- What is intelligence?

- What is intelligence?
  - Intellectual capability of humans

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude?

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent?

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
  - Intelligence may refer to different abilities.

- What is intelligence?
    - Intellectual capability of humans
    - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
    - Intelligence may refer to different abilities.
- What is Artificial intelligence?

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
  - Intelligence may refer to different abilities.
- What is Artificial intelligence?
  - Make a program capable of something:

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
  - Intelligence may refer to different abilities.
- What is Artificial intelligence?
  - Make a program capable of something:
  - It could be correct logical reasoning.

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
  - Intelligence may refer to different abilities.
- What is Artificial intelligence?
  - Make a program capable of something:
  - It could be correct logical reasoning.
  - It could be solving a puzzle in minimum number of steps.

- What is intelligence?
  - Intellectual capability of humans
  - Is it just the aptitude? Is Lionel Messi intelligent? Is A. R. Rehaman intelligent?
  - Intelligence may refer to different abilities.
- What is Artificial intelligence?
  - Make a program capable of something:
  - It could be correct logical reasoning.
  - It could be solving a puzzle in minimum number of steps.
  - It could be probabilistic inference.

Artificial Intelligence

- Includes all the topics in data science.

Artificial Intelligence

- Includes all the topics in data science.
- Scope of this course: Learn algorithms and techniques that will allow an agent (program) take optimal (intelligent) action in various environments.

Birth of AI: Initial conjecture

- *Every aspect of learning or any other feature of (human) intelligence can in principle be so precisely defined that a machine can be made to simulate it. (1956)*

Birth of AI: Initial conjecture

- *Every aspect of learning or any other feature of (human) intelligence can in principle be so precisely defined that a machine can be made to simulate it. (1956)*
- Most problems that are of interest are NP-hard

In this course

- We will define a problem.

In this course

- We will define a problem.
- We will represent the problem. (Usually, as a graph or a tree.)

In this course

- We will define a problem.
- We will represent the problem. (Usually, as a graph or a tree.)
- The problem turns out to be NP-hard.

In this course

- We will define a problem.
- We will represent the problem. (Usually, as a graph or a tree.)
- The problem turns out to be NP-hard.
- What are the general techniques (**heuristics**) we can use so that the problem can be solved more easily in practice?

In this course

- We will define a problem.
- We will represent the problem. (Usually, as a graph or a tree.)
- The problem turns out to be NP-hard.
- What are the general techniques (**heuristics**) we can use so that the problem can be solved more easily in practice?
- Questions?

Optimization in discrete search space (Chapter 4)

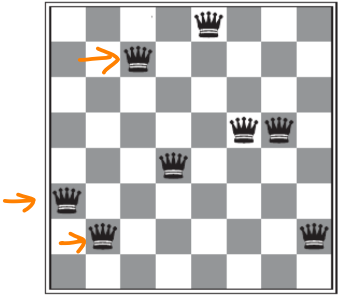
Optimization in discrete search space (Chapter 4)

- Objective function

Optimization in discrete search space (Chapter 4)

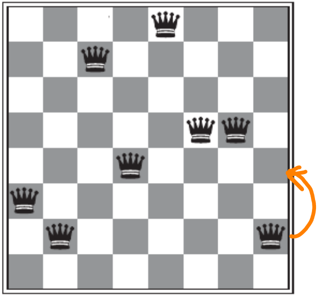
- Objective function
- Optimization over a discrete state space

Objective function: Cost vs. Fitness



3 2 7 4 8 5 5 2

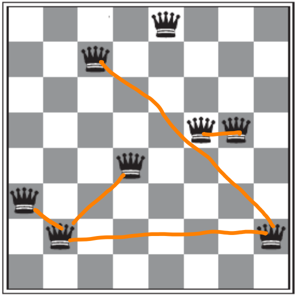
Objective function: Cost vs. Fitness



3 2 7 4 8 5 5 2 → (3 27 . . . 5,5,4)

- State and State space

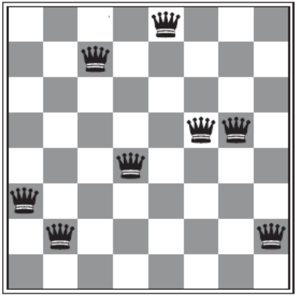
Objective function: Cost vs. Fitness



3 2 7 4 8 5 5 2

- State and State space
- Cost function $h = 5$

Objective function: Cost vs. Fitness

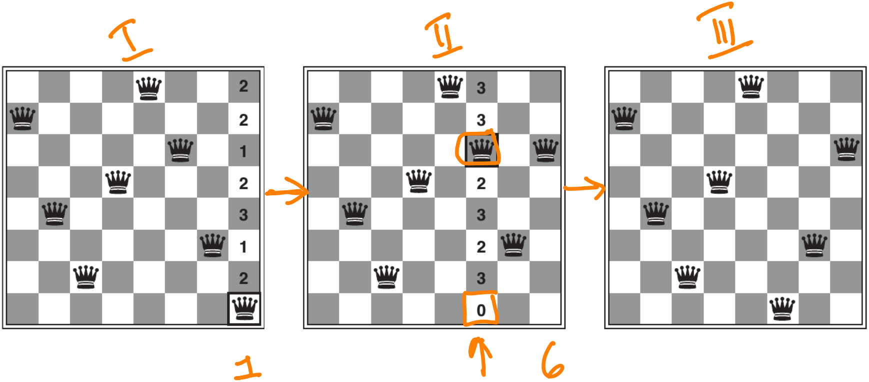


3 2 7 4 8 5 5 2

- State and State space
- Cost function $h = 5$
- Fitness function $= \binom{8}{2} - 5 = 23$

28

8 Queens Problem: 3 states



8 Queens Problem

- Total possible number of states? $8 \times 8 \times \ldots 8 = 8^8$

8 Queens Problem

- Total possible number of states?
- How many neighbours does each state have?

$7 \times 8 = 56$

8 Queens Problem

- Total possible number of states?
- How many neighbours does each state have?
- Objective function?

Four search algorithms

- Hill climbing

Four search algorithms

- Hill climbing
- Simulated annealing

Four search algorithms

- Hill climbing
- Simulated annealing
- Local beam search

Four search algorithms

- Hill climbing
- Simulated annealing
- Local beam search
- Genetic algorithm

Steepest ascent Hill climbing algorithm

function HILL-CLIMBING(*problem*)

current ← MAKE-NODE(*problem*.INITIAL-STATE)
loop do
  neighbor ← a highest-valued successor of *current*
  if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
  current ← *neighbor*

Steepest ascent Hill climbing algorithm

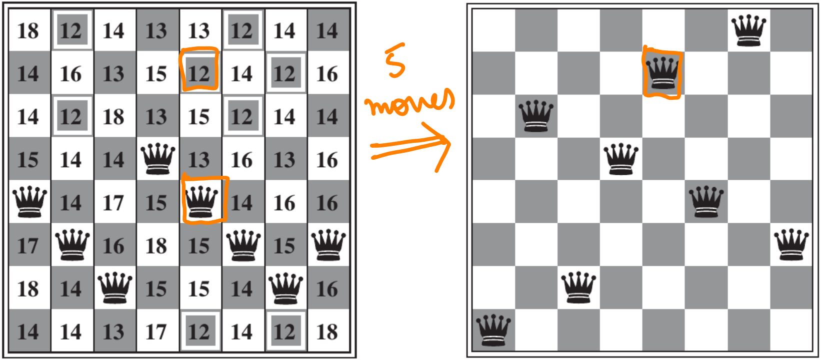
$8 \times 7 = 56$

function HILL-CLIMBING(*problem*)

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of *current*
    if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
    current ← *neighbor*

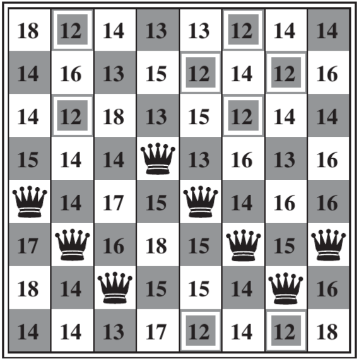
- Will this always work?

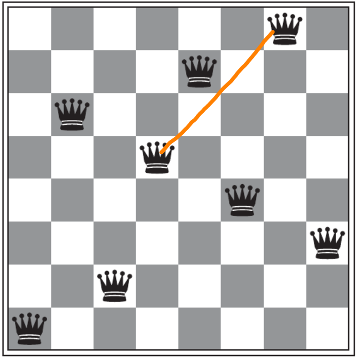
8-queens state



- 17 pairs of queens are in attacking position for the state on the left.

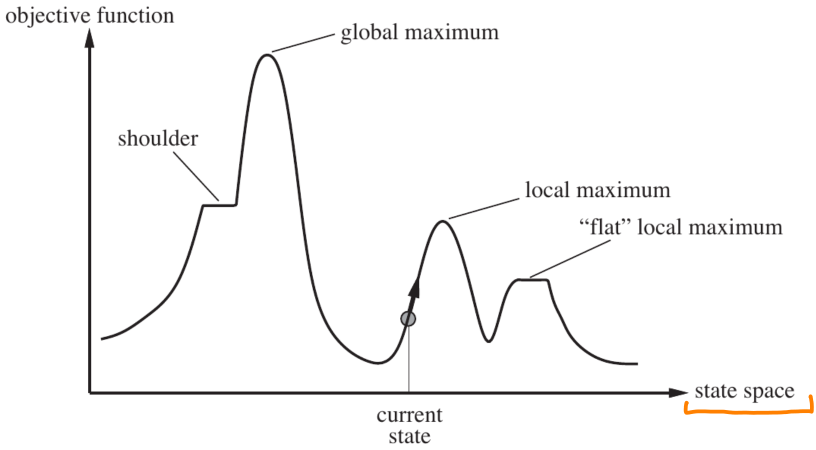
8-queens state





- 17 pairs of queens are in attacking position for the state on the left.
- After five steepest ascent steps, we reach a local maximum.

Landscape of the state-space



Success rate of steepest ascent hill climbing : 14%

Success rate of steepest ascent hill climbing : 14%
Possible ways to improve success:

Success rate of steepest ascent hill climbing : 14%

Possible ways to improve success:

- Sideways move

n-queen

n^n

Success rate of steepest ascent hill climbing : 14%

Possible ways to improve success:

- Sideways move
- N-consecutive sideways move

Success rate of steepest ascent hill climbing : 14%

Possible ways to improve success:

- Sideways move
- N-consecutive sideways move
  - For N=100, success rate: 94%

8-queens

Success rate of steepest ascent hill climbing : 14%

Possible ways to improve success:

- Sideways move
- N-consecutive sideways move
  - For N=100, success rate: 94%
- Stochastic hill climbing

Success rate of steepest ascent hill climbing : 14%

Possible ways to improve success:

- Sideways move
- N-consecutive sideways move
  - For N=100, success rate: 94%
- Stochastic hill climbing
- Random-restart hill climbing

Question

- Suppose, steepest-ascent hill climbing succeeds in reaching the goal state with probability p. What is the expected number of starts required before the random-restart hill climbing will succeed?

Question

- Suppose, steepest-ascent hill climbing succeeds in reaching the goal state with probability p. What is the expected number of starts required before the random-restart hill climbing will succeed?
- Suppose, we have a coin that gives a head with probability p. Suppose we repeatedly toss the coin. What is the expected number of coin tosses before we get a heads?

Question

- Suppose, steepest-ascent hill climbing succeeds in reaching the goal state with probability p. What is the expected number of starts required before the random-restart hill climbing will succeed?
- Suppose, we have a coin that gives a head with probability p. Suppose we repeatedly toss the coin. What is the expected number of coin tosses before we get a heads?
- Random-restart hill climbing

$$E(x) = p \times 1 + (1-p)(E(x)+1)$$

Question

- Suppose, steepest-ascent hill climbing succeeds in reaching the goal state with probability p. What is the expected number of starts required before the random-restart hill climbing will succeed?
- Suppose, we have a coin that gives a head with probability p. Suppose we repeatedly toss the coin. What is the expected number of coin tosses before we get a heads?
- Random-restart hill climbing
  - $p \approx .14$, Number of restarts $= \dfrac{1}{.14}$

Question

- Suppose, steepest-ascent hill climbing succeeds in reaching the goal state with probability p. What is the expected number of starts required before the random-restart hill climbing will succeed?
- Suppose, we have a coin that gives a head with probability p. Suppose we repeatedly toss the coin. What is the expected number of coin tosses before we get a heads?
- Random-restart hill climbing
  - $p \approx .14$, Number of restarts $= \dfrac{1}{.14} \approx 7$

Hill-climbing

- When will random-restart hill-climbing succeed in finding a good solution?

Simulated Annealing

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  current ← problem.INITIAL
  for t = 1 to ∞ do
    T ← schedule(t)
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE(current) – VALUE(next)
    if ΔE > 0 then current ← next
    else current ← next only with probability e^(−ΔE/T)
```

Some applications of Local search

- VLSI layout problem
  - optimize area (yield), power dissipation, etc.

Some applications of Local search

- VLSI layout problem
  - optimize area (yield), power dissipation, etc.
- Factory layout problem
  - Minimize total transportation of materials

Beam search

- **Local beam search**

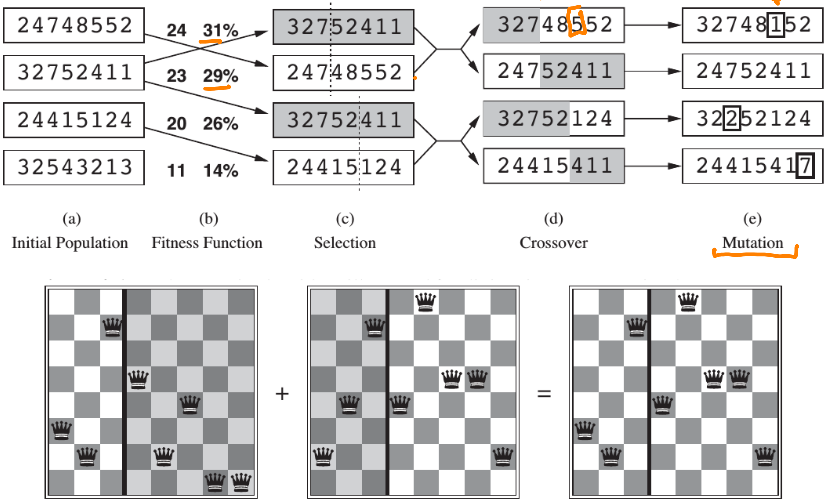
K
↑

Beam search

- **Local beam search**
- **Stochastic beam search**

24748552	24	31%
32752411	23	29%
24415124	20	26%
32543213	11	14%

Genetic Algorithm



Genetic Algorithm

```
function GENETIC-ALGORITHM(population, fitness) returns an individual
  repeat
      weights ← WEIGHTED-BY(population, fitness)
      population2 ← empty list
      for i = 1 to SIZE(population) do
          parent1, parent2 ← WEIGHTED-RANDOM-CHOICES(population, weights, 2)
          child ← REPRODUCE(parent1, parent2)
          if (small random probability) then child ← MUTATE(child)
          add child to population2
      population ← population2
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to fitness

