in Advances in Neural Information Processing Systems 15
S. Becker, S. Thrun and K. Obermayer (eds.),
MIT Press (2003)

Exponential Family PCA for Belief Compression in POMDPs

Nicholas Roy
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
nickr@ri.cmu.edu

Geoffrey Gordon
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
ggordon@cs.cmu.edu

Abstract

Standard value function approaches to finding policies for Partially Observable Markov Decision Processes (POMDPs) are intractable for large models. The intractability of these algorithms is due to a great extent to their generating an optimal policy over the entire belief space. However, in real POMDP problems most belief states are unlikely, and there is a structured, low-dimensional manifold of plausible beliefs embedded in the high-dimensional belief space.

We introduce a new method for solving large-scale POMDPs by taking advantage of belief space sparsity. We reduce the dimensionality of the belief space by exponential family Principal Components Analysis [1], which allows us to turn the sparse, high-dimensional belief space into a compact, low-dimensional representation in terms of learned features of the belief state. We then plan directly on the low-dimensional belief features. By planning in a low-dimensional space, we can find policies for POMDPs that are orders of magnitude larger than can be handled by conventional techniques.

We demonstrate the use of this algorithm on a synthetic problem and also on a mobile robot navigation task.

1 Introduction

Large Partially Observable Markov Decision Processes (POMDPs) are generally very difficult to solve, especially with standard value iteration techniques [2, 3]. Maintaining a full value function over the high-dimensional belief space entails finding the expected reward of every possible belief under the optimal policy. However, in reality most POMDP policies generate only a small percentage of possible beliefs. For example, a mobile robot navigating in an office building is extremely unlikely to ever encounter a belief about its pose that resembles a checkerboard. If the execution of a POMDP is viewed as a trajectory inside the belief space, trajectories for most large, real world POMDPs lie on low-dimensional manifolds embedded in the belief space. So, POMDP algorithms that compute a value function over the full belief space do a lot of unnecessary work.

Additionally, real POMDPs frequently have the property that the belief probability distributions themselves are sparse. That is, the probability of being at most states in the world is zero. Intuitively, mobile robots and other real world systems have local uncertainty (which can often be multi-modal), but rarely encounter global uncertainty. Figure 1 depicts a mobile robot travelling down a corridor, and illustrates the sparsity of the belief space.

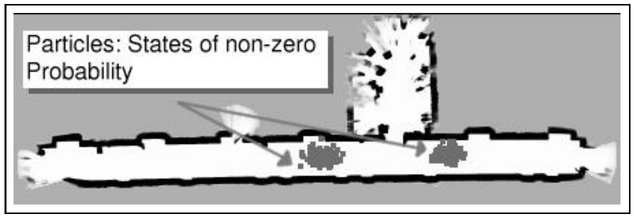


Figure 1: An example probability distribution of a mobile robot navigating in a hallway (map dimensions are 47m x 17m, with a grid cell resolution of 10cm). The white areas are free space, states where the mobile robot could be. The black lines are walls, and the dark gray particles are the output of the particle filter tracking the robot's position. The particles are located in states where the robot's belief over its position is non-zero. Although the distribution is multi-modal, it is still relatively compact: the majority of the states contain no particles and therefore have zero probability.

We will take advantage of these characteristics of POMDP beliefs by using a variant of a common dimensionality reduction technique, Principal Components Analysis (PCA). PCA is well-suited to dimensionality reduction where the data lies near a linear manifold in the higher-dimensional space. Unfortunately, POMDP belief manifolds are rarely linear; in particular, sparse beliefs are usually very non-linear. However, we can employ a link function to transform the data into a space where it does lie near a linear manifold; the algorithm which does so (while also correctly handling the transformed residual errors) is called Exponential Family PCA (E-PCA). E-PCA will allow us to find manifolds with only a handful of dimensions, even for belief spaces with thousands of dimensions.

