and} \quad 0.2 < \sqrt{z_1^2 + z_2^2} < 5.$$

The codomain of the target function makes each input point have two distinct solutions. IMLE was trained with random trajectories generated over the target function, and after 50,000 points, it achieved an RMSE of less than 0.065 on an independent random test set. The multivalued predicted outputs for this test set are shown in Figure 6a, while the 39 linear models allocated by the algorithm are represented in Figure 6b.

The final toy example we present consists of a randomly generated piecewise constant target function from $\mathbb{R}^2$ to $\mathbb{R}^1$, shown in Figure 7a. Although not multivalued, many function approximation algorithms will not be able to properly learn it, as their smoothness assumptions can conflict with the discontinuities of the target function. This will typically result in either an overfitting to the data, when the complexity of the model is increased to approximate the function in the vicinity of the discontinuities, or in an oversmoothing behavior, where the predictions average the two values of the target function near the function transitions.

[5] This behavior was unfortunately observed in the following experiments, and so ROGER was removed from the remaining tests. Note that the ROGER good results shown in Figure 4b were a consequence of a careful choice of its parameters: small changes in these values would typically lead to more than two solutions being predicted in many regions of the input space.



(a) Target function.

(b) Reconstruction (single-valued learner).

(c) Reconstruction (IMLE).

Figure 7: Learning functions with discontinuities.

Multivalued learning algorithms based on mixtures like IMLE can provide an elegant solution to this problem. During training, there is no output interference near the discontinuities, since in those regions, the data are simply interpreted as coming from a multivalued function. When predicting, a single-valued solution can nevertheless be provided by simply taking the most important solution from the multivalued prediction set, according to the sum of weights $w_j^x(z_q)$ of the experts that contribute to that solution. This procedure is able to generate sudden transitions of the prediction, like the algorithm developed by Toussaint and Vijayakumar (2005) to specifically deal with discontinuous functions. Prediction results are shown in Figure 7 for IMLE and a single-valued learner. We will not undergo a full comparison to other discontinuity function learning algorithms, as the main point here is the proof of the concept that IMLE (and other multivalued learning algorithms) can also efficiently approximate discontinuous functions without the need for any special modifications.

5.2.2 The Humanoid iCub Robot. The next experiment evaluates IMLE multivalued prediction capabilities in a higher-dimensional problem. We consider iCub, an anthropomorphic robot used for research into human

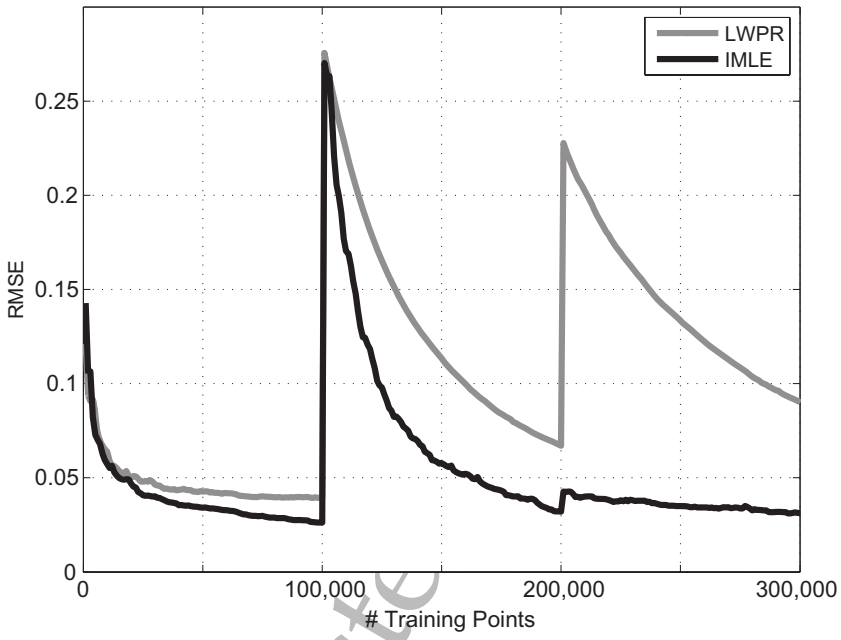


Figure 8: Online learning of a sensorimotor map under switching kinematic contexts.

cognition and artificial intelligence (Sandini, Metta, & Vernon, 2004). We simulated random trajectories in a seven-dimensional input space consisting of the joint angles of the robot waist (yaw, roll, and pitch) and right arm (shoulder yaw, pitch and roll, and elbow flexion), the output space being the 3D position of the end effector. During the training phase, after acquiring 100,000 data points, a 28 cm tool was introduced in the kinematic chain of the arm, effectively changing the end-effector position. After 100,000 more training points, the tool was removed and training resumed. From the viewpoint of the learning algorithm, this nonsignaled change of the kinematic structure can be represented by a multivalued function with two distinct branches corresponding to the tool and no-tool contexts. Traditional single-valued approximators must forget the previously learned map when the tool is introduced in order to learn the new kinematics. This wastes computational resources, as the original map must be learned again when the tool is removed and the original kinematics structure is once again presented to the robot. IMLE can keep both situations stored in its internal model: there is almost no increase in the RMSE when the tool is removed, as can be seen in Figure 8, where, for comparison with a single-valued learner, we also show the results of training LWPR using the same data set.