function REPRODUCE(parent1, parent2) returns an individual
  n ← LENGTH(parent1)
  c ← random number from 1 to n
  return APPEND(SUBSTRING(parent1, 1, c), SUBSTRING(parent2, c + 1, n))
```

Genetic Algorithm

There are several things that we can vary:

- Size of the population

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ ← $\rho = 2$

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents

n

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents
- Selecting a *crossover point*

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents
- Selecting a *crossover point*
- Mutation rate

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents
- Selecting a *crossover point*
- Mutation rate
- Make up of the next generation

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents
- Selecting a *crossover point*
- Mutation rate
- Make up of the next generation
    - Elitism

Genetic Algorithm

There are several things that we can vary:

- Size of the population
- Representation of each individual
- Mixing number, ρ
- Selection process : find ρ parents
- Selecting a *crossover point*
- Mutation rate
- Make up of the next generation
  - Elitism
  - Culling

(K + n)

Genetic Algorithm

- GA : schema and instances

2 4 6 * * * * *

↑ ↑ ↑

Genetic Algorithm

→ 2 4 6 * * * *

→ 1 2 3 * * * *

- GA : schema and instances
- If average fitness of the instances of a schema is above mean, then the number of instances of the schema in the population will grow over time.

Genetic Algorithm

- GA : schema and instances
- If average fitness of the instances of a schema is above mean, then the number of instances of the schema in the population will grow over time.
- Succesful use of GA requires careful engineering of representation.

Reinforcement Learning

B3: Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning – An Introduction, Second Edition*

Reinforcement Learning

B3: Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning – An Introduction, Second Edition*

Plan: Chapters 1, 2, 3 and 6

Reinforcement Learning

B3: Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning – An Introduction, Second Edition*

Plan: Chapters 1, 2, 3 and 6

Reminder : Python Tutorial on 05/09/21 (Sunday) at 5:30 PM

Introduction: Chapter 1 of **B3**

- What is Reinforcement Learning?

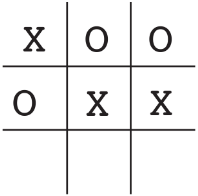
Introduction: Chapter 1 of **B3**

- What is Reinforcement Learning?
- Goal-directed learning through interaction with environment

Introduction: Chapter 1 of **B3**

- What is Reinforcement Learning?
- Goal-directed learning through interaction with environment
- Delayed reward; Trial-and-error search

- What is Reinforcement Learning?
- Goal-directed learning through interaction with environment
- Delayed reward; Trial-and-error search
- How to map states to actions such that the overall reward is maximized?



Comparision with other ML paradigms

- Supervised learning

Comparision with other ML paradigms

- Supervised learning
- Unsupervised learning

Features of Reinforcement Learning

- Trade-off between exploration and exploitation

Features of Reinforcement Learning

- Trade-off between exploration and exploitation
- Goal-seeking agent that interacts with an environment

Features of Reinforcement Learning

- Trade-off between exploration and exploitation
- Goal-seeking agent that interacts with an environment
- More similar to the learning that humans and other animals do

Examples

1. Trash-picking Robot

Examples

1. Trash-picking Robot

2. Person preparing breakfast

Examples

1. Trash-picking Robot
2. Person preparing breakfast

- There is interaction between an active decision-making agent and its environment

Elements of Reinforcement Learning

1. Policy :

Elements of Reinforcement Learning

1. Policy : Mapping from state to action

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal :

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal : Mapping from state and action to some number

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal : Mapping from state and action to some number
3. Value function :

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal : Mapping from state and action to some number
3. Value function : Mapping from a state to a number

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal : Mapping from state and action to some number
3. Value function : Mapping from a state to a number
    - Imp. component : A method for efficiently estimating the value function

Elements of Reinforcement Learning

1. Policy : Mapping from state to action
2. Reward signal : Mapping from state and action to some number
3. Value function : Mapping from a state to a number
   - Imp. component : A method for efficiently estimating the value function
4. (Optional) Model of the environment : additional information about the environment

   e.g. Mapping from state and action to state.
   Models are useful in planning.

Extended example : Tic-Tac-Toe

X	O	O
O	X	X
		X

Extended example : Tic-Tac-Toe

X	O	O
O	X	X
		X

- Assumption: We are playing against an imperfect player

Extended example : Tic-Tac-Toe

X	O	O
O	X	X
		X

- Assumption: We are playing against an imperfect player
- Goal: Construct a player that will discover its oponents' imperfections and learn to maximize its chances of winning.

Solving by estimating the value function

X	O	O
O	X	X
		X

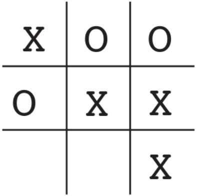
- How many states do we have?

Solving by estimating the value function

X	O	O
O	X	X
		X

- How many states do we have? 3^9

Solving by estimating the value function



- How many states do we have? 3^9
- Many states are infeasible.

Solving by estimating the value function

X	O	O
O	X	X
		X

- How many states do we have? 3^9
- Many states are infeasible.
- Many states are redundant.

Solving by estimating the value function

X	O	O
O	X	X
		X

- How many states do we have? 3^9
- Many states are infeasible.
- Many states are redundant.
- We need to consider only 765 unique game states.

Solving by estimating the value function

- Table contains a value corresponding to all the unique game states.

Solving by estimating the value function

- Table contains a value corresponding to all the unique game states.
- Value corresponds to probability of winning from a given state.

Solving by estimating the value function

- Table contains a value corresponding to all the unique game states.
- Value corresponds to probability of winning from a given state.
- Initially, the values are 0, 1 or 0.5 .

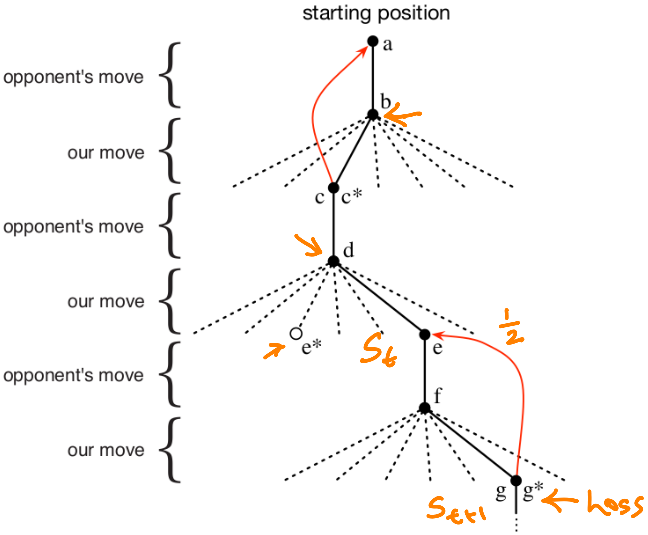
Solving by estimating the value function

- Table contains a value corresponding to all the unique game states.
- Value corresponds to probability of winning from a given state.
- Initially, the values are 0, 1 or 0.5 .
- We play many games against opponent.

Solving by estimating the value function

- Table contains a value corresponding to all the unique game states.
- Value corresponds to probability of winning from a given state.
- Initially, the values are 0, 1 or 0.5 .
- We play many games against opponent.
- Each move is either *greedy* or *exploratory*.

Solving by estimating the value function



Solving by estimating the value function

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

0 - 1/2

- α is a small positive fraction (step-size parameter); influences the learning rate

Solving by estimating the value function

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

- α is a small positive fraction (step-size parameter); influences the learning rate
- For convergence, step-size parameter is reduced over time.

Solving by estimating the value function

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

- α is a small positive fraction (step-size parameter); influences the learning rate
- For convergence, step-size parameter is reduced over time.
- Finds an optimal strategy against a particular (imperfect) opponent.

Solving by estimating the value function

$$V(S_t) \leftarrow V(S_t) + \alpha\Big[V(S_{t+1}) - V(S_t)\Big]$$

- α is a small positive fraction (step-size parameter); influences the learning rate
- For convergence, step-size parameter is reduced over time.
- Finds an optimal strategy against a particular (imperfect) opponent.
- We update only those states from where we chose a *greedy* move. Why?

- Instructive feedback vs. Evaluative feedback

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting
- K-armed Bandit problem

Ch. 2: Multi-armed Bandits

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting
- K-armed Bandit problem
  - K different actions

Ch. 2: Multi-armed Bandits

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting
- K-armed Bandit problem
  - K different actions
  - reward drawn from a probability distribution

Ch. 2: Multi-armed Bandits

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting
- K-armed Bandit problem
  - K different actions
  - reward drawn from a probability distribution
  - Goal: maximize expected total reward over 1000 time steps

- Instructive feedback vs. Evaluative feedback
- Evaluative feedback in *nonassociative* setting
- K-armed Bandit problem
  - K different actions
  - reward drawn from a probability distribution
  - Goal: maximize expected total reward over 1000 time steps
- One-armed Bandit / Slot machine:

K-armed Bandit Problem

- This problem has only one state.

K-armed Bandit Problem

- This problem has only one state.
- Expected reward (value) of each action:
  $q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$

K-armed Bandit Problem

- This problem has only one state.
- Expected reward (value) of each action:
  $q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$
- Estimated value of each action : $Q_t(a)$

K-armed Bandit Problem

- This problem has only one state.
- Expected reward (value) of each action:
  $q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$
- Estimated value of each action : $Q_t(a)$
  (Similar to value of each state $V(S_t)$)

K-armed Bandit Problem

- This problem has only one state.
- Expected reward (value) of each action:
  $q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$
- Estimated value of each action : $Q_t(a)$
  (Similar to value of each state $V(S_t)$)
- Goal : Find a good estimate, $Q_t(a)$, for the actual value $q_*(a)$.

K-armed Bandit Problem

- This problem has only one state.
- Expected reward (value) of each action:
  $q_*(a) \doteq \mathbb{E}[R_t | A_t = a]$
- Estimated value of each action : $Q_t(a)$
  (Similar to value of each state $V(S_t)$)
- Goal : Find a good estimate, $Q_t(a)$, for the actual value $q_*(a)$.
- Greedy moves and Exploratory moves.

Sample-average method for value estimation

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Sample-average method for value estimation

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Default value (0) if the denominator is 0

Sample-average method for value estimation

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Default value (0) if the denominator is 0
- Greedy action selection :

$$A_t \doteq \underset{a}{\text{argmax}}\, Q_t(a)$$

Sample-average method for value estimation

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Default value (0) if the denominator is 0
- Greedy action selection :

  $A_t \doteq \underset{a}{\operatorname{argmax}}\, Q_t(a)$
- ϵ-greedy action selection

ϵ $(1-\epsilon)$

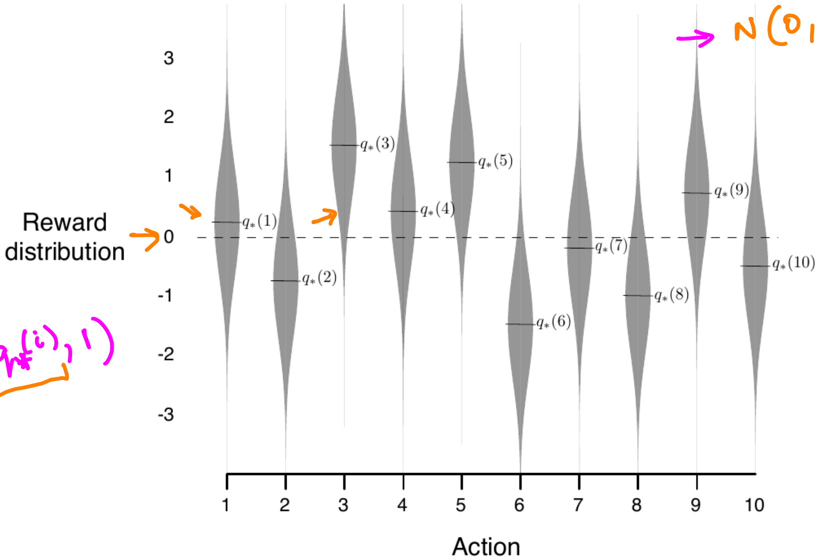
Sample-average method for value estimation

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

- Default value (0) if the denominator is 0
- Greedy action selection :

  $A_t \doteq \underset{a}{\operatorname{argmax}}\, Q_t(a)$
- ϵ-greedy action selection
- Assess the effectiveness of greedy and ϵ-greedy action-value methods : 10-armed testbed

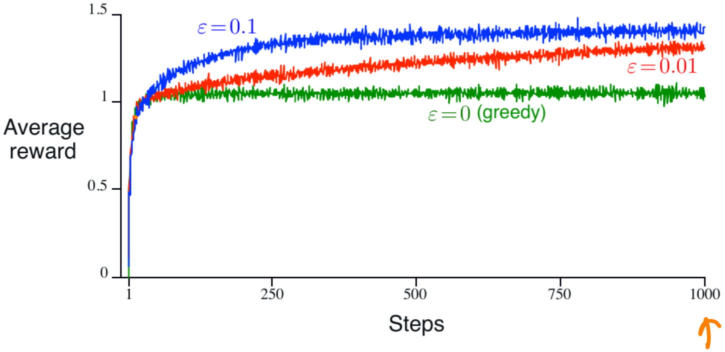
Random 10-armed bandit problem



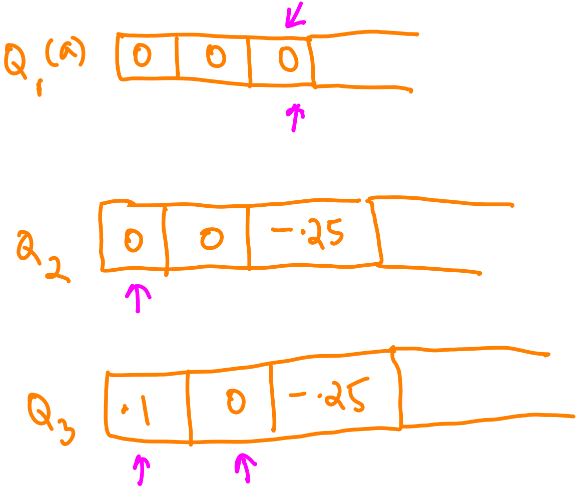
The 10-armed testbed

- A set of 2000 randomly generated 10-armed bandit problem.

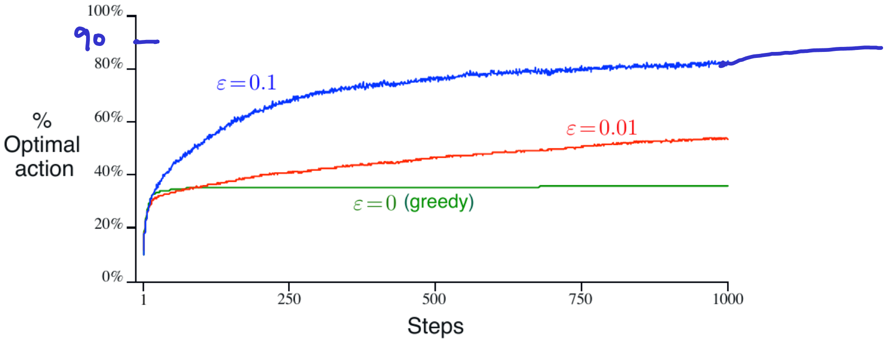
- A set of 2000 randomly generated 10-armed bandit problem.
- Action-value estimates were found using sample-average method



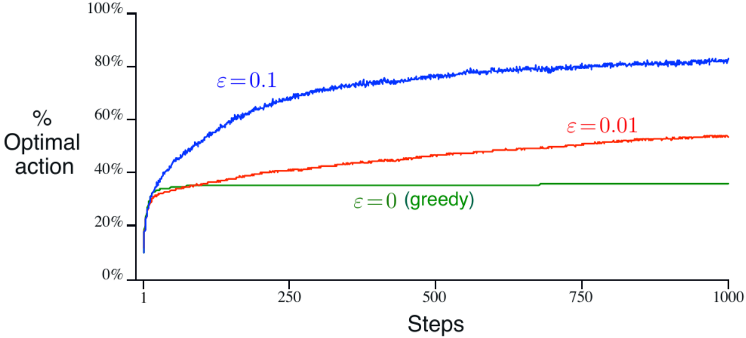
The 10-armed testbed



The 10-armed testbed

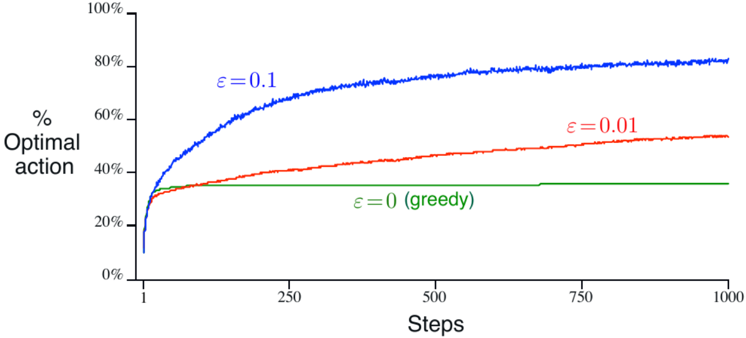


The 10-armed testbed



- Is it a good strategy to reduce the value of ϵ over time?

The 10-armed testbed



- Is it a good strategy to reduce the value of ϵ over time?
- If the reward probability distribution is nonstationary, it is better to keep exploring non-greedy actions.

Incremental Implementation

- Estimating action value : $Q_n \doteq \frac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$

Incremental Implementation

- Estimating action value : $Q_n \doteq \dfrac{R_1 + R_2 + \cdots + R_{n-1}}{n-1}$
- How to estimate the action values without storing all rewards?

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\Big(R_n + (n-1)Q_n\Big) \\
&= \frac{1}{n}\Big(R_n + nQ_n - Q_n\Big) \\
&= Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]
\end{aligned}
$$

Incremental Implementation

- $Q_{n+1} \doteq Q_n + \frac{1}{n}\left[R_n - Q_n\right]$

Incremental Implementation

- $Q_{n+1} \doteq Q_n + \frac{1}{n}[R_n - Q_n]$ (For a particular action)

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Loop forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$

Tracking a Nonstationary Problem

- Give more weight to recent rewards

$q_*(a)$

$\rightarrow N(q_*(a), 1)$

Tracking a Nonstationary Problem

- Give more weight to recent rewards
- $Q_{n+1} \doteq Q_n + \frac{1}{n}[R_n - Q_n]$

Tracking a Nonstationary Problem

- Give more weight to recent rewards
- $Q_{n+1} \doteq Q_n + \frac{1}{n}[R_n - Q_n]$
- $Q_{n+1} \doteq Q_n + \alpha [R_n - Q_n], \qquad \alpha \in (0, 1]$

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \quad \text{— (1)} \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}$$

$1 - (1-\alpha)^n$

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$
- Convergence conditions (stochastic approximation theory) :

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$
- Convergence conditions (stochastic approximation theory) :

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

- Conditions are satisfied for $\alpha_n(a) = \frac{1}{n}$

$q_*(a)$

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots = \infty$$

$$1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots = \frac{\pi^2}{6}$$

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$
- Convergence conditions (stochastic approximation theory) :

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

- Conditions are satisfied for $\alpha_n(a) = \frac{1}{n}$
- Conditions not satisfied for $\alpha_n(a) = \alpha$

$\frac{1}{2}$

$\frac{1}{2} + \frac{1}{2} + \cdots = \infty \qquad \frac{1}{4} + \frac{1}{4} + \cdots = \infty$

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$
- Convergence conditions (stochastic approximation theory) :

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

- Conditions are satisfied for $\alpha_n(a) = \frac{1}{n}$
- Conditions not satisfied for $\alpha_n(a) = \alpha$
- When $\alpha_n(a) = \alpha$, estimates don't converge but keep varying depending on the recent rewards.

Conditions required to assure convergence

- Step size parameter for an action : $\alpha_n(a)$
- Convergence conditions (stochastic approximation theory) :

$$\sum_{n=1}^{\infty} \alpha_n(a) = \infty \qquad \text{and} \qquad \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty$$

- Conditions are satisfied for $\alpha_n(a) = \frac{1}{n}$
- Conditions not satisfied for $\alpha_n(a) = \alpha$
- When $\alpha_n(a) = \alpha$, estimates don't converge but keep varying depending on the recent rewards.
  (A desirable property for nonstationary distribution.)

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.
  $Q_{n+1}(a) \doteq Q_n(a) + \frac{1}{n}[R_n - Q_n(a)]$

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.

$$Q_{n+1}(a) \doteq Q_n(a) + \frac{1}{n}[R_n - Q_n(a)]$$
$$Q_2(a) \doteq Q_1(a) + \frac{1}{1}[R_1 - Q_1(a)]$$

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.

  $Q_{n+1}(a) \doteq Q_n(a) + \frac{1}{n}[R_n - Q_n(a)]$

  $Q_2(a) \doteq Q_1(a) + \frac{1}{1}[R_1 - Q_1(a)] \ \doteq R_1$

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.

  $Q_{n+1}(a) \doteq Q_n(a) + \frac{1}{n}[R_n - Q_n(a)]$

  $Q_2(a) \doteq Q_1(a) + \frac{1}{1}[R_1 - Q_1(a)] \doteq R_1$
- However, when $\alpha_n(a)$ is a constant, the choice of $Q_1(a)$ matters.

Optimistic Initial values

- For sample-average methods (i.e. $\alpha_n(a) = \frac{1}{n}$), initial bias disappears.

  $Q_{n+1}(a) \doteq Q_n(a) + \frac{1}{n}[R_n - Q_n(a)]$

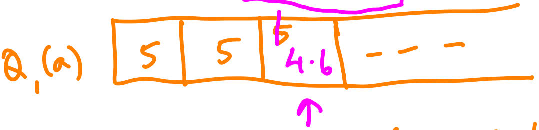
  $Q_2(a) \doteq Q_1(a) + \frac{1}{1}[R_1 - Q_1(a)] \doteq R_1$
- However, when $\alpha_n(a)$ is a constant, the choice of $Q_1(a)$ matters.
- Initial action values can be used to encourage exploration.

Optimistic Initial values

- Let $Q_1(a) = 5$ and $\alpha_n(a)$ be .1 for 10-armed testbed.

Optimistic Initial values

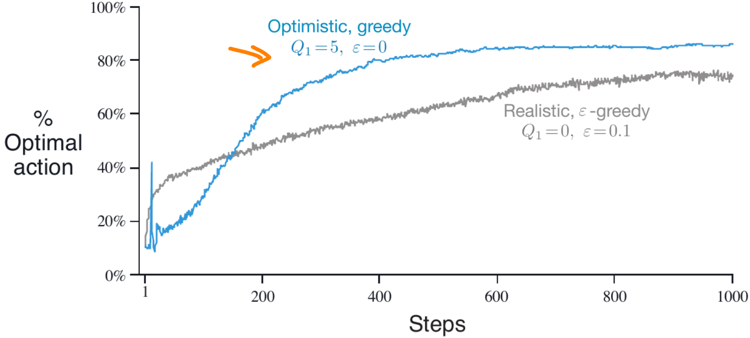
- Let $Q_1(a) = 5$ and $\alpha_n(a)$ be .1 for 10-armed testbed.
  Let the $q_*(a)$ be sampled from $\mathcal{N}(0, 1)$, and the reward distributions be $\mathcal{N}(q_*(a), 1)$.



$$Q_{n+1}(a) = Q_n(a) + \alpha(R_n - Q_n(a))$$
$$= 5 + .1(1 - 5)$$
$$= 5 - .4 = 4.6$$

Optimistic Initial values

- Let $Q_1(a) = 5$ and $\alpha_n(a)$ be .1 for 10-armed testbed.
  Let the $q_*(a)$ be sampled from $\mathcal{N}(0, 1)$, and the reward distributions be $\mathcal{N}(q_*(a), 1)$.



- Optimistic initial value technique with greedy action selection will only work for stationary distribution.

Upper-Confidence-Bound Action Selection

- Give more preference to actions whose values are uncertain

$$A_t \doteq \arg\max_a \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

Upper-Confidence-Bound Action Selection

- Give more preference to actions whose values are uncertain

$$A_t \doteq \underset{a}{\arg\max}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right]$$

- $c > 0$, controls the degree of exploration

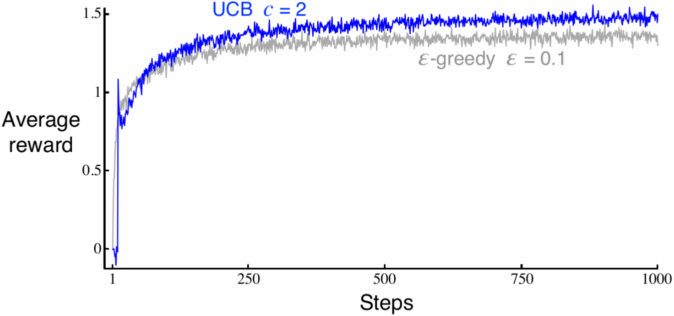
Upper-Confidence-Bound Action Selection

- Give more preference to actions whose values are uncertain

$$A_t \doteq \underset{a}{\arg\max} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

$\frac{ct}{N_t(a)}$

- $c > 0$, controls the degree of exploration
- Performance on 10-armed testbed :



Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$\frac{1}{k}$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially, $H_1(a) = 0$.

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially, $H_1(a) = 0$.
- Goal: maximize the expected reward:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

$\vec{H}$ $\quad \nabla_{\vec{H}} E[R_t]$

- Initially, $H_1(a) = 0$.
- Goal: maximize the expected reward:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

- Action preference update:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

Gradient Bandit Algorithms

- Action preference update:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \quad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

Gradient Bandit Algorithms

- Action preference update:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \quad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

$H_1(a) = 0$, $\alpha > 0$ and $\bar{R}_t$ is the average reward (baseline)

Gradient Bandit Algorithms

- 10-armed testbed; $q_*(a)$ sampled from $\mathcal{N}(4, 1)$, and reward distributions are $\mathcal{N}(q_*(a), 1)$.

N(0,1)

N(0,1000)

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially, $H_1(a) = 0$.

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially, $H_1(a) = 0$.
- Goal: maximize the expected reward:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

Gradient Bandit Algorithms

- Numerical preference for each action : $H_t(a)$
- Soft-max distribution:

$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a)$$

- Initially, $H_1(a) = 0$.
- Goal: maximize the expected reward:

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x)$$

- Action preference update:

$$H_{t+1}(a) \doteq H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)}$$

Gradient Bandit Algorithms

- Action preference update:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \quad \text{and}$$

$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

$H_1(a) = 0$, $\alpha > 0$ and $\bar{R}_t$ is the average reward (baseline)

Gradient Bandit Algorithms

- Action preference update:

$$H_{t+1}(A_t) \doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), \quad \text{and}$$
$$H_{t+1}(a) \doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), \qquad \text{for all } a \neq A_t$$

  $H_1(a) = 0$, $\alpha > 0$ and $\bar{R}_t$ is the average reward (baseline)

- How to estimate $\bar{R}_t$?

$\frac{1}{n}$ $\quad \alpha$

Example with two actions:

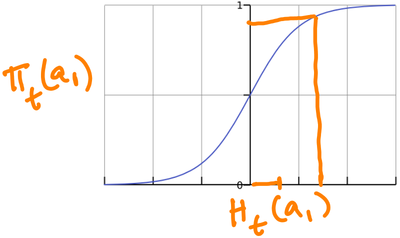
$$\begin{aligned}\mathbb{E}[R_t] &= \pi_t(a_1)q_*(a_1) + \pi_t(a_2)q_*(a_2) \\ &= \pi_t(a_1)q_*(a_1) + (1 - \pi_t(a_1))q_*(a_2)\end{aligned}$$

Gradient Bandit Algorithms

Example with two actions:

$$\begin{aligned}\mathbb{E}[R_t] &= \pi_t(a_1)q_*(a_1) + \pi_t(a_2)q_*(a_2) \\ &= \pi_t(a_1)q_*(a_1) + (1 - \pi_t(a_1))q_*(a_2)\end{aligned}$$

- $\pi_t(a_1) = \dfrac{e^{H_t(a_1)}}{e^{H_t(a_1)} + e^{H_t(a_2)}}$

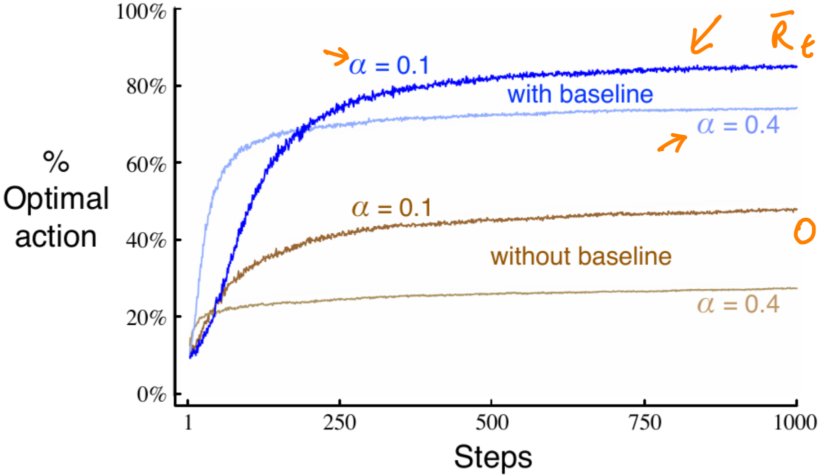


Effect of baseline in Gradient Bandit Algorithms

- Baseline: any value that does not depend on action *a*.

Effect of baseline in Gradient Bandit Algorithms

- Baseline: any value that does not depend on action a.
- 10-armed testbed; $q_*(a)$ sampled from $\mathcal{N}(4, 1)$, and reward distributions are $\mathcal{N}(q_*(a), 1)$.



Associative Search (Contextual Bandits)

- Nonassociative search

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.
- One among the two problem randomly selected in each time step.

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.
- One among the two problem randomly selected in each time step.
- Some clue about the identity of the task (state) is known.

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.
- One among the two problem randomly selected in each time step.
- Some clue about the identity of the task (state) is known.
- Choice of action should depend on previous rewards as well as on the current state.

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.
- One among the two problem randomly selected in each time step.
- Some clue about the identity of the task (state) is known.
- Choice of action should depend on previous rewards as well as on the current state.
- Each action affects only the immediate rewards and not subsequent rewards.

Associative Search (Contextual Bandits)

- Nonassociative search
- Two k-armed bandit tasks.
- One among the two problem randomly selected in each time step.
- Some clue about the identity of the task (state) is known.
- Choice of action should depend on previous rewards as well as on the current state.
- Each action affects only the immediate rewards and not subsequent rewards.
- Associative search vs. Full Reinforcement Learning problem

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.
- Tradeoff between immediate reward and delayed reward.

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.
- Tradeoff between immediate reward and delayed reward.
- In MDPs, we estimate:

Markov Decision Processes

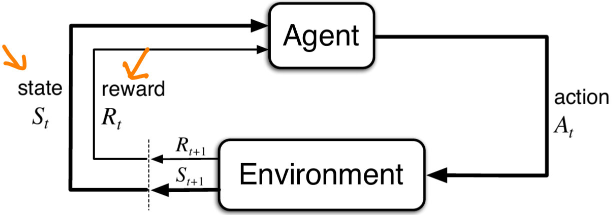
- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.
- Tradeoff between immediate reward and delayed reward.
- In MDPs, we estimate:
  - the value $q_*(s, a)$ for every action a in each state s.

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.
- Tradeoff between immediate reward and delayed reward.
- In MDPs, we estimate:
  - the value $q_*(s, a)$ for every action a in each state s.
  - the value $v_*(s)$ for each state

Markov Decision Processes

- MDPs : formalization of full reinforcement learning problem.
- Actions influence immediate reward, subsequent states and future rewards.
- Tradeoff between immediate reward and delayed reward.
- In MDPs, we estimate:
  - the value $q_*(s, a)$ for every action a in each state s.
  - the value $v_*(s)$ for each state
- Agent and Environment



Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- p is a joint probability distribution conditioned on $\mathcal{S}_t$ and $\mathcal{A}_t$

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- p is a joint probability distribution conditioned on $\mathcal{S}_t$ and $\mathcal{A}_t$
- Property

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- p is a joint probability distribution conditioned on $\mathcal{S}_t$ and $\mathcal{A}_t$
- Property

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

- Probability p completely represents the dynamics of a Markov decision process.

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- p is a joint probability distribution conditioned on $\mathcal{S}_t$ and $\mathcal{A}_t$
- Property

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

- Probability p completely represents the dynamics of a Markov decision process.
- *Markov property,* S_{t-1} S_t S_{t+1}

Markov Decision Process

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite
- Dynamics of a finite MDP

$$p(s', r \mid s, a) \doteq \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}$$

- p is a joint probability distribution conditioned on $\mathcal{S}_t$ and $\mathcal{A}_t$
- Property

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

- Probability p completely represents the dynamics of a Markov decision process.
- *Markov property*, *decision*

Markov Decision Process

- We can compute anything from the joint distribution p.

$$P(A, B \mid C)$$

$$P(A=a) = \sum_{B} P(a, B)$$

$$P(a \mid c) = \sum_{B} P(a, B \mid c)$$

Markov Decision Process

- We can compute anything from the joint distribution p.
- State-transition probability

$$p(s' \mid s, a) \doteq \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

Markov Decision Process

- We can compute anything from the joint distribution p.
- State-transition probability

$$p(s' \mid s, a) \doteq \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

- Expected rewards for state-action pairs

$$r(s, a) \doteq \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

Markov Decision Process

- We can compute anything from the joint distribution *p*.
- State-transition probability

$$p(s' \mid s, a) \doteq \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

- Expected rewards for state-action pairs

$$r(s, a) \doteq \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

- Expected rewards for state-action-next state triples

$$r(s, a, s') \doteq \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

Example: Bioreactor

- Goal is the production of some useful chemical.

Example: Bioreactor

- Goal is the production of some useful chemical.
- State has a structured representation which includes temperature, other sensory readings, ingrediants in the vat etc.

Example: Bioreactor

- Goal is the production of some useful chemical.
- State has a structured representation which includes temperature, other sensory readings, ingrediants in the vat etc.
- Action is a vector representing temperature and stirring rates.

Example: Bioreactor

- Goal is the production of some useful chemical.
- State has a structured representation which includes temperature, other sensory readings, ingrediants in the vat etc.
- Action is a vector representing temperature and stirring rates.
- Reward can be proportional to the production rate of some useful chemical.

Example: Bioreactor

- Goal is the production of some useful chemical.
- State has a structured representation which includes temperature, other sensory readings, ingrediants in the vat etc.
