2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

# An Online Algorithm for Simultaneously Learning Forward and Inverse Kinematics

Bruno Damas and José Santos-Victor

*Abstract*— **This paper proposes a supervised algorithm for online learning of input-output relations that is particularly suitable to simultaneously learn the forward and inverse kinematics of general manipulators — the multi-valued nature of the inverse kinematics of serial chains and forward kinematics of parallel manipulators makes it infeasible to apply state-of-the-art learning techniques to these problems, as they typically assume a single-valued function to be learned.**

**The proposed algorithm is based on a generalized expectation maximization approach to fit an infinite mixture of linear experts to an online stream of data samples, together with an outlier probabilistic model that dynamically grows the number of linear experts allocated to the mixture, this way 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, which can directly provide forward and inverse multi-valued estimates. The experiments presented in this paper show that it can achieve, for single-valued functions, a performance directly comparable to state-of-the-art online function approximation algorithms, while additionally providing inverse predictions and the capability to learn multi-valued functions in a natural manner. To our knowledge this is a distinctive property of the algorithm presented in this paper.**

## I. Motivation and Related Work

This work presents a new online learning algorithm that can successfully be applied to forward and inverse kinematics estimation for both serial and parallel manipulators. The forward kinematics learning problem can be stated as the estimation of a relation $\boldsymbol{X} = f(\boldsymbol{q})$ relating the task space coordinates $\boldsymbol{X}$ of the end effector of the robot with respect to a given reference frame (*e.g.* the 3D position and orientation of the end effector frame) to the joint space where $\boldsymbol{q}$, the controlled joint variables, are defined. Inverse kinematics, on the other side, describes the inverse problem of obtaining the joint variables corresponding to a given task space vector, *i.e.*, finding $\boldsymbol{q} = g(\boldsymbol{X})$.

The main problem that arises when learning such kinematics is that even for non redundant manipulators, where the task and joint space dimensions are equal, $f$ or $g$ may not be proper functions, exhibiting multiple solutions for the same input argument — these are known as multi-valued functions or *multimaps*, one-to-many maps where a single input can be associated with one or more different outputs. Inverse kinematics of serial robots, for instance, exhibit this multiple solution behaviour; on the other hand, the inverse situation often arises with parallel robots, where a single value for the actuation variables may correspond to distinct values of the task space vector [1], [2].

Although much work has been done to derive analytical solutions for a broad class of manipulators, there are situations where an analytical model fails to provide an accurate approximation to the real robot. These situations range from lack of knowledge of certain hard to measure physical parameters (*e.g.* friction) to highly non-linear physical interactions, such as actuator nonlinearities and unmodeled mass distributions [3], [4]. In such complex situations we must resort to modern supervised learning techniques to provide these systems with the necessary representation capability. In the context of robotic applications the need for online and incremental learning algorithms is often also a requirement, particularly when the prediction and learning are not two distinct phases: this makes some state-of-the-art non-linear function approximation methods, like Support Vector Regression and Gaussian Process Regression (GPR), non suitable for these kind of applications, as they require, in their original form, either the presence of all data points in memory or computational demands that are not suitable for online learning. Recently some modifications where suggested to address this problem, such as sparse and local versions of GP [4]: however, all of these methods can only provide single-valued predictions and are not able to provide solutions for the inverse problem.

*Locally Weighted Projection Regression* (LWPR) is another type of non-linear function approximation, very popular and widely used for online, real-time learning of robotic tasks: it performs spatially localized learning based on linear local models, making use of an incremental partial least squares algorithm to deal with redundant and high-dimensional input spaces [5]. Its excellent memory requirements and computational complexity have made LWPR a reference online learning algorithm. Yet it cannot deal with multi-valued functions, since LWPR adjustment of the size of its receptive fields is based on a stochastic gradient minimization of the output cross-validation error. Also, LWPR cannot provide inverse predictions in its original form: in [6] inverse kinematics for serial robots are learned using LWPR, but this approach, like other inverse kinematics learning methods, only obtains a single inverse solution by application of some sort of constraining or optimization criterion.

Unsupervised learning algorithms, such as mixture of Gaussians (MoG), can be applied to supervised learning tasks, using the conditional densities obtained from the

Bruno Damas and José Santos-Victor are with the Instituto de Sistemas e Robótica, Instituto Superior Técnico, Lisboa, Portugal. Bruno Damas is also with Escola Superior de Tecnologia de Setúbal, Portugal. {bdamas,jasv}@isr.ist.utl.pt. This work was partially funded by FCT (PEst-OE/EEI/LA0009/2011) and EU Projects POETICON++ (FP7-ICT-288382) and HANDLE (FP7-ICT-231640).

learned joint density function over both inputs and outputs [7], [8]: this, in principle, makes it easier to learn multi-valued functions, as no particular modification of the unsupervised algorithm is needed to deal with such structure in the input-output relation. However, unsupervised learning is a more difficult problem than its supervised counterpart and usually results in a worse performance, as it ignores that joint data, apart from noise corruption, lies in a lower dimensional manifold: not acknowledging this fact typically results in the convergence to sub-optimal solutions that do not take the problem structure into account.

Somewhat related to LWPR, another popular approach to nonlinear function learning is based on the Mixtures of Experts (ME) concept [9], 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. In its simplest form a ME is an offline algorithm, for which the number of components — number of experts — must be set beforehand. [10] suggests 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. However, it considers only single-value forward estimation, and the regularization and allocation of experts is performed in a somewhat heuristic way.

Still, the mixture of experts concept introduces a framework suitable for the multi-valued function learning and prediction, as different experts can be allocated to different solutions in the output space corresponding to the same input. In [11] this approach was followed, by considering each expert to be a sparse GP. Like the algorithm presented in this paper, it is one of the few supervised learning algorithms that is explicitly able to learn multi-valued functions. However, the inference procedure, based on Monte-Carlo integration over the posterior distribution, imposes some computational limitations on the algorithm, and no method to perform inverse prediction is given. Another approach [12] considers a mixture of neural networks to account for the multi-valued nature of the function to estimate: differently from our work, they use an iterative mode finding algorithm; also, their learning method cannot provide forward and inverse predictions from the same learned model.

