Learning Discontinuities with Products-of-Sigmoids for Switching between Local Models

Marc Toussaint MTOUSSAI@INF.ED.AC.UK
Sethu Vijayakumar SETHU.VIJAYAKUMAR@ED.AC.UK
School of Informatics, University of Edinburgh, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK.

Abstract

Sensorimotor data from many interesting physical interactions comprises discontinuities. While existing locally weighted learning approaches aim at learning smooth functions, we propose a model that learns how to switch discontinuously between local models. The local responsibilities, usually represented by Gaussian kernels, are learned by a product of local sigmoidal classifiers that can represent complex shaped and sharply bounded regions. Local models are incrementally added. A locality prior constrains them to learn only local data—which is the key ingredient for incremental learning with local models.

1. Introduction

Locally weighted learning techniques have successfully been employed as incremental, non-parametric approximation schemes for high-dimensional regression problems (Schaal & Atkeson, 1998; Vijayakumar & Schaal, 2000). Their robustness and efficient online versions are crucial in robotic domains where, for instance, an inverse model of an articulated dynamic robot has to be learned in real-time. Such models map a high-dimensional state (e.g., joint angles and velocities) and a desired change of state to the required motor signals (torques).

While typically such mappings are assumed to be smooth, in real world scenarios, there are many interesting cases where the functions of interest are truly discontinuous. Some examples include contact with other objects (and the ground), with other parts of the body, or with "joint limits". In fact, many interesting interactions with the environment manifest themselves through discontinuities in the sensorimotor data.

Switching between models can be modeled by introducing a latent switch variable i that indicates which model is responsible. Several existing methods address a scenario where i can be inferred from the temporal context by learning a latent temporal transition model $P(i'|i)$, e.g., in the context of state space models (Ghahramani & Hinton, 1998; Pavlovic et al., 2000) or multiple inverse models (Wolpert & Kawato, 1998). Here we want to address another scenario, where i can be inferred directly from the input $\boldsymbol{x}$ (e.g., the robot's configuration or sensor readings) by learning a probabilistic model $P(i|\boldsymbol{x})$. Learning a *deterministic* partitioning of the input space $\boldsymbol{x} \mapsto i$ has been addressed before: (Bemporad et al., 2003) use a greedy search algorithm to find a polygonal partitioning constrained by an absolute error bound. (Kavka & Schoenauer, 2004) recently proposed a Voronoi cell partitioning with local fuzzy controllers learned with an Evolutionary Algorithm. Also decision trees (Model Trees, (Quinlan, 1992)) are possible choices to represent a partitioning. None of these approaches provide a probabilistic model for $P(i|\boldsymbol{x})$ or for the local regression models (e.g., to account for outliers) and none of them provides an incremental learning scheme, where $P(i|\boldsymbol{x})$ (or the deterministic partitioning) can be adapted locally without interference effects.

Our approach will introduce a product-of-sigmoids model of $P(i|\boldsymbol{x})$ where the boundary between two neighboring models can be adapted independently from all others. A standard EM-algorithm is derived from a probabilistic regression model which includes a background noise (outlier) model as a mixture. Unlike standard mixture models, we introduce a locality prior which prevents non-neighboring local models to 'compete for data'—this greatly reduces interference in the learning process and is one of the key features of local learning techniques that allows for incremental learning (Schaal & Atkeson, 1998). In that respect, we follow the approach of locally weighted learn-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005. Copyright 2005 by the author(s)/owner(s).

ing (Vijayakumar & Schaal, 2000), where previously a Gaussian kernel associated with each local model represented its responsibility (although it is used as an averaging weight rather than a probabilistic mixture coefficient). Our key extension to these approaches is that, using a product of local sigmoidal classifiers as a model for the responsibilities $P(i|\boldsymbol{x})$ instead of the typically Gaussian kernel, we can learn complex shaped, sharply bounded responsibility regions for each local model.

The next section will give a brief overview of the model while Sections 3 and 4 introduce the details of learning a family of local models and learning the responsibilities. Section 5 briefly addresses the complexity of the model. In Section 6, we demonstrate the performance of the algorithm on test functions and a physical interaction problem.

2. Overview of the model

The complete model will be composed of n submodels ϕ_i, $i = 1, .., n$, and a set of sigmoidal functions $\psi_{ij}(x) \in [0, 1]$ that switch between two neighboring submodels i and j (see Fig. 1). The "cell" associated to a model i is bounded by the adjacent sigmoids ψ_{ij}, similar to the Voronoi cells of (Kavka & Schoenauer, 2004). However, adapting Voronoi sites incrementally to design appropriate boundaries is a complex coupled problem. Therefore, we represent every boundary by a separate sigmoid ψ_{ij}; hence, incorporating the ability to represent even non-convex polygonal regions.

