Project5: Learning ID and DID

Zeming Fang
661958922

1. Introduction

In this semester, we studied the Bayesian Network (BN) as a model to reflect the unilateral (sometimes causal) impact between real-world events. However, accurate prediction is always part of intelligent activities. For example, based on a BN model, one may know over 95% of it will be rainy tomorrow. However, if he does not take any sequential actions, like bring an umbrella, he will still catch the rain, and the prediction turns out to be meaningless.

Humans and animals continuously intervene in the process of the world by taking different actions. In this final project, I study a framework, the *influence diagram* (ID, also called a decision network), that allows us to model actions as an intelligent activity. This project starts with the introduction to a simple ID that includes only one decision and then shows how to stack up decisions to model intelligent decision-making in a sequential and dynamic task.

As a simple introduction to the ID, the project focuses on the discrete ID model, where all nodes has a set of mutual exclusive states.

2. Information Diagrams

ID can be understood as an extension to BN that allows the network to make decisions while reasoning. Like the BN, the structure of an ID consists of nodes and links $G = (X, A)$. X denotes all the nodes in the ID regardless of node type and A represents all links in ID. However, apart from the chance nodes (denoted as C, represents by oval shape) in the BN, an ID may also include a set of decision nodes (denoted as D, represents by rectangle shape), which an interface to incorporate decisions, and a set of value nodes (denoted as V, represents by diamond shape) represent the criteria we use to make decisions. In another word, we can see BN as a special case of ID when all nodes are chance nodes.

To specify an ID, we also need to assign parameters to quantify the values for V nodes and the conditional probability distributions (CPT) for C nodes. The links to the C nodes are sane as we learned in BN. The links to decision node D is called the information link. The semantic of this concept changed along with the development of the ID. Relatively early works explicitly pointed out that information links D should not be parameterized and only tell the temporal precedence (Shachter 1986; Korb and Nicholson 2010, page. 112; Tatman 1985), while later work lifts this hard constraints (Zhang et al. (1994). In section **??**, I will revisit this topic. Similarly, the semantic of the links to value changes over time. The value node is used to represent a deterministic mapping given the parent nodes, either C node or D (Shachter 1986; Korb and Nicholson 2010, page. 112; Tatman 1985; Koller and Friedman

2009). In the recent works on decision modeling, people started to shift the emphasis on the stochastic value function (Sutton and Barto, 2018, chapter. 2). Thee overlapping between BN and ID increases along with the increasing understanding of both frameworks.

Early ID models made the following two semantic constraints (Shachter, 1986):

- An ID should have no directed cycle as BN (Shachter, 1986).
- The value node V in ID should never have children nodes.
- An ID should be "no-forgetting". Intuitively, given a arbitrary decision node d_i, the diagram should have perfect memory of all precedent decision $d_{1:i-1}$ as well as their parents (Shachter, 1986). This "no-forgetting" have two further implication. First, the decisions sequence has been decided by the ID structure and any decision maker using the ID. Next, the 'no-forgetting' assumption assumes that all trajectories that go through the ID diagrams should go over the same decisions, called symmetric (Shachter and Bhattacharjya, 2010).

Some recent works lifted some of these constraints (Zhang et al. 1994; Jensen et al. 2006; Shenoy 2000). In this project, for the interest of simplicity, I would keep the acyclicity and leafy value function constraints. Meanwhile, I will propose a simple method to circumvent the no-forgetting assumption taking the advantage of MDP's dynamic structure in section 3.2.

The the most important problem of ID is the evaluation problem, which can be analogy to the BN inference problem. To evaluate an ID network, we need to find a sequence of the optimal decisions $\mathrm{d}^* = \{d_1^*, ..., d_2^*\}$ that maximize the expected utility (MEU). A general expression of MEU can be written:

$$\max_{\mathrm{d}} \sum_{\mathrm{c}} p(\mathrm{c}|\mathrm{d}, \mathrm{e}) U(\mathrm{c}, \mathrm{d}) \tag{1}$$

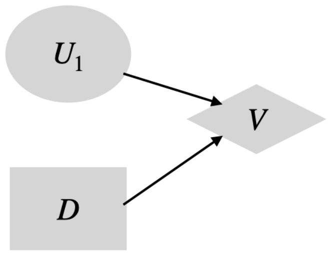
where e means the evidence information. Like the inference in BN, the crux of solving an ID evaluation problem is also *variable-elimination*.

In this section, we will first start by discussing one-shot ID, a simple extension of the BN, clarifying how the ID evaluation problem can benefit from the BN inference. Then, we will move forward to sequential ID with multiple decision inquiries, the sequential an dynamic ID.

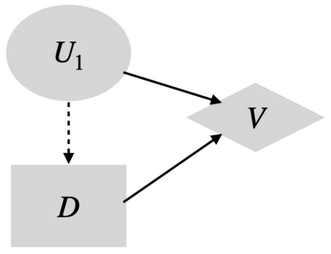
2.1 One-shot information diagrams

The BN can be easily extended to an ID with only one decision inquiry. In this section, I emphasize the understanding difference between directed and undirected information and explore in which conditions they are equivalent.

