Machine Intelligence

Lecture 1: Introduction to MI and Agents

Thomas Dyhre Nielsen

Aalborg University

Course Organization

Teacher

Thomas Dyhre Nielsen, tdn@cs.aau.dk, office 1.2.34

Literature

D. Poole and A. Mackworth: *Artificial Intelligence. Foundations of Computational Agents* (2nd edition)

`http://artint.info/`

Course Homepage

Can be found under Moodle

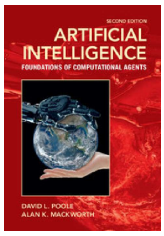
Times

- Mondays, 8.15-12.00 (typically Frb 7H, Aud): Exercises 8.15-10.00, Lecture 10.15-12.00
- Wednesdays 8.15–12.00 (group rooms): Extended exercise sessions

Exam

Exam in January

Why this book?



Several reasons, but:

> *We decided that it is better to clearly explain the foundations upon which more sophisticated techniques can be built, rather than present these more sophisticated techniques. This means that a larger gap may exist between what is covered in this book and the frontier of science. But it also means that the student will have a better foundation to understand current and future research.*

Tentative course overview

Topics:

- Introduction
- Problem solving as search
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Machine learning
- Planning
- Reinforcement learning
- Multi-agent systems

Learning goals

After having followed the course you should

- have knowledge about basic techniques within the field of machine intelligence and computational agents.
- be able to apply key machine intelligence techniques to a specific problem domain.
- be able to reason about computational agents that can operate in domains of varying complexity.

Learning goals

After having followed the course you should

- have knowledge about basic techniques within the field of machine intelligence and computational agents.
- be able to apply key machine intelligence techniques to a specific problem domain.
- be able to reason about computational agents that can operate in domains of varying complexity.

. . . and be well-prepared for the aMI course and the machine intelligence specialization!

AI

What is AI?

The Turing Test

A.M. Turing: "Computing Machinery and Intelligence", Mind Vol.59 (1950). Proposes an *imitation game*, which (slightly modified) has become known as the *Turing test*:

An *interrogator* is connected via one terminal to a real person, and by another terminal to a computer (both in another room). The interrogator does not know which terminal is connected to the machine. He or she can perform on both terminals a (natural language) dialogue with whatever is at the other end of the line. The machine passes the Turing test, if the interrogator is not able to identify, which terminal is connected to the machine.

$\Rightarrow$ The Turing test tests observable behavior, not cognitive processes!

What is AI?

Achievements: Deep Blue

- 30 IBM RS/6000 processors
- 480 custom chess processors
- able to examine 200 million moves per second
- database of 700.000 grandmaster games
- endgame database (covering all 5 piece positions)

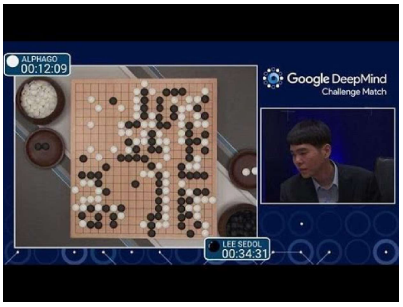
Results:

1996:	Kasparov 4	Deep Blue 2
1997:	Kasparov 2.5	Deep Blue 3.5

What is AI?

... two decades later

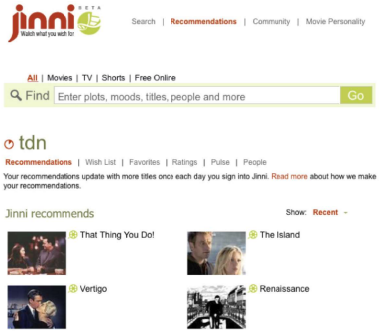
AlphaGo was developed by Google Deepmind to play Go. In March 2016 it payed a five game match against Lee Sedol with a final score of 4-1 in favor of AlphaGo.



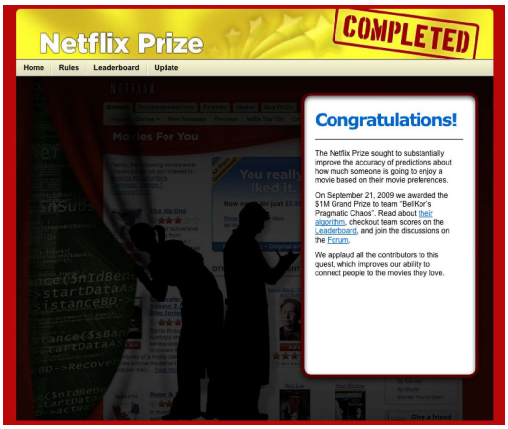
- AlphaGo combines Monte Carlo tree search with (deep) artificial neural networks.

Recommender systems

Systems for recommending items that users are likely to find interesting



Recommmending items to users



Jeopardy!: Watson

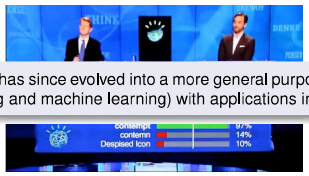
In 2011, Watson beat Brad Rutter, the biggest all-time money winner on Jeopardy!, and Ken Jennings, the record holder for the longest championship streak (75 days)

- Application of natural language processing, information retrieval, knowledge representation and reasoning, and machine learning
- Made up of a cluster of 90 IBM Power 750 servers with a total of 2880 POWER7 processor cores and 16 Terabytes of RAM. Each Power 750 server uses a 3.5 GHz POWER7 eight core processor, with four threads per core.

Taken from Wikipedia, August 25th, 2011

Jeopardy!: Watson

In 2011, Watson beat Brad Rutter, the biggest all-time money winner on Jeopardy!, and Ken Jennings, the record holder for the longest championship streak (75 days)



New developments Watson has since evolved into a more general purpose cognitive platform (using natural language processing and machine learning) with applications in e.g. health care.

- Application of natural language processing, information retrieval, knowledge representation and reasoning, and machine learning
- Made up of a cluster of 90 IBM Power 750 servers with a total of 2880 POWER7 processor cores and 16 Terabytes of RAM. Each Power 750 server uses a 3.5 GHz POWER7 eight core processor, with four threads per core.

Taken from Wikipedia, August 25th, 2011

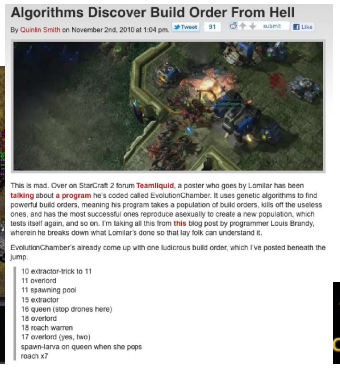
Computer games

AI in computer games (e.g. Starcraft)

`http://eis.ucsc.edu/StarCraftAICompetition`

Computer games

AI in computer games (e.g. Starcraft)



`http://eis.ucsc.edu/StarCraftAICompetition`

Computer games

The TrueSkill ranking system is a skill based ranking system for Xbox Live developed at Microsoft Research.

`https://research.microsoft.com/en-us/projects/trueskill/`

Troubleshooting

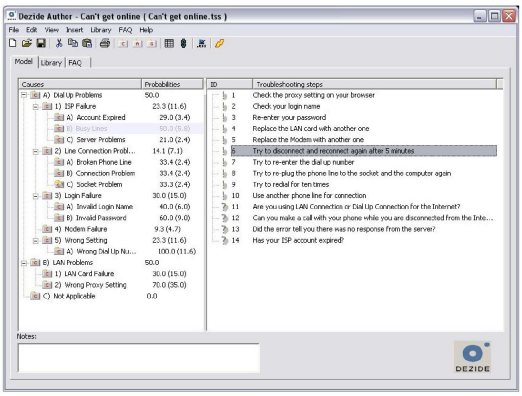
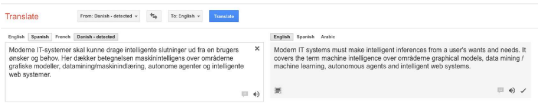


Figure borrowed from Dezide's homepage (`www.dezide.com`)

Automatic Translation

Achievements: Automatic Translation

Google translate:



Autonomous driving

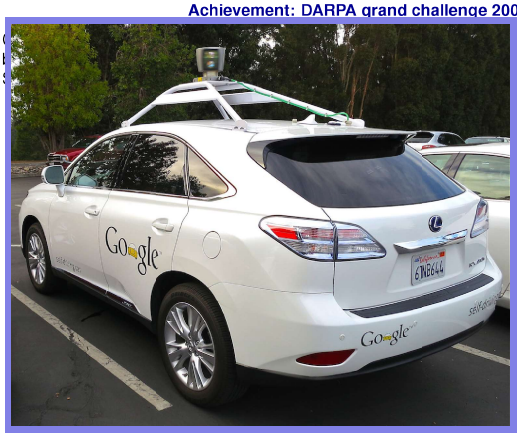
Achievement: DARPA grand challenge 2005

Competition for autonomous vehicles: navigate 132 miles through desert terrain (route specified by approx. 3000 "waypoints"). 5 out of 23 vehicles completed the task. Winner: *Stanley* of Stanford Racing Team in 6h 53m (19.2 mph).

- 7 Pentium M computers
- Sensors: 4 laser range finders, 1 radar system, 1 stereo camera pair, 1 monocular vision system, GPS, inertial measurement unit, wheel speed.

Autonomous driving

Achievement: DARPA grand challenge 2005



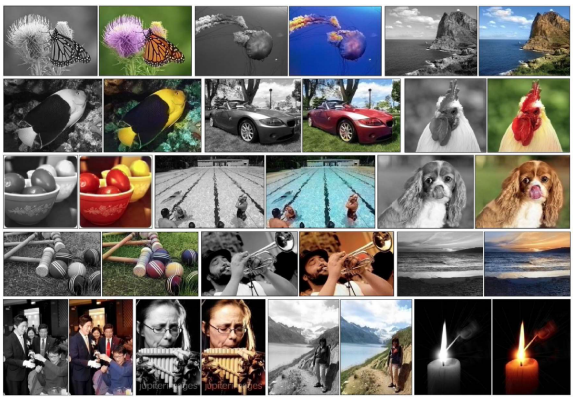
t terrain (route specified
Winner: *Stanley* of

ir, 1 monocular vision

June 2016: approx. 2.8 mill. km. autonomous driving.

Image analysis/processing

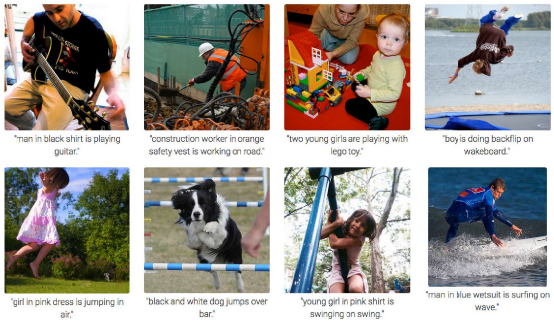
Colorization of images



Zhang, Isola, Efros. Colorful Image Colorization. In ECCV, 2016.

Image analysis/processing

Image captioning



Andrej Karpathy, Li Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions, CVPR 2015

Other application areas

- Medical diagnosis and advisory systems
- Information processing and filtering,
- Display of information for time-critical decisions
- Spam filtering
- Optical character recognition
- Profiling/Credit scoring: profiling customers
- Bioinformatics
- Real estate: Prediction of house prices
- Computer networks: intrusion detection
- Alert and monitoring systems
- Speech recognition
- Face recognition, image annotation
- Action recognition (in video sequences)
- . . .

Towards the Turing Test ...

Turing Test (Loebner Competition)

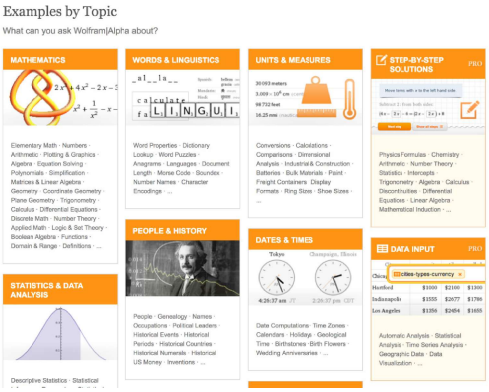
- Loebner Competition: (Non-scientific) competition for computer systems performing under Turing Test conditions.
- `http://www.pandorabots.com/pandora/talk?botid=f5d922d97e345aa1`

Online help

- Combine natural language interface with expert knowledge.
- Restricted Domain (Geography): CHAT-80 (1982)
- Broad Domain ("all factual knowledge"): Wolfram Alpha
  `http://www.wolframalpha.com/`

Wolfram Alpha: Reality

Some queries to try:

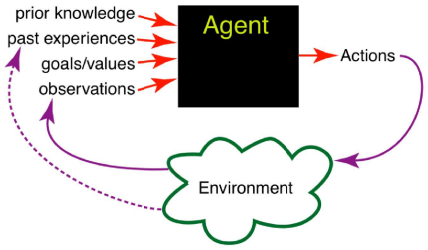


The Agent view of AI

AI and Agents

AI is the field that studies the synthesis and analysis of computational agents that act intelligently. [PM, p.3]

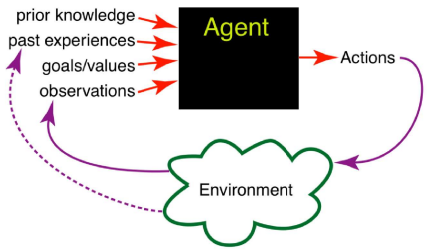
A coupling of perception, reasoning, and acting comprises an agent. [PM, p.10]



AI and Agents

AI is the field that studies the synthesis and analysis of computational agents that act intelligently. [PM, p.3]

A coupling of perception, reasoning, and acting comprises an agent. [PM, p.10]



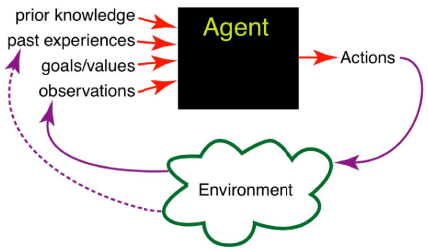
Some special flavors:

- Autonomous agents
- Intelligent agents
- Software agents
- Multi-agent systems

AI and Agents

AI is the field that studies the synthesis and analysis of computational agents that act intelligently. [PM, p.3]

A coupling of perception, reasoning, and acting comprises an agent. [PM, p.10]



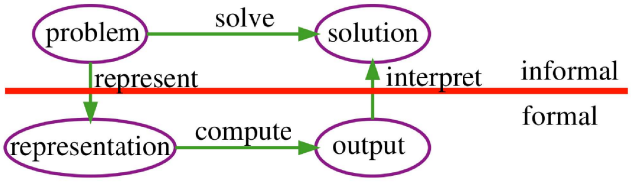
Some special flavors:

- Autonomous agents
- Intelligent agents
- Software agents
- Multi-agent systems

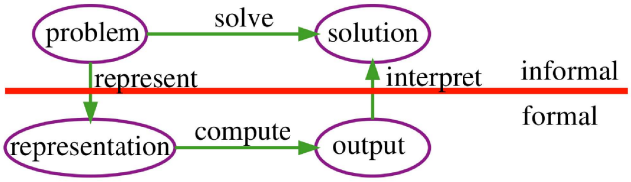
What is *not* an agent?

- perception $\sim$ input
- reasoning $\sim$ computation
- acting $\sim$ output
- $\rightsquigarrow$ "agent" a design metaphor, not a strict technical concept

Represent and Compute



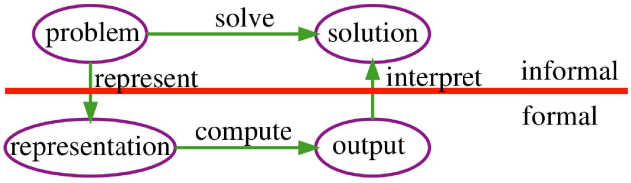
Represent and Compute



A *representation* should be:

- Sufficiently rich to encode the required knowledge.
- Be "close" to the problem.
- Amenable to efficient computation.
- Able to be acquired from people, data, or experience.

Represent and Compute



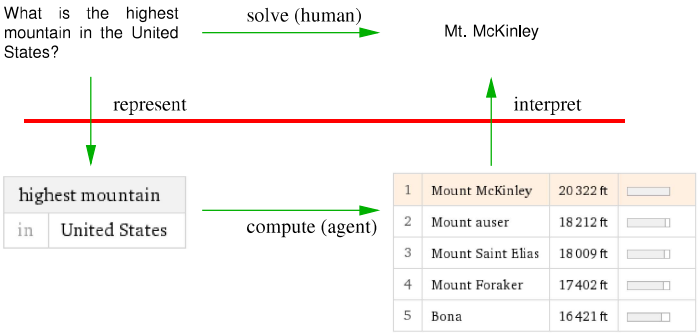
A *representation* should be:

- Sufficiently rich to encode the required knowledge.
- Be "close" to the problem.
- Amenable to efficient computation.
- Able to be acquired from people, data, or experience.

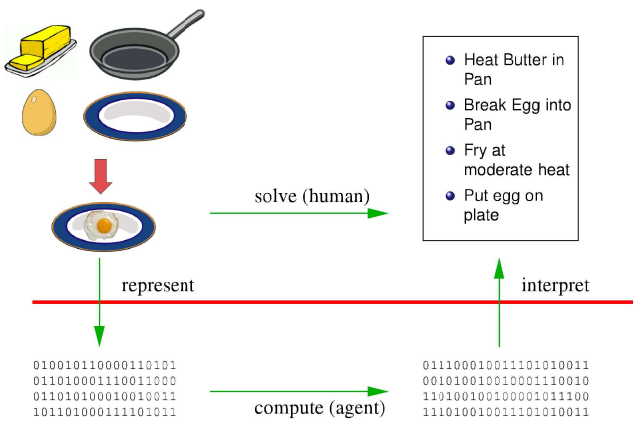
Questions to be considered:

- What is a solution and how good should it be (optimal, satisficing, approximately, probable)?
- How can the problem be represented?
- How can an output be computed (what properties should a solution have)?

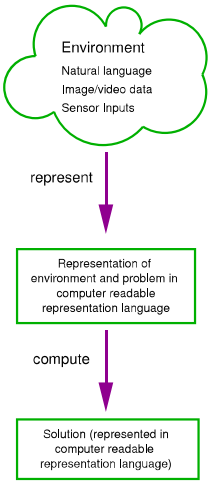
Represent and Compute



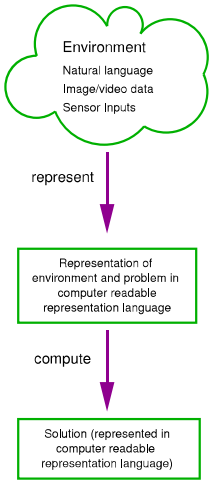
Represent and Compute



Problem Formalization

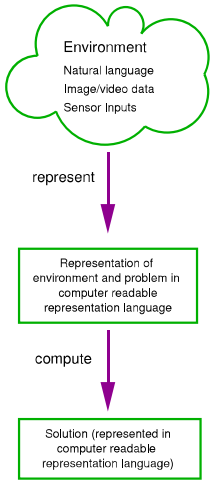


Problem Formalization



2. What formal representation languages can be used?

Problem Formalization

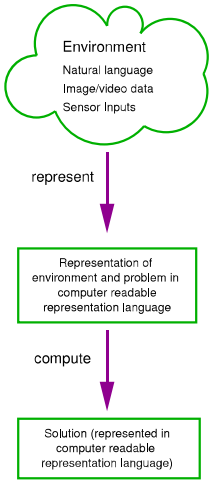


1. How to interpret input/observations from environment?

2. What formal representation languages can be used?

3. How to solve problems in the given representation language?

Problem Formalization



1. How to interpret input/observations from environment?

Natural Language Processing
Computer Vision
. . .

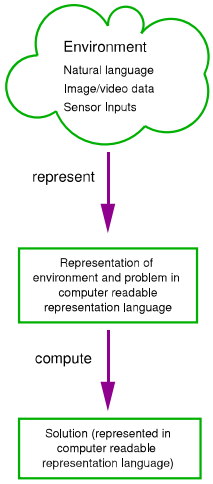
2. What formal representation languages can be used?

Knowledge Representation

3. How to solve problems in the given representation language?

Problem Solving
Automated Reasoning

Problem Formalization



1. How to interpret input/observations from environment?

Natural Language Processing
Computer Vision
. . .

2. What formal representation languages can be used?

Knowledge Representation

3. How to solve problems in the given representation language?

Problem Solving
Automated Reasoning

In this course the focus is on **2.** and **3.**!

Representing the problem

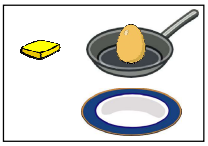
Levels of Representation

We consider 3 representation schemes

- State based
- Feature based
- Relational

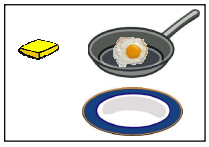
State based

State 01

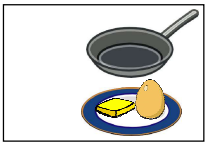


State 04

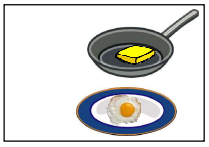
State 08



State 12



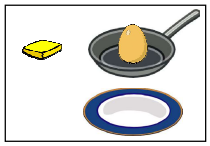
State 14



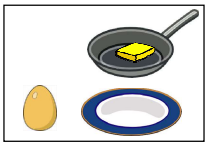
State 18

Feature based

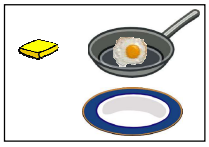
egg=whole,
butter_in=table,
egg_in=table



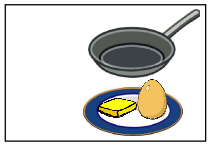
egg=whole,
butter_in=table,
egg_in=pan



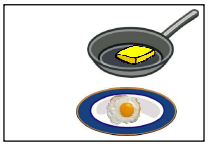
egg=whole,
butter_in=pan,
egg_in=table



egg=broken,
butter_in=table,
egg_in=pan



egg=whole,
butter_in=plate,
egg_in=plate

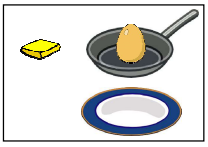


egg=broken,
butter_in=pan,
egg_in=plate

30 binary features represent $2^{30} = 1.073.741.824$ states.

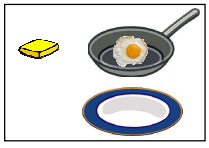
Relational

state(egg,whole),
in(butter,table),
in(egg,table)

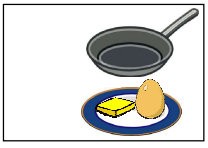


state(egg,whole),
in(butter,table),
in(egg,pan)

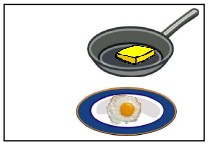
state(egg,whole),in(butter,pan),
in(egg,table)



state(egg,broken),
in(butter,table),
in(egg,pan)



state(egg,whole),
in(butter,plate),
in(egg,plate)



state(egg,broken),
in(butter,pan),
in(egg,plate)

1 binary relation and 100 individuals give $100^2 = 10.000$ boolean features, or 2^{10000} states.

Levels of Detail

Do we need to distinguish states

egg=whole,
butter_in=table,
egg_in=table,
butter_position_to_pan=left

egg=whole,
butter_in=table,
egg_in=pan,
butter_position_to_pan=right

- Not at "recipe level"
- Yes at robot control level: "*move arm to left, grab butter, . . .*"
- ⇝ may need hierarchical description of state space to reason at different levels of abstraction.

Other dimensions of complexity

Modularity

- Flat, modular, hierarchical

Other dimensions of complexity

Modularity

- Flat, modular, hierarchical

Planning horizon

- Non-planning, finite, indefinite, infinite

Other dimensions of complexity

Modularity
- Flat, modular, hierarchical

Planning horizon
- Non-planning, finite, indefinite, infinite

Domain uncertainty
- Fully observable vs. partially observable world.
- Deterministic vs. stochastic.

Other dimensions of complexity

Modularity
- Flat, modular, hierarchical

Planning horizon
- Non-planning, finite, indefinite, infinite

Domain uncertainty
- Fully observable vs. partially observable world.
- Deterministic vs. stochastic.

Preferences
- Achievement goals, maintenance goals.
- Complex ordinal or cardinal preferences.

MI Autumn 2019 Representing the problem 31

Other dimensions of complexity

Modularity
- Flat, modular, hierarchical

Planning horizon
- Non-planning, finite, indefinite, infinite

Domain uncertainty
- Fully observable vs. partially observable world.
- Deterministic vs. stochastic.

Preferences
- Achievement goals, maintenance goals.
- Complex ordinal or cardinal preferences.

Number of agents
- Single agent or multiple agents

MI Autumn 2019 Representing the problem 31

Other dimensions of complexity

Modularity
- Flat, modular, hierarchical

Planning horizon
- Non-planning, finite, indefinite, infinite

Domain uncertainty
- Fully observable vs. partially observable world.
- Deterministic vs. stochastic.

Preferences
- Achievement goals, maintenance goals.
- Complex ordinal or cardinal preferences.

Number of agents
- Single agent or multiple agents

Learning
- Knowledge is given at design time or learned from experience.

MI Autumn 2019 Representing the problem 31

Other dimensions of complexity

Modularity
- Flat, modular, hierarchical

Planning horizon
- Non-planning, finite, indefinite, infinite

Domain uncertainty
- Fully observable vs. partially observable world.
- Deterministic vs. stochastic.

Preferences
- Achievement goals, maintenance goals.
- Complex ordinal or cardinal preferences.

Number of agents
- Single agent or multiple agents

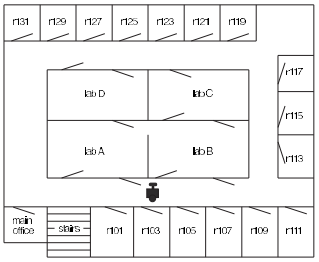
Learning
- Knowledge is given at design time or learned from experience.

Computational limits
- Perfect or bounded rationality

MI Autumn 2019 Representing the problem 31

Prototypical Applications

Autonomous Delivery Robot



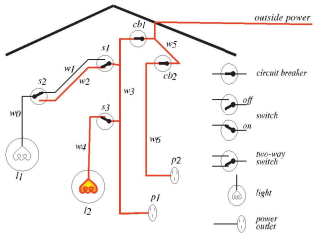
Inputs:
- Prior knowledge, past experience, goals, observations

Outputs:
- Motor control, speech, video display

Complexity issues:
- Hierarchical decomposition
- Planning horizon
- Goals
- Uncertainty
- ...

- Robot can move, pick up and put down objects
- Receives commands (in natural language)
- Can deliver packages, mail, coffee, ...
- Must interpret commands, develop and execute plans for action

Diagnostic Assistant



Inputs:
- Prior knowledge, past experience, goals, observations

Outputs:
- Recommendations on treatments and tests

Complexity issues:
- Sensing/effect uncertainty
- Knowledge representation
- Goal specification

- Advise a human about system, e.g. diagnose patient, troubleshoot electrical system, automobile, etc.
- Substitute of human expert

Trading Agent

Characteristics

- Automatically buy/sell goods for user/company (possibly at auction)
- Determine good strategy to procure necessary goods in time at best price

Inputs
- Prior knowledge about goods, past experience, preferences, observations

Output
- Proposals to the user

Tentative course overview

Topics:
- Introduction
- Problem solving as search
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Machine learning
- Planning
- Reinforcement learning
- Multi-agent systems

Machine Intelligence

Lecture 2: Search

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:
- Introduction
- **Search-based methods**
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Machine learning
- Planning
- Multi-agent systems

Problem Solving as Search

Problem Description

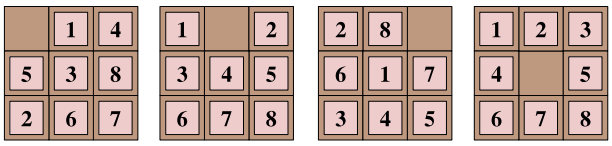
We consider problems where an agent

- has a state-based representation of its environment
- can observe with certainty which state it is in
- has a certain goal it wants to achieve
- can execute actions that have definite effects (no uncertainty)

The agent needs to find a sequence of actions that lead it to a **goal state**: a state in which its goal is achieved.

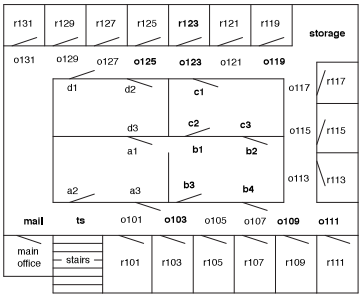
Example: 8 Puzzle

Problem: re-arrange tiles into goal configuration:



- There are 362880 states
- Actions: *move_up*, *move_down*,*move_left*, *move_right*

Example: Office Robot



- States: locations, e.g. r131, storage, o117, c3,...
- Actions: move to neighboring locations, e.g. *move_r131_o131*, *move_o119_storage*, *move_b2_c3*,...

State-Space Problem

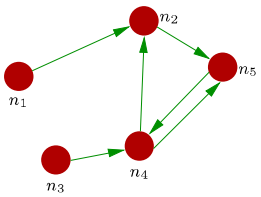
A **State-Space Problem** consists of

- A set of states
- A subset of **start states**
- A set of actions (not all actions available at all states)
- An **action function** that for a gives state s and action a returns the state reached when executing a in s
- A **goal test** that for any state s returns the boolean value *goal*(s) (true if s is a goal state)
- (optional) a **cost function** on actions
- (optional) a **value function** on goal states

A **Solution** consists of

- For any given start state, a sequence of actions that lead to a goal state
- (optional) a sequence of actions with minimal cost
- (optional) a sequence of actions leading to a goal state with maximal value

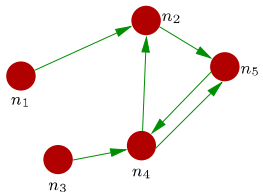
Graphs



A **directed graph** consists of

- a set of **nodes**
- a set of **arcs** (ordered pairs of nodes)

Graphs



A **directed graph** consists of

- a set of **nodes**
- a set of **arcs** (ordered pairs of nodes)

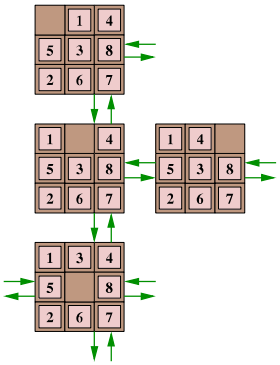
Further terminology:

- n_2 is a **neighbor** of n_4 (not the other way round!).
- n_3, n_4, n_2, n_5 is a **path** from n_3 to n_5.
- n_2, n_5, n_4, n_2 is a path that is a **cycle**.
- a graph is **acyclic** if it has no cycles.

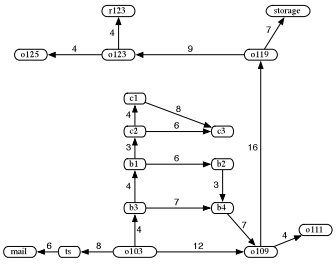
State-Space Problems as Graphs

- Nodes: states
- Arcs: possible state-transitions from actions (arcs can be labeled with actions)

8 puzzle graph (part)



Delivery Robot Graph



- Arcs labeled with costs (time to travel, fuel costs, ...)
- *Forward branching factor* of a node = number of arcs leaving that node.
- *Backward branching factor* of a node = number of arcs entering that node.

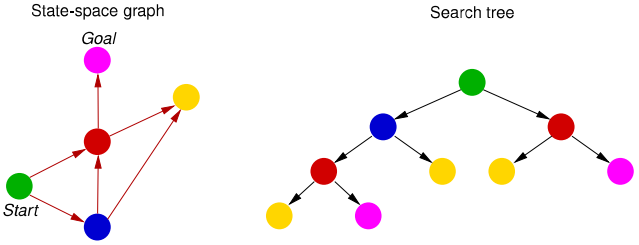
Graph Search

- A state-space problem can be solved by searching in the state-space graph for paths from start states to goal states.

Graph Search

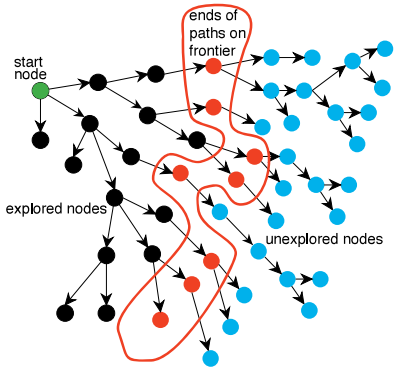
- A state-space problem can be solved by searching in the state-space graph for paths from start states to goal states.
- This does not require the whole graph at once: search may only locally generate neighbors of currently visited node.

From Graph to Search Tree



- Tree: special graph with
  - exactly one node that has no incoming arc (the **root**)
  - all other nodes have exactly one incoming arc (may have 0,1,2,3,... outgoing arcs)
- Nodes in the search tree correspond to paths in the graph beginning in the start state
- Nodes in the search tree also are labeled with states: the last state of the path

Snapshot of Search Tree Construction



Generic Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure $goal(n)$ that tests if n is a goal node.

frontier := $\{\langle s\rangle : s \text{ is a start node}\}$;
while *frontier* is not empty:
select and **remove** path $\langle n_0, \ldots n_k\rangle$ from *frontier*;
if $goal(n_k)$
return $\langle n_0, \ldots, n_k\rangle$;
for every neighbor n of n_k
add $\langle n_0 \ldots, n_k, n\rangle$ to *frontier*;
end while

Generic Search Algorithm

Input: a graph,
a set of start nodes,
Boolean procedure $goal(n)$ that tests if n is a goal node.

frontier := $\{\langle s\rangle : s \text{ is a start node}\}$;
while *frontier* is not empty:
select and **remove** path $\langle n_0, \ldots n_k\rangle$ from *frontier*;
if $goal(n_k)$
return $\langle n_0, \ldots, n_k\rangle$;
for every neighbor n of n_k
add $\langle n_0 \ldots, n_k, n\rangle$ to *frontier*;
end while

The algorithm does not require the complete graph as input: only needed are:

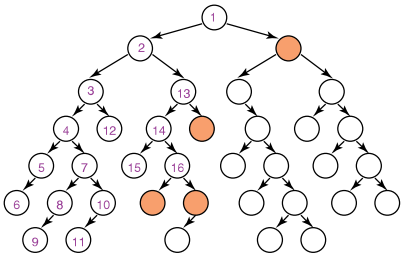
- List of start nodes (often only one)
- Boolean function *goal*(Node n)
- Function *get_neighbors*(Node n) returning list of neighbors of n.

Depth-first search

- Select from the frontier that path that was most recently added to the frontier (frontier implemented as **stack**).

Example

- Explored nodes with order of exploration
- Frontier (colored)
- Unexplored nodes



Depth-first search

Properties

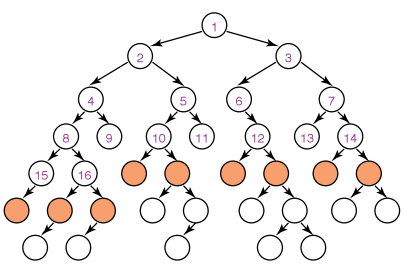
- Space used is linear in the length of the current path.
- May not terminate if state-space graph has cycles
- With a forward branching factor bounded by b and depth n, the worst-case time complexity of a finite tree is b^n.

Breadth-first search

- Select from the frontier that path that was earliest added to the frontier (frontier implemented as **queue**).

Example

- Explored nodes with order of exploration
- Frontier (colored)
- Unexplored nodes



Breadth-first search

- Will always find a solution if one exists
- Size of frontier always increases during search up to order of magnitude of total size of search tree.

Breadth-first search

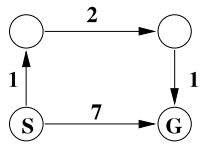
- Will always find a solution if one exists
- Size of frontier always increases during search up to order of magnitude of total size of search tree.
- Can be adapted to find a minimum cost path.

Problems with cost function

Problem

- Assume that for each action at each state we have an associated **cost**
- The cost of a solution is the sum of the costs of all actions on the path from start to goal state.
- A **minimum cost solution** is a solution with minimal cost.

Example



- Breadth first search will find the *shortest*, but not the *cheapest* solution.
- Depth first search may find either solution, depending on order of neighbor enumeration

Lowest-Cost-First Search

Simple modification of generic search algorithm:

- with each path in *frontier* store the cost of the path
- Modify one line of code:
  select and **remove** path $\langle n_0, \ldots, n_k \rangle$ with minimal cost from *frontier*;

Properties

- If all actions have non-zero cost, and solution exists, then a minimal cost solution will be found.
- Space requirement depends on cost structure, but usually similar to breadth-first search.

Iterative Deepening Search

Goal

- Termination guarantee of breadth-first search
- Space efficiency of depth-first search

Algorithm

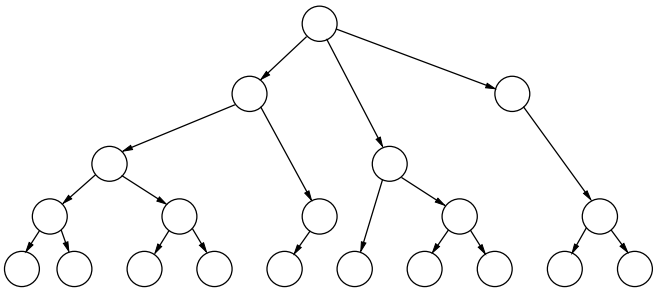
Depth-bounded search k

- As depth-first search, but
- do not add neighbors of selected node to frontier if selected node has depth k.

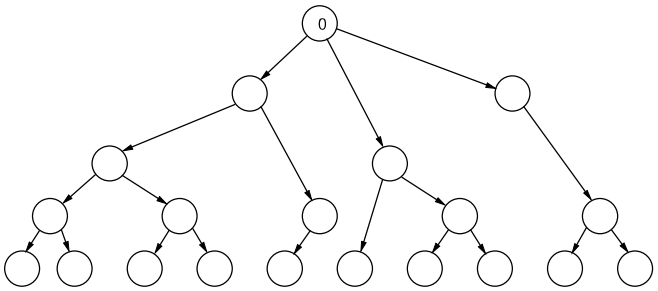
Iterative deepening search

- For $k = 1, 2, 3, \ldots$ perform depth-bounded search k.

Iterative deepening: an example



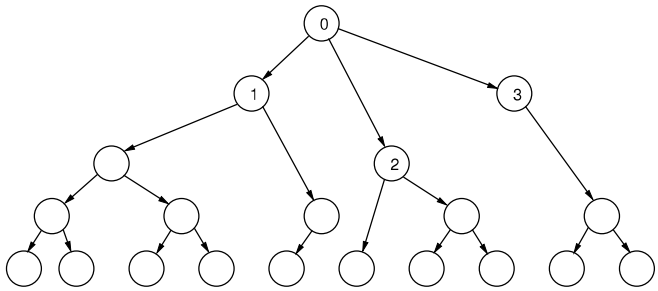
Iterative deepening: an example



Perform depth-bounded search to level $k = 1$.

NB: The node numbering corresponds to when the states are first visited.

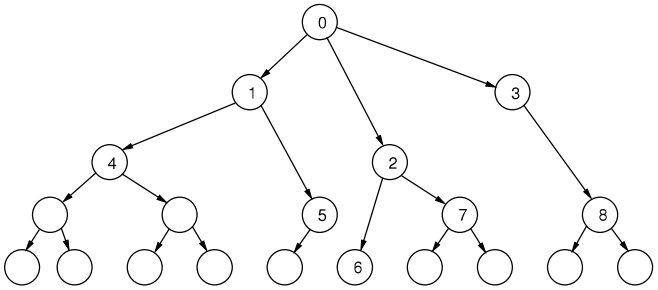
Iterative deepening: an example



Perform depth-bounded search to level $k = 2$.

NB: The node numbering corresponds to when the states are first visited.

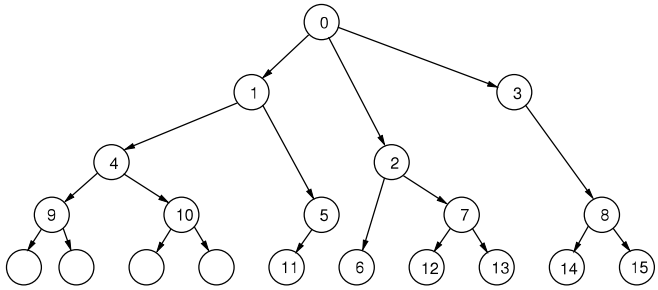
Iterative deepening: an example



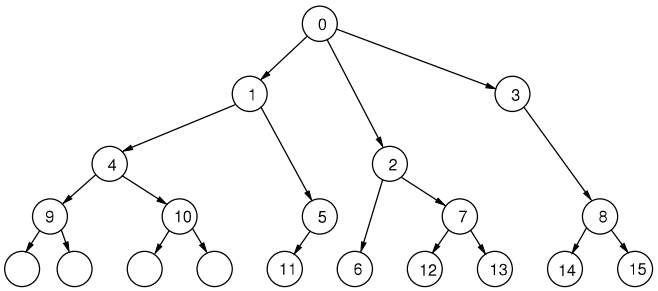
Perform depth-bounded search to level $k = 3$.

NB: The node numbering corresponds to when the states are first visited.

