

Lecture 1

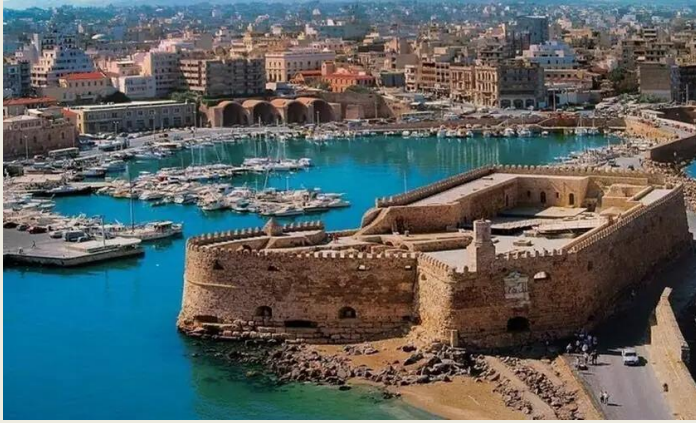
Introduction

Michail G. Lagoudakis

**Intelligent Systems Laboratory
School of Electrical and Computer Engineering
Technical University of Crete
Chania, Crete, Greece**



A few words about me (in pictures)



More words about me

- Research interests

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Decision Making under Uncertainty
- Reinforcement Learning (RL)
- Autonomous Robotics
- Multi-Agent Systems
- Robot Team “Kouretes”



A few words about the course

Course objectives

1. To introduce the students to the basic concepts of Artificial Intelligence (AI) and Machine Learning (ML).
2. To present to the students state-of-the-art applications of AI and ML in medicine and healthcare, focusing on decision-support systems for disease diagnosis and treatment.
3. To develop the students' interest for such systems and for AI and ML in general.

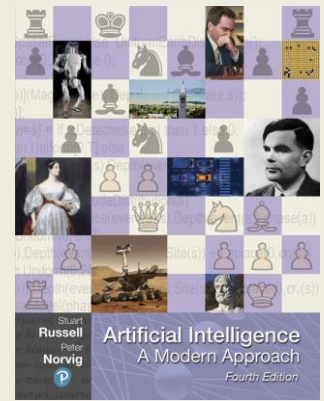


Course Content (7 lectures)

1. **Introduction to AI.** What is AI. History of AI. Symbolic AI and Expert Systems. Statistical AI. Data-driven AI. The Intelligent Agent paradigm. AI for medicine: an overview.
2. **Intelligent Search methods.** Informed Search and Uninformed Search.
3. **Constraint Satisfaction Problems.** Applications in the Medical Domain.
4. **Logic:** propositional logic and first-order logic. Logical Inference.
5. **Decision-making under uncertainty:** Bayesian reasoning, Markov Decision Processes.
6. **Expert Systems** for Medical Decision Support and Assistive Technologies (symbolic AI-based expert systems, Bayesian systems, hybrid approaches).
7. **Machine Learning:** Neural Networks, Deep Learning, and their applications in medicine and healthcare.



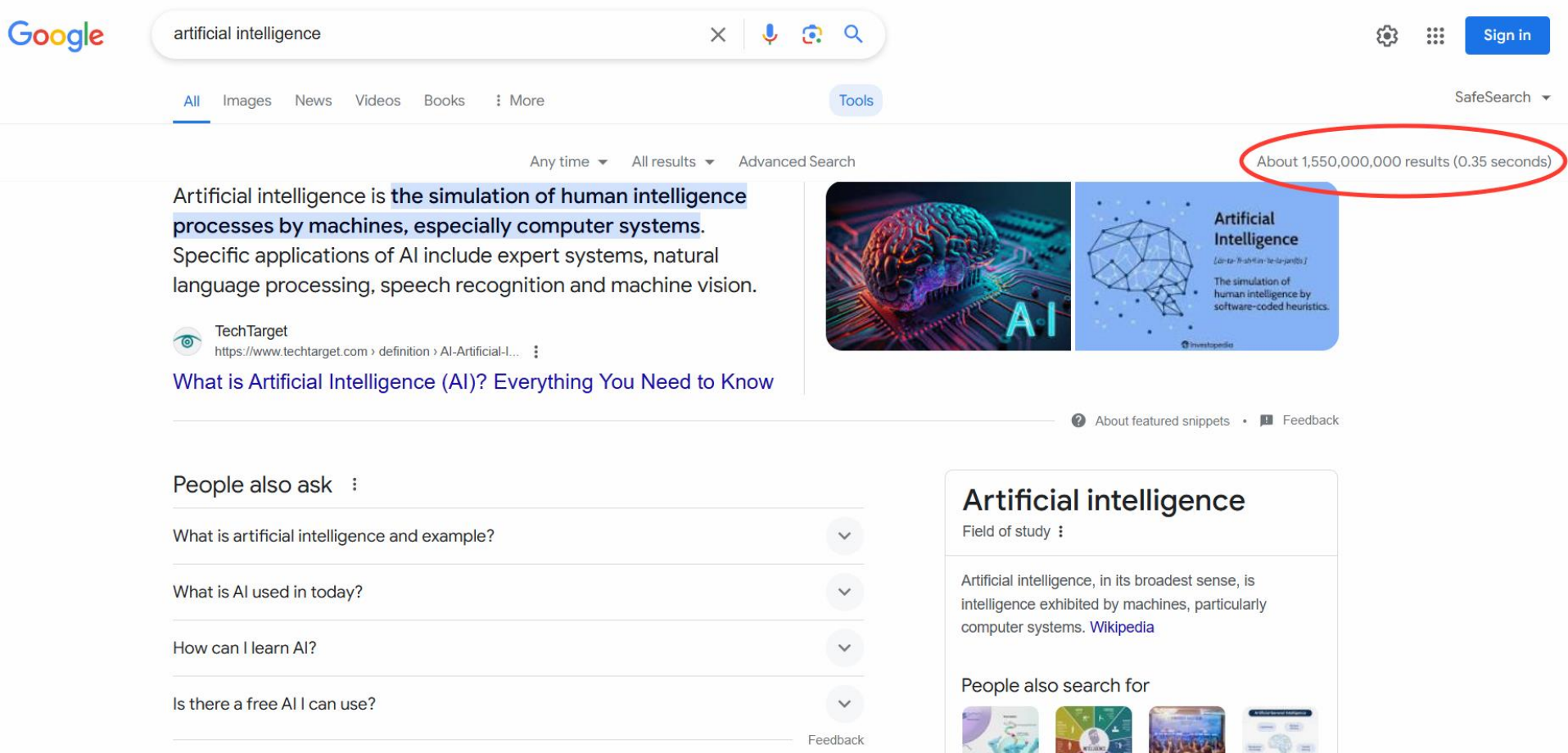
Bibliography



- *Artificial Intelligence: A Modern Approach.* Stuart Russell and Peter Norvig, 4th edition, Prentice Hall, 2020.
- *Artificial Intelligence: Structures and Strategies for Complex Problem Solving.* George Luger. 5th edition, Addison-Wesley, 2005.
- *Artificial Intelligence: A New Synthesis.* Nils Nilsson. Morgan Kaufmann, 1998.
- *Artificial Intelligence: Theory and Practice.* Dean, Allen, Aloimonos. Benjamin/Cummings Pub. Co., 1998.
- *Artificial Intelligence.* Patrick Winston, 3rd edition, Addison-Wesley, 1992.
- *Artificial Intelligence.* Rich and Knight. 2nd edition, McGraw Hill, 1991.
- *An Introduction to Multi-Agent Systems.* Michael Wooldridge. Wiley, 2nd edition, 2009.
- *Guide to Health Informatics.* Enrico Coiera, CRC Press, 3rd edition, 2015.
- Various research articles



AI is constantly in the news these days!



The screenshot shows a Google search for "artificial intelligence". The search bar is at the top, with the Google logo on the left and a "Sign in" button on the right. Below the search bar, there are tabs for "All", "Images", "News", "Videos", "Books", and "More". The "All" tab is selected. To the right of the tabs, there is a "Tools" button and a "SafeSearch" dropdown menu. The search results are displayed below the tabs. At the top of the results, there is a summary box with the text: "Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision." To the right of this text is a red circle containing the text: "About 1,550,000,000 results (0.35 seconds)". Below the summary box, there is a snippet from TechTarget with the title "What is Artificial Intelligence (AI)? Everything You Need to Know". To the right of the snippet is a small image of a brain on a circuit board with the letters "AI" in the bottom right corner. Below the snippet, there is a section titled "People also ask" with four questions: "What is artificial intelligence and example?", "What is AI used in today?", "How can I learn AI?", and "Is there a free AI I can use?". To the right of this section is a box titled "Artificial intelligence" with the subtitle "Field of study". Inside this box, there is a paragraph of text: "Artificial intelligence, in its broadest sense, is intelligence exhibited by machines, particularly computer systems. Wikipedia". Below this paragraph is a section titled "People also search for" with four small images representing different aspects of AI.

Google

artificial intelligence

Sign in

All Images News Videos Books More

Tools

SafeSearch

Any time All results Advanced Search

About 1,550,000,000 results (0.35 seconds)

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

TechTarget
https://www.techtarget.com/definition/AI-Artificial-I...

What is Artificial Intelligence (AI)? Everything You Need to Know

People also ask

What is artificial intelligence and example?

What is AI used in today?

How can I learn AI?

Is there a free AI I can use?

Feedback

Artificial intelligence

Field of study

Artificial intelligence, in its broadest sense, is intelligence exhibited by machines, particularly computer systems. Wikipedia

People also search for



Sooner or later, everyone gets a haircut...



Watch this video: <https://www.youtube.com/watch?v=IXUQ-DdSDoE>

“The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find **how to make machines use language**, **form abstractions and concepts**, **solve kinds of problems now reserved for humans**, and **improve themselves**.”

(From John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon, Proposal for the 1956 Dartmouth Workshop on AI, 1955)



So...what is AI?

Is it about Robots?

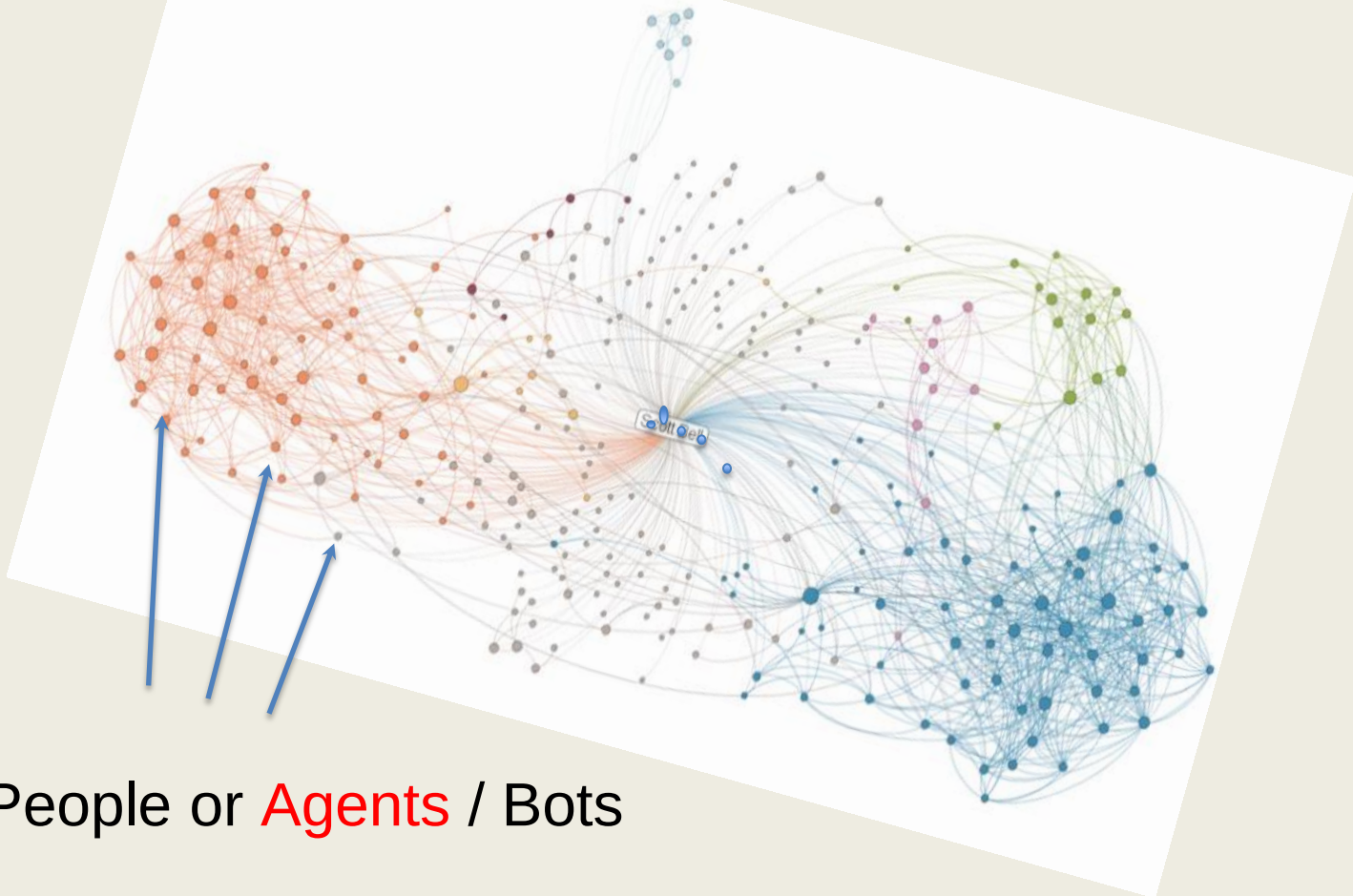
Or Robotic Waiters/Waitresses?

- ...the “strong AI” vision...





It's not a brain... it's a social network... part of the WWW... or is it our collective brain?



People or **Agents** / Bots





It's not a brain...it's a social network...part of the WWW... or is it our collective brain?

A Huge Multi-Agent System

Decision Making

Problem Solving

People or **Agents** / Bots





Who are these agents?

- **Rational Agent**

- Reactive
- Proactive

- **Socially-able**

- Acts to achieve the **best possible outcome**, given facts/knowledge + informed opinions/beliefs
- **Combines multiple abilities in the best possible manner**

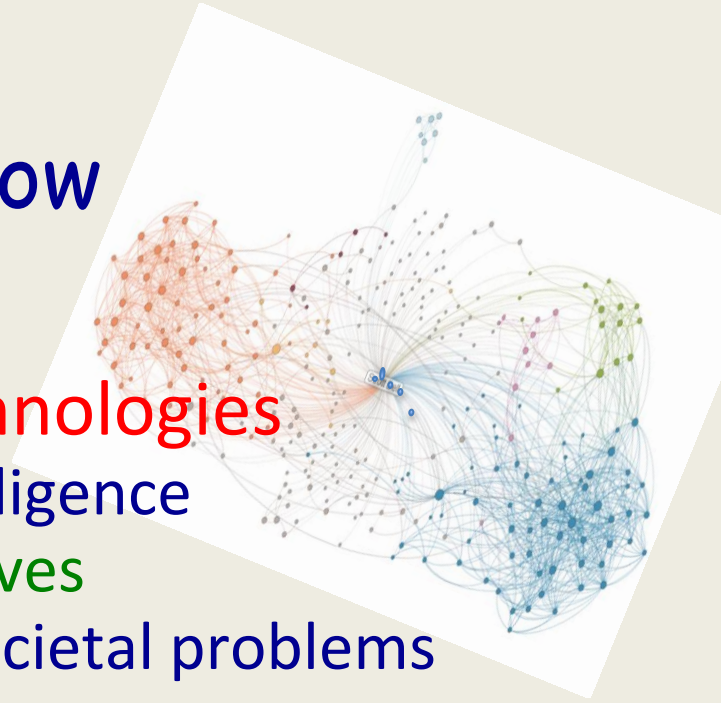
"We are social beings"

Aristotle



AI in the Web of Tomorrow

- **Man-Agents collaboration mediated by digital interaction technologies**
 - combining human **and** machine intelligence
 - **exploiting diverse large-scale collectives**
 - ...to solve complex computational/societal problems
- **Examples:**
 - optimizing the transportation system of a city
 - managing the treatment plans of a patient group
 - coordinating communities of
 - software developers;
 - activists;
 - Smart Grid consumers



Putting the human (back) in the loop?

- In a sense, this is yet another step towards abandoning the “strong AI” vision...



Thus, it's ok to not build the super-human,
omnipotent intelligent agent ...

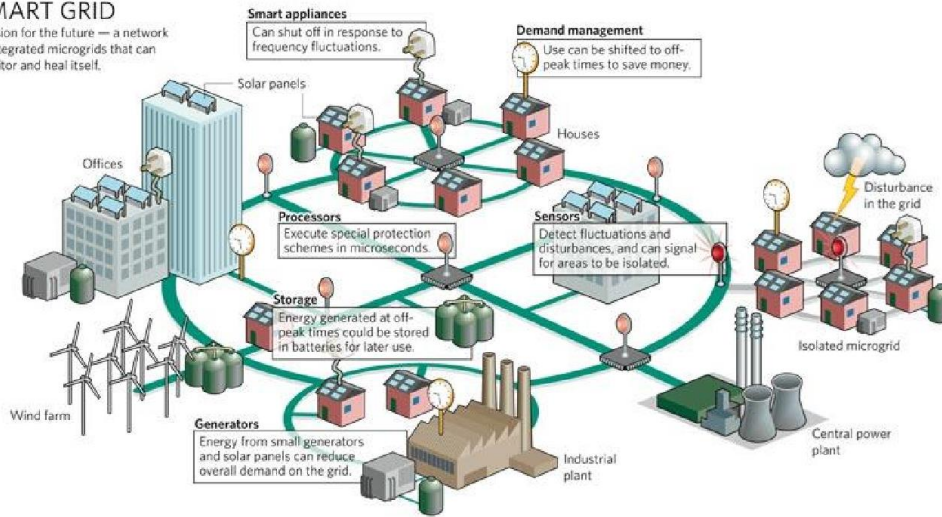


...as long as we get to get things done!...

- ...to ease our everyday living...and even save lives!

SMART GRID

A vision for the future — a network of integrated microgrids that can monitor and heal itself.



AI Applications

- Applications include (*but are not restricted to*) :
 - Medical Support Systems
 - Machine Learning-Based Medical Imaging
 - Decision Support Systems / Decision networks
 - Smart Electricity Grid, Energy efficiency, Virtual Power Plants
 - Sensor networks / wide area surveillance and security systems
 - Natural Language Processing, Understanding
 - Speech Recognition, Generation, Understanding
 - Autonomous Driving
 - Multi-agent coordination
 - Disaster management systems
 - Decentralized resource allocation
 - Self-organizing agent organizations for task execution
 - Personalized recommendation systems and web applications
 - Automated negotiations
 - Computer Games and Recreation
 - Personalized educational environments
 - e-Democracy – enhancing voters' decision making abilities
 - ...



So, can we really have smart machines? How can we build them?

- We had a glimpse at what AI is/might be/can do/cannot do...
- However:

Intelligence Itself is Difficult to Define

thus

AI is Difficult to Define



(Artificial) Intelligence: more than one dimensions

- Vision: The Intelligent Autonomous Agent
 - Multiple desirable characteristics and abilities



(Artificial) Intelligence: more than one dimensions

- AI is not just Machine Learning
- AI is not just Deep Learning
- AI is not just TensorFlow
- AI is not just ChatGPT, ...

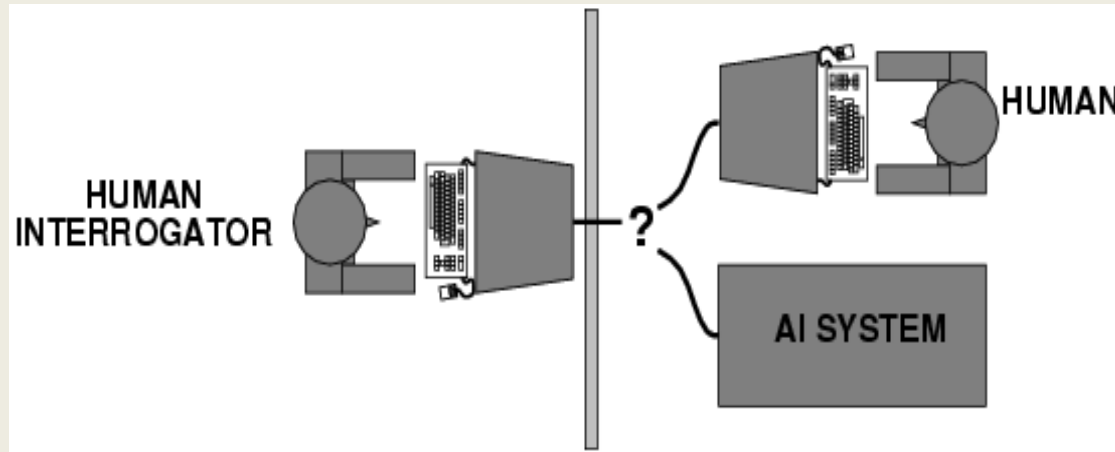


- **Need to combine:** Machine Vision, Machine Learning, Logic, Human-Computer Interaction, Big Data Science, Mathematical Optimization, Decision Theory, Game Theory, Social Choice Theory, Social Anthropology, ...



Can machines be smart?

Turing test (1950)



“Can machines think?”
... Define “machines” ...
... Define “think” ...



Can machines be smart?

- **The “Strong AI” hypothesis:** *if machines act intelligently, as humans do, that means that they also think or that they are smart, like humans are - and we will make such machines*
- Question: *Is “appearances” enough?*
 - (some) Philosophers: *AI simply emulates thinking*
 - **A response:** **expressing similar doubts for a human would have been rude** [Alan Turing]
 - Still, it’s fairly easy and ...a no-brainer to pass a Turing test!
 - *How to Pass a Turing Test...*
 - <http://www.cse.buffalo.edu/~rapaport/Papers/TURING.pdf>
 - “The Chinese Room”



The Chinese Room

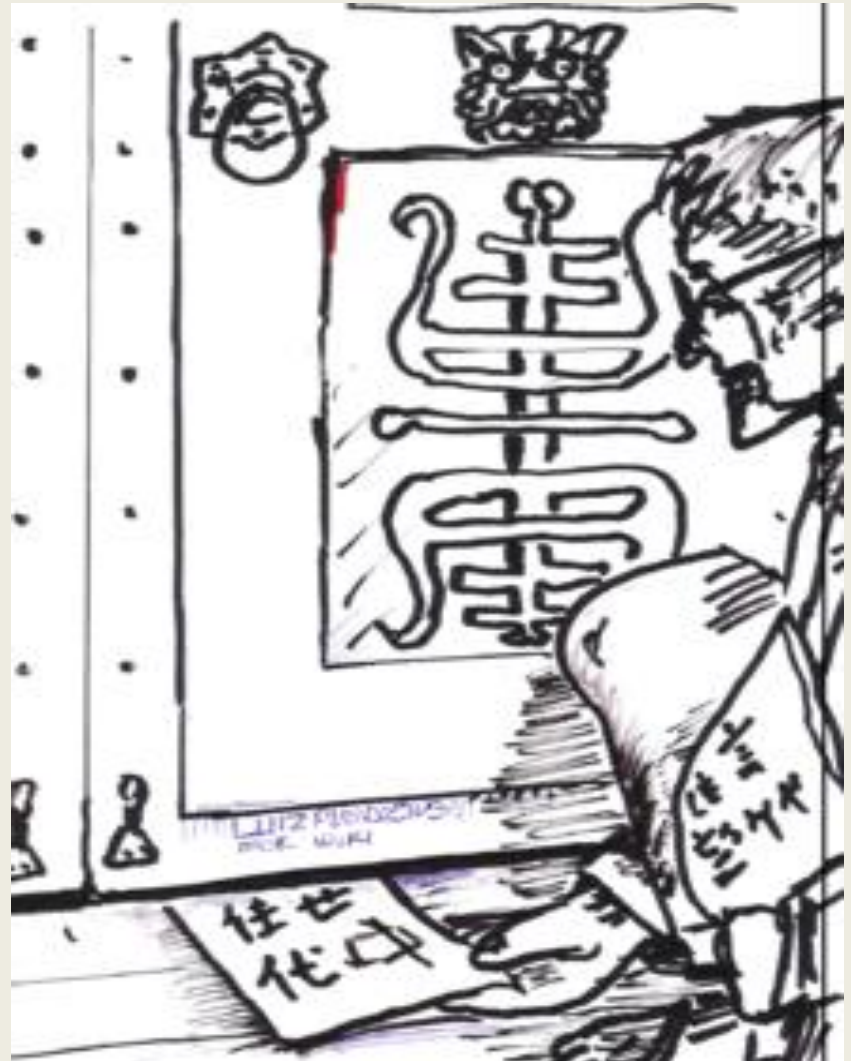


The Chinese Room

- But, ... if a machine uses a series of approximate computations and reaches a response, simply using “math magic” that appears **to have nothing to do with human thinking** (e.g. **probabilistic reasoning**) ... would that make it intelligent/rational?

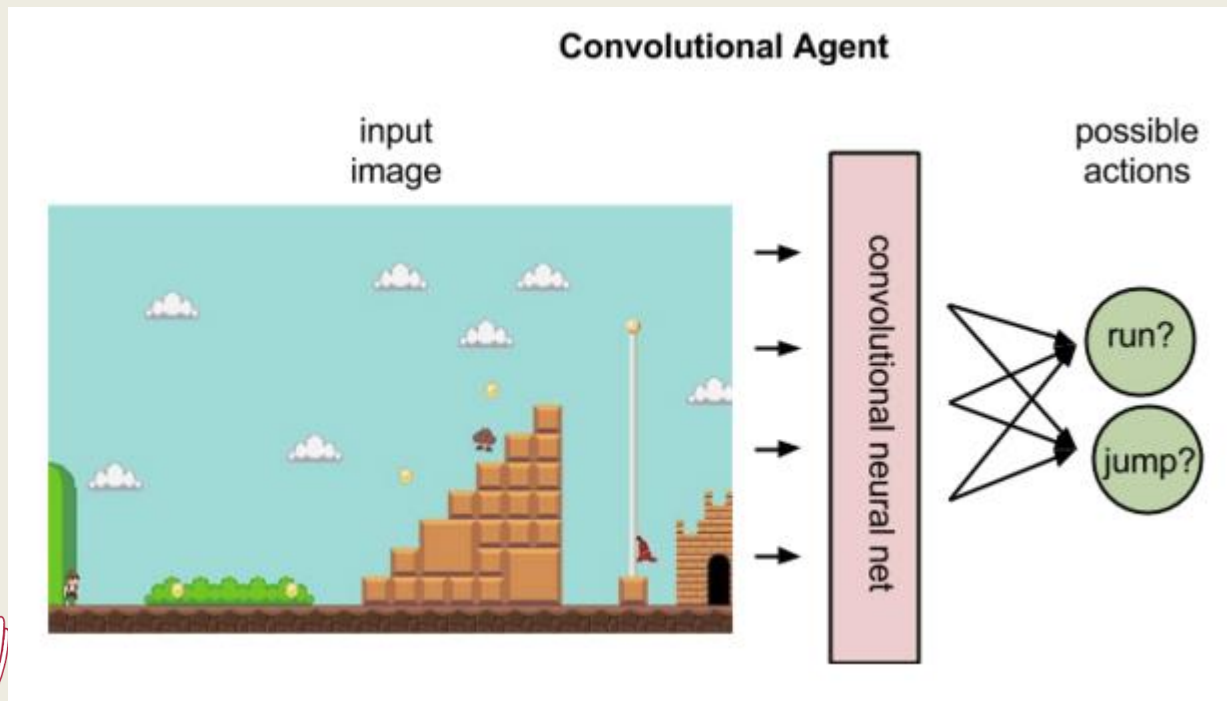
*- **Yes!** There are more possible combinations (or lemmas in the lexicon) than atoms in the universe! If that is not intelligent, then what is?*

*Judea Pearl, Turing Award Speech, 2012
(a liberal rendition by GC)*



Can machines be smart?

- Machines can only *follow (logical) rules*, they cannot be intelligent in the way humans are. [Dreyfus 1972, 1986, 1992]
 - Well, AI has moved far beyond that
 - machines nowadays can learn, even without a teacher



source:
<https://shivam5.github.io/drl/>



Can machines be smart?

- Dreyfus again: The chess grand master argument
 - The right move just “appears” in the mind of the grand master
 - no complex calculations required!
 - Ok...but... *how* does that happen? Philosophers cannot answer the question
 - ...plus grand masters have a tough time with chess-playing machines these days...



Success Stories

Deep Blue (1997) and Deep Fritz (2003)

“25 Years Ago, Chess Changed Forever When Deep Blue Beat Garry Kasparov”

online news, 2021

“Kasparov 'forced' to draw with X3D Fritz”, online news, 2003



Success Stories

Jeopardy!

*“It’s alive: IBM’s Watson
supercomputer
defeats humans in final
Jeopardy match”
online news, 2011*



Success Stories

Mars Rovers and Helicopter

“Seventh Landing Anniversary on Mars for the Twin NASA Rovers” (2011)



“Hilltop Panorama marks Mars Rover’s 11th Anniversary” (2015)

“NASA’s Ingenuity Mars Helicopter to Begin New Demonstration Phase” (2021)



mars.nasa.gov



Success Stories

Companion Robots Care Providing Robots

“Robot nannies play with kids and change their diapers...”

“Japan eyes robotic future for aged care”

“Robot nannies are here, but won’t replace your babysitter – yet”

“Demand for companion robots will hike from \$3.8 billion in 2015 to \$34.1 billion by 2022”

online news, 2010, 2014, 2017



Success Stories

ChatGPT



what do you know about Biomedical Engineering?



Biomedical Engineering (BME) is an interdisciplinary field that applies principles and problem-solving techniques from engineering, biology, and medicine to design and create equipment, devices, computer systems, and software used in healthcare. The goal of BME is to improve patient care by developing technologies that can prevent, diagnose, and treat medical conditions.

Key Areas of Biomedical Engineering:

1. Medical Imaging:



...sadly, not always...

Collateral Losses

“New massacres of civilians in Afghanistan in the last 24 hours with another 12 “collateral losses” to be added to the long list of victims ...”

online news, 2010



Defining AI



The Intelligent Autonomous Agent Paradigm

	human	rationality
thinking	Thinking like humans	Thinking rationally
acting	Acting like humans	Acting rationally

source: the AIMA book – by Russell and Norvig



Human Thinking / Human cognition

- Cognitive (neuro)science
 - Psychology + computer science: study information processing in the brain
 - cognition: trace reasoning steps to brain centers / functions
 - Develop a theory of the human mind
- Confirmation via:
 - psychological / medical experiments on humans
 - interpretation of experiments, predicting subjects' behaviour (top-down)
 - direct identification from neurological data (bottom-up)
- GPS (General Problem Solver): reason like a human
 - Newell and Simon, 1961



Thinking rationally / Logical thinking

- Laws of thought / laws of logic
 - Aristotelian reasoning (“...therefore, Socrates is a mortal”)
 - Theory of Logic, axioms, formal reasoning, proofs
- Logical AI
 - knowledge representation via sentences in some formal logic
 - logical reasoning = thinking
 - any problem can be solved if appropriately represented
- Obstacles
 - cannot adequately (or at all) represent/capture uncertainty or informal knowledge
 - *what’s the difference between theory and practice? ...well...*
 - *in theory there is no difference, in practice there is!*
 - insufficient computational resources to represent/solve problems formally
 - inefficient algorithms for effectively tackling large real-world problems



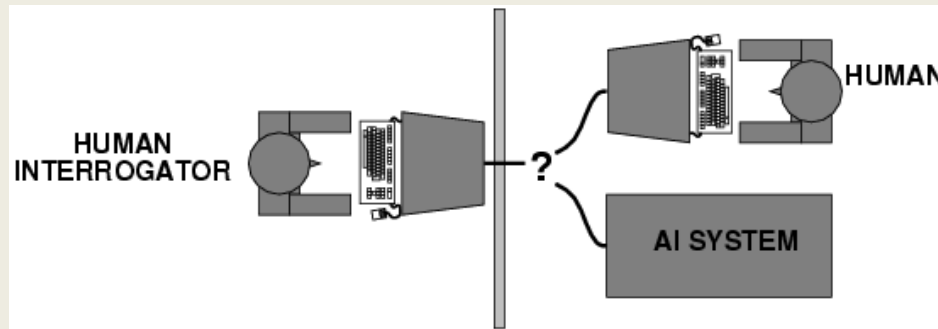
Acting like a human

...do it like a human does...

- Emulating human behavior
 - irrespectively of internal processing
 - focus on successful emulation, not methods/tools/processes used



Turing Test ++



- Natural language processing, knowledge representation, automated reasoning, machine learning ... perhaps computer vision and robotics...
- Yes, but ... does an airplane get mistaken for a bird? Which one flies better? Do their looks or styles matter?



Rational Action / Rational Behaviour

- Autonomous Agent(s)
 - agent = an entity that perceives and acts
 - rational agents
 - Do the Right Thing: perform an action (plan) whose outcome is best in expectation
- Rational Action
 - Define what is “the right thing”/ “best” in some objective manner
 - Rational reasoning: useful, but not a must
 - Some times, only using good reflexes is enough
 - Aim is to succeed in meeting goals and maximizing returns, not on methods/tools/processes used



Which definition is the best?

	human	rationality
thinking	Thinking like humans	Thinking Rationally
acting	Acting like humans	Acting rationally



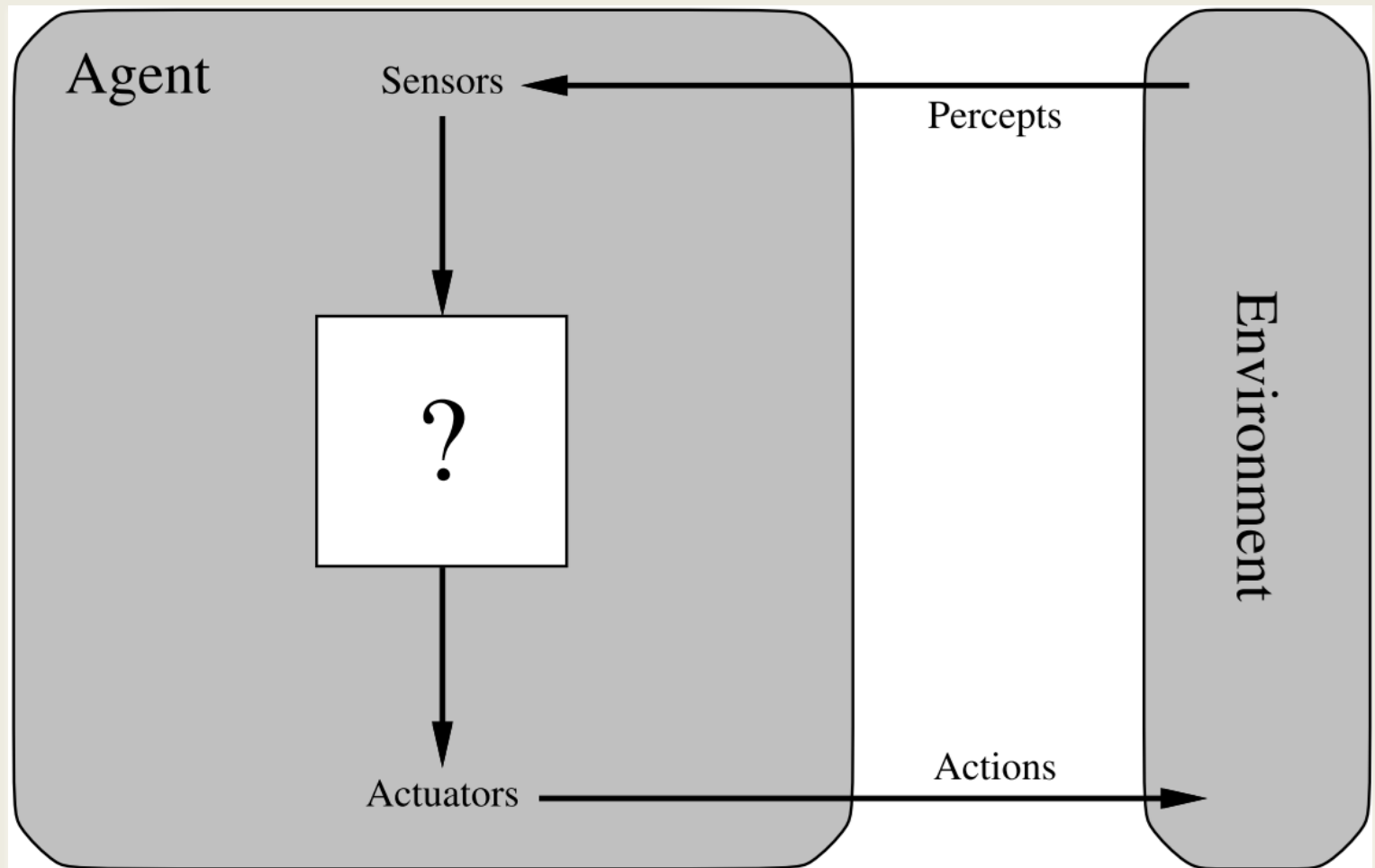
The intelligence we care about

- The “Weak AI” hypothesis:

***We do not care about machines that “think”,
it is enough if they behave “as if they are intelligent”,
that is they choose the “best” actions based on what
they know - in other words, “rationally”***



The Intelligent Autonomous Agent Paradigm ++

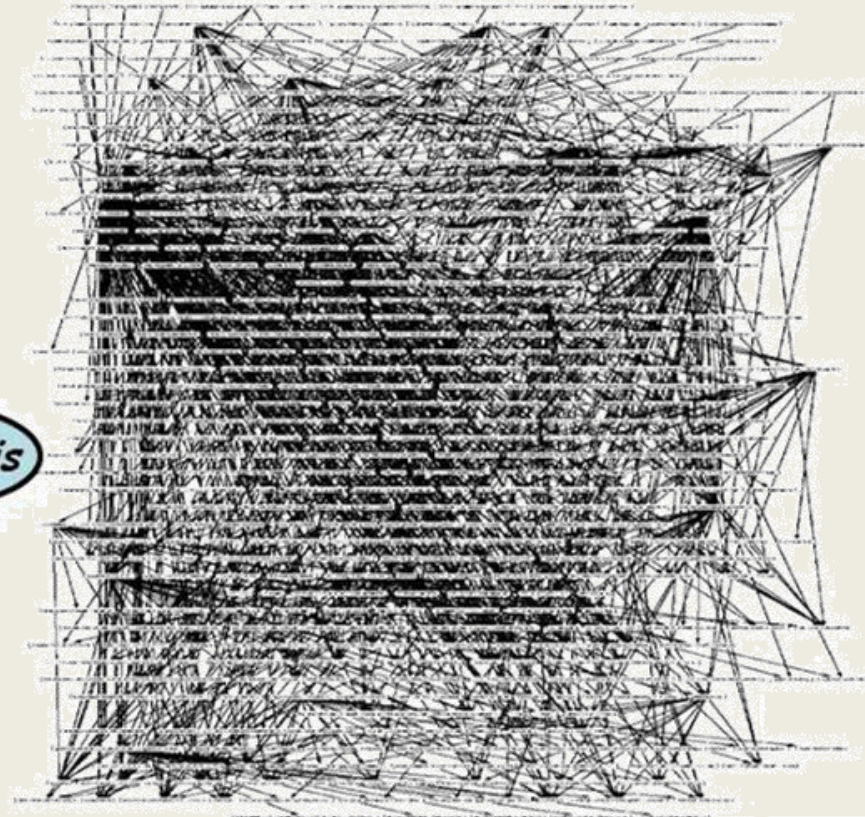
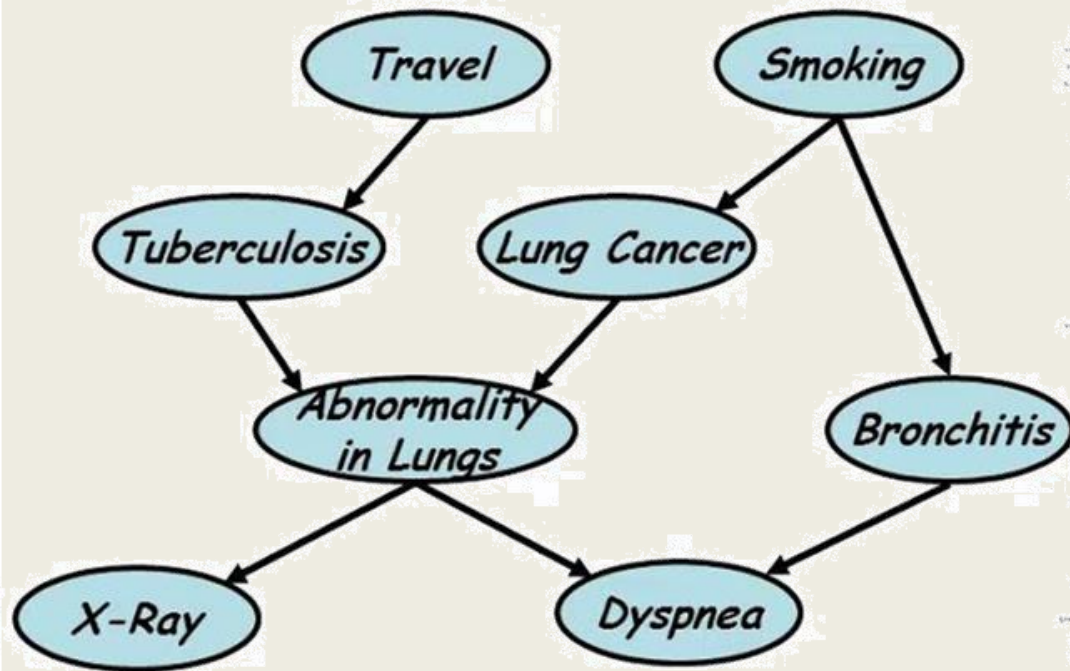


Some More Jargon and Terminology

- Symbolic AI
 - Logic-based + constraints-based AI
 - “deductive thought”:
 - deduces conclusions given “knowledge” and constraints
- Expert Systems
 - emulate experts’ opinions
 - use of appropriately represented built-in knowledge
- Statistical AI
 - probabilistic graphical models (“model-based AI”)
 - crunches numbers and data
 - “inductive thought”: discovers patterns and trends in data
- Data-Driven AI
 - statistical AI + neural networks



Graphical Models



Origins



Contributions to AI

Different fields have contributed to AI in the form of ideas, viewpoints and techniques.

- *Philosophy*: logic, reasoning, mind as a physical system
- *Mathematics*: formal representation and proof algorithms
- *Economics*: formal theory of rational decisions, game theory
- *Neurosciences*: physical substrate for mental activities
- *Psychology*: adaptation, phenomena of perception and motor control
- *Linguistics*: knowledge representation, grammars
- *Computer Engineering*: development of efficient computers
- *Control theory*: homeostatic systems, stability, optimal agent design



Philosophy (428 b.c. – today)

- **Are there any formal rules for drawing conclusions?**
 - Aristotle: exact set of laws for rational intelligence
- **How does intelligence come from the brain?**
 - materialism: intelligence is subject to the laws of nature
 - dualism: part of intelligence is outside the physical world
- **Where does knowledge come from?**
 - empiricism, logical positivism, theory of affirmation
- **How does knowledge lead to action?**
 - logical connection between objectives and knowledge of the effects of actions
 - Aristotle: first planning algorithm with backtracking!



Mathematics (c. 800 – today)

- **What are the typical rules for drawing conclusions?**
 - logic - George Boole (1847)
 - first-order logic - Gottlob Frege (1879)
- **What can be calculated and what not?**
 - Incompleteness Theorem - Godel (1931)
 - Algorithms, Theory of Computation, Solvability - Turing (1936)
 - NP-completeness - Cook (1971), Karp (1972)
- **How do we think with uncertain information?**
 - probability theory - Cardano, Fermat, Pascal, Bernoulli, Laplace
 - the Bayes rule - Thomas Bayes (1702-1761)



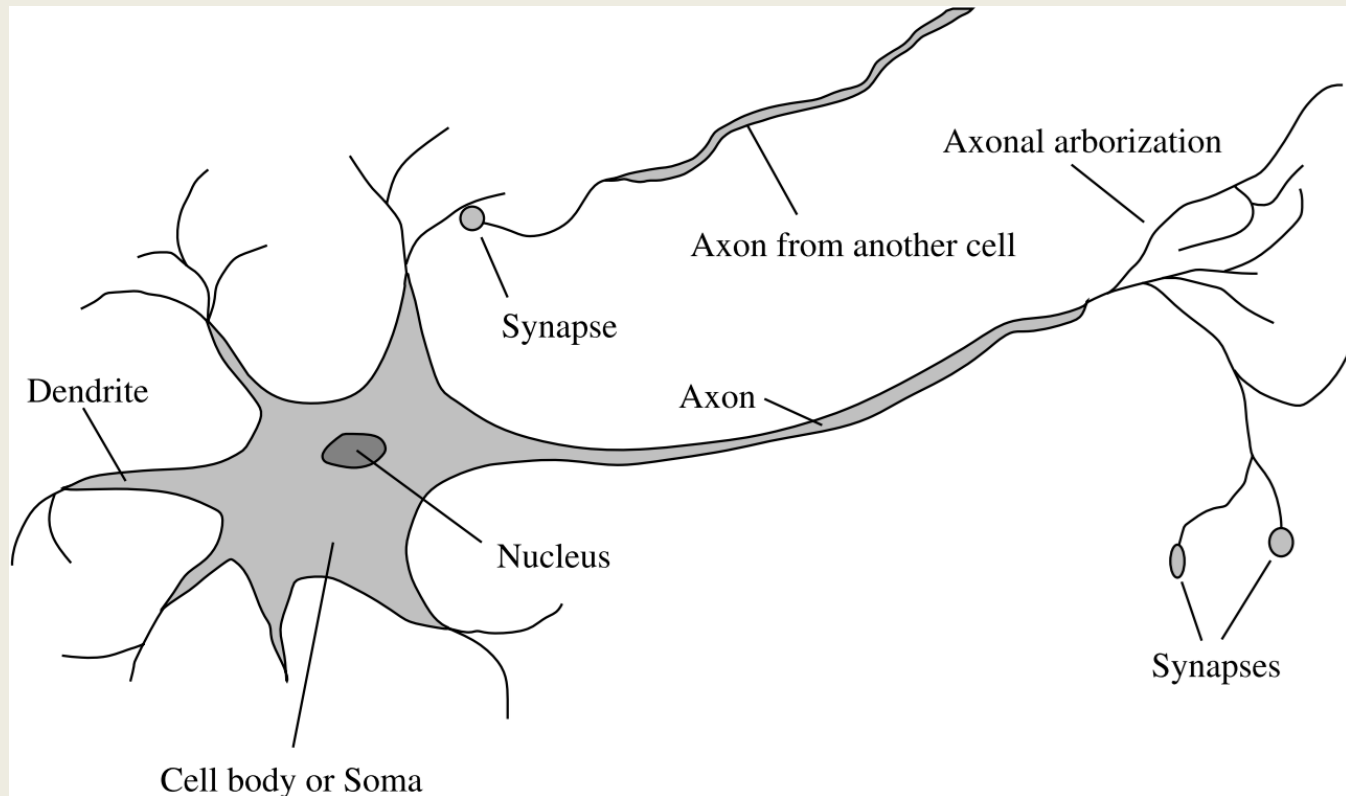
Economics (1776 - today)

- **How do we decide to maximize earnings?**
 - the concept of utility
 - decision theory = probability + utility
- **How do we decide when there is competition?**
 - game theory
- **How do we decide with a view to future earnings?**
 - operations research
 - Markov Decision Processes (MDP) - Bellman (1957)



Neurosciences (1861 - today)

- **How does the brain process information?**
 - a collection of neurons leading to intelligence (?)



Neurosciences (1861 - today)

	Computer 2003	Human Brain
Computational Units	1 CPU, 10^8 gates	10^{11} neurons
Storage Units	10^{10} bit RAM 10^{11} bit disk	10^{11} neurons 10^{14} synapses
Cycle Time	10^{-9} sec	10^{-3} sec
Bandwidth	10^{10} bit/sec	10^{14} bit/sec
Memory Updates/second	10^9	10^{14}

- the computer is 1,000,000 times faster than the brain
- the brain produces 100,000 more work due to parallelism
- there is also a difference in power consumption



Psychology (1879 - today)

- **How do humans and animals think and act?**
 - experimental psychology
 - body endoscopy
 - behaviorism
 - only input - output (stimuli - responses)
 - cognitive psychology
 - stimulus / representation / processing / action sequence
 - information processing device, first forms of agent
 - evolution to cognitive science
 - psychology and computational models
 - "a cognitive theory should look like a program"



Linguistics (1957 - today)

- **How does language relate to thought?**
 - Behavioral Language Learning - Skinner (1957)
 - syntactic language models - Chomsky (1950)
 - computational linguistics
 - natural language processing
 - natural language understanding: complex problem
 - understanding the structure of sentences
 - understanding of the subject
 - understanding of the meaning of the sentences
 - knowledge representation
 - how to represent knowledge to let the computer reason?



Computer Engineering (1940 - today)

- **How do we build an efficient computer?**
 - AI = intelligence + artifact
 - rapid development of efficient computers
 - mutual relationship between CE and AI
 - time sharing
 - interactive interpreters
 - symbolic programming
 - functional programming



Control Theory (1948 - today)

- **How can artifacts be self-controlled?**
 - hydraulic clock with regulator (Ktisivios of Alexandria, 250 BC)
 - thermostat (Drebbel, approx. 1600)
 - modifying behavior when data changes in the environment
 - control theory and government - Weiner (1894-1964)
 - intentional behavior: error minimization mechanism
 - stochastic optimal control
 - objective function maximization
 - difference between AI and modern control theory (CT)
 - AI: discrete systems (natural language, games, vision, ...)
 - CT: continuous systems (mechanical arms, electric current, ...)
 - AI: study / combination of many new fields



A Brief History of AI



Compact History of Artificial Intelligence

1943-1955	The gestation of AI
1956	The birth of AI
1952-1969	Early enthusiasm and high expectations
1966-1973	A dose of reality
1969-1979	Knowledge-based systems
1980-today	AI becomes an industry
1986-today	The return of neural networks
1987-today	AI becomes a science
1995-today	The emergence of intelligent agents
2010-today	Deep learning, ChatGPT, ...



History of Artificial Intelligence

- **1943-1955: The gestation of AI**

- 1943: artificial neural networks (McCulloch and Pitts)
- 1949: learning method for neural synapses (Hebb)
- 1950: “Computing Machinery and Intelligence” (Turing)
 - Turing test, machine learning, genetic algorithms
- 1951: first neural computer (Minsky and Edmonds)



History of Artificial Intelligence ++

- **1956: The birth of AI**

- Two-month workshop at Dartmouth College
 - John McCarthy, Marvin Minsky, Claude Shannon, Nathaniel Rochester, Trenchard More, Arthur Samuel, Ray Solomonoff, Oliver Selfridge, Allen Newell and Herbert Simon
- the Logic Theorist Program (Newell and Simon)
 - proofs for Chapter 2 of Principia Mathematica
- adoption of the term “Artificial Intelligence” (McCarthy)
- AI is studied as a separate scientific field



History of Artificial Intelligence ++

- **1952-1969: Early enthusiasm and high expectations**
 - 1952: a program for Checkers that learns (Samuel)
 - 1958: LISP programming language, Advice Taker (McCarthy)
 - 1959: Geometry Theorem Prover (Gelernter)
 - 1959: AI Lab at MIT (Minsky and McCarthy)
 - 1961: General Problem Solver (Newell and Simon)
 - 1960: Adalines neural networks (Widrow)
 - 1962: Perceptrons (Rosenblatt)
 - 1963: AI Lab at Stanford University (McCarthy)
 - 1963: AI microworlds
 - 1969: Shakey robot at Stanford



History of Artificial Intelligence ++

- **1966-1973: A dose of reality**
 - poor results, “curse of dimensionality”
 - 1966: automatic translation funding in the United States ceases
 - typical example: *"the spirit is willing, but the flesh is weak"*
 - automatic translation: *"the vodka is good, but the meat is rotten"!*
 - 1969: rejection of perceptrons (Minsky and Papert)
 - 1973: AI funding for research in Great Britain ceases
- **1969-1979: Knowledge-based systems**
 - 1969: finding molecular structure - DENDRAL (Buchanan et al.)
 - 1970: Diagnosis of Blood Infections - MYCIN (Stanford)
 - 1970: the Prolog language (logical programming)
 - 1975: frames as a method of representing knowledge



History of Artificial Intelligence ++

- **1980-present: AI becomes an industry**
 - 1982: R1 expert system for DEC sales (McDermott)
 - 40 installed expert systems
 - savings of \$40,000,000 per year (1986)
 - 1981: "Fifth Generation" program (Japan)
 - followed by "responses" from the US (MCC) and the UK
 - AI ?? No !!! It's... IKBS (Intelligent Knowledge-Based Systems)
 - end of 80's: the "winter" of AI due to many promises
- **1986-present: The return of neural networks**
 - 1982: Statistical engineering techniques for the analysis of neural network properties (Hopfield)
 - mid-1980s: Back-propagation (1969), applications in Computer Science and Psychology [Parallel Distributed Processing, 1986]



History of Artificial Intelligence ++

- **1987-present: AI becomes a science**
 - revolution in content and methodology
 - achievements are based on rigorous theories
 - acceptance only after proof or rigorous experimentation
 - cooperation with other fields
 - Hidden Markov Models, Data mining, Bayesian networks, Machine learning, Game theory
- **1995-present: The emergence of intelligent agents**
 - consolidation of all sub-sectors
- **2010-present: Deep Learning**
 - deep neural networks
- **2022-present: ChatGPT**
 - large language models



More AI Successes

- **Medical Diagnosis and Decision Support Systems**

- Expert systems, Bayesian network-based, neural network-based systems

- **Logistics Planning**

- During the Persian Gulf War (1991), the United States used the D.A.R.T. (Dynamic Analysis and Replanning Tool) for automated planning and scheduling of actions for their transfers.

- **Planning**

- Remote Agent: creating plans for NASA Exploration Vehicles (2000)

- **Games**

- AlphaGo defeats world Go champion in Korea (2016)

- **Robotics**

- Stanley autonomously drove 211 km in the desert in 10 hours (2005)

- **Crossword Puzzles**

- Proverb: automatic crossword puzzle solving with standard cues (1999)

- **Protein Folding**

- AlphaFold: a solution to a 50-year-old grand challenge in biology (2020)



Moral Questions

- **AI in the Defense industry: Killer Robots ?**
 - Need for principles to guide / restrain AI weapons research
- **Robots taking over...**
 - *The Matrix...*
 - Stephen Hawking: *“AI could spell the end of human race”*
- **Yet...**
 - “We have to colonize another planet within 100 years”
(again Stephen Hawking)
 - *And thus we need AI / Robots!...*
- **In which case...**
 - *“need for a robots civil rights movement”* (Spielberg’s AI movie)



An Experts' Debate on Intelligence

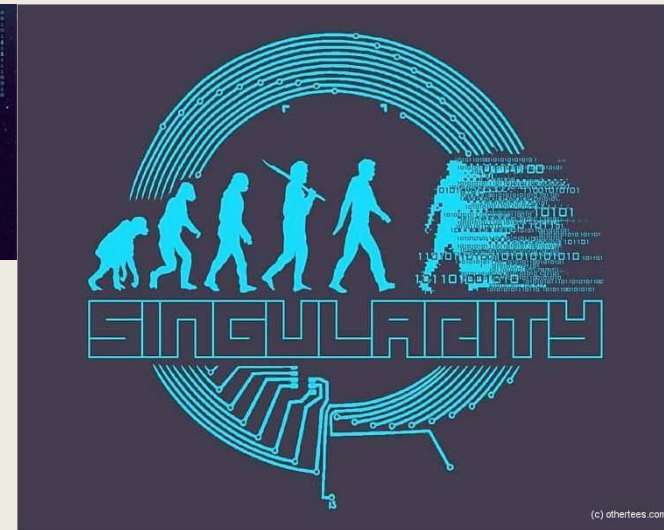
- Noam Chomsky vs. statistical / probabilistic AI
 - Chomsky's critique: *"The field's heavy use of statistical techniques to pick regularities in masses of data is behavioral in nature, and unlikely to yield the explanatory insight that science ought to offer. The "new AI" -- focused on using statistical learning techniques to better mine and predict data - - is unlikely to yield general principles about the nature of intelligent beings or about cognition."*

Brains, Minds and Machines Symposium, MIT, May 2011

- A response by Stuart Russell: *"AI's new methods and definition of progress is not far off from what happens in the other sciences."*



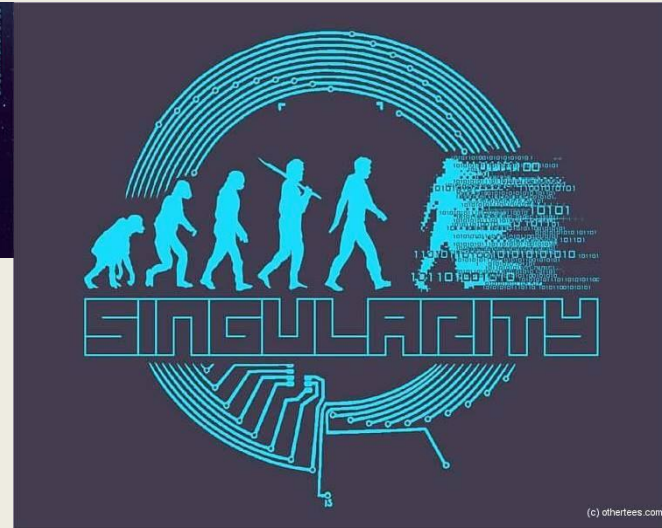
Will a “singularity” emerge?



Maybe we are not “very” far away from building AI
that surpasses human intelligence...
...maybe it’s just the next evolutionary stage...
...maybe not so bad given how human intelligence evolves...



Will a “singularity” emerge?



Maybe we are not “very” far away from building AI that surpasses human intelligence...

...maybe circa 2050 (in ~30 years) *

...not necessarily a bad thing...

...unless we are ready to tackle it ; we should at least *anticipate it*

* maybe not...maybe in 500 years...many questions regarding the computational power necessary and the energy that an AGI (artificial general intelligence) system would require are still open; thus, no good estimate / prediction exists!



The Asilomar AI principles

- 1) **Research Goal:** The goal of AI research should be to create not undirected intelligence, but beneficial intelligence.
- 2) **Research Funding:** Investments in AI should be accompanied by funding for research on ensuring its beneficial use, including thorny questions in computer science, economics, law, ethics, and social studies, such as:
 - How can we make future AI systems highly robust, so that they do what we want without malfunctioning or getting hacked?
 - How can we grow our prosperity through automation while maintaining people's resources and purpose?
 - How can we update our legal systems to be more fair and efficient, to keep pace with AI, and to manage the risks associated with AI?
 - What set of values should AI be aligned with, and what legal and ethical status should it have?



The Asilomar AI principles

- 3) **Science-Policy Link:** There should be constructive and healthy exchange between AI researchers and policy-makers.
- 4) **Research Culture:** A culture of cooperation, trust, and transparency should be fostered among researchers and developers of AI.
- 5) **Race Avoidance:** Teams developing AI systems should actively cooperate to avoid corner-cutting on safety standards.
- 6) **Safety:** AI systems should be safe and secure throughout their operational lifetime, and verifiably so where applicable and feasible.



The Asilomar AI principles

- 7) **Failure Transparency:** If an AI system causes harm, it should be possible to ascertain why.
- 8) **Judicial Transparency:** Any involvement by an autonomous system in judicial decision-making should provide a satisfactory explanation auditable by a competent human authority.
- 9) **Responsibility:** Designers and builders of advanced AI systems are stakeholders in the moral implications of their use, misuse, and actions, with a responsibility and opportunity to shape those implications.
- 10) **Value Alignment:** Highly autonomous AI systems should be designed so that their goals and behaviors can be assured to align with human values throughout their operation.



The Asilomar AI principles

- 11) **Human Values:** AI systems should be designed and operated so as to be compatible with ideals of human dignity, rights, freedoms, and cultural diversity.
- 12) **Personal Privacy:** People should have the right to access, manage and control the data they generate, given AI systems' power to analyze and utilize that data.
- 13) **Liberty and Privacy:** The application of AI to personal data must not unreasonably curtail people's real or perceived liberty.
- 14) **Shared Benefit:** AI technologies should benefit and empower as many people as possible.
- 15) **Shared Prosperity:** The economic prosperity created by AI should be shared broadly, to benefit all of humanity.



The Asilomar AI principles

- 16) **Human Control:** Humans should choose how and whether to delegate decisions to AI systems, to accomplish human-chosen objectives.
- 17) **Non-subversion:** The power conferred by control of highly advanced AI systems should respect and improve, rather than subvert, the social and civic processes on which the health of society depends.
- 18) **AI Arms Race:** An arms race in lethal autonomous weapons should be avoided.
- 19) **Capability Caution:** There being no consensus, we should avoid strong assumptions regarding upper limits on future AI capabilities.



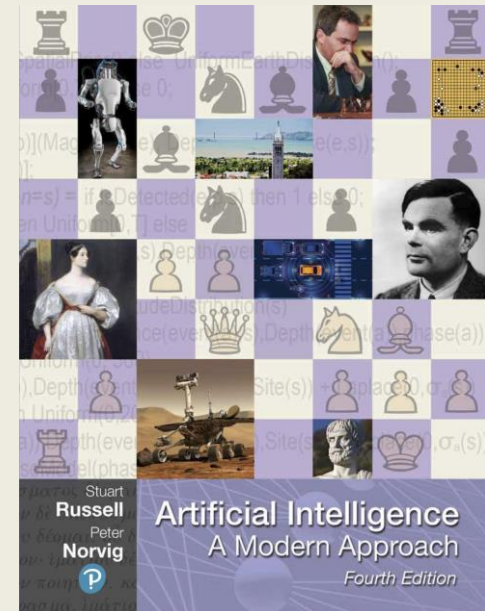
The Asilomar AI principles

- 20) **Importance:** Advanced AI could represent a profound change in the history of life on Earth, and should be planned for and managed with commensurate care and resources.
- 21) **Risks:** Risks posed by AI systems, especially catastrophic or existential risks, must be subject to planning and mitigation efforts commensurate with their expected impact.
- 22) **Recursive Self-Improvement:** AI systems designed to recursively self-improve or self-replicate in a manner that could lead to rapidly increasing quality or quantity must be subject to strict safety and control measures.
- 23) **Common Good:** Superintelligence should only be developed in the service of widely shared ethical ideals, and for the benefit of all humanity rather than one state or organization.



Suggested readings

- Artificial Intelligence: A Modern Approach,
Stuart Russell and Peter Norvig,
4th edition, Prentice Hall, 2020.
 - Chapter 1



Suggested viewings and listenings

- [Stephen Wolfram on Intelligence, AI, the meaning of life, and more](#)
- [Michael Wooldridge on “The Evolution of AI”](#)
- [Adnan Darwiche on "AI Education" or "AI vs Deep Learning”](#)
- [Michael Wooldridge on “An Introduction to Multiagent Systems”](#)



BME-16 Artificial Intelligence and Medical Decision Support Systems

Informed and Uninformed Intelligent Search

Michail G. Lagoudakis

Intelligent Systems Laboratory

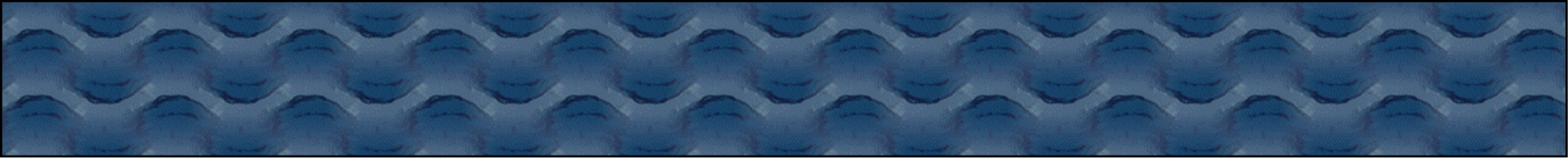
School of Electrical and Computer Engineering

Technical University of Crete

Chania, Crete, Greece

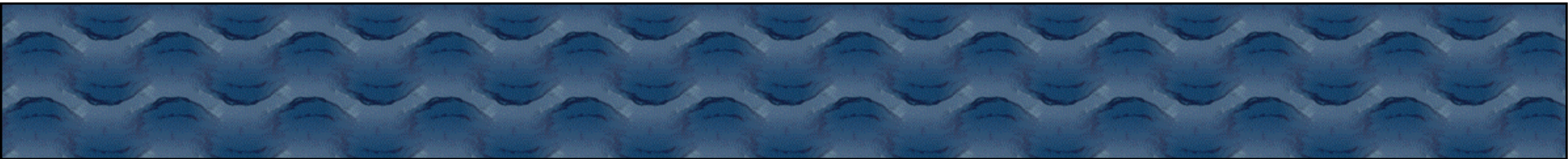


intelligence
INTELLIGENT SYSTEMS LABORATORY

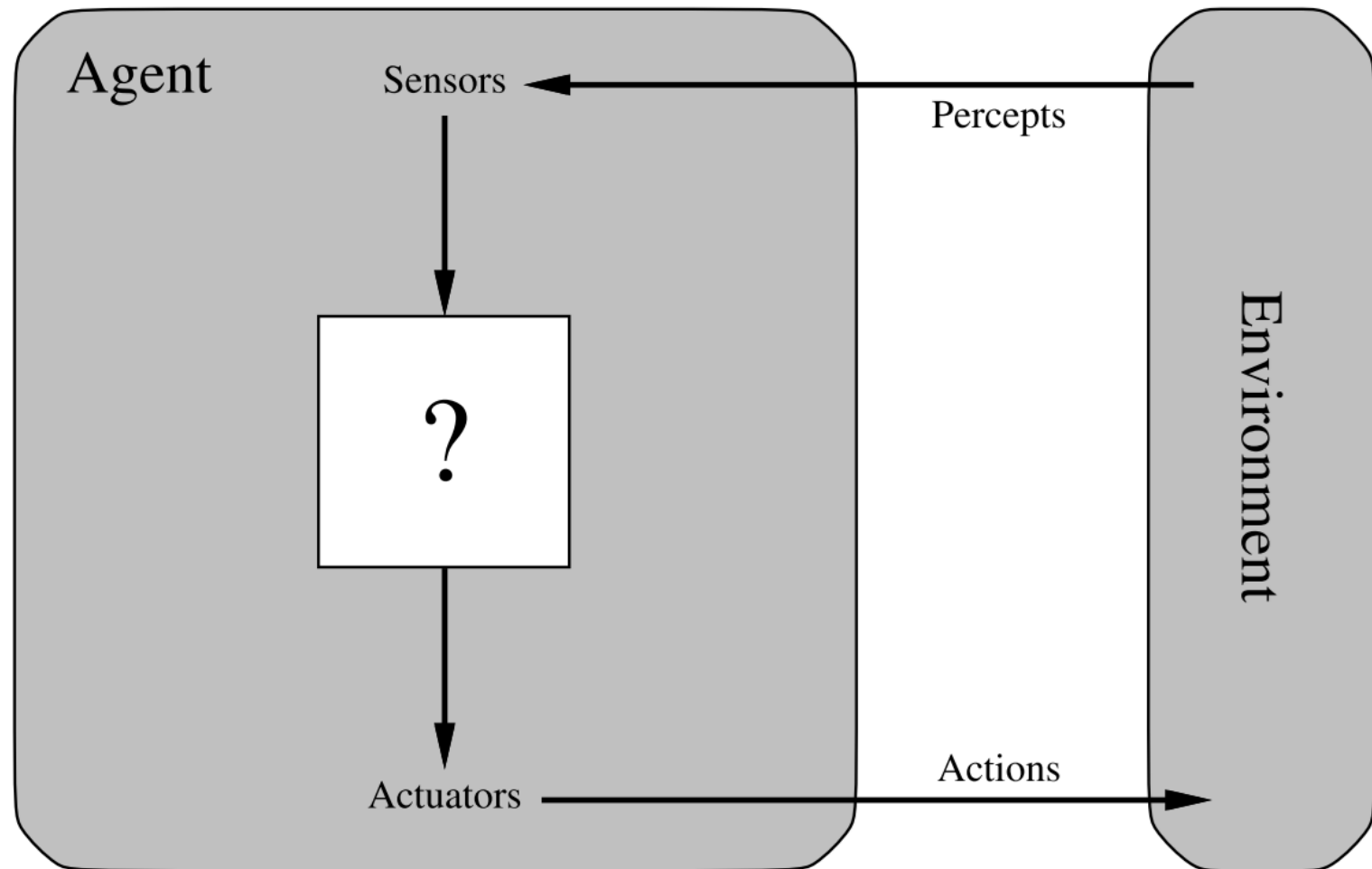


Agents and Environments

a unified view



Agent and Environment



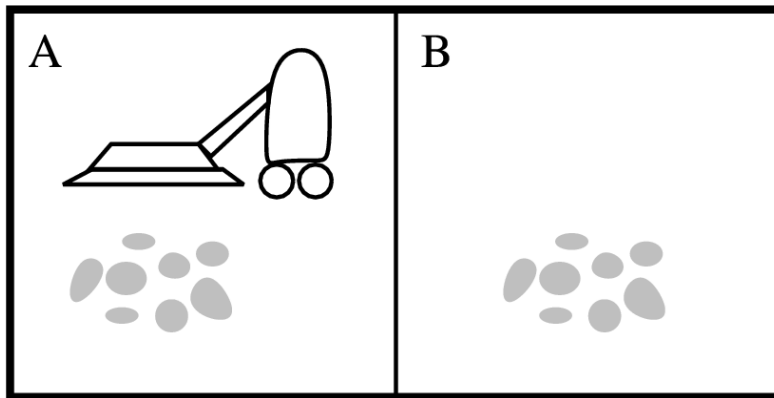
Agents

- **Agent**
 - from the Latin *agere* = act
 - entity with **sensors** and **actuators**
 - **perceives** the environment and **affects** the environment
- **Human agents**
 - sensors: eyes, ears, nose, skin, ...
 - actuators: arms, hands, legs, mouth, ...
- **Robotic agents**
 - sensors: cameras, laser, infrared, encoders, ...
 - actuators: motors, joints, wheels, tracks, ...
- **Software agents**
 - sensors: keyboard, mouse, files, network packages, ...
 - actuators: print on screen, write on disk, ...