Both sides agreed that the information links extend the ID evaluation to search for the most profitable decisions with explicit consciousness of the value of their parent configuration. As a consequence of of introducing the information link, the output is turned into a conditional decision table, called *policy*, rather than a single scalar.



(a) Basic one-shot decision problem (without information link)



(b) Basic one-shot decision problem with information link

Figure 1: One-shot ID structure examplars.

The misalignment exists at the representation of the information link. An early claim is that the information links should be treated differently from the links to the chance node, while more recent researches used parameterized links just like the chance node links. These different assumptions led to different evaluation algorithms and different evaluation results.

Algorithm 1 shows the pseudo-algorithm for the evaluation holding a non-parameterized information link assumption (adopted and modified based on (Korb and Nicholson 2010, algorithm. 4.2).

Algorithm 1 ID evaluation algorithm with information links (single decision)

```
1: Input A ID network $G = (X, A)$ with evidence e.
2: for each configuration of parents of the decision node do
3:     for For actions $d_i$ in the decision node do
4:         Set the decision node to that value $d = d_i$
5:         Calculate the posterior probabilities for the parent nodes of the utility node using
           Belief propagation algorithm.
6:         Calculate the resulting expected utility for the action
7: Return the action with the highest expected utility
```

Algorithm 1 used a loop to iteratively fix different configuration of decision-parent pair, d_i and $\pi(d_i) = j$, as evidence. We can only always turn a double loop with a scalar to a matrix $\mathbb{R}^{|\pi(d_i)|\times|d_i|}$. After normalizing the 2th dimension, we got a non-informative CPT for the decision node d_i. With this transformation we turned a non-parameterized information link into a CPT link.

After turning the information link as a CPT, we can use the sum-max-sum method discussed in the class (Ji, 2020, page. 91). The idea of the sum-max-sum method is pretty simple, it is the mixture of sum-product and max-product inference, except that we only use the sum operation to eliminate chance nodes and use the max operation to choose the optimize the decision policy. The most challenging part of the sum-max-sum method is to specify the sequence of variable-elimination. Unlike finding the elimination sequence in Bn, which merely increases some computational burden, an incorrect specification of the sequence might render faulty evaluation results. In section 2.2.1, I introduce an (Shachter,

1986) algorithm that automatically find the elimination sequence. We will use this algorithm to solve ID with parameterized information link.

2.2 Sequential Influence Diagrams

Thus far, we have considered only single decision problems. Once we increase the number of decision nodes, the ID automatically turns into a sequential diagram, called the sequential ID, due to the no-forgetting constraint. By assuming the parents of each decision include all earlier observed decisions and the variables Howard and Matheson (1984), no-forgetting describe the decision-making conditions for an individual decision-maker. The practical benefit of the no-forgetting assumption is it allows a backward evaluation method, start from the very last nodes in the decision sequence, partially overcoming the sequence identification challenge.

2.2.1 Evaluation in the SID

Due to the existence of more than one type of nodes, the variable elimination in ID requires different elimination rule. In this section, I introduce the elimination rule introduced in Shachter (2007) and Shachter (1986) as general rule to solve an sequential ID. In general, there are four graphical transformation operations that helps variables elimination: removal of decision nodes, removal of chance nodes, removal of utility nodes, and reverse arcs. The following content of this subsection are written based on Shachter (2007).

The prerequisite condition that we can *remove a decision node* is: The removal of the decision node D follows the optimal policy determination (Shachter, 2007, page. 36): when a decision node i has only one child, a value node j, and meanwhile all other parents of the value node j are also the parents of the decision node i, $\pi(i) \subseteq \pi(j)\backslash i$. When this condition is satisfied, we can eliminate the decision rule ID.

$$U_j(u|x_k) = \max_{d_i} U_j(d_i, x_K|x_{other}) \tag{2}$$

where $x_K = \pi(i) \subseteq \pi(j)\backslash i$ and x_{other} represents other existed node in the ID diagram. When we eliminate a decision node, we should record the decision d_i. As a result of our removal of the selected decision node i all the information carried by both node $\{d_i, u_j\}$ is now loaded on u_j.

The removal of an uncertain node is allowed when this uncertain node i has only one child, value node j (Shachter, 2007, page. 37). We can use the sum rule to remove the uncertain node i from the diagram,

$$U_j(x_J, x_K, x_L|x_{other}) = \sum_{x_i} p(x_i|x_J, x_K, x_{other}) U_j(x_i, x_K, x_L|x_{other}) \tag{3}$$

where $x_J = \pi(i)\backslash\pi(j)$, $x_K = \pi(i) \cap \pi(j)$, and $x_L = \pi(j) - (i \cup \pi(i))$.