5.2.3 Simultaneously Learning of Forward and Inverse Models. In the following experiments, we show how IMLE can use the learned mixture to predict both forward and inverse maps. We use the Puma 560 robot arm described in section 5.1, controlling its first three joints to position the end effector. The kinematic function for this robot configuration is a map from $\mathbb{R}^3$ to $\mathbb{R}^3$. While the forward map consists of a single-valued function, the inverse kinematics is multivalued and can exhibit up to four solutions. As a second example, we consider the 3-RPR parallel manipulator described in Merlet (2006). It consists of an end effector connected to a fixed base through three prismatic links, each connecting to the base and end effector using free, unactuated rotational joints. Its movement is restricted to the *x*-*y* plane. Actuating on link lengths L_1, L_2, and L_3 changes the *x*-*y* end-effector position and θ orientation on this plane. The kinematic for this mechanism is also a map from $\mathbb{R}^3$ to $\mathbb{R}^3$. Parallel robots typically exhibit a duality relation to serial chains with respect to the forward and inverse kinematics nature. While their inverse relation is usually unique and straightforward to calculate, obtaining a closed formula for the end-effector position and orientation as a function of actuator values is difficult and frequently yields multiple valid solutions. This mechanism is known to have up to six different solutions for the same actuator configuration, which makes learning its forward kinematics infeasible for most standard single-valued function approximation techniques.

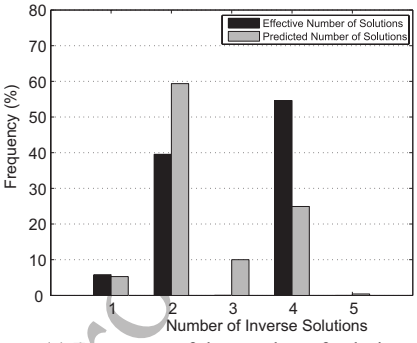
We trained IMLE over 1 million points taken from a simulated random trajectory for both manipulators, adding gaussian noise to the outputs, with standard deviation equal to 1/100 of the output range. We restricted the movement of the 3-RPR parallel manipulator to a square of 40 cm by 40 cm in the center of the mechanism, while the angle was constrained to the interval $[-\pi/2; \pi/2]$. We ran 10 different trials, testing the resulting final mixtures on a noiseless random sequence of 100,000 points. The results are shown in Figure 9. Figure 9c also presents, for the PUMA robot, the frequency of the number of solutions found by IMLE inverse prediction, comparing it to the real value, obtained by explicitly solving the inverse kinematics equations. The discrepancy between these numbers can be explained by the fact that close to the work space boundary of the PUMA 560, there are pairs of inverse solutions that become close to each other. In this situation, IMLE tends to merge these solutions. Lowering the value of α_{multi} would reduce this behavior; however, due to the curvature of the map to learn, this would have the undesired side effect of predictions of neighbor experts being erroneously taken as separate solutions. In general, choosing a value for α_{multi} is a compromise between this two effects. Nevertheless, IMLE still achieves a good inverse prediction error rate, since in the work space boundary, the merged solution provided by IMLE is approximately the average of two reasonable similar true solutions. As for the 3-RPR manipulator, IMLE found on average 1.88 forward solutions per test point, which, given the constrained work space, agrees with the expected number

#Models	Forward RMSE (cm)			Inverse RMSE (%)		
	x	y	z	θ_1	θ_2	θ_3
163.0	1.54	1.61	1.13	3.71	3.91	1.43

(a) Puma Serial robot.

#Models	Forward RMSE (%)			Inverse RMSE (cm)		
	x	y	θ	L_1	L_2	L_3
91.3	0.70	0.70	1.12	0.27	0.26	0.26

(b) Parallel 3-RPR robot.



(c) Frequency of the number of solutions found by IMLE inverse prediction.

Figure 9: Simultaneous learning of forward and inverse kinematic relations.

of solutions. For every point on the test set, only a single inverse solution was found. This is in total agreement with the single-valued nature of the inverse kinematics for these kind of mechanisms.

6 Discussion and Conclusion

The infinite mixture of linear models presented in this letter is, at its core, an algorithm that can efficiently deal with nonlinear function approximation in an online, incremental manner, comparable to current state-of-the-art online learning methods. It consists of a collection of linear experts, together with appropriate priors on the parameters being learned and a mechanism that efficiently grows the number of experts when the need to explain outfitted data arises for newly acquired samples. Its training is based on the generalized EM algorithm, where the expectation step is extended to allow for incremental updating of the sufficient statistics of the mixture of experts and the maximization step includes the allocation of a new expert each time a training point is poorly explained by the current mixture. Put together, this results in a very fast and scalable online learning algorithm, capable of processing hundreds or thousands of samples per second, coming from continuous streams of data. This is a difficult learning setting, since when no knowledge is provided about the characteristics of the function to learn, this information must be estimated from a sequence of correlated training data that may correspond to only a small subset of the full input-output space. We tested IMLE in this kind of situation for single-valued regression and showed how it could equal or even surpass LWPR, a current state-of-the-art online learning algorithm, in terms of convergence of prediction error and number of allocated models.

