TDT4171 Artificial Intelligence Methods

Lecture 12 – Summary

Norwegian University of Science and Technology

Helge Langseth
Gamle Fysikk 255
helge.langseth@ntnu.no

NTNU

TDT4171

Goals of the course:

"[...] The three main ways of reasoning (rule-based, model-based, and case-based), will be discussed, with most focus given to model-based reasoning. In particular, we work with reasoning with uncertain and/or partly missing information, as well as the basis for learning systems (machine learning)."

Syllabus (Tentative, as of Jan 17, 2025):

- The text-book *"Artificial Intelligence – A Modern Approach"* by S. Russell and P. Norvig, **4th ed.**, 2020. Approx. 300p.
- A. Aamodt and E. Plaza (1994): *"Case-based reasoning; Foundational issues, methodological approaches"*. 20p.
- D. Ganesan and S. Chakraborti (201 *Knowledge Tradeoffs in Case-Based*
- Also: *"Deep Learning"* by I. Goodfell 2016. Selected parts as supporting n

Added 2 sections on DL & NLP.

Took the book "Deep Learning" out again.

Assignments

- There will be **ten** assignments, to be solved **individually**.
- The assignments will contain both coding tasks and theory.
- Typically with a one-week deadline.
  - Assignments posted on BB Friday mornings.
  - Time of delivery is **Thursdays at 23:59**, unless **stated otherwise**.
  - We **plan** to have assignments every week; check BB.
- **Assignment hrs:** Will be announced on Blackboard. No assistants around before first assignment is out.

Requirement for exam

To be eligible for exam, you will need to have at least 7 of the 10 assignments accepted.

Examination

Confirmed 8/5 at 0900.

- Written, **4 hours**.
- Exam is available in **English**
- No written or handwritten examination support materials are permitted. A specified, "simple calculator" is permitted.
- We will use **INSPERA.**
- **Letter grades.**

Summary

- **Foundation:**
  - **Weak AI**: Can machines **act** intelligently?
  - **Strong AI**: Can machines really **think**?
  - Alan Turing proposed the **Turing Test** to verify intelligence
  - Arguments for and against Strong/Weak AI are **inconclusive**, but few believe this debate will have significant impact on AI
  - There are **ethical concerns** in AI, in particular related to **ultra-intelligent** systems

It is very common to include questions from the "soft" part in the examination set. Sometimes "What do you think and why"- questions, too. If so: Don't be shy! These are relevant also this year

Starting point: Uncertainty and probability calculus is a must

: Uncertainty

n message of the
pter: *"Uncertainty is everywhere"*

- Treatment here differs from the basic statistics course
  - We consider **high-dimensional** distributions
  - **More** focus on efficient representation & inference
  - **More** focus on modelling
  - **More** focus on making decisions
  - **Less** focus on statistical method (hypothesis tests, confidence intervals, etc.)

Probability

Probabilistic assertions **summarize** effects of

Laziness: Failure to enumerate exceptions, qualifications, etc.

Ignorance: Lack of relevant facts, initial conditions, etc.

Subjective or *Bayesian* probability:

- Probabilities relate propositions to one's own state of knowledge, e.g., $P(A_{15}|\text{no reported accidents}) = 0.06$
- These are **not** claims of a "probabilistic tendency" in the current situation (but might be learned from past experience of similar situations)
- Probabilities of propositions change with new evidence, e.g., $P(A_{15}|\text{no reported accidents, 5 a.m.}) = 0.15$

Summary

- Probability is a rigorous formalism for uncertain knowledge
- Joint probability distribution specifies probability of every atomic event
- Queries can be answered by summing over atomic events
- For nontrivial domains, we must find a way to reduce the joint size
- Independence and conditional independence provide the tools

Example – The Burglary Domain

I'm at work, neighbor John has called (to say my alarm is ringing), but neighbor Mary hasn't called. Sometimes it's set off by minor earthquakes. Query: Is there a burglar?

Variables:

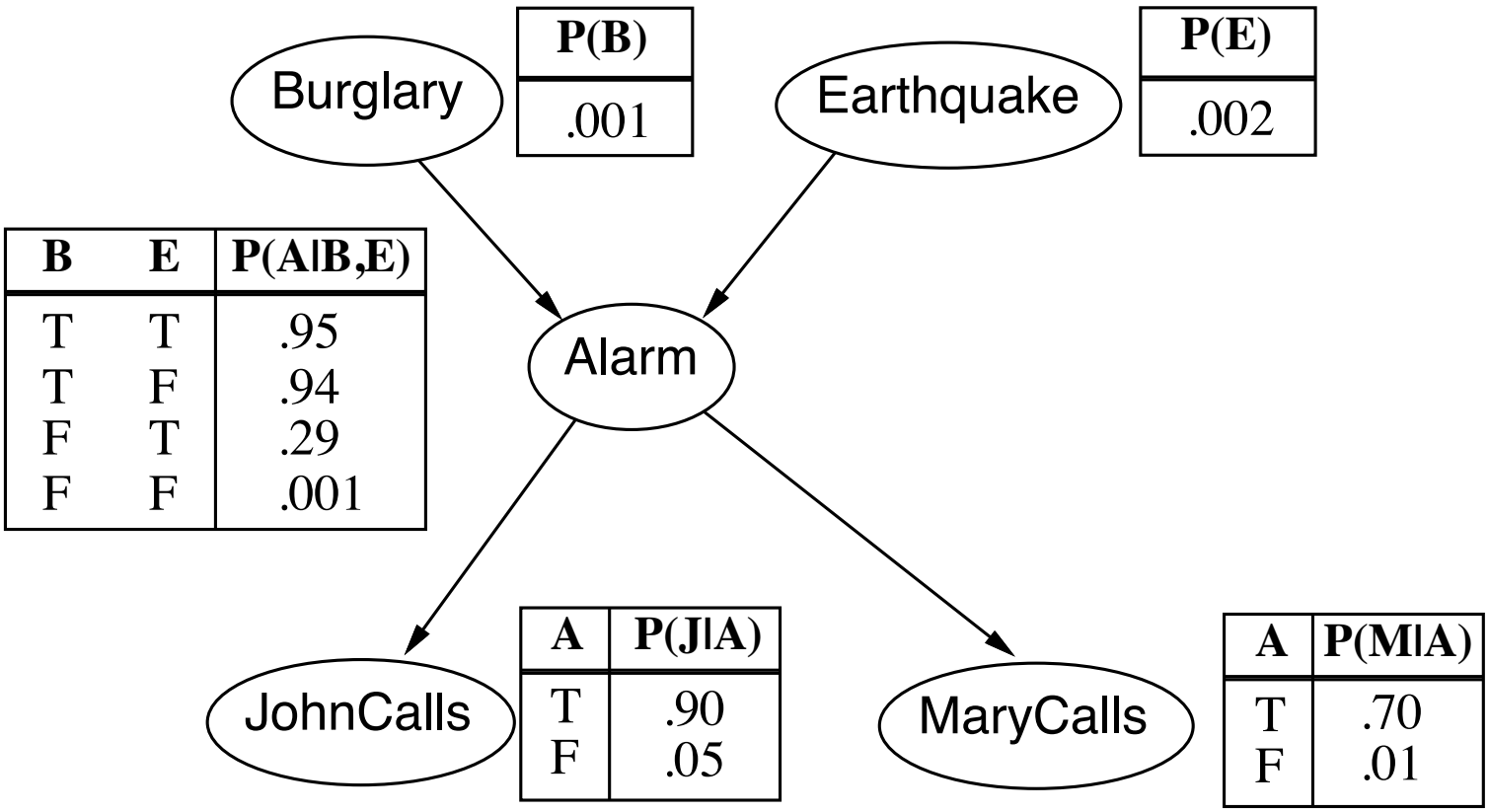
- `Burglar`, `Earthquake`
- `Alarm`
- `JohnCalls`, `MaryCalls`

Network topology reflects "causal" knowledge:

- A burglar can set the alarm off
- An earthquake can set the alarm off
- The alarm can cause Mary to call
- The alarm can cause John to call

VERY typical examination question: Model whatever using a BN. Sometimes only graph, sometimes also the probability tables.
You should really learn to do this!

Example contd.

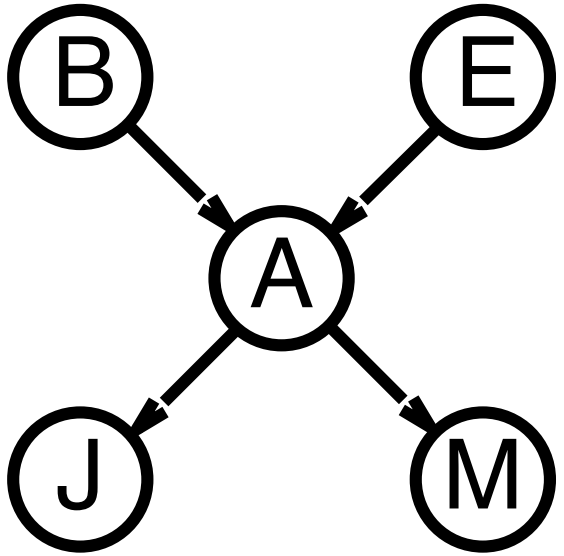


Compactness

A CPT for Boolean X_i with k Boolean parents has 2^k rows for the combinations of parent values