Apart from the individual prerequisite condition, there is a general condition in which we can always remove nodes, called the *barren node condition*: a decision or unobserved chance node has no children. Under the barren node condition, we can simply remove the barren node because it no longer impacts any value node.

Removal of the value nodes has no strict constraints. For any value nodes, i and j, we can always combine them with their sum (Shachter, 2007, page. 37),

$$U_i(x_J, x_K, x_L) = U_i(x_J, x_K) + U_j(x_K, x_L) \tag{4}$$

where $x_J = \pi(i) \backslash \pi(j)$, $x_K = \pi(i) \cap \pi(j)$, and $x_L = \pi(j) - (i \cup \pi(i))$.

However, the removals of the value nodes usually do not reduce computational complexity, so it should be delayed as long as possible (Shachter 2007; Koller and Friedman 2009). Ideally, we should combine the value nodes utill x_J and x_L are all empty set, $x_J = x_K = \varnothing$.

The last and the most complicated graphical operation is called *arc reversal*. The arc from node i to node j is reversible if there is no other directed path from i to j (Shachter, 2007, page. 19). The arc reversal transformation is an application of the Bayes theorem,

$$\begin{aligned} p(x_i, x_j | x_J, x_K, x_L, x_{other}) &= p(x_i | x_J, x_K, x_{other}) p(x_j | x_i, x_K, x_l, x_{other}) \\ p(x_i | x_j, x_J, x_K, x_L, x_{other}) &= \frac{p(x_i, x_j | x_J, x_K, x_L, x_{other})}{\sum_{x_i} p(x_i, x_j | x_J, x_K, x_L, x_{other})} \end{aligned} \tag{5}$$

where $x_J = \pi(i) \backslash \pi(j)$, $x_K = \pi(i) \cap \pi(j)$, and $x_L = \pi(j) - (i \cup \pi(i))$. And x_{other} represents other existed node in the ID diagram. As a result, node j will inherit both parents of node i and j and, thus, extra links needed to be added. The arc reversal method may create a new equivalent ID that allows node removal, when the ID diagram does not satisfy previous discussed conditions. It may meanwhile introduce extra computational complexity because of newly introduced links.

Given these operations, we can directly solve an acyclic and no-forgetting ID using the algorithm introduced in (Shachter, 1986) (see algorithm 2, adopted and modified based on (Shachter, 2007, page. 38)):

Algorithm 2 Acyclic and no-forgetting sequential ID evaluation algorithm

```
1: **Input** A ID network $G = (X, A)$ with evidence e.
2: **while** there are nodes remaining besides the value nodes $V$ and evidence e **do**
3:     **if** if there exists a barren node **then**
4:         remove a barren node
5:     **else**
6:         **if** conditions are satisfied to remove the latest decision **then**
7:             determine its optimal policy using equation 2
8:         **else**
9:             **if** there is unobserved chance node before the latest decision and has at most
   one value child **then**
10:                reversing any arcs to its non-value children in order
11:                remove the chance node
12:            **else**
13:                remove a value node, preferably a descendant of latest decision.
```

Unlike, algorithm 1, this is a quite general algorithm that theoretically can solve ID with arbitrary numbers o decisions. However, the downside the computational complexity may increase exponentially along with the size of the ID (Koller and Friedman, 2009, section. 23.2.3). An extension to the ID to incorporate more decision nodes usually impose additional constraints on the network structure and this conclusion results in the next section: a discussion of dynamic ID (DID).

2.3 Dynamic Influence Diagrams

The difference between the sequential ID and the DID is subtle since both diagrams describe a temporal decision making problem. The application of these models may vary a bit. The sequential ID is a more concrete model that represents a specific reasoning task within a small time window, while dynamic ID aims at growing the model into a long temporal sequence.

To ensure a highly dynamic, continuous setting, the DID usually adopted an extremely simple structure. All the nodes are highly abstracted to include a very small set of states, which is the same across the whole time domain. The number of cross-temporal unilateral links is also limited to a small number, which will repeat at every time-step. Unlike the sequential model which the model structure varies across time, the structure of DID in

To ensure a highly dynamic, continuous setting, the DID usually adopted an extremely simple structure. All the nodes are highly abstracted to include a very small set of states, which is the same across the whole time domain. The number of cross-temporal unilateral links is also limited to a small number, which will repeat at every time-step. Unlike the sequential model which the model structure varies across time, the structures of DID in every timestep are the same.

One of the most simple and practice DID model is the ID representation Markov decision process (MDP) (Shachter and Peot, 1992), as shown in figure 2. At each timestep t, there is only one chance node S_t, one decision node A_t, and one value node R_t. MDP to ID is the Hidden Markov Model (HMM) to Dynamic Bayesian Network (DBN). Its simple structure allows the researchers to develop efficient dynamic algorithms. In section 3, will focus on finding good temporal solution to the MDP.