The Infinite Mixture of Linear Experts (IMLE) proposed in this paper is an online, real-time supervised learning algorithm that is able to learn multi-valued functions while allocating experts to the mixture as needed, achieving performance comparable to some state-of-the-art methods. It is, at its core, an infinite mixture version of the model for mixture of linear experts given by [13], [10], with a careful choice of priors for some of its parameters and the adoption of a Generalized Expectation-Maximization (GEM) approach that allows, in the first place, an online operation based on the concepts present in [14], [15], and secondly, the automatic allocation of experts to the mixture in order to adapt to the complexity of the function to be learned. This probabilistic model is detailed in Section II, while the GEM algorithm to train it is presented in Section III. Section IV describes how to obtain forward and inverse multi-valued predictions for any query, given the current state of the IMLE model. IMLE is experimentally compared to LWPR in Section V, and some other experiments are conducted to provide evidence of the good performance of the proposed algorithm, focusing especially on the multi-valued function approximation. Finally, Section VI provides the concluding remarks.

## II. Probabilistic Model

The Infinite Mixture of Linear Experts assumes the following generative model for a sample point $(\boldsymbol{x}_i, \boldsymbol{z}_i)$, where $\boldsymbol{z}_i \in \mathbb{R}^d$ is the predictor, $\boldsymbol{x}_i \in \mathbb{R}^D$ the corresponding quantitative response and $\Theta$ the mixture parameters to learn:

$$p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}; \Theta) \sim \mathcal{N}\left(\boldsymbol{\mu}_j + \boldsymbol{\Lambda}_j(\boldsymbol{z}_i - \boldsymbol{\nu}_j), \boldsymbol{\Psi}_j\right) , \quad (1)$$

$$p(\boldsymbol{z}_i|w_{ij}; \Theta) \sim \mathcal{N}\left(\boldsymbol{\nu}_j, \boldsymbol{\Sigma}_j\right) , \quad (2)$$

$$p(w_{ij}; \Theta) = m_j . \quad (3)$$

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 above equations): an improper constant prior $m_j$ is assigned to this latent variable, equal to 1 if linear model $j$ is consider to be present in the mixture and 0 otherwise. This allows us to select which experts contribute to the overall mixture, this way controlling the mixture effective size, as it will be explained subsequently in the text. Given $w_{ij}$, input $\boldsymbol{z}_i$ follows a Normal distribution with location and shape parameters $\boldsymbol{\nu}_j$ and $\boldsymbol{\Sigma}_j$, while output $\boldsymbol{x}_i$ follows a linear relation from $\boldsymbol{z}_i$, with location parameter $\boldsymbol{\mu}_j$, design matrix $\boldsymbol{\Lambda}_j$ and diagonal covariance matrix $\boldsymbol{\Psi}_j$, corresponding to uncorrelated noise in the response $\boldsymbol{x}$. This model, apart the improper prior for $w_{ij}$, is similar to the one presented in [13], 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. $\Theta$ is the (infinite) parameter vector that defines this mixture, given by $\Theta = \sigma \cup \left\{\boldsymbol{\nu}_j, \boldsymbol{\Sigma}_j, \boldsymbol{\mu}_j, \boldsymbol{\Lambda}_j, \boldsymbol{\Psi}_j, m_j\right\}_{(1 \leq j \leq \infty)}$, to be learned from the data.

Additionally, the following priors on the parameters of the mixture are defined:

$$p(\boldsymbol{\nu}_j, \boldsymbol{\Sigma}_j|\sigma) \sim \mathcal{N}W^{-1}(\boldsymbol{\nu}_{0j},\ n_\nu,\ n_\Sigma \sigma \boldsymbol{I},\ n_\Sigma) , \quad (4)$$

$$p(\sigma) \sim \mathcal{G}^{-1}(n_\sigma, n_\sigma \sigma_0) , \quad (5)$$

$$p(\boldsymbol{\Lambda}_j^k, \boldsymbol{\Psi}_j^k) \sim \mathcal{N}\mathcal{G}^{-1}\left(\boldsymbol{\Lambda}_0 = 0, n_\Lambda^{-1}\boldsymbol{I}, \frac{n_\Psi}{2},\ \frac{n_\Psi}{2}\boldsymbol{\Psi}_0^k\right) . \quad (6)$$

Here, $\mathcal{N}W^{-1}$ and $\mathcal{N}\mathcal{G}^{-1}$ denote respectively multivariate Normal-Inverse Wishart and univariate Normal-Inverse Gamma distributions. $\boldsymbol{\Psi}_j^k$ and $\boldsymbol{\Psi}_0^k$ are the $k^{ths}$ elements of the respective diagonal matrices, while $\boldsymbol{\Lambda}^k$ corresponds to the $k^{th}$ row of the design matrix. $n_\nu$, $n_\Sigma$, $n_\Psi$ and $n_\Lambda$ are constants determining the "strength" of the respective priors, $\boldsymbol{\nu}_{0j}$, $\boldsymbol{\Sigma}_0 = \sigma I$, $\boldsymbol{\Psi}_0$ and $\boldsymbol{\Lambda}_0$, expressed as an equivalent

number of "fake" data points. Parameter $\sigma_0$ is an initial guess for $\sigma$ and $n_\sigma$ defines the strength of such belief.

The purpose of the common spherical prior for $\mathbf{\Sigma}_j$ is threefold: it introduces some regularization, so that $\mathbf{\Sigma}_j$ has always an inverse; second, it ensures that the experts input areas shapes do not differ too much from each other, and finally, it prevents non-neighboring experts from competing for the same data in the initial phase of the learning process of each expert — a serious problem occurring in mixtures of experts models, referred for instance in [5], thus enforcing the principle of localized learning. The inverse-gamma hyper-prior on $\sigma$ allow us to define an initial guess for $\sigma$, as well a corresponding degree of belief in this guess. The normal prior on $\boldsymbol{\nu}_j$ controls the degree of "mobility" of this parameter: $n_\nu = 0$ makes it dependent solely on the perceived data, while $n_\nu = \infty$ leads to a fixed input center for each expert, as occurs typically in Radial Basis Networks or in LWPR. The prior on $\mathbf{\Psi}_j$ defines the initial guess for the output noise and is closely related to the admissible error of IMLE prediction, ultimately controlling the degree of overfitting/oversmoothing of the algorithm. Finally, a prior $\mathbf{\Lambda}_0 = 0$ for the design matrix $\mathbf{\Lambda}$ performs a coefficient shrinkage similar to ridge regression. It's main purpose, however, is to impose a regularization mechanism, in order to make the matrix inversion required when estimating this parameter always full rank. Such prior introduces some bias in the expert prediction, and consequently $n_\Lambda$ should be kept to a low value in order to make such undesired effect negligible. Lastly, a special class $w_0$ is defined, corresponding to outliers that are not generated by any of the linear experts, with an improper prior $p(w_0) = 1$ and an improper constant distribution

$$p(\boldsymbol{x}_i, \boldsymbol{z}_i | w_0) = \frac{1}{(2\pi|\mathbf{\Psi}_0|)^{\frac{D}{2}}} \frac{1}{(2\pi|\sigma \boldsymbol{I}|)^{\frac{d}{2}}} e^{-0.5 K_{d+D}} \; .$$

