

Probabilistic Inference for Solving Markov Decision Processes

Toussaint, Storkey, ICML '06

Noam Siegel

Computer Science M.Sc. Seminar
Ben Gurion University

May 15, 2022

1. Prepare markers and eraser, erase the board.
2. Reset the timer.

Outline

1 Introduction

- Motivation & Prior work

- Main contribution

2 Research Problem

- Solving Markov Decision Processes

3 Research Plan

- Mixture of MDPs and likelihood

- An EM-algorithm for computing the optimal policy

- Relation to Policy Iteration

4 Experimental results

- Discrete maze

- Stochastic optimal control

5 Conclusion

1 Introduction

Motivation & Prior work

Main contribution


2 Research Problem

3 Research Plan


4 Experimental results

5 Conclusion

Motivation



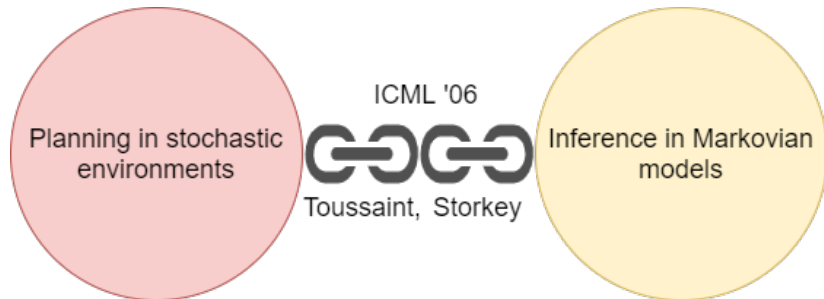
Planning in stochastic environments



Inference in Markovian models

1. Planning in stochastic environments, means solving a Markov Decision Process with a stochastic reward function.
2. Inference in Markovian models, means using Markovian assumptions to make estimating the values of probabilistic queries tractable.
3. similarities: both fields face the challenge of coping with very large state spaces spanned by multiple state variables, or working in continuous state spaces.

Motivation



1. The paper I am presenting to you bridges between the two fields, providing a solution to planning in stochastic environments using standard probabilistic inference techniques.

Prior work

- 1 *Bui et al. (2002)* used inference on Abstract Hidden Markov Models for policy recognition, *but not for computing an optimal policy*.
- 2 *Attias (2003)* got close to translating the problem of planning to a problem of inference. However, *the total time T had to be fixed* and the MAP action sequence that is proposed as a solution *is not optimal*.
- 3 *Verma and Rao (2006)* used inference to compute plans, but again T has to be fixed and the plan is not optimal.

1. existing ways to deal with challenges in each field: Factored MDPs or abstractions (in planning), approximate inference (inference).
2. describe and mention previous approaches to merging planning and inference, and mention their limitations.
3. Not optimal in the sense of maximizing an expected future reward

Main contribution

Contribution

- 1 Translate the problem of *maximizing the expected future return* exactly into a problem of *likelihood maximization in a latent variable model*, for arbitrary reward functions and episode lengths.

1. first contrib.: provide a framework that translates the problem of maximizing the expected future return exactly into a problem of likelihood maximization in a latent variable mixture model, for arbitrary rewards functions and without assuming a fixed time

Main contribution

Contribution

- 1 Translate the problem of *maximizing the expected future return* exactly into a problem of *likelihood maximization in a latent variable model*, for arbitrary reward functions and episode lengths.
- 2 Demonstrate the approach on discrete & continuous stochastic optimal control problems,

1. second contrib.: demonstrate the approach first on a discrete maze problem, then on a continuous stochastic optimal control problem.
2. In the discrete maze we will use exact inference in the E-step, and in the continuous case we will assume Gaussian belief state representations and use the unscented transform to handle non-linear dynamics.

Examples 1: Discrete maze

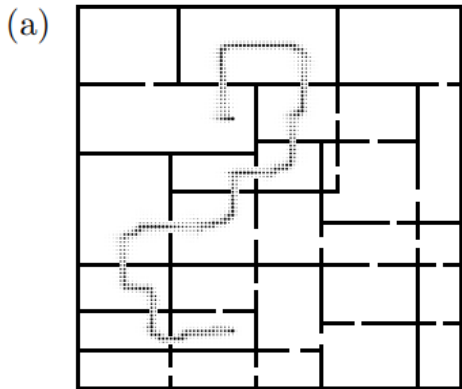


Figure 1: Posterior state-visiting-probabilities generated by our Probabilistic Inference Planner (PIP)

1. This figure is the posterior state-visiting-probabilities generated by our probabilistic inference planner (PIP) for a problem where rewards are given when some goal state is reached.
2. This is the image I want you to have in mind.

Markov Decision Processes

Definition (MDP)

state transition probability $P(x_{t+1} \mid a_t, x_t)$

reward probability $P(r_t \mid a_t; x_t), \quad r_t \in \{0, 1\}$

action probability $P(a_t \mid x_t; \pi), \quad \pi \text{ a parameter}$

1. This is an unrolled Dynamic Bayesian Network. Each node is a random variable, and each arrow denotes a causal relationship.
2. The graph encodes conditional independence relations between sets of variables. Examples $(a_1 \perp a_0 \mid x_1)$. These conditional independencies are what allow us to factor the probability distribution into fewer factors.
3. The r.v.s x and a can be either discrete or continuous
4. The reward variable r is, w.l.g. assumed to be binary, $r_t \in \{0, 1\}$
5. +++: w.l.g.: Assuming binary reward variables is sufficient to associate arbitrary reward expectations $P(r_t \mid a_t, x_t)$ in the interval $[0, 1]$ to states and actions, which is, modulo rescaling, the general case in Reinforcement Learning scenarios.

