Context-GMM: Incremental Learning of Sparse Priors for Gaussian Mixture Regression

Arturo Ribes, Jesús Cerquides Bueno, Yiannis Demiris and Ramón López de Mántaras

***Abstract*— Gaussian Mixture Models have been widely used in robotic control and in sensory anticipation applications. A mixture model is learnt from demonstrations and later used to infer the most likely control signals, or is also used as a forward model to predict the change in sensory signals over time. However, such models often are too big to be tractable in real-time applications. In this paper we introduce the Context-GMM, a method to learn sparse priors over the mixture components. Such priors are stable over large amounts of time and provide a way of selecting very small subsets of mixture components without significant loss in accuracy and with huge computational savings.**

I. INTRODUCTION

In order to perform a variety of tasks in their environment, robots must acquire a model of the consequences of its actions. This is important specially in the developmental robotics field, where the robot has no specific task but it still needs to learn a forward model of its environment that tells how state variables change over time depending on its own actions [1].

As robots become more dexterous and their sensory capabilities are enhanced, the acquired models become more and more complex.

In a need of managing this overwhelming complexity, models need to be compressed or partitioned, so the problem of using them to solve a specific task becomes tractable. This is very important in terms of interactivity of the robot with its environment, as it needs to compute a response in a limited amount of time or it may miss some important event.

Generative models enable the execution of complex behaviours by providing control signals obtained applying probabilistic inference on the distribution extracted from sensor and actuator data.

A Gaussian Mixture Model (GMM) is a kind of generative model which can be viewed as a compressed version of a dataset. In our case, this dataset is the sensorimotor history of a robot or a set of demonstrations of a particular task.

If our GMM models the joint distribution of perception-action tuples, we can use Gaussian Mixture Regression (GMR) to infer the most likely control signal for a given percept. This technique has proven very successful in learning by demonstration tasks [2][3][4][5].

However, as the space dimensionality grows, the resulting model often contains an intractable number of components, so the model cannot be evaluated in real time. This is very important in mobile robotics, where it is often the case that embedded CPUs do not have much computational power.

In robotics, sensorimotor variables provide a stream of data. This means that their values do not change abruptly very often, as they are governed by internal and external dynamics of both the robot and environment. Sensor data will exhibit a certain pattern when the robot is executing a behaviour like walking and another pattern when it is picking up some object. This translates in the activation of different parts of the model, usually very small, when a behaviour is being performed.

This insight leads to the conceptualization of contexts as the set of hidden factors that condition the activation of a small region of the model. Those factors induce sparse prior probabilities over the model components, that is, only a small subset of the model is likely to be activated under such conditions.

Sparse coding has been an active topic in the recent years. It provides representations based in a set of basis features or encoding units, where only a small amount of this units are actually used to code a pattern, hence the term sparsity. An interesting type of sparsity is group sparsity [6], where ones assumes that units in the same group tend to be zero or non-zero simultaneously. We observe that model units behave in a similar way, as they can be grouped by their temporally-correlated activation. We exploit this feature in our way to obtain big gains in the computational resources needed for the evaluation of our model.

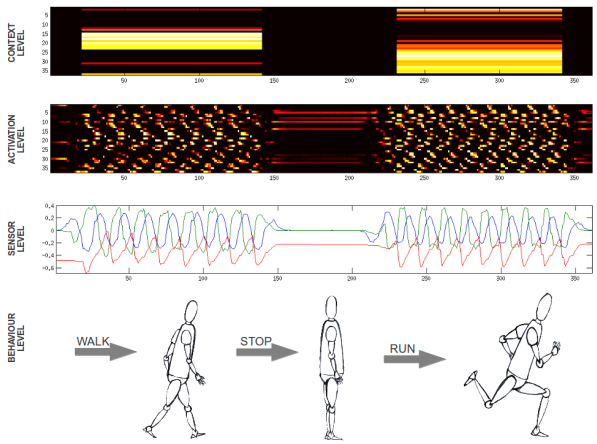


Fig. 1. Schematic view of different levels of abstraction. Different behaviours give rise to different sensory signals, which activate different parts of the model. This activity is summarised in the context level.

