The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Gaussian Mixture Models for Affordance Learning using Bayesian Networks

Pedro Osório Alexandre Bernardino Ruben Martinez-Cantin José Santos-Victor

Abstract— **Affordances are fundamental descriptors of relationships between actions, objects and effects. They provide the means whereby a robot can predict effects, recognize actions, select objects and plan its behavior according to desired goals. This paper approaches the problem of an embodied agent exploring the world and learning these affordances autonomously from its sensory experiences. Models exist for learning the structure and the parameters of a Bayesian Network encoding this knowledge. Although Bayesian Networks are capable of dealing with uncertainty and redundancy, previous work considered complete observability of the discrete sensory data, which may lead to hard errors in the presence of noise. In this paper we consider a probabilistic representation of the sensors by Gaussian Mixture Models (GMMs) and explicitly taking into account the probability distribution contained in each discrete affordance concept, which can lead to a more correct learning.**

I. Introduction

Affordances define the relationships between actions, objects and effects. As defined by James J. Gibson more than 30 years ago [7] affordances are *agent-dependent object usages*, i.e. the action possibilities given by an object to an agent with specific action capabilities. For example, a chair is only *sitable* by an individual of a certain height; a tree is only *climbable* by animals with specific capabilities. In fact, what matters in obtaining a certain desired effect are the properties of the objects more than the objects themselves.

The knowledge of object affordances is exploited in most of our decisions: to choose the most appropriate way of acting upon an object for a certain purpose; to search and select objects that best suit the execution of a task; to predict the effects of actions on objects; to recognize ambiguous objects or actions; etc. Affordances are at the core of high-level cognitive skills such as planning, recognition, prediction and imitation.

Thus, learning object affordances is an essential step for humans as they develop the required skills to interact with the environment and ultimately with each other. It is logical that affordances would be modeled in learning humanoid robots, emulating the human development process.

A few works have addressed the problem of learning affordances by autonomous exploration of the world. In [6], a robot learned the direction of motion of different objects when poked and used this information at a later stage to recognize actions performed by others. In [11] a set of predefined actions (tap, touch, grasp) are applied to objects of different shapes, colors and sizes, and the observed effects (object/hand velocities, contact) are used to learn a network of cause-effect relationships. However, these works do not address the noisy characteristics of the perceived sensory data, which may lead to difficulties in the learning stages. This work builds on [11] but extends it in order to model the learning process under noisy observations.

The paper is organized as follows. In the next section we present the affordance learning methodology in [11] where we base our work. Then, we describe our main contribution: to model the noisy nature of observations and the derivation of an EM algorithm for learning the parameters of the Bayesian Network. We present results from simulations where we can observe the improvements provided by the proposed methodology. Finally, we draw the conclusions and point directions for future work.

II. Learning Object Affordances

In this section we briefly describe the affordance learning process as described in [11]. It consists in a developmental approach composed by three stages: (i) learning motor primitives, (ii) learning object representations, (iii) learning effects representation and finally (iv) learning the object's affordances.

A. Developing Basic Skills

The first skills learned by the robot are the motor ones. It starts with a number of predefined actions such as grasping, tapping or touching and fine-tunes these, using its sensory abilities (i.e. relating the observed motion with the desired motion). After this, the robot is presented with different objects and forms perceptual categories. That is, for a number of properties of an object (color, shape, size) it uses unsupervised learning to form meaningful clusters (e.g. it may find 3 clusters in the size space – small, medium, big). In a subsequent stage the robot interacts with objects and, just as in the case of visual perception, it forms clusters for a number of properties (object velocity after contact, contact duration, etc.). Finally, when the robot has learned to execute actions (discrete) and categorize the objects and effects features in discrete categories, it is ready to start creating cause-effect links between actions on objects and the particular outcomes. It is on this stage that we concentrate our attention.

This work was partially supported by the Portuguese Government - Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the POS Conhecimento Program that includes FEDER funds, and through the EC Funded projects HANDLE and FIRST-MM.

