Learning Multiple Behaviors from Unlabeled Demonstrations in a Latent Controller Space

Javier Almingol ALMINGOL@UNIZAR.ES
I3A, Universidad de Zaragoza, Spain

Luis Montesano MONTESANO@UNIZAR.ES
I3A, Universidad de Zaragoza, Spain

Manuel Lopes MANUEL.LOPES@INRIA.FR
Inria Bordeaux Sud-Ouest, France

Abstract

In this paper we introduce a method to learn multiple behaviors in the form of motor primitives from an unlabeled dataset. One of the difficulties of this problem is that the measured behaviors can be very mixed, despite existing a latent representation where they can be separated more easily. We propose a mixture model based on a *Dirichlet Process* (DP) to simultaneously cluster the observed time-series and recover a sparse representation of the behaviors by using a Laplacian prior as the base measure of the DP. We show that for linear models, e.g potential functions generated by linear combinations of a large number of features, it is possible to compute analytically the marginal of the observations and derive an efficient sampler. The method was evaluated using robot behaviors and real data from human motion and compared to other techniques.

1. Introduction

Learning behaviors from multiple unlabeled data has applications in surveillance, monitoring systems, robotics and virtual agents, among others. In surveillance the number of behaviors is unknown and we need to infer a model in a way that allows to predict and interpret activities in new situations. In robotics and virtual agents it is common to use large time-series of human behavior to program robots or virtual characters. By simultaneously inferring the number of behaviors and how the *motion primitives* (MP) generate them, we can bootstrap a set of controllers for many activities.

There are several difficulties to learn multiple behaviors from unlabeled time-series. First, the number of underlying behaviors is usually not known. Second, it may be even difficult to label the data by hand based solely on observations due to perceptual aliasing, different reference frames or not knowing the relevant features of the problem. In this paper we argue that, given an unlabeled dataset of possibly multiple behaviors, there exist a latent representation of the motion primitives that allows to generate the different behaviors (see (Taylor et al., 2007) for an example of learned latent representations of dynamical behavior). A natural way to group these behaviors is to search common motion primitives that generate them accurately. Therefore, our objective is to cluster a collection of time-series and, at the same time, learn the latent control structure that generates them.

The contribution of the paper is a hierarchical model that combines a DP mixture of linear models with a Laplacian distribution as base measure to simultaneously cluster trajectories into behaviors and perform feature selection in a latent representation of the motion primitives. The motion primitives are represented as potential functions generated by linear combinations of features. We show that it is possible to derive an efficient sampler, despite non-conjugacy, by computing in closed form the marginal of each observation under the base measure of the DP. We also present an experimental validation of the proposed method in two applications: clustering robot behaviors and human trajectories. The results show that the approach robustly recovers meaningful behaviors by grouping common sparse latent representations and performs better than other methods such as a sparse EM algorithm or learning and clustering the motion primitives independently.

Appearing in *Proceedings of the 30^{th} International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013. Copyright 2013 by the author(s)/owner(s).

2. Related work

Few works have considered the problem of learning multiple behaviors from unlabeled time-series of data. (Babes et al., 2011) introduced an EM algorithm to cluster time-series in the space of rewards functions using Bayesian inverse reinforcement learning (Ramachandran & Amir, 2007). In (Dimitrakakis & Rothkopf, 2011) multi-task priors are defined on the policy and the rewards for finite MDPs but without considering the clustering problem and just learning a different reward function per trajectory. Another approach is to consider a prior directly on the policy (Doshi-Velez et al., 2010). (Choi & Kim, 2012) used a Bayesian formulation of IRL and a DP prior to perform simultaneously clustering and task learning, generalizing the previous works. As in our case, trajectories can be very mixed in the observation space, but have a compact representation in terms of rewards. Our approach differs from these works in two important aspects. First, they all relied on finite state-action representations while we consider continuous spaces for the actions and the latent space. Second, we incorporate sparsity into the parameter space allowing for more freedom when designing the features, for instance, when the relevant features are unknown.

Another different but complementary perspective has been followed where each time-series is explained by a set of simpler local models. Several authors approached this problem as a pure regression problem adding local models as needed (Schaal & Atkeson, 1998) or relying on non-parametric methods (Hannah et al., 2011; Rasmussen & Ghahramani, 2002; Shahbaba & Neal, 2009; Wood et al., 2008). Recently, more complex models have been considered to model each time series such as a switching system (Fox et al., 2008; Chiappa & Peters, 2010). In contrast with the the first type of methods, the difficulty here is the segmentation of the time series into an unknown number of local models rather than finding an unknown number of global ones. The sparsity property in these approaches aims at reducing the number of local models and not to perform feature selection at the behavior level.

The closest work is (Fox et al., 2008) which proposed a model to jointly extract the local behaviors from multiple unlabeled sequences. Sparsity is enforced using a Beta Process that allows to share local behaviors between the different time series. By using more data it is possible to better estimate these local behaviors and the possible transitions among them within each time series. Another important difference with our work is that in our model sparsity is enforced on the number of features defining a global behavior that explains as many examples as possible according to a DP prior instead of enforcing a sparse number of local behaviors. Nevertheless, this body of work is complementary to the one proposed here. An interesting avenue for future work would be to cluster global behaviors while modeling each behavior as a combination of local models instead of the global one used in this paper.