Arturo Ribes, Jesús Cerquides Bueno and Ramón López de Mántaras are with Artificial Intelligence Research Institute (IIIA-CSIC), Universitat Autónoma de Barcelona, 08290 Bellaterra, Barcelona, SPAIN aribes@iiia.csic.es

Yiannis Demiris is with the Department of Electrical and Electronic Engineering, Imperial College London, SW7 2BT, UK

In this paper we propose the Context-GMM, a method for applying general GMR to different environmental or behavioural conditions. Those conditions may come either from internal or external variables, that is, self-generated actions or environment hidden variables that affect the dynamics of the robot.

By detecting changes in the activation pattern of mixture components we can identify segments of temporally congruent activations, which we call contexts, and extract a prior distribution over component activations. Figure 1 depicts the different levels of abstraction in our system. At behaviour level we have the actions as perceived by an external human observer. At sensor level we have the raw stream of sensor data that is feed into into the learning system. In activation level we show the activation pattern of mixture components. It can be observed that although there are small differences in levels of activity, the patterns change abruptly when behaviour changes. Lastly, in context level we show the learnt context priors active at each time, where we can see that capture the stationary distribution of component activations. In our experiments we used a mobile robot, which also exhibits high variability in sensory signals, although the depiction of a humanoid is meant for illustration purposes.

II. RELATED WORK

Approaches based in statistical learning of behaviours have become increasingly popular thanks to their capability of dealing naturally with uncertainties in demonstrations.

Among different techniques, Gaussian Mixture Regression has been successfully applied in many imitation and control problems. Dynamical systems researchers approach the problem by learning a probabilistic model of the forces or velocities that need to be applied in order to reproduce or modulate the execution of a trajectory [3][7]. This has been done by learning the distribution of joint velocities in a latent space obtained by dimensionality reduction [3], or by learning an acceleration model that is used to modulate trajectories [7]. More recent work in the line of dynamical systems has focused in optimization of GMM parameters to obtain stable dynamical systems [2].

Another line of research is the use of local approaches to perform regression. Some authors proposed to learn the GMM on the fly by using search algorithms to query a set of data samples close to the input one [4]. Another approach is to learn different local models that are weighted using Gaussian kernels. Each of this models can use a different regression technique, such as local Gaussian Processes [8] or regression in a projected space [9].

In the field of Reinforcement Learning (RL) there has been much effort in recent years in breaking up a big model into multiple local specialised ones. While some work assumes that the number of environment conditions is known *a priori* [10] [11], others approach the problem by incrementally building new models as they detect changes in environment dynamics, either from state transition or reward changes [12]. However, the difference with our work is that our goal is to keep computational complexity very low without sacrificing predictive accuracy, while theirs is to minimize the amount of time spent in re-learning models when the context changes.

III. METHODOLOGY

The data comes in a stream defined by the set of samples $\boldsymbol{s}_t = (\boldsymbol{x}_t, \boldsymbol{y}_t)$ up to a time T. We make a distinction between the input part x and output part y of the data sample, as we want to use the model for probabilistic regression, that is, estimating:

$$\hat{y}(x) = \arg\max_y P(Y = y|X = x) \tag{1}$$

We start by learning a GMM using a state-of-the-art incremental method from [13]. In previous work [14] we studied how this method can be applied to provide long-term predictions of sensory consequences of actions and found that the models need to be large if we want to cover most of the environment and internal conditions.

The objective is to learn a forward model for the dynamical system that predicts the change in the optical flow perceived by the robot as it moves through its environment. The model is defined by the following ODE:

$$\frac{dOF(t)}{dt} = F(s(t), a(t)) \tag{2}$$

where $s(t) = (OF(t), V(t))$ are the sensor variables, in our case the optical flow and the robot velocities provided by the wheel encoders. As our aim is to make long-term predictions, i.e. anticipate the optical flow T time-steps in the future, we learn this as a mapping, changing the time-derivative of optical flow $\frac{dOF(t)}{dt}$ by the difference between the perceived optical flows at time t and time $t + T$:

$$\frac{dOF(t)}{dt} \approx \Delta OF_t^T = OF(t + T) - OF(t) \tag{3}$$

$F(s(t), a(t))$ turns to be the result of performing GMR on our model, conditioned in observing $(s(t), a(t))$.