Each row requires one number p for $X_i = \texttt{true}$ (the number for $X_i = \texttt{false}$ is just $1 - p$)

If each variable has no more than k parents, the complete network requires $O(n \cdot 2^k)$ numbers



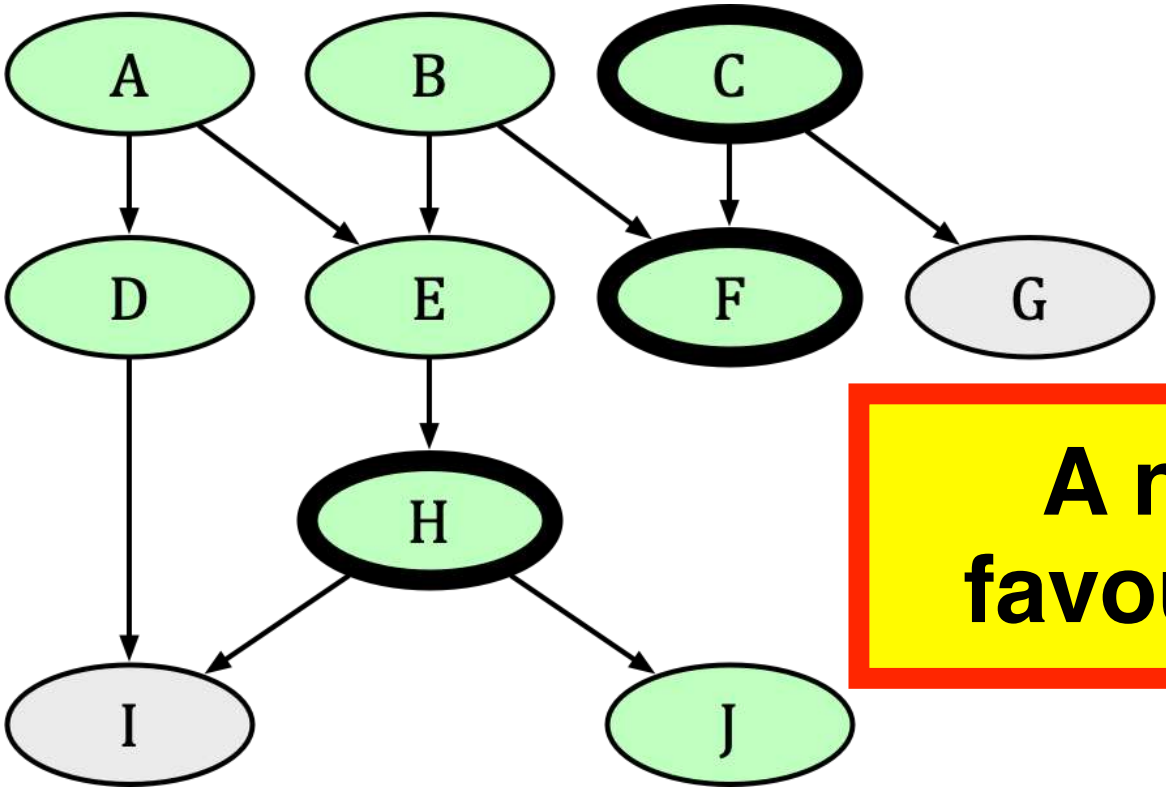
I.e., grows linearly with n, vs. $O(2^n)$ for the full joint distribution! For burglary net, $1 + 1 + 4 + 2 + 2 = 10$ numbers (vs. $2^5 - 1 = 31$)

Summary

- **Bayes nets** provide a natural representation for (causally induced) conditional independence
- Topology + CPTs = **compact representation** of joint distribution
- Generally **easy to construct** – also for non-experts
- **Canonical distributions** (e.g., noisy-OR) = compact representation of CPTs
- **Efficient inference** calculations are available (but the good ones are outside the scope of this course)

Typical exam questions:
* Build a model for some domain, assess the usefulness of the model.
* Assess how appropriate this tool is for a given task.
"Usefulness-things" also relevant later (e.g., CBR or deep learning?)

A slightly more involved example



A new favourite!

Is $I \perp\!\!\!\perp G$?	Yes!	Is $I \perp\!\!\!\perp G \mid H$?	Yes!
Is $I \perp\!\!\!\perp G \mid F$?	No!	Is $I \perp\!\!\!\perp G \mid \{H, F\}$?	No!
Is $I \perp\!\!\!\perp G \mid \{E, F\}$?	No!	Is $I \perp\!\!\!\perp G \mid \{H, F, C\}$?	Yes!