Markov Decision Processes

Definition (MDP)

state transition probability $P(x_{t+1} \mid a_t, x_t)$

reward probability $P(r_t \mid a_t; x_t), \quad r_t \in \{0, 1\}$

action probability $P(a_t \mid x_t; \pi), \quad \pi$ a parameter

Definition (Policy π)

The action probabilities are parameterized by a policy:

$$P(a_t \mid x_t = i; \pi) = \pi_{ai} \quad s.t. \sum_a \pi_{ai} = 1$$

1. So, let's remember the definition of a Markov Decision Process:
2. State transition probability is the same as usual, where the states and actions can be either continuous or discrete.
3. The reward probability is stochastic, and the reward is binary, without loss of generality. Because we can get every expected value between 0 and 1, and rescale if need.
4. The action probability is dependent on a parameter π which we call the Policy.
5. Throughout this paper we assume that the state transition and reward probabilities are given to us, i.e. known a priori.

Markov Decision Processes

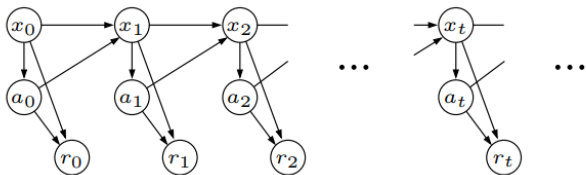


Figure 1. Dynamic Bayesian Network for a MDP. The x states denote the state variables, a the actions and r the rewards.

Definition (MDP)

state transition probability $P(x_{t+1} \mid a_t, x_t)$

reward probability $P(r_t \mid a_t; x_t), \quad r_t \in \{0, 1\}$

action probability $P(a_t \mid x_t; \pi), \quad \pi$ a parameter

1. The action probability $P(a_t \mid x_t; \pi)$ is parameterized by a policy π parameter such that $P(a_t \mid x_t = i; \pi) = \pi_{ai}$

Research problem

Definition (solving an MDP)

Solving an MDP means to find a parameter π of the graphical model in Figure 1 that maximizes the expected future return $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$, where $\gamma \in [0, 1]$ is a discount factor.

WRITE $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$ on the board

1. This definition is an equivalent definition to the standard definition, since reductions can be made in both directions.

Research problem

Definition (solving an MDP)

Solving an MDP means to find a parameter π of the graphical model in Figure 1 that maximizes the expected future return $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$, where $\gamma \in [0, 1]$ is a discount factor.

research problem

The problem is to *solve the MDP*, i.e. to find a policy that maximizes the expected future return.

WRITE $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$ on the board

1. The problem is to *solve the MDP*, i.e., to find a policy that maximizes the expected future return
2. The classical approach to solving MDPs is anchored in Bellman's equation, which simply reflects the recursive property of the future discounted return $R_T = \sum_{t=T}^{\infty} \gamma^t r_t = r_T + \gamma R_{T+1}$ and consequently of its expectation conditioned on the current state, $V_\pi = \sum_{j,a} P(j \mid a, i) P(a \mid i; \pi) [P(r_t = 1 \mid a, i) + \gamma V^\pi(j)]$.
3. Standard algorithms for computing value functions can be viewed as iterative schemes that converge towards the Bellman equation.

1 Introduction

2 Research Problem

3 Research Plan

Mixture of MDPs and likelihood

An EM-algorithm for computing the optimal policy

Relation to Policy Iteration

4 Experimental results

5 Conclusion

Mixture of MDPs

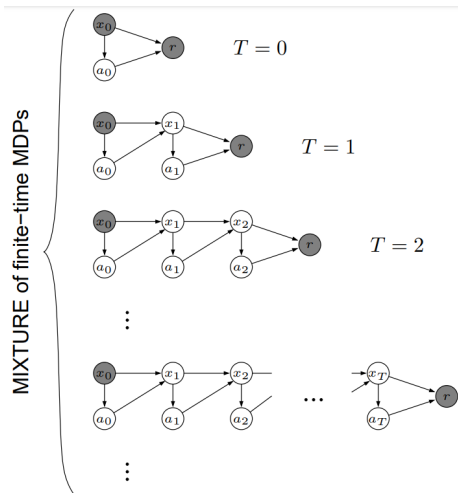


Figure 2. Mixture of finite-time MDPs.

DRAW Figure 2. on the board!

1. Our approach is to cast the problem of solving an MDP into a problem of optimizing the parameters of a graphical model with many hidden variables, namely all future states and actions.
2. The only observables are the current state and rewards
3. To achieve exact equivalence between solving the MDP and standard likelihood maximization we will need to define a likelihood that is proportional to the expected future return.
4. This is most easily done by considering a *mixture of finite-time MDPs*.
5. By a finite-time MDP we mean one which is limited in time by T , and which emits a reward variable only at the very final time step, as illustrated for various T in figure 2.

Representing the joint distribution $P(\mathcal{X})$

full joint for finite time MDP

$$P(r, x_{0:T}, a_{0:T} \mid T; \pi) =$$

$$P(r \mid a_T, x_T)P(a_0 \mid x_0; \pi)P(x_0) \cdot \prod_{t=1}^T P(a_t \mid x_t; \pi)P(x_t \mid a_{t-1}, x_{t-1})$$

1. This reads: The probability of seeing a reward r , a state sequence $x_{0:T}$, and an action sequence $a_{0:T}$, given an episode length T and a policy π , can be factored as the product of multiple conditional probabilities.
2. In this case we condition on the last variable (r), the first variables a_0, x_0 , and for each time step in between there's a constant number of variables conditioned at each time step, thanks to the Markov property.

