Content

1 Organisation 3
1.1 Topics 3
1.2 Literature 3
1.3 PÜ & PL 3
2 Overview 4
3 Rational Decision Agents 4
3.1 Agents 4
3.2 Environments 6
3.3 Structure of Agents 7
3.4 Types of Agents 7
3.5 Keywords 7
4 Probability 8
4.1 Sources of Uncertainty 8
4.2 Interpretations of Probability 9
4.3 Probability Theory 10
4.4 Inclusion Exclusion Principle 11
4.5 Probabilistic Reasoning 12
4.6 Stochastic Independence 14
4.7 Bayes’ Rule and Diagnostic Reasoning 15
5 Probabilistic Graphical Models 16
5.1 Bayesian Networks 16
5.2 Stochastic Sampling 20
5.3 Hidden Markov Models 22
5.4 Sum-Product Networks (SPNs) 25
6 Decision Theory and Decision Making 27
6.1 Definitions 27
6.2 Decision Networks 27
6.3 Value of Information 27

6.4 Markov Decision Processes (MDPs) 29

7 Reinforcement Learning 34

7.1 Batch learning 34

7.2 Q-Learning 34

8 Game Theory 36

8.1 Single Move Games 36

8.2 Nash Equilibria 37

8.3 Zero Sum Games 37

8.4 Repeated Games 39

8.5 Sequential Games 39

1 Organisation

1.1 Topics

- Probabilistic Graphical Models
- Decision Theory and Decision Networks
- Markov Decision Processes and Reinforcement Learning
- Game Theory and Mechanism Design

1.2 Literature

- Russel & Norvig – Artificial Intelligence – A modern Approach, Pearson 2013
    - Chapters 2 & 13-17

1.3 PÜ & PL

- Lecture
    - Pre-Recorded Videos
- Exercises
    - Sheets each Week
    - Solutions get discussed on Mondays 13:45 (in person) or 15:30 (online)
- Midterm
    - 45mins
    - not graded
    - covering half of the material
- Exam
    - 17.12. (100%)

2 Overview

“How to make good decisions automatically” (through algorithms)

- Is not about Decision Support Systems, OLAP/BA, Data Mining, Machine Learning, or human decision making
- Is about uncertainty in AI, Decision Theory, Game Theory, Alignment & AI Safety

3 Rational Decision Agents

3.1 Agents

An agent is something that *acts*.

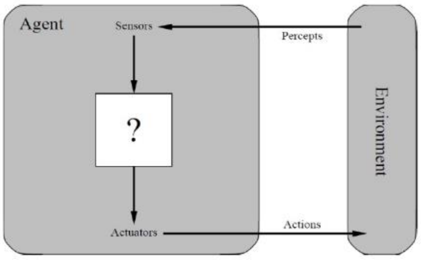
- Operates **autonomously**
- Perceives its **environment**
- Persists over a prolonged time period
- Adapts to change
- Creates and pursues goals

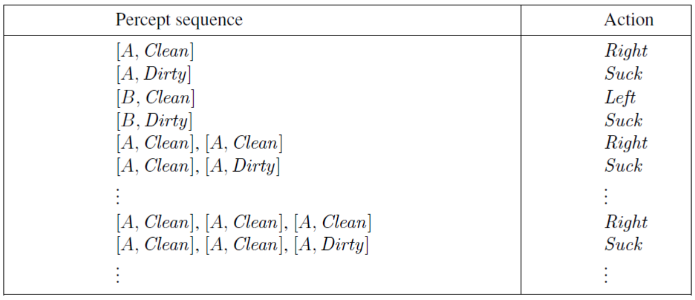
❗ A rational agent acts to achieve best expected outcome (some form of utility)

3.1.1 Agent-Environment Interface

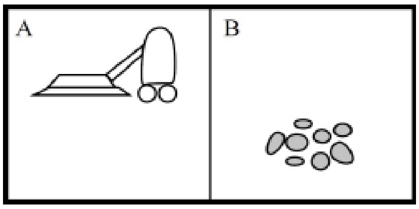
Agents perceive the environment through *sensors* and act in the environment through *actuators*.

Mathematically speaking, we say that an agent’s behavior is described by the *agent function* that maps any given *percept sequence* (complete history of everything the agent has ever perceived) to an action. The agent function specifies the agent’s action in response to every possible percept sequence.





Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮



The *agent program* is an implementation of the agent function. The program, which combined with a machine architecture, implements an agent function.

❗ The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

3.1.2 Concept of Rationality

“Rational agents do the right thing”. Conceptually speaking, every entry in the table for the *agent function* is filled out correctly (see picture above).

A sequence of an agent’s actions causes the environment to go through a sequence of *states.* A *performance measure* objectively evaluates the ‘goodness’ of a sequence of environment states.

- The environment’s designer must come up with individual problem-specific performance measures.
- How we define the performance measure influences the actions of the agent.

❗ If we define success only in terms of agent’s opinion of its own performance, an agent could achieve perfect rationality simply by deluding itself that its performance was perfect. Human agents in particular are notorious for “sour grapes” — believing they did not really want something after not getting it.

💡 As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.

📝 **Basis for Agent’s Decisions**

What is *rational* at a given time depends on:

1. Performance measure
2. Agent’s prior knowledge of the environment
3. Actions the agent can perform
4. Agent’s percept sequence to date

📝 **Rationality/Rational Agents**

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rationality maximizes expected performance, while perfection maximizes actual performance.

📝 **Autonomy**

A property of agents whose behavior is determined by their own experience rather than solely by their initial programming.

💡 A rational agent should be autonomous — it should learn what it can to compensate for partial or incorrect prior knowledge. After sufficient experience of its environment, the behavior of a rational agent can become effectively independent of its prior knowledge. Hence, the incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments.

🤔 Consider the lowly dung beetle. After digging its nest and laying its eggs, it fetches a ball of dung from a nearby heap to plug the entrance. If the ball of dung is removed from its grasp *en route*, the beetle continues its task and pantomimes plugging the nest with the nonexistent dung ball, never noticing that it is missing. Evolution has built an assumption into the beetle’s behavior, and when it is violated, unsuccessful behavior results.

3.1.3 Utility

📝 An agent’s *utility function* is basically an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

💡 Whereas the is always a performance measure, an agent does not necessarily have to have an explicit utility function at all. Furthermore, it is not necessary to have a utility function to be rational. Technically speaking, a rational utility-based agent chooses the action that maximizes the expected utility of the action’s outcomes.

3.2 Environments

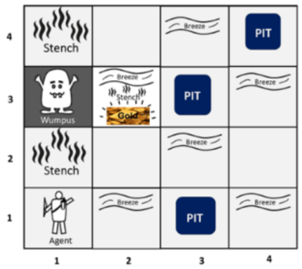
- Small set of *design principles* for agents
- The interaction between an agent and its environment is important
- Behavior of agent depends crucially on the *properties* of the environment

Properties of Environments

Fully observable	Agents’ sensors give it access to the complete state of the environment. → Chess
Partially observable	Agents’ sensors do not give it access to the complete state of the environment at each point in time. → Poker
Single agent	The key distinction is whether other agents’ behavior is best described as maximizing a performance measure whose value depends on agents’ behavior. Agents operating by themselves in an environment. → Vacuum cleaner
Multi agent	Agents is not operating by itself in an environment (cooperative, competitive). → Taxi driver
Deterministic	The next state of the environment is completely determined by the current state and the action executed by the agent. → Chess
Stochastic	The next state is not fully determined by the current state and the agents action. → Taxi driving
Episodic	In each episode the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in previous episodes. → Classification tasks
Sequential	The episodes are connected. → Chess
Static	The environment does not change while the agent is thinking. → Chess
Dynamic	The environment does change while the agent is thinking. → Taxi driver
Discrete	Agents’ actions and perceptions are limited. → Chess
Continuous	Agents’ actions and perceptions are unlimited. → Taxi driving
Known ***Unknown***	*This distinction refers not to the environment itself but to the agent’s state of knowledge about the “laws of physics” of the environment. In a known environment, the outcomes (or outcome probabilities) for all actions are given. In an unknown environment the agent will have to learn how it works to make good decisions.*

Example: The Wumpus world

The Wumpus world is *partially observable*, because the agent can only perceive the close environment such as an adjacent room. It is *deterministic*, as the next state is completely determined by the current state + action. The order is important, so it is *sequential*. It is *static* as the Wumpus and pits are not moving. The environment is *discrete*. The environment is *single agent* as we have one agent only and Wumpus is not considered as an agent.



3.3 Structure of Agents

📝 *Architecture* refers to some sort of computing device with physical sensors and actuators on which the *agent program* (implementation of the agent function) will run.

- $Agent = Architecture + Program$

In designing an agent, the first step must always be to specify the task environment as fully as possible. It consists of:

- **PEAS**: Environment, Actuators, Sensors, Performance measure

💡 The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table (agent function).

3.4 Types of Agents

- *Simple reflex agents* ignore percept history, their action depends solely on the current percept.
- *Model-based reflex agents* derive their actions directly from an internal model of the current world state that is updated over time.
- *Goal-based agents* select their actions so that they believe it will achieve explicitly represented goals.
- *Utility based agents* select actions that they believe will maximize the expected utility of the outcome state.

💡 All types of agents can improve their performance by learning from their own experiences.

3.5 Keywords

- Agent
- Percept sequence
- Agent function
- Agent program
- Performance measure
- Utility function
- Architecture
- Environment
- Task Environment
- Rationality
- Autonomy

4 Probability

An agent may never know for certain in which state it is on or where it will end up after a sequence of action. Therefore, agents need to handle uncertainty, such as partially observable environments or nondeterminism.

4.1 Sources of Uncertainty

Example of uncertain reasoning: Doctor (agent) wants to diagnose a patient that has toothache. He has defined 3 rules to find the cause of it:

1. Toothache → Cavity

   This rule is wrong! Not all patients witch toothaches have cavities. Some of the have gum problems, an abscess, or other problems. The rule misses many other possible causes.

2. Toothache → Cavity or GumProblem or Abscess or else…

   To make this rule true we have to add an almost infinite list of possible problems. It describes it adequately but does not help for coming to a decision

3. Cavity → Toothache

   This causal rule is not right either as not all cavities cause toothaches. This is only true in certain situations.

Trying to use logic to cope with a domain like medical diagnosis fails for three main reasons:

Laziness

It is too much work to make rules *exceptionless* and it is too hard to use such rules.

Theoretical ignorance

Medical science has no *complete* theory of the domain.

Practical ignorance

Even if we knew all the rules, we might still be uncertain about a specific patent due to uncertainty of tests and difficulty to run all possible tests.

💡 **Summarizing Uncertainty**

Often there is no *purely logical* relationship between different propositions. Instead, agents have *degrees of belief* in the relevant sentences. To deal with degrees of beliefs in a well-formalized way we use **probability theory**. It provides a way of summarizing the uncertainty that comes from our laziness and ignorance.

📑 **Degrees of Belief**

Degrees of belief formally represent the strength with which we believe the truth of various propositions. The higher an agent’s degree of belief for a particular proposition, the higher is its confidence in the truth of that proposition.

4.2 Interpretations of Probability

4.2.1 Frequentist Interpretation

📝 Probabilities are frequencies of events. The probability of an event is the fraction of times the event occurs if repeated indefinitely.

❗ But: What about “the probability that Germany will win the world cup is 0.2”?

4.2.2 Subjective Interpretation

📝 Probabilities are *subjective degrees of belief.* Meaning that each agent that has to make a decision has a subjective belief about the state of the world. This belief influences the decisions the agents make.

❗ The drawback of this interpretations is the lack of explanation of what exactly it means to hold a particular degree of belief. We need an explanation for how these beliefs are reflected in our actions.

4.2.3 Bayesian Interpretations

💡 One possible way of attributing degrees of belief is through betting games. For example, you belief that $P(\text{“}GER\ wins\ WC\text{”}) = 0{,}8$ and you would be willing to place a bet of $1 against $3 because the expected gain is 20ct (5 games, 4 won, 1 lost = $1 gain). So, by finding which bets agents are willing to place we can assess the agent’s degree of belief.

4.2.4 Optimality of Probability Theory

📝 If an agent does not follow the rules of probability when placing bets, we can construct a series of bets that would result in a sure negative outcome for it. Therefore, a rational agent must hold degrees of belief that satisfy the rules of probability.

4.2.5 Uncertainty and Rational Decisions

Agens must first have preferences between different possible outcomes (completely specified state) of the plans. *Utility theory* is used to represent and reason with preferences.

- Every state has a degree of usefulness/utility
- Preferences are relative to a specific agent

Preferences, expressed by utilities, are combined with probabilities in the general *theory of rational decisions*.

- $decision\ theory = probability\ theory + utility\ theory$

💡 Agents are rational, if and only if, they chose the action that yields the highest expected utility, averaged over all possible outcomes of the action. → Principle of maximum expected utility (MEU)

4.3 Probability Theory

4.3.1 Basic Probability Notation

📝 Probabilities are defined over possible worlds. They define about how *probable* possible worlds are. The set of all possible worlds is called the *sample space*, denoted with Ω.

Example: Throwing a die: $\Omega = \{1, 2, 3, 4, 5, 6\}$

📝 There are measurable events (propositions), which are subsets of Ω (e.g., {6} or {1, 3, 6} in the dice example). The set of all possible events is called the *event space*.

4.3.2 Event Spaces

Probability theory requires that every event space S satisfies three basic properties:

- Contains the empty event ∅ and the trivial event Ω
- It is closed under union: if $\alpha, \beta \in S\ then\ \alpha \cup \beta \in S$
- It is closed under complementation: if $\alpha \in S\ then\ so\ is\ \Omega - \alpha$

This implies that event spaces are also closed under other operations such as intersection and difference.