Iterative deepening: an example



Iterative deepening: an example



Properties

- Has desired termination and space efficiency properties
- Duplicates computations (depth-bounded search k repeats computaions of depth-bounded search $k - 1$). Not as problematic as it looks: constant overhead of $(b/(b - 1))$.

Uninformed Search

Depth-first, Breadth-first and Iterative deepening are **uninformed** search strategies:
they do not assume/use any knowledge of the search space except the pure graph structure.

Informed Search

Heuristic Search

Idea

- Lowest-Cost-First Search only considers cost of already constructed partial solution.
- Idea: Try to estimate the cost of optimal path from current state to goal.

Heuristic Search

Idea

- Lowest-Cost-First Search only considers cost of already constructed partial solution.
- Idea: Try to estimate the cost of optimal path from current state to goal.

Actual Cost

Given a cost function on actions, can define for any node n in the search tree:

opt(n)= *cost of optimal (minimal cost) path from* n *to a goal state (infinite if no path to goal exists).*

- The *opt* function can usually not be computed
- *opt*(n) only depends on the state at node n.

Heuristic Search

Idea

- Lowest-Cost-First Search only considers cost of already constructed partial solution.
- Idea: Try to estimate the cost of optimal path from current state to goal.

Actual Cost

Given a cost function on actions, can define for any node n in the search tree:

opt(n)= *cost of optimal (minimal cost) path from* n *to a goal state (infinite if no path to goal exists).*

- The *opt* function can usually not be computed
- *opt*(n) only depends on the state at node n.

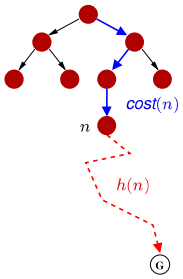
Heuristic Function

$h(n)$ (non-negative number): estimate of *opt*(n). $h(n)$ is an **underestimate** if for all nodes n:

$$h(n) \leq \textit{opt}(n)$$

A^* search

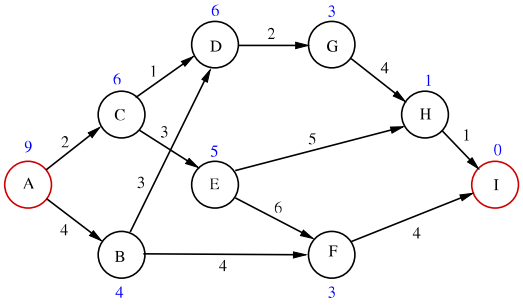
For any node n in the search tree we have:



- *cost*(n) : (true) cost of reaching n from the root node.
- $h(n)$: a heuristic function
- $f(n) :=$ *cost*$(n) + h(n)$

A^* search: always expand fringe node with minimal f-value.

A^* search: an example



For node D:

- *cost*$(D) = 3$
- $h(D) = 6$
- $f(D) :=$ *cost*$(D) + h(D) = 3 + 6 = 9$

A^* Admissibility

Admissibility

For finite state space graphs: if

- all actions have cost > 0
- $h(n)$ is an underestimate
- there exists a solution

then A^* will return an optimal solution.

A^* Admissibility

Admissibility

For finite state space graphs: if

- all actions have cost > 0
- $h(n)$ is an underestimate
- there exists a solution

then A^* will return an optimal solution.

Special Cases

- $h(n) = 0$ for all n: A^* becomes lowest-cost-first search
- $h(n) = opt(n)$: A^* directly constructs the optimal path

$\rightsquigarrow$ should find heuristic functions $h(n)$ that are "close underestimates" of $opt(n)$.

Constructing heuristic functions

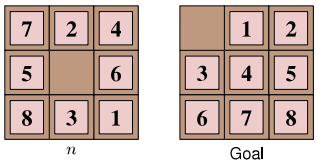
General strategy to design underestimates:

- define a simplified (easier) problem
  - add arcs, states to the state space graph
  - reduce cost of existing arcs
- use the exact function *opt* in the simplified problem as the heuristic function for the original problem
- requires: *opt* in the simplified problem must be easily computable

Heuristic functions for 8-puzzle

Simplified Problem 1: Can move any tile to any square (more states: can have several tiles on one square). Then

$h_1(n) = opt_1(n)$:= *Number of tiles that are not in their goal position.*



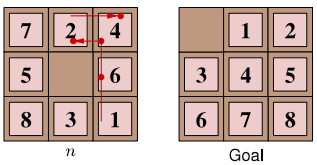
Heuristic functions for 8-puzzle

Simplified Problem 1: Can move any tile to any square (more states: can have several tiles on one square). Then

$h_1(n) = opt_1(n)$:= *Number of tiles that are not in their goal position.*

Simplified Problem 2: Can move any tile to an *adjacent* square.

$h_2(n) = opt_2(n)$:= *Sum of Manhatten distances of all tiles to their goal position.*



$h_1(n) = 8$

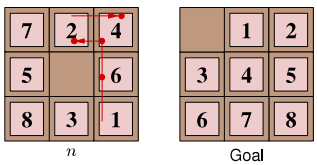
Heuristic functions for 8-puzzle

Simplified Problem 1: Can move any tile to any square (more states: can have several tiles on one square). Then

$h_1(n) = opt_1(n)$:= *Number of tiles that are not in their goal position.*

Simplified Problem 2: Can move any tile to an *adjacent* square.

$h_2(n) = opt_2(n)$:= *Sum of Manhatten distances of all tiles to their goal position.*



$h_1(n) = 8$
$h_2(n) = 3+1+2+2+2+3+3+2 = 18$

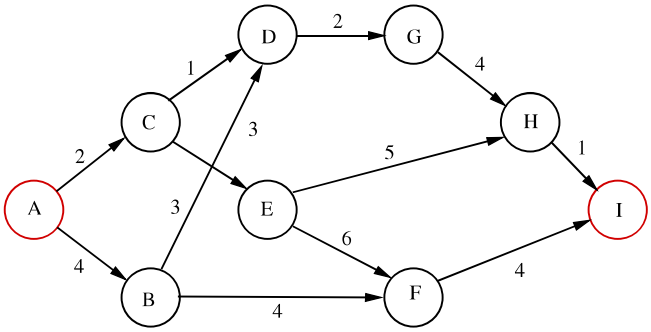
Dynamic programming

Idea: for statically stored graphs, build a table of $dist(n)$ the actual distance of the shortest path from node n to a goal.
This can be built backwards from the goal:

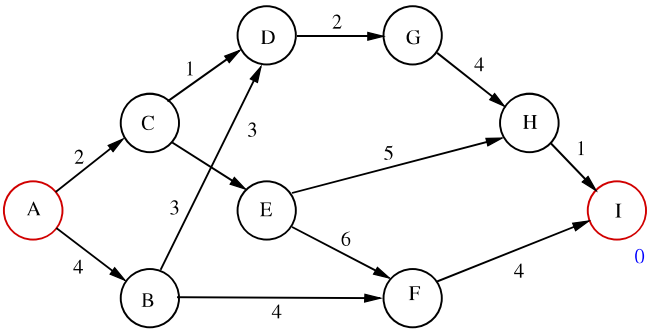
$$dist(n) = \begin{cases} 0 & \text{if } is\_goal(n), \\ \min_{\langle n,m\rangle \in A}(|\langle n,m\rangle| + dist(m)) & \text{otherwise.} \end{cases}$$

This can be used locally to determine what to do.

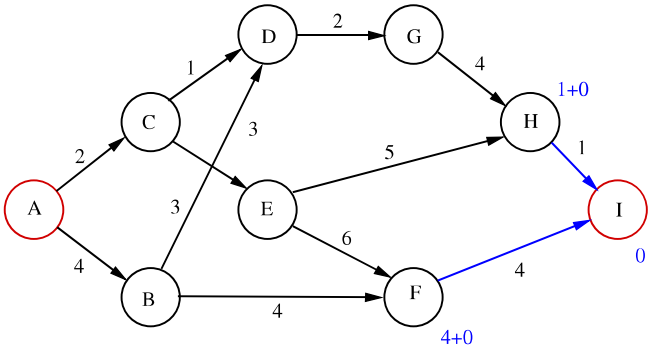
Dynamic programming: Example



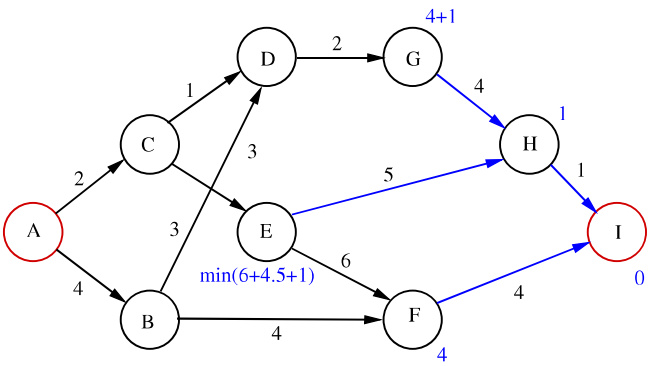
Dynamic programming: Example



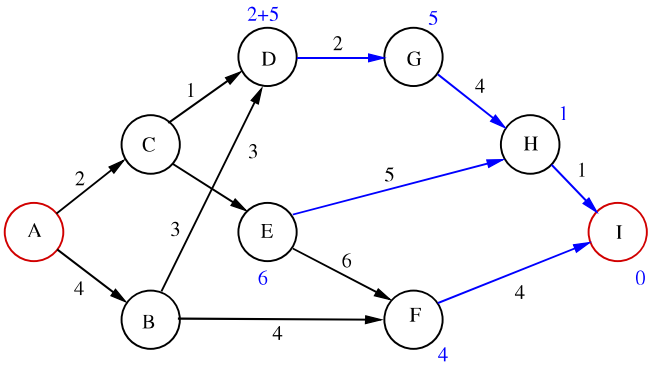
Dynamic programming: Example



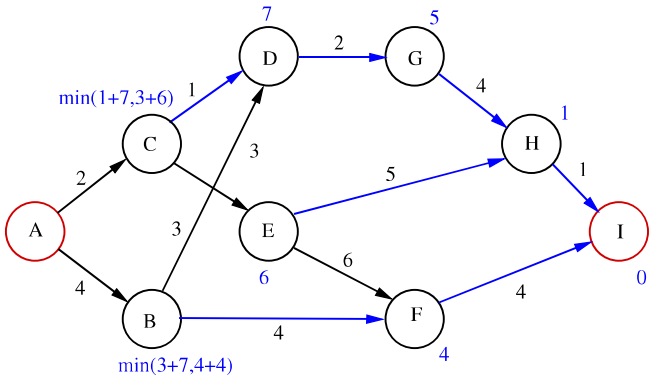
Dynamic programming: Example



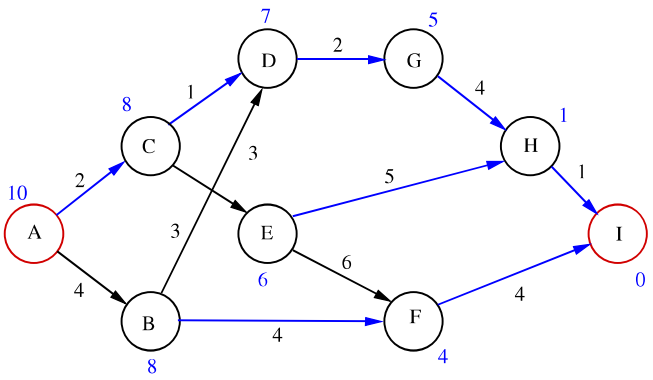
Dynamic programming: Example



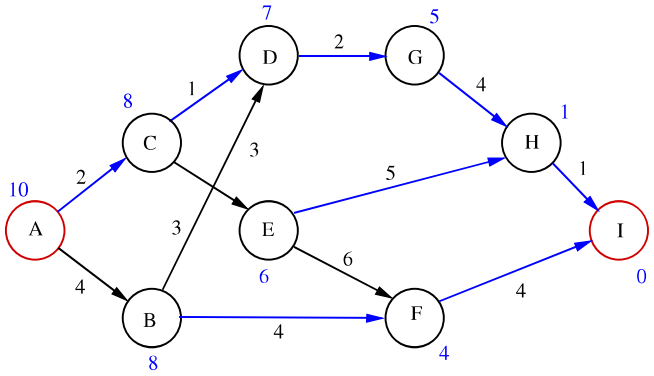
Dynamic programming: Example



Dynamic programming: Example



Dynamic programming: Example



There are two main problems:

- You need enough space to store the graph.
- The $dist$ function needs to be recomputed for each goal.

Direction of search

- The definition of searching is symmetric: find path from start nodes to goal node or from goal node to start nodes.
- Search complexity is b^n. Should use forward search if forward branching factor is less than backward branching factor, and vice versa.
- Note: sometimes when graph is dynamically constructed, you may not be able to construct the backwards graph.

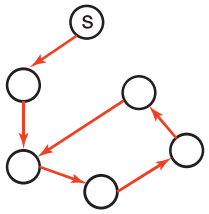
Direction of search

- The definition of searching is symmetric: find path from start nodes to goal node or from goal node to start nodes.
- Search complexity is b^n. Should use forward search if forward branching factor is less than backward branching factor, and vice versa.
- Note: sometimes when graph is dynamically constructed, you may not be able to construct the backwards graph.

Bi-directional search

- You can search backward from the goal and forward from the start simultaneously.
- This wins as $2b^{k/2} \ll b^k$. This can result in an exponential saving in time and space.
- The main problem is making sure the frontiers meet.
- This is often used with one breadth-first method that builds a set of locations that can lead to the goal. In the other direction another method can be used to find a path to these interesting locations.

Cycle checking



- A searcher can prune a path that ends in a node already on the path, without removing an optimal solution.
- Using depth-first methods, with the graph explicitly stored, this can be done in constant time.
- For other methods, the cost is linear in path length.

Machine Intelligence

Lecture 3: Constraint satisfaction problems

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- **Constraint satisfaction problems**
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Machine learning
- Planning
- Multi-agent systems

Features and Possible Worlds

Features and Variables

Describing the world (environment) by features:

Name (**Algebraic Variable**)	Domain
Symbol_on_square_1	$\{1, 2, 3, 4, 5, 6, 7, 8, \textit{empty}\}$
Robot_battery	$\{\textit{full}, \textit{half}, \textit{empty}\}$
Robot_position	$\{r131, \ldots, 0111\}$
Coffee_ready	$\{\textit{true}, \textit{false}\}$
No._of_undelivered_packages	$\{1, 2, 3, \ldots\}$
Temperature	$[-25, 40]$

- We will be mostly concerned with (algebraic) variables that have a *finite* domain.
- Special interest: **boolean** variable with domain $\{\textit{true}, \textit{false}\}$
- Numerical variables can be approximated:

$$\begin{array}{rcl} \{1, 2, 3, \ldots\} & \mapsto & \{1, 2, 3, 4, 5, > 5\} \\ [-25, 40] & \mapsto & \{-25, -24, \ldots, -1, 0, 1, \ldots 40\} \end{array}$$

Possible Worlds

From variables to possible worlds

A **possible world** for a set of variables is an assignment of a value to each variable.

Connection with state spaces

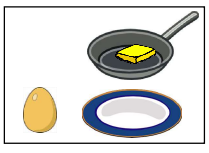
The set of all possible worlds for a given set of variables defines a state space (we can also call a possible world simply a state).

Example: Cooking

Variables:

egg	$\{\textit{whole}, \textit{broken}\}$
butter_in	$\{\textit{pan}, \textit{plate}, \textit{table}\}$
egg_in	$\{\textit{pan}, \textit{plate}, \textit{table}\}$

One out of $2 \cdot 3 \cdot 3 = 18$ possible worlds:



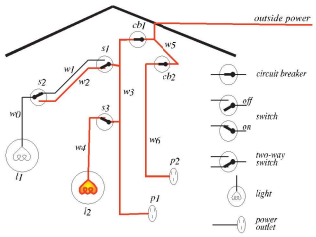
egg=whole
butter_in=pan
egg_in=table

Example: Electrical

Variables:

S_1_pos	{up, down}	S_1_st	{ok, broken, short}
S_2_pos	{up, down}	S_2_st	{ok, broken, short}
W_1_st	{ok, broken}	W_1_current	{yes, no}
...	...	...	...

One out of many possible worlds:



S_1_pos = down
S_1_st = ok
S_2_pos = up
S_2_st = ok
W_1_st = ok
W_1_current = no
...

Example: Schedule

Variables:

Teacher_MI	$\{PD, MJ, TDN\}$
Time_MI	{Mo_m, Mo_a, ..., Fr_m, Fr_a}
Room_MI	$\{0.2.12, 0.2.13, 0.2.90\}$
Teacher_AD	$\{PD, MJ, TDN\}$
Time_AD	{Mo_m, Mo_a, ..., Fr_m, Fr_a}
Room_AD	$\{0.2.12, 0.2.13, 0.2.90\}$

One possible world:

Mo	Tue	Wed	Thu	Fr
	MI, TDN, 0.2.13			
			AD, PD, 0.2.90	

Teacher_MI=TDN
Time_MI=Tue_m
Room_MI=0.2.13
Teacher_AD=PD
Time_AD=Thu_a
Room_AD=0.2.90

Constraint Satisfaction Problems

Constraints

A **constraint** is a condition on the values of variables in a possible world.

Extensional Constraint Specification

Explicitly list all allowed (or disallowed) combination of values:

Teacher_MI	Time_MI	Room_MI	Teacher_AD	Time_AD	Room_AD
PD	Mo_m	0.2.12	PD	Mo_a	0.2.12
PD	Mo_m	0.2.12	PD	Mo_a	0.2.13
...	...	...	...	...	...

Not on the list of allowed possible worlds:

Teacher_MI	Time_MI	Room_MI	Teacher_AD	Time_AD	Room_AD
PD	Mo_m	0.2.12	PD	Mo_m	0.2.12
PD	Mo_m	0.2.12	MJ	Mo_m	0.2.12
...	...	...	...	...	...

Constraints

Intensional Constraint Specification

Use logical expressions:

$$Teacher\_AD = Teacher\_MI \rightarrow Time\_AD \neq Time\_MI$$
$$Time\_AD = Time\_MI \rightarrow Room\_AD \neq Room\_MI$$

Example: Sudoku

A1	A2	1	A4	A5	A6	A7	A8	A9
B1	B2	2	B4	3	B6	B7	B8	4
C1	C2	C3	5	C5	C6	6	C8	7
5	D2	D3	1	4	D6	D7	D8	D9
E1	7	E3	E4	E5	E6	E7	2	E9
F1	F2	F3	F4	7	8	F7	F8	9
8	G2	7	G4	G5	9	G7	G8	G9
4	H2	H3	H4	6	H6	3	H8	H9
I1	I2	I3	I4	I5	I6	5	I8	I9

Constraints:

$$A1 = 2 \vee A2 = 2 \vee A4 = 2 \vee A5 = 2 \vee A6 = 2 \vee A7 = 2 \vee A8 = 2 \vee A9 = 2$$
$$A1 = 3 \vee A2 = 3 \vee A4 = 3 \vee A5 = 3 \vee A6 = 3 \vee A7 = 3 \vee A8 = 3 \vee A9 = 3$$
$$\dots$$
$$A1 = 3 \vee A2 = 3 \vee B1 = 3 \vee B2 = 3 \vee C1 = 3 \vee C2 = 3 \vee C3 = 3$$
$$\dots$$

Constraint Satisfaction Problem

A **Constraint Satisfaction Problem (CSP)** is given by

- a set of variables
- a set of constraints (usually intensional)

A **solution** to a CSP consists of a possible world that satisfies all the constraints (also called a **model** of the constraints).

Other tasks:

- Determine the number of models of the constraints
- Find an *optimal* model (given also a value function on possible worlds).
- ...

CSP as State Space Problem

A CSP can be represented as a state space problem:

- States are all partial assignments of values to variables that are consistent with the constraints
- For a state s: select some variable V not assigned a value in s, and let the neighbors of s be all states that assign a value to V (if any exist).
- The start state is the state that does not assign any values
- A goal state is a state that assigns values to all variables

CSP as State Space Problem

A CSP can be represented as a state space problem:

- States are all partial assignments of values to variables that are consistent with the constraints
- For a state s: select some variable V not assigned a value in s, and let the neighbors of s be all states that assign a value to V (if any exist).
- The start state is the state that does not assign any values
- A goal state is a state that assigns values to all variables

Solving the CSP

- A solution to the state space problem is a path with a goal state at the end: a solution to the CSP problem
- To solve the state space problem need only be able to:
  - enumerate all partial assignments that assign a value to one more variable than s
  - check whether a partial assignment is consistent with the constraints

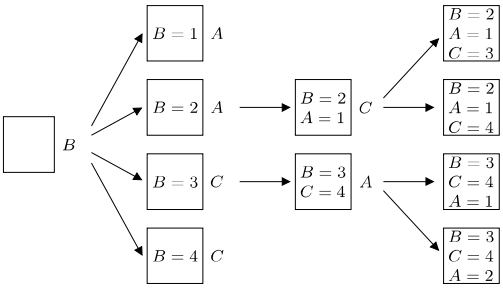
  (that is sufficient to implement the *get_neighbors* and *goal* functions needed in the generic search algorithm)

Example [PM 4.13]

Variables A, B, C; all with domain $\{1, 2, 3, 4\}$.

Constraints: $A < B$, $B < C$.

State Space Graph (showing at each node which variable was selected to generate the neighbors):



- The state space graph is a tree (= search tree)

Consistency Algorithms

Idea

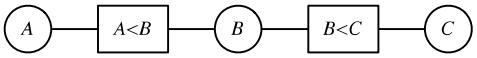
Example: Variables A, B, C; all with domain $\{1, 2, 3, 4\}$.

Constraints: $A < B$, $B < C$.

Observation: There is no solution with $A = 4$.

Approach to solving CSPs: iteratively eliminate value assignments that cannot be part of a solution.

Constraint Network



The **constraint network** for a CSP consists of

- One (oval) node for each variable X
- One (rectangular) node for each constraint c
- An (undirected) arc $\langle X, c \rangle$ between every constraint and every variable involved in the constraint

With each variable node X is associated a **(reduced) domain** D_X:

- Initially the domain of the variable
- Reduced by successively deleting values that cannot be part of a solution

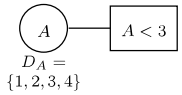
Arc Consistency

An arc $\langle X, c \rangle$ is **arc consistent**, if

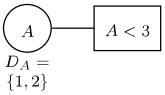
- for all $x \in D_X$ there exists values $y_1, \ldots, y_k$ for the other variables involved in c, such that $x, y_1, \ldots, y_k$ is consistent with c.

A constraint network is **arc consistent**, if all its arcs are arc consistent.

Examples

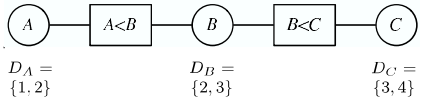


Not arc consistent

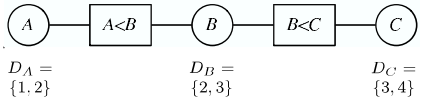


Arc consistent

Arc Consistency Examples

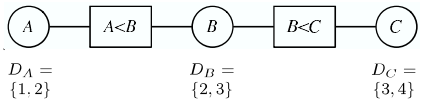


Arc Consistency Examples

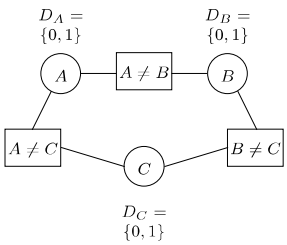


Arc consistent. Not every combination of values from D_A, D_B, D_C is a solution!

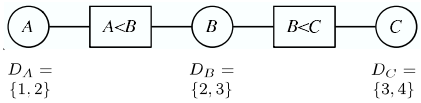
Arc Consistency Examples



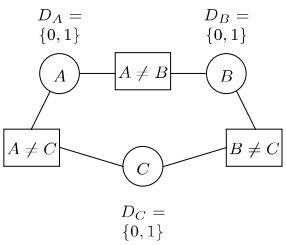
Arc consistent. Not every combination of values from D_A, D_B, D_C is a solution!



Arc Consistency Examples



Arc consistent. Not every combination of values from D_A, D_B, D_C is a solution!



Arc consistent. There exists no solution!

Generalized Arc Consistency Algorithm

Algorithm Outline

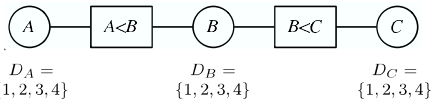
1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



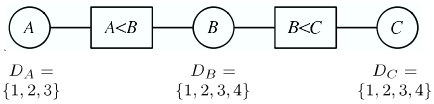
- *To-do-arcs* $= \{\langle A, A < B\rangle, \langle B, A < B\rangle, \langle B, B < C\rangle, \langle C, B < C\rangle\}$
- Selecting $\langle A, A < B\rangle$: For $A = 4$, no value of B satisfies $4 < B$.
  - Remove $\langle A, A < B\rangle$ from *To-do-arcs*.
  - Update D_A.

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



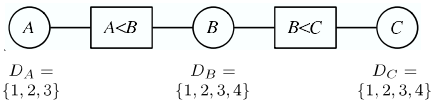
- *To-do-arcs* $= \{\langle B, A < B\rangle, \langle B, B < C\rangle, \langle C, B < C\rangle\}$

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



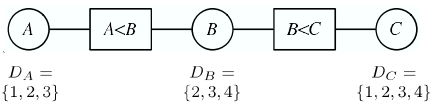
- *To-do-arcs* $= \{\langle B, A < B\rangle, \langle B, B < C\rangle, \langle C, B < C\rangle\}$
- Selecting $\langle B, A < B\rangle$: $B = 1$ can be pruned.
  - Remove $\langle B, A < B\rangle$ from *To-do-arcs*.
  - Update D_B.

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



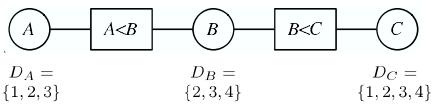
- *To-do-arcs* $= \{\langle B, B < C\rangle, \langle C, B < C\rangle\}$

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



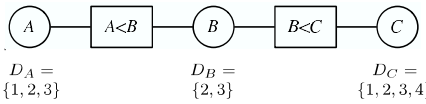
- *To-do-arcs* $= \{\langle B, B < C\rangle, \langle C, B < C\rangle\}$
- Selecting $\langle B, B < C\rangle$: $B = 4$ can be pruned.
  - Add $\langle A, A < B\rangle$ to *To-do-arcs* and remove $\langle B, B < C\rangle$.
  - Update D_B.

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



$D_A = \{1, 2, 3\}$ $D_B = \{2, 3\}$ $D_C = \{1, 2, 3, 4\}$

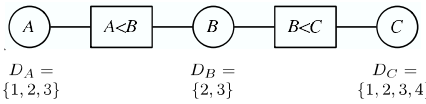
- *To-do-arcs* $= \{\langle A, A < B\rangle, \langle C, B < C\rangle\}$

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



$D_A = \{1, 2, 3\}$ $D_B = \{2, 3\}$ $D_C = \{1, 2, 3, 4\}$

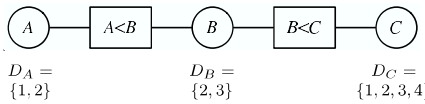
- *To-do-arcs* $= \{\langle A, A < B\rangle, \langle C, B < C\rangle\}$
- Selecting $\langle A, A < B\rangle$: $A = 3$ can be pruned.
  - Remove $\langle A, A < B\rangle$ from *To-do-arcs*.
  - Update D_A.

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



$D_A = \{1, 2\}$ $D_B = \{2, 3\}$ $D_C = \{1, 2, 3, 4\}$

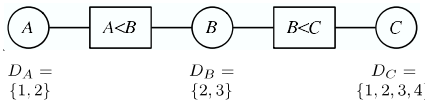
- *To-do-arcs* $= \{\langle C, B < C\rangle\}$

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



$D_A = \{1, 2\}$ $D_B = \{2, 3\}$ $D_C = \{1, 2, 3, 4\}$

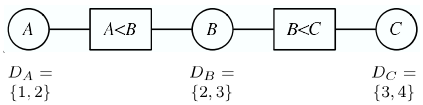
- *To-do-arcs* $= \{\langle C, B < C\rangle\}$
- Selecting $\langle C, B < C\rangle$: $C = 1$ and $C = 2$ can be pruned.
  - Remove $\langle B, B < C\rangle$ from *To-do-arcs*.
  - Update D_C.

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Example



$D_A = \{1, 2\}$ $D_B = \{2, 3\}$ $D_C = \{3, 4\}$

- *To-do-arcs* $= \{\}$
- DONE!

Generalized Arc Consistency Algorithm

Algorithm Outline

1. *To-do-arcs*= all arcs in constraint network // Potentially inconsistent arcs
2. **while** *To-do-arcs* $\neq \emptyset$
3. select and delete one arc $\langle X, c\rangle$ from *To-do-arcs*
4. make arc consistent by deleting values from D_X, if necessary
5. **if** values were deleted: add all other arcs $\langle Z, c'\rangle$ ($c \neq c'$, $X \in dom(c')$) to *To-do-arcs*

Algorithm Outcomes

Algorithm is guaranteed to terminate. Result independent of order in which arcs are processed. Possible cases at termination:

- $D_X = \emptyset$ for some X: CSP has no solution
- D_X contains exactly one value for each X: CSP has unique solution, given by the D_X values.
- Other: if the CSP has a solution, then the solution can only consist of current D_X values.

Variable Elimination

Idea

- Arc Consistency: simplify problem by eliminating values
- Variable Elimination: simplify problem by eliminating variables

Relational Operations

Variable Elimination operates on extensional (table) representations of constraints:

$A < B$:

A	B
1	2
1	3
1	4
2	3
2	4
3	4

$B < C$:

B	C
1	2
1	3
1	4
2	3
2	4
3	4

Algorithm requires **projection** and **join** operations on tables.

Projection

Projection of a table:

Course	Year	Student	Grade
cs322	2008	fran	77
cs111	2009	billie	88
cs111	2009	jess	78
cs444	2008	fran	83
cs322	2009	jordan	92

$\pi_{\{Student, Year\}}$ ⟶

Student	Year
fran	2008
billie	2009
jess	2009
jordan	2009

Join

Given two tables r_1, r_2 for variables $vars_1, vars_2$. The **join** is the table $r_3 = r_1 \bowtie r_2$ for variables $vars_1 \cup vars_2$ that

- contains all tuples, which restricted to $vars_1$ are in r_1, and restricted to $vars_2$ are in r_2.

Example

Course	Year	Student	Grade
cs322	2008	fran	77
cs111	2009	billie	88
cs111	2009	jess	78
cs444	2008	fran	83
cs322	2009	jordan	92

$\bowtie$

Course	Year	TA
cs322	2008	yuki
cs111	2009	sam
cs111	2009	chris
cs322	2009	yuki

=

Course	Year	Student	Grade	TA
cs322	2008	fran	77	yuki
cs111	2009	billie	88	sam
cs111	2009	jess	78	sam
cs111	2009	billie	88	chris
cs111	2009	jess	78	chris
cs322	2009	jordan	92	yuki

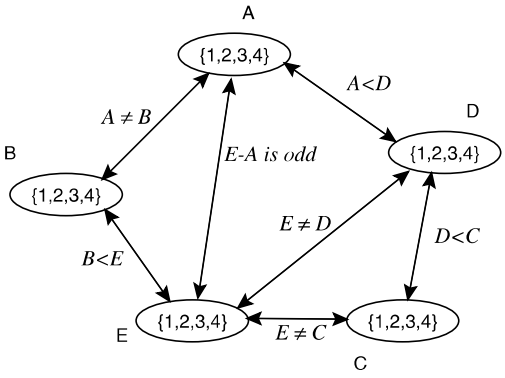
Variable Elimination Algorithm

Algorithm Outline

1. **Input:** C: set of constraints on variables *vars*
2. **while** C contains more than one element
3. select a variable $X \in vars$
4. delete X from *vars*
5. remove all constraints involving X from C and construct their join
6. **if** *vars* is not empty
7. project the join onto the variables other than X
8. add the (projected) join to C

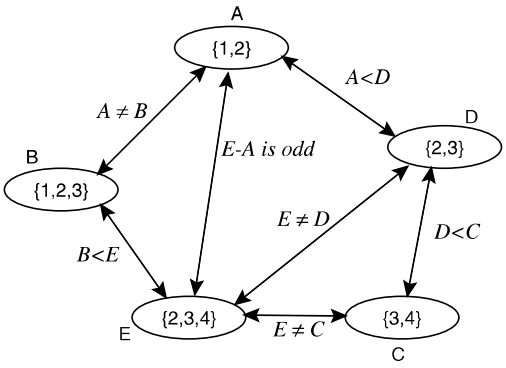
Intuition: the constraint constructed in line 5. summarizes the effect that all the constraints involving X have on variables other than X.

Example network



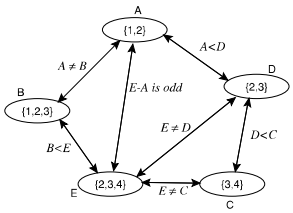
Example network

...now arc-consistent



Example network

Example: eliminating C



$r_1 : C \neq E$	C	E
	3	2
	3	4
	4	2
	4	3

$r_2 : C > D$	C	D
	3	2
	4	2
	4	3

Example network

Example: eliminating C

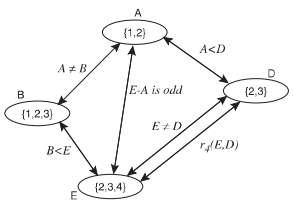
$r_1 : C \neq E$	C	E
	3	2
	3	4
	4	2
	4	3

$r_2 : C > D$	C	D
	3	2
	4	2
	4	3

$r_3 : r_1 \bowtie r2$	C	D	E
	3	2	2
	3	2	4
	4	2	2
	4	2	3
	4	3	2
	4	3	3

Example network

Example: eliminating C



$r_1 : C \neq E$	C	E
	3	2
	3	4
	4	2
	4	3

$r_2 : C > D$	C	D
	3	2
	4	2
	4	3

$r_3 : r_1 \bowtie r2$	C	D	E
	3	2	2
	3	2	4
	4	2	2
	4	2	3
	4	3	2
	4	3	3

$r_4 : \pi_{\{D,E\}} r_3$	D	E
	2	2
	2	3
	2	4
	3	2
	3	3

↪ new constraint

VE Properties

Properties

- The algorithm terminates
- The CSP has a solution if and only if the final constraint is non-empty
- The set of all solutions can be generated by joining the final constraint with the intermediate "summarizing" constraints generated in line 5.
- Algorithm operates on extensional constraint representations, therefore
  - constraints must not contain too many tuples (initial and constructed constraints)
- Worst case: VE is not more efficient than enumerating all possible worlds and checking whether they are solutions.

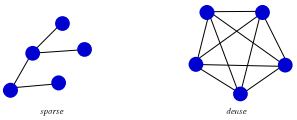
VE Properties

Properties

- The algorithm terminates
- The CSP has a solution if and only if the final constraint is non-empty
- The set of all solutions can be generated by joining the final constraint with the intermediate "summarizing" constraints generated in line 5.
- Algorithm operates on extensional constraint representations, therefore
  - constraints must not contain too many tuples (initial and constructed constraints)
- Worst case: VE is not more efficient than enumerating all possible worlds and checking whether they are solutions.

Constraint Graph

Consider the graph where
- there is one node for each variable
- two variables are connected when they appear together in one constraint



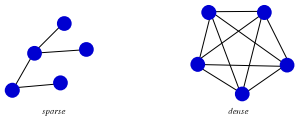
VE Properties

Properties

- The algorithm terminates
- The CSP has a solution if and only if the final constraint is non-empty
- The set of all solutions can be generated by joining the final constraint with the intermediate "summarizing" constraints generated in line 5.
- Algorithm operates on extensional constraint representations, therefore
  - constraints must not contain too many tuples (initial and constructed constraints)
- Worst case: VE is not more efficient than enumerating all possible worlds and checking whether they are solutions.

Constraint Graph

Consider the graph where
- there is one node for each variable
- two variables are connected when they appear together in one constraint



Then: VE will work better if the constraint graph is sparsely connected

Autumn 2019 Variable Elimination 27

Local Search

Systematic vs. Local Search

So far: all methods systematically explored the state space (possible worlds).

Problem: Time and space when search space is large.

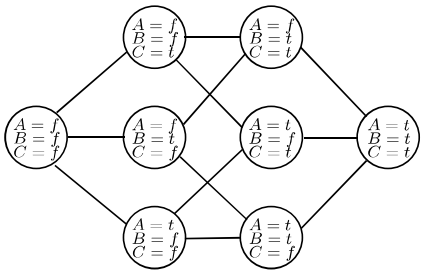
Local Search approach:
- explore state space without 'bookkeeping' (where have we been? what still needs to be explored?).
- no success/termination guarantees
- in practice, often the only thing that works

Autumn 2019 Local Search 28

State Space Graph for CSP

(Another) state space graph representation for CSPs:
- Nodes are possible worlds
- Neighbors are possible worlds that differ in the value of exactly one variable

State space graph for 3 boolean variables:



Local Search Approach

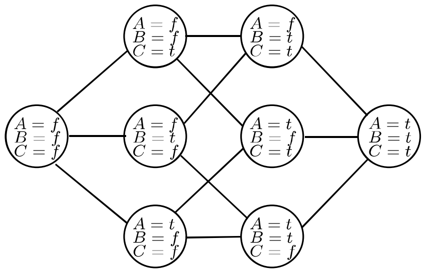
Algorithm Outline

1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*

Local Search Approach

Algorithm Outline

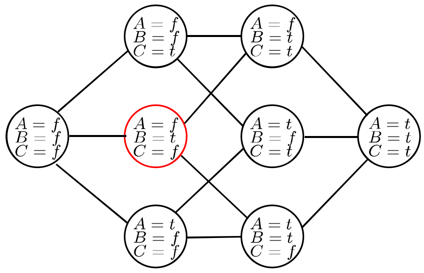
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

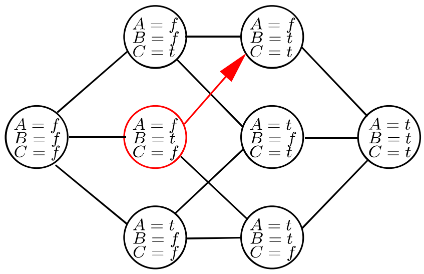
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

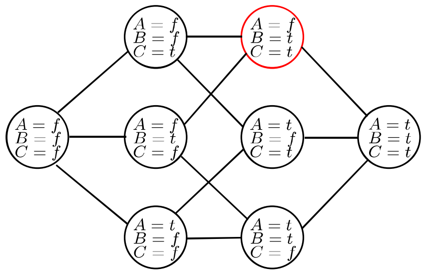
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

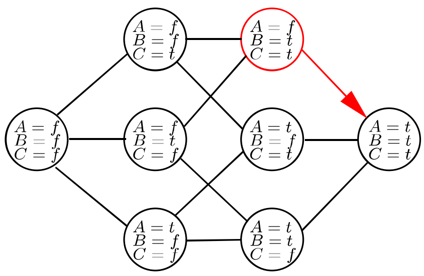
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

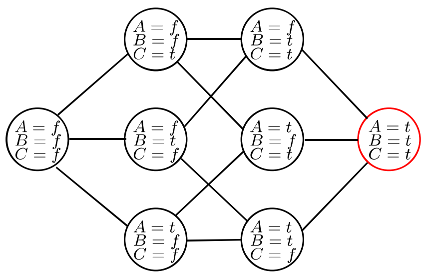
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

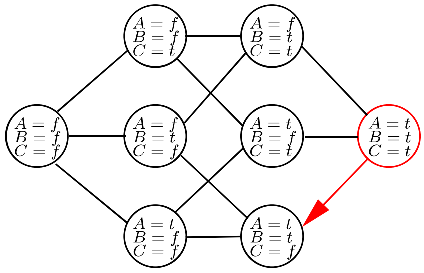
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

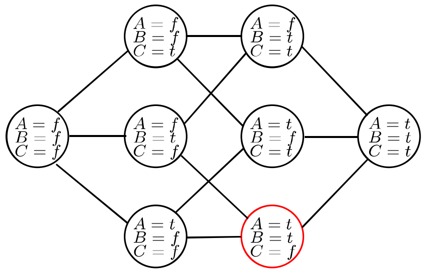
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

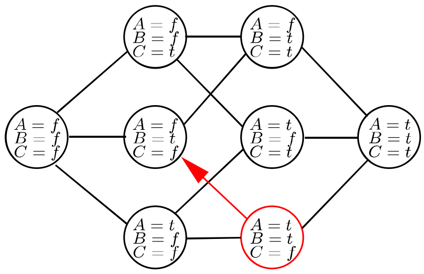
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

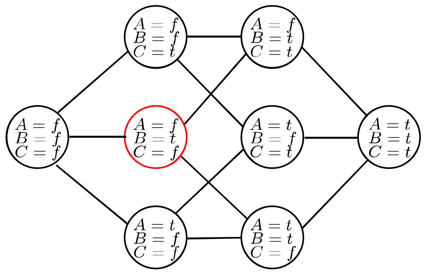
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

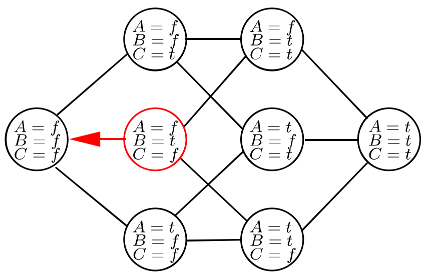
1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Local Search Approach

Algorithm Outline

1.Select some node in state space graph as *current state*
2.**while** *current state* is not a solution
3. *current state* = some neighbor of *current state*



Random Search

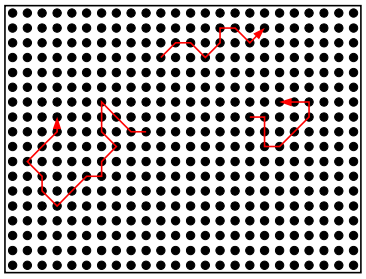
- Make choices in line 1. and 3. completely random
- “Random walk”
- Unlikely to find a solution if state space large with only few solutions

Greedy Search

Greedy Search or **Hill Climbing**:

- Use an *evaluation function* on states
- Example for evaluation function: number of constraints not satisfied by state
- Always choose neighbor with minimal evaluation function value
- Terminates when all neighbors have higher value than current state: current state is a **local minimum**.