P. Osório, student of Biomedical Engineering and collaborator of the Institute for Systems and Robotics, IST, Lisboa, Portugal pedro.osorio@ist.utl.pt

A. Bernardino, R. Martinez-Cantin and J. Santos-Victor are with the Institute for Systems and Robotics, IST, Lisboa, Portugal {alex,rmcantin,jasv}@isr.ist.utl.pt

978-1-4244-6676-4/10/$25.00 ©2010 IEEE

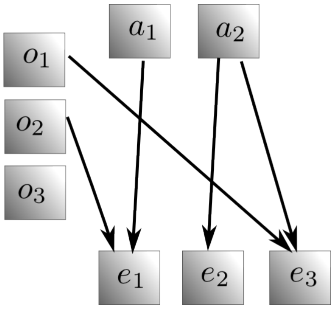


Fig. 1. BN with discrete,observed nodes

B. Affordance Modeling and Learning

Based on the work of [11], we use a graphical model known as Bayesian Network (BN) to model affordances. This model is used to encode probabilistic dependencies between actions, object features and effects of the actions. In this graphical model, nodes represent random variables and the arc structure encodes conditional independence assumptions between the variables. A fundamental result of BN is that the joint distribution of the variables decomposes in the following way:

$$p(X_1, ..., X_n) = \prod_{i=1}^{n} p(X_i | X_{Pa(X_i)}, \theta_i) \qquad (1)$$

where X_i is the random variable associated with node i, θ is the set of parameters of the probability distribution and $Pa(X_i)$ are the parents of node i.

Figure 1 is an example of a BN with the characteristics of the model used in this work. All random variables are observed and discrete-valued. a_i are the actions, o_i are the object features and e_i are the effects of the action.

Learning the affordances requires that, given a database of cases (observations of actions, objects and effects), we find the structure of the BN and the parameters θ_i that describe the relationships between each node and its parents. The structure can be efficiently approximated using Markov Chain Monte Carlo (MCMC) as well as the K2 algorithm.

Since this model only has discrete, observed nodes, we can model the conditional probability distributions of each node given its parents as a multinomial distribution and we can use the Dirichlet distribution as the prior distribution. Then, we can use conjugate Bayesian analysis to do the inference and update the parameters (see [8]).

C. Inference and Interaction Games

After the affordances are learned it is possible to use this knowledge during execution to infer incomplete data using a Bayesian inference strategy. For example, this can be used to interact with others. If a human performs an action on a given object, the robot can observe the effect. Even without a mechanism to identify actions carried out by others, the robot can use the object features, the effect and the learned affordances to infer the action. In fact, we can use the model to predict or infer any of the variables (actions, objects or effects) given the other two.

III. Probabilistic clustering for Affordances

The affordances model is based on the assumption that the variables are separable, because objects features and effects must have some semantic meaning. In some sense, it is the first step towards relational learning, where abstract classes define the properties for interaction. Therefore, the first step is to find the semantic classes by unsupervised learning. In the work of [11], this clustering is based on *X-means* [3], which represents the data distribution as a Gaussian Mixture Model. However, the work of [11] assigns the *maximum a posteriori* (MAP) cluster to each data point. Intuitively, this is a strong assumption considering many of the variables that are involved in the affordances model, like object features such us *color* or *size*.

For example, let us consider that our unsupervised learner defines a ball with a radius greater than 5 cm as *large*, and *small* otherwise. When the sensor detects an object with a radius of 4.9 cm, it automatically assign the semantic meaning of *small* object. Intuitively, we can see that the object in fact falls in between. If we consider the sensor noise, there is a high probability of the object being *small*, but also there is a non-negligible probability of being *large*. In our approach, we want to give a probabilistic assignment to the different classes, which is more robust and can be used to do Bayesian inference in the semantic level.