Here $K_{d+D}$ is obtained from the inverse of the chi-squared cumulative distribution function, with $d+D$ degrees of freedom, for a given probability $1-p_0$. This makes $p(\boldsymbol{x}_i, \boldsymbol{z}_i | w_0)$ directly comparable to the probability density $p(\boldsymbol{x}_i, \boldsymbol{z}_i | w_{ij})$ for a newly activated, not yet trained expert $j$, evaluated at a point $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ that lies over the equidensity contour that encircles the region, centered at $(\boldsymbol{\mu}_j, \boldsymbol{\nu}_j)$, corresponding to a $1-p_0$ probability. Parameter $p_0$ thus controls the probability of a point being generated by the outlier model: the lower the value of $p_0$ the lower $p(\boldsymbol{x}_i, \boldsymbol{z}_i | w_0)$ is, making a point less probable to be an outlier. As described in Section III, this class has an important role on the process of assigning new experts to the mixture, which heavily depends on the posterior probability of a point being generated by the outlier model: equation (II) has the important property of making such process approximately invariant with respect to input and output dimension, input scale and output noise.

## III. Learning

Training the IMLE model can be accomplished by the Expectation-Maximization (EM) algorithm [16]: assuming $\boldsymbol{Z}$, $\boldsymbol{X}$ and $\boldsymbol{W}$ to contain respectively the input sample $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$ and the corresponding output and hidden indicator samples, the log-likelihood of the complete data is given by

$$\begin{aligned} &l(\Theta; \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W}) = \log p(\Theta | \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W}) \\ &\propto \log \left[ p(\Theta) \prod_{i=1}^{N} p(\boldsymbol{x}_i | \boldsymbol{z}_i, \boldsymbol{w}_i; \Theta) p(\boldsymbol{z}_i | \boldsymbol{w}_i; \Theta) p(\boldsymbol{w}_i; \Theta) \right] \; , \end{aligned} \tag{7}$$

where $\boldsymbol{w}_i$ is defined as a vector whose entries are equal to $w_{ij}$ and probabilities appearing therein are given by $p(\boldsymbol{x}_i | \boldsymbol{z}_i, \boldsymbol{w}_i; \Theta) = \prod_{j=1}^{\infty} p(\boldsymbol{x}_i | \boldsymbol{z}_i, w_{ij}; \Theta)^{w_{ij}}$, $p(\boldsymbol{z}_i | \boldsymbol{w}_i; \Theta) = \prod_{j=1}^{\infty} p(\boldsymbol{z}_i | w_{ij}; \Theta)^{w_{ij}}$ and $p(\boldsymbol{w}_i; \Theta) = \prod_{j=1}^{\infty} p(w_{ij}; \Theta)^{w_{ij}}$.

The (batch) EM algorithm then produces a sequence of estimates $\hat{\Theta}^t$ by alternating between the expectation step (E-Step), which calculates the so called Q-function $Q(\Theta, \hat{\Theta}^t)$, the conditional expectation of $l(\Theta; \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W})$ with respect to the latent variables $w_{ij}$, for the current value of $\hat{\Theta}^t$, and the maximization step (M-Step), that finds the new value of $\hat{\Theta}^{t+1}$ given the previous expectation.

### A. E-Step

The log-likelihood is linear with respect to the latent variables $w_{ij}$, and hence it suffices, to obtain $Q(\Theta, \hat{\Theta}^t)$, to calculate $h_{ij}^t = E[w_{ij} | \boldsymbol{X}, \boldsymbol{Z}; \hat{\Theta}^t]$, 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 only $\boldsymbol{x}_i$ and $\boldsymbol{z}_i$ depend on $w_{ij}$, we have, using Bayes' theorem,

$$\begin{aligned} h_{ij}^t &\equiv E[w_{ij} | \boldsymbol{X}, \boldsymbol{Z}; \hat{\Theta}^t] = p(w_{ij} | \boldsymbol{x}_i, \boldsymbol{z}_i; \hat{\Theta}^t) \\ &= \frac{p(\boldsymbol{x}_i | \boldsymbol{z}_i, w_{ij}; \hat{\Theta}^t) p(\boldsymbol{z}_i | w_{ij}; \hat{\Theta}^t) \hat{m}_j^t}{\sum_{k=1}^{\infty} p(\boldsymbol{x}_i | \boldsymbol{z}_i, w_{ik}; \hat{\Theta}^t) p(\boldsymbol{z}_i | w_{ik}; \hat{\Theta}^t) \hat{m}_k^t} \; ; \end{aligned} \tag{8}$$

with this result, the Q-function to be maximized in the following M-Step becomes

$$\begin{aligned} &Q(\Theta, \hat{\Theta}^t) \equiv E_{\boldsymbol{W}}[l(\Theta; \boldsymbol{X}, \boldsymbol{Z}, \boldsymbol{W}); \hat{\Theta}^t] = \log p(\Theta) + \\ &\sum_{i=1}^{N} \sum_{j=1}^{\infty} h_{ij}^t \log \left[ p(\boldsymbol{x}_i | \boldsymbol{z}_i, w_{ij}; \Theta) p(\boldsymbol{z}_i | w_{ij}; \Theta) p(w_{ij}; \Theta) \right] \; . \end{aligned} \tag{9}$$

### B. M-Step

The Q-function in (9) belongs to the *exponential family*, and consequently can be expressed as a function of the expected value of the sufficient statistics $\boldsymbol{S}^t$, comprising the following terms:

$$\begin{aligned} S_{hj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \; , & \boldsymbol{S}_{hzzj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \boldsymbol{z}_i \boldsymbol{z}_i^T \; , \\ \boldsymbol{S}_{hzj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \boldsymbol{z}_i \; , & \boldsymbol{S}_{hxzj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \boldsymbol{x}_i \boldsymbol{z}_i^T \text{ and} \\ \boldsymbol{S}_{hxj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \boldsymbol{x}_i \; , & \boldsymbol{S}_{hxxj}^t &= \textstyle\sum_{i=1}^N h_{ij}^t \boldsymbol{x}_i \boldsymbol{x}_i^T \; . \end{aligned} \tag{10}$$