Agent Structure

- **Percepts**
 - the information from the sensors at any time point
- **Percept sequence**
 - the full history of percepts temporally ordered
- **Actions**
 - the effects of the actuators at any time point
- **Agent Function**
 - mapping from percept sequences to actions
 - definition: a huge look-up table
 - implementation: a computer program

A Simple Agent



Percepts

Room (A/B), Status (Dusty/Clean)

Actions

Left, Right, Suck

Percept Sequence	Action
[A, Clean]	Right
[A, Dusty]	Suck
[B, Clean]	Left
[B, Dusty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [B, Dusty]	Suck
...	...

Program Vacuum_Cleaner

...

if (Status==Dusty) **then** Suck

if (Room==A) **then** Right

if (Room==B) **then** Left

...

Rational Agents

- **Performance measure**
 - objective definition of successful agent behavior
 - design principle: **what** we want, not **how** it can be done
- **Rational agent**
 - chooses actions that maximize the performance measure
 - uses the percept sequence and the embedded knowledge
- **Rationality**
 - does not mean omniscience or perfection
 - maximization of the *expected* performance

Task Environment

- **Task environment**
 - environment = problem, agent = solution
 - PEAS description
 - Performance measure, Environment, Actuators, Sensors
- **Example: Agent for Internet Shopping**

Measure	Environment	Actuators	Sensors
price quality fitness efficiency	www e-shops comparison stores couriers	searching following links filling forms printing receipts	HTML pages product codes price offers payment methods

Environment Properties

- **Observability**
 - fully observable
 - partially observable
- **Predictiveness**
 - deterministic
 - stochastic
 - strategic
- **Dependency**
 - episodic
 - sequential

Environment Properties

- **Variability**
 - static
 - dynamic
 - semi-dynamic
- **Description**
 - discrete
 - continuous
- **Multitude**
 - single-agent
 - multi-agent
 - competitive or cooperative

Range of Environments

□ Hard environments

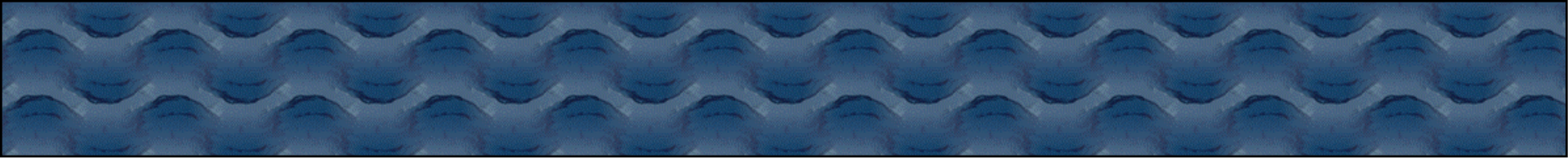
- partially observable
- stochastic
- sequential
- dynamic
- continuous
- multi-agent

□ Easy environments

- fully observable
- deterministic
- episodic
- static
- discrete
- single-agent

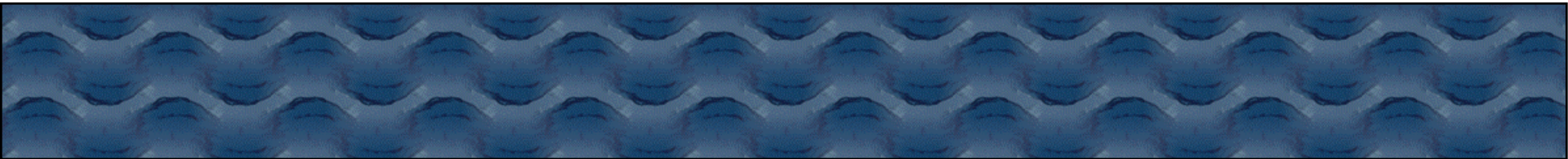
Environment Examples

Crossword Puzzles	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a timer	Fully	Strategic	Sequential	Semi Dynamic	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi Driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical Diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Image Analysis	Fully	Deterministic	Episodic	Semi Dynamic	Continuous	Single
Robotic Part Picker	Partially	Stochastic	Episodic	Dynamic	Continuous	Single
Plant Controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English Tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi



Problem-Solving Agents

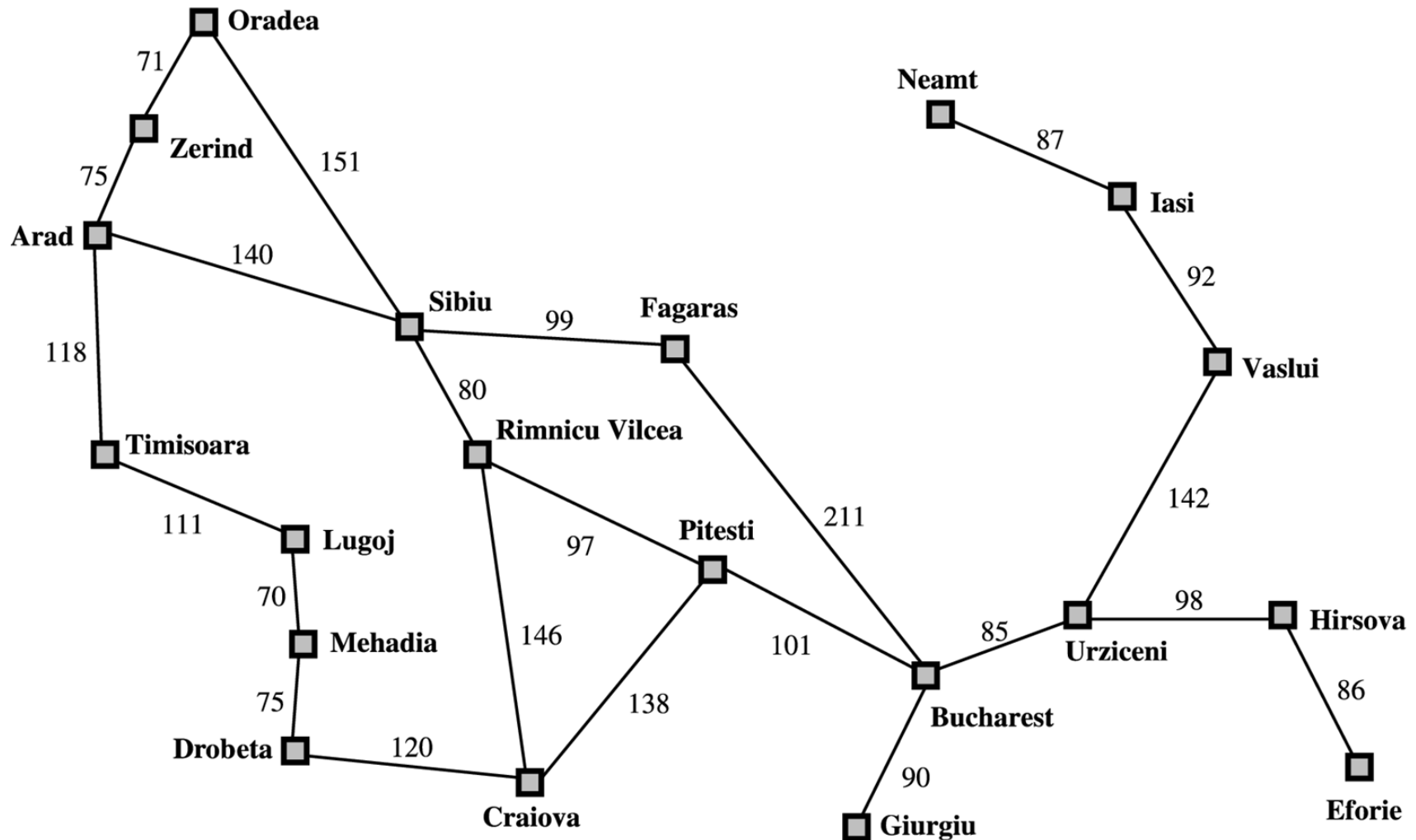
useful agents



Problem-Solving Agents

- **Problem solving**
 - the process of finding a solution for some problem
- **Problem formulation**
 - definition of states and actions
- **Goal formulation**
 - definition of a subset of states considered as solutions
- **Action selection**
 - action sequences (as opposed to instant actions)
- **Search**
 - the process of finding an appropriate action sequence that leads to a solution

Example: Routes in Romania



Problem-Solving Agent Code

function Simple-Problem-Solving-Agent(*percept*) **returns** *action*

inputs: *percept*, some percept

static: *sequence*, an action sequence, initially empty
state, description of the current state of the world
goal, some goal, initially empty
problem, a formulation of some problem

state $\leftarrow$ Update-State(*state*, *percept*)

if *sequence* is empty **then do**

goal $\leftarrow$ Formulate-Goal(*state*)

problem $\leftarrow$ Formulate-Problem(*state*, *goal*)

sequence $\leftarrow$ Search(*problem*)

action $\leftarrow$ First(*sequence*)

sequence $\leftarrow$ Rest(*sequence*)

return *action*

Conventions

- **Environment**

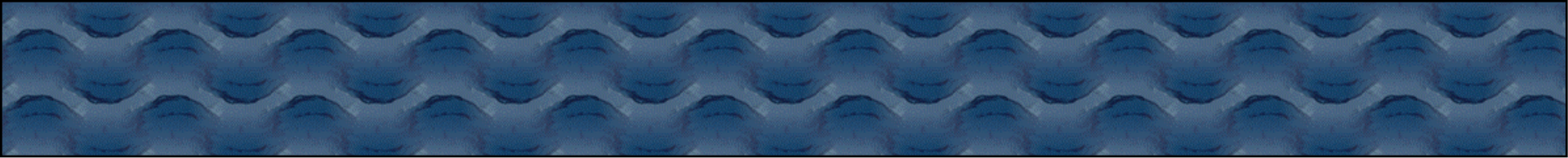
- static (no changes during the search)
- observable (known initial state)
- discrete (finite number of actions)
- deterministic (predictable transitions)

- **Agent**

- open-loop control
- action execution without perception (sensor-less)

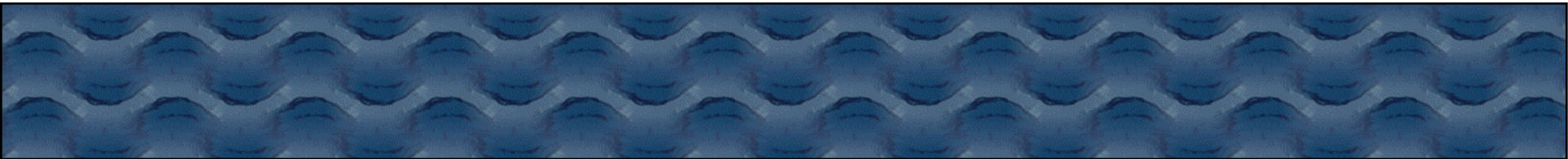
- **Abstraction**

- elimination of unnecessary details from the representation



Search

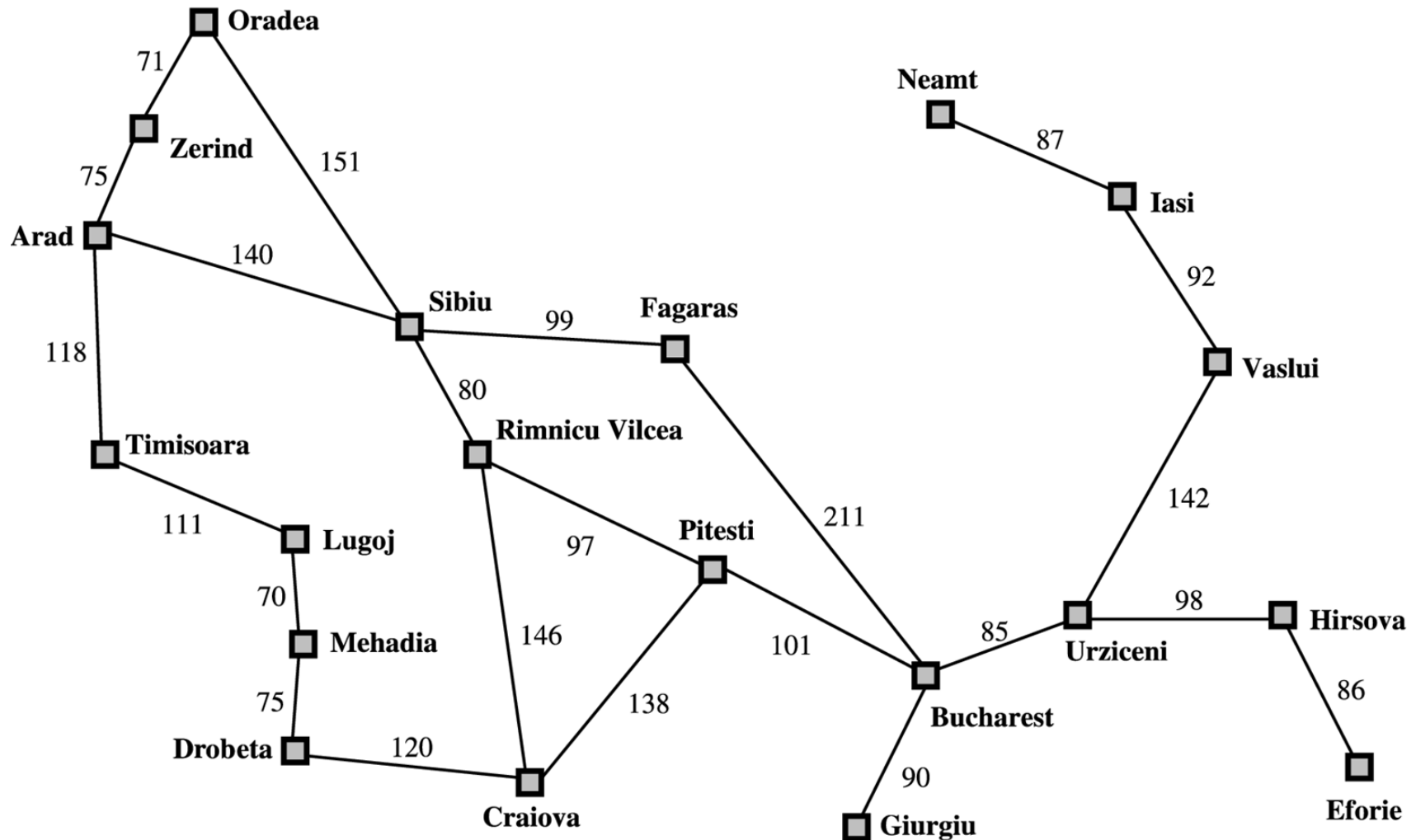
looking for something



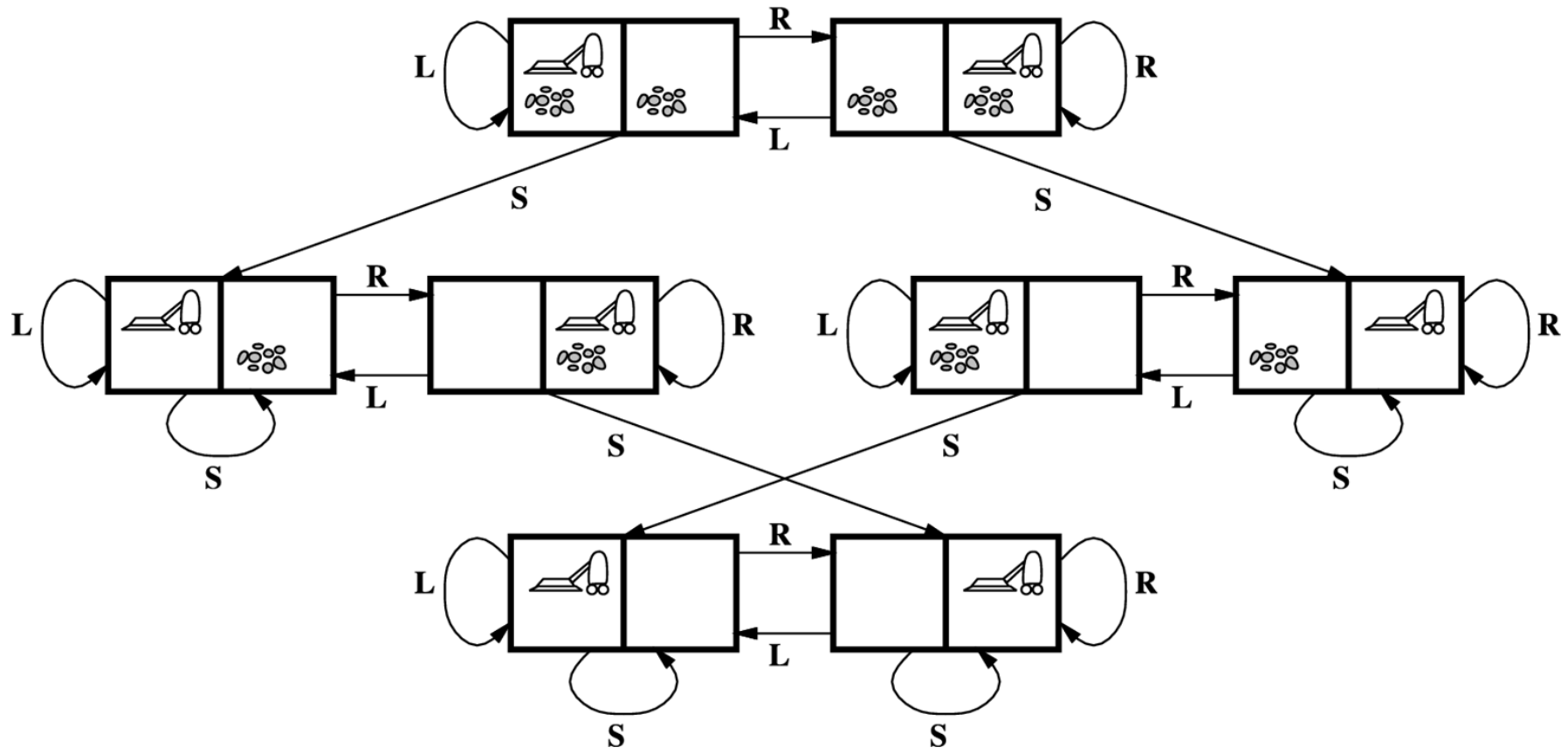
Problem Formulation

- **Initial state**
 - current state of environment and agent
- **State space**
 - all states accessible from the initial one
- **Successor function**
 - for each state, gives the valid actions and successor states
- **Goal test**
 - decides whether the goal is achieved in a state
- **Cost function**
 - determines the cost for each step of a sequence of actions

Example: Routes in Romania



The Vacuum Cleaner Micro-World



3x3 Tile Puzzle

7	2	4
5		6
8	3	1

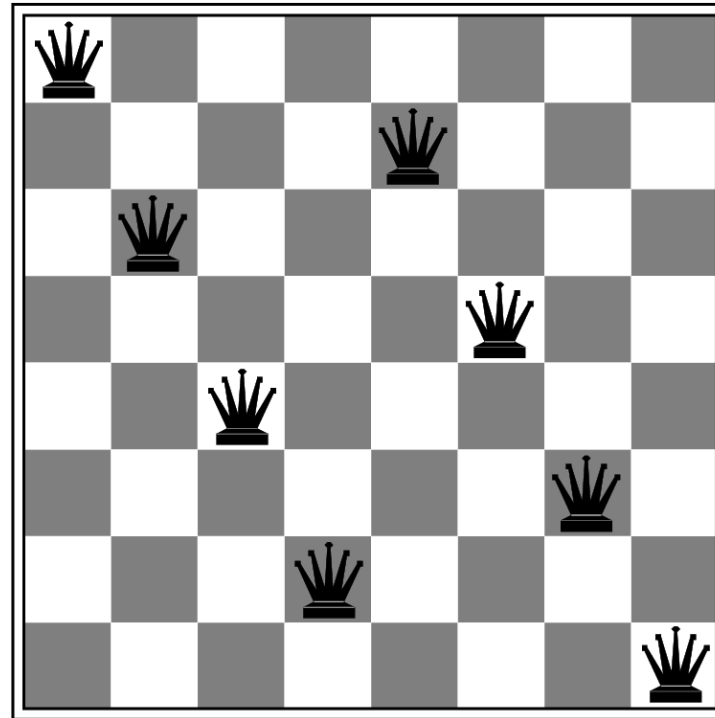
Start State

	1	2
3	4	5
6	7	8

Goal State

- generic class of *NP*-hard problems
- 4×4 puzzle: $16!/2 = 1,300,000,000,000$ states (easy)
- 5×5 puzzle: $25!/2 = 10^{25}$ states (hard)

The 8-Queen Problem



- incremental formulation (placing one queen at a time)
- full-state formulation (all queens placed on board)
- generalized problem: n queens on $n \times n$ board

Real-World Search Problems

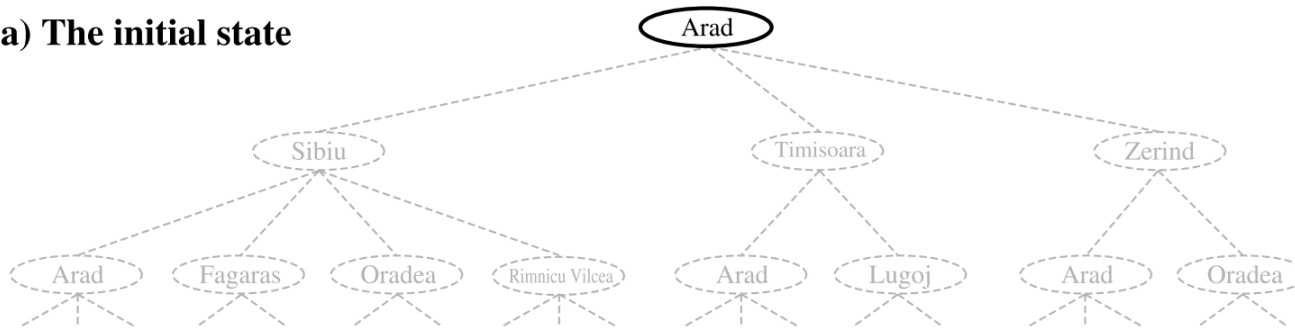
- **Routing problem**
- **Touring problem**
- **Traveling salesman problem**
- **VLSI layout**
- **Robot navigation**
- **Automatic assembly**
- **Protein synthesis**
- **Internet search**
- **...**

Search Tree

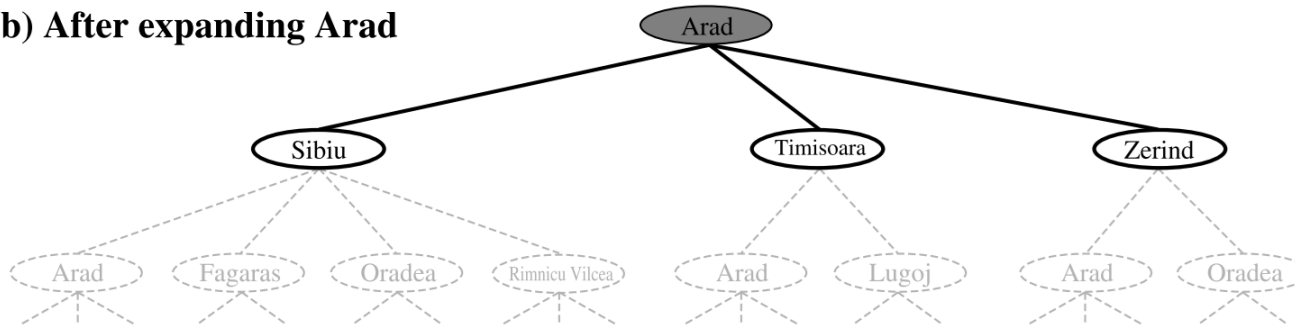
- **Search tree**
 - systematic generation of paths within the state space
 - search node: corresponds to a single state
 - root node: the node of the initial state
 - node expansion: generation of the successors of a node
 - generation method: successor function
 - search strategy: choosing where to expand the tree
- **Notes**
 - a node is different from a state
 - the search tree is different from the state space

Search Tree

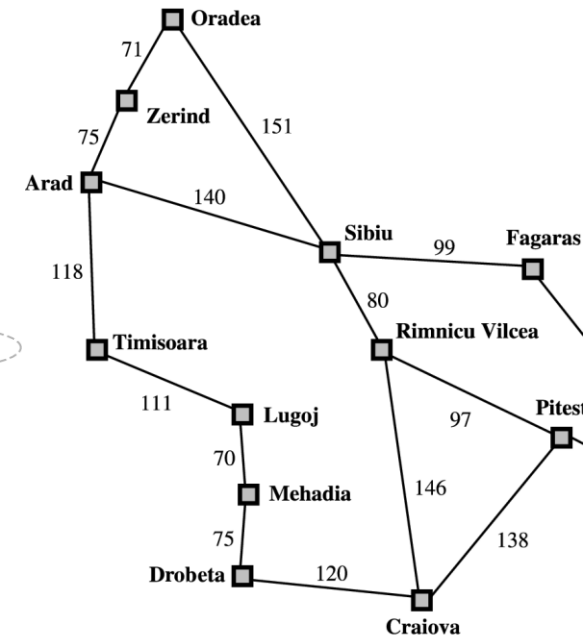
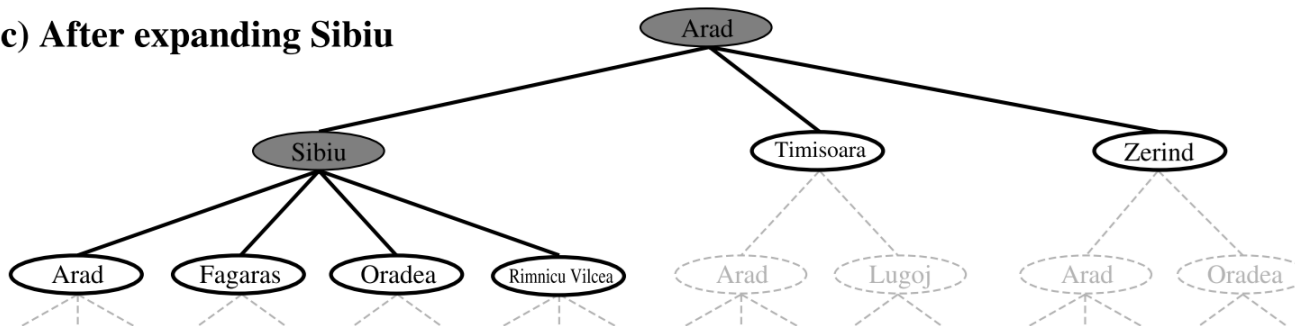
(a) The initial state



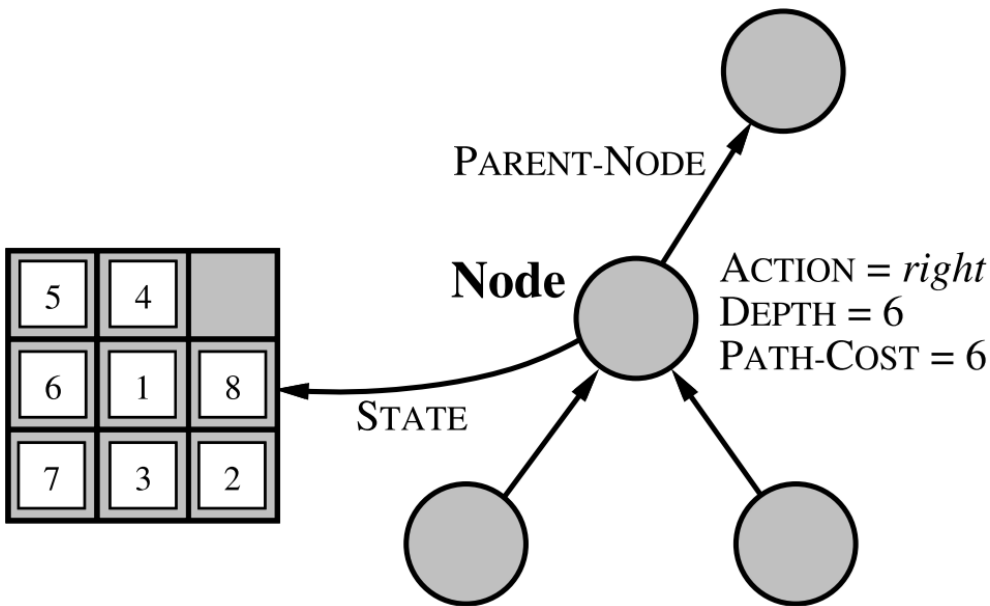
(b) After expanding Arad



(c) After expanding Sibiu



Search Node



■ Node: data structure

- state
- parent node
- action
- depth
- path cost

■ Fringe of nodes

- set of pending nodes
- waiting for expansion
- the leaves of the search
- implementation using some kind of queue

Informal Search Algorithm

function Tree-Search(*problem*, *strategy*) **returns** a solution or failure

 initialization of the search tree using the initial state of the *problem*

loop do

if there are no candidate nodes for expansion

then return failure

 choose a node from the fringe for expansion, according to the *strategy*

if the node contains a goal state

then return the corresponding solution

else expand the node and add the successor nodes in the search tree

Formal Search Algorithm (I)

function Tree-Search(*problem*, *fringe*) **returns** a solution or failure

fringe $\leftarrow$ InsertNode(MakeNode(InitialState[*problem*]), *fringe*)

loop do

if EmptyQueue(*fringe*) **then return** failure

node $\leftarrow$ RemoveFirstNode(*fringe*)

if GoalTest[*problem*] applied on State[*node*] succeeds

then return Solution(*node*)

fringe $\leftarrow$ InsertNodes(Expand(*node*, *problem*), *fringe*)

Formal Search Algorithm (II)

function Expand(*node*, *problem*) **returns** a set of nodes

successors $\leftarrow$ empty set

for each $\langle action, effect \rangle$ in Successor-Function[*problem*](State[*node*]) **do**

s $\leftarrow$ new Node

State[*s*] $\leftarrow$ *effect*

Parent-Node[*s*] $\leftarrow$ *node*

Action[*s*] $\leftarrow$ *action*

Path-Cost[*s*] $\leftarrow$ Path-Cost[*node*] + Step-Cost(*node*, *action*, *s*)

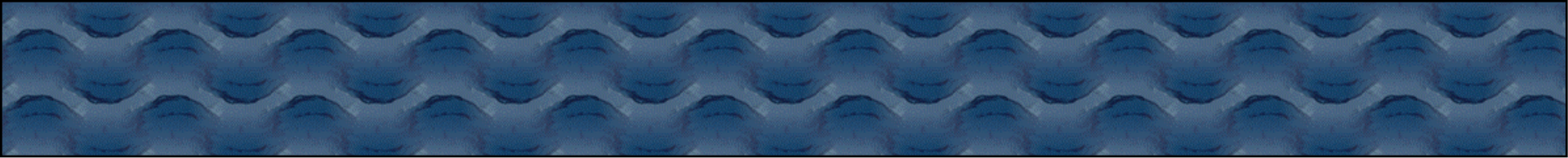
Depth[*s*] $\leftarrow$ Depth[*node*] + 1

addition of *s* to the set *successors*

return *successors*

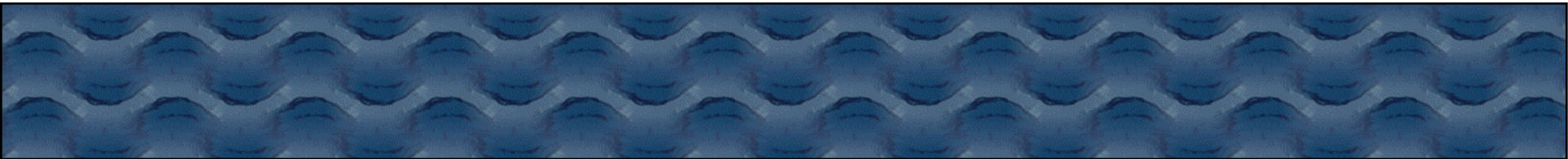
Measuring Search Performance

- **Completeness**
 - does it guarantee to find a solution, if one exists?
- **Optimality**
 - if it finds a solutions, does it guarantee to be an optimal one?
- **Time complexity**
 - how much time is required to find a solution?
- **Space complexity**
 - how much space (memory) is required to find a solution?
- **Problem size**
 - branching factor b (maximum number of children of a node)
 - depth of the closest solution d
 - maximum length of any path m

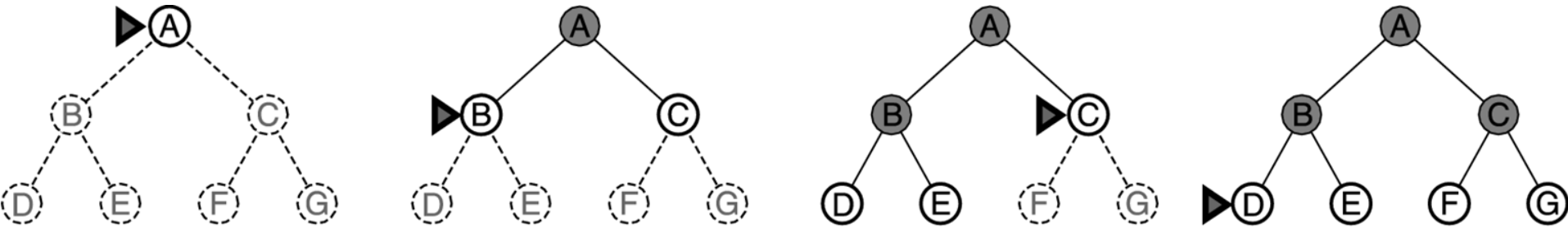


Uninformed or Blind Search

search with no guidance



Breadth-First Search



- expand the node with the least depth in the current fringe
- implementation with a FIFO queue (First In, First Out)
- complete, optimal (under conditions), exponential complexity $O(b^{d+1})$

Depth (d)	Nodes ($b=10$)	Time (10000 nodes/second)	Memory (1000 bytes/node)
2	1100	0.11 seconds	1 megabyte
4	111100	11 seconds	106 megabytes
6	10^7	19 minutes	10 gigabytes
8	10^9	31 hours	1 terabyte
10	10^{11}	129 days	101 terabytes
12	10^{13}	35 years	10 petabytes
14	10^{15}	3523 years	1 exabyte

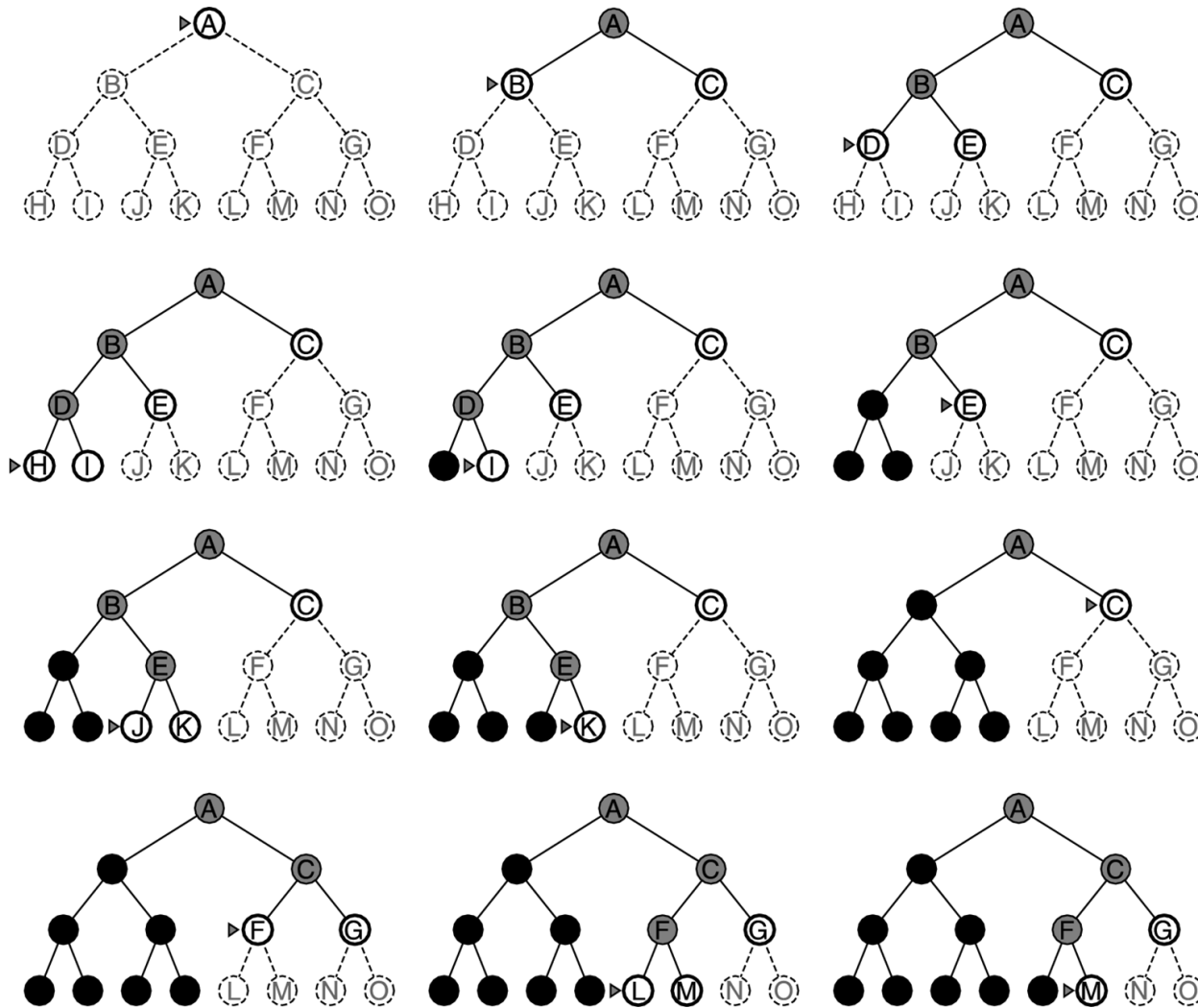
Uniform Cost Search

- variation of breadth-first search
- expand the node with the least path cost
- fringe: nodes with (almost) uniform cost
- complete, if the cost of each step is always positive
- optimal, if the cost of each step is always positive
- exponential complexity
 - depends on the optimal cost
 - depends on the minimum cost of each step
- possibly better, possibly worse performance

Depth-First Search

- expand the node with the maximum depth in the current fringe
- backtrack to the just previous node
- implementation with a LIFO queue (Last In, First Out) or a stack
- it is not complete (infinite-length paths)
- it is not optimal (proceeds to depth independently of cost)
- linear space complexity
 - improve with backtracking
 - improve with state change (requires the ability of undo)
- exponential time complexity

Depth-First Search



Depth-Limited Search

- variation of depth-first search
- cut-off expansion after a pre-determined depth
- heuristic determination of depth limit (graph diameter)
- advantage: elimination of infinite-length paths
- disadvantage: incompleteness
 - the solution may be located a little deeper
- disadvantage: suboptimality
 - an optimal solution may be located a little deeper

Iterative-Deepening Search

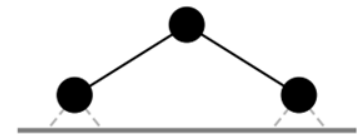
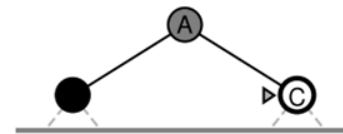
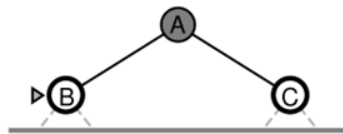
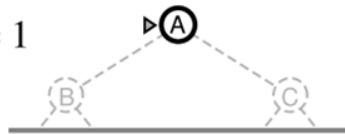
- smart management of depth-limited search
- incremental increase of depth limit ($L = 0, 1, 2, 3, \dots$)
- combines the advantages of depth-first and breadth-first search
- complete
- optimal (under conditions)
- space complexity $O(bd)$
 - linear, way less than the space complexity of breadth-first search!
- time complexity
 - exponential, but not worse $(d)b + (d-1)b^2 + \dots + (1)b^d = O(b^d)$!!!!
- variation: iterative-lengthening search

Iterative Deepening

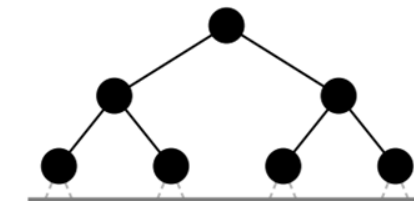
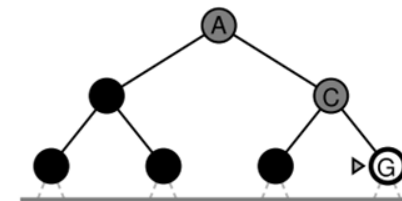
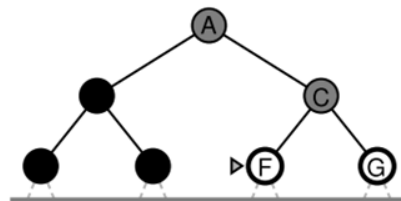
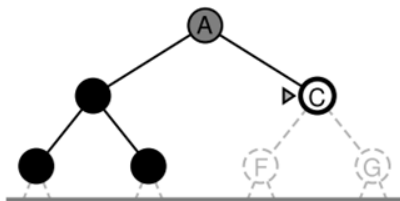
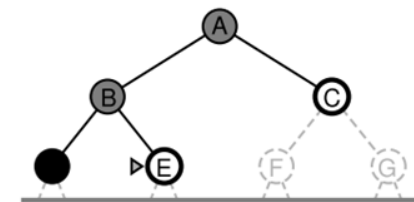
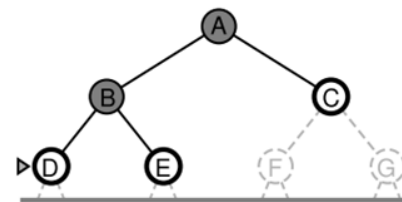
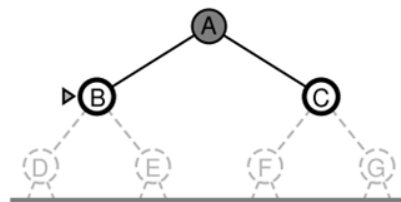
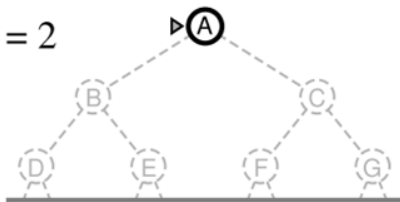
Limit = 0



Limit = 1

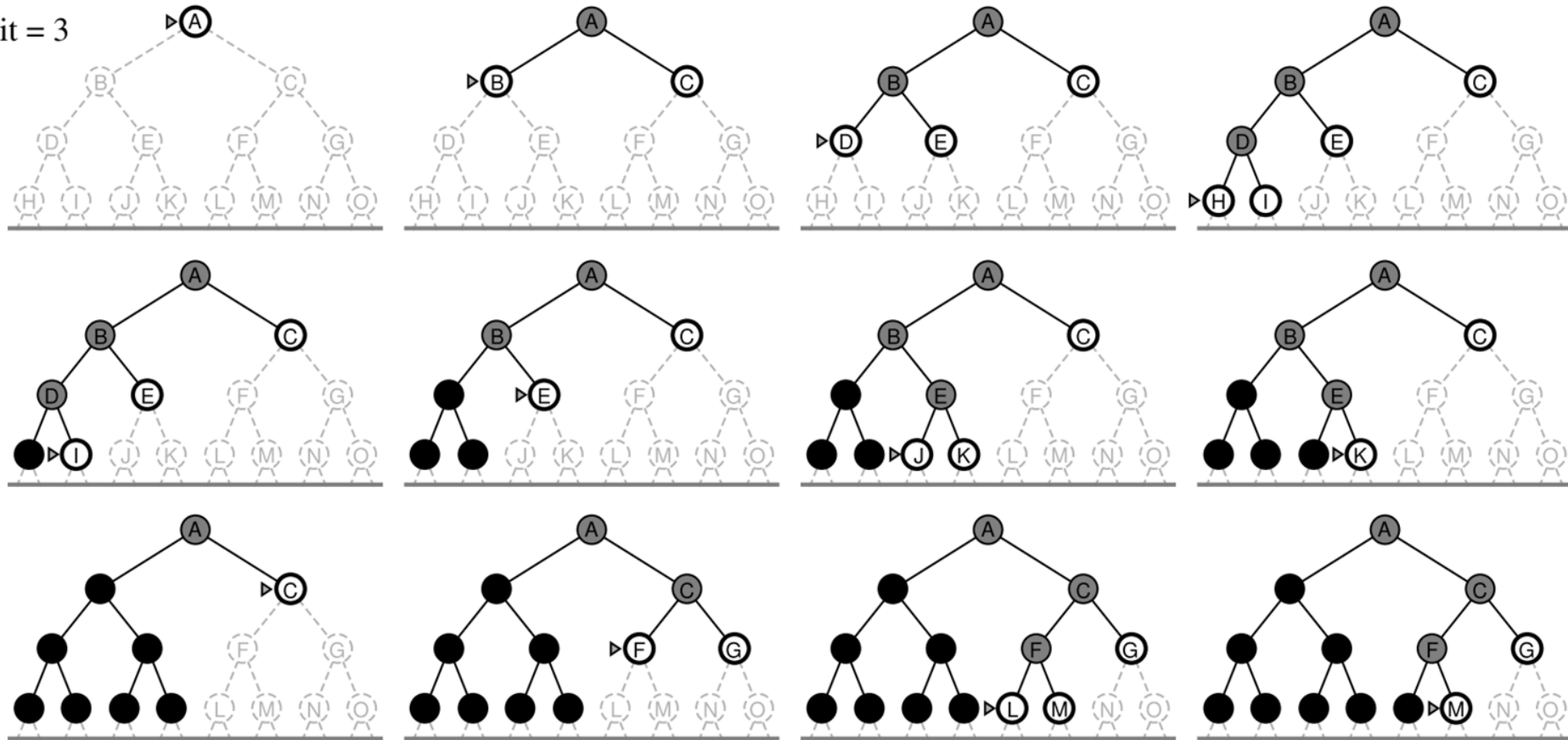


Limit = 2

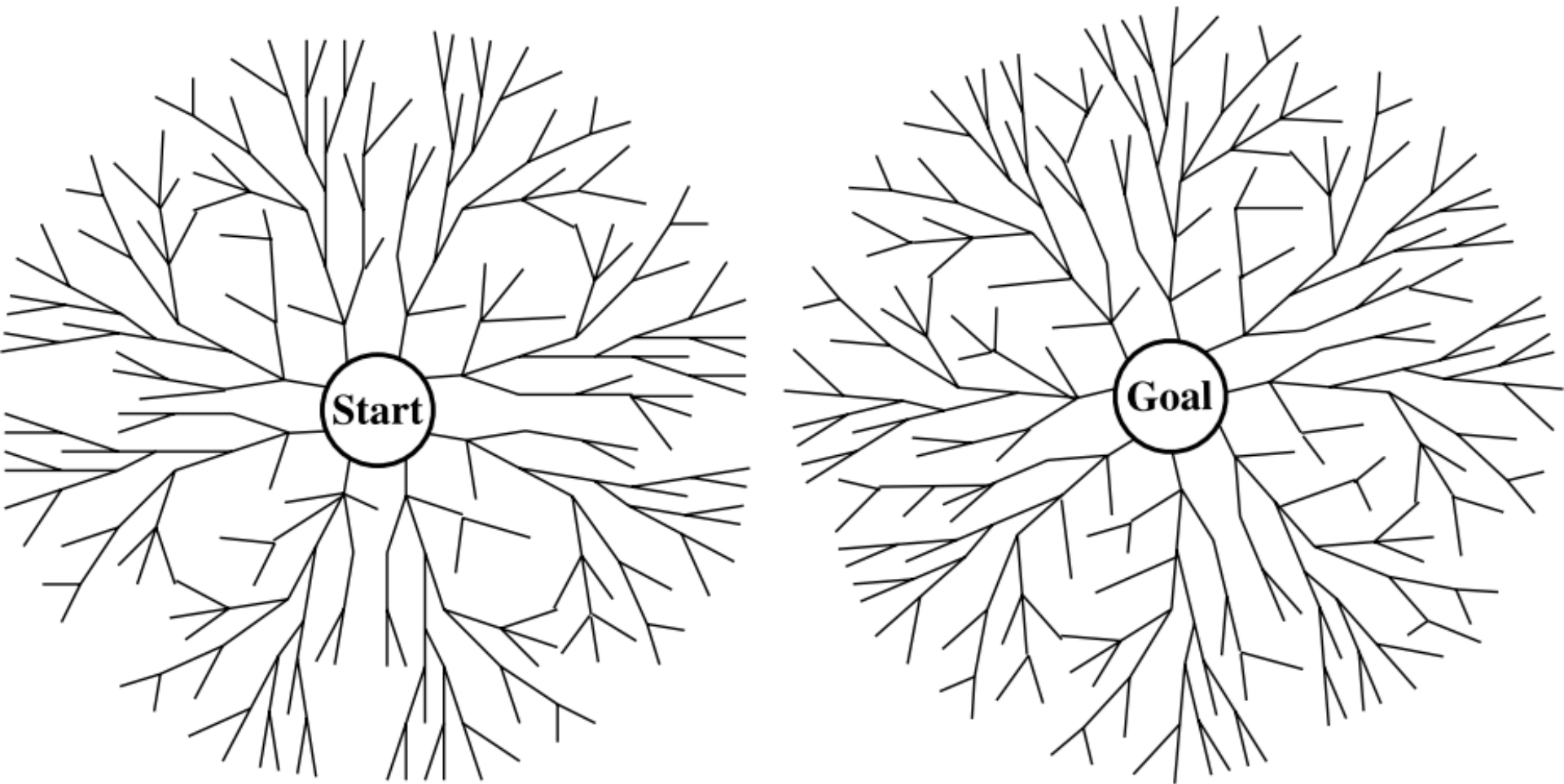


Iterative Deepening

Limit = 3

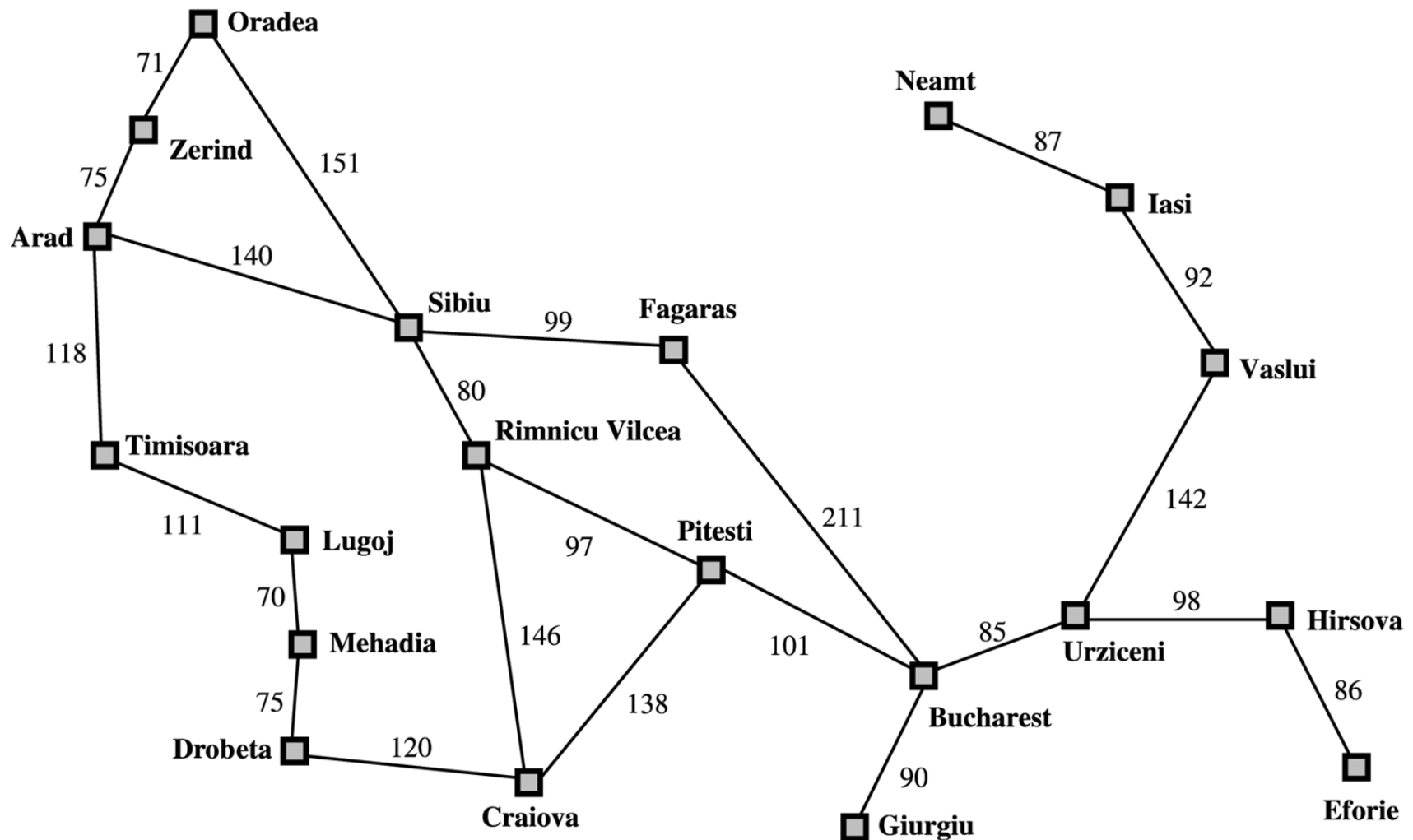


Bidirectional Search



$$b^{d/2} + b^{d/2} \ll b^d$$

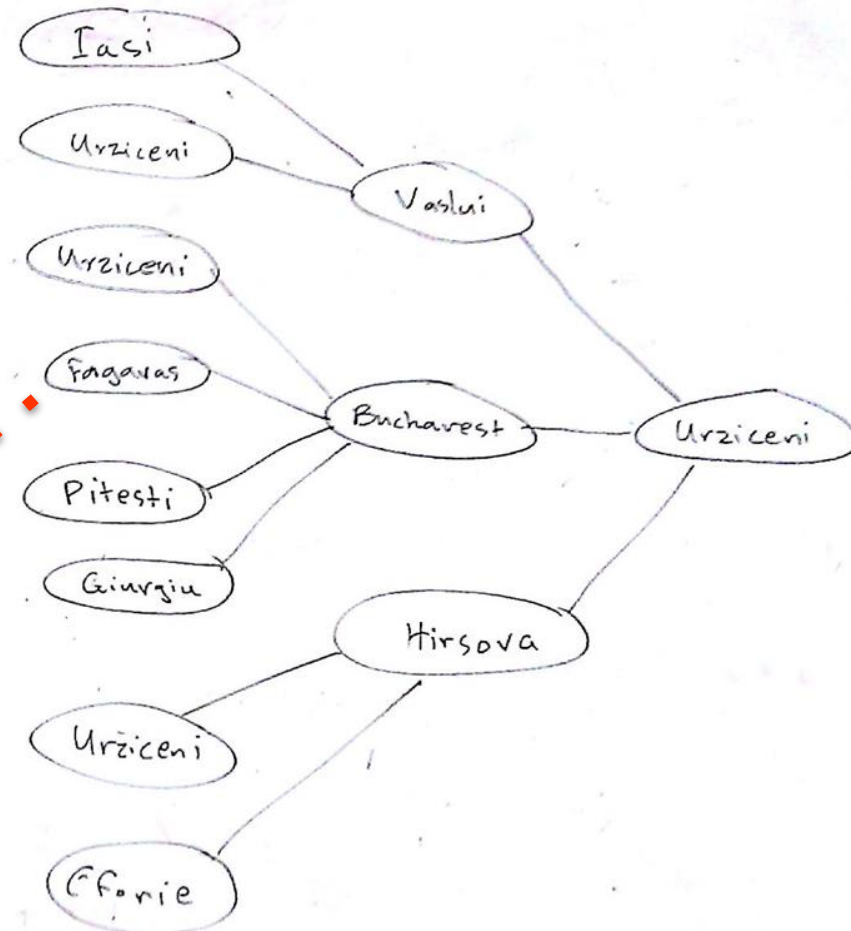
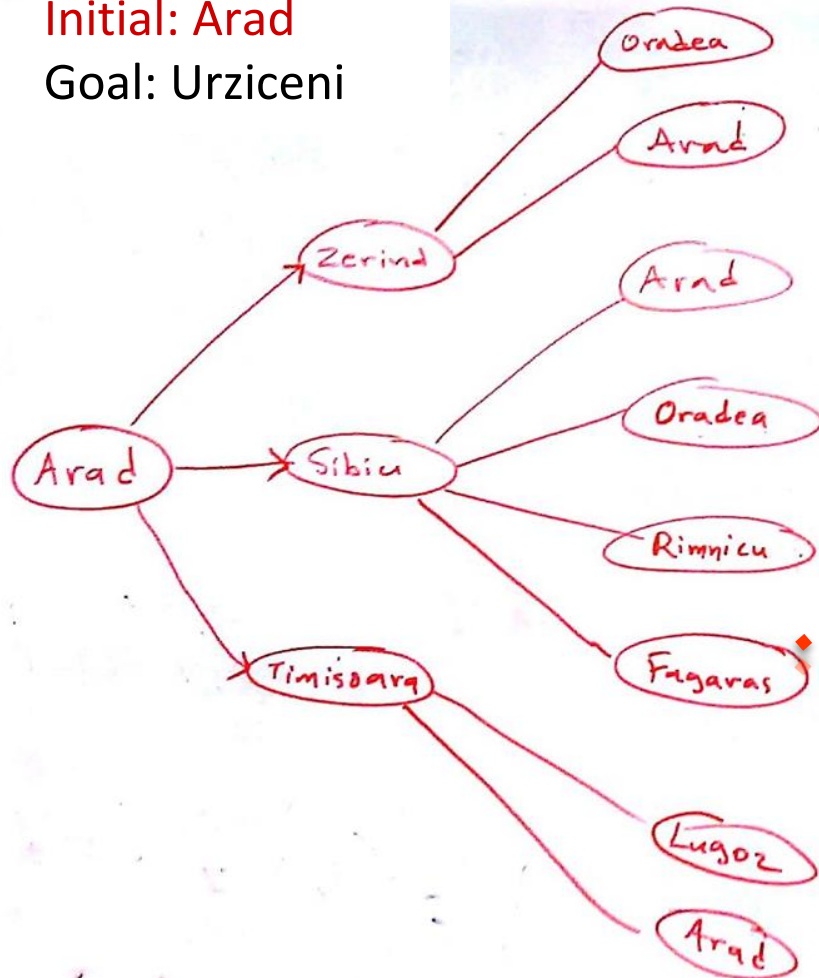
An Example: Routes in Romania



Bidirectional Search Example

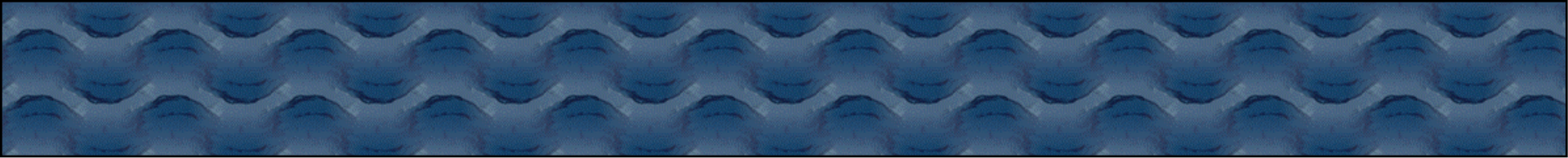
Initial: Arad

Goal: Urziceni



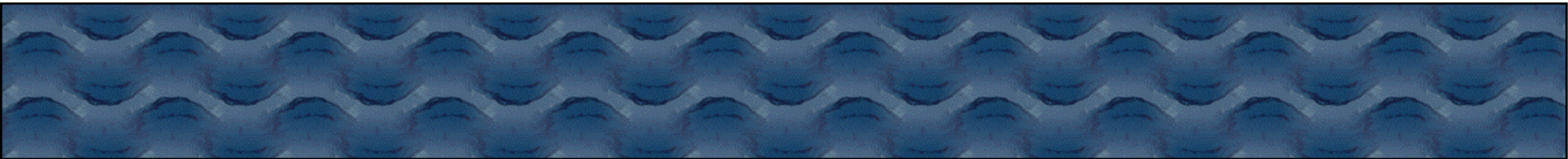
Comparing Uninformed Search Strategies

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes	Yes	No	No	Yes
Time	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(b^m)$	$O(b^l)$	$O(b^d)$
Space	$O(b^{d+1})$	$O(b^{\lceil C^*/\epsilon \rceil})$	$O(bm)$	$O(bl)$	$O(bd)$
Optimal?	Yes	Yes	No	No	Yes

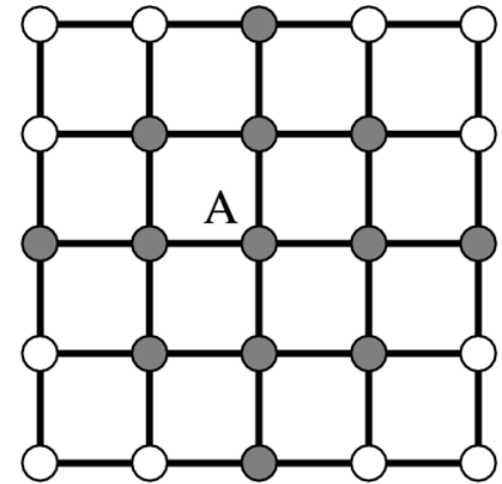
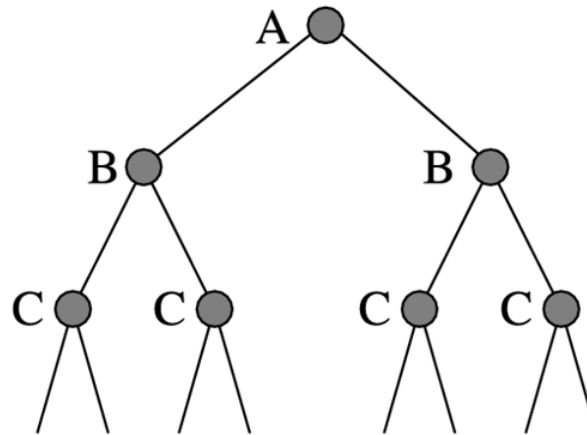
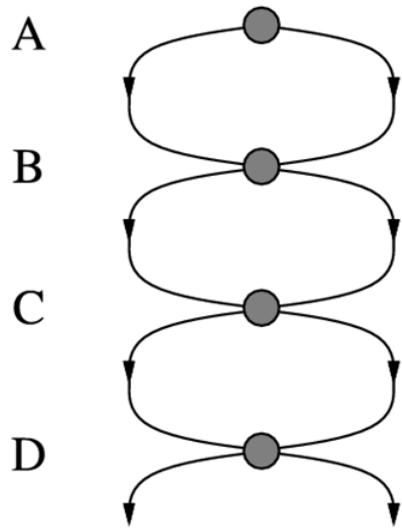


Repeated States

avoiding loops



Repeated States



- problem: a linear space starts looking exponential
- usually in problems with reversible actions
- simple comparison with nodes in the fringe is not enough

*algorithms that forget their history
are doomed to repeat it!*

Avoiding Repeated States

- **Memorization**

- closed list: expanded nodes
- open list: nodes to be expanded – fringe

- **Advantages**

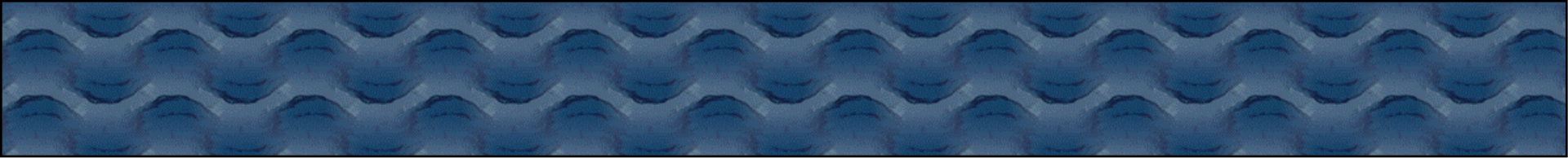
- space and time requirements proportional to state space
- all search algorithms become complete for finite spaces
- quick check in the closed list using a hash table

- **Disadvantages**

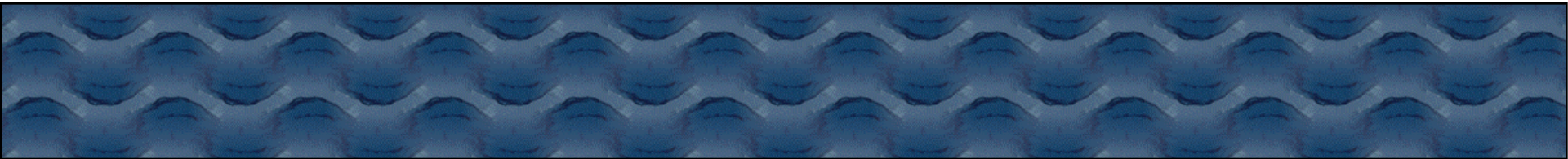
- no algorithm has linear time requirements anymore
- attention must be paid to alternative paths

Avoiding Repeated States: Algorithm

```
function GRAPH-SEARCH(problem, fringe) returns a solution, or failure
  closed ← an empty set
  fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node ← REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    if STATE[node] is not in closed then
      add STATE[node] to closed
      fringe ← INSERTALL(EXPAND(node, problem), fringe)
  end
```



Search with Partial Information



Problems with Partial Information

- **Sensorless problems**
 - the agent cannot perceive the current state
- **Contingency problems**
 - partially-observable environments or uncertain actions
 - percepts provide some information and restrict outcomes
- **Adversarial problems**
 - uncertainty comes from other agents
- **Exploration problems**
 - unknown state space and unknown actions
 - extreme case

Sensorless Problems

- **Features**

- known state space and known action effects
- unknown current state (initial, intermediate, final)

- **Confrontation**

- reasoning with sets of states, not just states
- belief space: powerset of state space
 - belief: the subset of currently possible states
 - actions: transitions between beliefs (union of simple transitions)
- initialization: belief with all possible states
- solution: search in the belief space
- goal: a belief where all states are goal states

Actions without Sensors

□ Coercion

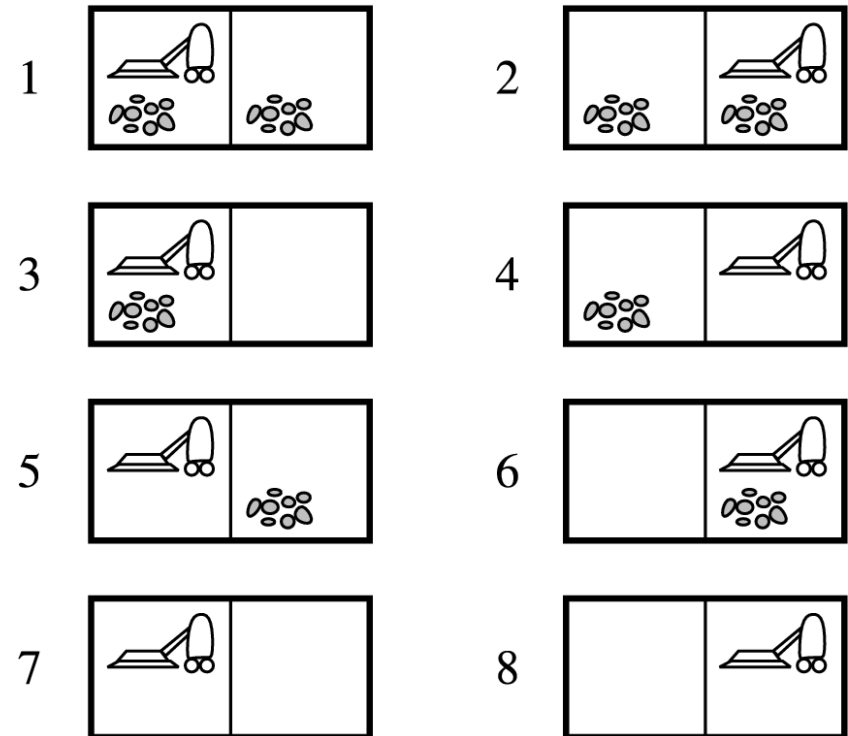
- choice of an action to exclude certain states

□ Vacuum Cleaner Example

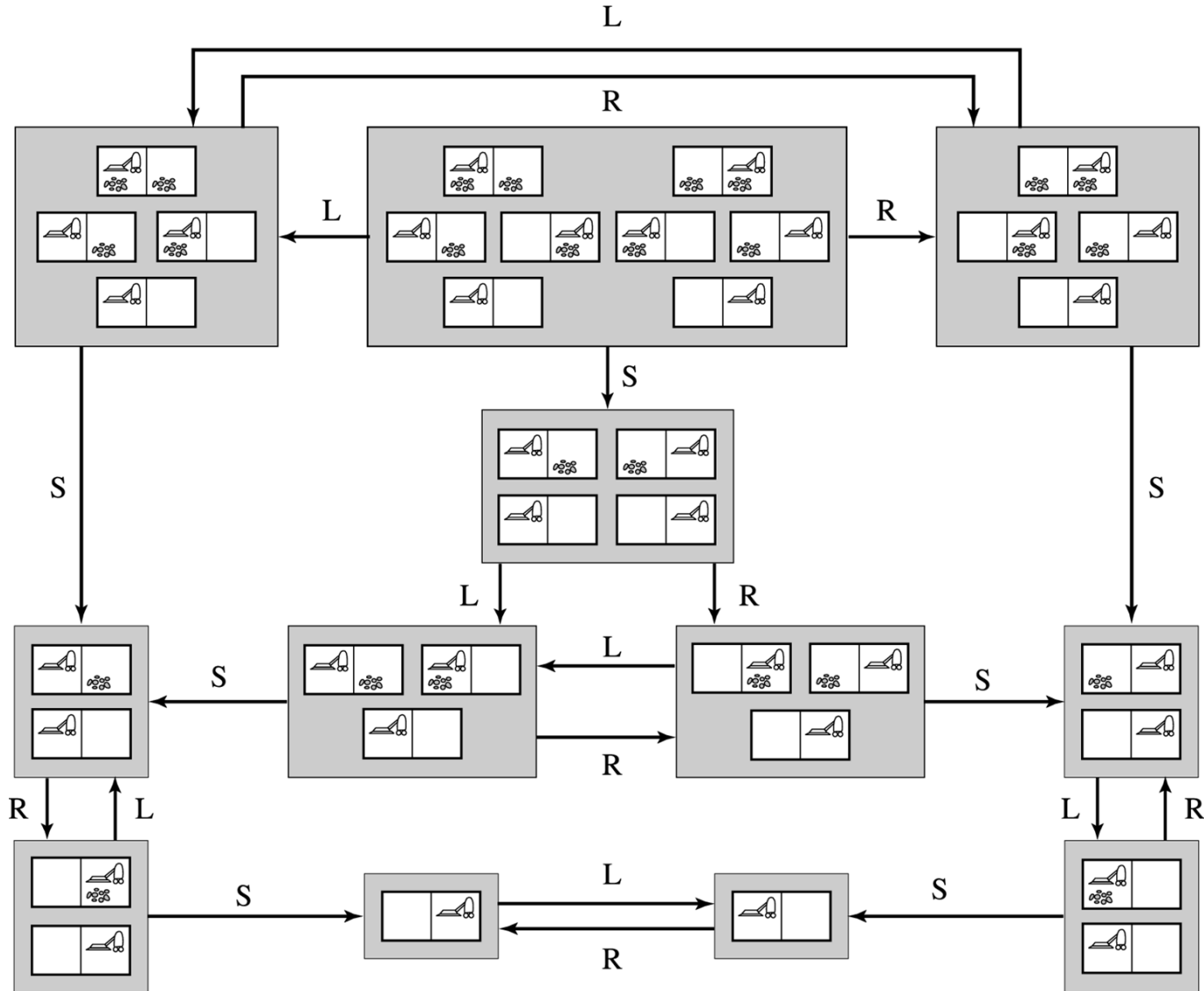
- initial belief {1,2,3,4,5,6,7,8}
- action: *Right*
- next belief: {2,4,6,8}
- action: *Suck*
- next belief: {4,8}
- ...