Given this setup, we will have to: (1) learn the set (or *family*) of submodels ϕ_i such that it is sufficient to explain the data, and (2) learn the sigmoidal switching functions ψ_{ij} between neighboring models. Following the policy to reduce interference, we decouple the learning process accordingly on two levels. On the first level (*family learning*) the goal is to learn a set of submodels such that for every datum at least one of them is a sufficient model. Family learning will assume a simple, uninformed locality prior $P_l(i|\boldsymbol{x})$ to formulate a generative mixture model and derive the EM-updates. The second level uses the learned family of submodels to learn an accurate predictive model $P_s(i|\boldsymbol{x})$ with the product-of-sigmoids approach. Separating these two levels prevents interference problems—typical for mixture of experts models (Jordan & Jacobs, 1994)—that arise when using a yet incorrect (gating) model $P_s(i|\boldsymbol{x})$ for the responsibility inference in the E-step: If initially the learned model $P_s(i|\boldsymbol{x})$ is incorrect, the inferred responsibilities are corrupted and the submodels ϕ_i are assigned hard to learn data. No stable set of submodels might develop and based on this a better model $P_s(i|\boldsymbol{x})$ could hardly be learned.

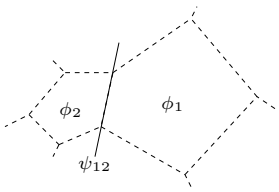

Figure 1. The complete model is composed of local models ϕ_i and classifiers ψ_{ij} between neighboring models (e.g., the solid line indicates a learned linear classification). Every local model receives responsibility in a region bounded by adjacent classifiers (dotted lines).

3. Learning a family of models

The first level goal of our algorithm is to learn a *family* $\Phi = \{\phi_1, .., \phi_N\}$ of models such that every datum can be explained by at least one model. Thinking in terms of a generative model, this can be captured as follows. First, an input $\boldsymbol{x}$ is drawn from some distribution $P(\boldsymbol{x})$. Then, with a probability ϵ, the output y is pure noise modeled by the uniform distribution $\mathcal{U}(y)$. Otherwise, with a probability $1-\epsilon$, the ith model of the family is selected according to a prior $P_l(i|\boldsymbol{x})$ and produces an output $P(y|i, \boldsymbol{x})$. Formally,

$$P(y|\boldsymbol{x}) = (1-\epsilon) \sum_{i=1}^{N} P_l(i|\boldsymbol{x})\ P(y|i, \boldsymbol{x}) + \epsilon\ P(y|i=0)\ ,$$

$$P(y|i, \boldsymbol{x}) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left\{-\frac{|y-\phi_i(\boldsymbol{x})|^2}{2\,\sigma^2}\right\}\ ,$$

$$P(y|i=0) = \mathcal{U}(y)\ .$$

The latent variable i, which we will call the *responsibility index*, thus decides on which model produced the data. Here, $i = 0$ corresponds to the case of the noise model.

On the level of family learning we assume a *locality prior* $P_l(i|\boldsymbol{x})$, describing which model i may possibly be responsible for a given input $\boldsymbol{x}$. This prior is rather uninformed and is not learned from data—the next section will exactly focus on replacing this prior by a more predictive model $P_s(i|\boldsymbol{x})$ learned from data. The locality prior implements a constraint such that the same model ϕ_i cannot be responsible for remote, disconnected regions of the input domain. This clearly follows the principle of localized learning, aiming to reduce interference, and is also a necessary constraint for the switching model explained later.

Let $\boldsymbol{c}_i$ denote the mean input (*center*) on which model ϕ_i has been trained on. For a given input $\boldsymbol{x}$, the ith

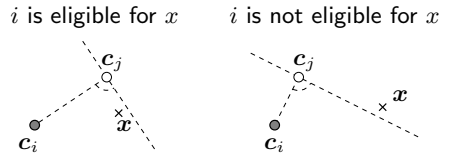

Figure 2. The constraint implemented by the locality prior: A model with center $\boldsymbol{c}_i$ is eligible for an input $\boldsymbol{x}$ only when there does not exist another model with center $\boldsymbol{c}_j$ "between" $\boldsymbol{c}_i$ and $\boldsymbol{x}$. A notion of "between" is that the scalar product in equation (1) is negative (which is equivalent to saying $\boldsymbol{c}_j$ is not in the ball between $\boldsymbol{x}$ and $\boldsymbol{c}_i$).

model is eligible (i.e., a candidate for explaining the data) if and only if there does not exist a jth model which has its center "between" $\boldsymbol{c}_i$ and $\boldsymbol{x}$. More precisely,

$$P_l(i|\boldsymbol{x}) = 0 \iff \exists j : \langle \boldsymbol{x} - \boldsymbol{c}_j, \boldsymbol{c}_i - \boldsymbol{c}_j \rangle < 0 \ , \quad (1)$$

where $\langle \cdot, \cdot \rangle$ is the scalar product in input space. This constraint is best explained by Fig. 2. Equal probability is then associated to all eligible models, i.e., $P_l(i|\boldsymbol{x})$ is uniform over all eligible i's. It would be straight-forward to introduce smoother locality constraints, but since this prior worked sufficiently and is parameter-free, we stick to this simple option.