Possible greedy search paths starting from different states:



Escaping Local Minima

Problem

- Search terminates with local minimum of evaluation function. This may not be a solution to the CSP.

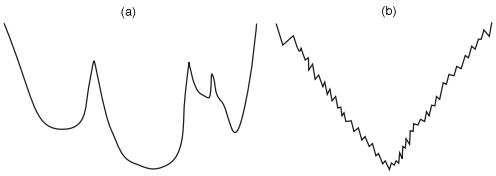
Escaping Local Minima

Problem

- Search terminates with local minimum of evaluation function. This may not be a solution to the CSP.

Solution Approaches

- Random restarts: repeat greedy search with several randomly chosen initial states
- Random moves: combine greedy moves with random steps



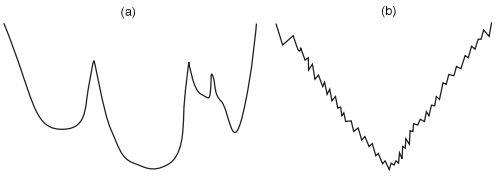
Escaping Local Minima

Problem

- Search terminates with local minimum of evaluation function. This may not be a solution to the CSP.

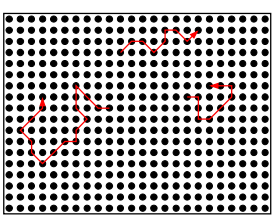
Solution Approaches

- Random restarts: repeat greedy search with several randomly chosen initial states
- Random moves: combine greedy moves with random steps
- **Example (a)**: Small number of random restarts will find global minimum
- **Example (b)**: Make random move when local minimum reached



Local search

- Maintain an assignment of a value to each variable.
- At each step, select a "neighbor" of the current assignment (e.g., one that improves some heuristic value).
- Stop when a satisfying assignment is found, or return the best assignment found.



Requires:

- What is a neighbor?
- Which neighbor should be selected?

Most improving step

Principle

Select the variable-value pair that gives the highest improvement.

Naive approach

- Linearly scan all variables and for each value of each variable determine the improvement (how many fewer constrains are violated).

Most improving step

Principle

Select the variable-value pair that gives the highest improvement.

Naive approach

- Linearly scan all variables and for each value of each variable determine the improvement (how many fewer constrains are violated).

Alternative

- Maintain a priority queue with variable-value pairs not part of the current assignment.
- Weight$\langle X, v\rangle$ = *eval(current assignment)* − *eval(current assignment but with* $X = v$).
- If X is given a new value, update the weight of all pairs participating in a changed constraint.

Two-stage choice

Principle

- First: choose variable
- Second: choose state

Data structure

- Maintain priority queue of variables; weight is the number of participating conflicts.
- After selecting a variable, pick the value minimizes the number of conflicts.
- Update weights of variables that participate in a conflict that is changed.

Simulated annealing

Algorithm

- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
  - With current assignment n and proposed assignment n' we move to n' with probability

$$e^{(h(n')-h(n))/T}$$

- Reduce the temperature.

Simulated annealing

Algorithm

- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
  - With current assignment n and proposed assignment n' we move to n' with probability

$$e^{(h(n')-h(n))/T}$$

- Reduce the temperature.

Probability of accepting a change

Temperature	1-worse	2-worse	3-worse
10	0.91	0.81	0.74
1	0.37	0.14	0.05
0.25	0.02	0.0003	0.000005
0.1	0.00005	0	0

Simulated annealing

Algorithm

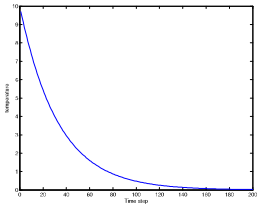
- Pick a variable at random and a new value at random.
- If it is an improvement, adopt it.
- If it isn't an improvement, adopt it probabilistically depending on a temperature parameter, T.
  - With current assignment n and proposed assignment n' we move to n' with probability

$$e^{(h(n')-h(n))/T}$$

- Reduce the temperature.

Probability of accepting a change

Temperature	1-worse	2-worse	3-worse
10	0.91	0.81	0.74
1	0.37	0.14	0.05
0.25	0.02	0.0003	0.000005
0.1	0.00005	0	0



$$y = 10 \cdot 0.97^x$$

Propositional Logic Basics

Intensional Constraints

Previously ...

Intensional representation of constraints:

$A < B$
Teacher_AD $=$ *Teacher_MI* $\rightarrow$ *Time_AD* $\neq$ *Time_MI*
...

CSP algorithms (arc-consistency algorithm) need to perform certain operations:

- *test* whether a certain value for one variable is *consistent* with a given constraint (and certain values for other variables)

To implement this:

- need a **formal language for representing constraints**

Propositional Logic

- provides a formal language for representing constraints on *binary variables*.

Propositional Logic: Syntax

Atomic Propositions

Boolean variables are now seen as **atomic propositions**. Convention: start with lowercase letter.

Constraints	Logic
$A =$ *true*	a
$A =$ *false*	$\neg a$

Propositions

Using **logical connectives** more complex propositions are constructed:

$\neg p$	**not** p
$(p \wedge q)$	p **and** q
$(p \vee q)$	p **or** q
$(p \rightarrow q)$	p **implies** q

A *set* of propositions is also called a **Knowledge Base**

Example

"If **it rains** I'll **take my umbrella**, or I'll **stay home**"

Propositional Logic: Syntax

Atomic Propositions

Boolean variables are now seen as **atomic propositions**. Convention: start with lowercase letter.

Constraints	Logic
$A =$ *true*	a
$A =$ *false*	$\neg a$

Propositions

Using **logical connectives** more complex propositions are constructed:

$\neg p$	**not** p
$(p \wedge q)$	p **and** q
$(p \vee q)$	p **or** q
$(p \rightarrow q)$	p **implies** q

A *set* of propositions is also called a **Knowledge Base**

Example

"If **it rains** I'll **take my umbrella**, or I'll **stay home**"

rains $\rightarrow$ (*umbrella* $\vee$ *home*)

Propositional Logic: Semantics I

An **interpretation** π for a set of atomic propositions $a_1, a_2, \ldots, a_n$ is an assignment of a truth value to each proposition (= possible world when atomic propositions seen as boolean variables):

$$\pi(a_i) \in \{\textit{true}, \textit{false}\}$$

An interpretation defines a truth value for all propositions:

$\pi(p)$	$\pi(\neg p)$
true	*false*
false	*true*

$\pi(p)$	$\pi(q)$	$\pi(p \wedge q)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*false*
false	*false*	*false*

$\pi(p)$	$\pi(q)$	$\pi(p \vee q)$
true	*true*	*true*
true	*false*	*true*
false	*true*	*true*
false	*false*	*false*

$\pi(p)$	$\pi(q)$	$\pi(p \rightarrow q)$
true	*true*	*true*
true	*false*	*false*
false	*true*	*true*
false	*false*	*true*

Propositional Logic: Semantics II

Models

A **model** of a proposition (a knowledge base) is an interpretation in which the proposition (all the propositions in the knowledge base) is true.

Propositions as constraints: a model is a possible world that satisfies the constraint.

Logical consequence

A proposition g is a **logical consequence** of a knowledge base *KB*, if every model of *KB* is a model of g. Written:

$$KB \models g$$

(whenever *KB* is true, then g also is true).

Example

KB=$\{$*man* $\rightarrow$ *mortal*, *man*$\}$. Then

$$KB \models \textit{mortal}$$

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	p	q	r	s
I_1	*true*	*true*	*true*	*true*
I_2	*false*	*false*	*false*	*false*
I_3	*true*	*true*	*false*	*false*
I_4	*true*	*true*	*true*	*false*
I_5	*true*	*true*	*false*	*true*

Model?

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	p	q	r	s	Model?
I_1	*true*	*true*	*true*	*true*	is a model of KB
I_2	*false*	*false*	*false*	*false*	not a model of KB
I_3	*true*	*true*	*false*	*false*	is a model of KB
I_4	*true*	*true*	*true*	*false*	is a model of KB
I_5	*true*	*true*	*false*	*true*	not a model of KB

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	p	q	r	s	Model?
I_1	*true*	*true*	*true*	*true*	is a model of KB
I_2	*false*	*false*	*false*	*false*	not a model of KB
I_3	*true*	*true*	*false*	*false*	is a model of KB
I_4	*true*	*true*	*true*	*false*	is a model of KB
I_5	*true*	*true*	*false*	*true*	not a model of KB

Which of p, q, r, q logically follow from KB?

Simple Example

$$KB = \begin{cases} p \leftarrow q. \\ q. \\ r \leftarrow s. \end{cases}$$

	p	q	r	s	Model?
I_1	*true*	*true*	*true*	*true*	is a model of KB
I_2	*false*	*false*	*false*	*false*	not a model of KB
I_3	*true*	*true*	*false*	*false*	is a model of KB
I_4	*true*	*true*	*true*	*false*	is a model of KB
I_5	*true*	*true*	*false*	*true*	not a model of KB

Which of p, q, r, q logically follow from KB?

$$KB \models p, KB \models q, KB \not\models r, KB \not\models s$$

Machine Intelligence

Lecture 4: Reasoning under uncertainty - probabilities

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- **Representing domains endowed with uncertainty**
- Bayesian networks
- Machine learning
- Planning
- Multi-agent systems

Certainty

Certainty in Search

Assumptions for using search for solving planning problems:

- Current state is fully known
- Actions have deterministic effects

Certainty in CSP and Logic

- Possible worlds are possible or impossible (according to given constraints/propositions)
- Propositions are fully known/believed, or unknown
- Generalization: soft constraints – possible worlds are more or less desirable

Uncertainty

More realistic scenarios

- Agents do not observe the world perfectly
- Actions have uncertain effects
- Propositions are believed only with a certain confidence

Degrees of Belief for Propositions

In reality, states of knowledge may better be represented by degrees of belief:

$$\begin{aligned} &\mathit{Bel}(\mathit{light\_on} \leftarrow \mathit{switch\_on} \wedge \mathit{breaker\_up}) = 0.7 \\ &\mathit{Bel}(\neg \mathit{umbrella} \rightarrow \mathit{rain}) = 1.5 \\ &\mathit{Bel}(\mathit{umbrella} \rightarrow \mathit{rain}) = 0.2 \\ &\mathit{Bel}(\mathit{global\_warming}) = 0.8 \end{aligned}$$

Question: what rules must (rational) degrees of belief obey?

Basic Probability Calculus

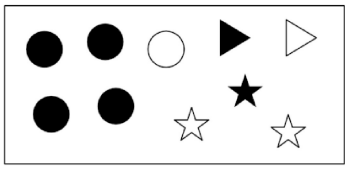
Probability Measures

Probability theory is built on the foundation of variables and worlds.

Worlds described by the variables:

- Filled: {true, false}
- Shape: {circle, triangle, star}

as well as position.



Probability measures

Ω: set of all possible worlds (for a given, fixed set of variables). A **probability measure over** Ω, is a function P, that assigns **probability values**

$$P(\Omega') \in [0, 1]$$

to subsets $\Omega' \subseteq \Omega$, such that

Axiom 1: $P(\Omega) = 1$.

Axiom 2: if $\Omega_1 \cap \Omega_2 = \emptyset$, then $P(\Omega_1 \cup \Omega_2) = P(\Omega_1) + P(\Omega_2)$.

Simplification for finite Ω

If all variables have a finite domain, then

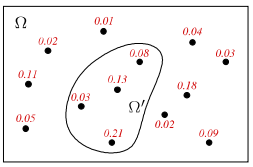
- Ω is finite, and
- a probability distribution is defined by assigning a probability value

$$P(\omega)$$

to each individual possible world $\omega \in \Omega$.

For any $\Omega' \subseteq \Omega$ then

$$P(\Omega') = \sum_{\omega \in \Omega'} P(\omega)$$



$P(\Omega') = 0.08 + 0.13 + 0.03 + 0.21 = 0.45$

From now on, we will only consider variables with finite domains.

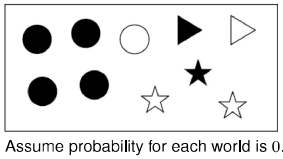
Probabilities of propositions

Probabilities of Propositions

A probability distribution over possible worlds defines probabilities for propositions α:

$$\begin{aligned} P(\alpha) &= P(\{\omega \in \Omega \mid \omega \models \alpha\}) \\ &= \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega) \end{aligned}$$

Example



Assume probability for each world is 0.1:

- $P(\mathit{Shape} = \mathit{circle}) =$

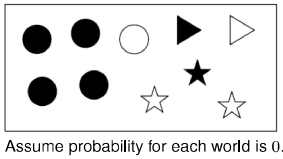
Probabilities of propositions

Probabilities of Propositions

A probability distribution over possible worlds defines probabilities for propositions α:

$$\begin{aligned} P(\alpha) &= P(\{\omega \in \Omega \mid \omega \models \alpha\}) \\ &= \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega) \end{aligned}$$

Example



Assume probability for each world is 0.1:

- $P(\mathit{Shape} = \mathit{circle}) = 0.5$
- $P(\mathit{Filled} = \mathit{false}) =$

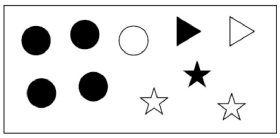
Probabilities of propositions

Probabilities of Propositions

A probability distribution over possible worlds defines probabilities for propositions α:

$$P(\alpha) = P(\{\omega \in \Omega \mid \omega \models \alpha\}) = \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega)$$

Example



Assume probability for each world is 0.1:

- $P(\textit{Shape} = \textit{circle}) = 0.5$
- $P(\textit{Filled} = \textit{false}) = 0.4$
- $P(\textit{Shape} = \textit{c} \wedge \textit{Filled} = \textit{f}) =$

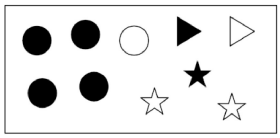
Probabilities of propositions

Probabilities of Propositions

A probability distribution over possible worlds defines probabilities for propositions α:

$$P(\alpha) = P(\{\omega \in \Omega \mid \omega \models \alpha\}) = \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega)$$

Example



Assume probability for each world is 0.1:

- $P(\textit{Shape} = \textit{circle}) = 0.5$
- $P(\textit{Filled} = \textit{false}) = 0.4$
- $P(\textit{Shape} = \textit{c} \wedge \textit{Filled} = \textit{f}) = 0.1$

MI Autumn 2019 Basic Probability Calculus 7

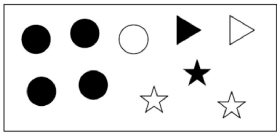
Probabilities of propositions

Probabilities of Propositions

A probability distribution over possible worlds defines probabilities for propositions α:

$$P(\alpha) = P(\{\omega \in \Omega \mid \omega \models \alpha\}) = \sum_{\omega : \alpha \text{ is true in } \omega} P(\omega)$$

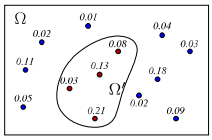
Example



Assume probability for each world is 0.1:

- $P(\textit{Shape} = \textit{circle}) = 0.5$
- $P(\textit{Filled} = \textit{false}) = 0.4$
- $P(\textit{Shape} = \textit{c} \wedge \textit{Filled} = \textit{f}) = 0.1$

Another example



$$P(\textit{Color} = \textit{red}) = 0.08 + 0.13 + 0.03 + 0.21 = 0.45$$

MI Autumn 2019 Basic Probability Calculus 7

Basic probability axioms

Axiom

If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B})$.

Example

Consider a deck with 52 cards. If $\mathcal{A} = \{2, 3, 4, 5\}$ and $\mathcal{B} = \{7, 8\}$, then

$$P(\mathcal{A} \cup \mathcal{B}) =$$

MI Autumn 2019 Basic Probability Calculus 8

Basic probability axioms

Axiom

If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B})$.

Example

Consider a deck with 52 cards. If $\mathcal{A} = \{2, 3, 4, 5\}$ and $\mathcal{B} = \{7, 8\}$, then

$$P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) = \frac{4}{13} + \frac{2}{13} = \frac{6}{13}.$$

MI Autumn 2019 Basic Probability Calculus 8

Basic probability axioms

Axiom

If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B})$.

Example

Consider a deck with 52 cards. If $\mathcal{A} = \{2, 3, 4, 5\}$ and $\mathcal{B} = \{7, 8\}$, then

$$P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) = \frac{4}{13} + \frac{2}{13} = \frac{6}{13}.$$

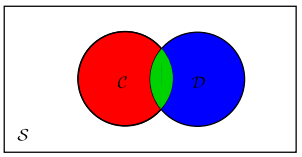
More generally

If $\mathcal{C}$ and $\mathcal{D}$ are not disjoint, then $P(\mathcal{C} \cup \mathcal{D}) = P(\mathcal{C}) + P(\mathcal{D}) - P(\mathcal{C} \cap \mathcal{D})$.

Example

If $\mathcal{C} = \{2, 3, 4, 5\}$ and $\mathcal{D} = \{\spadesuit\}$, then

$$P(\mathcal{C} \cup \mathcal{D}) =$$



MI Autumn 2019 Basic Probability Calculus 8

Basic probability axioms

Axiom

If $\mathcal{A}$ and $\mathcal{B}$ are disjoint, then $P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B})$.

Example

Consider a deck with 52 cards. If $\mathcal{A} = \{2, 3, 4, 5\}$ and $\mathcal{B} = \{7, 8\}$, then

$$P(\mathcal{A} \cup \mathcal{B}) = P(\mathcal{A}) + P(\mathcal{B}) = \frac{4}{13} + \frac{2}{13} = \frac{6}{13}.$$

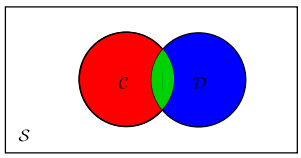
More generally

If $\mathcal{C}$ and $\mathcal{D}$ are not disjoint, then $P(\mathcal{C} \cup \mathcal{D}) = P(\mathcal{C}) + P(\mathcal{D}) - P(\mathcal{C} \cap \mathcal{D})$.

Example

If $\mathcal{C} = \{2, 3, 4, 5\}$ and $\mathcal{D} = \{\spadesuit\}$, then

$$P(\mathcal{C} \cup \mathcal{D}) = \frac{4}{13} + \frac{1}{4} - \frac{4}{52} = \frac{25}{52}.$$



Probability Updating

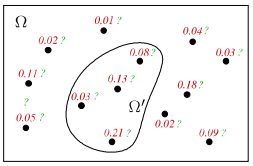
- Given new information (**evidence**), degrees of belief change.
- Evidence can consist of:
  - learning that a certain proposition p is true ("*switch_up*")
  - measuring the value of some variable ("room_ai = 0.2.90")
  - obtaining partial information on the value of some variable ("room_ai $\neq$ 0.2.90")
  - ...
- In all cases: evidence can be represented as the set of possible world Ω' not ruled out by the observation.

How should the probabilities be updated, when I observe Ω'?:

Probability Updating

- Given new information (**evidence**), degrees of belief change.
- Evidence can consist of:
  - learning that a certain proposition p is true ("*switch_up*")
  - measuring the value of some variable ("room_ai = 0.2.90")
  - obtaining partial information on the value of some variable ("room_ai $\neq$ 0.2.90")
  - ...
- In all cases: evidence can be represented as the set of possible world Ω' not ruled out by the observation.

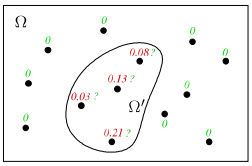
How should the probabilities be updated, when I observe Ω'?:



Probability Updating

- Given new information (**evidence**), degrees of belief change.
- Evidence can consist of:
  - learning that a certain proposition p is true ("*switch_up*")
  - measuring the value of some variable ("room_ai = 0.2.90")
  - obtaining partial information on the value of some variable ("room_ai $\neq$ 0.2.90")
  - ...
- In all cases: evidence can be represented as the set of possible world Ω' not ruled out by the observation.

How should the probabilities be updated, when I observe Ω'?:

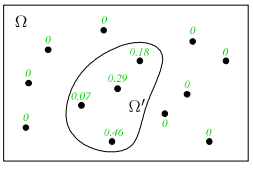


- worlds that are not consistent with evidence have probability 0

Probability Updating

- Given new information (**evidence**), degrees of belief change.
- Evidence can consist of:
  - learning that a certain proposition p is true ("*switch_up*")
  - measuring the value of some variable ("room_ai = 0.2.90")
  - obtaining partial information on the value of some variable ("room_ai $\neq$ 0.2.90")
  - ...
- In all cases: evidence can be represented as the set of possible world Ω' not ruled out by the observation.

How should the probabilities be updated, when I observe Ω'?:



- worlds that are not consistent with evidence have probability 0
- the probabilities of worlds consistent with evidence are proportional to their probability before observation, and they must sum to 1

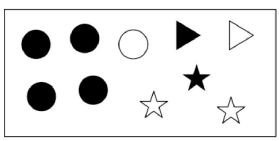
Conditional probability

Definition

The **conditional probability** of proposition p given e is

$$P(p \mid e) = \frac{P(p \wedge e)}{P(e)}$$

Example



(probability for each world is 0.1)

$$P(\text{S=circ.} \mid \text{Fill} = f) = \frac{P(\text{S=circ.} \wedge \text{Fill} = f)}{P(\text{Fill} = f)}$$

$$= \frac{0.1}{0.4} = 0.25$$

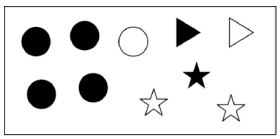
Conditional probability

Definition

The **conditional probability** of proposition p given e is

$$P(p \mid e) = \frac{P(p \wedge e)}{P(e)}$$

Example



(probability for each world is 0.1)

$$P(\textit{S=circ.} \mid \textit{Fill} = f) = \frac{P(\textit{S=circ.} \wedge \textit{Fill} = f)}{P(\textit{Fill} = f)}$$
$$= \frac{0.1}{0.4} = 0.25$$

What is the probability of $P(\textit{S=star} \mid \textit{Fill} = f)$?

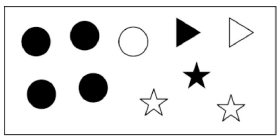
Conditional probability

Definition

The **conditional probability** of proposition p given e is

$$P(p \mid e) = \frac{P(p \wedge e)}{P(e)}$$

Example

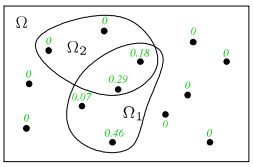


(probability for each world is 0.1)

$$P(\textit{S=circ.} \mid \textit{Fill} = f) = \frac{P(\textit{S=circ.} \wedge \textit{Fill} = f)}{P(\textit{Fill} = f)}$$
$$= \frac{0.1}{0.4} = 0.25$$

What is the probability of $P(\textit{S=star} \mid \textit{Fill} = f)$?

Another example



- e and p are represented by possible worlds Ω_1 and Ω_2
- division by $P(\Omega_1)$ already in green numbers

$$P(\Omega_2 \mid \Omega_1) = 0.18 + 0.29$$

Two important rules

Bayes rule

For propositions p, e:

$$P(p \mid e) = \frac{P(e \wedge p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e)}$$

Two important rules

Bayes rule

For propositions p, e:

$$P(p \mid e) = \frac{P(e \wedge p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e)}$$

Example

A doctor observes symptoms and wishes to find the probability of a disease:

$$P(\textit{disease} \mid \textit{symp.}) = \frac{P(\textit{symp.} \mid \textit{disease})P(\textit{disease})}{P(\textit{symp.})}$$

Two important rules

Bayes rule

For propositions p, e:

$$P(p \mid e) = \frac{P(e \wedge p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e \wedge p) + P(e \wedge \neg p)}$$

Example

A doctor observes symptoms and wishes to find the probability of a disease:

$$P(\textit{disease} \mid \textit{symp.}) = \frac{P(\textit{symp.} \mid \textit{disease})P(\textit{disease})}{P(\textit{symp.})}$$

Two important rules

Bayes rule

For propositions p, e:

$$P(p \mid e) = \frac{P(e \wedge p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e)} = \frac{P(e \mid p)P(p)}{P(e \wedge p) + P(e \wedge \neg p)}$$

Chain rule

For propositions $p_1, \ldots, p_n$:

$$P(p_1 \wedge \ldots \wedge p_n) = P(p_1)P(p_2 \mid p_1) \cdots P(p_i \mid p_1 \wedge \ldots \wedge p_{i-1}) \cdots P(p_n \mid p_1 \wedge \ldots \wedge p_{n-1})$$

Both rules are immediate consequences of the definition of conditional probability!

Random Variables and Distributions

Random Variables

Variables defining possible worlds on which probabilities are defined are called **random variables.**

Distributions

For a random variable A, and $a \in D_A$ we have the probability

$$P(A = a) = P(\{\omega \in \Omega \mid A = a \text{ in } \omega\})$$

The **probability distribution of** A is the function on D_A that maps a to $P(A = a)$. The distribution of A is denoted

$$P(A)$$

Joint Distributions

Extension to several random variables:

$$P(A_1, \ldots, A_k)$$

is the **joint distribution of** $A_1, \ldots, A_k$. The joint distribution maps tuples $(a_1, \ldots, a_k)$ with $a_i \in D_{A_i}$ to the probability

$$P(A_1 = a_1, \ldots, A_k = a_k)$$

Chain rule for distributions

With random variables instead of propositions, the chain rule becomes:

$$P(A_1, \ldots, A_n) = P(A_1)P(A_2 \mid A_1) \cdots P(A_i \mid A_1, \ldots, A_{i-1}) \cdots P(A_n \mid A_1, \ldots, A_{n-1})$$

Note: each $P(p_i \mid p_1 \wedge \ldots \wedge p_{i-1})$ was a *number*. Each $P(A_i \mid A_1, \ldots, A_{i-1})$ is a *function* on tuples $(a_1, \ldots, a_i)$.

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Example

Consider the variables

- Temp : $\mathrm{sp}(\text{Temp}) = \{l, m, h\}$
- Sensor : $\mathrm{sp}(\text{Sensor}) = \{l, m, h\}$

$P(\text{Temp}) = (0.1, 0.6, 0.3)$

$P(\text{Sensor}|\text{Temp}) =$

		Temp		
		l	m	h
Sensor	l	0.8	0.1	0.05
	m	0.15	0.8	0.1
	h	0.05	0.1	0.85

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Example

Consider the variables

- Temp : $\mathrm{sp}(\mathsf{Temp}) = \{l, m, h\}$
- Sensor : $\mathrm{sp}(\mathsf{Sensor}) = \{l, m, h\}$

Assume we observe $S = \mathsf{low}$:
$P(T|\mathsf{low}) = \frac{P(\mathsf{low}|T)P(T)}{\sum_T P(\mathsf{low}|T)P(T)} =$

$P(\mathsf{Temp}) = (0.1, 0.6, 0.3)$

$P(\mathsf{Sensor}|\mathsf{Temp}) =$

Sensor \ Temp	l	m	h
l	0.8	0.1	0.05
m	0.15	0.8	0.1
h	0.05	0.1	0.85

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Example

Consider the variables

- Temp : $\mathrm{sp}(\mathsf{Temp}) = \{l, m, h\}$
- Sensor : $\mathrm{sp}(\mathsf{Sensor}) = \{l, m, h\}$

Assume we observe $S = \mathsf{low}$:
$P(T|\mathsf{low}) = \frac{P(\mathsf{low}|T)P(T)}{\sum_T P(\mathsf{low}|T)P(T)} =$

$P(\mathsf{Temp}) = (0.1, 0.6, 0.3)$

$P(\mathsf{Sensor}|\mathsf{Temp}) =$

Sensor \ Temp	l	m	h
l	0.8	0.1	0.05
m	0.15	0.8	0.1
h	0.05	0.1	0.85

$P(\mathsf{low}|T)P(T) =$

$S = \mathsf{low}$	l	m	h
	0.08	0.06	0.015

$\rightarrow \quad \frac{P(S = \mathsf{low})}{0.155}$

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Example

Consider the variables

- Temp : $\mathrm{sp}(\mathsf{Temp}) = \{l, m, h\}$
- Sensor : $\mathrm{sp}(\mathsf{Sensor}) = \{l, m, h\}$

Assume we observe $S = \mathsf{low}$:
$P(T|\mathsf{low}) = \frac{P(\mathsf{low}|T)P(T)}{\sum_T P(\mathsf{low}|T)P(T)} =$

$P(\mathsf{Temp}) = (0.1, 0.6, 0.3)$

$P(\mathsf{Sensor}|\mathsf{Temp}) =$

Sensor \ Temp	l	m	h
l	0.8	0.1	0.05
m	0.15	0.8	0.1
h	0.05	0.1	0.85

$P(\mathsf{low}|T)P(T) =$

$S = \mathsf{low}$	l	m	h
	0.08 / 0.155	0.06 / 0.155	0.015 / 0.155

$P(\mathsf{low}|T)P(T) =$

$S = \mathsf{low}$	l	m	h
	0.08	0.06	0.015

$\rightarrow \quad \frac{P(S = \mathsf{low})}{0.155}$

Bayes' rule for variables

Bayes' rule can also be written for variables:

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

Example

Consider the variables

- Temp : $\mathrm{sp}(\mathsf{Temp}) = \{l, m, h\}$
- Sensor : $\mathrm{sp}(\mathsf{Sensor}) = \{l, m, h\}$

Assume we observe $S = \mathsf{low}$:
$P(T|\mathsf{low}) = \frac{P(\mathsf{low}|T)P(T)}{\sum_T P(\mathsf{low}|T)P(T)} =$

$P(\mathsf{Temp}) = (0.1, 0.6, 0.3)$

$P(\mathsf{Sensor}|\mathsf{Temp}) =$

Sensor \ Temp	l	m	h
l	0.8	0.1	0.05
m	0.15	0.8	0.1
h	0.05	0.1	0.85

$P(\mathsf{low}|T)P(T) =$

$S = \mathsf{low}$	l	m	h
	0.52	0.39	0.09

$P(\mathsf{low}|T)P(T) =$

$S = \mathsf{low}$	l	m	h
	0.08	0.06	0.015

$\rightarrow \quad \frac{P(S = \mathsf{low})}{0.155}$

Independence

Example: Football statistics

Results for Bayern Munich and SC Freiburg in seasons 2001/02 and 2003/04. (Not counting the matches Munich vs. Freiburg):

$$D_{\mathit{Munich}} = D_{\mathit{Freiburg}} = \{\mathit{Win}, \mathit{Draw}, \mathit{Loss}\}$$

2001/02

Munich: LWDWWWWWWWLDLDLDLWLDWWWDWDDWWWW
Freiburg: WLLDDWLDWDWLLLDDLWDDLLDLLLLLLWLW

2003/04

Munich: WDWWLDWWDWLWWDDWDWLWWWDDWWWLWWLL
Freiburg: LDDWDWLWLLLWWLWLWLLDWLDDWDLLLWLD

Summary:

Munich \ Freiburg	W	D	L	
W	12	9	15	36
D	3	4	9	16
L	6	4	2	12
	21	17	26	

Independence of Outcomes

The joint distribution of *Munich* and *Freiburg*:

$P(\textit{Munich},\textit{Freiburg})$:

Munich	*Freiburg* W	D	L	$P(\textit{Munich})$
W	.1875	.1406	.2344	.5625
D	.0468	.0625	.1406	.25
L	.0937	.0625	.0312	.1875
$P(\textit{Freiburg})$	.3281	.2656	.4062	

Independence of Outcomes

The joint distribution of *Munich* and *Freiburg*:

$P(\textit{Munich},\textit{Freiburg})$:

Munich	*Freiburg* W	D	L	$P(\textit{Munich})$
W	.1875 .571	.1406 .529	.2344 .577	.5625
D	.0468 .143	.0625 .235	.1406 .346	.25
L	.0937 .285	.0625 .235	.0312 .077	.1875
$P(\textit{Freiburg})$	.3281	.2656	.4062	

Conditional distribution: $P(\textit{Munich} \mid \textit{Freiburg})$

Independence of Outcomes

The joint distribution of *Munich* and *Freiburg*:

$P(\textit{Munich},\textit{Freiburg})$:

Munich	*Freiburg* W	D	L	$P(\textit{Munich})$
W	.1875 .571	.1406 .529	.2344 .577	.5625
D	.0468 .143	.0625 .235	.1406 .346	.25
L	.0937 .285	.0625 .235	.0312 .077	.1875
$P(\textit{Freiburg})$	.3281	.2656	.4062	

Conditional distribution: $P(\textit{Munich} \mid \textit{Freiburg})$

We have (almost):

$$P(\textit{Munich} \mid \textit{Freiburg}) = P(\textit{Munich})$$

The variables *Munich* and *Freiburg* are **independent**.

Independent Variables

Definition of Independence

The variables $A_1, \ldots, A_k$ and $B_1, \ldots, B_m$ are **independent** if

$$P(A_1, \ldots, A_k \mid B_1, \ldots, B_m) = P(A_1, \ldots, A_k)$$

This is equivalent to:

$$P(B_1, \ldots, B_m \mid A_1, \ldots, A_k) = P(B_1, \ldots, B_m)$$

and also to:

$$P(A_1, \ldots, A_k, B_1, \ldots, B_m) = P(A_1, \ldots, A_k) \cdot P(B_1, \ldots, B_m)$$

Compact Specifications by Independence

Independence properties can greatly simplify the specification of a joint distribution:

$M =$	$F =$ W	D	L	$P(M)$
W				.5625
D	M and F are independent			.25
L				.1875
$P(F)$	.3281	.2656	.4062	

The probability for each possible world then is defined, e.g.

$$P(M = D, F = L) = 0.25 \cdot 0.4062 = 0.10155$$

Example

Joint distribution for variables

Sex : $D_{\textit{Sex}} = \{\textit{male}, \textit{female}\}$
Hair length : $D_{\textit{Hair length}} = \{\textit{long}, \textit{short}\}$
Height : $D_{\textit{Height}} = \{\textit{tall}, \textit{medium}\}$

	Sex *male*		*female*	
	Hair length		*Hair length*	
Height	*long*	*short*	*long*	*short*
tall	0.06	0.24	0.07	0.03
medium	0.04	0.16	0.28	0.12

$P(\textit{Hair length}, \textit{Height})$ $P(\textit{Height})$, $P(\textit{Height} \mid \textit{Hair length})$:

Height	*Hair length* *long*	*short*	
tall	0.13	0.27	0.4
	0.289	0.49	
medium	0.32	0.28	0.6
	0.711	0.51	

$\rightsquigarrow$ *Hair length* and *Height* are not independent.

Example Continued

$P(\textit{Hair length}, \textit{Height} \mid \textit{Sex} = \textit{female})$, $P(\textit{Height} \mid \textit{Sex} = \textit{female})$, $P(\textit{Height} \mid \textit{Hair length}, \textit{Sex} = \textit{female})$:

	Hair length		
Height	*long*	*short*	
tall	0.14	0.06	0.2
	0.2	0.2	
medium	0.56	0.24	0.8
	0.8	0.8	

$\rightsquigarrow$ *Hair length* and *Height* are independent given *Sex=female*.

Also: *Hair length* and *Height* are independent given *Sex=male*.

$\rightsquigarrow$ *Hair length* and *Height* are independent given *Sex*.

Conditionally Independent Variables

Definition of Conditional Independence

The variables $A_1, \ldots, A_n$ are **conditionally independent** of the variables $B_1, \ldots, B_m$ **given** $C_1, \ldots, C_k$, if

$$P(A_1, \ldots, A_n \mid B_1, \ldots, B_m, C_1, \ldots, C_k) = P(A_1, \ldots, A_n \mid C_1, \ldots, C_k)$$

Agenda: use conditional independence to facillitate specification of probability distributions on complex state spaces

Machine Intelligence

Lecture 5: Bayesian networks

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- **Bayesian networks**
- Machine learning
- Planning
- Multi-agent systems

Bayesian Networks

Example

Random variables (all Boolean):

Tampering	fire alarm has been tampered with
Fire	fire in the building
Alarm	fire alarm ringing
Smoke	smoke in the building
Leaving	people leaving the building
Report	report of people leaving the building

Joint distribution according to chain rule:

$$\begin{aligned} P(&\textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}, \textit{Leaving}, \textit{Report}) = \\ &P(\textit{Tampering}) \cdot \\ &P(\textit{Fire} \mid \textit{Tampering}) \cdot \\ &P(\textit{Alarm} \mid \textit{Tampering}, \textit{Fire}) \cdot \\ &P(\textit{Smoke} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}) \cdot \\ &P(\textit{Leaving} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}) \cdot \\ &P(\textit{Report} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}, \textit{Leaving}) \end{aligned}$$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) =$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) = P(\mathit{Fire})$

$P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire}) =$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) = P(\mathit{Fire})$

$P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire}) = P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire})$

$P(\mathit{Smoke} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}) =$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) = P(\mathit{Fire})$

$P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire}) = P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire})$

$P(\mathit{Smoke} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}) = P(\mathit{Smoke} \mid \mathit{Fire})$

$P(\mathit{Leaving} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}, \mathit{Smoke}) =$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) = P(\mathit{Fire})$

$P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire}) = P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire})$

$P(\mathit{Smoke} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}) = P(\mathit{Smoke} \mid \mathit{Fire})$

$P(\mathit{Leaving} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}, \mathit{Smoke}) = P(\mathit{Leaving} \mid \mathit{Alarm})$

$P(\mathit{Report} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}, \mathit{Smoke}, \mathit{Leaving}) =$

Example continued

Conditional independence assumptions

$P(\mathit{Tampering}) = P(\mathit{Tampering})$

$P(\mathit{Fire} \mid \mathit{Tampering}) = P(\mathit{Fire})$

$P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire}) = P(\mathit{Alarm} \mid \mathit{Tampering}, \mathit{Fire})$

$P(\mathit{Smoke} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}) = P(\mathit{Smoke} \mid \mathit{Fire})$

$P(\mathit{Leaving} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}, \mathit{Smoke}) = P(\mathit{Leaving} \mid \mathit{Alarm})$

$P(\mathit{Report} \mid \mathit{Tampering}, \mathit{Fire}, \mathit{Alarm}, \mathit{Smoke}, \mathit{Leaving}) = P(\mathit{Report} \mid \mathit{Leaving})$

Example continued

Conditional independence assumptions

$P(\textit{Tampering}) = P(\textit{Tampering})$

$P(\textit{Fire} \mid \textit{Tampering}) = P(\textit{Fire})$

$P(\textit{Alarm} \mid \textit{Tampering}, \textit{Fire}) = P(\textit{Alarm} \mid \textit{Tampering}, \textit{Fire})$

$P(\textit{Smoke} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}) = P(\textit{Smoke} \mid \textit{Fire})$

$P(\textit{Leaving} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}) = P(\textit{Leaving} \mid \textit{Alarm})$

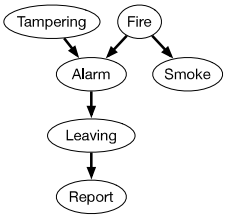
$P(\textit{Report} \mid \textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}, \textit{Leaving}) = P(\textit{Report} \mid \textit{Leaving})$

Pluggin this into chain rule give simplified representation of joint distribution:

$$P(\textit{Tampering}, \textit{Fire}, \textit{Alarm}, \textit{Smoke}, \textit{Leaving}, \textit{Report}) = P(\textit{Tampering}) \cdot P(\textit{Fire}) \cdot P(\textit{Alarm} \mid \textit{Tampering}, \textit{Fire}) \cdot P(\textit{Smoke} \mid \textit{Fire}) \cdot P(\textit{Leaving} \mid \textit{Alarm}) \cdot P(\textit{Report} \mid \textit{Leaving})$$

Graphical Representation

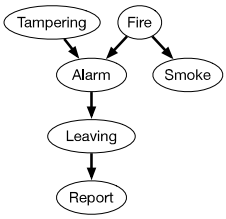
Representation of conditional dependencies in a graph:



The graph is **directed** and **acyclic**.

Graphical Representation

Representation of conditional dependencies in a graph:



The graph is **directed** and **acyclic**.

Bayesian Network

A **Bayesian Network** for variables $A_1, \ldots, A_k$ consists of

- a directed acyclic graph with nodes $A_1, \ldots, A_k$
- for each node a **conditional probability table** specifying the conditional distribution $P(A_i \mid \textit{parents}(A_i))$ ($\textit{parents}(A_i)$ denotes the **parents** of A_i in the graph)

and through the chain rule provides a compact representation of a joint probability distribution.

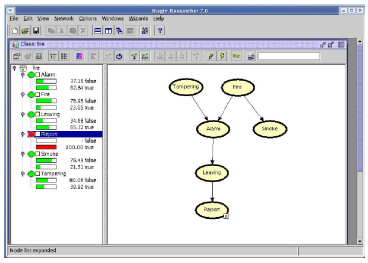
Observations and queries

A Bayesian network specifies a joint distribution from which arbitrary conditional probabilities can be derived.