A GMM M has two kinds of parameters, the set of likelihood functions $p(\boldsymbol{s}|m_j), j = 1..N$, where N is the number of Gaussian components, and the mixing weights, which can be viewed as a prior distribution over the mixture components $P(M)$. In that way, the GMM captures the density:

$$P(\boldsymbol{s}) = P(\boldsymbol{s}|M)P(M) = \sum_j^N P(\boldsymbol{s}|m_j)P(m_j) \tag{4}$$

The context learning module works from a learnt GMM and finds the latent contexts in the stream of data. In this work we manually freeze the model before learning the contexts, although we are working in making both process interact with each other. Figure 2 shows a diagram of how the system operates. Learning is performed in two steps, IGMM learning and context learning. After that, both the learnt GMM and context priors are used in the Context-GMM module to make efficient predictions.

A. Incremental GMM learning

The learning method used is an online formulation using the Robbins-Monro recursive equations [15], modified to deal with an unknown number of mixture components [13]. Although there are other existing methods, based in online formulations of the EM algorithm [16] or Kalman filtering [17], this method can be trained with very few exemplars and gives good results.

At each time-step, a new data sample $\boldsymbol{s}$ is received and reconstructed using the GMM, denoted as $\hat{\boldsymbol{s}}$.

If the normalized error is above the selected accuracy threshold, $|\boldsymbol{s} - \hat{\boldsymbol{s}}|_2 > \lambda^{rec}$, we create a new Gaussian component with mean $\mu = \boldsymbol{s}$ and covariance $\Sigma = \Sigma_{ini}$. In our experiments, we set the initial covariance to a proportional value of the sensor measurement noise. A new component is also added if the likelihood provided by the model is below a minimum threshold $\mathcal{L}(\boldsymbol{s}|M_t) < \lambda^{lhood}$.

B. Context-GMM

After executing different behaviours for some time, we have learnt a GMM that captures the relevant information from the sensorimotor experience of the robot. It can be used for performing tasks by reconstructing the control signals, but it can also be used as a forward model to predict how sensory variables change when the robot executes an action.

The learning method, based in EM, assumes that the dataset is composed from i.i.d. samples generated from a stationary distribution. In robotics, this assumption is almost never fulfilled, as the data comes in a stream while multiple behaviours are executed. This has a strong effect in the learnt prior over the components $P(M)$, which reflects an average distribution of mixture component activations.

At this point we introduce the concept of a context. It can be thought of as a set of factors, which may not be directly observed, that induce a stationary distribution over mixture components. A context may be just a simple action like looking up or a behaviour such as walking forward.

In that way, the learnt prior over the components obtained from incremental GMM learning is a weighted average of the priors induced by the different contexts present at different times:

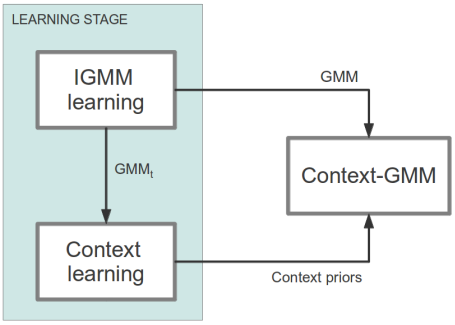


Fig. 2. Schematic view of different levels of abstraction.

$$P(M) = \sum_{c}^{|C|} \lambda_c P^c(M) \tag{5}$$

Where C is the set of contexts and λ_c are weights proportional to the amount of time a context is active over the whole sequence.

Another feature of this context-induced priors is that they are sparse, that is, they have a few non-zero entries. This is key in our approach, as the purpose of learning such priors is that we can evaluate only the components that have some probability of being active, resulting in huge computational savings. In Section III-D we explain in detail how to use those sparse priors. Figure 3 shows a sample of the learnt priors for one of the datasets used in our experiments.

C. Incremental learning of context priors

Context learning is performed in a similar fashion as the incremental GMM. Each context is represented as a prior distribution over the model components $P^c(M)$. We also store the number of samples that have been used to compute the context η_c. This count is used to weight the contribution of new samples when updating a context.

A standard GMM would be a special case of a Context-GMM, where there is only one context, corresponding to the prior distribution obtained when learning the GMM. For this reason, we initialise our context database with one context corresponding to the prior obtained from the incremental GMM learning step.

When a new sample $\boldsymbol{s}$ arrives we compute the likelihood vector for all the components $P(\boldsymbol{s}|M)$. Figure 4(a) shows the log-likelihood of some of the model components over a period of time, where different behaviours are performed. It can be appreciated that there are stable activity patterns while a behaviour is executed. Then we use the active context c^* to compute the activations of each mixture component using the corresponding prior $P^{c^*}(M)$.