4.3.3 Probability Distributions

A probability distribution ***P*** over (Ω, S) is a mapping from events in S to real values that satisfy the following conditions:

- $(P(\alpha) \geq 0, for\ all\ \alpha \in S$ (means that probabilities are not negative)
- $P(\Omega) = 1$
- $if\ \alpha, \beta \in S\ and\ \alpha \cap \beta = \emptyset, then\ P(\alpha \vee \beta) = P(\alpha) + P(\beta)$
  (means that the probability of two mutually disjoint events is the sum of the probabilities of each event)

📝 ***P*** indicates that a result is a vector of number and that there is predefined ordering.

$$P(weather = sunny) = 0.5 \qquad P(weather = cloudy) = 0.2$$

$$P(weather = rainy) = 0.2 \qquad P(weather = snowy) = 0.1$$

$$\text{can be rewritten as } \boldsymbol{P}(weather) =< 0.5, 0.2, 0.2, 0.1 >$$

📝 **P** defined a probability distribution for the random variable *weather* in this case. It is also used for conditional distributions: $P(X|Y)$ gives the values of $P(X = x_i|Y = y_i)$ for each possible i, j pair.

💡 In addition to distributions on single variables, we need notation for distributions on multiple variables. Commas are used for this. For example, $\boldsymbol{P}(Weather, Cavity)$ denotes the probabilities of all combinations of the values of Weather and Cavity. This is a 4×2 table of probabilities called the *joint probability distribution* of Weather and Cavity. We can also mix variables with and without values; $\boldsymbol{P}(sunny, Cavity)$ would be a two-element vector giving the probabilities of a sunny day with a cavity and a sunny day with no cavity.

4.3.4 Propositions

In AI, events are described by *propositions.* For each proposition, the corresponding set contains just those possible worlds in which the proposition holds.

$$\text{For any proposition } \Phi, P(\Phi) = \sum_{\omega \in \Phi} P(\omega)$$

Example

Throwing two dice; The proposition $SUM = 11$ corresponds to worlds $\{(5,6),\ (6,5)\}$

$$P(SUM = 11) = P(\{5,6\}) + P(\{6,5\})$$

4.3.5 Conditional Probability (including observations)

Probabilities such as $P(SUM = 11)$ are called *unconditional* or *prior* probabilities. In most cases though, we have prior *evidence* (observations). Then, we are interested in the *conditional* or *posterior* probability.

Example (1/2) - What is the probability of $SUM = 11$, given that the first die shows 5?

$$P(SUM = 11 | Die_1 = 5)$$

Conditional probabilities are defined in terms of unconditional probabilities:

$$P(A|B) = \frac{P(A,B)}{P(B)} \qquad P(A,B|C) = \frac{P(A,B,C)}{P(C)} \qquad P(A|B,C) = \frac{P(A,B,C)}{P(B,C)}$$

Product rules

Regular

$$P(a,b) = P(a|b) * P(b)$$

Conditionalized

$$P(a,b|c) = P(a|b,c) * P(b|c)$$

Example (2/2) – Solution using definition of conditional probabilities.

$$P(SUM = 11 | Die_1 = 5) = \frac{P(SUM=11, Die_1=5)}{P(Die_1=5)} = \frac{\frac{1}{36}}{\frac{1}{6}} = \frac{1}{6}$$

Given $Die_1 = 5$ is true and we have no further information, then $SUM = 11$ is true with probability of $\frac{1}{6}$ as it only relies on the second die landing on 6.

4.4 Inclusion Exclusion Principle

This rule is easily remembered by noting that the cases where a holds, together with the cases where b holds, certainly cover all the cases where $a \vee b$ holds; but summing the two sets of cases counts their intersection twice, so we need to subtract $P(a \wedge b)$.

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

4.5 Probabilistic Reasoning

4.5.1 Random Variables

- Variables in probability theory are called *random variables*.
- Each random variable has a *domain* – the set of possible values it can take on
- We write $P(cavity)$ for $P(cavity = true)$ in case of *Boolean random variables*
- We can define *probability distributions* for one random variable to specify the probability of all possible values.

4.5.2 Joint Distributions

A possible *world* is defined to be an *assignment* of values to all the random variables under consideration.

Probability distributions on multiple events are called *joint probability distributions. This r*esults in a big table where all combinations of values of the random variables and the respective probabilities are listed.

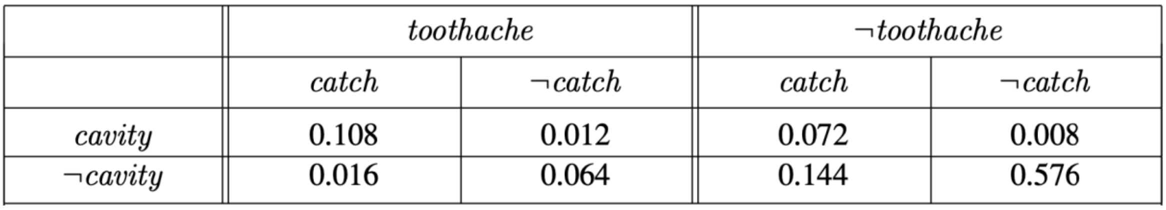
Probability models are *completely determined* by the *full joint probability distribution* over all random variables. $\boldsymbol{P}(X|Y)$ denotes the probability of all combinations of the values of X and Y. This is a 2×2 (for binary X and Y) table of probabilities.

Probabilistic Inference

The compilation of posterior probabilities for query propositions given observed evidence. Probabilistic inference can be used to calculate the probability of any proposition, simple or complex, by identifying those possible worlds in which the proposition is true and add up their probabilities.

Example (1/2)

The *full joint distribution* over 3 random variables (*cavity, toothache* and *catch*) consists of 8 values, each representing the possibility of a single possible world in our domain.



	toothache		*¬toothache*	
	catch	*¬catch*	*catch*	*¬catch*
cavity	0.108	0.012	0.072	0.008
¬cavity	0.016	0.064	0.144	0.576

There are six possible worlds where $P(cavity \vee toothache)$ holds:

$$P(cavity \vee toothache) = 0.108 + 0.012 + 0.016 + 0.064 + 0.072 + 0.008 = 0{,}24$$

Respectively there are only 2 worlds, where $P(cavity \wedge toothache)$ holds:

$$P(cavity, toothache) = 0{,}108 + 0{,}012 = 0{,}12$$

4.5.3 Marginalization

📝 To sum up the probabilities for each possible value of the other variables, thereby taking them out of the equation, is called *marginalization/summing out*.

Original – using joint probabilities

$$\boldsymbol{P}(\boldsymbol{Y}) = \sum_{\boldsymbol{z}\in\boldsymbol{Z}} \boldsymbol{P}(\boldsymbol{Y}, \boldsymbol{z})$$

Variant – using conditional probabilities

$$\boldsymbol{P}(\boldsymbol{Y}) = \sum_{\boldsymbol{z}\in\boldsymbol{Z}} \boldsymbol{P}(\boldsymbol{Y}|\boldsymbol{z}) * \boldsymbol{P}(\boldsymbol{z})$$

$\sum_{z\in Z}$ means to sum over all the possible combinations of values of the set of variables Z.

Example (2/2) - In this case:

$$P(cavity) = \sum_{z\,\in\,\{Catch, Toothache\}} P(cavity, z)$$

$$= P(cavity, catch, toothache) + P(cavity, \neg catch, toothache) + P(cavity, catch, \neg toothache) + P(cavity, \neg catch, \neg toothache)$$

$$= 0{,}108 + 0{,}012 + 0{,}072 + 0{,}008 = 0{,}2$$

4.5.4 Normalization

Furthermore, *conditional probabilities* can be computed from unconditional ones:

$$P(cavity|toothache) = \frac{P(cavity, toothache)}{P(toothache)} = \frac{0.12}{0.108+0.012+0.016+0.064} = 0.6$$

Just to check, we can also compute the probability that there is no cavity, given a toothache:

$$P(\neg cavity|toothache) = \frac{P(\neg cavity, toothache)}{P(toothache)} = \frac{0.016+0.064}{0.108+0.012+0.016+0.064} = 0.4$$

The two values sum to 1.0, as they should. Notice that in these two calculations the denominator remains constant, no matter which value of $Cavity$ we calculate.

If we’re interested in the full distribution over the possible *Cavity* variables and not just the value of $cavity = true$, we can save a part of the computation.

The constant term $\frac{1}{P(tootache)} = \frac{1}{0{,}2}$ can be viewed as a **normalization** constant α for the distribution $\boldsymbol{P}(Cavity \mid toothache\ )$, ensuring that it adds up to 1. Throughout the chapters dealing with probability, we use α to denote such constants. With this notation, we can write the two preceding equations in one:

$$\begin{aligned} \boldsymbol{P}(Cavity|toothache) &= \alpha * \boldsymbol{P}(Cavity, toothache) \\ &= \alpha * [\boldsymbol{P}(Cavity, toothache, catch) + \boldsymbol{P}(Cavity, toothache, \neg catch)] \\ &= \alpha * [\langle 0.108, 0.016\rangle + \langle 0.012, 0.064\rangle] = \alpha\langle 0.12, 0.08\rangle = \langle 0.6, 0.4\rangle \end{aligned}$$

In other words, we can calculate $\boldsymbol{P}(Cavity|toothache)$ even if we don’t know the value of $P(toothache)$! We temporarily forget about the factor $1/P(toothache)$ and add up the values for $cavity$ and $\neg cavity$, getting 0.12 and 0.08. Those are the correct relative proportions, but they don’t sum to 1, so we normalize them by dividing each one by $0.12 + 0.08$, getting the true probabilities of 0.6 and 0.4.

4.6 Stochastic Independence

4.6.1 Absolute independence

Two random variables are *independent* if:

$$\mathbf{P}(X|Y) = \mathbf{P}(X) \text{ or}$$

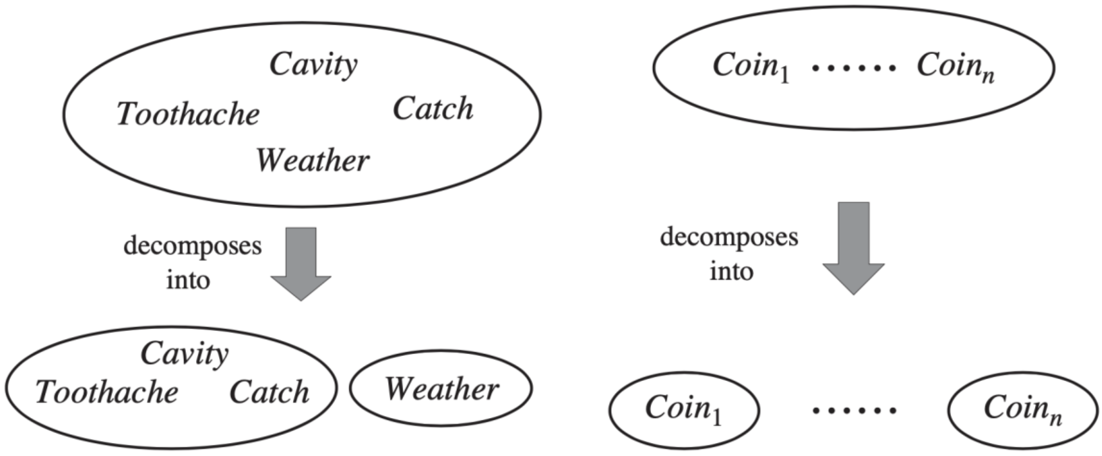
$$\mathbf{P}(Y|X) = \mathbf{P}(Y) \text{ or}$$

$$\mathbf{P}(X,Y) = \mathbf{P}(X) * \mathbf{P}(Y)$$

Independence assertions are usually based on knowledge of the domain. As the toothache – weather example illustrates, they can dramatically reduce the amount of information necessary to specify the full joint distribution. For example:

$$P(toothache, cavity, catch., cloudy) = P(cloudy) * P(toothache, cavity, catch)$$

This also can be generalized to sets of random variables.



4.6.2 Conditional Independence

Two random variables *X, Y* are *conditionally independent* given *Z* if:

$$P(X|Y,Z) = P(X|Z) \text{ or}$$

$$P(X,Y|Z) = P(X|Z)P(Y|Z)$$

X and Y become independent as soon as we know the value of Z

Example

Rolling 3 dice D_1, D_2, D_3, where Y is the value of D_1, Z is the sum of D_1 and D_2, X is the sum of all 3 dice D_1, D_2, D_3. The value of the sum of all 3 dice (*X*) is not independent from the value of the first die D_1 (*Y*). But as soon as we know the sum of D_1 and D_2 (*Z*), they become independent.

This also can be generalized to sets of random variables.

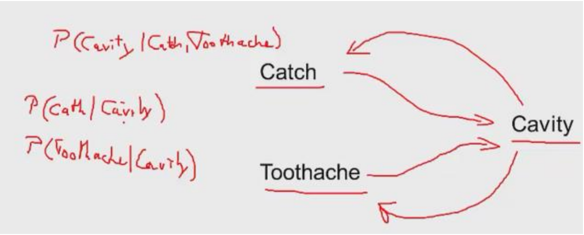
Conditional independencies are much more common than absolute independencies. Conditional independence assertions allow probabilistic systems to scale up.

4.7 Bayes' Rule and Diagnostic Reasoning

❗ Decision Agents must draw conclusions about the state of the world from observations. E.g.:

$$P(cavity \mid catch, toothache)$$

In typical situations, what we know about the world are not the symptoms and are looking for the probability of the cause. We rather have knowledge about the causal relation between symptoms and the effects. So, what we know is $P(catch|cavity)$ and/or $P(toothache|cavity)$.



❓ How can we use the knowledge we have about the world and its causal relations, to do diagnostic reasoning that gives us a true state of the world, given our observations.