Our algorithm begins with a set of beliefs from a POMDP. It uses these beliefs to find a decomposition of belief space into a small number of belief features. Finally, it plans over a low-dimensional space by discretizing the features and using standard value iteration to find a policy over the discrete beliefs.

2 POMDPs

A Partially Observable Markov Decision Process (POMDP) is a model given by a set of states $\mathcal{S} \in \{\vec{s}_1, \vec{s}_2, \dots \vec{s}_n\}$, actions $\mathcal{A} \in \{a_1, a_2, \dots, a_m\}$ and observations $\mathcal{Z} \in \{z_1, z_2, \dots, z_m\}$. Associated with these are a set of transition probabilities $T(\vec{s'}, a, \vec{s}) = p(\vec{s'} \mid \vec{s}, a)$ and observation probabilities $O(z, \vec{s}, a) = p(z \mid \vec{s}, a)$.

The objective of the planning problem is to find a policy that maximises the expected sum of future (possibly discounted) rewards of the agent executing the policy. There are a large number of value function approaches [2, 4] that explicitly compute the expected reward of every belief. Such approaches produce complete policies, and can guarantee optimality under a wide range of conditions. However, finding a value function this way is usually computationally intractable.

Policy search algorithms [3, 5, 6, 7] have met with success recently. We suggest that a large part of the success of policy search is due to the fact that it focuses computation on relevant belief states. A disadvantage of policy search, however, is that can be data-inefficient: many policy search techniques have trouble reusing sample trajectories generated from old policies. Our approach focuses computation on relevant belief states, but also allows us to use all relevant training data to estimate the effect of any policy.

Related research has developed heuristics which reduce the belief space representation. In particular, entropy-based representations for heuristic control [8] and full value-function planning [9] have been tried with some success. However, these approaches make strong assumptions about the kind of uncertainties that a POMDP generates. By performing principled dimensionality reduction of the belief space, our technique should be applicable to a wider range of problems.

3 Dimensionality Reduction

Principal Component Analysis is one of the most popular and successful forms of dimensionality reduction [10]. PCA operates by finding a set of feature vectors $U = \{u_1, \ldots, u_n\}$ that minimise the loss function

$$L(U, V) = ||X - UV||^2 \tag{1}$$

where X is the original data and V is the matrix of low-dimensional coordinates of X. This particular loss function assumes that the data lie near a linear manifold, and that displacements from this manifold are symmetric and have the same variance everywhere. (For example, i.i.d. Gaussian errors satisfy these requirements.)

Unfortunately, as mentioned previously, probability distributions for POMDPs rarely form a linear subspace. In addition, squared error loss is inappropriate for modelling probability distributions: it does not enforce positive probability predictions.

We use exponential family PCA to address this problem. Other nonlinear dimensionality-reduction techniques [11, 12, 13] could also work for this purpose, but would have different domains of applicability. Although the optimisation procedure for E-PCA may be more complicated than that for other models such as locally-linear models, it requires many fewer samples of the belief space. For real world systems such as mobile robots, large sample sets may be difficult to acquire.

3.1 Exponential family PCA

Exponential family Principal Component Analysis [1] (E-PCA) varies from conventional PCA by adding a link function, in analogy to generalised linear models, and modifying the loss function appropriately. As long as we choose the link and loss functions to match each other, there will exist efficient algorithms for finding U and V given X. By picking particular link functions (with their matching losses), we can reduce the model to an SVD.

We can use any convex function $F(z)$ to generate a matching pair of link and loss functions. The loss function which corresponds to F is

$$\sum_i [F(z_i) - y_i z_i + F^*(y_i)] \tag{2}$$

where F^* is defined so that the minimum over z of

$$F(z) - yz + F^*(y) \tag{3}$$

is always 0. (F^* is called the *convex dual* of F, and expression (3) is called a *generalised Bregman divergence* from z to y.)