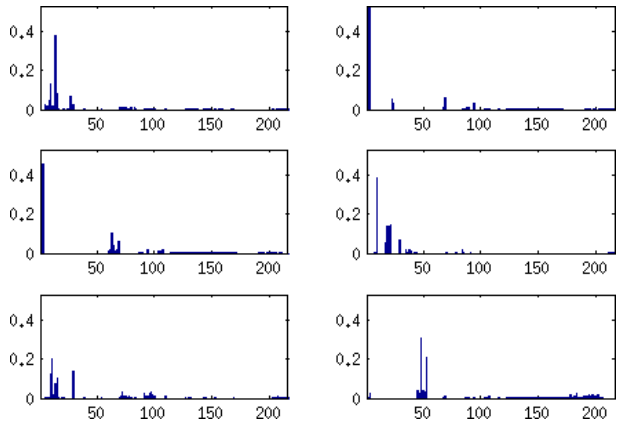
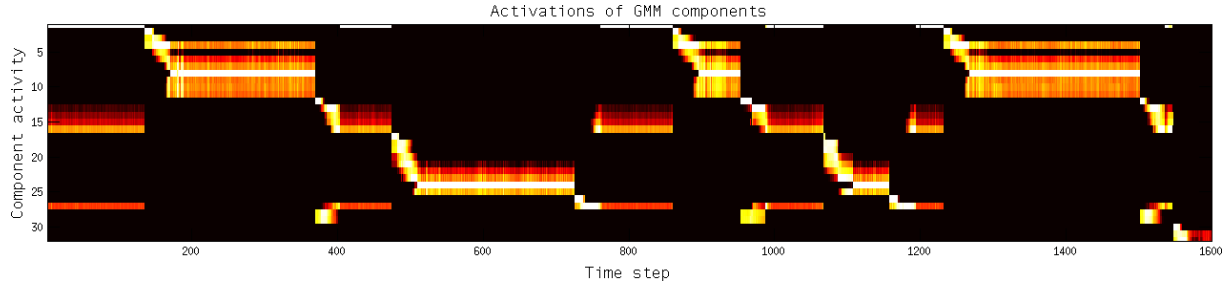
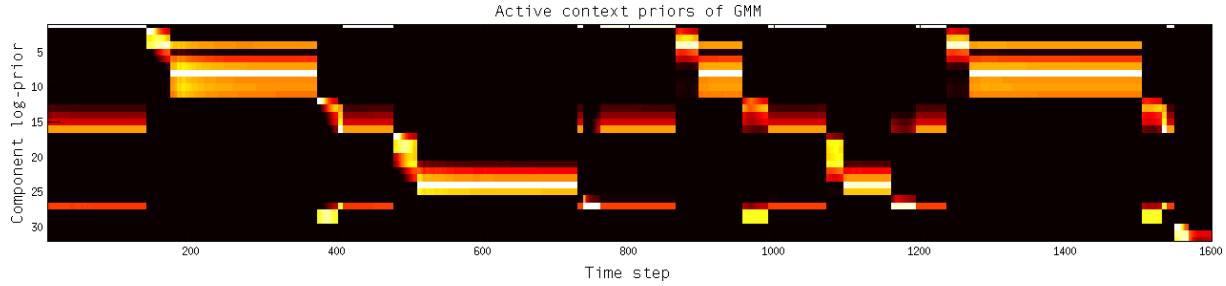


Fig. 3. Learnt prior distributions for the 6 most active context in second dataset, over a total of 16 contexts. Horizontal axis indicates the model component index, while vertical axis is the prior probability for a particular model component. Notice how few components have more than zero probability and, moreover, how increasing the threshold makes the resulting prior even more sparse.



(a) Activity of different model components.



(b) Priors learnt for the active context at each time step.

Fig. 4. Activities of a small subset of model components and its corresponding learnt context priors. Each row corresponds to a different model component and time is plotted in horizontal axis. It can be observed how only a few components present some activity pattern while the others remain inactive. The learnt priors capture the stationary distribution of component activity for each active context.

$$P(m_i|\boldsymbol{s}, c^*) = \frac{P(\boldsymbol{s}|m_i)P^{c^*}(m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)P^{c^*}(m_j)} \tag{6}$$

We also compute a confidence value for the current context, based on the likelihood that it generated the data:

$$err_{c^*} = -log(\sum_j^N P(\boldsymbol{s}|m_i)P^{c^*}(m_i) + \varepsilon) \tag{7}$$

where ε is used to establish an upper bound on err_{c^*}.