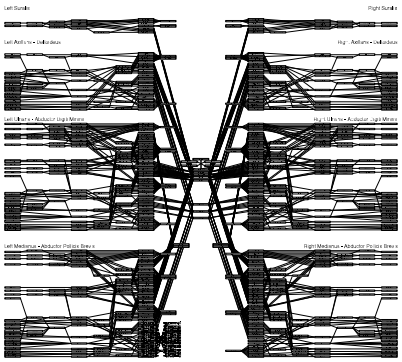
Inference

The most common task is to compute the posterior distribution over a query variable A given the observed values of some evidence nodes $E_i = e_i$, for $i = 1, \ldots, l$:

$$P(A \mid E_1 = e_1, \ldots, E_l = e_l).$$



The Munin network



Characteristics:

- Approximately 1100 variables.
- Each variable has between 2 and 20 states.
- 10^{600} possible state configurations!

A system for diagnosing neuro-muscular diseases.

Constructing a Bayesian Network

Construction via chain rule

1. put the random variables in some order
2. write the joint distribution using chain rule
3. simplify conditional probability factors by conditional independence assumptions. That determines the *parents* of each node, i.e. the graph structure
4. specify the conditional probability tables

Note: the structure of the resulting network strongly depends on the chosen order of the variables.

Construction via causality

- Draw and edge from variable A to variable B if A has a direct causal influence on A.

Note: this may not always be possible:

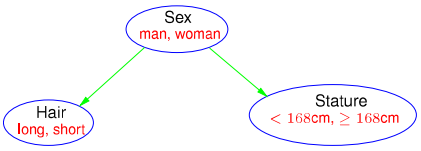
- *Inflation* $\rightarrow$ *salaries* or *salaries* $\rightarrow$ *Inflation* ?
- *Rain* doesn't cause *Sun*, and *Sun* doesn't cause *Rain*, but they are not independent either!

Transmission of evidence

Reasoning under uncertainty 1

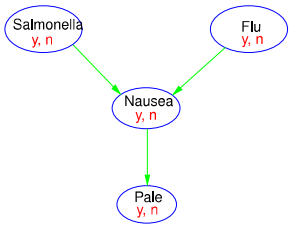
- If there has been a flooding does that tell me something about the amount of rain that has fallen?
- The water level is high: If there has been a flooding does that tell me anything new about the amount of rain that has fallen?

Reasoning under uncertainty 2



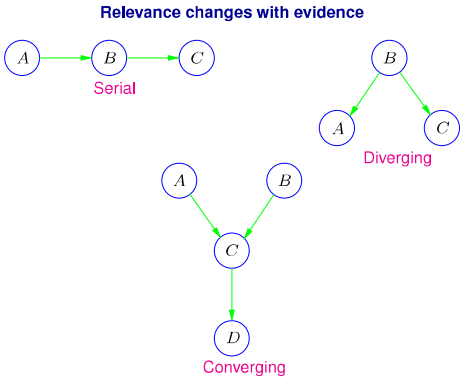
- If a person has long hair does that say something about his/her stature?
- It is a woman: If she has long hair does that say something about her stature?

Reasoning under uncertainty 3

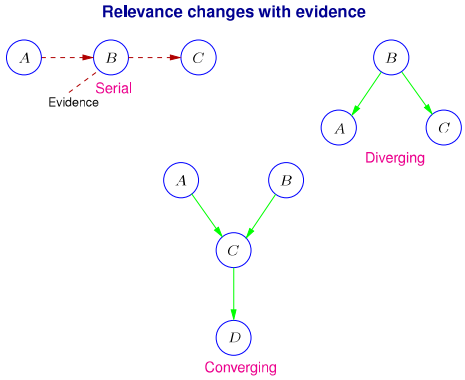


- Does salmonella have an impact on Flu?
- If a person is Pale, does salmonella then have an impact on Flu?

Transmission of evidence 1

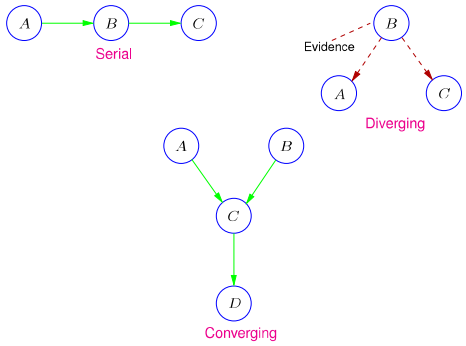


Transmission of evidence 1



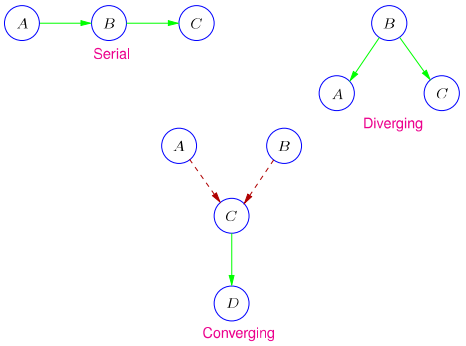
Transmission of evidence 1

Relevance changes with evidence



Transmission of evidence 1

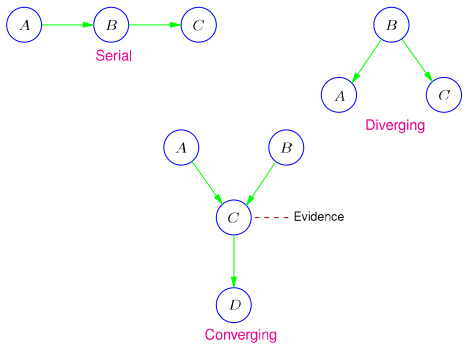
Relevance changes with evidence



MI Autumn 2019 Transmission of evidence 12

Transmission of evidence 1

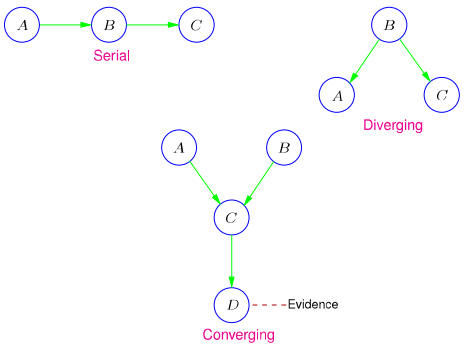
Relevance changes with evidence



MI Autumn 2019 Transmission of evidence 12

Transmission of evidence 1

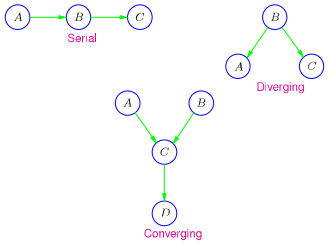
Relevance changes with evidence



MI Autumn 2019 Transmission of evidence 12

Transmission of evidence 1

Summary of transmission rules (d-separation rules)

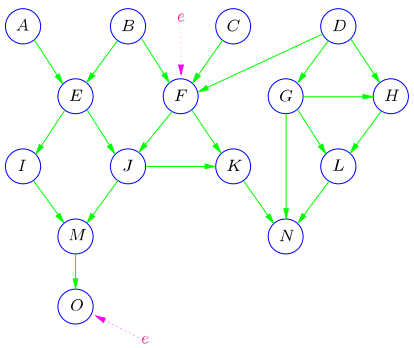


Rules for transmission of evidence

- Evidence may be transmitted through a serial or diverging connection unless it is instantiated.
- Evidence may be transmitted through a converging connection only if either the variable in the connection or one of its descendants has received evidence.

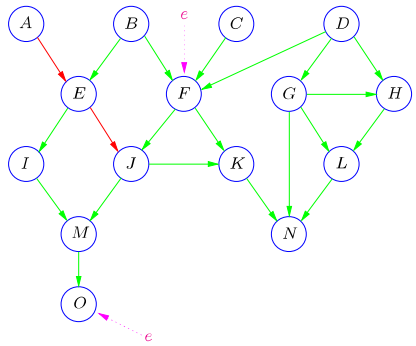
MI Autumn 2019 Transmission of evidence 12

Transmission of evidence 2



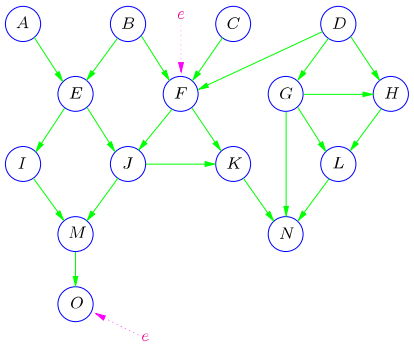
Can knowledge of A have an impact on our knowledge of J?

Transmission of evidence 2



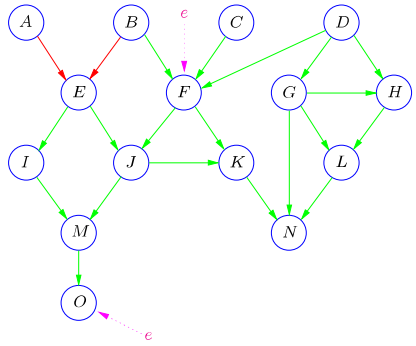
Can knowledge of A have an impact on our knowledge of J? yes!

Transmission of evidence 2



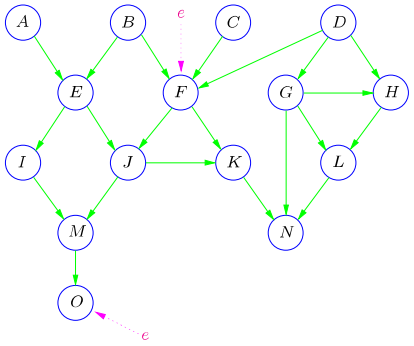
Can knowledge of A have an impact on our knowledge of B?

Transmission of evidence 2



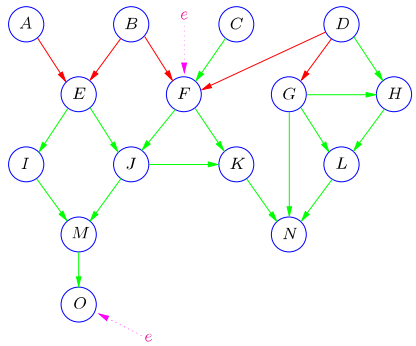
Can knowledge of A have an impact on our knowledge of B? yes!

Transmission of evidence 2



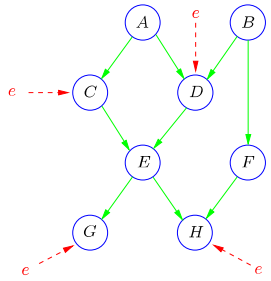
Can knowledge of A have an impact on our knowledge of G?

Transmission of evidence 2



Can knowledge of A have an impact on our knowledge of G? yes!

Transmission of evidence 3



Is E d-separated from A?

The d-separation theorem

Theorem

For all pairwise disjoint sets $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of nodes in a Bayesian network:

If $\mathbf{C}$ *d-separates* $\mathbf{A}$ *from* $\mathbf{B}$*, then* $P(\mathbf{A} \mid \mathbf{B}, \mathbf{C}) = P(\mathbf{A} \mid \mathbf{C})$.

There are no more general graphical conditions than d-separation for which such a result holds.

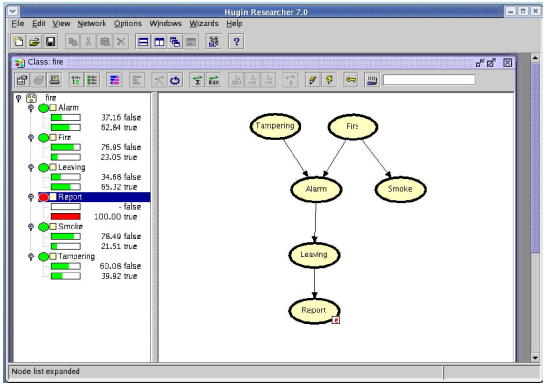
Use of d-Separation

Why is d-separation important?

- Gaining insight: given a (correct) Bayesian network model, can derive insight into the dependencies among the variables
- Debugging a model: given a Bayesian network model, check whether entailed independence relations are plausible
- Correctness of algorithms: certain computational procedures depend on validity of special independence relations

Fire Example

The Fire example in the HUGIN Bayesian Network system (`http://www.hugin.com/`)



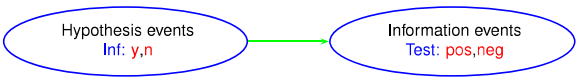
Specifying the structure of a Bayesian network

Building models

Milk from a cow may be infected. To detect whether or not the milk is infected, you can apply a test which may either give a positive or a negative test result. The test is not perfect: It may give false positives as well as false negatives.

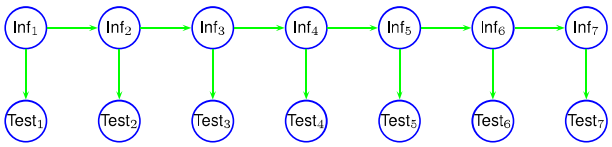
Building models

Milk from a cow may be infected. To detect whether or not the milk is infected, you can apply a test which may either give a positive or a negative test result. The test is not perfect: It may give false positives as well as false negatives.



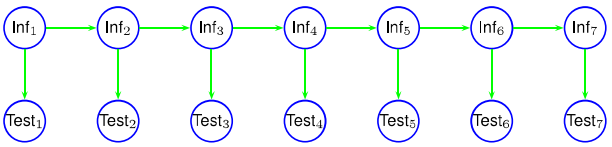
7-day model I

Infections develop over time:



7-day model I

Infections develop over time:



Assumption

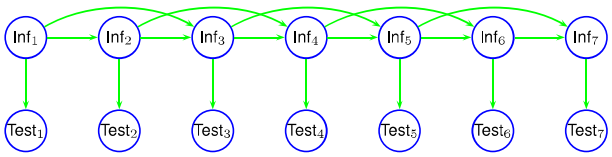
The Markov property: If I know the present, then the past has no influence on the future:

Inf_{i-1} is d-separated from Inf_{i+1} given Inf_i.

But what if yesterday's Inf-state has an impact on tomorrow's Inf-state?

7-day model II

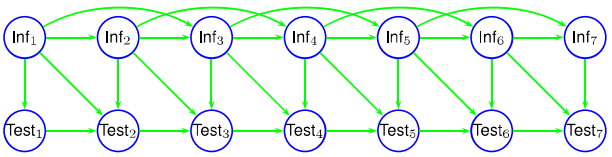
Non-Markov relations



Yesterday's Inf-state has an impact on tomorrow's Inf-state.

7-day model III

Relations between observations



The test-failure is dependent on whether or not the test failed yesterday.

Sore throat

I wake up one morning with a sore throat. It may be the beginning of a cold or I may suffer from angina. If it is a severe angina, then I will not go to work. To gain more insight, I can take my temperature and look down my throat for yellow spots.

Sore throat

I wake up one morning with a sore throat. It may be the beginning of a cold or I may suffer from angina. If it is a severe angina, then I will not go to work. To gain more insight, I can take my temperature and look down my throat for yellow spots.

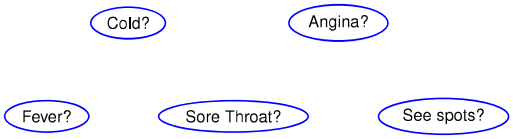
Hypothesis variables

Cold? - {n, y}
Angina? - {no, mild, severe}

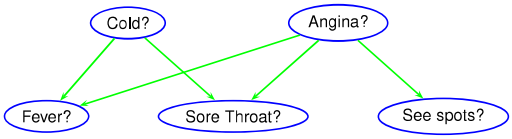
Information variables

Sore throat? - {n, y}
See spots? - {n, y}
Fever? - {no, low, high}

Model for sore throat

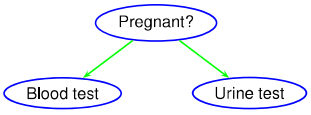


Model for sore throat



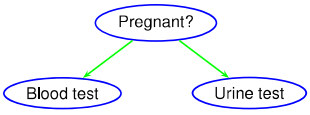
Insemination of a cow

Six weeks after the insemination of a cow, there are two tests: a Blood test and a Urine test.



Insemination of a cow

Six weeks after the insemination of a cow, there are two tests: a Blood test and a Urine test.

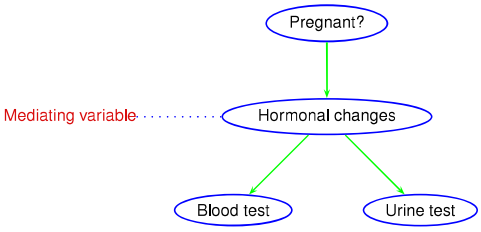


Check the conditional independences

If we know that the cow is pregnant, will a negative blood test then change our expectation for the urine test?

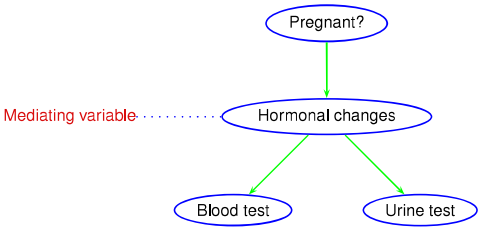
If it will, then the model does not reflect reality!

Insemination of a cow: A more correct model



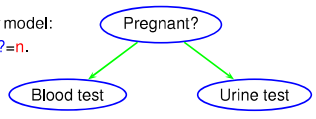
But does this actually make a difference?

Insemination of a cow: A more correct model



But does this actually make a difference?

Assume that both tests are negative in the *incorrect* model:
This will overestimate the probability for Pregnant?=n.

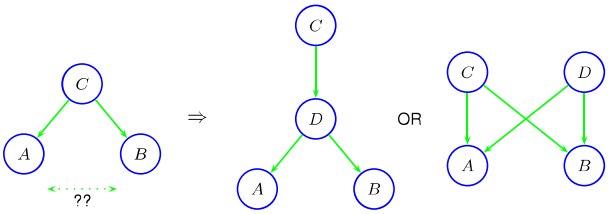


Why mediating variables?

Why do we introduce mediating variables:

- Necessary to catch the correct conditional independences.
- Can ease the specification of the probabilities in the model.

For example: If you find that there is a dependence between two variables A and B, but cannot determine a causal relation: Try with a mediating variable!



A simplified poker game

The game consists of:

- Two players.
- Three cards to each player.
- Two rounds of changing cards (max two cards in the second round)

What kind of hand does my opponent have?

A simplified poker game

The game consists of:

- Two players.
- Three cards to each player.
- Two rounds of changing cards (max two cards in the second round)

What kind of hand does my opponent have?

Hypothesis variable:

OH - {no, 1a, 2v, fl, st, 3v, sf}

Information variables:

FC - {0, 1, 2, 3} and SC - {0, 1, 2}

A simplified poker game

The game consists of:

- Two players.
- Three cards to each player.
- Two rounds of changing cards (max two cards in the second round)

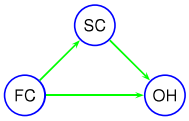
What kind of hand does my opponent have?

Hypothesis variable:

OH - {no, 1a, 2v, fl, st, 3v, sf}

Information variables:

FC - {0, 1, 2, 3} and SC - {0, 1, 2}



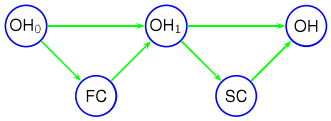
But how do we find:

$P(\text{FC})$, $P(\text{SC}|\text{FC})$ and $P(\text{OH}|\text{SC},\text{FC})$??

A simplified poker game: Mediating variables

Introduce mediating variables:

- The opponent's initial hand, OH_0.
- The opponent's hand after the first change of cards, OH_1.



Note

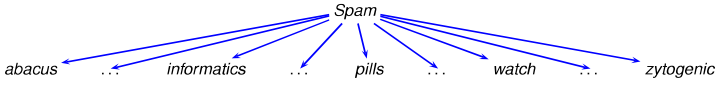
The states of OH_0 and OH_1 are different from OH.

Naive Bayes Model

Example: Spam filter

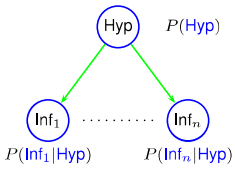
- A single *query variable*: *Spam*
- Many observable features (e.g. words appearing in the body of the message): *abacus*,. . . ,*informatics*, *pills*, . . . , *watch*,. . . , *zytogenic*

Network Structure:



- Inference with large number of variables possible
- Essentially how *Thunderbird* spam filter works

Naïve Bayes models

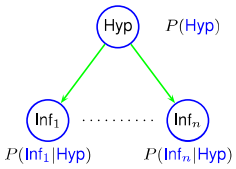


We want the posterior probability of the hypothesis variable Hyp given the observations $\{\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n\}$:

$$P(\text{Hyp}|\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n) = \frac{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})}{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n)}$$

Note: The model assumes that the information variables are independent given the hypothesis variable.

Naïve Bayes models



We want the posterior probability of the hypothesis variable Hyp given the observations $\{\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n\}$:

$$P(\text{Hyp}|\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n) = \frac{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})}{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n)}$$
$$= \mu \cdot P(\text{Inf}_1 = e_1|\text{Hyp}) \cdot \ldots \cdot P(\text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})$$

Note: The model assumes that the information variables are independent given the hypothesis variable.

Machine Intelligence

Lecture 6: Inference in Bayesian networks

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- **Inference in Bayesian networks**
- Machine learning
- Planning
- Multi-agent systems

Exact Inference

Inference Problem

Posterior Marginals

Inference Problem:

- Given: a Bayesian network
- Given: an assignment of values to some of the variables in the network: $E_i = e_i$ $(i = 1, \ldots, l)$
  - "Instantiation of the nodes $\mathbf{E}$"
  - "Evidence $\mathbf{E} = \mathbf{e}$ entered")
  - "Findings entered"
  - ...
- Want: for variables $A \notin \mathbf{E}$ the *posterior marginal* $P(A \mid \mathbf{E} = \mathbf{e})$.

Problem Reduction

According to the definition of conditional probability:

$$P(A \mid \mathbf{E} = \mathbf{e}) = \frac{P(A, \mathbf{E} = \mathbf{e})}{P(\mathbf{E} = \mathbf{e})}$$

It is sufficient to compute for each $a \in D_A$ the value

$$P(A = a, \mathbf{E} = \mathbf{e}).$$

Together with

$$P(\mathbf{E} = \mathbf{e}) = \sum_{a \in D_A} P(A = a, \mathbf{E} = \mathbf{e})$$

this gives the desired posterior distribution.

Inference: Basic Principles

Inference as summation

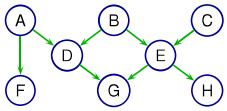
Let A be the variable of interest, $\mathbf{E}$ the evidence variables, and $\mathbf{Y} = Y_1, \ldots, Y_l$ the remaining variables in the network not belonging to $A \cup \mathbf{E}$. Then

$$P(A = a, \mathbf{E} = \mathbf{e}) = \sum_{y_1 \in D_{Y_1}} \ldots \sum_{y_l \in D_{Y_l}} P(A = a, \mathbf{E} = \mathbf{e}, Y_1 = y_1, \ldots, Y_l = y_l).$$

Note:

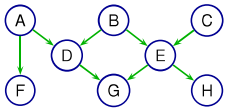
- For each $\mathbf{y}$ the probability $P(A = a, \mathbf{E} = \mathbf{e}, \mathbf{Y} = \mathbf{y})$ can be computed from the network (in time linear in the number of random variables).
- There number of configurations over $\mathbf{Y}$ is exponential in l.

Inference Problems I



Find $P(B|a, f, g, h) = \frac{P(B,a,f,g,h)}{P(a,f,g,h)}$

Inference Problems I

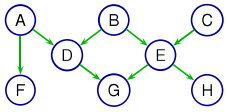


Find $P(B|a, f, g, h) = \frac{P(B,a,f,g,h)}{P(a,f,g,h)}$

We can if we have access to $P(A, B, C, D, E, F, G, H)$:

$$P(A, B, C, D, E, F, G, H) = P(A)P(B)P(C)P(D|A, B) \cdot \ldots \cdot P(H|E)$$

Inference Problems I



Find $P(B|a, f, g, h) = \frac{P(B,a,f,g,h)}{P(a,f,g,h)}$

We can if we have access to $P(A, B, C, D, E, F, G, H)$:

$$P(A, B, C, D, E, F, G, H) = P(A)P(B)P(C)P(D|A, B) \cdot \ldots \cdot P(H|E)$$

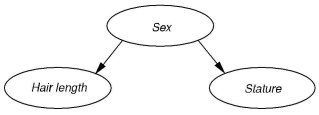
Inserting evidence we get:

$$P(B, a, f, g, h) = \sum_{C,D,E} P(a, B, C, D, E, f, g, h)$$

and

$$P(a, f, g, h) = \sum_{B} P(B, a, f, g, h)$$

Inference Problems II



Conditional probability tables:

Sex	
male	0.49
female	0.51

	Sex	
Hair length	*male*	*female*
long	0.05	0.6
short	0.95	0.4

	Sex	
Stature	*male*	*female*
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Posterior probability inference: Given the value of some observed variables (the evidence) compute the conditional distribution of some other variable:

$P(\textit{Stature} \mid \textit{Hair length} = \textit{long}) = ?$
$P(\textit{Sex} \mid \textit{Hair length} = \textit{short}, \textit{Stature} \leq 1.68) = ?$

Naive Solution Step 1

$P(Sex)$

Sex	
male	0.49
female	0.51

$P(Hair\ length \mid Sex)$

	Sex	
Hair length	male	female
long	0.05	0.6
short	0.95	0.4

$P(Stature \mid Sex)$

	Sex	
Stature	male	female
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Query: $P(Stature \mid Hair\ length = long) = ?$

Naive Solution Step 1

$P(Sex)$

Sex	
male	0.49
female	0.51

$P(Hair\ length \mid Sex)$

	Sex	
Hair length	male	female
long	0.05	0.6
short	0.95	0.4

$P(Stature \mid Sex)$

	Sex	
Stature	male	female
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Query: $P(Stature \mid Hair\ length = long) = ?$

Step 1: Construct joint distribution

$P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68				
> 1.68				

Naive Solution Step 1

$P(Sex)$

Sex	
male	0.49
female	0.51

$P(Hair\ length \mid Sex)$

	Sex	
Hair length	male	female
long	0.05	0.6
short	0.95	0.4

$P(Stature \mid Sex)$

	Sex	
Stature	male	female
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Query: $P(Stature \mid Hair\ length = long) = ?$

Step 1: Construct joint distribution

$P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68	0.00196			
> 1.68				

Naive Solution Step 1

$P(Sex)$

Sex	
male	0.49
female	0.51

$P(Hair\ length \mid Sex)$

	Sex	
Hair length	male	female
long	0.05	0.6
short	0.95	0.4

$P(Stature \mid Sex)$

	Sex	
Stature	male	female
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Query: $P(Stature \mid Hair\ length = long) = ?$

Step 1: Construct joint distribution

$P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68	0.00196	0.03724		
> 1.68				

Naive Solution Step 1

$P(Sex)$

Sex	
male	0.49
female	0.51

$P(Hair\ length \mid Sex)$

	Sex	
Hair length	male	female
long	0.05	0.6
short	0.95	0.4

$P(Stature \mid Sex)$

	Sex	
Stature	male	female
≤ 1.68	0.08	0.47
> 1.68	0.92	0.53

Query: $P(Stature \mid Hair\ length = long) = ?$

Step 1: Construct joint distribution

$P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68	0.00196	0.03724	0.14382	0.09588
> 1.68	0.02254	0.42826	0.16218	0.10812

Naive Solution Step 2

Joint distribution $P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68	0.00196	0.03724	0.14382	0.09588
> 1.68	0.02254	0.42826	0.16218	0.10812

Step 2 "Enter evidence" :

$P(Sex, Hair\ length, Stature)$

	Sex			
	male		female	
	Hair length		Hair length	
Stature	long	short	long	short
≤ 1.68	0.00196	0.03724	0.14382	0.09588
> 1.68	0.02254	0.42826	0.16218	0.10812

Hair length = long →

$P(Sex, Hair\ length{=}long, Stature)$

	Sex	
Stature	male	female
≤ 1.68	0.00196	0.14382
> 1.68	0.02254	0.16218

Note: the table on the right shows neither a joint nor a conditional distribution!

Naive Solution Steps 3 + 4

Step 3 Marginalize (sum out *Sex* variable):

$P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

	Sex	
Stature	male	female
≤ 1.68	0.00196	0.14382
> 1.68	0.02254	0.16218

$\xrightarrow{\sum_{\textit{Sex}}}$

$P(\textit{Hair length=long}, \textit{Stature})$

Stature	
≤ 1.68	0.14578
> 1.68	0.18472

Naive Solution Steps 3 + 4

Step 3 Marginalize (sum out *Sex* variable):

$P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

	Sex	
Stature	male	female
≤ 1.68	0.00196	0.14382
> 1.68	0.02254	0.16218

$\xrightarrow{\sum_{\textit{Sex}}}$

$P(\textit{Hair length=long}, \textit{Stature})$

Stature	
≤ 1.68	0.14578
> 1.68	0.18472

Step 4 Normalize

$P(\textit{Hair length=long}, \textit{Stature})$

Stature	
≤ 1.68	0.14578
> 1.68	0.18472

$\xrightarrow{\frac{1}{0.14578+0.18472}}$

$P(\textit{Stature} \mid \textit{Hair length=long})$

Stature	
≤ 1.68	0.441
> 1.68	0.559

Naive Solution: Summary

Construct Joint: $P(\textit{Sex}, \textit{Hair length}, \textit{Stature}) = P(\textit{Sex})P(\textit{Hair length} \mid \textit{Sex})P(\textit{Stature} \mid \textit{Sex})$

Naive Solution: Summary

Construct Joint: $P(\textit{Sex}, \textit{Hair length}, \textit{Stature}) = P(\textit{Sex})P(\textit{Hair length} \mid \textit{Sex})P(\textit{Stature} \mid \textit{Sex})$

Insert Evidence: $P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

Naive Solution: Summary

Construct Joint: $P(\textit{Sex}, \textit{Hair length}, \textit{Stature}) = P(\textit{Sex})P(\textit{Hair length} \mid \textit{Sex})P(\textit{Stature} \mid \textit{Sex})$

Insert Evidence: $P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

Marginalize: $P(\textit{Hair length=long}, \textit{Stature})=$

$$\sum_{s \in \{\textit{male},\textit{female}\}} P(\textit{Sex=s}, \textit{Hair length=long}, \textit{Stature})$$

Naive Solution: Summary

Construct Joint: $P(\textit{Sex}, \textit{Hair length}, \textit{Stature}) = P(\textit{Sex})P(\textit{Hair length} \mid \textit{Sex})P(\textit{Stature} \mid \textit{Sex})$

Insert Evidence: $P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

Marginalize: $P(\textit{Hair length=long}, \textit{Stature})=$

$$\sum_{s \in \{\textit{male},\textit{female}\}} P(\textit{Sex=s}, \textit{Hair length=long}, \textit{Stature})$$

Condition: $P(\textit{Stature} \mid \textit{Hair length=long}) =$

$$\frac{P(\textit{Hair length=long}, \textit{Stature})}{P(\textit{Hair length=long}, \textit{Stature} \leq 1.68) + P(\textit{Hair length=long}, \textit{Stature} > 1.68)}$$

Naive Solution: Summary

Construct Joint: $P(\textit{Sex}, \textit{Hair length}, \textit{Stature}) = P(\textit{Sex})P(\textit{Hair length} \mid \textit{Sex})P(\textit{Stature} \mid \textit{Sex})$

Insert Evidence: $P(\textit{Sex}, \textit{Hair length=long}, \textit{Stature})$

Marginalize: $P(\textit{Hair length=long}, \textit{Stature})=$

$$\sum_{s \in \{\textit{male},\textit{female}\}} P(\textit{Sex=s}, \textit{Hair length=long}, \textit{Stature})$$

Condition: $P(\textit{Stature} \mid \textit{Hair length=long}) =$

$$\frac{P(\textit{Hair length=long}, \textit{Stature})}{P(\textit{Hair length=long}, \textit{Stature} \leq 1.68) + P(\textit{Hair length=long}, \textit{Stature} > 1.68)}$$

Complexity

Complexity dominated by initial table $P(\textit{Sex}, \textit{Hair length}, \textit{Stature})$ (size 2^3).

For model with n binary variables:

$$O(2^n)$$

Variable Elimination

Problem

The joint probability distribution will contain exponentially many entries.

Idea

We can use

- the form of the joint distribution P
- the law of distributivity

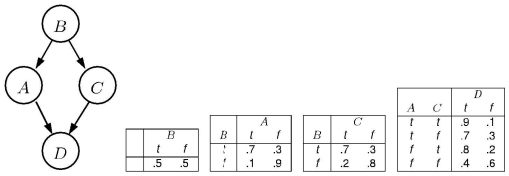
to make the computation of the sum more efficient.

Variable Elimination

Thus, we can adapt our elimination procedure so that:

- we marginalize out variables sequentially
- when marginalizing out a particular variable X, we only need to consider the factors involving X.

Example



B	
t	f
.5	.5

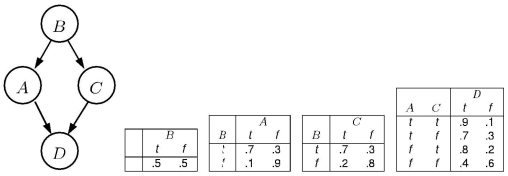
B	A	
	t	f
t	.7	.3
f	.1	.9

B	C	
	t	f
t	.7	.3
f	.2	.8

A	C	D	
		t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$P(A, D = f) =$$

Example



B	
t	f
.5	.5

B	A	
	t	f
t	.7	.3
f	.1	.9

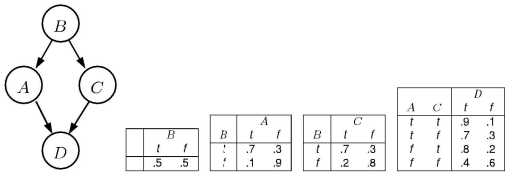
B	C	
	t	f
t	.7	.3
f	.2	.8

A	C	D	
		t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$P(A, D = f) =$$

$$\sum_{b \in \{t,f\}} \sum_{c \in \{t,f\}} P(B = b, A, C = c, D = f) =$$

Example



B	
t	f
.5	.5

B	A	
	t	f
t	.7	.3
f	.1	.9

B	C	
	t	f
t	.7	.3
f	.2	.8

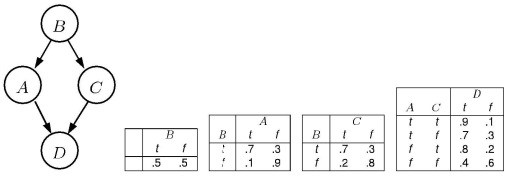
A	C	D	
		t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$P(A, D = f) =$$

$$\sum_{b \in \{t,f\}} \sum_{c \in \{t,f\}} P(B = b, A, C = c, D = f) =$$

$$\sum_{b \in \{t,f\}} \sum_{c \in \{t,f\}} P(B = b)P(A \mid B = b)P(C = c \mid B = b)P(D = f \mid A, C = c) =$$

Example



B	
t	f
.5	.5

B	A	
	t	f
t	.7	.3
f	.1	.9

B	C	
	t	f
t	.7	.3
f	.2	.8

A	C	D	
		t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

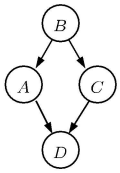
$$P(A, D = f) =$$

$$\sum_{b \in \{t,f\}} \sum_{c \in \{t,f\}} P(B = b, A, C = c, D = f) =$$

$$\sum_{b \in \{t,f\}} \sum_{c \in \{t,f\}} P(B = b)P(A \mid B = b)P(C = c \mid B = b)P(D = f \mid A, C = c) =$$

$$\sum_{b \in \{t,f\}} P(B = b)P(A \mid B = b) \sum_{c \in \{t,f\}} P(C = c \mid B = b)P(D = f \mid A, C = c)$$

Example continued



B	t	f
	.5	.5

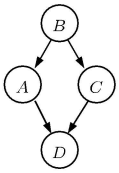
B	A	t	f
t	t	.7	.3
t	f	.1	.9

B	C	t	f
t	t	.7	.3
t	f	.2	.8

A	C	D t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$\sum_b P(B=b)P(A \mid B=b)\sum_c P(C=c \mid B=b)P(D=f \mid A, C=c) =$$
$$\sum_b P(B=b)P(A \mid B=b)F_1(B=b, A) = F_2(A)$$

Example continued



B	t	f
	.5	.5

B	A	t	f
t	t	.7	.3
t	f	.1	.9

B	C	t	f
t	t	.7	.3
t	f	.2	.8

A	C	D t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$$\sum_b P(B=b)P(A \mid B=b)\sum_c P(C=c \mid B=b)P(D=f \mid A, C=c) =$$
$$\sum_b P(B=b)P(A \mid B=b)F_1(B=b, A) = F_2(A)$$

where

B	C t	f
t	.7	.3
f	.2	.8

$\mapsto$

A	C	D t	f
t	t	.9	.1
t	f	.7	.3
f	t	.8	.2
f	f	.4	.6

$\mapsto$

b	a	$F_1(B, A)$
t	t	.7·.1 + .3·.3 = .16
t	f	.7·.2 + .3·.6 = .32
f	t	.2·.1 + .8·.3 = .26
f	f	.2·.2 + .8·.6 = .52

and

B	t	f
	.5	.5

B	A	t	f
t	t	.7	.3
t	f	.1	.9

$\mapsto$

b	a	$F_1(B, A)$
t	t	.16
⋮	⋮	⋮

$\mapsto$

a	$F_2(A)$
t	...
f	...

Factors

Calculus of factors

- The procedure operates on **factors**: functions of subsets of variables
- Required operations on factors:
  - *multiplication*
  - *marginalization* (summing out selected variables)
  - *restriction* (setting selected variables to specific values)

Complexity

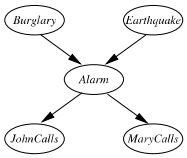
Call subsets $\mathbf{U}$ of $\mathbf{V}$ that are the arguments of factors $P(\ldots \mid \ldots)$ resp. $F_j(\ldots)$ which appear in the elimination process *factor sets*.

The complexity of variable elimination is exponential in the size of the largest factor set.

The size of the largest factor set can depend strongly on the order in which variables are summed out!

Alarm Example

Example

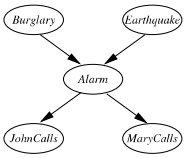


Bad ordering for computing $P(MC, B = t)$:

$$\sum_{eq \in \{t,f\}} \sum_{jc \in \{t,f\}} \sum_{a \in \{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

Alarm Example

Example

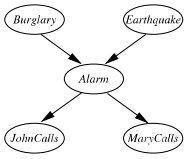


Bad ordering for computing $P(MC, B = t)$:

$$\sum_{eq \in \{t,f\}} \sum_{jc \in \{t,f\}} \sum_{a \in \{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

Alarm Example

Example

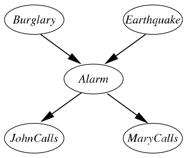


Bad ordering for computing $P(MC, B = t)$:

$$\sum_{eq \in \{t,f\}} \sum_{jc \in \{t,f\}} \sum_{a \in \{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$
$$\sum_{eq \in \{t,f\}} \sum_{jc \in \{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

Alarm Example

Example



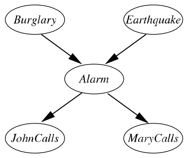
Bad ordering for computing $P(MC, B=t)$:

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}}\sum_{a\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

Alarm Example

Example



Bad ordering for computing $P(MC, B=t)$:

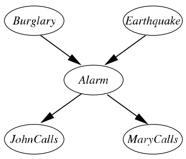
$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}}\sum_{a\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

$$\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)F_2(eq, MC) =$$

Alarm Example

Example



Bad ordering for computing $P(MC, B=t)$:

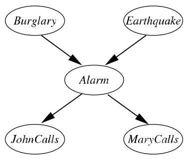
$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}}\sum_{a\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

$$\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)F_2(eq, MC) =$$

Alarm Example

Example



Bad ordering for computing $P(MC, B=t)$:

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}}\sum_{a\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

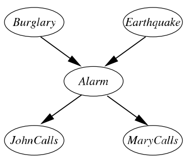
$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

$$\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)F_2(eq, MC) =$$

$$P(B=t)F_3(MC)$$

Alarm Example

Example



Bad ordering for computing $P(MC, B=t)$:

$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}}\sum_{a\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

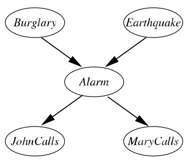
$$\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)F_1(eq, jc, MC) =$$

$$\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)F_2(eq, MC) =$$

$$P(B=t)F_3(MC)$$

Largest factor (F_1) is function of 3 variables!