However, a distinctive feature of IMLE, when compared to other online supervised learning algorithms, is its ability to deal with multivalued estimates for the same query data. The applications for such kind of problems range from learning forward models of parallel robots to learning switching models, where the function to be approximated can alternate between different configurations over time. This constitutes an even more challenging learning problem. Besides the limitations coming from learning online from a stream of data, we now face the expert allocation dilemma, where it is difficult to distinguish between noise and outliers, on one hand, and a mixture requiring a new component, on the other. Additionally, underlying nonstationary relations make the problem even more difficult to learn. Analyzing the influence of nonstationarity in multivalued prediction, relating it to separability of multivalued function branches, is perhaps a topic for further research.

We have also shown in this letter that the same procedure used to obtain a set of multivalued forward solutions can be applied to inverse queries, making IMLE capable of delivering both forward and inverse multivalued predictions from the same model, without need of further training. This is a consequence of directly learning a multivariate output relation from $\mathbb{R}^d$ to $\mathbb{R}^D$ instead of a set of D distinct univariate output maps. As seen in the previous section, such multivalued learning capability can also prove to be useful when learning discontinuous functions, for which an undesirable prediction smoothing typically occurs in the vicinity of the discontinuities when using standard function approximation algorithms.

A current limitation of the proposed algorithm is the lack of a mixture shrinking mechanism that would be responsible for removing either experts providing wrong predictions or redundant mixture components. The experiments presented in section 5 show that the IMLE model will activate new experts more and more sparingly as the training progresses, but eventually, after a lifetime of learning, too many experts may become activated, resulting in an increase of computational resources consumption. Moreover, in the event of some episodic outlier bursts, some experts may be activated to represent such erroneous training data. We do not yet have a definite answer to this problem. As discussed in section 3.2, growing the mixture under incremental assumptions, for multivalued data is a delicate matter, and shrinking the same mixture in a principled way is even more troublesome. The main difficulty here is that there is no simple way to detect experts wrongly activated by an outlier (or a sequence of outliers), as they cannot easily be distinguished from experts originating from a sporadic training on a new branch of the multivalued function being learned. Such "incorrect" experts can be detected, for instance, by a strong deviation from the characteristic input length scale or output noise (strong disagreement between $\mathbf{\Sigma}_j$ and $\bar{\mathbf{\Sigma}}$ or between $\mathbf{\Psi}_j$ and $\bar{\mathbf{\Psi}}$) or by low support on training data (low accumulation of sufficient statistics after a long period of

training). However, removing an expert under this condition does not come without the risk of lowering the likelihood of the training data, since no guarantees exist that such an expert does not represent the true distribution of the multivalued function being learned. In general, choosing an expert to be removed from the mixture—due, for instance, to capacity limitations imposed on the mixture—is not easy to do without access to the full set of training points. Creating a probabilistically supported shrinking mechanism for the mixture is surely an improvement to the IMLE model that we would like to investigate in subsequent work.

IMLE has a large space for customization by choosing different priors for the mixture parameters, the most evident being the prior on regression coefficients $\mathbf{\Lambda}_j$. Feature selection is a very desirable property of a function approximation algorithm that allows learning the input subspace that effectively contributes to output variation. This typically makes the prediction more stable and less influenced by irrelevant or strongly correlated input dimensions. In a Bayesian setup, feature selection can be implemented if adequate priors for loading matrices $\mathbf{\Lambda}_j$ are defined. Adopting a gaussian prior, as IMLE currently does, leads to the ridge regression (Hoerl & Kennard, 1970), while a Laplacian prior induces the LASSO (Tibshirani, 1996). Both priors involve the choice of a hyperparameter to control the degree of regularization and sparseness. Figueiredo (2003) uses a Jeffrey's prior to overcome the necessity of such a hyperparameter. More recently, Ting, D'Souza, Vijayakumar, and Schaal (2010) proposed a slightly different formulation of the generative model corresponding to the linear regression performed by each expert, which together with a careful choice of priors for the elements of $\mathbf{\Lambda}_j$ can lead to a fast and efficient high-dimensional feature selection and regression. In principle, any of these priors can be integrated into IMLE; this is a topic for future work. Another research direction is the inclusion of Dirichlet priors for mixture components, turning IMLE into a fully Bayesian learning method. In this situation, the major challenge would be retaining the online and scalability properties of the algorithm.

Appendix: Posterior Distributions for IMLE

We can use the fact that distributions 2.2 are conjugate to data likelihood (see equation 2.1) to derive the posterior distribution for the mixture parameters at iteration t, given the current estimates for $\bar{\mathbf{\Sigma}}$ and $\bar{\mathbf{\Psi}}$. Since the normal-inverse Wishart is a conjugate prior for the center and covariance matrix of a multivariate normal data distribution (Gelman et al., 2004), we have, for each expert j,

$$\boldsymbol{\nu}_j, \mathbf{\Sigma}_j | \boldsymbol{S}^t, \bar{\mathbf{\Sigma}} \sim \mathcal{NW}^{-1}(\boldsymbol{\nu}^*, n_\nu^*, \mathbf{\Sigma}^*, n_\Sigma^*),$$

where

$$
\begin{aligned}
n_{\nu}^{*} &= n_{\nu} + S_{hj}, \\
\boldsymbol{\nu}^{*} &= \frac{n_{\nu}\boldsymbol{\nu}_{0j} + \boldsymbol{S}_{hzj}}{n_{\nu} + S_{hj}}, \\
n_{\Sigma}^{*} &= n_{\Sigma} + S_{hj} \text{ and} \\
\boldsymbol{\Sigma}^{*} &= n_{\Sigma}\bar{\boldsymbol{\Sigma}} + n_{\nu}\boldsymbol{\nu}_{0j}\boldsymbol{\nu}_{0j}^{T} + \boldsymbol{S}_{hzzj} - n_{\nu}^{*}\boldsymbol{\nu}_{j}^{*}\boldsymbol{\nu}_{j}^{*T}.
\end{aligned}
$$