The Gaussian Mixture Model obtained from *X-means* and similar algorithms provides a density estimation of the probability distribution of the data. The advantage of using mixture models is that we can still assign a semantic meaning to every component, while recovering the whole distribution for Bayesian inference. In the *size* example, we may find that the distribution is the combination of two Gaussians, such us $\mathcal{N}(4, 1)$ for the *small* objects and $\mathcal{N}(6.5, 2)$ for the *large* ones. Therefore, our previous object of radius 4.9 cm has a probability of being *small* and a non-negligible probability of *large*. It is important to note that the inference model that we define in this paper could also deal with probabilistic inputs from the actions. However, in the robotic setup, control noise is negligible compared to sensor noise. Therefore, we consider deterministic actions to simplify the explanation.

Having defined a probabilistic assignment of the clusters to the sensing variables, we can extend the BN to consider the new dependencies (Fig. 2). The nodes that represent the discrete classes of object features and effects are no longer observed directly. Instead, the observed nodes are their children, the sensor nodes which represents continuous random variables.

A. Learning the Structure

By definition, we know that there is a dependency between any continuous node -sensor- and the corresponding discrete variable -feature-. Besides, the sensor is independent of the rest of the network given the value of the feature. Therefore, we can assume that this part of the structure is known and fixed, simplifying the structure learning problem. In fact, the structure to be learned is only the one that connects

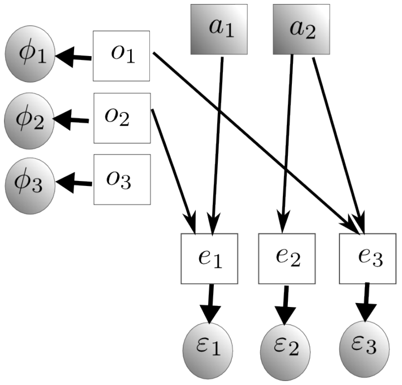


Fig. 2. Bayesian Network of the new clustering approach. The affordances are still represented as interactions between discrete variables (squares). The GMM is represented by the new continuous variables (circles). Shaded shapes represents observed variables during training $\mathcal{X} = (Actions, Objects, Effects)$. However, during execution, we can estimate any variable of $\mathcal{X}$, given the other two observations. The thick arrows represent the new dependence of the sensors and their associated features

discrete nodes, like in [11]. However, the problem is different because, in this case, some of the nodes are hidden. We can use a Bayesian method such us Structural-EM [5], which allows to estimate the structure of the network by integration over all possible hidden variables. However, this Bayesian approach is computationally very expensive and typically requires large datasets. Instead, as in [11] we assign the MAP value to any hidden variable. The MAP value is the one that corresponds to

$$\begin{aligned} e_i^* &= \arg\max_{e_i} p(\varepsilon_i|e_i) \cdot p(e_i) \\ o_i^* &= \arg\max_{o_i} p(\phi_i|o_i) \cdot p(o_i) \end{aligned} \tag{2}$$

where the likelihood $p(\varepsilon_1|e_1)$ or $p(\phi_i|o_i)$, and the prior $p(e_i)$ or $p(o_i)$, come from the estimation of the parameters of the GMM.

Once we have assigned the MAP value to the discrete variables -features-, we can treat them as observed variables. Therefore, we can apply the structure learning algorithms used the work of [11]. As commented in section II-B, we can use MCMC [10] and K2 [2] to learn the structure as if all nodes were observed. An important issue when doing structure learning appears when dealing with causality. In order to find causal relationship, we need to use interventional data. However, in robot interaction, causality is implied in the variables, that is, *acting on an object produces and effect.*

B. Learning the Parameters

Once we have learned the structure, we can proceed to learn the parameters of the BN as commented in section II-B. However, our new model has to deal with incomplete data from the sensors, that is, in terms of network, some variables are hidden. Therefore, we need to integrate over all the hidden variables. For that purpose, we can use the EM algorithm [4].