Context selection is greedy in a sense that if the error is below a minimum level, we assume that we are still in the same context. If it is above that threshold, it means that the current context is not applicable to the current situation, so we must find whether there is another context that explains the situation or we need to create a new context. We denote this threshold as θ_{err}, which we use to control the amount of context that will be created and also has an impact in the sparseness of the resulting contexts. At this point, we evaluate all the contexts in the database and pick the one with less error as the active context:

$$c^* = \arg\min_c err_c \tag{8}$$

If the best applicable context still has an error above the threshold, we create a new context and set it as the new active context. Its prior is set to:

$$P(m_i|\boldsymbol{s}) = \frac{P(\boldsymbol{s}|m_i)P(m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)P(m_j)} \tag{9}$$

If we could identify an active context from our database, we update incrementally its corresponding prior from time $t-1$ to:

$$P_t^{c^*}(m_i) = P_{t-1}^{c^*}(m_i) + \eta_c^{-1}\frac{P(\boldsymbol{s}|m_i)}{\sum_j^N P(\boldsymbol{s}|m_j)} \tag{10}$$

There are situations in which a context is learnt and never used again. We filter those spurious contexts if they have not been trained with minimum number of data samples, keeping only the stable ones. In our experiments, we set this amount to 10, which we found to work well empirically.

D. Use of sparse priors for GMR

Once context learning is finished, we can use the resulting set of sparse priors to perform GMR in a computationally efficient way.

In a similar way as we did for learning, we use the active context to compute the error score and check if we are still in the same context, changing to a more suitable one in case it is needed.

The components used in a context c are given by the set:

$$C^c = \{i \mid P^c(c_i) > \epsilon\}, i = 1..N \tag{11}$$

In our experiments we show that $|C^c| \ll N, \forall c = 1..|C|$, where N is the total number of mixture components.

The likelihood involved in the computation of err_{c^*} is weighted by $P^{c^*}(M)$, so the fact that we only evaluate a small subset of the model does not change significantly the error score.

Given that the contexts are used in a greedy fashion, we are only forced to compute the likelihoods for the whole model when we detect a change.

IV. EXPERIMENTAL RESULTS

We are interested in anticipating the changes in sensory variables when we perform an action in a given situation. In our case, we want to anticipate long-term changes in optical flow signals. Optical flow is defined as the pattern of apparent motion of objects in scene, caused by their relative motion with the observer. It is represented as a dense vector field, with one 2-D vector in each pixel representing where that pixel was in the previous image of the video sequence.

We want to model the density $p(\Delta OF_t^T, OF_t, V_t, A_t)$, which represents how likely is the change ΔOF_t^T in optical flow OF_t if we perform an action A_t while moving at a velocity V_t. It has to be noted that $\Delta OF_t^T = (OF_{t+T} - OF_t)$, where T is the prediction horizon, in time-steps.

Our experiments are done using a Pioneer Peoplebot with a mounted Kinect camera. We have attached a laptop with a Core 2 Duo 1.8Ghz processor, 2GB of RAM and an NVIDIA Quadro 570M GPU where the optical flow is computed for 320x240 images. No special arrangement of furniture or objects in the lab was done, with the aim of situating the robot in a realistic environment.

The robot is controlled using a joystick, so all the actions are performed by a human. We decided not to use any action decision algorithm because we are concerned with the learning capacity of our system, so we can drive it to challenging situations as required in order to stress its acquired knowledge.

The action space of the robot consists of its linear and angular velocities. In the experiments reported here, we limited the linear velocity to $0.3m/s$ for linear velocity and the angular velocity to $0.6rad/s$.

We captured three different sequences containing different behaviours with increasing levels of difficulty. The first one applies maximum speed commands and either linear or angular, but not both at the same time. The second one has more different combinations of control commands, mixing linear and angular velocity commands at different speeds. The third one is similar to the second one, but the robot approaches obstacles, so the optical flow signal presents different patterns and its predictive distribution is multi-modal.

The reconstruction error of the IGMM algorithm λ^{rec} was set to 5% of the range of the variables. The initial covariance matrix for new components Σ_{ini} is also set to 5% of the range of the variables. This choice was motivated by in an estimation of the amount of sensor noise present in data.