Finally, it is worth to discuss few works that have included sparsity at the feature level within a DP based clustering algorithm. Gaussian DPMM with variable relevance determination has been recently addressed in (Yau & Holmes, 2011). This was achieved by imposing a hierarchical prior that shrinks the means of the mean in each dimension to the same value for all clusters. The L1-norm was used for Multi-Task compressive sensing using linear models (Qi et al., 2008). The sparse prior was a Gamma distribution over each parameter and, due to non conjugacy, the authors used variational inference instead of Monte Carlo sampling to approximate the posterior with a truncated number of possible clusters.

3. Motion primitives representation

This section describes the latent controller space parametrization at the core of the clustering algorithm and how it can be learned from data of a single behavior within a Bayesian framework. We are interested in representations of motions with the following properties: i) can be learned efficiently from data; ii) can be used not just for discriminating motions but also to generate motions; and iii) can represent a large variety of motions.

We will represent motion as the movement generated by a particle moving along a potential field $V(X)$ (Brillinger, 2007). In other words, the velocity of the particle is given by the gradient of the potential field:

$$\dot{x} = -\nabla V(x) \tag{1}$$

where $x \in \mathcal{R}^d$ and V is the potential function generating the motion. If V is a smooth function then the observations can be approximated using the model $x(t+\nabla_t) - x(t) = -\nabla V(x(t))\nabla_t + \sqrt{\nabla_t}\omega$, where ω is an additive noise term, $\omega \sim N(0, \sigma^2 \mathcal{I}_d)$, and ∇_t is the sampling interval. In order to learn this function, we will consider a linear parametric model for V of the form $V(x) = \phi(x)^T\beta$ with $\phi(x) \in \mathcal{R}^p$ a vector of a priori given features and $\beta \in \mathcal{R}^p$ a vector of parameters to be estimated. Rearranging the terms of the previous equation one obtains:

$$\frac{x(t+\nabla_t) - x(t)}{\sqrt{\nabla_t}} = -\nabla\phi(x)^T\beta\frac{\nabla_t}{\sqrt{\nabla_t}} + \omega. \tag{2}$$

The advantage of this representation over other common approaches, such as (Ijspeert et al., 2003; Jetchev & Toussaint, 2011; Khansari-Zadeh & Billard, 2010), is that the global solution can be found efficiently even when including sparsity constraints and this will be exploited in next section to derive an efficient sampler for the DPMM. Also, it is a simple but general representation equivalent to representations widely used in robotics (Howard et al., 2009; Jetchev & Toussaint, 2011; Chaumette, 2004) and similar to a Lagrangian from physics. Note that this representation allows to encode any type of information within the features, for instance, it is possible to create large sets of non-linear features (e.g. combinations of attractors, polynomials or basis functions) to generate complex motions.

We can now construct a linear system for the whole trajectory $x_{1:T}$ of the form $Y = X\beta$, where matrices Y and X are obtained simply by stacking for all the points of a trajectory the corresponding terms $y_t = \frac{x(t+\nabla_t)-x(t)}{\sqrt{\nabla_t}}$ and $x_t = -\frac{\nabla_t}{\sqrt{\nabla_t}}\nabla\phi(x)^T$, respectively.

4. Sparse DPMM based clustering of trajectories in parameter space

In this section, we introduce our algorithm for clustering trajectories through the use of a linear representation of each behavior. Given a set of m observed trajectories, we can compute the corresponding input and feature gradients $\{Y_i, X_i\}_{i=1..m}$. We assume that they are generated from a mixture model of K ($K < m$) linear dynamical systems (namely a potential function $V_j(\cdot)$ with parameters β_k as described in Eq.1). The mixture model generating the trajectories is given by:

$$p(Y_i \mid X_i, \mathcal{B}) = \sum_{k=1}^{K} \pi_k p(Y_i \mid X_i, \beta_k)$$

where K is the number of components of the mixture, π_k are the mixing weights with $\sum_k \pi_k = 1$, $p(Y_i \mid X_i, \beta_k)$ is the distribution generating the observations for mixture component k with parameters β_k.

To estimate this mixture we use a Dirichlet Process (DP) (Ferguson, 1973) over the β parameters of each controller. A DP is a probability measure on the space of probability measures parameterized by the scale factor α_0 and the base distribution G_0,

$$\begin{aligned} \beta_k &\sim^{iid} G,\ k = 1..K, \\ G &\sim DP(G_0, \alpha_0). \end{aligned}$$

The base distribution is a continuous density that defines the prior over the distributions of the parameters. The DP induces a discrete probability over the samples drawn from the base measure and provides a clustering effect. The scale factor α_0, on the other hand, controls the effective number of classes.

We aim at recovering a sparse representation of each behavior to automatically select the relevant features. Sparsity can be incorporated using a Laplacian prior over the parameters $\beta_k = (\beta_{1,k} \cdots \beta_{p,k})^T$,

$$p(\beta_k \mid \sigma^2) = \prod_{j}^{p} \frac{\lambda}{2\sqrt{\sigma^2}} e^{-\left(\lambda|\beta_{j,k}|/\sqrt{\sigma^2}\right)}, \tag{3}$$

to automatically select relevant features independently for each cluster (Park & Casella, 2008). Conditioning on σ^2 is necessary to guarantee a unimodal posterior which eases the convergence of MCMC samplers. Since each trajectory represents a different behavior, the sparse latent representation is only appropriate within each cluster. For this, we incorporated directly in the base distribution G_0 the Laplacian prior over the parameters β_k of each cluster.