4.7.1 Bayes' Rule

General form:

$$\boldsymbol{P}(Y|X) = \frac{\boldsymbol{P}(X|Y) * \boldsymbol{P}(Y)}{\boldsymbol{P}(X)}$$

With background evidence e*:*

$$\boldsymbol{P}(Y|X, e) = \frac{\boldsymbol{P}(X|Y, \boldsymbol{e}) * \boldsymbol{P}(Y, \boldsymbol{e})}{\boldsymbol{P}(X, \boldsymbol{e})}$$

$$\left(= \frac{\boldsymbol{P}(X|Y, e) * \boldsymbol{P}(Y|\boldsymbol{e})}{\boldsymbol{P}(X|\boldsymbol{e})}\right)$$

On the surface, Bayes' rule does not seem very useful. It allows us to compute the single term $P(b|a)$ in terms of three terms: $P(a|b)$, $P(b)$, and $P(a)$. That seems like two steps backwards, but Bayes' rule is useful in practice because there are many cases where we do have good probability estimates for these three numbers and need to compute the fourth. Often, we perceive as evidence the *effect* of some unknown *cause* and we would like to determine that cause. In that case, Bayes' rule becomes.

$$P(cause \mid effect) = \frac{P(effect \mid cause)P(cause)}{P(effect)}$$

Example

Determining the probability of meningitis for a patient who has a stiff neck.

- s is the proposition that patient has a stiff neck & m that patient has meningitis
- Meningitis causes a patient to have a stiff neck in 50% of the cases → $P(s|m) = \frac{1}{2}$
- Prior probability that a patient has meningitis is 1 in 50000 → $P(m) = \frac{1}{50000}$
- Prior probability that a patient has a stiff neck is 1 in 20 → $P(s) = \frac{1}{20}$

$$P(m|s) = \frac{P(s|m) * P(m)}{P(s)} = \frac{\frac{1}{2} * \frac{1}{50000}}{\frac{1}{20}} = 0{,}0002 \; or \; 0{,}02\%$$

5 Probabilistic Graphical Models

5.1 Bayesian Networks

📝 A data structure representing the dependencies among variables. It can represent essentially any full joint probability distribution.

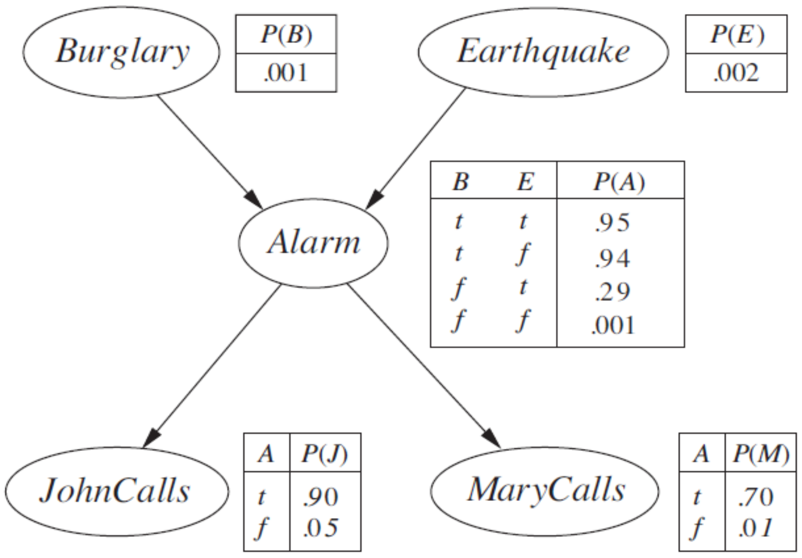
- Each *node* corresponds to a random variable, which may be discrete or continuous
- A set of directed links connect pairs of nodes.
  - If there is a link from node X to node y, X is said to be the *parent* of Y.
  - The graph has no cycles (directed acyclic graph)
- Each node X, has a conditional probability distribution $\boldsymbol{P}(X_i|parents(X_i))$

❗ A Bayesian network is a correct representation of the domain only if each node is conditionally independent of its other predecessors in the node ordering, given its parents.

$$\boldsymbol{P(x_1,\dots,x_n)} = \prod_{\boldsymbol{i}} \boldsymbol{P(x_i|\,parent(X_i))}$$

Example: Burglar Alarm

- An alarm A responds to burglars B often and sometimes to earthquakes E.
- Neighbor John J often hears the alarm A but sometimes confuses it with the telephone.
- Neighbor Mary M often misses the alarm A (due to loud music)



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Probabilities for each event of the example. Events “JohnCalls” and “MaryCalls” are independent from “Burglary” and “Earthquake” given the value of “Alarm”.

John and Mary do not notice Burglars or Earthquakes themselves; they only respond to the alarm.

The conditional distributions are shown as a ***conditional probability table****, or* ***CPT****. Each row in a CPT contains the conditional probability of each node value for a conditioning case.*

A conditioning case is just a possible combination of values for the parent nodes.

❓ What is the probability that the alarm (a) goes off and both John (j) and Mary (m) call, but there is neither a burglary $(\neg b)$ nor an earthquake $(\neg e)$?

💡 Searching $P(j,m,a,\neg b,\neg e)$ by using $P(x_1,\dots,x_n\,) = \prod_i P(x_i\mid parent(X_i))$:

$$\begin{aligned}P(j,m,a,\neg b,\neg e)&\\ &= P(john|alarm) * P(mary|alarm) * P(alarm|\neg burglar,\neg earthquake)\\ &* P(\neg burglar) * P(\neg earthquake) = 0.9 * 0.7 * 0.001 * 0.999 * 0.998\\ &= \boldsymbol{0.00063}\end{aligned}$$

5.1.1 Constructing Bayesian Networks

Algorithm

1. Determine the *set of variables* that are required to model the domain
2. Order them from X_1 to X_n
3. For *i*=1 to *n*:
    a. Choose from $X_1, \dots, X_{i-1}$ (all the predecessors of X_i) a minimal set of parents for X_i, such that $\prod_i P(x_i|x_{i-1}, \dots, x_i) = \prod_i P(x_i \mid parent(X_i))$ is true.
    b. For each parent insert a link the parent to X_i
    c. CPTs: write down the conditional probability table $P(x_i \mid parent(X_i))$

5.1.2 Reasoning in Bayesian Networks

❓ Given a probabilistic graphical model we want to compute the probability of query variables given some evidence. For example, $P(Burglary \mid Alarm) = ?$

📝 From the example in *4.5.4 Normalization*, we can extract a general inference procedure. We begin with the case in which the query involves a single variable, X ($Cavity$ in the example). Let $\boldsymbol{E}$ be the list of evidence variables (just $Toothache$ in the example), let $\boldsymbol{e}$ be the list of observed values for them, and let $\boldsymbol{Y}$ be the remaining unobserved variables (just $Catch$ in the example). The query is $\boldsymbol{P}(X \mid \boldsymbol{e})$ and can be evaluated as:

$$\boldsymbol{P}(X|\boldsymbol{e}) = \alpha * \sum_{y} P(X, \boldsymbol{e}, \boldsymbol{y})$$

where the summation is over all possible $\boldsymbol{y}$s (i.e., all possible combinations of values of the unobserved variables $\boldsymbol{Y}$). Notice that together the variables $X, \boldsymbol{E}$, and $\boldsymbol{Y}$ constitute the complete set of variables for the domain, so $\boldsymbol{P}(X, \boldsymbol{e}, \boldsymbol{y})$ is simply a subset of probabilities from the *full joint distribution*.

Notation:

- X is a query variable
- $\boldsymbol{E}$ is a set of observed (evidence) variables
- $\boldsymbol{Y}$ is set of remaining unobserved / hidden (non-evidence) variables
- $\boldsymbol{X}$ is complete set of variables $\boldsymbol{X} = \{X\} \cup \boldsymbol{E} \cup \boldsymbol{Y}$

A typical query would be $P(X, \boldsymbol{E} = \boldsymbol{e})$. This is called the marginal, a-posteriori probability of the variable X given evidence $\boldsymbol{e}$.

5.1.3 Inference by Enumeration

For a Bayesian network the joint distribution can be written as a product of conditional probabilities. For example:

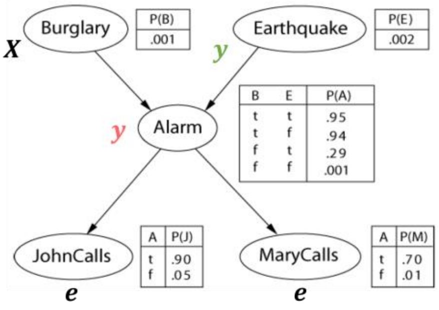
$$P(burglary|john, mary) = P(b|j,m)$$

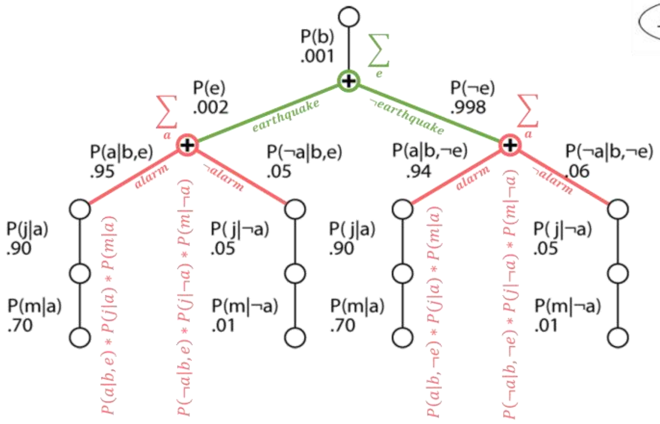
$$= \alpha * \sum_{y} P(\overset{\text{query variable}}{X}, \overset{\text{evidence variables}}{\boldsymbol{e}}, \overset{\text{hidden variables}}{\boldsymbol{y}})$$

↓ by using $P(x_1, \dots, x_n) = \prod_i P(x_i \mid parent(X_i))$

$$= \alpha * \sum_{a} \sum_{e} P(b) * P(e_i) * P(a_i|b,e_i) * P(j|a_i) * P(m|a_i)$$

$$= \alpha * P(b) * \sum_{e} P(e_i) * \sum_{a} P(a_i|b,e_i) * P(j|a_i) * P(m|a_i)$$





5.1.4 Inference by Variable Elimination

💡 If the Bayesian Network N is *single connected* (there is only one path between a pair of nodes), then, the complexity is in $O(n)$, with n being the number of random variables. If there are multiple paths, the complexity depends on the order of the variable elimination. In the worst case it is exponential $O(2^n)$.

❗ Using inference by enumeration results in an internal structure due to the summation over possible assignments, where certain sub-formulae can be re-used:

$$P(b|j,m) = \alpha * P(b) * \sum_{e_i \in E} P(e_i) * \sum_{a} P(a_i|b,e_i) * P(j|a_i) * P(m|a_i) =$$

$$\alpha * P(b) \begin{pmatrix} P(e) * \big(P(a|b,e) * P(j|a) * P(m|a) + P(\neg a|b,e) * P(j|\neg a) * P(m|\neg a)\big) + \\ P(\neg e) * \big(P(a|b,\neg e) * P(j|a) * P(m|a) \; + \; P(\neg a|b,\neg e) * P(j|\neg a) * P(m \,\big|\, \neg a)\big) \end{pmatrix}$$

These sub-formulae can be calculated and stored as intermediate results → **Factors**.

An n-dimensional ***factor*** f is a representation of a function from n random variables $X_1, \dots, X_n$ to a positive real number. A factor can be a probability distribution (summing up to 1) but does not need to be. Its notation for a factor f over $X_1, \dots, X_n$ is $f(X_1, \dots, X_n)$.

X	Y	Z	*val*
⊤	⊤	⊤	0.1
⊤	⊤	⊥	0.9
⊤	⊥	⊤	0.2
⊤	⊥	⊥	0.8
⊥	⊤	⊤	0.4
⊥	⊤	⊥	0.6
⊥	⊥	⊤	0.3
⊥	⊥	⊥	0.7

$f(X, Y, Z)$

Factors can have ***assigned values*** to its variables:

$f^{X=t}$:

Y	Z	*val*
⊤	⊤	0.1
⊤	⊥	0.9
⊥	⊤	0.2
⊥	⊥	0.8

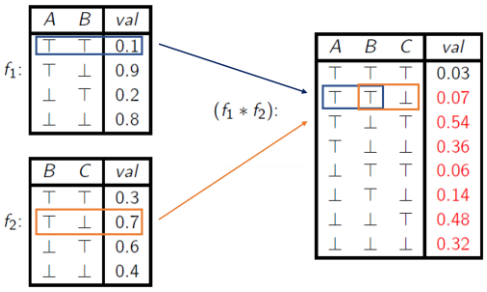
$f^{X=t,Z=f}$:

Y	*val*
⊤	0.9
⊥	0.8

$f^{X=t,Y=f,Z=f} = 0.8$

The ***product*** of factor $f_1(X, Y)$ and $f_2(Y, Z)$, where Y are the variables in common, is the factor $(f_1 * f_2)(X, Y, Z)$ defined by:

$$(\boldsymbol{f_1} * \boldsymbol{f_2})(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}) = \boldsymbol{f_1}(\boldsymbol{X}, \boldsymbol{Y})\boldsymbol{f_2}(\boldsymbol{Y}, \boldsymbol{Z})$$



f_1:

A	B	*val*
⊤	⊤	0.1
⊤	⊥	0.9
⊥	⊤	0.2
⊥	⊥	0.8

f_2:

B	C	*val*
⊤	⊤	0.3
⊤	⊥	0.7
⊥	⊤	0.6
⊥	⊥	0.4

$(f_1 * f_2)$:

A	B	C	*val*
⊤	⊤	⊤	0.03
⊤	⊤	⊥	0.07
⊤	⊥	⊤	0.54
⊤	⊥	⊥	0.36
⊥	⊤	⊤	0.06
⊥	⊤	⊥	0.14
⊥	⊥	⊤	0.48
⊥	⊥	⊥	0.32

