Radboud University Nijmegen

Faculty of Science

Modelling Non-Markovian Properties

-

Thesis MSc Computing Science

Author:
Serena Rietbergen

Supervisor:
dr. Nils Jansen

Second reader:
- -

-

Contents

Abstract

hoi hier moet iets

Chapter 1

Introduction

Motivating Example

Problem Formulation

Given a POMDP with a history-based reward function, obtain a policy that maximizes the expected reward.

Contribution

Structure

Chapter 2

Preliminaries

Set Theory

Let S be any countable set, then $|S|$ denotes the cardinality. We let S^* and S^ω denote the set of finite and infinite sequences over S, respectively. For a sequence $\pi \in S^*$ we can denote the length by $|\pi|$.

Probability Theory

For any countable set S we can define a *discrete probability distribution* as $\psi : S \to [0, 1]$ where $\sum_{s \in S} \psi(s) = 1$. The set of all possible probability distributions over S is denoted as $\Pi(S)$. We denote the support of a *probability distribution* as $supp(\psi) = \{s \in S \mid \psi(s) > 0\}$.

TO WRITE: random variable, expected value

Chapter 3

Background

Markov decision processes

TO WRITE: explain in words what an mdp is

Definition 3.1 (MDP)**.** A Markov decision process is a tuple $M = (S, s_I, Act, P)$ where

- S, the finite set of states;
- s_I, the initial state;
- Act, the finite set of actions;
- $T : S \times A \to \Pi(S)$, the probabilistic transition function.

Note that given $s \in S, a \in Act$, we assign a probability distribution over S through $T(s, a)$. To obtain the probability of ending up in a certain state s' when starting in state s and performing action a, we simply calculate $T(s, a, s')$ which we obtain through $T(s, a)(s')$.

The *available actions* for a state s are given by $Act(s) = \{a \in Act \mid \exists s' \in S : T(s, a, s') > 0\}$. We can give the *possible successors* of state s in a similar matter through $Succ(s) = \{s \in S | \exists a \in Act : T(s, a, s') > 0\}$.

A finite *trajectory* or *run* π of an MDP is realization of the stochastic process performed by the MDP denoted by the finite sequence $s_1 a_1 s_2 a_2 \dots s_{n-1} a_{n-1} s_n \in (S \times Act)^* \times S$. To obtain the last state of a trajectory we can use the following

$$last(\pi) = last(s_1 a_1 s_2 a_2 \dots s_{n-1} a_{n-1} s_n) = s_n$$

TO WRITE: Set notation for finite trajectories

Reward function

We can extend MDPs with a *reward function* R which assign a reward - usually in $\mathbb{R}$ for taking a certain action a in a state s.

TO WRITE: Intuitive explanation for reward function - including cost function

Let us look at *Markovian reward functions*, which can determine a reward based on the current state, action and obtained state, independent of its history. The most

conventional notation is $R : S \times Act \to \mathbb{R}$, where we consider the current state and the taken action. Another possible definition is $R : S \times Act \times S \to \mathbb{R}$, where in $R(s, a, s')$ we consider the specific transition from s to s' by using action a, or $R : Act \to \mathbb{R}$ where in $R(a)$ we only consider the action taken.

TO WRITE: Real life example of markovian reward function

A reward function wich is dependent of its history is called a *Non-Markovian reward function*. There are a number of different reward functions possible

- $R : Act^* \to \mathbb{R}$ - which only needs all the finite actions used, or;
- $R : S^* \to \mathbb{R}$ - which only looks at the finite states visited, or;
- $R : (S \times Act)^* \to \mathbb{R}$ - which looks at the finite (sub)trajectory without the last state, or;
- $R : (S \times Act)^* \times S \to \mathbb{R}$ - which looks at the finite (sub)trajectory.

All of these reward functions can be extended to the infinite sequences.

infinite needed?

The reward function we will be using is the Non-Markovian reward function which looks at trajectories of specific length k, namely $R_k : (S \times Act)^k \to \mathbb{R}$.

TO WRITE: Increasing k creates increased reward

Policy

As stated above, we use reward functions over a MDP to usually argue over an optimized expected reward. After retrieving such an optimum, the question remains on how to actually obtain this value. We wish to know what strategy to apply path to take to obtain this value. For this we use strategies, or often called policies.

Definition 3.2. A policy for a MDP M is a function $\sigma : (S \times Act)^* \times S \to \Pi(Act)$, which maps a trajectory π to a probability distribution over all actions.