M-Step assigns $\hat{\Theta}^{t+1}$ to the value of $\Theta$ that maximizes $Q(\Theta, \hat{\Theta}^t)$. We use the fact that most of the priors are conjugate to the data likelihood to derive the new estimates $\hat{\Theta}^{t+1}$: without entering into details, it can be shown that,

given $S^t$, the posterior for $(\boldsymbol{\nu}_j, \boldsymbol{\Sigma}_j)$ still follows a Normal-Inverse Wishart distribution; each element of the diagonal of $\boldsymbol{\Psi}_j$ has a posterior inverse Gamma distribution, and the posteriors for $\boldsymbol{\mu}_j$ and for each row of $\boldsymbol{\Lambda}_j$ now follow a t-Student distribution, that under the reasonable assumption of a large value of $n_\Psi$ can be approximated by a multivariate Normal distribution. Picking the modes of these distributions, for the set of active experts, maximizes $Q(\Theta, \hat{\Theta}^t)$:

$$\hat{\boldsymbol{\nu}}_j^{t+1} = \frac{\boldsymbol{S}_{hzj}^t + n_\nu \boldsymbol{\nu}_{0j}}{S_{hj}^t + n_\nu} , \tag{11}$$

$$\hat{\boldsymbol{\Sigma}}_j^{t+1} = \frac{\boldsymbol{S}_{hzzj}^t - (S_{hj}^t + n_\nu)\hat{\boldsymbol{\nu}}_j \hat{\boldsymbol{\nu}}_j^T + n_\sigma \hat{\sigma} \boldsymbol{I} + n_\nu \boldsymbol{\nu}_{0j} \boldsymbol{\nu}_{0j}^T}{S_{hj}^t + n_\Sigma + d + 2}, \tag{12}$$

$$\begin{aligned} \hat{\boldsymbol{\Lambda}}_j^{t+1} = & \ (\boldsymbol{S}_{hxzj}^t S_{hj}^t - \boldsymbol{S}_{hxj}^t \boldsymbol{S}_{hzj}^{t\,T}) \cdot \\ & \cdot (n_\Lambda S_{hj}^t \boldsymbol{I} + \boldsymbol{S}_{hzzj}^t S_{hj}^t - \boldsymbol{S}_{hzj}^t \boldsymbol{S}_{hzj}^{t\,T})^{-1} , \end{aligned} \tag{13}$$

$$\hat{\boldsymbol{\mu}}_j^{t+1} = \frac{\boldsymbol{S}_{hxj}^t}{S_{hj}^t} + \hat{\boldsymbol{\Lambda}}_j (\hat{\boldsymbol{\nu}}_j - \frac{\boldsymbol{S}_{hzj}^t}{S_{hj}^t}) \quad \text{and} \tag{14}$$

$$\hat{\boldsymbol{\Psi}}_j^{t+1} = \frac{n_\Psi \boldsymbol{\Psi}_0 + \text{diag}\left\{ \boldsymbol{S}_{hxxj} - \hat{\boldsymbol{\Lambda}}_j \boldsymbol{S}_{hxzj}^T - \hat{\boldsymbol{\mu}}_j \boldsymbol{S}_{hxj}^T \right\}}{n_\Psi + S_{hj} + 2} , \tag{15}$$

where diag$\{\cdot\}$ denotes a diagonal matrix equal to the diagonal of its argument. For notational convenience we omitted the iteration step $t+1$ for the parameter estimates $\hat{\boldsymbol{\nu}}_j$, $\hat{\boldsymbol{\Lambda}}_j$, $\hat{\boldsymbol{\mu}}_j$ and $\hat{\sigma}$ appearing in the right hand of the above formulas.

Unfortunately, no analytical expression exists for the posterior distribution of the common scale prior $\sigma$, and so we must obtain the partial derivatives of the Q-function with respect to $\sigma$ and equate them to zero, getting

$$\hat{\sigma}^{t+1} = \frac{A + \sqrt{A^2 + 2\frac{n_\sigma}{n_\Sigma}\sigma_0 B}}{B} , \tag{16}$$

where

$$A = \frac{M^{t+1} d}{2} - \frac{n_\sigma + 1}{n_\Sigma} \quad \text{and} \quad B = \sum_j \text{tr}\{(\hat{\boldsymbol{\Sigma}}_j^{t+1})^{-1}\} ;$$

$M^t$ is the effective number of experts contributing to the mixture at iteration $t$, *i.e.*, $M = \sum_{j=1}^{\infty} \hat{m}_j$, and the sum in the above equation is over the current set of active experts.

Solving for $\hat{m}_j^{t+1}$ that maximizes 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.

### C. Online Learning

The EM training for the infinite mixture of linear models presented in the previous section is a batch algorithm, unsuitable for real-time robotic applications. It can however easily be turned into an online, incremental learning algorithm if only a partial E-Step is implemented, updating, at iteration $t+1$, the sufficient statistics using solely the most recently acquired data point $i$. We will closely follow the results recently proposed in [15], where it is suggested that the sums in (10) be incrementally updating using the rule

$$\boldsymbol{S}^{t+1} = \boldsymbol{S}^t + \gamma_{t+1}(\boldsymbol{s}^{t+1} - \boldsymbol{S}^t) , \tag{17}$$

where $\boldsymbol{s}^{t+1}$ is the value of the sufficient statistics for the most recent point $(\boldsymbol{x}_i, \boldsymbol{z}_i)$ and with $\gamma_t$ being 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 is presented to the algorithm. Choosing $\alpha = 1$ results in an accumulation of the sufficient statistics with no forgetting over time and is equivalent to the generalized EM algorithm described in [14].

### D. Allocation of New Experts

Choosing the appropriate number of components for the mixture is a difficult problem and several methods have been proposed to deal with it: a Bayesian approach, for instance, typically assigns a Dirichlet process prior on the mixing proportions of the infinite mixture, responsible for the automatic generation of the correct number of components [17]. Training usually resorts to offline, computational expensive Markov Chain Monte Carlo sampling methods or to variational Bayesian inference algorithms.