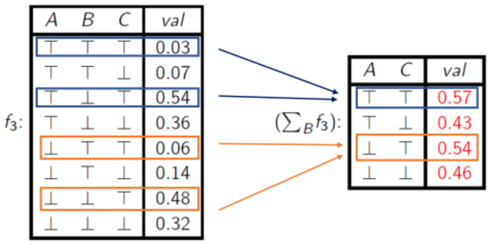
We can ***sum out*** a variable, say X₁ with domain $\{v_1, \dots, v_k\}$, from factor $f(X_1, \dots, X_n)$. This results in a factor on $X_2, \dots, X_j$, defined as follows:

$$\left(\sum_{x_1} f\right) * (X_2, \dots, X_j)$$

$$= f(X_1, \dots, X_j)^{X_1 = v_1} + \cdots$$

$$+ f(X_1, \dots, X_j)^{X_1 = v_k}$$

$$= \sum_{v \in dom(X_1)} f(X_1, X_2, \dots, X_j)^{X_1 = v}$$



f_3:

A	B	C	*val*
⊤	⊤	⊤	0.03
⊤	⊤	⊥	0.07
⊤	⊥	⊤	0.54
⊤	⊥	⊥	0.36
⊥	⊤	⊤	0.06
⊥	⊤	⊥	0.14
⊥	⊥	⊤	0.48
⊥	⊥	⊥	0.32

$(\sum_B f_3)$:

A	C	*val*
⊤	⊤	0.57
⊤	⊥	0.43
⊥	⊤	0.54
⊥	⊥	0.46

$$P(B \mid j, m) = c \cdot \underbrace{\underbrace{P(B)}_{f_1(B)} \cdot \underbrace{\sum_{e' \in dom_E} \underbrace{\underbrace{P(e')}_{f_2(E)} \cdot \underbrace{\sum_{a' \in dom_A} \underbrace{\underbrace{P(a' \mid b, e')}_{f_3(A,B,E)} \cdot \underbrace{\underbrace{\underbrace{P(j \mid a')}_{f_4(J,A)}}_{f_4'(A)=f_4^j} \cdot \underbrace{\underbrace{P(m \mid a')}_{f_5(M,A)}}_{f_5'(A)=f_5^m}}_{f_6(A)=(f_4' * f_5')}}_{f_7(A,B,E)=(f_3 * f_6)}}_{f_8(B,E)=(\sum_A f_7)}}_{f_9(B,E)=(f_2 * f_8)}}_{f_{10}(B)=(\sum_E f_9)}}_{f_{11}(B)=(f_1 * f_{10})}$$

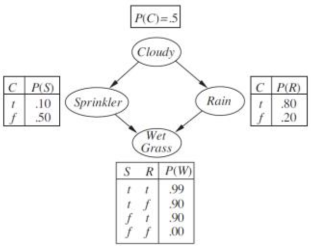
5.2 Stochastic Sampling

❗ Exact inference is intractable for larger multiply connected networks. Therefore, it is necessary to consider more efficient approximation algorithms.

5.2.1 Direct Sampling

Basic Idea: Draw *N* samples from a joint distribution and compute an approximate posterior probability *P’* by looking at the individual samples.

❓ How do we generate samples from the *BN* representation of a probability distribution (samples should be generated in accordance with their probability)?



- **Prior-Sample**: The simplest kind of random sampling process for Bayesian networks generates events from a network that has *no evidence* associated with it. The idea is to sample each variable in turn, in topological order. The probability distribution from which the value is sampled is conditioned on the values already assigned to the variable’s parents.

  1. Sample from $\mathbf{P}(Cloudy) = \langle 0.5, 0.5 \rangle$, value is $true$.
  2. Sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$, value is $false$.
  3. Sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$, value is $true$.
  4. Sample from $\mathbf{P}(WetGrass \mid Sprinkler = false, Rain = true) = \langle 0.9, 0.1 \rangle$, value is $true$.

  In this case, PRIOR-SAMPLE returns the event $[true, false, true, true]$.

  The sampling probability for this event is:

  $$S_{PS}(true, false, true, true) = 0.5 * 0.9 * 0.8 * 0.9 = 0.324$$

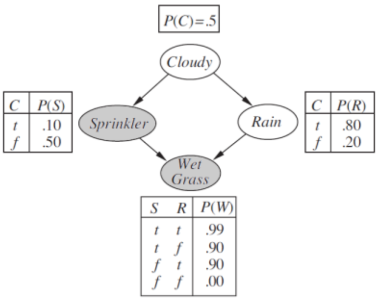
  Hence, in the limit of large N, we expect 32.4% of the samples to be of this event.

  The probability of the event can be estimated as the fraction of all complete events generated by the sampling process that match the partially specified event. For example, if we generate 1000 samples from the sprinkler network, and 511 of them have $Rain = true$, then the estimated probability of rain, written as $\hat{P}(rain) = 0.511$.

- **Rejection-Sample**: pick samples via Prior-Sample but remove samples where evidence does not match the value of the evidence in the query.

  The biggest problem with rejection sampling is that it rejects so many samples! The fraction of samples consistent with the evidence e drops exponentially as the number of evidence variables grows, so the procedure is simply unusable for complex problems.

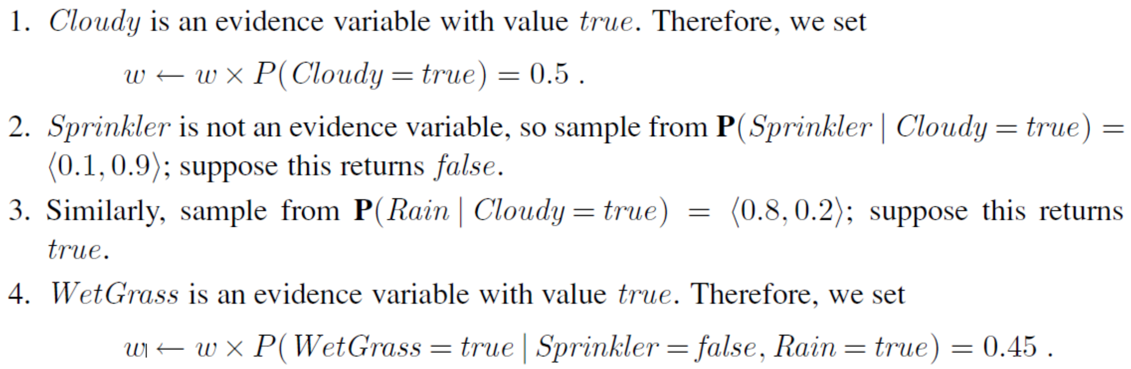
5.2.2 Likelihood Weighting



It is a particular instance of the general statistical technique of *importance sampling*, tailored for inference in Bayesian networks.

It fixes evidence variables beforehand and then samples only the hidden variables. Each resulting event is then weighted by the likelihood that the event occurs **according to the evidence**.

Example for the query $\boldsymbol{P}(Rain|cloudy, wetGrass)$:



1. $Cloudy$ is an evidence variable with value $true$. Therefore, we set

$$w \leftarrow w \times P(Cloudy = true) = 0.5 \ .$$

2. $Sprinkler$ is not an evidence variable, so sample from $\mathbf{P}(Sprinkler \mid Cloudy = true) = \langle 0.1, 0.9 \rangle$; suppose this returns $false$.
3. Similarly, sample from $\mathbf{P}(Rain \mid Cloudy = true) = \langle 0.8, 0.2 \rangle$; suppose this returns $true$.
4. $WetGrass$ is an evidence variable with value $true$. Therefore, we set

$$w \leftarrow w \times P(WetGrass = true \mid Sprinkler = false, Rain = true) = 0.45 \ .$$

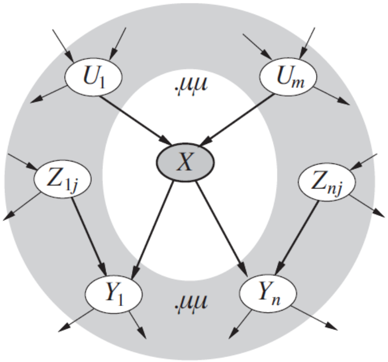
Here the sampling returns the event $[true, false, true, true]$ with weight:

$$P(cloudy) * P(wetGrass|\neg sprinler, rain) = 0.5 * 0.9 = 0.45$$

and this is tallied under $Rain = true$.

5.2.3 Markov Chain Monte Carlo (MCMC) - / Gibbs Sampling

MCMC generates samples by making a random change to the preceding sample. MCMC algorithms traverse a set of states (states correspond to a particular possible world, e.g., in a Bayesian network). The simplest form of MCMC is *Gibbs sampling*.



The topological semantics of a BN specifies that each variable is *conditionally independent* of its non-descendant given its parents.

This implies that a node is conditionally independent of all other nodes in the network, given its *parents, children, and children’s parents* → **Markov blanket.**

$$\begin{aligned} P(X|U_1, \dots, U_m, Y_1, \dots, Y_n, Z_{1j}, \dots, Z_{nj}) \\ &= P(X|MarkovBlanket(X)) \\ &= P(X|all\ variables) \end{aligned}$$

The algorithm wanders randomly around the state space—the space of possible complete assignments—flipping one variable at a time but keeping the evidence variables fixed.

Procedure

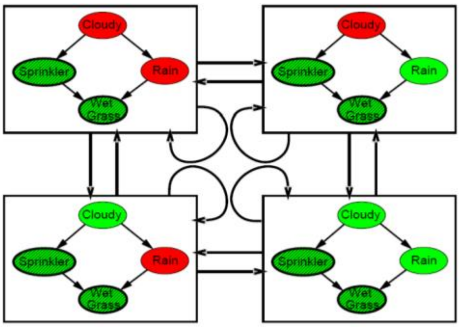
1. Start in State S
2. Randomly pick one nonevidence variable
3. Based on its M blanket, sample selected RV. The sampling is done conditioned on the current values of the variables in the Markov blanket of the RV.
4. Based on sampled RV: move to new state

Example

- MB for *Cloudy* is *Sprinkler* and *Rain*
- MB for *Rain* is *Cloudy*, *Sprinkler* and *WG*

Consider the query $\boldsymbol{P}(Rain|sprinkler, wetGrass)$. The evidence variables Sprinkler and WetGrass are fixed to their observed values and the nonevidence variables Cloudy and Rain are initialized randomly, let us say to true and false respectively. Thus, the initial state is $[true, true, false, true]$. Now the nonevidence variables are sampled repeatedly in an arbitrary order.

With *Sprinkler* = *true*, *WetGrass* = *true*, there are four states:



For example:

1. Cloudy is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\boldsymbol{P}(Cloudy|sprinkler, \neg rain)$. Suppose the result is $Cloudy = false$. Then the new current state is $[false, true, false, true]$.
2. Rain is sampled, given the current values of its Markov blanket variables: in this case, we sample from $\boldsymbol{P}(Rain|\neg cloudy, sprinkler, wetGrass)$. Suppose this yields $Rain = true$. The new current state is $[false, true, true, true]$.

Finally, out of 100 states visited: 31 have $Rain = true$ and 69 have $Rain = false$:

$$\widehat{\boldsymbol{P}}(Rain|sprinkler, wetGrass) = Normalize(\langle 31{,}69\rangle) = \langle 0.31, 0{,}69\rangle$$

5.3 Hidden Markov Models

Useful for modeling systems that evolve over time. Idea is using large Bayesian network with nodes for each time step and specialized inference algorithms that exploit the network structure.

5.3.1 Markov Chains

Assumption: States form a **first-order** Markov chain: State X_t only depends on state X_{t-1}. Furthermore, assume that the nature of that dependency does not change over time, i.e., the dependency can be expressed by a conditional distribution:

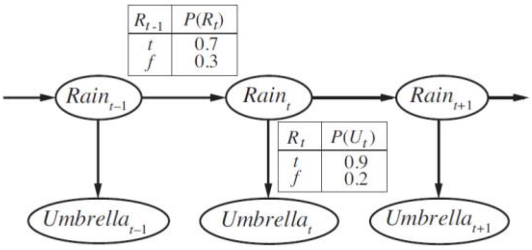
$$\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1}) \rightarrow \text{a } \textit{transition model}$$

Assumption: Observation Y_t depends only on state X_t. The nature of this dependency also does not change over time, it can also be expressed by a conditional distribution:

$$\boldsymbol{P}(\boldsymbol{Y}_t|\boldsymbol{X}_t) \rightarrow \textit{observation model.}$$

A *Hidden Markov Model* is defined by the transition model $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{X}_{t-1})$, the observation model $\boldsymbol{P}(\boldsymbol{Y}_t|\boldsymbol{X}_t)$ and a prior distribution $\boldsymbol{P}(\boldsymbol{X}_1)$

$$\boldsymbol{P}(\boldsymbol{X}_{0:t},\boldsymbol{Y}_{1:t}) = \boldsymbol{P}(\boldsymbol{X}_0)\prod_{i=1}^{t}\boldsymbol{P}(\boldsymbol{X}_i|\boldsymbol{X}_{i-1})\boldsymbol{P}(\boldsymbol{Y}_i\,|\boldsymbol{X}_i)$$



The structure in the figure is a first-order Markov process—the probability of rain is assumed to depend only on whether it rained the previous day.

5.3.2 Inference in HMMs: Filtering

Filtering is the task of computing the belief state—the posterior distribution over the most recent state—given all evidence to date. Filtering is also called state estimation. In our example, we wish to compute $\boldsymbol{P}(\boldsymbol{X}_t|\boldsymbol{y}_{1:t})$. In the umbrella example, this would mean computing the probability of rain today, given all the observations of the umbrella carrier made so far. Filtering is what a rational agent does to keep track of the current state so that rational decisions can be made.