- Action is a vector representing temperature and stirring rates.
- Reward can be proportional to the production rate of some useful chemical.
- States and actions can have structured representations. Reward must be a scalar.

Example: Recycling Robot

- Charge level of battery: $\mathcal{S} = \{high, low\}$

Example: Recycling Robot

- Charge level of battery: $\mathcal{S} = \{high, low\}$
- Available actions:

Example: Recycling Robot

- Charge level of battery: $\mathcal{S} = \{\mathit{high}, \mathit{low}\}$
- Available actions:

  $\mathcal{A}(\mathit{high}) = \{\mathit{search}, \mathit{wait}\}$,

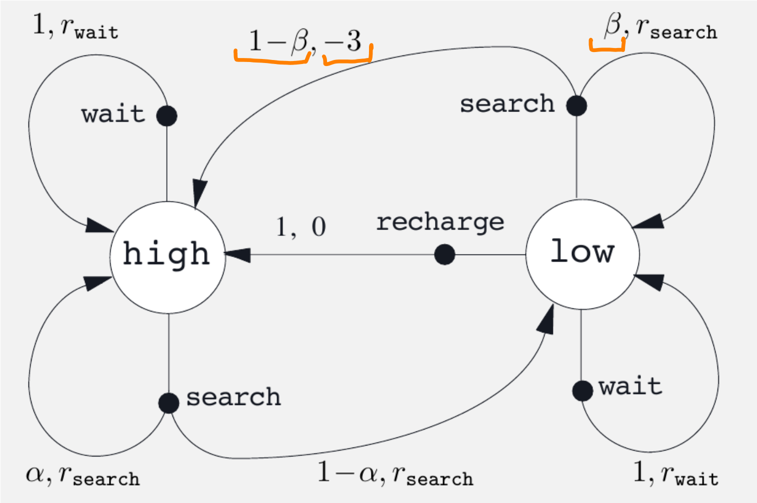
Example: Recycling Robot

- Charge level of battery: $\mathcal{S} = \{high, low\}$
- Available actions:

  $\mathcal{A}(high) = \{search, wait\}$, $\mathcal{A}(low) = \{search, wait, recharge\}$

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1 - \alpha$	$r_{\texttt{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	-

Example: Recycling Robot



Goals and Rewards

- Goal of an agent is determined in terms of rewards.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.
  - Robot escaping from maze: -1 reward for every time prior to escape.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.
  - Robot escaping from maze: -1 reward for every time prior to escape.
  - Collecting empty soda cans: 0 in every time step and 1 whenever an empty can is collected.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.
  - Robot escaping from maze: -1 reward for every time prior to escape.
  - Collecting empty soda cans: 0 in every time step and 1 whenever an empty can is collected.
  - Agent learning to play chess or checkers: 1 for winning, -1 for losing and 0 for remaining states.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.
  - Robot escaping from maze: -1 reward for every time prior to escape.
  - Collecting empty soda cans: 0 in every time step and 1 whenever an empty can is collected.
  - Agent learning to play chess or checkers: 1 for winning, -1 for losing and 0 for remaining states.
- Rewards must be set up such that maximizing them will achieve the goal.

Goals and Rewards

- Goal of an agent is determined in terms of rewards.
- Maximize the cummulative sum of rewards.
- Examples of rewards:
  - Robot learning to walk: reward proportional to forward motion in each time step.
  - Robot escaping from maze: -1 reward for every time prior to escape.
  - Collecting empty soda cans: 0 in every time step and 1 whenever an empty can is collected.
  - Agent learning to play chess or checkers: 1 for winning, -1 for losing and 0 for remaining states.
- Rewards must be set up such that maximizing them will achieve the goal.
- Rewards must convey *what* is to be achieved, and not *how* to achieve it.

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.
- *Episode* : any sort of repeated agent-environment interaction

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.
- *Episode* : any sort of repeated agent-environment interaction
  - Plays of a game

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.
- *Episode* : any sort of repeated agent-environment interaction
  - Plays of a game
  - Trips through a maze

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.
- *Episode* : any sort of repeated agent-environment interaction
  - Plays of a game
  - Trips through a maze
- Episodic task

Returns and Episodes

- Maximize *expected returns*

  $G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T$, where T is the final time step.
- *Episode* : any sort of repeated agent-environment interaction
  - Plays of a game
  - Trips through a maze
- Episodic task
- Each episode ends in a *Terminal state*, with a different reward for different outcomes.

Continuing task and Discounting

- *Continuing task*:

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞
- What should be the expected returns?

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞
- What should be the expected returns?
- *Discounted returns*

  $G_t \doteq \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots,$

  where $0 \leq \gamma \leq 1$ is the discount rate.

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞
- What should be the expected returns?
- *Discounted returns*
  $G_t \doteq \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots,$
  where $0 \leq \gamma \leq 1$ is the discount rate.
- If $\gamma = 0$, the agent is "myopic". If γ is close to 1, then agent is "farsighted".

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞
- What should be the expected returns?
- *Discounted returns*

  $G_t \doteq \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots,$

  where $0 \leq \gamma \leq 1$ is the discount rate.
- If $\gamma = 0$, the agent is "myopic". If γ is close to 1, then agent is "farsighted".
- $G_t \doteq R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \cdots)$

Continuing task and Discounting

- *Continuing task*: final time step T can be ∞
- What should be the expected returns?
- *Discounted returns*

  $G_t \doteq \gamma^0 R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \cdots,$

  where $0 \leq \gamma \leq 1$ is the discount rate.
- If $\gamma = 0$, the agent is "myopic". If γ is close to 1, then agent is "farsighted".
- $G_t \doteq R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \cdots)$

  $G_t \doteq R_{t+1} + \gamma G_{t+1}$

Problems

- How should the dynamics be modified to apply to episodic tasks?

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Problems

- How should the dynamics be modified to apply to episodic tasks?

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

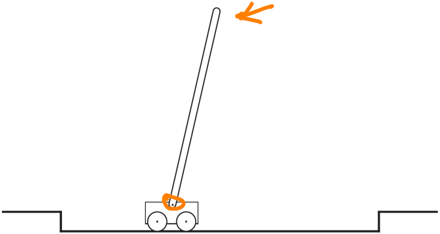
-

Exercise 3.7 Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve? □

$$G_t \doteq R_{t+1} + R_{t+2} + \ldots + R_T \qquad (3.7)$$

-1

Pole-Balancing



- Rewards would depend on whether this is an episodic task with short episodes or a continuous task.

Unified notation

- Unified notation for both Episodic and Continuous tasks

Unified notation

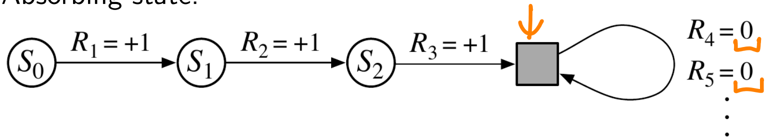
- Unified notation for both Episodic and Continuous tasks
- State representation for Episodic task $S_{t,i}$

Unified notation

- Unified notation for both Episodic and Continuous tasks
- State representation for Episodic task $S_{t,i}$
- We don't have to distinguish between different episodes.

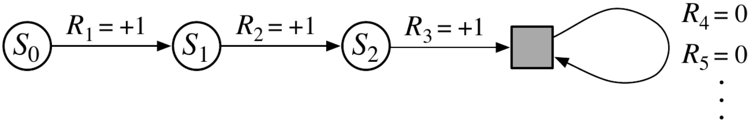
Unified notation

- Unified notation for both Episodic and Continuous tasks
- State representation for Episodic task $S_{t,i}$
- We don't have to distinguish between different episodes.
- Absorbing state:



Unified notation

- Unified notation for both Episodic and Continuous tasks
- State representation for Episodic task $S_{t,i}$
- We don't have to distinguish between different episodes.
- Absorbing state:



- We can now use discounted reward for both types of tasks

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

$$G_t = R_t + \gamma G_{t+1}$$

where $T = \infty$ or $\gamma = 1$ (but not both).

Policies and Value Functions

- Policy (π) : A mapping from states to probability distributions (over actions). Notation $\pi(a|s)$.

Policies and Value Functions

- Policy (π) : A mapping from states to probability distributions (over actions). Notation $\pi(a|s)$.

Q. If the current state is S_t, and actions are selected according to stochastic policy (π), then what is the expectation of R_{t+1} in terms of π and the four-argument function p ?

$$R_{t+1} = \sum_a \pi(a|s_t) \sum_{s'} \sum_r p(s',r|s_t,a) \cdot r$$

Policies and Value Functions

- Policy (π) : A mapping from states to probability distributions (over actions). Notation $\pi(a|s)$.

Q. If the current state is S_t, and actions are selected according to stochastic policy (π), then what is the expectation of R_{t+1} in terms of π and the four-argument function p ?

- *State-value function* of a state under a policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

Policies and Value Functions

- Policy (π) : A mapping from states to probability distributions (over actions). Notation $\pi(a|s)$.

Q. If the current state is S_t, and actions are selected according to stochastic policy (π), then what is the expectation of R_{t+1} in terms of π and the four-argument function p ?

- *State-value function* of a state under a policy π

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right]$$

- *Action-value function* under a policy π

$$q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

$$= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right]$$

Policies and value functions

Q. Give an equation for q_π in terms of v_π and the four-argument p.

$$q_\pi(s,a) = E[G_t \mid S_t = s, A_t = a]$$

$$= \sum_{s'} \sum_{r} p(s', r \mid s, a)[r + \gamma v_\pi(s')]$$

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2}$$

$$\gamma G_{t+1}$$

Policies and value functions

Q. Give an equation for q_π in terms of v_π and the four-argument p.

- In RL, we want to estimate the value functions v_π and q_π.

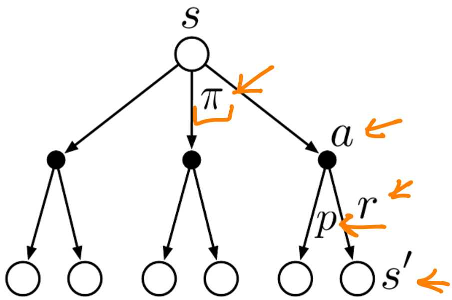
Policies and value functions

Q. Give an equation for q_π in terms of v_π and the four-argument p.

- In RL, we want to estimate the value functions v_π and q_π.
- *Bellman equation* for v_π

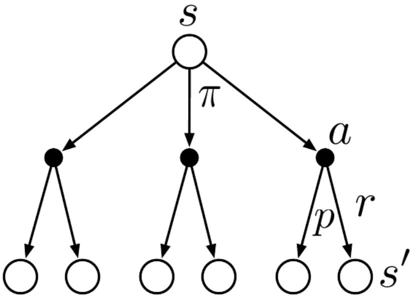
$$\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big] \\
&= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}
\end{aligned}$$

Policies and value functions



Backup diagram for v_π

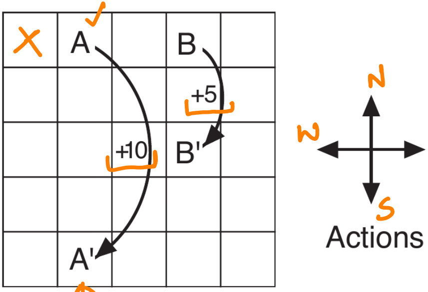
Policies and value functions



Backup diagram for v_π

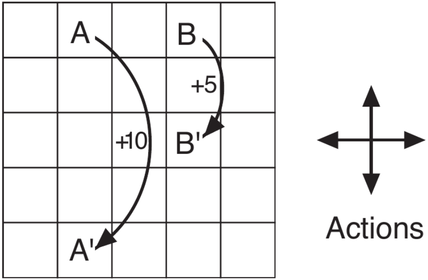
- Bellman equations form the basis of how we compute, approximate and learn v_π.

Gridworld Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

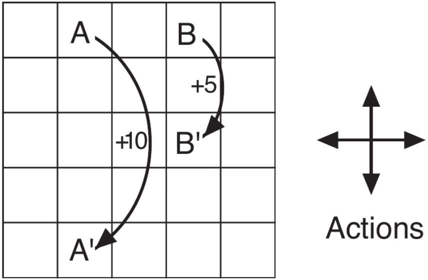
Gridworld Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- Policy (π): All four actions selected with equal probability. Discount rate: $\gamma = .9$.

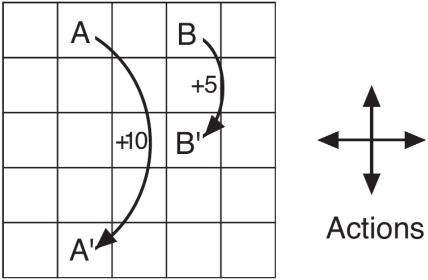
Gridworld Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

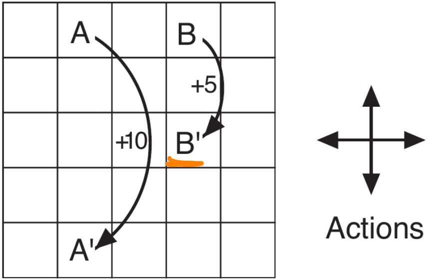
- Policy (π): All four actions selected with equal probability. Discount rate: $\gamma = .9$.
- Grid on the right shows the value function, $v_\pi(s)$, found for $\gamma = .9$.

Gridworld Example



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Gridworld Example

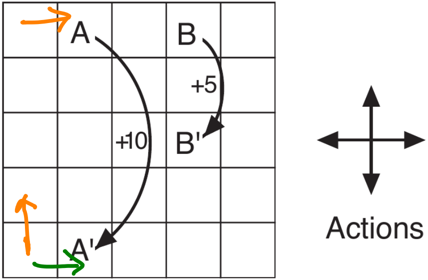


3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- How is $v_\pi(B')$ related to value of neighbouring states?

$$0 + \frac{1}{4} \times .9 \times (1.9 + .7 - .4 - .6) = .36$$

Gridworld Example

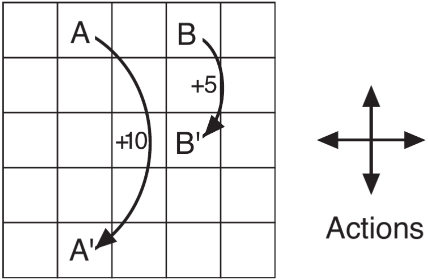


3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- How is $v_\pi(B')$ related to value of neighbouring states?
- Why is $v_\pi(A) < 10$?

$q(s, a)$

Gridworld Example

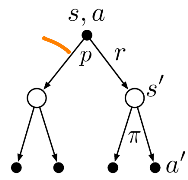


3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- How is $v_\pi(B')$ related to value of neighbouring states?
- Why is $v_\pi(A) < 10$?
- Why is $v_\pi(B) > 5$?

Action value

Exercise 3.17 What is the Bellman equation for action values, that is, for q_π? It must give the action value $q_\pi(s,a)$ in terms of the action values, $q_\pi(s',a')$, of possible successors to the state–action pair (s,a). Hint: the backup diagram to the right corresponds to this equation. Show the sequence of equations analogous to (3.14), but for action values. □



q_π backup diagram

$$q_\pi(s,a) = E\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid s, a\right]$$

$$= \sum_{s'} \sum_{r} p(s',r \mid s,a)\left[r + \gamma \sum_{a'} \pi(a' \mid s') \times q_\pi(s',a')\right]$$

Optimal Policies and Optimal Value Functions

- Policy π is better than $(\geq)$ policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$

Optimal Policies and Optimal Value Functions

- Policy π is better than ($\geq$) policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than ($\geq$) is a partial ordering over set of all policies.

Optimal Policies and Optimal Value Functions

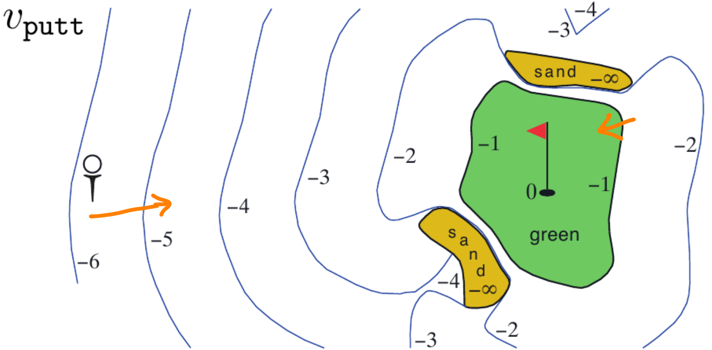
- Policy π is better than $(\geq)$ policy π' *iff* $v_{\pi}(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than $(\geq)$ is a partial ordering over set of all policies.
- There is always one policy which is better than all the other policies. (Optimal policy)

Optimal Policies and Optimal Value Functions

- Policy π is better than ($\geq$) policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than ($\geq$) is a partial ordering over set of all policies.
- There is always one policy which is better than all the other policies. (Optimal policy)
- State values of optimal policy, $v_*(s) \doteq \max_\pi v_\pi(s)$, for all $s \in \mathcal{S}$

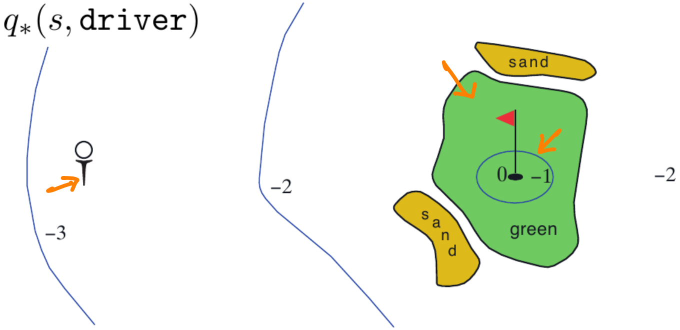
Golf Example

Golf: Only putter

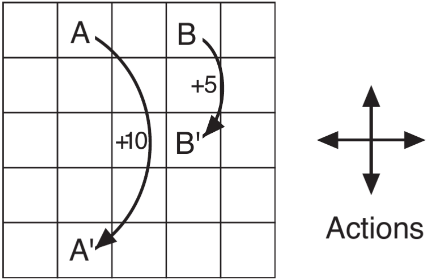


Golf: Driver first

$q_*(s, \texttt{driver})$



Gridworld Example



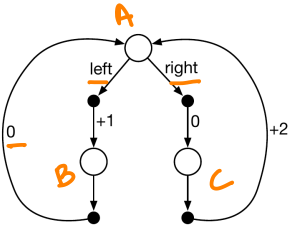
3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

- How to find the value of all the states?

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s'] \Big]$$

$$= \sum_a \pi(a|s) \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma v_\pi(s') \Big], \quad \text{for all } s \in \mathcal{S}$$



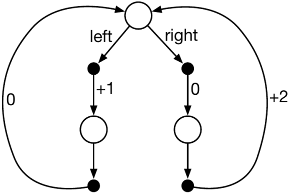
Let $\pi(left|A) = \pi(right|A) = 0.5$, $\gamma = .9$. Find $v_\pi(A)$.

$v(A) = \frac{1}{2} \times [1 + .9\, v(B)] + \frac{1}{2}[0 + .9\, v(C)]$

$v(B) = 0 + .9\, v(A)$; $v(C) = 2 + .9\, v(A)$

$v(A) = \frac{140}{19}$

Example



$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3}$

$\gamma = 0$

Let $\pi(left|A) = \pi(right|A) = 0.5$, $\gamma = .9$. Find $v_\pi(A)$.

$$v_\pi(A) = \frac{140}{19}$$

$\frac{180}{19}$ $\frac{100}{19}$

Optimal Policies and Optimal Value Functions

- Policy π is better than ($\geq$) policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than ($\geq$) is a partial ordering over set of all policies.
- There is always one policy which is better than all the other policies. (Optimal policy)
- State values of optimal policy, $v_*(s) \doteq \max_\pi v_\pi(s)$, for all $s \in \mathcal{S}$

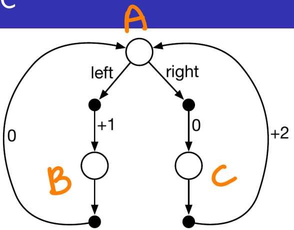
Optimal Policies and Optimal Value Functions

- Policy π is better than ($\geq$) policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than ($\geq$) is a partial ordering over set of all policies.
- There is always one policy which is better than all the other policies. (Optimal policy)
- State values of optimal policy, $v_*(s) \doteq \max_\pi v_\pi(s)$, for all $s \in \mathcal{S}$
- $q_*(s, a) \doteq \max_\pi q_\pi(s, a)$, for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$

Optimal Policies and Optimal Value Functions

- Policy π is better than ($\geq$) policy π' *iff* $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$
- Better than ($\geq$) is a partial ordering over set of all policies.
- There is always one policy which is better than all the other policies. (Optimal policy)
- State values of optimal policy, $v_*(s) \doteq \max_\pi v_\pi(s)$, for all $s \in \mathcal{S}$
- $q_*(s,a) \doteq \max_\pi q_\pi(s,a)$, for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$
- $q_*(s,a) \doteq \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$

Example



Let $\gamma = .9$. Find $v_*(A)$, $v_*(B)$, $v_*(C)$, $q_*(A, \mathit{left})$ and $q_*(A, \mathit{right})$.

Let $\pi(\text{right} \mid A) = 1$

$A = 0 + .9\,C$

$C = 2 + .9\,A$

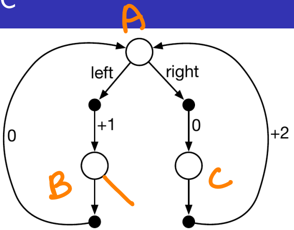
$B = 0 + .9\,A$

$A = \frac{180}{19}$

$B = \frac{162}{19}$

$C = \frac{200}{19}$

Example



Let $\gamma = .9$. Find $v_*(A)$, $v_*(B)$, $v_*(C)$, $q_*(A, left)$ and $q_*(A, right)$.

$\pi(left \mid A) = 1$

$A = 1 + .9B$

$B = 0 + .9A$

$C = 2 + .9A$

$A = \frac{100}{19}$

$B = \frac{90}{19}$

$C = \frac{128}{19}$

Example

$$v_*(A) = \frac{180}{19} \qquad v_*(C) = \frac{200}{19}$$

$$v_*(B) = \frac{162}{19}$$

$$q_*(A, left) = E[R_{t+1} + \gamma v_*(S_{t+1})]$$

$$= 1 + .9 \times \frac{162}{19} = 8.67$$

$$q_*(A, right) = 0 + .9 \times \frac{200}{19} = 9.47$$

Bellman Optimality Equations

$$
\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_a \sum_{s', r} p(s', r \mid s, a) \big[r + \gamma v_*(s') \big].
\end{aligned}
$$

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a \Big] \\
&= \sum_{s', r} p(s', r \mid s, a) \Big[r + \gamma \max_{a'} q_*(s', a') \Big].
\end{aligned}
$$

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function?

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$
- $f(3) + f(4) \neq f(3 + 4)$

  5 5 7

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$
- $f(3) + f(4) \neq f(3 + 4)$
- Bellman equation for $v_\pi(s)$ give us SLE for a policy π

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$
- $f(3) + f(4) \neq f(3 + 4)$
- Bellman equation for $v_\pi(s)$ give us SLE for a policy π
- Bellman optimality equation for $v_*(s)$ give us a system of non-linear equations.

Finding $v_*(s)$

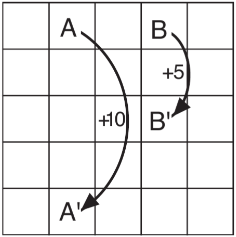
- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$
- $f(3) + f(4) \neq f(3 + 4)$
- Bellman equation for $v_\pi(s)$ give us SLE for a policy π
- Bellman optimality equation for $v_*(s)$ give us a system of non-linear equations.
- Optimal policy is easy to determine if we know $v_*(s)$.

$R_{t+1} + \gamma v_*(S_{t+1})$

Finding $v_*(s)$

- Let $f(x) = \max\{x, 5\}$
- Is $f(x)$ a linear function? $f(x) + f(y) = f(x + y)$
- $f(3) + f(4) \neq f(3 + 4)$
- Bellman equation for $v_\pi(s)$ give us SLE for a policy π
- Bellman optimality equation for $v_*(s)$ give us a system of non-linear equations.
- Optimal policy is easy to determine if we know $v_*(s)$. Assign non-zero probability to only those actions that maximize $q_*(s, a)$.

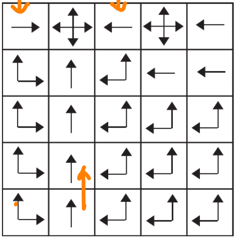
Optimal Gridworld



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

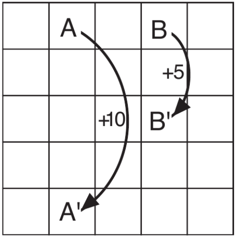
v_*



π_*

- Gridworld: $\gamma = 0.9$

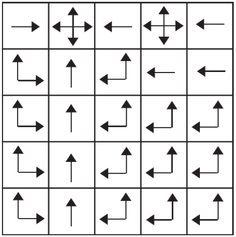
Optimal Gridworld



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

- Gridworld: $\gamma = 0.9$
- Example 3.9: Bellman Optimality Equations for the Recycling Robot

Solving the Belman optimality equation

- We won't solve non-linear equations

Solving the Belman optimality equation

- We won't solve non-linear equations
- In practice, we don't know the dynamics of the environment accurately.

Solving the Belman optimality equation

- We won't solve non-linear equations
- In practice, we don't know the dynamics of the environment accurately.
- Also, we don't have enough computational resources to find exact solutions.

Solving the Belman optimality equation

- We won't solve non-linear equations
- In practice, we don't know the dynamics of the environment accurately.
- Also, we don't have enough computational resources to find exact solutions.
- We are interested to find approximate solutions to Bellman optimality equation.

Solving the Belman optimality equation

- We won't solve non-linear equations
- In practice, we don't know the dynamics of the environment accurately.
- Also, we don't have enough computational resources to find exact solutions.
- We are interested to find approximate solutions to Bellman optimality equation.
- For small, finite state sets we can find approximations using tables with one entry for each state. Such methods are called tabular methods.

Solving the Belman optimality equation

$p(s', r \mid s, a)$

- We won't solve non-linear equations
- In practice, we don't know the dynamics of the environment accurately.
- Also, we don't have enough computational resources to find exact solutions.
- We are interested to find approximate solutions to Bellman optimality equation.
- For small, finite state sets we can find approximations using tables with one entry for each state. Such methods are called tabular methods.
- When there are too many states, we must use some parameterized function to represent states.

Tabular methods

- Chapter 4: Dynamic Programming
- Chapter 5: Monte Carlo Methods
- Chapter 6: Temporal-Difference Learning

G_t R_{t+1} + 1, 2, 3, 6

- Prediction problem: Estimating $v_\pi(\cdot)$ for a policy π.

- Prediction problem: Estimating $v_\pi(\cdot)$ for a policy π.
- We solved this using Bellman equation, which assumes that the *dynamics* $(p(s', r|s, a))$ is known.

- Prediction problem: Estimating $v_\pi(\cdot)$ for a policy π.
- We solved this using Bellman equation, which assumes that the *dynamics* $(p(s', r|s, a))$ is known.
- How to estimate $v_\pi(s)$ when dynamics is not known?

Chapter 6: Temporal-Difference Learning

- Prediction problem: Estimating $v_\pi(\cdot)$ for a policy π.
- We solved this using Bellman equation, which assumes that the *dynamics* $(p(s', r|s, a))$ is known.
- How to estimate $v_\pi(s)$ when dynamics is not known?

  Temporal-Difference (TD) Learning

Chapter 6: Temporal-Difference Learning

- Prediction problem: Estimating $v_\pi(\cdot)$ for a policy π.
- We solved this using Bellman equation, which assumes that the *dynamics* ($p(s', r|s, a)$) is known.
- How to estimate $v_\pi(s)$ when dynamics is not known?

  Temporal-Difference (TD) Learning
- We will be comparing TD with Monte Carlo Methods (MC)

Constant-α Monte Carlo

- Monte carlo methods wait till the end of an episode to update $V(S_t)$.

Constant-α Monte Carlo

- Monte carlo methods wait till the end of an episode to update $V(S_t)$.
- $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$ (Constant-α MC)

$$R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots \gamma^{n-1} R_n$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots$$

Constant-α Monte Carlo

- Monte carlo methods wait till the end of an episode to update $V(S_t)$.
- $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$ (Constant-α MC)
- Step-size parameter: *Exponential recency-weighted average*

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}$$

Constant-α Monte Carlo

- Monte carlo methods wait till the end of an episode to update $V(S_t)$.
- $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$ (Constant-α MC)
- Step-size parameter: *Exponential recency-weighted average*

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\Big[R_n - Q_n\Big] \\
&= \alpha R_n + (1-\alpha)Q_n \\
&= \alpha R_n + (1-\alpha)\left[\alpha R_{n-1} + (1-\alpha)Q_{n-1}\right] \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + (1-\alpha)\alpha R_{n-1} + (1-\alpha)^2 \alpha R_{n-2} + \\
&\qquad \cdots + (1-\alpha)^{n-1}\alpha R_1 + (1-\alpha)^n Q_1 \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}$$

- Update rule is suitable for non-stationary environments.

Temporal-Difference Learning

- Temporal-Difference methods update on every time step.

Temporal-Difference Learning

- Temporal-Difference methods update on every time step.
- $V(S_t) \leftarrow V(S_t) + \alpha[R_t + \gamma V(S_{t+1}) - V(S_t)]$

S_t R_{t+1} S_{t+1}

Temporal-Difference Learning

- Temporal-Difference methods update on every time step.
- $V(S_t) \leftarrow V(S_t) + \alpha[R_t + \gamma V(S_{t+1}) - V(S_t)]$
  (Tabular *TD*(0) or *one-step TD*)

Temporal-Difference Learning

- Temporal-Difference methods update on every time step.
- $V(S_t) \leftarrow V(S_t) + \alpha[R_t + \gamma V(S_{t+1}) - V(S_t)]$
(Tabular *TD*(0) or *one-step TD*)
- TD(0) is a *bootstrapping method* because the update is based on an existing update.

$G_t = R_{t+1} + \gamma G_{t+1}$

$$v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s] \qquad (6.3)$$
$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \qquad (\text{from } (3.9))$$
$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \qquad (6.4)$$

Tabular *TD*(0) or *one-step TD*

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Tabular *TD*(0) or *one-step TD*

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

- Policy π is given. We are evaluating policy π by estimating v_π (Prediction problem).

Driving Home Example

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

- Reward = Time-taken;

Driving Home Example

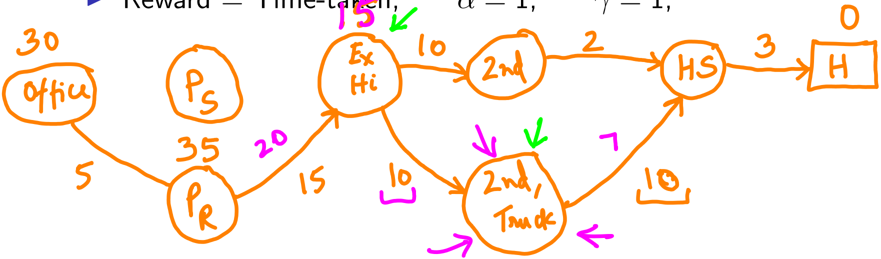
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

- Reward = Time-taken; $\alpha = 1$;

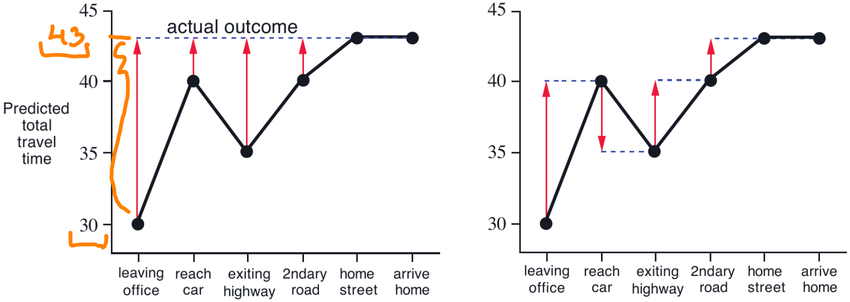
Driving Home Example

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

- Reward = Time-taken; $\alpha = 1$; $\gamma = 1$;



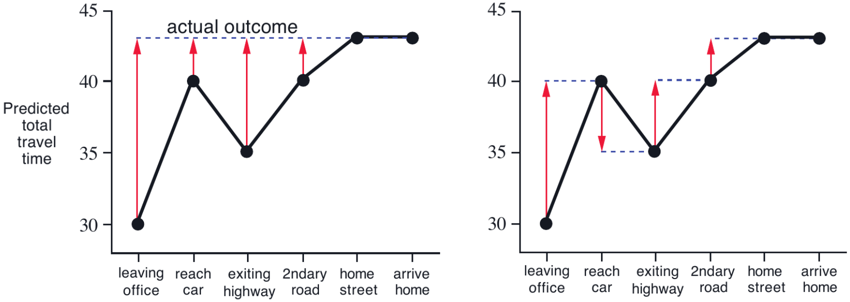
Driving Home Example



$G_t - V(S_t)$ $\quad \gamma = 1$ $\quad R_{t+1} + V(S_{t+1}) - V(S_t)$

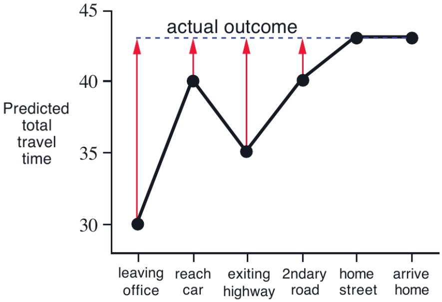
$$V(EH) = V(EH) + \alpha[G_t - V(EH)]$$
$$= 15 + \alpha[23 - 15]$$

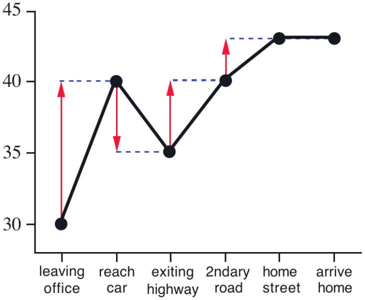
Driving Home Example



- MC may produce large updates to a node (and all the previous nodes).

Driving Home Example





- MC may produce large updates to a node (and all the previous nodes).
- TD update is proportional to the change over each time step.

Advantages of TD Prediction Methods

- We don't need to know the dynamics $p(s', a|s, a)$ of the environment.

Advantages of TD Prediction Methods

- We don't need to know the dynamics $p(s', a|s, a)$ of the environment.
- TD approach is more efficient for long episodes because updates are made in each time step.

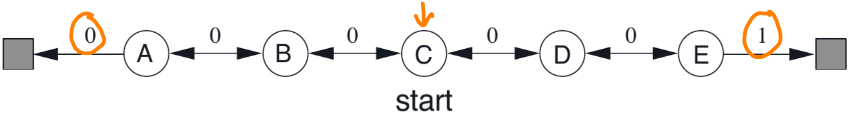
Advantages of TD Prediction Methods

- We don't need to know the dynamics $p(s', a|s, a)$ of the environment.
- TD approach is more efficient for long episodes because updates are made in each time step.
- Both TD and Monte Carlo methods converge asymptotically to the correct predictions.

Advantages of TD Prediction Methods

- We don't need to know the dynamics $p(s', a|s, a)$ of the environment.
- TD approach is more efficient for long episodes because updates are made in each time step.
- Both TD and Monte Carlo methods converge asymptotically to the correct predictions.
- Empirically, TD methods tend to converge faster compared to constant-α MC methods.

Markov Reward Process (MRP)

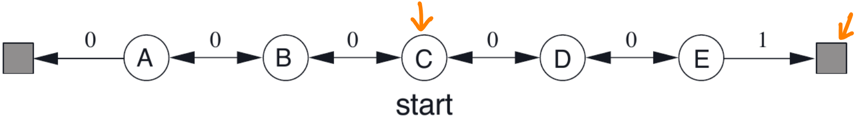


- Markov decision process without actions

Markov Reward Process (MRP)

- Markov decision process without actions
- A possible episode : C 0 B 0 C 0 D 0 E 1

Markov Reward Process (MRP)



$\gamma = 1$

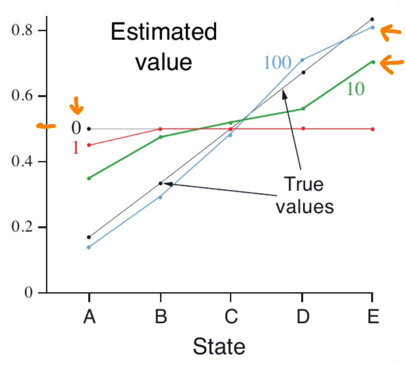
- Markov decision process without actions
- A possible episode : C 0 B 0 C 0 D 0 E 1
- Assuming that rewards are undiscounted, the actual rewards are the probability of reaching the terminal state on the right.

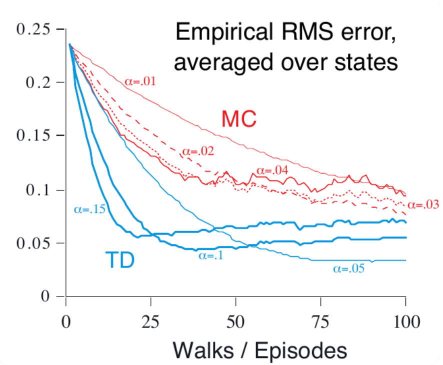
$v_\pi(C) = \frac{3}{6}$

Markov Reward Process (MRP)

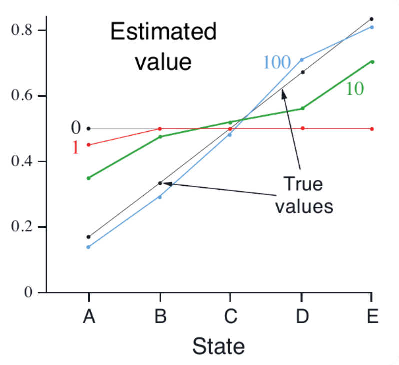
- Markov decision process without actions
- A possible episode : C 0 B 0 C 0 D 0 E 1
- Assuming that rewards are undiscounted, the actual rewards are the probability of reaching the terminal state on the right.
- True $v_\pi(\cdot)$ values for A, B, C, D and E are $\frac{1}{6}$, $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$ and $\frac{5}{6}$ respectively.

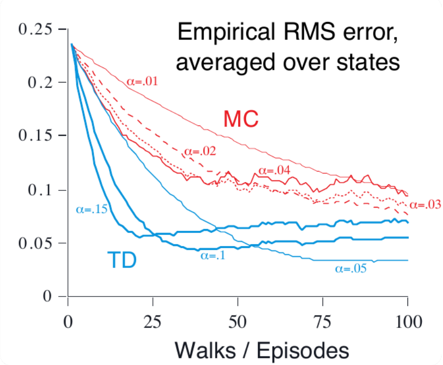
Markov Reward Process





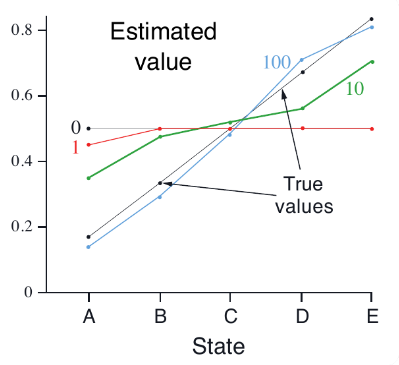
Markov Reward Process

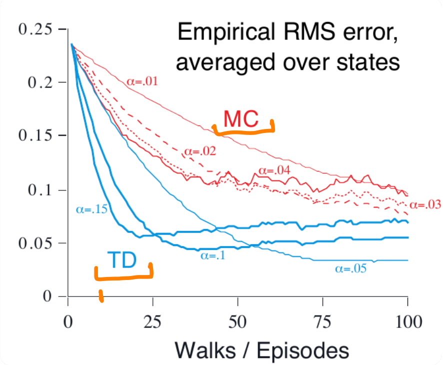




- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.

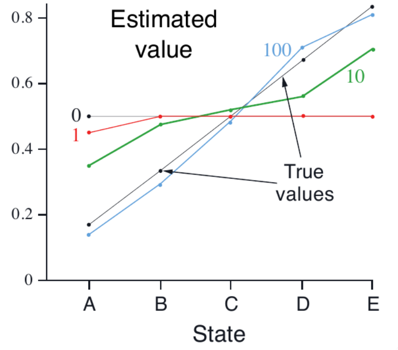
Markov Reward Process

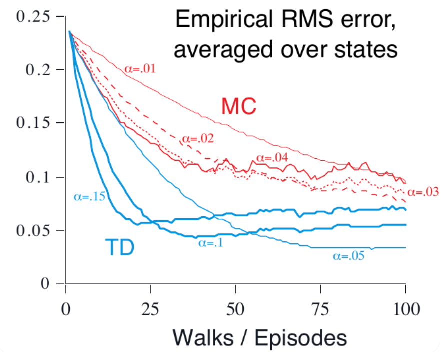




- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.
- Right graph: Root mean-squared (RMS) error between learned value function and true value function.

Markov Reward Process



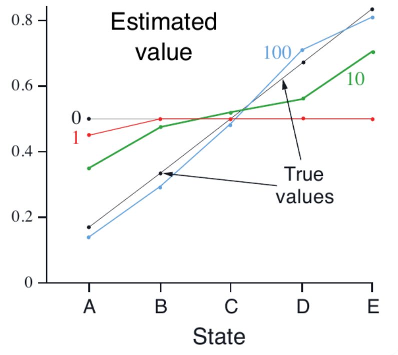


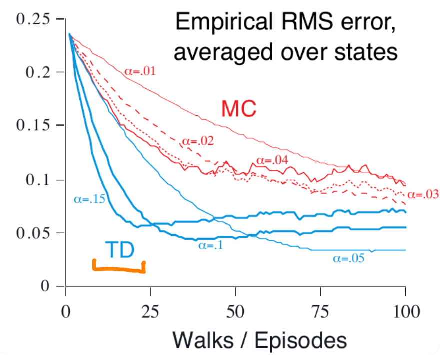
- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.
- Right graph: Root mean-squared (RMS) error between learned value function and true value function.
- Right graph: TD method performs better compared to MC.

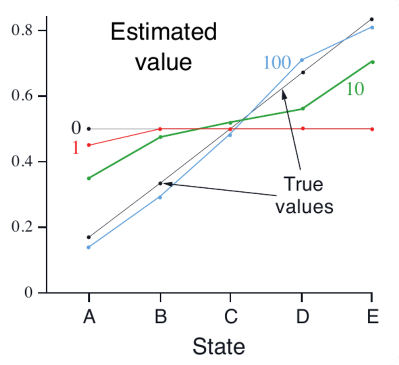