The proposed hierarchical model, shown in Fig. 1, uses the "stick breaking" construction representation of a DP to extend the Bayesian Lasso (Park & Casella, 2008) to an unknown number of models:

$$\begin{aligned} Y_i \mid X_i, c_i = k, \beta_k, \sigma^2 &\sim N(X_i\beta_k, \sigma^2\mathcal{I}_T) && (4)\\ \beta_k \mid \sigma^2, \tau_{1,k}^2, \cdots, \tau_{p,k}^2 &\sim N(0, \sigma^2 D_{\tau_k}) && (5)\\ D_{\tau_k} &= diag(\tau_{1,k}^2, \cdots, \tau_{p,k}^2) && (6)\\ \tau_{1,k}^2, \cdots, \tau_{p,k}^2 &\sim \prod_{j=1}^{p} \frac{\lambda^2}{2} e^{-\lambda^2\tau_{j,k}^2/2} d\tau_{j,k}^2 && (7)\\ \sigma^2 &\sim \frac{1}{\sigma^2} d\sigma^2 && (8)\\ c_i &\sim Multinom(\pi_1, ..., \pi_k) && (9)\\ \lambda^2 &\sim Gamma(a, b) && (10) \end{aligned}$$

where the indicator variables $c_i \in \{1..K\}$ assign each trajectory to one of the K components of the mixture. The model uses the representation of a Laplacian as a scale mixture of normals through the variables $\tau_{1,k}^2, \cdots, \tau_{p,k}^2$. Notice that the sparsity term affects independently each mixture component and, therefore, features selected in one component do not affect features selected in another component.

Under this model, the conditional distributions can be computed due to the conjugacy properties exploited in the Bayesian Lasso. In particular, the full conditionals of β_k, $\tau_{j,k}^{-2}$, σ^2 and λ^2 are respectively

$$N\big((X_{|k}^T X_{|k} + D_{\tau_k}^{-1})^{-1} X_{|k}^T Y_{|k}, \sigma^2 (X_{|k}^T X_{|k} + D_{\tau_k}^{-1})^{-1}\big), \tag{11}$$

$$Inv - Gaussian\left(\sqrt{\frac{\lambda^2\sigma^2}{\beta_{j,k}^2}}, \lambda^2\right), \tag{12}$$

$$IG\left(\frac{mT-1+Kp}{2}, \frac{\sum_k (Y_{|k} - X_{|k}\beta_k)^T (Y_{|k} - X_{|k}\beta_k) + \beta_k^T D_{\tau_k}^{-1} \beta_k}{2}\right), \tag{13}$$

$$Gamma(a + Kp, b + \textstyle\sum_{k=1}^{K} \sum_{j=1}^{p} \frac{\tau_{j,k}^2}{2}), \tag{14}$$

where $X_{|k}$ and $Y_{|k}$ denote the stacked matrices of all trajectories assigned to cluster k and IG denotes an

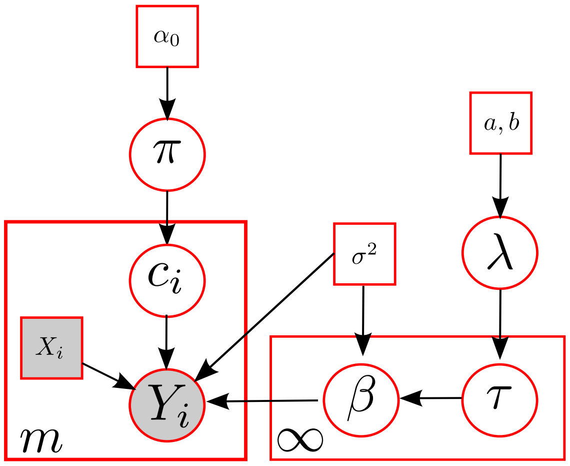

Figure 1. Graphical model.

Inverse-Gamma (see (Park & Casella, 2008) for a complete description of the parameterizations of the Inv-Gaussian, Inv-Gamma and Gamma distributions).

In order to implement the Gibbs sampler for the DPMM we also need to compute the marginal of each trajectory under the base measure G_0. In our case, this measure is the prior distribution defined on the space of the β and τ parameters as defined in Eqs.5 and 7. The corresponding likelihood function is defined in Eq.4. To compute this marginal we note that

$$\begin{aligned} q_0 &= \int_{\beta,\tau^2} f(\cdot|\beta)p(\beta,\tau^2 \mid \sigma^2)\; d\beta d\tau^2 \\ &= \int_{\beta} f(\cdot|\beta)p(\beta \mid \sigma^2)\; d\beta. \end{aligned}$$

since integrating out the $\tau^2 = (\tau_1^2 \cdots \tau_p^2)^T$ results in the conditional Laplace prior defined in Eq. 3.

Due to the use of a Laplacian prior, the model does not preserve conjugacy. However, it is possible to analytically compute q_0 (see Appendix A). The derivation requires to decompose the covariance matrix of the predictors to decorrelate the features and, due to the Laplacian prior, to consider the integral on the positive and negative values of each β_j independently. We provide the full derivation in the supplementary materials. With such computation we can define an efficient sampler as shown in Algorithm 1.