Suppose we already computed the previous time step $P(X_{t-1}|y_1,\dots,y_{t-1})$, now:

- Compute *Prediction* $P(x_t|y_{1:t-1})$ *without* considering new observation y_t

$$\boldsymbol{P(x_t|y_{1:t-1})} = \sum_{x_{t-1}} P(x_t, x_{t-1}|y_{1:t-1})$$

↓ by using the conditionalized product rule $P(X,Y|\,e) = P(X,Y|e) = P(X|Y,e) * P(Y|e)$

$$= \sum_{x_{t-1}} P(x_t|x_{t-1}, y_{1:t-1}) * P(x_{t-1}|y_{1:t-1})$$

by using the independence assumption that observation Y_t depends only on state X_t

$$= \sum_{x_{t-1}} P(x_t|x_{t-1}) * P(x_{t-1}|y_{1:t-1})$$

- Compute *Correction* $P(x_t|y_{1:t-1}, y_t)$

$$P(x_t|y_{1:t-1}, y_t)$$

↓ by using a modified version of the bayes theorem: $P(Y|X,Z) = \dfrac{P(X|Y,Z) * P(Y|Z)}{P(X|Z)}$

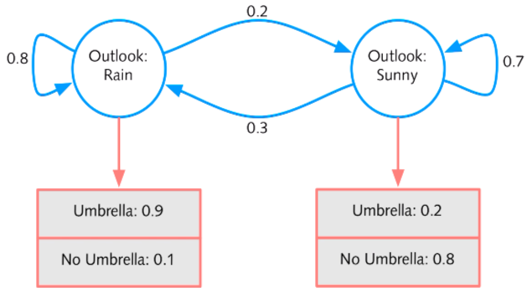
$$= \frac{P(y_t|x_t, y_{1:t-1}) * \boldsymbol{P(x_t|y_{1:t-1})}}{P(y_t|y_{1:t-1})}$$

↓ by using the independence assumption that observation Y_t depends only on state X_t

$$= \frac{P(y_t|x_t) * \boldsymbol{P(x_t|y_{1:t-1})}}{P(y_t)}$$

In the denominator, $P(y_t) = \sum_{x_t} P(y_t|x_t) * P(x_t|y_{1:t-1})$ can be written, which is a normalization factor that we get automatically when we compute the numerator (Zähler) for all x_t in the upper formula.

Example

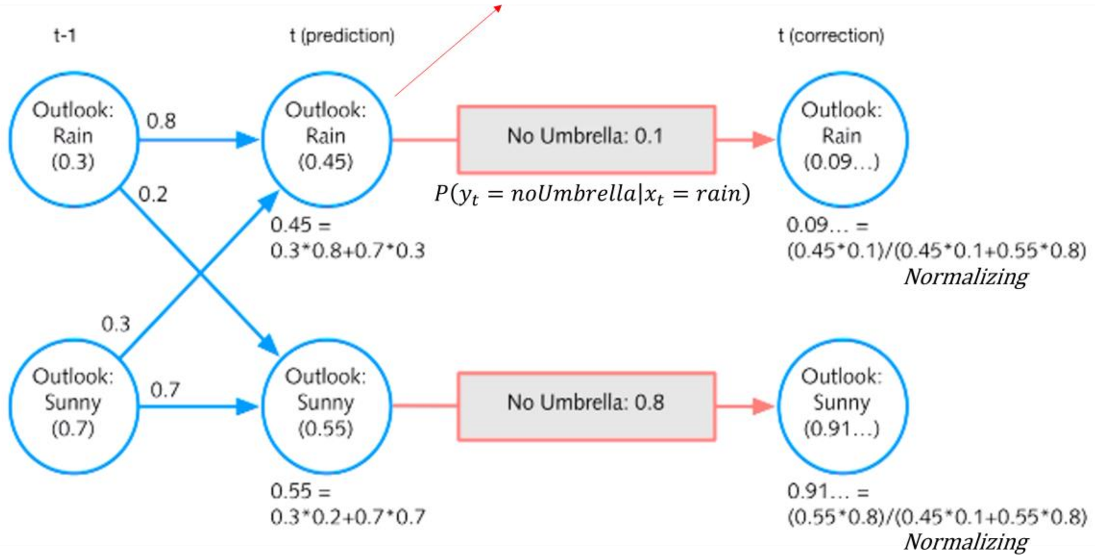


$$\mathbf{A} = \begin{array}{c|cc} & \multicolumn{2}{c}{x_{t-1}} \\ x_t & \mathit{Rain} & \mathit{Sunny} \\ \hline \mathit{Rain} & 0.8 & 0.3 \\ \mathit{Sunny} & 0.2 & 0.7 \end{array} \qquad \mathbf{O} = \begin{array}{c|cc} & \multicolumn{2}{c}{y_t} \\ x_t & \mathit{NoUmbr.} & \mathit{Umbr.} \\ \hline \mathit{Rain} & 0.1 & 0.9 \\ \mathit{Sunny} & 0.8 & 0.2 \end{array}$$

Not in the diagram:

$$p(x_1) = \begin{array}{c|c} \mathit{Rain} & 0.3 \\ \mathit{Sunny} & 0.7 \end{array}$$

$$P(x_t = rain \mid y_{1:t-1}) = \sum_{x_{t-1}} P(rain|x_{t-1}) * P(x_{t-1}|y_{1:t-1})$$
$$= 0.3 * 0.8 + 0.7 * 0.3 = 0.45$$



5.3.3 Inference in HMMs: Smoothing

The estimation of $P(x_t|y_{1:t})$ was called filtering. Another approach is *smoothing*, where we are looking for the probability $P(x_t|y_{1:T})$. The idea is to use the *whole observation sequence* when estimating the state at time t (even observations that occur later than a certain state, for example $y_4$when looking for state x_3, are considered).

$$\begin{aligned}
p(x_t \mid y_{1:T}) &= \sum_{x_{t+1}} p(x_t, x_{t+1} \mid y_{1:T}) && \text{Marginalisation} \\
&= \sum_{x_{t+1}} p(x_t \mid x_{t+1}, y_{1:T}) p(x_{t+1} \mid y_{1:T}) && \text{Chain rule} \\
&= \sum_{x_{t+1}} p(x_t \mid x_{t+1}, y_{1:t}) p_t(x_{t+1} \mid y_{1:T}) && \text{Independence in HMM} \\
&= \sum_{x_{t+1}} \frac{p(x_{t+1} \mid x_t, y_{1:t})\, p(x_t \mid y_{1:t})}{p(x_{t+1} \mid y_{1:t})} p(x_{t+1} \mid y_{1:T}) && \text{Bayes} \\
&= \sum_{x_{t+1}} \frac{p(x_{t+1} \mid x_t)\, p(x_t \mid y_{1:t})}{p(x_{t+1} \mid y_{1:t})} p(x_{t+1} \mid y_{1:T}) && \text{Independence in HMM} \\
&= p(x_t \mid y_{1:t}) \sum_{x_{t+1}} p(x_{t+1} \mid x_t) \frac{p(x_{t+1} \mid y_{1:T})}{p(x_{t+1} \mid y_{1:t})}
\end{aligned}$$

5.4 Sum-Product Networks (SPNs)

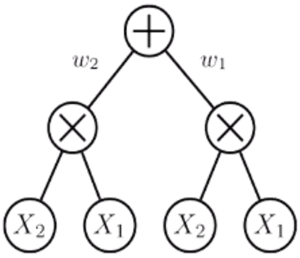
❗ While probabilistic graphical models are good at representing compact distributions, they are not very efficient at computing their marginals and modes.

SPNs are a type of probabilistic models. They are a class of deep probabilistic models that consist of many layers of hidden variables and can have unbounded treewidth. Inference in SPNs is guaranteed to be tractable (structure and parameters can be effectively and accurately learned).

5.4.1 Definition

📝 A *Sum-Product Network* S over *random variables* X is a rooted, weighted, directed acyclic graph consisting of distribution *leaves* (network inputs), *sum* and *product nodes* (inner nodes).

- A leaf n defines a tractable, possibly unnormalized, probability distribution ϕ_n over some RVs in X.
- A non-negative weight w_{nc} is associated to each edge linking a sum node n to $c \in ch(n)$
  - ○ $ch(n)$: child (input) nodes of a node n
  - ○ $pa(n)$: parent (output) nodes of a node n
  - ○ S_n: sub-network rooted at node n



5.4.2 Scopes

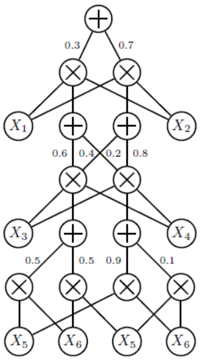
The *scope* of a node n in a SP-Network S is denoted as:

$$sc(n) \subseteq X$$

📝 The scope of a leaf node n is defined as the set of RVs over which ϕ_n is defined. For example: $sc(n) = \{X_6\}$. Meaning the scope is the node itself.

📝 The scope of an inner node n is defined as $sc(n) = \bigcup_{c \in ch(n)} sc(c)$. Meaning it unites the scope of its children c.

📝 The scope of S is the scope of the root, i.e.: X



5.4.3 Structural Properties

Let S be an SPN and let $S_{\oplus}$ (resp. $S_{\otimes}$) be the set of all sum nodes (resp. product nodes) in S.

1. S is *complete*, iff (*if and only if*, ⇔): $\forall\, n \in S_{\oplus}, \forall c_1, c_2 \in ch(n): sc(c_1) = sc(c_2)$
   “For all sum nodes, the children of the sum node have the same scope.”

2. S is *decomposable*, iff: $\forall\, n \in S_{\otimes}, \forall c_1, c_2 \in ch(n), c_1 \neq\ c_2: sc(c_1) \cap sc(c_2) = \emptyset$
   “For all product nodes, all children which are not identical have disjoint scopes.”

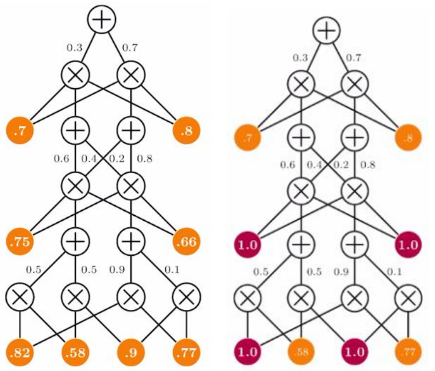
   💡 If an SPN is complete and decomposable it is called *valid*.

5.4.4 Complete Evidence Inference

To compute $p(X = x)$, evaluate S in a bottom-up (feedforward) fashion.

The root node’s output is $S(x) = p(X = x)$

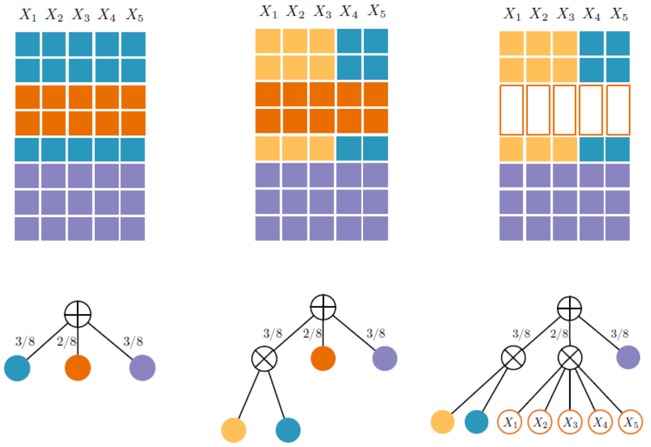
❗ This only works for valid SPNs



5.4.5 Marginal Inference

To compute a marginal query like $p(Q = q), Q \subset X$ evaluate S as before. Variables that should be marginalized get assigned 1 as their value.

5.4.6 Structure Learning


X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5
X_1 X_2 X_3 X_4 X_5
3/8
2/8
3/8
3/8
2/8
3/8
3/8
2/8
3/8
X_1
X_2
X_3
X_4
X_5

💡 Just starting with a data set: learn the whole structure of the SPN.

LearnSPN is a greedy, top down, constraint-based learner for tree SPNs. It builds a tree SPN by recursively splitting the data matrix:

- Splitting columns (when there’s independence) into pairs by a greedy G test with threshold p:

$$G(X_i, X_j) = 2 \sum_{x_i \sim X_i} \sum_{x_j \sim X_j} c(x_i, x_j) * \log \frac{c(x_i, x_j) * |T|}{c(x_i) * c(x_j)}$$

  If there is independence, we can create a product node in the SPN, which multiplies the independent columns.

- Clustering instances into $|C|$ sets with online Hard-EM, estimating weights as proportions with cluster penalty λ

$$p(X) = \sum_{C_i \in C} \prod_{X_j \in X} p(X_j | C_i) * p(C_i)$$

  This splits the data matrix by row and creates a sum node in the SPN.

- If there are less than m instances, put a naïve factorization over leaves
- Each univariate distribution get ML estimation smoothed by α

Hyperparameter space: $\{p, \lambda, m, a\}$

6 Decision Theory and Decision Making

6.1 Definitions

📝 There are actions A, outcomes S, and evidence observations $\boldsymbol{e}$. An essential part of decision theory is to compute the probability of certain outcomes of an action given evidence:

$$P(Result(a) = s' \mid a, \boldsymbol{e})$$

“Probability of outcome *s*’ given evidence observation *e* when action *a* is executed.”

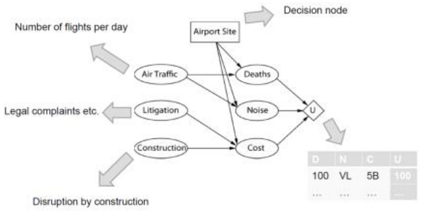
📝 The agents’ preferences are captured by a utility function $U(s)$. The function assigns a single number to express desirability of a state/outcome. This allows to calculate the expected utility EU of an action a given the evidence $\boldsymbol{e}$. The average utility value over all possible outcomes of the action is weighted by probabilities.

$$EU(a|\boldsymbol{e}) = \sum_{s'} P(Result(a) = s' \mid a, \boldsymbol{e}) * U(s')$$

📝 Principle of maximum expected utility (MEU): a rational agent should choose the action a that maximizes the agent’s expected utility EU given observation $\boldsymbol{e}$.

$$action = \underset{\mathrm{a}}{\operatorname{argmax}} EU(a|\boldsymbol{e})$$

6.2 Decision Networks



💡 Are an extension of Bayesian networks with additional decision nodes. A decision network represents information about the agent’s most current state, its possible actions, the state that will result from the actions and the utility of states.