Time and uncertainty

Motivation: The world changes; we need to track and predict it
Static (Vehicle diagnosis) vs. **Dynamic** (Diabetes management)

Basic idea: copy state and evidence variables for each time step

$\texttt{Rain}_t$ = Does it rain at time t

This assumes **discrete time**; step size depends on problem
Here: Timesteps = Days(?)

Good to know how we build these models!

Markov processes (Markov chains)

If we want to construct a Bayes net from these variables, then what are the parents?

- Assume we have observations of $\texttt{Rain}_0, \texttt{Rain}_1, \ldots, \texttt{Rain}_t$ and want to **predict** whether or not it rains at day $t+1$: $\mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_0, \texttt{Rain}_1, \ldots, \texttt{Rain}_t)$
- Try to build a BN over $\texttt{Rain}_0, \texttt{Rain}_1, \ldots, \texttt{Rain}_{t+1}$:
  - $\mathbf{P}(\texttt{Rain}_{t+1}) \neq \mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_t)$; base on $\texttt{Rain}_t$.
  - $\mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_t) \approx \mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_t, \texttt{Rain}_{t-1})$ **(Do you agree?)**

k'th-order Markov process:

$$\mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_0, \ldots, \texttt{Rain}_t) = \mathbf{P}(\texttt{Rain}_{t+1} | \texttt{Rain}_{t-k+1}, \ldots, \texttt{Rain}_t)$$

Markov processes; full set of assumptions

Stationary process:

Transition model $\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-1})$ and sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$ **fixed** for all t

k'th-order Markov process:

$$\mathbf{P}(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = \mathbf{P}(\mathbf{X}_t|\mathbf{X}_{t-k:t-1})$$

Sensor Markov assumption:

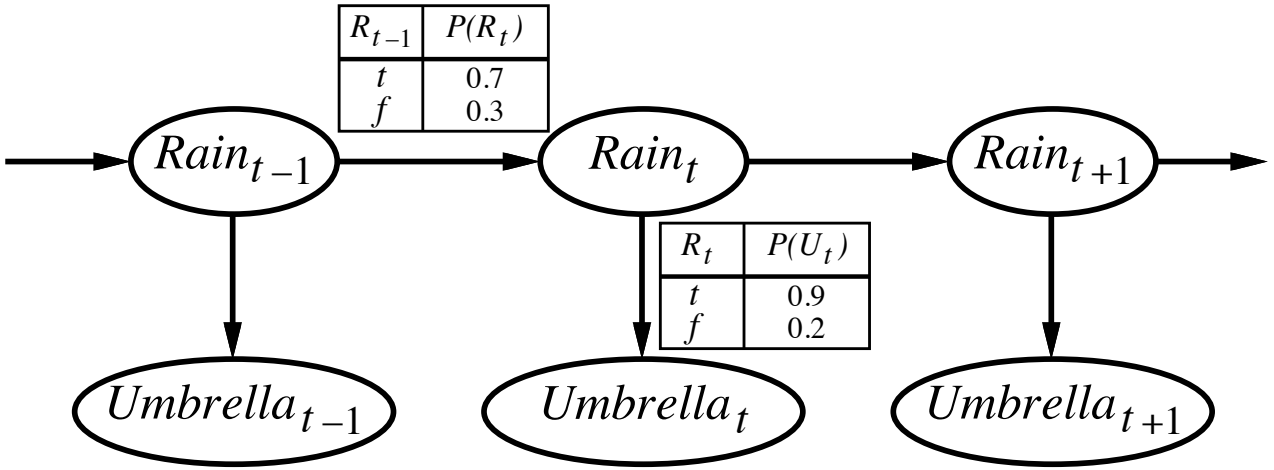
$$\mathbf{P}(\mathbf{E}_t|\mathbf{X}_{0:t}, \mathbf{E}_{0:t-1}) = \mathbf{P}(\mathbf{E}_t|\mathbf{X}_t)$$

Rather than asking you to memorize equations, my goal has typically been to check the understanding.

E.g., "What are the ASSUMPTIONS we make for this model class, and how do they help simplify the task?"

Some years, students have been asked to do detailed calculations in these models.