In this paper we take advantage of the fact that changing, at iteration $t$, the value of a particular parameter $\hat{m}_j$ from 0 to 1 will always increase the Q-function. Choosing the values $\hat{m}_j$ that maximize the Q-function is intractable, but relaxing this maximization requirement results in a generalized EM method, where the M-Step is modified to an update that improves the Q-function, without necessarily maximizing it [16]. This allows us to derive a broad class of criteria to decide when to activate a particular expert $j$, ranging from only allowing the existence of a single expert, equivalent to performing a global linear regression on the data, to the activation of a new expert for each new data point processed, which would correspond to some kind of Lazy Learning approach, where predictions are made resorting to all available data $(\boldsymbol{X}, \boldsymbol{Z})$. In online, incremental algorithms, however, a more sensible approach is to allocate new experts only when an acquired data point is poorly explained by the current probabilistic model. LWPR, for instance, achieves this by creating a new linear model each time a new input data point $z_i$ fails to activate the nearest receptive field by more than a given threshold. Of course, when dealing with multi-valued functions such activation scheme must take into account both the input and output part of the data point.

IMLE criterion for activating new experts starts by considering a training point $(\boldsymbol{z}_i, \boldsymbol{x}_i)$ to be an outlier if the posterior probability for class $w_0$ is dominant over the experts posterior probabilities, *i.e.*, if $p(w_0|\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{S}^t) > 0.5$. This quantity is given by

$$p(w_0|\boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{S}^t) = \frac{p(\boldsymbol{x}_i, \boldsymbol{z}_i|w_0)}{\sum_j p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{w}_j, \boldsymbol{S}^t) + p(\boldsymbol{x}_i, \boldsymbol{z}_i|w_0)} ,$$

with $p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{w}_j, \boldsymbol{S}^t) = p(\boldsymbol{x}_i|\boldsymbol{z}_i, w_{ij}, \boldsymbol{S}^t) p(\boldsymbol{z}_i|\boldsymbol{w}_j, \boldsymbol{S}^t)$, and so this condition is equivalent to $\sum_j p(\boldsymbol{x}_i, \boldsymbol{z}_i|\boldsymbol{w}_j, \boldsymbol{S}^t) <$

$p(\boldsymbol{x}_i, \boldsymbol{z}_i|w_0)$. We consider the probability of observing two consecutive outliers to be very low: when this happens we assume it to be caused by a lack of fit of the mixture to the current training points, and a new expert $j$ is then activated, by making $\hat{m}_j = 1$.

### E. Computational Complexity

For a novel 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; (2) assigning responsibilities $h_{ij}$ to active experts according to (8); (3) updating the sufficient statistics for each expert; and (4) obtaining the new values for mixture parameters presented in $\hat{\Theta}$. If the Sherman-Morrison formula is used to update the matrix inversions involved in the final steps, the learning algorithm is $\mathcal{O}(Md(d+D))$, *i.e.*, linear in the number of active experts $M$ and output dimensions and quadratic in the number of input dimensions, thus 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.

## IV. Prediction

### A. Forward Prediction

The conditional density of $\boldsymbol{x}$ for a given query point $\boldsymbol{z}_q$, at any iteration $t$ of the learning process, is

$$p(\boldsymbol{x}|\boldsymbol{z}_q) = \sum_j w_j^x(\boldsymbol{z}_q) p(\boldsymbol{x}|\boldsymbol{z}_q, w_{qj}) \ , \quad \text{with} \tag{18}$$

$$w_j^x(\boldsymbol{z}_q) \equiv p(w_{qj}|\boldsymbol{z}_q) = \frac{p(\boldsymbol{z}_q|w_{qj})}{\sum_k p(\boldsymbol{z}_q|w_{qk})} \tag{19}$$

and where the sums are performed over the active experts.

This conditional density 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. Without entering into details, it can be shown that $p(\boldsymbol{x}|\boldsymbol{z}_q, w_{qj})$ is approximately Normal, with mean and variance given by $\hat{\boldsymbol{x}}_j \equiv \hat{\boldsymbol{\Lambda}}_j(\boldsymbol{z}_q - \hat{\boldsymbol{\nu}}_j) + \hat{\boldsymbol{\mu}}_j$ and $\boldsymbol{R}_j^x \equiv [1 + \gamma_j(\boldsymbol{z}_q)]\hat{\boldsymbol{\Psi}}_j$, where $\gamma_j(\boldsymbol{z}_q)$ is a factor that accounts for expert j prediction uncertainty at location $\boldsymbol{z}_q$.

Single-valued prediction, together with an associated uncertainty, can be obtained from this density by simply taking its mean and variance — this is, for instance, the approach followed by LWPR — but for multi-valued prediction another mechanism has to be devised.

Assuming for the moment the existence of $N_{sol}$ different solutions for query $\boldsymbol{z}_q$ and dropping the dependence on this query for notational convenience, we propose, as a Bayesian model relating experts predictions to multi-valued solutions, that each expert prediction $\hat{\boldsymbol{x}}_j$ is generated from solution $k$, $1 \leq k \leq N_{sol}$, according to

$$\hat{\boldsymbol{x}}_j|s_{jk} \sim \mathcal{N}(\bar{\boldsymbol{x}}_k, \boldsymbol{R}_j \equiv \boldsymbol{R}_j^x / w_j^x) \ , \tag{20}$$

where $s_{jk}$ is a latent indicator variable that signals if expert $\hat{\boldsymbol{x}}_j$ was produced by solution $k$ and $\bar{\boldsymbol{x}}_k$ is the unknown $k^{th}$ solution to be found. The rationale for the definition of $\boldsymbol{R}_j$ follows from the traditional probabilistic view of weighted least squares and best linear unbiased estimators [18], [5], where we incorporate each expert weight $w_j^x$ in the respective predictor variance. Finding $\bar{\boldsymbol{x}}_k$ is the classical problem of clustering $M$ observations $\hat{\boldsymbol{x}}_j$ into $N_{sol}$ classes, where each observation has a different variance. A (possible non-optimal) solution for this problem is given by the EM algorithm, for which the following iterations can easily be found:

$$h_{jk}^t = E[s_{jk}|\hat{\boldsymbol{x}}_j, \hat{\bar{\boldsymbol{x}}}_k^t] = \frac{p(\hat{\boldsymbol{x}}_j|s_{jk}, \hat{\bar{\boldsymbol{x}}}_k^t)}{\sum_l p(\hat{\boldsymbol{x}}_j|s_{jl}, \hat{\bar{\boldsymbol{x}}}_l^t)} \quad \text{(E-Step)}, \tag{21}$$

$$\hat{\bar{\boldsymbol{x}}}_k^{t+1} = (\sum_j h_{jk}^t \boldsymbol{R}_j^{-1})^{-1} (\sum_j h_{jk}^t \boldsymbol{R}_j^{-1} \hat{\boldsymbol{x}}_j) \quad \text{(M-Step)}. \tag{22}$$