Representing the joint distribution $P(\mathcal{X})$

full joint for finite time MDP

$$P(r, x_{0:T}, a_{0:T} \mid T; \pi) = \\ P(r \mid a_T, x_T) P(a_0 \mid x_0; \pi) P(x_0) \cdot \prod_{t=1}^T P(a_t \mid x_t; \pi) P(x_t \mid a_{t-1}, x_{t-1})$$

full joint for mixture of finite-time MDPs

$$P(r, x_{0:T}, a_{0:T}, T; \pi) = P(r, x_{0:T}, a_{0:T} \mid T; \pi) P(T)$$

1. This reads: using the chain rule, the joint probability of the observed reward, state sequence and action sequence, at a general time T , parameterized by the policy π , is a product of the joint for finite time MDP we just derived, and the prior probability we give to T . And what prior do we choose?

Representing the joint distribution $P(\mathcal{X})$

full joint for finite time MDP

$$P(r, x_{0:T}, a_{0:T} \mid T; \pi) = \\ P(r \mid a_T, x_T) P(a_0 \mid x_0; \pi) P(x_0) \cdot \prod_{t=1}^T P(a_t \mid x_t; \pi) P(x_t \mid a_{t-1}, x_{t-1})$$

full joint for mixture of finite-time MDPs

$$P(r, x_{0:T}, a_{0:T}, T; \pi) = P(r, x_{0:T}, a_{0:T} \mid T; \pi) P(T)$$

prior over the total time

$$P(T) = \gamma^T (1 - \gamma)$$

WRITE $P(T) = \gamma^T (1 - \gamma)$ on board

1. In this project we choose the prior over the total time, $P(T)$, such that the relative counts are proportional to the discounting.
2. Note that each finite-time MDP shares the same transition probabilities and is parameterized by the same policy π

Defining the likelihood

1. What we are trying to do is solve an MDP, which is to find a policy that maximizes the future reward. The approach here is to define a likelihood we would like to maximize, and by maximizing the likelihood we will also maximize the expected future reward.

Defining the likelihood

Definition (likelihood for a finite-time MDP)

$$L_T^\pi(i) = P(r = 1 \mid x_0 = i, T; \pi) = E\{r \mid x_0 = i, T; \pi\}$$

1. What we are trying to do is solve an MDP, which is to find a policy that maximizes the future reward. The approach here is to define a likelihood we would like to maximize, and by maximizing the likelihood we will also maximize the expected future reward.
2. For a sequence length T , the time-bounded MDP allows us to formulate a likelihood proportional to the expected future return for that run.

Defining the likelihood

Definition (likelihood for a finite-time MDP)

$$L_T^\pi(i) = P(r = 1 \mid x_0 = i, T; \pi) = E\{r \mid x_0 = i, T; \pi\}$$

Definition (likelihood for mixture of MDPs)

$$L^\pi(i) = P(r = 1 \mid x_0 = i; \pi) = \sum_T P(T) E\{r \mid x_0 = i, T; \pi\}$$

1. What we are trying to do is solve an MDP, which is to find a policy that maximizes the future reward. The approach here is to define a likelihood we would like to maximize, and by maximizing the likelihood we will also maximize the expected future reward.
2. For a sequence length T , the time-bounded MDP allows us to formulate a likelihood proportional to the expected future return for that run.
3. The mixture of time-bounded MDPs allows us a likelihood proportional to the expected future return for any run, regardless of the episode length.

Defining the likelihood

Definition (likelihood for a finite-time MDP)

$$L_T^\pi(i) = P(r = 1 \mid x_0 = i, T; \pi) = E\{r \mid x_0 = i, T; \pi\}$$

Definition (likelihood for mixture of MDPs)

$$L^\pi(i) = P(r = 1 \mid x_0 = i; \pi) = \sum_T P(T) E\{r \mid x_0 = i, T; \pi\}$$

Corollary

$$L_T^\pi(i) = (1 - \gamma) V^\pi(i)$$

1. What we are trying to do is solve an MDP, which is to find a policy that maximizes the future reward. The approach here is to define a likelihood we would like to maximize, and by maximizing the likelihood we will also maximize the expected future reward.
2. For a sequence length T , the time-bounded MDP allows us to formulate a likelihood proportional to the expected future return for that run.
3. The mixture of time-bounded MDPs allows us a likelihood proportional to the expected future return for any run, regardless of the episode length.
4. This likelihood is, for the discounted time prior $P(T) = \gamma^T(1 - \gamma)$, proportional to the expected discounted future return.
5. +++: Note that the expectation term $E\{r \mid x_0 = i, T; \pi\}$ here is exactly the same as the terms $E\{r_t \mid x_0 = i; \pi\}$ for $t = T$ in definition 2: we are taking the expectation w.r.t. a full probabilistic forward-sweep through the MDP, from time 0 to time T , given the policy π . All MDPs share the same transition probabilities

Theoretical Guarantee

Reminders

- 1 *Solving an MDP* means to find a parameter π of the graphical model in Figure 1 that maximizes the expected future return $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$.
- 2 The *likelihood for a mixture of MDPs* is given by
$$L^\pi(i) = P(r = 1 \mid x_0 = i; \pi) = \sum_T P(T) E\{r \mid x_0 = i, T; \pi\}$$
- 3 This implies that $L_T^\pi(i) = (1 - \gamma) V^\pi(i)$

1. Let us remember what we've seen so far.

Theoretical Guarantee

Reminders

- 1 *Solving an MDP* means to find a parameter π of the graphical model in Figure 1 that maximizes the expected future return $V^\pi(i) = E\{\sum_{t=0}^{\infty} \gamma^t r_t \mid x_0 = i; \pi\}$.
- 2 The *likelihood for a mixture of MDPs* is given by
$$L^\pi(i) = P(r = 1 \mid x_0 = i; \pi) = \sum_T P(T) E\{r \mid x_0 = i, T; \pi\}$$
- 3 This implies that $L_T^\pi(i) = (1 - \gamma) V^\pi(i)$

Theorem

(proved) Maximizing the likelihood in the mixture of finite-time MDPs is equivalent to solving the MDP.