Markov Reward Process (MRP)

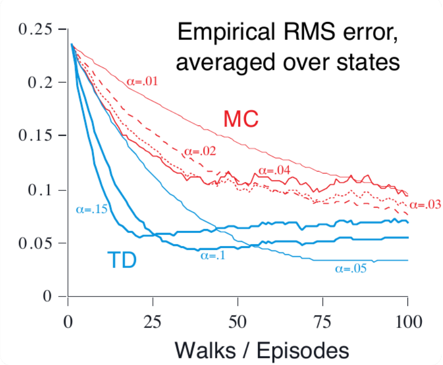
- Markov decision process without actions
- A possible episode : C 0 B 0 C 0 D 0 E 1
- Assuming that rewards are undiscounted, the actual rewards are the probability of reaching the terminal state on the right.
- True $v_{\pi}(\cdot)$ values for A, B, C, D and E are $\frac{1}{6}$, $\frac{2}{6}$, $\frac{3}{6}$, $\frac{4}{6}$ and $\frac{5}{6}$ respectively.

Markov Reward Process



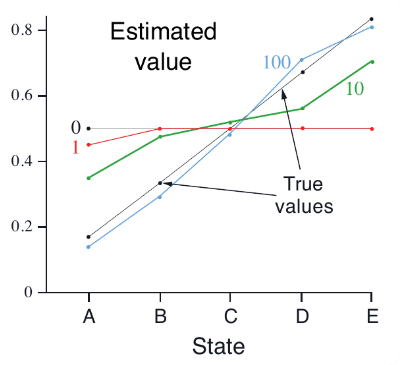


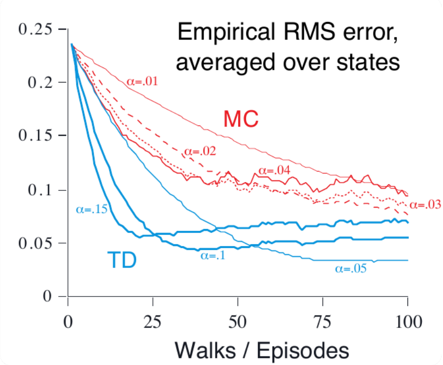




- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.

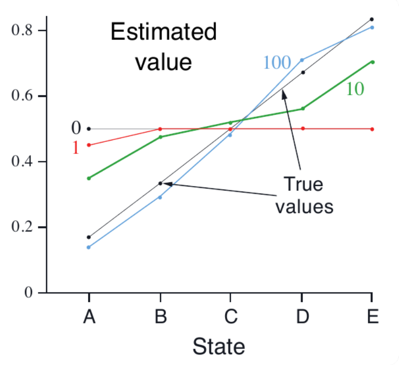
Markov Reward Process

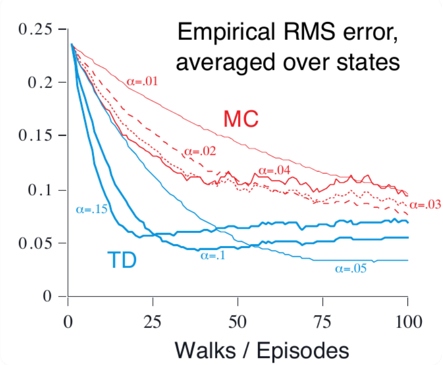




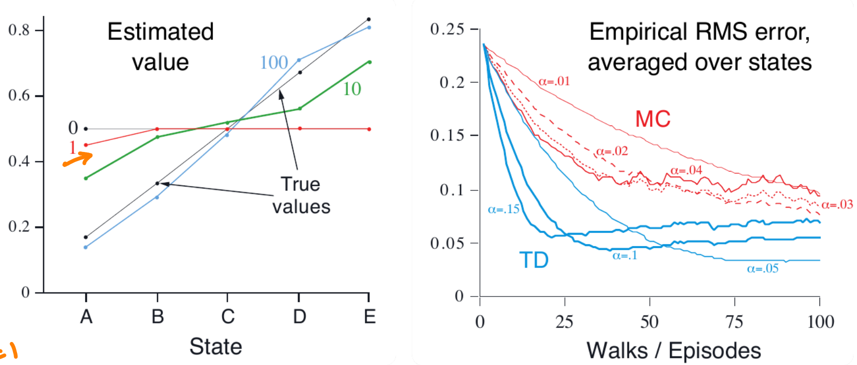
- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.
- Right graph: Root mean-squared (RMS) error between learned value function and true value function.

Markov Reward Process





- Left graph: $\alpha = .1$, Values will fluctuate indefinitely.
- Right graph: Root mean-squared (RMS) error between learned value function and true value function.
- Right graph: TD method performs better compared to MC.



$\gamma = 1$

Exercise 6.3 From the results shown in the left graph of the random walk example it appears that the first episode results in a change in only $V(\mathsf{A})$. What does this tell you about what happened on the first episode? Why was only the estimate for this one state changed? By exactly how much was it changed? □

$$v(S_t) = v(S_t) + \alpha\left[R_{t+1} + \gamma v(S_{t+1}) - v(S_t)\right]$$

$$= \frac{1}{2} + \alpha\left[0 + 0 - \frac{1}{2}\right]$$

Convergence under Batch updating

- Batch updating: Value function is changed only once by the sum of all the increments.

$$V(S) = V(S) + \underbrace{\alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]}$$

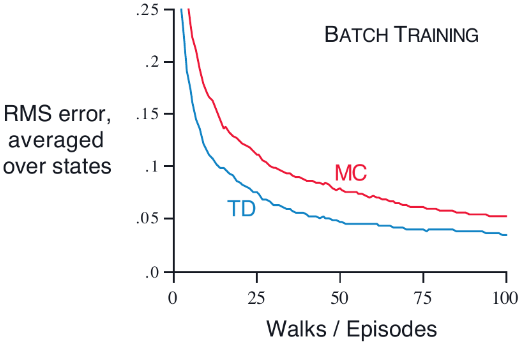
Convergence under Batch updating

- Batch updating: Value function is changed only once by the sum of all the increments.
- Under batch updating, both TD(0) and MC methods converge as long as α is small.

Markov Reward Process under Batch updating

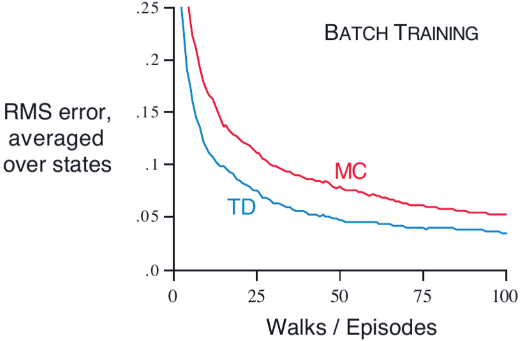
- After each new episode, all episodes seen so far are treated as a batch.

$v_\pi(s) - V(s)$



Markov Reward Process under Batch updating

- After each new episode, all episodes seen so far are treated as a batch.



- They converge to different answers.

Markov Reward Process under Batch updating

A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

Markov Reward Process under Batch updating

A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

- What will be the batch update under TD(0) method?

$V(A) = \frac{1}{2}$ $\quad V(B) = \frac{3}{4}$

$V(A) = \frac{1}{2} + .01\left[0 + V(B) - \frac{1}{2}\right]$

$= \frac{3}{4}$

Markov Reward Process under Batch updating

A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

- What will be the batch update under TD(0) method?

- Both V(A) and V(B) will converge to 0.75.

Markov Reward Process under Batch updating

$A, 0, B, 0$	$B, 1$
$B, 1$	$B, 1$
$B, 1$	$B, 1$
$B, 1$	$B, 0$

- What will be the batch update under MC method?

$V(A) = \frac{1}{2} + .01\left[-\frac{1}{2}\right]$

$= \frac{1}{2} + .01\left[0 - V(A)\right]$

$V(A) = 0$

Markov Reward Process under Batch updating

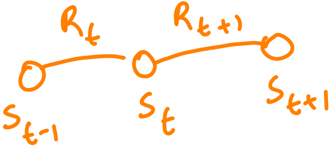
A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

- What will be the batch update under MC method?

- V(B) converges to 0.75, V(A) converges to 0.

Why is there a difference?

- MC method estimate depends on the peculiarites of the episodes (i.e. sequence of rewards). It is not making use of the fact that R_{t+1} is dependent only on S_t and is independent of R_t.



Why is there a difference?

- MC method estimate depends on the peculiarites of the episodes (i.e. sequence of rewards). It is not making use of the fact that R_{t+1} is dependent only on S_t and is independent of R_t.
- In other words, MC method is not making use of the *Markov property* assumption, because its estimate is based on the entire sequence of rewards in an episode.

Why is there a difference?

- MC method estimate depends on the peculiarites of the episodes (i.e. sequence of rewards). It is not making use of the fact that R_{t+1} is dependent only on S_t and is independent of R_t.
- In other words, MC method is not making use of the *Markov property* assumption, because its estimate is based on the entire sequence of rewards in an episode.
- TD method uses the current estimate for S_{t+1} to find the update (bootstrapping). So, the updates are not dependent on any particular episode(s).

Why is there a difference?

- MC method estimate depends on the peculiarites of the episodes (i.e. sequence of rewards). It is not making use of the fact that R_{t+1} is dependent only on S_t and is independent of R_t.
- In other words, MC method is not making use of the *Markov property* assumption, because its estimate is based on the entire sequence of rewards in an episode.
- TD method uses the current estimate for S_{t+1} to find the update (bootstrapping). So, the updates are not dependent on any particular episode(s).
- TD method will provide a better estimate (converge faster) when the underlying stochastic process has the Markov property.

Comparing MC and TD(0)

- If mean squared error is computed for actual $v_\pi(s)$ based on the underlying Markov Random Process, then TD(0) method will be better.

Comparing MC and TD(0)

- If mean squared error is computed for actual $v_\pi(s)$ based on the underlying Markov Random Process, then TD(0) method will be better.
- If we assume that the underlying stochastic process has the Markov property, then what is the Maximum Likelihood Estimate of the parameters of the stochastic process?

Comparing MC and TD(0)

- If mean squared error is computed for actual $v_\pi(s)$ based on the underlying Markov Random Process, then TD(0) method will be better.
- If we assume that the underlying stochastic process has the Markov property, then what is the Maximum Likelihood Estimate of the parameters of the stochastic process?
- $P(s'|s,a)$, $\mathbb{E}[R_{t+1}|s,a]$

Comparing MC and TD(0)

- If mean squared error is computed for actual $v_\pi(s)$ based on the underlying Markov Random Process, then TD(0) method will be better.
- If we assume that the underlying stochastic process has the Markov property, then what is the Maximum Likelihood Estimate of the parameters of the stochastic process?
- $P(s'|s,a)$, $\mathbb{E}[R_{t+1}|s,a]$
- TD(0) method gives the MLE of the parameters if the underlying process has the Markov property.

Comparing MC and TD(0)

- If mean squared error is computed for actual $v_\pi(s)$ based on the underlying Markov Random Process, then TD(0) method will be better.
- If we assume that the underlying stochastic process has the Markov property, then what is the Maximum Likelihood Estimate of the parameters of the stochastic process?
- $P(s'|s,a)$, $\mathbb{E}[R_{t+1}|s,a]$
- TD(0) method gives the MLE of the parameters if the underlying process has the Markov property.
- *Certainty-equivalence estimate*: The estimated value will be exactly correct if the assumed model was exactly correct.

Sarsa : On-policy TD Control

- Policy evaluation (Prediction problem) vs. Finding optimal policy (Control problem)

Sarsa : On-policy TD Control

- Policy evaluation (Prediction problem) vs. Finding optimal policy (Control problem)
- On-policy method: Use a policy π and then attempt to improve the same policy π.

Sarsa : On-policy TD Control

- Policy evaluation (Prediction problem) vs. Finding optimal policy (Control problem)
- On-policy method: Use a policy π and then attempt to improve the same policy π.
- Instead of $v_\pi(s)$ we will estimate $q_\pi(s, a)$.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

Sarsa : On-policy TD Control

- Policy evaluation (Prediction problem) vs. Finding optimal policy (Control problem)
- On-policy method: Use a policy π and then attempt to improve the same policy π.
- Instead of $v_\pi(s)$ we will estimate $q_\pi(s, a)$.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

- $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$

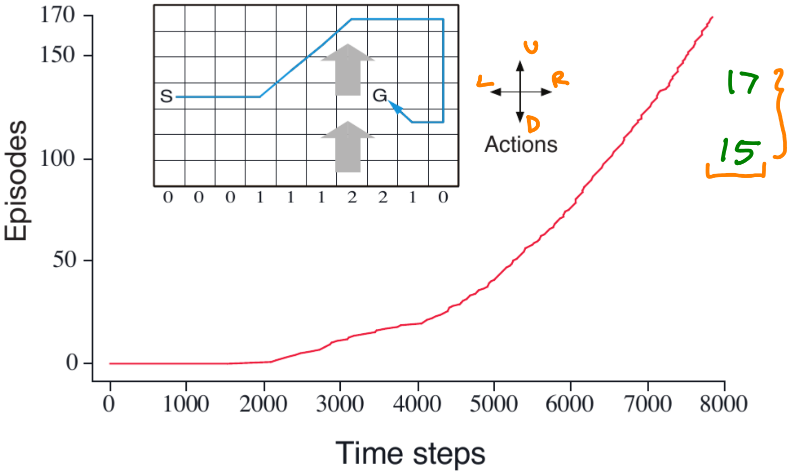
Sarsa : On-policy TD Control

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
    Initialize S
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Loop for each step of episode:
        Take action A, observe R, S'
        Choose A' from S' using policy derived from Q (e.g., ε-greedy)
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
        $S \leftarrow S'$; $A \leftarrow A'$;
    until S is terminal

- Sarsa converges to the optimal policy with probability 1 as long as all state-action pairs are visited an infinite number of times and ϵ decreases with time.

Applying ϵ-greedy Sarsa to Windy Gridworld



- Actions, Rewards, Wind
- Initial $Q(s, a) = 0$, $\epsilon = 0.1$, $\alpha = .5$, $\gamma = 1$, (constant ϵ)

ϵ-greedy and ϵ-soft policies

- ϵ-greedy policy: greedy action is selected with probability $1 - \epsilon$ and *any* action with probability $\frac{\epsilon}{|\mathcal{A}(s)|}$

ϵ-greedy and ϵ-soft policies

- ϵ-greedy policy: greedy action is selected with probability $1 - \epsilon$ and *any* action with probability $\frac{\epsilon}{|\mathcal{A}(s)|}$
- ϵ-soft policy: *all* actions have a probability $\geq \frac{\epsilon}{|\mathcal{A}(s)|}$

ϵ-greedy and ϵ-soft policies

- ϵ-greedy policy: greedy action is selected with probability $1 - \epsilon$ and *any* action with probability $\frac{\epsilon}{|\mathcal{A}(s)|}$
- ϵ-soft policy: *all* actions have a probability $\geq \frac{\epsilon}{|\mathcal{A}(s)|}$
- Is every ϵ-greedy policy an ϵ-soft policy? yes

Q-learning: Off-policy TD Control

- Q-learning update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Q-learning: Off-policy TD Control

- Q-learning update rule:
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$
- Directly approximates $q_*(s, a)$ independent of the policy being followed to select actions.

ε – greedy

Q-learning: Off-policy TD Control

- Q-learning update rule:
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$
- Directly approximates $q_*(s, a)$ independent of the policy being followed to select actions.
- Convergence to q_* is guaranteed if all state-action pairs are updated a large number of times and α is small.

Q-learning: Off-policy TD Control

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:

  Initialize S

  Loop for each step of episode:

    Choose A from S using policy derived from Q (e.g., ε-greedy)

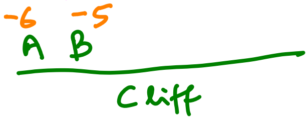
    Take action A, observe R, S'

    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

    $S \leftarrow S'$

  until S is terminal

$-6 + \alpha[-1 + -5 - (-6)]$



Q-learning: Off-policy TD Control

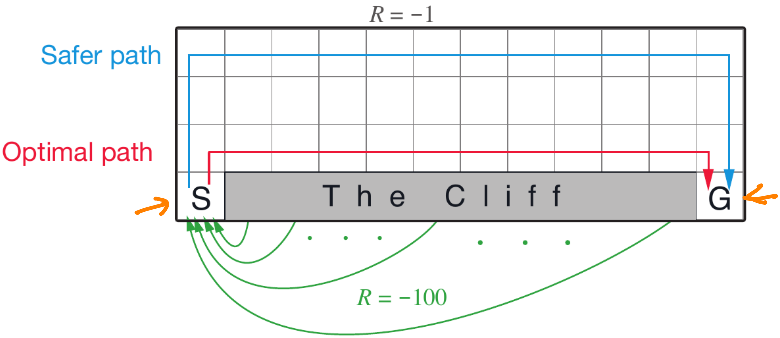
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

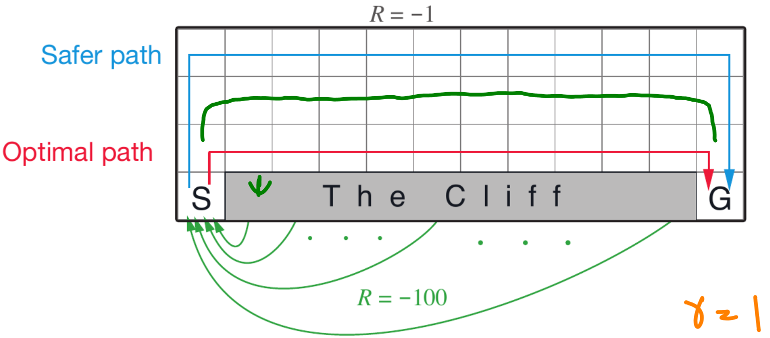
q_π π ε-greedy   0-greedy

- Which algorithm will converge to q_* in a faster manner? Sarsa or Q-learning.

Cliff Walking Example

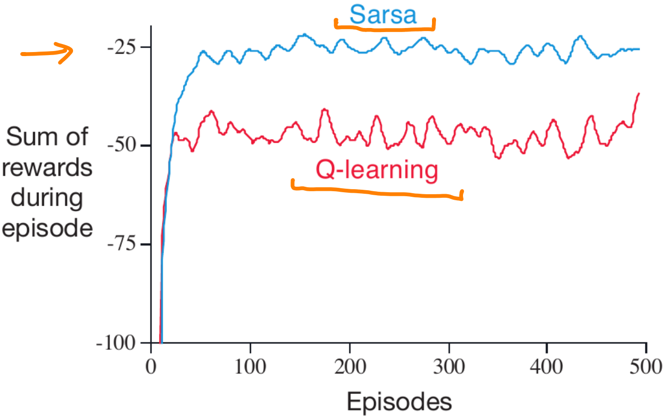


Cliff Walking Example

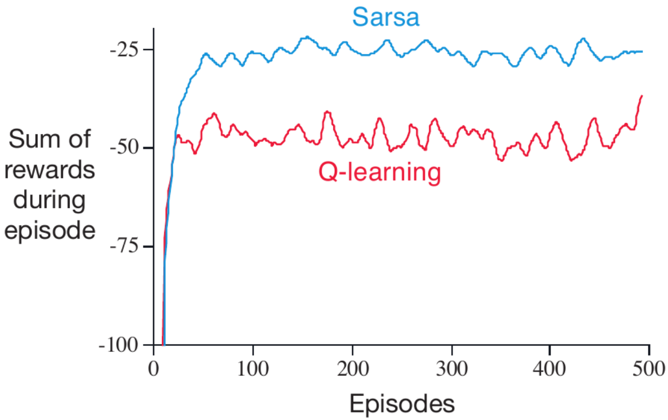


- Suppose we use ϵ-greedy action selection, $\epsilon = 0.1$

Sarsa vs. Q-learning

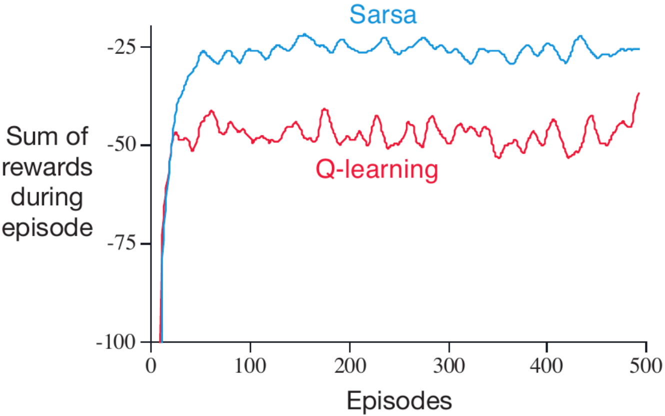


Sarsa vs. Q-learning



- Online performance of Q-learning can be worse than that of Sarsa.

Sarsa vs. Q-learning



- Online performance of Q-learning can be worse than that of Sarsa.
- If ϵ is decreased gradually, both algorithms will asymptotically converge to the optimal policy.

Expected Sarsa

- Just like Q-learning except the update rule:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big]$$
$$\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big]$$

ε-greedy

Q(S', A')

Expected Sarsa

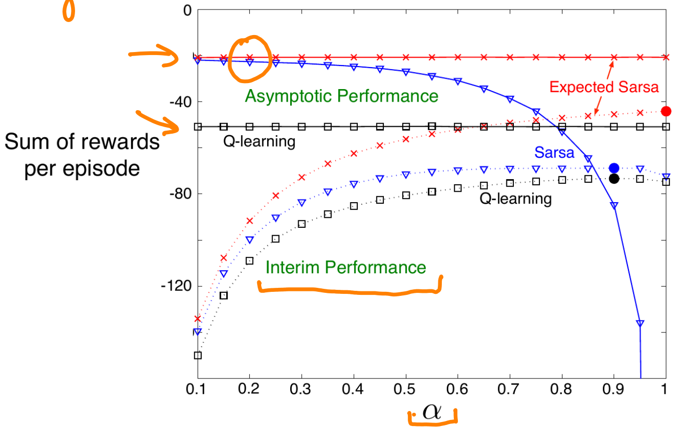


Figure 6.3: Interim and asymptotic performance of TD control methods on the cliff-walking task as a function of α. All algorithms used an ε-greedy policy with $\varepsilon = 0.1$. Asymptotic performance is an average over 100,000 episodes whereas interim performance is an average over the first 100 episodes. These data are averages of over 50,000 and 10 runs for the interim and asymptotic cases respectively. The solid circles mark the best interim performance of each method. Adapted from van Seijen et al. (2009).

Expected Sarsa

- Just like Q-learning except the update rule:

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \Big] \\
&\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \Big]
\end{aligned}
$$

$\epsilon/2$ - greedy

Target

$\epsilon = 0$

ϵ - greedy

ϵ - greedy, $\epsilon > 0$

Expected Sarsa

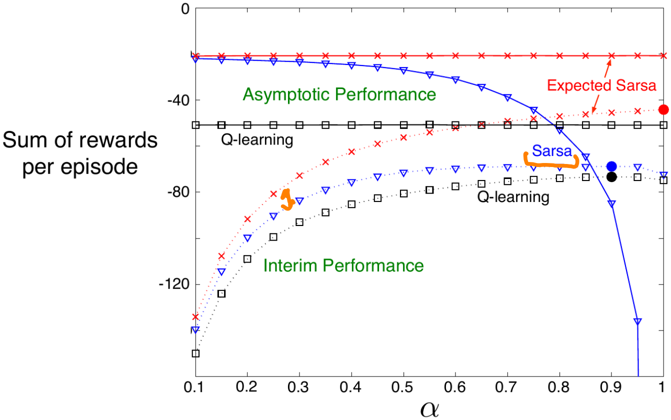


Figure 6.3: Interim and asymptotic performance of TD control methods on the cliff-walking task as a function of α. All algorithms used an ε-greedy policy with $\varepsilon = 0.1$. Asymptotic performance is an average over 100,000 episodes whereas interim performance is an average over the first 100 episodes. These data are averages of over 50,000 and 10 runs for the interim and asymptotic cases respectively. The solid circles mark the best interim performance of each method. Adapted from van Seijen et al. (2009).

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.
- What will happen if we gradually decrease ϵ ?

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.
- What will happen if we gradually decrease ϵ ?
- *Target policy* vs. *Behavior policy*

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.
- What will happen if we gradually decrease ϵ ?
- *Target policy* vs. *Behavior policy*
- The version of Expected Sarsa that we saw is on-policy or off-policy?

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.
- What will happen if we gradually decrease ϵ ?
- *Target policy* vs. *Behavior policy*
- The version of Expected Sarsa that we saw is on-policy or off-policy?
- We saw the On-policy version of Expected Sarsa; off-policy versions are also possible.

Expected Sarsa

- Online performance of Expected Sarsa is better than Sarsa and Q-learning for wide range of α values.
- Eliminates variance due to random selection of A_{t+1} in Sarsa.
- What will happen if we gradually decrease ϵ ?
- *Target policy* vs. *Behavior policy*
- The version of Expected Sarsa that we saw is on-policy or off-policy?
- We saw the On-policy version of Expected Sarsa; off-policy versions are also possible.
- Q-learning is a special case of Off-policy Expected Sarsa.

Maximization bias

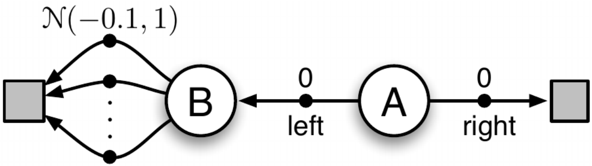
- We used ϵ-greedy behavior policy in all the algorithms.

Maximization bias

- We used ϵ-greedy behavior policy in all the algorithms.
- ϵ-greedy policy involves a maximization operation. This can lead to *maximization bias*.

Maximization bias

- We used ϵ-greedy behavior policy in all the algorithms.
- ϵ-greedy policy involves a maximization operation. This can lead to *maximization bias*.
- Maximization bias example:

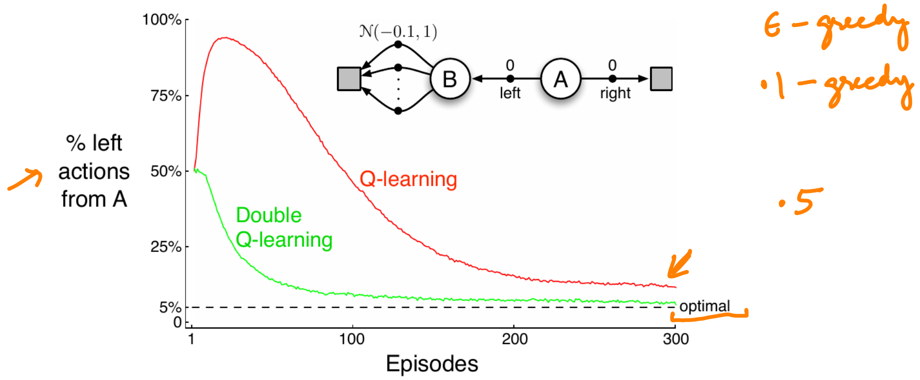


$\mu + 3\sigma$, $\alpha = .1$

$Q(B, a)$

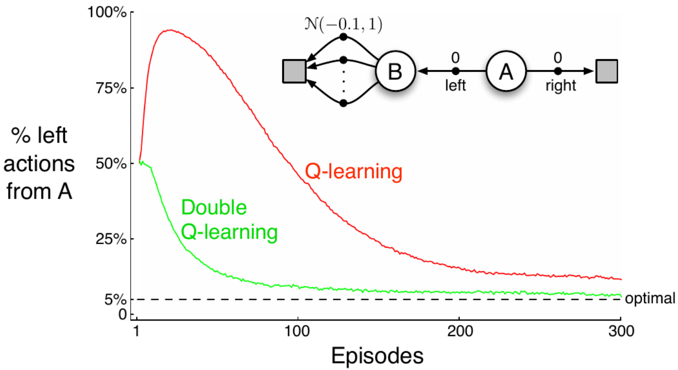
$Q(A, \text{left})$

Maximization bias



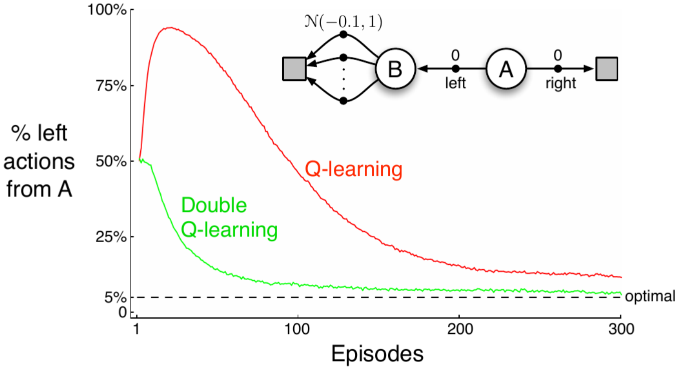
- ϵ-greedy behavior policy, $\epsilon = 0.1, \alpha = 0.1, \gamma = 1$
- averaged data over 10,000 runs

Maximization bias



- ϵ-greedy behavior policy, $\epsilon = 0.1, \alpha = 0.1, \gamma = 1$
- averaged data over 10,000 runs
- Solution: learn two estimates $Q_1(\cdot)$ and $Q_2(\cdot)$

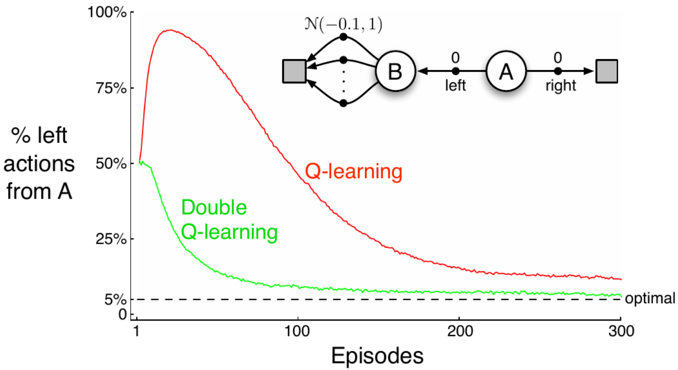
Maximization bias



- ϵ-greedy behavior policy, $\epsilon = 0.1, \alpha = 0.1, \gamma = 1$
- averaged data over 10,000 runs
- Solution: learn two estimates $Q_1(\cdot)$ and $Q_2(\cdot)$
- $Q_1(s, \text{argmax}\, Q_2(s, a))$

a

Maximization bias



- ϵ-greedy behavior policy, $\epsilon = 0.1, \alpha = 0.1, \gamma = 1$
- averaged data over 10,000 runs
- Solution: learn two estimates $Q_1(\cdot)$ and $Q_2(\cdot)$
- $Q_1(s, \operatorname{argmax}_a Q_2(s, a)) \leftarrow$ Won't have maximization bias

Double Q-learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\textit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

    $S \leftarrow S'$
  until S is terminal

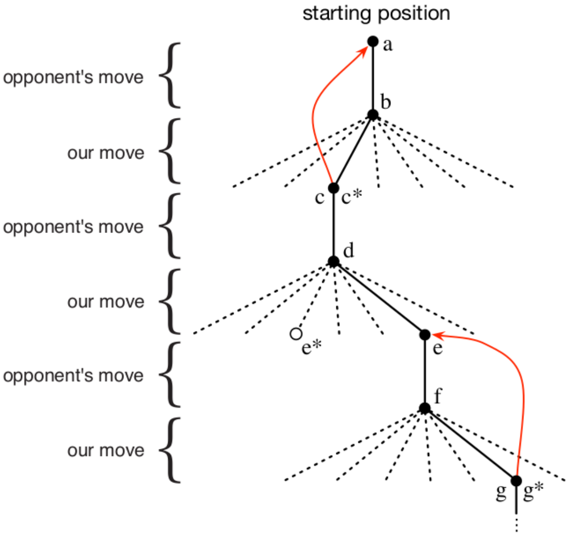
Double Q-learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$
    else:
      $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$
    $S \leftarrow S'$
  until S is terminal

- Doubles the memory requirement, but does not increase the amount of computation per step.

Tic-tac-toe example of Ch. 1



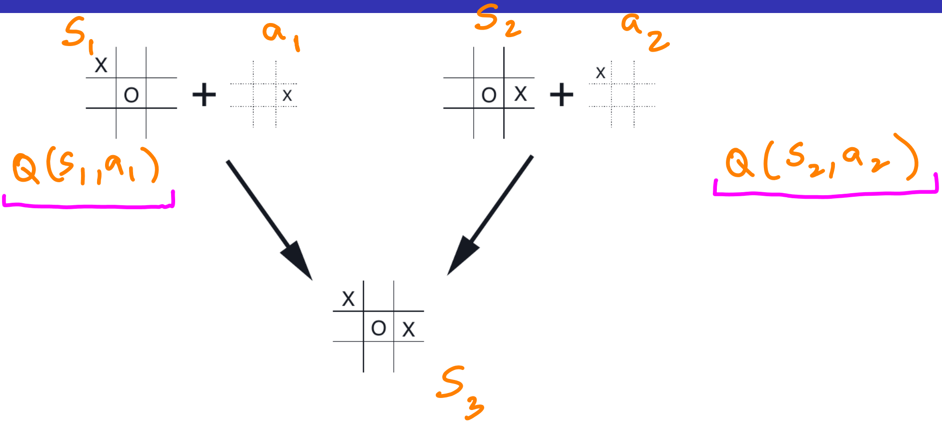
Tic-tac-toe example of Ch. 1

- Neither action-value nor state-value

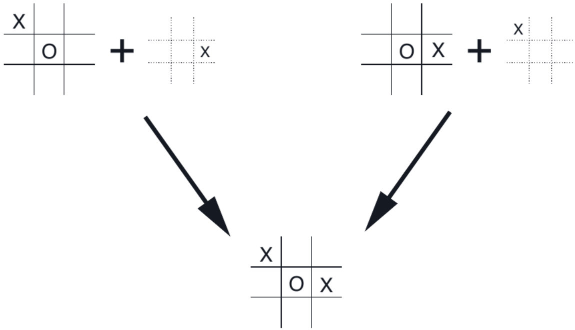
Tic-tac-toe example of Ch. 1

- Neither action-value nor state-value
- Evaluates board positions after the agent has made its move (*afterstates*).

Tic-tac-toe example of Ch. 1

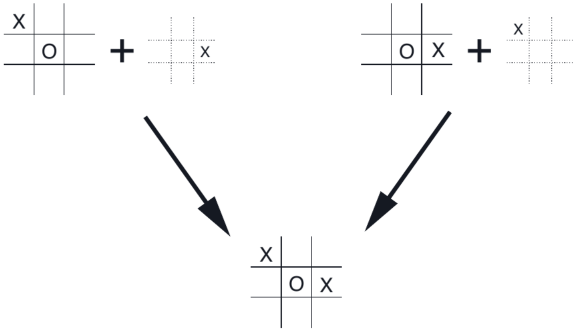


Tic-tac-toe example of Ch. 1



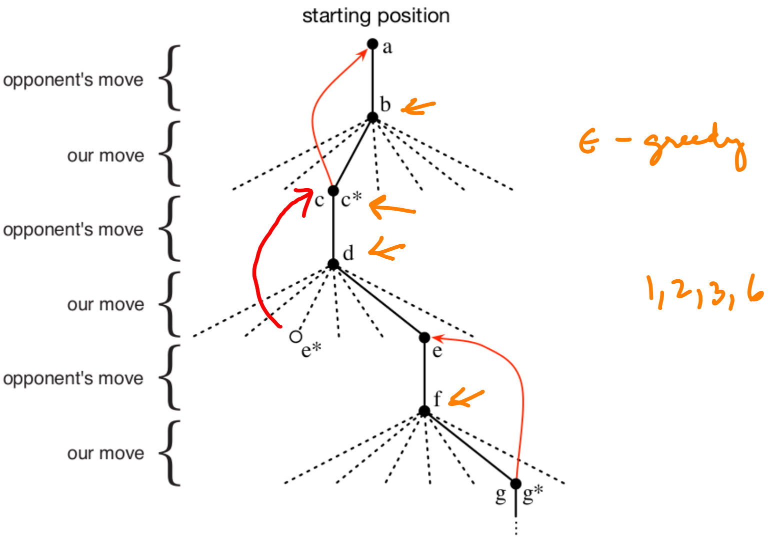
- Afterstates are useful when we are sure of the next state.

Tic-tac-toe example of Ch. 1



- Afterstates are useful when we are sure of the next state.
- This reduces the values that we have to estimate.

Is this Q-learning?



Swarm Intelligence

- Chapter 14, Richard E. Neapolitan and Xia Jiang, *Artificial Intelligence – With an Introduction to Machine Learning, Second Edition*.

Swarm Intelligence

- Chapter 14, Richard E. Neapolitan and Xia Jiang, *Artificial Intelligence – With an Introduction to Machine Learning, Second Edition*.
- Swarm Intelligence : a population of simple agents that interact locally to produce an intelligent collective behaviour.

Swarm Intelligence

- Chapter 14, Richard E. Neapolitan and Xia Jiang, *Artificial Intelligence – With an Introduction to Machine Learning, Second Edition*.
- Swarm Intelligence : a population of simple agents that interact locally to produce an intelligent collective behaviour.
- E.g. 1: Ants can find the shortest path between nest and food.

Swarm Intelligence

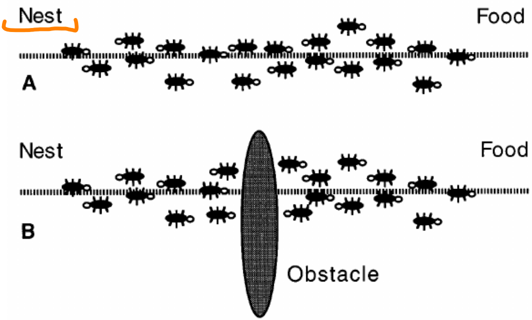
- Chapter 14, Richard E. Neapolitan and Xia Jiang, *Artificial Intelligence – With an Introduction to Machine Learning, Second Edition*.
- Swarm Intelligence : a population of simple agents that interact locally to produce an intelligent collective behaviour.
- E.g. 1: Ants can find the shortest path between nest and food.
- E.g. 2: Birds flock together in unison to avoid being preyed upon.

Swarm Intelligence

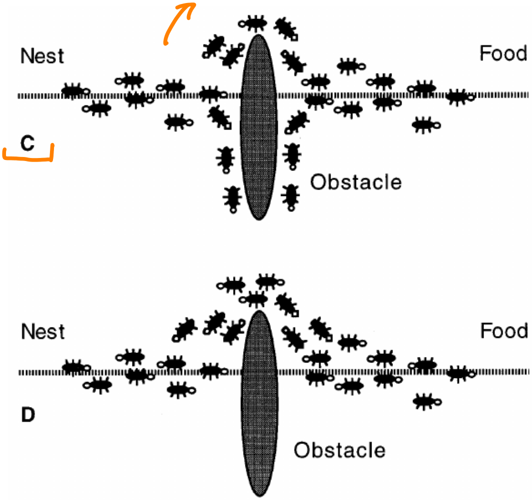
- Chapter 14, Richard E. Neapolitan and Xia Jiang, *Artificial Intelligence – With an Introduction to Machine Learning, Second Edition*.
- Swarm Intelligence : a population of simple agents that interact locally to produce an intelligent collective behaviour.
- E.g. 1: Ants can find the shortest path between nest and food.
- E.g. 2: Birds flock together in unison to avoid being preyed upon.
- Properties of swarm agents:

1. There is no top-down central command guiding the agents' behavior.
2. Each agent is able to generate some change in the environment.
3. Each agent is able to sense some change in the environment.

Ant System



Ant System



Artificial Ants for Solving the TSP

- [Dorigo and Gambardella, 1997]

Artificial Ants for Solving the TSP

- [Dorigo and Gambardella, 1997]
- Travelling salesman problem

Artificial Ants for Solving the TSP

- [Dorigo and Gambardella, 1997]
- Travelling salesman problem
- We have a complete graph.

Artificial Ants for Solving the TSP

- [Dorigo and Gambardella, 1997]
- Travelling salesman problem
- We have a complete graph.
- Artificial ants have the following additional properties:

1. Each agent k has a working memory M_k that contains the vertices the agent has already visited. The memory is emptied at the beginning of each new tour, and is updated each time a vertex is visited.

2. Each agent knows how far away vertices are from the agent's current vertex.

Steps that an Ant agent takes

1. Move to the "best" unvisited vertex (s) with probability p_0

$$s = \begin{cases} \arg\max\limits_{u \notin M_k} \left[\tau(r,u) \times \{\eta(r,u)\}^{\beta} \right] & \text{if } p \leq p_0 \\ S & \text{otherwise} \end{cases}$$

$\tau(r,u)$

$$\eta(r,u) = \left[\frac{1}{w(r,u)} \right]^{\beta}$$

Steps that an Ant agent takes

1. Move to the best unvisited vertex (s) with probability p_0

$$s = \begin{cases} \arg\max_{u \notin M_k} \left[\tau(r,u) \times \{\eta(r,u)\}^{\beta} \right] & \text{if } p \leq p_0 \\ S & \text{otherwise} \end{cases}$$

Otherwise, with probability $1 - p_0$ move to any unvisited vertex using the following probability distribution

$$p_{r,k}(s) = \begin{cases} \dfrac{\tau(r,s) \times \{\eta(r,s)\}^{\beta}}{\sum_{u \notin M_k} \tau(r,u) \times \{\eta(r,u)\}^{\beta}} & \text{if } s \notin M_k \\ 0 & \text{otherwise} \end{cases}$$

Pheromone updating

Happens when the m ant agents have completed their tour.

Pheromone updating

Happens when the m ant agents have completed their tour.

2. Global pheromone updating:
$\tau(r,s) \leftarrow (1-\alpha)\tau(r,s) + \alpha\Delta\tau(r,s)$

$\tau(r,s)$

$Q(r, a_s)$

$\alpha = .1$ $\qquad \Delta\tau(r,s) = \frac{1}{\text{length}(ST)}$

$\tau_0 = \frac{1}{n \times \text{Greedy}(ST)}$

Pheromone updating

Happens when the m ant agents have completed their tour.

2. Global pheromone updating:
   $\tau(r, s) \leftarrow (1 - \alpha)\tau(r, s) + \alpha\Delta\tau(r, s)$
   Local pheromone updating (trail evaporation):
   $\tau(r, s) \leftarrow (1 - \alpha)\tau(r, s) + \alpha\tau_0$

Performance of Ant colony system (ACS)

- Compared with Simulated Annealing (SA), Elastic Net (EN), Self organizing map (SOM) and Farthest insertion heuristic (FI).

Performance of Ant colony system (ACS)

- Compared with Simulated Annealing (SA), Elastic Net (EN), Self organizing map (SOM) and Farthest insertion heuristic (FI).
- Randomly generated five 50-vertex problem.

Problem Instance	ACS	SA	EN	SOM	FI
1	**5.86**	5.88	5.98	6.06	6.03
2	6.05	**6.01**	6.03	6.25	6.28
3	**5.57**	5.65	5.70	5.83	5.85
4	**5.70**	5.81	5.86	5.87	5.96
5	**6.17**	6.33	6.49	6.70	6.71

Is ACS similar to something we have studied?

Is ACS similar to something we have studied?

- The choice of next state is similar to ϵ-greedy strategy.

Is ACS similar to something we have studied?

- The choice of next state is similar to ϵ-greedy strategy.
- $\tau(r, s)$ is similar to $Q(r, a_s)$

Is ACS similar to something we have studied?

- The choice of next state is similar to ϵ-greedy strategy.
- $\tau(r, s)$ is similar to $Q(r, a_s)$
- Global pheromone update rule is similar to the update rule of Monte Carlo algorithm.

Is ACS similar to something we have studied?

- The choice of next state is similar to ϵ-greedy strategy.
- $\tau(r, s)$ is similar to $Q(r, a_s)$
- Global pheromone update rule is similar to the update rule of Monte Carlo algorithm.
- Important difference: The Global update is based on the shortest tour among the m swarm agents.

From the ants perspective

- Reward: $\frac{\text{food}}{\text{energy-spent}}$

From the ants perspective

- Reward: $\dfrac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently.

From the ants perspective

- Reward: $\dfrac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently. (ϵ-greedy, soft-max etc.)

From the ants perspective

- Reward: $\dfrac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently. (ϵ-greedy, soft-max etc.)
- Keep dropping pheromones along which ever path is taken.

From the ants perspective

- Reward: $\dfrac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently. (ϵ-greedy, soft-max etc.)
- Keep dropping pheromones along which ever path is taken.
- The above update rule is more like the SARSA algorithm.

From the ants perspective

- Reward: $\frac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently. (ϵ-greedy, soft-max etc.)
- Keep dropping pheromones along which ever path is taken.
- The above update rule is more like the SARSA algorithm. However, the update at each step is by a constant value.

From the ants perspective

- Reward: $\dfrac{\text{food}}{\text{energy-spent}}$
- Use some policy such that path with a higher concentration of pheromones is chosen more frequently. (ϵ-greedy, soft-max etc.)
- Keep dropping pheromones along which ever path is taken.
- The above update rule is more like the SARSA algorithm. However, the update at each step is by a constant value. Optimal action is discovered because more ants take the optimal action over time.

Co-ordinated movement of animals (Flocking)

- Birds fly in flocks, Fishes swim in schools

Co-ordinated movement of animals (Flocking)

- Birds fly in flocks, Fishes swim in schools
- Why do animals do this?

Co-ordinated movement of animals (Flocking)

- Birds fly in flocks, Fishes swim in schools
- Why do animals do this? The behaviour is primarily observed in prey animals.

Co-ordinated movement of animals (Flocking)

- Birds fly in flocks, Fishes swim in schools
- Why do animals do this? The behaviour is primarily observed in prey animals.
- Could one animal be controlling the overall behaviour of the group using some electromagnetic signal? (Some researchers actually suggested this possibility)

Co-ordinated movement of animals (Flocking)

- Birds fly in flocks, Fishes swim in schools
- Why do animals do this? The behaviour is primarily observed in prey animals.
- Could one animal be controlling the overall behaviour of the group using some electromagnetic signal? (Some researchers actually suggested this possibility)
- Can a simple model explain this complex behaviour?

Fish in a school

- Partridge (1982) : Lateral line

Fish in a school

- Partridge (1982) : Lateral line
- Blinded fish vs. Fish with lateral line removed

Fish in a school

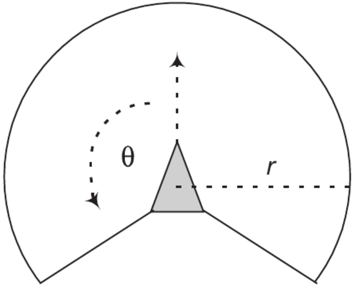
- Partridge (1982) : Lateral line
- Blinded fish vs. Fish with lateral line removed
- Reynolds (1987) : Flock's movement is determined by each individual member following simple rules.

Simulator of Bird flocking

- Member of a flock is called a bird-oid or simply boid.

Simulator of Bird flocking

- Member of a flock is called a bird-oid or simply boid.
- A given boid reacts only to other boids in a small region around itself.



Simple rules followed by Boid

1. Collision avoidance

Simple rules followed by Boid

1. Collision avoidance
2. Velocity matching

Simple rules followed by Boid

1. Collision avoidance
2. Velocity matching
3. Flock centering

Simple rules followed by Boid

1. Collision avoidance
2. Velocity matching
3. Flock centering

- (Flock model simulation)

Conclusion

- If a simple model can simulate a complex pattern, then it may have some explanatory power.

Conclusion

- If a simple model can simulate a complex pattern, then it may have some explanatory power.
- Turing's equations for patterns in nature (1954)
  https://www.weforum.org/agenda/2019/07/alan-turing-codebreaker-unlocked-secrets-of-nature/

Activator (Fire)

Inhibitor (Extinguisher)

How can we use Simulated Annealing for solving TSP?

- What should be the states?

n

$[1, 2, \cdots n]$ $n!$

How can we use Simulated Annealing for solving TSP?

- What should be the states?
- What should be the neighbouring states?

$^{n}C_{2}$

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*
- Plan: Chapter 12, 13, 14 and 16

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*
- Plan: Chapter 12, 13, 14 and 16
- Chapter 5: Adversarial search in Two-layer, Zero-sum Game

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*
- Plan: Chapter 12, 13, 14 and 16
- Chapter 5: Adversarial search in Two-layer, Zero-sum Game (Watch at 1.5x speed)

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*
- Plan: Chapter 12, 13, 14 and 16