Equations 3.7a and 3.7b directly follow from the preceding equations, as the maximum value for the normal-inverse Wishart distribution is achieved by $\hat{\boldsymbol{\nu}}_j = \boldsymbol{\nu}^*$ and $\hat{\boldsymbol{\Sigma}}_j = \boldsymbol{\Sigma}^*/(n_{\Sigma}^* + d + 2)$. The predictive distribution $p(\boldsymbol{z}_q|w_j, \boldsymbol{S}^t, \bar{\boldsymbol{\Sigma}})$ can then be obtained if we take the marginal distribution of $p(\boldsymbol{z}_q, \boldsymbol{\Sigma}_j, \boldsymbol{\nu}_j|w_j, \boldsymbol{S}^t, \bar{\boldsymbol{\Sigma}})$ with respect to unknown parameters $\boldsymbol{\Sigma}_j$ and $\boldsymbol{\nu}_j$, which results in a multivariate t-Student distribution with $n_{\Sigma}^* - d + 1$ degrees of freedom,

$$
\boldsymbol{z}_q|w_j, \boldsymbol{S}^t, \bar{\boldsymbol{\Sigma}} \sim t_{n_{\Sigma}^*-d+1}\left(\boldsymbol{\nu}^*, \frac{n_{\nu}^* + 1}{n_{\nu}^*(n_{\Sigma}^* - d + 1)}\boldsymbol{\Sigma}^*\right). \tag{A.1}
$$

The normal-inverse gamma distribution is a conjugate prior for the noise and regression coefficients under the standard linear regression likelihood model (O'Hagan & Forster, 1994). We can use a different parameterization of equation 2.1a to infer the posterior distribution of $\boldsymbol{\Lambda}_j$, $\boldsymbol{\mu}_j$, and $\boldsymbol{\Psi}_j$ at iteration t, by defining $\tilde{\boldsymbol{\Lambda}}_j \equiv [\boldsymbol{\Lambda}_j, \bar{\boldsymbol{\mu}}_j]$, where $\bar{\boldsymbol{\mu}}_j \equiv \boldsymbol{\mu}_j - \boldsymbol{\Lambda}_j\boldsymbol{\nu}_j$, and expanding the input vector to accommodate a constant term, $\tilde{\boldsymbol{z}} \equiv [\boldsymbol{z}^T, 1]^T$. We are then able to rewrite equation 2.1a as $\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}, \boldsymbol{\Theta} \sim \mathcal{N}(\tilde{\boldsymbol{\Lambda}}_j\tilde{\boldsymbol{z}}_i, \boldsymbol{\Psi}_j)$ and change the corresponding priors, equations 2.2c and 2.2d accordingly, $\tilde{\boldsymbol{\Lambda}}_j(k)|\boldsymbol{\Psi}_j(k) \sim \mathcal{N}(\tilde{\boldsymbol{\Lambda}}_0(k), \boldsymbol{\Psi}_j(k)\tilde{\boldsymbol{R}}_{\Lambda})$, with $\tilde{\boldsymbol{\Lambda}}_0 = [\boldsymbol{0}, \boldsymbol{\mu}_{0j}]$ and

$$
\tilde{\boldsymbol{R}}_{\Lambda}^{-1} = \begin{bmatrix} n_{\Lambda}\mathbf{I} & & \boldsymbol{0} \\ & & \vdots \\ \boldsymbol{0} & \cdots & n_{\mu} \end{bmatrix}.
$$

The posterior distribution for $(\tilde{\boldsymbol{\Lambda}}_j(k), \boldsymbol{\Psi}_j(k))$ then becomes

$$
\tilde{\boldsymbol{\Lambda}}_j(k), \boldsymbol{\Psi}_j(k)|\boldsymbol{S}^t, \bar{\boldsymbol{\Psi}} \sim \mathcal{N}G^{-1}\left(\tilde{\boldsymbol{\Lambda}}^*(k), \tilde{\boldsymbol{R}}_{\Lambda}^*, \frac{n_{\Psi}^*}{2}, \frac{\boldsymbol{\Psi}^*(k)}{2}\right), \tag{A.2}
$$