The loss functions themselves are only necessary for the analysis; our algorithm needs only the link functions and their derivatives. So, we can pick the loss functions and differentiate to get the matching link functions; or, we can pick the link functions directly and not worry about the corresponding loss functions.

Each choice of link and loss functions results in a different model and therefore a potentially different decomposition of X. This choice is where we should inject our domain knowledge about what sort of noise there is in X and what parameter matrices U and V are a priori most likely. In our case the entries of X are the number of particles from a large sample which fell into a small bin, so a Poisson loss function is most appropriate. The corresponding link function is

$$\bar{X} = f(UV) = \exp(UV) \tag{4}$$

(taken component-wise) and its associated loss function is

$$L(U, V) = \exp(UV) - X \circ UV \tag{5}$$

where the "matrix dot product" $A \circ B$ is the sum of products of corresponding elements. It is worth noting that using the Poisson loss for dimensionality reduction is related to Lee and Seung's non-negative matrix factorization [14].

In order to find U and V, we compute the derivatives of the loss function with respect to U and V and set them to 0. The result is a set of fixed-point equations that the optimal parameter settings must satisfy:

$$U^{\mathrm{T}}(X - f(UV)) = 0 \tag{6}$$

$$(X - f(UV))V^{\mathrm{T}} = 0 \tag{7}$$

There are many algorithms which we could use to solve our optimality equations (6) and (7). For example, we could use gradient descent. In other words, we could add a multiple of $U^{\mathrm{T}}(X - f(Z))$ to V, add a multiple of $(X - f(Z))V^{\mathrm{T}}$ to U, and repeat until convergence. Instead we will use a more efficient algorithm due to Gordon [15]; this algorithm is based on Newton's method and is related to iteratively-reweighted least squares. We refer the reader to this paper for further details.

4 Augmented MDP

Given the belief features acquired through E-PCA, it remains to learn a policy. We do so by using the low-dimensional belief features to convert the POMDP into a tractable MDP. Our conversion algorithm is a variant of the Augmented MDP, or Coastal Navigation algorithm [9], using belief features instead of entropy. Table 1 outlines the steps of this algorithm.

1. Collect sample beliefs
2. Use E-PCA to generate low-dimensional belief features
3. Convert low-dimensional space into discrete state space $\mathcal{S}$
4. Learn belief transition probabilities $\mathcal{T}(s_i, a, s_j)$, and reward function $\mathcal{R}(s_i)$.
5. Perform value iteration on new model, using states $\mathcal{S}$, transition probabilities $\mathcal{T}$ and $\mathcal{R}$.

Table 1: Algorithm for planning in low-dimensional belief space.

We can collect the beliefs in step 1 using some prior policy such as a random walk or a most-likely-state heuristic. We have already described E-PCA (step 2), and value iteration (step 5) is well-known. That leaves steps 3 and 4.

The state space can be discretized in a number of ways, such as laying a grid over the belief features or using distance to the closest training beliefs to divide feature space into Voronoi regions. Thrun [16] has proposed nearest-neighbor discretization in high-dimensional belief space; we propose instead to use low-dimensional feature space, where neighbors should be more closely related.

We can compute the model reward function $\mathcal{R}(s_i)$ easily from the reconstructed beliefs.

$$R(b) = b \cdot R(s) \tag{8}$$

To learn the transition function, we can sample states from the reconstructed beliefs, sample observations from those states, and incorporate those observations to produce new belief states.

One additional question is how to choose the number of bases. One possibility is to examine the singular values of the U matrix after performing E-PCA, and use only the features that have singular values above some cutoff. A second possibility is to use a model selection technique such as keeping a validation set of belief samples and picking the basis size with the best reconstruction quality. Finally, we could search over basis sizes according to performance of the resulting policy.