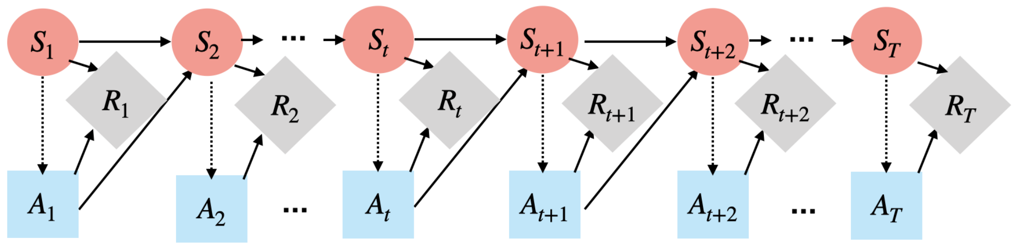


Figure 2: An ID representation of a MDP

If we add a hidden unobserved chance node at each timestep, we got a partially-observable Markov decision process (POMDP). The representation of POMDP is significantly more complex. Complicated as it is, the hidden chance nodes allow us to model the inference behaviors of the decision-makers and the model turns out to be powerful and have a lot of applications (Cassandra, 1998).

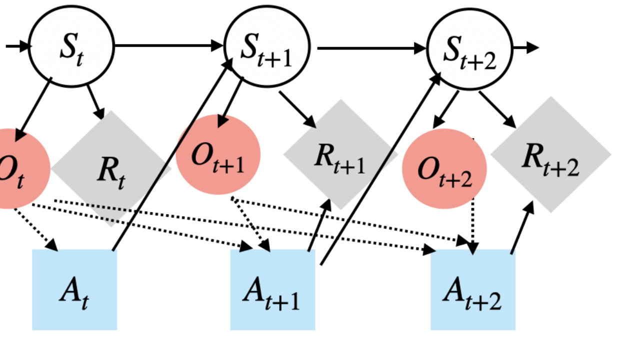


Figure 3: An ID representation of a MDP

3. Markov Decision Process as an ID model

A formal definition of MDP is a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$ (Ziebart, 2010, page. 18). S represents the value space the chance node state. A tells the value space of all possible decisions the decision makers can choose from. R is the utility function of state s and decision a. All time-steps share one utility function. The transition function $\mathcal{T}$ specifies the probability distribution $p(s_{t+1}|s_t, a_t)$ of the state in the next timestep s_{t+1} given the current state s_t and the current action a_t.

At each timestep t, a decision-maker observes a chance node for the state (S_t), then need to make an action (A_t, decision node), and then receive a reward (R_t), value node). Once an action is taken, the MDP makes a stochastic transition to a next state following the transition function $p(s'|s, a)$. for the decision-maker, the perfect information of state is informed via the information link from current state node S_t to current action decision node A_t.

An MDP can be either continuous or discrete. For the interest of simplicity, in this project, we only discuss the discrete scenario where both state node and action node have a finite mutual exclusive value space.

. The goal of the RL is to learn a parameterized behavior policy π that takes actions for each state in the environment. This can be described as a probability distribution over actions, given the context of the current state and the policy's parameters, $\pi(a|s, \theta)$

3.1 Evaluating the MDP

Like all ID diagram, the objective function is to maximize the expected utility. Due to dynamic structure of MDP, the decision sequence is defined as $\{a_1, a_2, ..., a_T\}$. We, thus, can write the MDP objective function as,

$$a^*_{1:T} = \arg\max_{a^*_{1:T}} \sum_{s_{1:T}} p(s_{1:T}|a_{1:T}) \sum_{i=1}^{T} R(s_i, a_i) \tag{6}$$

To solve this equation, we can use the direct method, by unrolling the dynamic process over time and start eliminating nodes from timestep T using algorithm 3. Due to its structure, the variables in MDP can be removed backward from the last timestep T to the very beginning without any arc reversal operation.

Alternatively, we can adopt the idea of dynamic programming and solve the evaluation problem recursively. In the following subsection, I will mimic the *backward algorithm* and *Viterbi algorithm* in HMM to derive a recursive solution for the MDP.

3.2 Recursive evaluation of MDP

The crux of proposing a recursive solution is to find a reusable term that is used in every timestep. Following the chain rule and the first-order Markov property the joint distribution $p(s_{1:T}|a_{1:T})$ can be written as,

$$p(s_{1:T}|a_{1:T}) = p(s_1)\prod_{i=1} p(s_{i+1}|s_i, a_i) \tag{7}$$

Follow the idea of (Ji, 2019, equation. 3.142-3.143), we can expand the objective function (the right-hand side of equation 6) based on equation 7 as (see the detailed expansion in appendix 1 equation 10),

$$\begin{aligned}
&\max_{a_{1:T}} \sum_{s_{1:T}} p(s_{1:T}|a_{1:T}) \sum_{i=1}^{T} R(s_i, a_i) \\
=&\sum_{s_1} \max_{a_1} p(s_1)[R(s_1, a_1)+ \\
&\sum_{s_2} p(s_2|s_1, a_1) \max_{a_2}[R(s_2, a_2) + \max_{a_{3:T}} p(s_{3:T}|a_{2:T}, s_2) \sum_{i=3}^{T} R(s_i, a_i)]
\end{aligned} \tag{8}$$