Having setup this model, learning follows the standard EM-machinery. For the E-step, given a current family of models and a new training datum $(\boldsymbol{x}, y)$, we can infer a posterior on the latent index i for this datum using Bayes rule:

$$\forall_{i \neq 0} : \ P(i|y, \boldsymbol{x}) = \frac{1-\epsilon}{Z} \ P(y|i, \boldsymbol{x}) \ P_l(i|\boldsymbol{x}), \quad (2)$$

$$P(i = 0|y, \boldsymbol{x}) = \frac{\epsilon}{Z} \ P(y|i = 0) \ , \quad (3)$$

where Z normalizes over i (including $i = 0$). As an M-step, model ϕ_i is then trained on the datum weighted by the $P(i|y, \boldsymbol{x})$ (we give more details on the local models below). It becomes clear how the locality prior in equation (2) reduces interference: a model i that is non-local to $\boldsymbol{x}$ has a weighting zero, is not trained, and does not participate in the competition to model the data.

New models are added incrementally to the family when needed: We use the MAP index

$$\hat{i} = \text{argmax}_i \ P(i|y, \boldsymbol{x}) \quad (4)$$

to label the training datum with the most likely model. A zero MAP index $\hat{i} = 0$ indicates that the training data is "yet unmodeled". All such data is collected into a set until it has a desired size sufficient for a new hypothesis. The generation of new family members is done, similarly to RANSAC (Fischler & Bolles, 1981), as follows: A datum $(\boldsymbol{x}, y)$ is selected randomly from the set of unmodeled data. Then the K closest neighbors[1] of $(\boldsymbol{x}, y)$ in this batch (w.r.t. Euclidean distance in input space) are determined. These K data points are chosen as initial training data for the new hypothesis. Thereafter, all points in the set of unmodeled data are reconsidered and the MAP index $\hat{i}$ recalculated for them according to equation (2), pretending the hypothesis is a member of the family. If the new hypothesis receives a sufficient count of responsibilities (it basically only has to compete with the noise model) this modeled data is deleted from the batch and the hypothesis added to the family. We considered $10\,d$ as a threshold for sufficient counts of responsibilities. This threshold naturally limits the number of instantiated models depending on the data. Additionally, a pruning of models that do not explain a sufficient amount of data is possible based on how often a model receives the MAP index. We did not use such heuristics though in the experiments.

We also define an error measure, the *family error*, which is the average squared error of the best fitting *eligible* model: For a test data set $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^M$ we define

$$E_f = \sum_{k=1}^{M} |y_k - \phi_{i_k}(\boldsymbol{x}_k)|^2 \ , \quad \text{where}$$
$$i_k = \underset{i}{\text{argmax}} \ P_l(i|\boldsymbol{x}_k) \ P(y_k|i, \boldsymbol{x}_k) \ .$$

In this way, the performance of the underlying family learning can be monitored independently in the training phase.

3.1. The local models

The general scheme of family learning can be realized with any type of models ϕ_i. In the experiments, we will choose ϕ_i to be linear functions, learned with Partial Least Squares (PLS) regression (de Jong, 1993). However, there exist standard techniques to *blend* between local linear functions in order to learn non-linearities (Schaal & Atkeson, 1998). Future version of our model will be extended to be able to do both, blending and switching. Here, we want to focus only on the novel aspect of the new model: the switching.

The sufficient statistics for PLS need to be collected in a weighted fashion: The variables of these statistics are the norm W, the input and output means $\boldsymbol{m}_x$ and m_y, and correlation matrices V and C. All of these

[1] We choose K to be a random Poisson number with mean $3\,d$, where d is the input dimensionality.

are initialized with zero. When the model receives an input $\boldsymbol{x}$, a target output y, and a weighting w ($= P(i|y,\boldsymbol{x})$) as new training data, the sufficient statistics are accumulated as

$$\begin{aligned} W &\leftarrow W + w \ , \\ \boldsymbol{m}_x &\leftarrow \boldsymbol{m}_x + w\,\boldsymbol{x} \ , \qquad & m_y &\leftarrow m_y + w\,y \ , \\ V &\leftarrow V + w\,\boldsymbol{x}\,\boldsymbol{x}^T \ , \qquad & C &\leftarrow C + w\,\boldsymbol{x}\,y \ . \end{aligned}$$

Clearly, the total input mean is $\boldsymbol{m}_x/W$, the output mean is m_y/W, the input covariance matrix $V/W - \boldsymbol{m}_x\,\boldsymbol{m}_x^T/W^2$ and the input-output covariance matrix $C/W - \boldsymbol{m}_x\,m_y/W^2$. Whenever the model has to be evaluated, the SIMPLS algorithm (de Jong, 1993) fits a linear regression to these statistics by computing a projection matrix P that maps $\boldsymbol{x}$ onto a lower-dimensional representation and a matrix Q that maps this representation to the final output. The composition $Q\,P:\ \boldsymbol{x} \mapsto y$ is the learned linear model.

