Planning Under Uncertainty: Decision Theory 2 (Sequential Decisions)

Shafiq Joty

Nanyang Technological University

srjoty@ntu.edu.sg

April 24, 2019

Overview

1. Recap
   - One-off Decisions
   - Decision Variables
   - Decision Networks
2. Sequential Decision Problem
   - Sequential Decisions
   - Decision Functions
   - Policy
3. Optimal policy
   - Possible worlds satisfying a policy
   - Expected Utility of a Policy
   - Computing Optimal Policy

One-off Decisions

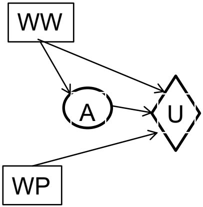
- **One-off Decisions:** Seq. of primitive decisions that can be treated as a single macro decision to be made before acting.
  $\Rightarrow$ Pick the sequence with the maximum expected utility.

Decision Variables

- **Decision variables** are like random variables that an agent gets to choose the value of.
- A possible world specifies a value for each decision variable and for each random variable.
- For each assignment of values to all decision variables, the measures of the worlds satisfying that assignment sum to 1.
- The probability of a proposition is undefined unless you condition on the values of all decision variables.

Decision Networks

- A **decision network** is a graphical representation of a finite sequential decision problem.
- Decision networks extend **belief networks** to include decision variables and utility.
- A decision network specifies what information is available when the agent has to act (decision nodes).
- A decision network specifies which variables the utility depends on.

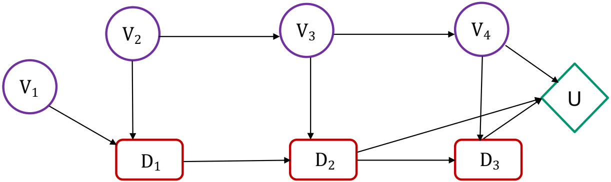


Sequential Decisions

- Agent doesn't make a multi-step decision and carry it out blindly. It takes new observations it makes into account.
- It does the following three steps iteratively in an environment.
  1. Makes observations
  2. Decides on an action
  3. Carries out the action
- Subsequent actions can depend on what is observed
  - What is observed often depends on previous actions
  - Often the sole reason for carrying out an action is to provide information for future actions.

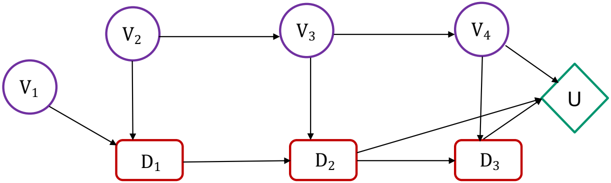
Sequential Decision Problem

- A **sequential decision problem** consists of a sequence of decision variables: $(D_1, \ldots, D_T)$.
- Each D_t has an information set of variables pD_t (parents of D_t), whose value is known at the time decision D_t is made.



Sequential Decision Problem

- A **sequential decision problem** consists of a sequence of decision variables: $(D_1, \ldots, D_T)$.
- Each D_t has an information set of variables pD_t (parents of D_t), whose value is known at the time decision D_t is made.



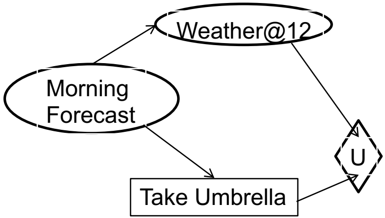
- **General Decision networks:** Just like single-stage decision networks, with one exception: the parents of decision nodes can include random variables

Sequential Decision Problem: One-Decision Example

- Early morning, I listen to the weather forecast, shall I take my umbrella today? (I'll have to go for a long walk at noon)

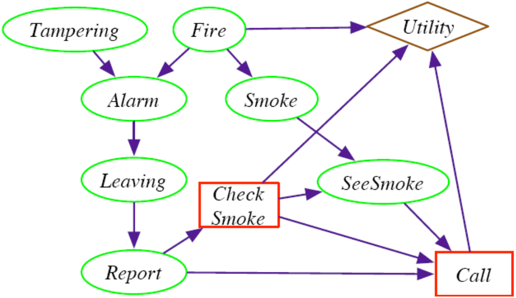
Sequential Decision Problem: One-Decision Example

- Early morning, I listen to the weather forecast, shall I take my umbrella today? (I'll have to go for a long walk at noon)
- A reasonable decision network (for a single-decision):



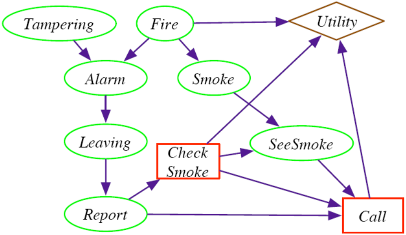
- Decision node (rectangle): Agent decides
- Chance node (circle): Chance decides

A more "Complete" Example



- Decision node (rectangle): Agent decides
- Chance node (circle): Chance decides

What should an agent do?