Example



Good to recognize this model, and understand why it looks this way —> Have a feel for the assumptions again

Inference tasks

Filtering: $\mathbf{P}(\mathbf{X}_t|\mathbf{e}_{1:t})$. This is the **belief state** – input to the decision process of a rational agent

Prediction: $\mathbf{P}(\mathbf{X}_{t+k}|\mathbf{e}_{1:t})$ for $k > 0$. Evaluation of possible action sequences; like filtering without the evidence

Smoothing: $\mathbf{P}(\mathbf{X}_k|\mathbf{e}_{1:t})$ for $0 \leq k < t$. Better estimate of *past* states – Essential for learning

Most likely explanation: $\arg\max_{\mathbf{x}_{1:t}} P(\mathbf{x}_{1:t}|\mathbf{e}_{1:t})$. Speech recognition, decoding with a noisy channel

Typical questions from this chapter:
- What are the assumptions
- WHY do we need these assumptions
- What will the model look like
- What inferences can be made

Rationality: Anything with decision-making

Learning goals

Understanding the relationship between

1. Rational behaviour – *"doing what is expected to maximize goal achievement, given the available information"*
2. Preference structures
3. Utilities

Being familiar with:

- Utility functions – Their foundation and definition
- Utility elicitation
- Influence diagrams

Maximizing expected utility

Theorem – The foundation for the *Utility function*

Given preferences satisfying the constraints there exists a real-valued function U such that

- $U(A) \geq U(B) \quad \Leftrightarrow \quad A \succsim B$
- $U([p_1, S_1; \ \ldots \ ; \ p_n, S_n]) = \Sigma_i \ p_i U(S_i)$

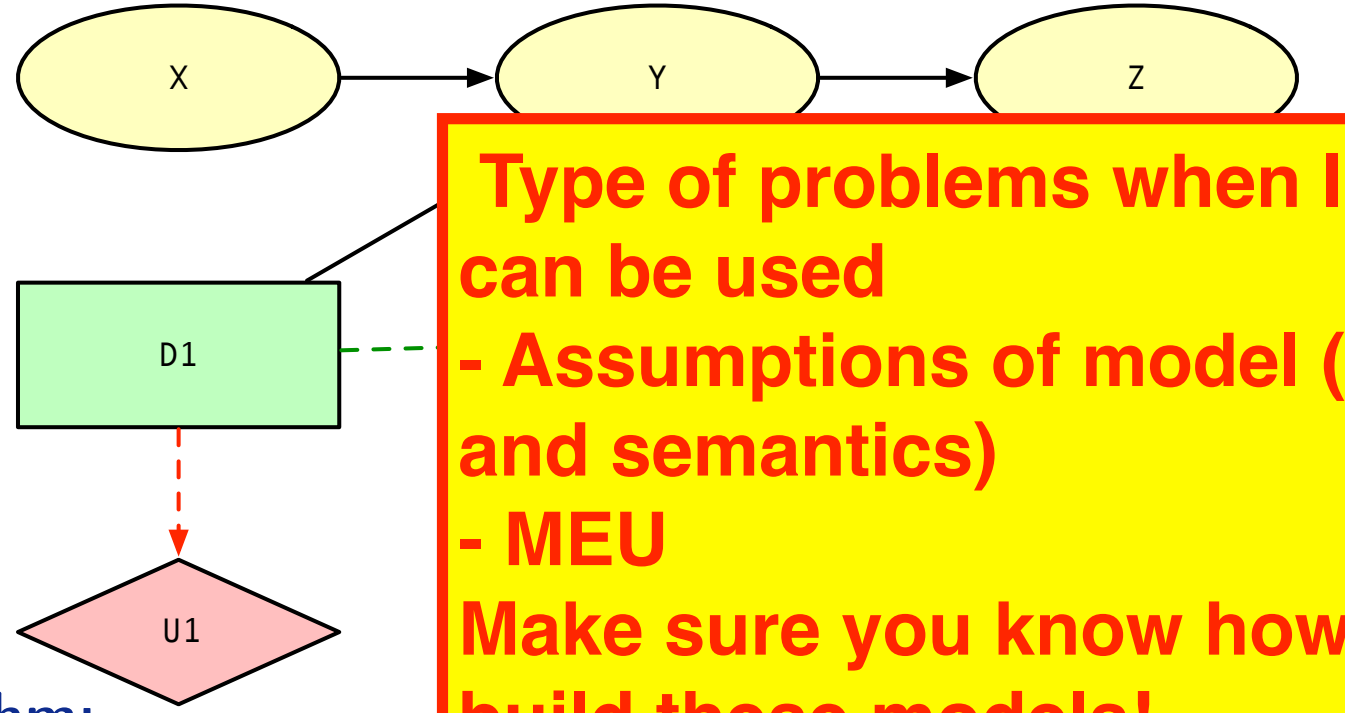
This gives rise to the MEU principle:
To be rational, the agent must choose the action that maximizes expected utility!

Note: an agent can be entirely rational (consistent with MEU) without ever representing or manipulating utilities and probabilities!
(For example, a lookup table for perfect tic-tac-toe)

You need to understand the MEU principle, and how it helps us design agents!