1. Let us remember what we've seen so far.
2. Thus, we have established an important theoretical guarantee: there is an exact equivalence between the maximization of the likelihood and expected future return.
3. Note that considering the mixture of finite-time models has a second advantage: the finite-time property of every mixture component makes the E-step in the full mixture model rather simple and efficient, as detailed in the next section.

An EM-algorithm for computing the optimal policy

What are EM algorithms?

A class of algorithms consisting of two modes:

- 1 E-step: estimates missing variables.
- 2 M-step: optimizes parameters of model to best explain the data.

1. The EM algorithm (or family of algorithms) is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
2. Formulating the objective function in terms of a likelihood allows us to apply Expectation-Maximization to find optimal parameters (the policy π) of our model. All action and state variables (except for x_0 are hidden variables).
3. The E-step will, for a given π , compute posteriors over state-action sequences as well as T conditioned on $x_0 = A$ and $r = 1$
4. The M-step then adapts the model parameters π to optimize the expected likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step.
5. Conceptually, the E-step in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and save us from performing separate inference sweeps in all finite-time MDPs

An EM-algorithm for computing the optimal policy

What are EM algorithms?

A class of algorithms consisting of two modes:

- 1 E-step: estimates missing variables.
- 2 M-step: optimizes parameters of model to best explain the data.

1. The EM algorithm (or family of algorithms) is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
2. Formulating the objective function in terms of a likelihood allows us to apply Expectation-Maximization to find optimal parameters (the policy π) of our model. All action and state variables (except for x_0 are hidden variables).
3. The E-step will, for a given π , compute posteriors over state-action sequences as well as T conditioned on $x_0 = A$ and $r = 1$
4. The M-step then adapts the model parameters π to optimize the expected likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step.
5. Conceptually, the E-step in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and save us from performing separate inference sweeps in all finite-time MDPs

An EM-algorithm for computing the optimal policy

What are EM algorithms?

A class of algorithms consisting of two modes:

- 1 E-step: estimates missing variables.
- 2 M-step: optimizes parameters of model to best explain the data.

1. The EM algorithm (or family of algorithms) is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
2. Formulating the objective function in terms of a likelihood allows us to apply Expectation-Maximization to find optimal parameters (the policy π) of our model. All action and state variables (except for x_0 are hidden variables).
3. The E-step will, for a given π , compute posteriors over state-action sequences as well as T conditioned on $x_0 = A$ and $r = 1$
4. The M-step then adapts the model parameters π to optimize the expected likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step.
5. Conceptually, the E-step in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and save us from performing separate inference sweeps in all finite-time MDPs

An EM-algorithm for computing the optimal policy

What are EM algorithms?

A class of algorithms consisting of two modes:

- 1 E-step: estimates missing variables.
- 2 M-step: optimizes parameters of model to best explain the data.

E-step

For a given π , compute posteriors $P(T \mid x_0 = A, r = 1; \pi)$ and $P(x_{1:T}, a_{1:T} \mid x_0 = A, r = 1, T; \pi)$

1. The EM algorithm (or family of algorithms) is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
2. Formulating the objective function in terms of a likelihood allows us to apply Expectation-Maximization to find optimal parameters (the policy π) of our model. All action and state variables (except for x_0 are hidden variables).
3. The E-step will, for a given π , compute posteriors over state-action sequences as well as T conditioned on $x_0 = A$ and $r = 1$
4. The M-step then adapts the model parameters π to optimize the expected likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step).
5. Conceptually, the E-step in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and save us from performing separate inference sweeps in all finite-time MDPs

An EM-algorithm for computing the optimal policy

What are EM algorithms?

A class of algorithms consisting of two modes:

- 1 E-step: estimates missing variables.
- 2 M-step: optimizes parameters of model to best explain the data.

E-step

For a given π , compute posteriors $P(T \mid x_0 = A, r = 1; \pi)$ and $P(x_{1:T}, a_{1:T} \mid x_0 = A, r = 1, T; \pi)$

M-step

Adapt parameters π to optimize $V^\pi(A)$

1. The EM algorithm (or family of algorithms) is an iterative approach that cycles between two modes. The first mode attempts to estimate the missing or latent variables, called the estimation step or E-step. The second mode attempts to optimize the parameters of the model to best explain the data, called the maximization-step or M-step.
2. Formulating the objective function in terms of a likelihood allows us to apply Expectation-Maximization to find optimal parameters (the policy π) of our model. All action and state variables (except for x_0 are hidden variables).
3. The E-step will, for a given π , compute posteriors over state-action sequences as well as T conditioned on $x_0 = A$ and $r = 1$
4. The M-step then adapts the model parameters π to optimize the expected likelihood (expectations then taken w.r.t. the posteriors calculated in the E-step).
5. Conceptually, the E-step in this Markovian model is straight-forward. However, the special structure of the finite-time MDPs will allow for certain simplifications and save us from performing separate inference sweeps in all finite-time MDPs