- What an agent should do now depends on what it will decide to do in the future; e.g., agent only needs to check for smoke if that will affect whether it calls
- What an agent does in the future depends on what it did before; e.g., when making the decision on whether to call, it needs to know whether it checked for smoke

What should an agent do?

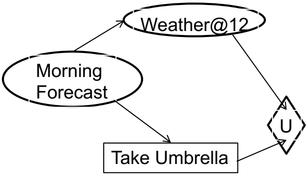
- What an agent should do now depends on what it will do in the future
- What an agent does in the future depends on what it did before

We will get around this problem as follows

- The agent maintains a conditional plan of what it will do in the future
- We will formalize this conditional plan as a policy or a sequence of decision functions.

Decision Functions

- A **decision function** specifies what an agent should do (i.e., its action) under each circumstance (parents' values)
- It is a function $\delta : \mathcal{D}(pD) \rightarrow \mathcal{D}(D)$, meaning that when the agent has observed $o \in \mathcal{D}(MF)$, it will do $\delta(o)$.



- **M**orning **F**orecast can be {**C**loudy, **S**unny, **R**ainy}, and **T**ake **U**mbrella can be {**T**rue, **F**alse}.
- How many decision functions are there?

Decision Functions

How many decision functions?

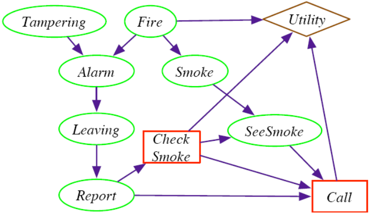
Morning **F**orecast can be {**C**loudy, **S**unny, **R**ainy};
Take **U**mbrella can be {**T**rue, **F**alse}.

MF	δ^1_{TU}	δ^2_{TU}	δ^3_{TU}	..	δ^8_{TU}
C	T	F	T	..	T
S	T	F	F	..	T
R	T	F	T	..	F

- How many instantiation of parents? $|\mathcal{D}(MF)| = 3$
- How many possible values for TU? $|\mathcal{D}(TU)| = 2$
- How many different decision functions? $|\mathcal{D}(TU)|^{|\mathcal{D}(MF)|} = 2^3$ because for each instantiation of the parent, the decision function could pick any of the 2 values

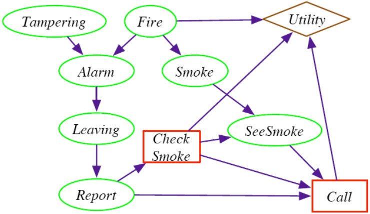
Policies for Sequential Decision Problems

A policy (π) is a sequence of (applied) **decision functions** $(\delta_1, \ldots, \delta_T)$, where each $\delta_t(D_t|pD_t) = \mathcal{D}(pD_t) \rightarrow \mathcal{D}(D_t)$, i.e., when the agent has observed $o \in \mathcal{D}(pD_t)$, it will do $\delta_t(o)$



- How many decision functions?
  $\Rightarrow 2^2$(*for CheckSmoke*) *and* 2^8(*for Call*)

Policies for Sequential Decision Problems

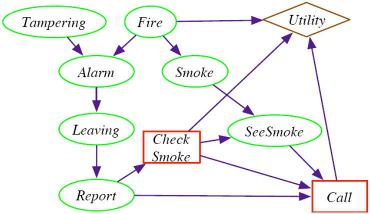


- 2^2 decision functions for Check Smoke

R	δ^1_{CS}	δ^2_{CS}	δ^3_{CS}	δ^4_{CS}
T	T	F	T	F
F	T	F	F	T

- Similarly, there are 2^8 different decision functions for Call

Policies for Sequential Decision Problems



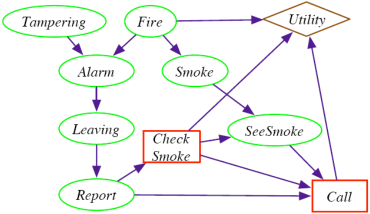
How many different policies are there?