Algorithm 1
Sparse Dirichlet Process Mixture Model (DPMM)

```
Given a set of trayectories $\{Y_i, X_i\}$, $i = 1 \cdots m$
  • Sample $s^{(t)}$, $t = 0$:
      1. Create one cluster per trajectory $K^{(t)} = m$; $c_i^{(t)} = i \quad i = 1, \cdots, m$
      2. Init $\{\beta_k^{(t)}, \tau_{jk}^{(t)}\}_{k=1}^{K^{(t)}}$,$\sigma^{2,(t)}$ and $\lambda^{2,(t)}$
  • Sample $s^{(t)}$, $t \geq 1$
      1. $s^{(t)} = s^{(t-1)}$
      2. For $i = 1, \ldots, m$
          (a) Compute $q_{0,i}$ as in Appendix A
          (b) Sample assignment $c_i^{(t)}$ and update $K^{(t)}$ following (Neal, 2000)
      3. For $k = 1, \ldots, K^{(t)}$
          (a) Sample ${\beta_k}^{(t)}$ y ${\tau_{jk}}^{(t)}$ with Eqs. 11 and 12.
      4. Sample $\sigma^{2,(t)}$ and $\lambda^{2,(t)}$ with Eqs. 13 and 14.
```

5. Results

5.1. Robot Behaviors

We consider a task where a mobile robot has to collect different types of objects from the environment and take them to a base. The behavior of the robot is defined by the type of objects it has to collect and the ones it has to avoid collisions with. For two classes of objects, we define five different controllers depending on which objects to pick (one of the two classes or both) and on the need for collision avoidance to get away of objects while fetching and carrying others (see Figure 2(a)). However, depending on the location of the objects, they might lead to ambiguities. For instance, consider two behaviors that differ in their obstacle avoidance policy. If they do not find an obstacle, they will generate identical trajectories. The trajectories are difficult to visualize because they are in a 6D space (the 2D locations $p = (p_u, p_v)$ of the robot and the objects). It is clear that any clustering made on the observed time-series can not work because the trajectories are context dependent making any comparison among observed trajectories meaningless. The controller of each behavior is implemented using servoing (Chaumette, 2004) to map each point of the 6D input space into the speed in each axis of the plane (v_u, v_v).

At a given time step we define the following features for the relative location of the robot p w.r.t. another location p_o: $\phi_o = \left\{e^{-\alpha\|p-p_o\|^2}, \|p - p_o\|\right\}$. An extra feature h indicates whether the robot is holding an object or not. The full feature vector is then:

$$\phi = \{\phi_o h, \phi_o(1-h),\; o \in \{o_t, o_1, o_2\}\},$$

where o_t represents the target location and o_1 and o_2 are the locations of the objects of each class. For two classes and one object per class, this amounts to a total of 12 features. We note that ϕ corresponds

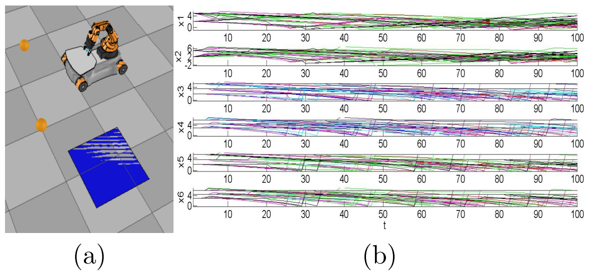

Figure 2. (a) Simulated robot arena. The robot collects objects of interest while avoiding or ignoring other objects. (b) Noisy observed robot trajectories for each dimension. Each color represents a class generated by a different servoing controller. Note that trajectories are mixed in observation space due to the dependency on the object and target location.

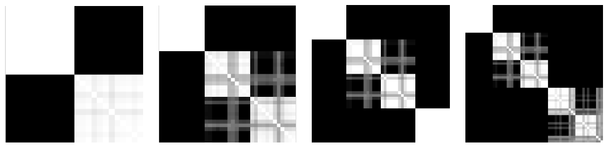

Figure 3. Averaged confusion matrix for assignments of the DPMM (top row) for 2,3,4 and 5 classes over ten runs. The lighter the component (i,j) is the more frequent trajectories i and j were assigned to the same cluster. The matrix would ideally be block diagonal.

to simple, generic, features not including any special domain knowledge. On the other hand, the feature h does include some domain knowledge, namely the fact that the robot controller has two modes of operation depending on whether the robot already picked up the object or not. However, we believe this is a very high level information that can be easily encoded in the algorithm. Furthermore, when comparing the algorithm to other approaches, all the methods benefit from this domain knowledge. For each behavior we simulated 200 time steps trajectories from 12 different initial configurations. Every time an object was taken to the base it reappeared in a random location. The measurements were corrupted with noise sampled from $N(0, 0.05\mathcal{I}_6)$. Figure 2(b) shows the trajectories of all five behaviors.

5.1.1. Clustering behaviors

We first analyze the clustering obtained by the DPMM algorithm for different number of classes. We show our results using a correspondence matrix where each component i, j of the matrix represents the frequency of assignment of trajectories i and j to the same cluster. The results shown are averaged over 500 samples after a burning period of 500. Figure 3 shows that the DP usually finds the right number of clusters and assigns the trajectories consistently to the same group. As mentioned before, there exist some behaviors which may generate indistinguishable trajectories and this is observed in the off diagonal blocks, specially for classes 2 and 3 and classes 4 and 5. Figure 4 shows the posterior distribution over the number of clusters provided by the DPMM algorithm. This figure shows that most of the probability is assigned to the right number of classes. The structure of the data makes difficult to estimate a smaller number of clusters without incurring a higher reconstruction error. However, it is possible to split some of the clusters into two separated ones.