Partial Least Squares can also be realized as an online learning scheme (Vijayakumar & Schaal, 2000) where the statistics are accumulated in the same way, but the projections P are learned by continuous adaptation. The benefit of PLS is that the projections P are learned in such a way that they are most "informative" (most correlated) with the output—which is a very powerful dimensionality reduction method to base the actual regression on. In addition, the multivariate regression breaks down into a series of univariate regressions due to the orthogonality of the projection directions.

4. Products-of-sigmoids for switching

On the second level of our algorithm, the goal is to learn a predictive model $P_s(i|\boldsymbol{x})$ of the latent responsibility index i that is more precise than the locality prior $P_l(i|\boldsymbol{x})$. One may think of exactly the same generative process as described in the previous section except that $P_l(i|\boldsymbol{x})$ is replaced with $P_s(i|\boldsymbol{x})$.

Traditionally, in localized learning one associates a kernel $\alpha_i(\boldsymbol{x})$ (typically Gaussian) to each local model such that $P(i|\boldsymbol{x})$ (or a model's "weighting") is proportional to $\alpha_i(\boldsymbol{x})$, but normalized over i. The shape and size of such kernels can also be learned (Schaal & Atkeson, 1998). Generally though, the shapes of responsibility regions may be complex and sharply bounded. Following the localized learning principle we still want this classification be represented in a localized way.

Our approach is to represent such kernels as a product of sigmoidal classifiers that are arranged locally around a model i. More precisely, given some data it is easy to decide whether two models are "potentially neighbored"—namely whether there exists data for which both models are eligible—based on their centers. For each pair (ij) of neighbored models, we learn a sigmoidal function $\psi_{ij}(\boldsymbol{x})$, where $\psi_{ij} \equiv 1 - \psi_{ji}$. The product of such sigmoids around a submodel i then defines $P_s(i|\boldsymbol{x})$ as

$$P_s(i|\boldsymbol{x}) = \frac{1}{Z'} \prod_j \psi_{ij}(\boldsymbol{x}) \ , \qquad (5)$$

$$\psi_{ij}(\boldsymbol{x}) = \frac{1}{1+\exp[-\eta_{ij}(\boldsymbol{x})]} \ ,$$

where Z' normalizes over i. As indicated, the sigmoids ψ_{ij} are defined by a scalar functions η_{ij} which we will assume to be linear (which makes ψ essentially a standard perceptron). Fig. 3 illustrates some kernels that can be represented as products of sigmoids.

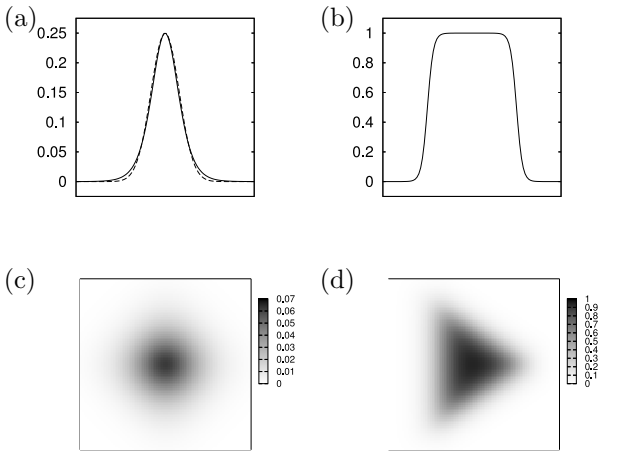

Figure 3. Kernels that can be represented as a product-of-sigmoids. Let $s(x) = 1/[1+\exp(-x)]$. (a): $s(x)\,s(-x)$ is compared with the Gaussian $.25\,\exp(-x^2/5)$ (dashed line). (b): $s(3(x+5))\,s(-3(x-5))$, the steepness of the kernel boundaries can be tuned with the slope of the linear function η_{ij}. (c): The product $s(x)s(-x)s(y)s(-y)$ in 2 dimensions is very smooth and almost radially symmetric. (d): $s(2(x+2))s(2(-x/1.5+y+2))s(2(-x/1.5-y+2))$ is displayed. In principle any polygonal kernel shape with any boundary steepness can be constructed.