Before describing in the detail how this is done for the previously described model, the notation used is the following:

- Uppercase letters are used to refer to variables, and lowercase to their values. However, we will often abuse of the notation and use the lowercase both for the variables and their respective values.
- $Y = (Y^{(1)}, \ldots, Y^{(k)})$ is the set of observed variables, where the superindex represent the number of experiment k.
- $Z = (Z^{(1)}, \ldots, Z^{(k)})$ is the set of hidden variables, where the superindex represent the number of experiment k. For each experiment, $Z_i^{(k)}$ being the i^{th} hidden node, where $i = 1, \ldots, N$.
- $X = (X^{(1)}, \ldots, X^{(k)})$ is the set of variables for an experiment, being $X = Y \cup Z$.
- $Pa(Z_i)$ is the set of the parents of Z_i and $\overline{Pa}(Z_i)$ is the set of hidden variables that are not parents of Z_i. Note that the structure is independent of the experiment, so $Pa(Z_i)$ and $\overline{Pa}(Z_i)$ are independent of k.

Since the log-likelihood of the parameters given the complete data, $\log[p(X|\theta)]$ cannot be computed (due to partial observability), the E-step computes its expected value with respect to the distribution $p(Z|Y, \theta^{old})$, where θ^{old} is the vector of parameters achieved on the last iteration of EM. The set of parameters θ that maximize this expected value are then computed in the M-step. In order to simplify calculations, we also assume that the action is only observed through a sensor. Therefore, there is also an auxiliary hidden node for the action, with the corresponding observed child node. This is equivalent to an observed action node but allows the separation of the observed nodes from the hidden ones on the joint probability of the network (1).

1) E step: – Compute the expected value of the complete log-likelihood with respect to the distribution of the complete data (on K experiments):

$$Q(\theta, \theta^{old}) = \alpha + \sum_{k=1}^{K} \sum_{z_k} \left[w_k \sum_{i=1}^{N} \log \theta_i^{(k)} \right] \tag{3}$$

where

$$\alpha = \sum_{k=1}^{K} \sum_{z_k} \left[w_k \cdot \log p(Y^{(k)}|Z^{(k)} \right]$$

being

$$w_k = p(Z^{(k)}|Y^{(k)}, \theta^{old})$$

and

$$\theta_i = p(Z_i^{(k)}|Pa(Z_i), \theta)$$

So the E-step consists of computing w_k. It should be noted that $w_k = w(Z^{(k)})$ and $\sum_{z_k} w(Z^{(k)}) = 1$. The derivation of equation (3) can be found in the appendix.