- Chance nodes (oval) represent standard random variables (as in Bayesian Networks)
- Decision nodes (rectangles) represent points where decision maker has choice of actions
- Utility nodes (diamonds) represent the agent’s utility function

6.3 Value of Information

More information typically leads to better decisions but acquiring information if often expensive. Knowing which information is important for a decision is essential.

📝 Knowing the value of certain random variables allows for better choices. Given the evidence *e*, the value of additionally knowing the value of a variable E_j is:

$$VPI_e(e_j) = \left(\sum_k \overbrace{P(e_{jk}|e)}^{\text{likelihood of new evidence } e_j \text{ having value } k} * \overbrace{EU\left(a_{e_{jk}}|e, e_{jk}\right)}^{\text{expected utility given new evidence with value k}}\right) - \underbrace{EU(a|e)}_{\text{previously expected utility}}$$

new evidence

all possible values of the new evidence variable e_j

action derived after obtaining new evidence e_{jk}

Example

- 5 areas can be acquired, only 1 contains oil (worth $\$1$M) →$P(oil) = 0{,}2$ $U(oil) = \$1M$
- Buying drilling rights for an area costs $\$200.000$ → $U(\neg oil) = -\$200k$

Expected profit of (“previously”):

$$EU(a|e) = \sum_{s'} P(Result(a) = s' \mid a, e) * U(s')$$

$$= P(oil) * U(oil) + P(\neg oil) * U(\neg oil) = \frac{1}{5} * (\$1\text{M} - \$200\text{k}) + \frac{4}{5} * (-\$200\text{k}) =$$

$$= \$160\text{k} - \$160\text{k} = \$\mathbf{0}$$

❓ “How much should the company pay for the information whether Block 3 contains oil?

Case 1: Block 3 contains oil with $P(oil) = 0.2$

- Company buys block for $\$200$k and makes $\$800$k profit.

Case 2: Block 3 does not contain oil with $P(\neg oil) = 0.8$

- Company does not buy block 3, $P(oil)$ for other areas changes to 0.25.
- New expected profit becomes $EU(a_{new}|e, \neg oil_{B3}) = \left(\frac{1}{4} * \$1\text{M}\right) - \$200\text{k} = \$50\text{k}$

Calculation:

$$VPI_e(E_j) = \left(\sum_k P(e_{jk}|e) * EU\left(a_{e_{jk}}|e, e_{jk}\right)\right) - EU(a|e)$$

$$= \left(P(oil_{B3}|e) * EU\left(a_{oil_{B3}}|e, oil_{B3}\right) + P(\neg oil_{B3}|e) * EU\left(a_{\neg oil_{B3}}|e, \neg oil_{B3}\right)\right) - EU(a|e)$$

$$= \left(\frac{1}{5} * \$800k + \frac{4}{5} * EU(a_{new}|e, \neg oil_{B3})\right) - \$0$$

$$= \left(\frac{1}{5} * \$800k + \frac{4}{5} * \left(\frac{1}{4} * (\$1\text{M} - \$200\text{k}) + \frac{3}{4} * (-\$200\text{k})\right)\right) - \$0$$

$$= \left(\frac{1}{5} * \$800\text{k} + \frac{4}{5} * (\$200\text{k} - \$150\text{k})\right) = \$200\text{k}$$

💡 The overall expected profit becomes: $\$\boldsymbol{200k}$. The company shouldn’t pay more for this info.

6.4 Markov Decision Processes (MDPs)

Whereas the previous chapter was concerned with one-shot or episodic decision problems, in which the utility of each action’s outcome was well known, we are concerned here with sequential decision problems, in which the agent’s utility depends on a sequence of decisions. Sequential decision problems incorporate utilities, uncertainty, and sensing, and include search and planning problems as special cases.

📝 A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model (S_{t+1} depends only on S_t) and additive rewards is called a *Markov decision process*. MDPs are a special case of decision networks to repeatedly make decisions over time.

6.4.1 System Dynamics

📝 As in HMMs: State s_{t+1} only depends on s_t (previous state) and action a_t (previous action). This dependency can be modeled as the following probability distribution: $P(S_{t+1}|S_t, A_t)$. This is called the *transition model.*

❗ Assumption: Agent making the decision knows its state at each time; *fully observable*.

📝 System setup: At each time step t the agent performs an action a_t and gets a reward $R(s, a)$. The reward function $R: S \times A \rightarrow \mathbb{R}$ specifies “how good” it is for an agent to choose action a in state s. Also denoted as $R(S_t, A_t, S_{t+1})$.

💡 For each point in time t, we want to choose an action → sequential decision problem. Therefore, defining an overall utility for sequences of states and actions is necessary. In principle, utility is the sum of rewards (+ slight adaptation). To achieve this, the agent is executing a policy π.

Policies

📝 A (stationary) *policy* $\pi: S \rightarrow A$ is a function $\pi(s)$ that selects an action based on a state s. The *optimal policy* π^* maximizes the expected utility. An agent acts rationally when it maximizes the expected utility.

The quality of a policy is therefore measured by the *expected utility* of the possible environment histories generated by that policy. A policy represents the agent function explicitly and is therefore a description of a simple reflex agent, computed from the information used for a utility-based agent.

Utility, Reward and Discount

📝 Often, we want to enforce agent behavior where an agent values immediate rewards higher than distant future awards. Therefore, a *discount factor* $0 < \gamma \leq 1$ is introduced. The new definition of utility is as follows:

$$U([(s_1, a_1), (s_2, a_2), (s_3, a_3), \ldots]) = R(s_1, a_1) + \gamma R(s_2, a_2) + \gamma^2 R(s_3, a_3) + \cdots$$

The decreasing γ^x factor reduces the utility of future rewards while also avoiding infinite utility. When γ is close to 0, rewards in the distant future are viewed as insignificant. When γ is 1, discounted rewards are exactly equivalent to additive rewards, so additive rewards are a special case of discounted rewards.

💡 Notice that $U(s)$ and $R(s)$ are quite different quantities; $R(s)$ is the “short term” reward for being in s, whereas $U(s)$ is the “long term” total reward from s onward. The utility is defined as the sums of discounted rewards.

6.4.2 Formal Definition

📝 An MDP is a 5-tuple (S, A, P, R, γ) with:

- set S of states
- set A of actions
- $P(S_{t+1}|S_t, A_t)$ the probability distribution specifying the system dynamics, also called transition model,
- $R(S_t, A_t, S_{t+1})$ specifies the reward at time t
  - $R(S_t, A_t, S_{t+1})$ is the expected reward received when the agent is in state s, does action a and ends up in state s'
  - To get from the more general definition $R(S_t, A_t, S_{t+1})$ to a simpler one $R(s, a)$ we need to sum the reward over all possible future states s', weighted by the probability of ending up in s'.

$$R(s,a) = \sum_{s'} P(s'|s,a) * R(s,a,s')$$

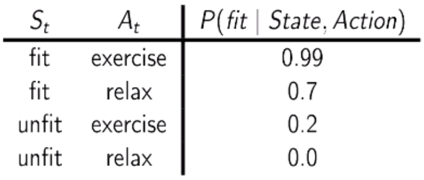
- γ is the discount factor

For a fully observable MDP with stationary dynamics (meaning that the transition model does not change over time) and rewards with infinite or indefinite horizon, there is always an optimal stationary policy.

6.4.3 Example

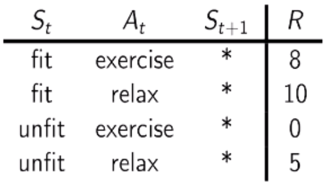
Each week Sam (agent) must decide whether to exercise or not:

- 2 states $S = \{fit, unfit\}$
- Dynamics $P(S_{t+1}|S_t, A_t) =$



S_t	A_t	$P(fit \mid State, Action)$
fit	exercise	0.99
fit	relax	0.7
unfit	exercise	0.2
unfit	relax	0.0

- 2 actions $A = \{exercise, relax\}$
- Reward $R(S_t, A_t, S_{t+1}) =$



S_t	A_t	S_{t+1}	R
fit	exercise	*	8
fit	relax	*	10
unfit	exercise	*	0
unfit	relax	*	5

❓ How many policies are there?

- Let s be the number of states, a be the number of actions, then there are a^s possible policies.

❓ What are they?

$States/Policies$	π_1	π_2	π_3	π_4
$unfit$	e	e	r	r
fit	e	r	e	r

6.4.4 Policy evaluation

We defined the utility of being in a state as *the expected sum of discounted rewards* from that point onwards. From this, it follows that there is a direct relationship between the utility of a state and the utility of its neighbor states: the utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.

Given a policy π:

- $V^{\pi}(s)$ is the expected discounted reward when following policy π in state s
- $Q^{\pi}(s, a)$ is the total expected discounted reward value of doing a in state s, then following π
- V^{π} and Q^{π} can be defined mutually recursively:

$$V^{\pi}(s) = Q^{\pi}(s, \pi(s))$$

$$Q^{\pi}(s, a) = \sum_{s'} P(s'|s, a) * \left(R(s, a, s') + \gamma * V^{\pi}(s')\right)$$

- $\sum_{s'}$ is every possible state s' that can be reached from state s with action a
- $P(s'|s, a)$ is the probability of ending up in state s' (from state s with action a).
- $R(s, a, s')$ is the immediate reward of ending up in state s'.
- $\gamma * V^{\pi}(s')$ is the value of being in state s' and then following policy π, weighted by the discount factor γ

Let π^*be the optimal policy. It always chooses the action a which maximizes the value:

$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) * \left(R(s, a, s') + \gamma * V^*(s')\right)$$

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s, a)$$

6.4.5 Example (1/2)

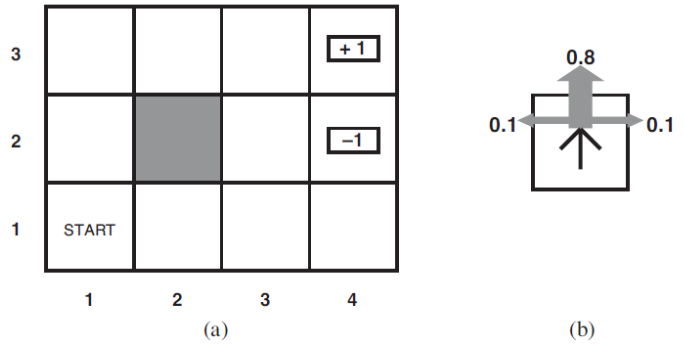


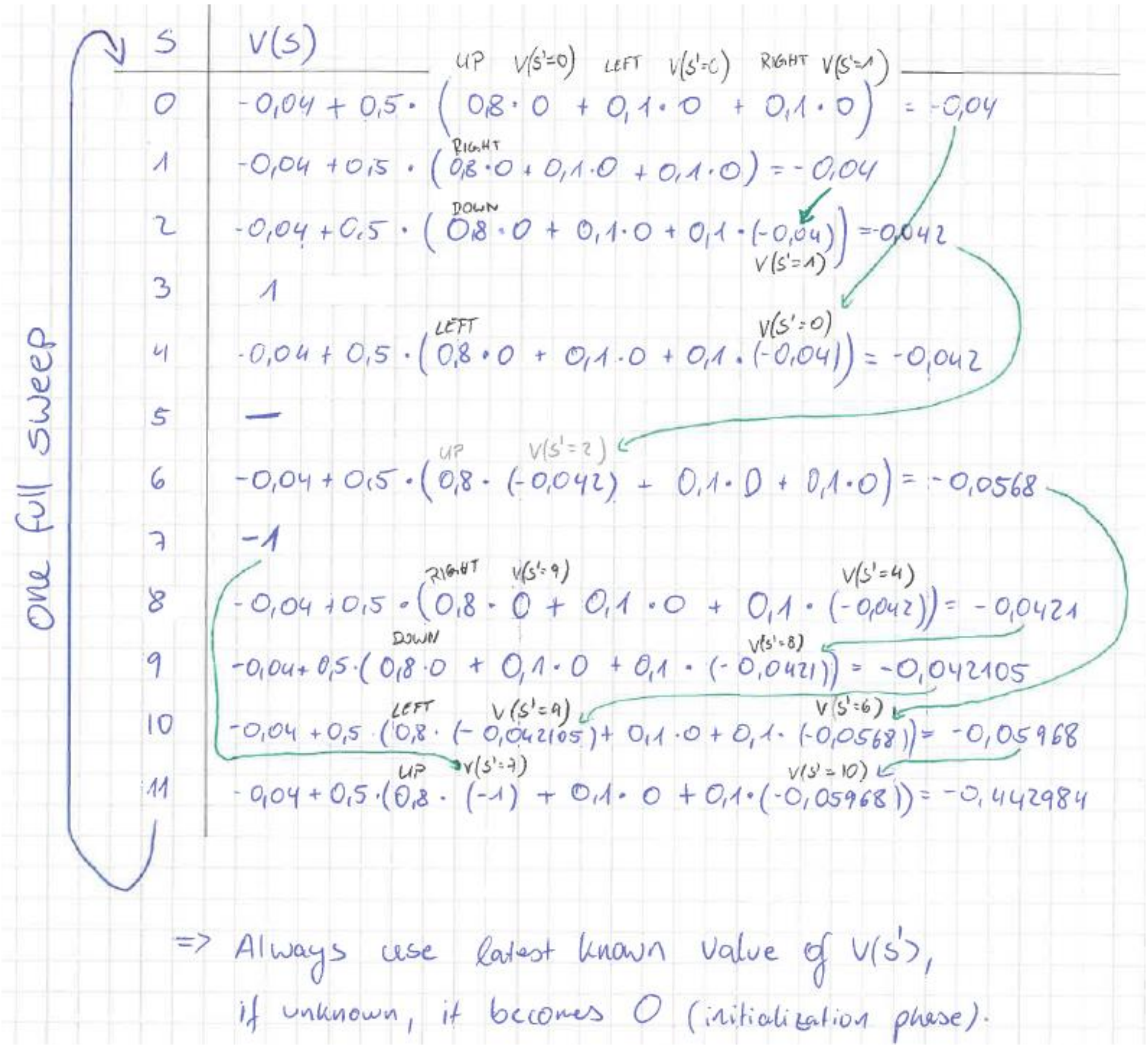
Figure 17.1 (a) A simple 4×3 environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the “intended” outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and –1, respectively, and all other states have a reward of –0.04.