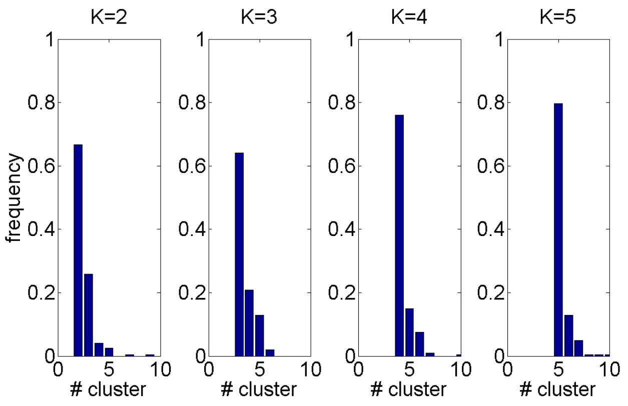

Figure 4. Frequency of the estimated number of clusters for different number of classes.

We will now compare our DPMM method with two baseline algorithms, both assuming that the number of clusters is known. The first one is a sparse EM algorithm that uses the L1-norm in the minimization step to compute the β parameters for each cluster and allows us to illustrate the benefits of using the DP prior. The second method corresponds to a naive solution to our problem, denoted Trajectory Reconstruction and Clustering (TRC). TRC estimates first a controller (i.e. the β parameters) for each trajectory and then clusters the resulting controllers using Kmeans (Macqueen, 1967). In this case, we analyze the benefits of simultaneously clustering and discovering the sparse latent space of the controller. We do not report results of a direct clustering on the observation and feature spaces since they results were systematically much worse.

To analyze the properties of the proposed method, we consider three cases using 50, 100 and 200 time steps, respectively. We perform this comparison because longer trajectories are more informative about the underlying behavior. The TRC should be able to estimate more accurately the parameters of each single trajectory making easier the clustering of the

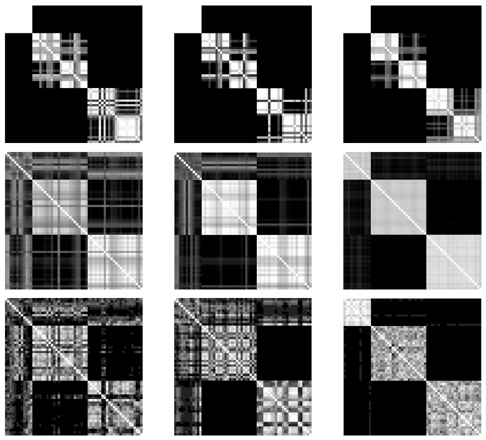

Figure 5. Averaged confusion matrix for assignments of the DPMM (top row), EM (middle row) and TRC (bottom row) for trajectories of five different behaviors. The figures of each row show the results for 50 (left), 100 (middle) and 200 (right) time steps, respectively. EM and the TRC were run with the real number of clusters.

estimated controllers. Figure 5 shows the correspondence matrices obtained by the DPMM, EM and TRC methods for five different behaviors with trajectories of different durations. For the DPMM, the results do not vary significantly. It is always able to recover the right number of clusters. The only variation is that the ambiguities between classes diminish for longer trajectories because the probability of having multiple indistinguishable behaviors decrease. Despite the number of clusters is known, the EM algorithm is not able to discriminate properly between similar behaviors [1]. It behaves better with longer trajectories with less assignments off the diagonal blocks, but still fails to distinguish classes 2,3 and 4,5 . Although we used multiple initializations, the EM seems to get stuck in a local minima with three dominating clusters instead of five independently of the trajectory length. The results for the TCR algorithm show that shorter trajectories make the estimated β less stable and more difficult to cluster. If trajectories are long enough, the TCR better estimates the controller parameters for each single trajectory and the confusion matrix is closer, but still not as good, as the EM one (and much worse than the DPMM).

The assignment matrix shows if trajectories belonging to the same class were grouped together. This is useful in applications where the behaviors need to be identified unambiguously, e.g. surveillance or robot control. Other applications will be more interested in the reconstruction quality of the trajectories, being the assignments whatever is needed for reconstruction, e.g. computer graphics. Thus, we also evaluate the prediction error on a test set using the behaviors learned for the different clusters.

[1]The correspondence matrix for the EM is computed using the correspondence probabilities of the E-step for each class.

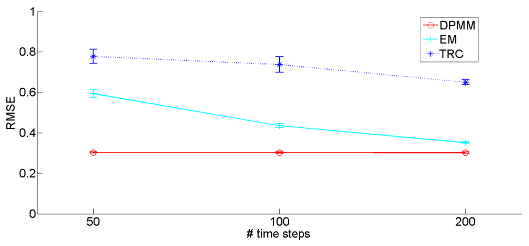

Figure 6. RMSE for different trajectory lengths for the DP, EM and TRC algorithms.

We created 12 test trajectories for each behavior starting from random points. We computed the root mean square error (RMSE) of the test trajectories on each of the estimated controllers using Eq. 2. The RMSE measure how well the potential created by the estimated parameters β_k matches the gradients from the test trajectories. To solve the correspondence problem, we selected the minimum RMSE. Figure 6 shows the RMSE obtained by the different algorithms. The DPMM behaves better than the other algorithms due to the better clustering properties. For 200 time step trajectories, the EM algorithm is close to the DPMM, but it degrades its performance with less informative trajectories (50 or 100 time steps). Note that the changes in RMSE per trajectory will not be very large even if the behavior is not correct since obstacle avoidance is a short and local maneuver that does not accumulate over time. However, not avoiding an obstacle may be a fatal failure for a robot controller. We conclude that DPMM provides a more accurate behavior assignment, is able to find the correct number of clusters, has better reconstruction properties and is more robust when few data is available.