The sigmoids $\psi_{ij}(\boldsymbol{x})$ are meant to represent the likelihood that a model i rather than j is responsible for an input $\boldsymbol{x}$, conditioned on the fact that either i or j is responsible. The product combination is comparable to an AND voting. The MAP index $\hat{i}$ we inferred in the previous section is now used to train these sigmoids: For every datum that is labeled with a non-zero MAP index $\hat{i}$, all sigmoids ψ_{ij} adjacent to the ith model are trained to classify this datum as 1 (or, equivalently, the ψ_{ji}'s are trained to classify this datum as 0). The pa-

rameters of the sigmoid (basically the weights of the perceptron) are trained with a fast gradient descent based adaptation (Rprop, (Igel & Hüsken, 2003)).

4.1. Error measures

In addition to the family error, we define the *classification error* E_c, which counts how often the product of sigmoids correctly predicts $\phi_{\hat{i}}$ to be the best fitting model for a given input: Let us define

$$i^* = \operatorname*{argmax}_i P_s(i|\boldsymbol{x}) \tag{6}$$

as the maximum likelihood *predicted* index, as opposed to the MAP index $\hat{i} = \operatorname{argmax}_i P(i|y, \boldsymbol{x})$ given the target output y. Then the classification error gives the ratio of data for which $i^* \neq \hat{i}$.

Our model could be interpreted as a deterministic function (instead of a probabilistic model) where the output y^* is given by the predicted best model, $y^* = \phi_{i^*}(\boldsymbol{x})$. The MSE w.r.t. this function could be used as an error measure for our model. However, this is inconsistent with the probabilistic framework and in the case of discontinuous functions, also misleading. For instance, consider the Heavy-side function $h(x < 0) = 0$ and $h(x \geq 0) = 1$. Very close to the discontinuity our model might predict an output of 0 *or* 1, each with a probability of 50%. Thereby, with a chance of 50% it produces a squared error of 1, or in average $\frac{1}{2}$. In contrast, a model that predicts an output of $\frac{1}{2}$ only produces a squared error of $\frac{1}{4}$ although it predicts consistently wrong. From the probabilistic point of view, it is clear that it makes no sense to assume a (unimodal) Gaussian output noise model at discontinuities. Instead, displaying both, the family and classification error, gives more insight and directly reflects the likelihood of the model.

5. Computational issues

For every training point, an E-step (eq. (2)) has to calculate responsibilities for each model by computing the product of the locality prior $P_l(i|\boldsymbol{x})$ and the model likelihood $P_i(y|\boldsymbol{x})$. Ordinarily, these would have to be evaluated for each model i. However, since the locality prior is non-zero only for a limited number of eligible models, we can reduce the computational cost by maintaining a graph data structure where each node is a local linear model ϕ_i and each edge represents a sigmoid ψ_{ij} between neighboring models. The graph allows us to quickly decide which models are eligible (by finding the closest node and considering its graph neighbors)—the likelihoods $P_i(y|\boldsymbol{x})$ need be evaluated only for those models. Evaluation of a likelihood is $O(d)$ for a linear model in d dimensions.

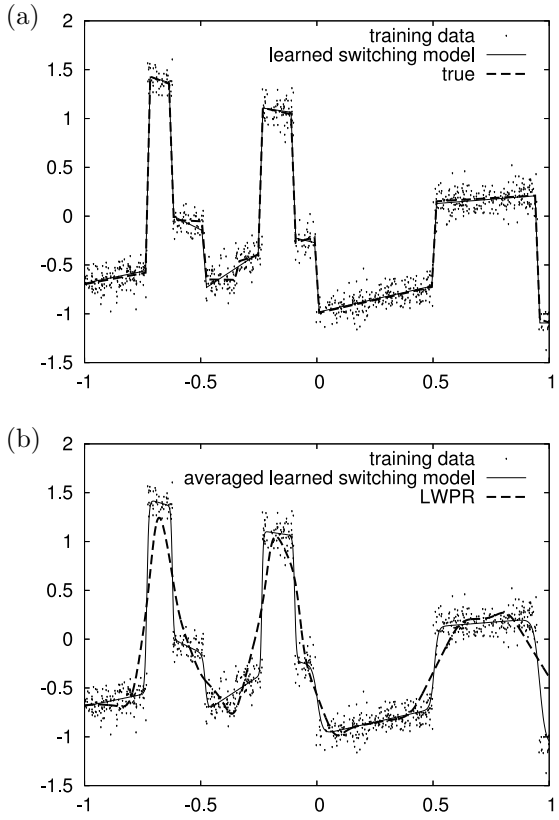

Figure 4. (a) A 1D test function with d=1, l=10, σ=0.1. Learned switching model after 20 iterations on 1000 training data points. (b) The averaged switching model: $y(\boldsymbol{x}) = \sum_i P_s(i|\boldsymbol{x})\,\phi_i(\boldsymbol{x})$ compared to LWPR.