2) M step: – perform the maximization

$$\theta^* = \arg\max_{\theta} \; Q(\theta, \theta^{old})$$

subject to the constraint

$$\sum_{z_i=1}^{r_i} \theta_i = 1$$

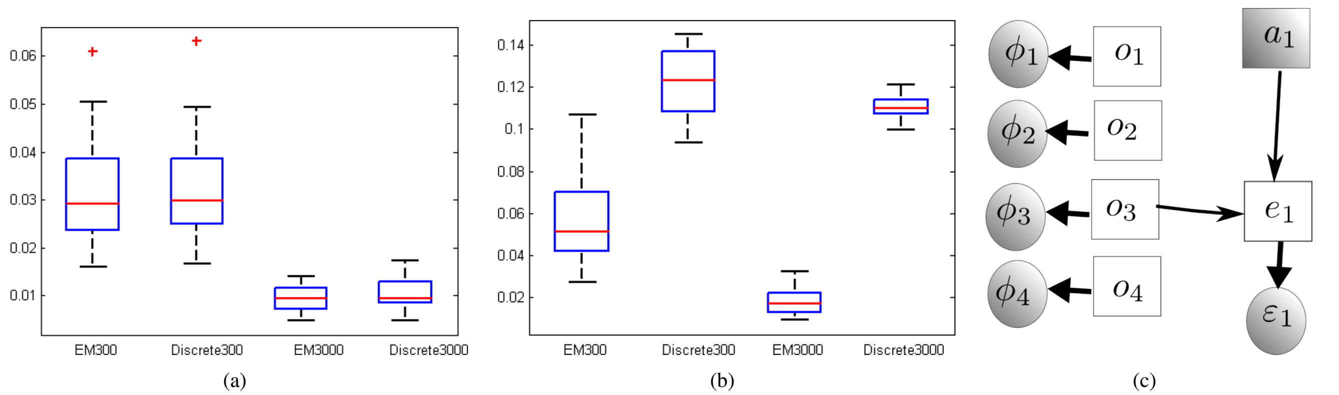


Fig. 3. RMS error of the difference between learned parameters and the real ones given a database with 300 or 3000 cases. a) Sensor noise distributed according to $\mathcal{N}(0,1)$, b) Sensor noise distributed according to $\mathcal{N}(0,9)$ c) The simple Bayesian network.

where r_i is the number of different values the i^{th} node can take. Basically this restriction corresponds to the partition function, that is, for a fixed configuration of its parents, the sum of the probabilities for all possible realizations has to be 1. The optimal value for θ_i is:

$$\theta_i = \frac{\sum_{k=1}^{K} p(z_i^{(k)}, Pa(z_i)|Y^{(k)}, \theta^{old})}{\sum_{z=1}^{r_i}\sum_{k=1}^{K} p(z_i^{(k)}, Pa(z_i)|Y^{(k)}, \theta^{old})} \tag{4}$$

If we consider the probability $p(z_i^{(k)}, Pa(z_i)|Y^{(k)}, \theta^{old})$ as the frequency of appearance of the realization, the solution is analogous to the solution of the complete-data problem. On the fully observable case (not considering the priors) we assigned:

$$\theta_i = \frac{\sharp(z_i, Pa(z_i))}{\sharp(Pa(z_i))} \tag{5}$$

where $\sharp(z_i, Pa(z_i))$ is the number of cases of the database that have $Z_i = z_i$ as well as $Pa(Z_i) = Pa(z_i)$ and $\sharp(Pa(z_i)) = \sum_{z_i=1}^{r_i} N(z_i, Pa(z_i))$.

With hidden-variables, instead of occurrence (1) or non-occurrence (0) of a specific configuration $z_i^{(k)}$, $Pa(z_i)$ on each case of the database, we have the probability of this configuration given the observed variables and the parameters of the network computed on the last step. The *counts* are replaced by the sum of these probabilities over all the cases of the database.

This algorithm *as is*, has a time complexity which is linear on the number of cases in the database, and on the number of iterations. It is, however, exponential on the number of nodes.

IV. Results

A. Procedure

In order to evaluate the differences between the first approach and the one proposed here, two types of Bayesian networks were used. The first (Fig. 3(c)) is a sparse graph (only two arcs, going to the same variable) with seven discrete variables. Each has two possible values which leads to 10 degrees of freedom in the parameters. The second one (Fig. 4(b)) is the Bayesian network learned on [11]. It has eight variables and ten arcs, the discrete variables have between two and four possible values, the parameters of this model have 125 degrees of freedom.

Each of the discrete nodes in these Bayesian networks has a sensor node, similar to the ones described earlier. The purpose of this work is to learn the parameters of the discrete network, assuming that we know the distributions of the sensor models. Thus, the conditional distribution of each sensor given the value, v, of the associated discrete variable was arbitrarily defined to be a normal with mean $5 \cdot v$, and variance constant for all v[1].