Let $Q^t(s_t, a_t) = R(s_t, a_t) + \max_{a_{t+1:T}} \sum_{s_{t+1:T}} p(s_{t+1:T}|s_t, a_{t:T}) \sum_{i=t+1}^{T} R(s_i, a_i)$, which can be extended into the following recursive form (see equation 11 in appendix 2 for the detailed deviation):

$$\begin{aligned}
Q^t(s_t, a_t) =& R(s_t, a_t) + \max_{a_{t+1:T}} \sum_{s_{t+1:T}} p(s_{t+1:T}|a_{t:T}, s_t) \sum_{i=t+1}^{T} R(s_i, a_i) \\
=& R(s_t, a_t) + \sum_{s_{t+1}} \max_{a_{t+1}} \max_{a_{t+2:T}} \\
&\sum_{s_{t+2:T}} p(s_{t+1}, s_{t+2:T}|a_{t+1:T}, a_t, s_t) \sum_{i=t+1}^{T} R(s_i, a_i) \\
=& R(s_t, a_t) + \sum_{s_{t+1}} p(s_{t+1}|a_t, s_t) \max_{a_{t+1}} Q^{t+1}(s_{t+1}, a_{t+1})
\end{aligned} \tag{9}$$

Note that $Q(s_T, a_T) = R(s_T, a_T)$ because there is no more successor nodes. According to equation 9, we know that the early Q function always depends on its successor, which means

Algorithm 3 Backward recursive solution to MDP

```
1: Input An MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$
2: Initialize $Q^T(s, a) = R(s, a)$
3: Start from the last step $T$
4: Recursively compute $Q^{t-1}$ using equation 9
5: termination $EU = \sum_{s_1} p_0(s) \max_a Q^1(s, a)$
6: return $EU$, $\{a_1, a_2, ..., a_T\}$
```

this recursive solution is a backward method. Algorithm 3 provides the pseudo-code for the backward solution using the recursive method.

A close look at the algorithm 3, we may found it is extremely similar to the exact method we discussed in algorithm 2. Both start from the very last decision, first remove the decision node $\max_{a_{t+1}}$ using max operation , then remove chance nodes by marginalization $\sum_{s_{t+1}}$, and finally combine the value nodes $Q(s_{t+1}, a_{t+1})$.

3.3 Recursive methods without fully acknowledging the decision orders

Both algorithm 2 and algorithm 3 requires us to start with the last decision state, which means we need to have a bird-view picture of the whole decision-making process. Some interpreted this full acknowledgment of the decision order as a by-product of the no-forgetting assumption (Shachter and Bhattacharjya, 2010).

However, I consider the assumption of fully acknowledging the decision to order an aggressive assumption, and the algorithm developed under this assumption may have limited practical applications. To find the last state, the decision processes are required to 1) first observe all the precedence states and decisions; 2) have the same length for each trajectory (known as the symmetric problem). However, it is hard to guarantee that every unrolls of the dynamic process returns a fixed length of trajectories (e.g. unrolling the HMM). What's more, the termination condition of most of the real-world decision making is to reach a certain state instead of undergoing a certain period of time. It is necessary to develop an algorithm that makes no assumption of full observation of the decision process.

Inspired by the belief propagation and reinforcement learning algorithm, I make a subtle modification to algorithm 3 to get rid of the fully acknowledging assumption (see algorithm 4). This method initializes the Q function to make a guess of the value and iterate to correct the biases brought by the initialization, just like belief propagation.

Algorithm 4 Forward recursive solution to MDP

```
1: Input An MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R)$
2: Initialize $\{Q^1(s, a), Q^2(s, a), ...\}$
3: while not converge do
4:     Randomly choose a starting timestep $t$, usually start $t = 1$
5:     Recursively compute $Q^t(s, a)$ using equation 9 and the stored $Q^{t+1}$
6:     Save the new $Q^t(s, a)$
7: return $EU$, $\{a_1, a_2, ..., a_T\}$
```

This method is very similar but is simpler than the policy iteration method (Sutton and Barto, 2018, section. 4.3).

3.4 Learning of MDP

Learning in MDP is to figure out the parameters that represent the transition function and the reward function. The complexity of the learning problem usually depends not on the input representations. In this project, we restrict the value space of all nodes as a small discrete set so the learning problem can be easily solved by counting and normalize.

Based on the lecture slide (Ji, 2020, page. 93), the key in ID learning is to separate the decision and chance nodes learning and the value node learning.