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 solution $k$, where the sums are over experts assigned to this solution:

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

A good indicator for the validity of the set of solutions found by the previous step is the dispersion of experts estimates around the corresponding solution $\hat{\bar{\boldsymbol{x}}}_k$: large deviations from this value indicate the need to increase the value of $N_{sol}$. Under the hypothesis that each $\hat{\boldsymbol{x}}_j$ assigned to solution $k$ was effectively generated according to (20), the statistic $T_k$ follows a chi-squared distribution for every solution $k$,

$$T_k = \sum_j (\hat{\boldsymbol{x}}_j - \hat{\bar{\boldsymbol{x}}}_k)^T \boldsymbol{R}_j^{-1} (\hat{\boldsymbol{x}}_j - \hat{\bar{\boldsymbol{x}}}_k) \quad \sim \chi^2_{(M_k - 1)D} \ , \tag{23}$$

where the sums are over experts assigned to solution $k$ and $M_k$ is the number of experts belonging to that solution. If the p-value for any solution $k$ is lower than a given significance level $\alpha_{multi}$, the current set of solutions $\hat{\bar{\boldsymbol{x}}}_k$ is considered to be badly explained by the data.

Forward prediction starts with the single-valued prediction: $N_{sol}$ is then successively increased, repeating steps (21-22), until the above hypothesis fails to be rejected for any of the obtained solutions.

### B. Inverse Prediction

Defining $\boldsymbol{y} = [\boldsymbol{z}^T, \ \boldsymbol{x}^T]^T$ to be the joint input-output vector, an inverse prediction can be obtained, for each expert, from its distribution $p(\boldsymbol{y})$, given by

$$p(\boldsymbol{y}|w_{qj}) = p(\boldsymbol{x}|\boldsymbol{z}, w_{qj}) p(\boldsymbol{z}|w_{qj}) = \mathcal{N}(\bar{\boldsymbol{y}}_j, \boldsymbol{R}_j^y) \ ,$$

where $\bar{\boldsymbol{y}}_j = [\boldsymbol{\nu}_j^T, \ \boldsymbol{\mu}_j^T]^T$ and

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

From the above expression it results that $p(\boldsymbol{z}|\boldsymbol{x}_q, w_{qj})$ follows a Normal distribution with mean and variance given

by

$$\hat{\boldsymbol{z}}_j(\boldsymbol{x}_q) = \hat{\boldsymbol{\nu}}_j + \boldsymbol{R}_j^z \, \hat{\boldsymbol{\Lambda}}_j^T \hat{\boldsymbol{\Psi}}_j^{-1} (\boldsymbol{x}_q - \hat{\boldsymbol{\mu}}_j) \quad \text{and} \tag{24}$$

$$\boldsymbol{R}_j^z = (\hat{\boldsymbol{\Sigma}}_j^{-1} + \hat{\boldsymbol{\Lambda}}_j^T \hat{\boldsymbol{\Psi}}_j^{-1} \hat{\boldsymbol{\Lambda}}_j)^{-1} \; ; \tag{25}$$

this allow us to define a Bayesian model similar to the one presented for forward prediction, where now we have $\hat{\boldsymbol{z}}_j$, given by (24), instead of $\hat{\boldsymbol{x}}_j$, following a Normal distribution whose unknown mean $\bar{\boldsymbol{z}}_k$ is the $k^{th}$ solution to be predicted, and where we have now $\boldsymbol{R}_j \equiv \boldsymbol{R}_j^z / w_j^z$: the prediction procedure is the same as in the forward prediction situation and will be omitted here for brevity.

## V. Experimental Results

In this section we evaluate IMLE in several different experimental settings, comparing its performance to LWPR whenever possible, as this is probably the most widely used state-of-the-art online learning method for robotic applications. LWPR implementation in C++ is freely available for download, and the most recent version to this date was used in our comparisons (Version 1.2.3). Unless otherwise stated, we keep the default values for LWPR parameters provided in the example included in their code for the approximation of the cross function described in the following text, setting $\alpha_{init} = 250$ and $w_{gen} = 0.2$. $D_{init}$, the initial receptive field size, is a parameter that greatly affects the performance and the number of receptive fields created and will be set separately for each experience. We also adjust LWPR to update a full metric $D$, setting the parameter $diagOnly$ to false.

IMLE has 9 parameters that can be tuned to change the resulting behaviour of the algorithm: some of them typically don't need any tweaking and the following experiences, unless otherwise noted, will kept them with their default values, namely: $n_\Lambda = 0.1$, $n_\sigma = 1$, $n_\nu = 0$, $\alpha = 0.99$ and $n_\Psi = \infty$. $n_\Sigma$ will usually be set to a value equal to $2^d$, providing a regularization of the input covariance matrices $\boldsymbol{\Sigma}_j$. The remaining parameters, $\boldsymbol{\Psi}_0$, $\sigma_0$ and $p_0$, have a strong influence on the outlier model, and ultimately on the performance and the number of local experts created during the training phase[1].

### A. Multi-valued Function Approximation:

The toy example presented in Figure 1 is an example of a multi-valued function to be learned, consisting of a multimap comprising two branches, $f_1(z) = cos(z)$ and $f_2(z) = 4 + cos(z)$, for $0 < z < 4\pi$. Standard function approximation learning methods, like LWPR, GPR or SVM, will typically fail here, since they all assume the single-valued model with noise, $x_i = f(z_i) + \epsilon_i$. Motivated by the typical sequential nature of robotic data acquisition, training data was generated by alternating sequential sweeps over each of the two branches, and each output $x_i$ was corrupted with Gaussian noise with standard deviation equal to 0.1 — generating instead training data from random input locations

[1]Source code in C++ for the IMLE algorithm is publicly available at `http://users.isr.ist.utl.pt/~bdamas/IMLE`.