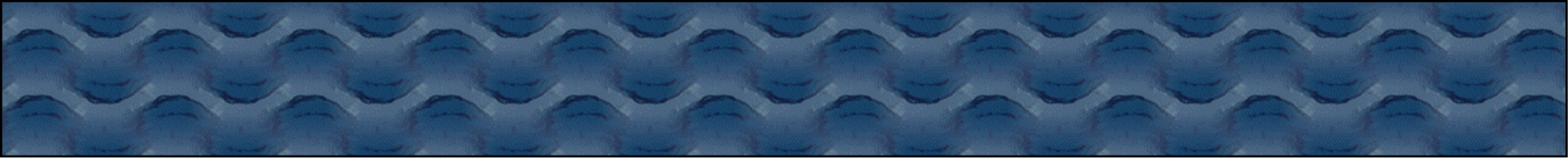
□ Generalization

- non-deterministic actions with uncertain effects



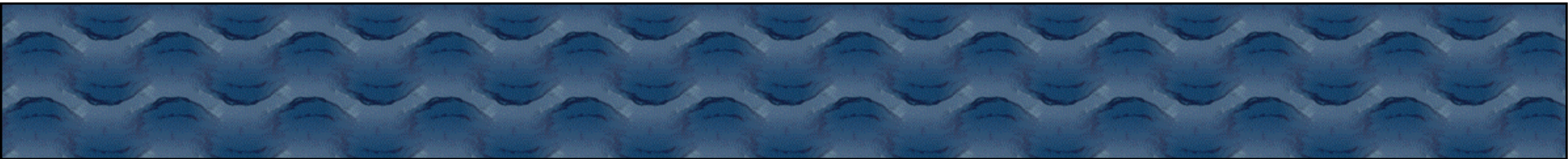
Vacuum Cleaner without Sensors





Informed Search

searching with guidance



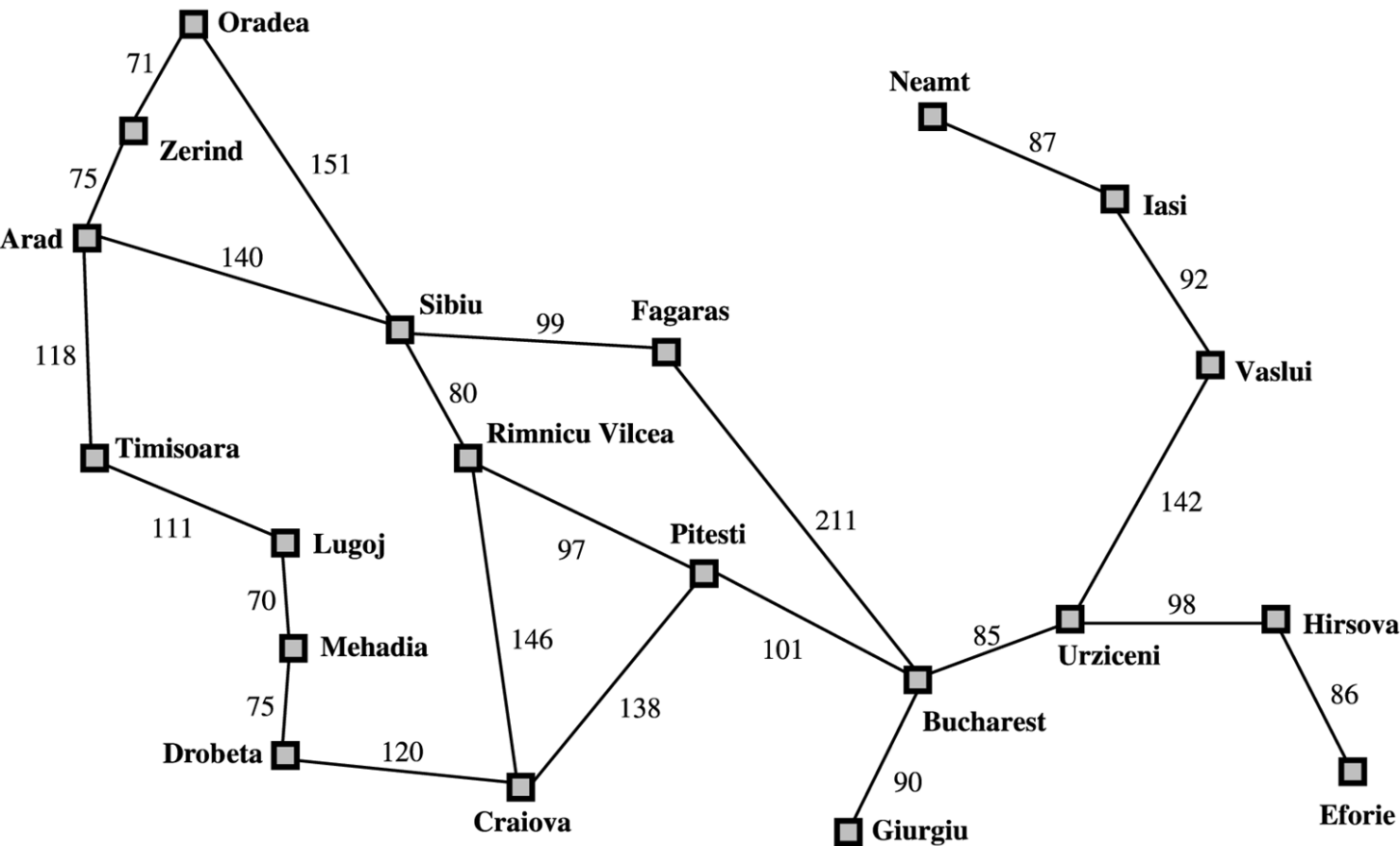
Best-First Search

- **Informed search**
 - use of domain knowledge about the problem, not just definition
 - uninformed search: depth, cost, repeat, etc.
 - informed search: extraction of information from the state, exploitation of domain knowledge
- **Best-first search**
 - evaluation function for the “best”
 - evaluation of each node n with the evaluation function $f(n)$
 - expand the node with the minimum value
 - implementation using a priority queue for the nodes in fringe

Greedy Best-First Search

- **Evaluation function**
 - heuristic function $h(n)$, $h(G)=0$
 - estimated cost of cheapest path from node n to the goal
 - the estimate depends only on node n and the goal
 - evaluation: $f(n) = h(n)$
- **Example: Routes in Romania**
 - $h_{\text{SLD}}(n)$: Straight-Line Distance (SLD) heuristic
 - e.g. SLD of city n from Bucharest (goal) in Routes in Romania
 - it cannot be computed easily from the search problem definition
 - it can be computed easily from the city coordinates on a map

Routes in Romania: SLD from Bucharest



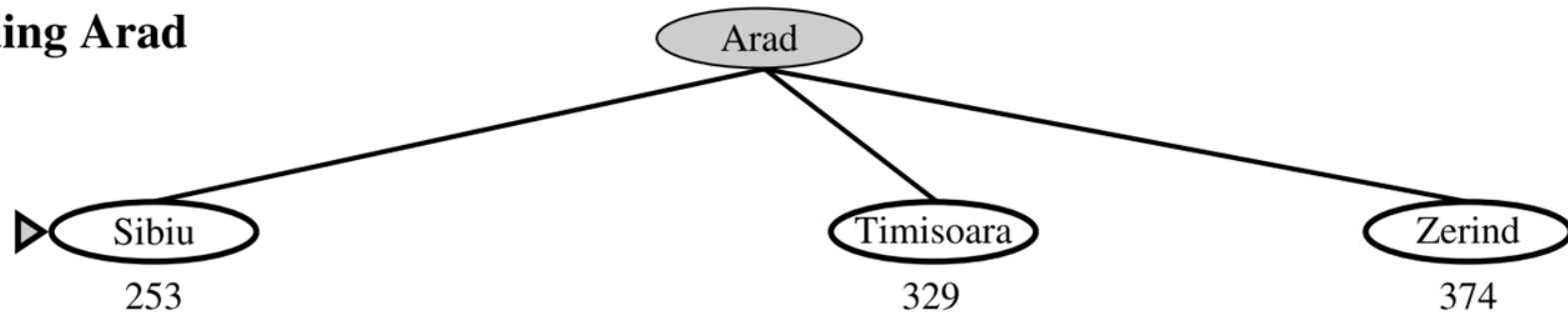
City	h_{SLD}	City	h_{SLD}
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Dobreta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Routes in Romania (GBFS)

(a) The initial state

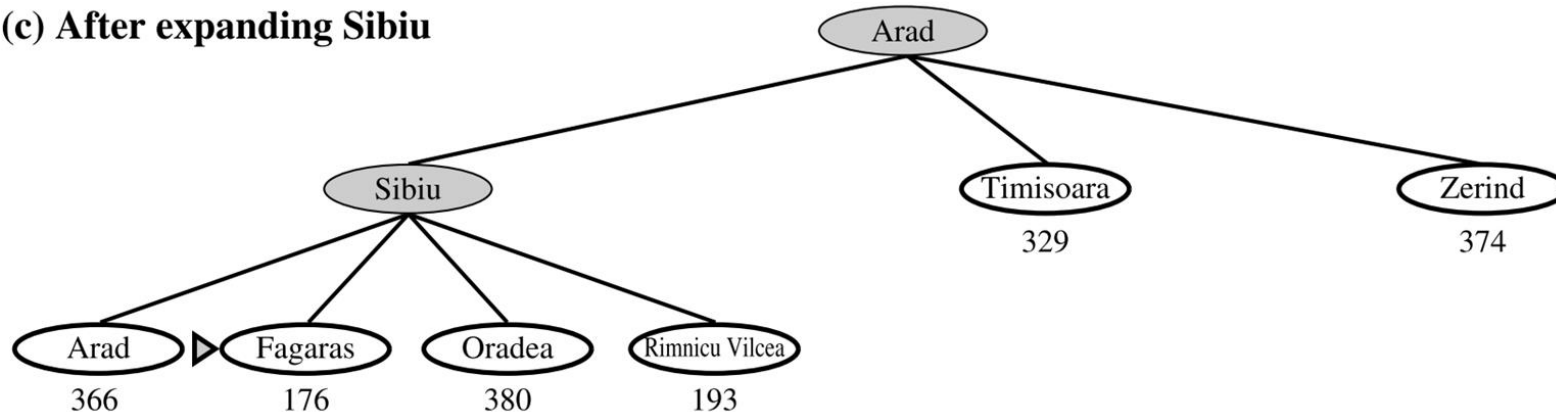


(b) After expanding Arad

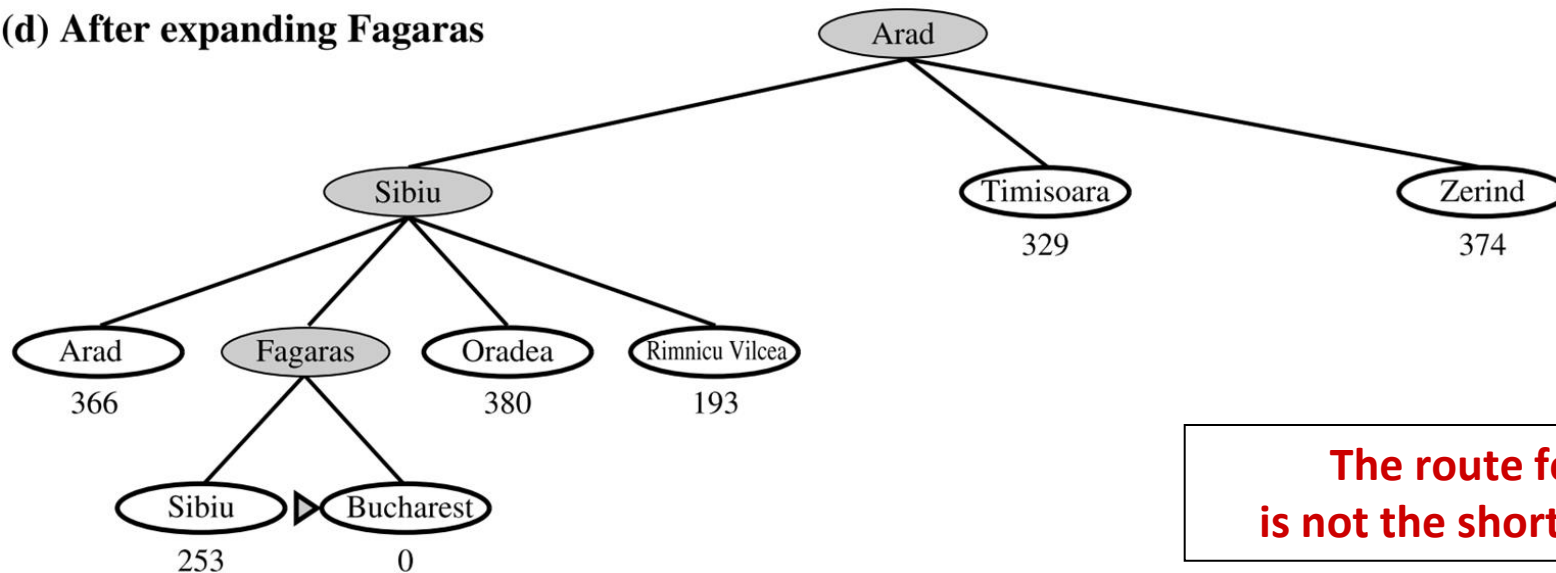


Routes in Romania (GBFS)

(c) After expanding Sibiu



(d) After expanding Fagaras



**The route found
is not the shortest one!**

Greedy Best-First Search

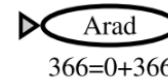
- **Completeness**
 - no in general, can get stuck in infinite loops (Iasi → Fagaras)
 - complete in finite spaces, if repeated nodes are avoided
- **Optimality**
 - no, subject to erroneous estimates (Sibiu → Bucharest)
- **Time and Space Complexity**
 - exponential $O(b^m)$, where m is the maximum depth of the space
- **Overall**
 - looks like depth-first search

A* Search

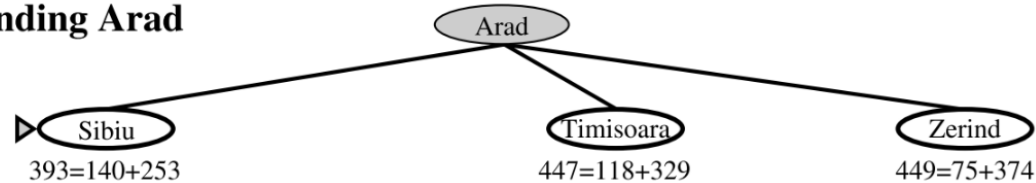
- **Evaluation function**
 - estimated cost $h(n)$ of cheapest path from node n to the goal
 - real cost $g(n)$ of current path from initial state to node n
 - evaluation: $f(n) = g(n) + h(n)$ – total path cost through node n
- **Admissible heuristic**
 - always makes optimistic estimates
 - does not overestimate costs
- **Theorem**
 - A* search without pruning of repeated states is optimal, if the heuristic function $h(n)$ is admissible.

Routes in Romania (A*)

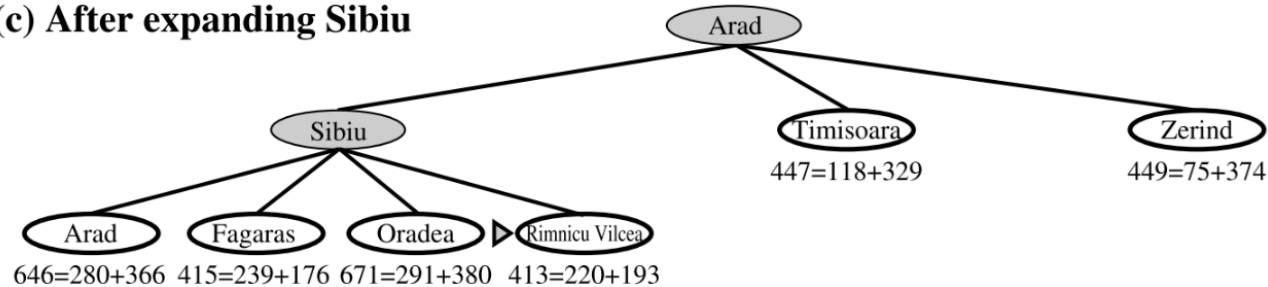
(a) The initial state



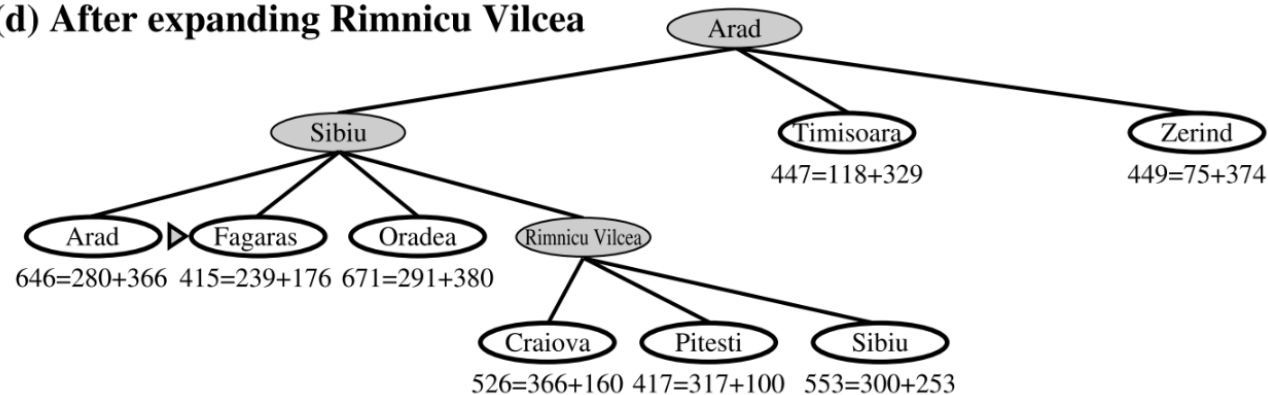
(b) After expanding Arad



(c) After expanding Sibiu

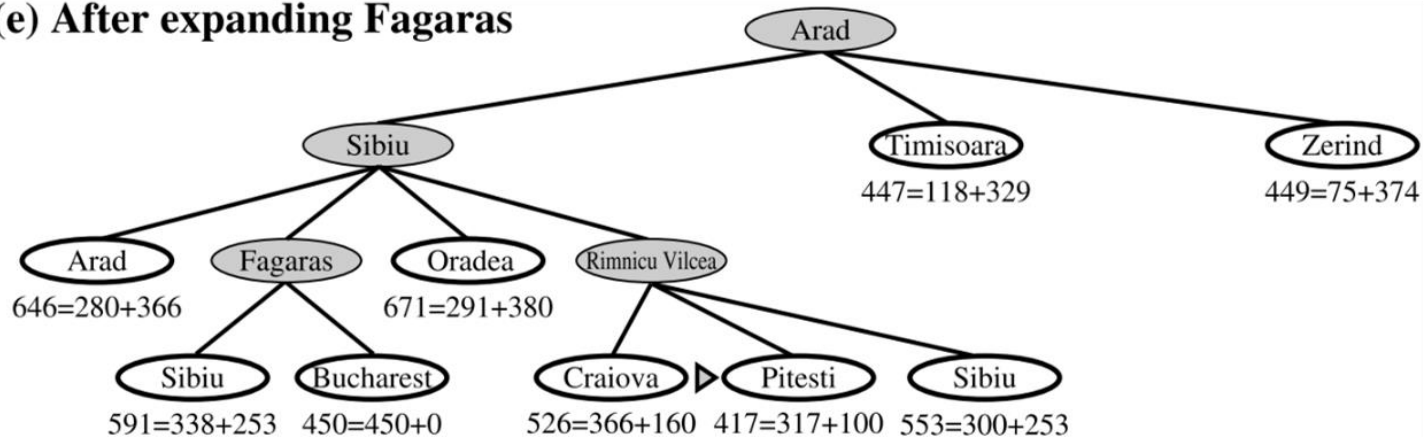


(d) After expanding Rimnicu Vilcea

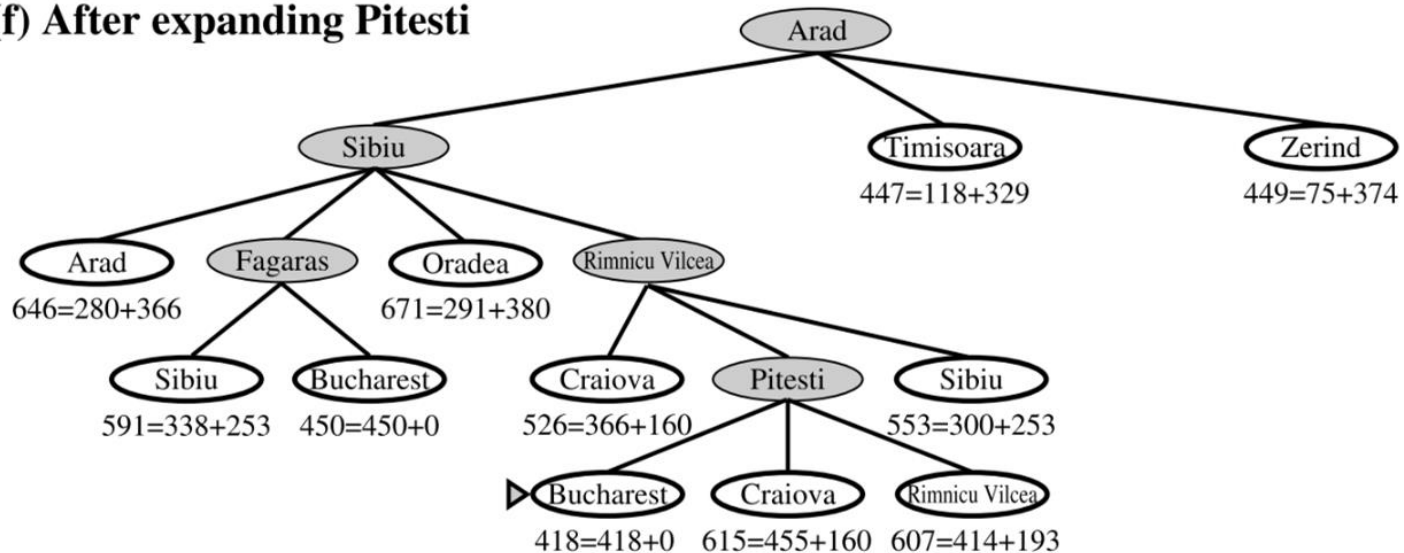


Routes in Romania (A*)

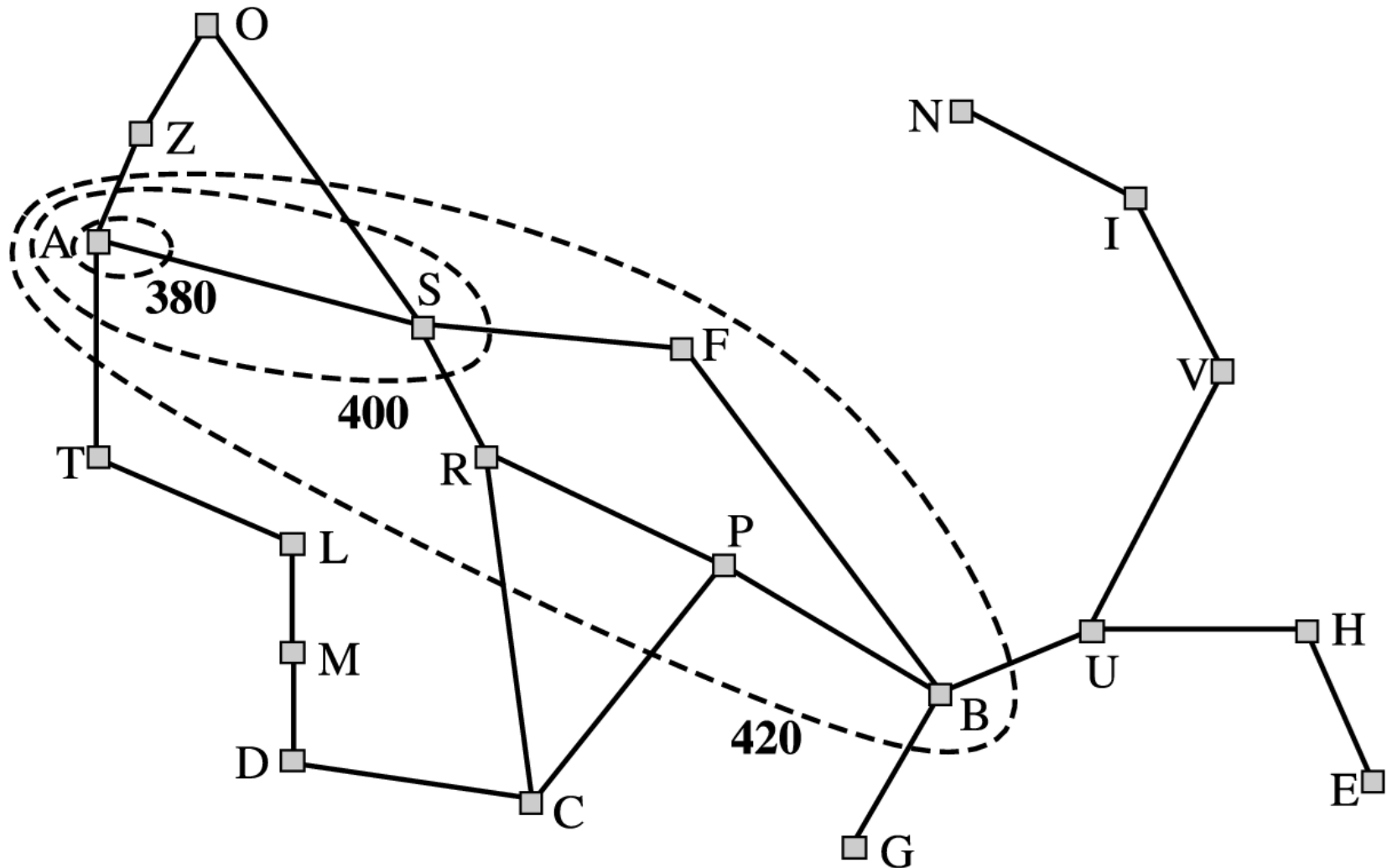
(e) After expanding Fagaras



(f) After expanding Pitesti

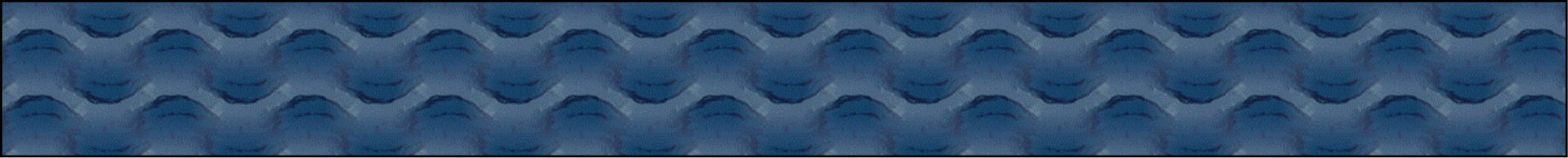


Cost Contour Lines



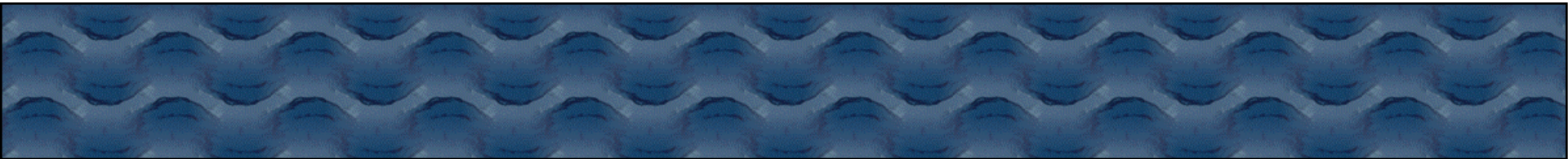
A* Performance

- **Completeness**
 - yes, unless there are infinite nodes n with $f(n) \leq f(G) = C^*$
- **Optimality**
 - yes, with an admissible heuristic
 - expansion: all with $f(n) < C^*$, some with $f(n) = C^*$, none with $f(n) > C^*$
- **Complexity**
 - exponential requirements in time and (mostly) in memory
- **Optimally efficient**
 - for the same problem, A* explores the fewest states of all the optimal and complete search algorithms



Informed Search with Bounded Memory

can we save on memory?



Iterative-Deepening A* - IDA*

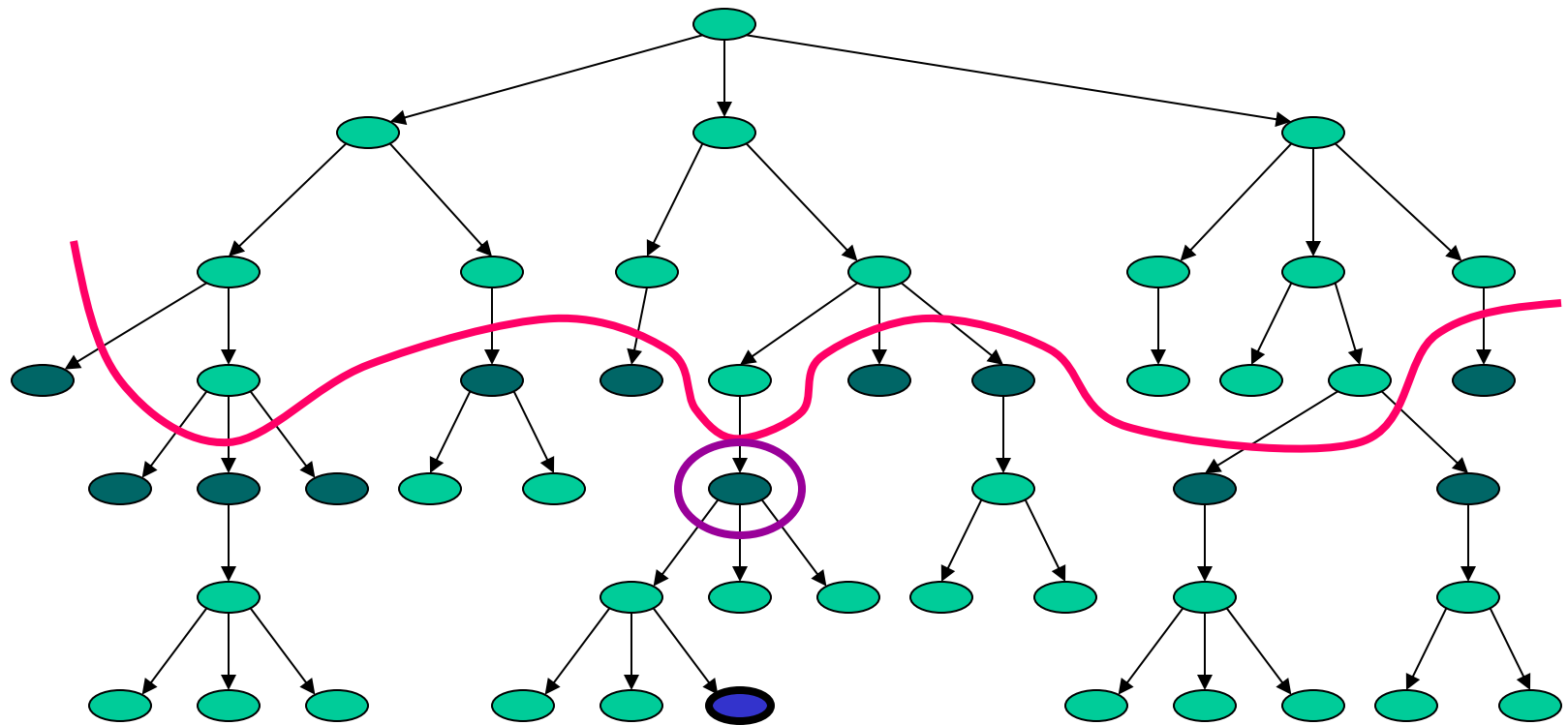
▪ Methodology

- analogous to iterative lengthening search
- search up to a depth with limited cost $f(n) = g(n) + h(n)$
- gradual increase of the cut-off limit $0, f_1, f_2, f_3, \dots$
- new limit f_i : the least $f(n)$ over the limit f_{i-1} at stage $i-1$
- complete and optimal with admissible h
- linear space complexity $O(bd)$
- exponential time complexity $O(b^d)$

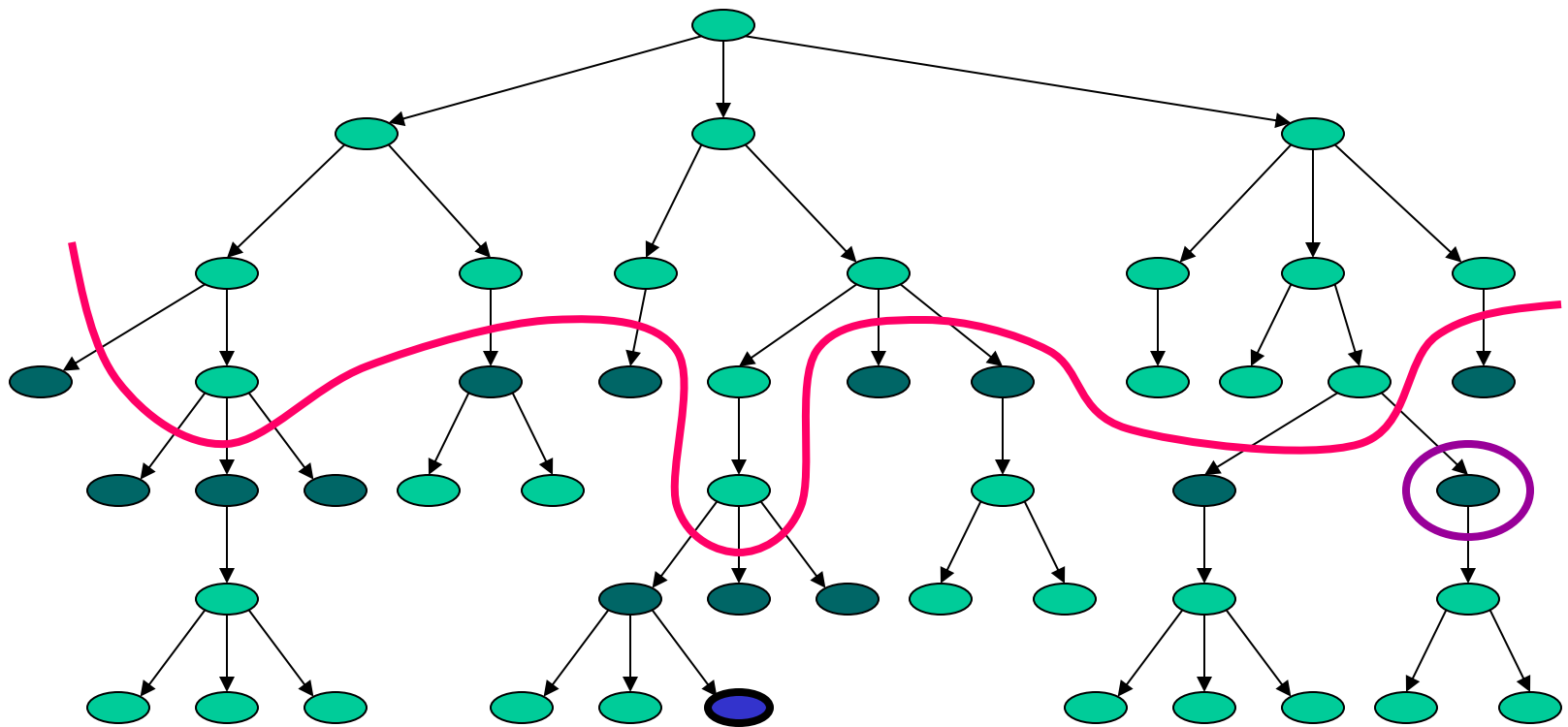
▪ Disadvantages

- possible, slow increase of the cut-off limit $\Rightarrow$ slow progress
- does not detect repeated states

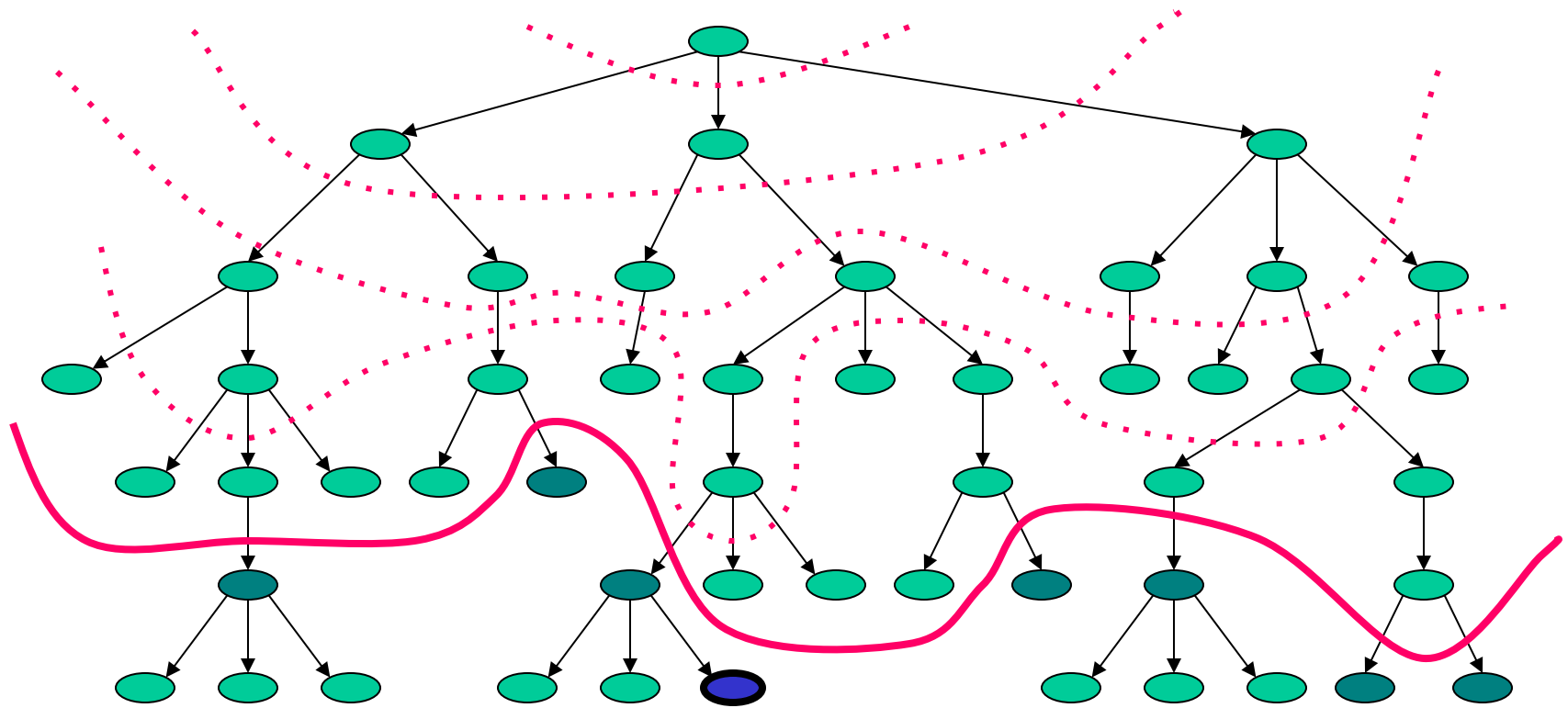
IDA* - Searching up to a Cut-Off Limit

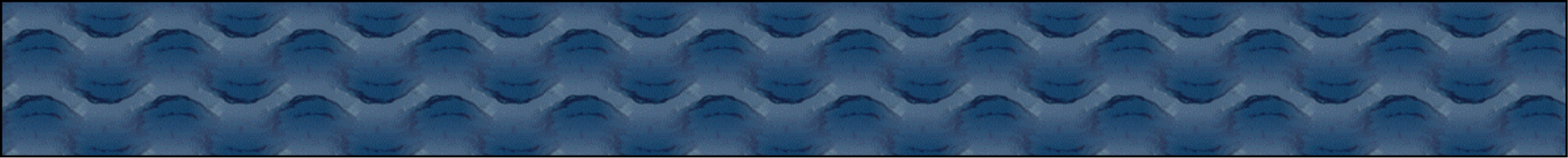


IDA* - Increasing the Cut-Off Limit



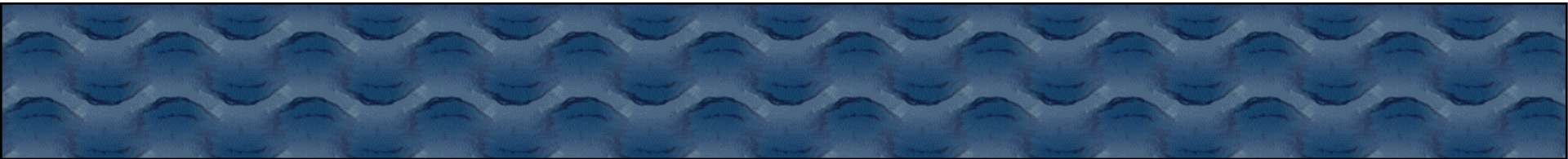
IDA* Search Stages





Heuristic Functions

providing guidance



8 Tile Puzzle

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

$$h_1 = 8$$

$$h_2 = 18$$

$$h^* = 26$$

Heuristic functions

- h_1 = the number of tiles not in the correct place
- h_2 = the sum of distances of the tiles from their destination
 - Manhattan distance (distance of building blocks)
- admissible heuristic functions (does not overestimate)
- $h_1 \leq h_2 \Rightarrow h_2$ dominates over $h_1 \Rightarrow h_2$ expands less nodes

Comparing Heuristic Functions

– uniform tree: $N + 1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$

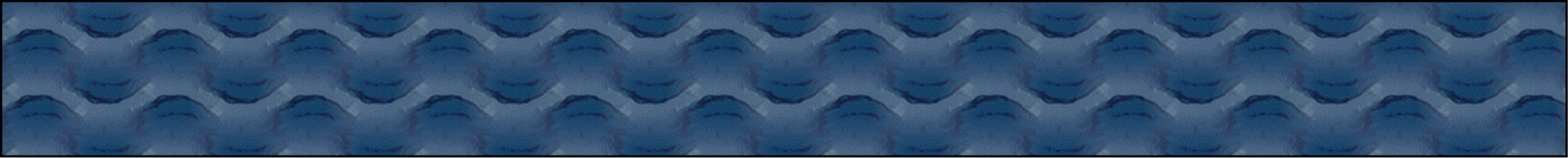
d	Search Cost N			Active Branching Factor b^*		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2,45	1,79	1,79
4	112	13	12	2,87	1,48	1,45
6	680	20	18	2,73	1,34	1,30
8	6384	39	25	2,80	1,33	1,24
10	47127	93	39	2,79	1,38	1,22
12	3644035	227	73	2,78	1,42	1,24
14	—	539	113	—	1,44	1,23
16	—	1301	211	—	1,45	1,25
18	—	3056	363	—	1,46	1,26
20	—	7276	676	—	1,47	1,27
22	—	18094	1219	—	1,48	1,28
24	—	39135	1641	—	1,48	1,26

Inventing Admissible Heuristics

- **Relaxed problems**
 - simplification or relaxation of the rules of the problem
 - in general, the relaxed problem can be easily solved optimally
 - $f(\text{optimal solution of relaxed}) \leq f(\text{optimal solution of original})$
 - admissible h : cost of an optimal solution for the relaxed problem
- **Relaxation example**
 - 8 tile puzzle original rule: *a tile can be moved from A to B, if A borders horizontally or vertically with B **and** B is empty*
- **Collection of heuristic functions**
 - admissible heuristic functions $h_1, h_2, \dots, h_m$
 - $h = \max\{h_1, h_2, \dots, h_m\}$ is admissible and dominates over $h_1, \dots, h_m$

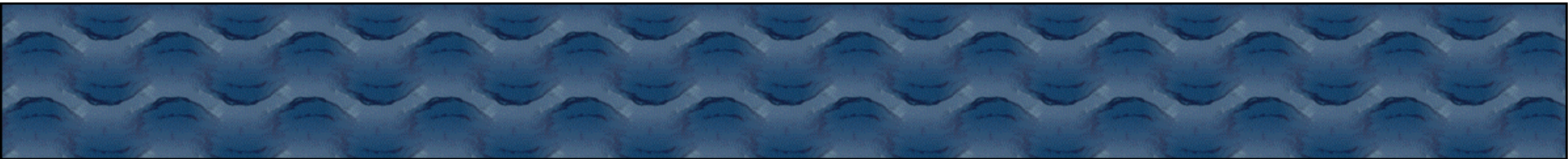
Learning Heuristic Functions

- **Experience**
 - solve specific problem instances, compute optimal cost
 - form a training set $\{ \text{state } n, \text{ cost } h^*(n) \}$
- **Features**
 - enhance state description
 - e.g. tiles not in place, sum of distances, ...
 - enhanced training set $\{ [\text{state } n, \text{ features of } n], \text{ cost } h^*(n) \}$
- **Learning**
 - supervised learning for generalization over the entire space
 - decision trees, SVMs, RVMS, deep neural networks, ...



Local Search

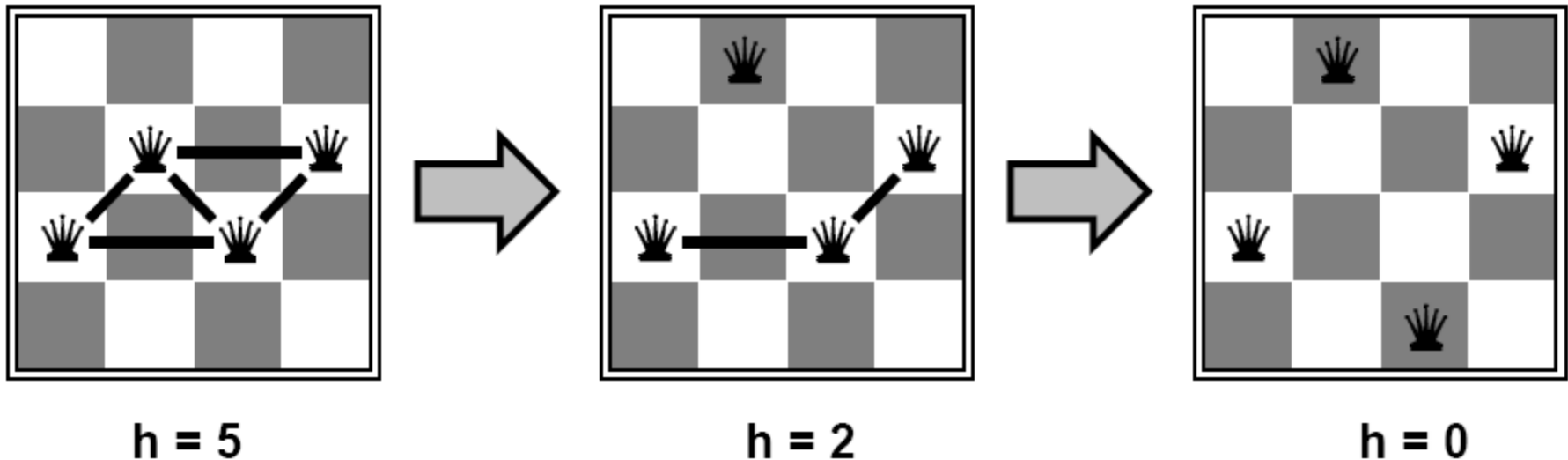
**looking for the goal itself!
not for the path**



Local Search

- **Goal-oriented problems**
 - we care to find the goal, the path to the goal is not important
 - e.g., 8 queens problem, various scheduling tasks, ...
- **Local search algorithms**
 - keep track of the current state only
 - (mostly) move to “neighbouring” states only
 - local search algorithms are not *complete*
 - no guarantees that a solution will be returned, even if it exists
- **Advantages**
 - space-efficient: they use constant amount of memory
 - very good for search in very large or infinite state spaces

An Example: The n-Queens Problem



h = total number of threats (pairs of queens threatening each other)

Optimization

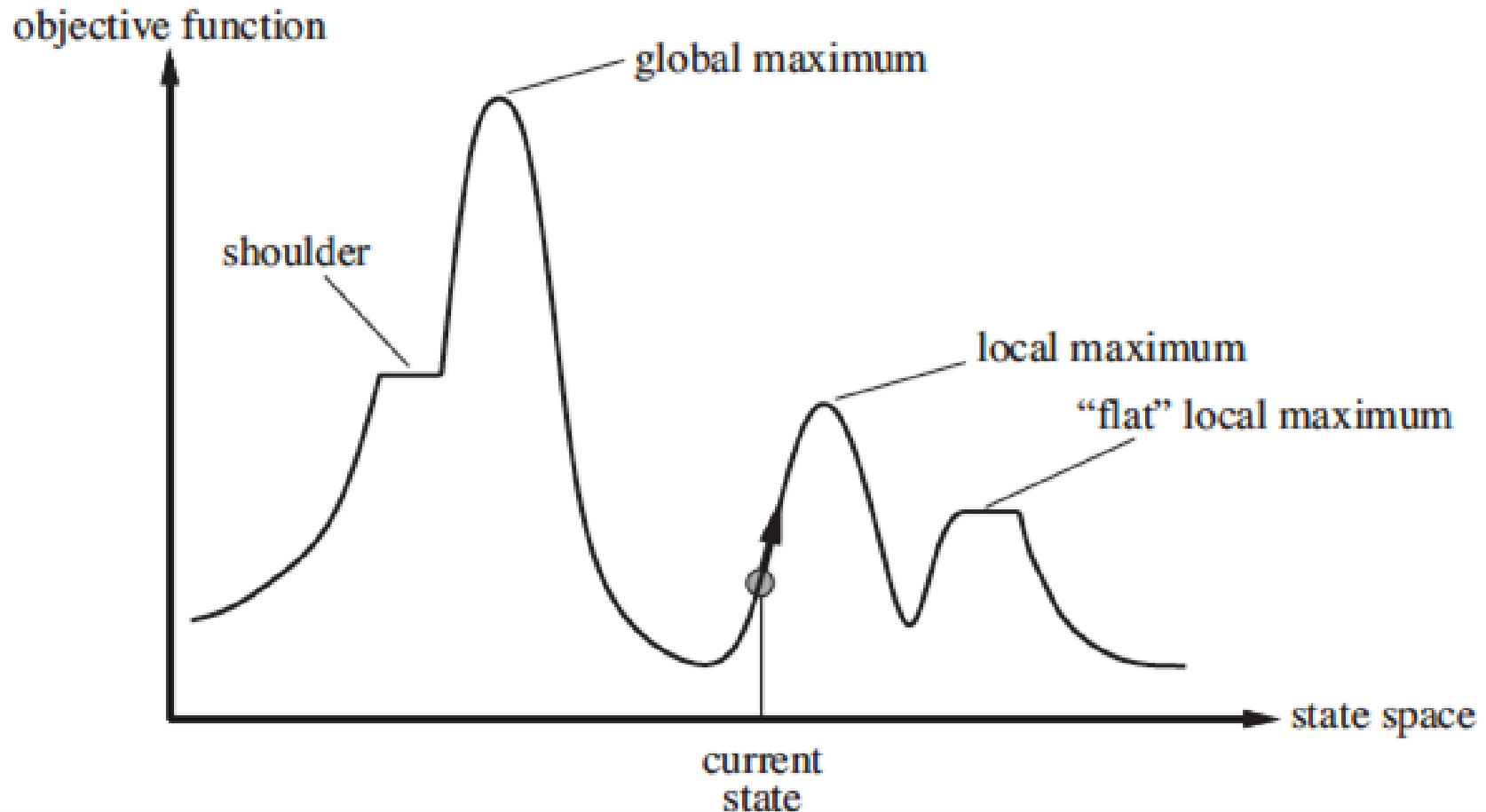
▪ Optimization Problems

- use of an *objective function*
- objective function: maps a state to a number
- goal state: optimizes the objective function
 - optimization: *maximization* or *minimization*
 - just change the sign to switch!

▪ State Space landscape

- position = state ; altitude = objective function value
- extrema: global maximum / minimum
- local extrema: local maximum / minimum
- optimization: find a global maximum / minimum

State Space Landscape



Hill Climbing

- **Algorithm**
 - move to the neighbouring state with the “best value”
 - stop, when no neighbour can improve on current value

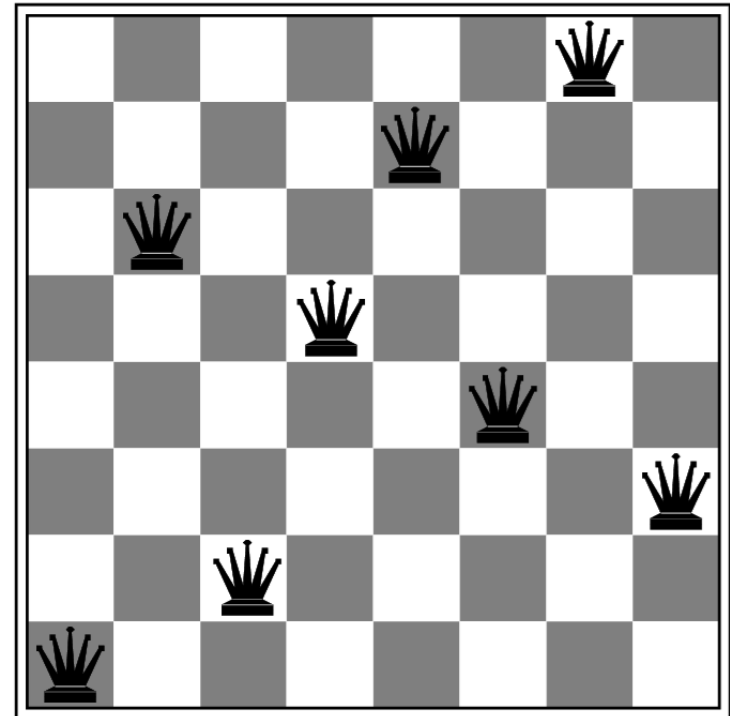
```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                  neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
    current ← neighbor
  end
```


Hill Climbing – An Example

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14	♔	13	16	13	16
♔	14	17	15	♔	14	16	16
17	♔	16	18	15	♔	15	♔
18	14	♔	15	15	14	♔	16
14	14	13	17	12	14	12	18

current value $h=17$

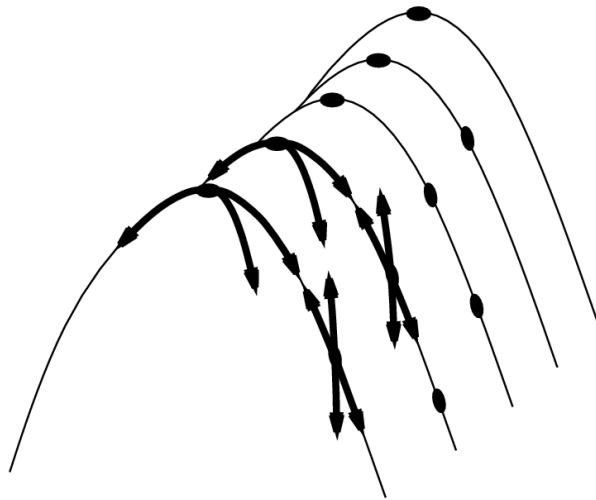


a local minimum with $h=1$

Hill Climbing Problems

□ Challenges

- local maxima
- plateaus
- ridges



□ Treatment

- side moves (up to some limit)
- stochastic search
 - choose among “promising” moves according to some probability distribution (probability of choice : depends on the move’s/state’s value)
 - (when many neighbours) create moves randomly, pick first “good” move
- random restarts
 - a complete algorithm
 - p : probability a search succeeds
 - $1/p$: expected number of restarts

8 queens: 94% success instead of 14% (1.06 restarts instead of 7.14) with side moves!

Simulated Annealing

- **History**

- invented by Nicholas Metropolis in 1953
- based on physics/metallurgy (solid state transition models)
- a variant of stochastic hill climbing

- **Algorithm**

- pre-selects an action randomly (irrespective of value)
- if value better than current, it is accepted as the next move
- if not, it is accepted with probability $p=e^{\Delta E/T}$
 - $\Delta E < 0$ the distance to current value; $T > 0$ a “temperature” parameter
- temperature is decreased progressively with time
- many “sub-optimal” actions initially
- focus progressively moves to “expected-to-be-best” moves
- the decreasing schedule of T is important for optimality

Simulated Annealing

```

function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”
  local variables: current, a node
                    next, a node
                    T, a “temperature” controlling prob. of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
     $\Delta E \leftarrow \text{VALUE}[\textit{next}] - \text{VALUE}[\textit{current}]$ 
    if  $\Delta E > 0$  then current ← next
    else current ← next only with probability  $e^{\Delta E/T}$ 

```

Local Beam Search

- **Algorithm**
 - climb hills from different points in-parallel
 - k random initial states are selected
 - k local searches are performed in-parallel
 - at each step: select the k best among all previous step's children
- **Features**
 - funnel resources to most promising areas
 - problem: searches are concentrated in the same sub-area
 - treatment: choose the k children nodes *stochastically*
 - selection probability proportional to the child's value

Genetic Algorithms (GAs)

▪ Algorithm and Glossary

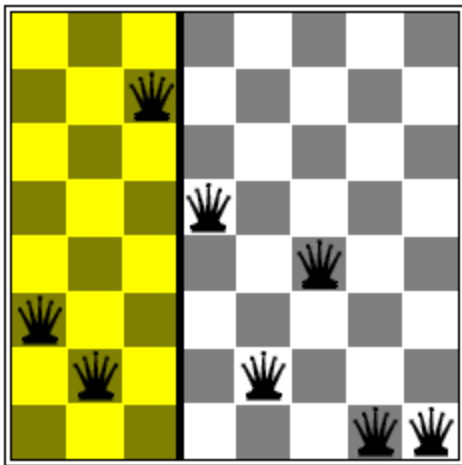
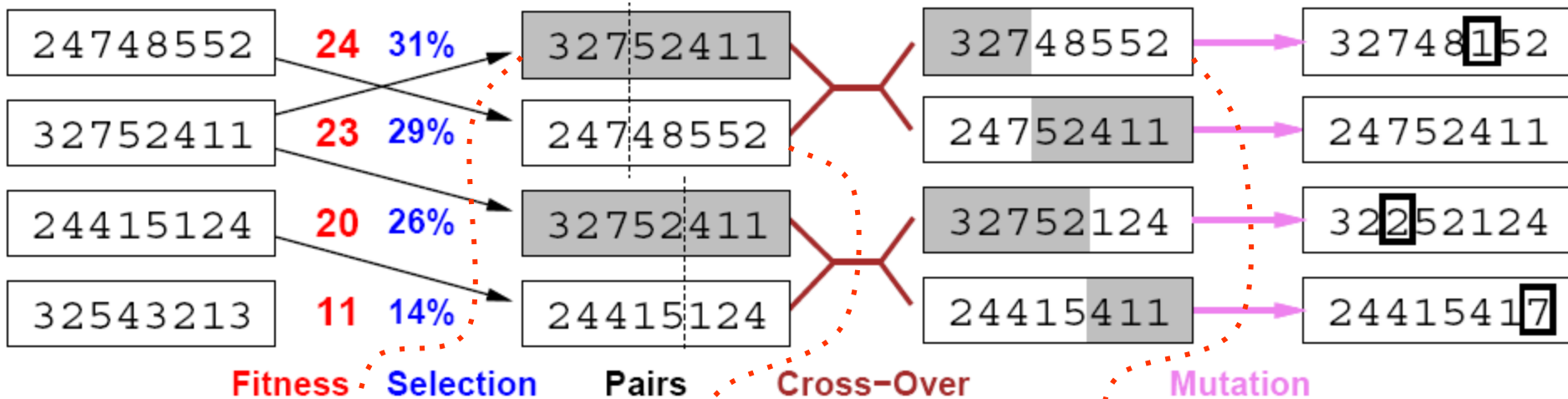
- stochastic beam search + genetic reproduction of successors
- **population**: k **individuals** or states (initially random)
- **chromosomes**: a representation of each state (strings)
- **fitness**: a heuristic function to evaluate individuals
- **selection**: the k “best” individuals survive
- **crossover**: “mate” two strong individuals to give birth to a new one
- **mutation**: a random change in an individual’s chromosome

▪ Features

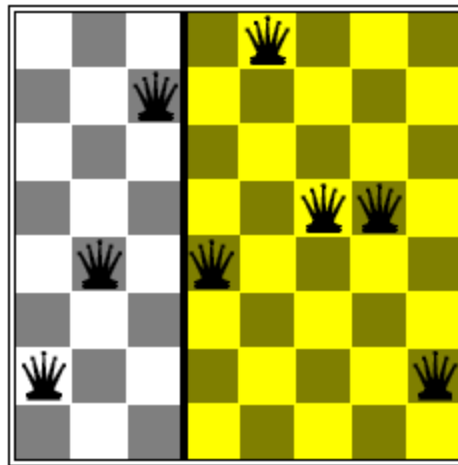
- careful engineering of the representation and other parameters
- 8 queens: position (1-8) of each queen at its column (8 symbols)

GA for the 8-Queens Problem

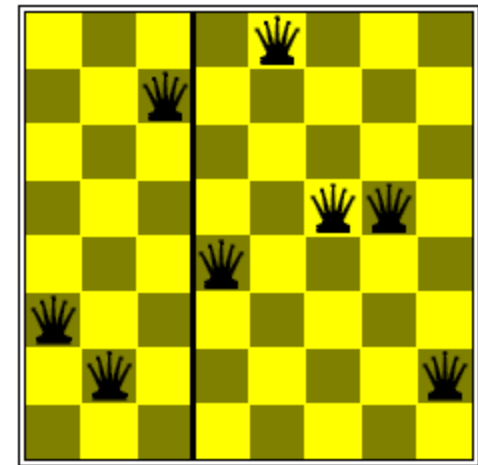
fitness = pairs of queens **not** threatening each other (max value = 28)



+

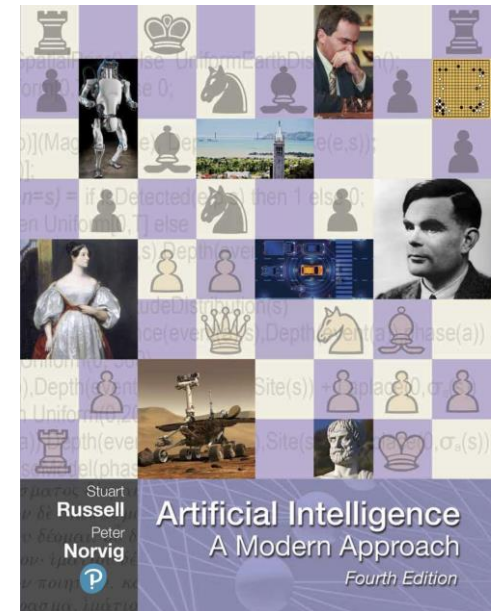


=



Related Bibliography

- **Artificial Intelligence: A Modern Approach, Stuart Russell and Peter Norvig, 4th edition, Prentice Hall, 2020**
 - Sections 2.1 – 2.4
 - Sections 3.1 – 3.3



Constraint Satisfaction Problems

Michail G. Lagoudakis

Intelligent Systems Laboratory

School of Electrical and Computer Engineering

Technical University of Crete

Chania, Crete, Greece



Constraint Satisfaction Problem (CSP)

■ CSP Definition

- variables: $X_1, X_2, \dots, X_n$
- (non-empty) domains of possible values: $D_1, D_2, \dots, D_n$
- constraints: $C_1, C_2, \dots, C_m$
 - allowable combinations of values to subsets of variables
- value assignment: $\{X_i = v_i, X_j = v_j, \dots\}$
 - can be partial or complete
- consistent (or legal) assignment: no constraint is violated
- solution: complete and consistent assignment

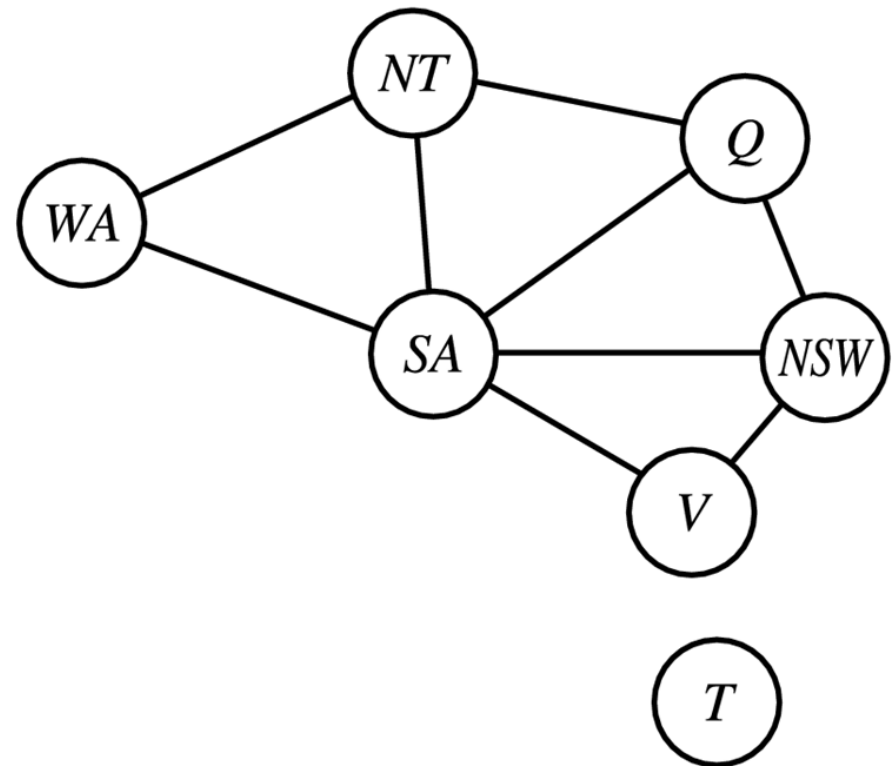
■ Extension

- in addition, optimization of an objective function

Example: Map Coloring



Map of Australia



Constraint Graph

goal: assign colors so that no neighboring regions have the same color

Example: Map Coloring

■ Variables

- *WA, NT, Q, NSW, V, SA, T*

■ Domains

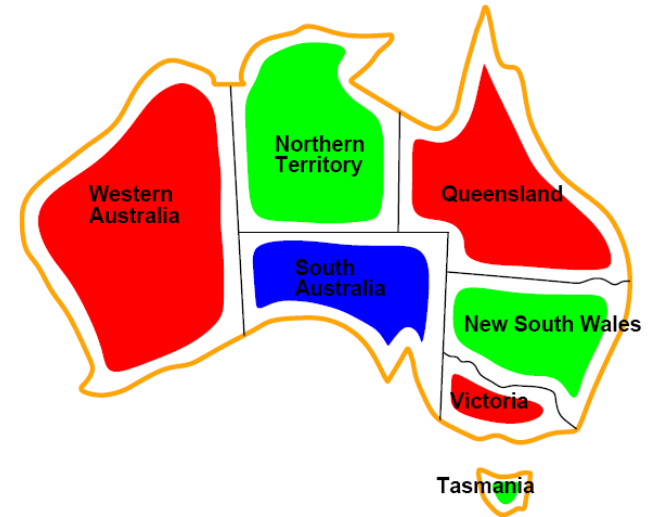
- {*red, green, blue*} for all variables

■ Constraints

- neighboring regions cannot have the same color
- representation by enumeration of allowable combinations:
 $\{ (red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green) \}$
- representation by using specialized notation: $WA \neq NT, \dots$

■ Solution (one of many)

- $WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green$



Benefits

■ Advantages

- generic, formal language of representation
- general-purpose algorithms
- utilization of the structure of problems

■ Applications

- task scheduling
- curriculum development and course assignment
- semester timetable for a university
- coloring graph
- hardware configuration

Constraint Satisfaction as Search I

■ States

- partial or complete (consistent) value assignments to variables
- initial state: empty value assignment

■ Successor function

- value assignment that does not cause any conflict

■ Goal state

- complete (consistent) value assignment to variables

■ Path cost

- irrelevant (solution is independent of the search path)

■ Algorithm

- depth-first search (maximum depth = number of variables)

Constraint Satisfaction as Search II

■ States

- complete value assignments to variables
- initial state: random value assignment (possible conflicts)

■ Local Moves

- new value assignment to some variable to reduce conflicts

■ Goal state

- (complete) consistent value assignment to variables

■ Path cost

- irrelevant (solution is independent of the search path)

■ Algorithm

- local search

Problem Characteristics

■ Formulation

- incremental (depth-first search)
- complete states (local search)

■ Variables

- discrete / continuous variables
- finite / infinite domains of values

■ Constraints

- unary / binary / ternary / n -ary (higher-order)
- soft constraints
 - can be violated, but prefer to satisfy
 - reduction to optimization with weights

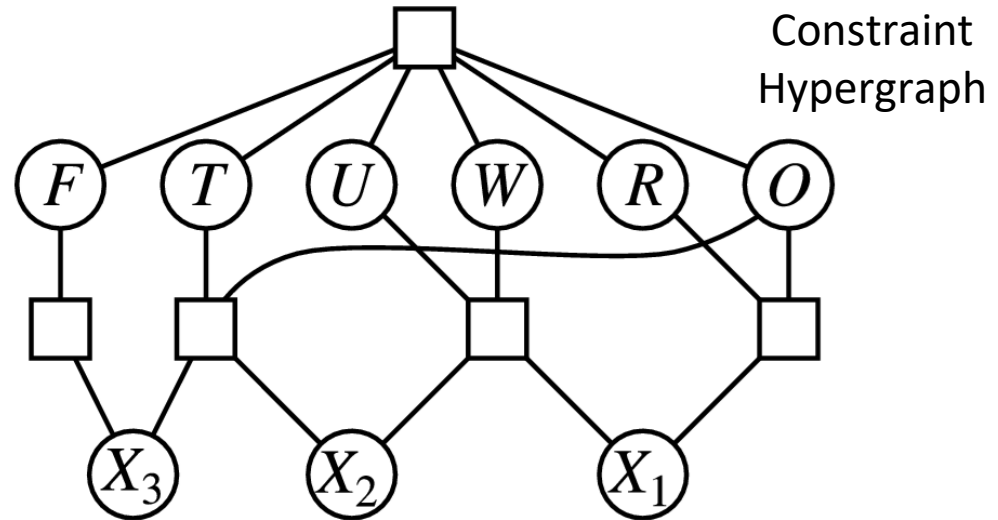
Higher-Order Constraints

Cryptarithmic Puzzle

$$\begin{array}{r}
 765 \\
 + 765 \\
 \hline
 1530
 \end{array}
 \quad
 \begin{array}{r}
 TWR \\
 + TWR \\
 \hline
 FOUR
 \end{array}$$

$$\begin{array}{r}
 132 \\
 + 132 \\
 \hline
 0264
 \end{array}$$

$$\begin{array}{r}
 734 \\
 + 734 \\
 \hline
 1468
 \end{array}$$

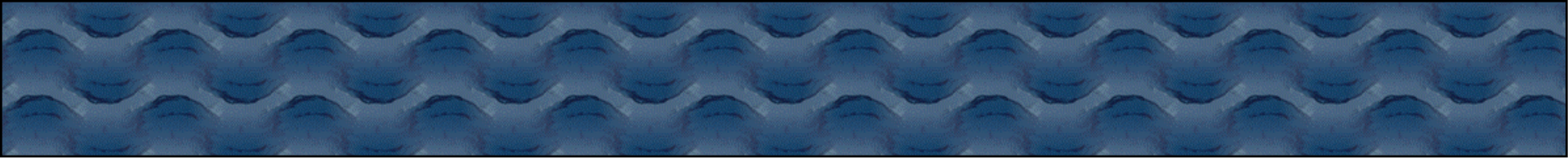


Variables

- F, T, U, W, R, O
 - $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- X_1, X_2, X_3
 - $D = \{0, 1\}$

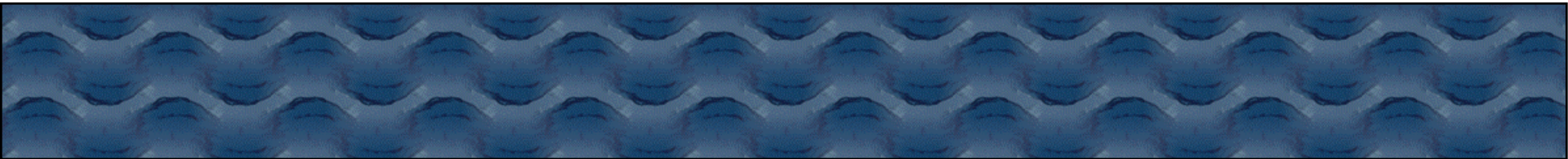
Constraints

- $Alldiff(F, T, U, W, R, O)$
- $O + O = R + 10 \times X_1$
- $X_1 + W + W = U + 10 \times X_2$
- $X_2 + T + T = O + 10 \times X_3$
- $X_3 = F$



Backtracking Search

Backtracking at Dead-Ends



Commutativity and Backtracking

■ Naive search

- depth-first search
- $b = nd, (n-1)d, (n-2)d, \dots \Rightarrow n!d^n$ leaves, but only d^n assignments!