4. Experiments

4.1 Experiment1: Single shot decision making

In experiment 1, we will test algorithm 1 and algorithm 2 to solve this fever example (see figure 4)

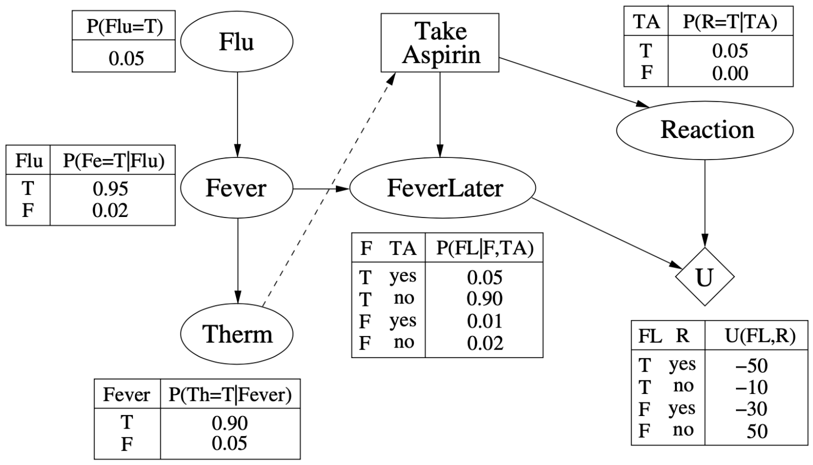


Figure 4: (adopted from (Korb and Nicholson, 2010, section. 4.3.5)) A simple decision problem

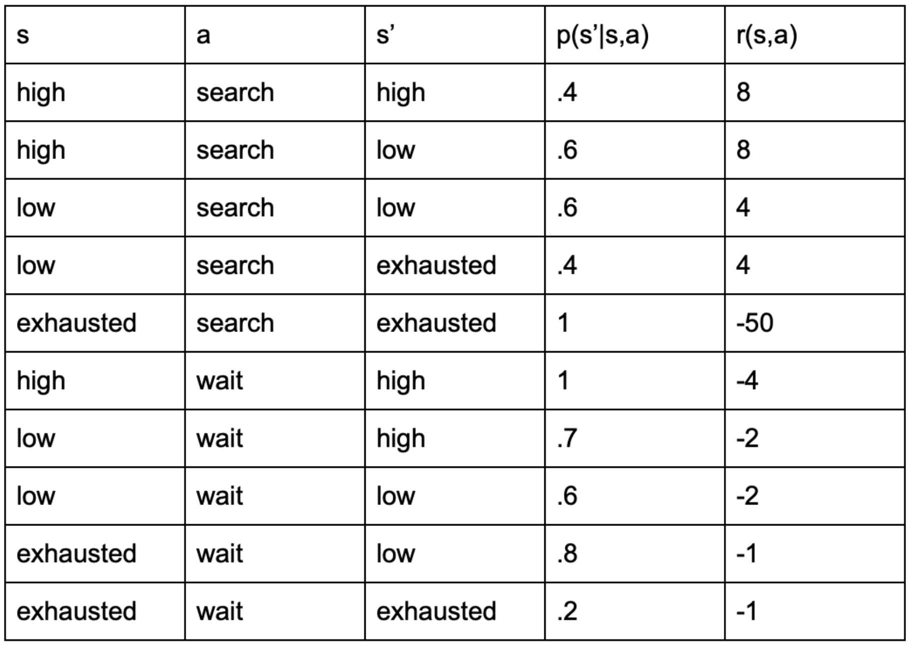
The goal is to solve

- Evaluation MEU
- Compare the performance between algorithm 1 (with non-parameterized) and algorithm 2 (allow parameterized information link).

4.2 Experiment2:

Run the algorithm 3 and algorithm 4 to conduct a self-defined sequential decision game with fixed length: Survival.

Supposed you are attending a *Desert Island Survival* competition. On the island your all available action is $\mathcal{A} = search, wait$, search means go out to search treasure, and wait is to stay in a cave to restore energy. All possible body conditions is $\mathcal{S} = high - energy, low - energy, exhausted$. For example, in the high energy condition, the searching efficiency will be high, but you have probability to enter the low energy condition. If you have a rest at high energy condition, you will get more penalty. The underlying transition matrix and the reward functions are shown in figure 5. Note that the $p(s_0 = \text{high}) = 1$.



s	a	s'	p(s'|s,a)	r(s,a)
high	search	high	.4	8
high	search	low	.6	8
low	search	low	.6	4
low	search	exhausted	.4	4
exhausted	search	exhausted	1	-50
high	wait	high	1	-4
low	wait	high	.7	-2
low	wait	low	.6	-2
exhausted	wait	low	.8	-1
exhausted	wait	exhausted	.2	-1

Figure 5: transition function and reward function for survival game, all the unlisted conditions have the probability of 0.

Gain as much reward as we can within 7 days. To show the backward algorithm (algorithm 3) and forward algorithm (algorithm 4) give the same prediction.

4.3 Experiment3:

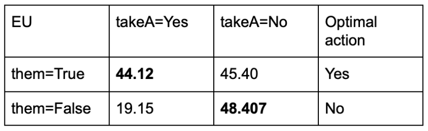
Learn the transition function and reward function in figure 5 using the counting method. The data we fed to the algorithm is the synthesized data generated by the MDP with the same parameters.