Context learning is analysed in terms of error threshold to detect context changes θ_{err} and the sparsity threshold ϵ. The sensitivity to context changes influences how many contexts are created and their sparsity.

We provide prediction results in two different measurements. One is the mean reconstruction error of the sequence. The second one is a normalised error measure defined as $err_{norm} = 1 - \frac{err_{GMM}}{err_{trivial}}$, where $err_{trivial}$ is the error of a trivial predictor that always predicts that nothing will change, which in terms of our model means $\Delta OF_t^T = 0$ or $OF_{t+T} = OF_t$. It can be viewed as how much using our GMM improves the trivial prediction.

TABLE I
NRMSE RESULTS FOR THE DIFFERENT DATASETS USED IN EXPERIMENTS.

Dataset	nRMSE trivial	nRMSE GMM	Error decrease
Sequence 1	0.088	0.048	45.7%
Sequence 2	0.081	0.048	40.6%
Sequence 3	0.060	0.036	38.5%

In Table I we show the error of the trivial predictor to compare with the full GMM before context learning is applied. The relative error is also included to show how much the GMM decreases the trivial predictor errors.

Figure 5 shows the error reduction respect the trivial predictor for different number of contexts and increasing value of sparsity threshold ϵ. It can be seen that error is very stable across as we increase the minimum prior probability that a model component needs to have in order to be used ($P^c(m_i) > \epsilon$). Furthermore, before the error degrades due to the use of too few components, there is an increase of performance, attributed to using the best model components for prediction in a particular context, or in other words, because we get rid of spurious model components that perturb predictions.

The sparsity results, as can be seen in Figure 5, represent the portion of the model, regarded as sparsity index, that is evaluated during the whole sequence. We can observe three different behaviours depending on the value of ϵ. The first part corresponds to thresholds nearly zero, where only non-zero probability components are used. We see that, in average, only 20% of the model needs to be evaluated to have almost the same accuracy than evaluating the whole model at every time-step.

After a certain point ($\epsilon \approx 10^{-4}$), increasing the threshold steadily decreases the portion of model that is evaluated, meaning that it is using more sparse priors.

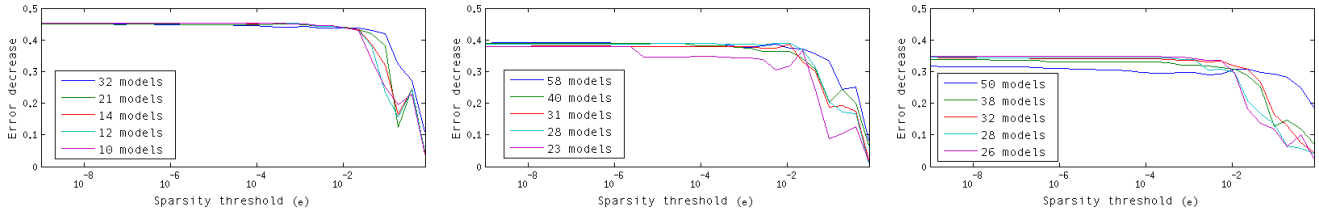
However, we can observe an increasing pattern in the sparsity index. This is explained by the fact that we rely in too few components, so a lot of context change detections are triggered. Those context changes force the system to re-evaluate the whole model to find the new active context, thus loosing the benefits of using only a few components.

Looking at this results, it is desirable to set the threshold ϵ before the error starts dropping suddenly, as we can obtain very good accuracy while evaluating only 10% of the model.

V. CONCLUSIONS AND FUTURE WORKS

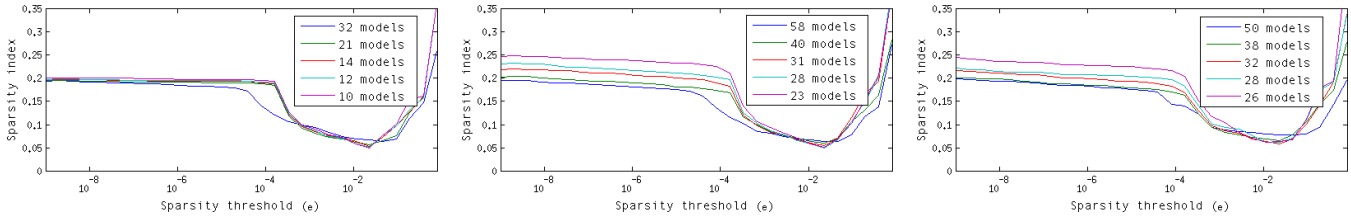
A. Conclusions

In this paper we proposed a method to reduce the computational demand when performing GMR for anticipating changes in sensor variables of a robot. After learning a predictive model based in an incremental GMM, our initial observation was that the execution of behaviours over a significant amount of time, i.e. a few seconds at least, activated only a small region of the internal predictive model.