where

$$\tilde{\boldsymbol{R}}_{\Lambda}^{*} = (\tilde{\boldsymbol{R}}_{\Lambda}^{-1} + \tilde{\boldsymbol{S}}_{hzzj})^{-1},$$
$$\tilde{\boldsymbol{\Lambda}}^{*} = (\tilde{\boldsymbol{S}}_{hxzj} + \tilde{\boldsymbol{\Lambda}}_{0}\tilde{\boldsymbol{R}}_{\Lambda}^{-1})\tilde{\boldsymbol{R}}_{\Lambda}^{*} = [\boldsymbol{\Lambda}^{*}, \bar{\boldsymbol{\mu}}^{*}],$$
$$n_{\Psi}^{*} = n_{\Psi} + S_{hj},$$
$$\boldsymbol{\Psi}^{*} = n_{\Psi}\bar{\boldsymbol{\Psi}} + \operatorname{diag}\{\boldsymbol{S}_{hxxj} + \tilde{\boldsymbol{\Lambda}}_{0}\tilde{\boldsymbol{R}}_{\Lambda}^{-1}\tilde{\boldsymbol{\Lambda}}_{0}^{T} - \tilde{\boldsymbol{\Lambda}}^{*}(\tilde{\boldsymbol{R}}_{\Lambda}^{*})^{-1}\tilde{\boldsymbol{\Lambda}}^{*T}\},$$

with $\tilde{\boldsymbol{S}}_{hxzj} = [\boldsymbol{S}_{hxzj}, \boldsymbol{S}_{hxj}]$ and

$$\tilde{\boldsymbol{S}}_{hzzj} = \begin{bmatrix} \boldsymbol{S}_{hzzj} & \boldsymbol{S}_{hzj} \\ \boldsymbol{S}_{hzj}^{T} & S_{hj} \end{bmatrix}$$

accounting for the extended input vector $\tilde{\boldsymbol{z}}$. Noting that $\tilde{\boldsymbol{R}}_{\Lambda}^{*}$ can be written as

$$\tilde{\boldsymbol{R}}_{\Lambda}^{*} = \begin{bmatrix} \boldsymbol{R}_{\Lambda}^{*} & -\boldsymbol{R}_{\Lambda}^{*}\frac{\boldsymbol{S}_{hzj}}{S_{hj}+n_{\mu}} \\ -\frac{\boldsymbol{S}_{hzj}^{T}}{S_{hj}+n_{\mu}}\boldsymbol{R}_{\Lambda}^{*} & R_{\bar{\boldsymbol{\mu}}}^{*} \end{bmatrix}, \quad \text{where}$$
$$\boldsymbol{R}_{\Lambda}^{*} = \left(n_{\Lambda}\mathbf{I} + \boldsymbol{S}_{hzzj} - \frac{\boldsymbol{S}_{hzj}\boldsymbol{S}_{hzj}^{T}}{S_{hj}+n_{\mu}}\right)^{-1} \quad \text{and}$$
$$R_{\bar{\boldsymbol{\mu}}}^{*} = \frac{1}{S_{hj}+n_{\mu}} + \frac{\boldsymbol{S}_{hzj}^{T}}{S_{hj}+n_{\mu}}\boldsymbol{R}_{\Lambda}^{*}\frac{\boldsymbol{S}_{hzj}}{S_{hj}+n_{\mu}},$$

we easily arrive at

$$\boldsymbol{\Lambda}^{*} = \left(\boldsymbol{S}_{hxzj} - \frac{\boldsymbol{S}_{hxj} + n_{\mu}\boldsymbol{\mu}_{0j}}{S_{hj}+n_{\mu}}\boldsymbol{S}_{hzj}^{T}\right)\boldsymbol{R}_{\Lambda}^{*},$$
$$\bar{\boldsymbol{\mu}}^{*} = \frac{\boldsymbol{S}_{hxj} + n_{\mu}\boldsymbol{\mu}_{0j}}{S_{hj}+n_{\mu}} - \boldsymbol{\Lambda}^{*}\frac{\boldsymbol{S}_{hzj}}{S_{hj}+n_{\mu}} \quad \text{and}$$
$$\boldsymbol{\Psi}^{*} = n_{\Psi}\bar{\boldsymbol{\Psi}} + \operatorname{diag}\{\boldsymbol{S}_{hxxj} - \boldsymbol{\Lambda}^{*}\boldsymbol{S}_{hxzj}^{T} - \bar{\boldsymbol{\mu}}^{*}(\boldsymbol{S}_{hxj} + n_{\mu}\boldsymbol{\mu}_{0j})^{T}\}.$$

Equation A.2 can be split as $p(\tilde{\boldsymbol{\Lambda}}_{j}(k)|\boldsymbol{\Psi}_{j}(k), \boldsymbol{S}^{t})p(\boldsymbol{\Psi}_{j}(k)|\boldsymbol{S}^{t}, \bar{\boldsymbol{\Psi}})$, where

$$\tilde{\boldsymbol{\Lambda}}_{j}(k)|\boldsymbol{\Psi}_{j}(k), \boldsymbol{S}^{t} \sim \mathcal{N}(\tilde{\boldsymbol{\Lambda}}^{*}(k), \boldsymbol{\Psi}_{j}(k)\tilde{\boldsymbol{R}}_{\Lambda}^{*}) \quad \text{and}$$
$$\boldsymbol{\Psi}_{j}(k)|\boldsymbol{S}^{t}, \bar{\boldsymbol{\Psi}} \sim \mathcal{G}^{-1}\left(\frac{n_{\Psi}^{*}}{2}, \frac{\boldsymbol{\Psi}^{*}(k)}{2}\right),$$