For the M-step (for every training point), the statistics of each of the eligible models has to be updated using incremental PLS weighted with the non-zero responsibility. In (Vijayakumar & Schaal, 2000) details are found on how to realize incremental PLS in $O(d)$ by adapting the number p of projections needed for regression (as a result, the sufficient statistics are $d \times p$ matrices rather than $d \times d$). For every training datum, we also adapt the sigmoids $\psi_{\hat{i}j}$ adjacent to the MAP model $\phi_{\hat{i}}$, which is also $O(d)$ when implemented using gradient based adaptation.

6. Experiments

6.1. Discontinuous test functions

First we tested our algorithm on random, piecewise linear, discontinuous test functions. A test function has 3 parameters: the input dimensionality d, the number l of linear pieces it is composed of, and the output noise σ. They are generated as follows. We draw l

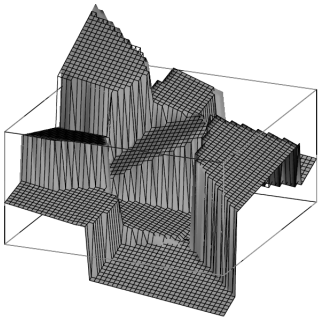

Figure 5. A 2D test function composed of 10 pieces. The wire-frame represents the learned switching model which deviates from the true test function (gray-shaded surface) only at a few corners. The noise level was $\sigma = .01$.

basis points $\hat{\boldsymbol{x}}_i$, $i = 1, .., l$ uniformly from the input domain $[-1, 1]^d$. To each basis point $\hat{\boldsymbol{x}}_i$ we associate a random offset $\beta_i^0 \sim \mathcal{U}([-1, 1])$ and a random vector $\boldsymbol{\beta}_i \sim \mathcal{U}([-1, 1]^n)$, where $\mathcal{U}$ is the uniform distribution. Thus, each i represents a random linear function $\boldsymbol{x} \mapsto \beta_i^0 + \langle \boldsymbol{\beta}_i, \boldsymbol{x} - \hat{\boldsymbol{x}}_i \rangle$. For a given input $\boldsymbol{x}$, the target output is then the output of the ith linear piece with minimal distance $|\boldsymbol{x} - \hat{\boldsymbol{x}}_i|$ plus Gaussian noise with standard deviation σ.

Fig. 4(a) display a 1D test function (dashed) with 10 linear pieces together with the 1000 training data points sampled from it and the approximation (solid) learned by our algorithm (i.e., the most likely predicted output y^*). The fit between the learned model and the true test function is almost perfect, except of the region around $x \approx -.4$, where the true test function has as a step which is not recognized by the model. Given the noise variance of the actual training data though, this is not surprising.

Fig. 4(b) displays the same 1D test function, now compared against a model learned by Locally Weighted Projection Regression (LWPR, (Vijayakumar & Schaal, 2000)) and an "averaged output" of our model, $y(\boldsymbol{x}) = \sum_i P_s(i|\boldsymbol{x})\, \phi_i(\boldsymbol{x})$. As mentioned in the section 4.1, this averaging is probabilistically questionable. Still, displaying this output on the one hand gives a concrete idea of the sigmoids, in particular their slopes, that are implicitly learned, and on the other hand makes it more comparable to the weighted averaging usually employed by local regression models.

Fig. 5 displays a 2D test function showing how the complexity of the kernels (or cells) may increase in higher dimensions.

In Fig. 6 we display results for 10 independent trials on random test functions in 2, 5 and 10 dimensions. The family error (upper graphs) consistently decays

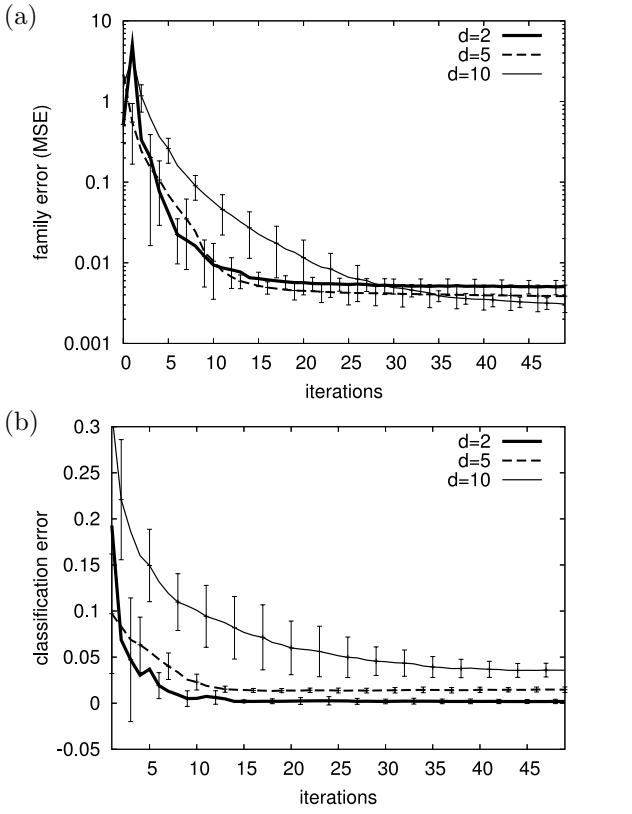

Figure 6. The family error (a) and the classification error (b) for 10 runs on random 2D, 5D and 10D test functions with $l = 10$ and $\sigma = .1$. We used 1000 training point in the 2D case and 10 000 in the 5D and 10D. Each iteration corresponds to one cycle through the training data set. The curves are average errors standard deviations for the 10 runs on independent test data sets.