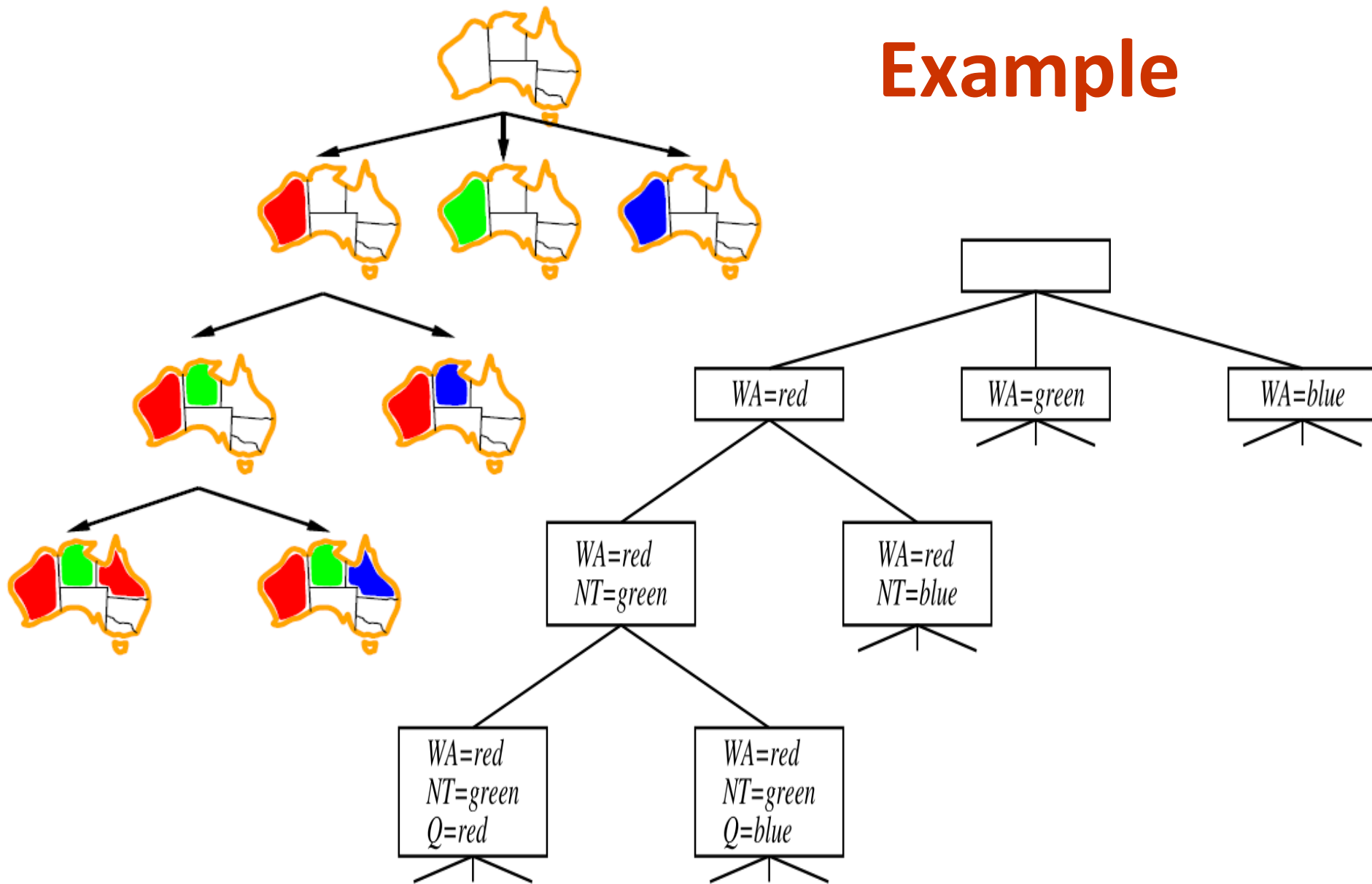
■ Commutativity

- the ordering of actions (assignments) does not affect the result
- assignment to a *single* variable per level $\Rightarrow b=d \Rightarrow d^n$ leaves

■ Backtracking search

- depth-first search
- assignment to a *single* variable per level
- variables can be ordered arbitrarily or heuristically
- backtracking when some variable has no legal values left

Example



Backtracking Search Algorithm

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment given CONSTRAINTS[csp] then
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```

Improvements

■ Conventional search

- use of heuristic functions
- exploitation of domain knowledge

■ Constraint Satisfaction Problems

- use of generic techniques
 - in which variable will the next assignment be made?
 - in what order will the legal values be assigned?
 - what are the effects of the current assignment to the other variables?
 - how can a definite failure be detected early?
 - how can we avoid repeated failures?
 - how can we exploit the structure of the problem?
 - ...

Variable and Value Ordering

var $\leftarrow$ SELECT-UNASSIGNED-VARIABLE(VARIABLES[*csp*], *assignment*, *csp*)

- **Variable selection: the most-constrained**

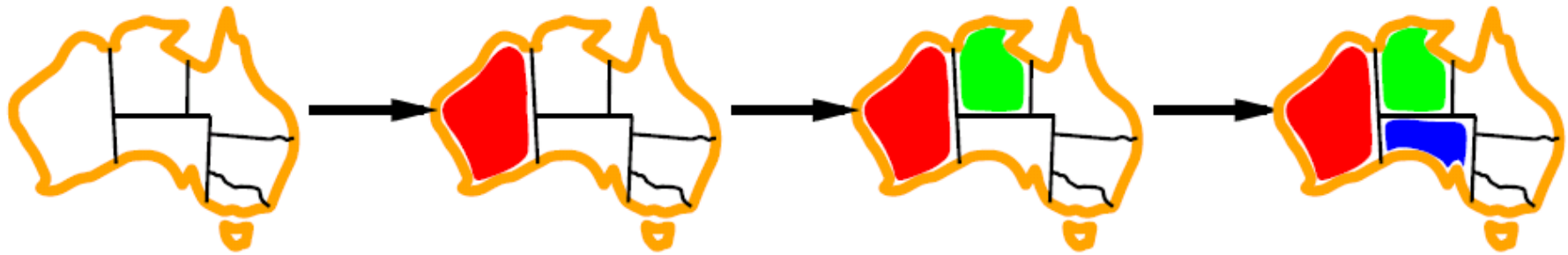
- minimum remaining values (MRV) heuristic
- degree heuristic

for each *value* **in** ORDER-DOMAIN-VALUES(*var*, *assignment*, *csp*) **do**

- **Value selection: the least-constraining**

- least-constraining value (LCV) heuristic

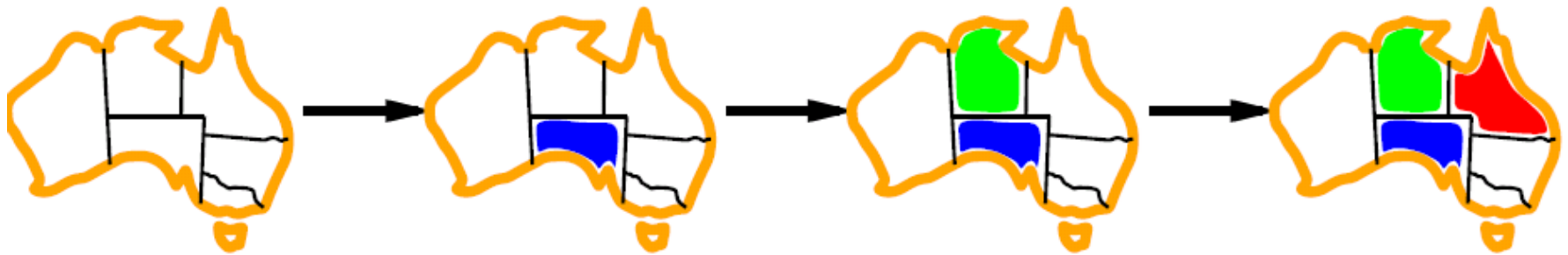
Minimum Remaining Values Heuristic



■ Heuristic mechanism

- rule: *choose the variable with the fewest legal values*
- known also as the “most-constrained variable” heuristic
- known also as the “fail-first” heuristic
- attempts to detect a possible failure early

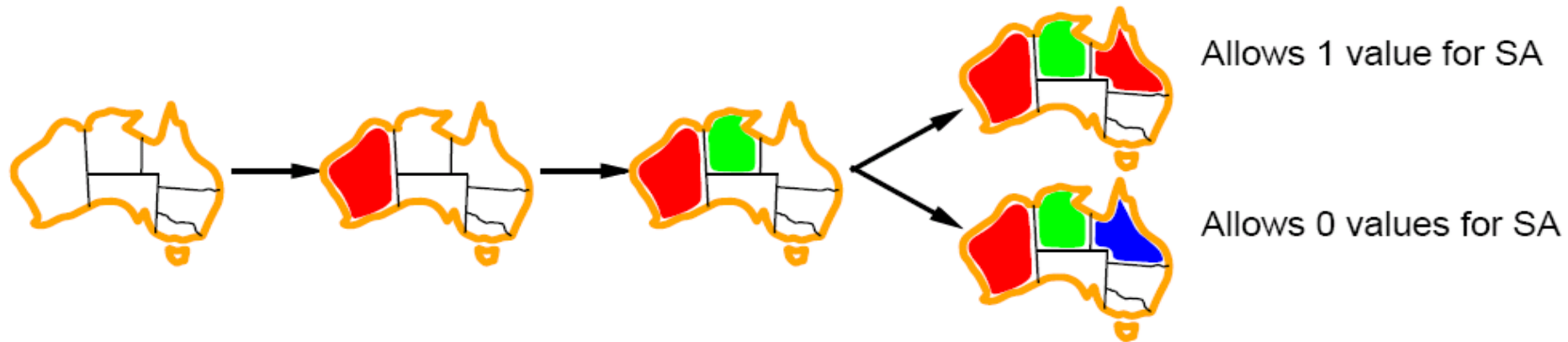
Degree Heuristic



■ Heuristic mechanism

- rule: *choose the variable involved in most constraints compared to the other unassigned variables*
- it relates to node degree in the constraint graph
- attempts to reduce the branching factor in future choices
- acts as tie-breaker in case of an equal number of legal values

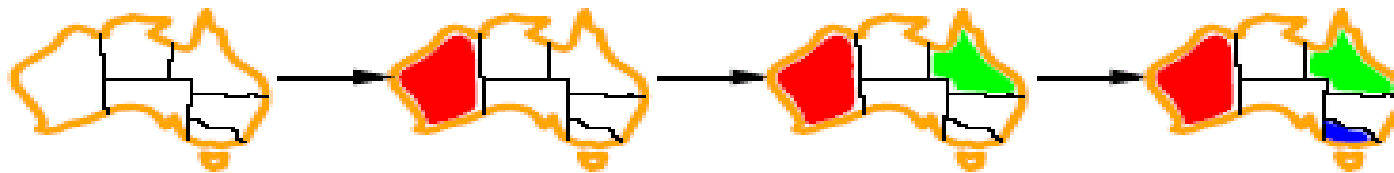
Least-Constraining Value Heuristic



■ Heuristic mechanism

- rule: *choose a value for the variable that excludes the fewest options for neighboring free variables*
- attempts to allow maximum flexibility in the next steps
- useful when we ask for only *one* solution, not all solutions

Forward Checking

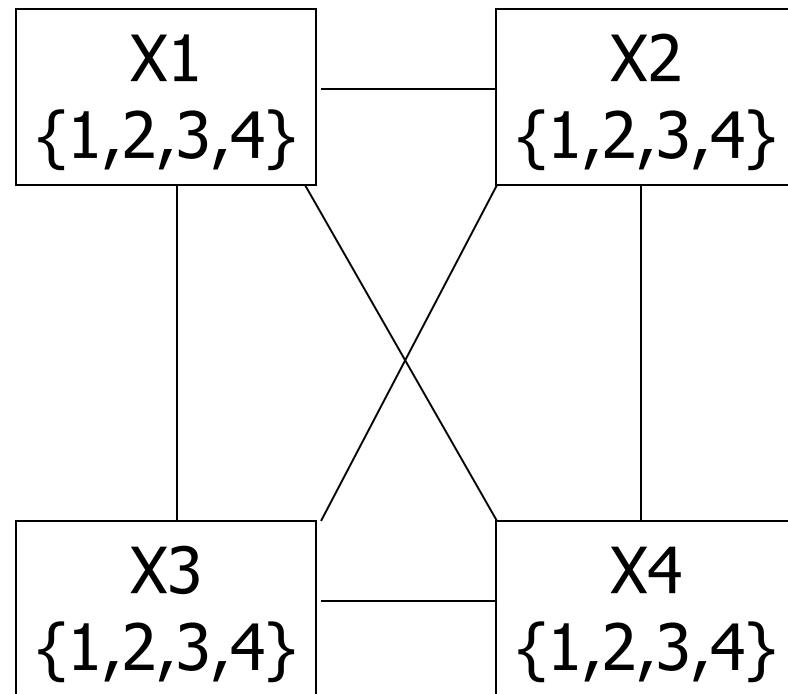


WA	NT	Q	NSW	V	SA	T

- rule: *when assigning a value to a variable X , we check the variables associated with X and remove values from their domains*
- propagation of information from bound to free variables
- makes better use of constraints

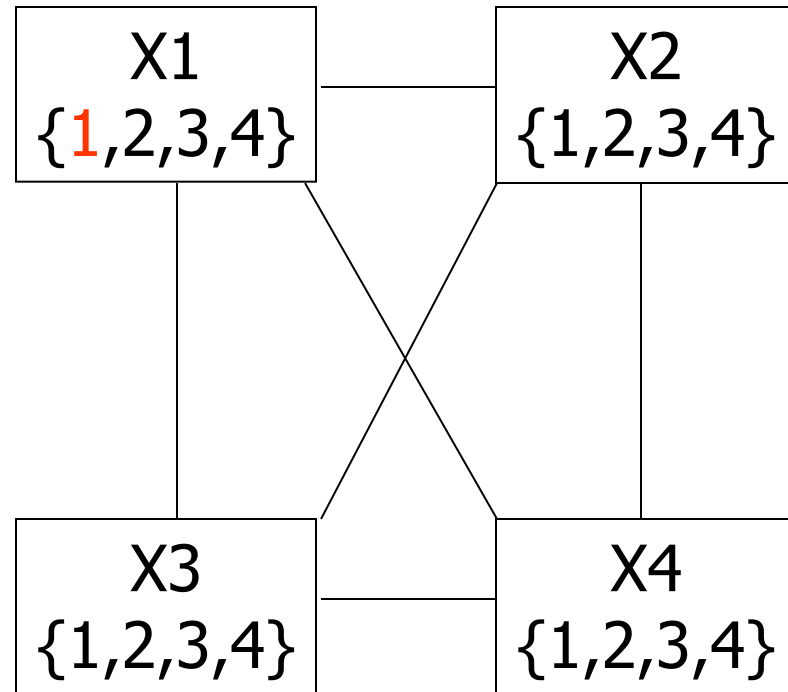
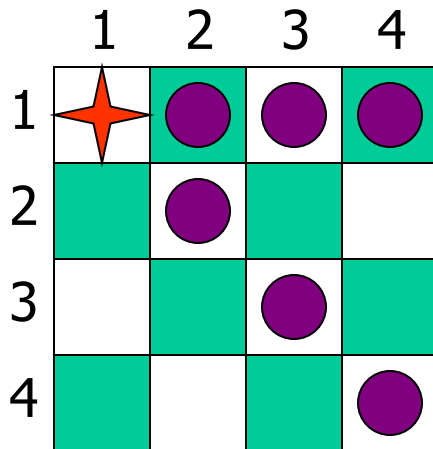
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



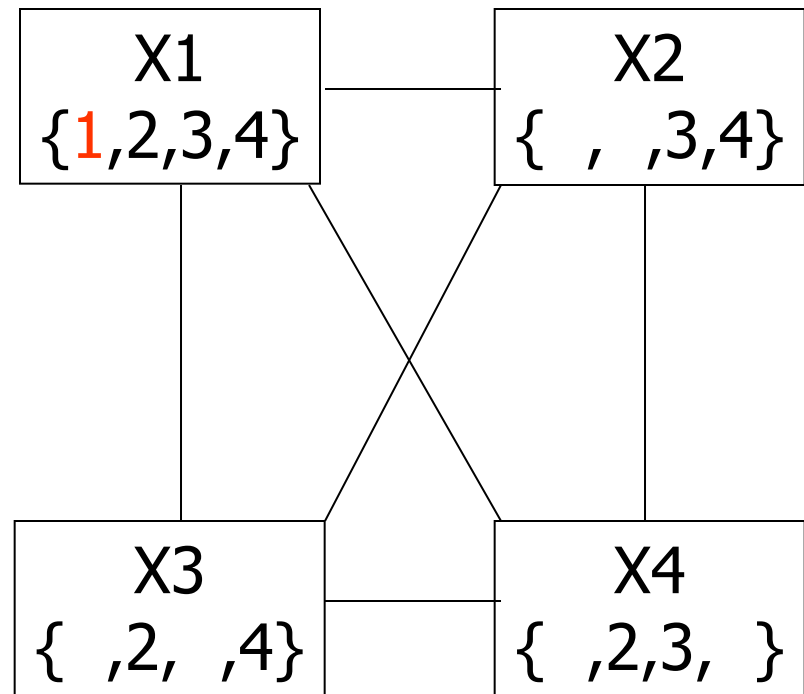
[these slides are taken from the CMSC 421 course of B. J. Dorr]

Example: 4-Queens Problem

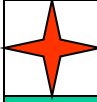
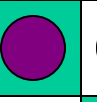
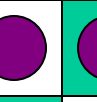
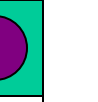
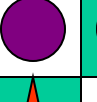
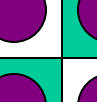


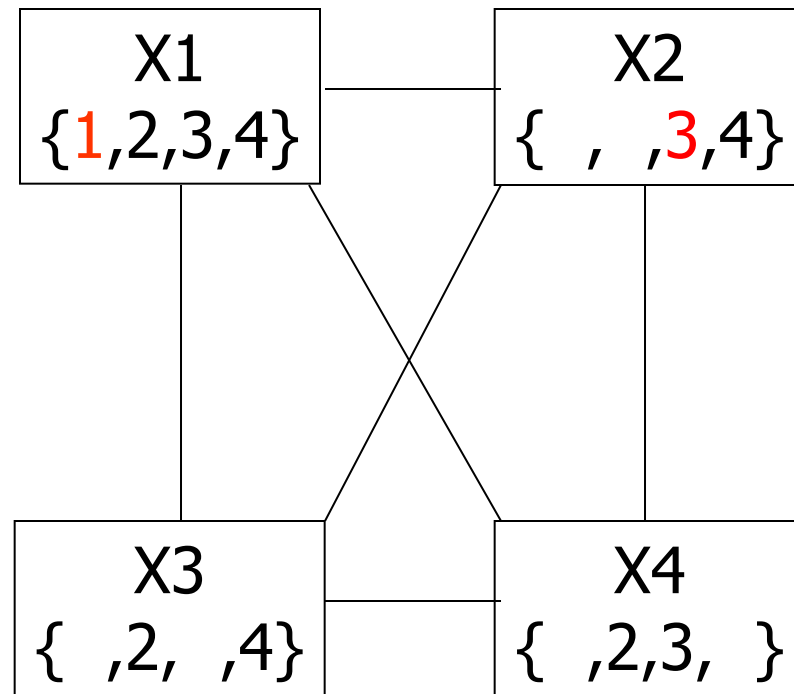
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				

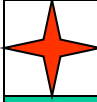
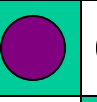
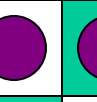
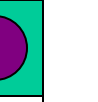
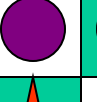
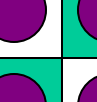


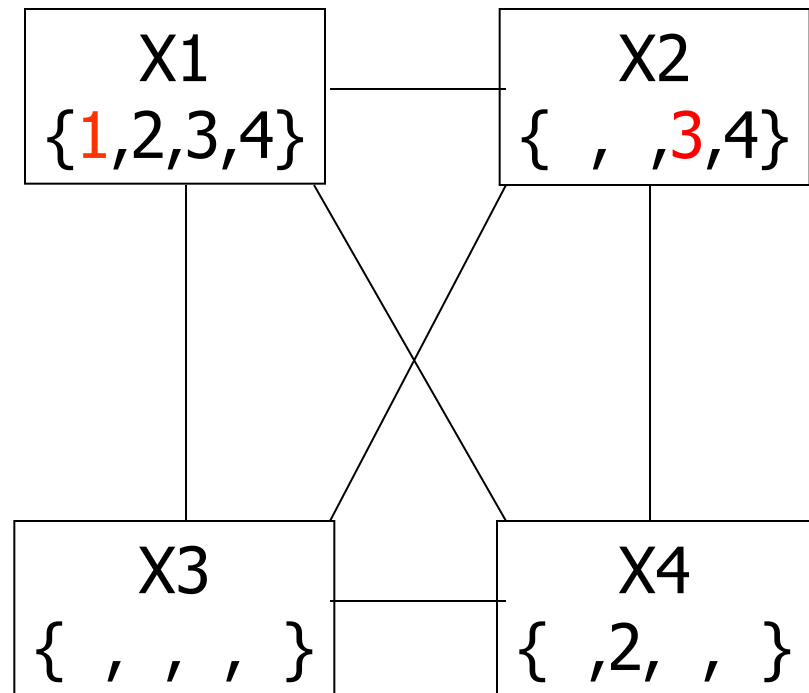
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				

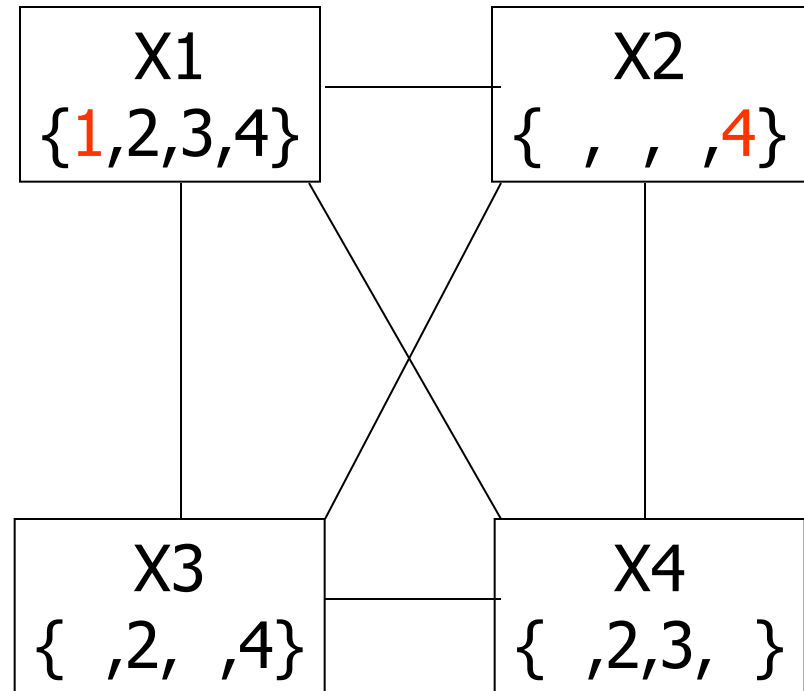
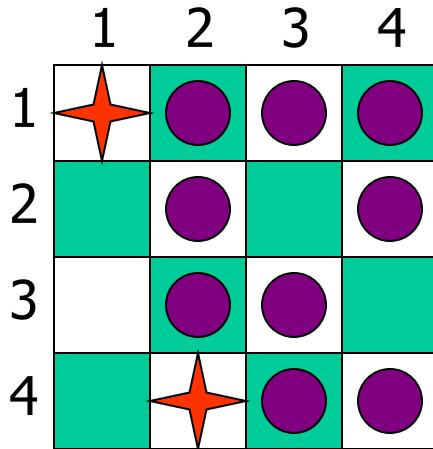


Example: 4-Queens Problem

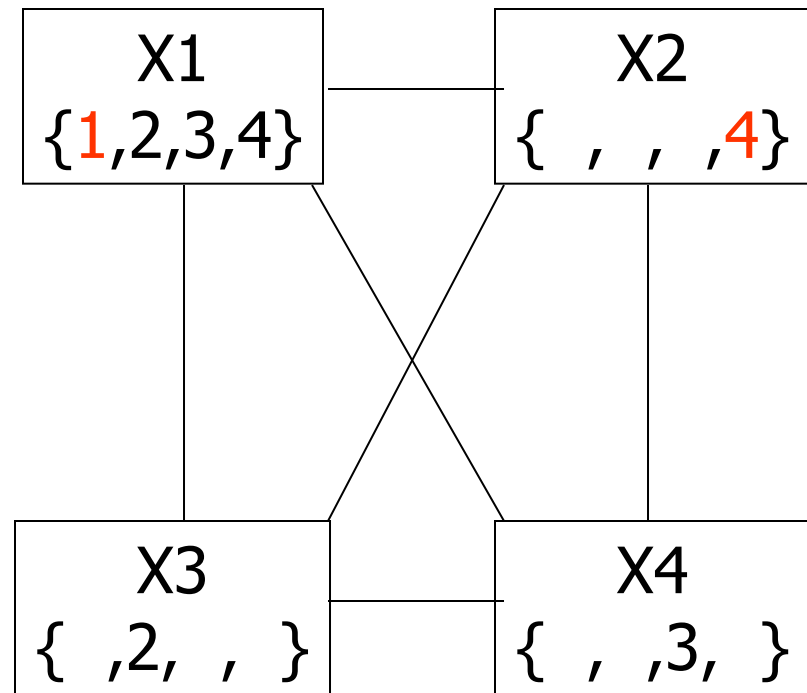
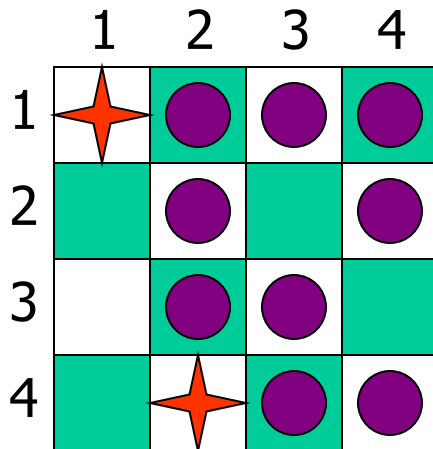
	1	2	3	4
1				
2				
3				
4				



Example: 4-Queens Problem

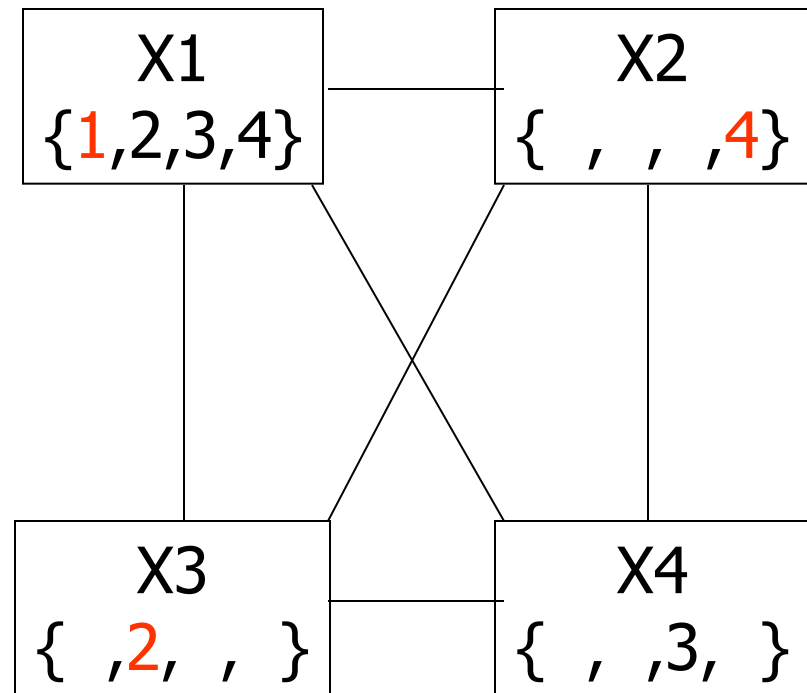


Example: 4-Queens Problem

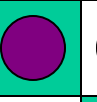
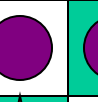
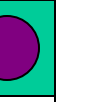
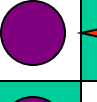


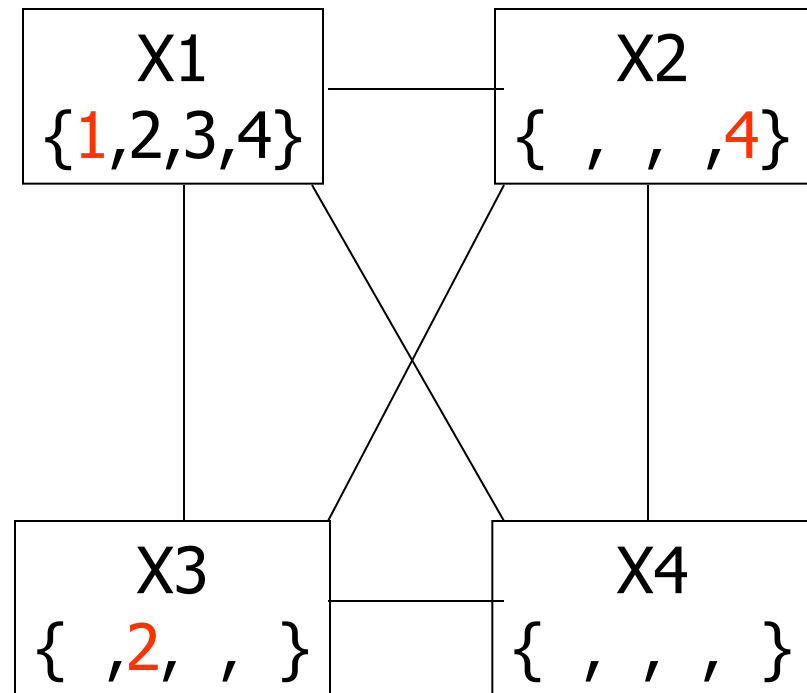
Example: 4-Queens Problem

	1	2	3	4
1	★	●	●	●
2		●	★	●
3		●	●	●
4		★	●	●



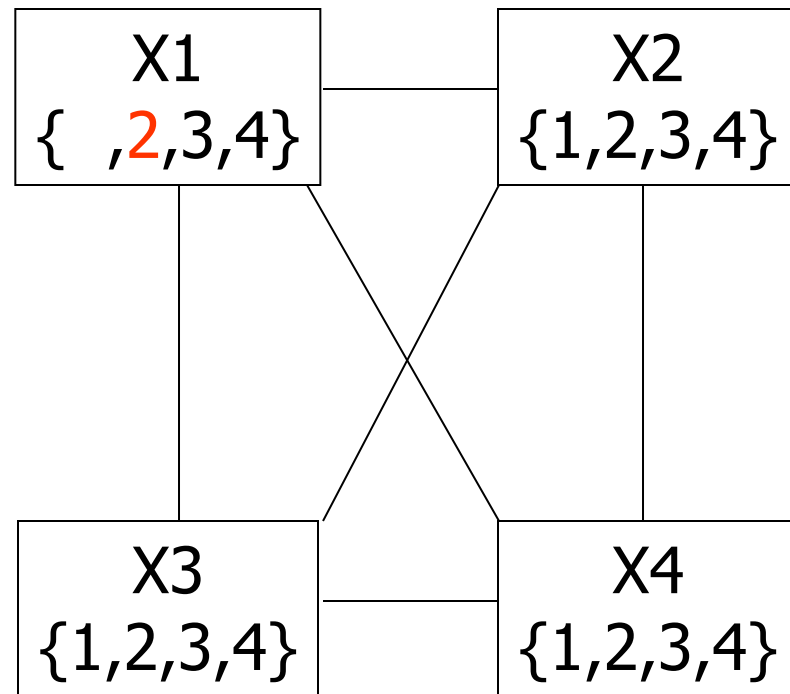
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



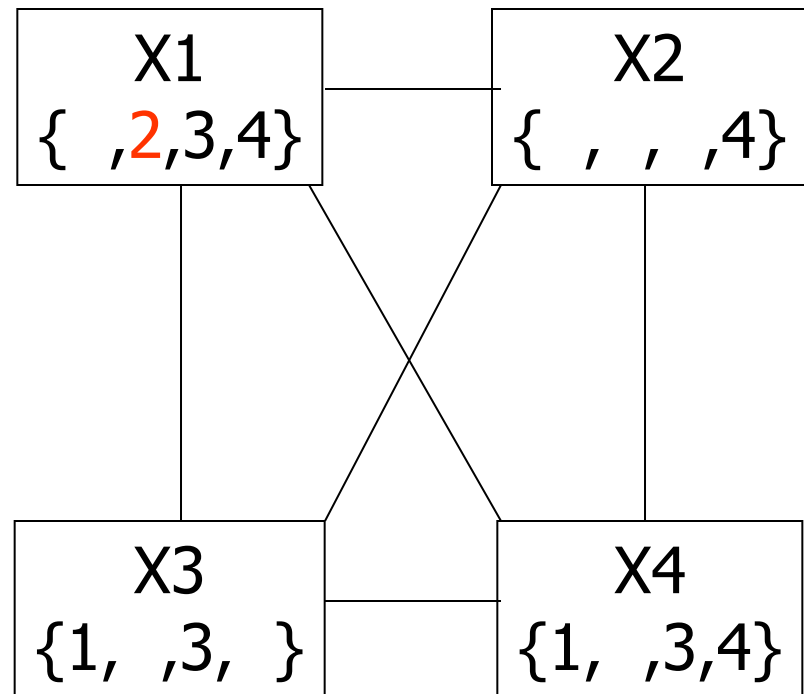
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



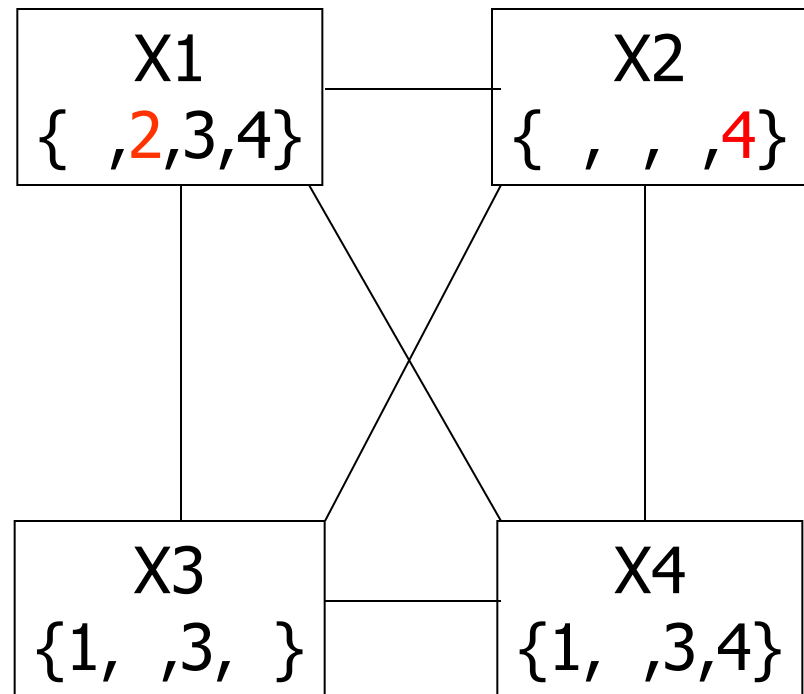
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



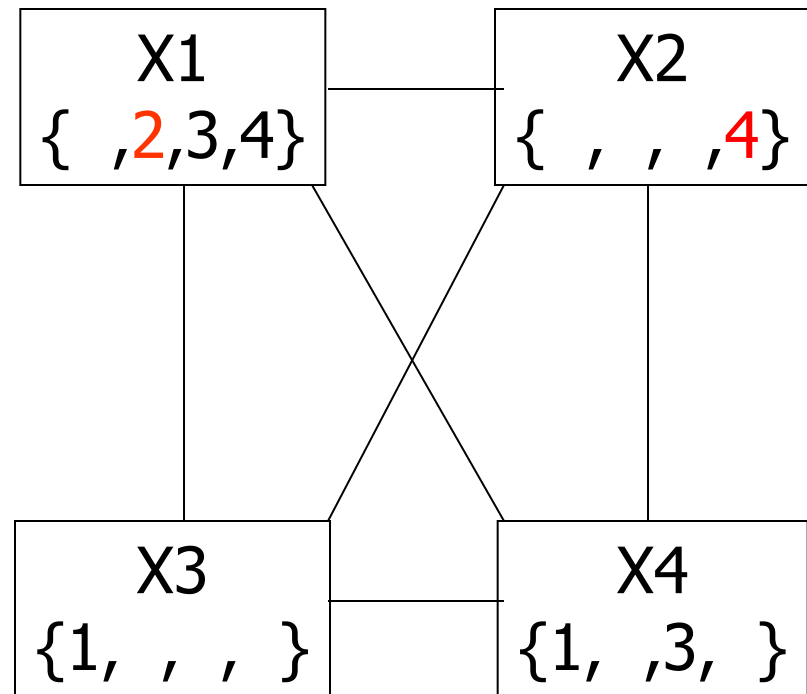
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



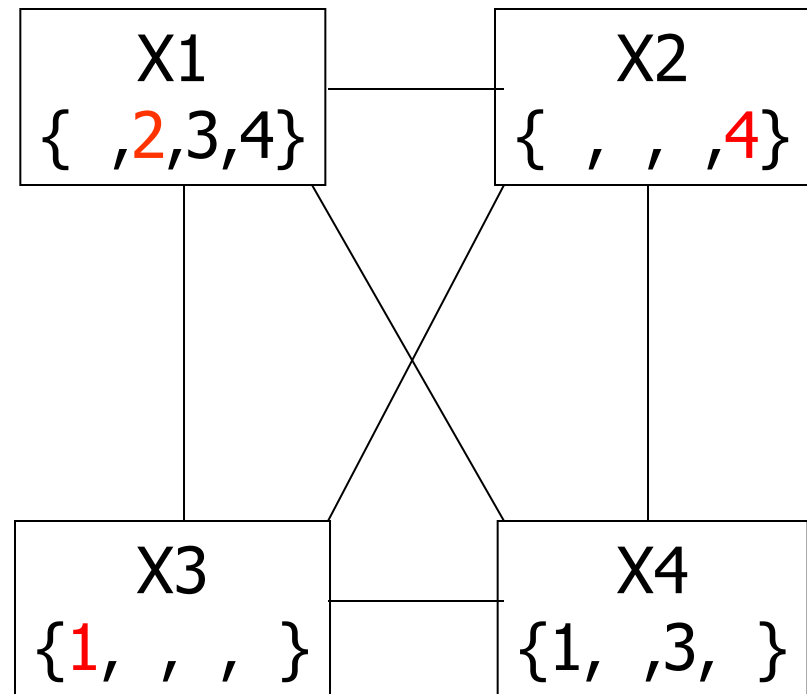
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



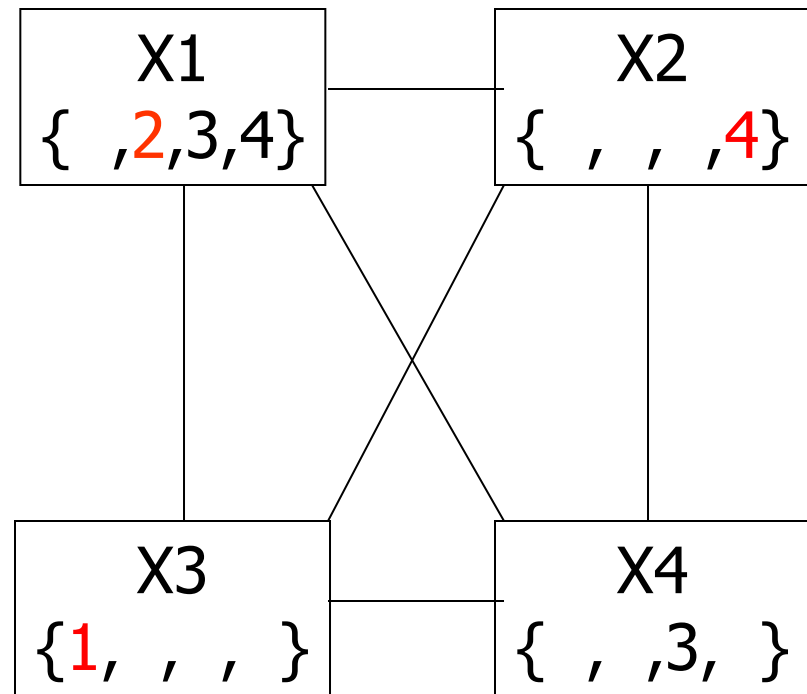
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



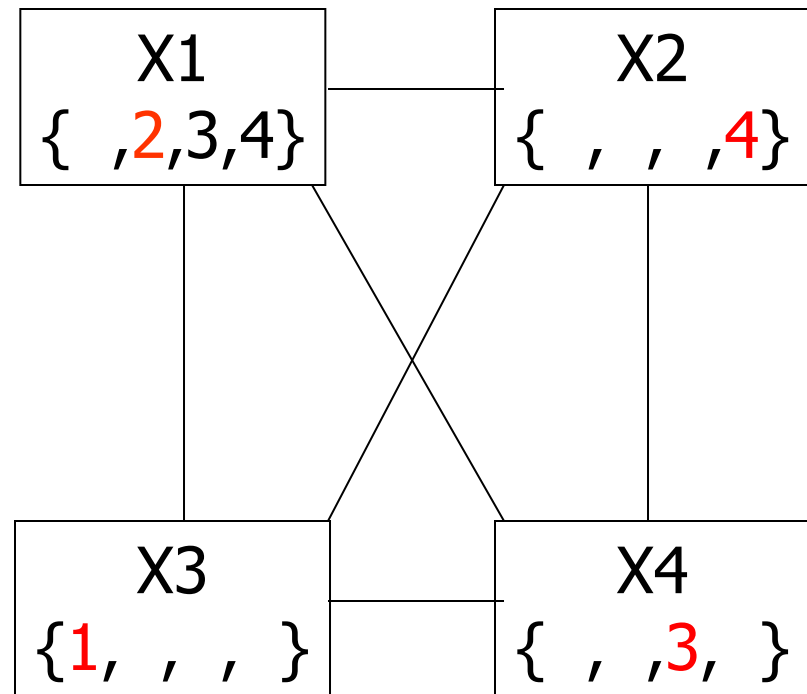
Example: 4-Queens Problem

	1	2	3	4
1				
2				
3				
4				



Example: 4-Queens Problem

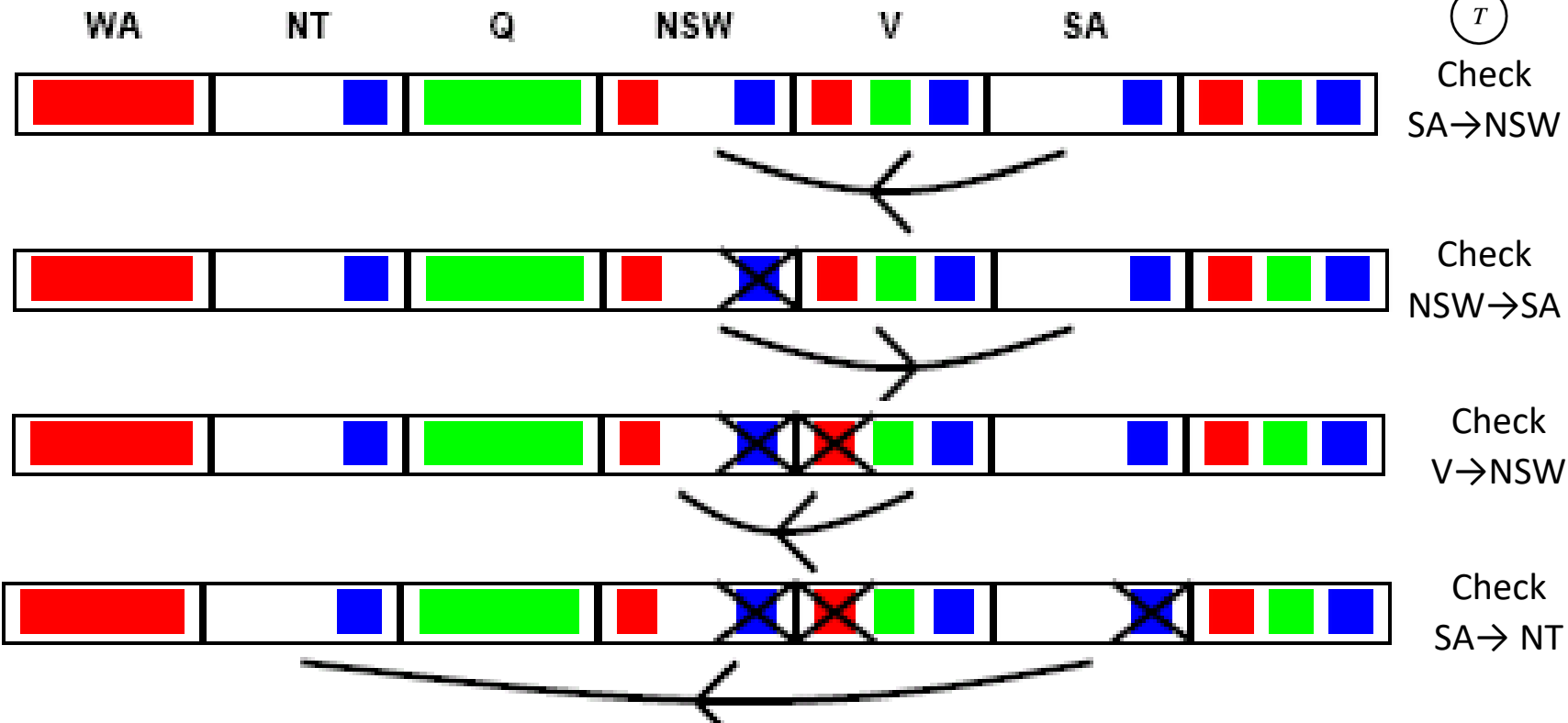
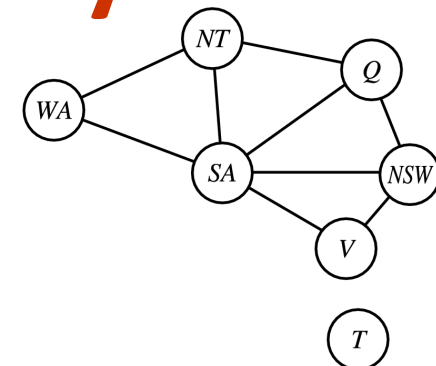
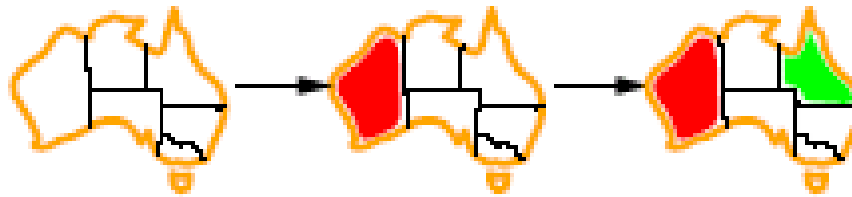
	1	2	3	4
1				
2				
3				
4				



Constraint Propagation

- recursive extension of forward checking to variables for which a value removal occurred at a connected variable
- **arc consistency, 2-consistency**
 - the arc $X \rightarrow Y$ (from the binary constraint $X-Y$) is consistent, if for *each* value of X there exists *some* legal value of Y
 - values of X that lead to 2-inconsistency are removed from X
- **k -consistency**
 - for any value assignment to some set of $k-1$ variables, there exists some legal value for the k -th variable
- **strong k -consistency**
 - k -consistency **and** $(k-1)$ -consistency **and** $(k-2)$ -... **and** 1-consistency

Example: Arc Consistency



Arc Consistency Algorithm

function AC-3(*csp*) **returns** the CSP, possibly with reduced domains

inputs: *csp*, a binary CSP with variables $\{X_1, X_2, \dots, X_n\}$

local variables: *queue*, a queue of arcs, initially all the arcs in *csp*

while *queue* is not empty **do**

$(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\textit{queue})$

if REMOVE-INCONSISTENT-VALUES(X_i, X_j) **then**

for each X_k **in** NEIGHBORS[X_i] **do**

 add (X_k, X_i) to *queue*

function REMOVE-INCONSISTENT-VALUES(X_i, X_j) **returns** true iff succeeds

removed $\leftarrow$ false

for each x **in** DOMAIN[X_i] **do**

if no value y in DOMAIN[X_j] allows (x, y) to satisfy the constraint $X_i \leftrightarrow X_j$

then delete x from DOMAIN[X_i]; *removed* $\leftarrow$ true

return *removed*

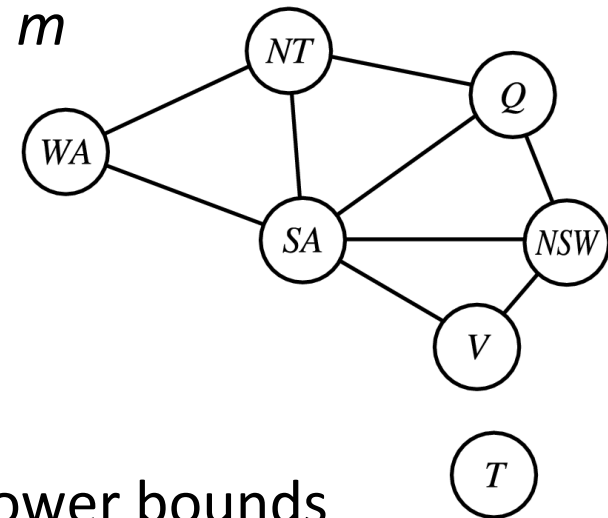
Special Constraints

■ *AllDiff* constraint

- count the number of different values for the variables
- *AllDiff*(NT, SA, Q) inconsistent with {WA=red, NSW=red}

■ Resource constraint: *atmost*

- $atmost(m, X_1, X_2, \dots, X_n) : X_1 + X_2 + \dots + X_n \leq m$
- e.g. *atmost*(10, A, B, C, D):
 - if each variable has domain {3,4,5,6}, the problem is inconsistent
 - if each variable has domain {2,3,4,5,6}, values 5 and 6 can be safely removed



■ Bound propagation

- consistency with respect to upper and lower bounds
 - Flight271 ∈ [0,165], Flight272 ∈ [0,385], Flight271+Flight272 ∈ [420,420]
 - new bounds after propagation: Flight271 ∈ [35,165], Flight272 ∈ [255,385]

Backjumping

■ Conflict set

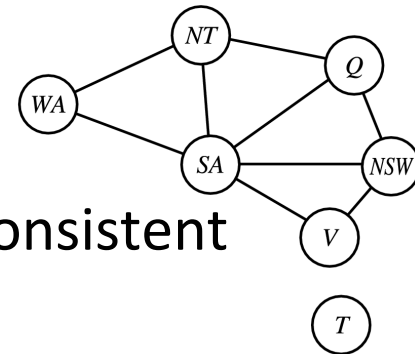
- for each variable Y , it is the set of variables with assigned values, which are connected to Y through constraints (can be built easily during forward checking)

■ Backtracking methods from variable Y

- chronological backtracking, if all assignments were tried and failed
 - back to the most recent variable (the one just before Y)
- backjumping, if all value assignments are inconsistent
 - back to the most recent variable of the conflict set of Y

■ Example

- $\{Q = \text{red}, \text{NSW} = \text{green}, V = \text{blue}, T = \text{red}\}$
- current step: value assignment to SA , all values inconsistent
- conflict set $SA = \{Q, \text{NSW}, V\}$, backjumping to V



Conflict-Directed Backjumping

- backjumping: when values are inconsistent with current assignment
- same thing could have been detected by forward checking
- backjumping is redundant when forward checking is applied

■ Conflict-Directed Backjumping

- assume value inconsistency at Y with conflict set $conflict(Y)$
- backjumping from Y to the most recent variable Z of $conflict(Y)$
- update to Z : $conflict(Z) \leftarrow conflict(Z) \cup conflict(Y) - \{Z\}$
- backjumping even from Z based on $conflict(Z)$ on failure!

■ Characteristics

- never chronological backtracking, even on failure
- $conflict(Y)$: set of variables responsible for inconsistency or failure

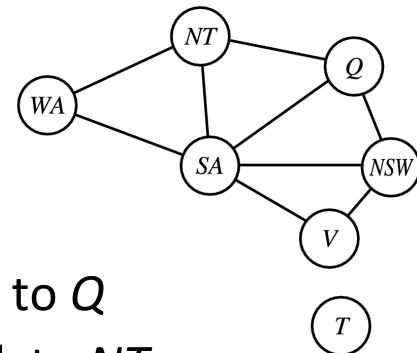
Example: Conflict-Directed Backjumping

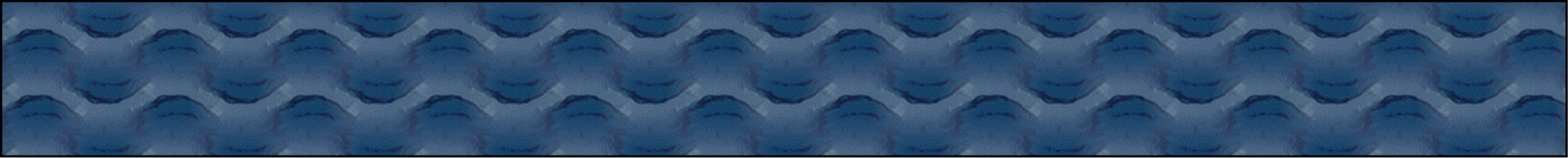
■ Conflict-Directed Backjumping

- assume value inconsistency at Y with conflict set $\text{conflict}(Y)$
- backjumping from Y to the most recent variable Z of $\text{conflict}(Y)$
- update to Z : $\text{conflict}(Z) \leftarrow \text{conflict}(Z) \cup \text{conflict}(Y) - \{Z\}$
- backjumping even from Z based on $\text{conflict}(Z)$ on failure!

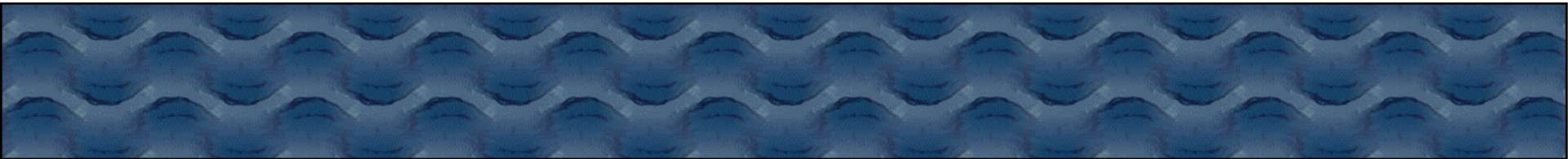
■ Example

- current assignment $\{WA=red, NSW=red, T=red\}$
- variable ordering: WA, NSW, T, NT, Q, SA, V
 - $\text{conflict}(SA)=\{WA, NSW, NT, Q\}$, inconsistency at SA , back to Q
 - $\text{conflict}(Q)=\{NSW, NT\} \Rightarrow \{WA, NSW, NT\}$, failure at Q , back to NT
 - $\text{conflict}(NT)=\{WA\} \Rightarrow \{WA, NSW\}$, failure at NT , back to NSW (!!!)
 - the source of failure was correctly detected from NT afterwards
- simple backjumping: after failure at NT , back to T ...





Local Search for CSPs



Local Search for CSPs

■ States

- complete value assignments to variables
- possible constraint violations

■ Local moves

- assignment of a new value to some variable

■ Choices

- variable: random choice
 - among the variables involved in conflicts
- value: minimum conflict
 - value that leads to minimum number of conflicts

Min-Conflicts Algorithm

function MIN-CONFLICTS(*csp*, *max_steps*) **returns** solution or failure

inputs: *csp*, a constraint satisfaction problem

max_steps, the number of steps allowed before giving up

current $\leftarrow$ an initial complete assignment for *csp*

for *i* = 1 **to** *max_steps* **do**

if *current* is a solution for *csp* **then return** *current*

var $\leftarrow$ a randomly chosen, conflicted variable from VARIABLES[*csp*]

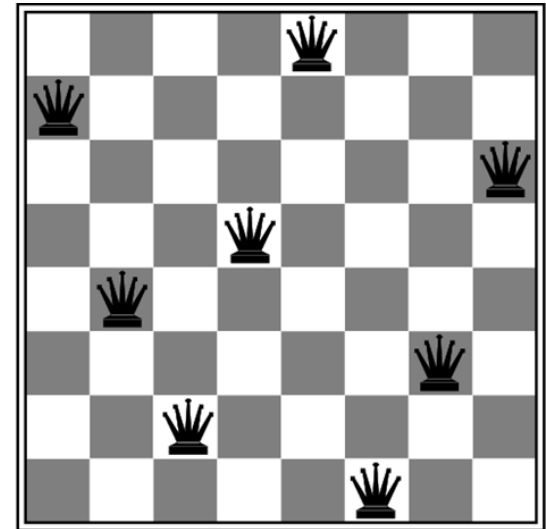
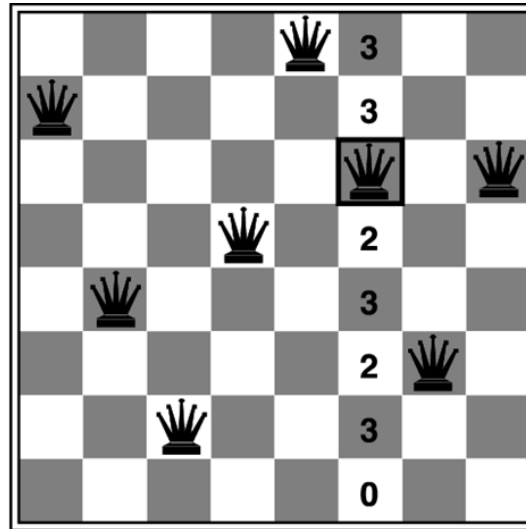
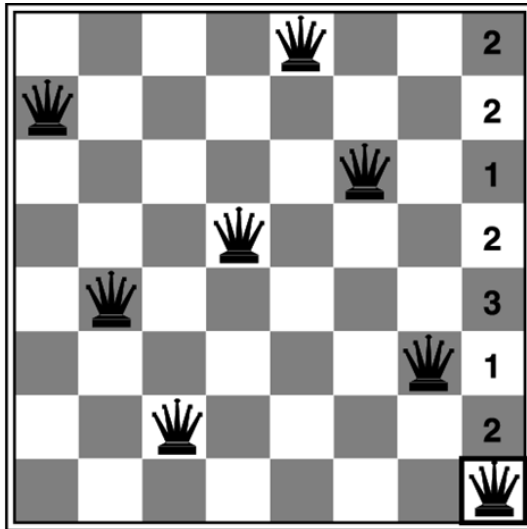
value $\leftarrow$ the value *v* for *var* that minimizes

 CONFLICTS(*var*, *v*, *current*, *csp*)

 set *var* = *value* in *current*

return *failure*

Min-Conflicts Example: n-Queens

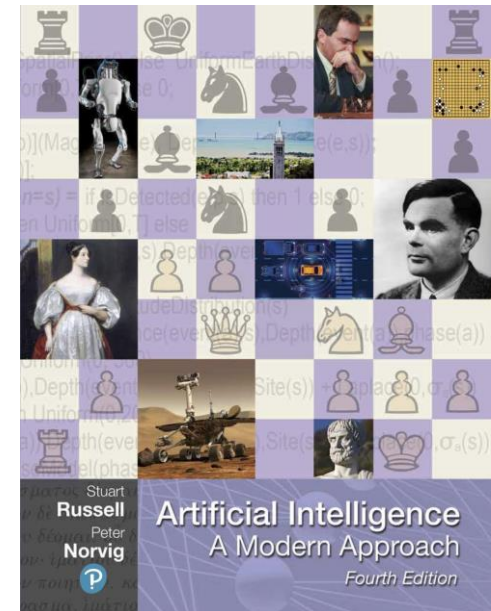


■ Characteristics

- very effective (though not complete)
 - 1,000,000,000 queens in 50 steps!
 - Hubble observation tasks-scheduling: 3 weeks-> 10 minutes
- suitable for “online” problems!

Related Bibliography

- **Artificial Intelligence: A Modern Approach,**
Stuart Russell and Peter Norvig,
4th edition, Prentice Hall, 2020
 - Sections 5.1 – 5.2



Lecture 4

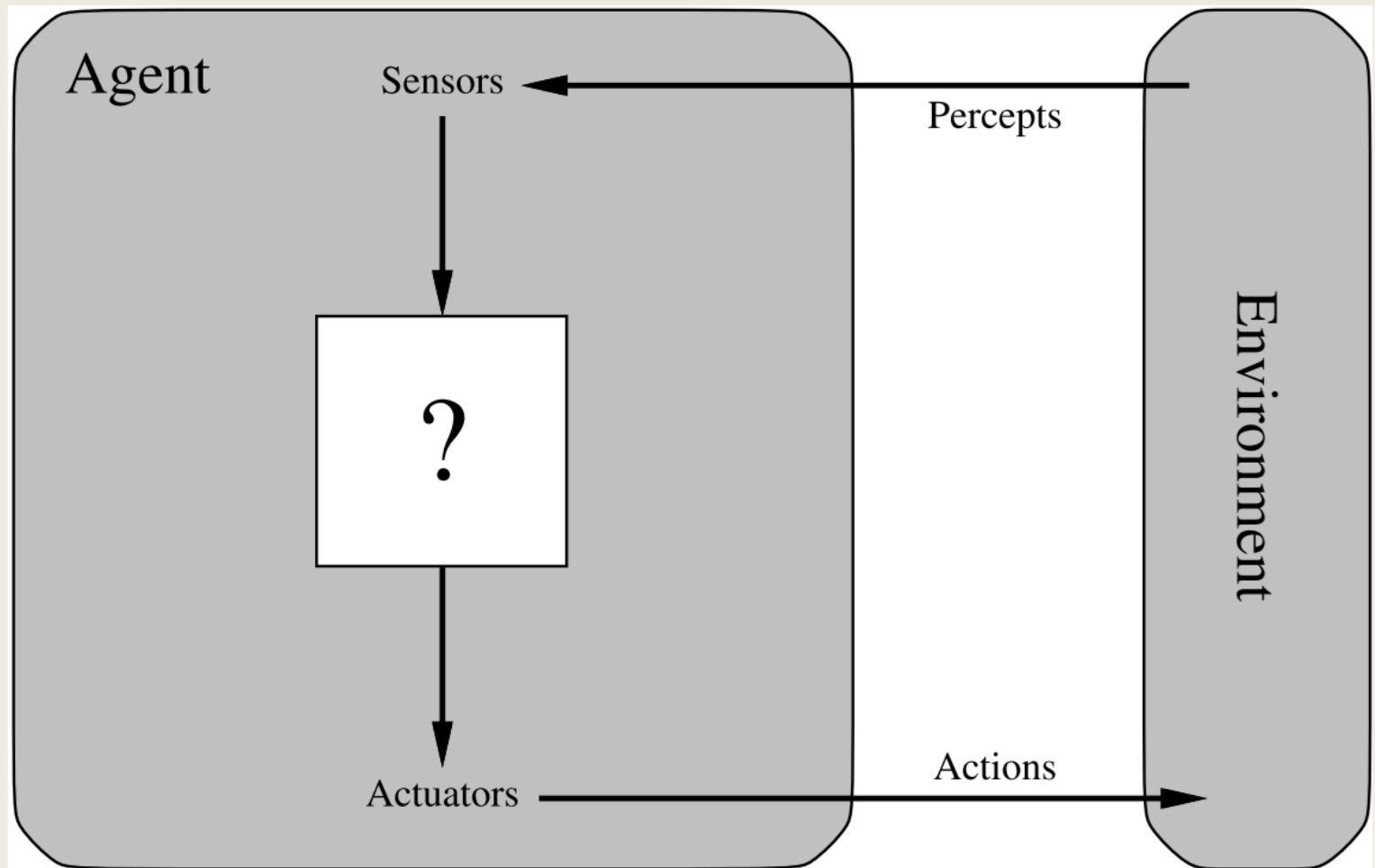
Statistical Artificial Intelligence

Michail G. Lagoudakis

**Intelligent Systems Laboratory
School of Electrical and Computer Engineering
Technical University of Crete
Chania, Crete, Greece**



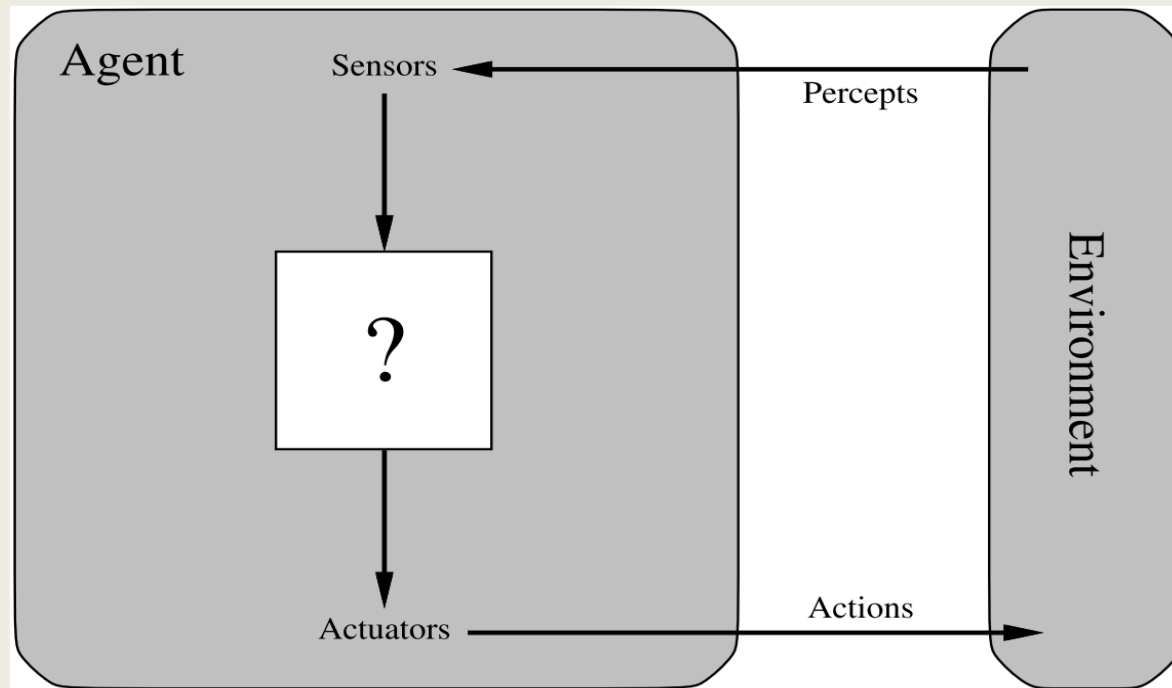
The Intelligent Autonomous Agent Paradigm



The Intelligent Autonomous Agent Paradigm

Symbolic AI (GOOF-AI)

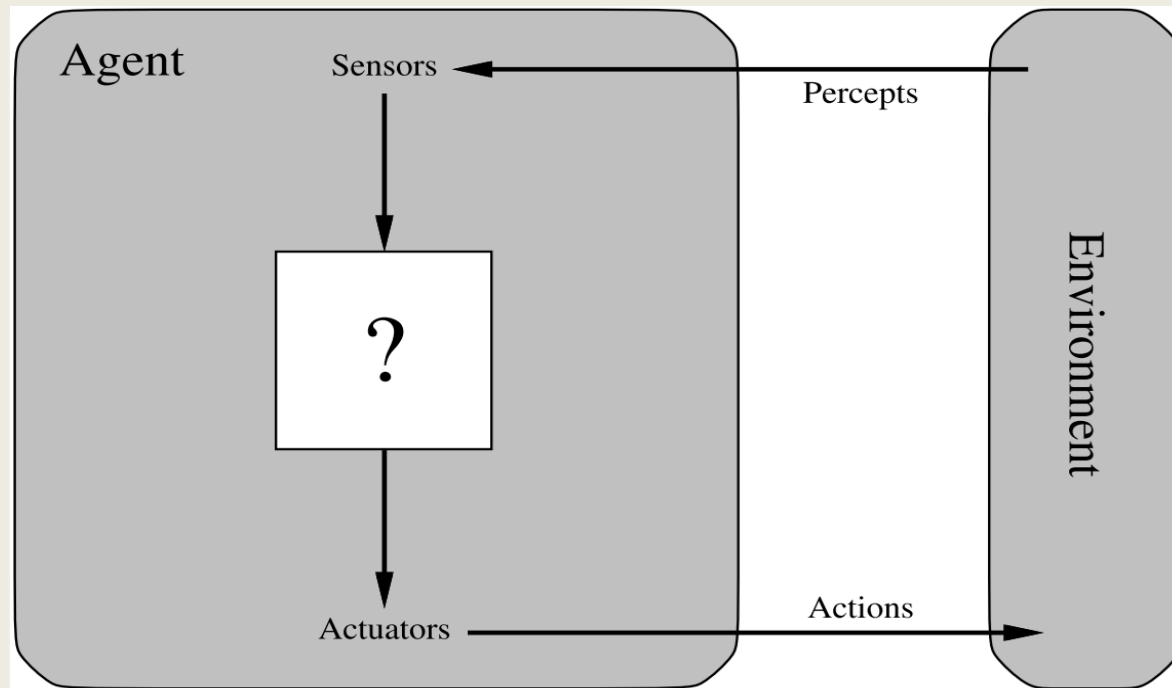
- since the very beginning in the 1960s
- everything can be represented using propositions and truth values
- reasoning is simply a procedure of logical inference
- ... using axioms of logic, logical inference, derivation of proofs, ...
- problems: huge needs of resources, no handling of uncertainty, ...