and consequently $\mathbf{\Lambda}_j(k)$ and $\bar{\boldsymbol{\mu}}_j(k)$ are jointly normal given $\mathbf{\Psi}_j(k)$, with

$$\mathbf{\Lambda}_j(k)|\mathbf{\Psi}_j(k), \boldsymbol{S}^t \sim \mathcal{N}(\mathbf{\Lambda}^*(k), \mathbf{\Psi}_j(k)\boldsymbol{R}^*_\Lambda),$$
$$\bar{\boldsymbol{\mu}}_j(k)|\mathbf{\Psi}_j(k), \boldsymbol{S}^t \sim \mathcal{N}(\bar{\boldsymbol{\mu}}^*(k), \mathbf{\Psi}_j(k)R^*_{\bar{\boldsymbol{\mu}}})$$

and cross-variance equal to $-\boldsymbol{R}^*_\Lambda \frac{S_{hzj}}{S_{hj}+n_\mu}$. As a consequence, $\boldsymbol{\mu}_j(k)$ is also normally distributed given $\mathbf{\Psi}_j(k)$, with $\boldsymbol{\mu}_j(k)|\mathbf{\Psi}_j(k), \boldsymbol{S}^t \sim \mathcal{N}(\boldsymbol{\mu}^*(k), \mathbf{\Psi}_j(k)R^*_\mu)$, where

$$\boldsymbol{\mu}^* = \frac{\boldsymbol{S}_{hxj} + n_\mu \boldsymbol{\mu}_{oj}}{S_{hj} + n_\mu} + \mathbf{\Lambda}^* \left(\hat{\mathbf{v}}_j - \frac{\boldsymbol{S}_{hzj}}{S_{hj} + n_\mu} \right) \text{ and}$$
$$R^*_\mu = \frac{1}{S_{hj} + n_\mu} + \left(\frac{\boldsymbol{S}_{hzj}}{S_{hj} + n_\mu} - \hat{\mathbf{v}}_j \right)^T \boldsymbol{R}^*_\Lambda \left(\frac{\boldsymbol{S}_{hzj}}{S_{hj} + n_\mu} - \hat{\mathbf{v}}_j \right).$$

Equations 3.7c to 3.7e arise from the previous posterior distributions, since $\hat{\mathbf{\Lambda}}_j = \mathbf{\Lambda}^*$, $\hat{\boldsymbol{\mu}}_j = \boldsymbol{\mu}^*$ and $\hat{\mathbf{\Psi}}_j = \mathbf{\Psi}^*/(n^*_\Psi + 2)$ are the modes of the respective distributions.

In order to obtain $p(\mathbf{\Lambda}_j|\boldsymbol{S}^t)$ and $p(\boldsymbol{\mu}_j|\boldsymbol{S}^t)$, we must marginalize equation A.2 with respect to $\mathbf{\Psi}_j$. A well-known result states that the resulting distribution becomes a multivariate t-Student with n^*_Ψ degrees of freedom. When this value is large, the distribution can be approximated by a multivariate normal. For $\mathbf{\Lambda}_j$ and $\boldsymbol{\mu}_j$ this results in

$$\mathbf{\Lambda}_j(k)|\boldsymbol{S}^t \underset{a}{\sim} \mathcal{N}(\mathbf{\Lambda}^*(k), \boldsymbol{R}^*_\Lambda \ \mathbf{\Psi}^*(k)/n^*_\Psi) \quad \text{and}$$
$$\boldsymbol{\mu}_j(k)|\boldsymbol{S}^t \underset{a}{\sim} \mathcal{N}(\boldsymbol{\mu}^*(k), R^*_\mu \ \mathbf{\Psi}^*(k)/n^*_\Psi).$$

Finally, to obtain the posterior predictive distribution for $\boldsymbol{x}|\boldsymbol{z}_q$, we use the fact that the marginalization of $p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}, \mathbf{\Theta})$ with respect to the parameters $\mathbf{\Psi}_j$, $\boldsymbol{\mu}_j$, and $\mathbf{\Lambda}_j$ yields again a t-Student distribution, with n^*_Ψ degrees of freedom, mean equal to

$$\hat{\boldsymbol{x}}_j(\boldsymbol{z}_q) = \tilde{\mathbf{\Lambda}}^* \tilde{\boldsymbol{z}}_q = \hat{\mathbf{\Lambda}}_j(\boldsymbol{z}_q - \hat{\mathbf{v}}_j) + \hat{\boldsymbol{\mu}}_j \tag{A.3}$$

and variance given by

$$\boldsymbol{R}^x_j(\boldsymbol{z}_q) = (1 + \tilde{\boldsymbol{z}}^T_q \tilde{\boldsymbol{R}}^*_\Lambda \tilde{\boldsymbol{z}}_q) \frac{\mathbf{\Psi}^*}{n^*_\Psi} = (1 + \gamma_j(\boldsymbol{z}_q)) \frac{\mathbf{\Psi}^*}{n^*_\Psi} \approx (1 + \gamma_j(\boldsymbol{z}_q)) \hat{\mathbf{\Psi}}_j, \tag{A.4}$$

where the factor

$$\gamma_j(z_q) = \frac{1}{S_{hj} + n_\mu} + \left(z_q - \frac{S_{hzj}}{S_{hj} + n_\mu}\right)^T \boldsymbol{R}^*_\Lambda \left(z_q - \frac{S_{hzj}}{S_{hj} + n_\mu}\right)$$