5.1.2. Feature selection

We next analyze the feature selection properties of the DPMM. Table 1 shows that the parameters found by the DPMM algorithm were quite similar to the sparse original ones. Furthermore, the correlation between both parameters was always over .98 since the DPMM recovered the right coefficients. However, the Laplacian prior did not shrink to zero all parameters not needed by the controller and some of them kept small values (under 0.05). Removing the Laplacian prior resulted in much larger L1-norms and the recovered parameters did not match the true parameters. However, it did correctly group the trajectories and the reconstruction error increased only slightly. EM also founds sparse parameters but with a worse reconstruction and assignment matrix.

Table 1. L1-norms of estimated parameters $\|\hat{\beta}\|_1$, correlation $corr(\beta, \hat{\beta})$ with the true ones and reconstruction error (RMSE) on the test set.

		class					RMSE
		c1	c2	c3	c4	c5	
true	$\|\beta\|_1$	15	13	10	13	10	-
EM	$\|\beta\|_1$	15.1	10.8	10.7	10.8	10.8	0.35
	$corr$	1	0.98	0.99	0.98	0.99	
$DPMM$	$\|\beta\|_1$	15.4	12.5	11.0	12.5	10.6	0.3
$\hat{\lambda} = .15$	$corr$	0.99	0.98	0.99	0.98	0.99	
$DPMM$	$\|\beta\|_1$	15.56	49.3	352.4	182.2	10.7	0.33
$\lambda = 0$	$corr$	0.99	0.38	0.22	-0.08	0.99	

We further tested the ability of the DPMM algorithm by expanding the feature vector with 78 correlated features resulting of cross products of the original ones. Table 2 shows the L1-norm of the estimated parameters and the correlation with the original ones. The first row shows the results for the sampled values of λ, while in the other two λ was fixed manually to a lower value. The Laplacian prior has a strong effect as the L1-norm increases and the correlation with the original parameters decrease. Also, the reconstruction error is better which indicates that sparsity avoids over-fitting. However, there were no differences in terms of clustering and correspondence matrices. Therefore, for this dataset sparsity helps to recover a compact and interpretable controller and to reduce the reconstruction error but does not play a crucial role in grouping trajectories together.

Table 2. Feature selection with 78 correlated features.

		class					RMSE
		c1	c2	c3	c4	c5	
DPMM	$\|\hat{\beta}\|_1$	18.4	15.4	12.8	14.9	13.3	0.3
$\hat{\lambda} = 0.28$	$corr$	0.99	0.98	0.99	0.98	0.99	
DPMM	$\|\hat{\beta}\|_1$	125	92.3	105	101	90.2	0.31
$\lambda = 0.01$	$corr$	0.42	0.47	0.38	0.42	0.51	
DPMM	$\|\hat{\beta}\|_1$	606	644	603	646	514	0.35
$\lambda = 0$	$corr$	0.07	0.07	0.06	0.06	0.06	

5.2. Edinburgh Informatics Forum Pedestrian Database

We now use the Edinburgh Informatics Forum Pedestrian Database (Majecka, 2009) to cluster the trajectories of people in a open public space (see Fig. 7(a)). The input data is a set of 474 trajectories in $2D$. Each trajectory was generated by sampling 22 points from a six point spline approximation of each tracked people in normalized pixel coordinates (u, v). Two type of features were computed from the trajectory points using: 1) Gaussian attractors as in Section 5.1 over a $8x8$ grid on the image plane located at $o_t, t = 1 \dots 64$, and 2) a polynomial of degree two. The dimension of the feature space is therefore $64 + 7$.

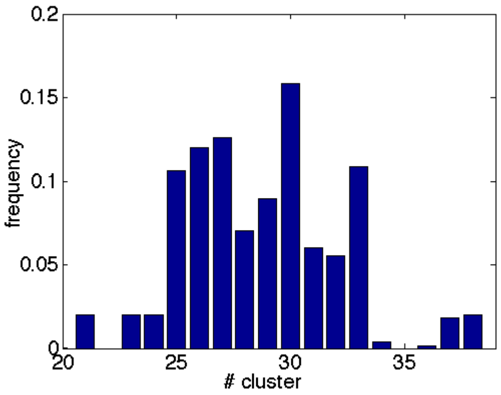

Figure 8. Frequency of the estimated number of clusters for the EIFPD dataset.

$$\phi(u, v) = \left\{1; u; v; uv; uv^2; u^2v; u^2v^2; e^{-\alpha\|(u,v)-o_t\|^2}|_1^{64}\right\}.$$

To evaluate the algorithm, we used the hand-made clustering provided in (Majecka, 2009) where trajectories were grouped in 51 classes based solely on manually selected entry and exit points. The correspondence matrix is shown in Fig. 7(b). The reconstruction error (RMSE) obtained with the potential functions estimated from the manually labeled trajectories was 1.95. We also ran a KMeans with 51 classes, the number of classes provided by manual labeling, on the trajectories to assess the difficulty of the clustering on the observed trajectories space. The correspondence matrix for KMeans is shown in Fig. 7(c). Although the matrix has some structure similar to the ground truth, the assignments are more randomly distributed and the reconstruction error is higher (3.04).