We call a policy *memoryless* if the function only considers $last(\pi)$. The notation $\sigma_k : (S \times Act)^{k-1} \times S \to \Pi(S)$, gives a probability distribution when given the trajectory of length k, which includes the k visited states and $k - 1$ actions. $\sigma_k(\pi, a) = \sigma_k(\pi)(a)$ gives the probability of using action a after the trajectory π.

TO WRITE: induced markov chain for removing non-determinism??

Partial observability

TO WRITE: Introduce pomdp

Definition 3.3 (POMDP)**.** A partially observable Markov decision process (POMDP) is a tuple $\mathcal{M} = (M, Z, O)$ where

- $M = (S, s_I, Act, T)$, the hidden MDP;
- Z, the finite set of observations;
- $O : S \to Z$, the observation function.

Without loss of generality we assume that states with the same observations have the same set of available actions, thus $O(s_1) = O(s_2) \Rightarrow Act(s_1) = A(s_2)$.

Since the actual states in a trajectory of the hidden MDP are not visible to the observes, we argue about an *obversed trajectory* of the POMDP $\mathcal{M}$. This is not consist of a sequence of states and actions, but instead a sequence of observations are actions, thus an element of $(Z \times Act)^* \times Z$. The set of all possible finite observed trajectories of will be denoted as $ObsSeq^{\mathcal{M}}$.

We can argue about the observed trajectory through the observation function, which will be extended over trajectories, like so

$$O(\pi) = O(s_1 a_1 s_2 a_2 \dots s_{n-1} a_{n-1} s_n) = O(s_1) a_1 O(s_2) a_2 \dots O(s_{n-1}) a_{n-1} O(s_n)$$

Policy

Definition 3.4. An observation-based strategy of a POMDP $\mathcal{M}$ is a function $\sigma : ObsSeq^{\mathcal{M}} \to \Pi(Act)$ such that $supp(\sigma(O(\pi))) \subseteq Act(last(\pi))\ \forall \pi \in (S \times Act)^* \times S$.

Chapter 4

Reward Finite State Controllers

4.1 Definition

Idea: transform the history-based reward function into something more tangible. Problem: we can only work with the observations, since we are working with a pomdp.

Based on the history-based reward function $R_k : (S \times Act)^k \to \mathbb{R}$ of a POMDP $\mathcal{M}$, we build a finite-state controller that mimics its behavior.

Definition 4.1. A reward finite-state controller (R-FSC) for a reward function of a POMDP $\mathcal{M}$ is a tuple $\mathcal{F} = (N, n_I, \delta, \mathcal{R})$ where

- N, the finite set of memory nodes;
- n_I, the initial memory node;
- $\delta : N \times Z \times Act \to \Pi(N)$, the memory update;
- $\mathcal{R} : N \times Z \times Act \times N \to \mathbb{R}$, the reward update.

TO WRITE: explain $\mathcal{R}$

We use the notation $\delta(n, o, a, n')$ for the probability of ending up in the memory node n' starting in the memory node n, showing observation o and performing action a. Thus, $\delta(n, o, a, n') = \delta(n, o, a)(n')$.

When working with the sequence $s_0 a_0 s_1 a_1 \dots s_{k-1} a_{k-1} s_k \in (S \times Act)^k \times S$ we also obtain two sequences from the connected R-FSC $\mathcal{F}$. First we obtain the memory sequence $n_0 n_1 \dots n_{k-1}$, where $n_0 = n_I$ and n_{i+1} is obtained with probability $\delta(n_i, O(n_i), a_i, n_{i+1})$. Second, we obtain the reward sequence $r_0 r_1 \dots r_{k-1}$. In this reward sequence we claim that $r_i = R(n_i, O(s_i), a_i, n_{i+1})$.

Since $\mathcal{F}$ simulates R_k of $\mathcal{M}$, we want the reward of R_k and of $\mathcal{F}$ to be the same when using the same trajectory. Thus $R_k(s_1 a_1 \dots s_k a_k) = \mathcal{R}(n_{k-1}, O(s_k), a_k, n_k) = r_k$.