reflects the uncertainty on the estimates $\hat{\boldsymbol{\mu}}_j$ and $\hat{\boldsymbol{\Lambda}}_j$ used in the posterior prediction. Note that as the training size increases, this term vanishes in equation A.4, while the fundamental source of noise due to $\boldsymbol{\Psi}_j$ remains. Put together, this results in

$$\boldsymbol{x}|z_q, w_j, \boldsymbol{S}^t, \bar{\boldsymbol{\Psi}} \underset{a}{\sim} \mathcal{N}(\hat{\boldsymbol{x}}_j(z_q), \boldsymbol{R}^x_j(z_q)), \tag{A.5}$$

where again we approximate the t-Student distribution to a normal one, under the assumption of a large value of n^*_Ψ. It is useful to view this result under a different and equivalent formulation, given by the hierarchical model

$$\boldsymbol{x}|\boldsymbol{x}_j, w_j, \boldsymbol{S}^t, \bar{\boldsymbol{\Psi}} \underset{a}{\sim} \mathcal{N}(\boldsymbol{x}_j, \hat{\boldsymbol{\Psi}}_j), \tag{A.6}$$

$$\boldsymbol{x}_j|z_q, \boldsymbol{S}^t, \bar{\boldsymbol{\Psi}} \underset{a}{\sim} \mathcal{N}(\hat{\boldsymbol{x}}_j(z_q), \gamma_j(z_q)\hat{\boldsymbol{\Psi}}_j), \tag{A.7}$$

which can be interpreted as a sample point $\boldsymbol{x}(z_q)$ being generated, with noise $\hat{\boldsymbol{\Psi}}_j$, from an unknown mean $\boldsymbol{x}_j$, for which posterior distribution A.7 is available given the current set of sufficient statistics and input query z_q.

Acknowledgments

We thank the anonymous reviewers for their valuable comments and suggestions, which greatly contributed to improving the quality of this letter. We are also grateful to Lorenzo Jamone for sharing the iCub data set.

References

Antoniak, C. (1974). Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *Annals of Statistics*, *2*(6), 1152–1174.

Atkeson, C., Moore, A., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*(1), 11–73.

Bishop, C. (1994). *Mixture density networks* (Tech. Rep.). Birmingham, UK: Neural Computing Research Group.

Bishop, C. (1995). *Neural networks for pattern recognition*. New York: Oxford University Press.

Brouwer, R. (2004). Feed-forward neural network for one-to-many mappings using fuzzy sets. *Neurocomputing*, *57*, 345–360.

Cappé, O., & Moulines, E. (2009). Online EM algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *71*(3), 593–613.

Carreira-Perpiñán, M. (2000). Mode-finding for mixtures of gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(11), 1318–1323.

Chernova, S., & Veloso, M. (2008). Learning equivalent action choices from demonstration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008* (pp. 1216–1221). Piscataway, NJ: IEEE.

Craig, J. (1989). *Introduction to robotics: Mechanics and control*. Reading, MA: Addison-Wesley Longman.

Csató, L., & Opper, M. (2002). Sparse on-line gaussian processes. *Neural Computation*, *14*(3), 641–668.

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

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Learning inverse kinematics. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Volume 1, pp. 298–303). Piscataway, NJ: IEEE.

Figueiredo, M. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*(9), 1150–1159.

Figueiredo, M., & Jain, A. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 381–396.

Gelman, A., Carlin, J., Stern, H., & Rubin, D. (2004). *Bayesian data analysis*. Boca Raton, FL: Chapman and Hall/CRC.

Ghahramani, Z., & Hinton, G. (1997). *The EM algorithm for mixtures of factor analyzers* (Tech. Rep. CRG-TR-96-1). Toronto: University of Toronto.

Ghahramani, Z., & Jordan, M. (1994). Supervised learning from incomplete data via an EM approach. In J. D. Cowan, G. Tesauro, & J. Alspector (Eds.), *Advances in neural information processing systems, 6*. San Mateo, CA: Morgan Kaufmann.

Gomes, R., Welling, M., & Perona, P. (2008). Incremental learning of nonparametric Bayesian mixture models. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008*. Piscataway, NJ: IEEE.

Grollman, D., & Jenkins, O. (2010). Incremental learning of subtasks from unsegmented demonstration. In *International Conference on Intelligent Robots and Systems 2010* (pp. 261–266). Piscataway, NJ: IEEE.

Hastie, T., & Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, *8*(2), 120–129.

Hoerl, A., & Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*, 55–67.

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Jordan, M., & Jacobs, R. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*(2), 181–214.

Kendall, M. G., Stuart, A., Ord, K., Arnold, S., & O'Hagan, A. (1998). *Kendall's advanced theory of statistics, vol. 2. Classical inference and the linear model*. New York: Wiley.

Lee, K., & Lee, T. (2001). Design of neural networks for multi-value regression. In *Proceedings of the International Joint Conference on Neural Networks, 2001* (Vol. 1, pp. 93–98). Piscataway, NJ: IEEE.

Lopes, M., & Damas, B. (2007). A learning framework for generic sensory-motor maps. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007* (pp. 1533–1538). Piscataway, NJ: IEEE.