Decision networks (a.k.a. "influence diagrams")

Add **decision nodes** and **utility nodes** to belief networks to enable rational decision making.



Type of problems when IDs can be used
- Assumptions of model (syntax and semantics)
- MEU
Make sure you know how to build these models!

Algorithm:

- For each value of action node(s):
  - Compute expected value of sum of utility nodes given action(s), evidence
- Return MEU action(s)

Complex decisions: "Rationality + time"

Learning goals

Understanding the relationship between

- One-shot decisions
- Sequential decisions
  - Decisions for finite horizon
  - Decisions for infinite horizon

Being familiar with:

- Markov Decision processes
- Value iteration
- Policy iteration

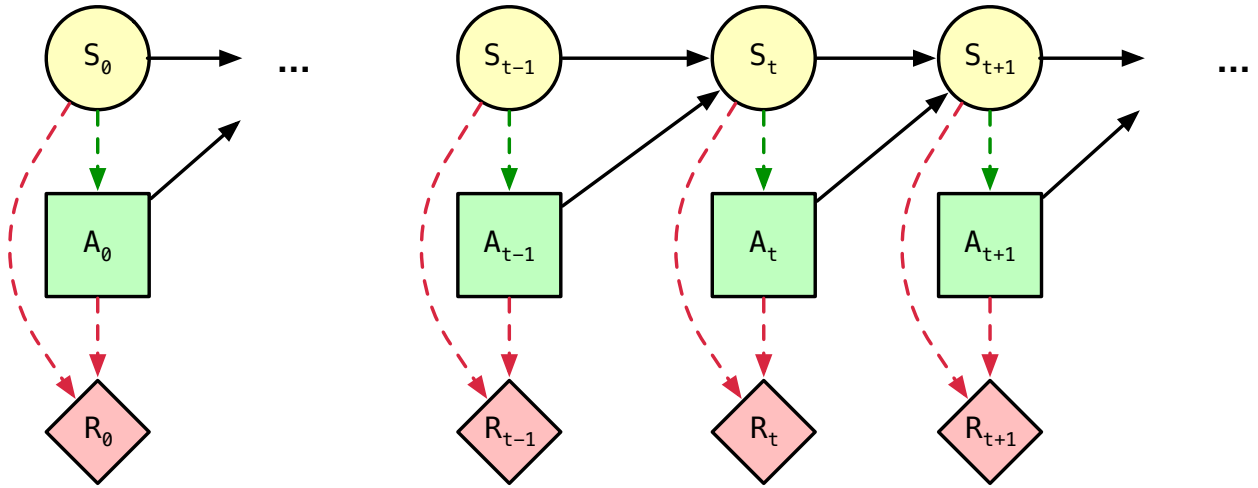
Know about:

- Partially Observable Markov Decision processes (not covered in slides, **but part of the curriculum**)

MDPs in general

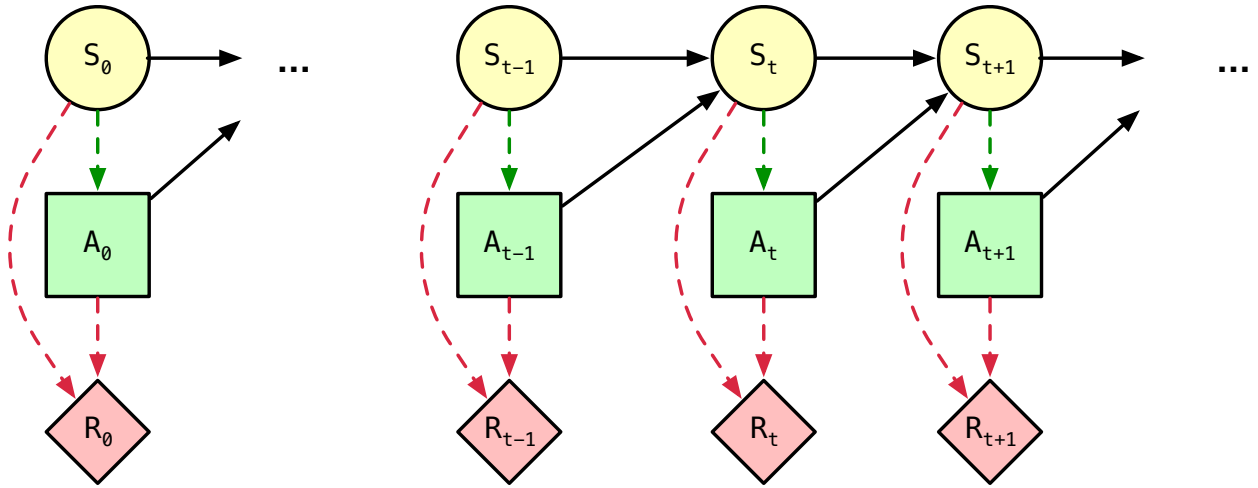
In general, in a Markov decision process . . .