The Intelligent Autonomous Agent Paradigm

Statistical AI

- 1990s: let's embrace uncertainty + let's make AI work in the real world
- probability theory, utility theory, decision theory, information theory, statistics, numerical optimization, ...
- celebrated “tools”: Bayesian Networks, MDPs, HMMs, CNNs, LLMs, ...
- ...but also game theory, auctions, mechanisms for multi-agent systems



Bayesian Probabilities

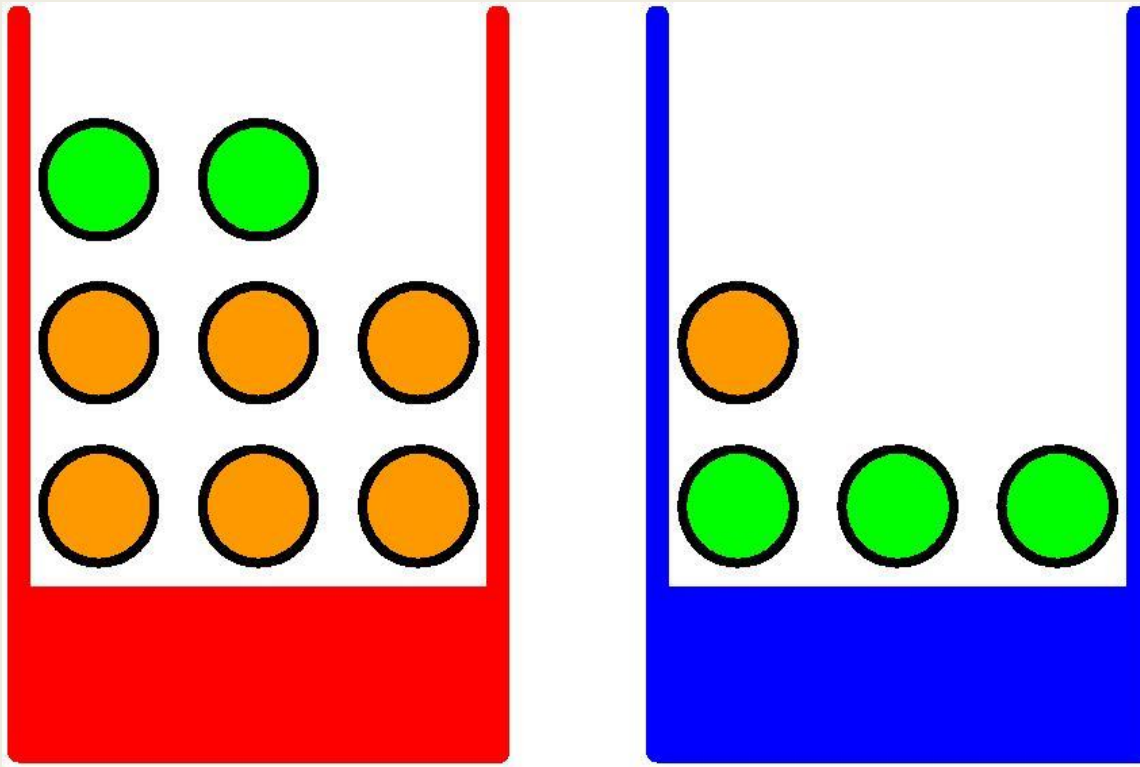
Bayes Rule

Bayesian Approach



Probability Theory

Apples and Oranges



Probability Theory

- **Procedure**

- choose a box: red (40%) or blue (60%)
- pick up a fruit: randomly (equally likely)
- replace the fruit and repeat

- **Random Variables**

- B : r or b , red or blue box
- F : a or o , apple or orange fruit
- probability of event: fraction of times it occurs in the limit

- **Queries**

- what is the probability of picking an apple?
- what is the probability that the box was red, given an apple?
- ...



Frequentist Probability

- **Frequentist probability** is the interpretation of probability that defines an event's probability as the limit of its relative frequency in a large number of trials.



Bayesian Probability

- **Bayesian probability** is the interpretation of the probability that supports the idea that probability can be defined as the degree to which a person (or community) believes that a proposition is true.



Bayesian vs Frequentist Probability

Both agree with the statement that $P(\text{heads})=1/2$.



- The **Bayesian** would say that given there are two sides to the coin, and there is no reason **to believe** that one side is more favored than the other (or if one is favored, there is no way to know which side is favored), then I am completely **uncertain** as to which side will fall, so the probability of it coming up heads must be $1/2$.
- The **frequentist** would say that if this is a fair coin and it would be flipped a very large number of times, then **the long term ratio of frequencies** of heads and tails is $1/2$.



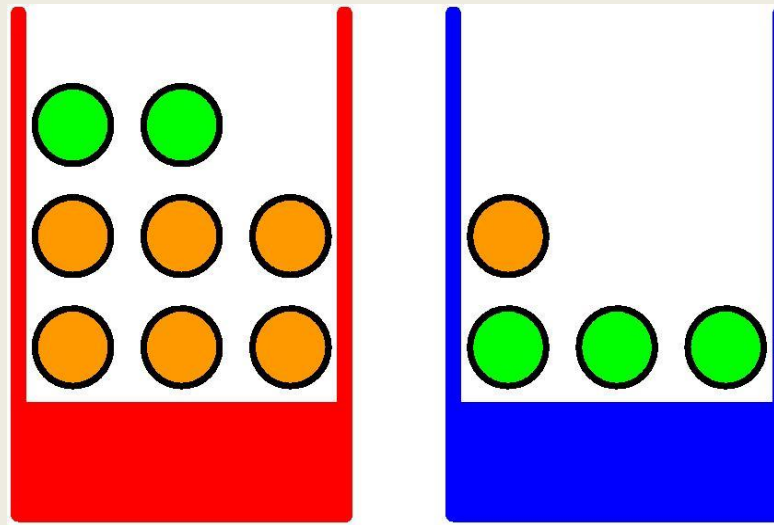
Towards Bayesian Probability

- **Frequentist probability**
 - frequencies of random, repeatable events
- **Considerations**
 - can we reason about rare, non-repeatable events?
 - example: will the Arctic ice cap disappear in 1000 years?
 - what if we have some idea about the rate of ice melting?
 - what if this knowledge is revised after 100 years?
 - need to quantify uncertainty
 - need to make revisions in light of new evidence
- **Bayesian probability**
 - quantification and revision of uncertainty



Bayesian Thinking

Apples and Oranges



- blue box most likely chosen, before looking at fruit (prior)
- red box most likely chosen, after picking an orange (posterior)

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$



Probability Theory as Extended Logic

- Framework for **plausible reasoning** developed by three physicists:
 - Harold Jeffreys (1891 – 1989)
 - Richard Threlkeld Cox (1898 - 1991)
 - Edwin Thompson Jaynes (1922 – 1998)



Probability Theory as Extended Logic

1. Degrees of **plausibility** are represented by real numbers.
2. Qualitative correspondence with common sense:
 - Increasing the **plausibility** of a statement, decreases the **plausibility** of its negative.
3. Consistency:
 - If a conclusion can be reasoned out in more than one way, then every possible way must lead to the same result.
 - It must be possible to incorporate all the evidence relevant to a question into the reasoning process.
 - All equivalent states of knowledge are represented by equivalent **plausibility** assignments.



Probability Theory as Extended Logic

- These simple statements (known as Cox's Theorems) mean that the mathematical description of **plausible reasoning** is completely determined.
 - It turns out to be exactly the same as that of standard probability theory
 - Probability theory is a general form of logic where statements are not just true or false, but have degrees of certainty.
 - ...therefore, *use probability to measure beliefs*



Bayes Theorem

- **Rev. Thomas Bayes (1702 – 1761)**
 - a British mathematician and Presbyterian minister



Bunhill Fields Cemetery in London



Bayes Theorem

Likelihood
(the Bayes factor)

Prior

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

Posterior

Evidence



Bayes Theorem: Derivation

From conditional probabilities ...

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

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

... we get:

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

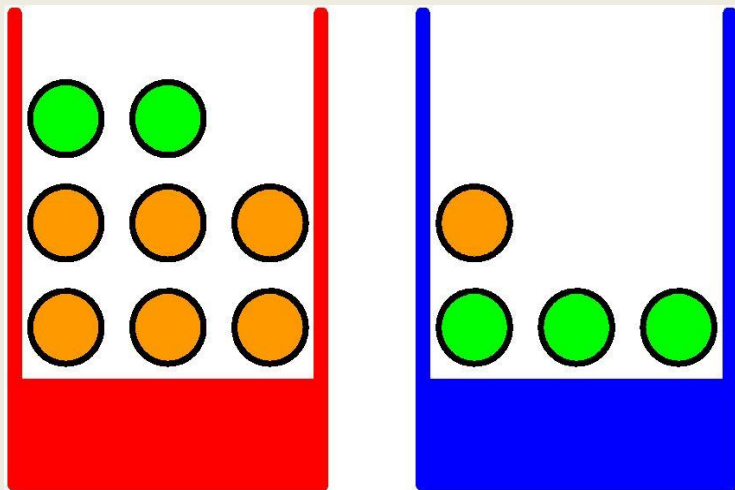
(this is known as the *product* or *chain rule*)

... and then, rearranging:

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



Probability Examples



$$p(B = r) = 4/10 \quad p(B = b) = 6/10$$

$$p(F = a|B = r) = 1/4$$

$$p(F = o|B = r) = 3/4$$

$$p(F = a|B = b) = 3/4$$

$$p(F = o|B = b) = 1/4$$

$$p(F = a) = p(F = a|B = r)p(B = r) + p(F = a|B = b)p(B = b) = 11/20$$

$$p(F = o) = p(F = o|B = r)p(B = r) + p(F = o|B = b)p(B = b) = 9/20$$

$$p(B = b|F = a) = \frac{p(F = a|B = b)p(B = b)}{p(F = a)} = \frac{(3/4)(6/10)}{11/20} = \frac{9}{11}$$



Why is this useful?

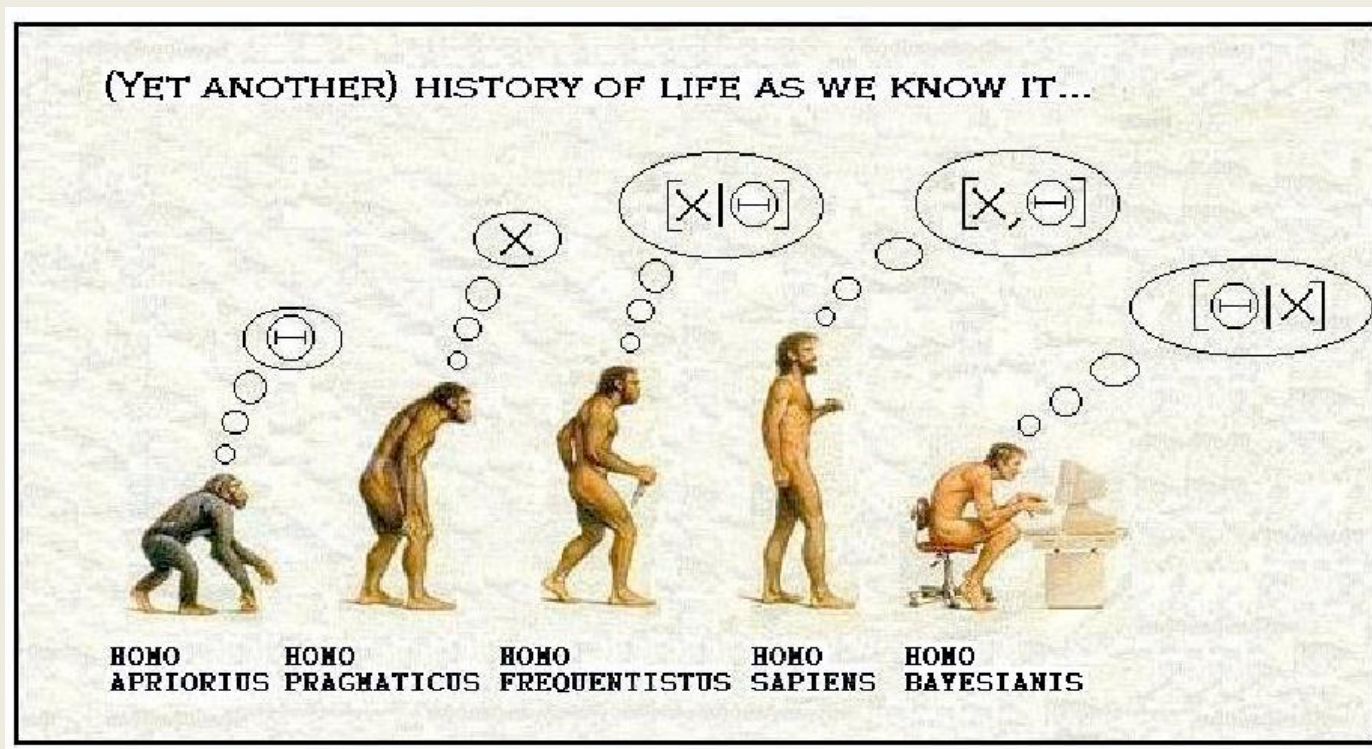
$$P(Y | X) = \frac{P(X | Y)P(Y)}{P(X)}$$

- Because usually $P(X | Y)$
 - is easier to obtain ... **X is data (evidence / effects / symptoms)**
 - provided **given an assumption (hypothesis / cause)**
- In many cases we have info on: $P(\text{effects} | \text{cause})$
- but we really care to find this: $P(\text{cause} | \text{effects})$
- Bayes rule turns a diagnostic probability to a causal one!

$$P(\text{cause} | \text{effects}) = \frac{P(\text{effects} | \text{cause})P(\text{cause})}{P(\text{effects})}$$



The Bayesian Approach



- The Homo Apriorius establishes the probability of a hypothesis, no matter what data says.
- The Homo Pragmaticus is interested in the probability of the data only.
- The Homo Frequentist measures the probability of the data, given the hypothesis.
- The Homo Sapiens is interested in the probability of the data and of the hypothesis.
- The Homo Bayesian measures the probability of the hypothesis, given the data.

with thanks to M. West and to S. Andreon



Frequentists vs. Bayesians

- **Frequentist approach**

- measures the probability of the data, given the hypothesis
- predictions may be misled (e.g. three coin tosses, all heads)
- being somewhat objective (just believe the data)
- parameters are fixed and unknown constants (point estimates)
- uncertainty is quantified by confidence intervals
- predictions using point estimates of the parameters

- **Bayesian approach**

- measures the probability of the hypothesis, given the data
- makes smoother predictions (e.g. three coin tosses, all heads)
- being somewhat subjective (dependence on prior belief)
- parameters are described as probability distributions
- uncertainty is directly included in the distribution
- predictions by marginalization over the space of parameters



Example 1



- Suppose that Bob can decide to go to work by one of three modes of transportation, car, bus, or train.
 - Because of high traffic, if he decides to go by car, there is a 50% chance he will be late.
 - If he goes by bus, which has special reserved lanes but is sometimes overcrowded, the probability of being late is only 20%.
 - The train is almost never late, with a probability of only 1%, but is more expensive than the bus.



Example 1

- Suppose that Bob is late one day, and his boss wishes to estimate the probability that he drove to work that day by car.
- Since he does not know which mode of transportation Bob usually uses, he gives a prior probability of 1 of 3 to each of the three possibilities.
- What is the boss' estimate of the probability that Bob drove to work?



Example 1

- First list all the facts available:

$$P(\text{car}) = \frac{1}{3}$$

$$P(\text{late}|\text{car}) = \frac{1}{2}$$

$$P(\text{bus}) = \frac{1}{3}$$

$$P(\text{late}|\text{bus}) = \frac{1}{5}$$

$$P(\text{train}) = \frac{1}{3}$$

$$P(\text{late}|\text{train}) = \frac{1}{100}$$



Example 1

- Then apply Bayes theorem:

$$P(\text{car}|\text{late}) = \frac{P(\text{late}|\text{car})P(\text{car})}{P(\text{late})}$$

$$= \frac{P(\text{late}|\text{car})P(\text{car})}{P(\text{late}|\text{car})P(\text{car}) + P(\text{late}|\text{bus})P(\text{bus}) + P(\text{late}|\text{train})P(\text{train})}$$

$$= \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{2} \times \frac{1}{3} + \frac{1}{5} \times \frac{1}{3} + \frac{1}{100} \times \frac{1}{3}}$$

$$\approx 0.704$$

P(evidence) is given by the Law of Total Probability:

a “weighted average” that captures the overall probability of an outcome that can be realized via several distinct events



Example 2

- A clinical test, designed to diagnose a specific illness, comes out positive for a certain patient.
- We are told that:
 - The test is 79 percent reliable
(that is, it misses 21 percent of actual cases)
 - On average, this illness affects 1 percent of the population in the same age group as the patient
 - The test has a false positive rate of 10 percent.
- Taking this into account and assuming you know nothing about the patient's symptoms or signs, what is the probability that this patient actually has the illness?



Example 2

- First list all the facts available:

$$P(D) = 0.01$$

$$P(T|D) = 0.79$$

$$P(T|\neg D) = 0.1$$



Example 2

- Then apply Bayes theorem:

$$\begin{aligned}P(D|T) &= \frac{P(T|D)P(D)}{P(T)} \\&= \frac{P(T|D)P(D)}{P(T|D)P(D) + P(T|\neg D)P(\neg D)} \\&= \frac{0.79 \times 0.01}{0.79 \times 0.01 + 0.1 \times 0.99} \\&\approx 0.074\end{aligned}$$

A good argument against random screening if the test does not discriminate very well!



Optimal Learning: Overview

- Behavior that maximizes expected total reward, while interacting with an uncertain world.
- Behave well while learning, learn while behaving well.
- Optimal learning $\Leftrightarrow$ using a Bayesian approach



Optimal Learning: Bayesian Approach

Bayesian approach

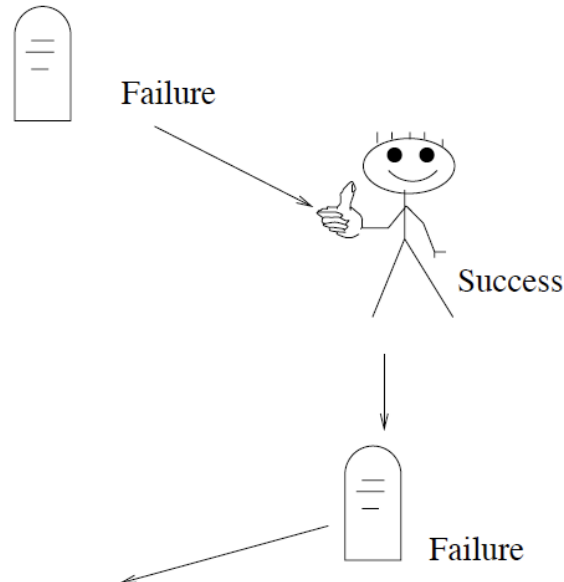
- Evolve uncertainty about *unknown process parameters* of the world model (e.g., transitions, rewards, ...)
- Uncertainty over the parameters is captured by *prior distributions*.
- The values of the (hyper-) parameters of these priors correspond to *information states* (describing our uncertainty about the world)
- Using the prior, the likelihood of observed data, and Bayes rule ...
- ... the belief is updated and a *posterior distribution* is formed,
- ... which serves as the new prior distribution in the next step!
- ... still capturing our (reduced) uncertainty about the world!



A simple example

$$\theta_1 = \Pr\{\text{Cure} \mid \text{Drug 1}\}$$

$$\theta_2 = \Pr\{\text{Cure} \mid \text{Drug 2}\}$$



example due to
M. O' Gordon Duff (2002)

- Bernoulli process parameters θ_1 , θ_2 describe the actual (but unknown) probabilities of success
- **Bayesian approach:** Uncertainty about parameters?
describe it by conjugate prior distributions



Conjugate Priors

- A prior is **conjugate** to a likelihood function, if the posterior is in the same family with the prior.

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

- Prior in the family $\Rightarrow$ posterior in the family
- A simple update of the **hyper-parameters** is typically enough to get the posterior!

- **Remark:** The uncertainty over the parameters is captured by these *prior-turned-posterior* distributions.

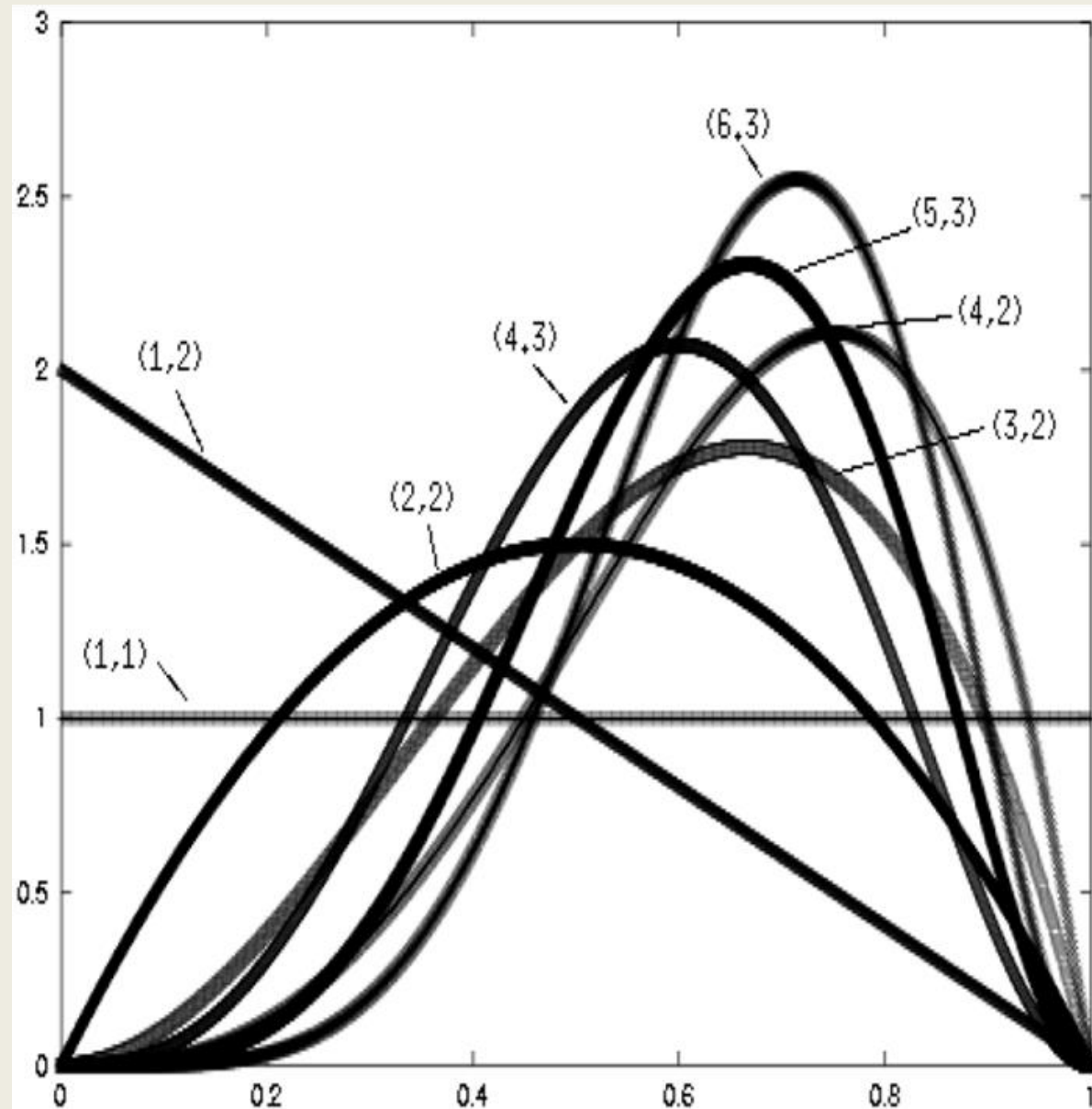


Distribution Beta(α, β)

$$\frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}$$

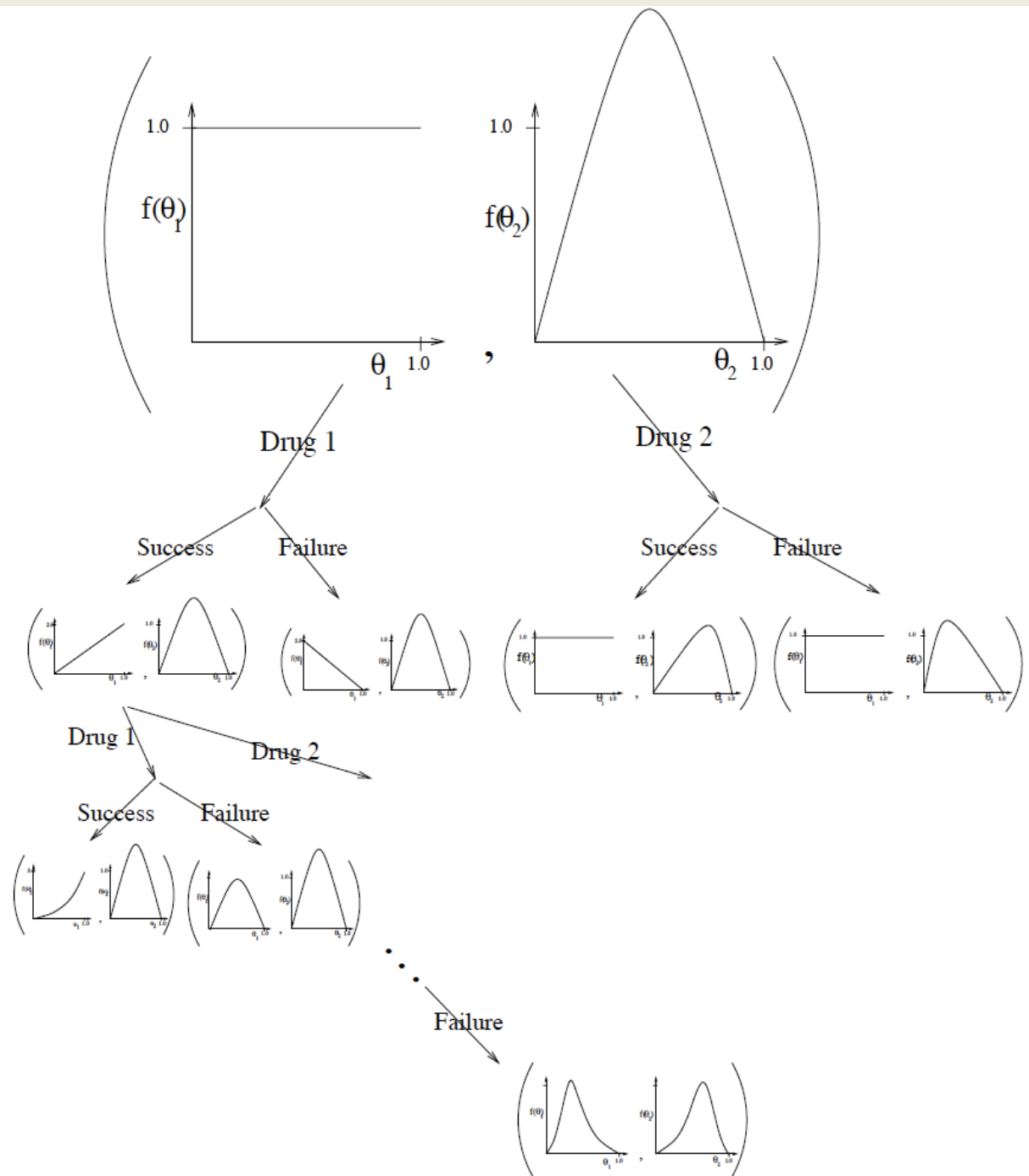
$$B(\alpha, \beta)$$

$$\text{where } B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)}$$



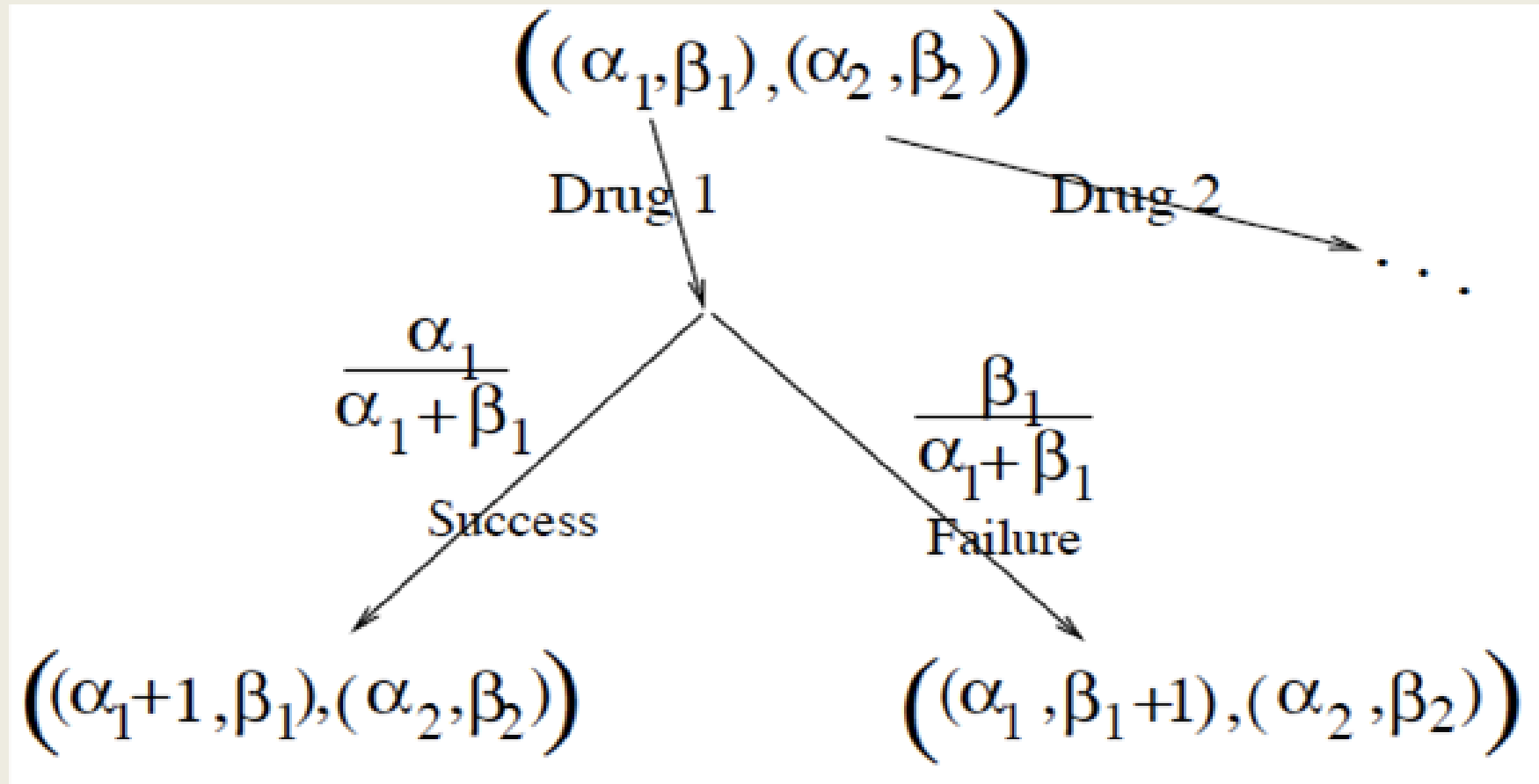
example due to
M. O' Gordon Duff (2002)

Information- State Transition Diagram



example due to
M. O' Gordon Duff (2002)

For Beta distribution, it simply becomes



example due to
M. O' Gordon Duff (2002)



Acting under Uncertainty



What should an agent do when operating under uncertainty? Decision theory to the rescue

Decision Theory = Utility Theory + Probability Theory

```
function DT-AGENT(percept) returns an action
  persistent: belief_state, probabilistic beliefs about the current state of the world
               action, the agent's action

  update belief_state based on action and percept
  calculate outcome probabilities for actions,
    given action descriptions and current belief_state
  select action with highest expected utility
    given probabilities of outcomes and utility information
  return action
```



A Definition of Rationality

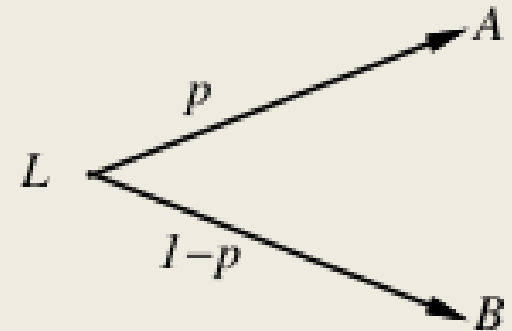
An agent is rational *if-f* it picks its actions adhering to the **maximum expected utility principle**:

*when choosing among (stochastic) actions,
always pick the action with maximum expected utility.*



Justification: Lotteries and Stochastic Actions

- A standard lottery L leads to a prize A with probability p , and to a prize B with probability $1-p$.



- Generally: $L = \{(p_1, X_1), (p_2, X_2), \dots, (p_n, X_n)\}$
- If an outcome X_i has utility $U(X_i)$...
- ... then Lottery L has expected utility $U(L) = \sum p_i U(X_i)$
- Due to the Ramsey-von Neumann and Morgenstern expected utility theorem, you can construct a real-valued utility function that corresponds to a lottery's expected utility, *such that:*

$$U(L1) \geq U(L2) \text{ if-} f \text{ } L1 \text{ preferable than } L2$$

- **Thus, if $U(L1) > U(L2)$, which lottery would you prefer?**
- Now, think of a lottery as an action with uncertain outcomes
- Hence, the *Maximum Expected Utility Principle*



Inference and Decision

- **Inference**

- input $\mathbf{x}$, target output $\mathbf{t}$
- determine either $p(\mathbf{x}, \mathbf{t})$ or $p(\mathbf{t} | \mathbf{x})$
- *example*: probability of cancer ($\mathbf{t}$), given an x-ray image ($\mathbf{x}$)

- **Decision**

- for given $\mathbf{x}$, determine optimal decision based on predicted $\mathbf{t}$
- *example*: give cancer treatment, given an x-ray image
- the decision step is easy, if we have solved the inference step

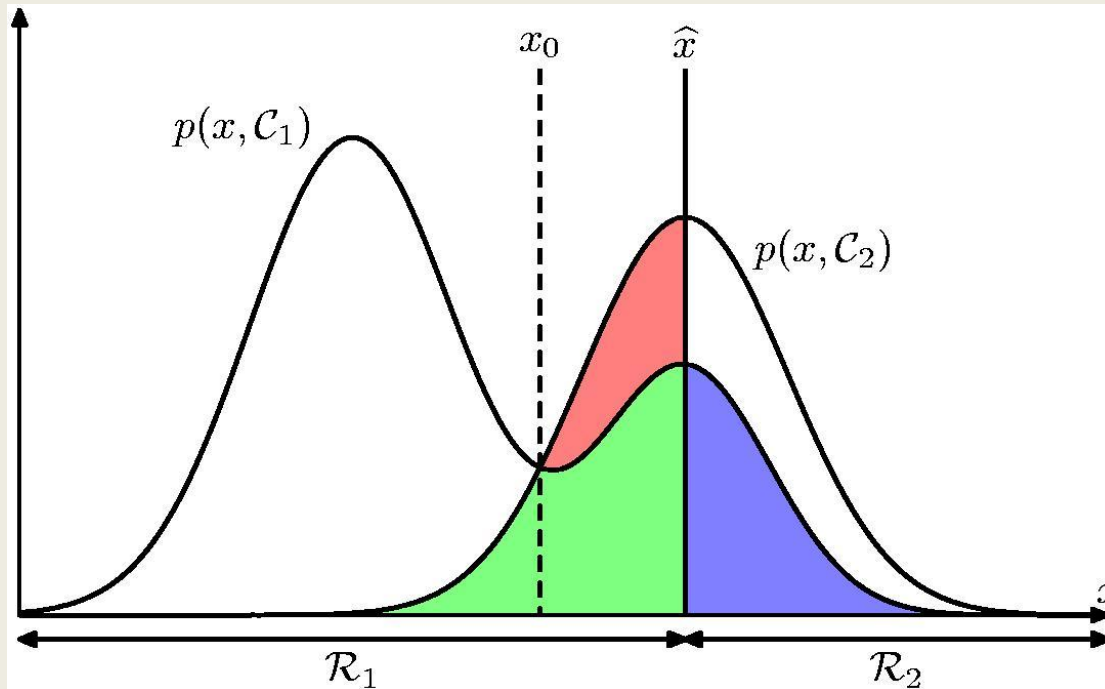
- **Example: Classification Rule**

- decision region R_k over $\mathbf{x}$
- corresponding to class C_k (decision)

$$p(C_k | \mathbf{x}) = \frac{p(\mathbf{x} | C_k) p(C_k)}{p(\mathbf{x})}$$



Minimum Misclassification Rate

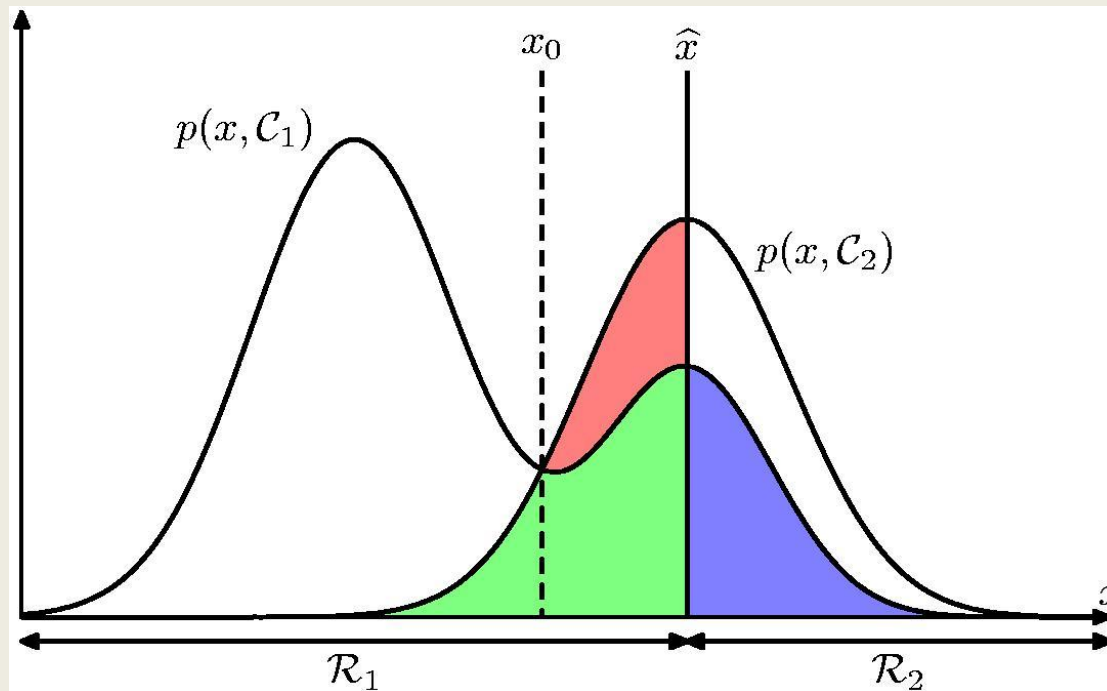


Choose the class
with the
highest
probability

$$\begin{aligned} p(\text{mistake}) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x}. \end{aligned}$$



Maximum Correct Classification Rate



$$\mathcal{C}^* = \arg \max_k p(\mathbf{x}, \mathcal{C}_k)$$

$$p(\mathbf{x}, \mathcal{C}_k) = p(\mathcal{C}_k | \mathbf{x}) p(\mathbf{x})$$

$$\mathcal{C}^* = \arg \max_k p(\mathcal{C}_k | \mathbf{x})$$

Choose the class
with the highest
posterior probability

$$p(\text{correct}) = \sum_{k=1}^K p(\mathbf{x} \in \mathcal{R}_k, \mathcal{C}_k) = \sum_{k=1}^K \int_{\mathcal{R}_k} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$



Minimum Expected Loss

Decision

cancer normal

Truth	cancer	0	1000
	normal	1	0

- **Loss Function** L_{kj}
 - associates decisions with costs
 - depends on decision and truth

- **Minimum Expected Loss**

- expected cost due to mistakes

$$\mathbb{E}[L] = \sum_k \sum_j \int_{\mathcal{R}_j} L_{kj} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

- choose regions that minimize the expected loss

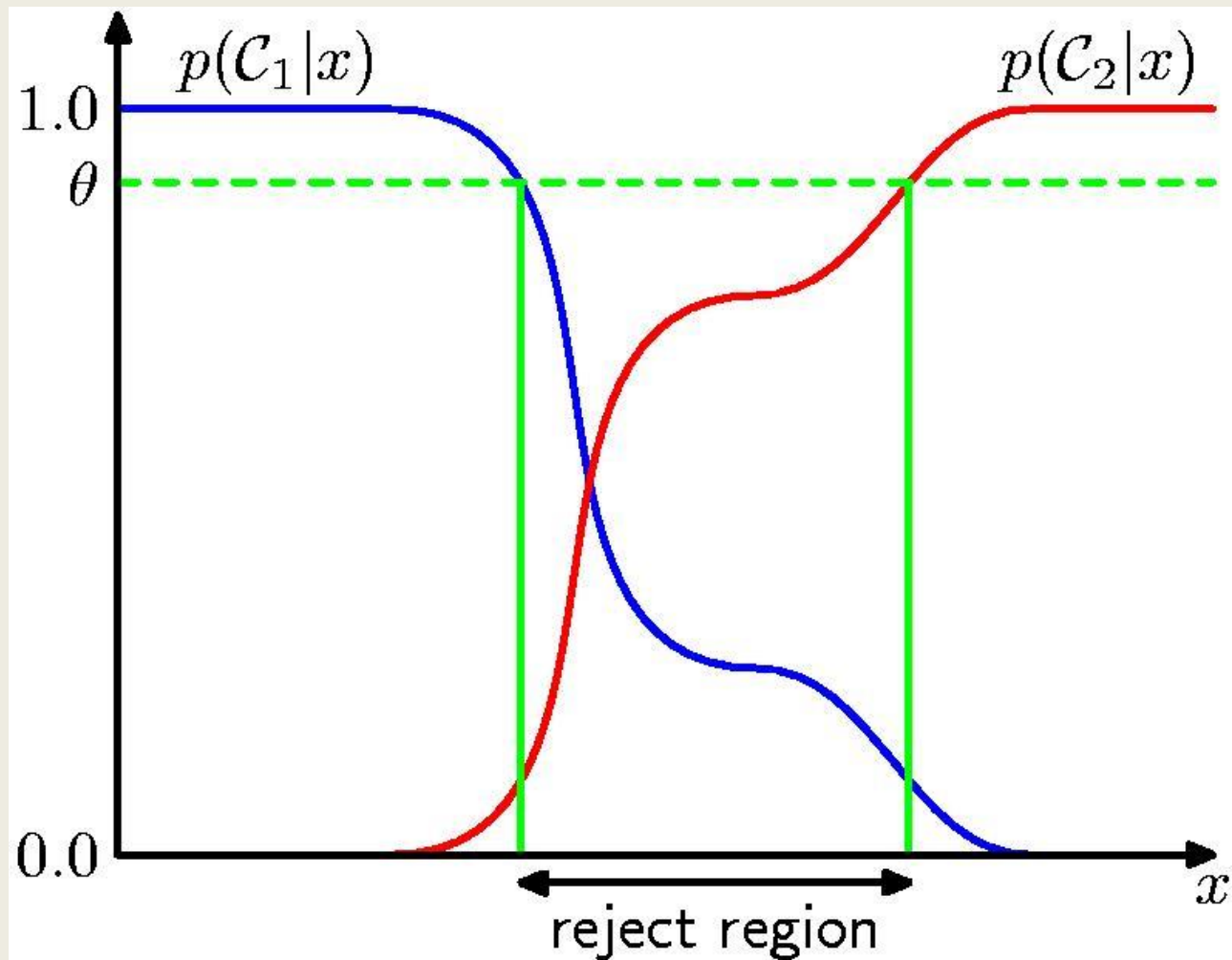
- **Decision**

- choose class that minimizes loss

$$C^* = \arg \min_j \sum_k L_{kj} p(\mathcal{C}_k | \mathbf{x})$$



Reject Option



Approaches to Classification Decision Problem

- **Approach I: generative models**
 - determine the class-conditional densities $p(\mathbf{x}|C_k)$ (and priors)
 - infer the posterior class probabilities $p(C_k|\mathbf{x})$ (through Bayes)
 - use decision theory to make decision
- **Approach II: discriminative models**
 - determine the posterior class probabilities $p(C_k|\mathbf{x})$
 - use decision theory to make decision
- **Approach III: discriminant functions**
 - determine a function that maps inputs to classes directly
 - use the discriminant function to make decision

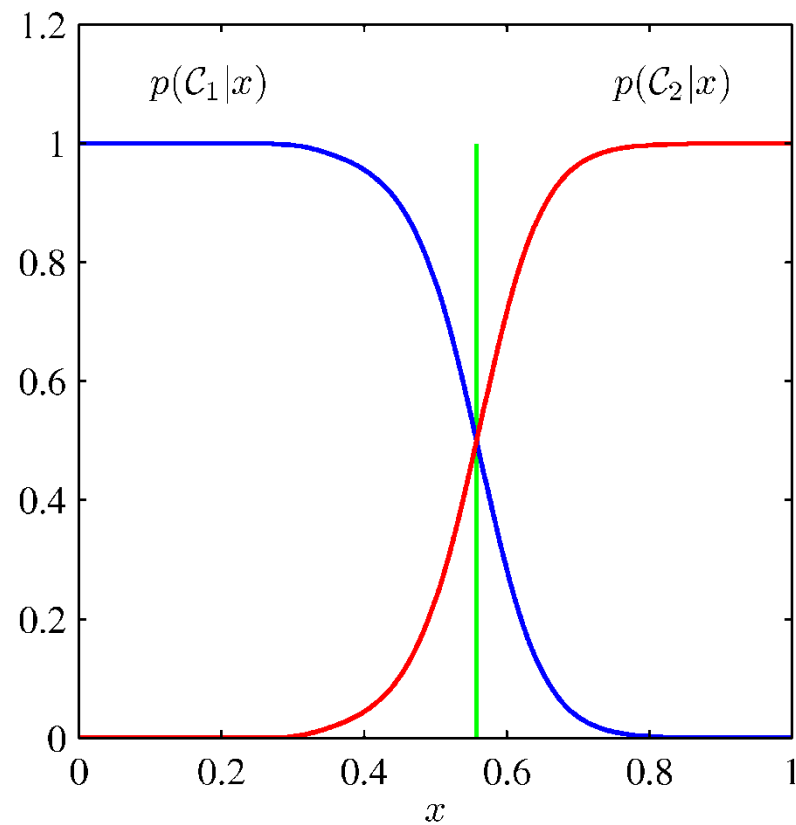
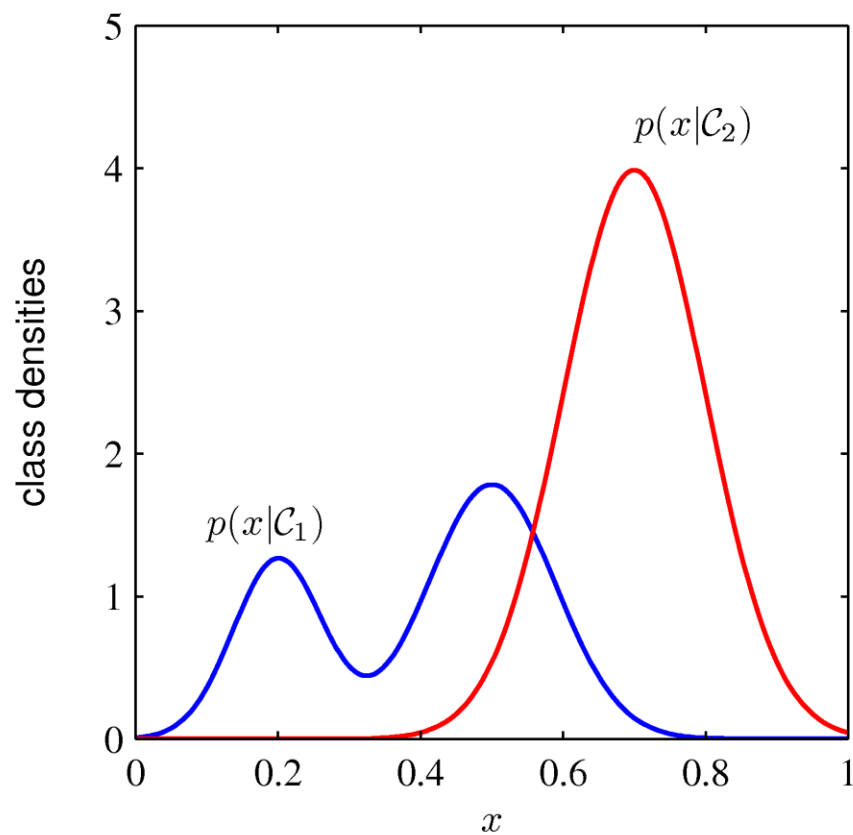


Pros and Cons

- **Approach I: generative models**
 - pros: can be used to generate synthetic data
 - cons: solves a much bigger problem, quite demanding
 - pros: can be used for outlier/novelty detection
- **Approach II: discriminative models**
 - pros: makes no waste of resources, avoids complex models
 - cons: solves a slightly bigger problem
- **Approach III: discriminant functions**
 - pros: a single (learning) problem
 - pros (cons?): may not use probabilities at all!
 - cons: black-box approach, provides no intuition



Comparison: $p(x|C_k)$ vs. $p(C_k|x)$ vs. x^*



Separating Inference and Decision

- **Minimizing risk**
 - loss matrix may change over time
 - changes can be easily introduced in the minimum risk criterion
- **Reject option**
 - posterior probabilities allow for a rejection criterion
- **Unbalanced class priors**
 - posterior probabilities can be transformed to other populations
- **Combining models**
 - split the problem, solve separately, model for each sub-problem
 - combine sub-models using probability theory to solve problem
 - exploit conditional independence



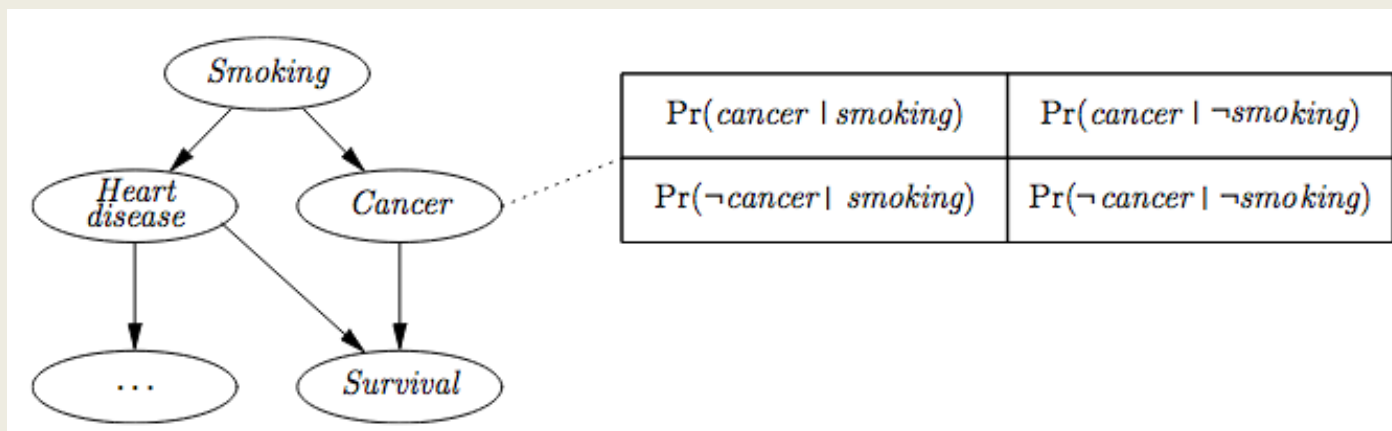
Bayesian Networks

A celebrated decision-making under uncertainty paradigm and tool



Bayesian Networks (BNs) or Probabilistic Graphical Models (PGMs)

- Events can be characterized by/linked to random variables.
How to represent/easily calculate their probability?
- There can be (in-)dependencies among the variables
- Graphical representation of the (in-)dependencies (via **DAGs**):

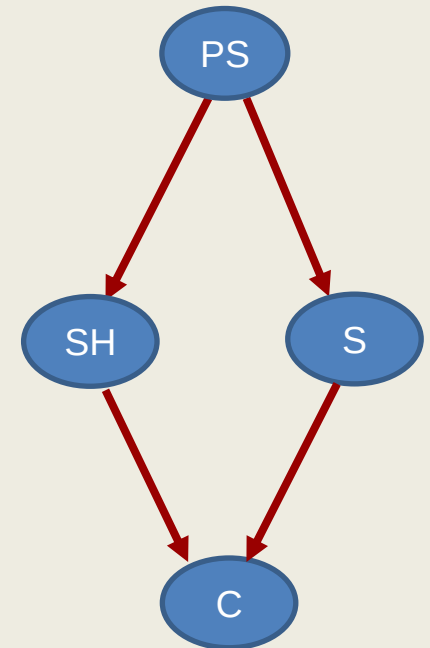


- *This can help provide structure to a problem!*
- *Makes it easier to reason about*
- *Makes it easier to elicit information from experts*
- *Could require far fewer numbers to represent the probability*



Example: reasoning about smoking causing cancer - using Bayesian networks for inference

- C: “has cancer”
 - SH: “exposed to second-hand smoking”
 - PS: “at least one parent smokes”
 - S: “smokes”
- ☐ a qualitative representation of independencies
- Whether or not the patient has cancer is directly influenced by S and SH
 - S and SH are influenced by PS
 - PS influences C also... However,
 - ***once values of SH and S are known, finding out about PS provides no additional information***
- We say that “C is independent of PS, given SH and S”



Why do we care about independence?

We want to represent probability distributions compactly

- State spaces characterized by $X_1, \dots, X_n$ random variables
- States have form $(x_1, \dots, x_n) : X_1 = x_1, \dots, X_n = x_n$
- Example: suppose $X_1, \dots, X_5$ independent *binary* variables
 - We can completely characterize a distribution by 5 numbers!
 - $\Pr(X_i = 0)$ for $i = 1, \dots, 5$
 - If $\Pr(X_i = 0) = a_i$, then $\Pr(X_i = 1) = 1 - a_i$
 - Independence: $\Pr(1, 0, 0, 1, 0) = (1 - a_1) * a_2 * a_3 * (1 - a_4) * a_5$
 - Can compute probability of a set of states by adding things

Thus, we can compactly describe the distribution using only n numbers (the $\Pr(X_i = 0)$ ones) instead of 2^n



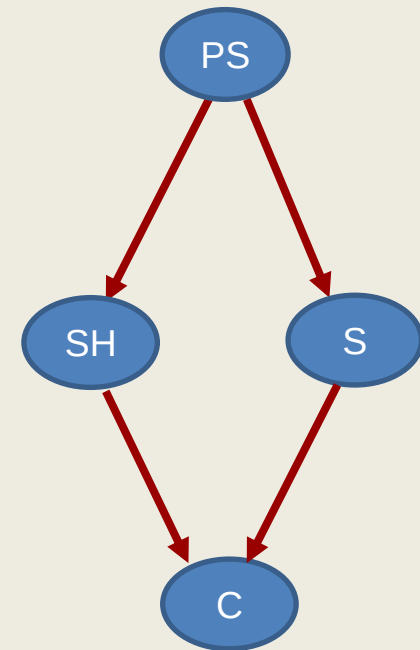
Quantitative BNs: Qualitative BNs + conditional probability tables (CPTs)

S	SH	C=1
1	1	0.6
1	0	0.4
0	1	0.1
0	0	0.01

PS	S=1
1	0.4
0	0.2

PS	SH=1
1	0.8
0	0.3

PS = 1
0.3



- The tables include only values for $\Pr(C=1 \mid S, SH)$, $\Pr(S=1 \mid PS)$, $\Pr(SH=1 \mid PS)$, $\Pr(S=1)$, but:
 - $\Pr(C=0 \mid SH) = 1 - \Pr(C=1 \mid SH)$
 - Can similarly compute other values of interest

via the chain rule and exploiting independencies from non-descendant nodes given parent nodes...

$$\Pr(PS = 0, S=0, SH=1, C=1) =$$

$$\Pr(C=1 \mid S=0, SH=1) * \Pr(S=0 \mid PS=0) * \Pr(SH=1 \mid PS=0) * \Pr(PS=0) = 0.1 * 0.8 * 0.3 * 0.7 = 0.0168$$



Recap: What do BNs buy us?

- **Conciseness of representation**
 - If a node has $< k$ parents, then we need $< 2^k n$ numbers to represent the distribution
 - if k is not too large, then $2^k n \ll 2^n$
- **Information tends to be easier to elicit**
 - experts are better in giving information about dependencies, rather than about numbers
- **Information easier to understand**
- **Many computational tools**
 - for computing probabilities given information
 - for learning Bayesian networks
 - many commercial packages exist
- **Numerous applications**
 - notably for medical decision making and diagnosis



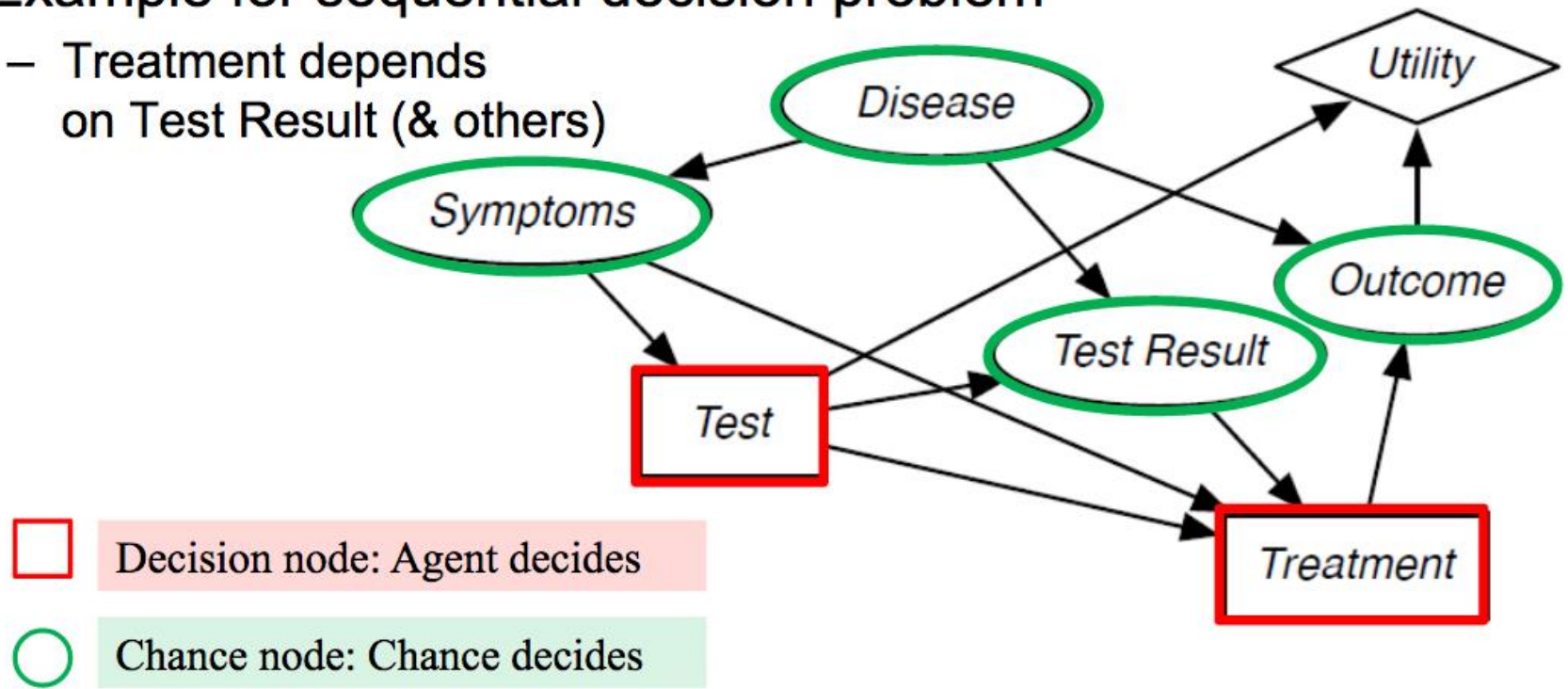
Making Sequentially Optimal Decisions



Decision Networks

Example for sequential decision problem

- Treatment depends on Test Result (& others)



- Each decision has an *information set* of parent variables (pa), whose value will be known at the time the decision is to be made
- $pa(\text{Test}) = \{\text{Symptoms}\}$, $pa(\text{Treatment}) = \{\text{Test}, \text{Symptoms}, \text{Test Result}\}$

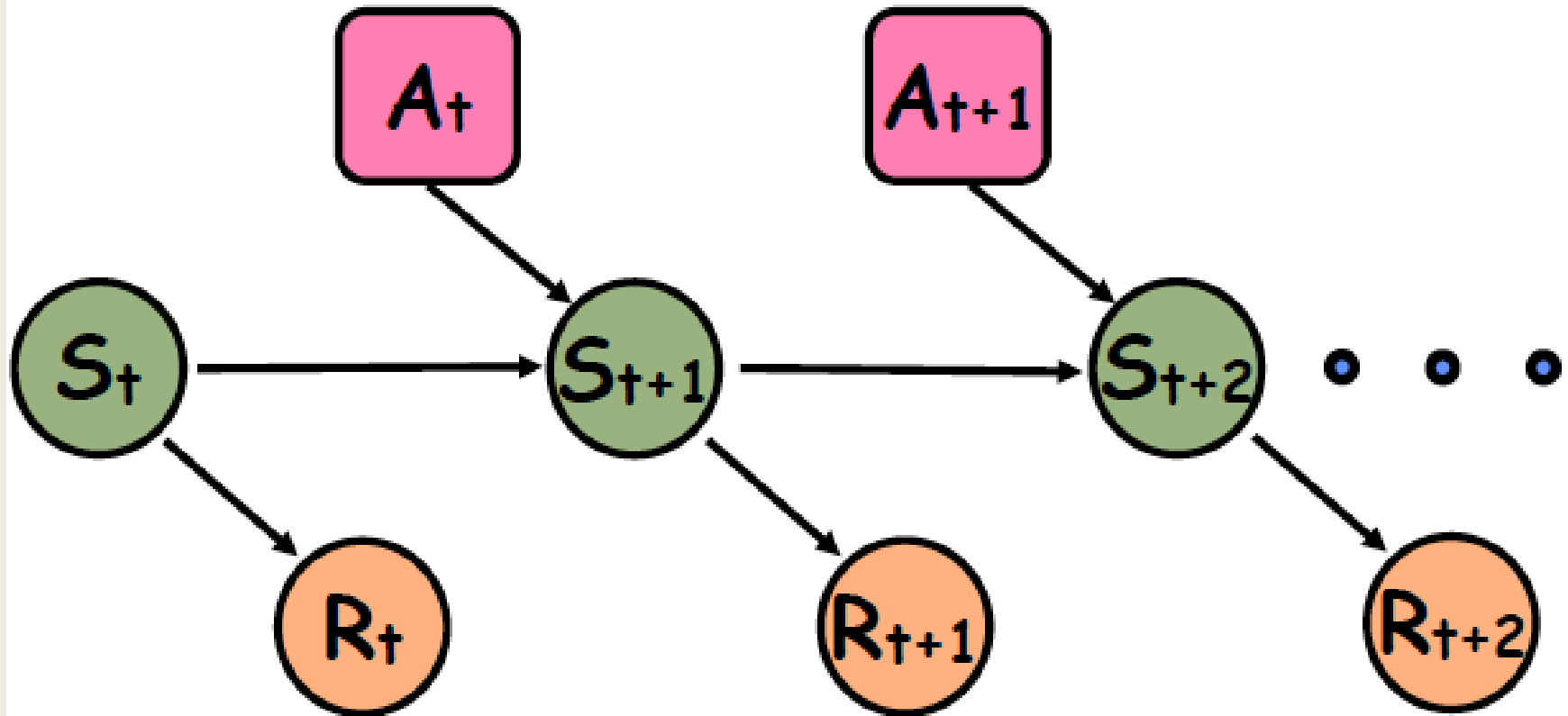


Markov Decision Processes (MDPs)

- MDPs are a family of nondeterministic search problems
- An MDP has four components: **S , A , R , Pr**
 - (finite) state set S ($|S| = n$)
 - (finite) action set A ($|A| = m$)
 - transition function $Pr(s' | s, a)$
 - each $Pr(s' | s, a)$ is a distribution over S
 - represented by set of $n \times n$ stochastic matrices
 - probability of transition to next state s' , when choosing action a in current state s
 - bounded, real-valued reward function $R(s)$
 - represented by an n -vector
 - can be generalized to include action costs: $R(s, a)$
 - can be stochastic (replaceable by expectation)



Graphical view of MDPs



What is Markov about MDPs

- Andrey Markov (1856-1922)
- “Markov” chains: given present state, future is independent from the past
- In MDPs: past states/actions are irrelevant, when taking decisions at present



Assumptions

- Markovian dynamics (history independence)
 - $\Pr(S^{t+1} \mid A^t, S^t, A^{t-1}, S^{t-1}, \dots, S^0) = \Pr(S^{t+1} \mid A^t, S^t)$
- Markovian reward process
 - $\Pr(R^{t+1} \mid A^t, S^t, A^{t-1}, S^{t-1}, \dots, S^0) = \Pr(R^{t+1} \mid A^t, S^t)$
- Stationary transition and reward dynamics
 - $\Pr(S^{t+1} \mid A^t, S^t) = \Pr(S^{t'+1} \mid A^{t'}, S^{t'})$ for all t, t'
 - $\Pr(R^{t+1} \mid A^t, S^t) = \Pr(R^{t'+1} \mid A^{t'}, S^{t'})$ for all t, t'
- Full observability
 - though we can't predict what state we will reach when we execute an action, once it is realized, we know what it is



Policies

- **Non-stationary policy**
 - $\pi: S \times T \rightarrow A$
 - $\pi(s,t)$ is action to do at state s **with t-steps-to-go**
- **Stationary policy**
 - $\pi: S \rightarrow A$
 - $\pi(s)$ is action to do at state s (regardless of time)
 - analogous to reactive or universal plan
- These assume or have these properties:
 - full observability
 - history-independence
 - deterministic action choice

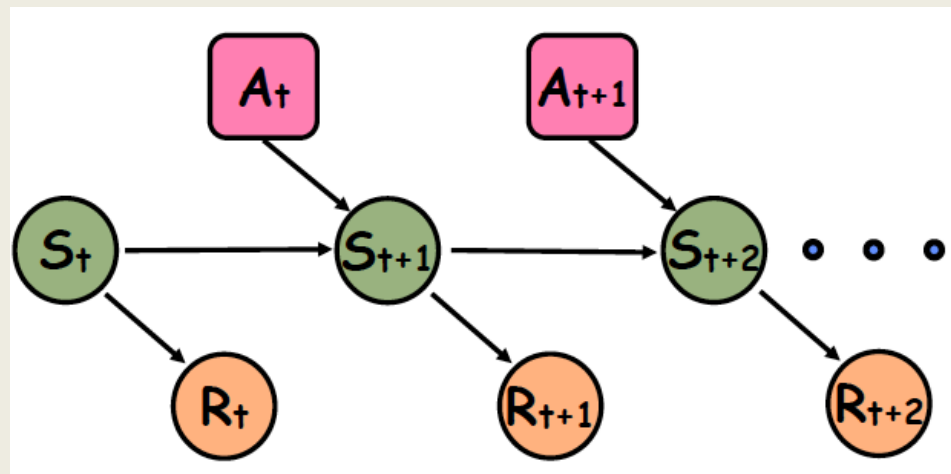


Value (utility) of a policy

- How good is a policy π ?
- Does it choose actions that bring a lot of reward in the long run?
- How do we measure “accumulated” reward?
- **Value function $V: S \rightarrow R$ (real-valued function)**
 - Associates each state with a value
- $V^\pi(s)$ denotes value of policy π at state s
 - **expected accumulated reward over horizon of interest**
 - Note that $V^\pi(s) \neq R(s)$: it measures (long-term) utility



MDPs are commonly "solved" via Dynamic Programming



What is Dynamic Programming?