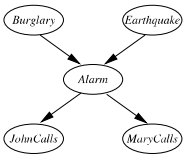
Alarm Example continued



Good ordering for computing $P(MC, B=t)$:

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

Alarm Example continued

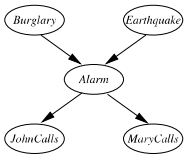


Good ordering for computing $P(MC, B = t)$:

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(MC \mid A=a)F_1(a) =$$

Alarm Example continued



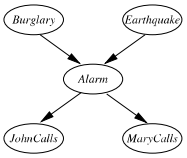
Good ordering for computing $P(MC, B = t)$:

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(MC \mid A=a)F_1(a) =$$

$$\sum_{a\in\{t,f\}} P(B=t)P(MC \mid A=a)F_1(a)F_2(a) =$$

Alarm Example continued



Good ordering for computing $P(MC, B = t)$:

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}}\sum_{jc\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(JC=jc \mid A=a)P(MC \mid A=a) =$$

$$\sum_{a\in\{t,f\}}\sum_{eq\in\{t,f\}} P(B=t)P(EQ=eq)P(A=a \mid B=t, EQ=eq)P(MC \mid A=a)F_1(a) =$$

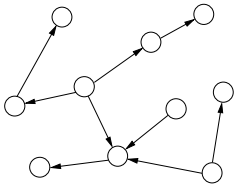
$$\sum_{a\in\{t,f\}} P(B=t)P(MC \mid A=a)F_1(a)F_2(a) =$$

$$P(B=t)F_3(MC)$$

Largest factor ($P(A \mid B = t, EQ)$) is function of 2 variables!

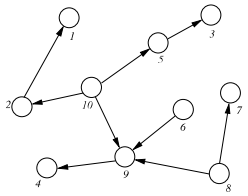
Singly connected networks

A **singly connected network** is a network in which any two nodes are connected by at most one path of undirected edges:



Singly connected networks

A **singly connected network** is a network in which any two nodes are connected by at most one path of undirected edges:



For singly connected network: any elimination order that "peels" variables from outside will only create factors of only one variable.

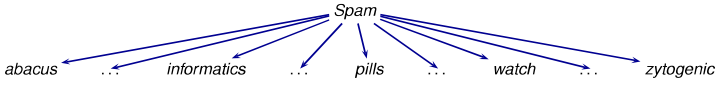
The complexity of inference is therefore linear in the total size of the network (= combined size of all conditional probability tables).

Naive Bayes Model

Example: Spam filter

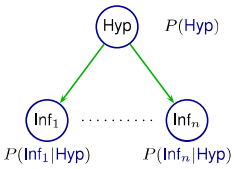
- A single *query variable*: *Spam*
- Many observable features (e.g. words appearing in the body of the message): *abacus*,. . . ,*informatics*, *pills*, . . . , *watch*,. . . , *zytogenic*

Network Structure:



- Inference with large number of variables possible
- Essentially how *Thunderbird* spam filter works

Naïve Bayes models

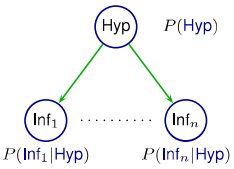


We want the posterior probability of the hypothesis variable Hyp given the observations $\{\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n\}$:

$$P(\text{Hyp}|\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n) = \frac{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})}{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n)}$$

Note: The model assumes that the information variables are independent given the hypothesis variable.

Naïve Bayes models



We want the posterior probability of the hypothesis variable Hyp given the observations $\{\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n\}$:

$$P(\text{Hyp}|\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n) = \frac{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})}{P(\text{Inf}_1 = e_1, \ldots, \text{Inf}_n = e_n)}$$
$$= \mu \cdot P(\text{Inf}_1 = e_1|\text{Hyp}) \cdot \ldots \cdot P(\text{Inf}_n = e_n|\text{Hyp})P(\text{Hyp})$$

Note: The model assumes that the information variables are independent given the hypothesis variable.

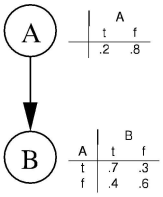
Approximate Inference

Sampling Inference

Sample Generator

Observation: can use Bayesian network as random generator that produces states $\mathbf{X} = \mathbf{x}$ according to distribution P defined by the network.

Example:



- Generate random numbers r_1, r_2 uniformly from [0,1].
- Set $A = t$ if $r_1 \leq .2$ and $A = f$ else.
- Depending on the value of A and r_2 set B to t or f.

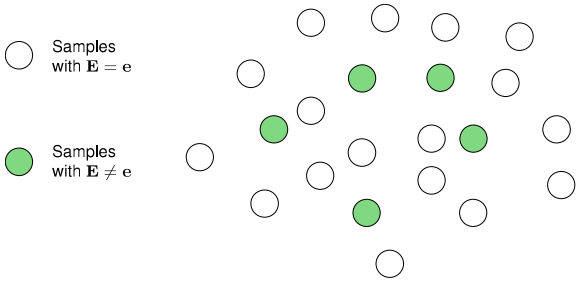
Random generation of one state: linear in size of network.

Sampling Inference

Approximate Inference from Samples

To compute an approximation of $P(\mathbf{E} = \mathbf{e})$ ($\mathbf{E}$ a subset of the variables in the Bayesian network):

- generate a (large) number of random states
- count the frequency of states in which $\mathbf{E} = \mathbf{e}$.



Accuracy

Hoeffding Bound

- p: true probability $P(\mathbf{E} = \mathbf{e})$
- s: estimate for p from sample of size n
- ϵ: an error bound > 0.

Then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

Accuracy

Hoeffding Bound

- p: true probability $P(\mathbf{E} = \mathbf{e})$
- s: estimate for p from sample of size n
- ϵ: an error bound > 0.

Then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

Required Sample Size

To obtain an estimate that *with probability at most* δ has an *accuracy at least* ϵ, it is sufficient to take

$$n = -ln(\delta/2)/(2\epsilon^2) \text{ samples.}$$

Accuracy

Hoeffding Bound

- p: true probability $P(\mathbf{E} = \mathbf{e})$
- s: estimate for p from sample of size n
- ϵ: an error bound > 0.

Then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

Required Sample Size

To obtain an estimate that *with probability at most* δ has an *accuracy at least* ϵ, it is sufficient to take

$$n = -ln(\delta/2)/(2\epsilon^2) \text{ samples.}$$

Example

To get an error ϵ of less than 0.1 in 95% of the cases ($\delta = 0.05$), we need:

$$n > -ln(0.05/2)/(2 \cdot 0.1^2) \approx 184 \text{ samples}$$

Accuracy

Hoeffding Bound

- p: true probability $P(\mathbf{E} = \mathbf{e})$
- s: estimate for p from sample of size n
- ϵ: an error bound > 0.

Then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

Required Sample Size

To obtain an estimate that *with probability at most* δ has an *accuracy at least* ϵ, it is sufficient to take

$$n = -ln(\delta/2)/(2\epsilon^2) \text{ samples.}$$

Example

To get an error ϵ of less than 0.1 in 95% of the cases ($\delta = 0.05$), we need:

$$n > -ln(0.05/2)/(2 \cdot 0.1^2) \approx 184 \text{ samples}$$

How many samples do we need if the error should be less than 0.01?

Accuracy

Hoeffding Bound

- p: true probability $P(\mathbf{E} = \mathbf{e})$
- s: estimate for p from sample of size n
- ϵ: an error bound > 0.

Then

$$P(|s - p| > \epsilon) \leq 2e^{-2n\epsilon^2}$$

Required Sample Size

To obtain an estimate that *with probability at most* δ has an *accuracy at least* ϵ, it is sufficient to take

$$n = -ln(\delta/2)/(2\epsilon^2) \text{ samples.}$$

Example

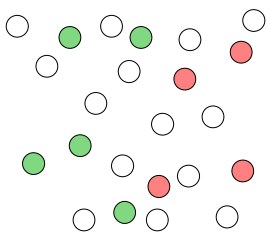
To get an error ϵ of less than 0.1 in 95% of the cases ($\delta = 0.05$), we need:

$$n > -ln(0.05/2)/(2 \cdot 0.1^2) \approx 184 \text{ samples}$$

How many samples do we need if the error should be less than 0.01? 18444 samples

Rejection Sampling

The simplest approach: **Rejection Sampling**



Sample with

not $\mathbf{E} = \mathbf{e}$

$\mathbf{E} = \mathbf{e}, A \neq a$

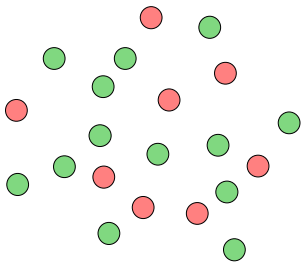
$\mathbf{E} = \mathbf{e}, A = a$

Approximation for $P(A = a \mid \mathbf{E} = \mathbf{e})$: $\frac{\#\ \bigcirc}{\#\ \bigcirc \cup \bigcirc}$

Sampling from the conditional distribution

Problem with rejection sampling: samples with $\mathbf{E} \neq \mathbf{e}$ are useless!

Ideally: would draw samples directly from the conditional distribution $P(\mathbf{A} \mid \mathbf{E} = \mathbf{e})$.



Likelihood weighting I

First idea (*not* to be followed)

- Fix evidence variables to their observed states.
- Sample from non-evidence variables.

Likelihood weighting I

First idea (*not* to be followed)

- Fix evidence variables to their observed states.
- Sample from non-evidence variables.

Problem: This gives a sampling distribution

$$\prod_{X \in \mathbf{X} \setminus \mathbf{E}} P(X \mid \mathrm{pa}(X) \setminus \mathbf{E}, \mathrm{pa}(X) \cap \mathbf{E})$$

somewhere between $P(\mathbf{X})$ and $P(\mathbf{X} \mid \mathbf{e})$.

Likelihood weighting I

First idea (*not* to be followed)

- Fix evidence variables to their observed states.
- Sample from non-evidence variables.

Problem: This gives a sampling distribution

$$\prod_{X \in \mathbf{X} \setminus \mathbf{E}} P(X \mid \mathrm{pa}(X) \setminus \mathbf{E}, \mathrm{pa}(X) \cap \mathbf{E})$$

somewhere between $P(\mathbf{X})$ and $P(\mathbf{X} \mid \mathbf{e})$.

Likelihood weighting

We would like to sample from

$$P(\mathbf{X}, \mathbf{e}) = \underbrace{\prod_{X \in \mathbf{X} \setminus \mathbf{E}} P(X \mid \mathrm{pa}(X) \setminus \mathbf{E}, \mathrm{pa}(X) \cap \mathbf{E})}_{\text{Part 1}} \cdot \underbrace{\prod_{E \in \mathbf{E}} P(E = e \mid \mathrm{pa}(E) \setminus \mathbf{E}, \mathrm{pa}(E) \cap \mathbf{E})}_{\text{Part 2}}$$

So instead weigh each generated sample with a weight corresponding to Part 2.

Likelihood weighting II

Likelihood weighting

Estimate $P(X = e \mid \mathbf{e})$ as

$$\hat{P}(X = e \mid \mathbf{e}) = \frac{\sum_{\textit{sample}: X = x} w(\textit{sample})}{\sum_{\textit{sample}} w(\textit{sample})},$$

where

$$w(\textit{sample}) = \prod_{E \in \mathbf{E}} P(E = e \mid \mathrm{pa}(E) = \pi) \qquad \text{(Part 2 on the previous slide)}$$

and π is the values of $\mathrm{pa}(E)$ under *sample* and $\mathbf{e}$.

Likelihood weighting II

Likelihood weighting

Estimate $P(X = e \mid \mathbf{e})$ as

$$\hat{P}(X = e \mid \mathbf{e}) = \frac{\sum_{\textit{sample}: X = x} w(\textit{sample})}{\sum_{\textit{sample}} w(\textit{sample})},$$

where

$$w(\textit{sample}) = \prod_{E \in \mathbf{E}} P(E = e \mid \mathrm{pa}(E) = \pi) \qquad \text{(Part 2 on the previous slide)}$$

and π is the values of $\mathrm{pa}(E)$ under *sample* and $\mathbf{e}$.

Importance sampling

Likelihood weighting is an instance of importance sampling, where

- samples are weighted and can come from (almost) any proposal distribution.

Likelihood weighting II

Likelihood weighting

Estimate $P(X = e \mid \mathbf{e})$ as

$$\hat{P}(X = e \mid \mathbf{e}) = \frac{\sum_{\textit{sample}: X = x} w(\textit{sample})}{\sum_{\textit{sample}} w(\textit{sample})},$$

where

$$w(\textit{sample}) = \prod_{E \in \mathbf{E}} P(E = e \mid \mathrm{pa}(E) = \pi) \qquad \text{(Part 2 on the previous slide)}$$

and π is the values of $\mathrm{pa}(E)$ under *sample* and $\mathbf{e}$.

Machine Intelligence

Lecture 7: Learning - Introduction and decision trees

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- **Machine learning**
- Planning
- Reinforcement learning
- Multi-agent systems

Supervised Learning

So far

The general pattern so far:

Problem/Domain description	State Space Problem	Variables, Constraints	Probabilistic Model
Inference Algorithms	Search $(A^*, \ldots)$	Arc Consistency, Variable Elimination	Variable Elimination
Solutions	Goal states, plans, diagnoses, predictions,...		

- Problem/Domain description and algorithms designed by human
- Agent will always act the same in the same situation

So far

The general pattern so far:

Problem/Domain description	State Space Problem	Variables, Constraints	Probabilistic Model
Inference Algorithms	Search $(A^*, \ldots)$	Arc Consistency, Variable Elimination	Variable Elimination
Solutions	Goal states, plans, diagnoses, predictions,...		

- Problem/Domain description and algorithms designed by human
- Agent will always act the same in the same situation

Objective of (machine) **learning**:

- Agent can *learn by experience*: improve performance over time
- Agent (program) can be automatically constructed from *examples* (rather than designed by expert)

Tasks and Model Types

The models constructed by machine learning algorithms are used for several kinds of tasks:

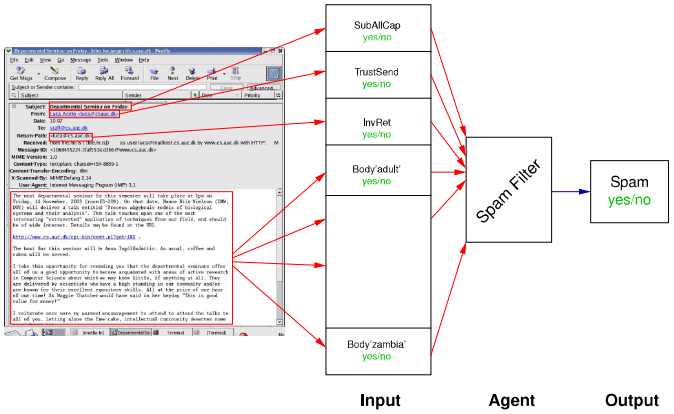
Predictive tasks/models

- Task: predict some (unobserved) target or class variable based on observed values of (predictor) attributes
  - Regression, if target is continuous
  - Classification, if target is discrete
- Examples: Spam filtering, Character recognition, ...
- Model types e.g.: Decision trees, k nearest neighbors, Neural networks, Naive Bayes,...

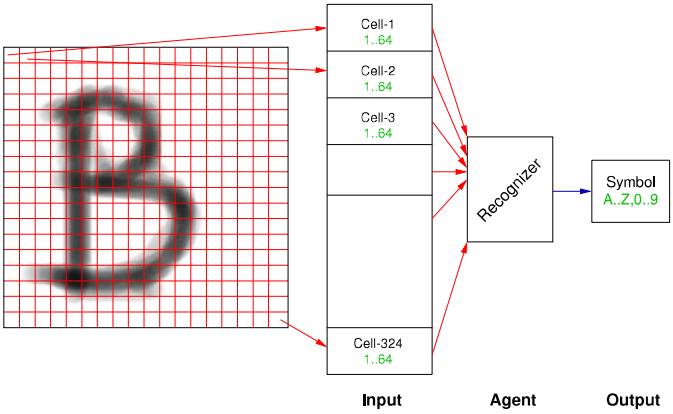
Descriptive tasks/models

- Task: Clustering: identify coherent subgroups in data
- Examples: Recommender systems,
- Model types e.g.: k-means, hierarchical clustering, Self-organizing maps, probabilistic clustering,...

Example: Spam filter



Example: Character Recognition



Experience, Data

In order to show a certain behavior, or perform certain tasks, an agent needs to

- make (the right) decisions based on possible observations

Experience from which the right behavior can be learned consists of

- **Examples** or **Cases** of inputs that are **labeled** with the correct decisions (outputs).

This is also called **Labeled data**.

Data

We assume that data (experience) consists of an attribute-value table:

Input Features (Attributes, Predictor Variables)					**Target Feature** (Class variable)
SubAllCap	TrustSend	InvRet	...	B'zambia'	Spam
y	n	n	...	n	y
n	n	n	...	n	n
n	y	n	...	n	y
n	n	n	...	n	n
...	...	...	...	...	...

- Columns correspond to attributes given by a name and a state space (attributes are basically the same as (chance) variables).
- Rows correspond to examples (also called cases or instances): observations of joint occurrences of values of the attributes.
- In prediction problems, there is a distinguished target attribute. When the target attribute is discrete, it is usually called the class variable. The attributes used for prediction are then called predictor attributes.

In table above: *Spam* is class variable for the prediction problem: predict whether a mail is spam, given characteristics of the mail.

Continuous Attributes

current temp	*pressure change (24h)*	*rain tomorrow*	*temp tomorrow*
16.8	-8.5	yes	15.3
21.7	2.1	no	22.5
19.5	-1.4	no	20.4
8.4	0.5	yes	7.2
...	...	...	...

- Classification problem: predict whether it rains tomorrow, given current temperature and pressure change (*rain tomorrow* is class variable).
- Regression problem: predict temperature tomorrow, given current temperature and pressure change (*temperature tomorrow* is target attribute).
- Clustering problem: identify groups of observations representing similar weather patterns (e.g. stable winter high pressure situations).

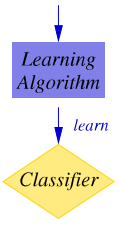
⇝ one data set can be the basis for many different learning tasks!

Classification, Regression

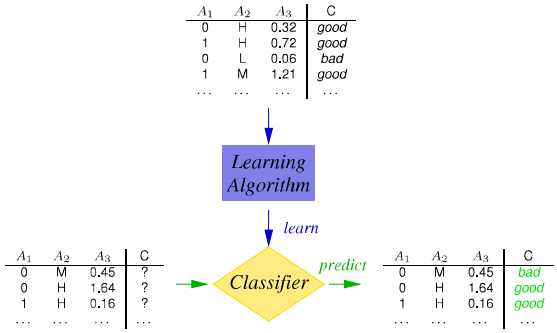
- Predicting a discrete target feature: **classification**
- Agent/Model/Program that classifies: **classifier**
- Predicting a continuous (numeric) target feature: **regression**
- Agent/Model/Program that performs regression: **regression model**

Learning and Using a Classifier

A_1	A_2	A_3	C
0	H	0.32	*good*
1	H	0.72	*good*
0	L	0.06	*bad*
1	M	1.21	*good*
...	...	...	...



Learning and Using a Classifier



Ingredients

Key ingredients of a learning method are:

- a **Hypothesis Space**: set of all possible classifiers that could be learned based on a given **Representation**
- an **Evaluation Measure** that is used to decide how good a candidate hypothesis is
- a **Search** or **Optimization** method used to find a hypothesis that scores high according to the evaluation measure.

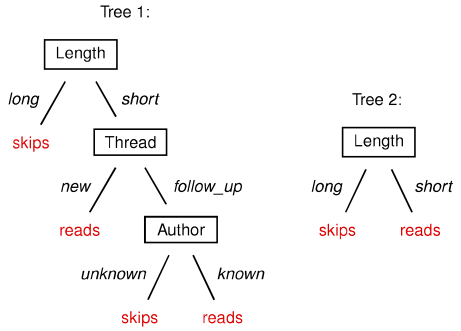
Decision Trees

Example: Training Data

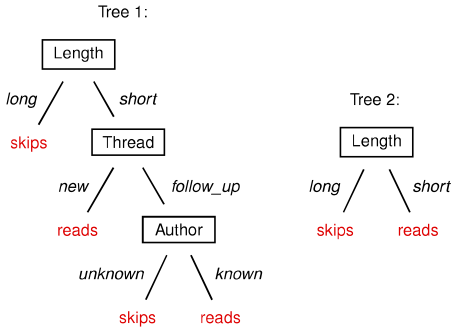
User preference data:

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads
e_{19}	unknown	new	long	work	?
e_{20}	unknown	follow Up	long	home	?
e_{21}	unknown	follow Up	short	home	?

Example: Decision Trees



Example: Decision Trees



Tree 1 is equivalent to the following logic program:

$$skips \leftarrow long$$
$$reads \leftarrow short \wedge new$$
$$reads \leftarrow short \wedge follow\_up \wedge known$$
$$skips \leftarrow short \wedge follow\_up \wedge unknown$$

Example: Classifications

Example	Author	Thread	Length	WhereRead	UserAction	Tree 1	Tree 2
e_1	known	new	long	home	skips	skips	skips
e_2	unknown	new	short	work	reads	reads	reads
e_3	unknown	follow Up	long	work	skips	skips	skips
e_4	known	follow Up	long	home	skips	skips	skips
e_5	known	new	short	home	reads	reads	reads
e_6	known	follow Up	long	work	skips	skips	skips
e_7	unknown	follow Up	short	work	skips	skips	reads
e_8	unknown	new	short	work	reads	reads	reads
e_9	known	follow Up	long	home	skips	skips	skips
e_{10}	known	new	long	work	skips	skips	skips
e_{11}	unknown	follow Up	short	home	skips	skips	reads
e_{12}	known	new	long	work	skips	skips	skips
e_{13}	known	follow Up	short	home	reads	reads	reads
e_{14}	known	new	short	work	reads	reads	reads
e_{15}	known	new	short	home	reads	reads	reads
e_{16}	known	follow Up	short	work	reads	reads	reads
e_{17}	known	new	short	home	reads	reads	reads
e_{18}	unknown	new	short	work	reads	reads	reads
e_{19}	unknown	new	long	work	?	skips	skips
e_{20}	unknown	follow Up	long	home	?	skips	skips
e_{21}	unknown	follow Up	short	home	?	skips	reads

Learning Decision Trees

Top-down construction:

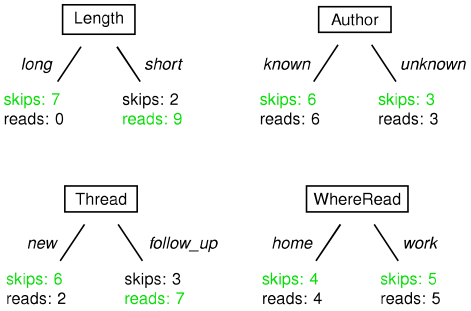
procedure *DecisionTreeLearner*$(\mathbf{X}, Y, E)$
  // $\mathbf{X} = \{X_1, \ldots, X_n\}$: input features
  // Y: target feature
  // E: set of training examples
1. **if** stopping criterion is true
2. **return** *leaf node* labeled with most frequent target feature value in E
3. **else**
4. select feature $X_i \in \mathbf{X}$
  // let v_1, v_2 be the possible values of X_i
5. $E_1 = \{e \in E : \mathit{val}(e, X_i) = v_1\}$
6. $T_1 = \mathit{DecisionTreeLearner}(\mathbf{X}, Y, E_1)$
7. $E_2 = \{e \in E : \mathit{val}(e, X_i) = v_2\}$
8. $T_2 = \mathit{DecisionTreeLearner}(\mathbf{X}, Y, E_2)$
9. **return** X_i

T_1 T_2

Choosing X_i

Key question: which X_i to choose in line 4.?

Approach: choose the feature that would provide the best classifier if construction would terminate with that feature.



Showing

- Number of examples with class labels skips,reads belonging to different sub-trees
- Green: predicted class label (possibly a tie between two labels)

Class Purity

Principle: Prefer features that split the examples into *class pure* subsets:

pure:	*nearly pure*:	*impure*:
skips: 7, reads: 0	skips: 6, reads: 1	skips: 6, reads: 5
skips: 0, reads: 5	skips: 2, reads: 15	skips: 7, reads: 7
skips: 11, reads: 0	skips: 11, reads: 1	skips: 13, reads: 12

Normalized to probabilities:

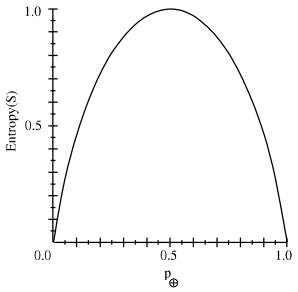
pure:	*nearly pure*:	*impure*:
skips: 1, reads: 0	skips: 0.85, reads: 0.15	skips: 0.54, reads: 0.46
skips: 0, reads: 1	skips: 0.12, reads: 0.88	skips: 0.5, reads: 0.5
skips: 1, reads: 0	skips: 0.91, reads: 0.09	skips: 0.52, reads: 0.48

Entropy

Purity Measure

For a probability distribution $(p, 1-p)$ of a two-valued class label, define

$$h(p, 1-p) = -p \cdot \mathit{log}(p) - (1-p) \cdot \mathit{log}(1-p)$$



- High values for impure distributions
- Maximal for $(0.5, 0.5)$
- Zero for $(1, 0)$ and $(0, 1)$

Examples:

- Entropy$(0.5, 0.5) = -0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5) = 0.5 \cdot 1 + 0.5 \cdot 1 = 1$
- Entropy$(0.35, 0.65) = -0.35 \cdot \log_2(0.35) - 0.65 \cdot \log_2(0.65) = 0.93$
- Entropy$(0, 1) = -0 \cdot \log_2(0) - 1 \cdot \log_2(1) = -0 - 0 = 0$
- Entropy$(1, 0) = -1 \cdot \log_2(1) - 0 \cdot \log_2(0) = -0 - 0 = 0$

Entropy II

Generalization to larger domains

For a probability distribution on domain with n elements:

$$\mathbf{p} = (p_1, p_2, \ldots, p_n) \qquad (p_n = 1 - \sum_{i=1}^{n-1} p_i)$$

define entropy:

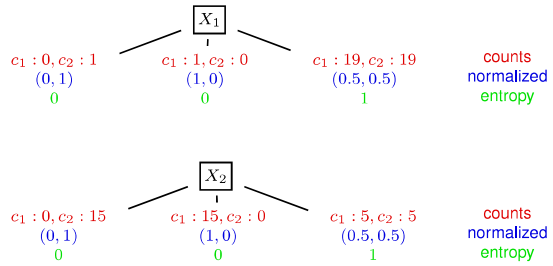
$$h(\mathbf{p}) = -\sum_{i=1}^{n} p_i \cdot \mathit{log}(p_i)$$

Again:

- Maximal for $\mathbf{p} = (1/n, \ldots, 1/n)$
- Zero for $\mathbf{p} = (1, 0, \ldots, 0)$, $\mathbf{p} = (0, 1, 0, \ldots, 0)$,..., $\mathbf{p} = (0, \ldots, 0, 1)$

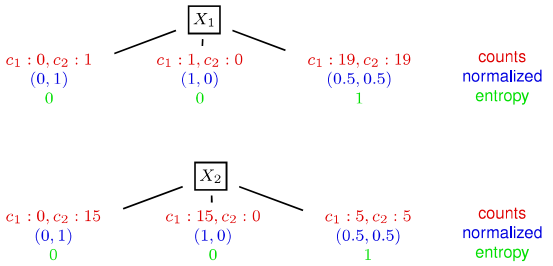
Expected Entropy Example

We prefer features that split into subsets with low entropy, but consider example for binary class variable (values c_1, c_2 with initial counts $c_1 : 20, c_2 : 20$), and two 3-valued features X_1, X_2:



Expected Entropy Example

We prefer features that split into subsets with low entropy, but consider example for binary class variable (values c_1, c_2 with initial counts $c_1 : 20, c_2 : 20$), and two 3-valued features X_1, X_2:



X_2 provides a better division of examples than X_1. It gives a lower *expected entropy*:

$$(1/40) \cdot 0 + (1/40) \cdot 0 + (38/40) \cdot 1 > (15/40) \cdot 0 + (15/40) \cdot 0 + (10/40) \cdot 1$$

Expected Entropy and Information Gain

For feature X with domain $v_1, \ldots, v_n$, let:

- E_i be the set of examples with $X = v_i$
- $q_i = |E_i|/|E|$
- h_i the entropy of the class label distribution in E_i

The **expected entropy** from splitting on X then is:

$$h(\textit{Class} \mid X) = \sum_{i=1}^{n} q_i \cdot h_i$$

Let

- $h(\textit{Class})$: entropy of class label distribution before splitting

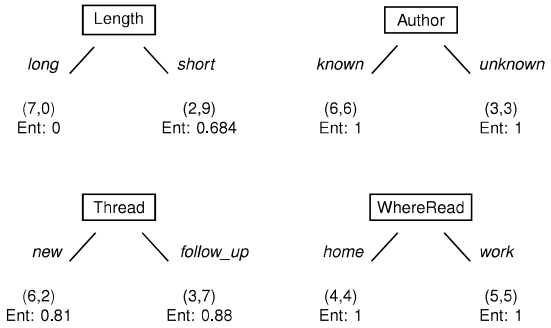
The **Information Gain** from splitting on X then is:

$$h(\textit{Class}) - h(\textit{Class} \mid X)$$

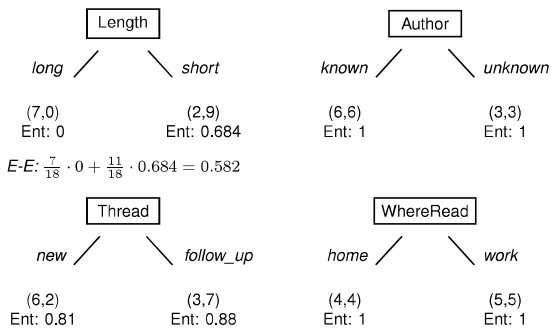
Information Gain in Decision Tree Learning

- In line 4. of algorithm choose feature X_i that gives highest information gain.

Information gain

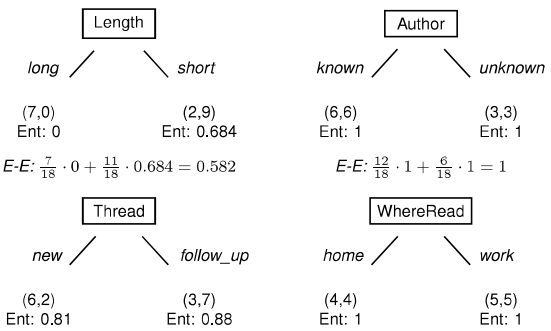


Information gain



E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

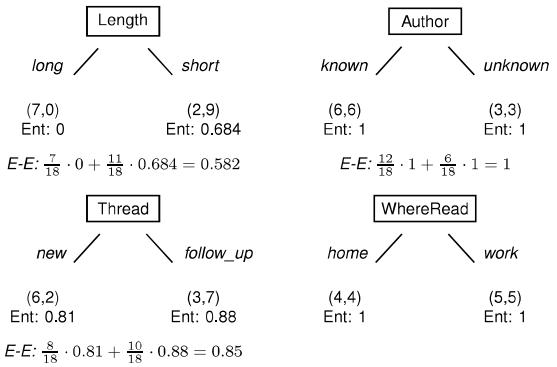
Information gain



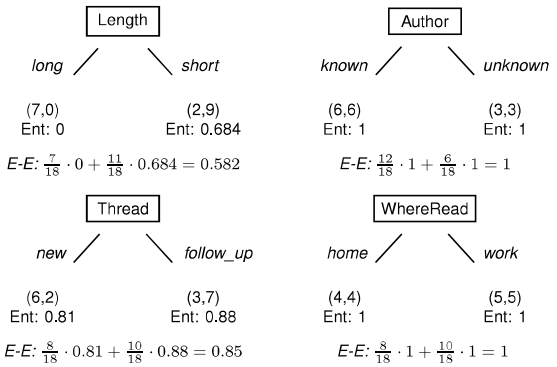
E-E: $\frac{7}{18} \cdot 0 + \frac{11}{18} \cdot 0.684 = 0.582$

E-E: $\frac{12}{18} \cdot 1 + \frac{6}{18} \cdot 1 = 1$

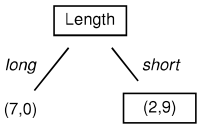
Information gain



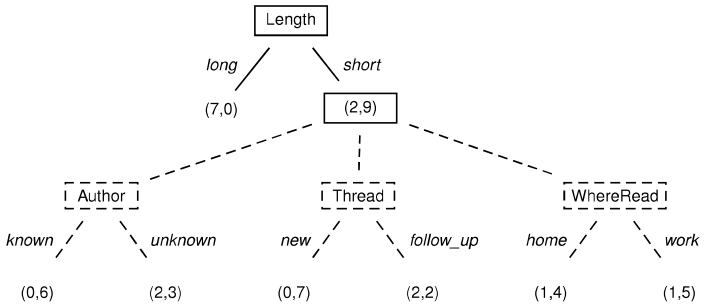
Information gain



Constructing the decision tree



Constructing the decision tree



Continuous/many-valued attributes I

The information gain measure favors attributes with many values:

- For example, the attribute Date (with the possible dates as states) will have a very high information gain but is unable to generalize!

One approach for avoiding this problem is to select attributes based on GainRation:

$$\text{GainRation}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|},$$

where S_i is the subset of examples produced by splitting on the i'th value of A.

Note that SplitInformation is the entropy of S w.r.t. the values of A.

Continuous/many-valued attributes I

We require that the attributes being tested are discrete valued. So in order to test a continuous valued attribute we need to "discretize" it.

Suppose that the training examples are associated with the attribute Temperature:

Temperature:	40	48	60	72	80	90
Rain tomorrow:	*yes*	*yes*	*no*	*no*	*no*	*yes*

Continuous/many-valued attributes I

We require that the attributes being tested are discrete valued. So in order to test a continuous valued attribute we need to "discretize" it.

Suppose that the training examples are associated with the attribute Temperature:

Temperature:	40	48	60	72	80	90
Rain tomorrow:	*yes*	*yes*	*no*	*no*	*no*	*yes*

Create a new boolean valued attribute by first testing the two candidate thresholds:

- (48+60)/2
- (80+90)/2

Next, pick the one with highest information gain (i.e., $\text{Temperature}_{>54}$)

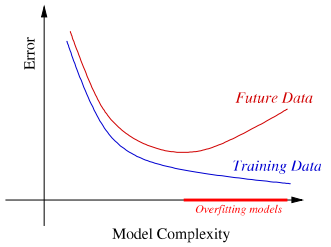
Overfitting

Noise in data may lead to a bad classifier. In particularly, if the decision tree fits the data perfectly. This is called overfitting.

Definition

A hypothesis h is said to overfit the training data if there exists some alternative hypotheses h', such that:

- h has smaller error than h' over the training data, but
- h' has a smaller error than h over the entire distribution of instances.



Machine Intelligence

Lecture 8: Learning II

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

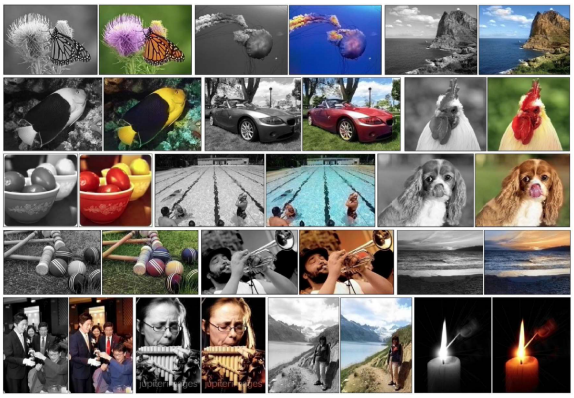
Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- **Machine learning**
- Planning
- Reinforcement learning
- Multi-agent systems

Neural Networks

Example applications

Colorization of images



Zhang, Isola, Efros. Colorful Image Colorization. In ECCV, 2016.

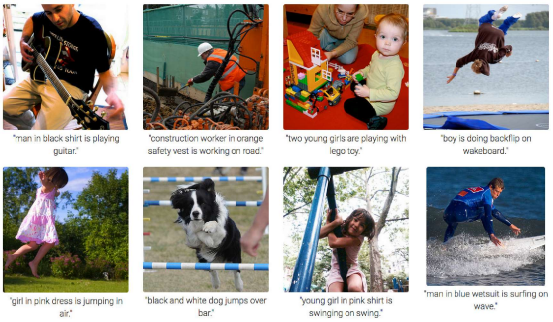
Example applications

Text-image translation

https://research.googleblog.com/2015/07/how-google-translate-squeezes-deep.html

Example applications

Image captioning



Andrej Karpathy, Li Fei-Fei, Deep Visual-Semantic Alignments for Generating Image Descriptions, CVPR 2015

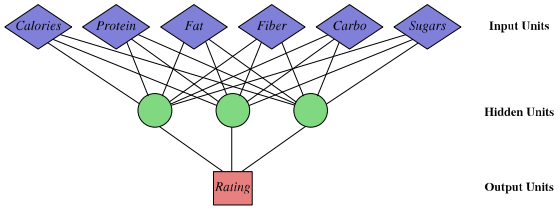
Regression: Example

Cereals Data

Name	Calories	Protein	Fat	Fiber	Carbo	Sugars	Rating
All-Bran	70	4	1	9	7	5	59.42
All-Bran_with_Extra_Fiber	50	4	0	14	8	0	93.70
Almond_Delight	110	2	2	1	14	8	34.38
Apple_Cinnamon_Cheerios	110	2	2	1.5	10.5	10	29.50
Apple_Jacks	110	2	0	1	11	14	33.17
Basic_4	130	3	2	2	18	8	37.03
Bran_Chex	90	2	1	4	15	6	49.12
Bran_Flakes	90	3	0	5	13	5	53.31
Cap_n_Crunch	120	1	2	0	12	12	18.04
Cheerios	110	6	2	2	17	1	50.76
...	...	...	...	...	...	...	...

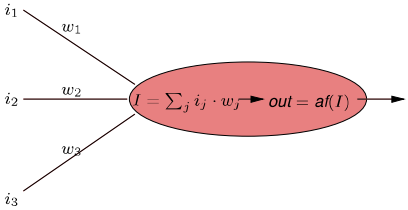
Problem: predict nutritional rating of cereal.

Neural Network



- Layered network of computational **units** (or *neurons*)
- Each unit has outputs of all units in preceding layer as inputs
- With each connection in the network there is an associated **weight**

Single Neuron



Two step computation:

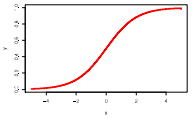
- Combine inputs as *weighted sum*
- Compute output by **activation function** of combined inputs

Activation Functions

The most common activation functions are:

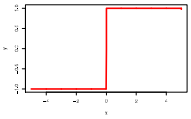
Sigmoid

$af(x) = 1/(1 + e^{-x})$



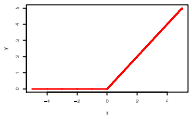
Sign

$af(x) = sign(x)$



Relu

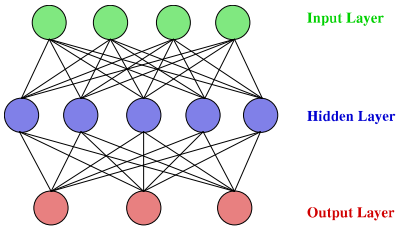
$af(x) = \max(0, x)$



- If activation function is sigmoid, i.e. $out = \sigma(\sum_j i_j \cdot w_j)$, we also talk of *squashed linear function*.
- For the output neuron also the **identity function** is used: $af(x) = id(x) = x$

Neural Networks

Neural Network Semantics



Given

- the network structure,
- the weights associated with links/nodes,
- the activation function (usually the same for all hidden/output nodes)

a neural network with n input and k output nodes defines k real-valued functions on continuous input attributes:

$$o_i(a_1, \ldots, a_n) \in \mathbb{R} \qquad (i = 1, \ldots, k).$$

Discrete Inputs

If the regression should also use discrete predictor attributes, e.g.:

Calories	*Protein*	*Sugars*	*Vitamins*	*Manufacturer*	*Rating*
70	105	8	135	Kellogs	59.3
110	80	23	99	Nabisco	43.6
...	...	...	...	...	...

Discrete Inputs

If the regression should also use discrete predictor attributes, e.g.:

Calories	*Protein*	*Sugars*	*Vitamins*	*Manufacturer*	*Rating*
70	105	8	135	Kellogs	59.3
110	80	23	99	Nabisco	43.6
...	...	...	...	...	...

replace discrete attributes with 0-1-valued indicator variables for their possible values:

Calories	*Protein*	*Sugars*	*Vitamins*	*M_Kellogs*	*M_Nabisco*	*M_xxx*	*Rating*
70	105	8	135	1	0	...	59.3
110	80	23	99	0	1	...	43.6
...	...	...	...	...	...	...	...