5 Experimental Results

We tested our approach on two models: a synthetic 40 state world with idealised action and observations, and a large mobile robot navigation task. For each problem, we compared E-PCA to conventional PCA for belief representation quality, and compared E-PCA to some heuristics for policy performance. We are unable to compare our approach to conventional value function approaches, because both problems are too large to be solved by existing techniques.

5.1 Synthetic model

The abstract model has a two-dimensional state space: one dimension of position along a circular corridor, and one binary orientation. States $s_0 \ldots s_19$ inclusive correspond to one orientation, and states $s_19 \ldots s_40$ correspond to the other. The reward is at a known position along the corridor; therefore, the agent needs to discover its orientation, move to the appropriate position, and declare it has arrived at the goal. When the goal is declared the system resets (regardless of whether the agent is actually at the goal). The agent has 4 actions: `left`, `right`, `sense_orientation`, and `declare_goal`. The observation and transition probabilities are given by von Mises distributions, an exponential family distribution defined over $[-\pi : \pi)$. The von Mises distribution is the "wrapped" analog of a Gaussian; it accounts for the fact that the two ends of the corridor are connected, and because the sum of two von Mises variates is another von Mises variate, we can guarantees that the true belief distribution is always a von Mises distribution over the corridor for each orientation.

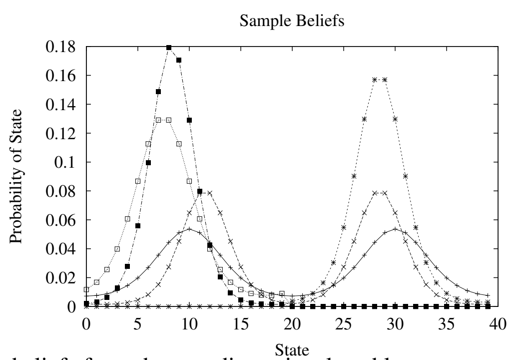
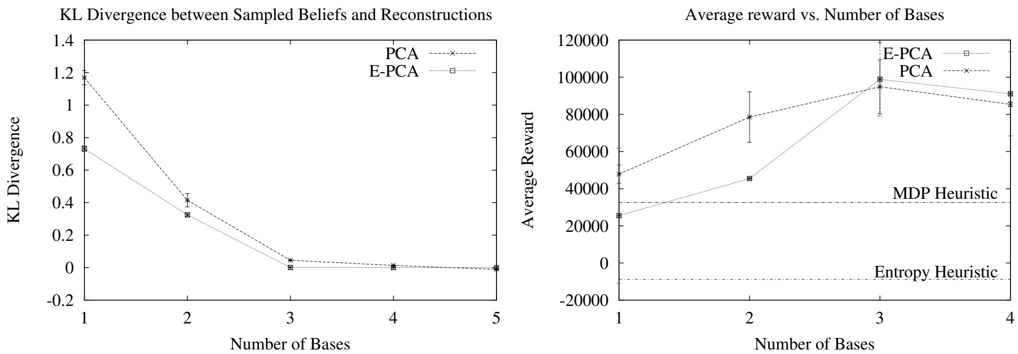


Figure 2: Some sample beliefs from the two-dimensional problem, generated from roll-outs of the model. Notice that some beliefs are bimodal, whereas others are unimodal in one half or the other of the state space.

Figure 2 shows some sample beliefs from this model. Notice that some of the beliefs are bimodal, but some beliefs have probability mass over half of the state space only—these unimodal beliefs follow the `sense_orientation` action.

Figure 3(a) shows the reconstruction performance of both the E-PCA approach and conventional PCA, plotting average KL-divergence between the sample belief and its reconstruction against the number of bases used for the reconstruction. PCA minimises squared error, while E-PCA with the Poisson loss minimises unnormalised KL-divergence, so it is no surprise that E-PCA performs better. We believe that KL-divergence is a more appropriate measure since we are fitting probabilities. Both PCA and E-PCA reach near-zero error at 3 bases (E-PCA hits zero error, since an l-basis E-PCA can fit an $(l-1)$-parameter exponential family exactly). This fact suggests that both decompositions should generate good policies using only 3 dimensions.