Dynamic Programming (DP)

- methods for successively approximating optimal evaluation functions and decision rules

Used for:

- optimization problems with **objectives=sequences of states**
- DP solves these by solving recurrence relations
- **Basic DP step:**
backing-up state evaluations to solve the recurrence relations
- **Bootstrapping:** *updating estimates based on other (previous) estimates (of other quantities)*



(cons) requires repeated generation and expansion of all possible next states from each current state

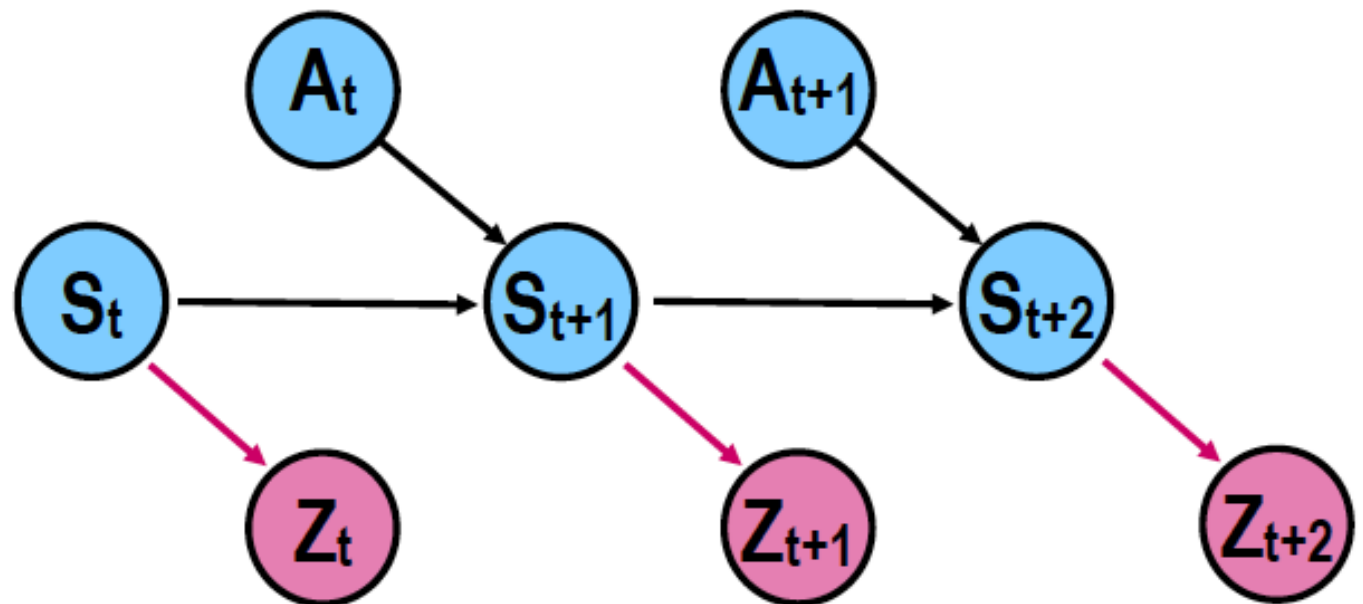
Enter Partial Observability

- Classical planning: observable states, deterministic actions
- MDPs: observable states, stochastic actions
- POMDPs: **partially observable states**, stochastic actions
 - *“The world is a huge POMDP. You just have to solve it.”*
[Craig Boutilier, circa 2002]

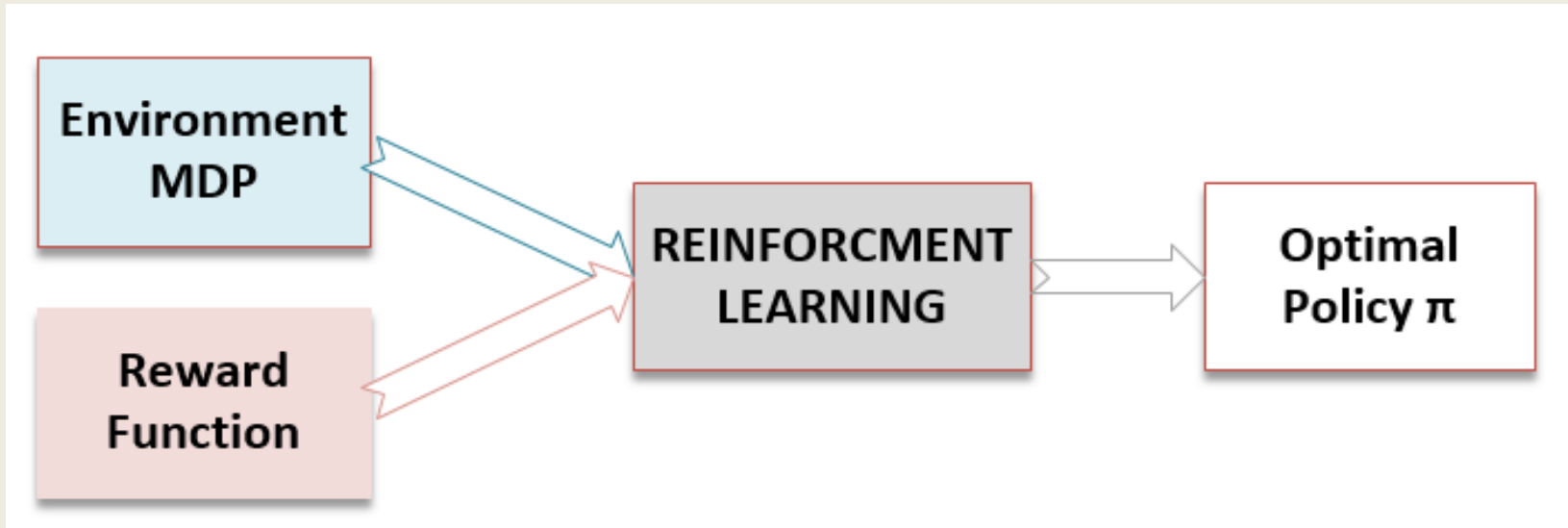


POMDP

- As in MDPs: $S, A, p_{ij}^a, r_i^a, r_i^T$
- Observation space: Z (or Z_a)
- Observation probabilities: p_{ijz}^a for $z \in Z_a$



Reinforcement Learning



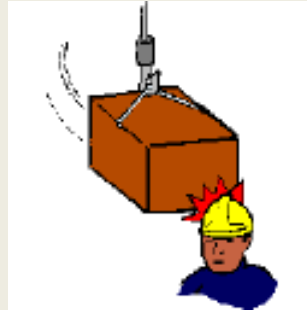
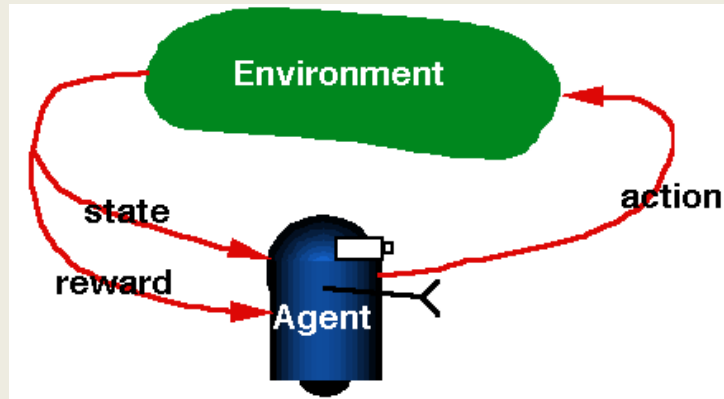
- Agent interacts with the environment (Markov Decision Process): chooses action, transitions to new state, and obtains a reward.
- The action choice in each situation (state) is based on its *value*: the (estimated or learned) long-term reward from similar situations.

Goal : maximize the expected reward obtained over long term

Goal of RL: policy π that specifies an action for each possible state



Reinforcement Learning



“ You do something, the world bangs you on the head; then you do something else, the world bangs you on the head again...”

Sam Roweis, 2002

✦ RL is learning by trial and error, i.e. learns how to map situations to actions, so as to maximize a sequence of rewards.

Approaches

- **model-based methods:** use observed rewards+transitions to build models of the world and use these to derive policies
- **model-free methods:** approximate the quality of choices directly from sampled reward-transition observation (e.g. “Q-learning”)



The Exploration vs. Exploitation Dilemma

- To explore or to exploit?
 - Stay / be content with what I know, or
 - ... attempt to test other states-action pairs?



Exploration vs. Exploitation++

- **RL**: the agent should explicitly **explore** the environment in order to acquire knowledge about it
- Act to **improve the estimate of the value function** (exploration) or to **get high (anticipated) payoffs** (exploitation) ? **Dilemma!**
- Rewards maximization ***requires*** exploration; but **too much exploration of irrelevant parts can waste time!**
 - Choice dependent on particular learning problem, as well as the particular learning technique employed
 - famous way to treat exploration vs exploitation problem:
 - GLIE: greedy in the limit exploration methods
 - better: the Bayesian approach



Tetris: an example of RL Results

Handcoded Player

Average Score ≈ 675

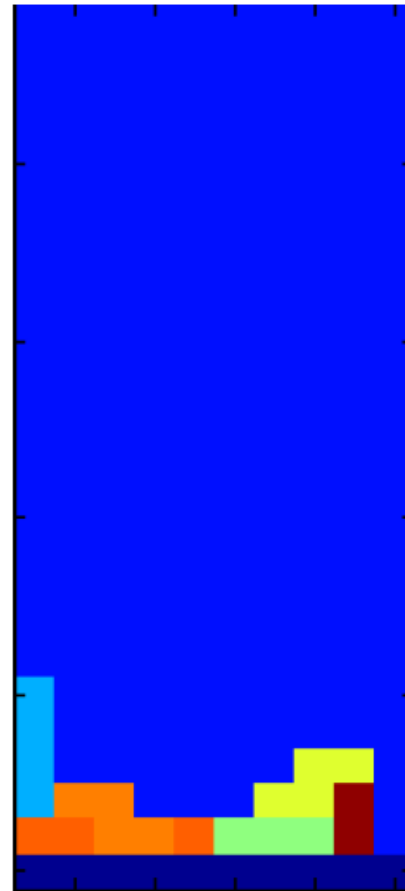
Score = 49
(after 150 steps)



Learned Player

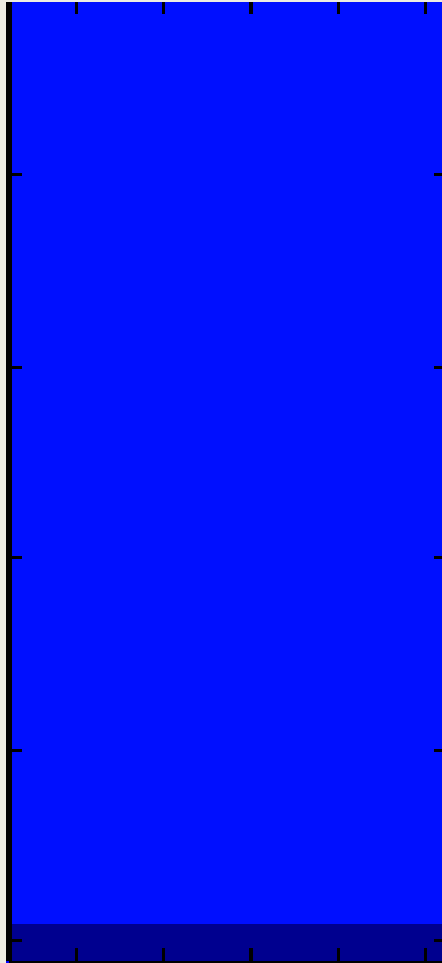
Average Score ≈ 3000

Score = 58
(after 150 steps)

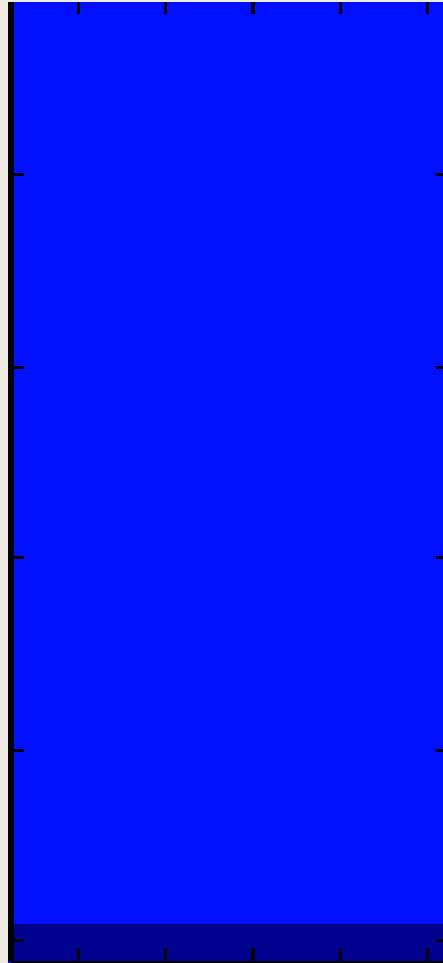


Tetris: Handcoded player and LSPI

Handcoded
Video

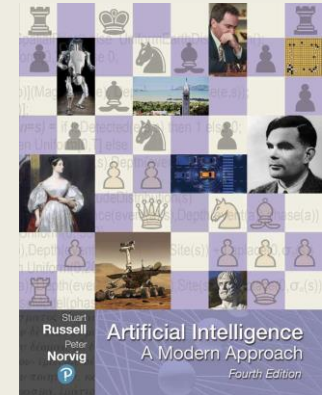


LSPI
Video



Suggested readings

- AI: A Modern Approach, 4th Edition
 - Chapters 12, 13, 16, and 17



Lecture 5

AI in Medicine and Healthcare

Michail Lagoudakis

**Intelligent Systems Laboratory
School of Electrical and Computer Engineering
Technical University of Crete
Chania, Crete, Greece**



Knowledge-Based Agents

- late 1970's
- search limitations?
- computational complexity barrier?
 - ⇒ capture and use explicit knowledge of a problem
- The birth of expert systems (e.g., **MYCIN**, **CYC**...)
- Need for **rich** and **precise** ways to represent knowledge
- ... and to make *deductions: from true premises, via the laws of logic, reach necessarily true conclusions*



hence, the birth of “logical AI”

Knowledge-Based Agents

- **Logical AI**
 - knowledge representation
 - inference procedures
 - mainly via logic
- **Drawbacks**
 - definite knowledge, no way to deal with uncertainty



Definite knowledge $\Rightarrow$ no retractions

Common-sense reasoning

You are told Tweetie is a bird. You conclude Tweetie can fly. You are then, later, told that Tweetie is a penguin. Thus, you retract your conclusion.

- logical reasoning $\neq$ common-sense reasoning
- logical reasoning is inappropriate or an overkill for many tasks
 - e.g., driving a car, recognizing faces, analyzing malware, ...



Knowledge-Based Agents

- **Logical AI**

- knowledge representation
- inference procedures
 - mainly via logic

- **Drawbacks**

- definite knowledge, no way to deal with uncertainty

- **Advantages**

- Generic representation $\Rightarrow$ flexibility, adaptability
- Combining/blending information $\Rightarrow$ discovery of new knowledge
- Knowledge + Percepts $\Rightarrow$ discovery/realization of world state



Knowledge Base

- **A set of sentences**
 - Initialization: background knowledge
- **Sentence**
 - A claim/assertion for the agent's world
 - Representation language: some formal logic
- **Functions**
 - ...TELL and ASK...