- 2^2 decision functions for Check Smoke
- 2^8 decision functions for Call

$\Rightarrow$ total # of policies: $2^8 \times 2^2$

Policies for Sequential Decision Problems

If there are d decision variables, each with k binary parents and b possible actions, how many policies are there?



- # of parent assignments for each decision variable: 2^k
- # of decision functions for each decision variable: b^{2^k}

$\Rightarrow$ total # of policies: $(b^{2^k})^d$

Possible worlds satisfying a policy

A possible world ω specifies a value for each random variable and decision variable.

A possible world satisfies a policy π (written $\omega \models \pi$), if the world assigns the value to each decision node that the policy specifies.

Possible worlds satisfying a policy

A possible world ω specifies a value for each random variable and decision variable.

A possible world satisfies a policy π (written $\omega \models \pi$), if the world assigns the value to each decision node that the policy specifies.

- Consider the following policy:

- Check smoke (i.e. CheckSmoke=true) iff Report=true
- Call iff Report=true, CheckSmoke=true, SeeSmoke=true

Possible worlds satisfying a policy

A possible world ω specifies a value for each random variable and decision variable.
A possible world satisfies a policy π (written $\omega \models \pi$), if the world assigns the value to each decision node that the policy specifies.

- Consider the following policy:

- Check smoke (i.e. CheckSmoke=true) iff Report=true
- Call iff Report=true, CheckSmoke=true, SeeSmoke=true

Does the following possible worlds satisfy this policy?

1. $\neg$tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call $\Rightarrow$ Yes

Possible worlds satisfying a policy

A possible world ω specifies a value for each random variable and decision variable.

A possible world satisfies a policy π (written $\omega \models \pi$), if the world assigns the value to each decision node that the policy specifies.

- Consider the following policy:

- Check smoke (i.e. CheckSmoke=true) iff Report=true
- Call iff Report=true, CheckSmoke=true, SeeSmoke=true

Does the following possible worlds satisfy this policy?

1. ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call ⇒ Yes
2. ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, ¬call ⇒ No

Possible worlds satisfying a policy

A possible world ω specifies a value for each random variable and decision variable.

A possible world satisfies a policy π (written $\omega \models \pi$), if the world assigns the value to each decision node that the policy specifies.

- Consider the following policy:

- Check smoke (i.e. CheckSmoke=true) iff Report=true
- Call iff Report=true, CheckSmoke=true, SeeSmoke=true

Does the following possible worlds satisfy this policy?

1. ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, call ⇒ Yes
2. ¬tampering, fire, alarm, leaving, report, smoke, checkSmoke, seeSmoke, ¬call ⇒ No
3. ¬tampering, fire, alarm, leaving, ¬report, ¬smoke, ¬checkSmoke, ¬seeSmoke, ¬call ⇒ Yes,

Expected Utility of a Policy

For $X_1, \ldots, X_n$ random variables and $D_1, \ldots, D_m$ decision variables:

- The expected utility of policy π is

$$\mathbb{E}(U|\pi) = \sum_{\omega \models \pi} p(\omega) U(\omega)$$

$$\mathbb{E}(U|\pi) = \sum_{X_1,\ldots,X_n; D_1,\ldots,D_m} \prod_{i=1}^{n} p(X_i|pX_i) \prod_{j=1}^{m} \delta_j^{\pi}(D_j|pD_j) U(pU)$$

$\delta_j^{\pi}(D_j|pD_j) = 1$ if the possible world satisfies π else 0.

Expected Utility of a Policy

For $X_1, \ldots, X_n$ random variables and $D_1, \ldots, D_m$ decision variables:

- The expected utility of policy π is

$$\mathbb{E}(U|\pi) = \sum_{\omega \models \pi} p(\omega) U(\omega)$$

$$\mathbb{E}(U|\pi) = \sum_{X_1,\ldots,X_n;D_1,\ldots,D_m} \prod_{i=1}^{n} p(X_i|pX_i) \prod_{j=1}^{m} \delta_j^{\pi}(D_j|pD_j) U(pU)$$

$\delta_j^{\pi}(D_j|pD_j) = 1$ if the possible world satisfies π else 0.

- An optimal policy is the one with the highest expected utility. $\pi^* = \text{argmax}_{\pi}\, \mathbb{E}(U|\pi)$

Maxing out vs. Summing out

Maxing out a variable is similar to marginalization (summing out), but instead of taking the *sum* of some values, we take the *max*.