- The world is **fully observable**,
- Any uncertainty in the system is due to **non-deterministic actions**
- For each decision we get a **reward** (which may be negative); may depend on current world state and chosen action, but is independent of time (stationarity of reward-model).



Decision policies

A decision policy for A_t is in general a function over the entire past, $\{S_0, A_0, S_1, A_1, S_2, A_2, \ldots, S_t\}$.

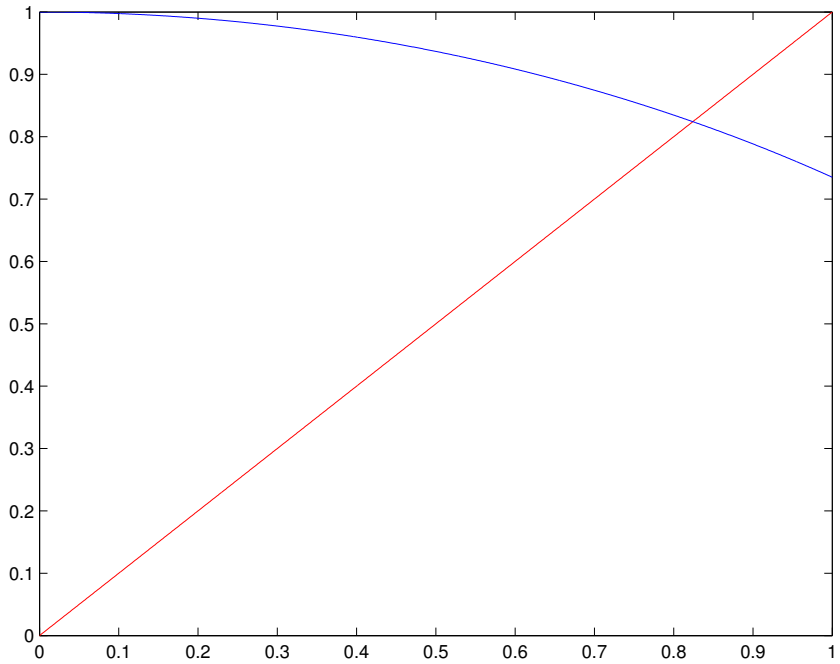


However, from the conditional independence properties we see that the **relevant** past is reduced to S_t.

Note! While we do not consider S_{t-1} when choosing A_t, we **do** think about the future (S_{t+1}, A_{t+1}, S_{t+2}, A_{t+2}, ...).
This is similar to **prediction** in the dynamic models.

A side-step: Fix-point iterations

→ Rather solve $x = \sqrt{\cos(x)}$ (same solution for $x \in [0, 1]$).

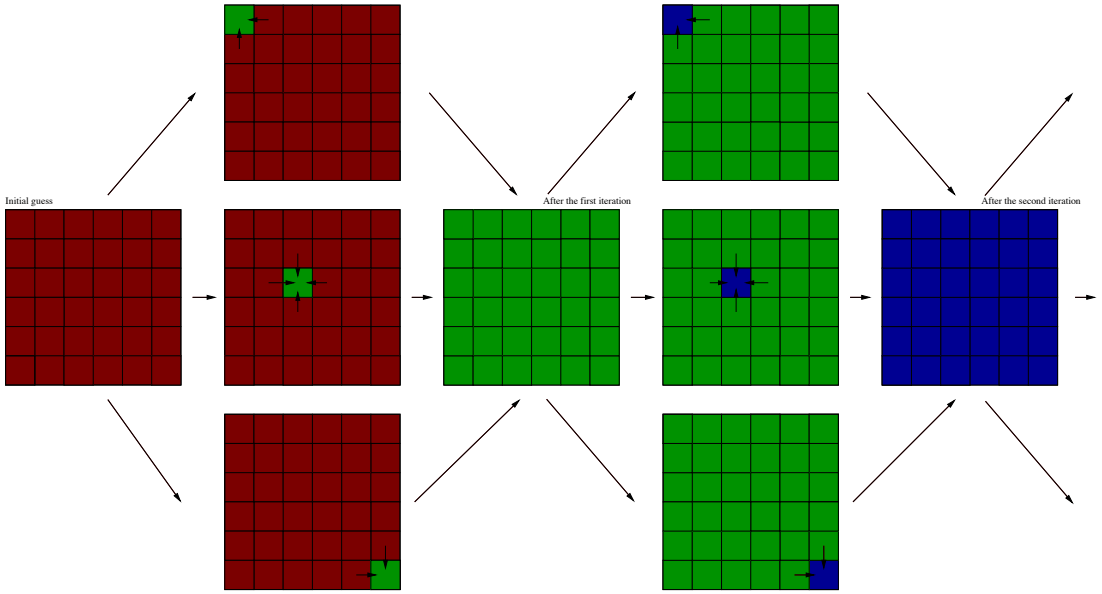


It is a good thing if you understand this…

Why is that easier?