5. Results

5.1 Result of Experiment1

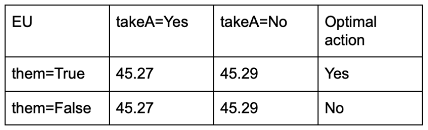
The policy learned with algorithm 1 can only calculate the Expected utility (see figure 6a.

However, using the algorithm 2 I can only compute expected utility by sum over the therm conditions. It seems like my code has some problems with accepting therm as evidence.



EU	takeA=Yes	takeA=No	Optimal action
them=True	**44.12**	45.40	Yes
them=False	19.15	**48.407**	No

(a) Learned policy using algorithm1



EU	takeA=Yes	takeA=No	Optimal action
them=True	45.27	45.29	Yes
them=False	45.27	45.29	No

(b) Learned policy using algorithm1

Figure 6: policy learned in both algorithm1 and algorithm2

Another observation is the dependency from therm to decision did affect the EU estimation, we can think of this as prior knowledge of the policy. I guess when we specific the dependency as [[1,1], [1,1]]. Algorithm 2 is equivalent to algorithm1.

5.2 Result of Experiment2

Under the condition 1: gain as much reward as possible in 7 days, both algorithm returns the same maximized expected utility $MEU = 30.565784$. However, the backward algorithm can make the correct estimation within 1 sweep, while the forward algorithm requires 7 iterations (see figure 7).

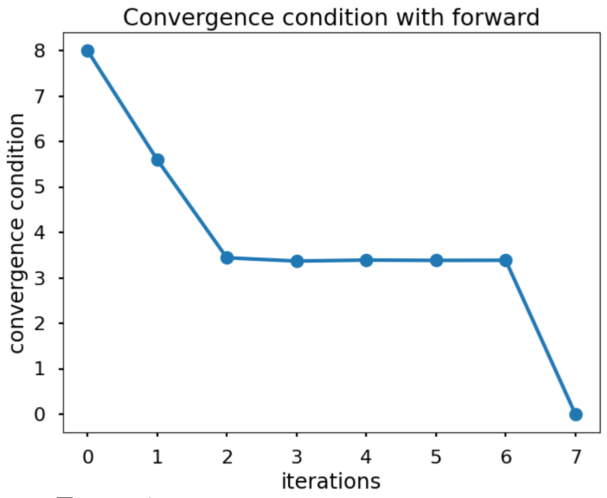


Figure 7: Convergence of the forward algorithm

The policy they learned is the same. In the first 5 days, the policy is conservative, only search under high energy conditions (see figure 8 a). When the end of MDP is approaching, the policy tends to become aggressive only wait when exhausted (see figure 8 b).

5.3 Result of Experiment3

Here I use the condition1 in survival game: gain as much reward as possible in 7 days. I simulated 100 trajectories each with 7 time steps as the training data.

The learned transition function and reward functions are shown in figure 9

high	search
low	wait
exhausted	wait

(a) Learned policy in first 5 days

high	search
low	search
exhausted	wait

(b) Learned policy in last 2 days

Figure 8: policy learned in both forward and backward algorithm

s	a	s’	p(s’|s,a)	r(s,a)
high	search	high	.4485	8
high	search	low	.5515	8
low	search	low	.6076	4
low	search	exhausted	.3924	4
exhausted	search	exhausted	1	-50
high	wait	high	1	-4
low	wait	high	.6591	-2
low	wait	low	.3409	-2
exhausted	wait	low	.8421	-1
exhausted	wait	exhausted	.1579	-1

Figure 9: Learn the transition function and the reward function separately by counting.

6. Conclusions

In this project, I implement the automatic ID evaluation algorithm introduced in Shachter (1986). Also, I derive two evaluation algorithm for ID representation of MDPs. Finaly, I use the counting method to learn the transition function and reward function in MDP environment.

References

Anthony R Cassandra. A survey of pomdp applications. In Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes, volume 1724, 1998.

Ronald A Howard and James E Matheson. The principles and applications of decision analysis. Strategic Decisions Group, Palo Alto, CA, pages 719–762, 1984.

Finn V Jensen, Thomas D Nielsen, and Prakash P Shenoy. Sequential influence diagrams: A unified asymmetry framework. International Journal of Approximate Reasoning, 42(1-2): 101–118, 2006.

Qiang Ji. Probabilistic Graphical Models for Computer Vision. Academic Press, 2019.

Qiang Ji. Fall20 other directed bn. 2020.

Daphne Koller and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

Kevin B Korb and Ann E Nicholson. Bayesian artificial intelligence. CRC press, 2010.

Ross D Shachter. Evaluating influence diagrams. Operations research, 34(6):871–882, 1986.

Ross D Shachter. 10 Model Building with Belief Networks and Influence Diagrams. Citeseer, 2007.

Ross D Shachter and Debarun Bhattacharjya. Solving influence diagrams: Exact algorithms. Wiley Encyclopedia of Operations Research and Management Science, 2010.