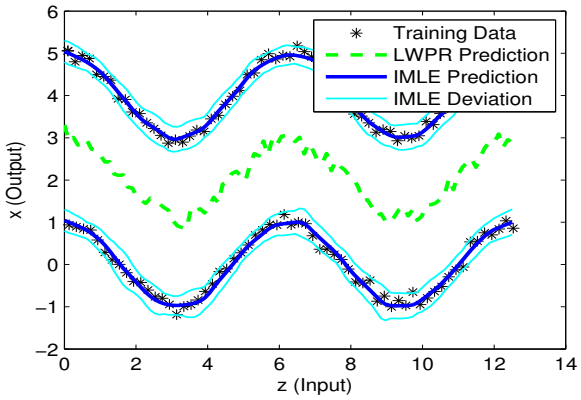


Fig. 1. Some training data for the multi-valued function being learned, together with estimates provided by IMLE and LWPR. Confidence intervals for IMLE, shown in the figure, correspond to one standard deviation.

did not significantly change the experimental results. Figure 1 depicts a sample of the training data corresponding to a single sweep of the multi-function, together with estimates from both IMLE and LWPR, taken after a training period of 10,000 points. As expected, LWPR behaves poorly, producing an estimate that is approximately an average of the two different multi-function branches, together with a very large prediction variance, not depicted in the figure. Multi-valued function approximation algorithms like IMLE, on the other hand, correctly identify the two different solutions for each input value. In this experiment LWPR created 123 different linear models during the training phase, while IMLE only allocated 30 experts.

### B. Single-valued Function Approximation:

We ran LWPR on sequential data taken from the cross function suggested in [5], displayed in Figure 2. This function (Cross 2D) has a two-dimensional input and a univariate output. We adopted all the suggested values for LWPR

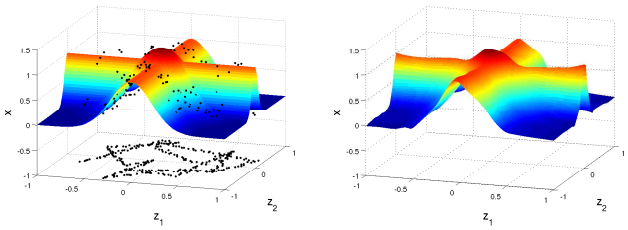

Fig. 2. The cross 2D function. In the left image the target function is depicted, together with some training points and their corresponding projections on the z-plane, in order to enhance the trajectory nature of data acquisition. To the right is the reconstructed function using IMLE.

parameters presented in the Cross 2D example bundled with LWPR code, namely $D_{init} = 50.0$. As for IMLE, we chose $\sigma_0 = 0.02$ to make the initial input covariance matrix guess equal to LWPR. We also defined $p_0 = 0.3$ and $\boldsymbol{\Psi}_0 = 0.01$. Since the output was known to be single-valued, we used $\alpha_{multi} = 1.0$, which resulted in a single-valued solution. The training set consisted of points sampled from a random

trajectory performed in the input space and corresponding output data, for which we added white noise with 0.1 standard deviation. To obtain the learning curves we trained both IMLE and LWPR on a set of 200,000 such points: the resulting normalized root mean square error (nRMSE) and number of models created are shown in Figure 3. Evaluation

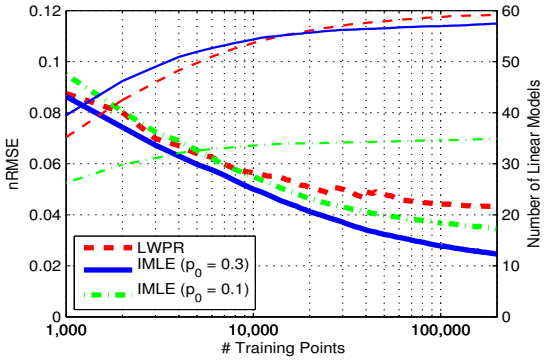


Fig. 3. Learning curves for the Cross 2D function: nRMSE and number of created models as a function of the number of processed training points.

of nRMSE was performed on a 200 by 200 grid like noiseless test set, and to increase the accuracy of the results we ran one hundred different trials with such randomly generated training sets: Figure 3 shows the average nRMSE and number of models over these experiments. IMLE achieves an average nRMSE of less than 0.03, a better result than LWPR, with a value over 0.04. We also present, for illustrative purposes, the learning curves for IMLE when $p_0 = 0.1$: as expected, the average number of experts decreases, and as a consequence the performance of the algorithm is slightly worse in terms of nRMSE.

### C. The PUMA 560 Serial Manipulator:

In this section we use IMLE to learn direct and inverse kinematics for the Unimation PUMA 560, a popular six degrees of freedom industrial robot arm [1]. We use its first three joints to position the end-effector in the workspace, while the remaining three are left untouched since they only control the end-effector orientation. We thus have a kinematics map to be learned that relates the first three joint angles vector $\boldsymbol{z}$ to the 3-dimensional end-effector position $\boldsymbol{x}$. We simulated a random trajectory in the normalized input space comprising one million points to use as the training set, and using Corke's robotic toolbox for MatLab [19] we obtained the corresponding output points. The same procedure was used to obtain the 100,000 points test set. After that, noise with 2cm standard deviation in each position coordinate was added to the output of the training set — that corresponds approximately to 1/100 of the output range.

We trained the IMLE model using $\alpha = 0.99$, $n_\Psi = 500$, $p_0 = 0.01$ and $\sigma_0 = 4$; we also set the significance $\alpha_{multi}$ for the multivalued test to 0.99. Two different runs were made, one using a value for $\boldsymbol{\Psi}_0$ equal to 9 times the noise variance ($\text{IMLE}_\text{A}$) and the other setting this value to 4 times the value of the same noise ($\text{IMLE}_\text{B}$). Results are presented in Table I. We also present the error achieved by LWPR in the same setting: two values of $D_{init}$ were chosen, $D_{init} = 20$ ($\text{LWPR}_\text{A}$) and $D_{init} = 200$ ($\text{LWPR}_\text{B}$) that approximately match the final number of activated linear models of IMLE, while $\alpha_{init}$ was tuned to provide the best error rate. Note how IMLE outperforms LWPR when learning the forward single-valued kinematics.

| | #Models | Forward nRMSE (cm) | | | Inverse nRMSE (%) | | |
|---|---|---|---|---|---|---|---|
| | | x | y | z | $\theta_1$ | $\theta_2$ | $\theta_3$ |
| $\text{LWPR}_\text{A}$ | 200 | 5.36 | 5.01 | 3.56 | — | — | — |
| $\text{LWPR}_\text{B}$ | 475 | 3.76 | 3.68 | 2.73 | — | — | — |
| $\text{IMLE}_\text{A}$ | 151 | 1.44 | 1.39 | 1.03 | 3.43 | 3.73 | 1.43 |
| $\text{IMLE}_\text{B}$ | 487 | 0.83 | 0.86 | 0.73 | 3.08 | 3.47 | 1.26 |

TABLE I
IMLE AND LWPR RESULTS FOR PUMA 560 DATA.