E-step: forward-backward in all MDPs synchronously

Simplifying notation

- 1 $p(j \mid a, i) = P(x_{t+1} = j \mid a_t = a, x_t = i)$
- 2 $p(j \mid i; \pi) = P(x_{t+1} = j \mid x_t = i; \pi) = \sum_a p(j \mid a, i) \pi_{ai}$

1. Since we assume the transition probabilities to be stationary, we may use the simpler notations.

E-step: forward-backward in all MDPs synchronously

Forward Propagation

$$\begin{aligned}\alpha_0(i) &= \delta_{i=A} \\ \alpha_t(i) &= P(x_t = i \mid x_0 = A; \pi) \\ &= \sum_j p(i \mid j; \pi) \alpha_{t-1}(j)\end{aligned}$$

1. We will only show the algorithm for the discrete case, but I promise you that the EM algorithm works also for the continuous case. Examples in the paper.
2. Specifically, for the discrete case, the algorithm used is called the forward-backward algorithm. It is a form of belief propagation through message passing.
3. In the E-step, we consider a fixed given policy π and all the quantities we compute depend on π even if not explicitly annotated.
4. For a single MDP of finite time T we perform a forward and backward propagation as shown. α_t is the "location" belief state at time t , i.e., the probability distribution over states, conditioned on the initial state and parameterized by policy π).

E-step: forward-backward in all MDPs synchronously

Forward Propagation

$$\alpha_t(i) = P(x_t = i \mid x_0 = A; \pi)$$

$$\alpha_0(i) = \delta_{i=A}$$

$$= \sum_j p(i \mid j; \pi) \alpha_{t-1}(j)$$

Backward Propagation (temp.)

$$\tilde{\beta}_t(i) = P(r = 1 \mid x_t = i; \pi)$$

$$\tilde{\beta}_T(i) = \hat{\beta}(i)$$

$$= \sum_j p(j \mid i; \pi) \tilde{\beta}_{t+1}(j)$$

Where

$$\hat{\beta}(i) = P(r = 1 \mid x_T = i; \pi) = \sum_a P(r = 1 \mid a_T = a, x_T = i) \pi_{ai}$$

1. We will only show the algorithm for the discrete case, but I promise you that the EM algorithm works also for the continuous case. Examples in the paper.
2. Specifically, for the discrete case, the algorithm used is called the forward-backward algorithm. It is a form of belief propagation through message passing.
3. In the E-step, we consider a fixed given policy π and all the quantities we compute depend on π even if not explicitly annotated.
4. For a single MDP of finite time T we perform a forward and backward propagation as shown. α_t is the "location" belief state at time t , i.e., the probability distribution over states, conditioned on the initial state and parameterized by policy π).
5. β_t is the reward belief state, i.e. we assume that at time t we are at state x_t , and we ask what is the probability of observing a reward, parameterized by policy π . $\hat{\beta}$ is the starting "seed" for the process. And then we go backward one time step at a time, using β_{t+1} to calculate β_t .
6. We find that all the α -quantities do not depend on T in any way, i.e., they are valid for all MDPs of any finite time T . This is not true for the $\tilde{\beta}$ -quantities when defined as above.

E-step: forward-backward in all MDPs synchronously

Backward Propagation (temp.)

$$\begin{aligned}\tilde{\beta}_T(i) &= \hat{\beta}(i) \\ \tilde{\beta}_t(i) &= P(r = 1 \mid x_t = i; \pi) \\ &= \sum_j p(j \mid i; \pi) \tilde{\beta}_{t+1}(j)\end{aligned}$$

Where

$$\hat{\beta}(i) = P(r = 1 \mid x_T = i; \pi) = \sum_a P(r = 1 \mid a_T = a, x_T = i) \pi_{ai}$$

Backward Propagation (corrected)

$$\begin{aligned}\beta_0(i) &= \hat{\beta}(i) \\ \beta_\tau(i) &= P(r = 1 \mid x_{T-\tau} = i; \pi) \\ &= \sum_j p(j \mid i; \pi) \beta_{\tau-1}(j)\end{aligned}$$

1. β_t is the reward belief state, i.e. we assume that at time t we are at state x_t , and we ask what is the probability of observing a reward, parameterized by policy π . $\hat{\beta}$ is the starting "seed" for the process. And then we go backward one time step at a time, using β_{t+1} to calculate β_t .
2. We find that all the α -quantities do not depend on T in any way, i.e., they are valid for all MDPs of any finite time T . This is not true for the $\tilde{\beta}$ -quantities when defined as above.
3. However, we can use a simple trick, namely define the β 's to be indexed backward in time (with the 'time-to-go' τ), and get:
4. Defined in that way, all β -quantities do indeed not depend on T . For a specific MDP of finite time T , setting $\tau = T - t$ would allow us to retrieve the traditional forward-indexed $\tilde{\beta}$ -quantities.
5. This means that we can perform α - and β - propagation in parallel, incrementing t and τ synchronously, and can retrieve the α 's and β 's for all MDPs of any finite time T . Although we introduce a mixture of MDPs we only have to perform a single forward and backward sweep. This procedure is, from the point of view of ordinary HMMs, quite unusual - it is possible because in our specific setup we only condition on the very first ($x_0 = A$) and very last state ($r = 1$).

M-step: the policy update

Definition (expected complete log-likelihood)

$$Q(\pi^*, \pi) = \sum_T \sum_{x_{0:T}, a_{0:T}} P(x_{0:T}, a_{0:T}, T \mid r = 1; \pi)$$

1. The standard M-step in an EM-algorithm maximizes the *expected complete log-likelihood* w.r.t. the new parameters π^* , where expectations over the latent variables ($T, x_{0:T}, a_{0:T}$) were taken w.r.t. the posterior given the old parameters π .