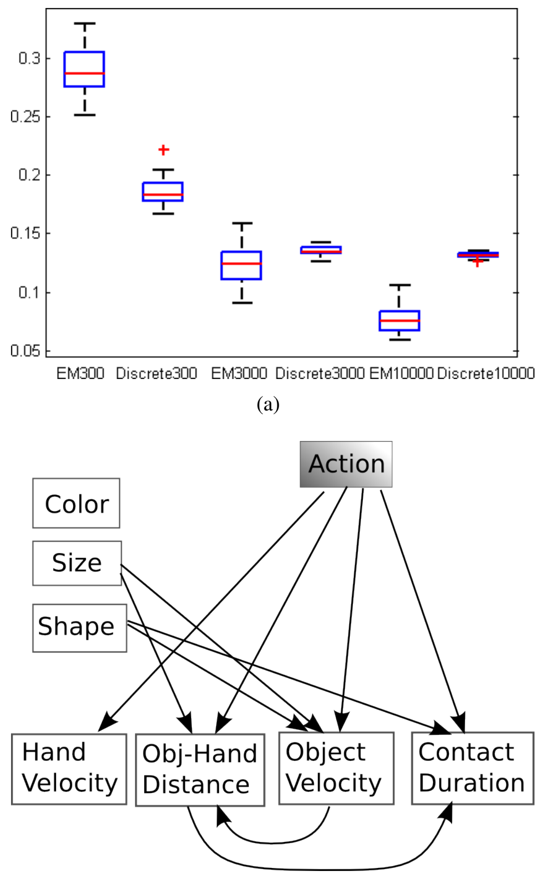
In order to test the EM algorithm against the discrete representation used in [11], hereinafter called *Discrete*, we generate a database of *simulated experiments* based on a known sensors model and Bayesian network, which we will use as ground truth. In order to avoid errors from the clustering algorithm, we assume that the clusters are known. The *Discrete* case takes the most probable cluster for each data point. The EM algorithm is applied to the data and let run until the root mean square (RMS) error of the difference between the parameters obtained in consecutive iterations is smaller than 0.001 to a maximum of 100 iterations. The log-likelihood of the parameters given the data is computed for the parameters obtained using each of the algorithms. The RMS error of the difference between the learned parameters and the ground truth is then computed and used as a metric to evaluate the results of EM and *Discrete*.

B. Simple Bayesian Network Results

The procedure described above was applied 30 times to the first kind of Bayesian network, and the results are on Fig.3(a) and Fig.3(b) as boxplots with median, as well as first and third quartiles. The network is exactly the same in both cases, except that the normal distributions of the sensors have standard deviation 1 and 3, respectively.

Focusing on the first case, it can be seen that, independently of the database size, EM very slightly outperforms the *Discrete* (the RMS error is reduced by less than 2%) algorithm and, as expected, the error is much smaller when more information is provided.

[1] Here, v is assumed to take all integer values between 0 and n-1, if the discrete variable has n possible values



(a)

(b)

Fig. 4. a) RMS error of the difference between learned parameters and the real ones given a database with 300, 3000 or 10000 cases. b) Discrete variables of the realistic Bayesian network. The continuous variables (sensors) have been removed for clarification.

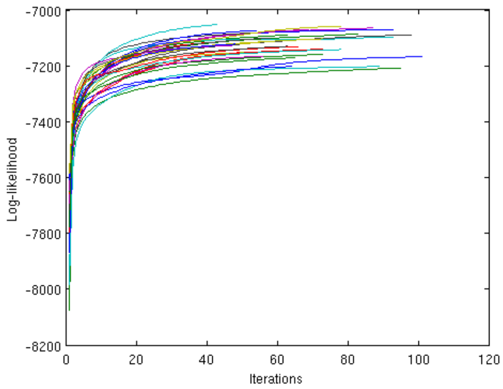


Fig. 5. Evolution of the log-likelihood of the parameters for the realistic network with a database of 300 cases

However, the second case has sensors with bigger variance and hence it is not as straightforward to infer the discrete values from the sensors. Thus, both algorithms perform worse than they did with the first case. However, now EM clearly outperforms *Discrete* (the RMS error is reduced by 58.5% and 84.6% with 300 and 3000 cases); moreover, the EM algorithm greatly benefits from the increase of information whereas *Discrete* has similar errors on both situations.