Calculating the first sweep of policy iteration, given the following policy

s	0	1	2	4	6	8	9	10	11
$\pi(s_i)$	UP	RIGHT	DOWN	LEFT	UP	RIGHT	DOWN	LEFT	UP

and the formula to compute utilities:

$$v(s) = r(s) + \sum_{s} P\left(s' \middle| s, a = \pi(s)\right) * v(s')$$



One full sweep

S	V(S)
0	$-0{,}04 + 0{,}5 \cdot (\overset{\text{UP}}{0{,}8} \cdot \overset{V(s'=0)}{0} + \overset{\text{LEFT}}{0{,}1} \cdot \overset{V(s'=0)}{0} + \overset{\text{RIGHT}}{0{,}1} \cdot \overset{V(s'=1)}{0}) = -0{,}04$
1	$-0{,}04 + 0{,}5 \cdot (\overset{\text{RIGHT}}{0{,}8} \cdot 0 + 0{,}1 \cdot 0 + 0{,}1 \cdot 0) = -0{,}04$
2	$-0{,}04 + 0{,}5 \cdot (\overset{\text{DOWN}}{0{,}8} \cdot 0 + 0{,}1 \cdot 0 + 0{,}1 \cdot \underset{V(s'=1)}{(-0{,}04)}) = -0{,}042$
3	1
4	$-0{,}04 + 0{,}5 \cdot (\overset{\text{LEFT}}{0{,}8} \cdot 0 + 0{,}1 \cdot 0 + 0{,}1 \cdot \overset{V(s'=0)}{(-0{,}04)}) = -0{,}042$
5	—
6	$-0{,}04 + 0{,}5 \cdot (\overset{\text{UP}}{0{,}8} \cdot \overset{V(s'=2)}{(-0{,}042)} + 0{,}1 \cdot 0 + 0{,}1 \cdot 0) = -0{,}0568$
7	-1
8	$-0{,}04 + 0{,}5 \cdot (\overset{\text{RIGHT}}{0{,}8} \cdot \overset{V(s'=9)}{0} + 0{,}1 \cdot 0 + 0{,}1 \cdot \overset{V(s'=4)}{(-0{,}042)}) = -0{,}0421$
9	$-0{,}04 + 0{,}5 \cdot (\overset{\text{DOWN}}{0{,}8} \cdot 0 + 0{,}1 \cdot 0 + 0{,}1 \cdot \overset{V(s'=8)}{(-0{,}0421)}) = -0{,}042105$
10	$-0{,}04 + 0{,}5 \cdot (\overset{\text{LEFT}}{0{,}8} \cdot \overset{V(s'=9)}{(-0{,}042105)} + 0{,}1 \cdot 0 + 0{,}1 \cdot \overset{V(s'=6)}{(-0{,}0568)}) = -0{,}05968$
11	$-0{,}04 + 0{,}5 \cdot (\overset{\text{UP}}{0{,}8} \cdot \overset{V(s'=7)}{(-1)} + 0{,}1 \cdot 0 + 0{,}1 \cdot \overset{V(s'=10)}{(-0{,}05968)}) = -0{,}442984$

=> Always use latest known value of V(s'),
if unknown, it becomes 0 (initialization phase).

6.4.6 Solution Technique: Value Iteration

💡 Value iteration is an algorithm for *finding and calculating an optimal policy*. The basic idea is to calculate the utility of each state and then use the state utilities to select an optimal action in each state.

Idea: Let V_k and Q_k be k-step lookahead value and Q functions. Given an estimate of the k-step lookahead value function, determine the k+1 step lookahead value function.

Set V_0 arbitrarily. Compute $Q^{(i+1)}$ and $V^{(i+1)}$ from $V^{(i)}$.

If $R(s,a,s') = R(s,a)$:

$$V^{(i+1)}(s) = \underbrace{R\left(s,\pi^{(i)}(s)\right)}_{\substack{\text{immediate reward of}\\ \text{following the current}\\ \text{policy in state } s}} + \gamma \underbrace{\sum_{s'} \underbrace{P\left(s'|\pi^{(i)}(s),s\right)}_{\substack{\text{probability of}\\ \text{ending up in } s'}} * \underbrace{V^{(i)}(s')}_{\substack{\text{value of state } s' \text{ given by}\\ \text{the old value function}}}}_{\substack{\text{expectation of all possible}\\ \text{states we can end up in}}}$$

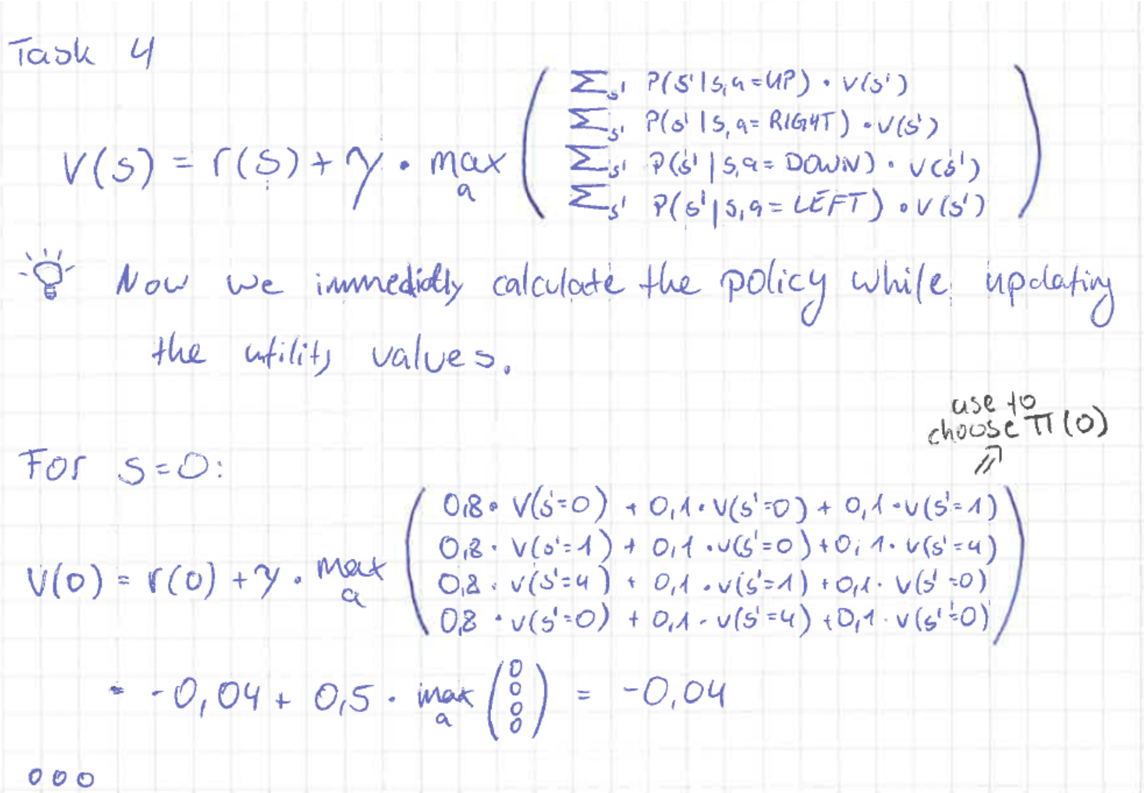
with $\pi^{(i)}$ being the optimal policy with respect to $V^{(i)}$. This term computes a better value function $V^{(i+1)}(s)$. Or alternatively **if rewards depend on s'**:

$$V^{(i+1)}(s) = \sum_{s'} P\left(s' \mid \pi^{(i)}(s), s\right) * \left(R\left(s,\pi^{(i)}(s),s'\right) + \gamma * V^{(i)}(s')\right)$$

This converges exponentially fast (in k) to the optimal value functions ($V^*$and Q^*).

The error reduces proportionally to $\frac{\gamma^k}{1-\gamma}$

6.4.7 Example: 2/2



Task 4

$$V(s) = r(s) + \gamma \cdot \max_a \begin{pmatrix} \Sigma_{s'} \; P(s' \mid s, a = UP) \cdot v(s') \\ \Sigma_{s'} \; P(s' \mid s, a = RIGHT) \cdot v(s') \\ \Sigma_{s'} \; P(s' \mid s, a = DOWN) \cdot v(s') \\ \Sigma_{s'} \; P(s' \mid s, a = LEFT) \cdot v(s') \end{pmatrix}$$

💡 Now we immedidtly calculate the policy while updating the utility values.

use to choose $\pi(0)$

For $s = 0$:

$$V(0) = r(0) + \gamma \cdot \max_a \begin{pmatrix} 0{,}8 \cdot V(s'=0) + 0{,}1 \cdot V(s'=0) + 0{,}1 \cdot v(s'=1) \\ 0{,}8 \cdot V(s'=1) + 0{,}1 \cdot v(s'=0) + 0{,}1 \cdot v(s'=4) \\ 0{,}8 \cdot v(s'=4) + 0{,}1 \cdot v(s'=1) + 0{,}1 \cdot V(s'=0) \\ 0{,}8 \cdot v(s'=0) + 0{,}1 \cdot v(s'=4) + 0{,}1 \cdot v(s'=0) \end{pmatrix}$$

$$= -0{,}04 + 0{,}5 \cdot \max_a \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = -0{,}04$$

000

💡 For $s = 1$ we then can use the updated $v(s\text{^}')$ values from the states calculated in $s = 0$

7 Reinforcement Learning

❓ In many cases we don’t know properties such as the state space S, the action set A, the transition model or the expected rewards to solve a problem by using an MDP. What do we do in case we’re thrown into an unknown world, and we have to learn these models from observations?

7.1 Batch learning

Assume that there is a sequence of transition samples $(s, a, r, s')_{1:N}$, so called *experiences*. We can use a sequence of these experiences to create an empirical model:

- $\hat{S} \coloneqq \{s_i\} \cup \{s_i'\}$ collect all possible states we observed as the *state set*
- $\hat{A} \coloneqq \{a_i\}$ collect all possible actions we observed as the *action set*
- $\hat{P}_{ss'}^{a} \coloneqq \frac{N_{ss'}^{a}}{N_s^a}$ Compute an estimate of the *transition model*. Out of all states s we’ve been in: in how many of these situations did we end up in s'.
- $\hat{R}_{ss'}^{a} \coloneqq \frac{1}{N_{ss'}^{a}} * \sum_{i=1}^{N} r_i[s_i = s, a_i = a, s_i' = s']$ Computes the average *reward*. Out of all states s we’ve been in, selected action a, and ended up in s'.

where

- $N_{ss'}^{a} \coloneqq \sum_{i=1}^{N} r_i[s_i = s, a_i = a, s_i' = s']$
- $N_s^a \coloneqq \sum_{s' \in \hat{S}} N_{ss'}^{a}$

Experiences $(s, a, r, s')_{1:N}$ can be generated by interacting with the environment. This can be done by following an arbitrary policy if this policy guarantees that, as $N \to \infty$, every action will be tried infinitely often in any state.

❗ Batch learning takes a lot of time/experiences to obtain a reliable estimate of the MDP via experiences. Storing these experiences takes a lot of time and explicitly storing the Q-function takes a lot of space.

7.2 Q-Learning

How can we learn an estimate of the Q-function without having to learn the full MDP first? Consider the Bellman optimality equation for the optimal Q^*-function:

$$Q^*(s,a) = \sum_{s'} P_{ss'}^{a} * \left(R_{ss'}^{a} + \gamma * \max_{a'} Q^*(s', a')\right)$$

Now consider a single experience $(s, a, r, \tilde{s}')$. Using this tuple, we can first build very coarse approximations of the transition model $P_{ss'}^{a} = p(s'|s,a)$ and the reward model $R_{ss'}^{a}$, by defining:

$$P_{ss'}^{a} \approx [s' = \tilde{s}'] = \begin{cases} 1: & s' = \tilde{s}' \\ 0: & else \end{cases}$$

The probability is 1 for the observed state and 0 for everything else.

$$R_{s\tilde{s}'}^{a} \approx r$$

The expected reward is the one reward we just observed.

Entering this into the equation above gives: $Q^*(s,a) = \sum_{s'} [s' = \tilde{s}'] * \left(r + \gamma * \max_{a'} Q^*(s', a')\right)$

Due to the assumption of $P^{a}_{ss'} \approx [s' = \tilde{s}'] = \begin{cases} 1: & s' = \tilde{s}' \\ 0: & else \end{cases}$ each part of the sum becomes zero, except for the one case where $s' = \tilde{s}'$ holds. Therefore:

$$Q^{*}(s,a) = r + \gamma * \max_{a'} Q^{*}(\tilde{s}', a')$$

This approximation based of just one experience is obviously not very good. It can be improved by using a weighted average of both values (this one and the following approximation):

$$Q_{i+1}(s,a) \leftarrow (1-\alpha) * Q_i(s,a) + \alpha \left(r + \gamma * \max_{a'} Q_i(s',a') \right)$$

7.2.1 Definition

Q-Learning

$$\overbrace{Q_{i+1}(s,a)}^{\substack{\text{updated Q-value for} \\ \text{state } s \text{ and action } a}} \leftarrow \underbrace{Q_i(s,a)}_{\substack{\text{a previous} \\ \text{estimate of the} \\ \text{Q-function}}} + \overset{\substack{\text{weighting factor} \\ \downarrow}}{\alpha} \left(\overbrace{\underbrace{r + \gamma * \max_{a'} Q_i(s',a')}_{\substack{\text{newly generated estimate of} \\ \text{the Q-function}}} - Q_i(s,a)}^{\substack{\text{difference of new and} \\ \text{old q-functions}}} \right)$$

This version emphasizes the idea that the new action value is the old value plus a move of step size α in the *direction of the difference between the old value and the target* created by the new experience. I.e., we move the action value gradually to the target, using the temporal difference.