M-step: the policy update

Definition (expected complete log-likelihood)

$$Q(\pi^*, \pi) = \sum_T \sum_{x_{0:T}, a_{0:T}} P(x_{0:T}, a_{0:T}, T \mid r = 1; \pi)$$

Fact

Maximizing $Q(\pi^*, \pi)$ w.r.t. π^* is achieved by setting
 $\pi_{ai}^* = P(a_t = a \mid x_t = i, r = 1; \pi)$

1. The standard M-step in an EM-algorithm maximizes the *expected complete log-likelihood* w.r.t. the new parameters π^* , where expectations over the latent variables ($T, x_{0:T}, a_{0:T}$ were taken w.r.t. the posterior given the old parameters π .
2. In our case, in strong analogy to the standard HMM case, this turns out to assign the new parameters to the posterior action probability

M-step: the policy update

Definition (expected complete log-likelihood)

$$Q(\pi^*, \pi) = \sum_T \sum_{x_{0:T}, a_{0:T}} P(x_{0:T}, a_{0:T}, T \mid r = 1; \pi)$$

Fact

Maximizing $Q(\pi^*, \pi)$ w.r.t. π^* is achieved by setting

$$\pi_{ai}^* = P(a_t = a \mid x_t = i, r = 1; \pi)$$

However

exploiting the structure of the MDP, we can also write the likelihood as

$$P(r = 1 \mid x_0 = i; \pi) = \sum_{aj} P(r = 1; a_t = a, x_t = j; \pi^*) \pi_{aj}^* \\ \cdot P(x_t = j \mid x_0 = i; \pi^*)$$

1. The standard M-step in an EM-algorithm maximizes the *expected complete log-likelihood* w.r.t. the new parameters π^* , where expectations over the latent variables ($T, x_{0:T}, a_{0:T}$ were taken w.r.t. the posterior given the old parameters π .
2. In our case, in strong analogy to the standard HMM case, this turns out to assign the new parameters to the posterior action probability
3. Maximizing this expression w.r.t. π_{aj}^* can be done separately for every j and is thus independent of $P(x_t = j \mid x_0 = i; \pi^*)$ and thereby also independent of t because $P(r = 1 \mid a_t = a, x_t = j; \pi^*)$ is so.

M-step: the policy update

However

exploiting the structure of the MDP, we can also write the likelihood as

$$P(r = 1 \mid x_0 = i; \pi) = \sum_{aj} P(r = 1; a_t = a, x_t = j; \pi^*) \pi_{aj}^* \cdot P(x_t = j \mid x_0 = i; \pi^*)$$

$$\pi_{ai}^* = \delta_{a=a^*(i)} \quad \alpha^*(i) = \arg \max_a P(r = 1 \mid a_t = a, x_t = i; \pi)$$

1. The standard M-step in an EM-algorithm maximizes the *expected complete log-likelihood* w.r.t. the new parameters π^* , where expectations over the latent variables ($T, x_{0:T}, a_{0:T}$ were taken w.r.t. the posterior given the old parameters π .
2. In our case, in strong analogy to the standard HMM case, this turns out to assign the new parameters to the posterior action probability
3. Maximizing this expression w.r.t. π_{aj}^* can be done separately for every j and is thus independent of $P(x_t = j \mid x_0 = i; \pi^*)$ and thereby also independent of t because $P(r = 1 \mid a_t = a, x_t = j; \pi^*)$ is so.
4. Using the E-step we can approximate the first term and maximize $\sum_a P(r = 1 \mid a_t = a, x_t = j; \pi) \pi_{aj}^*$ via:
5. where $a^*(i)$ maximizes the action-conditioned likelihood. Given the relations between equations 15&16, the main difference to the standard M-step is that this update is much greedier and converges faster in the MDP case. We will use this update rather than the previous one shown in the experiments.
6. This is the standard policy update in Policy Iteration (Sutton & Barto, 1998)

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$
- 2 $V^{pi}(i) = \frac{1}{1-\gamma} \sum_T P(T) \beta_T(i).$

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$
- 2 $V^{pi}(i) = \frac{1}{1-\gamma} \sum_T P(T) \beta_T(i).$
- 3 Hence, *the E-step performs a policy evaluation.*

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$
- 2 $V^{pi}(i) = \frac{1}{1-\gamma} \sum_T P(T) \beta_T(i).$
- 3 Hence, *the E-step performs a policy evaluation.*
- 4 Additionally, *it yields the time, state and action posteriors.*

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$
- 2 $V^{pi}(i) = \frac{1}{1-\gamma} \sum_T P(T) \beta_T(i).$
- 3 Hence, *the E-step performs a policy evaluation.*
- 4 Additionally, *it yields the time, state and action posteriors.*

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

E-step: Policy Evaluation

- 1 $\beta_\tau(i) \propto (V^\pi(i) \text{ of the MDP of time } T = \tau).$
- 2 $V^{pi}(i) = \frac{1}{1-\gamma} \sum_T P(T) \beta_T(i).$
- 3 Hence, *the E-step performs a policy evaluation.*
- 4 Additionally, *it yields the time, state and action posteriors.*

M-step: Policy Update

- 1 Maximizing the Q-function w.r.t. the action a and state i .

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

Thus,

the EM-algorithm using exact inference and belief representation is effectively equivalent to Policy Iteration but computes the necessary quantities in a different way.

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

Relation to Policy Iteration

Thus,

the EM-algorithm using exact inference and belief representation is effectively equivalent to Policy Iteration but computes the necessary quantities in a different way.