C. Complex Bayesian Network Results

Apart from these tests with the simple Bayesian network, similar tests were done with the more complex (125 degrees of freedom) network. The means of the sensors distributions are the same as before, the standard deviations are 3, as in the second case. This time around, the procedure was applied exactly as described 30 times, with database sizes of 300 and 3000, and additionally, with a database size of 10000. The results of this procedure are summarized on Fig.4(a).

Just looking at the results for the database size of 300, one could claim that, when applied to this more complex network, the *Discrete* algorithm is superior to EM. However, given that the network has 125 degrees of freedom, it is very hard to believe that any meaningful information can be extracted from only 300 experiences, that would allow the inference of the parameters of the model. As can be seen in Fig.5, with such a short amount of experiments, even though the log-likelihood of the parameters converges, the result is still far from good.

This suspicion is confirmed, and the trend that was observed on the previous case is seen again when the 3000 and 10000 database size results are taken into account. On the 3000 case, the RMS error for the *Discrete* is slightly lower but for the EM algorithm it becomes less than half (compared to the 300 case), the RMS error is 7.7% lower for the EM than for the *Discrete*. When one observes the 10000 database size, it becomes evident that EM makes use of the extra information to perform a much better inference (the RMS error is reduced by 43.1%), while the *Discrete* algorithm performs almost at the same level as with the 3000 database size.

V. Conclusions and Future Work

It has been shown that considering the full distribution of the GMM applied to affordance learning performs better than the MAP estimate, especially in situations where there is significant overlap between the various values that descriptors of object features and effects can take.

However, there are a number of flaws in the application of the algorithm described. Namely, its execution time of the EM algorithm is much slower than the close form solution of the network parameters using the discrete nodes directly. EM is also a batch algorithm, hence there can be no iterative learning in the sense that the algorithm update the parameters as new information becomes available. However, there are some algorithm for online-EM which could be applied in this setup [9], [1].

Interesting future work would include the development of an algorithm with comparable performance to that of EM but more efficient, particularly with regards to its time complexity dependence on the number of nodes, in order to make the algorithm scalable. This algorithm would ideally be able to store the previously performed inference as a prior, allowing iterative learning. This algorithm might also be extended to deal with the structure of the problem similarly to [5].

Also of interest is the question of whether it is feasible to acquire the amounts of information that EM appears to need in order to perform a satisfactory inference of complex Bayesian Networks, given the context, a robot that performs experiments just like a child would.

VI. ACKNOWLEDGMENTS

This work has been partly supported by PTDC/EEA-ACR/70174/2006, grant SFRH/BPD/48857/2008 from FCT and EU projects HANDLE and FIRST-MM.

APPENDIX

In this section, we derive the equations for the EM algorithm.

1) E step: – Compute the expected value of the complete log-likelihood with respect to the distribution of the complete data (on K experiments):

$$\begin{aligned}
Q(\theta,\theta^{old}) &= E_{p(z|y,\theta^{old})}\left[\log p(x|\theta)\right]\\
&= \sum_{k=1}^{K} E_{p(z^{(k)}|y^{(k)},\theta^{old})}\left[\log p\left(x^{(k)}|\theta\right)\right]\\
&= \sum_{k=1}^{K}\sum_{z^{(k)}} \log\left[p\left(y^{(k)},z^{(k)}|\theta\right)\right]\underbrace{p\left(z^{(k)}|y^{(k)},\theta^{old}\right)}_{w_k}\\
&= \sum_{k=1}^{K}\sum_{z^{(k)}} w_k\left[\log p\left(y^{(k)}|z^{(k)}\right)+\log p\left(z^{(k)}|\theta\right)\right]\\
&= \underbrace{\sum_{k=1}^{K}\sum_{z^{(k)}}\left[w_k\log p\left(y^{(k)}|z^{(k)}\right)\right]}_{\alpha} +\\
&+ \sum_{k=1}^{K}\sum_{z^{(k)}} w_k \log p\left(z^{(k)}|\theta\right)\\
&= \alpha + \sum_{k=1}^{K}\sum_{z^{(k)}}\left[w_k\sum_{i=1}^{N}\log\underbrace{p\left(z_i^{(k)}|Pa_i,\theta\right)}_{\theta_i^{(k)}}\right]
\end{aligned}$$