(a) Errors for first dataset. (b) Errors for second dataset. (c) Errors for third dataset.

Fig. 5. Reduction in prediction error, in vertical axis, compared against the trivial predictor for different number of contexts and different sparsity thresholds, in horizontal axis. After some point, there are too few components used, so the reconstruction becomes equivalent to the trivial predictor.



(a) Sparsities for first dataset. (b) Sparsities for second dataset. (c) Sparsities for third dataset.

Fig. 6. Sparsity indices, in vertical axis, for different number of contexts and different sparsity thresholds, in horizontal axis. It can be seen that if we set the prior probability threshold too high, then the context change detection will produce a lot of false positives, causing the system to evaluate the whole model too often.

This motivated the learning of sparse priors induced by different behaviours being executed by the robot. The sparsity enabled us to select only a subset of model components that have some probability of being active while maintaining the same performance level.
Depending on how many contexts were learnt, we showed that the system achieved good results using less than 10% of the model.

The models obtained by applying the learnt context priors are more compact and fit better the data, while sharing the same basis components. Another feature of contexts is that their mean duration is high enough to benefit from a greedy selection mechanism.

B. Future Works

We plan to interleave the process of learning the underlying GMM and the contexts concurrently. If we take advantage of the computational savings that our method provides, then the underlying model can scale up almost independently of the current prediction, as the working part of the model will be very small.

Contexts provide a high-level representation of the current situation, so we can build more complex models on top of it, like Hidden Markov Models, to further restrict the regions of the model that need to be evaluated when a context change is signalled.

References

[1] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini, "Developmental robotics: a survey," *Connection Science*, vol. 15, no. 4, pp. 151–190, 2003.

[2] S. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *Robotics, IEEE Transactions on*, vol. 27, no. 5, pp. 943–957, 2011.

[3] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 2, pp. 286–298, 2007.

[4] T. Cederborg, M. Li, A. Baranes, and P. Oudeyer, "Incremental local online gaussian mixture regression for imitation learning of multiple tasks," pp. 267–274, 2010.

[5] M. Heinen and P. Engel, "An incremental probabilistic neural network for regression and reinforcement learning tasks," *Artificial Neural Networks–ICANN 2010*, pp. 170–179, 2010.

[6] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 417–424.

[7] M. Hersch, F. Guenter, S. Calinon, and A. Billard, "Dynamical system modulation for robot learning via kinesthetic demonstrations," *Robotics, IEEE Transactions on*, vol. 24, no. 6, pp. 1463–1467, 2008.

[8] 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.

[9] S. Schaal, C. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Applied Intelligence*, vol. 17, no. 1, pp. 49–60, 2002.

[10] S. Choi, D. Yeung, and N. Zhang, "Hidden-mode markov decision processes for nonstationary sequential decision making," *Sequence Learning*, pp. 264–287, 2001.

[11] K. Doya, K. Samejima, K. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural computation*, vol. 14, no. 6, pp. 1347–1369, 2002.

[12] E. Basso and P. Engel, "Reinforcement learning in non-stationary continuous time and space scenarios," vol. 7, pp. 1–8, 2009.

[13] P. Engel and M. Heinen, "Incremental learning of multivariate gaussian mixture models," *Advances in Artificial Intelligence–SBIA 2010*, pp. 82–91, 2011.

[14] A. Ribes, J. Cerquides, Y. Demiris, and R. López de Màntaras, "Incremental learning of an optical flow model for sensorimotor anticipation in a mobile robot," *IEEE International Conference on Development and Learning (ICDL)*, 2012.

[15] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[16] M. Sato and S. Ishii, "On-line em algorithm for the normalized gaussian network," *Neural Computation*, 2000.

[17] R. López de Màntaras and J. Aguilar-Martin, "Self-learning pattern classification using a sequential clustering technique," *Pattern Recognition*, vol. 18, no. 3-4, pp. 271–277, 1985.