Simple Knowledge-Based Agent (high-level view)

```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
         t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

- **Declarative approach**

- Provide the agent with useful/necessary knowledge about world
 - Goals, state changes, irrevocable assertions, ...
- The agent decides, given the declared sentences
 - This is in contrast to the “procedural” approach



Expert Systems: Definition

- Software programs that use AI techniques to simulate the judgement and decisions of human experts
- Historically / usually: highly specialized
- Historically / originally: stand-alone
- Historically/ originally: emulate the behavior/decision making processes of human experts
- Usually / often that can act parallel to human experts
- Developed using a multitude of AI technologies



Expert Systems: Technologies

- Rule -based
- Symbolic AI / Logic -based
- Bayesian networks -based
- Decision theoretic (e.g. MDP-based)
- Machine learning / neural networks -based
 - exploit big data / data analytics
 - can lead to new knowledge
 - re: diseases (e.g. via pattern recognition / detection in data)
 - however, can suffer from overfitting / overgeneralization problems
- **Issues**
 - data acquisition
 - expert opinions' acquisition
 - privacy-related issues
 - explainability



“Expert system”: kind of old-fashioned

- historical term, not commonly-used anymore
- refers mainly to rule-based / logic-based systems that hit the “complexity barrier” one way or another, e.g.
 - too few experts / experts’ opinions / rules
 - inefficient inference algorithms
 - combinations thereof
- better/modern terms
 - intelligent systems
 - AI-based systems



AI in Medicine: the beginnings (1959 - c. 1990)

“Medical artificial intelligence is primarily concerned with the construction of AI programs that perform diagnosis and make therapy recommendations. Unlike medical applications based on other programming methods, such as purely statistical and probabilistic methods, medical AI programs are based on symbolic models of disease entities and their relationship to patient factors and clinical manifestations.”

Clancey and Shortliffe, 1984



AI in Medicine:

a shift from diagnosis to broader decision support tasks and more (1990s - present)

Intelligent systems today are thus found supporting:

- **medical diagnosis**; very useful in/for:
 - complex/rare cases, when data complex or contradictory, or when physicians inexperienced
 - early diagnoses
- **medication prescribing**
 - in clinical laboratories and educational settings
 - can check for drug-drug interactions, dosage errors, and for other prescribing contraindications, such as allergies
- **clinical surveillance**
 - therapy critiquing
 - check for inconsistencies, errors, omissions in treatment plans
 - therapy planning
 - ...



AI in Medicine:

a shift from diagnosis to broader decision support tasks and more (1990s - present)

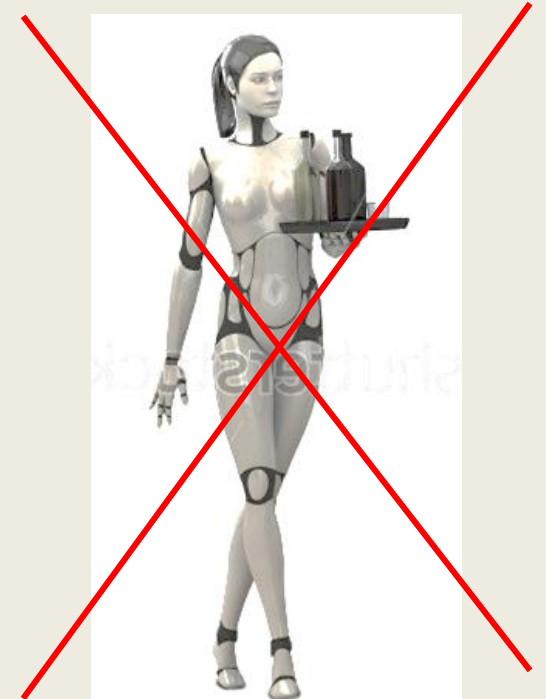
...

- **alerts and decision support in the intensive care**
- **intelligent information retrieval**
 - formulating appropriately specific and accurate clinical questions
 - information filtering
 - important use in medical educational settings
- **image recognition and interpretation**
- **surgery tasks** (increasingly)
- **discovery of new phenomena / creation of clinical knowledge**
 - analysis of “big” data
 - discovery of unknown patterns and associations
- **other peripheral but important medical/healthcare tasks**
 - embedded use in electronic patient record
 - to raise alerts and modify planned treatments “online”
 - determining level of medical record access privileges



"Stand alone" AI systems in Medicine?

- No, we have simply moved (as elsewhere) from the "strong AI" to "weak AI" paradigm
- e.g., decision support systems deeply integrated into an electronic patient record system
- integration reduces the barriers to using such a system, by crafting them more closely into clinical working processes, rather than expecting workers to create new processes to use them.



Examples of pre-2000s AI Systems in Medicine and Healthcare

ACUTE CARE SYSTEMS

POEMS (Sawar et al., 1992)	Post-operative care decision support
VIE-PNN (Miksch et al., 1993)	Parenteral nutrition planning for neonatal ICU
NéoGanesh (Dojat et al., 1996)	ICU ventilator management
SETH (Darmoni, 1993)	Clinical toxicology advisor



Examples of pre-2000s AI Systems in Medicine and Healthcare

LABORATORY SYSTEMS

GERMWATCHER (Kahn et al., 1993)	Analysis of nosocomial infections
HEPAXPERT I, II (Adlassnig et al., 1991)	Interprets tests for hepatitis A and B
Acid-base expert system (Pince, et al., 1990)	Interpretation of acid-base disorders
MICROBIOLOGY/PHARMACY (Morrell et al., 1993)	Monitors renal active antibiotic dosing
PEIRS (Edwards et al., 1993)	Chemical pathology expert system
PUFF (Snow et al., 1988)	Interprets pulmonary function tests
Pro.M.D.- CSF Diagnostics (Trendelenburg, 1994)	Interpretation of CSF findings



Examples of pre-2000s AI Systems in Medicine and Healthcare

EDUCATIONAL SYSTEMS

DXPLAIN (Barnett et al., 1987)	Internal medicine expert system
ILLIAD (Warner et al., 1988)	Internal medicine expert system
HELP (Kuperman et al., 1991)	Knowledge-based hospital information system



Examples of pre-2000s AI Systems in Medicine and Healthcare

QUALITY ASSURANCE AND ADMINISTRATION

COLORADO MEDICAID Utilization Review System	Quality review of drug prescribing practices
Managed Second Surgical Opinion System	Aetna Life and Casualty assessor system

MEDICAL IMAGING

PERFEX (Ezquerro et al., 1992)	Interprets cardiac SPECT data
SCINTIgrams (Lindahl et al., 1999)	Classification of scintigrams



AI Systems in Medicine and Healthcare: More Recent Developments 2005 - present

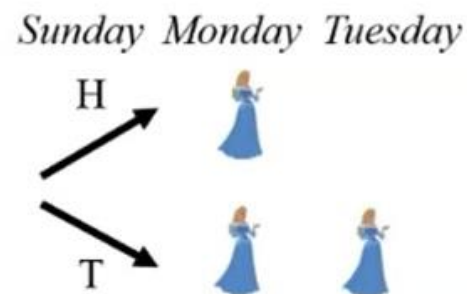
Systems embracing/tackling uncertainty,
using deep learning, and more



Deciding under Imposed Uncertainty: The Sleeping Beauty Problem

The Sleeping Beauty problem [Elga'00]

- There is a participant in a study (call her Sleeping Beauty)
- On Sunday, she is given drugs to fall asleep
- A coin is tossed (H or T)
- If H, she is awoken on Monday, then made to sleep again
- If T, she is awoken Monday, made to sleep again, then **again** awoken on Tuesday
- Due to drugs she **cannot remember what day it is or whether she has already been awoken once**, but she remembers all the rules
- Imagine **you** are SB and you've just been awoken. What is your (subjective) probability that the coin came up H?



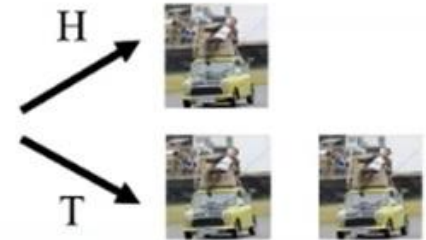
*don't do this at
home / without
IRB approval...*

The Sleeping Beauty Problem: Why do we care?

Modern version

- **Low-level autonomy** cars with AI that intervenes when driver makes major error
- Does not keep record of such event
- Two types of drivers: Good (1 major error), Bad (2 major errors)
- Upon intervening, what probability should the AI system assign to the driver being good?

Sunday Monday Tuesday



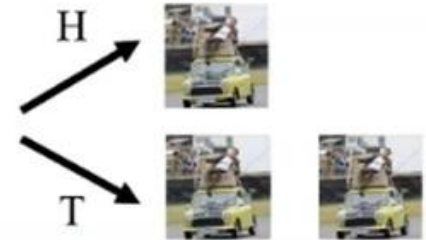
The Sleeping Beauty Problem: Why do we care?

Similar problems for the medical domain!!!

Modern version

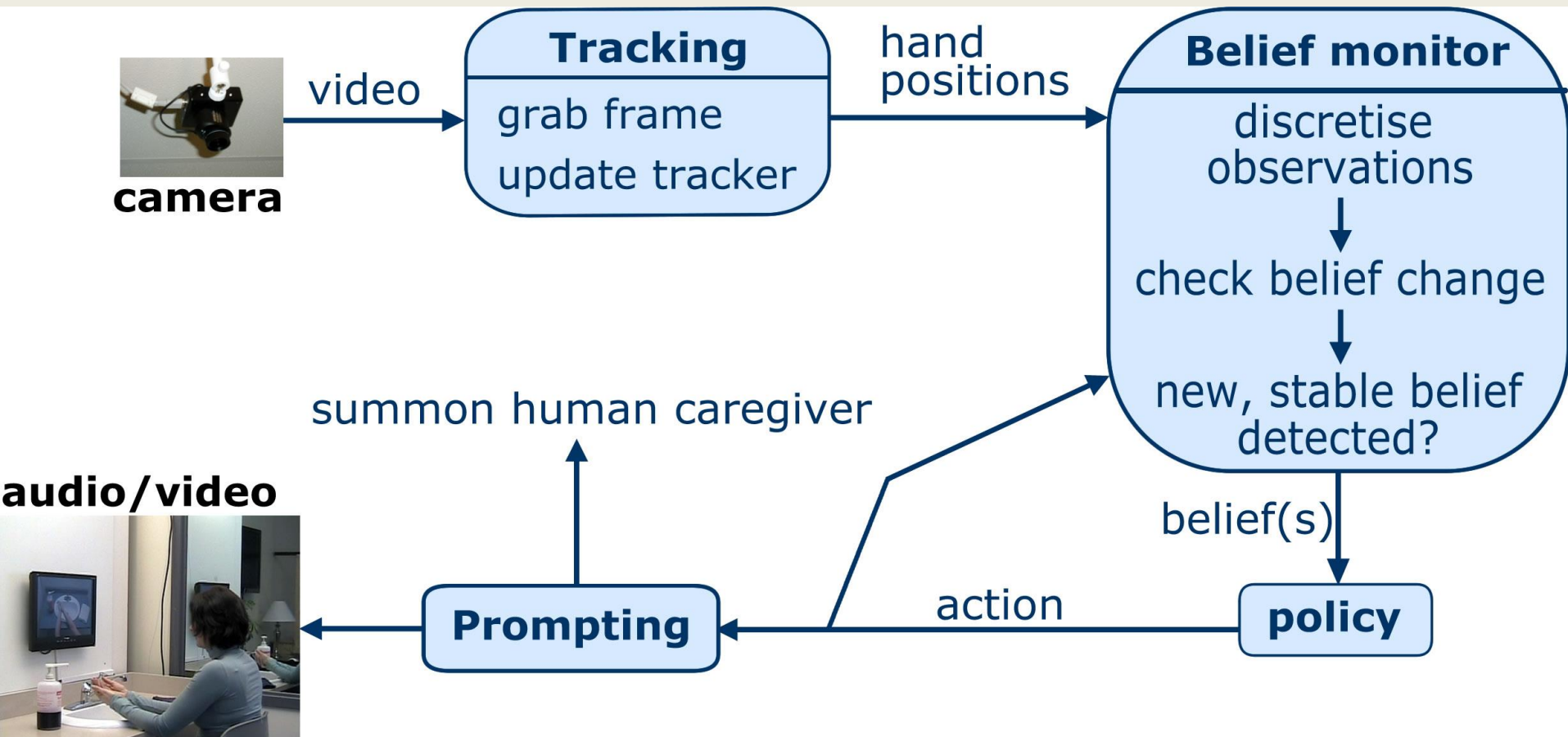
- **Low-level autonomy** cars with AI that intervenes when driver makes major error
- Does not keep record of such event
- Two types of drivers: Good (1 major error), Bad (2 major errors)
- Upon intervening, what probability should the AI system assign to the driver being good?

Sunday Monday Tuesday

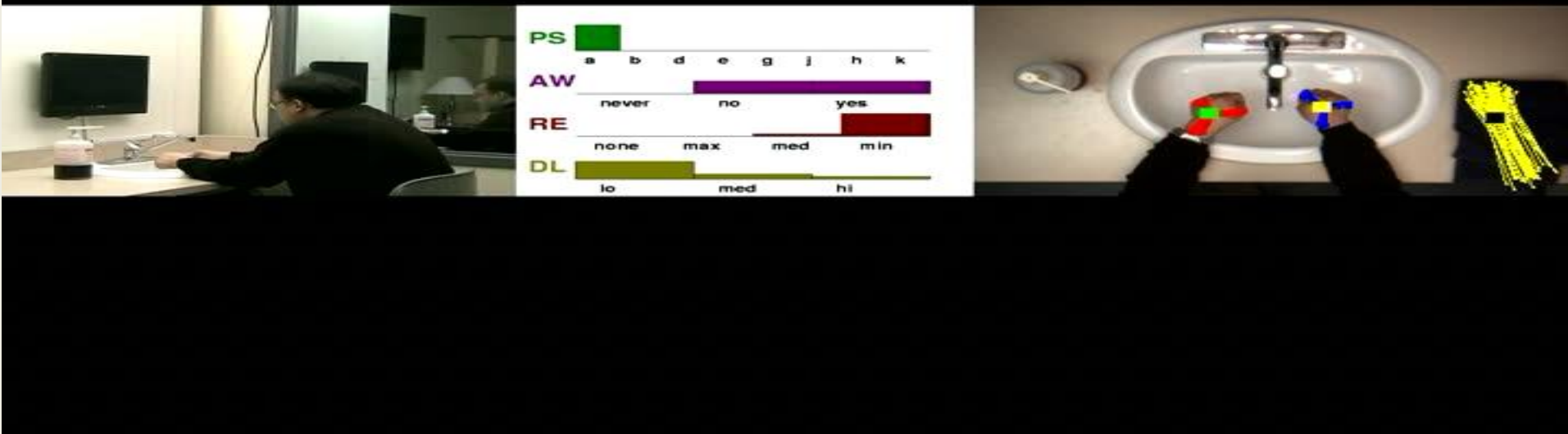


COACH: Assisted Handwashing using a POMDP - for people with cognitive disabilities (eg dementia)

<https://cs.uwaterloo.ca/~jhoey/research/coach/index.php>



COACH: Assisted Handwashing using a POMDP - for people with cognitive disabilities (eg dementia)



belief state in the planstep (PS), the awareness (AW),
the responsiveness (RE) and the overall dementia level (DL)

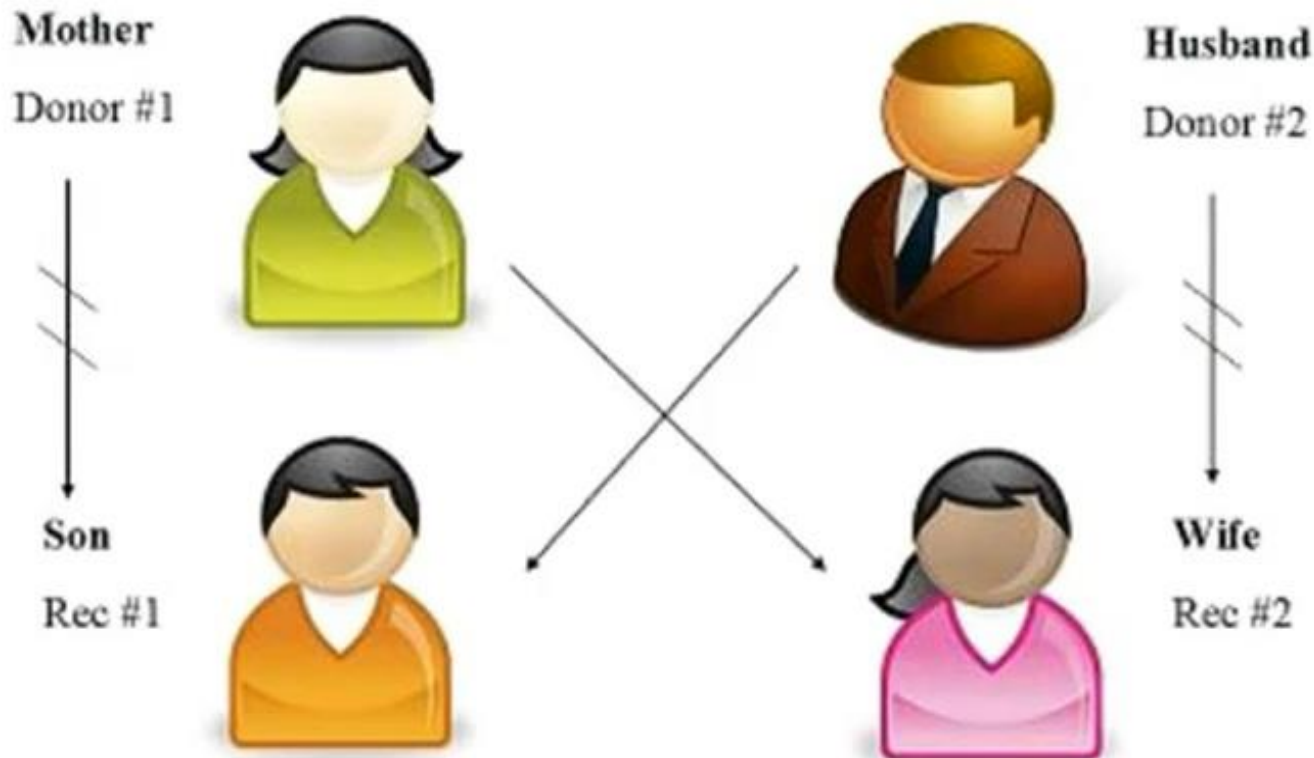
Actor trial - person needs assistance for most steps, and is only responsive to video prompts. The system learns this after an initial attempt with audio prompts, and then switches to using video only.



AI Facilitating Kidney Exchange

Kidney exchange [Roth, Sönmez, and Ünver 2004]

- Kidney exchanges allow patients with willing but incompatible live donors to swap donors



AI Facilitating Kidney Exchange

Kidney exchange [Roth, Sönmez, and Ünver 2004]

- Kidney exchanges allow patients with willing but incompatible live donors to swap donors

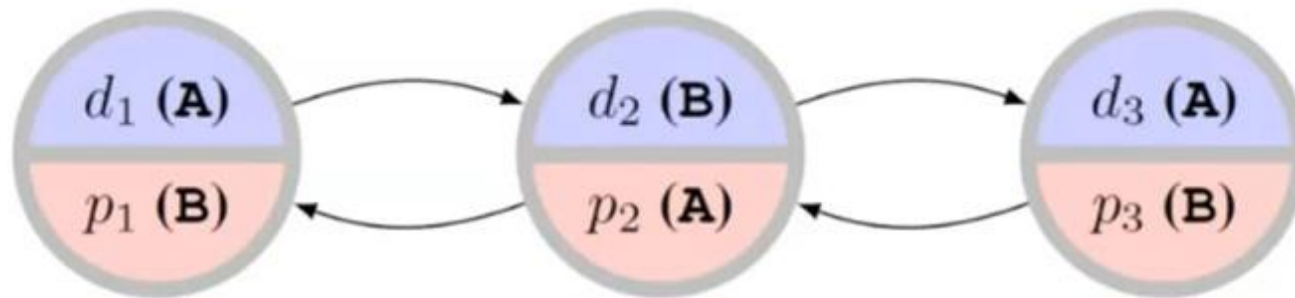
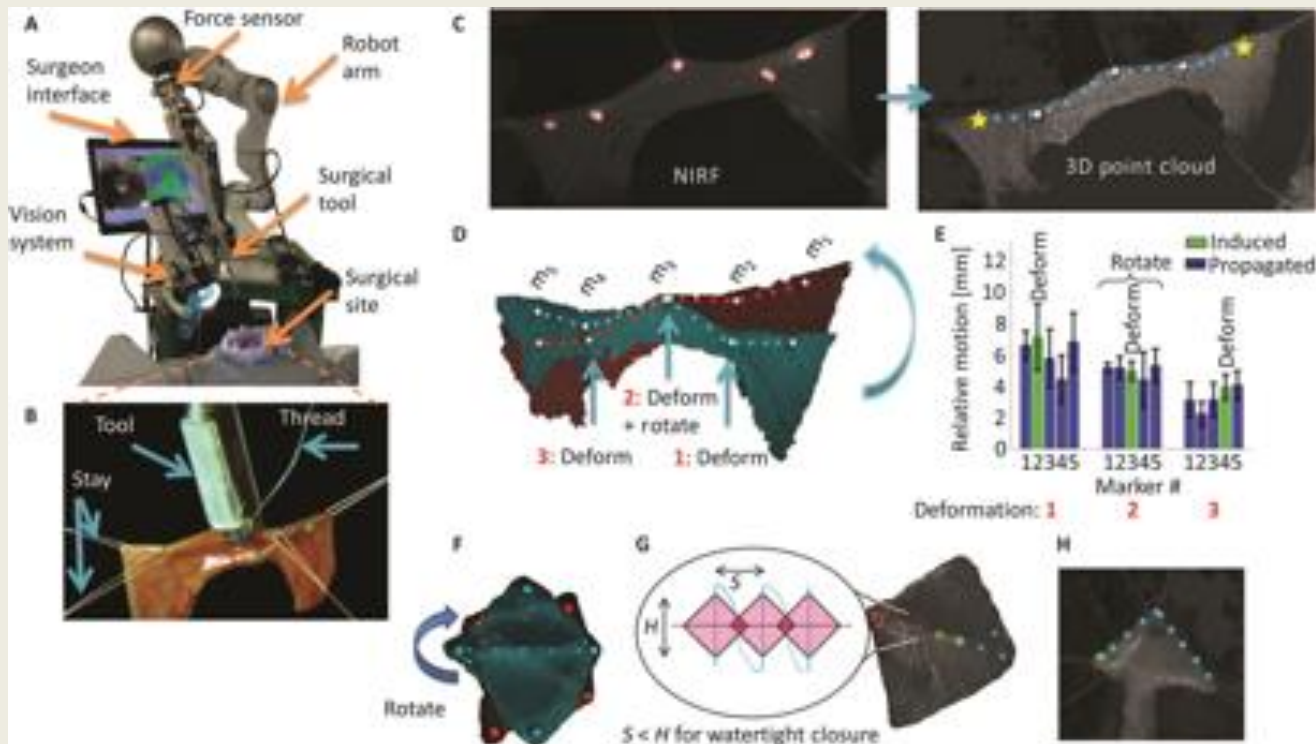


Figure 1: A compatibility graph with three patient-donor pairs and two possible 2-cycles. Donor and patient blood types are given in parentheses.

- Algorithms developed in the AI community are used to find optimal matchings (starting with Abraham, Blum, Sandholm [2007])

Robotic Surgery

- [Shademan et al 2016] **Supervised autonomous robotic soft tissue surgery**
- Possesses “an ability to perform intended tasks based on current state and sensing without human intervention”
- Findings: “the outcome of supervised autonomous procedures is superior to surgery performed *by expert surgeons and Robot-Assisted Surgery* techniques in ex vivo porcine tissues and in living pigs”



Machine Learning for Potentially New Clinical Knowledge

- **discover hidden relationships within raw data**
 - unsupervised learning methods can be used to discover patterns in genomic sequences $\Rightarrow$ infer connections with specific diseases
- **automatically construct pathophysiological models**
 - via inferring the functional relationships between various measurements
- **develop knowledge bases used by expert systems**
 - given a set of example clinical cases $\Rightarrow$ discover the clinical features that uniquely characterize the clinical conditions
- **discovery and development of new drugs**
 - given examples of one or more drugs that weakly exhibit a particular activity, and based upon a description of the chemical structure of those compounds, the learning system suggests which of the chemical attributes are necessary for that pharmacological activity
 - bootstrap with experimentation $\Rightarrow$ **enormous speed-up in development**
 - **also reduction of costs**
- **development of clinical guidelines**
 - identify features of treatment methods responsible for better results



AI Systems in Medicine: Measured Impact and Benefits

1. *Improved patient safety*
2. *Improved quality of care*
3. *Improved efficiency in health care delivery*



AI Systems in Medicine: Measured Impact and Benefits

1. *Improved patient safety*

- reduced medication errors and adverse drug events
- improved medication and test ordering
- enhancement of drug prescribing behaviors/processes



AI Systems in Medicine: Measured Impact and Benefits

2. *Improved quality of care*

- increasing clinicians' available time for direct patient care
- increased application of clinical pathways and guidelines
- facilitating the use of up-to-date clinical evidence
- improved clinical documentation and patient satisfaction



AI Systems in Medicine: Measured Impact and Benefits

3. *Improved efficiency in health care delivery*

- reducing costs through faster order processing
- reductions in test duplication
- improvements via changed patterns of drug prescribing
 - favoring cheaper, but equally effective generic brands



Non-Monotonic Logic and Argumentation for AI- Supported Patient Data Access and Sharing



Reminder: Strict Logic

- A statement either is or isn't a logical conclusion.
- If a statement is a logical conclusion (or solution to a problem), then it is still a logical conclusion, when we add any new knowledge!
- e.g. Once proven, mathematical theorems hold forever!
- Thus, we say that classical logic is **monotonic**.



Remember VIKI from "I, Robot"?



MY LOGIC IS UNDENIABLE

Non-Monotonic Logic

- However, when we reason with common sense, new information leads us to change our conclusion!
 - this is non-monotonic reasoning
- Common sense rules are not strict
- They are “For the most part” or “Usually” rules (**DEFAULT RULES**)
- A rule, **p :- q**, is interpreted as
“Usually, if we know that q holds then p holds”
- Example: **fly(X) :- bird(X)**, holds “for the most part”

(PROLOG notation “ :- ” equivalent to “inverted $\Rightarrow$ ”)



Defeasible Knowledge and Argumentation

Slides in this section are based on material by Dr Nikos Spanoudakis and other researchers working on the development of the Gorgias system



Defeasible Knowledge

- **Properties of the world**
 - “Usually, birds fly”
 - “Usually, penguins don’t fly”
 - “Usually, nestlings don’t fly”
 - “Usually, hurt birds don’t fly”
- **Results of actions**
 - “Usually, when we shoot a bird, it is hurt”
 - “Usually, when the gun is not loaded, the bird is not hurt”
 - “Usually, the gun is loaded”
 - “Usually, when the bird is far away, it is not hurt”



What color is my orange?

- I have it in my bag...



What color is my orange?

- I have it in my bag...



What color is my orange?

- I have it in my bag...
- It is not ripe...



What color is my orange?

- I have it in my bag...
- It is not ripe...



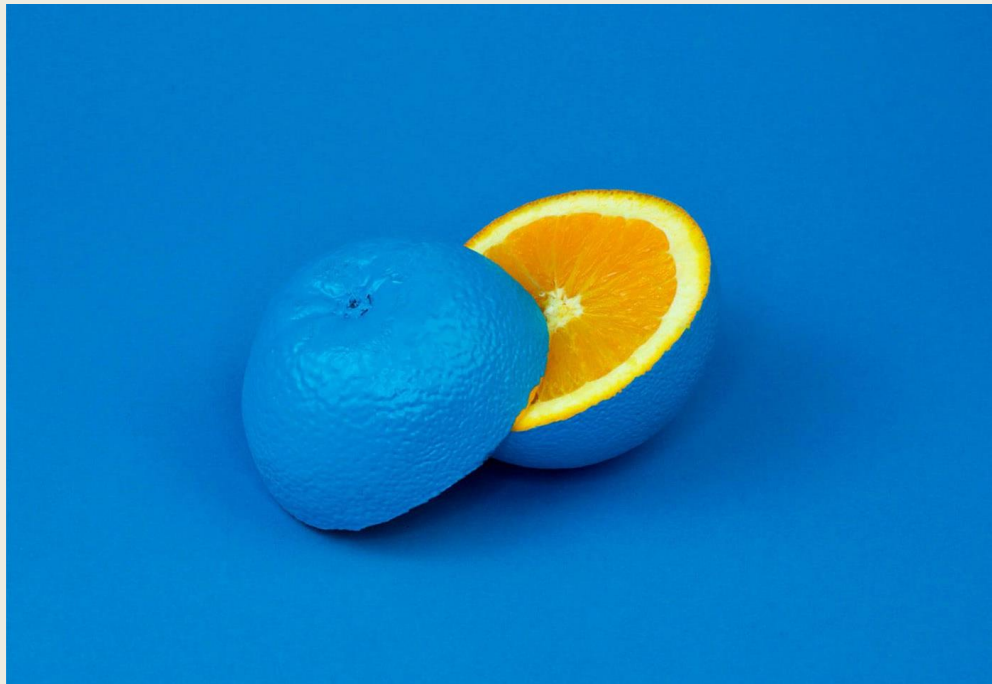
What color is my orange?

- I have it in my bag...
- It is not ripe...
- I spilled blue color in my bag...



What color is my orange?

- I have it in my bag...
- It is not ripe...
- I spilled blue color in my bag...



What is an argument?

- An argument is a link between
 - Some premises
 - A conclusion supported by them
- Premises → Conclusion



Acceptable Arguments

- Based on the informal meaning:
 - “A valid argument is one whose counter-arguments are not valid”
 - “A valid argument is one whose counter-arguments are, or rendered by it, not valid”
- What is a good/acceptable argument?
 - An argument that builds a coherent case for its position.
 - An argument that can defend itself against all its counter-arguments.
 - An argument that renders its counter-arguments incoherent / invalid
- We call this an **Admissible** argument.



Example of Argumentation

- “Sellers who deliver on time are trustworthy”
 - $a1 = \{\text{trusted}(\text{Seller}) :- \text{timely}(\text{Seller})\}$
- “Sellers who deliver wrong are not trustworthy”
 - $a2 = \{\neg \text{trusted}(\text{Seller}) :- \text{wrong_delivery}(\text{Seller})\}$
- Suppose we “observe”:
 - $\text{timely}(\text{bob})$: $a1$ supports $\text{trusted}(\text{bob})$.
 - $\text{wrong_delivery}(\text{bob})$: $a2$ supports $\neg \text{trusted}(\text{bob})$.
 - $a1$ attacks $a2$ and vice-versa.
- “Sellers who are trusted get large orders”
 - $a = \{\text{large_orders}(X) :- \text{trusted}(X)\}$
 - $A = \{a1, a\}$ supports $\text{large_orders}(\text{bob})$
 - $B = \{a2\}$ attacks A - **B undercuts A**



Argumentation: Foundations

- Logical Entailment via argument acceptability
 - Existence of an acceptable argument for conclusion ϕ .
 - **Credulous entailment**
 - Non-Existence of an acceptable argument for $\neg\phi$.
 - **Skeptical entailment**



Preference-Based Argumentation

- Logic Programming Rules and Priorities
- An extension of Logic Programming
- Arguments are sets of rules
- Attacks between arguments are defined via:
 - Conflicts between conclusions of arguments
 - Strength relation on the subsets of rules, used in each argument to derive the conflicting conclusion, based on the priority relation between the individual rules in the subsets.



An Example

(r1): $\text{fly}(x) \leftarrow \text{bird}(x)$

(r2): $\neg \text{fly}(x) \leftarrow \text{penguin}(x)$

(r3): $\text{penguin}(x) \leftarrow \text{walkslikepeng}(x)$

(r4): $\neg \text{penguin}(x) \leftarrow \neg \text{flatfeet}(x)$

(r5): $\text{bird}(x) \leftarrow \text{penguin}(x)$

(r6): $\text{bird}(\text{tweetie})$

(r7): $\text{walkslikepeng}(\text{tweetie})$

(r8): $\neg \text{flatfeet}(\text{tweetie})$

? fly(tweetie)

Argument for:

$A1 = \{r6, r1\}$

Against A1:

$A2 = \{r7, r3, r2\}$

Against A2:

$A3 = \{r8, r4\}$

Yes, fly(tweetie) can be supported by A1 U A3. (credulous)



With preferences (priorities)

(r1): $\text{fly}(x) \leftarrow \text{bird}(x)$

(r2): $\neg \text{fly}(x) \leftarrow \text{penguin}(x)$

(r3): $\text{penguin}(x) \leftarrow \text{walkslikepeng}(x)$

(r4): $\neg \text{penguin}(x) \leftarrow \neg \text{flatfeet}(x)$

(r5): $\text{bird}(x) \leftarrow \text{penguin}(x)$

(r6): $\text{bird}(\text{tweedy})$

(r7): $\text{walkslikepeng}(\text{tweetie})$

(r8): $\neg \text{flatfeet}(\text{tweetie})$

(r9): $r2 > r1$

(r10): $r4 > r3$

? $\text{fly}(\text{tweetie})$

Argument for:

$A1 = \{r6, r1\}$

Against A1:

$A2 = \{r7, r3, r2, r9\}$

Against A2:

$A3 = \{r8, r4, r10\}$

**Yes, $\neg \text{fly}(\text{tweetie})$ is not supported.
(skeptical)**



Argumentation Summary

- **An argument is:**
 - A set of sentences/rules, S , in some background logic $(L, \vdash)$ from which we can derive a conclusion (i.e. $S \vdash \phi$)
- **Attacking Relation:**
 - Specifies that one argument (i.e. a set S_1 of rules) attacks another S_2 , when they have some contrary conclusion and S_1 is “as strong” as S_2 .
- **An argument S is Admissible:**
 - S is conflict-free (i.e. it does not attack itself) and
 - S attacks (counter-attacks) all its attacks
- **Credulous or Skeptical Reasoning:**
 - A conclusion holds in one or all admissible/acceptable extensions.



The Gorgias-B System and its Application to the Medical Record and Data Protection Domain



Medical Data Access/Sharing

- **Problem**
 - Decide **Level of Access** according to
 - **user** and
 - **current circumstances**
- **Challenge**
 - **Formalize and automate the Legislation**
 - **Facilitate the management/compliance of policy**
 - **Cognitive Interaction**
 - **Explain and Persuade the user**



Medical Data Access: Legislation

- There are **6 Access Levels** (Read and Write)
 - Full Access
 - Partial Access
 - Read Only Access
 - Restricted Read Access
 - Suspended Access
 - No Access
- Decide **Level of Access** according to
 - **user** and
 - **current circumstances.**



Medical Data Access: Legislation

Law 138(I)/2001: Personal Data Protection

Law N. 1(I)/2005: Patient Rights

- Example

“Generally, no one has access to medical files.
But, the owner CAN have full access.”

- Formalization and Execution Code in Gorgias

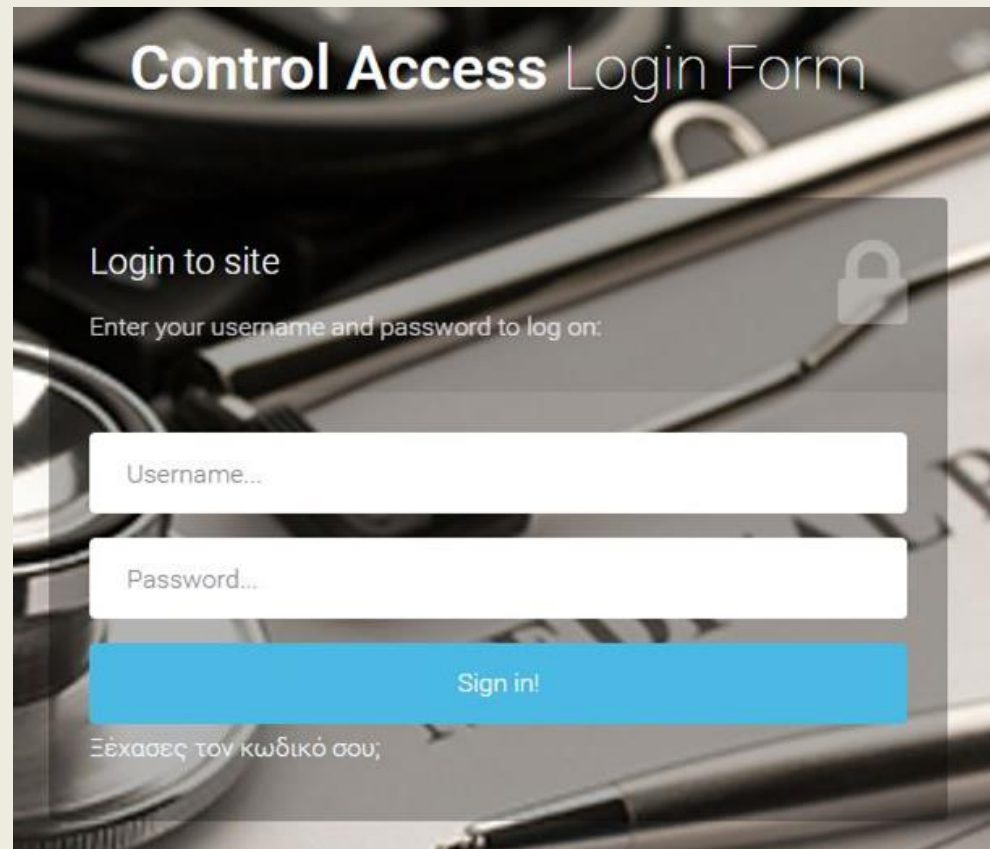
```
rule(d1(Agn), access(Agn, DataID, full_access), []).  
rule(d2(Agn), access(Agn, DataID, none_access), []).  
rule(hpr21(Agn), prefer(d2(Agn), d1(Agn)), []).  
rule(hpr12(Agn), prefer(d1(Agn), d2(Agn)), []) :- owner(Agn)  
rule(hpr_12_21(Agn), prefer(hpr12(Agn), hpr21(Agn)), []).
```



Medical Data Access: **MEDICA**

- **MEDICA**

<http://medica.cs.ucy.ac.cy>



The image shows a login form for the MEDICA system. The background is a blurred image of medical equipment. The form is titled "Control Access Login Form". It contains the following elements:

- Control Access Login Form** (Title)
- Login to site** (Text)
- Enter your username and password to log on:** (Text)
- Username...** (Input field)
- Password...** (Input field)
- Sign in!** (Button)
- Ξέχασες τον κωδικό σου;** (Text)



Development of MEDICA

- Under the **SoDA (Software Development for Argumentation) methodology** for applications of Argumentation.
- Over the argumentation framework of **Logic Programs with Priorities** and the **Gorgias** system.
- Using the **Gorgias-B tool**: a high-level declarative “programming” – problem solving – environment.



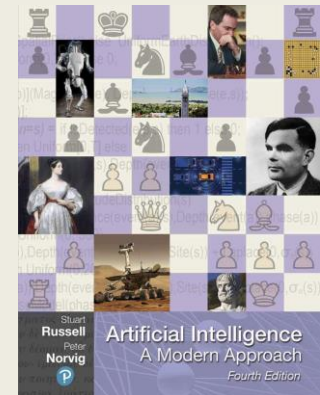
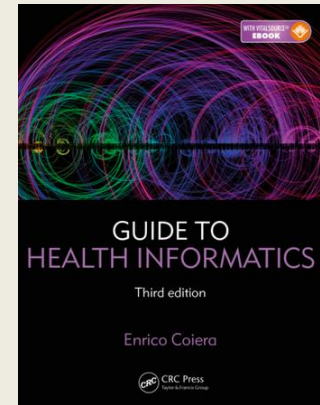
Decision Making via Argumentation

- **Options**
 - e.g. different **levels** of medical access
- **Preferences**
 - **Dynamic preferences** over **changing environment** of the application
 - **Multi-Level preferences** over different **CONTEXTS** in the application
- **General form** of Preferences
 - “**Generally**, in **SITUATION** prefer O_i , **but** in **particular** **CONTEXT** prefer O_j .”
 - “**Generally**, deny access **but** for the **{owner}** give full access.”
 - “**Generally**, allow full access to **{owner}**, **but** when **{critical tests}** suspend access.”



Suggested Readings

- The Guide to Health Informatics, 3rd Edition
 - Chapters 26 and 27
- The Guide to Health Informatics, 2nd Edition
 - Chapter 25
- AI: A Modern Approach, 4th Edition
 - Chapter 1



Lecture 6

Neural Networks and Deep Learning

Michail G. Lagoudakis

Intelligent Systems Laboratory
School of Electrical and Computer Engineering
Technical University of Crete
Chania, Crete, Greece



Machine Learning



Machine Learning

- learn from data to make predictions (and take decisions)
- **Supervised Learning**
 - set of training data with inputs and targets
 - classification, regression, ...
- **Unsupervised Learning**
 - set of training data with inputs, but without targets
 - clustering, density estimation, dimensionality reduction, ...
- **Reinforcement Learning**
 - set of training trials of interaction with feedback by a critic
 - value function, decision policy, exploration vs. exploitation, ...
- **Learning Theory**
 - theoretical investigations: what can be learned? how fast? ...



Machine Learning Terminology

- **Dataset**

- data for training, validation, testing

- **Model**

- structure/artifact created through a training process

- **Cost/Loss/Objective function**

- measures model performance

- **Optimization Procedure**

- minimizes error over (training) data



Machine Learning Terminology ++

- **Training error:** error rate (over the training set)
- **Testing error:** generalization error (over the testing set)
(the expected value of the error on new inputs)
- **Parameters/Estimators:** a vector of estimated parameter values
(e.g. weights of a neural network) giving the "best" prediction
- **Hyperparameters:** parameter settings to control the algorithm's
behavior, learned during **validation** (**k-fold cross-validation**)
- **Bias:** expectation over data included in the estimator
- **Maximum Likelihood:** common principle for parameter estimation

$$\theta_{ML} = \arg \max p_{model}(\mathbb{X}; \theta)$$

- find parameter values to make observations most probable given the model

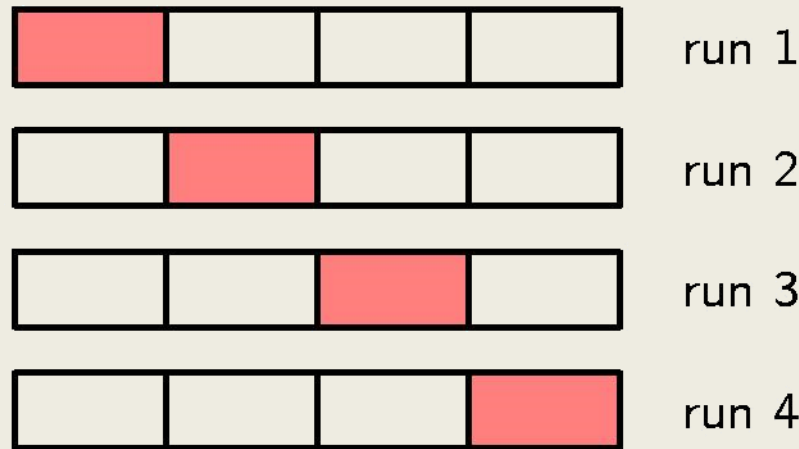


Model Selection

- **Models**
 - polynomials, mixture distributions, neural networks, ...
- **Complexity**
 - the number of (free) model parameters
- **Control**
 - selecting the order of a polynomial
 - selecting the number of distributions in the mixture
 - selecting the number of hidden units in a neural network
 - ...
- **Approach**
 - train several models on the same training set
 - look for best predictive performance on independent data



k -Fold Cross-Validation



example: $k=4$

- training on $k-1$ groups, testing on the remaining group
- averaging performance over all k runs
- all data are used for both training and testing
- for $k=N$ (number of data): leave-one-out technique
- drawback: computationally expensive, large number of runs

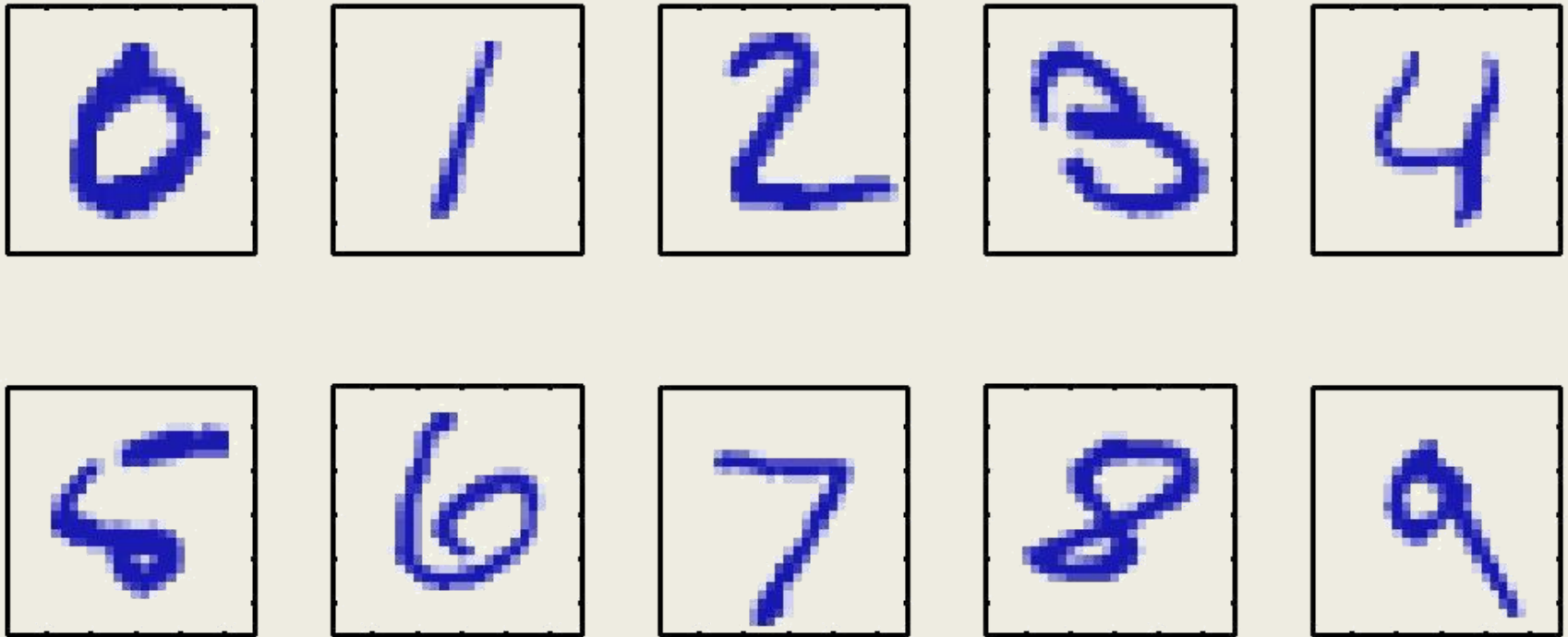


Machine Learning Examples



Hand-Written Digit Recognition

input: digital image (28x28 pixels)



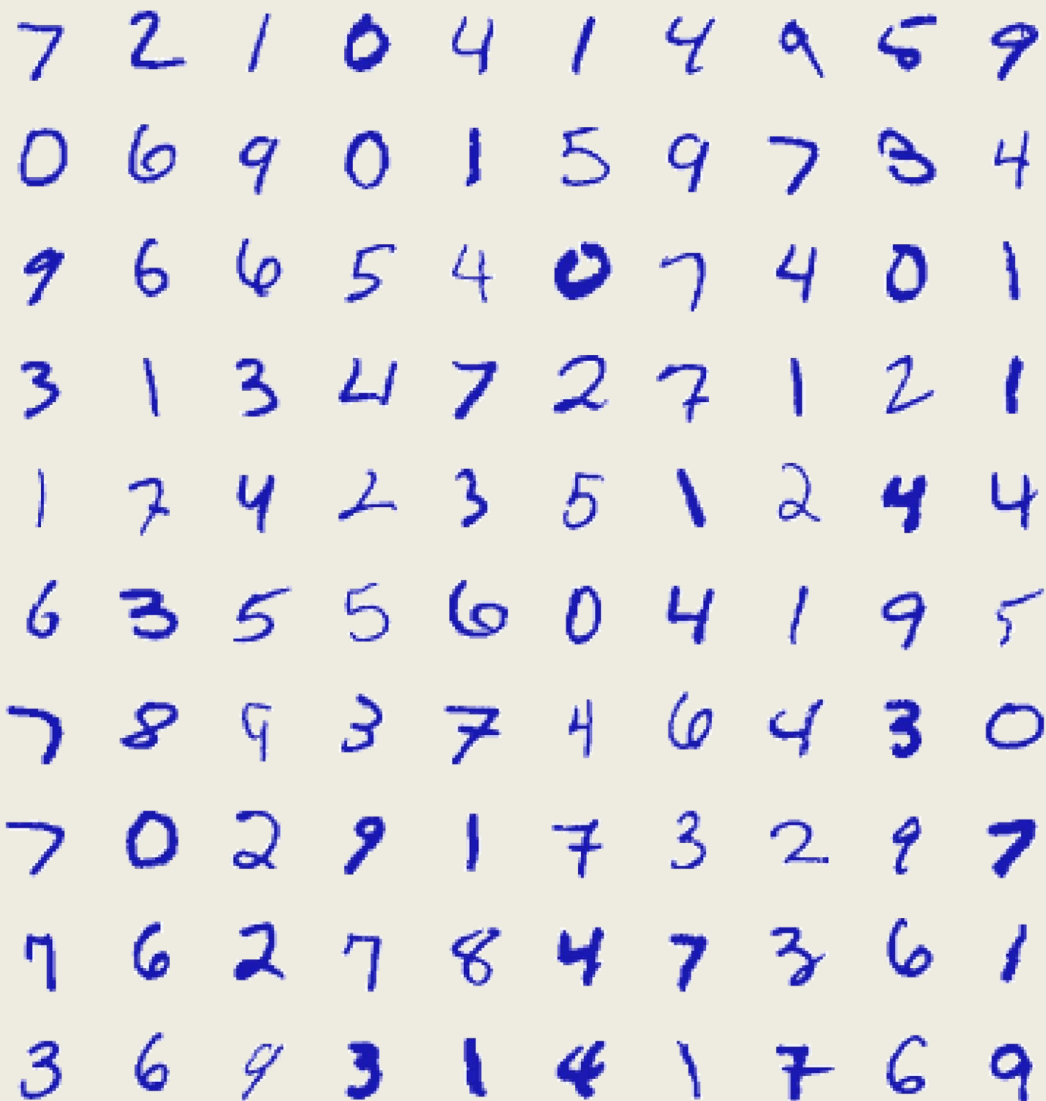
target: label 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9



Classification: Hand-Written Digit Recognition

• MINST Data Set

- originated by USPS mail
- 60000 training examples
- 10000 test examples
- normalized to 20×20
- fit to 28×28 square
- solved problem today
- scale?
- rotation?
- invariance?



Heart Abnormalities

No abnormalities



Atrial fibrillation

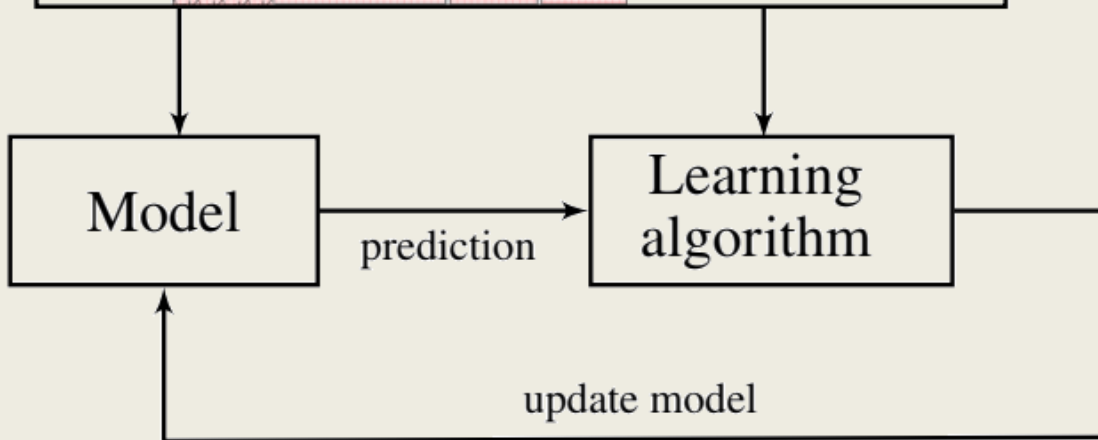


Right bundle branch block

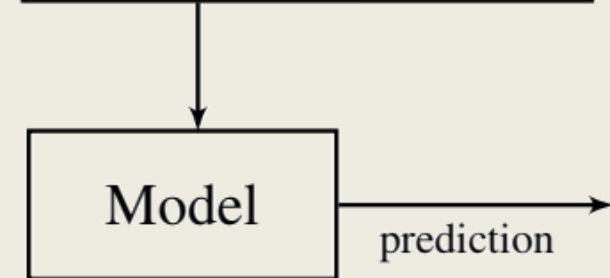
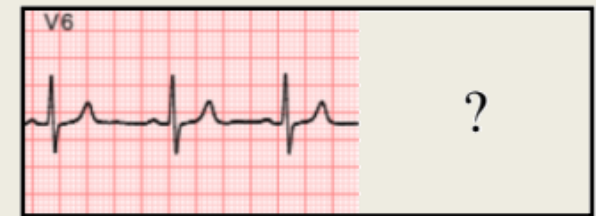


Classification: Heart Abnormalities

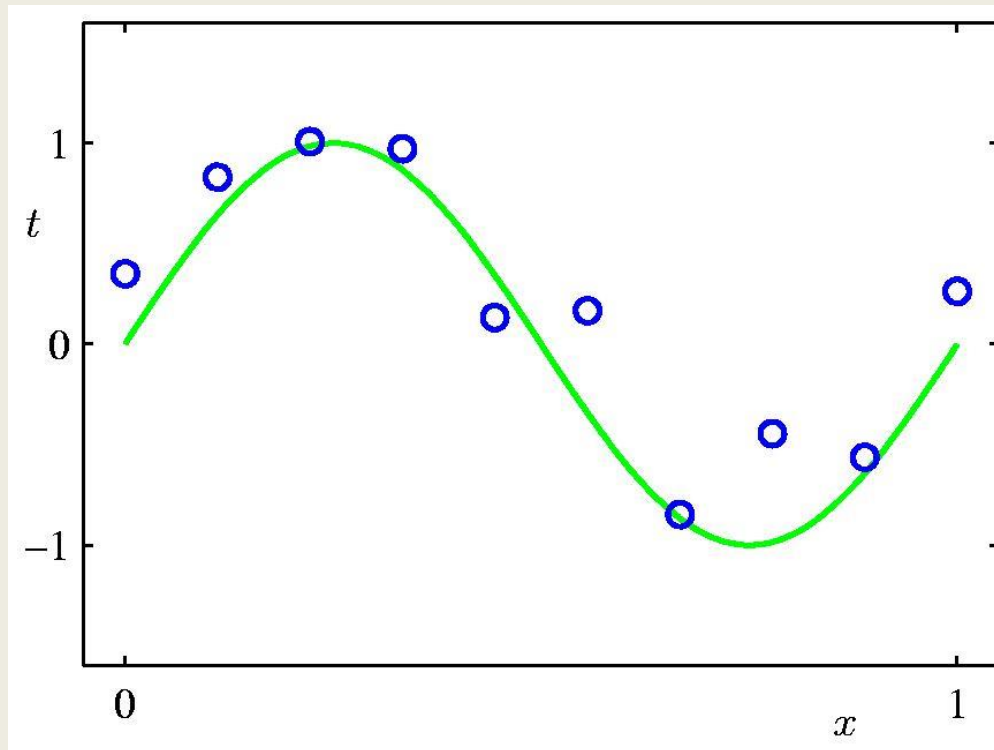
Training data



Unseen data



Regression: Polynomial Curve Fitting



$$t = \sin(2\pi x)$$

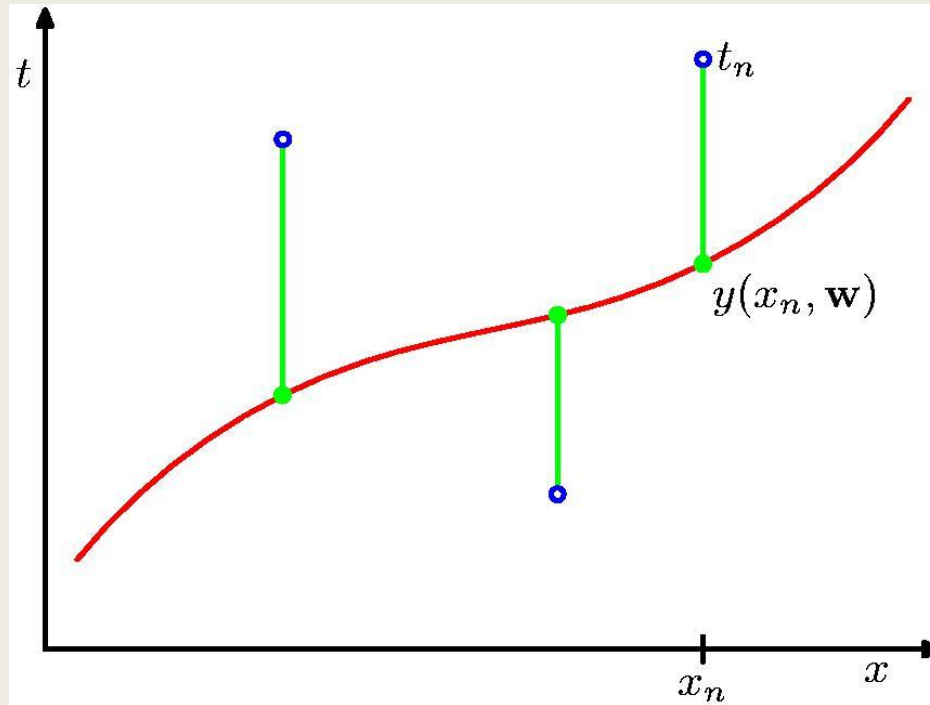
training data:
10
noisy samples
(Gaussian noise
added to targets)

model: polynomial of degree M

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$



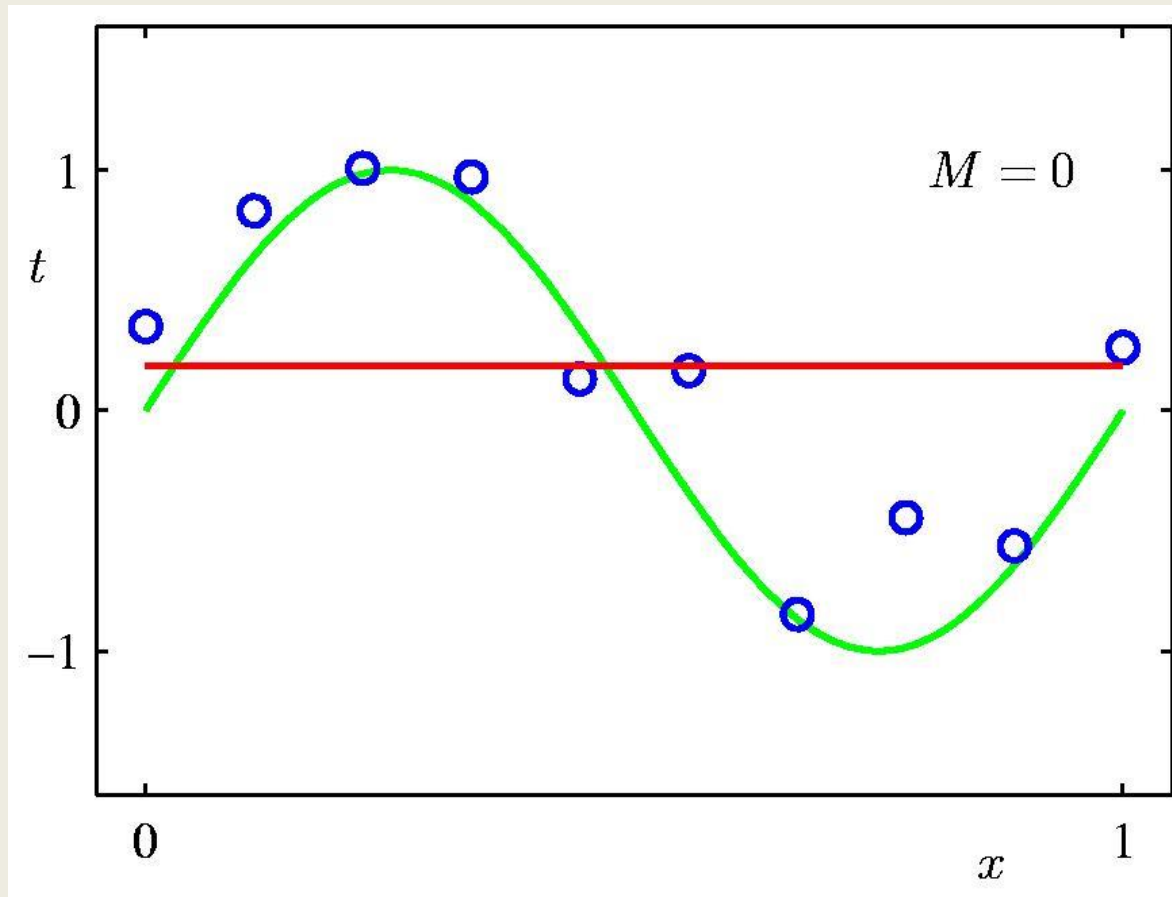
Sum-of-Squares Error Function



$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$



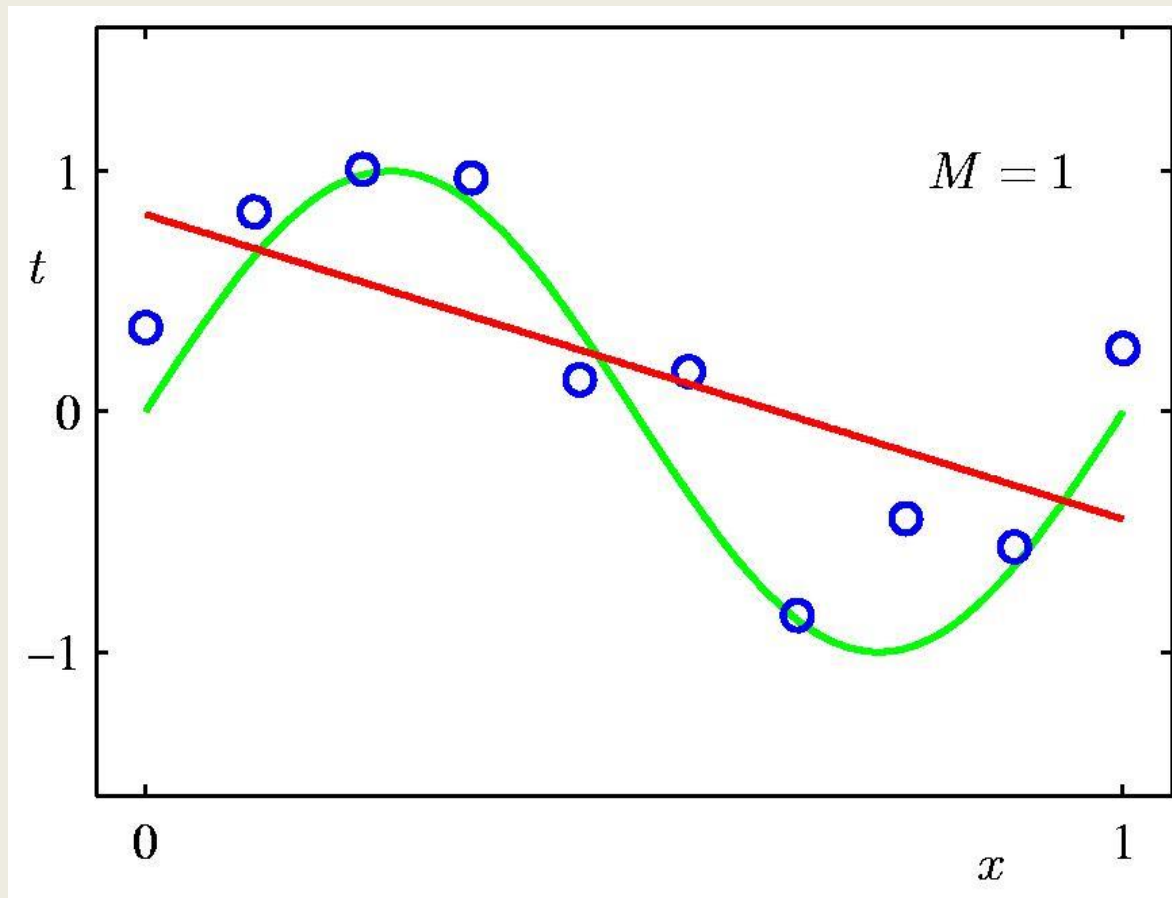
Model: 0th Order Polynomial



$$y(x, \mathbf{w}) = w_0 = \sum_{j=0}^0 w_j x^j$$



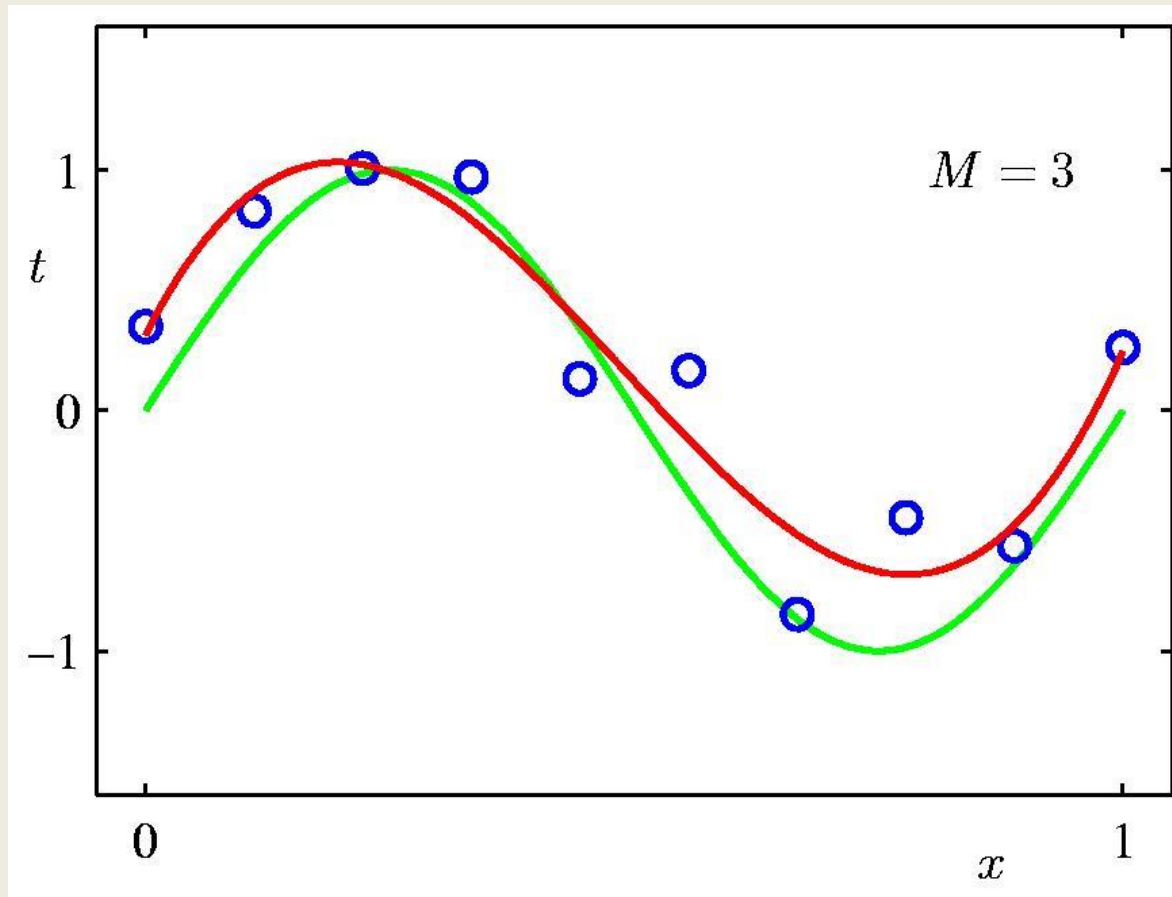
Model: 1st Order Polynomial



$$y(x, \mathbf{w}) = w_0 + w_1 x = \sum_{j=0}^1 w_j x^j$$



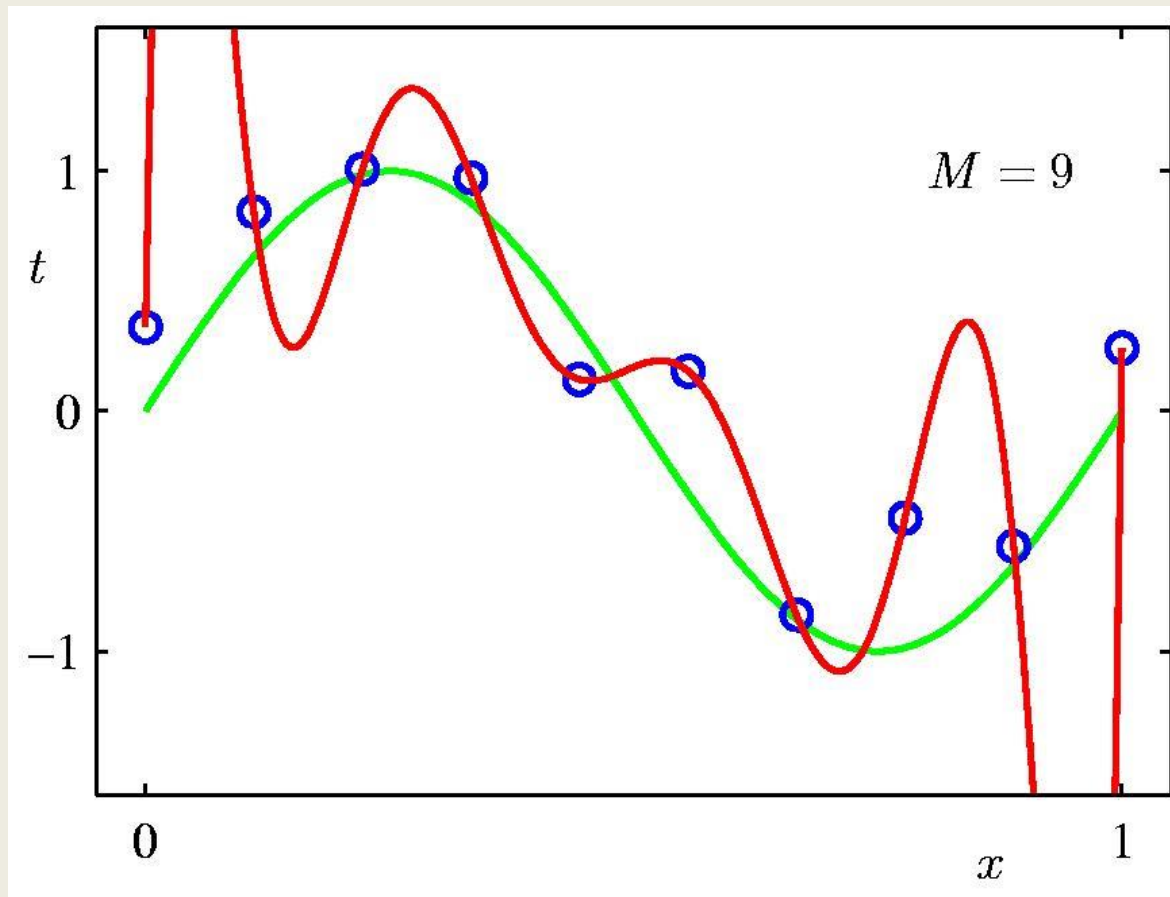
Model: 3rd Order Polynomial



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + w_3x^3 = \sum_{j=0}^3 w_j x^j$$



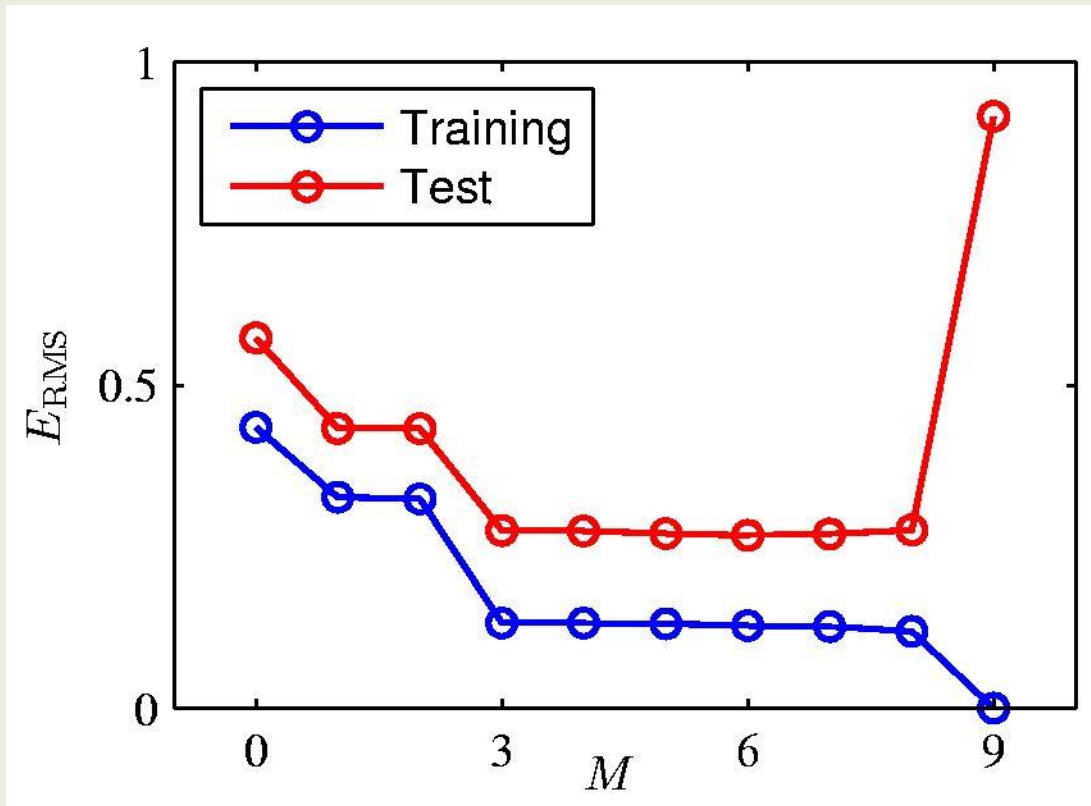
Model: 9th Order Polynomial