(a) Reconstruction error (b) Policy performance

Figure 3: (a) A comparison of the average KL divergence between the sample beliefs and their reconstructions, against the number of bases used, for 500 samples beliefs. (b) A comparison of policy performance using different numbers of bases, for 10000 trials. Policy performance was given by total reward accumulated over trials.

Figure 3(b) shows a comparison of the policies from different algorithms. The PCA techniques do approximately twice as a well as the naive Maximum Likelihood heuristic. This is because the ML-heuristic must guess its orientation, and is correct about half the time. In comparison, the Entropy heuristic does very poorly because it is unable to distinguish between a unimodal belief that has uncertainty about its orientation but not its position, and a bimodal belief that knows its position but not its orientation.

5.2 Mobile Robot Navigation

Next we tried our algorithm on a mobile robot navigating in a corridor, as shown in figure 1. As in the previous example, the robot can detect its position, but cannot determine its orientation until it reaches the lab door approximately halfway down the corridor. The robot must navigate to within 10cm of the goal and declare the goal to receive the reward. The map is shown in figures 1 and 4, and is 47m $\times$ 17m, with a grid cell resolution of 0.1m. The total number of unoccupied cells is 8250, generating a POMDP with a belief space of 8250 dimensions. Without loss of generality, we restrict the robot's actions to the forward and backward motion, and similarly simplified the observation model. The reward structure of the problem strongly penalised declaring the goal when the robot was far removed from the goal state.

The initial set of beliefs was collected by a mobile robot navigating in the world, and then post-processed using a noisy sensor model. In this particular environment, the laser data used for localisation normally gives very good localisation results; however, this will not be true for many real world environments [17].

Figure 4 shows a sample robot trajectory using the policy learned using 5 basis functions. Notice that the robot drives past the goal to the lab door in order to verify its orientation before returning to the goal. If the robot had started at the other end of the corridor, its orientation would have become apparent on its way to the goal.

Figure 5(a) shows the reconstruction performance of both the E-PCA approach and con-

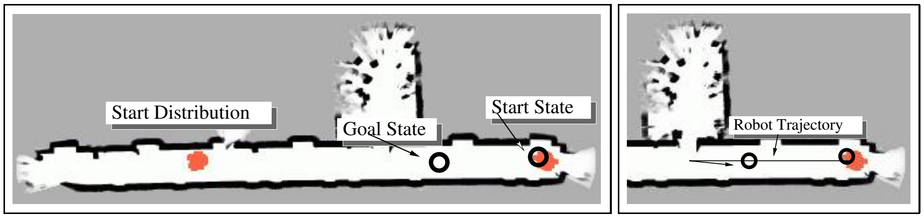
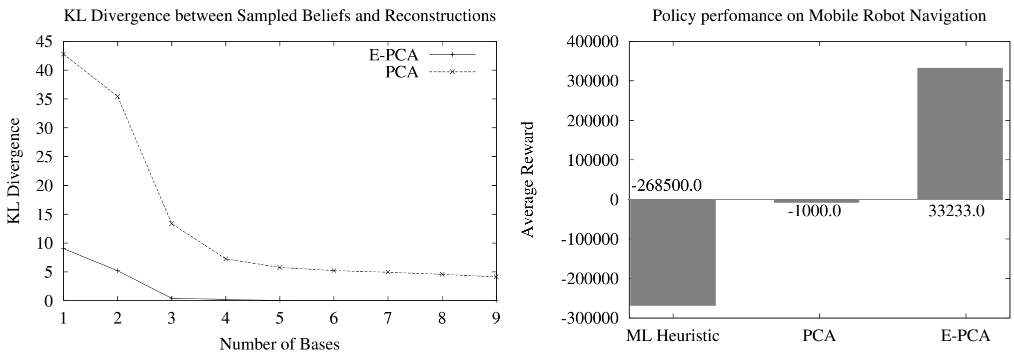


Figure 4: An example robot trajectory, using the policy learned using 5 basis functions. On the left are the start conditions and the goal. On the right is the robot trajectory. Notice that the robot drives past the goal to the lab door to localise itself, before returning to the goal.

ventional PCA, plotting average KL-divergence between the sample belief and its reconstruction against the number of bases used for the reconstruction.