Discrete Features in general

Neural networks can also handle discrete features as inputs or outputs:

Indicator variables

- For each value x_i of X with domain $\{x_1, \ldots, x_k\}$ introduce a binary feature *X_is_*x_i with values 0,1.
- Encode input $X = x_i$ by inputs

$$X\_is\_x_0 = 0, \ldots, X\_is\_x_{i-1} = 0, X\_is\_x_i = 1, X\_is\_x_{i+1} = 0, \ldots, X\_is\_x_k = 0$$

Discrete Features in general

Neural networks can also handle discrete features as inputs or outputs:

Indicator variables

- For each value x_i of X with domain $\{x_1, \ldots, x_k\}$ introduce a binary feature *X_is_*x_i with values 0,1.
- Encode input $X = x_i$ by inputs

$$X\_is\_x_0 = 0, \ldots, X\_is\_x_{i-1} = 0, X\_is\_x_i = 1, X\_is\_x_{i+1} = 0, \ldots, X\_is\_x_k = 0$$

Numerical Encoding

Translate values into numbers, e.g.:

- *true*,*false* $\mapsto$ 1,0
- *low*,*medium*,*high* $\mapsto$ 0,1,2

Discrete Features in general

Neural networks can also handle discrete features as inputs or outputs:

Indicator variables

- For each value x_i of X with domain $\{x_1, \ldots, x_k\}$ introduce a binary feature *X_is_*x_i with values 0,1.
- Encode input $X = x_i$ by inputs

$$X\_is\_x_0 = 0, \ldots, X\_is\_x_{i-1} = 0, X\_is\_x_i = 1, X\_is\_x_{i+1} = 0, \ldots, X\_is\_x_k = 0$$

Numerical Encoding

Translate values into numbers, e.g.:

- *true*,*false* $\mapsto$ 1,0
- *low*,*medium*,*high* $\mapsto$ 0,1,2

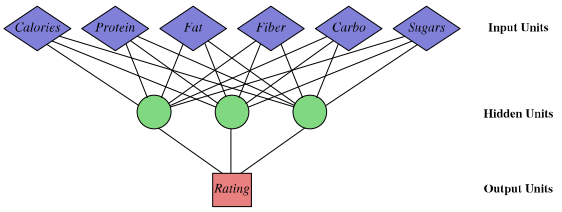
Probably not sensible:

- *red*,*green*,*blue* $\mapsto$ 0,1,2

because *blue* is not "two times *green*"

Neural Networks for Regression

Task: predict the (health-)*rating* of breakfast cereals based on their contents.
NN regression model, all predictor attributes are continuous:

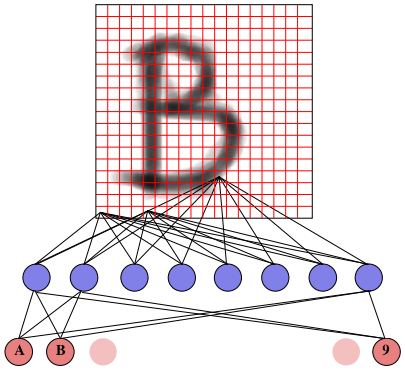


Neural Networks for Classification

Task: hand-written character recognition. Predictor attributes: (continuous) grey-scale values for 18×18 grid cells. Class label: one of A,...,Z,0,...,9.

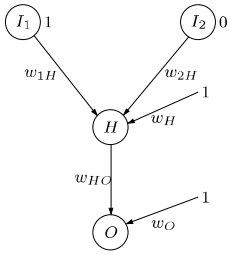
Neural Networks for Classification

Task: hand-written character recognition. Predictor attributes: (continuous) grey-scale values for 18×18 grid cells. Class label: one of A,...,Z,0,...,9.



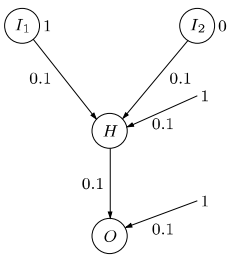
Use one output node for each class label.
Classify instance by class label associated with output node with highest output value.

Propagation in Neural Networks



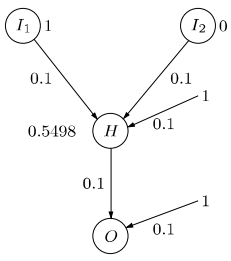
The input nodes are set to 1 and 0, respectively.

Propagation in Neural Networks



The input nodes are set to 1 and 0, respectively.

Propagation in Neural Networks



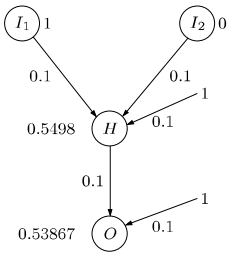
The output of of neuron H is:

$$o_H = \sigma(1 \cdot 0.1 + 0 \cdot 0.1 + 1 \cdot 0.1) = 0.5498.$$

The input nodes are set to 1 and 0, respectively.

Neural Networks

Propagation in Neural Networks



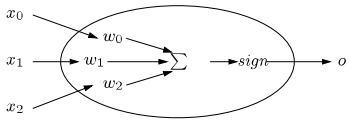
The output of of neuron O is:

$$o_H = \sigma(0.5498 \cdot 0.1 + 1 \cdot 0.1) = 0.53867.$$

The input nodes are set to 1 and 0, respectively.

Expressive power

The perceptron



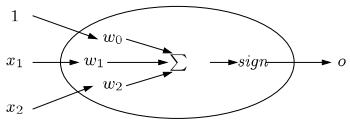
- No hidden layer
- One output neuron o
- *sign* activation function

Function computed:

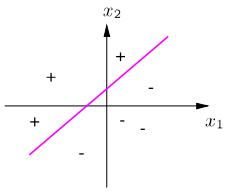
$$O(x_1, \ldots x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

Convention: from now on assume that x_0 is an input neuron with constant input value 1. Then write $\mathbf{w} \cdot \mathbf{x}$ for $w_0 x_0 + w_1 x_1 + \ldots w_n x_n$.

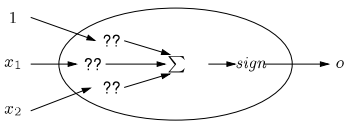
Expressive power

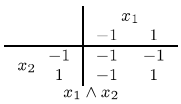


The decision surface of a two-input perceptron $a(x_1, x_2) = \text{sign}(x_1 \cdot w_1 + x_2 \cdot w_2 + w_0)$ is given by a straight line, separating positive and negative examples.



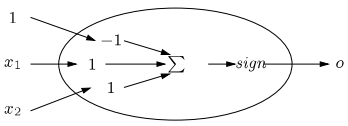
Expressive power

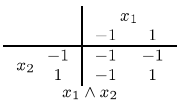




Can the perceptron represent the Boolean function?

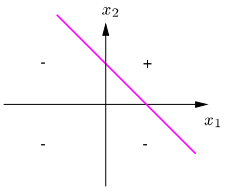
Expressive power



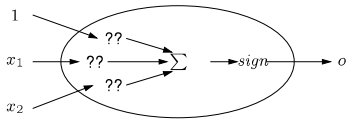


This perceptron specifies the decision surface:

$$-1 + 1 \cdot x_1 + 1 \cdot x_2 = 0$$



Perceptron examples

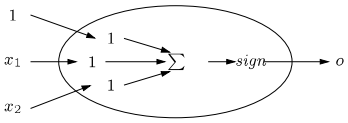


x_2 \ x_1	-1	1
-1	-1	1
1	1	1

$x_1 \vee x_2$

Can the perceptron represent the Boolean function?

Perceptron examples

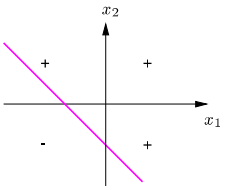


x_2 \ x_1	-1	1
-1	-1	1
1	1	1

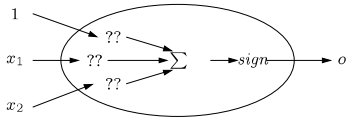
$x_1 \vee x_2$

This perceptron specifies the decision surface:

$$1 + 1 \cdot x_1 + 1 \cdot x_2 = 0$$



Expressive power

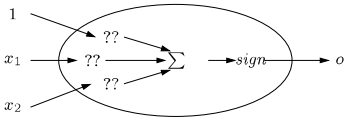


x_2 \ x_1	-1	1
-1	-1	1
1	1	-1

x_1 xor x_2

Can the perceptron represent the Boolean function?

Expressive power



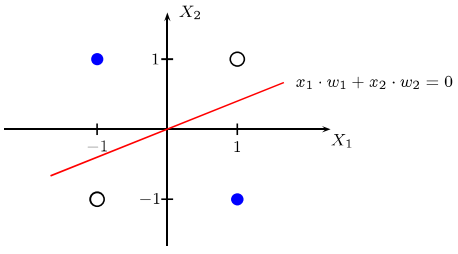
x_2 \ x_1	-1	1
-1	-1	1
1	1	-1

x_1 xor x_2

We cannot specify any values for the weights so that the perceptron can represent the "xor" function:

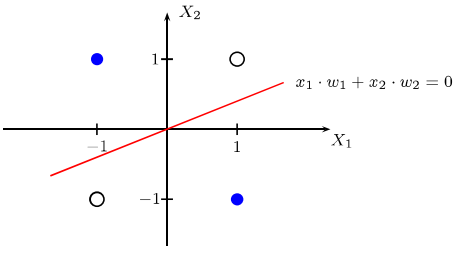
The examples are not linear separable!

Linear separable



$x_1 \cdot w_1 + x_2 \cdot w_2 > 0$

Linear separable

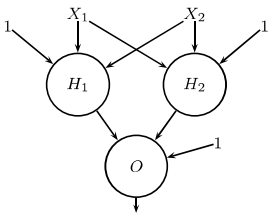


$x_1 \cdot w_1 + x_2 \cdot w_2 > 0$

Only linearly separable functions can be computed by a single perceptron.

XOR again

Can multiple neurons help?

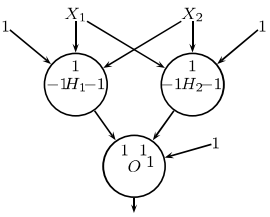


X_2 \ X_1	-1	1
-1	-1	1
1	1	-1

X_1 xor X_2

XOR again

Can multiple neurons help?

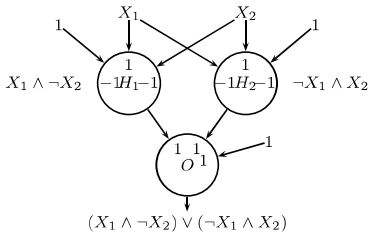


X_2 \ X_1	-1	1
-1	-1	1
1	1	-1

X_1 xor X_2

XOR again

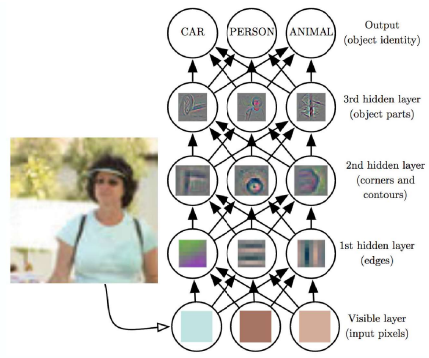
Can multiple neurons help?



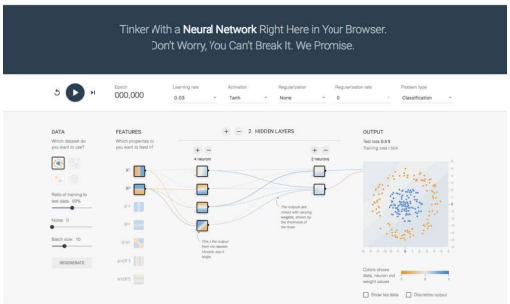
X_2 \ X_1	-1	1
-1	-1	1
1	1	-1

X_1 xor X_2

Depth: repeated composition



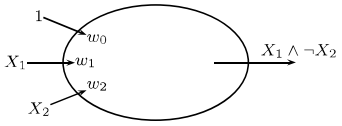
Playground



playground.tensorflow.org

The task of learning

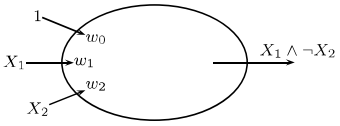
Learning weights and threshold I



		X_1	
		-1	1
X_2	-1	-1	1
	1	-1	-1

$X_1 \wedge \neg X_2$

Learning weights and threshold I



		X_1	
		-1	1
X_2	-1	-1	1
	1	-1	-1

$X_1 \wedge \neg X_2$

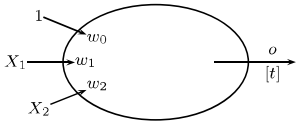
We have:

- $\mathcal{D} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4)$ input vectors (cases).
- $\mathbf{t} = (-1, 1, -1, -1)$ vector of target outputs.
- $\mathbf{w} = (w_0, w_1, w_2)$ vector of current parameters.
- $\mathbf{o} = (o_1, o_2, o_3, o_4)$ vector of current outputs.

We request:

- $\mathbf{w}^*$ parameters yielding $\mathbf{o} = \mathbf{t}$.

Learning weights and threshold I



Error: $E = t - o$

		X_1	
		-1	1
X_2	-1	-1	1
	1	-1	-1

$X_1 \wedge \neg X_2$

Weight updating procedure

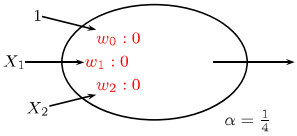
The weights are updated as follows:

- $E > 0 \Rightarrow o$ shall be increased $\Rightarrow \mathbf{x} \cdot \mathbf{w}$ up $\Rightarrow \mathbf{w} := \mathbf{w} + \alpha E \mathbf{x}$
- $E < 0 \Rightarrow o$ shall be decreased $\Rightarrow \mathbf{x} \cdot \mathbf{w}$ down $\Rightarrow \mathbf{w} := \mathbf{w} + \alpha E \mathbf{x}$

α is called the learning rate.

With $\mathcal{D}$ linearly separable and α not too large this procedure will converge to a correct set of parameters.

Example: $X_1 \wedge X_2$



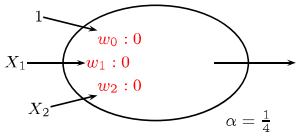
$$o = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

		X_1	
		-1	1
X_2	-1	-1	-1
	1	-1	1

$X_1 \wedge X_2$

Cases:	$(1,-1,-1)$	$(1,1,-1)$	$(1,-1,1)$	$(1,1,1)$
$\mathbf{t}$:	-1	-1	-1	1
o:				
E:				

Example: $X_1 \wedge X_2$



$$o = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

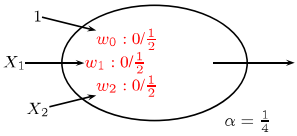
		X_1	
		-1	1
X_2	-1	-1	-1
	1	-1	1

$X_1 \wedge X_2$

Cases:	$(1,-1,-1)$	$(1,1,-1)$	$(1,-1,1)$	$(1,1,1)$
$\mathbf{t}$:	-1	-1	-1	1
o:	-1	-1	-1	-1
E:	0	0	0	$\underline{2}$

$$w := \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \frac{1}{4} \cdot 2 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

Example: $X_1 \wedge X_2$



$$o = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

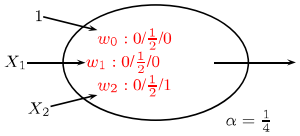
		X_1	
		-1	1
X_2	-1	-1	-1
	1	-1	1

$X_1 \wedge X_2$

Cases:	$(1,-1,-1)$	$(1,1,-1)$	$(1,-1,1)$	$(1,1,1)$
$\mathbf{t}$:	-1	-1	-1	1
o:	-1	1	1	1
E:	0	$\underline{-2}$	-2	0

$$w := \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} + \frac{1}{4} \cdot -2 \cdot \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Example: $X_1 \wedge X_2$



$$o = \begin{cases} 1 & \text{if } w_1 x_1 + w_2 x_2 + w_0 > 0 \\ -1 & \text{otherwise} \end{cases}$$

		X_1	
		-1	1
X_2	-1	-1	-1
	1	-1	1

$X_1 \wedge X_2$

Cases:	$(1,-1,-1)$	$(1,1,-1)$	$(1,-1,1)$	$(1,1,1)$
$\mathbf{t}$:	-1	-1	-1	1
o:	-1	-1	1	1
E:	0	0	$\underline{-2}$	0

ETC... for a finite number of steps

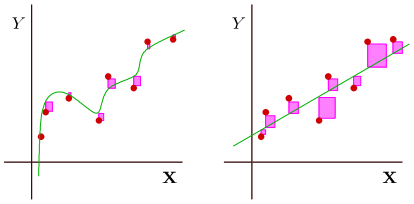
Sum of Squared Errors

Given:

- data (training examples): $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, N)$, with
  - $\mathbf{x}_i$: value of input features $\mathbf{X}$
  - y_i: value of output feature Y
- a neural network that computes outputs $o_i = \mathit{out}(\mathbf{x}_i)$

define **sum of squares error**

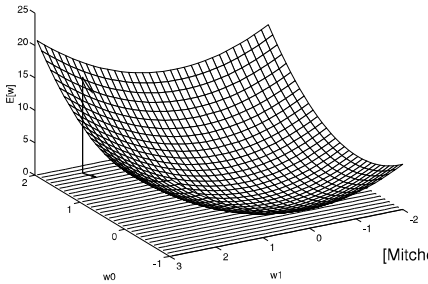
$$\mathit{SSE} = \sum_{i=1}^{N} (y_i - o_i)^2$$



Set of training examples (red dots), function defined by neural network (green), squared errors for each training example (area of magenta squares).

Minimizing SSE

For a given dataset, the SSE is a smooth, convex function of the weights.
Example for $n = 2$ (and a linear activation function):



[Mitchell, Fig.4.4]

$\rightsquigarrow$ weights $\mathbf{w}$ that minimize $E(\mathbf{w})$ can be found by gradient descent.

Gradient Descent Learning

The gradient is the vector of partial derivatives:

$$\nabla E[\mathbf{w}] = \left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n} \right)$$

The partial derivatives are (with a linear activation function):

$$\frac{\partial E}{\partial w_k} = \sum_{i=1}^{N} (t_i - \mathbf{w} \cdot \mathbf{x}_i)(-x_{i,k}).$$

Gradient descent rule:

Initialize $\mathbf{w}$ with random values
repeat
  $\mathbf{w} := \mathbf{w} - \eta \nabla E(\mathbf{w})$
until $\nabla E(\mathbf{w}) \approx 0$

(η is again a small constant, the learning rate).

Properties:

- The procedure converges to the weights $\mathbf{w}$ that minimize the SSE (if η is small enough).

Stochastic Gradient Descent

Variation of gradient descent: instead of following the gradient computed from the whole dataset:

$$\frac{\partial E}{\partial w_i} = \sum_{k=1}^{N} (t_k - \mathbf{w} \cdot \mathbf{x}_k)(-x_{k,i}),$$

iterate through the data instances one by one, and in one iteration follow the gradient defined by a single data instance $(\mathbf{x}_k, t_k)$:

$$\frac{\partial E}{\partial w_i} = (t_k - \mathbf{w} \cdot \mathbf{x}_k)(-x_{k,i}),$$

Neural Networks

The Task of Learning

Given: structure and activation functions. To be learned: weights.
Goal: given the training examples

Input				Output			
X_1	X_2	$\ldots$	X_n	X_1	X_2	$\ldots$	X_m
$x_{1,1}$	$x_{2,1}$	$\ldots$	$x_{n,1}$	$t_{1,1}$	$t_{2,1}$	$\ldots$	$t_{m,1}$
$x_{1,2}$	$x_{2,2}$	$\ldots$	$x_{n,2}$	$t_{1,2}$	$t_{2,2}$	$\ldots$	$t_{m,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$x_{1,N}$	$x_{2,N}$	$\ldots$	$x_{n,N}$	$t_{1,N}$	$t_{2,N}$	$\ldots$	$t_{m,N}$

Find the weights that minimize the *sum of squared errors (SSE)*

$$\sum_{i=1}^{N} \sum_{j=1}^{m} (t_{j,i} - o_{j,i})^2,$$

where $o_{j,i}$ is the value of the jth output neuron for the ith data instance.

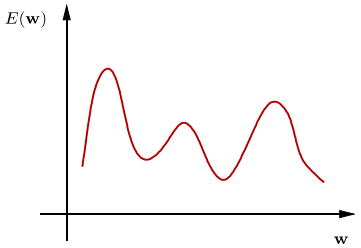
Gradient Descent for Multilayer NN

As for perceptron with SSE error:

- Error is smooth function of weights $\mathbf{w}$
- Can use gradient descent to optimize weights

but:

- Error no longer convex, can have multiple local minima:

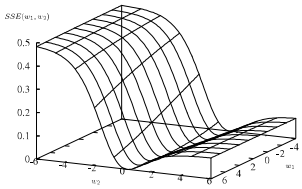


- Partial derivatives more difficult to compute

Neural Networks

Learning

Basic principle: SSEis a differentiable function of the weights (for differentiable activation functions such as the sigmoid function!). Use *gradient descent* to optimize SSE:



$$\nabla SSE(\mathbf{w}) = \left(\frac{\partial SSE}{\partial w_0}, \ldots, \frac{\partial SSE}{\partial w_n}\right)$$

specifies the direction of steepest increase in SSE.

Hence, our new training rule becomes:

$$w_i := w_i + \Delta w_i,$$

where

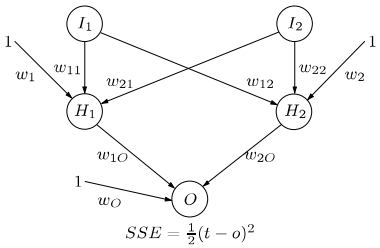
$$\Delta w_i = -\eta \frac{\partial SSE}{\partial w_i}$$

In practice: use the *back propagation algorithm* (approximation of gradient descent)

Neural Networks

The Principle of Back Propagation

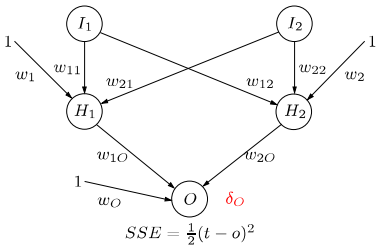
Training examples provide target values for only network outputs, so no target values are directly available for indicating the error of the hidden units' values.



Neural Networks

The Principle of Back Propagation

Training examples provide target values for only network outputs, so no target values are directly available for indicating the error of the hidden units' values.

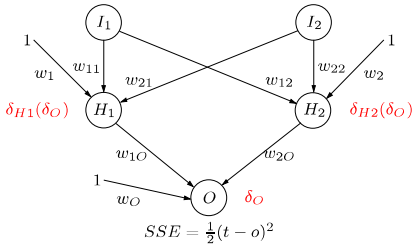


Idea: Calculate an error term δ_h for a hidden unit by taking the weighted sum of the error terms, δ_k, for each output units it influences.

Neural Networks

The Principle of Back Propagation

Training examples provide target values for only network outputs, so no target values are directly available for indicating the error of the hidden units' values.



Idea: Calculate an error term δ_h for a hidden unit by taking the weighted sum of the error terms, δ_k, for each output units it influences.

Neural Networks

Updating Rules

When using a sigmoid activation function we can derive the following updating rule:

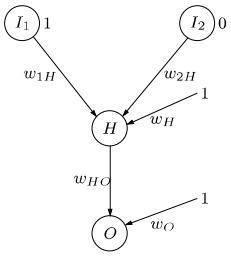
$$w_{ij}^{\text{new}} := w_{ij}^{\text{current}} + \eta \cdot \delta_j \cdot x_{ij},$$

learning rate, error term, input

where

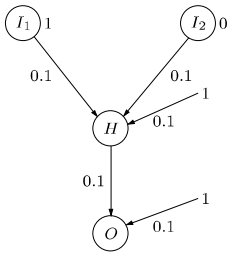
$$\delta_j = \begin{cases} o_j(1-o_j)(t-o_j) & \text{for output nodes,} \\ o_j(1-o_j)\sum_{k=1}^{m} w_{jk}\delta_k & \text{for hidden nodes.} \end{cases}$$

Back Propagation Example



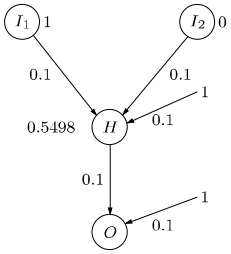
Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Back Propagation Example



Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Back Propagation Example

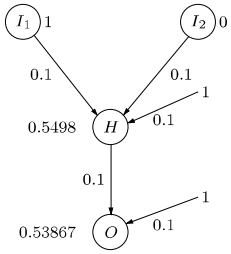


The output of of neuron H is:

$$o_H = \sigma(1 \cdot 0.1 + 0 \cdot 0.1 + 1 \cdot 0.1) = 0.5498.$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Back Propagation Example

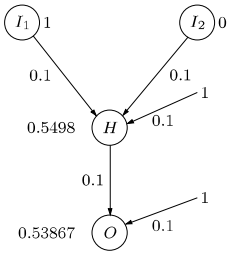


The output of of neuron O is:

$$o_H = \sigma(0.5498 \cdot 0.1 + 1 \cdot 0.1) = 0.53867.$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Back Propagation Example

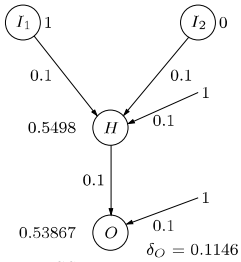


The SSEvalue is:

$$SSE = (1 - 0.53867)^2 = 0.21283.$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Back Propagation Example



The error term for node O is:

$$\delta_O = 0.53867 \cdot (1 - 0.53867) \cdot (1 - 0.53867) = 0.1146.$$

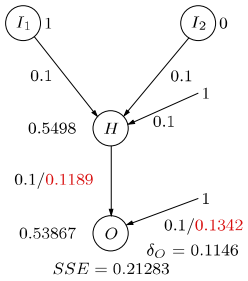
Recall:

$$\delta_O = o_j(1 - o_j)(O - o_j).$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Neural Networks

Back Propagation Example



The updated weights are:

$$w_O = 0.1 + [0.3 \cdot 0.1146 \cdot 1] = 0.1342,$$
$$w_{HO} = 0.1 + [0.3 \cdot 0.1146 \cdot 0.5498] = 0.1189.$$

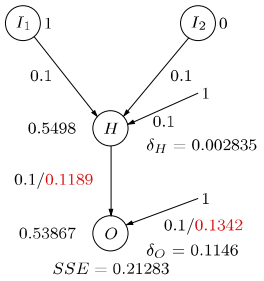
Recall:

$$w_{ij}^{\text{new}} := w_{ij}^{\text{current}} + [\eta \cdot \delta_j \cdot x_{ij}].$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Neural Networks

Back Propagation Example



The error term for node H is:

$$\delta_H = 0.5498 \cdot (1 - 0.5498) \cdot 0.1 \cdot 0.1146 = 0.002836.$$

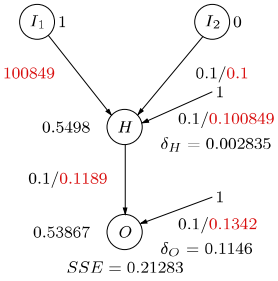
Recall:

$$\delta_H = o_j(1 - o_j) \sum_{k=1}^{m} w_{jk}\delta_k.$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Neural Networks

Back Propagation Example



The updated weights are:

$$w_{1H} = 0.1 + [0.3 \cdot 0.00283 \cdot 1] = 0.100849,$$
$$w_{2H} = 0.1 + [0.3 \cdot 0.00283 \cdot 0] = 0.1,$$
$$w_H = 0.1 + [0.3 \cdot 0.00283 \cdot 1] = 0.100849$$

Recall:

$$w_{ij}^{\text{new}} := w_{ij}^{\text{current}} + [\eta \cdot \delta_j \cdot x_{ij}].$$

Assume that we have the training example $(I_1 = 1, I_2 = 0, O = 1)$.

Machine Intelligence

Lecture 9: Learning

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- **Machine learning**
- Planning
- Clustering
- Multi-agent systems

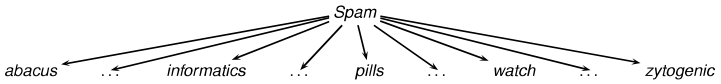
Probabilistic Models

Spam Data

Word occurrence in emails (thousands of input features!):

Mail	abacus	...	informatics	...	pills	...	watch	...	zytogenic	Spam
m_1	n	...	y	...	n	...	n	...	n	no
m_2	n	...	n	...	n	...	n	...	n	yes
m_3	n	...	n	...	n	...	n	...	y	no
m_4	n	...	n	...	n	...	n	...	n	yes
m_5	n	...	n	...	n	...	y	...	n	yes
m_6	n	...	n	...	n	...	n	...	n	yes
m_7	n	...	n	...	n	...	n	...	n	no
m_8	n	...	n	...	n	...	n	...	n	yes
m_9	n	...	y	...	n	...	y	...	n	yes

Naive Bayes Classifier



Classify email as *spam* if

$$P(\textit{Spam} = \textit{yes} \mid \mathbf{X} = \mathbf{x}) > \textit{threshold}$$

($\mathbf{X} = (\textit{abacus}, \ldots, \textit{zytogenic})$, $\mathbf{x}$ a corresponding set of y/n values)

Structural assumption

$$P(a_1, \ldots, a_n, \textit{Spam}) = P(a_1 \mid \textit{Spam}) \cdot P(a_2 \mid \textit{Spam}) \cdots P(a_n \mid \textit{Spam}) \cdot P(\textit{Spam})$$

Learning

- Need to learn entries in conditional probability tables.
- Simplest approach: use **empirical frequencies**, e.g:

$$P(\textit{pills} = y \mid \textit{Spam} = \textit{yes}) = \frac{\#\text{mails with } \textit{pills} = y \text{ and } \textit{Spam} = \textit{yes}}{\#\text{mails with } \textit{Spam} = \textit{yes}}$$

Example: Learning a Naive Bayes

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads

Probabilities to estimate:

NB model:

Example: Learning a Naive Bayes

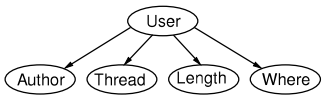
Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads

Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$P(\text{known}|\text{reads}) =$

NB model:



Example: Learning a Naive Bayes

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads

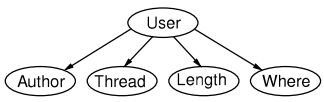
Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{2}{3}$$

$P(\text{known}|\text{skips}) =$

NB model:



Example: Learning a Naive Bayes

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads

Probabilities to estimate:

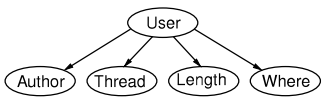
$$P(\text{reads}) = \frac{9}{18}$$

$$P(\text{known}|\text{reads}) = \frac{2}{3}$$

$$P(\text{known}|\text{skips}) = \frac{2}{3}$$

$P(\text{new}|\text{reads}) =$

NB model:



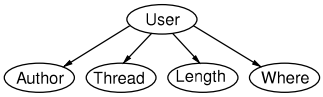
Example: Learning a Naive Bayes

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads

Probabilities to estimate:

$$P(\text{reads}) = \frac{9}{18}$$
$$P(\text{known}|\text{reads}) = \frac{2}{3}$$
$$P(\text{known}|\text{skips}) = \frac{2}{3}$$
$$P(\text{new}|\text{reads}) = \frac{7}{9}$$
$$P(\text{new}|\text{skips}) = \frac{1}{3}$$
$$P(\text{long}|\text{reads}) = \frac{0}{9}$$
$$P(\text{long}|\text{skips}) = \frac{7}{9}$$
$$P(\text{home}|\text{reads}) = \frac{4}{9}$$
$$P(\text{home}|\text{skips}) = \frac{4}{9}$$
$$\dots$$

NB model:



Making predictions using Naive Bayes

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

Making predictions using Naive Bayes

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

$$P(\text{skips}|\text{unknown}, \text{followUp}, \text{short}, \text{home}) = \frac{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home}) + P(\text{reads}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}$$

Making predictions using Naive Bayes

In order to classify a new instance

[Author=unknown, Thread=followUp, Length=short, Where=home]

we calculate

$$P(\text{skips}|\text{unknown}, \text{followUp}, \text{short}, \text{home}) = \frac{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}{P(\text{skips}, \text{unknown}, \text{followUp}, \text{short}, \text{home}) + P(\text{reads}, \text{unknown}, \text{followUp}, \text{short}, \text{home})}$$

For the numerator and denominator we have

$$P(\text{read})P(\text{unknown}|\text{read})P(\text{followUp}|\text{read})P(\text{short}|\text{read})P(\text{home}|\text{read}) = \frac{9}{18}\cdot\frac{1}{3}\cdot\frac{2}{9}\cdot\frac{9}{9}\cdot\frac{4}{9} = 0.0165;$$

$$P(\text{skips})P(\text{unknown}|\text{skips})P(\text{followUp}|\text{skips})P(\text{short}|\text{skips})P(\text{home}|\text{skips}) = \frac{9}{18}\cdot\frac{1}{3}\cdot\frac{2}{3}\cdot\frac{2}{9}\cdot\frac{4}{9} = 0.0110,$$

which gives

$$P(\text{skips}|\text{unknown}, \text{followUp}, \text{short}, \text{home}) = \frac{0.0110}{0.0110 + 0.0165} = 0.4.$$

Estimating probabilities: zero probabilities

When learning the naive Bayes model we estimated $P(\text{long}|\text{reads})$ as

$$P(\text{long}|\text{reads}) = \frac{0}{9},$$

based on 9 cases only $\rightsquigarrow$ unreliable parameter estimates and risk of zero probabilities.

Estimating probabilities: zero probabilities

When learning the naive Bayes model we estimated $P(\text{long}|\text{reads})$ as

$$P(\text{long}|\text{reads}) = \frac{0}{9},$$

based on 9 cases only $\rightsquigarrow$ unreliable parameter estimates and risk of zero probabilities.

Using pseudo counts

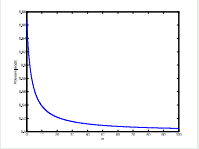
$$P(A = a|C = c) = \frac{N(A = a, C = c) + p_{ac} \cdot m}{N(C = c) + m}$$

where

- p_{ac} is our prior estimate of the probability (often chosen as a uniform distribution) and
- m is a virtual sample size (determining the weight of p_{ac} relative to the observed data).

Example

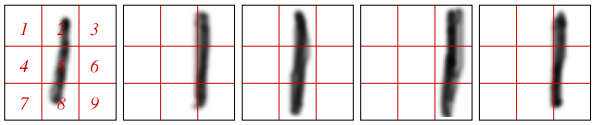
$$P(\text{known}|\text{reads}) = \frac{2 + 0.5 \cdot m}{3 + m}$$



The naive Bayes assumption I

Target: *Symbol* $\in \{A, \ldots, Z, 0, \ldots, 9\}$

Predictors: *Cell-1*,...,*Cell-9* $\in \{b, w\}$.

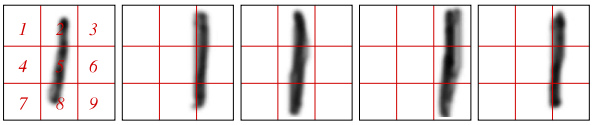


Naive Bayes

The naive Bayes assumption I

Target: *Symbol* $\in \{A, \ldots, Z, 0, \ldots, 9\}$

Predictors: *Cell-1*,...,*Cell-9* $\in \{b, w\}$.



For example:

$$P(Cell-2=b \mid Cell-5=b, Symbol=1) > P(Cell-2=b \mid Symbol=1)$$

Attributes not independent given *Symbol*=1!

MI E19 Probabilistic Models 8

Naive Bayes

The naive Bayes assumption II

Target: *Spam* $\in \{y, n\}$

Predictors: *Subject-all-caps*, *Known-spam-server*, ..., *Contains'Money'* $\in \{y, n\}$.

Naive Bayes

The naive Bayes assumption II

Target: *Spam* $\in \{y, n\}$

Predictors: *Subject-all-caps*, *Known-spam-server*, ..., *Contains'Money'* $\in \{y, n\}$.

For example:

$$P(\text{Body'nigeria'=y} \mid \text{Body'confidential'=y, Spam=y}) \gg P(\text{Body'nigeria'=y} \mid \text{Spam=y})$$

Attributes not independent given *Spam*=yes!

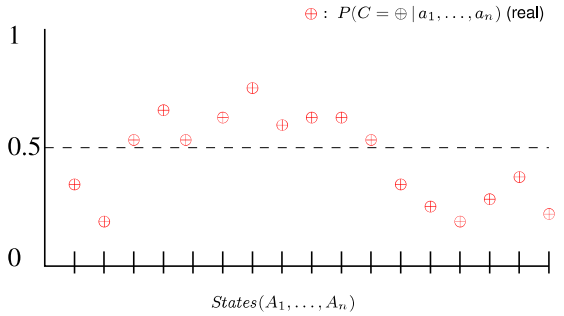
$\rightsquigarrow$ Naive Bayes assumption often not realistic. Nevertheless, Naive Bayes often successful.

MI E19 Probabilistic Models 9

Naive Bayes

The paradoxical success of Naive Bayes

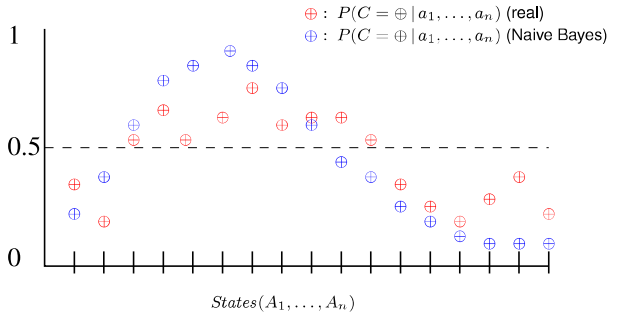
One explanation for the surprisingly good performance of Naive Bayes in many domains: do not require exact distribution for classification, only the right decision boundaries [Domingos, Pazzani 97]



Naive Bayes

The paradoxical success of Naive Bayes

One explanation for the surprisingly good performance of Naive Bayes in many domains: do not require exact distribution for classification, only the right decision boundaries [Domingos, Pazzani 97]



MI E19 Probabilistic Models 10

Naive Bayes

When Naive Bayes must fail

No Naive Bayes Classifier can produce the following classification:

A	B	Class
yes	yes	$\oplus$
yes	no	$\ominus$
no	yes	$\ominus$
no	no	$\oplus$

because assume it did, then:

1. $P(A = y \mid \oplus)P(B = y \mid \oplus)P(\oplus) > P(A = y \mid \ominus)P(B = y \mid \ominus)P(\ominus)$
2. $P(A = y \mid \ominus)P(B = n \mid \ominus)P(\ominus) > P(A = y \mid \oplus)P(B = n \mid \oplus)P(\oplus)$
3. $P(A = n \mid \ominus)P(B = y \mid \ominus)P(\ominus) > P(A = n \mid \oplus)P(B = y \mid \oplus)P(\oplus)$
4. $P(A = n \mid \oplus)P(B = n \mid \oplus)P(\oplus) > P(A = n \mid \ominus)P(B = n \mid \ominus)P(\ominus)$

Naive Bayes

Multiplying the four left sides and the four right sides of these inequalities:

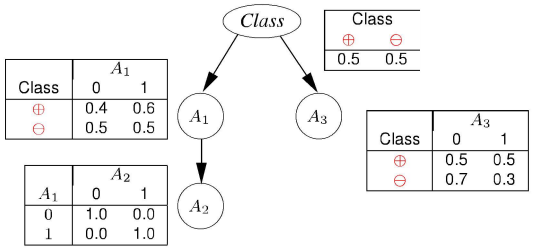
$$\prod_{i=1}^{4}(\textit{left side of } i.) > \prod_{i=1}^{4}(\textit{right side of } i.)$$

But this is false, because both products are actually equal.

A word of caution

When features don't help

Data generated by process described by Bayesian network:

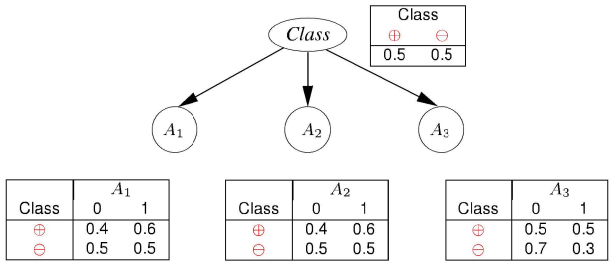


Attribute A_2 is just a duplicate of A_1. Conditional class probability for example:

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.461$$

A word of caution

The Naive Bayes model learned from data:



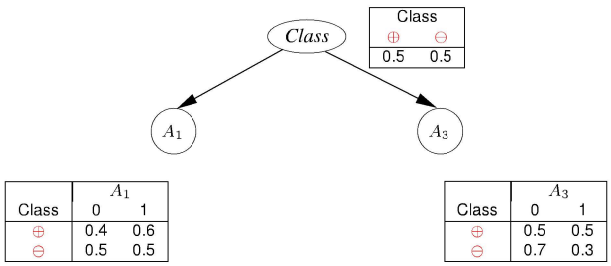
In Naive Bayes model:

$$P(\oplus \mid A_1 = 1, A_2 = 1, A_3 = 0) = 0.507$$

Intuitively: the NB model double counts the information provided by A_1, A_2. Recall our previous discussion on the use of intermediate variables!

A word of caution

The Naive Bayes model with selected features A_1 and A_3:



In this Naive Bayes model:

$$P(\oplus \mid A_1 = 1, A_3 = 0) = 0.461$$

(and all other posterior class probabilities also are the same as in the true model).

Case-based Reasoning

Idea

To predict the output feature of a new example e:

- find among the training examples the one (several) most similar to e
- predict the output for e from the known output values of the similar cases

Several names for this approach:

- Instance based learning
- Lazy learning
- Case-based reasoning

Required: **distance measure** on values of input features.