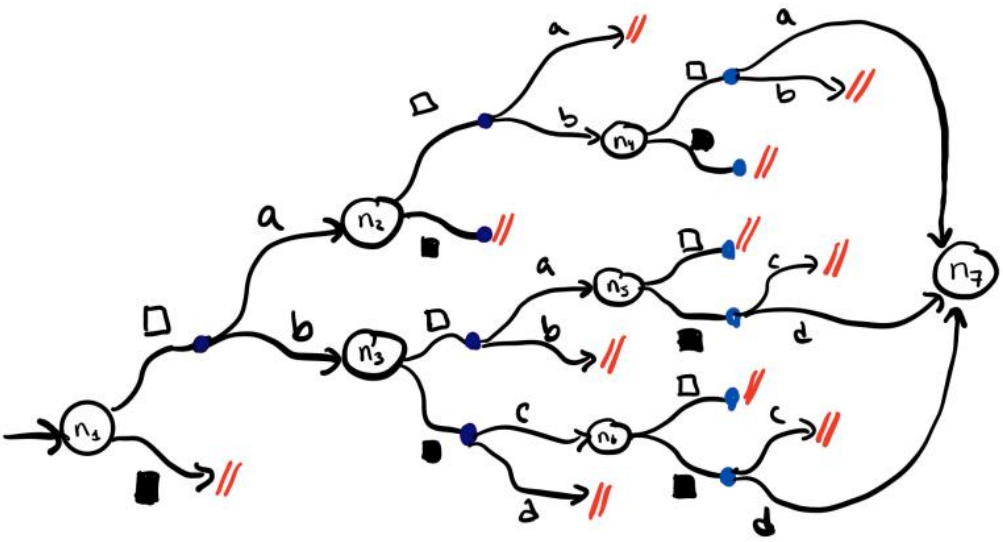
4.2 Obtaining a R-FSC

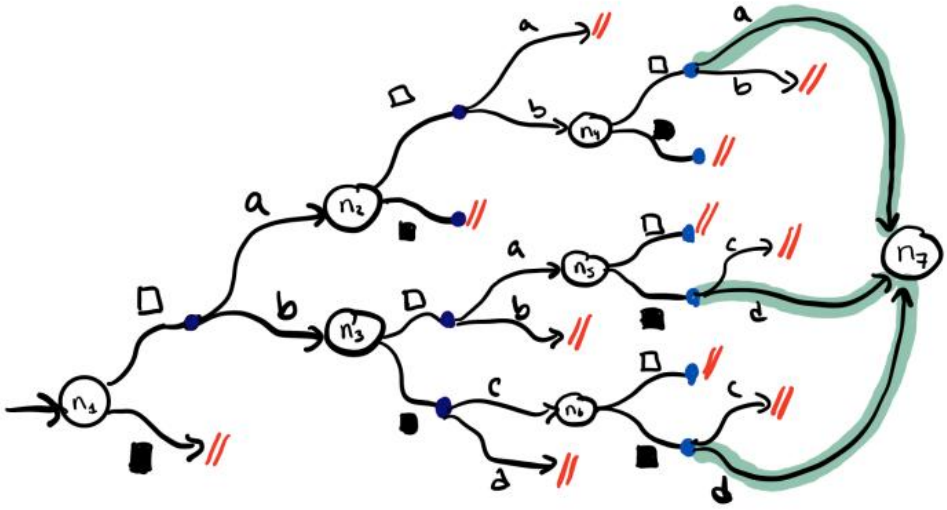
Starting from a fully known R_k and POMDP $\mathcal{M}$

Example

Let R_k be defined over the following sequences

- $\square\ a\ \square\ b\ \square\ a$
- $\square\ b\ \square\ a\ \blacksquare\ d$
- $\square\ b\ \blacksquare\ c\ \blacksquare\ d$





Chapter 5

Obtaining policy

Definition 5.1. The induced Reward Machine for R-FSC on a labeled POMDP is a tuple $\mathcal{A} = (V, v_I, \mathbb{R}, \hat{\delta})$ where

- $V = S \times N$, the finite set of states;
- $v_I = \langle s_I, n_I \rangle$, the initial state;
- $\mathbb{R}$, the output alphabet;
- $\hat{\delta} : V \times Act \to (V, \mathbb{R})$, the transition function.
  $\hat{\delta}(\langle s_1, n_1 \rangle, a) = (\langle s_2, n_2 \rangle, r)$ where
  - $o = O(s_1)$
  - s_2 obtained with probability $T(s_1, a, s_2)$
  - n_2 obtained with probability $\delta(n_1, o, a, n_2)$
  - $r = \mathcal{R}(n_1, o, a, n_2)$

After obtaining the reward machine, you can use known methods to obtain the optimal policy.

Corollary 5.1.1. *Given a POMDP $\mathcal{M}$ and the R-FSC $\mathcal{F}$ obtained through the history-based reward function R_k of $\mathcal{M}$, we create the Reward Machine $\mathbb{A}$ with the help of Definition 5.1. The expected reward of $\mathcal{A}$ is the same as the expected reward of $\mathcal{M}$.*

Bibliography

[1] Mauricio Araya, Olivier Buffet, Vincent Thomas, and Françcois Charpillet. A pomdp extension with belief-dependent rewards. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23, pages 64–72. Curran Associates, Inc., 2010.

[2] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503 – 515, 1954.

[3] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.