Figure 4 shows the histogram of the number of solutions found by inverse prediction on the test set, comparing it to real number of solutions, obtained by explicitly solving the inverse kinematics equations. The discrepancy between

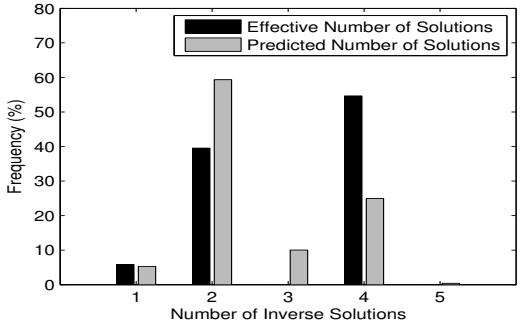


Fig. 4. Frequency of the number of solutions found by IMLE inverse prediction on the test set.

these numbers can be explained by the fact that close to the workspace 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. Increasing $\alpha_{multi}$ would reduce this behaviour; however, due to the curvature of the map to learn, this has the undesired side effect of predictions of neighbour experts being erroneously taken as separate solutions. Choosing a value for $\alpha_{multi}$ is a compromise between this two effects; still, despite this merged solutions phenomena, IMLE still achieves a good inverse prediction error rate, since in the workspace boundary the merged solution provided by IMLE is approximately the average of two reasonable similar true solutions.

### D. A 3-RPR Parallel Robot:

Parallel robots typically exhibit a duality relation to serial chains with respect to the forward and inverse kinematics nature: while the 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. The 3-RPR manipulator described for instance in [2] is an example of such mechanism: 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 end-effector position and orientation on the x-y plane. This mechanism is known to have up to six different solutions for the same actuator configuration [2], which makes learning its forward kinematics unfeasible for most state-of-the-art nonlinear regression techniques like GPR or LWPR.

We trained IMLE with a sequence of one million points taken from a simulated random trajectory in the output space of this parallel robot: the input vector $\boldsymbol{z}$ consisted of the three actuated link lengths, while angle $\theta$ and x-y coordinates of the moving platform composed the output vector $\boldsymbol{x}$. Movement was restricted to a square of 40cm by 40cm in the center of the mechanism, while the angle was constrained to the interval $[-\pi/2; \pi/2]$. We added Gaussian noise to the outputs, again with standard deviation equal to 1/100 of the output range. IMLE parameters where the same as the ones used in the previous experiment, with exception of $\sigma_0 = 1.0$; $\boldsymbol{\Psi}_0$ was set to 4 and 9 times the true noise variance, as before. After the training phase took place, a random sequence of 100,000 test points was presented to the algorithm: forward and inverse prediction errors are presented in Table II, where, to calculate both forward and inverse prediction errors, we picked the predicted solution closest to the previous observed sample. IMLE found on average 1.8 forward solutions per test point, which, given the constrained workspace, agrees with the expected number of solutions; inverse prediction only found a single solution for every point on the test set, as expected.

| | #Models | Forward nRMSE (%) | | | Inverse nRMSE (cm) | | |
|---|---|---|---|---|---|---|---|
| | | x | y | $\theta$ | $L_1$ | $L_2$ | $L_3$ |
| $\text{IMLE}_\text{A}$ | 92 | 1.12 | 1.21 | 2.31 | 0.30 | 0.30 | 0.31 |
| $\text{IMLE}_\text{B}$ | 137 | 1.07 | 1.05 | 1.86 | 0.26 | 0.25 | 0.26 |

TABLE II
IMLE AND LWPR RESULTS FOR 3-RPR PARALLEL ROBOT DATA.

## VI. DISCUSSION AND CONCLUDING REMARKS

The Infinite Mixture of Linear Models presented in this text is, at its core, an algorithm that can efficiently deal with nonlinear function approximation in an online, incremental manner, with performance comparable to current state-of-the-art online learning methods. However, where IMLE performance really shines is in its ability to deal with multi-valued estimates for the same query data. The applications for such kind of problems range from simultaneously learning forward and inverse models of serial and parallel robots to learning switching models, where the function to be approximated can alternate between different configurations over time. To our knowledge this is a distinctive property of the algorithm presented in this paper.

Further directions of research include the choice of different priors for the regression coefficients in $\boldsymbol{\Lambda}_j$: it is well known that when higher dimensional input spaces are considered an increase on the variance of the estimator for these coefficients is observed, degrading the learning performance. In these situations LWPR regularization via PLS may give this algorithm an edge when comparing performances for single-valued forward prediction. Different kind of priors have been suggested in the literature to cope with this problem, and we intend to study how to integrate them in the IMLE model without increasing the algorithm computational complexity.

## REFERENCES


[1] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] J. Merlet, *Parallel robots*. Springer-Verlag New York Inc, 2006.

[3] J. Peters and S. Schaal, "Learning Operational Space Control," *Robotics: Science and Systems (RSS 2006)*, 2006.

[4] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 380–385.

[5] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental Online Learning in High Dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[6] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2001, pp. 298–303.

[7] Z. Ghahramani and M. Jordan, "Supervised Learning from Incomplete Data via an EM approach," *Advances in Neural Information Processing Systems 6*, 1994.

[8] M. Lopes and B. Damas, "A learning framework for generic sensory-motor maps," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*. IEEE, 2007, pp. 1533–1538.

[9] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[10] M. Sato and S. Ishii, "On-line EM Algorithm for the Normalized Gaussian Network," *Neural Computation*, vol. 12, no. 2, pp. 407–432, 2000.

[11] D. Grollman and O. Jenkins, "Incremental learning of subtasks from unsegmented demonstration," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 261–266.

[12] C. Qin and M. Carreira-Perpinán, "Trajectory inverse kinematics by conditional density modes," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1979–1986.

[13] L. Xu, M. Jordan, and G. Hinton, "An Alternative Model for Mixtures of Experts," *Advances in Neural Information Processing Systems*, pp. 633–640, 1995.

[14] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," *Learning in graphical models*, pp. 355–368, 1999.

[15] O. Cappé and E. Moulines, "Online EM Algorithm for Latent Data Models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 71, no. 3, pp. 593–613, 2009.

[16] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977.

[17] E. Meeds and S. Osindero, "An alternative infinite mixture of gaussian process experts," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006, pp. 883–890.

[18] A. Gelman, J. Carlin, H. Stern, and D. Rubin, "Bayesian Data Analysis," *Champan and Hall/CRC*, 2004.

[19] P. Corke, "Matlab toolboxes: robotics and vision for students and teachers," *Robotics & Automation Magazine, IEEE*, vol. 14, no. 4, pp. 16–17, 2007.