The, we can do the E step by computing

$$Q(\theta,\theta^{old}) == \alpha + \sum_{k=1}^{K}\sum_{z^{(k)}}\left[w_k\sum_{i=1}^{N}\log\theta_i^{(k)}\right] \tag{6}$$

2) M step: – perform the maximization

$$\theta^{*} = \arg\max_{\theta}\ Q(\theta,\theta^{old})$$

subject to the constraints $\sum_{z_i^{(k)}=1}^{r_i}\theta_i = 1$, being r_i is the number of different values the i^{th} node can take.

We can use Lagrange Multipliers λ_i to solve the constrained optimization problem. For simplicity in the derivation, we replace $z_i^{(k)}$ with z.

$$\Lambda = Q(\theta,\theta^{old}) - \sum_{i=1}^{n}\sum_{Pa_i}\lambda_i\left(\sum_{z=1}^{r_i}\theta_i\ -1\right) \tag{7}$$

Now, we can derivate Λ with respect to θ_i and make it equal to 0. Then, developing the equation for a fixed $z_i^{(k)}$ and $Pa(z_i^{(k)}) = Pa_i$:

$$-\lambda_i + \sum_{k=1}^{K}\sum_{\overline{Pa_i}}\frac{w_k}{\theta_i} = 0 \Leftrightarrow$$

$$\Leftrightarrow \lambda_i\theta_i = \sum_{k=1}^{K}\sum_{\overline{Pa_i}} w_k \Leftrightarrow \tag{8}$$

$$\Leftrightarrow \lambda_i\sum_{z=1}^{r_i}\theta_i = \lambda_i = \sum_{z=1}^{r_i}\sum_{k=1}^{K}\sum_{\overline{Pa_i}} w_k \tag{9}$$

Replacing (9) in (8):

$$\theta_i = \frac{\sum_{k=1}^{K}\sum_{\overline{Pa_i}}\ w_k}{\sum_{z=1}^{r_i}\sum_{k=1}^{K}\sum_{\overline{Pa_i}}\ w_k} \tag{10}$$

Thus, we can directly estimate the parameters θ_i. Furthermore, we have defined w_k such us

$$w_k \equiv p\left(z^{(k)}|y^{(k)},\theta^{old}\right)$$

where, we can integrate over all hiden variables that are not parents of $z_i^{(k)}$. Thus,

$$\sum_{\overline{Pa_i}} w_k = p\left(z_i^{(k)},Pa_i|y^{(k)},\theta^{old}\right) \tag{11}$$

By replacing (11) on (10) we obtain equation (4).

REFERENCES

[1] O. Cappé and E.Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B*, 71(3):593–613, 2009.

[2] G.F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

[3] Andrew Moore Dan Pelleg. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society B*, 39:1–38, 1977.

[5] N. Firedman. The bayesian structural em algorithm. In *Uncertainty in Artificial Intelligence, UAI*, 1998.

[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini. Learning about objects through action: Initial steps towards artificialcognition. In *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, 2003.

[7] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979.

[8] D. Heckerman. A tutorial on learning with bayesian networks. Technical report, Microsoft Research Advanced Techonlogy Division, 1996.

[9] P. Liang and D. Klein. Online EM for unsupervised models. In *North American Association for Computational Linguistics (NAACL)*, 2009.

[10] D. Madigan and J. York. Bayesian graphical models for discrete data. *Intl. Statistical Review*, 63:215–232, 1995.

[11] Luis Montesano, Manuel Lopes, Alexandre Bernardino, and José Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *IEEE Transactions on Robotics*, 24(1):15–26, Feb. 2008.