Distance Measures

Distances for numeric features

If all features $\mathbf{X}$ are numeric:

- Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_i (x_i - x'_i)^2}$
- Manhatten distance: $d(\mathbf{x}, \mathbf{x}') = \sum_i |x_i - x'_i|$

Distances for discrete features

For a single feature X with domain $\{x_1, \ldots, x_k\}$:

- Zero-One distance: $d(x_i, x_j) = 0$ if $j = i$, $d(x_i, x_j) = 1$ if $j \neq i$
- Distance matrix: specify for each pair x_i, x_j the distance $d(x_i, x_j)$ in a $k \times k$-matrix. Example:

	low	*medium*	*high*
low	0	2	5
medium	2	0	1
high	5	1	0

For a set of discrete features $\mathbf{X}$:

- Define distance d_i and weight w_i for each $X_i \in \mathbf{X}$
- Define $d(\mathbf{x}, \mathbf{x}') = \sum_i w_i d_i(x_i, x'_i)$

K-Nearest-Neighbor Classifier

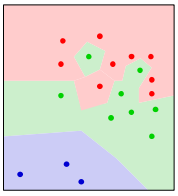
Given

- training examples $(\mathbf{x}_i, y_i)$ $(i = 1, \ldots, N)$
- a new case $\mathbf{x}$ to be classified
- a distance measure $d(\mathbf{x}, \mathbf{x}')$

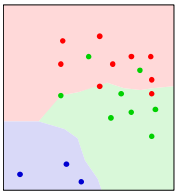
classify $\mathbf{x}$ as follows:

- find the K training examples $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_K}$ with minimial distance to $\mathbf{x}$
- predict for $\mathbf{x}$ the most frequent value in $y_{i_1}, \ldots, y_{i_K}$.

Example



1-Nearest Neighbor



5-Nearest Neighbor

- Output feature: *red,blue,green*
- Two numeric input features
- Euclidean distance
- Colored dots: training examples
- Colored regions: regions of input values that will be classified as that color

Overfitting

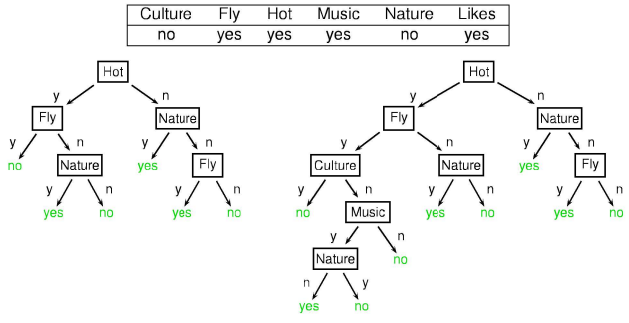
Holiday Data

Culture	Fly	Hot	Music	Nature	Likes
no	no	yes	no	no	no
no	yes	yes	no	no	no
yes	yes	yes	yes	yes	no
no	yes	yes	yes	yes	no
no	yes	yes	no	yes	no
yes	no	no	yes	yes	yes
no	no	no	no	no	no
no	no	no	yes	yes	yes
yes	yes	yes	no	no	no
yes	yes	no	yes	yes	yes
yes	yes	no	no	no	yes
yes	no	yes	no	yes	yes
no	no	no	yes	no	no
yes	no	yes	yes	no	no
yes	yes	yes	yes	no	no
yes	no	no	yes	no	no
yes	yes	yes	no	yes	no
no	no	no	no	yes	yes
no	yes	no	no	no	yes

Overfitting: Decision Trees

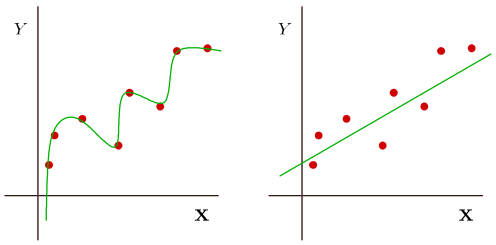
Decision tree learned from holiday data (left) and holiday data augmented with one more example

Culture	Fly	Hot	Music	Nature	Likes
no	yes	yes	yes	no	yes



Trees provide very accurate representation of training examples, but are they the best trees to predict the preferences of *future* customers?

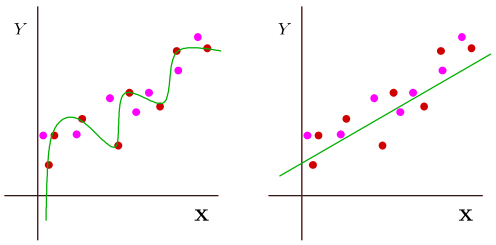
Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Right model: may have smaller SSE on future observations

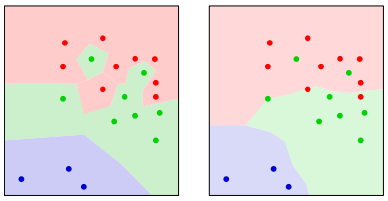
Overfitting: Neural Networks



Left model: minimizes SSE on training examples

Right model: may have smaller SSE on future observations

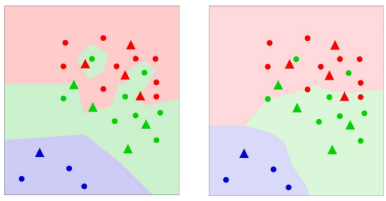
Overfitting: Nearest Neighbor



1-Nearest Neighbor 5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

Overfitting: Nearest Neighbor

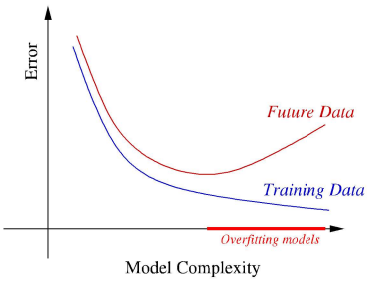


1-Nearest Neighbor 5-Nearest Neighbor

1-Nearest Neighbor correctly classifies all training examples

5-Nearest Neighbor may be better for future observations (triangles)

The General Picture



Model	Complexity	Error
Decision Tree	Depth, # Nodes	Classification error
Neural Network	# hidden nodes/layers	SSE
Nearest Neighbor	Decision regions	Classification error

A model **overfits** the training data, if a smaller error on future data would be obtained by a less complex model.

Avoiding overfitting

Reduced hypothesis space

Do not allow overly complex models:

- Naive Bayes model
- K-NN for "large" K.

Modified Search/Scoring

Do not allow to learn models that are too complex (relative to the available data):

- Decision Trees: use an early stopping criterion, e.g. stop construction when (sub-) set of training examples contains fewer than k elements (for not too small k).
- Add to evaluation measure a **penalty term** for complexity. This penalty term can have an interpretation as a **prior model probability**, or **model description length**

These techniques will usually lead to more complex models only when the data strongly supports it (especially: large number of examples).

Validation data

Basic idea

Reserve part of the training examples as a **validation set**:

- not used during model search
- used as proxy for future data in model evaluation

Train/Validation split

Simplest approach: split the available data randomly into a **training** and a **validation** set.
Typically: 50%/50% or 66%/33% split.

Example: Decision Trees

Post pruning

Use of validation set in decision tree learning:

- Build decision tree using *training* set
- For each internal node:
  - test whether accuracy on *validation* set is improved by replacing sub-tree rooted at this node by single leaf (labeled with most frequent target feature value of *training* examples in this sub-tree)
  - if yes: replace sub-tree with leaf (*prune* sub-tree)

Example: K-NN and Neural Networks

Selection of K

for $K = 1, 2, 3, \ldots$:

- measure accuracy of K-NN based on *training* examples for *validation* examples
- use K with best performance on *validation* examples
- *validation* examples can now be merged with *training* examples for predicting future cases

Example: K-NN and Neural Networks

Selection of K

for $K = 1, 2, 3, \ldots$:

- measure accuracy of K-NN based on *training* examples for *validation* examples
- use K with best performance on *validation* examples
- *validation* examples can now be merged with *training* examples for predicting future cases

Selection of Neural Network Structure

for #*hl* = 1,2,. . . , max, and #*nd* = 1,2,. . . , max:

- learn Neural Network with #*hl* hidden layers and #*nd* nodes per hidden layer using *training* examples
- evaluate SSE of learned network on *validation* examples
- select network structure with minimal SSE
- re-learn weights using merged training and validation examples

Cross-Validation

Disadvantage of training/validation splits (for small datasets):

- the examples in the validation set are partly "wasted"
- (unrepresentative) patterns in the validation set can bias the model selection

Cross-Validation

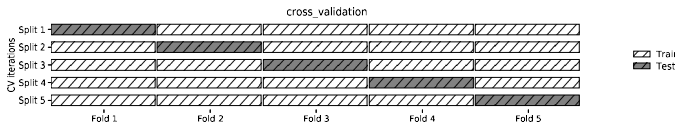
Disadvantage of training/validation splits (for small datasets):

- the examples in the validation set are partly "wasted"
- (unrepresentative) patterns in the validation set can bias the model selection

Cross Validation

The n**-fold cross-validation** approach:

- Divide the examples into n equal sized subsets, or *folds* (e.g. $n = 10$)
- for $i = 1, \ldots, n$:
  - learn model (with given "complexity setting") using folds $1, \ldots, i-1, i+1, \ldots, n$
  - evaluate using fold i
  - average the evaluation scores from the n folds
- Choose "complexity setting" that obtained highest average evaluation score
- Learn final model with chosen "complexity setting" using all available examples



Machine Intelligence

Lecture 10: Clustering

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

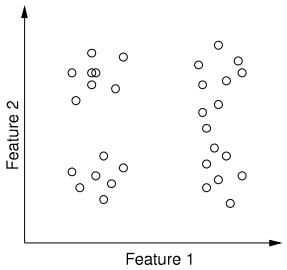
Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- Machine learning: Classification
- **Machine learning: Clustering**
- Planning
- Multi-agent systems

Clustering

Clustering: Introduction

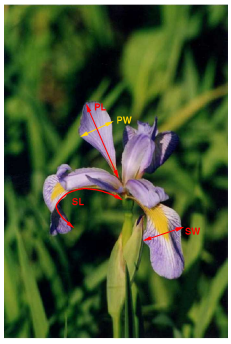
The objective of clustering is to find structure in the data.



Examples:

- Based on customer data, find groups of customers with similar profiles.
- Based on image data, find groups of images with similar motif.
- Based on article data, find groups of articles with the same topics.
- ...

Iris Data



Measurement of petal width/length and sepal width/length for 150 flowers of 3 different species of Iris.

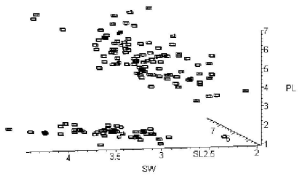
first reported in:
Fisher,R.A. "The use of multiple measurements in taxonomic problems" Annual Eugenics, 7 (1936).

Attributes				Class variable
SL	SW	PL	PW	Species
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
6.3	2.9	6.0	2.1	Virginica
6.3	2.5	4.9	1.5	Versicolor
...	...	...	...	...

Unlabeled Iris

The Iris data with class labels removed:

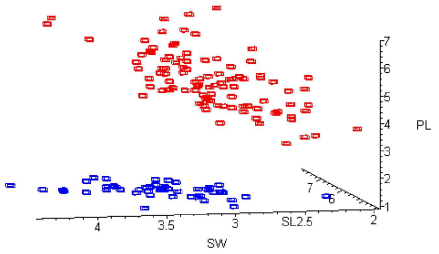
Attributes			
SL	SW	PL	PW
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
6.3	2.9	6.0	2.1
6.3	2.5	4.9	1.5
...	...	...	...



Clustered Iris

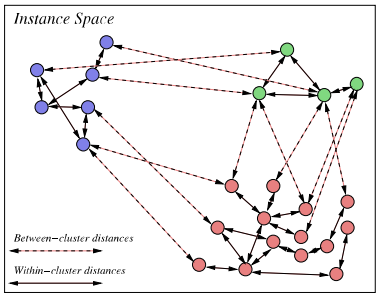
A clustering of the data $S = \mathbf{a}_1, \ldots, \mathbf{a}_N$ consists of a set $C = \{c_1, \ldots, c_k\}$ of *cluster labels*, and a *cluster assignment* $ca : S \to C$.

Clustering Iris with $C = \{blue, red\}$:



The k-means algorithm

Distance Function and Clustering



A candidate clustering (indicated by colors) of data cases in instance space. Arrows indicate between- and within-cluster distances (selected).

General goal: find clustering with

- large between-cluster variation (sum of between-cluster distances)
- small within-cluster variation (sum of within-cluster distances)

The k-means algorithm

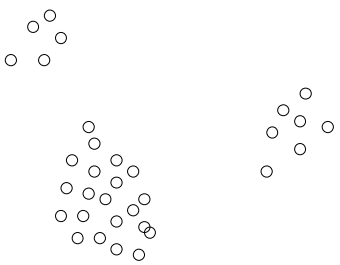
We consider the scenario, where

- the number k of clusters is known.
- we have a distance measure $d(\mathbf{x}_i, \mathbf{x}_j)$ between pairs of data points (feature vectors).
- we can calculate a centroid for a collection of data points $S = \{\mathbf{x}_1, \ldots \mathbf{x}_n\}$.

Initialize: *randomly pick k data points as initial cluster centers* $\mathbf{c} = c_1, \ldots, c_k$ from S
`repeat`
    Form k clusters by assigning each point in S to its closest centroid.
    Recompute the centroid for each cluster.
`until` Centroids do not change

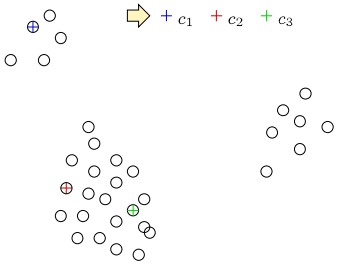
The k-means algorithm: Example

$k = 3$:



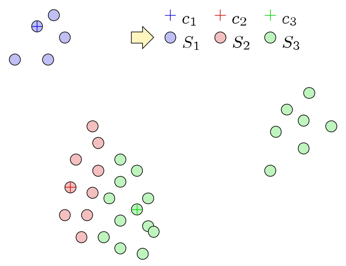
The k-means algorithm: Example

$k = 3$:



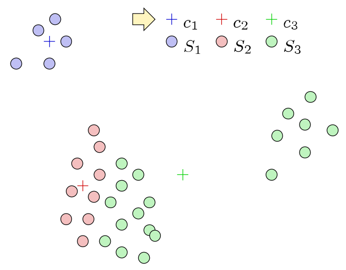
The k-means algorithm: Example

$k = 3$:



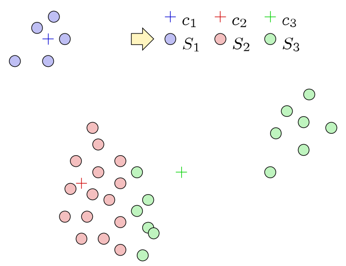
The k-means algorithm: Example

$k = 3$:



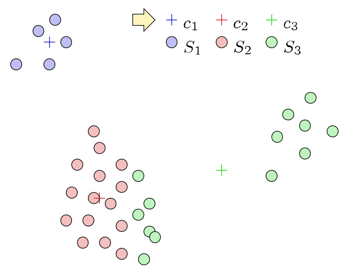
The k-means algorithm: Example

$k = 3$:



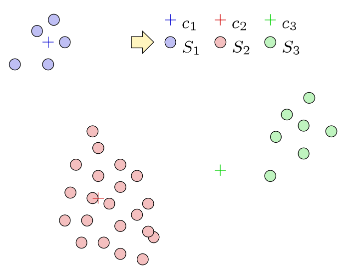
The k-means algorithm: Example

$k = 3$:



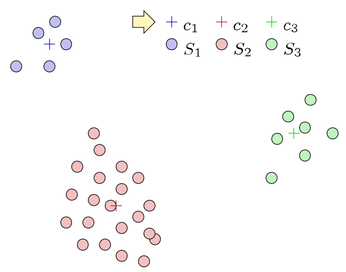
The k-means algorithm: Example

$k = 3$:



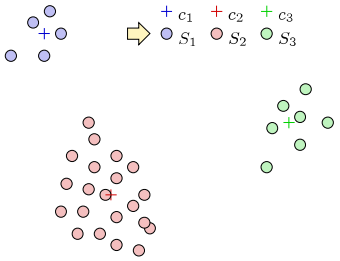
The k-means algorithm: Example

$k = 3$:



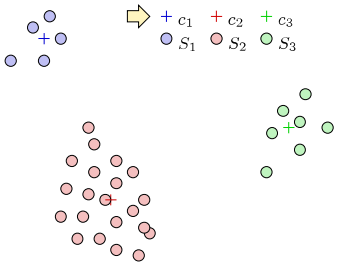
The k-means algorithm: Example

$k = 3$:



The k-means algorithm: Example

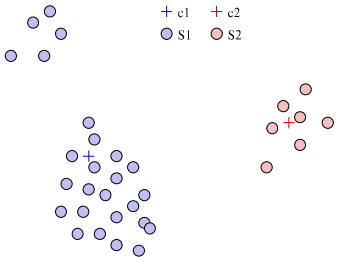
$k = 3$:



MI Autumn 2019 The k-means algorithm 9

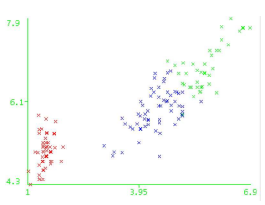
Different k

Result for clustering the same data with $k = 2$:

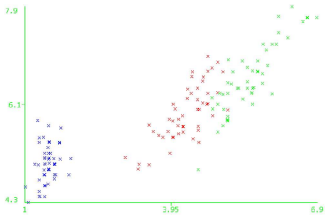


Result can depend on choice of initial cluster centers!

Clustering Iris



Iris 3-means clustered



Iris true classes

The k-means algorithm: Background

k-means as an optimization problem

Assume that we use the Euclidean distance d as proximity measure and that the quality of the clustering is measured by the sum of squared errors:

$$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{c}_i, \mathbf{x})^2,$$

where:

- $\mathbf{c}_i$ is the i'th centroid
- $C_i \subseteq S$ is the points closets to $\mathbf{c}_i$ according to d.

In principle ...

We can minimize the SSE by looking at all possible partitionings $\rightsquigarrow$ not feasible!

The k-means algorithm: Background

k-means as an optimization problem

Assume that we use the Euclidean distance d as proximity measure and that the quality of the clustering is measured by the sum of squared errors:

$$SSE = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{c}_i, \mathbf{x})^2,$$

where:

- $\mathbf{c}_i$ is the i'th centroid
- $C_i \subseteq S$ is the points closets to $\mathbf{c}_i$ according to d.

In principle ...

We can minimize the SSE by looking at all possible partitionings $\rightsquigarrow$ not feasible!

Instead, k-means

The centroid that minimizes the SSE is the *mean* of the data-points in that cluster:

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Local optimum found by alternating between cluster assignments and centroid estimation.

MI Autumn 2019 The k-means algorithm 12

The k-means algorithm: Background

Convergence

The k-means algorithm is guaranteed to converge

- Each step reduces the sum of squared errors.
- There is only a finite number of cluster assignments.

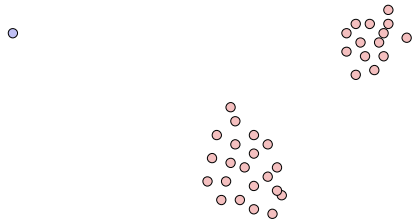
There is no guarantee of reaching the global optimum:

- Improve by running with multiple random restarts.

Some practical issues

Outliers

The result of partitional clustering can be skewed by outliers. Example with $k = 2$:

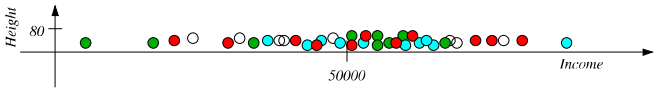


$\rightsquigarrow$ useful preprocessing: outlier detection and elimination.

Different Measuring Scales

Instances defined by attributes

$A_1 =$ *height in inches* and $A_2 =$ *annual income in \$* :



- all distance functions for continuous attributes dominated by *income* values
- $\rightsquigarrow$ may need to *rescale* or *normalize* continuous attributes

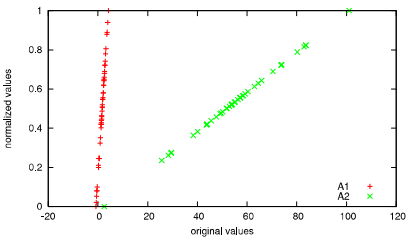
Min-Max Normalization

Min-Max Normalization

replace A_i with

$$\frac{A_i - \min(A_i)}{\max(A_i) - \min(A_i)}$$

$(\min(A_i), \max(A_i)$ are min/max values of A_i appearing in the data)

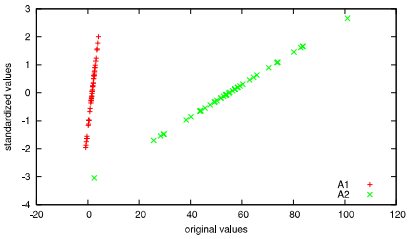


Z-Score Standardization

Z-score Standardization

replace A_i with

$$\frac{A_i - \textit{mean}(A_i)}{\textit{standard deviation}(A_i)}$$



where

$$\textit{mean}(A_i) = \frac{1}{n}\sum_{j=1}^{n} a_{j,i}$$

$$\textit{standard deviation}(A_i) = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(a_{j,i} - \textit{mean}(A_i))^2}$$

Soft clustering

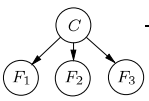
Soft clustering

The k-means algorithm generates a *hard* clustering: each example is assigned to a single cluster.

Alternatively: In *soft* clustering each example is assigned to a cluster with a certain probability.

The naive Bayes model for clustering

Model



Data

F_1	F_2	F_3	C
t	t	t	?
t	f	t	?
t	f	f	?
f	f	t	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- C is the hidden cluster variable.
- F_1, F_2, and F_3 are the feature variables.

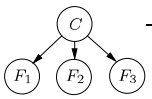
Soft clustering

The k-means algorithm generates a *hard* clustering: each example is assigned to a single cluster.

Alternatively: In *soft* clustering each example is assigned to a cluster with a certain probability.

The naive Bayes model for clustering

Model



Data

F_1	F_2	F_3	C
t	t	t	?
t	f	t	?
t	f	f	?
f	f	t	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- C is the hidden cluster variable.
- F_1, F_2, and F_3 are the feature variables.

Procedure

- Set the number of clusters, i.e., the states of C
- Learn the probabilities of the model:
  - $P(C)$, $P(F_1|C)$, $P(F_2|C)$, and $P(F_3|C)$
- Use the learned probabilities to cluster the (future) instances.

The EM algorithm

When learning the probability distributions of the model, the variable C is hidden

- $\rightsquigarrow$ we *cannot* directly estimate the probabilities using frequency counts

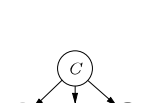
Instead we employ the *Expectation-maximization algorithm*

The EM-algorithm

The main idea:

- Use hypothetical completions of the data using the current probability estimates.
- Infer the maximum likelihood probabilities for the model based on completed data set.

EM for soft clustering: an example



Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

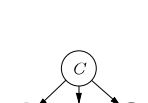
Also $P_0(F_2|C)$ and $P_0(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	
t	f	t	
t	f	f	
f	f	t	

Maximization

EM for soft clustering: an example



Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2|C)$ and $P_0(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

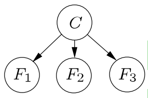
F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	
t	f	f	
f	f	t	

Maximization

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

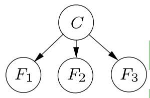
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	
f	f	t	

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

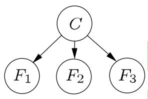
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

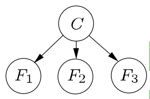
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_0(C) = (0.6, 0.4)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

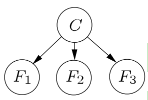
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

$$P_1(C) = \frac{1}{4} \sum_{F_1, F_2, F_3} A(F_1, F_2, F_3, C) = \frac{1}{4}(0.84 + 0.69 + 0.5 + 0.5, 0.16 + 0.31 + 0.5 + 0.5)$$
$$= (0.63, 0.37)$$

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

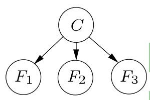
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

$$P_1(C) = \frac{1}{4} \sum_{F_1, F_2, F_3} A(F_1, F_2, F_3, C) = \frac{1}{4}(0.84 + 0.69 + 0.5 + 0.5, 0.16 + 0.31 + 0.5 + 0.5)$$
$$= (0.63, 0.37)$$

EM for soft clustering: an example



Expectation

Maximization

Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_0(F_1 \mid C)$	$C=1$	$C=2$
$F_1 = t$	0.6	0.4
$F_1 = f$	0.4	0.6

Also $P_0(F_2 \mid C)$ and $P_0(F_3 \mid C)$

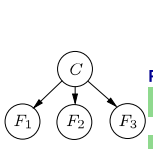
Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C \mid F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

$$P_1(F_1 \mid C) = \frac{\sum_{F_2, F_3} A(F_1, F_2, F_3, C)}{\sum_{F_1, F_2, F_3} A(F_1, F_2, F_3, C)} = \frac{\begin{pmatrix} 0.84 + 0.69 + 0.5 + 0 & 0.16 + 0.31 + 0.5 + 0 \\ 0 + 0 + 0 + 0.5 & 0 + 0 + 0 + 0.5 \end{pmatrix}}{(2.53, 1.47)}$$
$$= \begin{pmatrix} 0.8 & 0.65 \\ 0.2 & 0.35 \end{pmatrix}$$

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_0(F_2|C)$ and $P_0(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

Data:

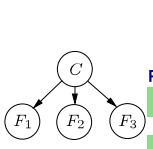
F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Maximization

$$P_1(F_1|C) = \frac{\sum_{F_2,F_3} A(F_1,F_2,F_3,C)}{\sum_{F_1,F_2,F_3} A(F_1,F_2,F_3,C)} = \frac{\begin{pmatrix} 0.84+0.69+0.5+0 & 0.16+0.31+0.5+0 \\ 0+0+0+0.5 & 0+0+0+0.5) \end{pmatrix}}{(2.53, 1.47)}$$

$$= \begin{pmatrix} 0.8 & 0.65 \\ 0.2 & 0.35 \end{pmatrix}$$

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_0(F_2|C)$ and $P_0(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

Data:

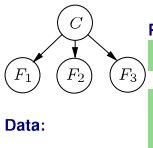
F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Maximization

$$P_1(F_2|C) = \ldots\ldots = \begin{pmatrix} 0.33 & 0.11 \\ 0.67 & 0.89 \end{pmatrix}$$

$$P_1(F_3|C) = \ldots\ldots = \begin{pmatrix} 0.80 & 066 \\ 0.20 & 0.34 \end{pmatrix}$$

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_1(F_2|C)$ and $P_1(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.84, 0.16)$
t	f	t	$(0.69, 0.31)$
t	f	f	$(0.5, 0.5)$
f	f	t	$(0.5, 0.5)$

Maximization

Data:

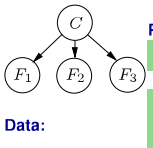
F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Maximization

$$P_1(F_2|C) = \ldots\ldots = \begin{pmatrix} 0.33 & 0.11 \\ 0.67 & 0.89 \end{pmatrix}$$

$$P_1(F_3|C) = \ldots\ldots = \begin{pmatrix} 0.80 & 066 \\ 0.20 & 0.34 \end{pmatrix}$$

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_1(F_2|C)$ and $P_1(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.88, 0.12)$
t	f	t	
t	f	f	
f	f	t	

Maximization

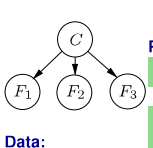
Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_1(F_2|C)$ and $P_1(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.88, 0.12)$
t	f	t	$(0.66, 0.34)$
t	f	f	
f	f	t	

Maximization

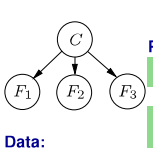
Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

- Fractional counts are being calculated by probability updating.

EM for soft clustering: an example



Expectation

Probability tables:

$P_1(C) = (0.63, 0.37)$

$P_1(F_1\|C)$	$C=1$	$C=2$
$F_1 = t$	0.8	0.65
$F_1 = f$	0.2	0.35

Also $P_1(F_2|C)$ and $P_1(F_3|C)$

Count table $A(F_1, F_2, F_3, C)$:

F_1	F_2	F_3	$P(C\|F_1, F_2, F_3)$
t	t	t	$(0.88, 0.12)$
t	f	t	$(0.66, 0.34)$
t	f	f	$(0.48, 0.52)$
f	f	t	

Maximization

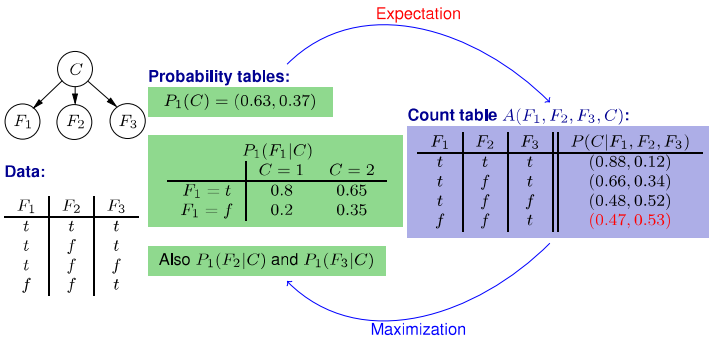
Data:

F_1	F_2	F_3
t	t	t
t	f	t
t	f	f
f	f	t

Expectation

- Fractional counts are being calculated by probability updating.

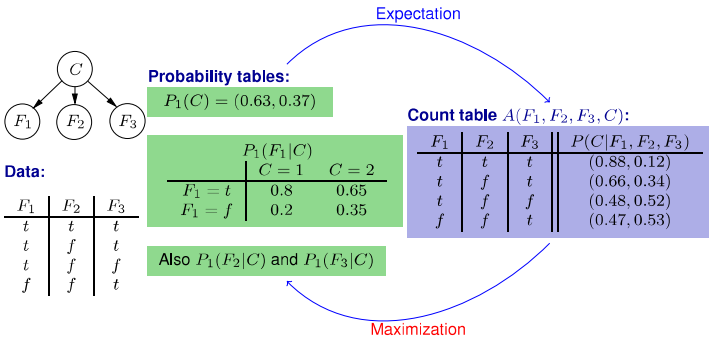
EM for soft clustering: an example



Expectation

- Fractional counts are being calculated by probability updating.

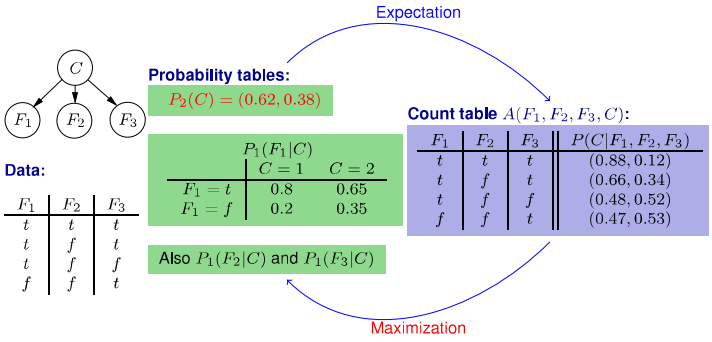
EM for soft clustering: an example



Maximization

$$P_2(C) = \frac{1}{4}\sum_{F_1,F_2,F_3} A(F_1,F_2,F_3,C) = \frac{1}{4}(0.88+0.66+0.48+0.47, 0.12+0.34+0.52+0.53)$$
$$= (0.62, 0.38)$$

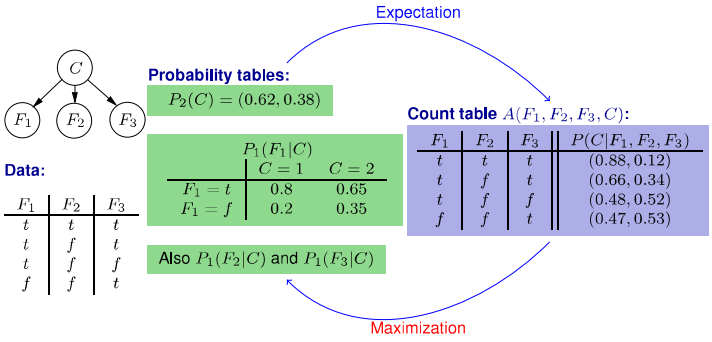
EM for soft clustering: an example



Maximization

$$P_2(C) = \frac{1}{4}\sum_{F_1,F_2,F_3} A(F_1,F_2,F_3,C) = \frac{1}{4}(0.88+0.66+0.48+0.47, 0.12+0.34+0.52+0.53)$$
$$= (0.62, 0.38)$$

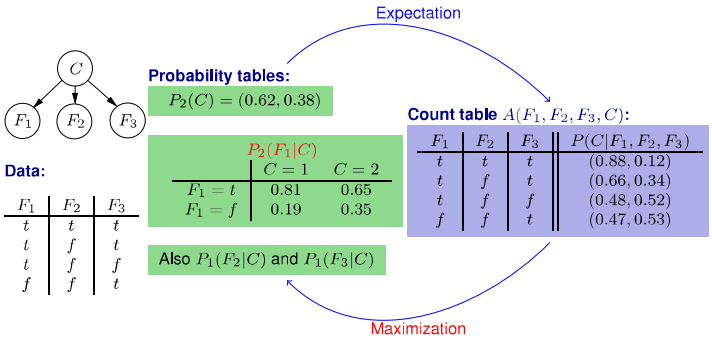
EM for soft clustering: an example



Maximization

$$P_2(F_1|C) = \frac{\sum_{F_2,F_3} A(F_1,F_2,F_3,C)}{\sum_{F_1,F_2,F_3} A(F_1,F_2,F_3,C)} = \frac{\begin{pmatrix} 0.88+0.66+0.48+0 & 0.12+0.34+0.52+0 \\ 0+0+0+0.47 & 0+0+0+0.53) \end{pmatrix}}{(2.49, 1.51)}$$
$$= \begin{pmatrix} 0.81 & 0.65 \\ 0.19 & 0.35 \end{pmatrix}$$

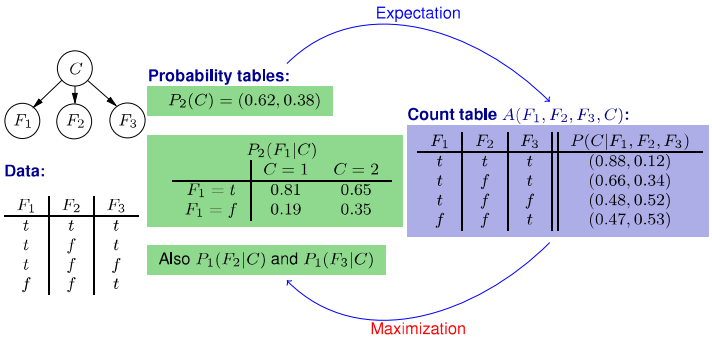
EM for soft clustering: an example



Maximization

$$P_2(F_1|C) = \frac{\sum_{F_2,F_3} A(F_1,F_2,F_3,C)}{\sum_{F_1,F_2,F_3} A(F_1,F_2,F_3,C)} = \frac{\begin{pmatrix} 0.88+0.66+0.48+0 & 0.12+0.34+0.52+0 \\ 0+0+0+0.47 & 0+0+0+0.53) \end{pmatrix}}{(2.49, 1.51)}$$
$$= \begin{pmatrix} 0.81 & 0.65 \\ 0.19 & 0.35 \end{pmatrix}$$

EM for soft clustering: an example



Maximization

. . . and so we continue until a termination criterion is reached.

The EM algorithm

Correctness

- The sequence of probability estimates generated by the EM algorithm converges to a local maximum (in rare cases: a saddle point) of the marginal likelihood given the data.
- To avoid sub-optimal local maxima: run EM several times with different starting points.

The EM algorithm

Correctness

- The sequence of probability estimates generated by the EM algorithm converges to a local maximum (in rare cases: a saddle point) of the marginal likelihood given the data.
- To avoid sub-optimal local maxima: run EM several times with different starting points.

Notes

- Any permutation of the cluster labels of a local maximum will also be a local maximum.
- Rather than keeping track of a full count table, it suffices to store counts for the variable families, $fa(X) = \{X\} \cup pa(X)$. Only one pass through the data is necessary.
- Clustering an existing or new instance $\mathbf{x}$ amounts to calculating $P(C|\mathbf{x})$.

Cluster evaluation

Cluster evaluation

A clustering algorithm applied to a dataset will return a clustering - even if there is no meaningful structure in the data!

Question: Do the clusters actually correspond to meaningful groups of data instances?
Question: Are all the clusters relevant, or are there some real and some meaningless clusters?

Supervised vs. Unsupervised Evaluation

Unsupervised

- Uses only the data as given to the clustering algorithm, and the resulting clustering
- The realistic scenario
- Can be guided by considering changes in evaluation score.

Supervised

- Uses external information, e.g. a true class label as the "gold standard" for actual groups in the data
- Not representative for actual clustering applications
- Can be useful to evaluate *clustering algorithms*
- Caveat: no guarantee that the class labels actually describe the most natural or relevant groups in the data

Machine Intelligence

Lecture 11: Planning

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- Machine learning: classification
- Machine learning: Clustering
- **Planning**
- Multi-agent systems

Utility

A small quiz

Which of the following two lotteries would you prefer?:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.

A small quiz

Which of the following two lotteries would you prefer?:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.

What about these two?:

- Lottery $C = 0.11[\$1\text{mill.}] + 0.89[\$0]$,
- Lottery $D = 0.1[\$5\text{mill.}] + 0.9[\$0]$.

Preferences and Lotteries

Values with certainty

What do you prefer

- $100 or $1000000 ?
- A 4 or 10 grade in the exam?

Lotteries

A **lottery** is a probability distribution over **outcomes**. E.g.

$$[0.4 : \$100,\ 0.6 : -\$20]$$

means: you win $100 with probability 0.4, and loose $ 20 with probability 0.6.

$$[0.3 : 00,\ 0.5 : 7,\ 0.2 : 10]$$

means: with probability 0.3 you get a 00, with probability 0.5 a 7, and with probability 0.2 a 10.

Preferences and Lotteries

Values with certainty

What do you prefer

- $100 or $1000000 ?
- A 4 or 10 grade in the exam?

Lotteries

A **lottery** is a probability distribution over **outcomes**. E.g.

$$[0.4 : \$100,\ 0.6 : -\$20]$$

means: you win $100 with probability 0.4, and loose $ 20 with probability 0.6.

$$[0.3 : 00,\ 0.5 : 7,\ 0.2 : 10]$$

means: with probability 0.3 you get a 00, with probability 0.5 a 7, and with probability 0.2 a 10.

Values with uncertainty

What do you prefer

- [1:$1000000] or [0.5:$ 0, 0.5:$2100000] ?

Preferences and Lotteries

Values with certainty

What do you prefer
- $100 or $1000000 ?
- A 4 or 10 grade in the exam?

Lotteries

A **lottery** is a probability distribution over **outcomes**. E.g.

$$[0.4 : \$100,\ 0.6 : -\$20]$$

means: you win $100 with probability 0.4, and loose $ 20 with probability 0.6.

$$[0.3 : 00,\ 0.5 : 7,\ 0.2 : 10]$$

means: with probability 0.3 you get a 00, with probability 0.5 a 7, and with probability 0.2 a 10.

Values with uncertainty

What do you prefer
- [1:$1000000] or [0.5:$ 0, 0.5:$2100000] ?
- [0.4:00, 0.1:7, 0.5:10] or [0.1:00, 0.8:7, 0.1:10]?

Utilities

Typically:

[1:$1000000]is preferred over [0.5:$ 0, 0.5:$2100000]

Thus: preferences between lotteries with "money outcomes" are not always determined by **expected monetary value**.

Utilities

Typically:

[1:$1000000]is preferred over [0.5:$ 0, 0.5:$2100000]

Thus: preferences between lotteries with "money outcomes" are not always determined by **expected monetary value**.

Preferences from utilities

The following is a classical result:

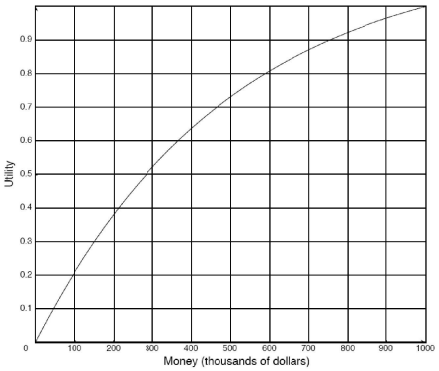
> *If preferences between lotteries obey a certain set of plausible rules, then there exists an assignment of real numbers (***utilities***) to all outcomes, such that one lottery is preferred over another if and only if it has a higher* **expected utility**.