(a) Reconstruction performance (b) Policy performance

Figure 5: (a) A comparison of the average KL divergence between the sample beliefs and their reconstructions against the number of bases used, for 400 samples beliefs for a navigating mobile robot.(b) A comparison of policy performance using E-PCA, conventional PCA and the Maximum Likelihood heuristic, for 1,000 trials.

Figure 5(b) shows the average policy performance for the different techniques, using 5 bases. (The number of bases was chosen based on reconstruction quality of E-PCA: see [15] for further details.) Again, the E-PCA outperformed the other techniques because it was able to model its belief accurately. The Maximum-Likelihood heuristic could not distinguish orientations, and therefore regularly declared the goal in the wrong place. The conventional PCA algorithm failed because it could not represent its belief accurately with only a few bases.

6 Conclusions

We have demonstrated an algorithm for planning for Partially Observable Markov Decision Processes by taking advantage of particular kinds of belief space structure that are prevalent in real world domains. In particular, we have shown this approach to work well on an abstract small problem, and also on a 8250 state mobile robot navigation task which is well beyond the capability of existing value function techniques.

The heuristic that we chose for dimensionality reduction was simply one of reconstruction error, as in equation 5: a reduction that minimises reconstruction error should allow near-optimal policies to be learned. However, it may be possible to learn good policies with even fewer dimensions by taking advantage of transition probability structure, or cost function structure. For example, for certain classes of problems, a loss function such as

$$L(U,V) = \sum_A \|T(a_i) \cdot F(UV) - T(a_i) \cdot X \circ UV\|^2 \quad (9)$$

would lead to a dimensionality reduction that maximises predictability. Similarly,

$$L(U,V) = \|V(F(UV)) - V(X \circ UV)\|^2 \tag{10}$$

where $V(\cdot)$ is some heuristic cost function (such as from a previous iteration of dimensionality reduction) would lead to a reduction that maximises ability to differentiate states with different values.

Acknowledgments

Thanks to Sebastian Thrun for many suggestions and insight. Thanks also to Drew Bagnell, Aaron Courville and Joelle Pineau for helpful discussion. Thanks to Mike Montemerlo for localisation code.

References

[1] M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal components analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, Cambridge, MA, 2002. MIT Press.

[2] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[3] Andrew Ng and Michael Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2000.

[4] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[5] Andrew Ng, Ron Parr, and Daphne Koller. Policy search via density estimation. In *Advances in Neural Information Processing Systems 12*, 1999.

[6] Jonathan Baxter and Peter Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proc. the 17th International Conference on Machine Learning*, 2000.

[7] J. Andrew Bagnell and Jeff Schneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Proceedings of the International Conference on Robotics and Automation*, 2001.

[8] Anthony R. Cassandra, Leslie Pack Kaelbling, and James A. Kurien. Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In *Proceedings of the IEEE/RSJ Interational Conference on Intelligent Robotic Systems (IROS)*, 1996.

[9] Nicholas Roy and Sebastian Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, pages 1043–1049, 1999.

[10] I. T. Joliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[11] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[12] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[13] S. T. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of local linear models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, Cambridge, MA, 2002. MIT Press.

[14] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[15] Geoffrey Gordon. Generalized^2 linear^2 models. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.

[16] Sebastian Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems 12*, 1999.

[17] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hhnel, C. Rosenberg, N. Roy, J. Schulte, , and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.