Ross D. Shachter and Mark A. Peot. Decision making using probabilistic inference methods. In 8th Conferece on Uncertainty in Artificial Intelligence, pages 276–283, 1992.

Prakash P Shenoy. Valuation network representation and solution of asymmetric decision problems. European Journal of Operational Research, 121(3):579–608, 2000.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

Joseph A Tatman. Decision processes in influence diagrams: Formulation and analysis. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH, 1985.

Nevin Lianwen Zhang, Runping Qi, and David Poole. A computational theory of decision networks. International Journal of Approximate Reasoning, 11(2):83–158, 1994.

Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

7. Appendix

7.1 Expanding the objective function of MDP

$$
\begin{aligned}
&\max_{a_{1:T}} \sum_{s_{1:T}} p(s_{1:T}|a_{1:T}) \sum_{i=1}^{T} R(s_i, a_i) \\
=&\max_{a_{1:T}} \sum_{s_{1:T}} p(s_{1:T}|a_{1:T})[R(s_1, a_1) + \sum_{i=2}^{T} R(s_i, a_i)] \\
=&\sum_{s_1} \max_{a_1} p(s_1) R(s_1, a_1) + \sum_{s_1} \max_{a_1} \max_{a_{2:T}} \sum_{s_{2:T}} p(s_{2:T}|a_{2:T}, s_1, a_1) \sum_{i=2}^{T} R(s_i, a_i) \\
=&\sum_{s_1} p(s_1) \max_{a_1}[R(s_1, a_1) + \max_{a_{2:T}} \sum_{s_{2:T}} p(s_{3:T}, s_2|s_1, a_{2:T}, a_1) \sum_{i=2}^{T} R(s_i, a_i)] \\
=&\sum_{s_1} p(s_1) \max_{a_1}[R(s_1, a_1) + \max_{a_{2:T}} \sum_{s_{2:T}} p(s_2|s_1, a_{2:T}, a_1) p(s_{3:T}|s_2, s_1, a_{2:T}, a_1) \sum_{i=2}^{T} R(s_i, a_i)] \\
=&\sum_{s_1} p(s_1) \max_{a_1}[R(s_1, a_1) + \max_{a_{2:T}} \sum_{s_{2:T}} p(s_2|s_1, a_1) p(s_{3:T}|s_2, a_{2:T}) \sum_{i=2}^{T} R(s_i, a_i)] \\
=&\sum_{s_1} p(s_1) \max_{a_1}[R(s_1, a_1) + \sum_{s_2} p(s_2|s_1, a_1) \max_{a_2}[R(s_2, a_2) + \max_{a_{3:T}} p(s_{3:T}|a_{2:T}, s_2) \sum_{i=3}^{T} R(s_i, a_i)]
\end{aligned}
\tag{10}
$$

7.2 Recursive solution to the MDP evaluation

$$\begin{aligned}
Q^t(s_t,a_t) =& R(s_t,a_t) + \max_{a_{t+1:T}} \sum_{s_{t+1:T}} p(s_{t+1:T}|a_{t:T},s_t) \sum_{i=t+1}^{T} R(s_i,a_i) \\
=& R(s_t,a_t) + \sum_{s_{t+1}} \max_{a_{t+1}} \max_{a_{t+2:T}} \\
& \sum_{s_{t+2:T}} p(s_{t+1}, s_{t+2:T}|a_{t+1:T}, a_t, s_t) \sum_{i=t+1}^{T} R(s_i,a_i) \\
=& R(s_t,a_t) + \sum_{s_{t+1}} \max_{a_{t+1}} \max_{a_{t+2:T}} \sum_{s_{t+2:T}} p(s_{t+1}|a_{t+1:T}, a_t, s_t) \\
& p(s_{t+2:T}|s_{t+1}, a_{t+1:T}, a_t, s_t) \sum_{i=t+1}^{T} R(s_i,a_i) \\
=& R(s_t,a_t) + \sum_{s_{t+1}} \max_{a_{t+1}} \max_{a_{t+2:T}} \sum_{s_{t+2:T}} \\
& p(s_{t+1}|a_t, s_t) p(s_{t+2:T}|s_{t+1}, a_{t+1:T}) \sum_{i=t+1}^{T} R(s_i,a_i) \\
=& R(s_t,a_t) + \sum_{s_{t+1}} p(s_{t+1}|a_t,s_t) \max_{a_{t+1}} [R(s_{t+1},a_{t+1}) + \\
& \max_{a_{t+2:T}} \sum_{s_{t+2:T}} p(s_{t+2:T}|s_{t+1},a_{t+1:T}) \sum_{i=t+2}^{T} R(s_i,a_i)] \\
=& R(s_t,a_t) + \sum_{s_{t+1}} p(s_{t+1}|a_t,s_t) \max_{a_{t+1}} Q^{t+1}(s_{t+1},a_{t+1})
\end{aligned} \tag{11}$$