The DPMM algorithm was run with 1000 samples and a burn-in period of 300. Figure 8 shows the posterior distribution for the number of clusters. The average number of clusters is 28.7, but there is no clear peak indicating multiple possible groupings. The number of cluster was less than the ground truth partition, due to the grouping of different entry/exit points when their trajectories could be explained by a single potential function. The correspondence matrix is shown in Fig. 7(d). It shows more structure than the KMeans and recovers a more similar pattern to the hand-made clustering. The reconstruction error was 1.58, lower than the one provided by the ground truth. This improvement is due to the fact that the DPMM was able to consider the whole trajectory and fit functions that approximated the global behavior measured in the image, not just entry and exit points. Figure 7(e) shows the reconstructed trajectories from each initial point given the parameters obtained with the most likely particle.

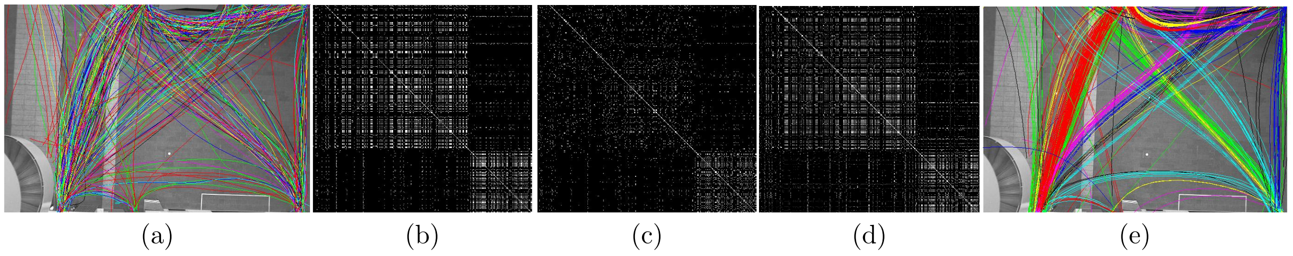

Figure 7. EIFPD dataset. (a) Trajectories of the EIFPD to be clustered (color is non-informative). (b-d) correspondence matrix for the 474 trajectories for the labeled ground truth, the KMeans in measurement space and the DPMM, respectively. (e) Reconstructed trajectories from the initial point using the estimated parameters of the DPMM algorithm. Due to the large number of clusters, colors are repeated for different clusters.

We see that they represent very accurately the average behavior per class of which each individual demonstrations was a noisy observation. This shows how the reconstructed trajectories can indeed represent the behavior of each class and be used to generate new motions and potentially be used for behavioral prediction. We did more experiments to test the sensitivity with the choice of features. Training the DPMM only with Gaussian attractors resulted in a slightly higher $RMSE$ of 1.69 and a mean number of clusters of 26. For polynomials, the $RMSE$ decreased to 1 due to a higher number of clusters (mean of 61.2) and a correspondence matrix with less structure. These results show that the algorithm is able to deal with different feature spaces with good accuracy in terms of $RMSE$. Depending on the problem domain, different choices of features will lead to different interpretations of the resulting behavior models.

We also evaluated the effect of the sparsity prior on the clustering process by fixing different sparsity levels. Contrary to the robot example, sparsity decreased the total number of clusters, the mean of the estimated λ was 1.573 and the sum of the L1-norm of the estimated parameters of 10.41 for the whole feature vector (0.236 and 68.64 with only Gaussian attractors). When λ was fixed to 10^{-4}, the mean number of clusters was 38, the $RMSE$ was 1.37 and the sum of the L1-norm of the estimated parameters was $1.22 \cdot 10^5$. This is due to the fact that the number of clusters is not so clearly defined by the data and a sparse β regularize trajectories and make their clustering less costly.

6. Conclusions

In this paper we presented a novel algorithm for learning an unknown number of behaviors from a set of unlabeled trajectories which can be highly mixed in the measurement space. It estimates the number of behaviors in a latent controller space and selects the relevant dimensions for each one. By selecting a linear representation, in this paper a potential function generated by a linear combination of features, it is possible to derive a sparse version of the DPMM based on Laplacian priors over the parameters and compute the marginal of the observations in closed form.

The results show that the latent controller space allows to reliably recover groups of behaviors with a sparse representation that can be used to generate new instances of the learned behaviors without losing in terms of reconstruction error. The effect of the sparsity prior on the clustering depends on the problem at hand and the selected features. Although it may not always change the clustering, the recovered representation is always simple and more interpretable with similar accuracy. The results also show that the proposed method behaves better than a sparse EM version for mixture models and naive approaches such as clustering directly on the feature space or fitting a controller for each trajectory and then cluster the estimated parameters.

There are some interesting future directions. First, the method can be applied to any linear representation. The potentials used in this paper do not directly model sequential data and it would be interesting to extend this to models that do so, for instance, linear auto-regressive models. Second, it may be possible to combine this type of clustering of global trajectories with recent work on learning multiple behaviors within a single sequence using multiple sequences as in (Fox et al., 2008) or (Chiappa & Peters, 2010).