- Chapter 5: Adversarial search in Two-layer, Zero-sum Game (Watch at 1.5x speed)
- 12/10/21 (Tuesday) : Doubt clearing for Chapter 5

- Part IV : Uncertain knowledge and reasoning (Russell and Norvig)
- Chapter 12: Stuart Russell and Peter Norvig, *Artificial Intelligence – A Modern Approach, Fourth Edition*
- Plan: Chapter 12, 13, 14 and 16
- Chapter 5: Adversarial search in Two-layer, Zero-sum Game (Watch at 1.5x speed)
- 12/10/21 (Tuesday) : Doubt clearing for Chapter 5
- 13/10/21 (Wednesday) : Doubt clearing for any topic that was covered

Propositions vs. Degree of belief

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \lor GumProblem \lor Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains:

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief:

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief: Probability theory

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief: Probability theory
  - Ontological commitments

Toothache = True

P (Toothache = True)

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief: Probability theory
  - Ontological commitments
  - Epistemological commitments

P (weather = Sunny)

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief: Probability theory
  - Ontological commitments
  - Epistemological commitments
- Probability: summarize the uncertainty due to laziness and ignorance.

Propositions vs. Degree of belief

- $Toothache \Rightarrow Cavity$
- $Toothache \Rightarrow Cavity \vee GumProblem \vee Abscess \ldots$
- $Cavity \Rightarrow Toothache$
- Problem typical of judgmental domains: medical domain, gardening, automobile repair etc.
- Degree of belief: Probability theory
  - Ontological commitments
  - Epistemological commitments
- Probability: summarize the uncertainty due to laziness and ignorance.
- The probability that a patient has a cavity, *given that she has a toothache*, is 0.8.

Probability of a proposition

- Sample space:

Probability of a proposition

- Sample space: *mutually exclusive* and *exhaustive* outcomes

Probability of a proposition

- Sample space: *mutually exclusive* and *exhaustive* outcomes
- e.g. Throw of a pair of dice: $(1,1),(1,2),\ldots,(6,6)$

Probability of a proposition

- Sample space: *mutually exclusive* and *exhaustive* outcomes
- e.g. Throw of a pair of dice: $(1,1),(1,2),\ldots,(6,6)$
- Fully specified probability model

$$0 \leq P(\omega) \leq 1 \text{ for every } \omega \text{ and } \sum_{\omega \in \Omega} P(\omega) = 1$$

Probability of a proposition

- Sample space: *mutually exclusive* and *exhaustive* outcomes
- e.g. Throw of a pair of dice: $(1,1),(1,2),\ldots,(6,6)$
- Fully specified probability model

$$0 \leq P(\omega) \leq 1 \text{ for every } \omega \text{ and } \sum_{\omega \in \Omega} P(\omega) = 1$$

- Probability of a proposition

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

Conditional probability

- Conditional probability

Conditional probability

- Conditional probability e.g. probability of rolling doubles *given that the first die is a 5*.

Conditional probability

- Conditional probability e.g. probability of rolling doubles *given that the first die is a 5.*
- $P(doubles|Die_1 = 5)$

$\neg doubles$

Double = True

↑

Conditional probability

- Conditional probability e.g. probability of rolling doubles *given that the first die is a 5.*
- $P(doubles|Die_1 = 5)$ ($Doubles$ vs. $doubles$)

Conditional probability

- Conditional probability e.g. probability of rolling doubles *given that the first die is a 5*.
- $P(doubles|Die_1 = 5)$ ($Doubles$ vs. $doubles$)
- $P(cavity) = 0.2$, $P(cavity|toothache) = 0.6$

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)} ,$$

which holds whenever $P(b) > 0$. For example,

$$P(doubles \mid Die_1 = 5) = \frac{P(doubles \wedge Die_1 = 5)}{P(Die_1 = 5)}$$

Conditional probability

- Conditional probability e.g. probability of rolling doubles *given that the first die is a 5*.
- $P(doubles|Die_1 = 5)$ $\quad$ ($Doubles$ vs. $doubles$)
- $P(cavity) = 0.2$, $P(cavity|toothache) = 0.6$

$$P(a \mid b) = \frac{P(a \wedge b)}{P(b)} ,$$

which holds whenever $P(b) > 0$. For example,

$$P(doubles \mid Die_1 = 5) = \frac{P(doubles \wedge Die_1 = 5)}{P(Die_1 = 5)}$$

- Product rule : $P(a \wedge b) = P(a|b)P(b)$

- Probability of all possibilities for *Weather*:

$$P(Weather = sunny) = 0.6$$
$$P(Weather = rain) = 0.1$$
$$P(Weather = cloudy) = 0.29$$
$$P(Weather = snow) = 0.01\ ,$$

but as an abbreviation we will allow

$$\mathbf{P}(Weather) = \langle 0.6, 0.1, 0.29, 0.01 \rangle$$

Joint Probability Distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

A, B, C

3^3

3

$P(\neg cavity \vee \neg catch \mid toothache)$

Joint Probability Distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

$2^3 = 8$

- Full joint probability distribution: d^n entries

Joint Probability Distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

- Full joint probability distribution: d^n entries
- Number of entries for $P(Cavity, Toothache, Weather)$? 16

2 2 4

Joint Probability Distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

- Full joint probability distribution: d^n entries
- Number of entries for $P(Cavity, Toothache, Weather)$? $= 16$.

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

a. $P(cavity \lor toothache)$?

Inference using full joint distribution

	toothache		$\neg$*toothache*	
	catch	$\neg$*catch*	*catch*	$\neg$*catch*
cavity	0.108	0.012	0.072	0.008
$\neg$*cavity*	0.016	0.064	0.144	0.576

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

a. $P(cavity \vee toothache)$?

$$0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$$

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

a. $P(cavity \vee toothache)$?

$0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$

b. $P(cavity)$?

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

a. $P(cavity \vee toothache)$?

$0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$

b. $P(cavity)$?

$0.108 + 0.012 + 0.072 + 0.008 = 0.2$

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

b. $P(cavity)$?

$0.108 + 0.012 + 0.072 + 0.008 = 0.2$

- Marginal probability

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

b. $P(cavity)$?

$0.108 + 0.012 + 0.072 + 0.008 = 0.2$

- Marginal probability

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

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

b. $P(cavity)$?

$0.108 + 0.012 + 0.072 + 0.008 = 0.2$

- Marginal probability

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

- Conditioning

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

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)$?

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

Inference using full joint distribution

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

d. $P(\neg cavity|toothache)?$

Inference using full joint distribution

	toothache		$\neg$*toothache*	
	catch	$\neg$*catch*	*catch*	$\neg$*catch*
cavity	0.108	0.012	0.072	0.008
$\neg$*cavity*	0.016	0.064	0.144	0.576

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

d. $P(\neg cavity|toothache)? = \dfrac{P(\neg cavity \wedge toothache)}{P(toothache)}$

Inference using full joint distribution

	toothache		$\neg$*toothache*	
	catch	$\neg$*catch*	*catch*	$\neg$*catch*
cavity	0.108	0.012	0.072	0.008
$\neg$*cavity*	0.016	0.064	0.144	0.576

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$$= \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

d. $P(\neg cavity|toothache)? = \dfrac{P(\neg cavity \wedge toothache)}{P(toothache)}$

$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$= (0.108 + 0.012)\alpha = 0.12\alpha$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$= (0.108 + 0.012)\alpha = 0.12\alpha$

$P(\neg cavity|toothache)?$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$= (0.108 + 0.012)\alpha = 0.12\alpha$

$P(\neg cavity|toothache)? = \dfrac{P(\neg cavity \wedge toothache)}{P(toothache)}$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

c. $P(cavity|toothache)? = \dfrac{P(cavity \wedge toothache)}{P(toothache)}$

$= (0.108 + 0.012)\alpha = 0.12\alpha$

$P(\neg cavity|toothache)? = \dfrac{P(\neg cavity \wedge toothache)}{P(toothache)}$

$= (0.016 + 0.064)\alpha = 0.08\alpha$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

$$0.12\alpha + 0.08\alpha = 1$$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

$$0.12\alpha + 0.08\alpha = 1 \ ; \qquad \alpha = 5$$

Using normalization constant

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

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

$$0.12\alpha + 0.08\alpha = 1 \; ; \qquad \alpha = 5$$

$$P(cavity|toothache) = 0.12\alpha = 0.6$$

Independence

- Suppose we add a fourth R.V. : *Weather*.

Independence

- Suppose we add a fourth R.V. : *Weather*.
- $P(Toothache, Catch, Cavity, Weather)$

Independence

- Suppose we add a fourth R.V. : $Weather$.
- $P(Toothache, Catch, Cavity, Weather)$
- $P(toothache, catch, \neg cavity, cloudy) =$
  $P(cloudy | toothache, catch, \neg cavity) P(toothache, catch, \neg cavity)$

  P(cloudy)

Independence

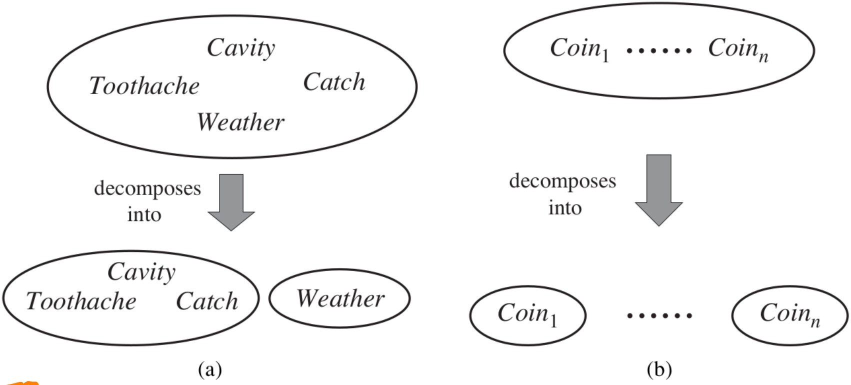
- Suppose we add a fourth R.V. : *Weather*.
- $P(Toothache, Catch, Cavity, Weather)$
- $P(toothache, catch, \neg cavity, cloudy) =$ $P(cloudy|toothache, catch, \neg cavity)P(toothache, catch, \neg cavity)$
- $P(cloudy|toothache, catch, \neg cavity) = P(cloudy)$

Independence

- Suppose we add a fourth R.V. : $Weather$.
- $P(Toothache, Catch, Cavity, Weather)$
- $P(toothache, catch, \neg cavity, cloudy) =$
  $P(cloudy|toothache, catch, \neg cavity)P(toothache, catch, \neg cavity)$
- $P(cloudy|toothache, catch, \neg cavity) = P(cloudy)$
- Independent random variables
  $P(X|Y) = P(X)$ or $P(Y|X) = P(Y)$ or
  $P(X, Y) = P(X)P(Y)$

Factoring the full joint distribution

- $P(Toothache, Catch, Cavity, Weather) = P(Toothache, Catch, Cavity)P(Weather)$



2^{10} =

10

Bayes' rule and its use

- $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)}$

Bayes' rule and its use

- $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)}$
- A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the prior probability that a patient has meningitis is 1/50,000. The prior probability that any patient has a stiff neck is 1%. What is the probability that the patient has meningitis if the patient has a stiff neck?

$P(s|m) = 70\%$

Bayes' rule and its use

- $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$
- A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the prior probability that a patient has meningitis is 1/50,000. The prior probability that any patient has a stiff neck is 1%. What is the probability that the patient has meningitis if the patient has a stiff neck?

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.7 \times 1/50000}{.01} = 0.0014$$

Bayes' rule and its use

- $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)}$
- A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the prior probability that a patient has meningitis is 1/50,000. The prior probability that any patient has a stiff neck is 1%. What is the probability that the patient has meningitis if the patient has a stiff neck?

  $$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.7 \times 1/50000}{.01} = 0.0014$$
- Notice that though $P(s|m)$ is high, $P(m|s)$ is small.

Bayes' rule and its use

- $P(Y|X) = \dfrac{P(X|Y)P(Y)}{P(X)}$
- A doctor knows that the disease meningitis causes the patient to have a stiff neck 70% of the time. The doctor also knows that the prior probability that a patient has meningitis is 1/50,000. The prior probability that any patient has a stiff neck is 1%. What is the probability that the patient has meningitis if the patient has a stiff neck?

  $$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0.7 \times 1/50000}{.01} = 0.0014$$

- Notice that though $P(s|m)$ is high, $P(m|s)$ is small.
- Useful in finding $P(cause|effect)$ e.g. $P(cavity|toothache)$

More general Bayes' rule

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

Conditional Independence

- Conditional Independence
  $P(X, Y|Z) = P(X|Z)P(Y|Z)$

Conditional Independence

- Conditional Independence
  $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- Conditional independence (like factoring) helps in reducing the size of the joint probability distribution table.
  $P(X, Y, Z) = P(X, Y|Z)P(Z) = P(X|Z)P(Y|Z)P(Z)$

Conditional Independence

- Conditional Independence

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

- Conditional independence (like factoring) helps in reducing the size of the joint probability distribution table.

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

- $P(Toothache, Catch, Cavity) =$
  $P(Toothache, Catch|Cavity)P(Cavity)$
  $= P(Toothache|Cavity)P(Catch|Cavity)P(Cavity)$

  P(t | c) P(ct | c) P(c)

  P(t | ¬c) P(ct | ¬c)

Conditional Independence

- Conditional Independence
  $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- Conditional independence (like factoring) helps in reducing the size of the joint probability distribution table.
  $P(X, Y, Z) = P(X, Y|Z)P(Z) = P(X|Z)P(Y|Z)P(Z)$
- $P(Toothache, Catch, Cavity) =$
  $P(Toothache, Catch|Cavity)P(Cavity)$
  $= P(Toothache|Cavity)P(Catch|Cavity)P(Cavity)$
- Size of KB is $O(n)$ instead of $O(2^n)$.

Conditional Independence

- Conditional Independence
  $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- Conditional independence (like factoring) helps in reducing the size of the joint probability distribution table.
  $P(X, Y, Z) = P(X, Y|Z)P(Z) = P(X|Z)P(Y|Z)P(Z)$
- $P(Toothache, Catch, Cavity) =$
  $P(Toothache, Catch|Cavity)P(Cavity)$
  $= P(Toothache|Cavity)P(Catch|Cavity)P(Cavity)$
- Size of KB is $O(n)$ instead of $O(2^n)$.

Q Which of the following is/are True?

a. $P(toothache|cavity) = 1 - P(\neg toothache|cavity)$
b. $P(toothache|cavity) = 1 - P(toothache|\neg cavity)$

$P(a|b) = 1 - P(\neg a|b)$

$P(a|\neg b) = 1$

$P(a|b) = 1 - P(a|\neg b)$, $P(a|b) = 1$

Naive Bayes model

- $P(Cause, Effect_1, \ldots, Effect_n) = P(Cause) \prod_i P(Effect_i | Cause)$

Naive Bayes model

- $P(Cause, Effect_1, \ldots, Effect_n) = P(Cause) \prod_i P(Effect_i | Cause)$
- How will we find $P(Cause | Effect1, \ldots, Effect_n)$?

Naive Bayes model

- $P(Cause, Effect_1, \ldots, Effect_n) = P(Cause) \prod_i P(Effect_i | Cause)$
- How will we find $P(Cause | Effect1, \ldots, Effect_n)$?
- $\dfrac{P(Cause, Effect_1, \ldots, Effect_n)}{P(Effect_1, \ldots, Effect_n)}$ ←

Naive Bayes model

- $P(Cause, Effect_1, \ldots, Effect_n) = P(Cause)\prod_i P(Effect_i|Cause)$
- How will we find $P(Cause|Effect1, \ldots, Effect_n)$?
- $\dfrac{P(Cause, Effect_1, \ldots, Effect_n)}{P(Effect_1, \ldots, Effect_n)}$
  $= \alpha P(Cause, Effect_1, \ldots, Effect_n)$

$= \alpha P(\neg Cause, Effect1, \ldots, Effectn)$

Main points.

- Full joint distribution table can act as a KB.

Main points.

- Full joint distribution table can act as a KB.
- Conditional independence assumption helps us in storing fewer values in the KB.

Main points.

- Full joint distribution table can act as a KB.
- Conditional independence assumption helps us in storing fewer values in the KB.
- Factoring the joint distribution has two advantages.

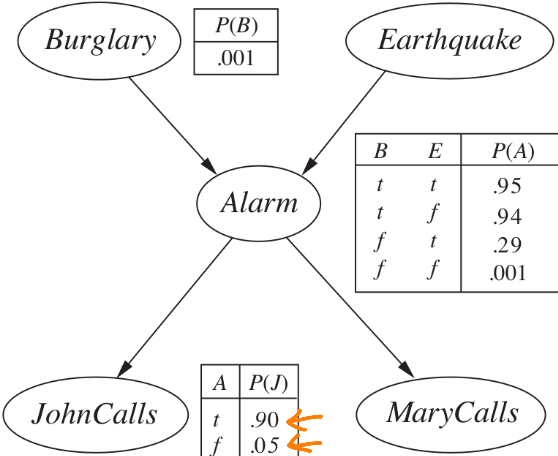
Main points.

- Full joint distribution table can act as a KB.
- Conditional independence assumption helps us in storing fewer values in the KB.
- Factoring the joint distribution has two advantages.
  - Prior probabilities can be easily updated.

Main points.

- Full joint distribution table can act as a KB.
- Conditional independence assumption helps us in storing fewer values in the KB.
- Factoring the joint distribution has two advantages.
  - Prior probabilities can be easily updated.
  - Helps in storing fewer values in the KB.

Bayesian Network



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

$2^5 = 32$

10

$P(a \mid b, e)$

$P(a \mid b, \neg e)$

$P(j \mid a)$ $P(j \mid \neg a)$

Bayesian Network

$$P(j|a) + P(j|\neg a) = 1\ ?$$

$$P(a|e, b) + P(a|e, \neg b) + P(a|\neg e, b) + P(a|\neg e, \neg b) = 1\ ?$$

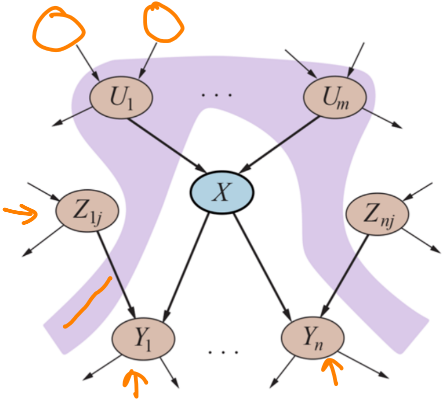
- If *parents*(X) is given then X is independent of any non-descendant random variable Y.

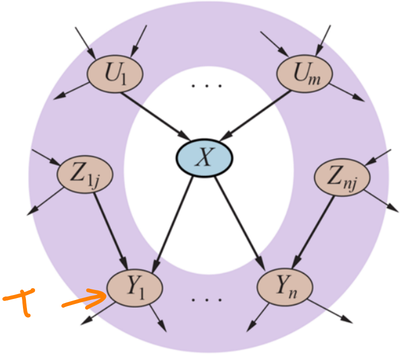
  $P(J|A) = P(J|A, Y)$

- If *parents*(X) is given then X is not independent of any descendant random variable Y.

  $P(A|B, E) = P(A|B, E, J)$?

Independence Properties

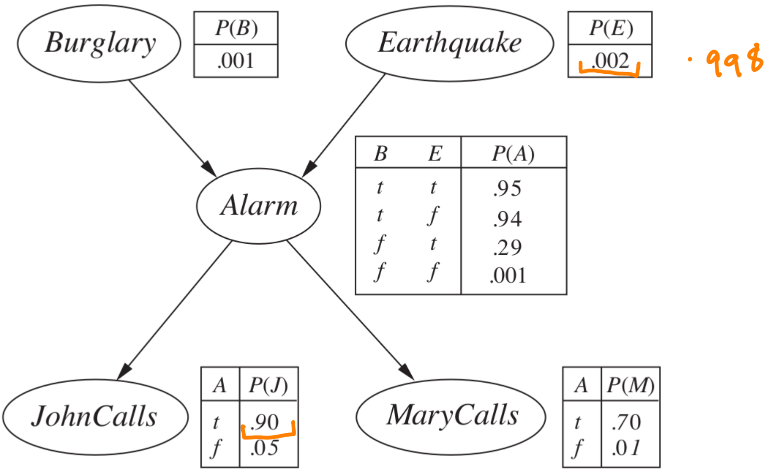




(a) Non-descendants property

(b) Markov blanket property

Bayesian Network



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

Joint Probability

- $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(x_i))$

Joint Probability

- $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(x_i))$
- $P(J, M, A, B, E) = P(J|A)P(M|A)P(A|B, E)P(B)P(E)$

Joint Probability

- $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(x_i))$
- $P(J, M, A, B, E) = P(J|A)P(M|A)P(A|B, E)P(B)P(E)$
- What is $P(j, m, a, \neg b, \neg e)$?

$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

$$= .9 \times .7 \times .001 \times .999 \times .998$$

$$= 0.000628$$

Joint Probability

- $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(x_i))$
- $P(J, M, A, B, E) = P(J|A)P(M|A)P(A|B, E)P(B)P(E)$
- What is $P(j, m, a, \neg b, \neg e)$?

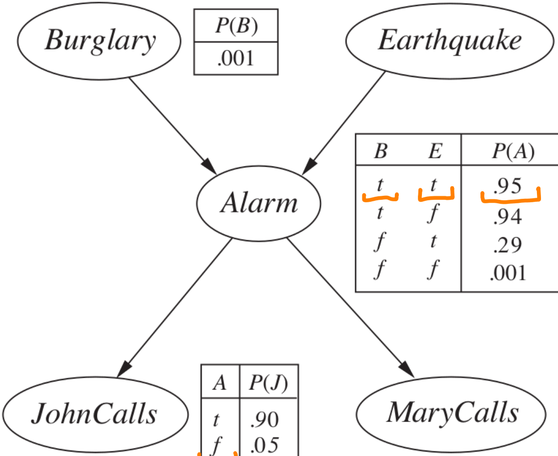
$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$

$$= .9 \times .7 \times .001 \times .999 \times .998$$

$$= 0.000628$$

- What is $P(b|j, m)$?

Bayesian Network



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

$P(\neg a \mid b, e)$

$p(a \mid b, e) = .95$

$P(\neg a \mid b, e) = 1 - .95$

$= .05$

$.05 \times .01 \times .05$

Inference

Find the probability $P(b|j, m)$?

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)} =$$

Inference

Find the probability $P(b|j, m)$?

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)} = \alpha P(b, j, m)$$

Inference

Find the probability $P(b|j, m)$?

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)} = \alpha P(b, j, m)$$

$$P(b, j, m) = \sum_a \sum_e P(b, j, m, a, e)$$

Inference

Find the probability $P(b|j, m)$?

$$P(b|j,m) = \frac{P(b,j,m)}{P(j,m)} = \alpha P(b,j,m)$$

$$P(b,j,m) = \sum_a \sum_e P(b,j,m,a,e)$$

$$= \sum_a \sum_e P(j|a)P(m|a)P(a|b,e)P(b)P(e)$$

Inference

Find the probability $P(b|j,m)$?

$$P(b|j,m) = \frac{P(b,j,m)}{P(j,m)} = \alpha P(b,j,m)$$

$$P(b,j,m) = \sum_a \sum_e P(b,j,m,a,e)$$

$$= \sum_a \sum_e P(j|a)P(m|a)P(a|b,e)P(b)P(e)$$

$$= P(b)\sum_e P(e)\sum_a P(j|a)P(m|a)P(a|b,e)$$

Find the probability $P(b|j, m)$?

$$P(b|j,m) = \frac{P(b,j,m)}{P(j,m)} = \alpha P(b,j,m)$$

$$P(b,j,m) = \sum_a \sum_e P(b,j,m,a,e)$$

$$= \sum_a \sum_e P(j|a)P(m|a)P(a|b,e)P(b)P(e)$$

$$= P(b) \sum_e P(e) \sum_a P(j|a)P(m|a)P(a|b,e)$$

$$= P(b)($$

$$P(e)[$$

$$P(j|a)P(m|a)P(a|b,e) + P(j|\neg a)P(m|\neg a)P(\neg a|b,e)]$$

$$+ P(\neg e)[$$

$$P(j|a)P(m|a)P(a|b,\neg e) + P(j|\neg a)P(m|\neg a)P(\neg a|b,\neg e)])$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) +$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) +$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$
$$P(b, j, m) = .000592243$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05)+$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$
$$P(b, j, m) = .000592243$$
$$P(b|j, m) = .000592243\alpha$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) +$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$P(b, j, m) = .000592243$

$P(b|j, m) = .000592243\alpha$

Similarly, $P(\neg b | j, m)$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) + .998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$$P(b, j, m) = .000592243$$

$$P(b|j, m) = .000592243\alpha$$

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) + \\ .998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$$P(b, j, m) = .000592243$$

$$P(b|j, m) = .000592243\alpha$$

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$

$$P(\neg b, j, m) = P(\neg b) \sum_e P(e) \sum_a P(j|a)P(m|a)P(a|\neg b, e)$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) +$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$$P(b, j, m) = .000592243$$

$$P(b|j, m) = .000592243\alpha$$

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$

$$P(\neg b, j, m) = P(\neg b) \sum_e P(e) \sum_a P(j|a)P(m|a)P(a|\neg b, e)$$

$$P(\neg b, j, m) = .0014919$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05)+$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$
$$P(b, j, m) = .000592243$$
$$P(b|j, m) = .000592243\alpha$$ — ①

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$
$$P(\neg b, j, m) = P(\neg b)\sum_e P(e)\sum_a P(j|a)P(m|a)P(a|\neg b, e)$$
$$P(\neg b, j, m) = .0014919$$
$$P(\neg b|j, m) = .0014919\alpha$$ — ②

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) + .998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$$P(b, j, m) = .000592243$$

$$P(b|j, m) = .000592243\alpha$$

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$

$$P(\neg b, j, m) = P(\neg b) \sum_e P(e) \sum_a P(j|a)P(m|a)P(a|\neg b, e)$$

$$P(\neg b, j, m) = .0014919$$

$$P(\neg b|j, m) = .0014919\alpha$$

$$\alpha = 479.81$$

Inference

$$= .001(.002(.90 \times .70 \times .95 + .05 \times .01 \times .05) +$$
$$.998(.90 \times .70 \times .94 + .05 \times .01 \times .06))$$

$$P(b, j, m) = .000592243$$

$$P(b|j, m) = .000592243\alpha$$

Similarly,

$$P(\neg b|j, m) = \frac{P(\neg b, j, m)}{P(j, m)} = \alpha P(\neg b, j, m)$$

$$P(\neg b, j, m) = P(\neg b) \sum_e P(e) \sum_a P(j|a)P(m|a)P(a|\neg b, e)$$

$$P(\neg b, j, m) = .0014919$$

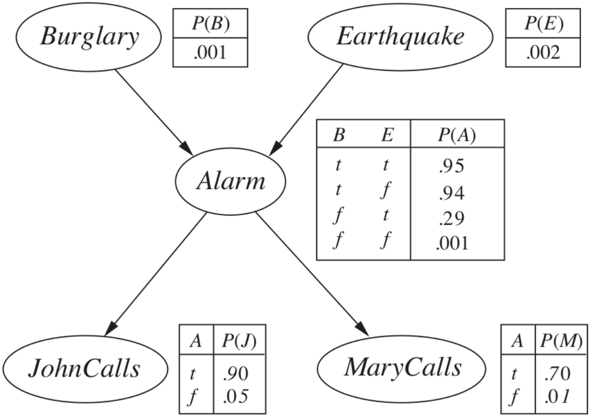
$$P(\neg b|j, m) = .0014919\alpha$$

$$\alpha = 479.81$$

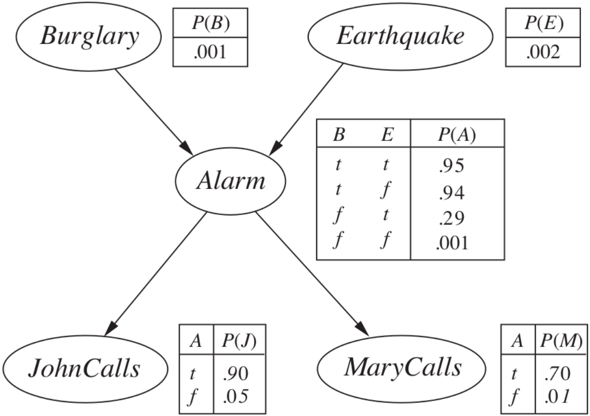
So, $P(b|j, m) = 0.284$

.001

Constructing a Bayesian network



Constructing a Bayesian network



- $P(X_i|parents(X_i))$

Constructing a Bayesian network

E B A J M
B E A M J

$$P(x_1, \ldots, x_n) = P(x_n | x_{n-1}, \ldots, x_1) P(x_{n-1}, \ldots, x_1)$$

$P(x_{n-1} | x_{n-2} \cdots x_1)$

Constructing a Bayesian network

$$P(x_1, \ldots, x_n) = P(x_n | x_{n-1}, \ldots, x_1) P(x_{n-1}, \ldots, x_1)$$

$$P(x_1, \ldots, x_n) = P(x_n | x_{n-1}, \ldots, x_1) P(x_{n-1} | x_{n-2}, \ldots, x_1) \cdots P(x_2 | x_1) P(x_1)$$

$$= \prod_{i=1}^{n} P(x_i | x_{i-1}, \ldots, x_1)$$

Constructing a Bayesian network

$$P(x_1,\ldots,x_n) = P(x_n|x_{n-1},\ldots,x_1)P(x_{n-1},\ldots,x_1)$$

$$\begin{aligned} P(x_1,\ldots,x_n) &= P(x_n|x_{n-1},\ldots,x_1)P(x_{n-1}|x_{n-2},\ldots,x_1)\cdots P(x_2|x_1)P(x_1) \\ &= \prod_{i=1}^{n} P(x_i|x_{i-1},\ldots,x_1) \end{aligned}$$

$$P(X_i|X_{i-1},\ldots,X_1) = P(X_i|Parent(X_i)) \text{ , where}$$

$$Parent(X_i) \subseteq \{X_{i-1},\ldots,X_1\}$$

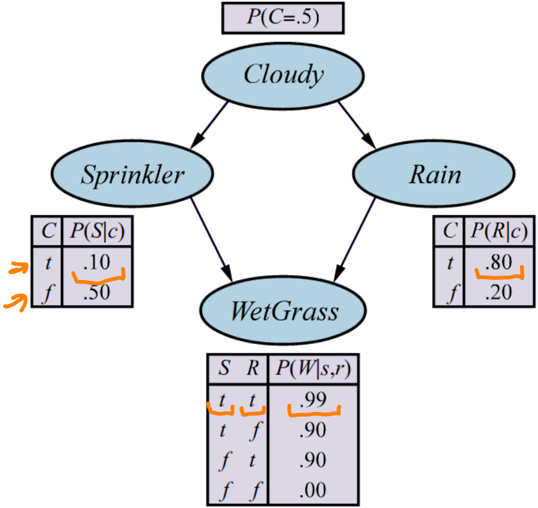
Constructing a Bayesian network

1. Order RVs such that causes precede effects.

Constructing a Bayesian network

1. Order RVs such that causes precede effects.
2. For i=1 to n do:
   - Find a minimal set of parents such that $Parent(X_i) \subseteq \{X_{i-1}, \ldots, X_1\}$.
   - For each parent add a directed edge from parent to X_i.
   - Write down the conditional probability table $P(X_i|Parent(X_i))$.

Practice problem



Practice problem

What is the probability that the Sprinkler is on if the grass is Wet (i.e. $P(s|w)$)?

Partial soln.:

$$P(s|w) = \frac{P(s \wedge w)}{P(w)} = \alpha P(s \wedge w)$$

Practice problem

What is the probability that the Sprinkler is on if the grass is Wet (i.e. $P(s|w)$)?

Partial soln.:

$$P(s|w) = \frac{P(s \wedge w)}{P(w)} = \alpha P(s \wedge w)$$

$$P(\neg s|w) = \alpha P(\neg s \wedge w)$$

Practice problem

What is the probability that the Sprinkler is on if the grass is Wet (i.e. $P(s|w)$)?

Partial soln.:

$$P(s|w) = \frac{P(s \wedge w)}{P(w)} = \alpha P(s \wedge w)$$

$$P(\neg s|w) = \alpha P(\neg s \wedge w)$$

$$P(s \wedge w) = .2781 \; , \; P(\neg s \wedge w) = .369$$

Practice problem

What is the probability that the Sprinkler is on if the grass is Wet (i.e. $P(s|w)$)?

Partial soln.:

$$P(s|w) = \frac{P(s \wedge w)}{P(w)} = \alpha P(s \wedge w)$$

$$P(\neg s|w) = \alpha P(\neg s \wedge w)$$

$$P(s \wedge w) = .2781 \;,\; P(\neg s \wedge w) = .369$$

$$\alpha = 1.5454$$

Practice problem

What is the probability that the Sprinkler is on if the grass is Wet (i.e. $P(s|w)$)?

Partial soln.:

$$P(s|w) = \frac{P(s \wedge w)}{P(w)} = \alpha P(s \wedge w)$$

$$P(\neg s|w) = \alpha P(\neg s \wedge w)$$

$$P(s \wedge w) = .2781 \;,\; P(\neg s \wedge w) = .369$$

$$\alpha = 1.5454$$

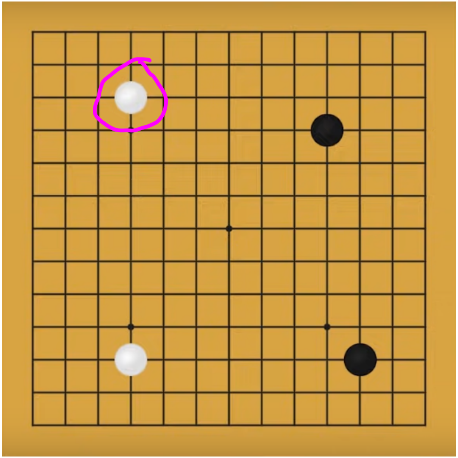
Ans. P(s|w) = 0.4298

Monte Carlo Tree Search

- Chapter 5, Russell and Norvig, 4th Edition

Monte Carlo Tree Search

- Chapter 5, Russell and Norvig, 4th Edition
- Game of Go



Monte Carlo Tree Search

- Heuristic alpha-beta search won't work well when:

Monte Carlo Tree Search

- Heuristic alpha-beta search won't work well when:

1. Branching factor is large

Monte Carlo Tree Search

- Heuristic alpha-beta search won't work well when:

1. Branching factor is large
2. Good evaluation function is not available

Monte Carlo Tree Search

- Heuristic alpha-beta search won't work well when:

1. Branching factor is large
2. Good evaluation function is not available

- AlphaGo : first computer Go program to beat a human professional Go player (October 2015).

Monte Carlo Tree Search

- Heuristic alpha-beta search won't work well when:

1. Branching factor is large
2. Good evaluation function is not available

- AlphaGo : first computer Go program to beat a human professional Go player (October 2015).
- AlphaGo used Monte Carlo Tree Search and Deep Neural Network.

Monte Carlo Tree Search

- A special case of Monte Carlo method in reinforcement learning: value of each state is updated at the end of an episode.

Monte Carlo Tree Search

- A special case of Monte Carlo method in reinforcement learning: value of each state is updated at the end of an episode.
- No heuristic evaluation function

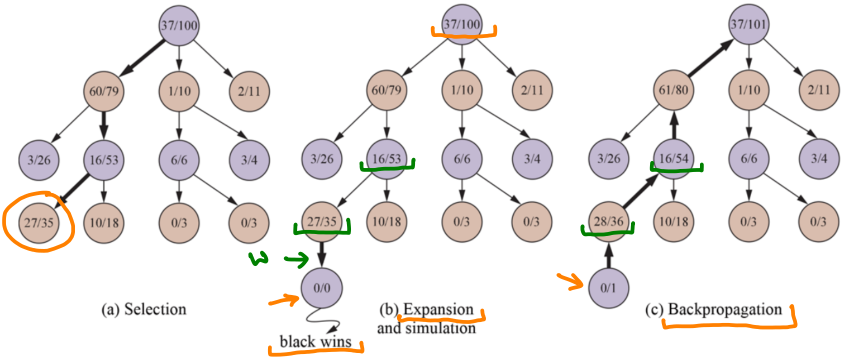
Monte Carlo Tree Search

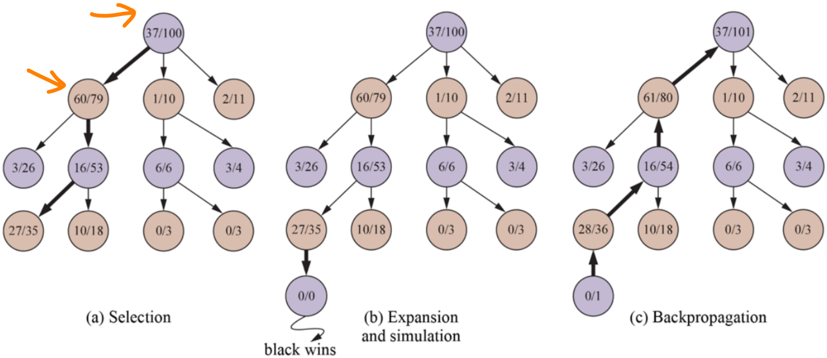
- A special case of Monte Carlo method in reinforcement learning: value of each state is updated at the end of an episode.
- No heuristic evaluation function
- We use simulations (rollout/playout) of the game. (similar to Episodes)

Monte Carlo Tree Search

- A special case of Monte Carlo method in reinforcement learning: value of each state is updated at the end of an episode.
- No heuristic evaluation function
- We use simulations (rollout/playout) of the game. (similar to Episodes)
- We need an action selection policy that balances exploration and exploitation.

Monte Carlo Tree Search





The above four steps are repeated for a set number of iterations, or until the allotted time has expired.

Selection policy at each node

- Upper-Confidence-Bound Action Selection

Selection policy at each node

- Upper-Confidence-Bound Action Selection
- Give more preference to actions whose values are uncertain

$$A_t \doteq \underset{a}{\arg\max} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

$N_t(a) > kt$, $k = .001$

$\frac{\ln t}{N_t(a)} < \frac{\ln t}{kt}$

$\lim_{t \to \infty} \frac{\ln t}{kt} = \frac{1/t}{k} = \frac{1}{kt}$

$= 0$

Selection policy at each node

- Upper-Confidence-Bound Action Selection
- Give more preference to actions whose values are uncertain

$$A_t \doteq \underset{a}{\arg\max} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right]$$

where $c > 0$ controls the degree of exploration

Selection policy : UCT

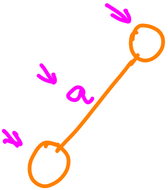
- Upper-Confidence-Bound applied to trees (UCT)

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

1.4 1.5

N

$\sqrt{2}$



$\frac{36}{45} + C \times$

Selection policy : UCT

- Upper-Confidence-Bound applied to trees (UCT)

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

The parameter C is usually set to be between 1 and 2.

Monte Carlo Tree Search Algo.