Example

				Expected Utility
Outcomes:	00	7	10	
Utilities:	-5	5	10	
Lottery 1:	0.4	0.1	0.5	3.5
Lottery 2:	0.1	0.8	0.1	4.5

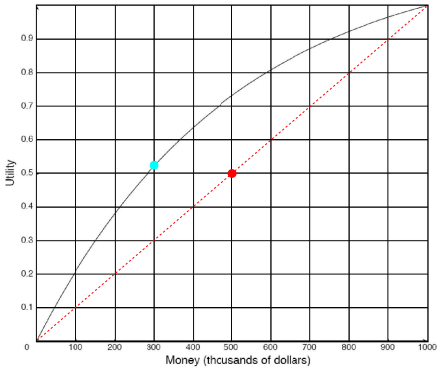
Utility of Money

Also "money outcomes" have a utility. Utility function of a **risk-averse** agent:



Utility of Money

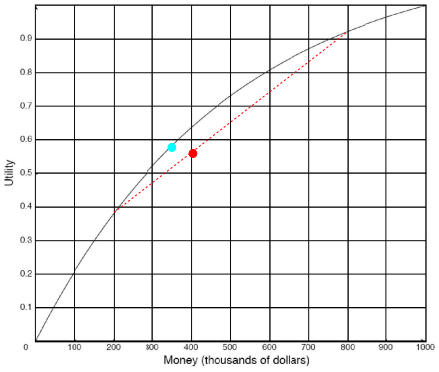
Also "money outcomes" have a utility. Utility function of a **risk-averse** agent:



- Sure $ 300k preferred over [0.5:$ 0, 0.5:$ 1000k]

Utility of Money

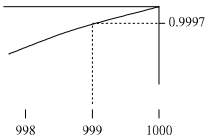
Also "money outcomes" have a utility. Utility function of a **risk-averse** agent:



- Sure $ 300k preferred over [0.5:$ 0, 0.5:$ 1000k]
- Sure $ 350k preferred over [2/3:$ 200,1/3:$ 800k]

Digression: Insurance business

Assume Utility($ 999k)=0.9997:



Then agent is indifferent between lotteries

[1:$ 999k] and [0.0003:0, 0.9997:$ 1000k]

and prefers

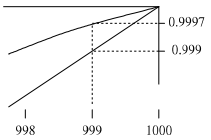
[1:$ 999.4k] over [0.0003:0, 0.9997:$ 1000k]

Interpretation:

- right lottery: 0.03 risk of loosing a $ 1000k property
- left lottery: buying insurance against that risk for $ 600.

Digression: Insurance business

Assume Utility($ 999k)=0.9997:



Then agent is indifferent between lotteries

[1:$ 999k] and [0.0003:0, 0.9997:$ 1000k]

and prefers

[1:$ 999.4k] over [0.0003:0, 0.9997:$ 1000k]

Interpretation:

- right lottery: 0.03 risk of loosing a $ 1000k property
- left lottery: buying insurance against that risk for $ 600.

The *insurance company* prefers

[0.0003:0, 0.9997:$ 1000k] over [1:$ 999.4k]

$\rightsquigarrow$ the insurance company has a different utility function (near linear).

Do we maximize expected utility?

Recall:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.
- Lottery $C = 0.11[\$1\text{mill.}] + 0.89[\$0]$,
- Lottery $D = 0.1[\$5\text{mill.}] + 0.9[\$0]$.

Do we maximize expected utility?

Recall:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.
- Lottery $C = 0.11[\$1\text{mill.}] + 0.89[\$0]$,
- Lottery $D = 0.1[\$5\text{mill.}] + 0.9[\$0]$.

A preferred over $B \Rightarrow$

$$\begin{aligned} & U(\$1m) > 0.1U(\$5m) + 0.89U(\$1m) + 0.01U(\$0) \\ \Rightarrow \quad & 0.11U(\$1m) > 0.1U(\$5m) + 0.01U(\$0) \end{aligned}$$

Do we maximize expected utility?

Recall:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.
- Lottery $C = 0.11[\$1\text{mill.}] + 0.89[\$0]$,
- Lottery $D = 0.1[\$5\text{mill.}] + 0.9[\$0]$.

A preferred over $B \Rightarrow$

$$\begin{aligned} & U(\$1m) > 0.1U(\$5m) + 0.89U(\$1m) + 0.01U(\$0) \\ \Rightarrow \quad & 0.11U(\$1m) > 0.1U(\$5m) + 0.01U(\$0) \end{aligned}$$

D preferred over $C \Rightarrow$

$$\begin{aligned} & 0.1U(\$5m) + 0.9U(\$0) > 0.11U(\$1m) + 0.89U(\$0) \\ \Rightarrow \quad & 0.1U(\$5m) + 0.01U(\$0) > 0.11U(\$1m) \end{aligned}$$

Do we maximize expected utility?

Recall:

- Lottery $A = [\$1\text{mill.}]$,
- Lottery $B = 0.1[\$5\text{mill.}] + 0.89[\$1\text{mill.}] + 0.01[\$0]$.
- Lottery $C = 0.11[\$1\text{mill.}] + 0.89[\$0]$,
- Lottery $D = 0.1[\$5\text{mill.}] + 0.9[\$0]$.

A preferred over $B \Rightarrow$

$$\begin{aligned} & U(\$1m) > 0.1U(\$5m) + 0.89U(\$1m) + 0.01U(\$0) \\ \Rightarrow \quad & 0.11U(\$1m) > 0.1U(\$5m) + 0.01U(\$0) \end{aligned}$$

D preferred over $C \Rightarrow$

$$\begin{aligned} & 0.1U(\$5m) + 0.9U(\$0) > 0.11U(\$1m) + 0.89U(\$0) \\ \Rightarrow \quad & 0.1U(\$5m) + 0.01U(\$0) > 0.11U(\$1m) \end{aligned}$$

Contradiction! Explanations:

- People do not maximize expected utility
- The utility of $ 0 also depends on in which lottery $ 0 were "won"

Factored Utility

So far: outcomes seen as unstructured states. When states are described by features, then overall utility often a combination of utility factors derived from different features.

Example

Two component utility function:

RHC	*SWC*	Utility
rhc	*swc*	5
rhc	$\overline{swc}$	3
$\overline{rhc}$	*swc*	0
$\overline{rhc}$	$\overline{swc}$	5

RLoc	*MW*	Utility
cs	*mw*	0
cs	$\overline{mw}$	2
off	*mw*	0
off	$\overline{mw}$	2
lab	*mw*	0
lab	$\overline{mw}$	2
mr	*mw*	5
mr	$\overline{mw}$	2

Utility of full outcome (state) is sum of utility factors:

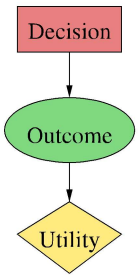
$$U(\langle off, rhc, \overline{swc}, \overline{mw}, rhm\rangle) = 3 + 2 = 5$$

Assumption: the utility contribution from one factor is independent of the values of other factors. E.g.: (*rhc*,*swc*) should perhaps be worth less than 5 when at the same time (*mr*,*mw*), because mail needs to be delivered first (the two utility factors are **substitutes**).

Single-Stage Decision Networks

Decisions and Lotteries

Simple decisions can be seen as choices over lotteries. Graphical representation of study example:



prepare_some,*prepare_all*

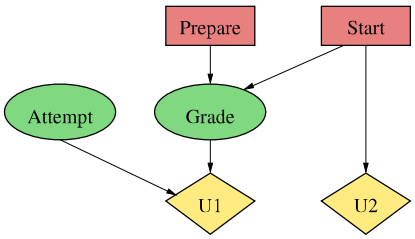
Decision	*Outcome* 00	7	10
prepare_some	0.4	0.1	0.5
prepare_all	0.1	0.8	0.1

Outcome	Utility
00	-5
7	5
10	10

Feature Based Models

Decisions, outcomes and utilities can all be composed of features or factors:

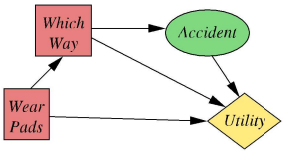
Two components of decision: prepare *some*/*all*, start preparations *sooner*/*later*
Two utility factors: utility of grade, and utility (cost) of preparation time
Outcome composed of *Grade*, *Attempt*



Graph represents:

- One utility factor depends on *Attempt* and *Grade*, another only on *Start*
- Both the *Prepare* and *Start* decision influence the probabilities for *Grade*.

Robot Example



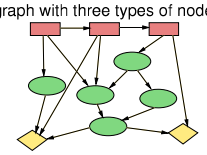
WhichWay	*Accident* true	false
short	0.2	0.8
long	0.01	0.99

WearPads	*WhichWay*	*Accident*	Utility
true	short	true	35
true	short	false	95
true	long	true	30
true	long	false	75
false	short	true	3
false	short	false	100
false	long	true	0
false	long	false	80

SSDN in General

Structure

A **Single-Stage Decision Network** is a directed acyclic graph with three types of nodes:



- Decision Nodes $\mathbf{D}$
- Chance Nodes $\mathbf{C}$
- Utility Nodes $\mathbf{U}$

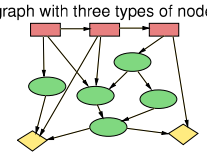
The graph must have the following structure:

- All decision nodes are connected in one linear sequence (representing the order in which the different sub-decisions are taken)
- The only parent of a decision node is its predecessor in the order
- Chance Nodes can have Decision Node and Chance Node parents
- Utility Nodes can have Decision Node and Chance Node parents

SSDN in General

Structure

A **Single-Stage Decision Network** is a directed acyclic graph with three types of nodes:



- Decision Nodes $\mathbf{D}$
- Chance Nodes $\mathbf{C}$
- Utility Nodes $\mathbf{U}$

The graph must have the following structure:

- All decision nodes are connected in one linear sequence (representing the order in which the different sub-decisions are taken)
- The only parent of a decision node is its predecessor in the order
- Chance Nodes can have Decision Node and Chance Node parents
- Utility Nodes can have Decision Node and Chance Node parents

Tables

- No table is associated with decision nodes (only the list of available decisions)
- A chance node is labeled with a conditional probability table that specifies for each value assignment to its parents (decision and chance nodes) a probability distribution over the domain of the chance node.
- A utility node is labeled with a utility table that specifies for each value assignment to its parents (decision and chance nodes) a utility value.

SSDN Semantics and solutions

SSDN Semantics

A **possible world** ω is an assignment of values to all decision and chance variables.
An SSDN defines:

- For each assignment $\mathbf{D} = \mathbf{d}$ of values to the decision nodes a probability distribution

$$P(\omega \mid \mathbf{D} = \mathbf{d})$$

over possible worlds.

- For each possible world ω a utility value

$$U(\omega)$$

SSDN Semantics and solutions

SSDN Semantics

A **possible world** ω is an assignment of values to all decision and chance variables.
An SSDN defines:

- For each assignment $\mathbf{D} = \mathbf{d}$ of values to the decision nodes a probability distribution

$$P(\omega \mid \mathbf{D} = \mathbf{d})$$

over possible worlds.

- For each possible world ω a utility value

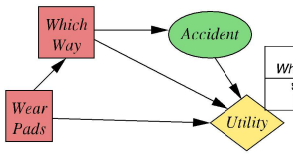
$$U(\omega)$$

Solving an SSDN

To **solve** a single-stage decision network (or problem) means to find the decisions $\mathbf{d}$ that maximize the expected utility

$$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d}) = \sum_{\omega} U(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$$

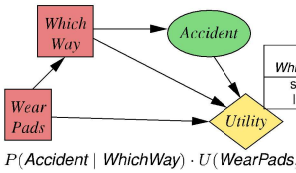
Robot Example



	Accident	
WhichWay	true	false
short	0.2	0.8
long	0.01	0.99

WearPads	*WhichWay*	*Accident*	Utility
true	short	true	35
true	short	false	95
true	long	true	30
true	long	false	75
false	short	true	3
false	short	false	100
false	long	true	0
false	long	false	80

Robot Example



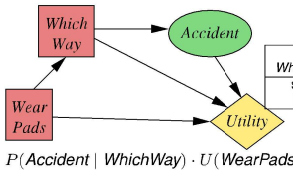
	Accident	
WhichWay	true	false
short	0.2	0.8
long	0.01	0.99

WearPads	*WhichWay*	*Accident*	Utility
true	short	true	35
true	short	false	95
true	long	true	30
true	long	false	75
false	short	true	3
false	short	false	100
false	long	true	0
false	long	false	80

$P(\mathit{Accident} \mid \mathit{WhichWay}) \cdot U(\mathit{WearPads}, \mathit{WhichWay}, \mathit{Accident}) =$

WearPads	*WhichWay*	*Accident*	$P(\omega \mid \mathbf{D} = \mathbf{d}) \cdot U(\omega)$
true	short	true	0.2· 35
true	short	false	0.8· 95
true	long	true	0.01· 30
true	long	false	0.99· 75
false	short	true	0.2· 3
false	short	false	0.8· 100
false	long	true	0.01· 0
false	long	false	0.99· 80

Robot Example



	Accident	
WhichWay	true	false
short	0.2	0.8
long	0.01	0.99

WearPads	*WhichWay*	*Accident*	Utility
true	short	true	35
true	short	false	95
true	long	true	30
true	long	false	75
false	short	true	3
false	short	false	100
false	long	true	0
false	long	false	80

$P(\mathit{Accident} \mid \mathit{WhichWay}) \cdot U(\mathit{WearPads}, \mathit{WhichWay}, \mathit{Accident}) =$

WearPads	*WhichWay*	*Accident*	$P(\omega \mid \mathbf{D} = \mathbf{d}) \cdot U(\omega)$
true	short	true	0.2· 35
true	short	false	0.8· 95
true	long	true	0.01· 30
true	long	false	0.99· 75
false	short	true	0.2· 3
false	short	false	0.8· 100
false	long	true	0.01· 0
false	long	false	0.99· 80

$\sum_{\mathit{Accident}} P(\mathit{Accident} \mid \mathit{WhichWay}) \cdot U(\mathit{WearPads}, \mathit{WhichWay}, \mathit{Accident}) =$

WearPads	*WhichWay*	$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d})$
true	short	0.2· 35 + 0.8· 95 = 83
true	long	0.01· 30 + 0.99· 75 = 74.55
false	short	0.2· 3 + 0.8· 100= 80.6
false	long	0.01· 0 + 0.99· 80= 79.2

Solving SSDNs in general

To **solve** a single-stage decision network (or problem) means to find the decisions $\mathbf{d}$ that maximize the expected utility

$$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d}) = \sum_{\omega} U(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$$

Solving SSDNs in general

To **solve** a single-stage decision network (or problem) means to find the decisions $\mathbf{d}$ that maximize the expected utility

$$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d}) = \sum_{\omega} U(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$$

Solving by (Chance) variable elimination

Let $P(C_1 \mid \mathit{par}(C_1)), \ldots, P(C_n \mid \mathit{par}(C_n))$ be the conditional probability tables associated with the chance nodes, and $U_1(\mathit{par}(U_1)), \ldots, U_k(\mathit{par}(U_k))$ the utility tables of the utility nodes. Then

- For each $j = 1, \ldots, k$:

$$\prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

is a table in the variables $\mathbf{D}, \mathbf{C}$. Each row in the table corresponds to a possible world $\omega \sim (\mathbf{D} = \mathbf{d}, \mathbf{C} = \mathbf{c})$. The entry for that row is equal to $U_j(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$.

Solving SSDNs in general

To **solve** a single-stage decision network (or problem) means to find the decisions $\mathbf{d}$ that maximize the expected utility

$$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d}) = \sum_{\omega} U(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$$

Solving by (Chance) variable elimination

Let $P(C_1 \mid \mathit{par}(C_1)), \ldots, P(C_n \mid \mathit{par}(C_n))$ be the conditional probability tables associated with the chance nodes, and $U_1(\mathit{par}(U_1)), \ldots, U_k(\mathit{par}(U_k))$ the utility tables of the utility nodes. Then

- For each $j = 1, \ldots, k$:

$$\prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

is a table in the variables $\mathbf{D}, \mathbf{C}$. Each row in the table corresponds to a possible world $\omega \sim (\mathbf{D} = \mathbf{d}, \mathbf{C} = \mathbf{c})$. The entry for that row is equal to $U_j(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$.

- The expected utility factor U_j of decision $\mathbf{D} = \mathbf{d}$ is obtained by summing the chance variables out of that table:

$$\mathcal{E}(U_j \mid \mathbf{D} = \mathbf{d}) = \sum_{\mathbf{C}} \prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

Solving SSDNs in general

To **solve** a single-stage decision network (or problem) means to find the decisions $\mathbf{d}$ that maximize the expected utility

$$\mathcal{E}(U \mid \mathbf{D} = \mathbf{d}) = \sum_{\omega} U(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$$

Solving by (Chance) variable elimination

Let $P(C_1 \mid \mathit{par}(C_1)), \ldots, P(C_n \mid \mathit{par}(C_n))$ be the conditional probability tables associated with the chance nodes, and $U_1(\mathit{par}(U_1)), \ldots, U_k(\mathit{par}(U_k))$ the utility tables of the utility nodes. Then

- For each $j = 1, \ldots, k$:

$$\prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

is a table in the variables $\mathbf{D}, \mathbf{C}$. Each row in the table corresponds to a possible world $\omega \sim (\mathbf{D} = \mathbf{d}, \mathbf{C} = \mathbf{c})$. The entry for that row is equal to $U_j(\omega) P(\omega \mid \mathbf{D} = \mathbf{d})$.

- The expected utility factor U_j of decision $\mathbf{D} = \mathbf{d}$ is obtained by summing the chance variables out of that table:

$$\mathcal{E}(U_j \mid \mathbf{D} = \mathbf{d}) = \sum_{\mathbf{C}} \prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

- The decisions that maximize the expected utility can be found from the sum of the expected utility factors:

$$\mathit{maxEU}(\mathbf{D}) = \max_{\mathbf{D}=\mathbf{d}} \sum_{j=1}^{k} \sum_{\mathbf{C}} \prod_{i=1}^{n} P(C_i \mid \mathit{par}(C_i)) U_j(\mathit{par}(U_j))$$

Sequential Decisions

Generalization

SSDNs are generalized to sequential decision problems by:

- several decisions are taken (in a fixed order)
- some chance variables may be *observed* before the next decision is taken

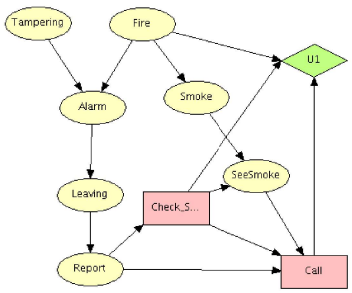
Examples

- Doctor first decides which test to perform, then observes test outcome, then decides which treatment to prescribe
- Before we decide to take the umbrella with us, we observe the weather forecast
- A company first decides whether to develop a certain product, then observes the customer reaction in a test market, then decides whether to go into full production.

Example: Fire scenario

Fire alarm example extended with

- two decisions: *CheckSmoke* $\in \{yes, no\}$, *Call* $\in \{call, do\_not\_call\}$
- one more random variable *SeeSmoke* $\in \{yes, no\}$
- a utility function depending on *Fire*,*CheckSmoke*,*Call* (composed of two factors: one depending on *CheckSmoke*, one on *Fire* and *Call*)



CheckSmoke	*Fire*	*Call*	Utility
yes	yes	call	-220
yes	yes	do_not_call	-5020
yes	no	call	-220
yes	no	do_not_call	-20
no	yes	call	-200
no	yes	do_not_call	-5000
no	no	call	-200
no	no	do_not_call	0

We assume that the decision maker doesn't forget: the *no-forgetting assumption*

Poker Example

Chance variables:

- $OH_0, OH_1, OH_2, MH_0, MH_1, MH_2 \in \{$*nothing*, *ace*, *2 of a kind*, *2 aces*, *flush*, *straight*, *straight flush*$\}$ represent my and opponent's hand after 0,1,2 card exchanges
- $BH \in \{$*me*, *opponent*, *draw*$\}$ represents who has the better hand after the exchanges
- $OFC \in \{0, 1, 2, 3\}$, $OSC \in \{0, 1, 2\}$ represent how many cards opponent changes in first/second exchange

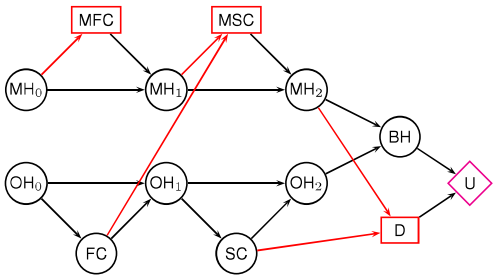
Decision variables:

- $MFC \in \{0, 1, 2, 3\}$, $MSC \in \{0, 1, 2\}$ represent how many cards I exchange in first/second exchange
- $D \in \{$*fold*, *call*$\}$ represents whether I decide to fold or call after second exchange

Utility function:

	D	
BH	*fold*	*call*
me	-1	2
opponent	-1	-2
draw	-1	0

Possible decision model



Induced order on observations and decisions:

$$\{MH_0\} \prec MFC \prec \{MH_1, OFC\} \prec MSC \prec \{MH_2, OSC\} \prec D \prec \{OH_0, OH_1, OH_2, BH\}$$

Policies

Decision Function

A **Decision Function** for a decision node D is an assignment of a decision d to each possible configuration of D's parents.

Example:

Report	*CheckSmoke*	*SeeSmoke*	*Call*
yes	yes	yes	call
yes	yes	no	do_not_call
yes	no	yes	do_not_call
yes	no	no	do_not_call
no	yes	yes	call
no	yes	no	do_not_call
no	no	yes	do_not_call
no	no	no	do_not_call

Policy

Report	*CheckSmoke*	*SeeSmoke*	*Call*
yes	yes	yes	(1,0)
yes	yes	no	(0,1)
yes	no	yes	(0,1)
yes	no	no	(0,1)
no	yes	yes	(1,0)
no	yes	no	(0,1)
no	no	yes	(0,1)
no	no	no	(0,1)

Policy encoding

Policies

A **Policy** π consists of one decision function for each decision node.

- General strategy for actions (decisions), taking into account the possible (uncertain) effects of previous actions

Semantics

Expected Utility

As before: possible worlds ω are assignments for all decision and chance variables.

- A policy π defines a probability distribution

$$P(\omega \mid \pi)$$

over possible worlds:

  - if ω contains assignments to a decision node D and its parents which is not consistent with the decision function for D: $P(\omega \mid \pi) = 0$.
  - Otherwise: $P(\omega \mid \pi)$ is the product of all conditional probability values for the assignments to chance nodes C, given the assignment to the parents of C

- Each possible world has a utility

$$U(\omega)$$

- Obtain expected utility of a policy

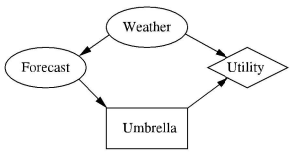
$$\mathcal{E}(U \mid \pi) = \sum_{\omega} U(\omega) P(\omega \mid \pi)$$

Optimal Policy

An **optimal policy** is a policy with maximal expected utility (among all possible policies).

Solving Sequential Decision Problems I

Initial distributions:

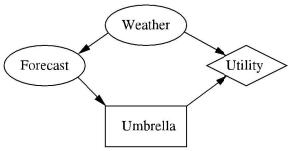


Weather	Value
norain	0.7
rain	0.3

Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Solving Sequential Decision Problems I



Initial distributions:

Weather	Value
norain	0.7
rain	0.3

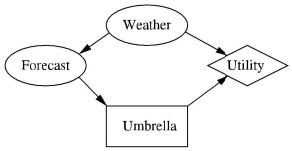
Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Combine the factors

Umb	*Weather*	*Fcast*	Value
take	norain	sunny	9.8
take	norain	cloudy	2.8
take	norain	rainy	1.4
take	rain	sunny	3.15
take	rain	cloudy	5.25
take	rain	rainy	12.6
leave	norain	sunny	49
leave	norain	cloudy	14
leave	norain	rainy	7
leave	rain	sunny	0
leave	rain	cloudy	0
leave	rain	rainy	0

Solving Sequential Decision Problems I



Initial distributions:

Weather	Value
norain	0.7
rain	0.3

Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Combine the factors

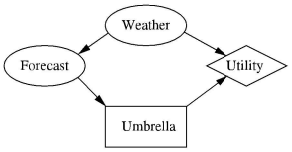
Umb	*Weather*	*Fcast*	Value
take	norain	sunny	9.8
take	norain	cloudy	2.8
take	norain	rainy	1.4
take	rain	sunny	3.15
take	rain	cloudy	5.25
take	rain	rainy	12.6
leave	norain	sunny	49
leave	norain	cloudy	14
leave	norain	rainy	7
leave	rain	sunny	0
leave	rain	cloudy	0
leave	rain	rainy	0

$\Rightarrow \sum_{Weather}$

Umb	*Fcast*	Value
take	sunny	12.95
take	cloudy	8.05
take	rainy	14
leave	sunny	49
leave	cloudy	14
leave	rainy	7

MI Autumn 2019 Sequential Decisions 23

Solving Sequential Decision Problems I



Initial distributions:

Weather	Value
norain	0.7
rain	0.3

Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Marginalising out *Umb*

Umb	*Fcast*	Value
take	sunny	12.95
take	cloudy	8.05
take	rainy	14
leave	sunny	49
leave	cloudy	14
leave	rainy	7

$\max_{Umb} f$:

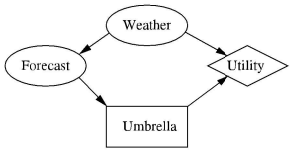
Fcast	Val
sunny	49.0
cloudy	14.0
rainy	14.0

$\arg\max_{Umb} f$:

Fcast	*Umb*
sunny	leave
cloudy	leave
rainy	take

MI Autumn 2019 Sequential Decisions 23

Solving Sequential Decision Problems I



Initial distributions:

Weather	Value
norain	0.7
rain	0.3

Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Marginalising out *Umb*

Umb	*Fcast*	Value
take	sunny	12.95
take	cloudy	8.05
take	rainy	14
leave	sunny	49
leave	cloudy	14
leave	rainy	7

$\max_{Umb} f$:

Fcast	Val
sunny	49.0
cloudy	14.0
rainy	14.0

$\arg\max_{Umb} f$:

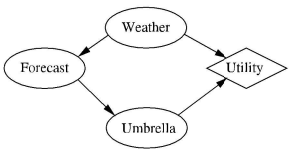
Fcast	*Umb*
sunny	leave
cloudy	leave
rainy	take

$\equiv$

Fcast	*Umb*
sunny	(0,1)
cloudy	(0,1)
rainy	(1,0)

MI Autumn 2019 Sequential Decisions 23

Solving Sequential Decision Problems I



Initial distributions:

Weather	Value
norain	0.7
rain	0.3

Weather	*Umb*	Value
norain	take	20
norain	leave	100
rain	take	70
rain	leave	0

Weather	*Fcast*	Value
norain	sunny	0.7
norain	cloudy	0.2
norain	rainy	0.1
rain	sunny	0.15
rain	cloudy	0.25
rain	rainy	0.6

Marginalising out *Umb*

Umb	*Fcast*	Value
take	sunny	12.95
take	cloudy	8.05
take	rainy	14
leave	sunny	49
leave	cloudy	14
leave	rainy	7

$\max_{Umb} f$:

Fcast	Val
sunny	49.0
cloudy	14.0
rainy	14.0

$\arg\max_{Umb} f$:

Fcast	*Umb*
sunny	leave
cloudy	leave
rainy	take

$\equiv$

Fcast	*Umb*
sunny	(0,1)
cloudy	(0,1)
rainy	(1,0)

We now have a new decision problem with one decision less. This decision problem can be solved using the same procedure until no decisions are left!

MI Autumn 2019 Sequential Decisions 23

Solving Sequential Decision Problems II

Intuition

- Given values assigned to its parents, the last decision node can be seen as a single-stage decision.
- When all decisions following a given decision D are taken according to fixed decision rules, then D also behaves like a single-stage decision.
- Backward strategy:
  - find the decision rule for the last decision D that is not yet eliminated.
  - eliminate D by replacing it with the resulting utility factor
- Formal way of "What would I do if ..." reasoning

Variable Elimination

1. *DFs* $= \emptyset$ // Set of decision functions
2. *Fs*= all conditional probability and utility tables
3. **while** there are decision nodes
4. sum out all random variables that are not parents of a decision node
   // *Fs now contains a factor F that depends on one decision node D*
   // *and (a subset of) its parents*"/
5. Add $max_D F$ to *Fs*
6. Add $arg\,max_D F$ to *DFs*
7. Sum out remaining random variables
8. Return *DFs* and product of remaining factors (expected utility of optimal policy)

Value of Information

Test Decisions

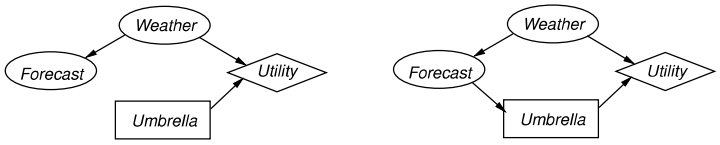
Collecting Information

- *CheckSmoke* is aimed at determining (with some uncertainty) the true state of *Smoke* (or even *Fire*)
- *Test* (medical example) is aimed at determining the true state of *Disease*

Value of information

Question: what is it worth to know the exact state of *Forecast* F when making decision *Umbrella*?

Answer: Compare maximal expected utilities of



Test Decisions

Collecting Information

- *CheckSmoke* is aimed at determining (with some uncertainty) the true state of *Smoke* (or even *Fire*)
- *Test* (medical example) is aimed at determining the true state of *Disease*

Value of information

Question: what is it worth to know the exact state of a random variable C when making decision D?

Answer: compute

- the expected value val_0 of optimal policy in given decision network
- the expected value val_1 of optimal policy in modified decision network:
  - add an edge from C to D and all subsequent decisions
- $val_1 - val_0$ is the value of knowing C.

Properties of Value of Information

- Value of information is always non-negative
- Value of knowing C for decision D is zero, if no observed value of C can change the decision rule, i.e. for all values $\mathbf{p}$ of existing parents of D, and all values c of C, the optimal decision given $(\mathbf{p}, c)$ is the same as the optimal decision given $(\mathbf{p})$.

Machine Intelligence

Lecture 12: Multi-agent systems

Thomas Dyhre Nielsen

Aalborg University

Tentative course overview

Topics:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Representing domains endowed with uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- Machine learning: classification
- Machine learning: clustering
- Planning
- **Multi-agent systems**

Multi-Agent Systems

From Single to Multi Agent Systems

So far ...

We have modeled an agent that decides/plans in a world with/without uncertainty.

New Dimension

Agent acts in an environment containing other agents. Other agents might have competing/conflicting objectives.

Using Uncertainty

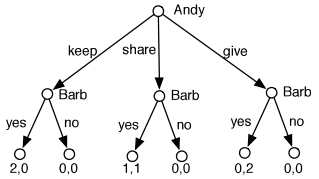
The actions of other agents can partly be represented as uncertainty in effects of own actions (uncertainty of state transitions).

Better: take explictly into account

- Competing objectives of other agents
- Reasoning about what other agents will do (reduce uncertainty)
- Possibility to collaborate to achieve common objectives

Game Trees

The sharing "game": Andy and Barb share two pieces of pie:



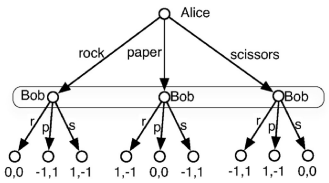
Extensive Form Representation

Representation by **game tree**:

- tree whose nodes are labeled with agents
- outgoing arcs labeled by actions of agent
- leaves labeled with one utility value for each agent
- (can also have *nature* nodes that represent uncertainty from random effects, e.g. dealing of cards, rolling of dice)

Imperfect Information Games

Representation of game with simultaneous moves:



Collect in an **information set** the nodes that the agent (Bob) can not distinguish (at all nodes in an information set the same actions must be possible).

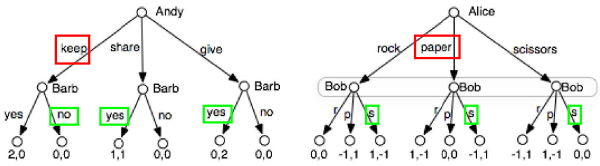
Other sources for imperfect information:

- Unobserved, random moves by nature (dealing of cards).
- Hidden moves by other agent

Strategies

A **(pure) strategy** for one agent is a mapping from information sets to (possible) actions.

Example strategies for A and strategies for B:



(A strategy is essentially the same as a policy)

A **strategy profile** consists of a strategy for each agent.

Utility

Utility for each agent given a strategy profile:

- each node has the utilities that will be reached at a leaf by following the strategy profile
- the utilities at the node represent the outcome of the game (given the strategy profile)
- (utilities at a *nature* node are computed by taking the *expectation* over the utilities of its successors)

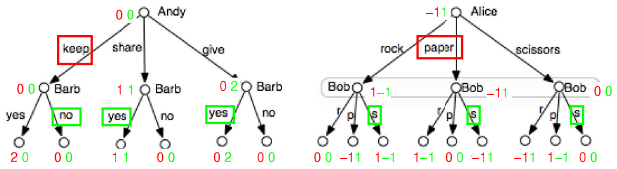


Figure shows the utilities for A and utilities for B at all nodes.

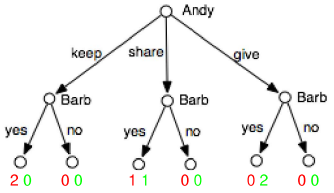
Solving Perfect Information Gain

If

- game is perfect information (no information sets with more than 1 node)
- both agents play rationally (optimize their own utility)

then the optimal strategies for both players are determined by

- bottom-up propagation of utilities under optimal strategies, where
- each player selects the action that leads to the child with the highest utility (for that player)



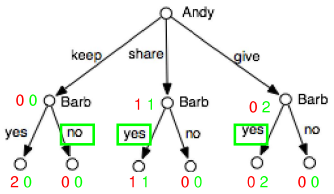
Solving Perfect Information Gain

If

- game is perfect information (no information sets with more than 1 node)
- both agents play rationally (optimize their own utility)

then the optimal strategies for both players are determined by

- bottom-up propagation of utilities under optimal strategies, where
- each player selects the action that leads to the child with the highest utility (for that player)



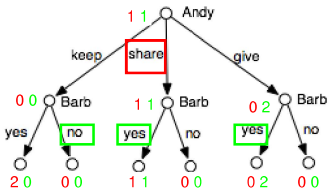
Solving Perfect Information Gain

If

- game is perfect information (no information sets with more than 1 node)
- both agents play rationally (optimize their own utility)

then the optimal strategies for both players are determined by

- bottom-up propagation of utilities under optimal strategies, where
- each player selects the action that leads to the child with the highest utility (for that player)



Zero Sum Games

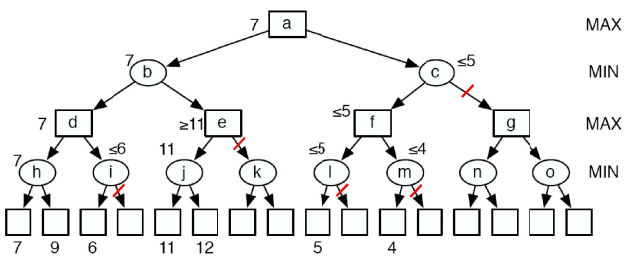
Zero Sum Game for two players:

$$\textit{utility of player 1} = -\textit{utility of player 2}$$

In this case:

- need only one utility value at the leaves
- one player (called *Max*) wants to reach leaf with maximal value, the other (*Min*) wants to reach leaf with minimal value.

In the bottom-up utility computation some sub-trees can then be **pruned** (α-β-pruning):



Solving Checkers

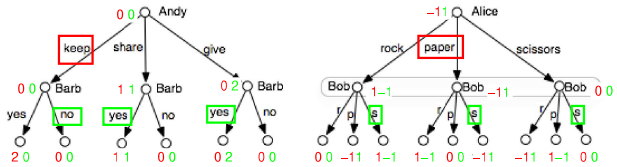
J.Schaeffer et al.: *Checkers Is Solved*. Science, July 2007

- Schaeffer et al. proved: there is no winning strategy for either player: perfect play by both players will always result in a draw
- checkers has approximately $5 \cdot 10^{20}$ different positions
- in the proof only about 10^{14} positions were explored
- reduction by several techniques, including α-β-pruning.

Imperfect Information

Normal Form

For each strategy profile, utilities of game are determined:



Share game:

- Strategy Andy: *keep*
- Strategy Barb: *no if keep, yes if share, yes if give*
- Utilities: 0 for Andy, 0 for Barb

Rock Paper Scissors:

- Strategy Alice: *paper*
- Strategy Bob: *scissors*
- Utilities: -1 for Alice, 1 for Bob

Can view game simply as consisting of

- Choice of action by A (possibly: action=strategy)
- Choice of action by B (possibly: action=strategy)
- Utilities determined by these choices

Normal form representations

Share game

		Andy	
Barb	keep	share	give
$k \to y,s \to y,g \to y$	2 0	1 1	0 2
$k \to y,s \to y,g \to n$	2 0	1 1	0 0
$k \to y,s \to n,g \to y$	2 0	0 0	0 2
$k \to y,s \to n,g \to n$	2 0	0 0	0 0
$k \to n,s \to y,g \to y$	0 0	1 1	0 2
$k \to n,s \to y,g \to n$	0 0	1 1	0 0
$k \to n,s \to n,g \to y$	0 0	0 0	0 2
$k \to n,s \to n,g \to n$	0 0	0 0	0 0

Rock Paper Scissors

		Alice	
Bob	rock	paper	scissors
rock	0 0	1 -1	-1 1
paper	-1 1	0 0	1 -1
scissors	1 -1	-1 1	0 0

Difference between perfect and imperfect information not directly visible in normal form representation!

Nash Equilibrium

Consider optimal strategy profile for share game:

		Andy	
Barb	keep	share	give
$k \to y,s \to y,g \to y$	2 0	1 1	0 2
$k \to y,s \to y,g \to n$	2 0	1 1	0 0
$k \to y,s \to n,g \to y$	2 0	0 0	0 2
$k \to y,s \to n,g \to n$	2 0	0 0	0 0
$k \to n,s \to y,g \to y$	0 0	1 1	0 2
$k \to n,s \to y,g \to n$	0 0	1 1	0 0
$k \to n,s \to n,g \to y$	0 0	0 0	0 2
$k \to n,s \to n,g \to n$	0 0	0 0	0 0

The two strategies are in **Nash equilibrium**:

- no agent can improve utility by switching strategy while other agent keeps its strategy
- this also means: agent will stick to strategy when it knows the strategy of the other player

Prisoner's Dilemma

Alice and Bob are arrested for burglary. They are separately questioned by police. Alice and Bob are both given the offer to *testify*, in which case

- they will receive a sentence of 5 years each if both testify
- if only one testifies, that person will receive 1 year, and the other 10 years
- if neither testifies, both will get 2 years

	Alice	
Bob	*testify*	*not testify*
testify	-5 -5	-10 -1
not testify	-1 -10	-2 -2

- The only Nash equilibrium is Alice:*testify*, Bob:*testify*
- Nash equilibria do not represent cooperative behavior!

Mixed Strategies

No pure strategy Nash equilibrium in Rock Paper Scissors:

	Alice		
Bob	rock	paper	scissors
rock	0 0	1 -1	-1 1
paper	-1 1	0 0	1 -1
scissors	1 -1	-1 1	0 0

A **mixed strategy** is a probability distribution over actions.

Mixed Strategy for Alice: $r : 1/3\ p : 1/3\ s : 1/3$
Mixed Strategy for Bob: $r : 1/3\ p : 1/3\ s : 1/3$

Expected utility for Alice = expected utility for Bob =

$$1/9(0 + 1 - 1 - 1 + 0 + 1 + 1 - 1 + 0) = 0$$

Mixed Strategies

No pure strategy Nash equilibrium in Rock Paper Scissors:

	Alice		
Bob	rock	paper	scissors
rock	0 0	1 -1	-1 1
paper	-1 1	0 0	1 -1
scissors	1 -1	-1 1	0 0

A **mixed strategy** is a probability distribution over actions.

Mixed Strategy for Alice: $r : 1/3\ p : 1/3\ s : 1/3$
Mixed Strategy for Bob: $r : 1/3\ p : 1/3\ s : 1/3$

Expected utility for Alice = expected utility for Bob =

$$1/9(0 + 1 - 1 - 1 + 0 + 1 + 1 - 1 + 0) = 0$$

Suppose Alice plays some other strategy: $r : p_r\ p : p_p\ s : p_s$. Expected utility for Alice then:

$$1/3(p_r \cdot 0 + p_p \cdot 1 - p_s \cdot 1 - p_r \cdot 1 + p_p \cdot 0 + p_s \cdot 1 p_r \cdot + 1 - p_p \cdot 1 + p_s \cdot 0) =$$
$$1/3(p_p + p_r + p_s - p_p - p_r - p_s) = 0$$

- If Bob plays $r : 1/3\ p : 1/3\ s : 1/3$, Alice can not do better than playing $r : 1/3\ p : 1/3\ s : 1/3$ also.
- Same for Bob
- Both playing $r : 1/3\ p : 1/3\ s : 1/3$ is a (the only) Nash equilibrium

Key Results

- Every (finite) game has a Nash equilibrium (using mixed strategies)
- There can be multiple Nash equilibria
- Playing a Nash equilibrium strategy profile does not necessarily lead to optimal utilities for the agents (prisoner's dilemma)

The Exam

The exam

Some practical issues

- January 8th, 2019.
- Written exam with internal censor.
- Graded exam.
- Answers should be written in English.
- A "question session" is scheduled for ??.

What the course has covered

The course has covered the following issues:

- Introduction
- Search-based methods
- Constrained satisfaction problems
- Logic-based knowledge representation
- Reasoning under uncertainty.
- Bayesian networks
- Inference in Bayesian networks
- Machine learning
- Planning
- Multi-agent systems

This corresponds to the following literature:

- David L. Poole and Alan K. Mackworth, Artificial Intelligence: Foundations of computational agents (Second edition): Preface, Ch. 1, 3-3.6, 3.7-3.7.1, 3.7.3, 3.8.2 - 3.8.3, 4-4.7.3, 5-5.2, 7-7.5 (except 7.4.2), 7.7, 8-8.4.1, 8.6-8.6.5, 9-9.4 (except 9.1.3), 10.1.2, 10.2, 11-11.4, A.3
- Finn V. Jensen and Thomas D. Nielsen, Bayesian networks and decision graphs: Sections 2-2.2, 3-3.1.
- The slides from the course.