A. Expression for $\int_\beta f(\cdot|\beta)p(\beta \mid \sigma^2)\, d\beta$

This appendix provides the expression for $q_0 = \int_\beta f(\cdot|\beta)p(\beta \mid \sigma^2)\, d\beta$ when the likelihood model is a Multivariate Normal and the prior is a Laplacian distribution. After resolving the integral over β we get the following expression

$$q_0 \quad = \quad \left(\frac{\lambda}{2\sigma}\right)^p \frac{1}{\sqrt[T]{2\pi\sigma^2}} exp\left(-\frac{Y^TY}{2\sigma^2}\right) I$$

$$
\begin{aligned}
I &= \prod_{i=1}^{\hat{p}} (T_{1i} + T_{2i}) \quad \prod_{i=\hat{p}+1}^{p} -\frac{2}{b_i} \\
T_{1i} &= exp\left(\frac{(b_i - e_i)^2}{2d_i}\right)\sqrt{\frac{\pi}{2d_i}} erfc\left(\frac{-(b_i - e_i)}{\sqrt{2d_i}}\right) \\
T_{2i} &= exp\left(\frac{(b_i + e_i)^2}{2d_i}\right)\sqrt{\frac{\pi}{2d_i}} erfc\left(\frac{-(b_i + e_i)}{\sqrt{2d_i}}\right)
\end{aligned}
$$

where $\hat{p}$ is the rank of matrix $A = \frac{1}{\sigma^2} X^T X$ and d_i are the eigenvalues of matrix A. The integral requires to decompose matrix $A = SDS^{-1}$ to compute the vector $E^T = \frac{Y^T}{\sigma^2} XS$ and the vector $B^T = (\frac{-\lambda}{\sigma}, \cdots, \frac{-\lambda}{\sigma})R$, where matrix R is defined as the component wise absolute value of matrix S.

References

Babes, M., Marivate, V., Littman, M., and Subramanian, K. Apprenticeship learning about multiple intentions. ICML, 2011.

Brillinger, D.R. Learning a potential function from a trajectory. *Signal Processing Letters, IEEE*, 14(11): 867–870, 2007.

Chaumette, F. Image moments: a general and useful set of features for visual servoing. *Robotics, IEEE Transactions on*, 20(4):713–723, 2004.

Chiappa, S. and Peters, J. Movement extraction by detecting dynamics switches and repetitions. *Advances in neural information processing systems*, 23: 388–396, 2010.

Choi, Jaedeug and Kim, Kee-Eung. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *Neural Information Processing Systems (NIPS)*, 2012.

Dimitrakakis, C. and Rothkopf, C. Bayesian multitask inverse reinforcement learning. *Arxiv preprint arXiv:1106.3655*, 2011.

Doshi-Velez, Finale, Wingate, David, Roy, Nicholas, and Tenenbaum, Joshua. Nonparametric Bayesian Policy Priors for Reinforcement Learning. In *Neural Information Processing Systems (NIPS)*, 2010.

Ferguson, T. A bayesian analysis of some nonparametric problems. *The annals of statistics*, 1:209–230, 1973.

Fox, E.B., Sudderth, E.B., Jordan, M.I., and Willsky, A.S. Nonparametric bayesian learning of switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 21:457–464, 2008.

Hannah, Lauren A., Blei, David M., and Powell, Warren B. Dirichlet process mixtures of generalized linear models. *J. Mach. Learn. Res.*, pp. 1923–1953, July 2011.

Howard, M., Klanke, S., Gienger, M., Goerick, C., and Vijayakumar, S. A novel method for learning policies from variable constraint data. *Autonomous Robots*, 27:105–121, 2009.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. Learning attractor landscapes for learning motor primitives. In *Neural Information Processing Systems (NIPS)*, Cambridge, MA, 2003.

Jetchev, N. and Toussaint, M. Task space retrieval using inverse feedback control. In *Intl. Conf. on Machine Learning (ICML)*, 2011.

Khansari-Zadeh, S.M. and Billard, A. Bm: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models. In *Robotics and Automation, IEEE Inter. Conf. on*, 2010.

Macqueen, J. B. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Math, Statistics, and Probability.* University of California Press, 1967.

Majecka, B. Statistical models of pedestrian behaviour in the forum. Master's thesis, School of Informatics, University of Edinburgh, 2009.

Neal, R. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9:249–265, 2000.

Park, T. and Casella, G. The Bayesian Lasso. *J. Am. Stat. Assoc.*, 103(482):681–686, June 2008. ISSN 0162-1459.

Qi, Y., Liu, D., Dunson, D., and Carin, L. Multi-task compressive sensing with dirichlet process priors. In *International conference on Machine learning*, pp. 768–775, 2008.

Ramachandran, Deepak and Amir, Eyal. Bayesian inverse reinforcement learning. In *20th Int. Joint Conf. Artificial Intelligence*, India, 2007.

Rasmussen, C. E. and Ghahramani, Z. Infinite Mixtures of Gaussian Process Experts. In *Advances in Neural Information Processing Systems 14*, 2002.

Schaal, S. and Atkeson, C. G. Constructive incremental learning from only local information. *Neural Comput.*, 10(8):2047–2084, 1998.

Shahbaba, Babak and Neal, Radford. Nonlinear models using dirichlet process mixtures. *J. Mach. Learn. Res.*, pp. 1829–1850, August 2009.

Taylor, G.W., Hinton, G.E., and Roweis, S.T. Modeling human motion using binary latent variables. *Advances in neural information processing systems*, 19:1345, 2007.

Wood, F., Grollman, D. H., Heller, K. A., Jenkins, O. C., and Black, M. Incremental nonparametric bayesian regression, 2008.

Yau, C. and Holmes, C. Hierarchical bayesian nonparametric mixture models for clustering with variable relevance determination. *Bayesian Anal*, 6:329–352, 2011.