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_9x^9 = \sum_{j=0}^9 w_j x^j$$



Problem: Over-Fitting



|training| = 10 examples

|test|=100 examples

Root-Mean-Square (RMS) Error: $E_{\text{RMS}} = \sqrt{2E(\mathbf{w}^*)/N}$

$$E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^N \{y(x_n, \mathbf{w}) - t_n\}^2$$



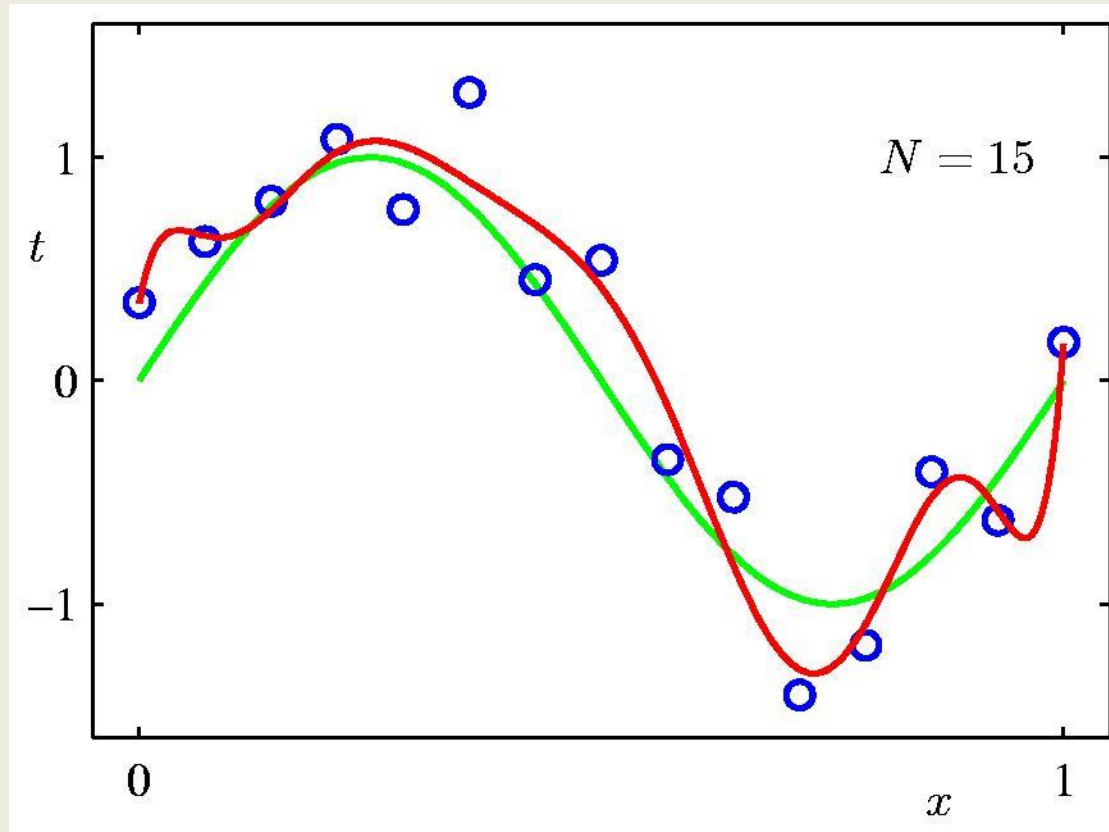
Learned Values of Polynomial Coefficients

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43



Increasing the Data Set Size: $N=15$

9th Order Polynomial

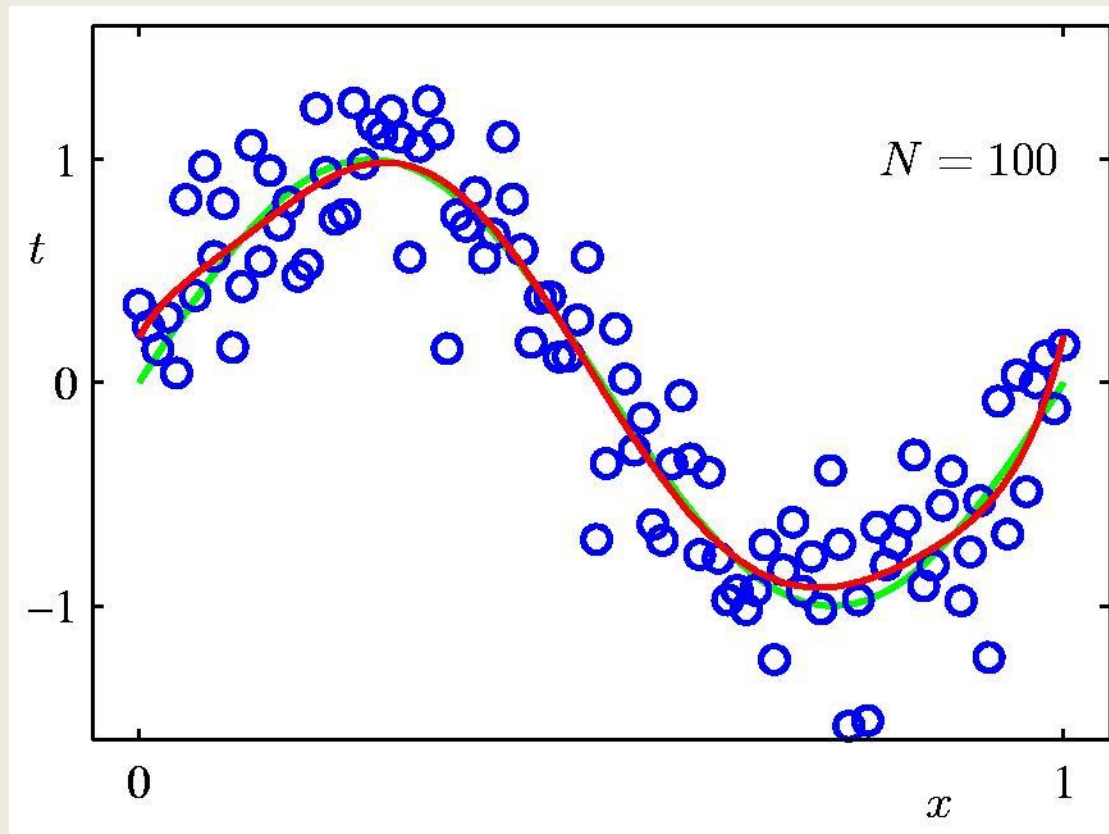


$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_9x^9 = \sum_{j=0}^9 w_j x^j$$



Increasing the Data Set Size: $N=100$

9th Order Polynomial



$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_9x^9 = \sum_{j=0}^9 w_j x^j$$



Real-Time Image Segmentation



Segmented Image (cars, signs, road, ...)

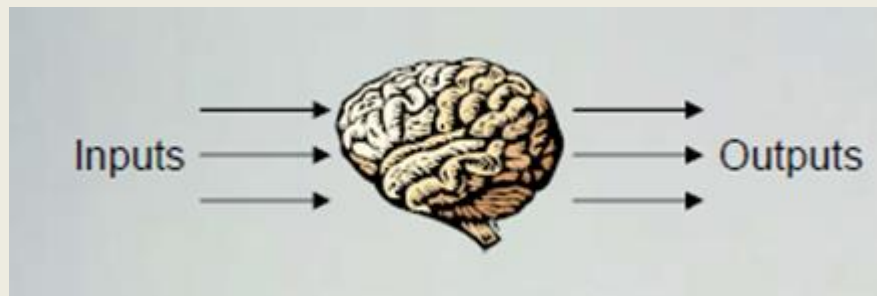


Artificial Neural Networks

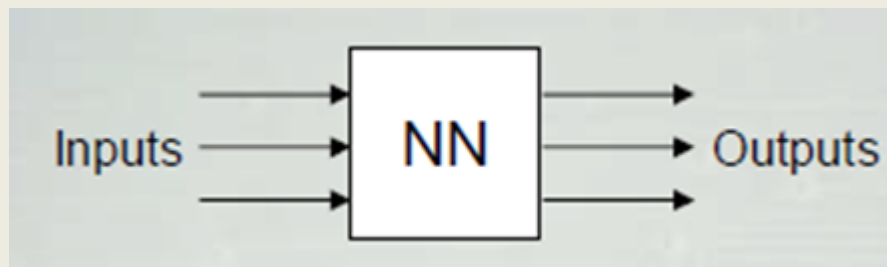


Artificial Neural Networks (ANNs)

- ANNs are a simplified model of human brain

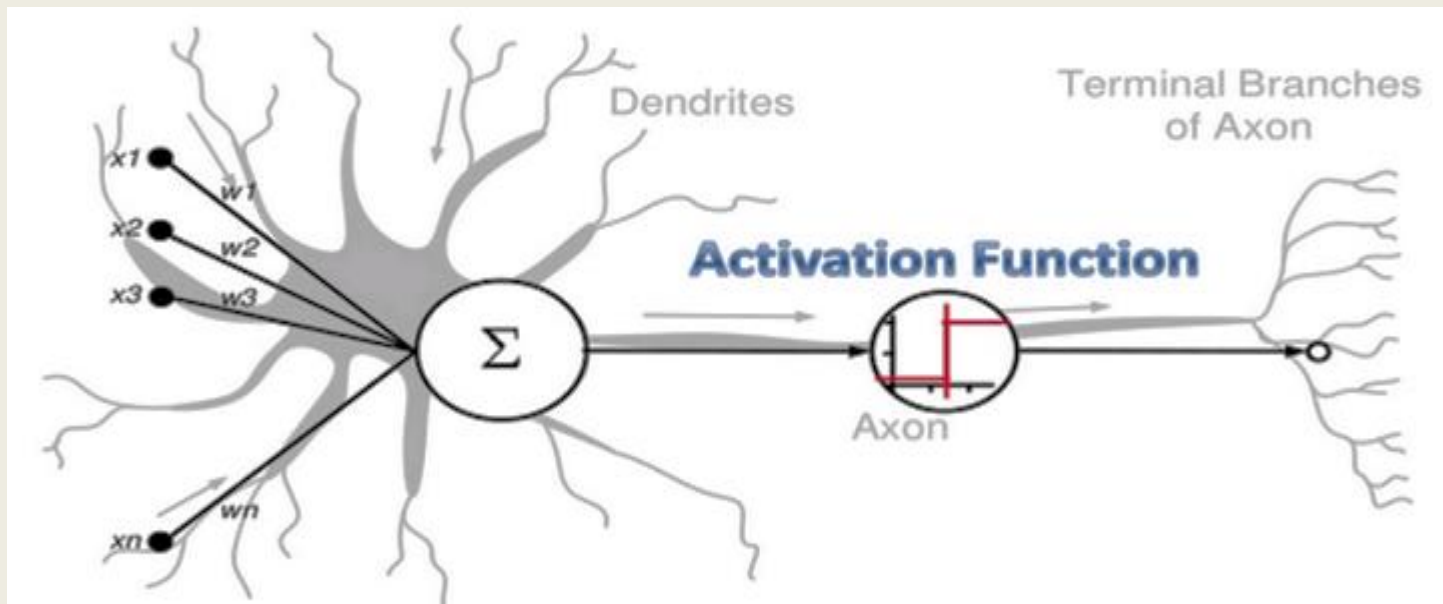


- A function which transforms input data to output data based on training over many correct examples



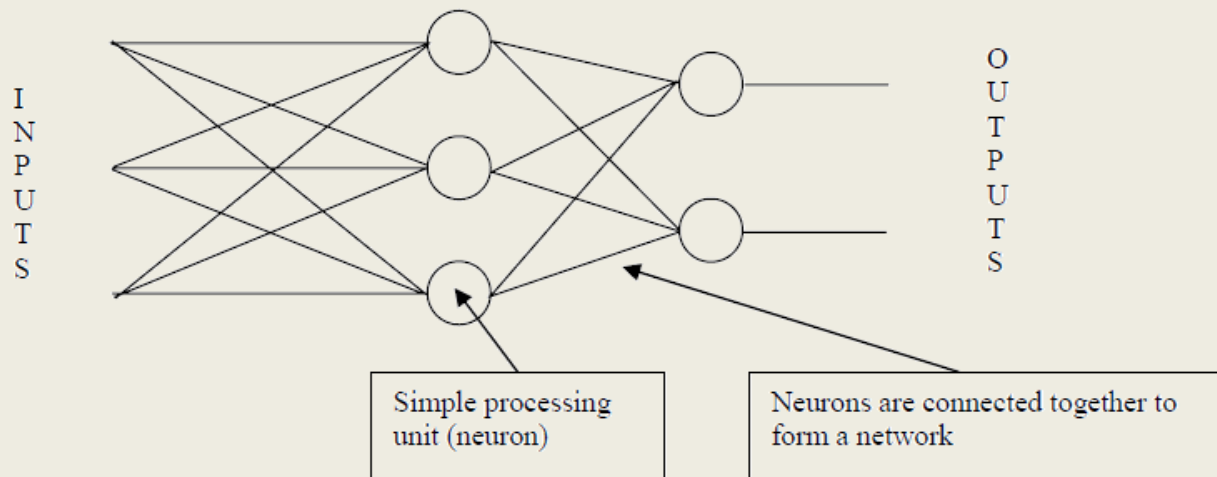
From Biology to ANNs

- Each neuron has 10^4 synapses, each with a synaptic weight
- Weights between neurons can be positive or negative
- Input: sum of the weighted input signals (linear combination)
- Depending on the total input signal the neuron is activated (stimulated) or not (inhibited): output activation function

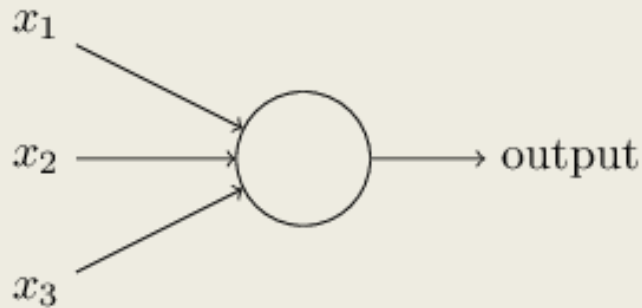


From Biology to ANNs ++

- In 1943, Warren McCulloch and Walter Pitts suggested the computational model of a neuron.
- Network: collection of connected processing units (neurons)
- Each processing unit is able to execute a simple and specific mathematical operation.
- Although individual neurons are simple, the entire neural network exhibits great processing potential.
- Flexible, non-linear systems, which exhibit “artificial intelligence”



Perceptrons



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

“bias” parameter $b \equiv -\text{threshold}$
i.e., how likely is the neuron to “fire”

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \quad w \cdot x \equiv \sum_j w_j x_j$$

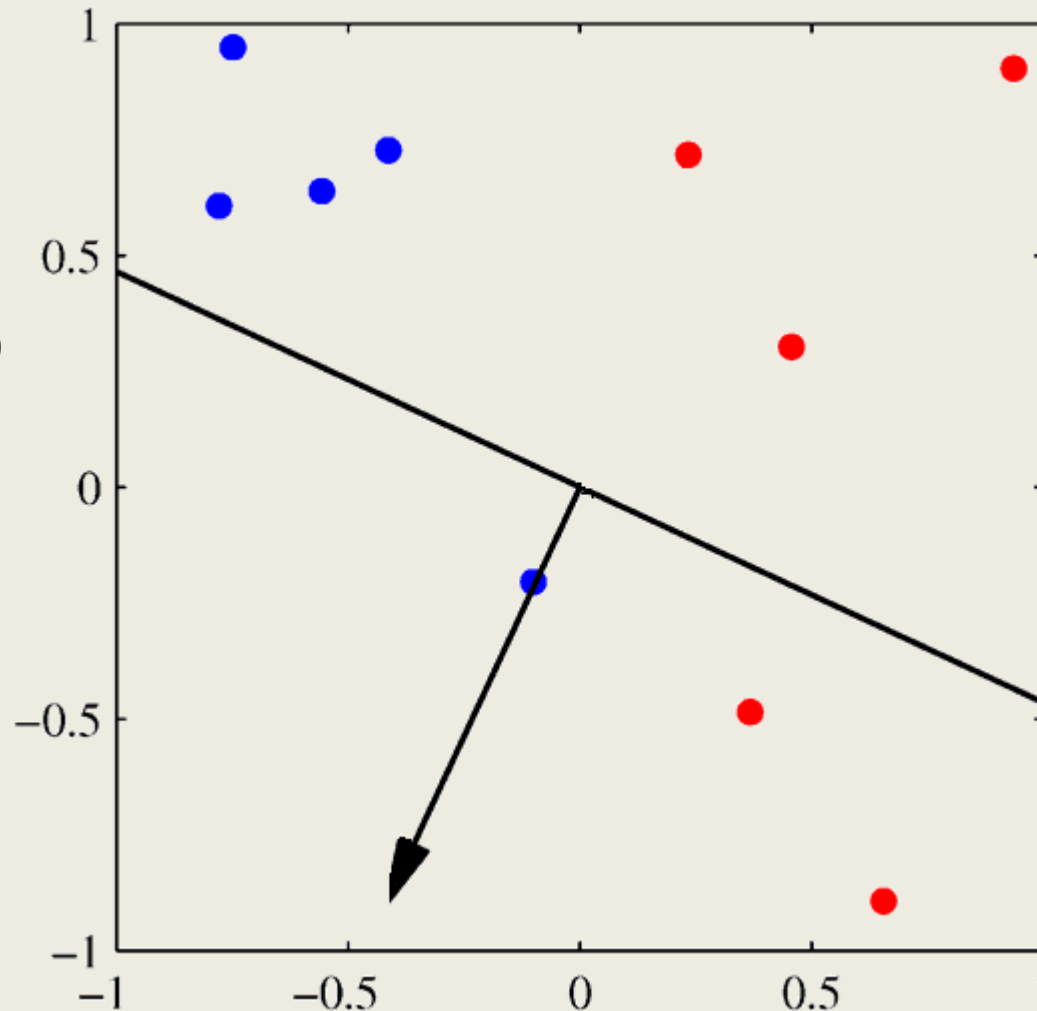
Perceptron learning

- iteratively go through the misclassified training data points
- update weights using the perceptron rule to reduce the error
- converges to a clear decision boundary, if data are separable



Perceptron Iterative Learning Example (0)

black arrow
(weight vector)
indicates the
classification
region of the
red class; the
other is **blue**

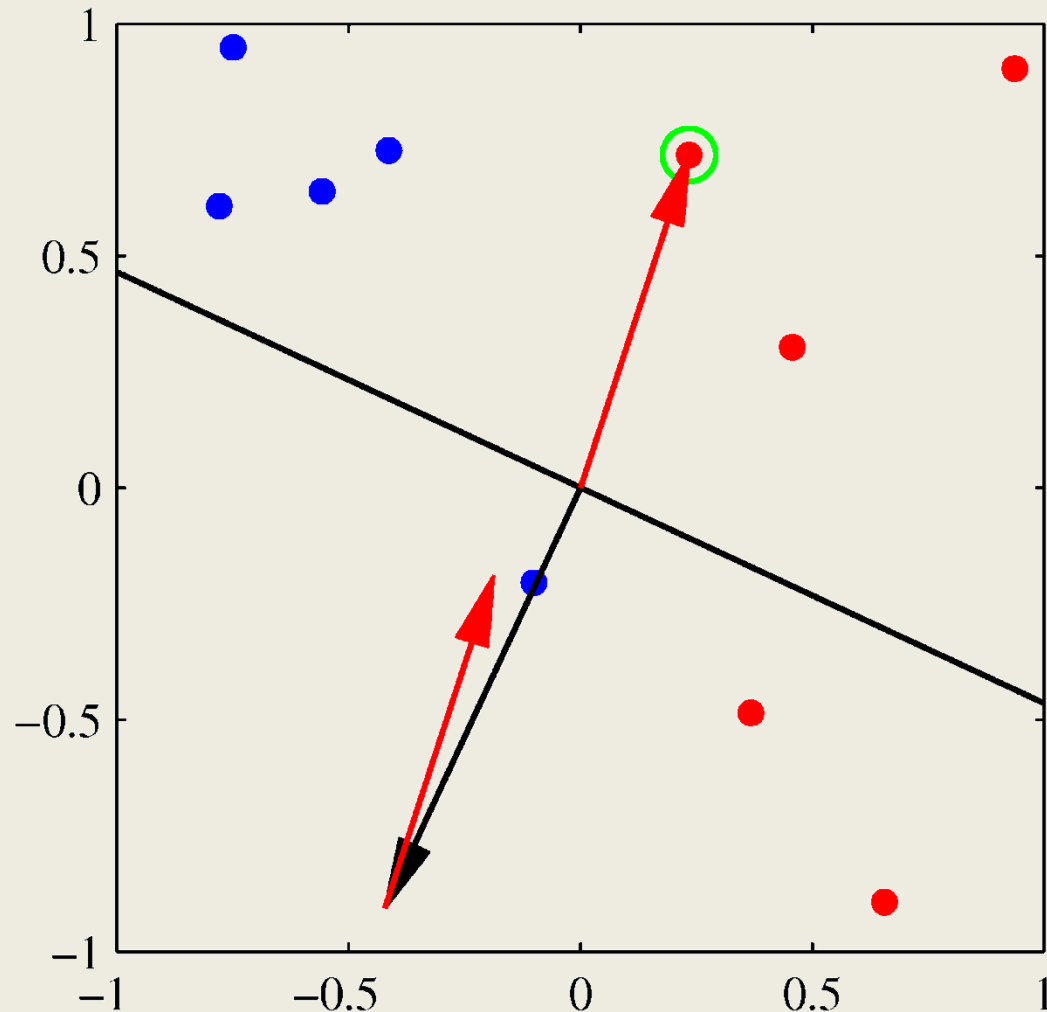


4 misclassified
data points
(3 red, 1 blue)



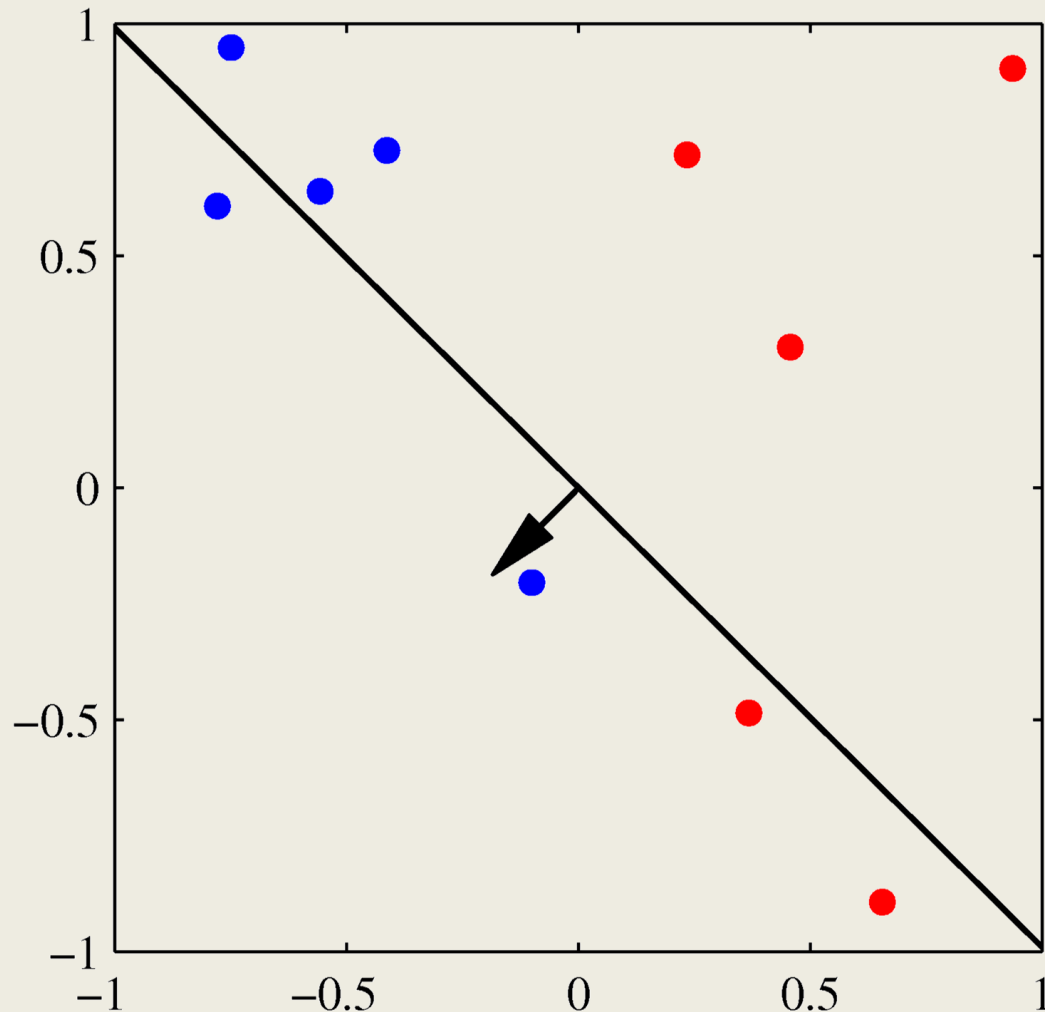
Perceptron Iterative Learning Example (1)

pick some
misclassified
data point,
compute the
error vector
(**red** arrow),
update the
weight vector



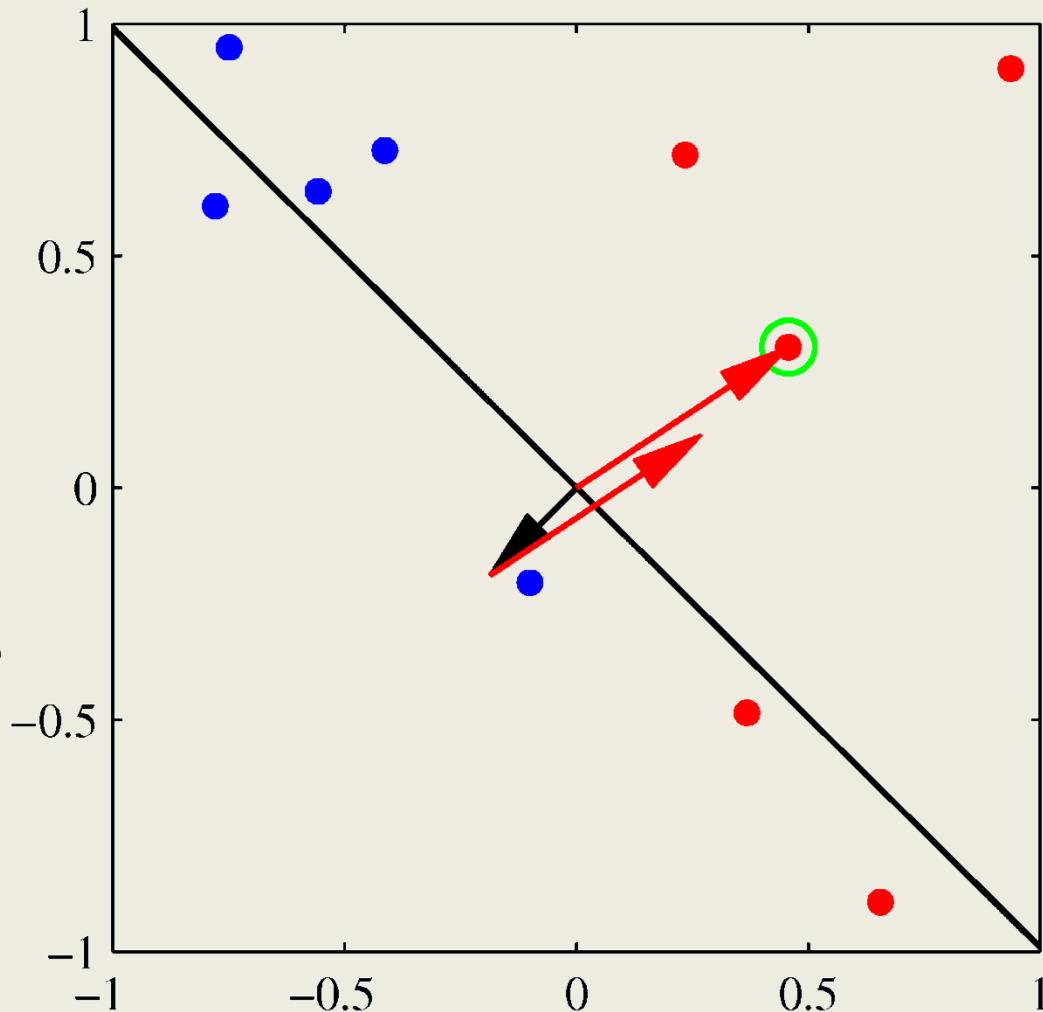
Perceptron Iterative Learning Example (2)

black arrow
(weight vector)
indicates the
classification
region of the
red class; the
other is **blue**

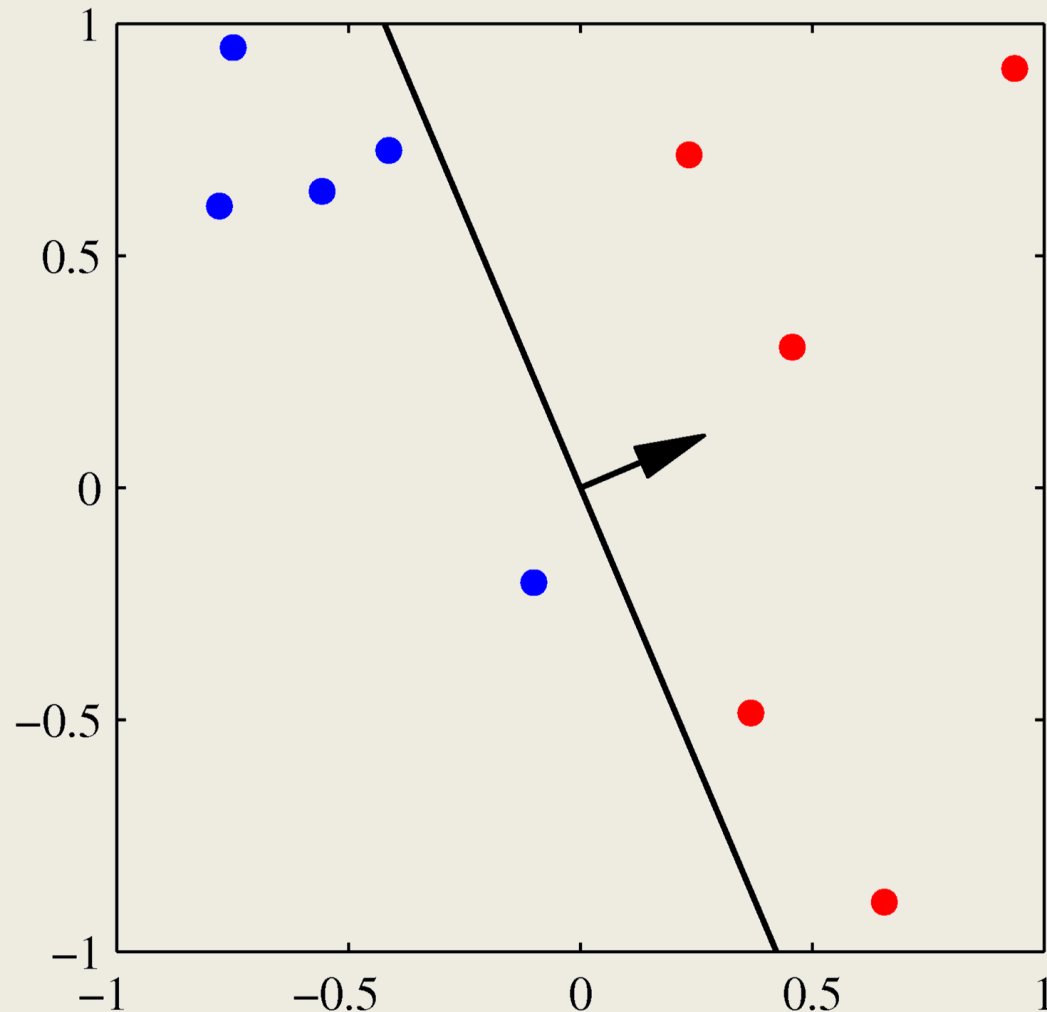


Perceptron Iterative Learning Example (3)

pick another
misclassified
data point,
compute the
error vector,
and update the
weight vector



Perceptron Iterative Learning Example (4)

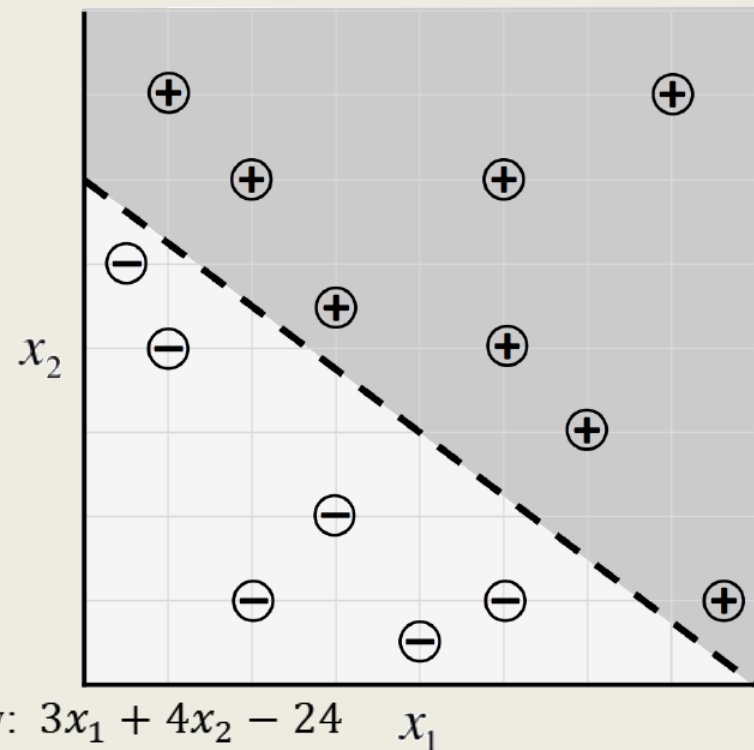
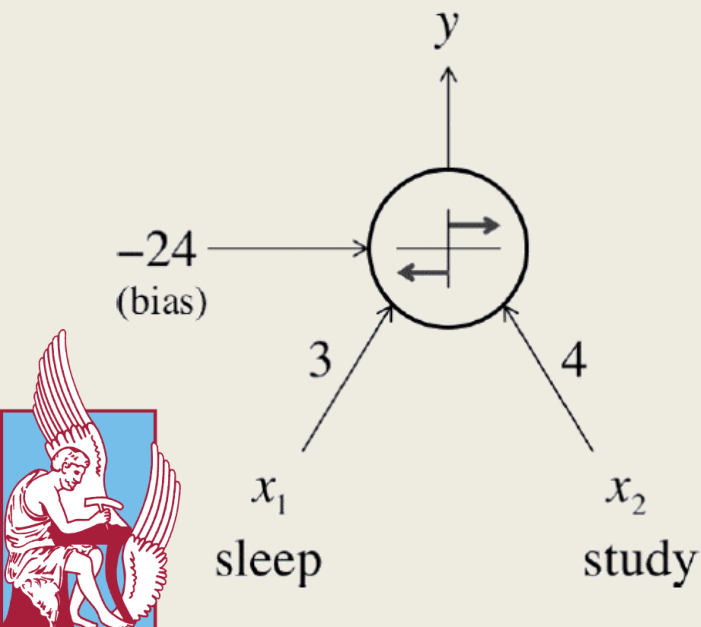


Linear Perceptron Example: Exam Performance

- Predict exam performance (above-below average) based on $\mathbf{x} = [x_1, x_2]$
 - x_1 = number of hours of sleep
 - x_2 = number of hours of study
- Define the linear classifier with parameters θ :

$$h(\mathbf{x}, \theta) = \begin{cases} -1, & \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1, & \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

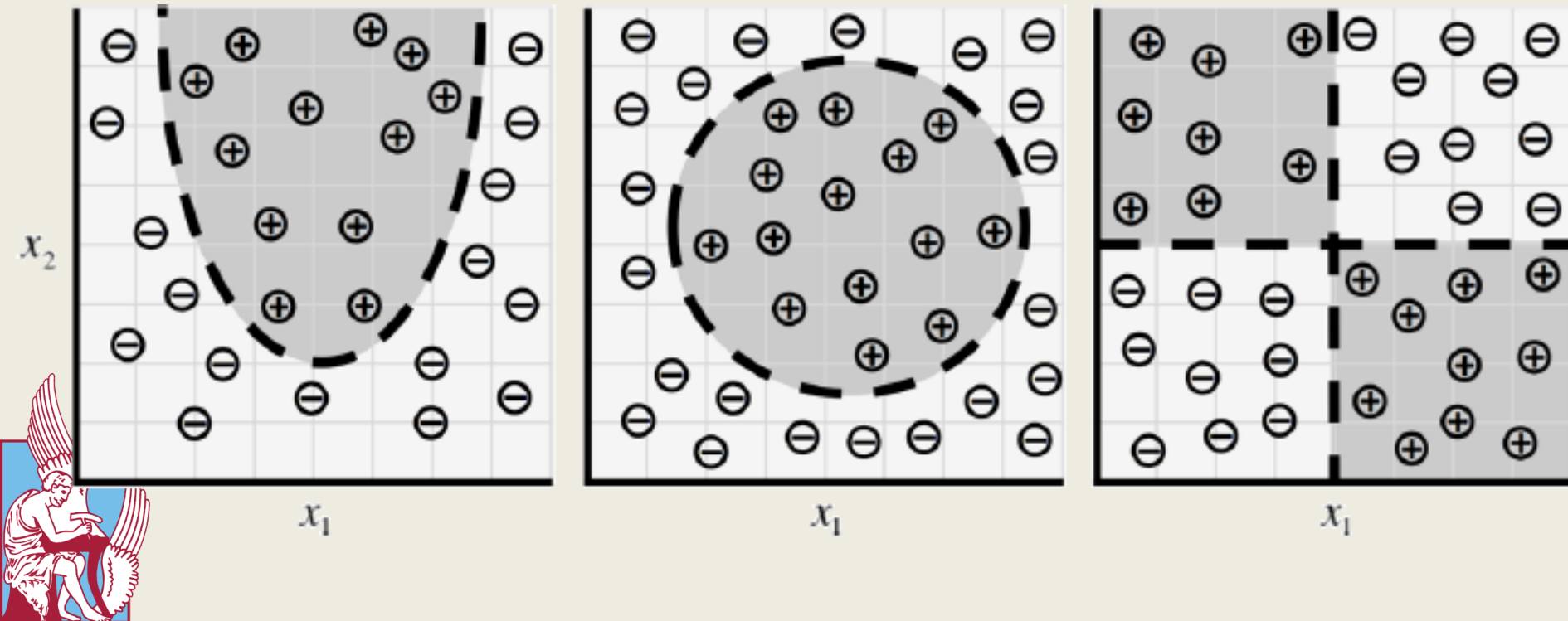
Possible solution: $\theta = [-24, 3, 4]^T$



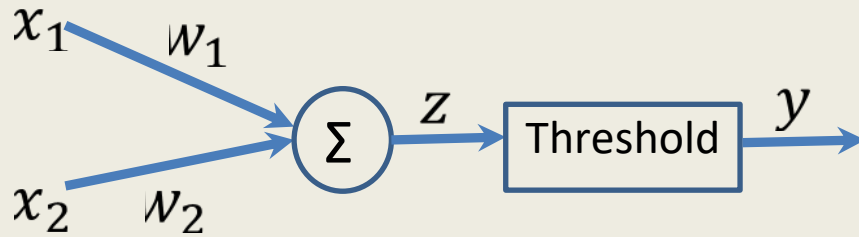
Decision boundary: $3x_1 + 4x_2 - 24$

Linear Perceptron Limitations

- perceptrons cannot handle non-separable data
- perceptrons can handle only binary classification problems
- perceptrons can only learn linear decision boundaries
- many data distributions cannot be separated linearly



Neuron's Resilience to Noise



Threshold neuron
with weights (0,1)
and threshold 0.5

x_1	x_2	Z	Output
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	1

x_1	x_2	Z	Output
0	0	0	0
0	1	0.8	1
1	0	0.2	0
1	1	1	1

Threshold neuron with
noisy weights (0.2,0.8)
and a threshold of 0.5

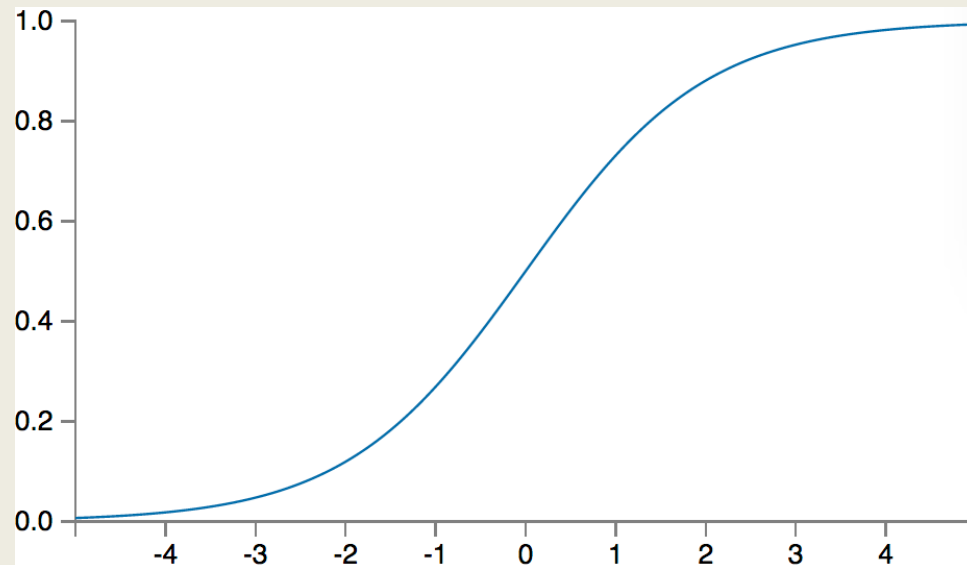
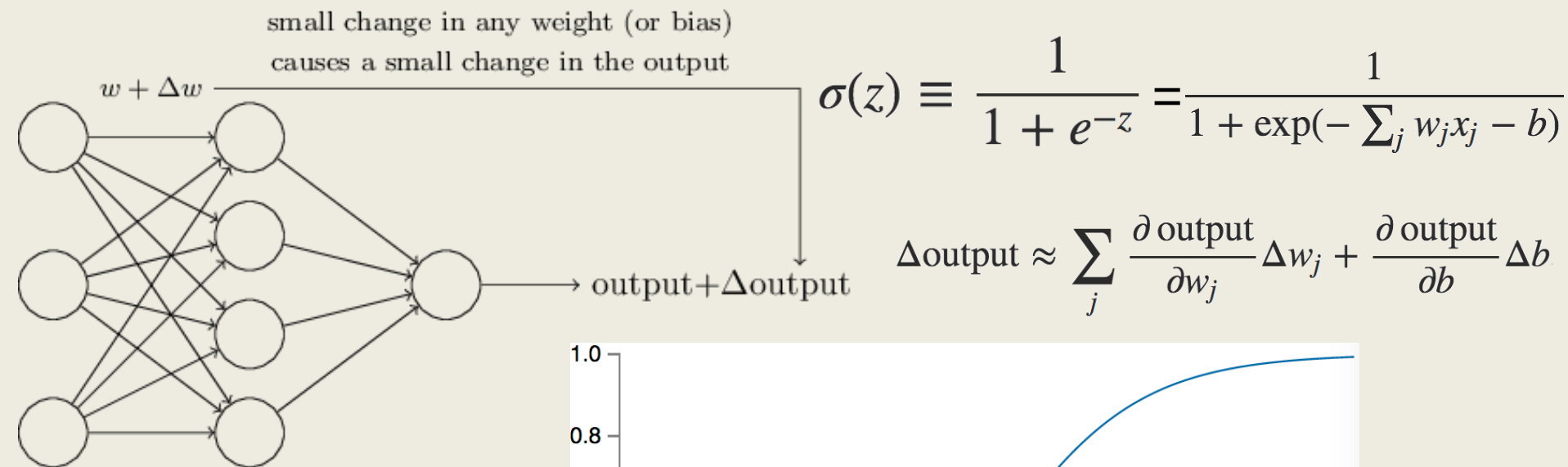
x_1	x_2	Z	Output
0.2	0.2	0.2	0
0.2	0.8	0.8	1
0.8	0.2	0.2	0
0.8	0.8	0.8	1

Threshold neuron with weights (0,1) and a
threshold of 0.5 when the input signal is noisy



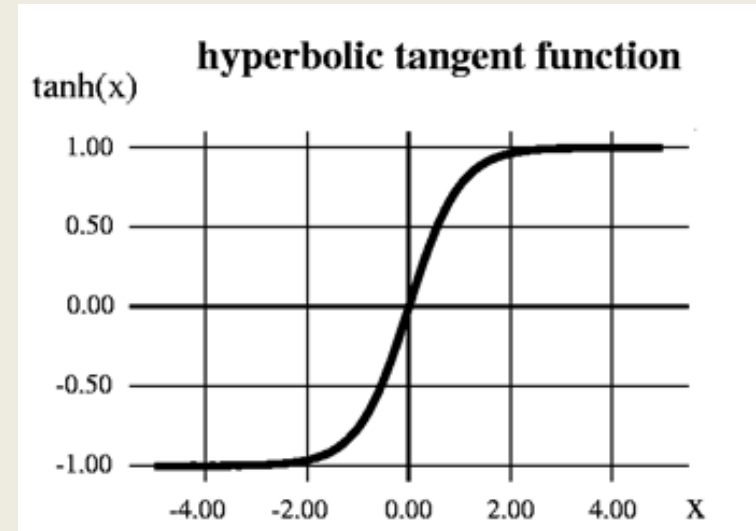
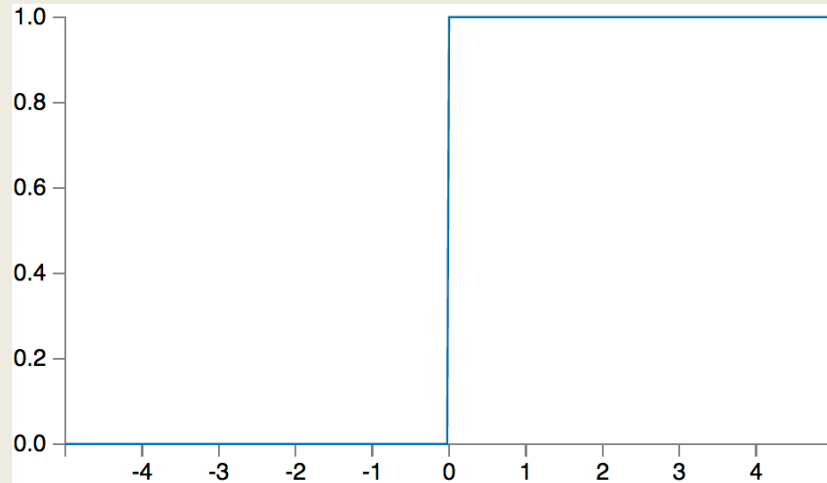
Sigmoid Activation Function

- Use *logistic* or other *sigmoid functions* for smooth behavior!



More Non-Linear Activation Functions

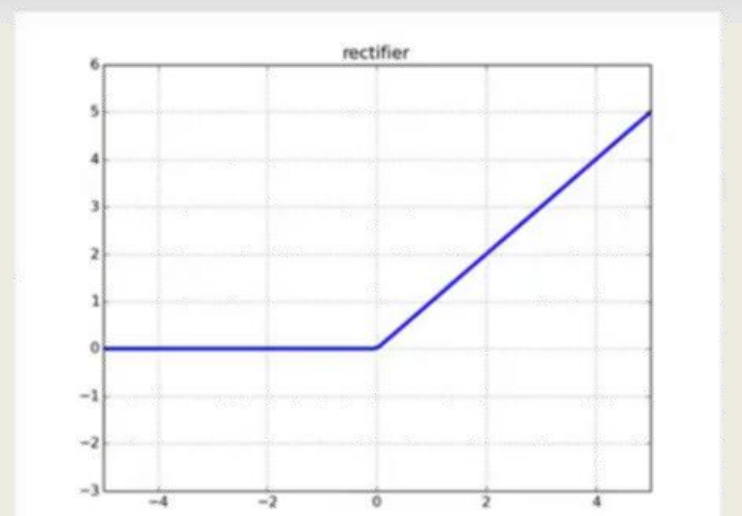
- Step/Threshold function



- Rectified Linear Unit (ReLU)

ReLU

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



SoftMax Neurons

- Activation Function:

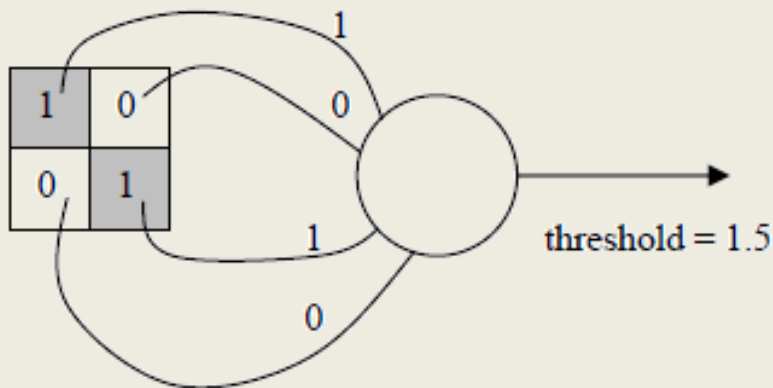
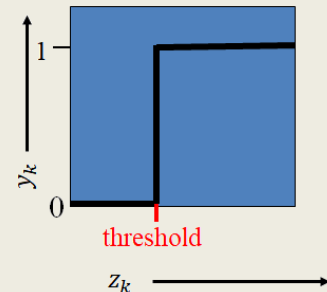
$$y_l(k) = \frac{e^{z_l(k)}}{\sum_{i=1}^{d_l} e^{z_l(i)}}$$

- d_l number of neurons
- Usually used at the neurons of the output layer
- Converts a vector of real numbers to a vector of posterior probabilities
 - Output values are between 0 and 1 and sum to unit
 - The output of a neuron depends the outputs of all other neurons in its layer
 - It is a measure of uncertainty of the output
 - Strong prediction → single entry in the output vector close to 1
 - Weak prediction → multiple entries almost equally likely



Recognizing Simple Patterns

- Image with 4 grey scaled pixels (from 0=white to 1=black)
 - $z = (1 \times 1) + (0 \times 0) + (1 \times 1) + (0 \times 0) = 2$
 - Define a threshold value $\theta = 1.5$ to emulate the action potential
 - $z = 2 > 1.5 \Rightarrow \text{Output} = 1$
 - Even if some white pixels had values a bit more than 0 the neuron could recognize the pattern correctly



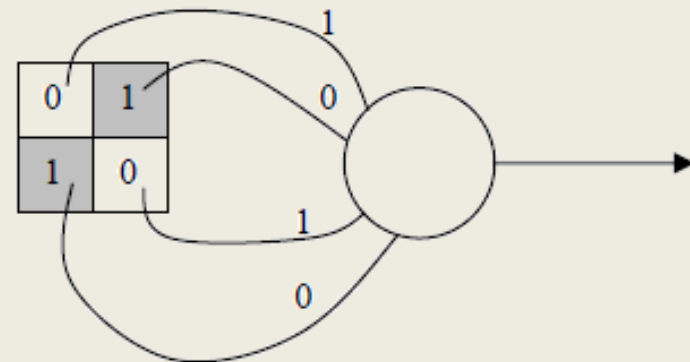
Output = 1 if pattern
is recognised

Output = 0 if pattern
is not recognised

- Rotate the same image (pattern):

- $z = (0 \times 1) + (1 \times 0) + (0 \times 1) + (1 \times 0) = 0 < 1.5$

- Fail to recognize



Recognizing Simple Patterns

- Can it recognize the noisy image? →

0.7	0.1
0.2	0.9

– $z = (0.7 \times 1) + (0.9 \times 1) = 1.6 > 1.5 \rightarrow$ marginally successful

- Use the logistic function as activation:

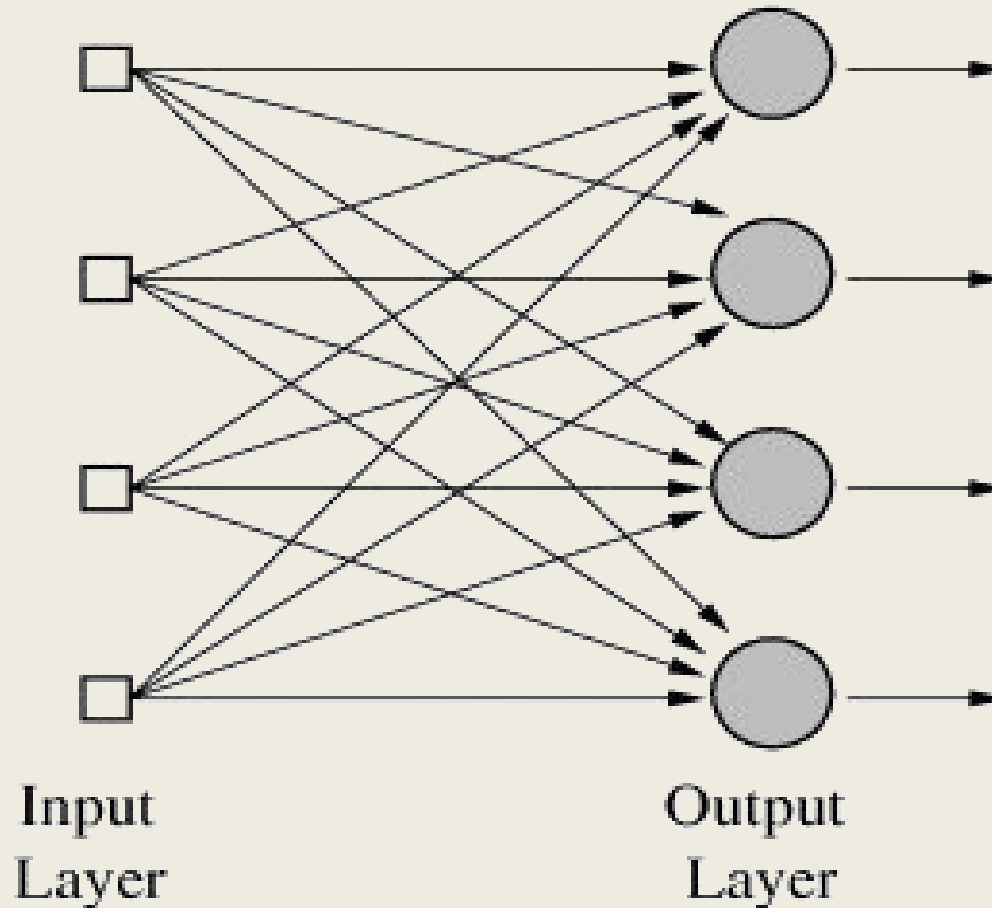
– $\text{output} = \frac{1}{1+e^{-1.6}} = 0.8320$ (close to max=1)

$$y_k = \frac{1}{1 + \exp(-z_k)}$$

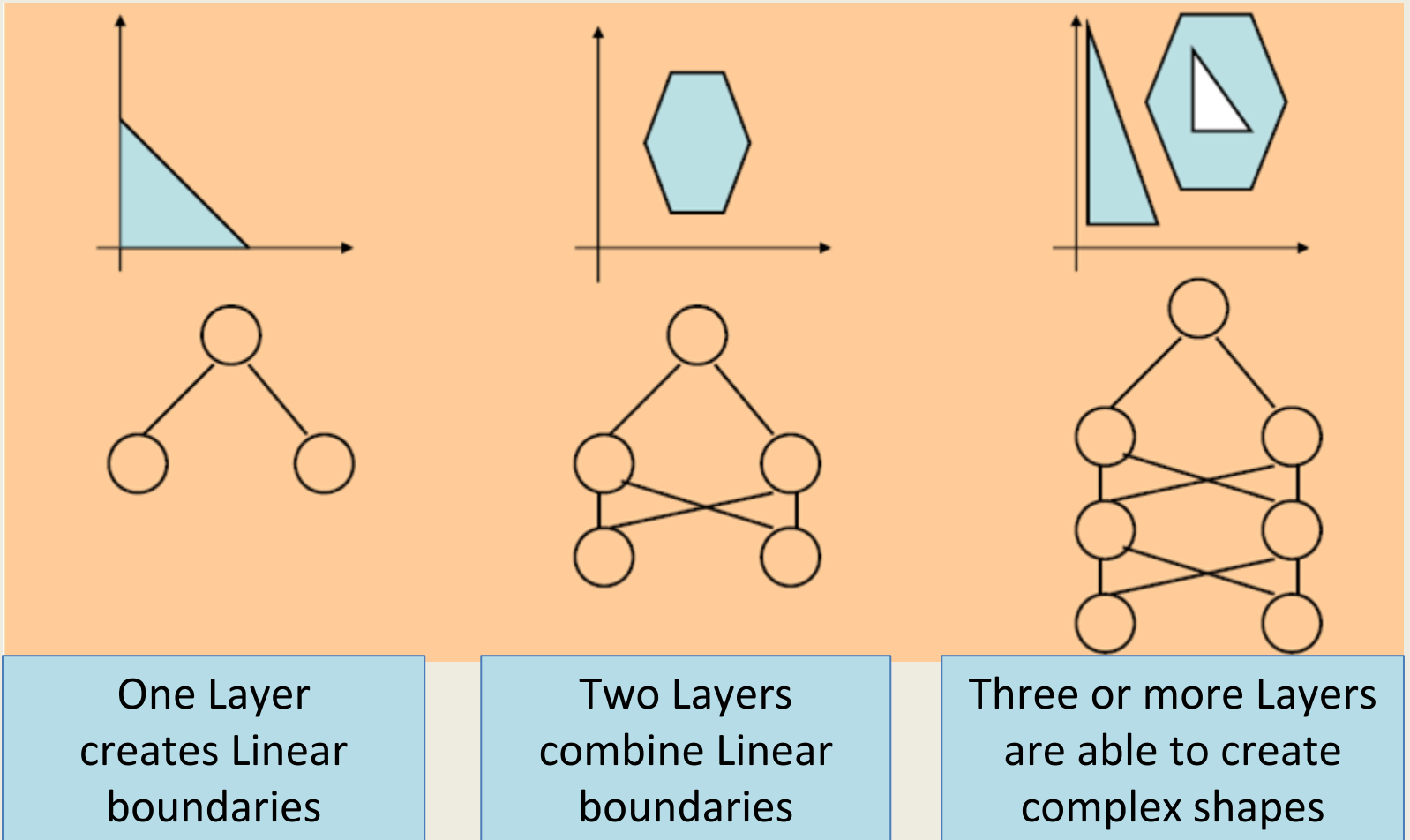


Neural Network Structure

- Neurons are organized in layers
 - simplest form: 1 Input layer and 1 output layer

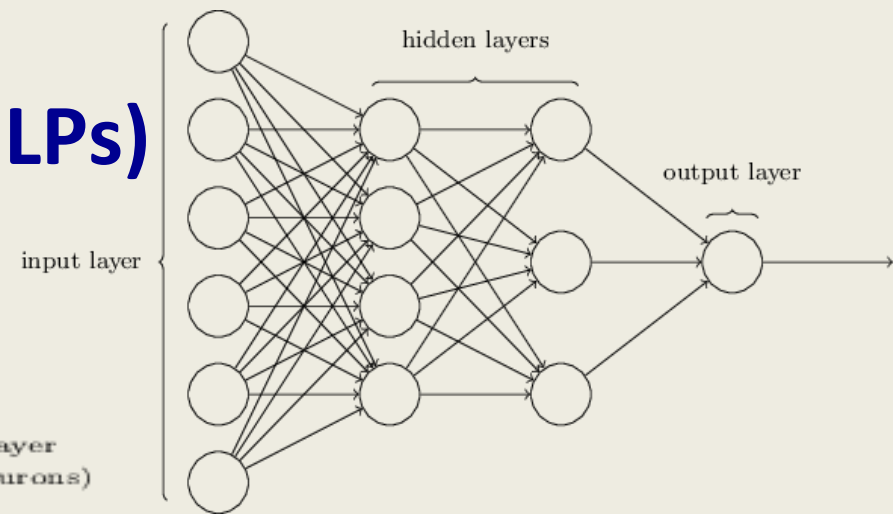


Why do we need hidden layers?

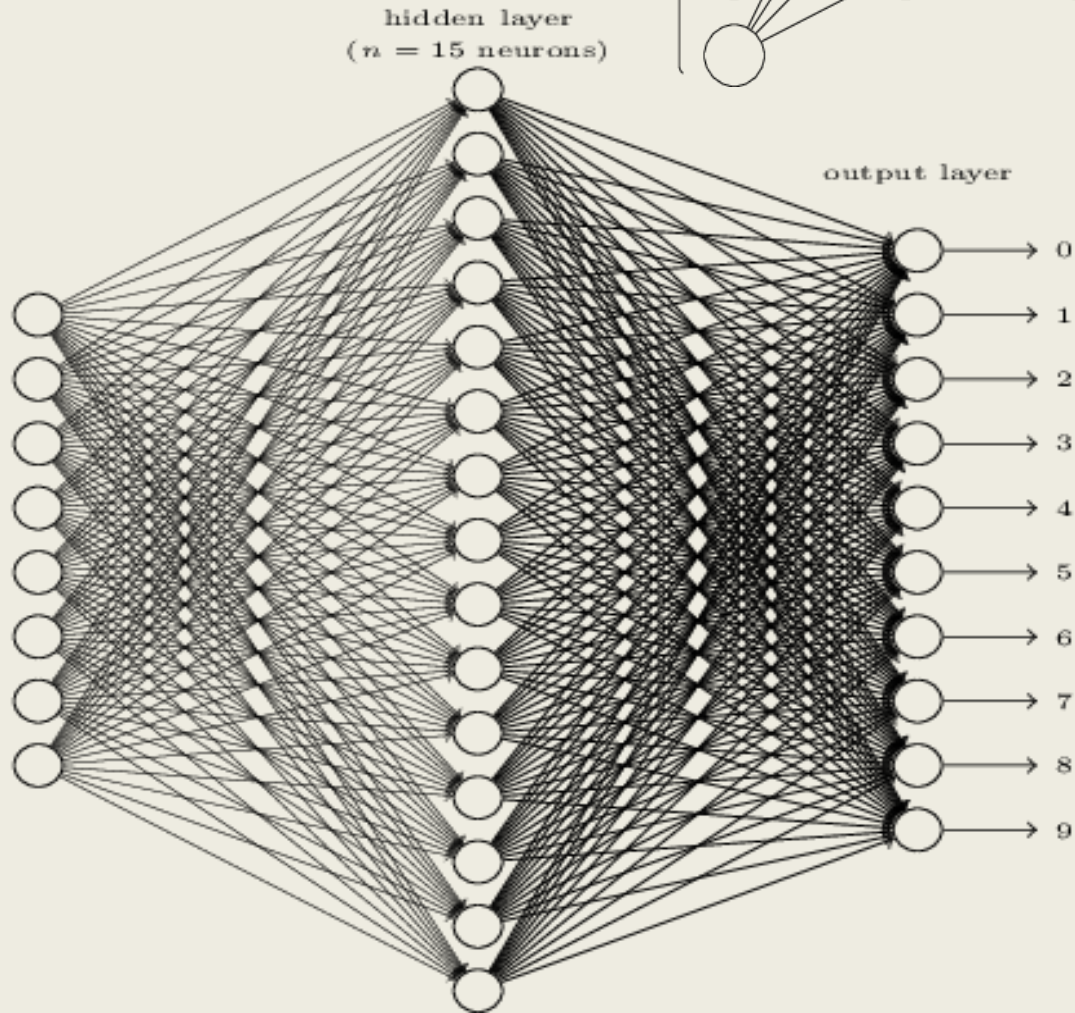


Multi-Layer Perceptrons (MLPs)

- Terminology confusing! Neurons in MLPs use the sigmoid function for their output.



input layer
(784 neurons)



Learning Process of Neural Networks

- The NN structure is determined first (inference model)
- how many layers? how many units per layer? activation functions?...

- **Learning process**

- The NN learns a specific task by adjusting its parameters (weights) from the correctly labeled training examples (supervised learning)

- **Adjusting the parameters**

1. randomly initialize the weights
2. for some input, obtain a predicted output from the network
3. if the output matches the real label, do not change the weights
4. if the output is larger than the real label,
adjust the weights that contribute to large output values
5. if the output is smaller than the real label,
adjust the weights that contribute to small output values
6. repeat the previous steps, until the error converges to a minimum



<http://playground.tensorflow.org>



Epoch
000,254

Learning rate

0.1

Activation

Tanh

Regularization

None

Regularization rate

0

Problem type

Classification

DATA

Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



REGENERATE

FEATURES

Which properties do you want to feed in?

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$

$\sin(X_1)$

$\sin(X_2)$

+ - 2 HIDDEN LAYERS

+ -

4 neurons

+ -

2 neurons

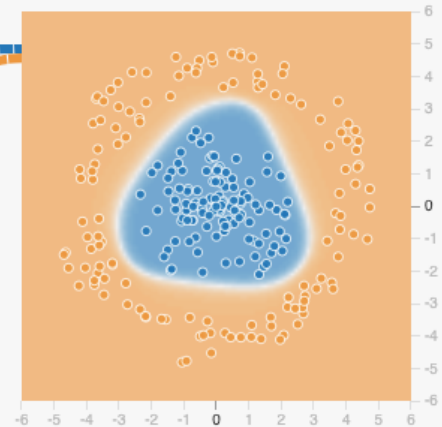
The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

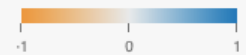
OUTPUT

Test loss 0.003

Training loss 0.001



Colors shows data, neuron and weight values.



☐ Show test data

☐ Discretize output



Backpropagation



How to learn?

Use gradient descent on the cost function

The key idea

- adjust a model's weights in response to the error it produces, until you can't reduce the error any more

How do the errors vary, as the weights are adjusted?

- calculate how a change in *weight* affects a change in *Error*
- first, calculate how a change in *activation* affects a change in *Error*,
- and then, how a change in *weight* affects a change in *activation*

$$\frac{dError}{dweight} = \frac{dError}{dactivation} * \frac{dactivation}{dweight}$$



How to learn?

Use gradient descent on the cost function

- **use a smooth cost function** → helps figure out how to make small changes in weights and biases to get an improvement in the cost

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

- **perform stochastic gradient descent** to guarantee reduction of cost

$X_i \leftarrow$ training inputs as **a mini-batch**
(small set of m training inputs)

A training epoch

Train over

a randomly chosen mini-batch,
then on another,
until you exhaust all mini-batches

[repeat]

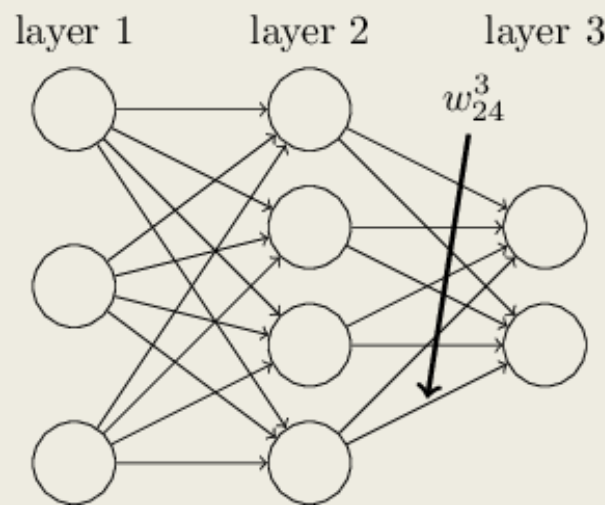
$$w_k \rightarrow w'_k = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l},$$



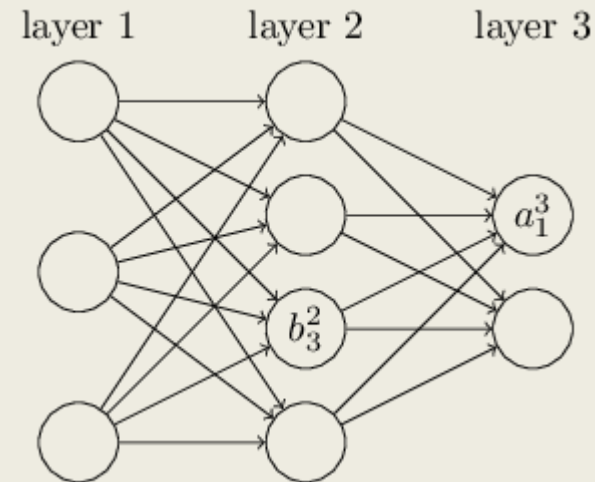
Computing the gradient of the cost function: The Backpropagation Algorithm

- But, how to compute the gradient of the cost function?
- Use the backpropagation algorithm!



w_{jk}^l is the weight from the k^{th} neuron in the $(l-1)^{\text{th}}$ layer to the j^{th} neuron in the l^{th} layer

$$a_j^l = \sigma \left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$



$$a^l = \sigma(w^l a^{l-1} + b^l)$$

how do activations in one layer relate to those in the previous layer? apply the weight matrix to the activations, add the bias vector, and apply σ

$$a^l = \sigma(z^l) \quad \text{with the weighted input } z^l \quad z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l$$



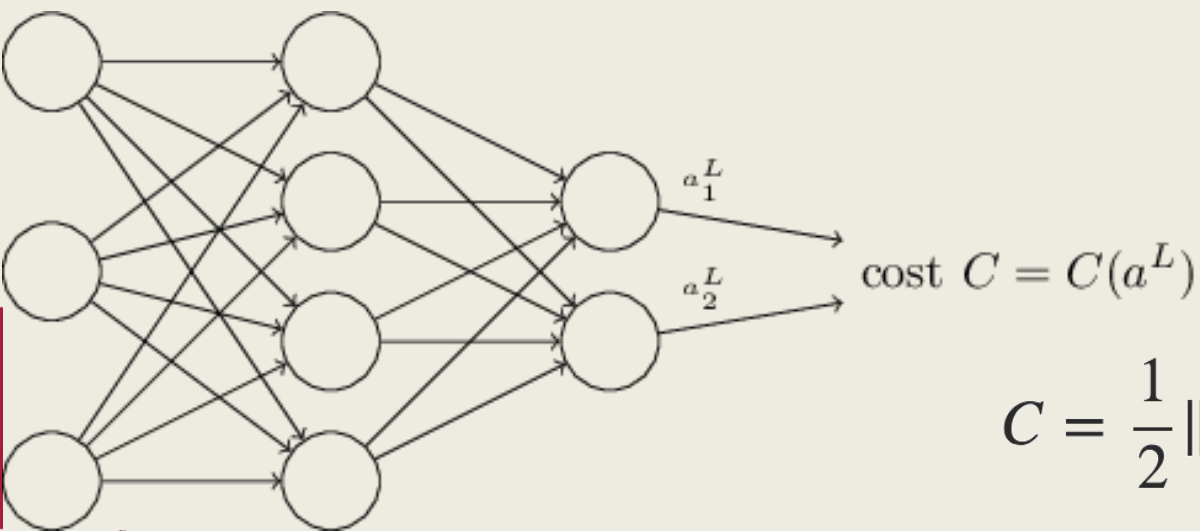
Let's define the cost function

$$C = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2$$

L : number of layers; n : total number of training examples x ; $y(x)$: desired output;
 $a^L(x)$: vector of activations that are output from the network, when x is input

We take $C = \frac{1}{n} \sum_x C_x$ (required for averaging over all training examples)

We also take C to be a quadratic function of the output activations



$$C = \frac{1}{2} \|y - a^L\|^2 = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

Computing the gradient of the cost function

the error δ_j^l of neuron j in layer l $\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$

$$\frac{d\text{Error}}{d\text{weight}} = \frac{d\text{Error}}{d\text{activation}} * \frac{d\text{activation}}{d\text{weight}}$$

the error in the output layer

$$\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$$

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

$s \odot t$ denotes the *elementwise* product of the two vectors

$$(s \odot t)_j = s_j t_j \quad \begin{bmatrix} 1 \\ 2 \end{bmatrix} \odot \begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 * 3 \\ 2 * 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 8 \end{bmatrix}$$

For the quadratic cost function

$$C = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

we can easily compute the gradient with respect to the activation

$$\partial C / \partial a_j^L = (a_j^L - y_j)$$



Computing the gradient of the cost function

The equation for the error δ^l in terms of the error δ^{l+1} in the next layer

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

propagates the error backward (from layer $l+1$ to layer l),
first assessing it on l layer's output, then to l 's weighted input via σ .

The rate of change of the cost with respect to any **bias** in the network

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

The rate of change of the cost with respect to any **weight** in the network

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$



The Backpropagation Algorithm

1. **Input x :** Set the corresponding activation a^1 for the input layer.
2. **Feedforward:** For each $l = 2, 3, \dots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma(z^l)$.
3. **Output error δ^L :** Compute the vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.
4. **Backpropagate the error:** For each $l = L - 1, L - 2, \dots, 2$ compute $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$.
5. **Output:** The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.



The Backpropagation Algorithm

In other words:

- We compute the error vectors δ' backward, layer by layer, starting from the final layer.
- The backward propagation is a consequence of the fact that the cost is a function of outputs from the network.

Now, we can combine backpropagation with stochastic gradient descent for adjusting weights and biases, in which we compute the gradient for **many** training examples.

In practice, we also need:

- an outer loop generating mini-batches of training examples
- another outer loop stepping through multiple epochs of training



Gradient descent with backpropagation for a mini-batch of m training examples



1. Input a set of training examples

2. For each training example x : Set the corresponding input activation $a^{x,1}$, and perform the following steps:

- **Feedforward:** For each $l = 2, 3, \dots, L$ compute

$$z^{x,l} = w^l a^{x,l-1} + b^l \text{ and } a^{x,l} = \sigma(z^{x,l}).$$

- **Output error $\delta^{x,L}$:** Compute the vector

$$\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L}).$$

- **Backpropagate the error:** For each

$$l = L - 1, L - 2, \dots, 2 \text{ compute}$$

$$\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l}).$$

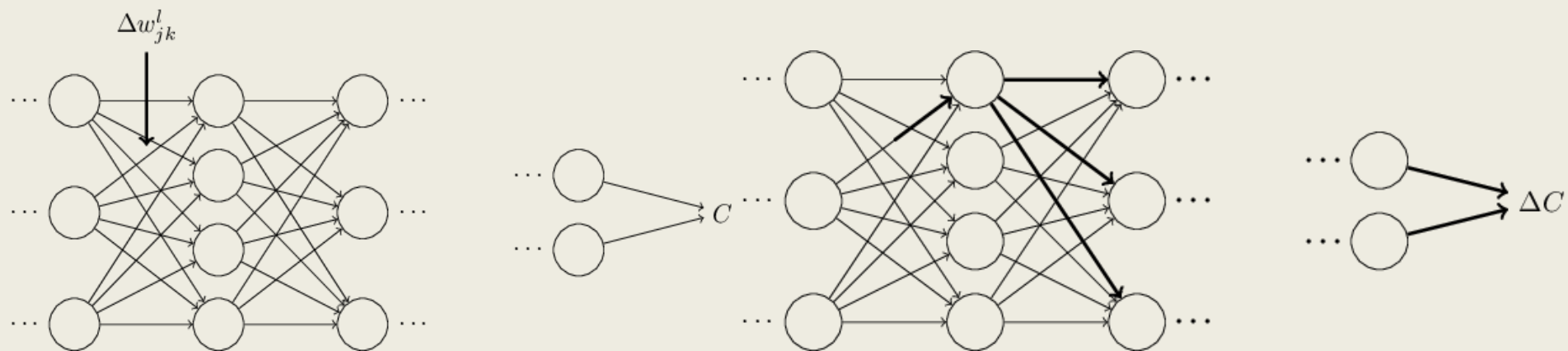
3. Gradient descent: For each $l = L, L - 1, \dots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Backpropagation: More intuitions

- Every edge between two neurons in the network is associated with a **rate factor**, which is just **the partial derivative of one neuron's activation with respect to the other neuron's activation**.
- The rate factor for a forward path in the network is just the product of the rate factors along the entire path.
- The total rate of change of C with respect to a weight in the network, is just the sum of the rate factors of all paths from the initial weight to the final cost.
- The backpropagation algorithm provides a way of computing this sum over the rate factor for all these paths.



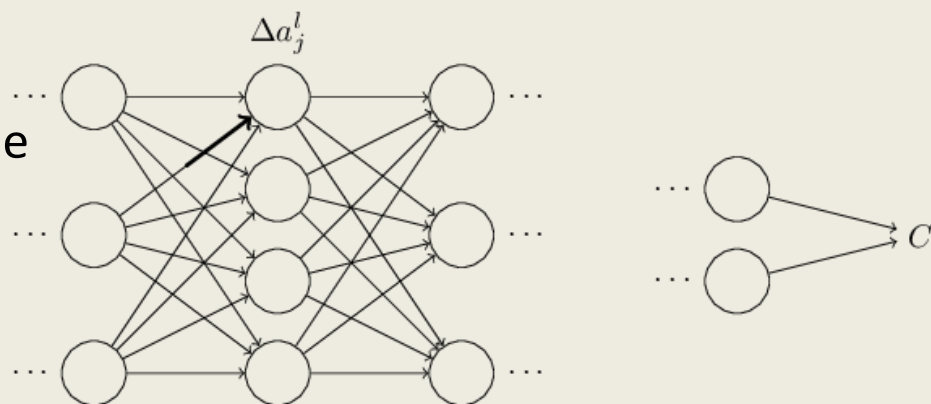
Backpropagation: More intuitions



Given the change, overall: $\Delta C \approx \frac{\partial C}{\partial w_{jk}^l} \Delta w_{jk}^l$

First, the change causes a small change in the activation of the j th neuron in the l th layer:

$$\Delta a_j^l \approx \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

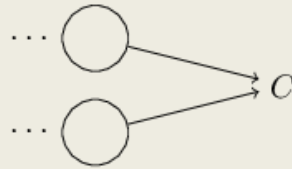
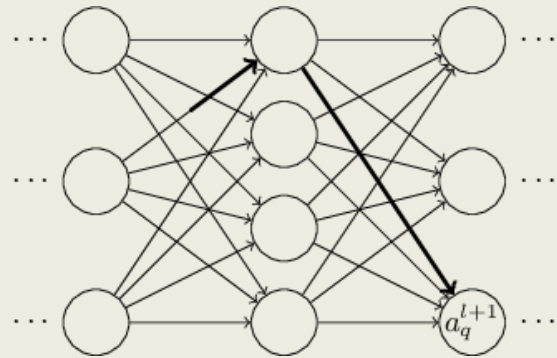


This change in turn, will cause changes in *all* the activations in the next layer. Let's concentrate on one of them...



$$\Delta a_j^l \approx \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Backpropagation: More intuitions



Imagining a path
all the way to C:

$$\Delta C \approx \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$

Since the change propagates across all possible paths to affect C:

$$\Delta C \approx \sum_{mnp \dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l$$



Backpropagation: More intuitions

$$\Delta C \approx \sum_{mnp\dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l} \Delta w_{jk}^l \quad \Delta C \approx \frac{\partial C}{\partial w_{jk}^l} \Delta w_{jk}^l$$

By combining the two equations above, it follows that:

$$\frac{\partial C}{\partial w_{jk}^l} = \sum_{mnp\dots q} \frac{\partial C}{\partial a_m^L} \frac{\partial a_m^L}{\partial a_n^{L-1}} \frac{\partial a_n^{L-1}}{\partial a_p^{L-2}} \cdots \frac{\partial a_q^{l+1}}{\partial a_j^l} \frac{\partial a_j^l}{\partial w_{jk}^l}$$

- We are computing the rate of change of C with respect to a weight in the network.
- This is just the sum of the rate factors of all paths from that weight to the final cost.
- The backpropagation algorithm computes the sum over the rate factor for all these paths.

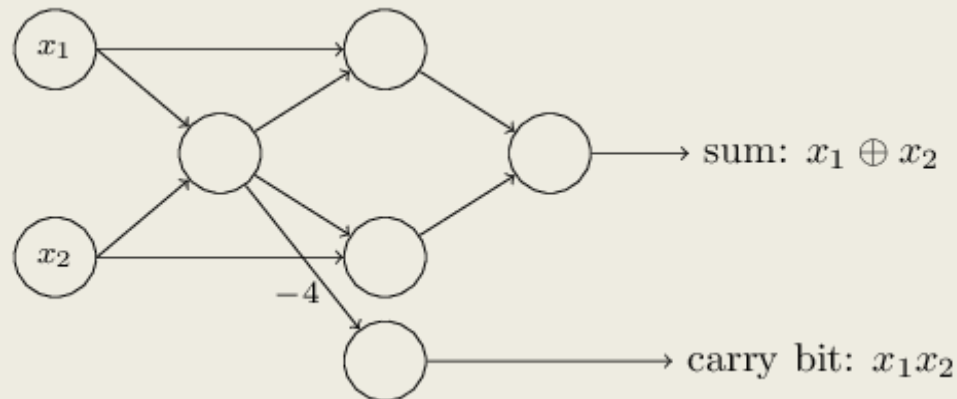
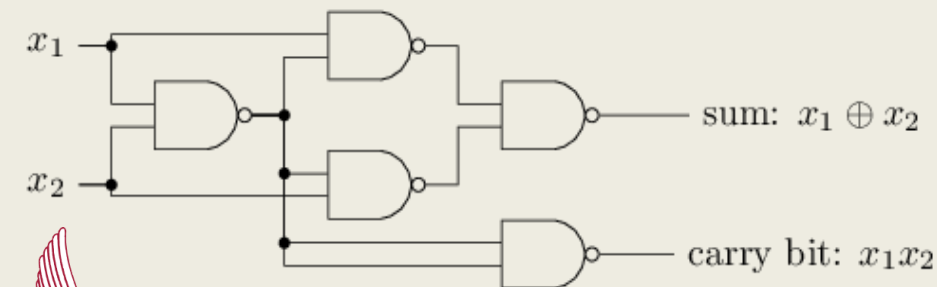


Deep Considerations ...



The Universal Approximation Theorem

- “A feed-forward network with a single hidden layer containing a finite number of neurons, can approximate continuous functions on compact subsets of R^n , under mild assumptions on the activation function.”
 - shown, e.g., for sigmoid functions (Cybenko, 1989)
- **Intuition:** a network of perceptrons can simulate a circuit containing NAND gates. Since any Boolean function can be implemented using NAND gates, it follows that perceptrons are universal computers!



other weights: -2; biases= 3



The Universal Approximation Theorem

- **Kolmogorov Theorem**

- *A neural network with at least 3 layers of sufficiently large number of neurons is able to learn any mathematical function!*

- **Thus, in theory**

- Simple networks can represent essentially any function!
- All we need is networks of perceptrons, likely MLPs.
- But, these may require a huge number of neurons!
- Also, no recipe on how exactly to structure the network.
- Also, no recipe on how to actually do the learning/computations.
- In a sense, we probably just confirm that a network of perceptrons is just a new type of NAND gate!



- **Practice+new theory shows that deep networks are good**

Why Deep NN?

- First ... what does “deep” mean?
- Well, **deep** means **more than 3** (input+hidden+output) layers
- So, strictly speaking, it is not “a new thing”!
- Why “**3**”? With depth 3 we have a universal approximator.
- Recent theoretical results show the power of “deep”.
- Deep architectures can represent a function with exponentially fewer units than a NN with 3 layers!



Deep Learning and Deep Neural Networks

■ Why now (i.e., after 2010)?

- more powerful CPUs + appearance of GPUs + big data
- high-quality software (tensorflow, PyTorch, MXNet, ...)
- new theoretical results that suggest their power

• A Spiral Effect

- Deep NNs do well →
- more and more people now get interested →
- people get to learn how to tweak Deep NNs →
- Deep NNs get even better in more domains →
- [repeat]



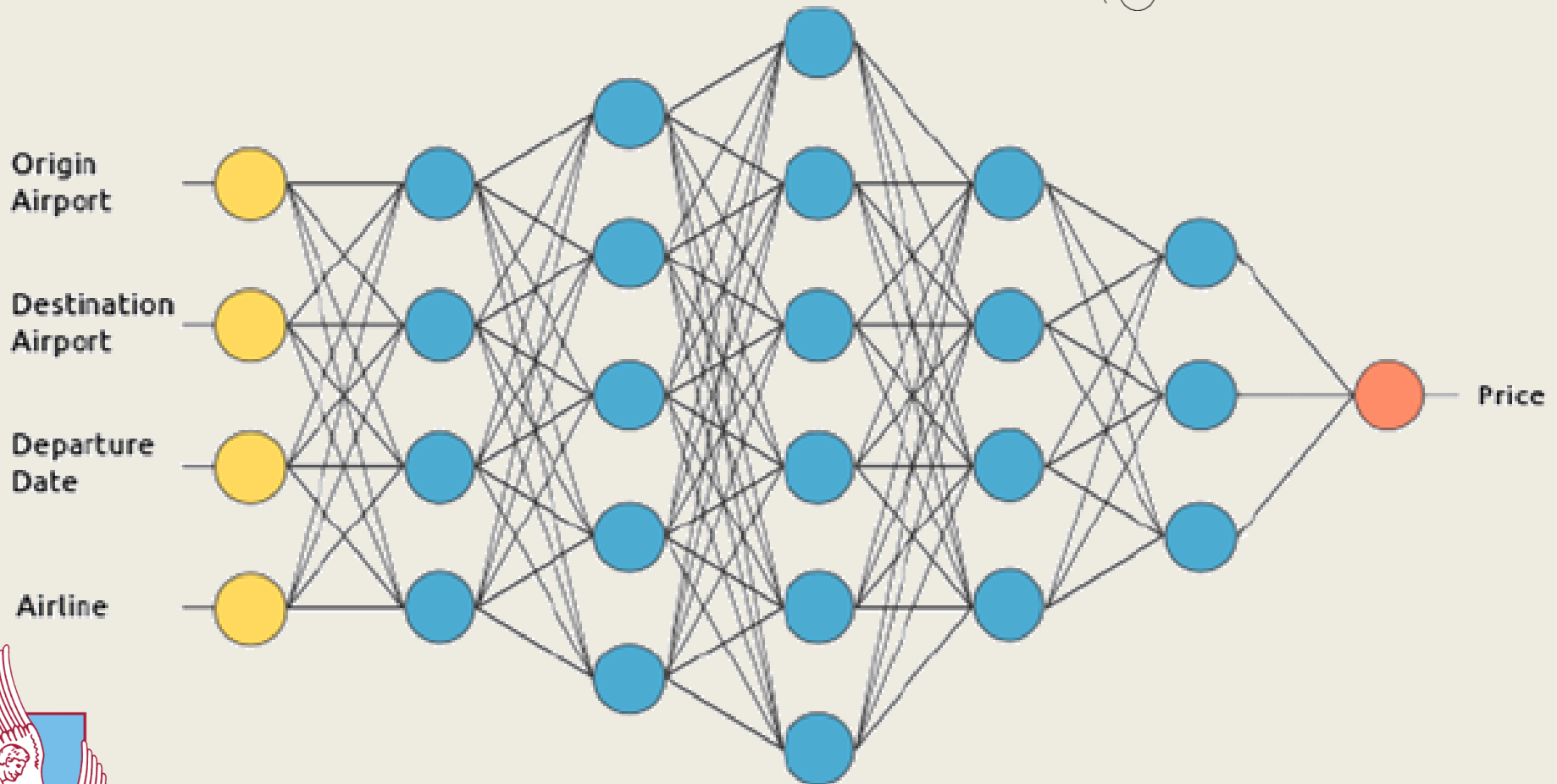
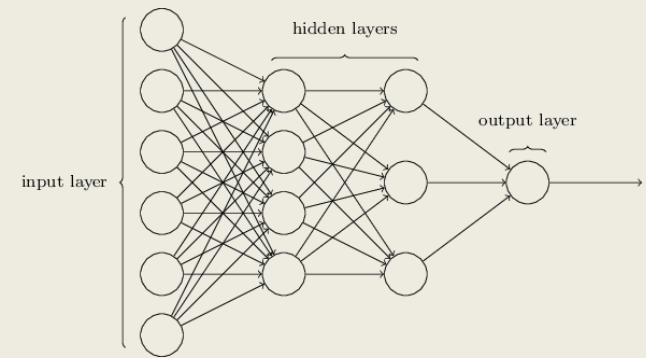
Deep Neural Networks in a Nutshell

input layer + a number of hidden layers + output layer

- Each unit's input: a **weighted sum of outputs** of units from previous layers (weights are on links between layers)
- Units' outputs: **nonlinear transformation** of weighted inputs by **activation functions** (tanh, logistic, rectified linear unit, ...)
- Compute **error derivatives** and **backpropagate gradients** from the output layer towards the input layer.
- **Update weights** from gradients to optimize loss function.
- **Repeat** until convergence.

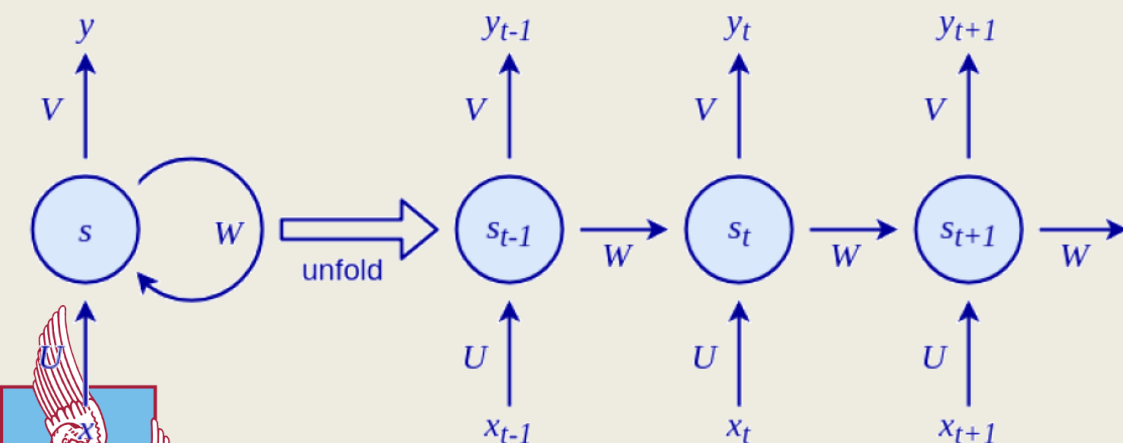


Deep Feedforward NNs



Recurrent Neural Networks (RNNs)

- In feed-forward NNs we assume that all inputs (and outputs) are independent of each other → Could be a bad idea
- RNNs have at least one feedback node $s_t = f(s_{t-1}, x_t)$
 - Can process sequential data with variable length
 - Consider time dependencies via the recurrence relation
- They might exhibit complicated dynamics and is difficult to train
- They act similarly to the biological model
- Increasing interest on RNN architectures

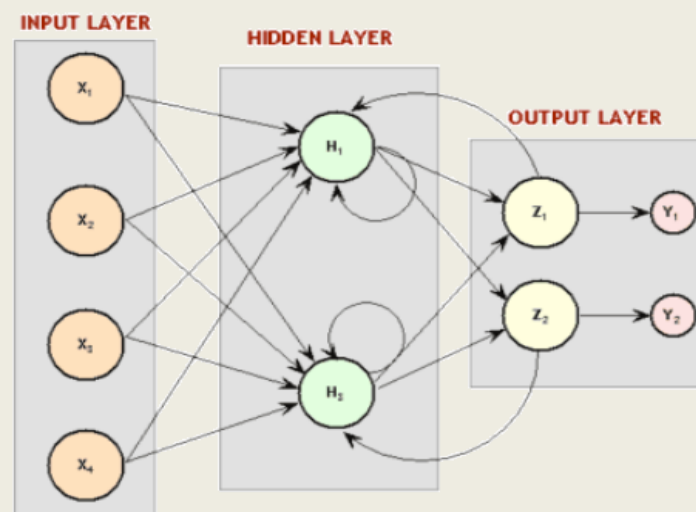


Recurrence relation:

$$s_t = f(W \cdot s_{t-1} + U \cdot x_t)$$

Final Output:

$$y_t = V \cdot s_t$$

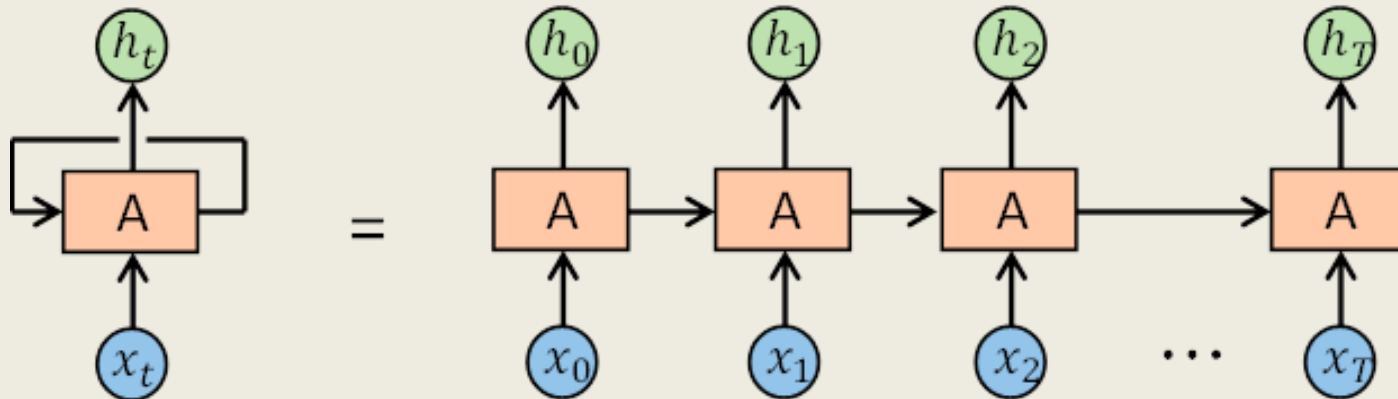


$$\{\widehat{TD}_{1(t+1)}, \widehat{TD}_{2(t+1)}, \dots, \widehat{TD}_{N(t+1)}\} \quad \{\widehat{TD}_{1(T+1)}, \widehat{TD}_{2(T+1)}, \dots, \widehat{TD}_{N(T+1)}\}$$

Prediction for t+1

Prediction for T+1

Recurrent NNs



- **Idea:** capture sequential dependencies among inputs, add “memory” for calculations
- layer A (at t) looks at input x_t , outputs h_t ; loop allows sequence info to propagate
- Loop $\equiv$ multiple copies of the same network
- At each t , a hidden state C_t (memory) is calculated given x_t and the previous state C_{t-1}
- The weights are *the same at all steps* – since same task at each step, different inputs
- **Problem:** “vanishing gradient”, *same as in feed-forward networks*

Example

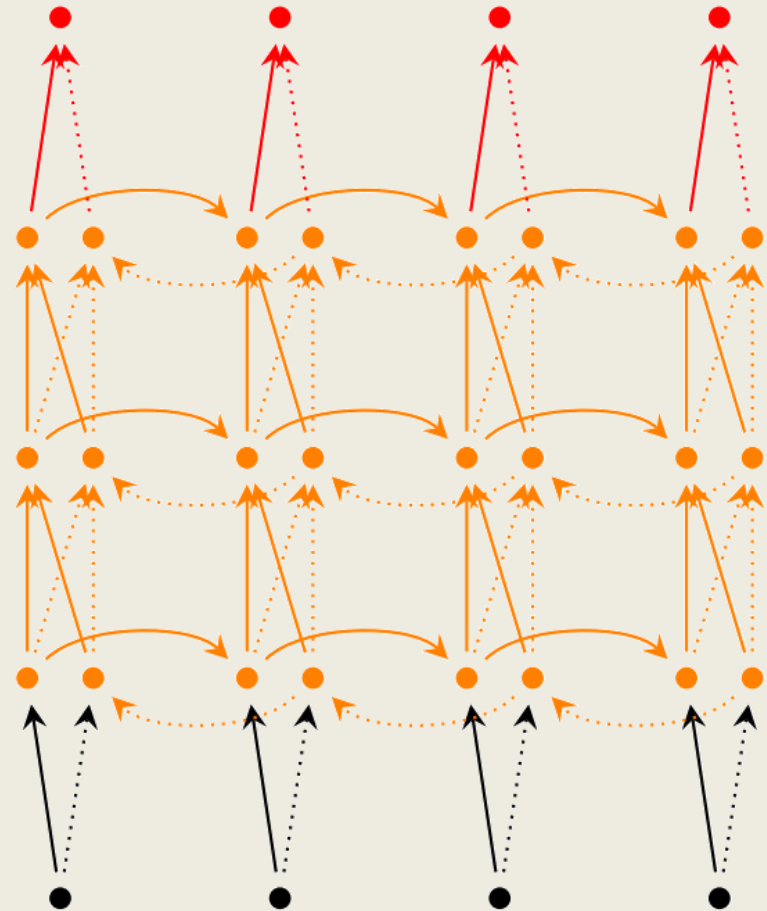
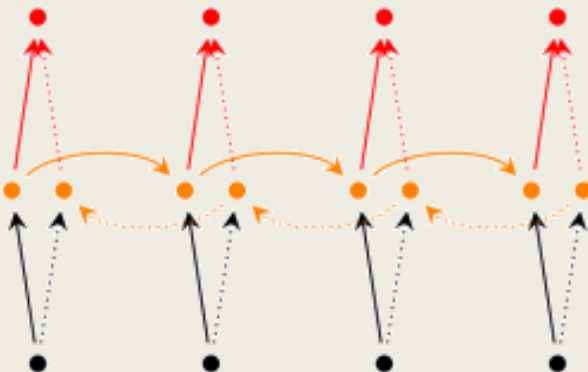
- Trip demand prediction problem
- x_t = at t : trip demands at N zones, hour, day of week, precipitation, temperature

$$\{TD_{1t}, TD_{2t}, \dots, TD_{Nt}, H_t, DW_t, P_t, T_t\}$$

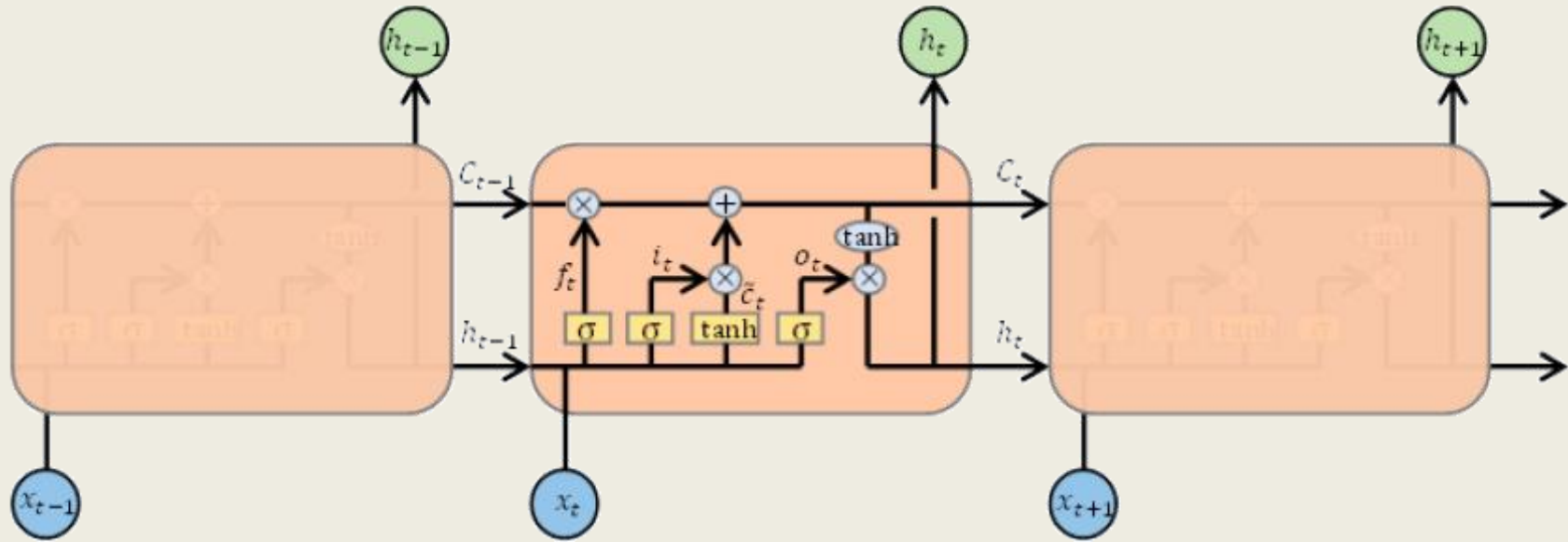
RNNs made deeper

- RNNs are already deep, but perhaps not deep enough!
- Deep bidirectional RNNs →

Bidirectional RNN
(captures dependencies on future elements)



Long Short-Term Memories (LSTMs)



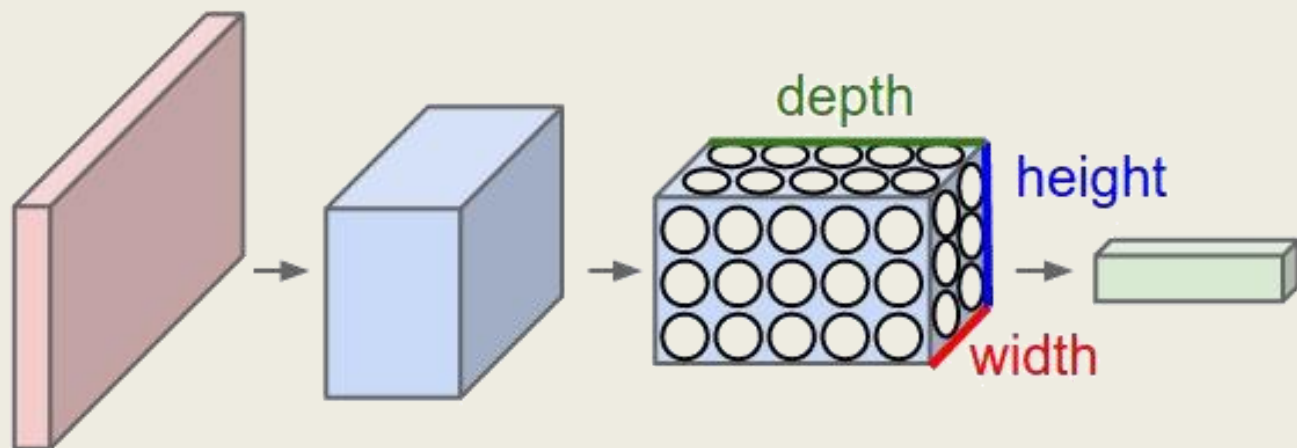
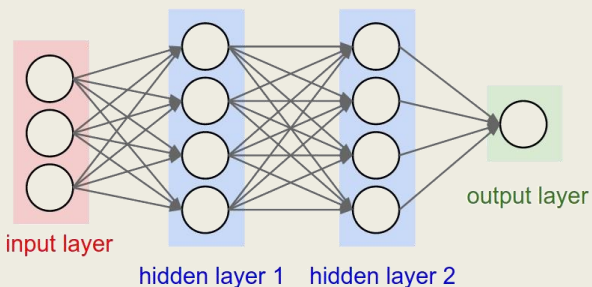
- **Idea:** capture sequential dependencies among inputs, add “memory” for calculations, and **overcome the vanishing gradient problem (their short-term memory can last long) ...how?**
- **Four layers in each repeating module**
 - A logistic “forget gate layer” decides the extent to which a value in C_{t-1} remains in C_t
 - A logistic “input gate layer” decides the extent to which input values will be updated
 - A $\tanh$ layer outputs an array of new candidate values to add to the forget layer output
 - A logistic “output gate layer” decides the extent to which values are used in the output



Excellent results for NLP and speech recognition tasks!

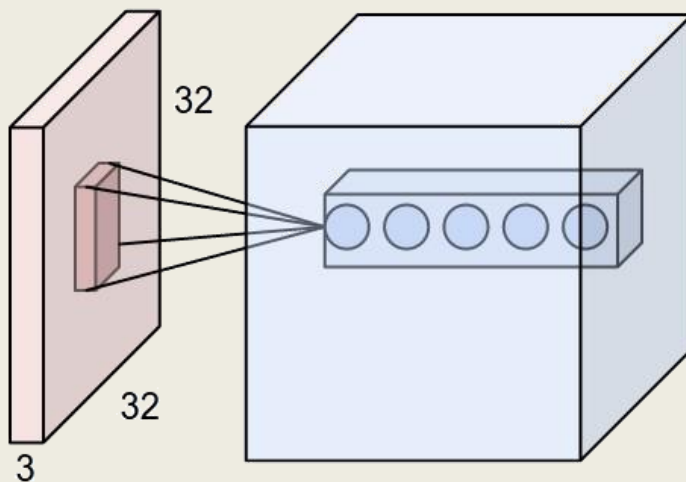
Convolutional NNs

A 3D neural architecture!



Biologically-inspired:

- resemble the architecture of the visual cortex
- they use "receptive fields"

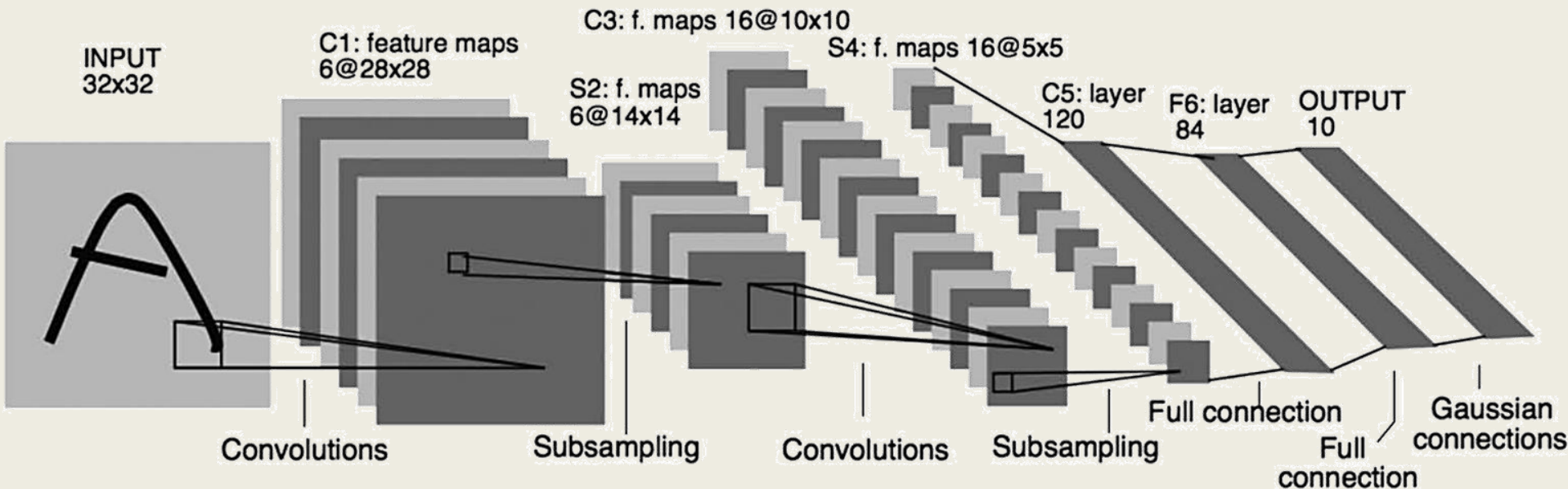


One key common technique used:
depth-slicing+parameter sharing

Obviously good for image/video recognition,
but also for NLP and recommender systems!



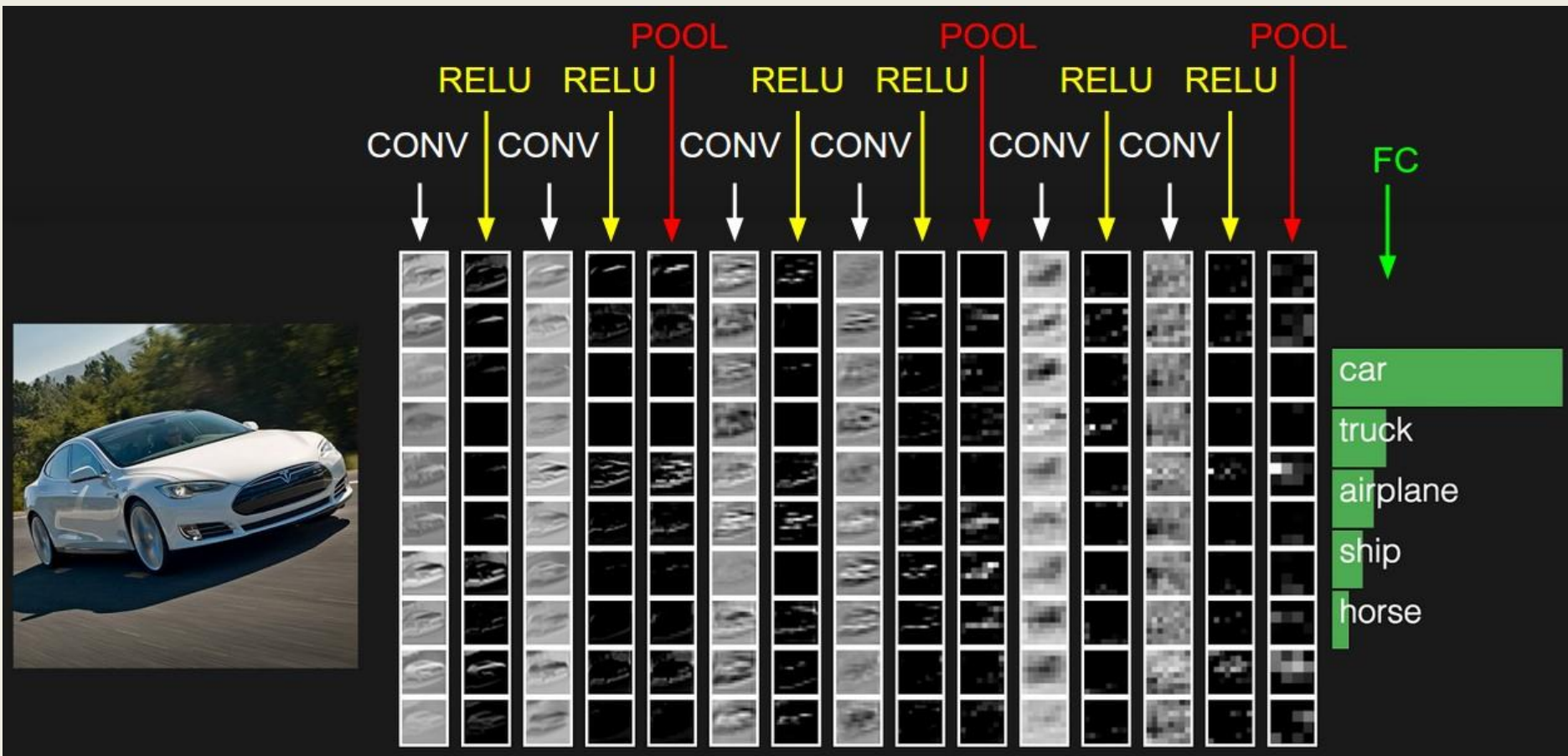
CNNs: an example Conv-Net architecture



<http://incompleteideas.net/book/the-book-2nd.html>

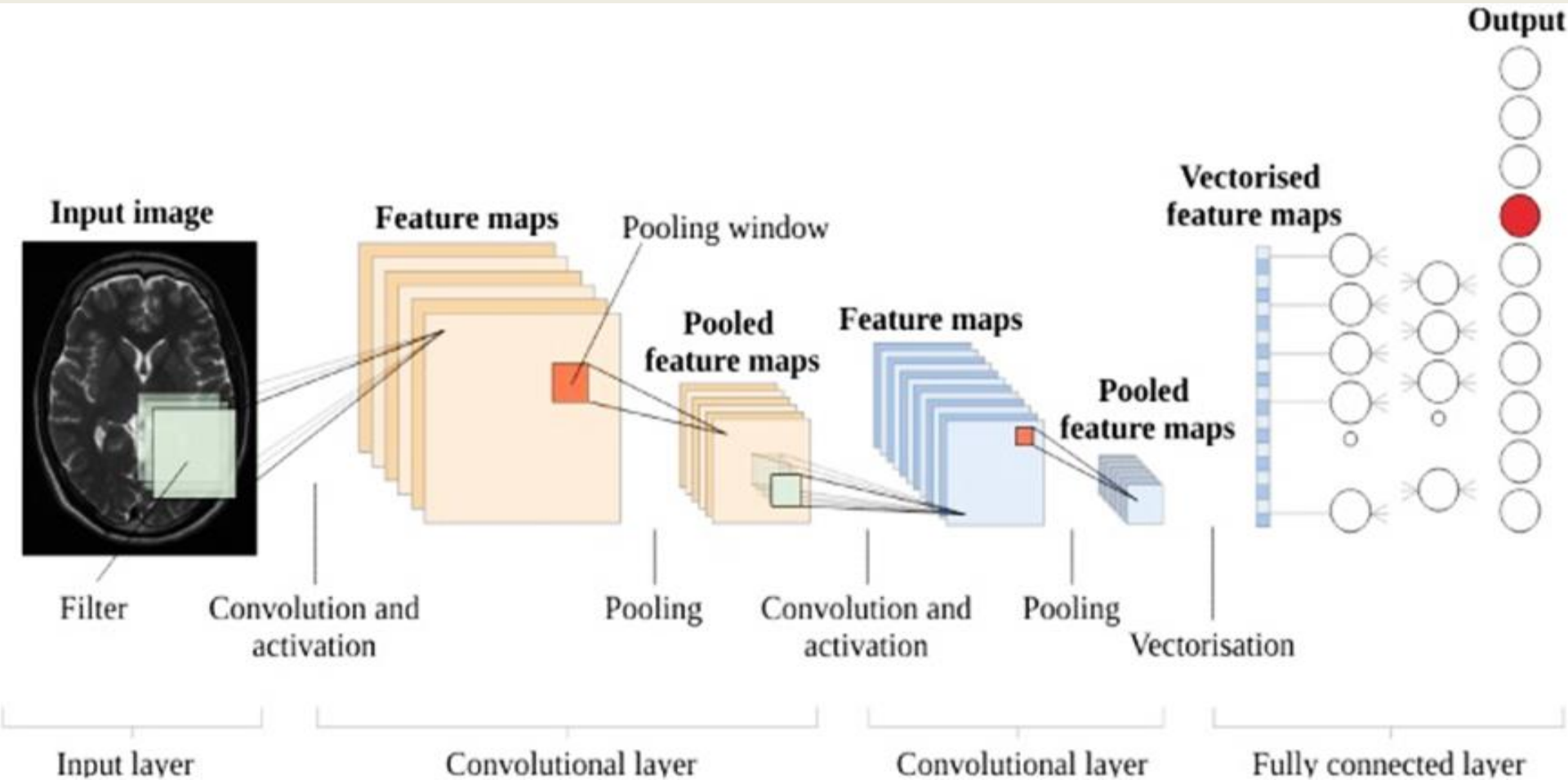


CNN use in yet another Conv-Net architecture

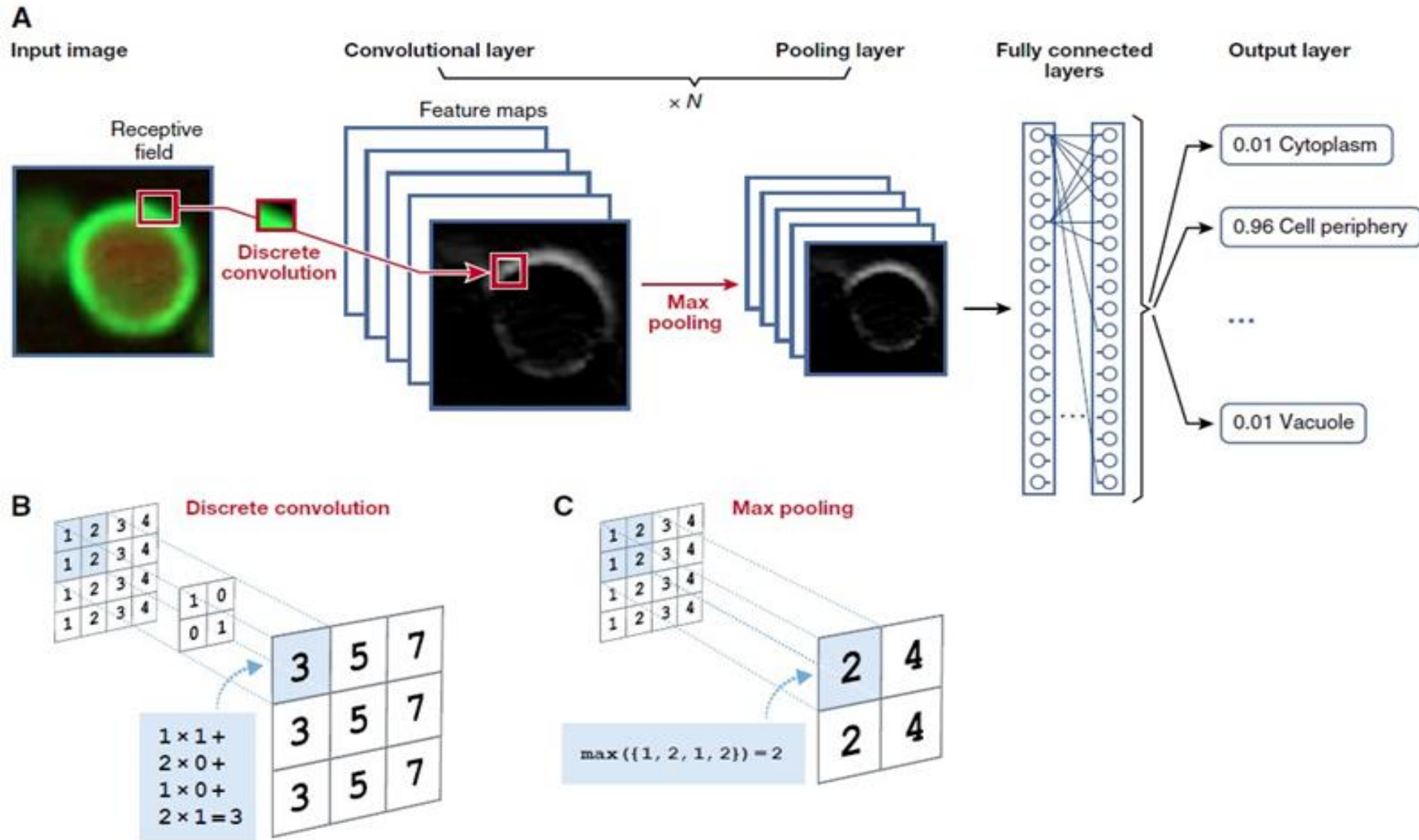


<http://cs231n.github.io/convolutional-networks/>

Typical CNN



Typical CNN Diagram



CNNs for Medical Imaging

- **BrainNet**: developed based on TensorFlow, aims to train deep neural networks to segment GM (Gray Matter) and WM (White Matter) from brain MR images
- **LiviaNet**: developed based on Theano, aims to train 3-D fully convolutional neural networks by using MR images to segment sub-cortical brain structures
- **DIGITS**: developed to rapidly train accurate deep neural networks for image segmentation, classification, and tissue detection tasks (e.g. Alzheimer's disease detection)
- **DeepMedic**: developed based on Theano, aims to train multi-scale 3-D convolutional neural networks for brain lesion segmentation from MR images (winner of the ISLES 2015 competition)



Main References

- Michael Nielsen (2015): **Neural Networks and Deep Learning** *Free online book.* neuralnetworksanddeeplearning.com
- Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016): **Deep Learning** *Free online book.* www.deeplearningbook.org

A few other good reads

- Delalleau, O. and Bengio, Y. (2011). *Shallow vs. deep sum-product networks*. In Neural Information Processing Systems (NeurIPS)
- Hinton, G., Deng, L., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012). *Deep neural networks for acoustic modeling in speech recognition*. IEEE Signal Processing Magazine, 29 (6), 82–97
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). *Maxout networks*. in Intl Conf on ML (ICML)