```
function MONTE-CARLO-TREE-SEARCH(state) returns an action
  tree ← NODE(state)
  while IS-TIME-REMAINING() do
    leaf ← SELECT(tree)
    child ← EXPAND(leaf)
    result ← SIMULATE(child)
    BACK-PROPAGATE(result, child)
  return the move in ACTIONS(state) whose node has highest number of playouts
```

Monte Carlo Tree Search Algo.

function MONTE-CARLO-TREE-SEARCH(*state*) **returns** *an action*
  tree ← NODE(*state*)
  while IS-TIME-REMAINING() **do**
    leaf ← SELECT(*tree*)
    child ← EXPAND(*leaf*)
    result ← SIMULATE(*child*)
    BACK-PROPAGATE(*result*, *child*)
  return the move in ACTIONS(*state*) whose node has highest number of playouts

- We want to prefer node with total utility $= \frac{65}{100}$ over a node with total utility $= \frac{2}{3}$

Monte Carlo Tree Search Algo.

```
function MONTE-CARLO-TREE-SEARCH(state) returns an action
  tree ← NODE(state)
  while IS-TIME-REMAINING() do
    leaf ← SELECT(tree)
    child ← EXPAND(leaf)
    result ← SIMULATE(child)
    BACK-PROPAGATE(result, child)
  return the move in ACTIONS(state) whose node has highest number of playouts
```

- We want to prefer node with total utility $= \frac{65}{100}$ over a node with total utility $= \frac{2}{3}$
- Due to UCT selection policy, the node with the highest playout very often has a high total utility.

Comparison between MCTS and Alpha-beta

- Time to compute a playout is linear in maximum depth of the game tree.

Comparison between MCTS and Alpha-beta

- Time to compute a playout is linear in maximum depth of the game tree.
- This allows us to have plenty of playouts before deciding an action

Comparison between MCTS and Alpha-beta

- Time to compute a playout is linear in maximum depth of the game tree.
- This allows us to have plenty of playouts before deciding an action
- If we have a good evaluation function, then alpha-beta search may do better.

Comparison between MCTS and Alpha-beta

- Time to compute a playout is linear in maximum depth of the game tree.
- This allows us to have plenty of playouts before deciding an action
- If we have a good evaluation function, then alpha-beta search may do better.
- Otherwise, MCTS algorithm might be a better option where millions of playouts can be tried before making a move.

Using Neural Network

- With MCTS

Using Neural Network

- With MCTS
- With Q-learning

Bayesian Networks

- Represents the joint probabilities by making use of Cause-effect relations and conditional independence.

Bayesian Networks

- Represents the joint probabilities by making use of Cause-effect relations and conditional independence.
- Inferencing using Bayesian Networks.

Chapter 14: Probabilistic Reasoning over Time

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)

Chapter 14: Probabilistic Reasoning over Time

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)

- Hidden Markov models

Chapter 14: Probabilistic Reasoning over Time

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)
- Hidden Markov models
- Some applications:

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)
- Hidden Markov models
- Some applications:

1. Speech recognition

Chapter 14: Probabilistic Reasoning over Time

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)

- Hidden Markov models

- Some applications:

1. Speech recognition
2. Handwriting recognition

Chapter 14: Probabilistic Reasoning over Time

- Chapter 14: Probabilistic Reasoning over Time (Russell and Norvig, 4th edition)
- Hidden Markov models
- Some applications:

1. Speech recognition
2. Handwriting recognition
3. Gene annotation and sequence alignment in Bioinformatics

Time and Uncertanity

- Speech to text translation :

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers
- A variable for each time slice

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers
- A variable for each time slice
- Simpler example: You are a security guard stationed at a secret underground installation

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers
- A variable for each time slice
- Simpler example: You are a security guard stationed at a secret underground installation
- Predict whether it's raining today based on director coming with, or without, an umbrella.

Time and Uncertainty

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers
- A variable for each time slice
- Simpler example: You are a security guard stationed at a secret underground installation
- Predict whether it's raining today based on director coming with, or without, an umbrella.
  1. (hidden) state variables ($\mathbf{X_t}$) : raining

Time and Uncertanity

- Speech to text translation :
  1. (hidden) state variables : syllables
  2. (observable) evidence variables : sounds
     (Evidence variables are affected by accent, pitch, volume, background noise etc.)
- Discrete-time models : time slice denoted by integers
- A variable for each time slice
- Simpler example: You are a security guard stationed at a secret underground installation
- Predict whether it's raining today based on director coming with, or without, an umbrella.
  1. (hidden) state variables ($\mathbf{X_t}$) : raining
  2. (observable) evidence variables ($\mathbf{E_t}$) : umbrella

Time and Uncertanity

- We will assume that state sequence starts at $t = 0$, and evidence sequence starts at $t = 1$

Time and Uncertanity

- We will assume that state sequence starts at $t = 0$, and evidence sequence starts at $t = 1$
- State variables : $R_0, R_1, R_2, \ldots$

Time and Uncertanity

- We will assume that state sequence starts at $t = 0$, and evidence sequence starts at $t = 1$
- State variables : $R_0, R_1, R_2, \ldots$
- Evidence variables : $U_1, U_2, U_3, \ldots$

Time and Uncertanity

- We will assume that state sequence starts at $t = 0$, and evidence sequence starts at $t = 1$
- State variables : $R_0, R_1, R_2, \ldots$
- Evidence variables : $U_1, U_2, U_3, \ldots$
- Notation : $U_{1:3}$ denotes U_1, U_2, U_3

Transition model

- Transition model: how the world evolves?

Transition model

- Transition model: how the world evolves?
- Transition model: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}})$

P(R_t |

Transition model

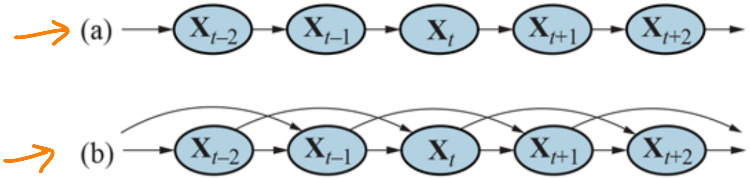
- Transition model: how the world evolves?
- Transition model: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}})$
- Assumption: Transition model is a first-order Markov process:

Transition model

- Transition model: how the world evolves?
- Transition model: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}})$
- Assumption: Transition model is a first-order Markov process: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$

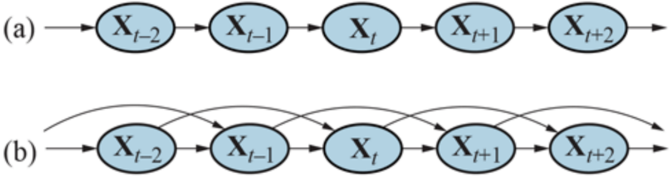
Transition model

- Transition model: how the world evolves?
- Transition model: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}})$
- Assumption: Transition model is a first-order Markov process:
  $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$
- What is second-order Markov process?



Transition model

- Transition model: how the world evolves?
- Transition model: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}})$
- Assumption: Transition model is a first-order Markov process: $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$
- What is second-order Markov process?



- Markov assumption: present state depends on only a finite fixed number of previous states.

Transition model

- $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$

Transition model

- $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$
- A different distribution for every time step?

Transition model

- $\mathbf{P(X_t|X_{0:t-1}) = P(X_t|X_{t-1})}$
- A different distribution for every time step?
- Time-homogeneous process: Process of state change is governed by laws that do not themselves change over time

Transition model

- $\mathbf{P}(\mathbf{X_t}|\mathbf{X_{0:t-1}}) = \mathbf{P}(\mathbf{X_t}|\mathbf{X_{t-1}})$
- A different distribution for every time step?
- Time-homogeneous process: Process of state change is governed by laws that do not themselves change over time
- Probability distribution for the transition model remains the same across time steps:

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

Sensor (observation) models

- Sensor model: how the evidence variables get their value?

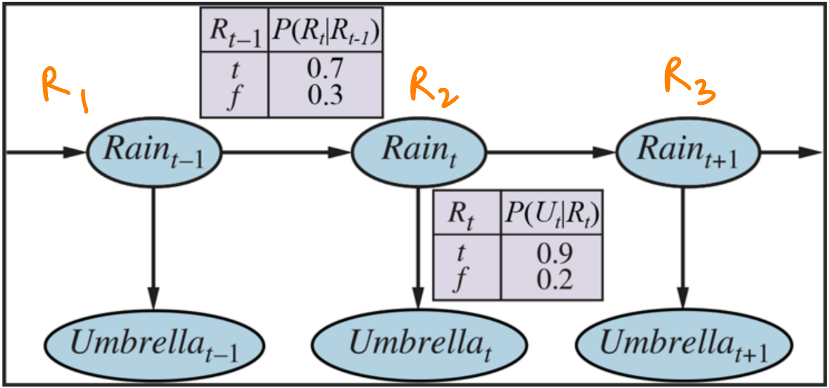
Sensor (observation) models

- Sensor model: how the evidence variables get their value?
- Sensor model: $\mathbf{P}(\mathbf{E_t}|\mathbf{X_{0:t}}, \mathbf{E_{1:t-1}})$

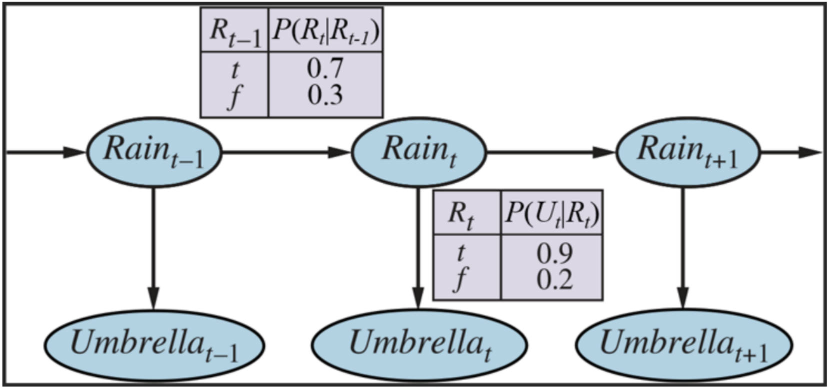
Sensor (observation) models

- Sensor model: how the evidence variables get their value?
- Sensor model: $\mathbf{P}(\mathbf{E_t}|\mathbf{X_{0:t}}, \mathbf{E_{1:t-1}})$
- Sensor Markov assumption: $\mathbf{P}(\mathbf{E_t}|\mathbf{X_{0:t}}, \mathbf{E_{1:t-1}}) = \mathbf{P}(\mathbf{E_t}|\mathbf{X_t})$

Bayesian network for transition and sensor models



Bayesian network for transition and sensor models

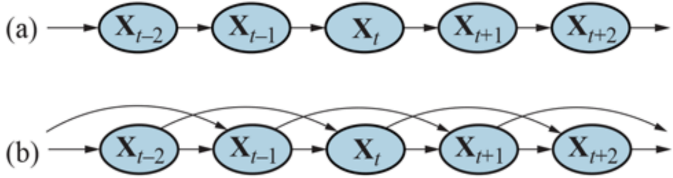


- Joint distribution:

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

Improving the accuracy of the Markov process

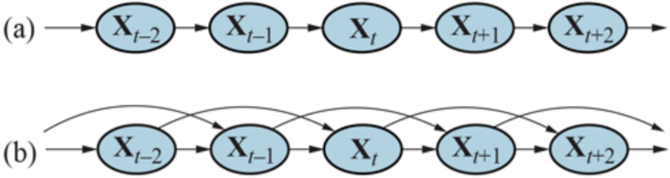
- Increasing the order of the Markov process model



$N_t = (T, F)$

Improving the accuracy of the Markov process

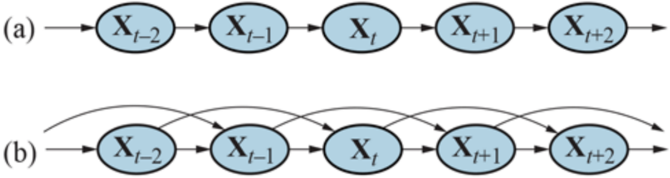
- Increasing the order of the Markov process model



- Add new state variables and sensor variables

Improving the accuracy of the Markov process

- Increasing the order of the Markov process model

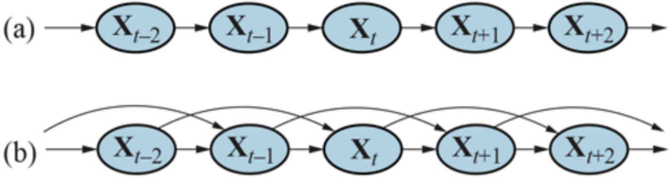


- Add new state variables and sensor variables

  e.g $XTemperature_t$, $XHumidity_t$, $ETemperature_t$, $EHumidity_t$

Improving the accuracy of the Markov process

- Increasing the order of the Markov process model

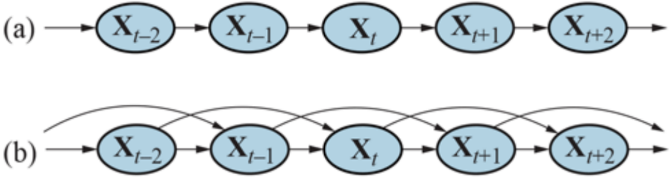


- Add new state variables and sensor variables

  e.g $XTemperature_t$, $XHumidity_t$, $ETemperature_t$, $EHumidity_t$
- The state variables should be able to predict the evidence (sensor) variables.

Improving the accuracy of the Markov process

- Increasing the order of the Markov process model



- Add new state variables and sensor variables

  e.g $XTemperature_t$, $XHumidity_t$, $ETemperature_t$, $EHumidity_t$
- The state variables should be able to predict the evidence (sensor) variables.
- The designer must have some understanding the "physics" (rules) underlying the process being modeled.

Inference in temporal models

- Filtering or state estimation (computing the belief state): $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$

Inference in temporal models

- Filtering or state estimation (computing the belief state):
  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$
- Prediction
  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$

R 4

Inference in temporal models

- Filtering or state estimation (computing the belief state):

  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$
- Prediction

  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$
- Smoothing

  $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

Inference in temporal models

- Filtering or state estimation (computing the belief state):
  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$
- Prediction
  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$
- Smoothing
  $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$
- Most likely explanation
  $\arg\max_{\mathbf{x_{1:t}}} \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})$

$P(E_t | X_t)$

$P(X_t | X_{t-1}) \ldots P(X_0)$

$P(X_{0:t} | E_{1:t})$

Inference in temporal models

- Filtering or state estimation (computing the belief state):
  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$
- Prediction
  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$
- Smoothing
  $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$
- Most likely explanation
  $\arg\max_{\mathbf{x_{1:t}}} \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})$
- Learning

More general Bayes' rule

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

More general Bayes' rule

- $P(Y|X,e) = \dfrac{P(X|Y,e)P(Y|e)}{P(X|e)}$
- $P(Y|X,e) = \alpha P(X|Y,e)P(Y|e)$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}} | \mathbf{e_{1:t+1}})$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

$P(e_{t+1}|X_{t+1})$

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}, \mathbf{e_{t+1}})$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

$$\begin{aligned}\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) &= \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\ &= \alpha\mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}})\end{aligned}$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

$$\begin{aligned}
\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) &= \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\
&= \alpha \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \\
&= \alpha \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \qquad (1)
\end{aligned}$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

$$\begin{aligned}\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) &= \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\ &= \alpha\mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \\ &= \alpha\mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \qquad (1)\end{aligned}$$

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) = \sum_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t})\mathbf{P}(\mathbf{x_t}|\mathbf{e_{1:t}}) \qquad (2)$$

Filtering (State estimation) and Prediction

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \mathbf{f}(\mathbf{e_{t+1}}, \mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}}))$$

$$\begin{aligned}
\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) &= \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\
&= \alpha \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \\
&= \alpha \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}})\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) \qquad (1)
\end{aligned}$$

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t}}) = \sum_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t})\mathbf{P}(\mathbf{x_t}|\mathbf{e_{1:t}}) \qquad (2)$$

$$\mathbf{P}(\mathbf{X_{t+1}}|\mathbf{e_{1:t+1}}) = \alpha \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \sum_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t})\mathbf{P}(\mathbf{x_t}|\mathbf{e_{1:t}}) \qquad (3)$$

Filtering (State estimation) and Prediction

$$\mathbf{f}_{1:t+1} = \mathrm{FORWARD}(\mathbf{f}_{1:t}, \mathbf{e}_{t+1}) \tag{4}$$

Filtering (State estimation) and Prediction

$$\mathbf{f_{1:t+1}} = \text{FORWARD}(\mathbf{f_{1:t}}, \mathbf{e_{t+1}}) \quad (4)$$

- For each update, the time and space requirements is a constant.

Filtering (State estimation) and Prediction

$$\mathbf{f_{1:t+1}} = \text{FORWARD}(\mathbf{f_{1:t}}, \mathbf{e_{t+1}}) \tag{4}$$

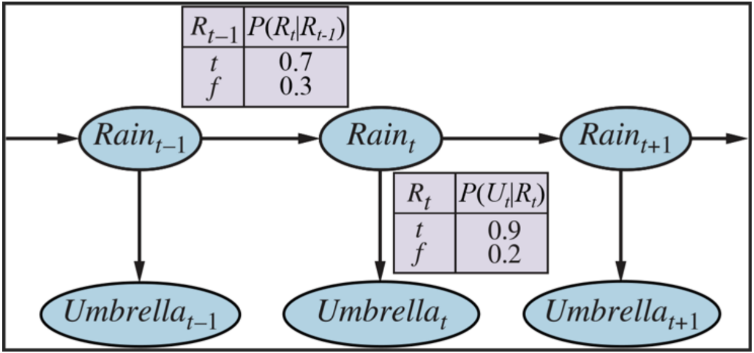
- For each update, the time and space requirements is a constant.
- This helps a finite agent keep track of current state estimate distribution indefinitely.

Filtering (State estimation) and Prediction

$$\mathbf{f}_{\mathbf{1:t+1}} = \mathrm{FORWARD}(\mathbf{f}_{\mathbf{1:t}}, \mathbf{e}_{\mathbf{t+1}}) \qquad (4)$$

- For each update, the time and space requirements is a constant.
- This helps a finite agent keep track of current state estimate distribution indefinitely.
- Eqn. (2) gives one step prediction.

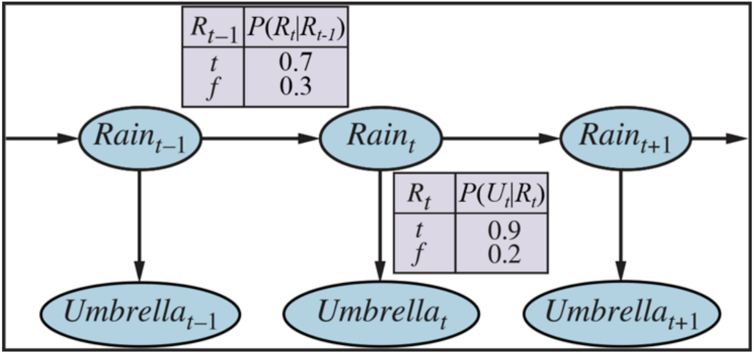
Filtering process for two steps



T F

$$\mathbf{P}(\mathbf{R_0}) = < .5, .5 >$$

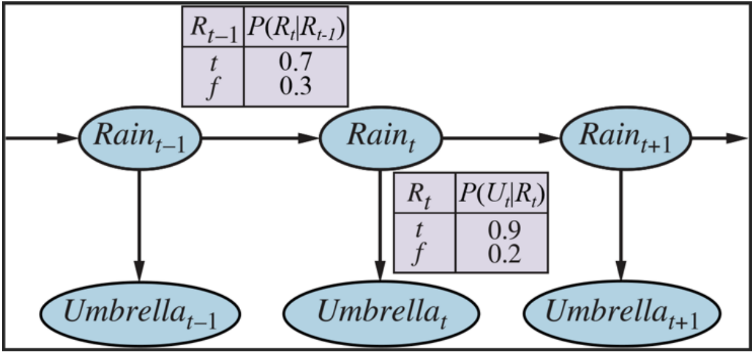
Filtering process for two steps



$\mathbf{P}(\mathbf{R_0}) = < .5, .5 >$

$\mathbf{P}(\mathbf{R_1})$

Filtering process for two steps

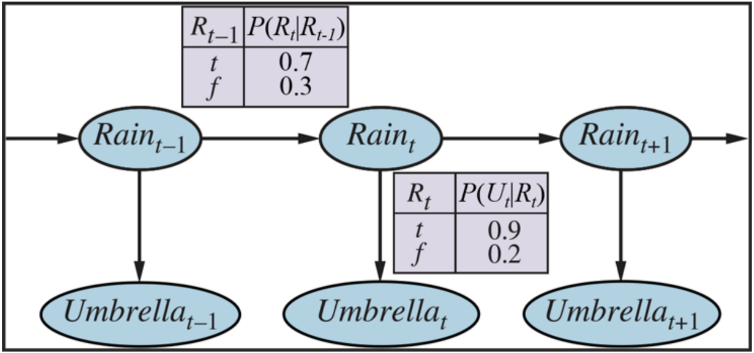


$$\mathbf{P}(\mathbf{R_0}) = < .5, .5 >$$

$$\mathbf{P}(\mathbf{R_1}) = \mathbf{P}(\mathbf{R_1}|\mathbf{R_0})\mathbf{P}(\mathbf{R_0})$$

.5 <0.7 .3> + .5 <.3 .7>

Filtering process for two steps



$$\mathbf{P}(\mathbf{R_0}) = < .5, .5 >$$

$$\begin{aligned}\mathbf{P}(\mathbf{R_1}) &= \mathbf{P}(\mathbf{R_1}|\mathbf{R_0})\mathbf{P}(\mathbf{R_0}) \\ &= .5 < .7, .3 > + .5 < .3, .7 > \\ &= < .5, .5 >\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >,$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$P(R_1 \mid U_1)$

$$\mathbf{P(R_1)} = < .5, .5 >, \mathbf{U_1} = \textbf{True}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P(R_1)} = <.5, .5>, \mathbf{U_1} = \textbf{True}$$

$$\mathbf{P(R_1 \mid u_1)} =$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >, \mathbf{U_1} = \mathbf{True}$$

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1})$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >, \mathbf{U_1} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1})$$

$$= \alpha < .9, .2 >< .5, .5 >$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >, \mathbf{U_1} = \mathbf{True}$$

$$\begin{aligned} \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 > \end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = <.5, .5>, \mathbf{U_1} = \textbf{True}$$

$$\begin{aligned} \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 > \end{aligned}$$

$R_1 = T$

What is $\mathbf{P}(\mathbf{r_1} \mid \mathbf{u_1})$? $.45\alpha$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) =< .5, .5 >, \mathbf{U_1} = \mathbf{True}$$

$$\begin{aligned} \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 > \end{aligned}$$

What is $\mathbf{P}(\mathbf{r_1} \mid \mathbf{u_1})$? What is $\mathbf{P}(\neg \mathbf{r_1} \mid \mathbf{u_1})$? $.1\alpha$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >, \mathbf{U_1} = \textbf{True}$$

$$\begin{aligned} \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 > \end{aligned}$$

What is $\mathbf{P}(\mathbf{r_1} \mid \mathbf{u_1})$? What is $\mathbf{P}(\neg \mathbf{r_1} \mid \mathbf{u_1})$?

$\alpha \approx 1.8182$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P(R_1)} = < .5, .5 >, \mathbf{U_1} = \textbf{True}$$

$$\mathbf{P(R_1|u_1)} = \alpha \mathbf{P(u_1|R_1)P(R_1)}$$
$$= \alpha < .9, .2 >< .5, .5 >$$
$$= \alpha < .45, .10 >$$

What is $\mathbf{P(r_1|u_1)}$? What is $\mathbf{P(\neg r_1|u_1)}$?

$$\alpha \approx 1.8182$$
$$\mathbf{P(R_1|u_1)} \approx < .8182, .1818 >$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = < .5, .5 >, \mathbf{U_1} = \textbf{True}$$

$$\begin{aligned}\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1} \mid \mathbf{R_1}) \mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 >\end{aligned}$$

What is $\mathbf{P}(\mathbf{r_1} \mid \mathbf{u_1})$? What is $\mathbf{P}(\neg \mathbf{r_1} \mid \mathbf{u_1})$?

$$\alpha \approx 1.8182$$

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) \approx < .8182, .1818 > \quad (\mathbf{f_{1:1}})$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}) = <.5, .5>, \mathbf{U_1} = \textbf{True}$$

$$\begin{aligned}\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) &= \alpha \mathbf{P}(\mathbf{u_1}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}) \\ &= \alpha < .9, .2 >< .5, .5 > \\ &= \alpha < .45, .10 >\end{aligned}$$

What is $\mathbf{P}(\mathbf{r_1}|\mathbf{u_1})$? What is $\mathbf{P}(\neg \mathbf{r_1}|\mathbf{u_1})$?

$$\alpha \approx 1.8182$$

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \approx < .8182, .1818 > \quad (\mathbf{f_{1:1}})$$

$$\mathbf{f_{1:2}} = \text{FORWARD}(\mathbf{f_{1:1}}, \mathbf{e_2})$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = < .8182, .1818 >,$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P(R_1|u_1)} = < .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

Filtering process for two steps

P(R2|R1)

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = <.8182, .1818>, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2} \mid \mathbf{u_1}) =$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P(R_1|u_1)} = < .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P(R_2|u_1)} = \mathbf{P(R_2|R_1)P(R_1|u_1)}$$

$$.8182 < .7 \quad .3 > + .1818 < .3 \quad .7 >$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P(R_1|u_1)} =< .8182, .1818 >, \mathbf{U_2} = \mathbf{True}$$

$$\mathbf{P(R_2|u_1)} = \mathbf{P(R_2|R_1)P(R_1|u_1)}$$

$$= .8182 < .7, .3 > + .1818 < .3, .7 >$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = < .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}\mathbf{P}(\mathbf{R_2} \mid \mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2} \mid \mathbf{R_1})\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) \\ &= .8182 < .7, .3 > + .1818 < .3, .7 > \\ &= < .6273, .3727 >\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = \langle .8182, .1818 \rangle, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2} \mid \mathbf{u_1}) = \mathbf{P}(\mathbf{R_2} \mid \mathbf{R_1})\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1})$$

$$= .8182 \langle .7, .3 \rangle + .1818 \langle .3, .7 \rangle$$

$$= \langle .6273, .3727 \rangle \quad \text{(One step prediction)}$$

$P(R_3 \mid u_1) = P(R_3 \mid R_2) P(R_2 \mid u_1)$

$t+k+1$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) = \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1})$$

$$= .8182 < .7, .3 > + .1818 < .3, .7 >$$

$$=< .6273, .3727 > \text{ (One step prediction)}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) =$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) = < .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2} \mid \mathbf{u_1}) = \mathbf{P}(\mathbf{R_2} \mid \mathbf{R_1})\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1})$$

$$= .8182 < .7, .3 > + .1818 < .3, .7 >$$

$$= < .6273, .3727 > \text{ (One step prediction)}$$

$$\mathbf{P}(\mathbf{R_2} \mid \mathbf{u_{1:2}}) = \mathbf{P}(\mathbf{R_2} \mid \mathbf{u_2}, \mathbf{u_1})$$

$P(U_2 \mid R_2)$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) = < .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) = \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1})$$

$$= .8182 < .7, .3 > + .1818 < .3, .7 >$$

$$= < .6273, .3727 > \text{ (One step prediction)}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) = \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) =$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\ &= .8182 < .7, .3 > + .1818 < .3, .7 > \\ &=< .6273, .3727 > \quad \text{(One step prediction)}\end{aligned}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) = \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1})$$

Filtering process for two steps

.5

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$
\begin{aligned}
\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) &= < .8182, .1818 >, \mathbf{U_2} = \mathbf{True} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > + .1818 < .3, .7 > \\
&= < .6273, .3727 > \text{(One step prediction)} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1})
\end{aligned}
$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > + .1818 < .3, .7 > \\
&=< .6273, .3727 > \quad \text{(One step prediction)} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha\mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha\mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha < .9, .2 >< .6273, .3727 >
\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) = \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1})$$

$$= .8182 < .7, .3 > + .1818 < .3, .7 >$$

$$=< .6273, .3727 > \quad \text{(One step prediction)}$$

$$\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) = \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1})$$

$$= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1})$$

$$= \alpha < .9, .2 >< .6273, .3727 >$$

$$= \alpha < .5646, .0745 >$$

$P(r_2|U_{1:2})$

$P(\neg r_2|U_{1:2})$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) = < .8182, .1818 >, \mathbf{U_2} = \mathbf{True}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > + .1818 < .3, .7 > \\
&= < .6273, .3727 > \quad \text{(One step prediction)}
\end{aligned}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha < .9, .2 >< .6273, .3727 > \\
&= \alpha < .5646, .0745 > \quad , \alpha \approx 1.5647
\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > +.1818 < .3, .7 > \\
&=< .6273, .3727 > \text{ (One step prediction)} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha < .9, .2 >< .6273, .3727 > \\
&= \alpha < .5646, .0745 > \quad , \alpha \approx 1.5647 \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &\approx < .8834, .1166 >
\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > + .1818 < .3, .7 > \\
&=< .6273, .3727 > \quad \text{(One step prediction)} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha < .9, .2 >< .6273, .3727 > \\
&= \alpha < .5646, .0745 > \quad , \alpha \approx 1.5647 \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &\approx< .8834, .1166 > \quad (\mathbf{f_{1:2}})
\end{aligned}$$

Filtering process for two steps

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

$$\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) =< .8182, .1818 >, \mathbf{U_2} = \textbf{True}$$

$$\begin{aligned}
\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{R_1})\mathbf{P}(\mathbf{R_1}|\mathbf{u_1}) \\
&= .8182 < .7, .3 > + .1818 < .3, .7 > \\
&=< .6273, .3727 > \quad \text{(One step prediction)} \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &= \mathbf{P}(\mathbf{R_2}|\mathbf{u_2}, \mathbf{u_1}) = \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2}, \mathbf{u_1})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha \mathbf{P}(\mathbf{u_2}|\mathbf{R_2})\mathbf{P}(\mathbf{R_2}|\mathbf{u_1}) \\
&= \alpha < .9, .2 >< .6273, .3727 > \\
&= \alpha < .5646, .0745 > \quad , \alpha \approx 1.5647 \\
\mathbf{P}(\mathbf{R_2}|\mathbf{u_{1:2}}) &\approx< .8834, .1166 > \quad (\mathbf{f_{1:2}}) \\
\mathbf{f_{1:3}} &= \text{FORWARD}(\mathbf{f_{1:2}}, \mathbf{e_3})
\end{aligned}$$

Filtering process for two steps

- The probability that it rained has gone up after observing the evidence variable for two days.

Filtering process for two steps

- The probability that it rained has gone up after observing the evidence variable for two days.
- We can repeat the one step prediction procedure to predict the probability of rain on a future day.

Filtering process for two steps

- The probability that it rained has gone up after observing the evidence variable for two days.
- We can repeat the one step prediction procedure to predict the probability of rain on a future day.
- Prediction:

$$\mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{e_{1:t}}) = \sum_{\mathbf{x_{t+k}}} \mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{x_{t+k}})\mathbf{P}(\mathbf{x_{t+k}}|\mathbf{e_{1:t}})$$

K > 0

Filtering process for two steps

- The probability that it rained has gone up after observing the evidence variable for two days.
- We can repeat the one step prediction procedure to predict the probability of rain on a future day.
- Prediction:
$$\mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{e_{1:t}}) = \sum_{\mathbf{x_{t+k}}} \mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{x_{t+k}})\mathbf{P}(\mathbf{x_{t+k}}|\mathbf{e_{1:t}})$$
- Predicting further and further into the future leads to **stationary distribution** of the Markov process defined by the transition model.

Filtering process for two steps

- The probability that it rained has gone up after observing the evidence variable for two days.
- We can repeat the one step prediction procedure to predict the probability of rain on a future day.
- Prediction:
  $$\mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{e_{1:t}}) = \sum_{\mathbf{x_{t+k}}} \mathbf{P}(\mathbf{X_{t+k+1}}|\mathbf{x_{t+k}})\mathbf{P}(\mathbf{x_{t+k}}|\mathbf{e_{1:t}})$$
- Predicting further and further into the future leads to **stationary distribution** of the Markov process defined by the transition model.
- The stationary distribution is $< .5, .5 >$ for the Rain-umbrella model.

Filtering process for two steps

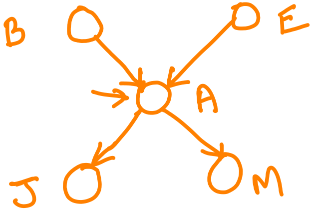
- The probability that it rained has gone up after observing the evidence variable for two days.
- We can repeat the one step prediction procedure to predict the probability of rain on a future day.
- Prediction:
  $$\mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{e}_{1:t}) = \sum_{\mathbf{x}_{t+k}} \mathbf{P}(\mathbf{X}_{t+k+1}|\mathbf{x}_{t+k})\mathbf{P}(\mathbf{x}_{t+k}|\mathbf{e}_{1:t})$$
- Predicting further and further into the future leads to **stationary distribution** of the Markov process defined by the transition model.
- The stationary distribution is $< .5, .5 >$ for the Rain-umbrella model.
- **Mixing time** is the time taken to reach the stationary distribution

Smoothing

$$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}), \text{ for } 0 \leq k < t$$

More general Bayes' rule

- $$P(Y|X,e) = \frac{P(X|Y,e)P(Y|e)}{P(X|e)}$$
- $$P(Y|X,e) = \alpha P(X|Y,e)P(Y|e)$$



$$P(J) \neq P(J|M)$$

$$P(J|A) = P(J|A,M)$$

Inference in temporal models

- Filtering or state estimation (computing the belief state): $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$

Inference in temporal models

- Filering or state estimation (computing the belief state):

  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$

  $\mathbf{f_{1:t+1}} = \mathrm{FORWARD}(\mathbf{f_{1:t}}, \mathbf{e_{t+1}})$

Inference in temporal models

- Filtering or state estimation (computing the belief state):

  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$

  $\mathbf{f_{1:t+1}} = \mathrm{FORWARD}(\mathbf{f_{1:t}}, \mathbf{e_{t+1}})$
- Prediction

  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$

Inference in temporal models

- Filtering or state estimation (computing the belief state):
  $\mathbf{P}(\mathbf{X_t}|\mathbf{e_{1:t}})$
  $\mathbf{f_{1:t+1}} = \mathrm{FORWARD}(\mathbf{f_{1:t}}, \mathbf{e_{t+1}})$
- Prediction
  $\mathbf{P}(\mathbf{X_{t+k}}|\mathbf{e_{1:t}})$, for $k > 0$
- Smoothing
  $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

Smoothing

$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

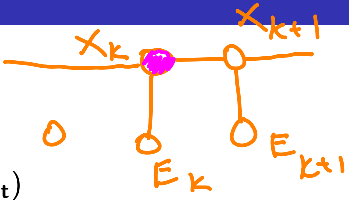
Smoothing

$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}) = \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}, \mathbf{e_{k+1:t}})$

$P(X_k | e_{1:k})$

Smoothing



$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

$$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}) = \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}, \mathbf{e_{k+1:t}})$$
$$= \alpha \mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k}, \mathbf{e_{1:k}})\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}})$$

Smoothing

$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$, for $0 \leq k < t$

$$\begin{aligned}\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}) &= \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}, \mathbf{e_{k+1:t}}) \\ &= \alpha \mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k}, \mathbf{e_{1:k}})\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}) \\ &= \alpha \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}})\mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k})\end{aligned}$$

Smoothing

$$\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}),\ \text{for } 0 \le k < t$$

$$\begin{aligned}
\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}}) &= \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}, \mathbf{e_{k+1:t}}) \\
&= \alpha \mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k}, \mathbf{e_{1:k}})\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}}) \\
&= \alpha \mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:k}})\mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k}) \\
&= \alpha \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}}
\end{aligned}$$

Smoothing

$P(e_{k+1}|X_{k+1})$

$$\mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t}|X_k)}$$

Smoothing

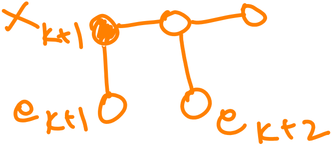
$$\mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t}|X_k)}$$
$$= \sum_{\mathbf{x_{k+1}}} \mathbf{P(e_{k+1:t}|x_{k+1}, X_k)P(x_{k+1}|X_k)}$$

Smoothing

$P(e_{k+1} | x_{k+1})$

$$
\begin{aligned}
\mathbf{b}_{k+1:t} &= \mathbf{P}(\mathbf{e}_{k+1:t} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} | \mathbf{x}_{k+1}, \mathbf{X}_k) \mathbf{P}(\mathbf{x}_{k+1} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t} | \mathbf{x}_{k+1}) \mathbf{P}(\mathbf{x}_{k+1} | \mathbf{X}_k)
\end{aligned}
$$

Smoothing



$$
\begin{aligned}
\mathbf{b}_{k+1:t} &= P(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1}, \mathbf{X}_k) P(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1}) P(\mathbf{x}_{k+1}|\mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t}|\mathbf{x}_{k+1}) P(\mathbf{x}_{k+1}|\mathbf{X}_k)
\end{aligned}
$$

Smoothing

P(A, B | C) = P(A|C) P(B|C)

$$
\begin{aligned}
\mathbf{b}_{k+1:t} &= P(\mathbf{e}_{k+1:t} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} | \mathbf{x}_{k+1}, \mathbf{X}_k) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1}, \mathbf{e}_{k+2:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k) \\
&= \sum_{\mathbf{x}_{k+1}} P(\mathbf{e}_{k+1} | \mathbf{x}_{k+1}) P(\mathbf{e}_{k+2:t} | \mathbf{x}_{k+1}) P(\mathbf{x}_{k+1} | \mathbf{X}_k)
\end{aligned}
$$

k ← k+1

$b_{k+2:t}$

Smoothing

$$\begin{aligned}
\mathbf{b}_{k+1:t} &= P(\mathbf{e}_{k+1:t}|\mathbf{X}_k) \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t}|x_{k+1}, \mathbf{X}_k) P(x_{k+1}|\mathbf{X}_k) \\
&= \sum_{x_{k+1}} P(\mathbf{e}_{k+1:t}|x_{k+1}) P(x_{k+1}|\mathbf{X}_k) \\
&= \sum_{x_{k+1}} P(e_{k+1}, \mathbf{e}_{k+2:t}|x_{k+1}) P(x_{k+1}|\mathbf{X}_k) \\
&= \sum_{x_{k+1}} P(e_{k+1}|x_{k+1}) \underbrace{P(\mathbf{e}_{k+2:t}|x_{k+1})} P(x_{k+1}|\mathbf{X}_k) \\
&= \sum_{x_{k+1}} \underbrace{P(e_{k+1}|x_{k+1})}\, \mathbf{b}_{k+2:t} \underbrace{P(x_{k+1}|\mathbf{X}_k)}
\end{aligned}$$

Smoothing

$$
\begin{aligned}
b_{k+1:t} &= P(e_{k+1:t} | X_k) \\
&= \sum_{x_{k+1}} P(e_{k+1:t} | x_{k+1}) P(x_{k+1} | X_k) \\
&= \sum_{x_{k+1}} P(e_{k+1} | x_{k+1}) \, b_{k+2:t} \, P(x_{k+1} | X_k)
\end{aligned}
$$

Smoothing

$b_{k+1:t}$ $b_{k+2:t}$ ... $b_{t:t}$

$$\mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t}|X_k)}$$
$$= \sum_{\mathbf{x_{k+1}}} \mathbf{P(e_{k+1:t}|x_{k+1})P(x_{k+1}|X_k)}$$
$$= \sum_{\mathbf{x_{k+1}}} \mathbf{P(e_{k+1}|x_{k+1})\, b_{k+2:t}\, P(x_{k+1}|X_k)}$$

Substituting $k = t - 1$ we get :

Smoothing

$$\begin{aligned}
\mathbf{b_{k+1:t}} &= \mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{X_k}) \\
&= \sum_{\mathbf{x_{k+1}}} \mathbf{P}(\mathbf{e_{k+1:t}}|\mathbf{x_{k+1}})\mathbf{P}(\mathbf{x_{k+1}}|\mathbf{X_k}) \\
&= \sum_{\mathbf{x_{k+1}}} \mathbf{P}(\mathbf{e_{k+1}}|\mathbf{x_{k+1}})\, \mathbf{b_{k+2:t}}\, \mathbf{P}(\mathbf{x_{k+1}}|\mathbf{X_k})
\end{aligned}$$

Substituting $k = t - 1$ we get :

$$\mathbf{b_{t:t}} = \mathbf{P}(\mathbf{e_{t:t}}|\mathbf{X_{t-1}})$$

Smoothing

$$\mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1:t}|\mathbf{x}_{k+1})\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$
$$= \sum_{\mathbf{x}_{k+1}} \mathbf{P}(\mathbf{e}_{k+1}|\mathbf{x}_{k+1})\,\mathbf{b}_{k+2:t}\,\mathbf{P}(\mathbf{x}_{k+1}|\mathbf{X}_k)$$

Substituting $k = t - 1$ we get :

$$\mathbf{b}_{t:t} = \mathbf{P}(\mathbf{e}_{t:t}|\mathbf{X}_{t-1})$$
$$= \sum_{\mathbf{x}_t} \mathbf{P}(\mathbf{e}_{t:t}|\mathbf{x}_t)\mathbf{P}(\mathbf{x}_t|\mathbf{X}_{t-1})$$

1

$b_{t-1:t}$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) =$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) = \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}}$$

$P(R_1 \mid u_1)$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) = \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}}$$
$$= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) = \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}}$$
$$= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}}$$

$$\mathbf{f_{1:1}} = \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}) \approx < .8182, .1818 >$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P}(\mathbf{e_{k+1:t}} | \mathbf{X_k})$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$e_{2:2}$