Ma, J., Theiler, J., & Perkins, S. (2003). Accurate on-line support vector regression. *Neural Computation*, *15*(11), 2683–2703.

McLachlan, G., & Peel, D. (2000). *Finite mixture models*. New York: Wiley.

Meeds, E., & Osindero, S. (2006). An alternative infinite mixture of gaussian process experts. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in neural information processing systems, 18* (pp. 883–890). Cambridge, MA: MIT Press.

Merlet, J. 2006). *Parallel robots*. New York: Springer-Verlag.

Neal, R., & Hinton, G. (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 355–368). Norwell, MA: Kluwer.

Nguyen-Tuong, D., & Peters, J. (2008). Local gaussian process regression for real-time model-based robot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008* (pp. 380–385). Piscataway, NJ: IEEE.

O'Hagan, A., & Forster, J. (1994). *Kendall's advanced theory of statistics: Vol. 2B; Bayesian inference*. Ultimo, NSW, Australia: Halsted Press.

Peters, J., & Schaal, S. (2006). Learning operational space control. *Robotics: Science and Systems (RSS 2006)*. Cambridge, MA: MIT Press.

Qin, C., & Carreira-Perpinán, M. (2008). Trajectory inverse kinematics by conditional density modes. In *IEEE International Conference on Robotics and Automation, 2008* (pp. 1979–1986). Piscataway, NJ: IEEE.

Quiñonero-Candela, J., & Rasmussen, C. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, *6*, 1939–1959.

Rasmussen, C. (2000). The infinite gaussian mixture model. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems, 12* (pp. 554–560). Cambridge, MA: MIT Press.

Rasmussen, C., & Ghahramani, Z. (2002). Infinite mixtures of gaussian process experts.

Rasmussen, C., & Williams, C. (2006). *Gaussian processes for machine learning*. Cambridge, MA: MIT Press.

Ray, S., & Lindsay, B. (2005). The topography of multivariate normal mixtures. *Annals of Statistics*, *33*(5), 2042–2065.

Sandini, G., Metta, G., & Vernon, D. (2004). Robotcub: An open framework for research in embodied cognition. In *IEEE-RAS/RSJ International Conference on Humanoid Robots (Humanoids 2004)*. Piscataway, NJ: IEEE.

Sato, M. (2001). Online model selection based on the variational Bayes. *Neural Computation*, *13*(7), 1649–1681.

Sato, M., & Ishii, S. (2000). On-line EM algorithm for the normalized gaussian network. *Neural Computation*, *12*(2), 407–432.

Schaal, S., & Atkeson, C. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*(8), 2047–2084.

Schaal, S., Vijayakumar, S., & Atkeson, C. (1998). Local dimensionality reduction. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems, 10* (pp. 633–639). Cambridge, MA: MIT Press.

Shizawa, M. (1994). Regularization networks for approximating multi-valued functions: learning ambiguous input-output mappings from examples. In *1994 IEEE International Conference on Neural Networks* (Vol. 1, pp. 137–142). Piscataway, NJ: IEEE.

Shizawa, M. (1996). Multivalued regularization network-a theory of multilayer networks for learning many-to-h mappings. *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, *79*(9), 98–113.

Smola, A., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society (B)*, *58*, 267–288.

Ting, J., D'Souza, A., Vijayakumar, S., & Schaal, S. (2010). Efficient learning and feature selection in high-dimensional regression. *Neural Computation*, *22*(4), 831–886.

Tomikawa, Y., & Nakayama, K. (1998). Approximating many valued mappings using a recurrent neural network. *Proc. of IJCNN'98,* (Vol. *2*, pp. 1494–1497).

Toussaint, M., & Vijayakumar, S. (2005). Learning discontinuities with products-of-sigmoids for switching between local models. In *Proceedings of the 22nd International Conference on Machine Learning* (pp. 904–911). New York: ACM.

Vijayakumar, S., D'Souza, A., & Schaal, S. (2005). Incremental online learning in high dimensions. *Neural Computation*, *17*(12), 2602–2634.

Vijayakumar, S., D'Souza, A., Shibata, T., Conradt, J., & Schaal, S. (2002). Statistical learning for humanoid robots. *Autonomous Robots*, *12*(1), 55–69.

Vlassis, N., & Likas, A. (2002). A greedy EM algorithm for gaussian mixture learning. *Neural Processing Letters*, *15*(1), 77–87.

Wang, C., & Blei, D. (2012). Truncation-free online variational inference for Bayesian nonparametric models. In P. Bartlett, F.C.N. Pereira, C.J.C. Burges, L. Bottou, & K.Q. Weinberger (Eds.), *Advances in neural information processing systems, 25* (pp. 422–430). Red Hook, NY: Curran.

Whitehead, S., & Ballard, D. (1991). Learning to perceive and act by trial and error. *Machine Learning*, *7*(1), 45–83.

Wilson, S. (2002). Classifiers that approximate functions. *Natural Computing*, *1*(2), 211–234.

Xu, L., Jordan, M., & Hinton, G. (1995). An alternative model for mixtures of experts. In *Advances in neural information processing systems, 7* (pp. 633–640). Cambridge, MA: MIT Press.

Received June 28, 2012; accepted April 13, 2013.