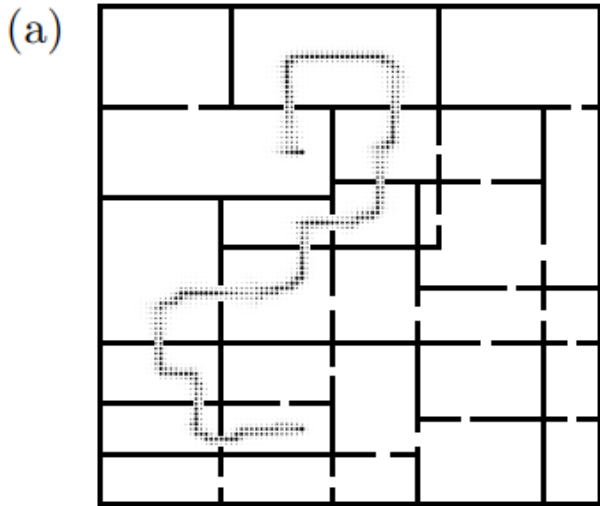
However,

when using approximate inference or belief representations

1. The β -quantities computed during backward-propagation are actually the value function for a single MDP of finite time T .
2. More precisely, comparing eq (9) with the reward likelihood (3) for the MDP of finite time T and the and the reward definition 2.1 of the value function, we have that *the β -quantity with τ time to go at the (i th state) is proportional to the Value function of the MDP at time $T=\tau$.*
3. Accordingly, the full value function is the mixture of the β 's, when a discount time prior is chosen
4. Additionally, yields ... which have no traditional analogue.

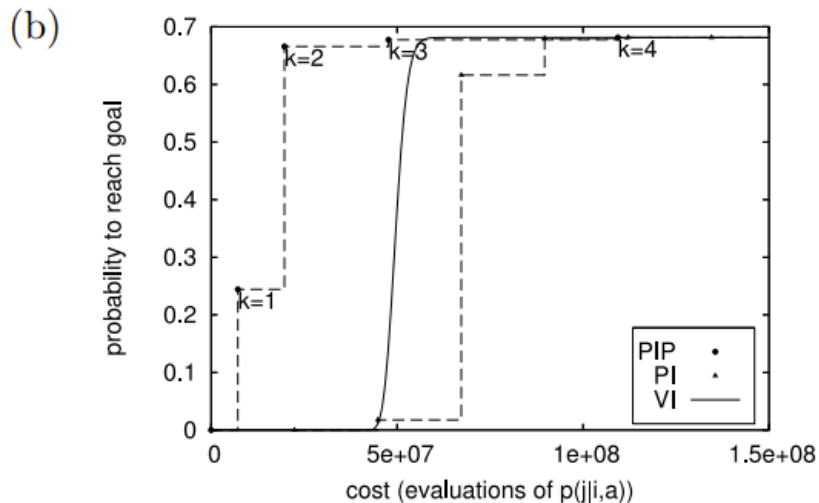
- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ 🔍 ↺

Discrete maze



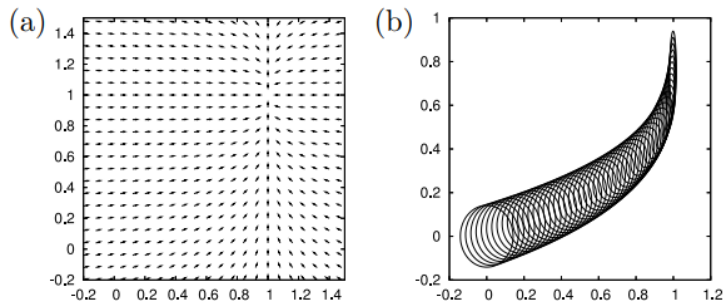
1. We tested our probabilistic inference planning (PIP) algorithm on a discrete maze of size 100 X 100 and compared it to standard value iteration (VI) and policy iteration (PI).
2. Walls of the maze are considered to be trap states (leading to unsuccessful trials) and actions (north, south, east, west, stay) are highly noisy in that with a probability of 0.2 they lead to random transitions. In the experiment we chose a uniform time prior (discount factor $\gamma = 1$ and iterated the policy update $k = 5$ times).
3. To increase computational efficiency we exploited that the algorithm explicitly calculates posteriors which can be used to prune unnecessary message passings as explained in appendix A.
4. For policy evaluation in PI we performed 100 iterations of standard value function updates.
5. Figure 3(a) displays the posterior state visiting probabilities generated by our probabilistic inference planner (PIP) for a problem where rewards are given when some goal state is reached.

Discrete maze



1. Computational costs are measure by the number of evaluations of the environment $p(j | i, a)$ needed during the planning procedure. Figure 3(b) displays the probability of reaching the goal $P(r = 1 | x_j = A; \pi)$ against these costs for the same two start/goal state configurations.
2. Note that for PIP (and PI) we can give this information only after a complete E- and M-step cycle (policy evaluation and update) which are the discrete dots (triangles) in the graph.
3. The graph also displays the curve for VI, where the currently calculated value V_A of the start state (which converges to $P(B | A)$ for the optimal policy) is plotted against how often VI evaluated $p(j | i, a)$.
4. In contrast to VI and PI, the PIP algorithm takes considerable advantage of knowing the start state in this planning scenario: the forward propagation allows for the pruning and the early decision on cutoff times of the E-step as described in Appendix A. It should thus not be a surprise and not overstated that PIP is significantly more efficient in this specific scenario. Certainly, some kind of forward propagations could also be introduced for VI or PI to achieve similar efficiency. Nonetheless, our approach provides a principled way of pruning by exploiting the computation of proper posteriors.