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P}(\mathbf{e_{k+1:t}} \mid \mathbf{X_k})$$

$$= \mathbf{P}(\mathbf{u_2} \mid \mathbf{R_1}) = \sum_{r_2} \mathbf{P}(\mathbf{u_2} \mid \mathbf{r_2}) \mathbf{P}(\mathbf{r_2} \mid \mathbf{R_1})$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t} \mid X_k)}$$

$$= \mathbf{P(u_2 \mid R_1)} = \sum_{r_2} \mathbf{P(u_2 \mid r_2) P(r_2 \mid R_1)}$$

$$\mathbf{(R_2 = True) + (R_2 = False)}$$

$= .9 \times \langle .7, .3 \rangle$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P}(\mathbf{e_{k+1:t}} \mid \mathbf{X_k})$$

$$= \mathbf{P}(\mathbf{u_2} \mid \mathbf{R_1}) = \sum_{r_2} \mathbf{P}(\mathbf{u_2} \mid \mathbf{r_2}) \mathbf{P}(\mathbf{r_2} \mid \mathbf{R_1})$$

$$(\mathbf{R_2} = \mathbf{True}) + (\mathbf{R_2} = \mathbf{False})$$

$$= .9 < .7, .3 > + .2 \times < .3, .7 >$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t} \mid X_k)}$$

$$= \mathbf{P(u_2 \mid R_1)} = \sum_{r_2} \mathbf{P(u_2 \mid r_2) P(r_2 \mid R_1)}$$

$$(\mathbf{R_2} = \mathbf{True}) + (\mathbf{R_2} = \mathbf{False})$$

$$= .9 < .7, .3 > + .2 < .3, .7 >$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b_{2:2}} = \mathbf{b_{k+1:t}} = \mathbf{P(e_{k+1:t} \mid X_k)}$$

$$= \mathbf{P(u_2 \mid R_1)} = \sum_{r_2} \mathbf{P(u_2 \mid r_2) P(r_2 \mid R_1)}$$

$$(\mathbf{R_2} = \textbf{True}) + (\mathbf{R_2} = \textbf{False})$$

$$= .9 < .7, .3 > + .2 < .3, .7 > = < .63, .27 > + < .06, .14 >$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{b}_{2:2} = \mathbf{b}_{k+1:t} = \mathbf{P}(\mathbf{e}_{k+1:t} \mid \mathbf{X}_k)$$

$$= \mathbf{P}(\mathbf{u}_2 \mid \mathbf{R}_1) = \sum_{r_2} \mathbf{P}(\mathbf{u}_2 \mid \mathbf{r}_2)\mathbf{P}(\mathbf{r}_2 \mid \mathbf{R}_1)$$

$$(\mathbf{R}_2 = \mathbf{True}) + (\mathbf{R}_2 = \mathbf{False})$$

$$= .9 < .7, .3 > + .2 < .3, .7 > = < .63, .27 > + < .06, .14 >$$

$$= < .69, .41 >$$

$P(u_2 \mid R_1 = T) = .69$

$P(u_2 \mid R_1 = F) = .41$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) = \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\begin{aligned}\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) &= \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\ &= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}}\end{aligned}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\begin{aligned} \mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) &= \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\ &= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}} \\ &= \alpha\ \mathbf{P(R1 \mid u1)} \times \mathbf{P(u2 \mid R1)} \end{aligned}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\begin{aligned}
\mathbf{P}(\mathbf{R_1}|\mathbf{u_1},\mathbf{u_2}) &= \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\
&= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}} \\
&= \alpha\ \mathbf{P(R1|u1)} \times \mathbf{P(u2|R1)} \\
&\approx \alpha < .8182, .1818 >< .69, .41 >
\end{aligned}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$
\begin{aligned}
\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) &= \alpha\, \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\
&= \alpha\, \mathbf{f_{1:1}} \times \mathbf{b_{2:2}} \\
&= \alpha\, \mathbf{P}(\mathbf{R1} \mid \mathbf{u1}) \times \mathbf{P}(\mathbf{u2} \mid \mathbf{R1}) \\
&\approx \alpha < .8182, .1818 >< .69, .41 > \\
&= \alpha < .5646, .0754 >
\end{aligned}
$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$\begin{aligned}
\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) &= \alpha\ \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\
&= \alpha\ \mathbf{f_{1:1}} \times \mathbf{b_{2:2}} \\
&= \alpha\ \mathbf{P}(\mathbf{R1} \mid \mathbf{u1}) \times \mathbf{P}(\mathbf{u2} \mid \mathbf{R1}) \\
&\approx \alpha < .8182, .1818 >< .69, .41 > \\
&= \alpha < .5646, .0754 > \ , \ \alpha \approx 1.5647
\end{aligned}$$

Smoothing Example

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Q. Find the smoothed estimate for the probability of rain in time slice $k = 1$, given that the umbrella was observed on days 1 and 2.

$$
\begin{aligned}
\mathbf{P}(\mathbf{R_1} \mid \mathbf{u_1}, \mathbf{u_2}) &= \alpha\, \mathbf{f_{1:k}} \times \mathbf{b_{k+1:t}} \\
&= \alpha\, \mathbf{f_{1:1}} \times \mathbf{b_{2:2}} \\
&= \alpha\, \mathbf{P}(\mathbf{R1} \mid \mathbf{u1}) \times \mathbf{P}(\mathbf{u2} \mid \mathbf{R1}) \\
&\approx \alpha < .8182, .1818 >< .69, .41 > \\
&= \alpha < .5646, .0754 > \quad , \alpha \approx 1.5647 \\
&=< .8834, .1166 >
\end{aligned}
$$

- The smoothed estimate for $R_1 = True$ is higher than the filtered estimate.

- The smoothed estimate for $R_1 = True$ is higher than the filtered estimate.
- Time complexity for smoothing w.r.t $e_{1:t}$ for a given time step k : $O(t)$

$f_{1:k}$ $O(k)$ $b_{k+1:t}$

- The smoothed estimate for $R_1 = True$ is higher than the filtered estimate.
- Time complexity for smoothing w.r.t $e_{1:t}$ for a given time step k : $O(t)$
- Time complexity for smoothing state variable in all the time steps $O(t^2)$

$f_{1:k}$ 1 .. t $b_{t:t}$

- The smoothed estimate for $R_1 = True$ is higher than the filtered estimate.
- Time complexity for smoothing w.r.t $e_{1:t}$ for a given time step $k : O(t)$
- Time complexity for smoothing state variable in all the time steps $O(t^2)$
- Can we do better than $O(t^2)$ for finding smoothed estimates for all the time steps?

Forward-backward algorithm

function FORWARD-BACKWARD(**ev**, *prior*) **returns** a vector of probability distributions
  inputs: **ev**, a vector of evidence values for steps $1, \ldots, t$
    prior, the prior distribution on the initial state, $\mathbf{P}(\mathbf{X}_0)$
  local variables: **fv**, a vector of forward messages for steps $0, \ldots, t$
    b, a representation of the backward message, initially all 1s
    sv, a vector of smoothed estimates for steps $1, \ldots, t$

$\mathbf{fv}[0] \leftarrow prior$
for $i = 1$ **to** t **do**
  $\mathbf{fv}[i] \leftarrow \text{FORWARD}(\mathbf{fv}[i-1], \mathbf{ev}[i])$
for $i = t$ **down to** 1 **do**
  $\mathbf{sv}[i] \leftarrow \text{NORMALIZE}(\mathbf{fv}[i] \times \mathbf{b})$
  $\mathbf{b} \leftarrow \text{BACKWARD}(\mathbf{b}, \mathbf{ev}[i])$
return **sv**

< 1 1 >

Forward-backward algorithm

- Forward-backward algorithm is very useful in applications that deal with sequence of noisy observations.

Forward-backward algorithm

- Forward-backward algorithm is very useful in applications that deal with sequence of noisy observations.
- Fixed-lag smoothing $\mathbf{P}(\mathbf{X_{t-d}}|\mathbf{e_{1:t}})$

$P(X_k | e_{1:t})$ $1 \leq k < t$

Finding the most likely sequence

- Observed umbrella sequence : $[true, true, false, true, true]$

Finding the most likely sequence

- Observed umbrella sequence : $[true, true, false, true, true]$
- What weather sequence is most likely to explain the observed data?

Finding the most likely sequence

- Observed umbrella sequence : [*true*, *true*, *false*, *true*, *true*]
- What weather sequence is most likely to explain the observed data?

$$\arg\max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}} | \mathbf{e_{1:t}})$$

Finding the most likely sequence

- Observed umbrella sequence : $[true, true, false, true, true]$
- What weather sequence is most likely to explain the observed data?

  $$\arg\max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})$$

- Naive approach: Iterate over all the 2^t possible sequence of state variables and find $x_{1:t}$ that maximizes $\mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})$.

Finding the most likely sequence

- Another approach: Use smoothing to find $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$ for all the time steps k in $O(t)$ time. For each variable X_k pick a value that has the maximum probability.

$e_{1:t}$ $\quad$ $P(X_k|e_{1:t}) =$

Finding the most likely sequence

- Another approach: Use smoothing to find $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$ for all the time steps k in $O(t)$ time. For each variable X_k pick a value that has the maximum probability.
- Is there any problem with this approach?

Finding the most likely sequence

- Another approach: Use smoothing to find $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$ for all the time steps k in $O(t)$ time. For each variable X_k pick a value that has the maximum probability.
- Is there any problem with this approach?
- Marginal probabilities can be misleading. We need to look at the joint probabilities.

Finding the most likely sequence

- Another approach: Use smoothing to find $\mathbf{P}(\mathbf{X_k}|\mathbf{e_{1:t}})$ for all the time steps k in $O(t)$ time. For each variable X_k pick a value that has the maximum probability.
- Is there any problem with this approach?
- Marginal probabilities can be misleading. We need to look at the joint probabilities.

	$\mathbf{X_2}$ = **True**	$\mathbf{X_2}$ = **False**
$\mathbf{X_1}$ = **True**	**.40**	**.05**
$\mathbf{X_1}$ = **False**	**.25**	**.30**

.65 .35
.45 .55
$x_1 = F \quad x_2 = T$

Finding the most likely sequence

$$\arg\max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}} | \mathbf{e_{1:t}}) =$$

Finding the most likely sequence

$$\arg\max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}}) = \arg\max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})\mathbf{P}(\mathbf{e_{1:t}})$$

Finding the most likely sequence

$$\underset{x_{1:t}}{\arg\max}\, \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}}) = \underset{x_{1:t}}{\arg\max}\, \mathbf{P}(\mathbf{x_{1:t}}|\mathbf{e_{1:t}})\mathbf{P}(\mathbf{e_{1:t}})$$

$$= \underset{x_{1:t}}{\arg\max}\, \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

Finding the most likely sequence

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Observed umbrella sequence : [*true*, *true*, *false*, *true*, *true*]

Finding the most likely sequence

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Observed umbrella sequence : [*true*, *true*, *false*, *true*, *true*]

	R_0	R_1	R_2	R_3
True	·5 →	·315		
False	·5 →	·07		
		$u1$	$u2$	$\neg u3$

$R_1 = T$

$\cdot 5 \times 0{\cdot}7 \times \cdot 9 = \cdot 315$

$\cdot 5 \times 0{\cdot}3 \times \cdot 9$

$R_1 = F$

$\cdot 5 \times \cdot 3 \times 0 \cdot 2$

$\rightarrow \cdot 5 \times \cdot 7 \times 0 \cdot 2 = \cdot 07$

Finding the most likely sequence

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2

Observed umbrella sequence : $[true, true, false, true, true]$

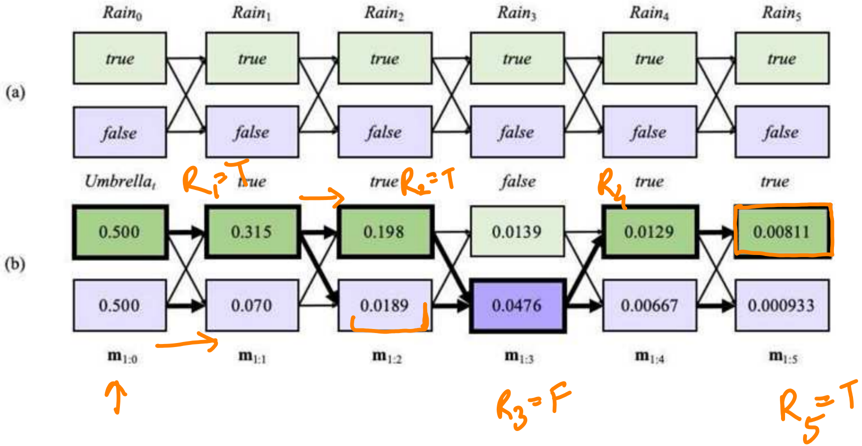
	R_0	R_1	R_2	R_3
True		.315 →	.198	
False		.07		
		$u1$	$u2$	$\neg u3$

$R_2 = T$

Finding the most likely sequence

R_{t-1}	$P(R_t \mid R_{t-1})$
t	0.7
f	0.3

R_t	$P(U_t \mid R_t)$
t	0.9
f	0.2



Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \underbrace{\mathbf{e_{1:t+1}}})$$

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}})$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}})$$

$$= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$\begin{aligned}
m_{1:t+1} &= \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\
&= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})
\end{aligned}$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$\begin{aligned}
m_{1:t+1} &= \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\
&= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}} | \mathbf{x_{1:t}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})
\end{aligned}$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}})$$

$$= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}} | \mathbf{x_{1:t}}, \mathbf{e_{1:t}}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}} | \mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}} | \mathbf{x_t}) \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$\begin{aligned}
m_{1:t+1} &= \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}}) \\
&= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_{1:t}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}}) \\
&= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})
\end{aligned}$$

Finding the most likely sequence

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t+1}})$$

$$m_{1:t+1} = \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}}, \mathbf{e_{t+1}})$$

$$= \max_{x_{1:t}} \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})$$

$$= \underbrace{\mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}})} \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{X_{t+1}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_{1:t}}, \mathbf{e_{1:t}})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_{1:t}} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t})\mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t}) \max_{x_{1:t}} \mathbf{P}(\mathbf{x_{1:t}}, \mathbf{e_{1:t}})$$

$$= \mathbf{P}(\mathbf{e_{t+1}}|\mathbf{X_{t+1}}) \max_{x_t} \mathbf{P}(\mathbf{X_{t+1}}|\mathbf{x_t}) \max_{x_{1:t}} \underbrace{\mathbf{P}(\mathbf{x_{1:t-1}}, \mathbf{x_t}, \mathbf{e_{1:t}})}$$

Finding the most likely sequence

- For each state, we need to record the best state that leads to it.

Finding the most likely sequence

- For each state, we need to record the best state that leads to it.
- **Viterbi algorithm**

Finding the most likely sequence

- For each state, we need to record the best state that leads to it.
- **Viterbi algorithm**
- Time complexity $O(t)$,

Finding the most likely sequence

- For each state, we need to record the best state that leads to it.
- **Viterbi algorithm**
- Time complexity $O(t)$, Space complexity $O(t)$

Finding the most likely sequence

- For each state, we need to record the best state that leads to it.
- **Viterbi algorithm**
- Time complexity $O(t)$, Space complexity $O(t)$
- Section 14.3 not needed.

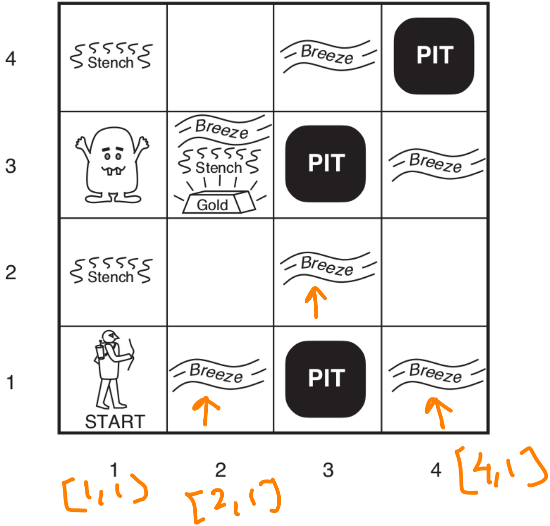
- Knowledge base

- Knowledge base
- Propositional logic

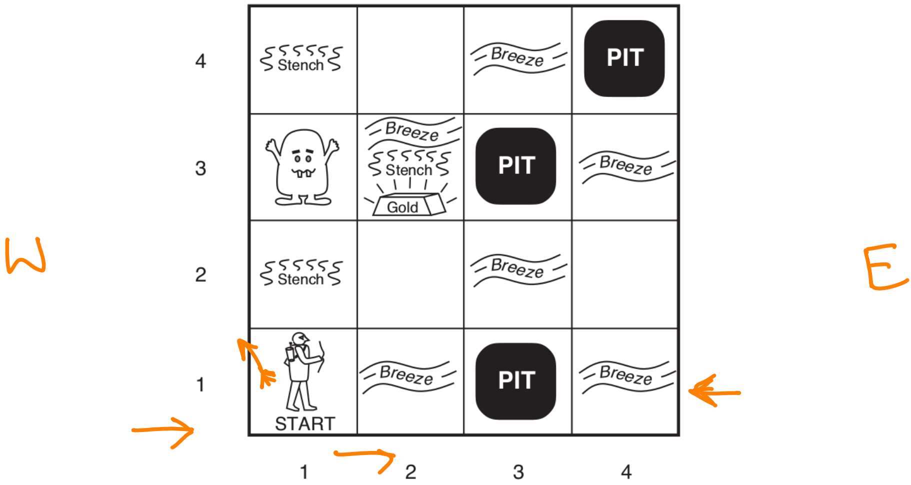
Chapter 7: Logical Agents

- Knowledge base
- Propositional logic
- Inference

Logical Agents



Logical Agents



Percept in each time step: $[Stench, Breeze, Glitter, Bump, Scream]$

First Two Steps

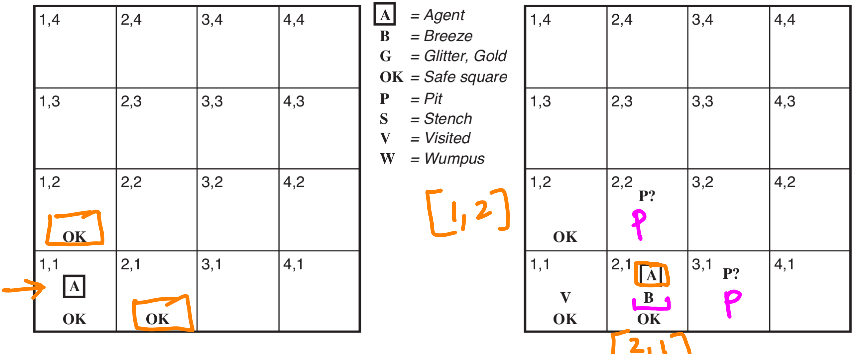


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept [*None*, *None*, *None*, *None*, *None*]. (b) After one move, with percept [*None*, *Breeze*, *None*, *None*, *None*].

Next Steps

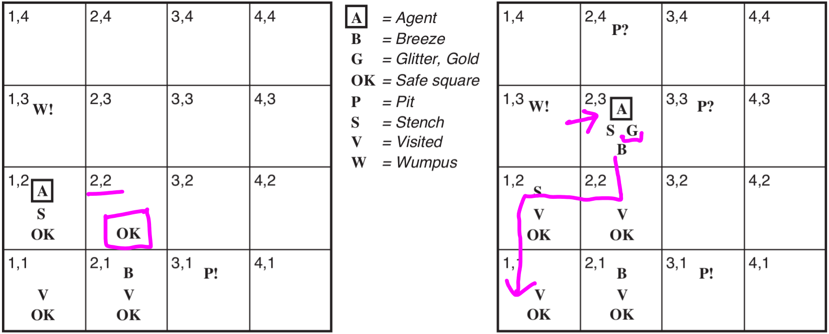
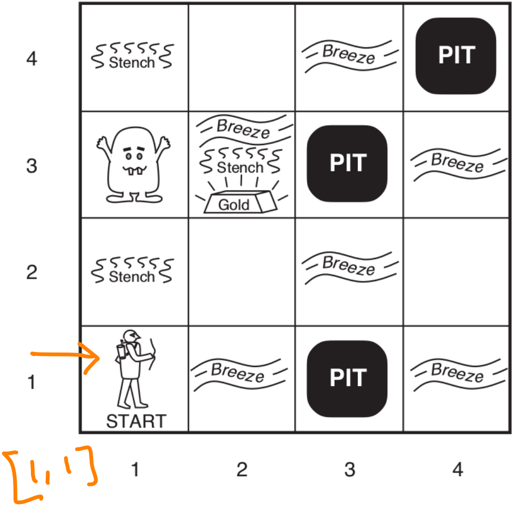
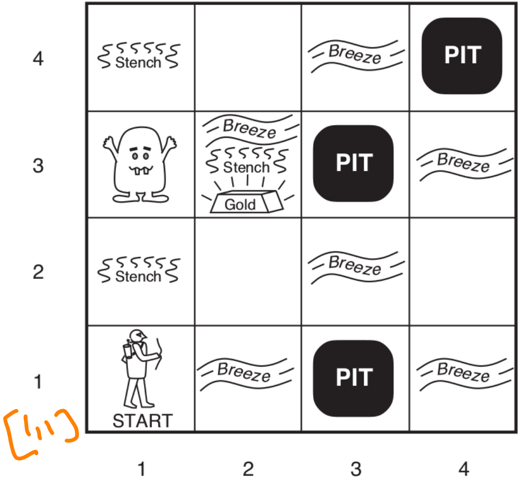


Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept $[Stench, None, None, None, None]$. (b) After the fifth move, with percept $[Stench, Breeze, Glitter, None, None]$.

Logical Agents





Percept in each time step: $[Stench, Breeze, Glitter, Bump, Scream]$

First Two Steps

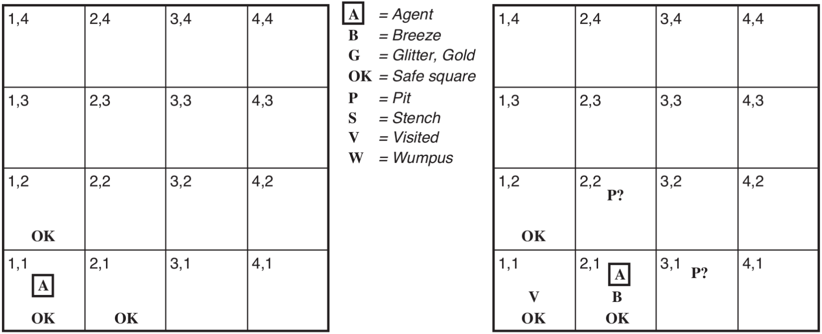


Figure 7.3 The first step taken by the agent in the wumpus world. (a) The initial situation, after percept $[None, None, None, None, None]$. (b) After one move, with percept $[None, Breeze, None, None, None]$.

Next Steps

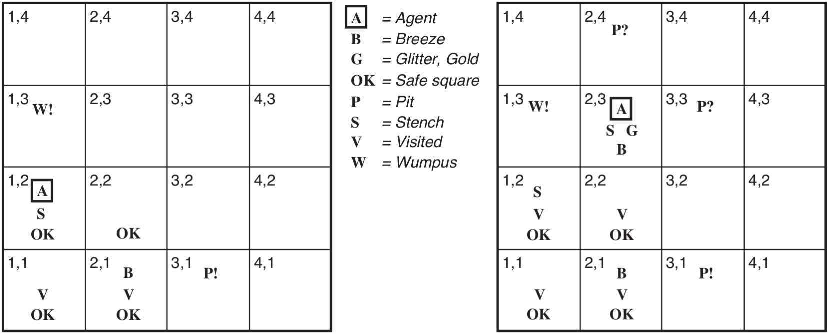


Figure 7.4 Two later stages in the progress of the agent. (a) After the third move, with percept $[Stench, None, None, None, None]$. (b) After the fifth move, with percept $[Stench, Breeze, Glitter, None, None]$.

Propositional Logic

$$
\begin{aligned}
Sentence &\rightarrow AtomicSentence \mid ComplexSentence \\
AtomicSentence &\rightarrow True \mid False \mid P \mid Q \mid R \mid \ldots \\
ComplexSentence &\rightarrow (\ Sentence\ ) \mid [\ Sentence\ ] \\
&\mid \neg\ Sentence \\
&\mid Sentence \land Sentence \\
&\mid Sentence \lor Sentence \\
&\mid Sentence \Rightarrow Sentence \\
&\mid Sentence \Leftrightarrow Sentence \\
\textsc{Operator Precedence} &: \neg, \land, \lor, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Propositional Logic Connectives

NEGATION
LITERAL

$\neg$ (not). A sentence such as $\neg W_{1,3}$ is called the **negation** of $W_{1,3}$. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).

CONJUNCTION

$\wedge$ (and). A sentence whose main connective is $\wedge$, such as $W_{1,3} \wedge P_{3,1}$, is called a **conjunction**; its parts are the **conjuncts**.

DISJUNCTION

$\vee$ (or). A sentence using $\vee$, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a **disjunction** of the **disjuncts** $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$.

IMPLICATION
PREMISE
CONCLUSION
RULES

$\Rightarrow$ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$.

BICONDITIONAL

$\Leftrightarrow$ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**.

Propositional Logic Connectives

NEGATION
LITERAL

$\neg$ (not). A sentence such as $\neg W_{1,3}$ is called the **negation** of $W_{1,3}$. A **literal** is either an atomic sentence (a **positive literal**) or a negated atomic sentence (a **negative literal**).

CONJUNCTION

$\wedge$ (and). A sentence whose main connective is $\wedge$, such as $W_{1,3} \wedge P_{3,1}$, is called a **conjunction**; its parts are the **conjuncts**.

DISJUNCTION

$\vee$ (or). A sentence using $\vee$, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a **disjunction** of the **disjuncts** $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$.

IMPLICATION
PREMISE
CONCLUSION
RULES

$\Rightarrow$ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$.

BICONDITIONAL

$\Leftrightarrow$ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**.

Semantics of PL

$\alpha \equiv$

Logic

1. Model
2. $M(\alpha_1)$
3. Entailment $(\alpha \models \beta)$
4. $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$

$M(\alpha_1)$

1. Model
2. $M(\alpha_1)$
3. Entailment $(\alpha \models \beta)$
4. $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$
5. Does $(a \vee b) \models (a \vee b \vee c)$?

$m(a \vee b) = \{a=T, b=F, c=T\}$

$m(\alpha) \subseteq m(\beta)$

Logic

1. Model
2. $M(\alpha_1)$

$\{a=F, b=F, c=T\}$

$\in M(\alpha)$ $\quad$ $\notin M(\beta)$

3. Entailment ($\alpha \models \beta$)
4. $\alpha \models \beta$ iff $M(\alpha) \subseteq M(\beta)$
5. Does $(a \vee b) \models (a \vee b \vee c)$?
6. Does $(a \vee b \vee c) \models (a \vee b)$?

$M(\alpha) \not\subseteq M(\beta)$

Example

A knowledge-based agent knows that whenever there is a *party* (P), then there is *food* (F) and *soft drinks* (D). When there is no *party*, then either there is *food* or there are *games* (G) (or both). The agent perceives that there are no *games*.

Example

A knowledge-based agent knows that whenever there is a *party* (P), then there is *food* (F) and *soft drinks* (D). When there is no *party*, then either there is *food* or there are *games* (G) (or both). The agent perceives that there are no *games*.

- What propositional logic sentences must be present in the agent's knowledge base after the agent has perceived that there are no *games*? Use the symbols P, F, D and G to construct the sentences.

Example

A knowledge-based agent knows that whenever there is a *party* (P), then there is *food* (F) and *soft drinks* (D). When there is no *party*, then either there is *food* or there are *games* (G) (or both). The agent perceives that there are no *games*.

- What propositional logic sentences must be present in the agent's knowledge base after the agent has perceived that there are no *games*? Use the symbols P, F, D and G to construct the sentences.

  R1: $P \Rightarrow F \wedge D$

Example

A knowledge-based agent knows that whenever there is a *party* (P), then there is *food* (F) and *soft drinks* (D). When there is no *party*, then either there is *food* or there are *games* (G) (or both). The agent perceives that there are no *games*.

- What propositional logic sentences must be present in the agent's knowledge base after the agent has perceived that there are no *games*? Use the symbols P, F, D and G to construct the sentences.

  R1: $P \Rightarrow F \wedge D$
  R2: $\neg P \Rightarrow F \vee G$

Example

A knowledge-based agent knows that whenever there is a *party* (P), then there is *food* (F) and *soft drinks* (D). When there is no *party*, then either there is *food* or there are *games* (G) (or both). The agent perceives that there are no *games*.

- What propositional logic sentences must be present in the agent's knowledge base after the agent has perceived that there are no *games*? Use the symbols P, F, D and G to construct the sentences.

  R1: $P \Rightarrow F \wedge D$
  R2: $\neg P \Rightarrow F \vee G$
  R3: $\neg G$

Example

KB: R1: $P \Rightarrow F \land D$
R2: $\neg P \Rightarrow F \lor G$
R3: $\neg G$

Example

KB: R1: $P \Rightarrow F \land D$

R2: $\neg P \Rightarrow F \lor G$

R3: $\neg G$

- Find the models in which the knowledge base is true?

$2^4 = 16$

Example

KB: R1: $P \Rightarrow F \wedge D$
R2: $\neg P \Rightarrow F \vee G$
R3: $\neg G$

- Find the models in which the knowledge base is true?

P	F	D	G	KB
False	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*True*	*True*	*False*	*True*

T T F F

Example

KB: R1: $P \Rightarrow F \land D$
R2: $\neg P \Rightarrow F \lor G$
R3: $\neg G$

- Find the models in which the knowledge base is true?

P	F	D	G	KB
False	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*True*	*True*	*False*	*True*

- Can we infer that there is a *party*? Does $KB \models P$?

$m(KB) \not\subseteq m(P)$ $m(KB) \subseteq m(\neg P)$

Example

KB: R1: $P \Rightarrow F \wedge D$

R2: $\neg P \Rightarrow F \vee G$

R3: $\neg G$

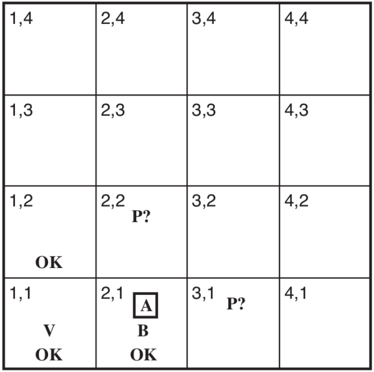
- Find the models in which the knowledge base is true?

P	F	D	G	KB
False	*True*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*True*	*True*	*False*	*True*

- Can we infer that there is a *party*? Does $KB \models P$?
- Can we infer that there is *food*? Does $KB \models F$?

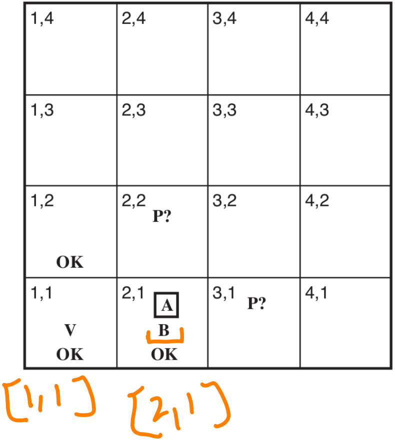
$m(KB) \subseteq m(F)$

Wupus-world inference example



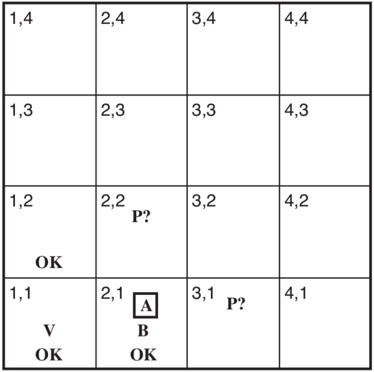
- KB contains agent's percepts (in the first 2 steps) and rules of the Wumpus world

Wupus-world inference example



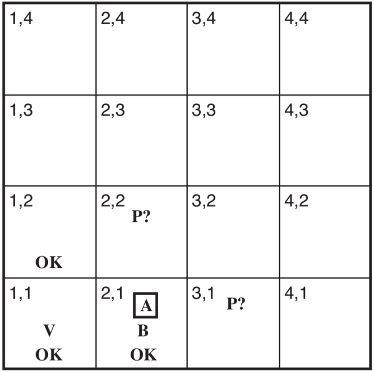
- KB contains agent's percepts (in the first 2 steps) and rules of the Wumpus world
- Agent wants to know whether pit is present in [1,2] and [2,2].

Wupus-world inference example



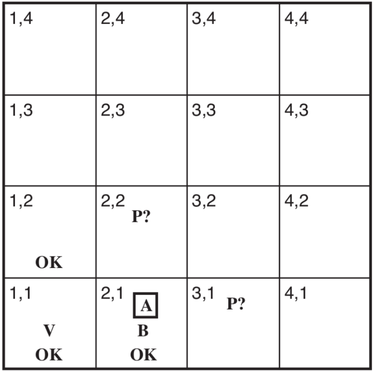
- KB contains agent's percepts (in the first 2 steps) and rules of the Wumpus world
- Agent wants to know whether pit is present in [1,2] and [2,2].
- $\alpha_1 \equiv$ "No pit in [1,2]"
- $\alpha_2 \equiv$ "No pit in [2,2]"

Wupus-world inference example



- KB contains agent's percepts (in the first 2 steps) and rules of the Wumpus world
- Agent wants to know whether pit is present in [1,2] and [2,2].
- $\alpha_1 \equiv$ "No pit in [1,2]"
- $\alpha_2 \equiv$ "No pit in [2,2]"
- $KB \models \alpha_1$?

Wupus-world inference example

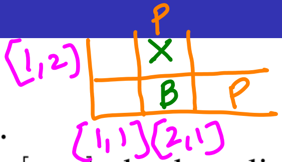


- KB contains agent's percepts (in the first 2 steps) and rules of the Wumpus world
- Agent wants to know whether pit is present in [1,2] and [2,2].
- $\alpha_1 \equiv$ "No pit in [1,2]"
- $\alpha_2 \equiv$ "No pit in [2,2]"
- $KB \models \alpha_1$?
- $KB \models \alpha_2$?

Simple Knowledge Base

$P_{x,y}$ is true if there is a pit in $[x, y]$.
$W_{x,y}$ is true if there is a wumpus in $[x, y]$, dead or alive.
$B_{x,y}$ is true if the agent perceives a breeze in $[x, y]$.
$S_{x,y}$ is true if the agent perceives a stench in $[x, y]$.

Simple Knowledge Base



$P_{x,y}$ is true if there is a pit in $[x, y]$.
$W_{x,y}$ is true if there is a wumpus in $[x, y]$, dead or alive.
$B_{x,y}$ is true if the agent perceives a breeze in $[x, y]$.
$S_{x,y}$ is true if the agent perceives a stench in $[x, y]$.

KB:

$$
\begin{aligned}
R_1 &: \quad \neg P_{1,1} \\
R_2 &: \quad B_{1,1} \quad \Leftrightarrow \quad (P_{1,2} \lor P_{2,1}) \\
R_3 &: \quad B_{2,1} \quad \Leftrightarrow \quad (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \\
R_4 &: \quad \neg B_{1,1} \\
R_5 &: \quad B_{2,1}
\end{aligned}
$$

Simple Knowledge Base

Does KB $\models P_{1,2}$?

Simple Knowledge Base

Does KB $\models P_{1,2}$?

Does KB $\models P_{2,2}$?

$KB \models \neg P_{1,2}$

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Figure 7.9 A truth table constructed for the knowledge base given in the text. KB is true if R_1 through R_5 are true, which occurs in just 3 of the 128 rows (the ones underlined in the right-hand column). In all 3 rows, $P_{1,2}$ is false, so there is no pit in [1,2]. On the other hand, there might (or might not) be a pit in [2,2].

$m(KB) \subseteq m(\neg P_{1,2}) \quad KB \models \neg P_{1,2}$

Logical inference algorithms

- Model checking

Logical inference algorithms

- Model checking
- Inference algorithm ($KB \vdash_i \alpha$) (algorithm i derives α from KB)

Logical inference algorithms

- Model checking
- Inference algorithm ($KB \vdash_i \alpha$) (algorithm i derives α from KB)
- Soundness :

Logical inference algorithms

- Model checking
- Inference algorithm ($KB \vdash_i \alpha$) (algorithm i derives α from KB)
- Soundness : If $KB \vdash_i \alpha$, then $KB \models \alpha$

Logical inference algorithms

- Model checking
- Inference algorithm ($KB \vdash_i \alpha$) (algorithm i derives α from KB)
- Soundness : If $KB \vdash_i \alpha$, then $KB \models \alpha$
- Completeness :

Logical inference algorithms

n 2^n

- Model checking
- Inference algorithm ($KB \vdash_i \alpha$) (algorithm i derives α from KB)
- Soundness : If $KB \vdash_i \alpha$, then $KB \models \alpha$
- Completeness : If $KB \models \alpha$, then $KB \vdash_i \alpha$

Logical equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Inference rules

Modus Ponens :

Inference rules

Modus Ponens :

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

Inference rules

Modus Ponens :

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

And-Elimination :

$$\frac{\alpha \wedge \beta}{\alpha}$$

Inference rules

Modus Ponens :

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

And-Elimination :

$$\frac{\alpha \wedge \beta}{\alpha}$$

Resolution :

Inference rules

Modus Ponens :

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

And-Elimination :

$$\frac{\alpha \wedge \beta}{\alpha}$$

Resolution :

$$\frac{a \vee b \vee \neg c, \qquad c \vee d}{a \vee b \vee d}$$

Concepts for inference

Concepts for inference

- Logical equivalences ($\equiv$)

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem
  $\alpha \models \beta$ if and only if $(\alpha \Rightarrow \beta)$ is valid.

$M(\alpha) \subseteq M(\beta)$

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem
  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem
  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity
  - Suppose $KB \models \alpha$. Is it possible to add a sentence to KB such that $KB' \not\models \alpha$?

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem

  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity
  - Suppose $KB \models \alpha$. Is it possible to add a sentence to KB such that $KB' \not\models \alpha$?

Suppose KB' is obtained by adding more sentences to KB.

$m(KB') \subseteq m(KB)$

Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem

  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity
  - Suppose $KB \models \alpha$. Is it possible to add a sentence to KB such that $KB' \not\models \alpha$?

  Suppose KB' is obtained by adding more sentences to KB.

$$M(KB) \subseteq M(\alpha)$$

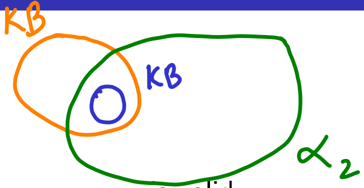
Concepts for inference

- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem
  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity
  - Suppose $KB \models \alpha$. Is it possible to add a sentence to KB such that $KB' \not\models \alpha$?

  Suppose KB' is obtained by adding more sentences to KB.

$$M(KB) \subseteq M(\alpha)$$
$$M(KB') \subseteq M(KB)$$

Concepts for inference



- Logical equivalences ($\equiv$)
- Validity or Tautology
- Deduction theorem

  $\alpha \models \beta$ if and only if ________________ is valid.
- Monotonicity
  - Suppose $KB \models \alpha$. Is it possible to add a sentence to KB such that $KB' \not\models \alpha$?

  Suppose KB' is obtained by adding more sentences to KB.

$$M(KB) \subseteq M(\alpha)$$
$$M(KB') \subseteq M(KB)$$
$$\therefore M(KB') \subseteq M(\alpha)$$

$KB' \models \alpha$

Conjunctive Normal Form

- Clause

$a \lor b \lor \neg c$

Conjunctive Normal Form

- Clause
- Conjuctive Normal Form (CNF) : Conjunction of Clauses

$(a \vee b) \wedge (\neg b \vee c)$

Conjunctive Normal Form

- Clause
- Conjuctive Normal Form (CNF) : Conjunction of Clauses
- Can every sentence α be written in a logically equivalent CNF?

Conjunctive Normal Form

- Clause
- Conjuctive Normal Form (CNF) : Conjunction of Clauses
- Can every sentence α be written in a logically equivalent CNF?
- What is the CNF of $B_{1,1} \Leftrightarrow P_{2,1} \vee P_{1,2}$?

Resolution Algorithm

- Deduction theorem : $M(\beta) \subseteq M(\alpha)$

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

Resolution Algorithm

- Deduction theorem :

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

Resolution Algorithm

- Deduction theorem :

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\beta \wedge \neg\alpha$ is a contradiction.

Resolution Algorithm

- Deduction theorem :

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

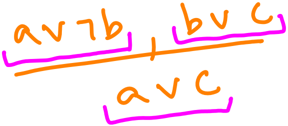
  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\beta \wedge \neg\alpha$ is a contradiction.

- Is this sentence in CNF?

  $(a \vee \neg b) \wedge (\neg a \vee \neg b) \wedge (b)$

Resolution Algorithm



- Deduction theorem :

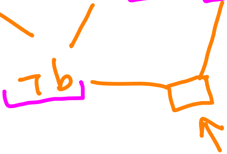
  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\beta \wedge \neg\alpha$ is a contradiction.

- Is this sentence in CNF? Is it a contradiction?

  $(a \vee \neg b) \wedge (\neg a \vee \neg b) \wedge (b)$



Resolution Algorithm

- Deduction theorem :

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\beta \wedge \neg\alpha$ is a contradiction.

- Is this sentence in CNF? Is it a contradiction?

  $(a \vee \neg b) \wedge (\neg a \vee \neg b) \wedge (b)$

- Factoring

Resolution Algorithm

- Deduction theorem :

  $\beta \models \alpha$ if and only if $\beta \Rightarrow \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\neg\beta \vee \alpha$ is valid.

  $\beta \models \alpha$ if and only if $\beta \wedge \neg\alpha$ is a contradiction.

- Is this sentence in CNF? Is it a contradiction?

  $(a \vee \neg b) \wedge (\neg a \vee \neg b) \wedge (b)$

- Factoring
- Ground resolution theorem

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.
  KB:

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.

  KB:

  R1: $C_1 \wedge C_2$

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \land \neg\alpha$ is a contradiction.

  KB:

  R1: $C_1 \land C_2$

  R2: $C_3 \land C_4 \land C_5$

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.

  KB:

  R1: $C_1 \wedge C_2$
  R2: $C_3 \wedge C_4 \wedge C_5$
  R3: C_6

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.

  KB:

  R1: $C_1 \wedge C_2$
  R2: $C_3 \wedge C_4 \wedge C_5$
  R3: C_6
- $KB \equiv C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6$

Resolution Algorithm

- How can we use the Resolution Algorithm to check whether $KB \models \alpha$?
- $KB \models \alpha$ if and only if $KB \wedge \neg\alpha$ is a contradiction.

KB:

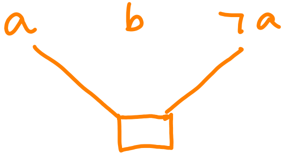
R1: $C_1 \wedge C_2$
R2: $C_3 \wedge C_4 \wedge C_5$
R3: C_6

$\neg\alpha \equiv C_9 \wedge C_{10} \wedge \ldots$

- $KB \equiv C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6$
- $KB \wedge \neg\alpha \equiv C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge \neg\alpha$

(i) Check whether $a \wedge b \models a$

$KB \wedge \neg \alpha \equiv a \wedge b \wedge \neg a$

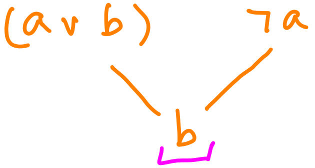


Resolution Algorithm

(i) Check whether $a \wedge b \models a$

(ii) Check whether $a \vee b \models a$

$KB \wedge \neg\alpha = (a \vee b) \wedge \neg a$



Resolution Algorithm Inference

KB:

R1: $\neg B_{1,1}$

R2: $B_{1,1} \Leftrightarrow P_{2,1} \vee P_{1,2}$

Resolution Algorithm Inference

KB:

R1: $\neg B_{1,1}$

R2: $B_{1,1} \Leftrightarrow P_{2,1} \vee P_{1,2}$

$KB \wedge P_{1,2}$

- Does $KB \models \neg P_{1,2}$?

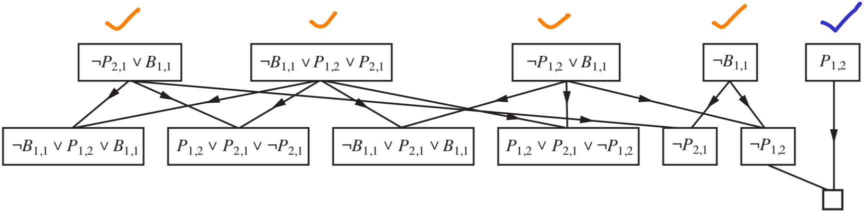


Figure 7.13 Partial application of PL-RESOLUTION to a simple inference in the wumpus world. $\neg P_{1,2}$ is shown to follow from the first four clauses in the top row.

Resolution Algorithm

$KB \models \alpha$

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
      α, the query, a sentence in propositional logic

  $clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  $new \leftarrow \{\ \}$
  loop do
    for each pair of clauses C_i, C_j **in** $clauses$ **do**
      $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
      if $resolvents$ contains the empty clause **then return** $true$
      $new \leftarrow new \cup resolvents$
    if $new \subseteq clauses$ **then return** $false$
    $clauses \leftarrow clauses \cup new$

$KB \not\models \alpha$

Soundness and Completeness of Resolution

- Is resolution algorithm sound? Deduction theorem

α

$KB \wedge \neg\alpha$

$KB \models \alpha$

Soundness and Completeness of Resolution

- Is resolution algorithm sound? Deduction theorem
- Complete? Ground resolution theorem

$KB \models \alpha$ $\quad$ $(KB \wedge \neg\alpha)$

Resolution Algorithm

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
inputs: KB, the knowledge base, a sentence in propositional logic
α, the query, a sentence in propositional logic

$clauses \leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
$new \leftarrow \{\ \}$
loop do
  for each pair of clauses C_i, C_j **in** $clauses$ **do**
    $resolvents \leftarrow$ PL-RESOLVE(C_i, C_j)
    if $resolvents$ contains the empty clause **then return** *true*
    $new \leftarrow new \cup resolvents$
  if $new \subseteq clauses$ **then return** *false*
  $clauses \leftarrow clauses \cup new$

Resolution Algorithm

Factoring :

Resolution Algorithm

Factoring :

$$\frac{a \lor b \lor \neg c, \qquad \neg a \lor b \lor d}{b \lor \neg c \lor d}$$

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

n $\qquad$ $2n$

2^{2n}

Resolution Algorithm

Factoring :

$$\frac{a \vee b \vee \neg c, \qquad \neg a \vee b \vee d}{b \vee \neg c \vee d}$$

Maximum possible number of clauses?

$$2^{2n}$$

A more efficient algorithm

- SAT is NP-complete.
- Can we come up with a more efficient algorithm?

Effective algorithm for Satisfiability

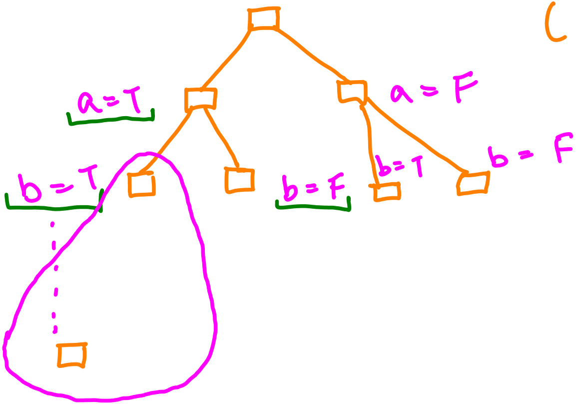
- SAT is NP-complete.

Effective algorithm for Satisfiability

- SAT is NP-complete.
- $(\neg a \vee \neg b) \wedge (a \vee b \vee \neg c \vee d) \wedge \dots$

Effective algorithm for Satisfiability

- SAT is NP-complete.
- $(\neg a \vee \neg b) \wedge (a \vee b \vee \neg c \vee d) \wedge \ldots$
- Backtracking algorithm



Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF

Output : Is the sentence satisfiable?

- Early termination

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic

e.g. 1 : $(a \vee \neg b) \wedge (\neg b \vee \neg c) \wedge (c \vee a)$

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic

e.g. 1 : $(a \vee \neg b) \wedge (\neg b \vee \neg c) \wedge (c \vee a)$
e.g. 2 : $(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee a \vee \ldots) \wedge \ldots$

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF
Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic

e.g. 1 : $(a \vee \neg b) \wedge (\neg b \vee \neg c) \wedge (c \vee a)$
e.g. 2 : $(a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee a \vee \ldots) \wedge \ldots$

- Unit clause heuristic

Davis, Putnam, Logemann and Loveland (DPLL) Algorithm

Input : A sentence in CNF

Output : Is the sentence satisfiable?

- Early termination
- Pure symbol heuristic

e.g. 1 : $(a \lor \neg b) \land (\neg b \lor \neg c) \land (c \lor a)$

e.g. 2 : $(a \lor \neg b) \land (b \lor \neg c) \land (c \lor a \lor \ldots) \land \ldots$

- Unit clause heuristic

e.g. : $a \land (\neg a \lor \neg b \lor c \lor \neg d) \land \ldots$

a = True

a = T
b = T
c = F
d = F

DPLL Algorithm