to the noise level ($\sigma^2 = 0.01$). (A family error below the noise level is possible because the MAP index $\hat{i}$ is a posterior given the target output.) The classification is not perfect but still rather robust as shown by the variance between trials. The 10D case (with a considerably large input space $[-1, 1]^{10}$) indicates that eventually this classification becomes the challenge in high dimensions.

6.2. Physical interaction problem

We use a realistic physical simulation[2] of a movement system with contact dynamics (cf. Fig. 7(a)) to test our algorithm. The configuration has 4 degrees of freedom, an angular position and velocity for both levers. As a control signal, we applied random Gaussian torques on the cylindrical lever. When the cylindrical lever comes in contact and pushes the black lever, contact forces

[2]The *Open Dynamics Engine* (http://ode.org/) written by Russell Smith et al.

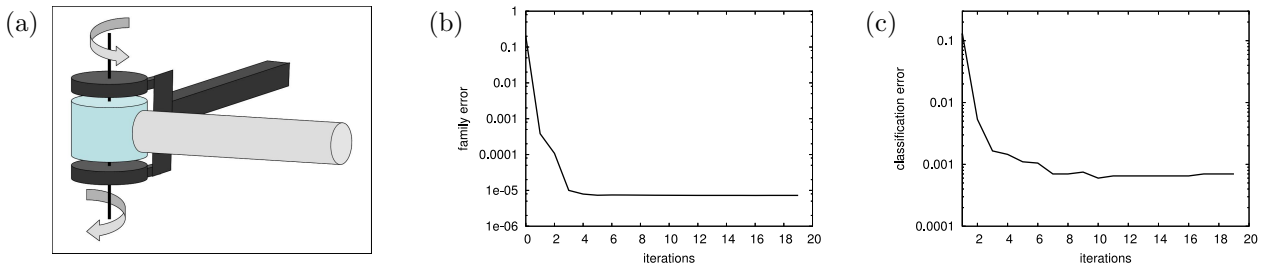

Figure 7. (a) The physical configuration we used to generate the data. The 1D control signal specifies the torque applied on the (light coloured) cylindrical lever which rotates friction-less about the axis. An unactuated (black) lever rotates about the same axis with significant friction. No torques are exerted on the black lever unless the cylinder pushes it. The family error (b) and the classification error (c) for the physical interaction problem.

discontinuously change the dynamics of the whole system.

The problem is to learn the inverse dynamic model which maps from the current state (positions and velocities of both levers) and the desired acceleration (of the cylindrical lever) to the motor signal that is needed to realize this acceleration. 10 000 data points were collected for training. Eventually, the model instantiated three submodels to learn the data. The family error given in Fig. 7 corresponds to the noise-level of the physical simulation; the classification error shows that only very few of the independent 10 000 test points were wrongly associated. More insight is gained by investigating how the sigmoidal classifiers learned to split up the data among the local models. In Fig. 8, the full training data set is displayed, as well as the split up. We find that the data is separated in three parts corresponding to the three submodels. The simplest part is the contact-free data (with non-extremal relative angle, Fig. 8(b)) which is learned as a linear relationship between desired acceleration and motor signal. The two other parts correspond to the contact situations (from the left and from the right). Here, the target motor signal additionally depends on the current velocities (due to the black lever's friction) and two separate models are learned for the cases of left and right contact.

7. Discussion

The presented model addresses the problem of handling the discontinuities that naturally arise, for example, in sensorimotor data during interaction with a structured environment. Our model extends earlier local incremental learning approaches in several ways: The responsibility region associated with each local model (learned with the product-of-sigmoids) has a much more versatile boundary shape compared to typical Gaussian kernels. Problems associated with initialization of kernel shapes or widths and the heuristic choice of an ad hoc number of submodels are circumvented by the robust incremental allocation of new models.

In incremental learning problems, locality is the key to reduce interference since one local model can be adapted to new local data without destroying remote models. The sigmoid-of-products representation of $P(i|\boldsymbol{x})$ follows this principle: A new datum leads to an adaptation of only the local classifiers around $\hat{i}$ and leaves remote classifiers unaffected. This contrasts to previously investigated global approaches (Bemporad et al., 2003; Quinlan, 1992) to learn a deterministic partitioning of the input space.