The value α, where $0 \leq \alpha \leq 1$, is called the *learning rate.* If α is appropriately adjusted over time such that $\lim_{t \to \infty} \alpha = 0$, the Q-learning algorithm will converge to Q^{*}. Fortunately, to π^{*} will (usually) be discovered long before Q^{*} has been reached.

7.2.2 Action Selection

Q-learning will converge to the true optimal policy (and the true optimal value function), regardless of policy, as long as the policy will try every action infinitely often in any state.

However, in order to optimize reward, it seems advisable to start using actions with high value as soon as possible. This is known as the *exploration-exploitation dilemma*: if one starts to greedily use the currently optimal moves, action options leading to maybe even better results will never be reached.

One simple solution is to always reserve a probability of ϵ (e.g., $\epsilon = 0{,}01$) for experimentally trying a non-optimal action. This is called epsilon-greedy action selection:

$$\pi(s,a) = \begin{cases} 1 - \epsilon & if\ a\ is\ optimal \\ \dfrac{\epsilon}{n-1} & otherwise \end{cases}$$

Where $n > 1$ is the number of actions available in state s (if $n = 1$ there is obviously nothing to explore).

8 Game Theory

Game Theory studies decision making of multiple actors, it tells us:

- How an agent should decide
- How the decision situation should be designed to let actors make good decisions

8.1 Single Move Games

The simplest form of a game is where each actor makes exactly one move. A game is defined by players (decision makers), possible actions and a payoff function.

Example: Two Finger Morra

Players O and E simultaneously display one or two fingers. If the total amount of fingers f shown

- is *odd*, player O gets f dollar from player E,
- is *even*, player O gives f dollar to player E,

Analyzing with *normal form*:

	O: one finger	**O: two fingers**
E: one finger	E: +2, O: -2	E: -3, O: +3
E: two fingers	E: -3, O: +3	E: +4, O: -4

Strategies:

- *Pure*: always make the same choice, e.g., E: one finger
- *Mixed*: A probability distribution over choices, e.g. [0,5: one; 0.5: two]

Example: The prisoner’s dilemma

Two prisoners are accused of committing a crime. Based on their willingness to cooperate, they will do more or less time.

The corresponding *normal form*:

	A: testify	**A: refuse**
B: testify	A: -5; B: -5	A: -10; B: 0
B: refuse	A: 0; B: -10	A: -1; B: -1

From prisoner A’s perspective, testifying is the best option, because no matter what B does it is the better choice (“testify” *strongly dominates* “refuse”). The same holds for prisoner B!

💡 Both prisoners testifying is called *pareto optimal*.

8.2 Nash Equilibria

When each player has a dominant strategy, their combination is always a *Nash equilibrium*. In general, a combination of strategies is already a Nash equilibrium, if no player can gain by changing the strategy.

❗ Equilibria are not necessarily the best solution for all players. Both prisoners would be better off. If they would both refuse, still this is not the best rational strategy.

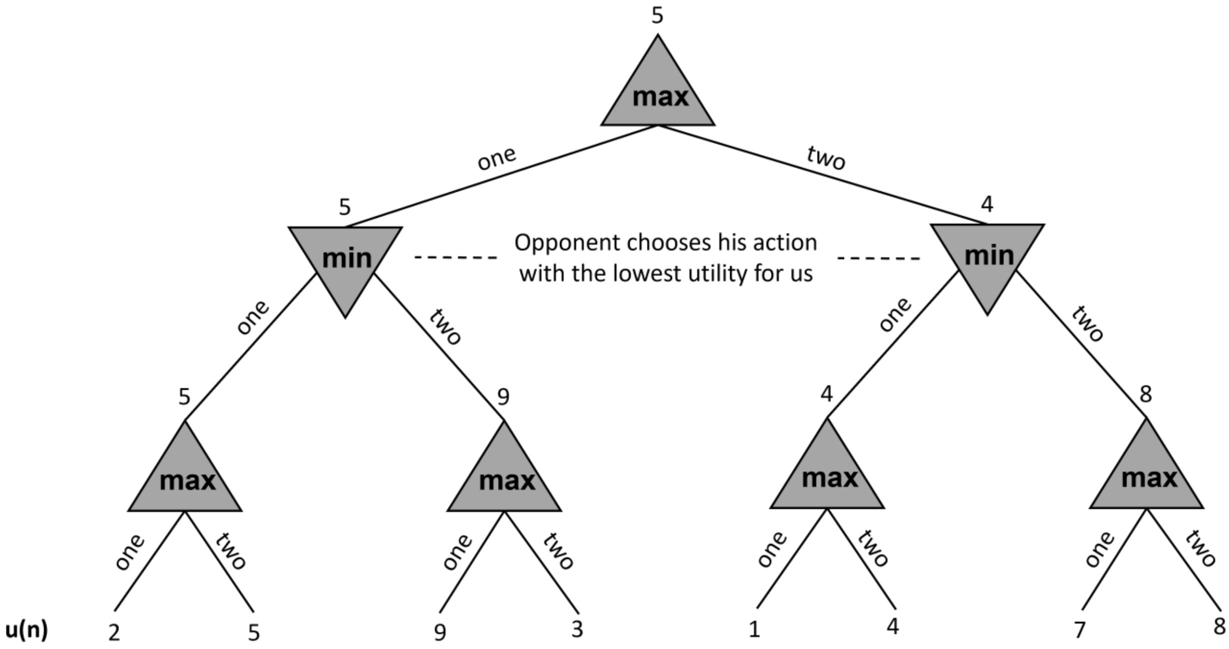
8.3 Zero Sum Games

📝 Zero Sum Games are a special kind of game where the sum of the rewards for the players is zero. That means if one player wins amount *x*, other players have to lose *x*. For example: Two Finger Morra. There is no pure strategy but its possible to determine an optimal mixed strategy.

❗ Assumption: players act rationally (players choose the move that maximizes utility/reward).

📝 *MinMax Strategy*: maximize reward given that the other player will try to minimize it.

8.3.1 Game Trees and Utility Propagation

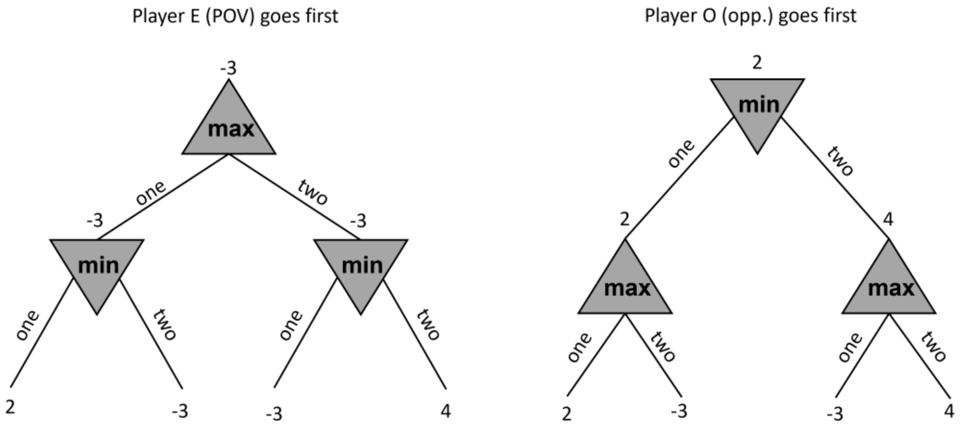


Game tree method to calculate possible rewards. Players can choose between move “one” and “two”.

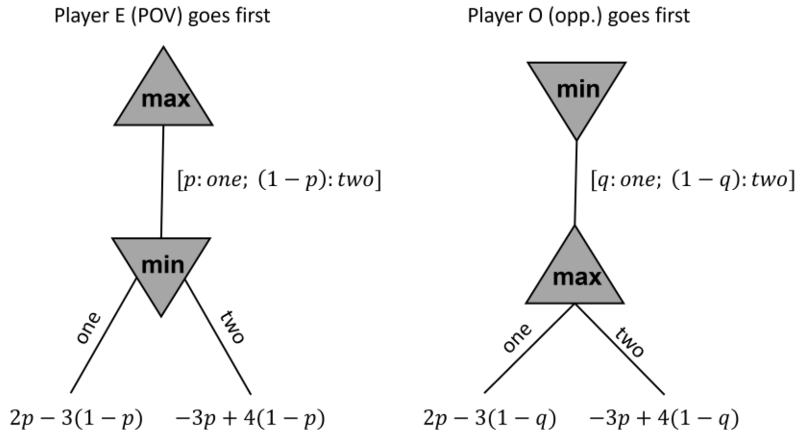
8.3.2 Two Finger Morra: Continued

Theoretical conversion into a turn-based game to use game trees:

This gives us upper and lower bounds for the utility of the optimal solution: $[-3, 2]$.



Now let’s look at the case where the first player has a mixed strategy:



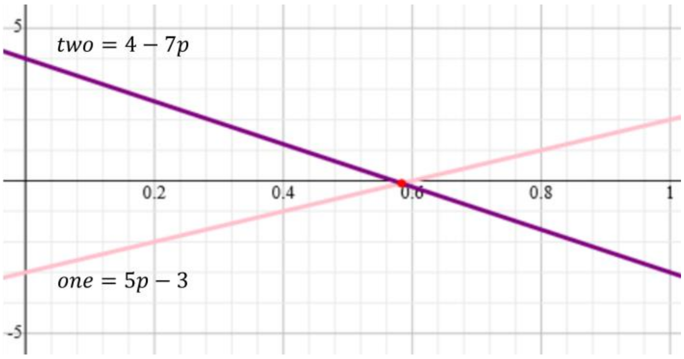
The optimal strategy can now be determined by solving a system of linear equations modelling the possible gains. The optimal strategy is the intersection of the two lines.

If player E chooses first the expected payoff for “one” is $5p - 3$ and for “two” $4 - 7p$.

$$5p - 3 = 4 - 7p \Rightarrow p = \frac{7}{12}$$

If player O chooses first the expected payoff for “one” is $5q - 3$ and for “two” $4 - 7q$

$$5q - 3 = 4 - 7q \Rightarrow q = \frac{7}{12}$$



This means that the best mixed strategy for both players is $[\frac{7}{12}: one, \frac{5}{12}: two]$

8.4 Repeated Games

Simplest form of multi-move games are repeated games. E.g., the prisoner’s dilemma game with a 99% chance of playing another round with ongoing and accumulating rewards/penalties. This allows for different strategies that take the (past) behavior of the other player into account:

8.4.1 Perpetual punishment

Both players choose the action that possibly gets them to the best result, i.e., “refuse” in the prisoner’s dilemma, until the other chooses “testify” once. From then on, only choose “testify” as well.

This strategy rewards cooperation: the total payoff for both players when sticking to “refuse” is:

$$\sum_{t=0}^{\infty} 0,99^t * (-1) = -100$$

It punishes non-cooperative behavior: the total payoff for a player choosing “testify” is:

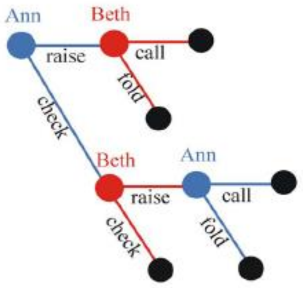
$$0 + \sum_{t=1}^{\infty} 0,99^t * (-5) = -495$$

8.4.2 Tit-for-Tat

Start with “refuse” and always mimic the choice of the other player in the next round. Very robust strategy with a less strong but still working punishing effect.

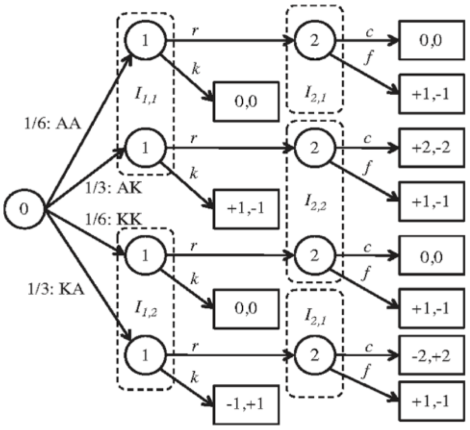
8.5 Sequential Games

In general, games consist of sequences of turns that don’t need to be the same. Such games are best represented in terms of a game tree.



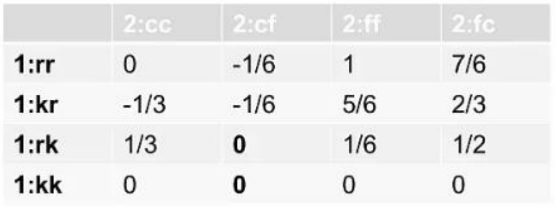
Example: Simplified Poker

Consists of a deck of four cards containing two aces and two kings. One card is dealt to each player. Then, the player starting this turn has options “raise” and “check” while the other player then has options to “fold” or “call”.



Game Tree for Simplified Poker

Sequential games can be converted to normal form games:



	2:cc	2:cf	2:ff	2:fc
1:rr	0	-1/6	1	7/6
1:kr	-1/3	-1/6	5/6	2/3
1:rk	1/3	**0**	1/6	1/2
1:kk	0	**0**	0	0