- Summing out X_i: $\sum_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$
- Maxing out X_i: $max_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$

Maxing out vs. Summing out

Maxing out a variable is similar to marginalization (summing out), but instead of taking the *sum* of some values, we take the *max*.

- Summing out X_i: $\sum_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$
- Maxing out X_i: $max_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$

B	A	C	$f_3(A,B,C)$
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14
f	f	t	0.48
f	f	f	0.32

$\boxed{max_B f_3(A,B,C) = f_4(A,C)}$

A	C	$f_4(A,C)$
t	t	0.54
t	f	0.36
f	t	?
f	f	

0.32 0.06 0.48 0.14

Maxing out vs. Summing out

Maxing out a variable is similar to marginalization (summing out), but instead of taking the *sum* of some values, we take the *max*.

- Summing out X_i: $\sum_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$
- Maxing out X_i: $max_{j \in \mathcal{D}(X_i)} f(X_1, \ldots, X_i = j, \ldots, X_n)$

B	A	C	$f_3(A,B,C)$
t	t	t	0.03
t	t	f	0.07
f	t	t	0.54
f	t	f	0.36
t	f	t	0.06
t	f	f	0.14
f	f	t	0.48
f	f	f	0.32

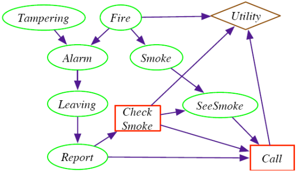
$\max_B f_3(A,B,C) = f_4(A,C)$

A	C	$f_4(A,C)$
t	t	0.54
t	f	0.36
f	t	?
f	f	

0.32 0.06 0.48 0.14

$\Rightarrow 0.48, 0.32$

Optimal Policies with VE: Overall Idea



Dynamic programming: precompute optimal future decisions

- Consider the last decision D (e.g., Call) to be made
  - Find optimal decision $D = d$ for each instantiation of pD.
  (For each instantiation of pD, this is just a single-stage decision problem)
  - Create a factor of these maximum values: max out D (i.e., for each instantiation of the parents, what is the best utility I can achieve by making this last decision optimally?)
- Recurse to find optimal policy for the reduced network (now with one less decision node)

Optimal Policies with VE: More Formally

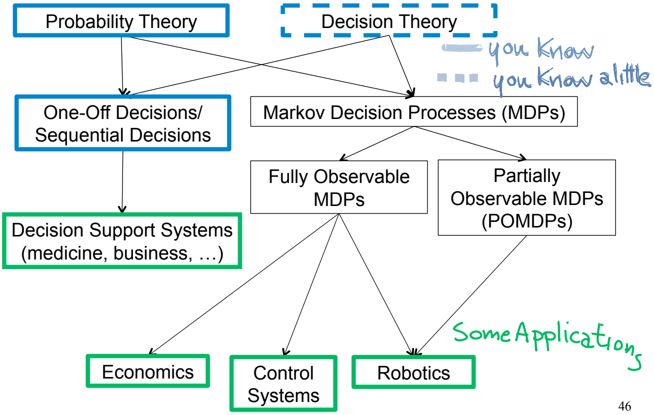
Dynamic programming:

1. Create a factor for each CPT and a factor for the utility.
2. While there are still decision variables
   1. Sum out random variables that are not parents of a decision node (e.g., Tampering, Fire, Alarm, Smoke, Leaving)
   2. Max out last decision variable D in the total ordering (keep track of decision functions).
3. Sum out any remaining random variables. This is the maximum expected utility. By keeping track of the decision functions, we can get the optimal policy.

Optimal Policies with VE: Complexity

- For d decision variables (each with k binary parents and b possible actions), there are $(b^{2^k})^d$ policies
- VE saves the final exponent
  1. Dynamic programming: consider each decision functions only once
  2. Resulting complexity: $O(d * b^{2^k})$
- Much faster than enumerating policies (or search in policy space), but still doubly exponential.
- CS422: approximation algorithms for finding optimal policies.

Big Picture: Planning Under Uncertainty



Learning Goals

- Sequential decision networks
  - Represent sequential decision problems as decision networks
- Policies
  - Verify whether a possible world satisfies a policy
  - Define the expected utility of a policy
  - Compute the number of policies for a decision problem
  - Compute the optimal policy by Variable Elimination

The End