```
function DPLL-SATISFIABLE?(s) returns true or false
   inputs: s, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of s
   symbols ← a list of the proposition symbols in s
   return DPLL(clauses, symbols, { })
```

```
function DPLL(clauses, symbols, model) returns true or false

   if every clause in clauses is true in model then return true
   if some clause in clauses is false in model then return false
   P, value ← FIND-PURE-SYMBOL(symbols, clauses, model)
   if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
   P, value ← FIND-UNIT-CLAUSE(clauses, model)
   if P is non-null then return DPLL(clauses, symbols – P, model ∪ {P=value})
   P ← FIRST(symbols); rest ← REST(symbols)
   return DPLL(clauses, rest, model ∪ {P=true}) or
          DPLL(clauses, rest, model ∪ {P=false}))
```

DPLL Algorithm

Further enhancements:

DPLL Algorithm

Further enhancements:

- Component Analysis

DPLL Algorithm

Further enhancements:

- Component Analysis
  - 10 unassigned symbols : S_1 to S_5, and S_6 to S_{10}

DPLL Algorithm

Further enhancements:

- Component Analysis
  - 10 unassigned symbols : S_1 to S_5, and S_6 to S_{10}
  - $C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7 \wedge C_8$

$2^5 + 2^5 = 64$

$2^{10} = 1024$

DPLL Algorithm

Further enhancements:

- Component Analysis
  - 10 unassigned symbols : S_1 to S_5, and S_6 to S_{10}
  - $C_1 \wedge C_2 \wedge C_3 \wedge C_4 \;\; \wedge \;\; C_5 \wedge C_6 \wedge C_7 \wedge C_8$
- Variable and value ordering

DPLL Algorithm

Further enhancements:

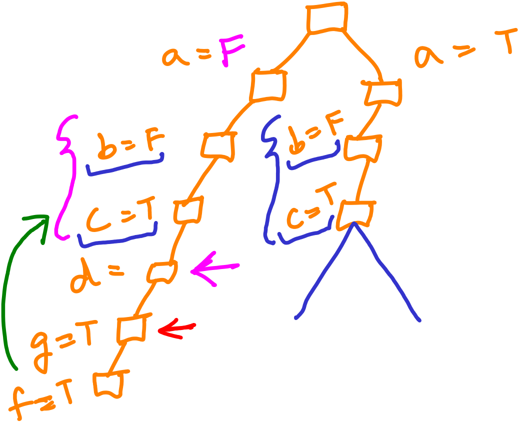
- Intelligent Backtracking

DPLL Algorithm

Further enhancements:

- Intelligent Backtracking

e.g. : $\ldots \wedge (b \vee \neg c \vee g) \wedge (b \vee \neg c \vee f) \wedge (\neg g \vee \neg f) \wedge \ldots$



DPLL Algorithm

Further enhancements:

- Intelligent Backtracking

e.g. : $\ldots \wedge (b \vee \neg c \vee g) \wedge (b \vee \neg c \vee f) \wedge (\neg g \vee \neg f) \wedge \ldots$

- Conflict clause learning

DPLL Algorithm

Further enhancements:

- Intelligent Backtracking

e.g. : $\ldots \wedge (b \vee \neg c \vee g) \wedge (b \vee \neg c \vee f) \wedge (\neg g \vee \neg f) \wedge \ldots$

- Conflict clause learning
- Random restarts

DPLL Algorithm

Further enhancements:

- Intelligent Backtracking

e.g. : $\ldots \wedge (b \vee \neg c \vee g) \wedge (b \vee \neg c \vee f) \wedge (\neg g \vee \neg f) \wedge \ldots$

- Conflict clause learning
- Random restarts
- Clever indexing

DPLL Algorithm

Further enhancements:

- Intelligent Backtracking

e.g. : $\ldots \wedge (b \vee \neg c \vee g) \wedge (b \vee \neg c \vee f) \wedge (\neg g \vee \neg f) \wedge \ldots$

- Conflict clause learning
- Random restarts
- Clever indexing
- SAT Solvers

Local Search : WALKSAT Algorithm

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

Figure 7.18 The WALKSAT algorithm for checking satisfiability by randomly flipping the values of variables. Many versions of the algorithm exist.

$(a \vee \neg b \vee c)$

p

SAT Problems

$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

3-CNF

5 symbol

$2^5 = 32$

16

$16/32 = 1/2$

SAT Problems

$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

- Underconstrained SAT problem

SAT Problems

$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

- Underconstrained SAT problem
- $CNF_k(m, n)$

k-CNF

m clause

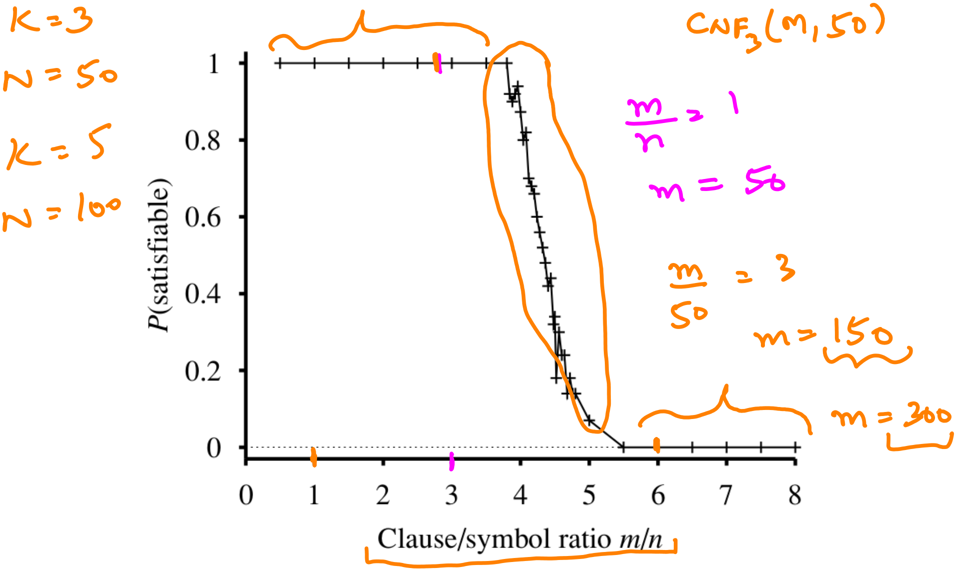
n symbols

SAT Problems

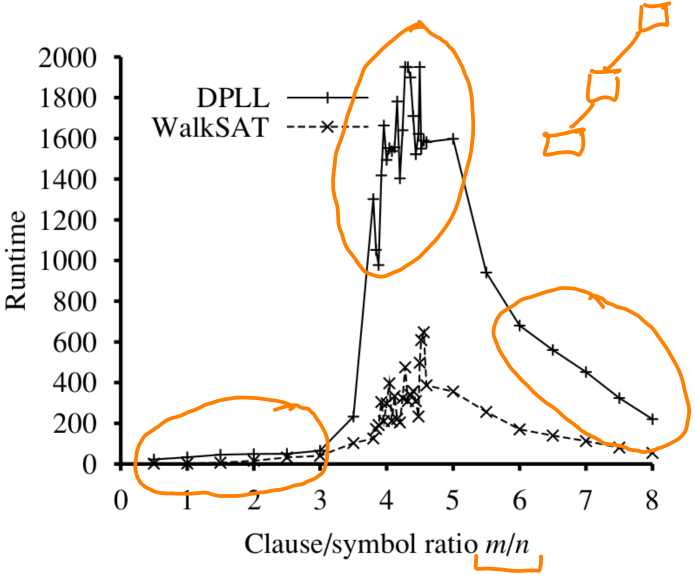
$$(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$$

- Underconstrained SAT problem
- $CNF_k(m, n)$
- $CNF_3(m, 50)$ ↑   n = 50   3-CNF

Satisfiability of Random SAT Problems



Where are the hard problems?



Representing the state of the world

- Background knowledge

Representing the state of the world

- Background knowledge

  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

  $S_{1,1} \Leftrightarrow (W_{1,2} \vee P_{W,1})$ $W_{2,1}$

  ...

Representing the state of the world

$S_{1,1}$

- Background knowledge

  $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
  $S_{1,1} \Leftrightarrow (W_{1,2} \vee P_{W,1})$
  ...

- Exactly one Wumpus

$W_{1,1} \vee W_{2,1} \vee W_{1,2} \vee \ldots$

$W_{1,1} \Rightarrow \neg W_{2,1}$ $\neg W_{1,1} \vee \neg W_{2,1}$

$W_{1,1} \Rightarrow \neg W_{1,2}$

Representing the state of the world

- Background knowledge

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$S_{1,1} \Leftrightarrow (W_{1,2} \vee P_{W,1})$$
$$\ldots$$

- Exactly one Wumpus

$$W_{1,1} \vee W_{1,2} \vee \ldots \vee W_{4,3} \vee W_{4,4}$$
$$\neg W_{1,1} \vee \neg W_{1,2}$$
$$\neg W_{1,1} \vee \neg W_{1,3}$$
$$\ldots$$

Representing the state of the world

- Fluent (or Temporal) variables

$WumpusAlive^0$

Representing the state of the world

- Fluent (or Temporal) variables
  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.

HaveArrow^0

HaveArrow^5

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.

  $W_{2,1}$ $P_{3,1}$

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

$Forward^0$

$L^0_{1,1}$

$L^1_{1,1}$

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.

- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$

Representing the state of the world

- Fluent (or Temporal) variables

  FacingEast^0, *HaveArrow*^{0}, *WumpusAlive*^{0} etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like *Forward*^{0}.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.

- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$

- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1})$

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1}) =$ ~~True~~ False

$KB \models \neg L^1_{2,1}$

$KB \wedge \neg \alpha$

Representing the state of the world

- Fluent (or Temporal) variables
  FacingEast^0, *HaveArrow*^{0}, *WumpusAlive*^{0} etc.
- Atemporal variables.
- Effect axioms
  Describe effects of actions like *Forward*^{0}.
  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1}) = True$
  - $Ask(KB, HaveArrow^1)$

Have Arrow^0

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1}) = True$
  - $Ask(KB, HaveArrow^1) = False$

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1}) = True$
  - $Ask(KB, HaveArrow^1) = False$
- Frame Problem

Representing the state of the world

- Fluent (or Temporal) variables

  $FacingEast^0$, $HaveArrow^0$, $WumpusAlive^0$ etc.
- Atemporal variables.
- Effect axioms

  Describe effects of actions like $Forward^0$.

  $L^0_{1,1} \wedge FacingEast^0 \wedge Forward^0 \Rightarrow (L^1_{2,1} \wedge \neg L^1_{1,1})$
- Suppose we make the following queries:
  - $Ask(KB, L^1_{2,1}) = True$
  - $Ask(KB, HaveArrow^1) = False$
- Frame Problem
- Can the following sentence fix the frame problem?

$HaveArrow^t \wedge Forward^t \Rightarrow HaveArrow^{t+1}$

Forward⁰ = F

⇒ HaveArrow¹

Solving the Frame problem

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB?

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB? $m \times n$

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB? $m \times n$

2. Successor-state axioms

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB? $m \times n$

2. Successor-state axioms

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB? $m \times n$

2. Successor-state axioms

$$F^{t+1} \Leftrightarrow ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$$
$$HaveArrow^{t+1} \Leftrightarrow ReloadArrow^t \vee (HaveArrow^t \wedge \neg Shoot^t)$$

$$\begin{aligned} L_{1,1}^{t+1} \quad \Leftrightarrow \quad & (L_{1,1}^t \wedge (\neg Forward^t \vee Bump^{t+1})) \\ \vee \quad & (L_{1,2}^t \wedge (South^t \wedge Forward^t)) \\ \vee \quad & (L_{2,1}^t \wedge (West^t \wedge Forward^t)) \,. \end{aligned}$$

Solving the Frame problem

1. Frame axioms

$$Forward^t \Rightarrow (HaveArrow^t \Leftrightarrow HaveArrow^{t+1})$$
$$Forward^t \Rightarrow (WumpusAlive^t \Leftrightarrow WumpusAlive^{t+1})$$
$$\ldots$$

- If there are m actions and n fluent variables, then how many frame axioms should we add to KB? $m \times n$

2. Successor-state axioms

$$F^{t+1} \Leftrightarrow ActionCausesF^t \lor (F^t \land \neg ActionCausesNotF^t)$$
$$HaveArrow^{t+1} \Leftrightarrow ReloadArrow^t \lor (HaveArrow^t \land \neg Shoot^t)$$

$$\begin{aligned} L_{1,1}^{t+1} \quad \Leftrightarrow \quad & (L_{1,1}^t \land (\neg Forward^t \lor Bump^{t+1})) \\ \lor \quad & (L_{1,2}^t \land (South^t \land Forward^t)) \\ \lor \quad & (L_{2,1}^t \land (West^t \land Forward^t)) . \end{aligned}$$

- Axioms are templates for new variables.

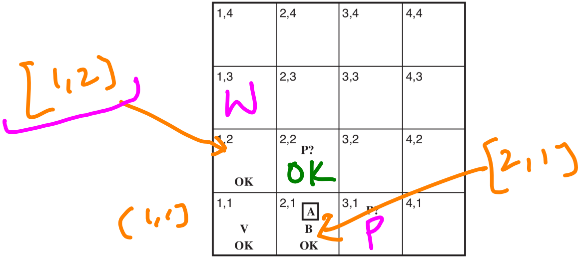
Action exclusion axioms

- We need to add additional sentences to ensure that only one action can be taken at each time step. What should these sentences be?

- $W_{1,1} \Rightarrow \neg W_{2,1}$

  $A_i^t \Rightarrow \neg A_j^t$

Queries about the Current State



$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \quad ; \quad Forward^0$$

$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \quad ; \quad TurnRight^1$$

$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \quad ; \quad TurnRight^2$$

$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \quad ; \quad Forward^3$$

$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \quad ; \quad TurnRight^4$$

$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \quad ; \quad Forward^5$$

$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

Queries about the Current State

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \; ; \; Forward^0$$

$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \; ; \; TurnRight^1$$

$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \; ; \; TurnRight^2$$

$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \; ; \; Forward^3$$

$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \; ; \; TurnRight^4$$

$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \; ; \; Forward^5$$

$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

$Ask(KB, L^6_{1,2})$

Queries about the Current State

$$\neg Stench^0 \land \neg Breeze^0 \land \neg Glitter^0 \land \neg Bump^0 \land \neg Scream^0 \;\; ; \;\; Forward^0$$
$$\neg Stench^1 \land Breeze^1 \land \neg Glitter^1 \land \neg Bump^1 \land \neg Scream^1 \;\; ; \;\; TurnRight^1$$
$$\neg Stench^2 \land Breeze^2 \land \neg Glitter^2 \land \neg Bump^2 \land \neg Scream^2 \;\; ; \;\; TurnRight^2$$
$$\neg Stench^3 \land Breeze^3 \land \neg Glitter^3 \land \neg Bump^3 \land \neg Scream^3 \;\; ; \;\; Forward^3$$
$$\neg Stench^4 \land \neg Breeze^4 \land \neg Glitter^4 \land \neg Bump^4 \land \neg Scream^4 \;\; ; \;\; TurnRight^4$$
$$\neg Stench^5 \land \neg Breeze^5 \land \neg Glitter^5 \land \neg Bump^5 \land \neg Scream^5 \;\; ; \;\; Forward^5$$
$$Stench^6 \land \neg Breeze^6 \land \neg Glitter^6 \land \neg Bump^6 \land \neg Scream^6$$

$$Ask(KB, L^6_{1,2}) = True,$$

Queries about the Current State

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \ ; \ Forward^0$$
$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \ ; \ TurnRight^1$$
$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \ ; \ TurnRight^2$$
$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \ ; \ Forward^3$$
$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \ ; \ TurnRight^4$$
$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \ ; \ Forward^5$$
$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

$$Ask(KB, L^6_{1,2}) = True, \ Ask(KB, W_{1,3})$$

Queries about the Current State

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \;\; ; \;\; Forward^0$$
$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \;\; ; \;\; TurnRight^1$$
$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \;\; ; \;\; TurnRight^2$$
$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \;\; ; \;\; Forward^3$$
$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \;\; ; \;\; TurnRight^4$$
$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \;\; ; \;\; Forward^5$$
$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

$Ask(KB, L^6_{1,2}) = True$, $Ask(KB, W_{1,3}) = True$,

Queries about the Current State

$$\neg Stench^0 \land \neg Breeze^0 \land \neg Glitter^0 \land \neg Bump^0 \land \neg Scream^0 \;\; ; \;\; Forward^0$$
$$\neg Stench^1 \land Breeze^1 \land \neg Glitter^1 \land \neg Bump^1 \land \neg Scream^1 \;\; ; \;\; TurnRight^1$$
$$\neg Stench^2 \land Breeze^2 \land \neg Glitter^2 \land \neg Bump^2 \land \neg Scream^2 \;\; ; \;\; TurnRight^2$$
$$\neg Stench^3 \land Breeze^3 \land \neg Glitter^3 \land \neg Bump^3 \land \neg Scream^3 \;\; ; \;\; Forward^3$$
$$\neg Stench^4 \land \neg Breeze^4 \land \neg Glitter^4 \land \neg Bump^4 \land \neg Scream^4 \;\; ; \;\; TurnRight^4$$
$$\neg Stench^5 \land \neg Breeze^5 \land \neg Glitter^5 \land \neg Bump^5 \land \neg Scream^5 \;\; ; \;\; Forward^5$$
$$Stench^6 \land \neg Breeze^6 \land \neg Glitter^6 \land \neg Bump^6 \land \neg Scream^6$$

$Ask(KB, L^6_{1,2}) = True$, $Ask(KB, W_{1,3}) = True$,
$Ask(KB, P_{3,1})$

Queries about the Current State

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \;;\; Forward^0$$

$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \;;\; TurnRight^1$$

$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \;;\; TurnRight^2$$

$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \;;\; Forward^3$$

$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \;;\; TurnRight^4$$

$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \;;\; Forward^5$$

$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

$Ask(KB, L^6_{1,2}) = True$, $Ask(KB, W_{1,3}) = True$,
$Ask(KB, P_{3,1}) = True$,

Queries about the Current State

$$\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 \;;\; Forward^0$$
$$\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 \;;\; TurnRight^1$$
$$\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 \;;\; TurnRight^2$$
$$\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 \;;\; Forward^3$$
$$\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 \;;\; TurnRight^4$$
$$\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 \;;\; Forward^5$$
$$Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6$$

$Ask(KB, L^6_{1,2}) = True$, $Ask(KB, W_{1,3}) = True$,
$Ask(KB, P_{3,1}) = True$,

$$OK^t_{x,y} \Leftrightarrow \neg P_{x,y} \wedge \neg(W_{x,y} \wedge WumpusAlive^t)$$

Queries about the Current State

$$\begin{array}{ll}
\neg Stench^0 \wedge \neg Breeze^0 \wedge \neg Glitter^0 \wedge \neg Bump^0 \wedge \neg Scream^0 & ; \; Forward^0 \\
\neg Stench^1 \wedge Breeze^1 \wedge \neg Glitter^1 \wedge \neg Bump^1 \wedge \neg Scream^1 & ; \; TurnRight^1 \\
\neg Stench^2 \wedge Breeze^2 \wedge \neg Glitter^2 \wedge \neg Bump^2 \wedge \neg Scream^2 & ; \; TurnRight^2 \\
\neg Stench^3 \wedge Breeze^3 \wedge \neg Glitter^3 \wedge \neg Bump^3 \wedge \neg Scream^3 & ; \; Forward^3 \\
\neg Stench^4 \wedge \neg Breeze^4 \wedge \neg Glitter^4 \wedge \neg Bump^4 \wedge \neg Scream^4 & ; \; TurnRight^4 \\
\neg Stench^5 \wedge \neg Breeze^5 \wedge \neg Glitter^5 \wedge \neg Bump^5 \wedge \neg Scream^5 & ; \; Forward^5 \\
Stench^6 \wedge \neg Breeze^6 \wedge \neg Glitter^6 \wedge \neg Bump^6 \wedge \neg Scream^6 &
\end{array}$$

$Ask(KB, L^6_{1,2}) = True$, $Ask(KB, W_{1,3}) = True$,
$Ask(KB, P_{3,1}) = True$,

$OK^t_{x,y} \Leftrightarrow \neg P_{x,y} \wedge \neg(W_{x,y} \wedge WumpusAlive^t)$

$Ask(KB, OK^6_{2,2})$?

ASK(KB,α)

- When is it True? $KB \models \alpha$

α

ASK(KB,α)

$KB \not\models \alpha$

- When is it True?
- When is it False?

$\neg\alpha$

Hybrid Wumpus Agent

Inference in Wumpus World

- Need temporal variables $HaveArrow^t$, $WumpusAlive^t$ etc.

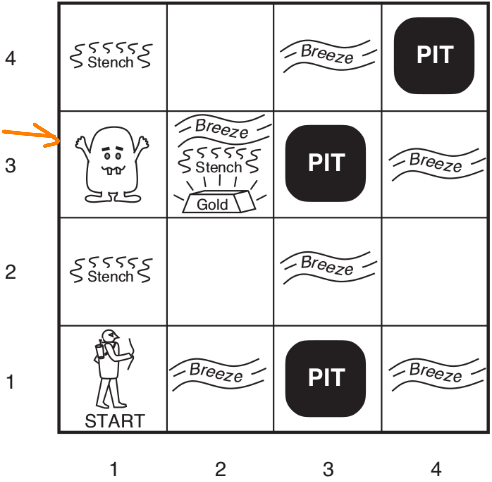
Inference in Wumpus World

- Need temporal variables $HaveArrow^t$, $WumpusAlive^t$ etc.
- Effect axioms

Inference in Wumpus World

- Need temporal variables $HaveArrow^t$, $WumpusAlive^t$ etc.
- Effect axioms
- Successor state axioms to address the frame problem

Making Plans by Propositional Inference



Making Plans by Propositional Inference

- Planning vs. Inference
  - Fully observable environment

Making Plans by Propositional Inference

- Planning vs. Inference
  - Fully observable environment
  - Satisfiability

Making Plans by Propositional Inference

- Planning vs. Inference
  - Fully observable environment
  - Satisfiability
- KB

$$L^0_{1,1}$$
$$L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$$
$$L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$$

Making Plans by Propositional Inference

- Planning vs. Inference
  - Fully observable environment
  - Satisfiability
- KB

  $L^0_{1,1}$

  $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$

  $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
- Goal: $L^2_{1,3}$

$L^2_{1,3} = T$

Making Plans by Propositional Inference

- Planning vs. Inference
  - Fully observable environment
  - Satisfiability
- KB

$$L^0_{1,1}$$
$$L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$$
$$L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$$

- ~~Goal: $L^2_{1,3}$~~
- Goal: $L^t_{1,3}$ ~~$L^0_{1,3}$~~ $L^1_{1,3}$

Making Plans by Propositional Inference

1. Construct a sentence that includes
   (a) $Init^0$, a collection of assertions about the initial state;
   (b) $Transition^1, \ldots, Transition^t$, the successor-state axioms for all possible actions at each time up to some maximum time t;
   (c) the assertion that the goal is achieved at time t: $HaveGold^t \wedge ClimbedOut^t$.
2. Present the whole sentence to a SAT solver. If the solver finds a satisfying model, then the goal is achievable; if the sentence is unsatisfiable, then the planning problem is impossible.
3. Assuming a model is found, extract from the model those variables that represent actions and are assigned *true*. Together they represent a plan to achieve the goals.

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans?

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$

  $KB \models L^1_{1,3}$

  Possible assignment: $\ldots, L^1_{1,2} = True, L^1_{1,3} = True, \ldots$

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$
  Possible assignment: $\ldots, L^1_{1,2} = True, L^1_{1,3} = True, \ldots$
- (Not a problem if we want to check whether $KB \models L^1_{1,3}$.)

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$

  Possible assignment: $\ldots, L^1_{1,2} = True, L^1_{1,3} = True, \ldots$
- (Not a problem if we want to check whether $KB \models L^1_{1,3}$.)
- Location Exclusion Axioms

$\neg L^t_{1,2} \vee \neg L^t_{1,3}$ $\quad (L^t_{1,2} \Rightarrow \neg L^t_{1,3})$

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$
  Possible assignment: $\ldots, L^1_{1,2} = True, L^1_{1,3} = True, \ldots$
- (Not a problem if we want to check whether $KB \models L^1_{1,3}$.)
- Location Exclusion Axioms

  Another assignment:
  $\ldots, Shoot^0 = True, Forward^0 = True, \ldots$

Making Plans

- Suppose we have Effect axioms and Successor state axioms. Can we come up with valid plans? No.
- R1. $L^0_{1,1}$
  R2. $L^0_{1,1} \wedge Forward^0 \Leftrightarrow L^1_{1,2}$
  R3. $L^1_{1,2} \wedge Forward^1 \Leftrightarrow L^2_{1,3}$
  Goal. $L^1_{1,3}$

  Possible assignment: $\ldots, L^1_{1,2} = True, L^1_{1,3} = True, \ldots$
- (Not a problem if we want to check whether $KB \models L^1_{1,3}$.)
- Location Exclusion Axioms

  Another assignment:
  $\ldots, Shoot^0 = True, Forward^0 = True, \ldots$
- Action Exclusion Axioms
  $\neg A^t_i \vee \neg A^t_j$

Making Plans

- Successor state axioms

Making Plans

- Successor state axioms

$$HaveArrow^{t+1} \Leftrightarrow ReloadArrow^{t} \lor (HaveArrow^{t} \land \neg Shoot^{t})$$

Shoot² = True

HaveArrow² = False

Making Plans

- Successor state axioms

  $HaveArrow^{t+1} \Leftrightarrow ReloadArrow^{t} \vee (HaveArrow^{t} \wedge \neg Shoot^{t})$
- Precondition axioms

Making Plans

- Successor state axioms

  $HaveArrow^{t+1} \Leftrightarrow ReloadArrow^{t} \vee (HaveArrow^{t} \wedge \neg Shoot^{t})$

- Precondition axioms

  $Shoot^{t} \Rightarrow HaveArrow^{t}$

function SATPLAN(*init*, *transition*, *goal*, $T_{\max}$) **returns** solution or failure
  inputs: *init*, *transition*, *goal*, constitute a description of the problem
    $T_{\max}$, an upper limit for plan length

  for $t = 0$ **to** $T_{\max}$ **do**
    cnf ← TRANSLATE-TO-SAT(*init*, *transition*, *goal*, *t*)
    model ← SAT-SOLVER(*cnf*)
    if *model* is not null **then**
      return EXTRACT-SOLUTION(*model*)
  return *failure*

Figure 7.22 The SATPLAN algorithm. The planning problem is translated into a CNF sentence in which the goal is asserted to hold at a fixed time step t and axioms are included for each time step up to t. If the satisfiability algorithm finds a model, then a plan is extracted by looking at those proposition symbols that refer to actions and are assigned *true* in the model. If no model exists, then the process is repeated with the goal moved one step later.

Representational Languages

Desirable properties of a representational language:

- Domain independent knowledge representation

Representational Languages

Desirable properties of a representational language:

- Domain independent knowledge representation
- Inferencing

Representational Languages

Desirable properties of a representational language:

- Domain independent knowledge representation
- Inferencing
- Compositionality

Representational Languages

Desirable properties of a representational language:

- Domain independent knowledge representation
- Inferencing
- Compositionality

First-order Logic:

- More concise compared to PL

Representational Languages

Desirable properties of a representational language:

- Domain independent knowledge representation
- Inferencing
- Compositionality

First-order Logic:

- More concise compared to PL
- More expressive compared to PL

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?
- In PL and FOL, symbols have precise meaning.

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?
- In PL and FOL, symbols have precise meaning.
- Natural language is ambiguous.

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?
- In PL and FOL, symbols have precise meaning.
- Natural language is ambiguous.

Eg. *Most people are shocked when they find out how bad I am as an electrician.*

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?
- In PL and FOL, symbols have precise meaning.
- Natural language is ambiguous.

Eg. *Most people are shocked when they find out how bad I am as an electrician.*

- Can all human thoughts be expressed in a natural language?

Comparisons with natural language and human thought

- Can natural language sentences be represented using PL or first-order logic?

8.1

- In PL and FOL, symbols have precise meaning.
- Natural language is ambiguous.

Eg. *Most people are shocked when they find out how bad I am as an electrician.*

- Can all human thoughts be expressed in a natural language?
  - Without (natural) language there can be no thought.
  - Language is inessential for thought. (Language evolved *for* thought.)

First-order Logic

- Some domain or universe.

objects

First-order Logic

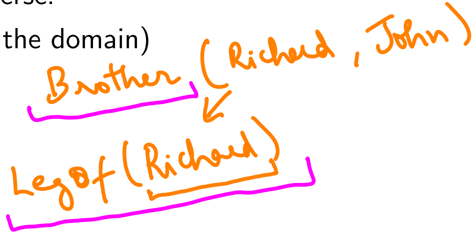
- Some domain or universe.
- Objects (elements of the domain)

First-order Logic

- Some domain or universe.
- Objects (elements of the domain)
- Relations

First-order Logic

- Some domain or universe.
- Objects (elements of the domain)
- Relations
- Functions



First-order Logic Example

OnHead(Crown, John)

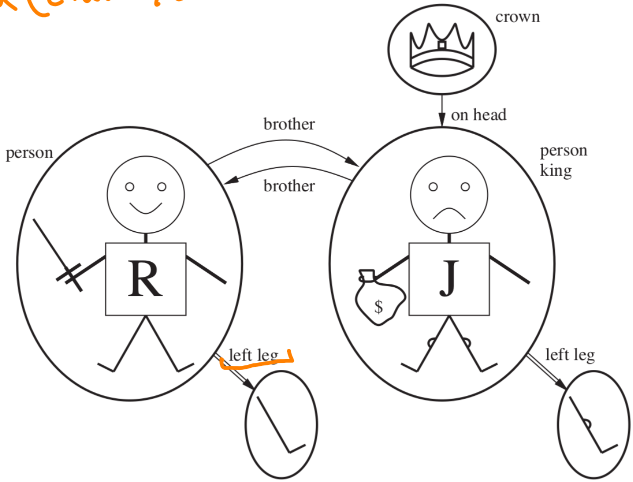


Figure 8.2 A model containing five objects, two binary relations, three unary relations (indicated by labels on the objects), and one unary function, left-leg.

Syntax of First-order Logic

- Defined relative to a *signature*.
- A signature σ consists of:

1. A set of *constant symbols*
2. A set of *predicate symbols*
3. A set of *function symbols*
4. Each function and predicate symbol has an *arity* $k > 0$

Richard →
Crown →
Brother(·, ·)
OnHead(·, ·)
LefLeg(·)

Semantics of First-order Logic

- We are refering to the standard FOL semantics.
- A model (or structure or assignment) consists of:

1. A non-empty set U called the *universe* (or the domain) of the structure.
2. Each k-ary predicate symbol is mapped to a k-ary relation.
3. Each k-ary function symbol is mapped to a k-ary function.
4. Each constant symbol is mapped to an element of the universe.
5. Existentially quantified variable is mapped to an element of the universe.

Example

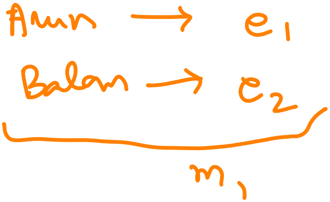
KB_1 : R1. $Male(Arun)$
R2. $Male(Balan)$

Example

KB_1 : R1. $Male(Arun)$
R2. $Male(Balan)$

- Does $KB_1 \models (Arun = Balan)$?

$m(KB_1) \subseteq m(\alpha)$



Example

KB_1 : R1. $Male(Arun)$
R2. $Male(Balan)$

- Does $KB_1 \models Arun = Balan$?
- Does $KB_1 \models \neg(Arun = Balan)$?

Arun → e_1
Balan → e_1
Male( ) = { e_1, e_2 ... }

First-order Logic: Inference

KB: $Brother(Richard, John)$
$OnHead(Crown, John)$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$

$m(KB) \subseteq m(\alpha)$

KB: $Brother(Richard, John)$
$OnHead(Crown, John)$
$\forall x, y \; Brother(x, y) \Rightarrow \neg(x = y)$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$

First-order Logic: Inference

KB: $Brother(Richard, John)$
$OnHead(Crown, John)$
$\forall x, y \; Brother(x, y) \Rightarrow \neg(x = y)$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$
$$KB \models \neg Brother(Crown, John)$$

First-order Logic: Inference

KB:

$$Brother(Richard, John)$$
$$OnHead(Crown, John)$$
$$\forall x, y \; Brother(x, y) \Rightarrow \neg(x = y)$$
$$\forall x, y \; Brother(x, y) \Rightarrow Person(x) \wedge Person(y)$$
$$\forall x, y \; OnHead(x, y) \Rightarrow \neg Person(x) \wedge Person(y)$$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$
$$KB \models \neg Brother(Crown, John)$$

First-order Logic: Inference

KB:

$$Brother(Richard, John)$$
$$OnHead(Crown, John)$$
$$\forall x, y \;\; Brother(x, y) \Rightarrow \neg(x = y)$$
$$\forall x, y \;\; Brother(x, y) \Rightarrow Person(x) \wedge Person(y)$$
$$\forall x, y \;\; OnHead(x, y) \Rightarrow \neg Person(x) \wedge Person(y)$$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$
$$KB \models \neg Brother(Crown, John)$$
$$KB \models \neg OnHead(Crown, Richard)$$

First-order Logic: Inference

KB:
$$Brother(Richard, John)$$
$$OnHead(Crown, John)$$
$$\forall x, y \; Brother(x, y) \Rightarrow \neg(x = y)$$
$$\forall x, y \; Brother(x, y) \Rightarrow Person(x) \wedge Person(y)$$
$$\forall x, y \; OnHead(x, y) \Rightarrow \neg Person(x) \wedge Person(y)$$
$$\forall x, y \; OnHead(Crown, x) \wedge OnHead(Crown, y) \Rightarrow x = y$$

- Does the following entailment hold?

$$KB \models \neg(Richard = John)$$
$$KB \models \neg Brother(Crown, John)$$
$$KB \models \neg OnHead(Crown, Richard)$$

First-order Logic: Syntax

Universal and Existential Quantifiers

- *All kings are persons.*

Universal and Existential Quantifiers

- *All kings are persons.*
  1. $\forall x \; King(x) \land Person(x)$

Universal and Existential Quantifiers

- *All kings are persons.*
  1. $\forall x\, King(x) \wedge Person(x)$
  2. $\forall x\, King(x) \Rightarrow Person(x)$

Universal and Existential Quantifiers

- *All kings are persons.*
  1. $\forall x\, King(x) \wedge Person(x)$
  2. $\forall x\, King(x) \Rightarrow Person(x)$
- *There is a person who has a crown on his/her head.*

Universal and Existential Quantifiers

- *All kings are persons.*
  1. $\forall x\, King(x) \wedge Person(x)$
  2. $\forall x\, King(x) \Rightarrow Person(x)$
- *There is a person who has a crown on his/her head.*
  1. $\exists x\, Person(x) \wedge OnHead(Crown, x)$

Universal and Existential Quantifiers

- *All kings are persons.*
  1. $\forall x\, King(x) \wedge Person(x)$
  2. $\forall x\, King(x) \Rightarrow Person(x)$
- *There is a person who has a crown on his/her head.*
  1. $\exists x\, Person(x) \wedge OnHead(Crown, x)$
  2. $\exists x\, Person(x) \Rightarrow OnHead(Crown, x)$

Nested Quantifiers

- *Everybody loves someone.*

Nested Quantifiers

- *Everybody loves someone.*

  $\forall x\ \exists y\ Loves(x, y)$

Nested Quantifiers

- *Everybody loves someone.*

  $\forall x \, \exists y \; Loves(x, y)$

  $\exists y \, \forall x \; Loves(x, y)$

Nested Quantifiers

- *Everybody loves someone.*

  $\forall x \, \exists y \; Loves(x, y)$

  $\exists y \, \forall x \; Loves(x, y)$

- *There is someone who is loved by everybody.*

Connections between ∃ and ∀

- *Everybody loves Icecream.*

Connections between $\exists$ and $\forall$

- *Everybody loves Icecream.*

  $\forall x\ Loves(x, Icecream)$

Connections between ∃ and ∀

- *Everybody loves Icecream.*

  $\forall x \, Loves(x, Icecream)$

  $\neg\exists x \, \neg Loves(x, Icecream)$

Connections between ∃ and ∀

- *Everybody loves Icecream.*

  $\forall x \; Loves(x, Icecream)$

  $\neg \exists x \; \neg Loves(x, Icecream)$

- More generally

Connections between $\exists$ and $\forall$

- *Everybody loves Icecream.*

  $\forall x\, Loves(x, Icecream)$

  $\neg\exists x\, \neg Loves(x, Icecream)$

- More generally

  $\forall x\, P \equiv \neg\exists x\, \neg P$

Connections between ∃ and ∀

- *Everybody loves Icecream.*

  $\forall x\, Loves(x, Icecream)$

  $\neg\exists x\, \neg Loves(x, Icecream)$

- More generally

  $\forall x\, P \equiv \neg\exists x\, \neg P$

  $\exists x\, P \equiv \neg\forall x\, \neg P$

First order logic sentences

- $\forall y\, P(x, y)$
  The above is a first order logic formula where x is a free variable and y is a bound variable.

First order logic sentences

- $\forall y\, P(x, y)$
  The above is a first order logic formula where x is a free variable and y is a bound variable.
- An FOL sentence is a formula with no free variables.

First order logic sentences

- $\forall y \, P(x, y)$
  The above is a first order logic formula where x is a free variable and y is a bound variable.
- An FOL sentence is a formula with no free variables.
- We will be constructing a KB using FOL sentences that represents the relevant facts.

Queries in FOL

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

- $KB \models Person(John)$

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

- $KB \models Person(John)$
- $Ask(KB, Person(John))$

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

- $KB \models Person(John)$
- $Ask(KB, Person(John))$
- $AskVars(KB, Person(x))$

  Two answers: $\{x/John\}$ and $\{x/Richard\}$

$KB \models Person(x)$

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

- $KB \models Person(John)$
- $Ask(KB, Person(John))$
- $AskVars(KB, Person(x))$

  Two answers: $\{x/John\}$ and $\{x/Richard\}$
  (Substitution or Binding list)

Queries in FOL

KB: $King(John)$
$King(Richard)$
$\forall x King(x) \Rightarrow Person(x)$

- $KB \models Person(John)$
- $Ask(KB, Person(John))$
- $AskVars(KB, Person(x))$

  Two answers: $\{x/John\}$ and $\{x/Richard\}$
  (Substitution or Binding list)
- Knowledge representation in kinship domain (Section 8.3.2)

Inference: Propositionalization

KB:

$King(John)$
$King(Richard)$
$Greedy(John)$
~~$\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$~~

$\rightarrow K(J) \land G(J) \Rightarrow E(J) \leftarrow$
$\rightarrow K(R) \land G(R) \Rightarrow E(R) \leftarrow$
} K_2

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\, King(x) \land Greedy(x) \Rightarrow Evil(x)$

$$KB \models Evil(John)?$$

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)$?

- Universal instantiation
  - Ground term

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)?$

- Universal instantiation
  - Ground term
  - Substitution :

$$\frac{\forall x\ \alpha}{Subst(\{x/g\}, \alpha)}$$

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\; King(x) \land Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)$?

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\; \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\ King(x) \land Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)?$

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\ \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation
  - $\exists x\ Crown(x) \land OnHead(x, John)$

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)?$

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\; \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation
  - $\exists x\; Crown(x) \wedge OnHead(x, John)$
  - $\dfrac{\exists x\; \alpha}{Subst(\{x/C\}, \alpha)}$

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\ King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)$?

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\ \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation
  - $\exists x\ Crown(x) \wedge OnHead(x, John)$ (1)
  - $\dfrac{\exists x\ \alpha}{Subst(\{x/C\}, \alpha)}$
  - $Crown(C_1) \wedge OnHead(C_1, John)$ (2)

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\, King(x) \land Greedy(x) \Rightarrow Evil(x)$

KB₂

$KB \models Evil(John)?$

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\ \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation
  - $\exists x\, Crown(x) \land OnHead(x, John)$
  - $\dfrac{\exists x\ \alpha}{Subst(\{x/C\}, \alpha)}$
  - $Crown(C_1) \land OnHead(C_1, John)$
  - Skolemization, skolem constant

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$
- Therefore, $M(K_2) \subseteq M(\alpha)$. $K_2 \models \alpha$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$
- Therefore, $M(K_2) \subseteq M(\alpha)$.
- We say K_1 is Inferentially Equivalent to K_2.

$K_2 \models \alpha$

Inference: Propositionalization

KB: $King(John)$
$King(Richard)$
$Greedy(John)$
$\forall x\, King(x) \land Greedy(x) \Rightarrow Evil(x)$

$KB \models Evil(John)$?

- Universal instantiation
  - Ground term
  - Substitution : $\dfrac{\forall x\ \alpha}{Subst(\{x/g\}, \alpha)}$
- Existential instantiation
  - $\exists x\, Crown(x) \land OnHead(x, John)$
  - $\dfrac{\exists x\ \alpha}{Subst(\{x/C\}, \alpha)}$
  - $Crown(C_1) \land OnHead(C_1, John)$
  - Skolemization, skolem constant

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$ (1)
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$ (2)
- Therefore, $M(K_2) \subseteq M(\alpha)$

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$
- Therefore, $M(K_2) \subseteq M(\alpha)$ and $K_2 \models \alpha$.

Inferentially Equivalent

- Suppose we obtain K_2 from K_1.
- If some model m satisfies K_2, then we are sure that m will satisfy K_1.
- So, $M(K_2) \subseteq M(K_1)$
- Now, suppose $K_1 \models \alpha$. Then $M(K_1) \subseteq M(\alpha)$
- Therefore, $M(K_2) \subseteq M(\alpha)$ and $K_2 \models \alpha$.
- So, instead of checking whether $K_1 \models \alpha$ we can check whether $K_2 \models \alpha$.

Propositionalization

- Functions can lead to infinite number of ground terms.

Propositionalization

- Functions can lead to infinite number of ground terms.

  FatherOf(*Richard*), *FatherOf*(*FatherOf*(*Richard*)) etc.

Propositionalization

- Functions can lead to infinite number of ground terms. $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.

Propositionalization

- Functions can lead to infinite number of ground terms.
  $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.
  $\forall x\, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

Propositionalization

- Functions can lead to infinite number of ground terms.
  $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.
  $\forall x\, King(x) \land Greedy(x) \Rightarrow Evil(x)$
- We can iteratively increase the depth of nested ground terms to check whether $KB \models \alpha$.

Propositionalization

- Functions can lead to infinite number of ground terms.
  $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.
  $\forall x\, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- We can iteratively increase the depth of nested ground terms to check whether $KB \models \alpha$.
- Is the algorithm sound?

Propositionalization

- Functions can lead to infinite number of ground terms.
  $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.
  $\forall x\, King(x) \land Greedy(x) \Rightarrow Evil(x)$
- We can iteratively increase the depth of nested ground terms to check whether $KB \models \alpha$.
- Is the algorithm sound? complete?

Propositionalization

- Functions can lead to infinite number of ground terms.

  $FatherOf(Richard)$, $FatherOf(FatherOf(Richard))$ etc.
- Therefore, universal instantiation can generate infinite number of sentences.

  $\forall x \, King(x) \wedge Greedy(x) \Rightarrow Evil(x)$
- We can iteratively increase the depth of nested ground terms to check whether $KB \models \alpha$.
- Is the algorithm sound? complete?
- Inferencing in FOL is semidecidable.

Unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

Unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

$$Unify(Knows(J, x), Knows(J, A)) \quad \{x | A\}$$

Unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

$$Unify(Knows(J, x), Knows(J, A)) = \{x/A\}$$

Unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

$$Unify(Knows(J, x), Knows(J, A)) = \underbrace{\{x/A\}}_{\text{unifier}}$$

Unification

$$\text{UNIFY}(p, q) = \theta \text{ where } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)$$

$$\mathit{Unify}(\mathit{Knows}(J, x), \mathit{Knows}(J, A)) = \underbrace{\{x/A\}}_{\text{unifier}}$$

- Most general unifier:

  $\mathit{Unify}(\mathit{Knows}(J, x), \mathit{Knows}(y, z))$

Unification

$$\text{UNIFY}(p,q) = \theta \text{ where } \text{SUBST}(\theta,p) = \text{SUBST}(\theta,q)$$

$$Unify(Knows(J,x), Knows(J,A)) = \underbrace{\{x/A\}}_{\text{unifier}}$$

- Most general unifier:

$$Unify(Knows(J,x), Knows(y,z))$$
$$= \{y/J, x/J, z/J\}$$

$$\text{UNIFY}(p,q) = \theta \text{ where } \text{SUBST}(\theta,p) = \text{SUBST}(\theta,q)$$

$$Unify(Knows(J,x), Knows(J,A)) = \underbrace{\{x/A\}}_{\text{unifier}}$$

- Most general unifier:

$$Unify(Knows(J,x), Knows(y,z))$$
$$= \{y/J, x/J, z/J\}$$
$$= \{y/J, x/z\} \text{ (Most general unifier)}$$

Unification

$$\text{UNIFY}(p,q)=\theta \text{ where } \text{SUBST}(\theta,p)=\text{SUBST}(\theta,q)$$

$$Unify(Knows(J,x), Knows(J,A)) = \underbrace{\{x/A\}}_{\text{unifier}}$$

- Most general unifier:

  $Unify(Knows(J,x), Knows(y,z))$

  $= \{y/J, x/J, z/J\}$

  $= \{y/J, x/z\}$ (Most general unifier)

- We have polynomial time algorithms that can find a unifier (if one exists) for two expressions.

Assumptions

Resolution Algorithm for FOL

Assumptions

- Only universal quantifiers

Resolution Algorithm for FOL

Assumptions

- Only universal quantifiers
- Sentences in CNF form

Skolemization

- *Everyone is loved by someone.*

Skolemization

- *Everyone is loved by someone.*

  $\forall x \, \exists y \; Loves(y, x)$

Skolemization

- *Everyone is loved by someone.*
  $\forall x \exists y \, Loves(y, x)$
- After skolemization:

Skolemization

- *Everyone is loved by someone.*

  $\forall x \exists y \, Loves(y, x)$
- After skolemization:

  $\forall x \, Loves(C_1, x)$

Skolemization

- *Everyone is loved by someone.*

  $\forall x \exists y \, Loves(y, x)$

- After skolemization:

  ~~$\forall x \, Loves(C_1, x)$~~ (wrong!)

Skolemization

- *Everyone is loved by someone.*

  $\forall x \exists y \; Loves(y, x)$

- After skolemization:

  ~~$\forall x \; Loves(C_1, x)$~~ (wrong!)

- Skolem function:

  $\forall x \; Loves(F(x), x)$

Skolemization

- *Everyone is loved by someone.*

  $\forall x \exists y \, Loves(y, x)$
- After skolemization:

  ~~$\forall x \, Loves(C_1, x)$~~ (wrong!)
- Skolem function:

  $\forall x \, Loves(F(x), x)$
-

  $Loves(F(x), x)$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$
2. $\forall x\, [\forall y\, Animal(y) \land Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$
2. $\forall x\, [\forall y\, Animal(y) \land Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

- Sentence 2. will always be True if there is a y such that $\neg Animal(y)$ is True.

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall\, x\, [\forall\, y\; Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists\, z\; Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall\, x\, [\forall\, y\ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists\, z\ Loves(z, x)]$

   $\forall\, x\, \neg[\forall\, y\ Animal(y) \Rightarrow Loves(x, y)] \vee [\exists\, z\ Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, Animal(y) \Rightarrow Loves(x, y)] \lor [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, \neg Animal(y) \lor Loves(x, y)] \lor [\exists z\, Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$
   $\forall x\, \neg[\forall y\, Animal(y) \Rightarrow Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, \neg Animal(y) \vee Loves(x, y)] \vee [\exists z\, Loves(z, x)]$
   $\forall x\, [\exists y\, Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, Animal(y) \Rightarrow Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, \neg Animal(y) \vee Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, [\exists y\, Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   Skolem constant or skolem function?

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, Animal(y) \Rightarrow Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, \neg Animal(y) \vee Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, [\exists y\, Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   Skolem constant or skolem function?

   $\forall x\, [\, Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee [\, Loves(G(x), x)]$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\, [\forall y\, Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists z\, Loves(z, x)]$
   $\forall x\, \neg[\forall y\, Animal(y) \Rightarrow Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   $\forall x\, \neg[\forall y\, \neg Animal(y) \vee Loves(x, y)] \vee [\exists z\, Loves(z, x)]$
   $\forall x\, [\exists y\, Animal(y) \wedge \neg Loves(x, y)] \vee [\exists z\, Loves(z, x)]$

   Skolem constant or skolem function?
   $\forall x\, [\, Animal(F(x)) \wedge \neg Loves(x, F(x))] \vee [\, Loves(G(x), x)]$
   $\forall x\, (Animal(F(x)) \vee Loves(G(x), x)) \wedge (\neg Loves(x, F(x)) \vee Loves(G(x), x))$

More complex sentence

- *Everyone who loves all animals is loved by someone.*

1. $\forall x\,[\forall y\, Animal(y) \Rightarrow Loves(x,y)] \Rightarrow [\exists z\, Loves(z,x)]$

   $\forall x\,\neg[\forall y\, Animal(y) \Rightarrow Loves(x,y)] \vee [\exists z\, Loves(z,x)]$

   $\forall x\,\neg[\forall y\,\neg Animal(y) \vee Loves(x,y)] \vee [\exists z\, Loves(z,x)]$

   $\forall x\,[\exists y\, Animal(y) \wedge \neg Loves(x,y)] \vee [\exists z\, Loves(z,x)]$

   Skolem constant or skolem function?

   $\forall x\,[\,Animal(F(x)) \wedge \neg Loves(x,F(x))] \vee [\,Loves(G(x),x)]$

   $\forall x\,(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

- $(Animal(F(x)) \vee Loves(G(x),x)) \wedge (\neg Loves(x,F(x)) \vee Loves(G(x),x))$

Resolution Inference Rule

$$\frac{\ell_1 \vee \cdots \vee \ell_k, \qquad m_1 \vee \cdots \vee m_n}{\textsc{Subst}(\theta, \ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)}$$

where $\textsc{Unify}(\ell_i, \neg m_j) = \theta$. For example, we can resolve the two clauses

$$[Animal(F(x)) \vee Loves(G(x), x)] \quad \text{and} \quad [\neg Loves(u, v) \vee \neg Kills(u, v)]$$

by eliminating the complementary literals $Loves(G(x), x)$ and $\neg Loves(u, v)$, with unifier $\theta = \{u/G(x), v/x\}$, to produce the **resolvent** clause

$$[Animal(F(x)) \vee \neg Kills(G(x), x)] \,.$$

Factoring

- $\dfrac{\neg King(x) \lor Greedy(x), \;\; King(J) \lor Greedy(J)}{}$

Factoring

- $$\frac{\neg King(x) \lor Greedy(x),\;\; King(J) \lor Greedy(J)}{Greedy(J)}$$

Another sentence

- Anyone who kills an animal is loved by no one.

Another sentence

- Anyone who kills an animal is loved by no one.
  $[\exists\, y\; Animal(y) \wedge Kills(x, y)]$

Another sentence

- Anyone who kills an animal is loved by no one.

$\forall x$ $[\exists\, y\, Animal(y) \wedge Kills(x,y)] \Rightarrow [\forall\, z\, \neg Loves(z,x)]$

Another sentence

- Anyone who kills an animal is loved by no one.
  $\forall x\,[\exists y\, Animal(y) \land Kills(x,y)] \Rightarrow [\forall z\, \neg Loves(z,x)]$

Curiosity: Example 2

Everyone who loves all animals is loved by someone.
Anyone who kills an animal is loved by no one.
Jack loves all animals.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

Does $KB \models Kills(Curiosity, Tuna)$?

Curiosity: FOL sentences

KB ∧ ¬α

KB ⊨ α

A. $\forall x\ [\forall y\ \ Animal(y) \Rightarrow Loves(x, y)] \Rightarrow [\exists y\ \ Loves(y, x)]$

B. $\forall x\ [\exists z\ \ Animal(z) \wedge Kills(x, z)] \Rightarrow [\forall y\ \ \neg Loves(y, x)]$

C. $\forall x\ \ Animal(x) \Rightarrow Loves(Jack, x)$

D. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

E. $Cat(Tuna)$

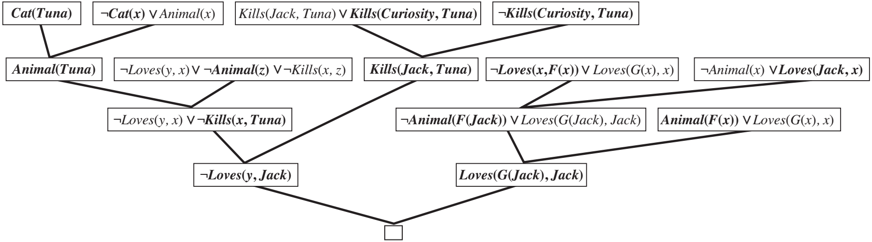
F. $\forall x\ \ Cat(x) \Rightarrow Animal(x)$

¬G. $\neg Kills(Curiosity, Tuna)$

Curiosity: FOL sentences CNF

A1.	$Animal(F(x)) \lor Loves(G(x), x)$
A2.	$\neg Loves(x, F(x)) \lor Loves(G(x), x)$
B.	$\neg Loves(y, x) \lor \neg Animal(z) \lor \neg Kills(x, z)$
C.	$\neg Animal(x) \lor Loves(Jack, x)$
D.	$Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
E.	$Cat(Tuna)$
F.	$\neg Cat(x) \lor Animal(x)$
¬G.	$\neg Kills(Curiosity, Tuna)$

Curiosity: Resolution proof



Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

Curiosity example

1. Query: Who killed the cat?

   KB $\models$ *Kills*$(x, Tuna)$?

- Nonconstructive proofs:

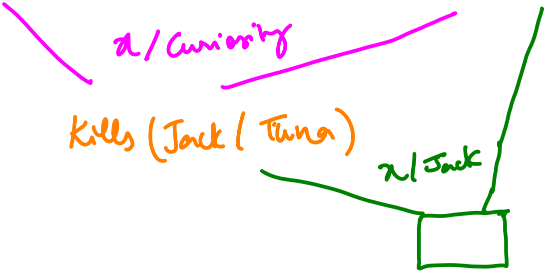
Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$



Curiosity example

1. Query: Who killed the cat?

   KB $\models$ $Kills(x, Tuna)$?

- Nonconstructive proofs:

  $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$, $\neg Kills(x, Tuna)$

- Bind once and backtrack