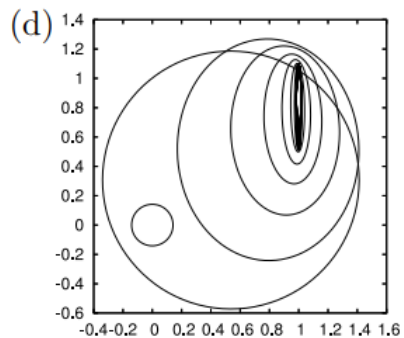
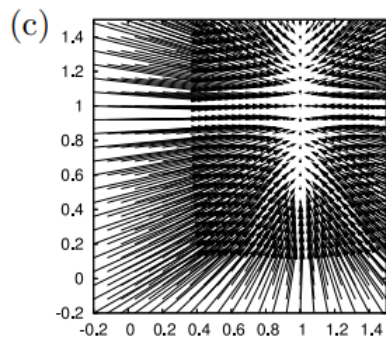
Stochastic optimal control - 'walker'



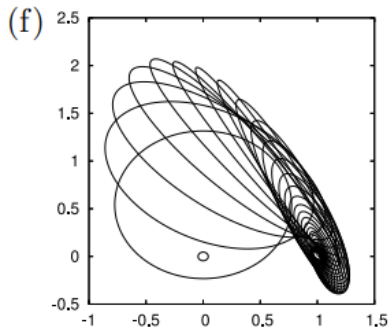
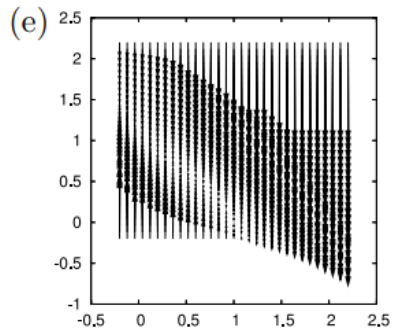
- $P(x' | u, x) = \mathcal{N}(x', \phi(u, x), Q + (|u|/\mu)^2 I)$ is *transitions*.
- $\alpha_0(x) = \mathcal{N}(x, (0, 0), .01 I)$ is *start-state*.
- $P(r = 1 | x) = \mathcal{N}(x, (1, 1), \text{diag}(.0001, .1))$ is *goal*.
- $\phi(u, x) = x + .1u$ is *control-law*.

1. We address the problem of *stochastic optimal control* in the case of a *continuous state and control space*. (both continuous state and continuous control).
2. Figure 4ab shows the learned control policy π and the forward simulation given this policy by displaying the covariance ellipses for $\alpha_{0:T}(x)$ after $k = 3$ EM iterations.
3. *What we find is a control policy that reduces errors in x -dimension more strongly than in y -direction, leading to a tangential approach.*
4. As a transition model we assume (eq). where $\phi(u, x)$ is a non-linear function, Q is a constant noise covariance, and we introduced a parameter μ for an additional noise term that is squared in the control signal.
5. We assume Gaussian belief states as representations for α 's and β 's and propagate forward-backward.
6. This goal region is heavily skewed in that rewards depend more on the precision in the x -dimension than the y -dimension.
7. When choosing $\mu = 0$, (no control-dependent noise), the optimal control policy will try to jump directly to the goal $(1, 1)$. Hence, we first consider the solution when manually constraining the norm $|u|$ to be small ('walker')
8. This example shows that the framework naturally allows to

Stochastic optimal control - 'golfer'



Stochastic optimal control - 'phase space'



1 Introduction

2 Research Problem

3 Research Plan

4 Experimental results

5 Conclusion

Conclusion

We present a model that translates the problem of planning into a problem of probabilistic inference.

1. The key step to our approach is to introduce a mixture of finite-time MDPs as a model that is equivalent to the original time-unbounded MPD but has two advantages: it allows us to formulate a likelihood proportional to the expected future return, and inference in the mixture model can efficiently be realized with a single synchronous forward-backward propagation without having to fix a finite total time in advance.

Conclusion

We present a model that translates the problem of planning into a problem of probabilistic inference.

Main contributions

- 1 we do not have to fix a total time
- 2 likelihood maximization is equivalent to maximization of the expected future return

1. The key step to our approach is to introduce a mixture of finite-time MDPs as a model that is equivalent to the original time-unbounded MPD but has two advantages: it allows us to formulate a likelihood proportional to the expected future return, and inference in the mixture model can efficiently be realized with a single synchronous forward-backward propagation without having to fix a finite total time in advance.

Conclusion

We present a model that translates the problem of planning into a problem of probabilistic inference.

Main contributions

- 1 we do not have to fix a total time
- 2 likelihood maximization is equivalent to maximization of the expected future return

We can compute posteriors over actions, states, and the total time.

1. The key step to our approach is to introduce a mixture of finite-time MDPs as a model that is equivalent to the original time-unbounded MPD but has two advantages: it allows us to formulate a likelihood proportional to the expected future return, and inference in the mixture model can efficiently be realized with a single synchronous forward-backward propagation without having to fix a finite total time in advance.

Conclusion

We present a model that translates the problem of planning into a problem of probabilistic inference.

Main contributions

- 1 we do not have to fix a total time
- 2 likelihood maximization is equivalent to maximization of the expected future return

We can compute posteriors over actions, states, and the total time.

The full variety of existing inference techniques can be applied to solving MDPs.

1. The key step to our approach is to introduce a mixture of finite-time MDPs as a model that is equivalent to the original time-unbounded MPD but has two advantages: it allows us to formulate a likelihood proportional to the expected future return, and inference in the mixture model can efficiently be realized with a single synchronous forward-backward propagation without having to fix a finite total time in advance.

End

Thank You