Learning a family of models was realized on a separate level, decoupled from the learning of the responsibility prediction which is built on top. The fixed locality prior $P_l(i|\boldsymbol{x})$ that we introduced at the level of family learning has a decoupling effect which contrasts to standard mixture models: Since the prior is zero for non-local data, non-neighboring submodels do not compete for modeling this data. For instance, when a new submodel is added to the family, it only interferes with its direct neighbors (which may have to readapt) but does not interfere with remote submodels. This is the key ingredient for incremental learning with local models.

Finally, we assumed the local models to be linear. If one can specify an appropriate set of basis functions (like low order polynomials) for the given problem, then it is straight-forward to extend the approach to local non-linear models following Generalized Additive Models (Hastie & Tibshirani, 1990). Generally, existing approaches addressed the problem of blending (with associated kernels) local linear models in order to represent smooth non-linear function. Here we focused on how to discontinuously switch between linear models. A future version of our model will have to combine

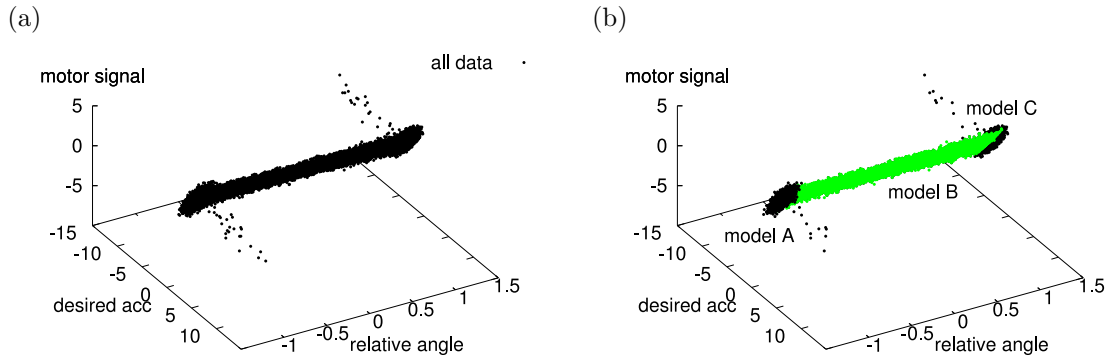

Figure 8. (a) The (normalized) training data collected from the physical simulation projected on the desired acceleration (which is an input), the relative angle between the levers (which is not an input but used here for better visibility), and the motor signal (which is the target output of the inverse model). When the relative angle is extremal (around ± 1 in the given scale) the two levers have contact (from the left or from the right). (b) The separation of the test data set as learned by the products-of-sigmoids. The model learned to associate the test data points to the three different local models ϕ_i. More precisely, the test data is split up according to the maximum likelihood predicted index i^* (cf. eq. (6)), which is computed from the product of the learned sigmoids (cf. eq. (5)). The data labeled with "model B" corresponds to the contact free dynamics and is a linear dependency between desired acceleration and motor signal. The data assigned to models A and C correspond to contact from the left or the right, respectively.

both, blending and switching, in order to represent discontinuous and piece-wise non-linear functions.

Acknowledgments

We would like to thank the anonymous reviewer for their helpful comments. The first author is grateful to the German Research Foundation (DFG) for the Emmy Noether fellowship TO 409/1-1.

References

Bemporad, A., Garulli, A., Paoletti, S., & Vicino, A. (2003). A greedy approach to identification of piecewise affine models. *Hybrid Systems: Computation and Control; LNCS* (pp. 97–112). Springer.

de Jong, S. (1993). SIMPLS: An alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, *18*, 251–263.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, *24*, 381–395.

Ghahramani, Z., & Hinton, G. (1998). Variational learning for switching state-space models. *Neural Computation*, *12*, 963–996.

Hastie, T., & Tibshirani, R. (1990). *Generalized additive models*. Chapman and Hall, New York.

Igel, C., & Hüsken, M. (2003). Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, *50(C)*,, 105–123.

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

Kavka, C., & Schoenauer, M. (2004). Evolution of voronoi-based fuzzy controllers. *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Birmingham, UK; LNCS*. Springer.

Pavlovic, V., Rehg, J. M., & MacCormick, J. (2000). Learning switching linear models of human motion. *NIPS* (pp. 981–987).

Quinlan, J. R. (1992). Learning with Continuous Classes. *5th Australian Joint Conference on Artificial Intelligence* (pp. 343–348).

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

Vijayakumar, S., & Schaal, S. (2000). Locally weighted projection regression: An *o(n)* algorithm for incremental real time learning in high dimensional space. *Proc. Int. Conf. on Machine Learning (ICML)* (pp. 1079–1086).

Wolpert, D., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, *11*, 1317–1329.