Value iteration

Start with an initial guess at the utility function, and iteratively refine this:



* Problems to use this for
*Assumptions
* Top level algo for solution

The updating function:

$$U^{j+1}(s) := R(s) + \gamma \cdot \max_a \sum_{s'} P(s' \mid a, s) U^j(s').$$

Learning goals for today

Being familiar with:

- **Motivation** and **Formalization** for learning
- **Fundamental issues** like learning bias, overfitting
- **Neural net** basics and main idea for learning
- **Deep Learning** basics

New part: Learning!

Why do Learning?

- Learning **modifies the agent's decision mechanisms** to improve performance
- Learning is essential for **unknown environments** (when designer lacks omniscience)
- Learning is useful as a **system construction** method (expose the agent to reality rather than trying to write it down)

Currently we see lots of work in machine learning, due to:

- Availability of data massively increasing
- Increased utilization of hardware architectures (like GPUs)
- Method development (like deep learning)
- Clever definition of new tasks as machine learning problems

Overfitting

Consider error of hypothesis h over

- Training data: $\text{error}_t(h)$
- Entire distribution $\mathcal{D}$ of data (often approximated by measurement on test-set): $\text{error}_{\mathcal{D}}(h)$

Overfitting

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$\text{error}_t(h) < \text{error}_t(h') \text{ and } \text{error}_{\mathcal{D}}(h) > \text{error}_{\mathcal{D}}(h')$$

Connectionist Models

Some facts about the human brain:

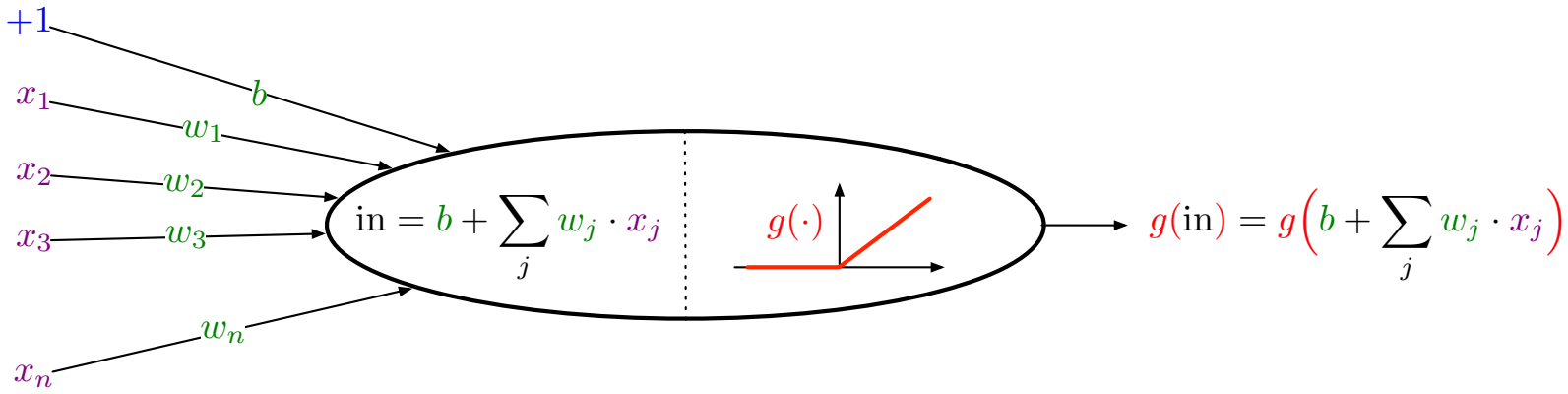
- Number of neurons about 10^{11}
- Connections per neuron about 10^4 – 10^5
- Scene recognition time about $.1$ second
- Neuron switching time about $.001$ second
- 100 inference steps doesn't seem like enough for scene recognition $\rightarrow$ much parallel computation

Properties of artificial neural nets (ANN's):

- Many neuron-like threshold switching units
- Many weighted interconnections among units
- Highly parallel, distributed process
- Emphasis on tuning weights automatically

Perceptron

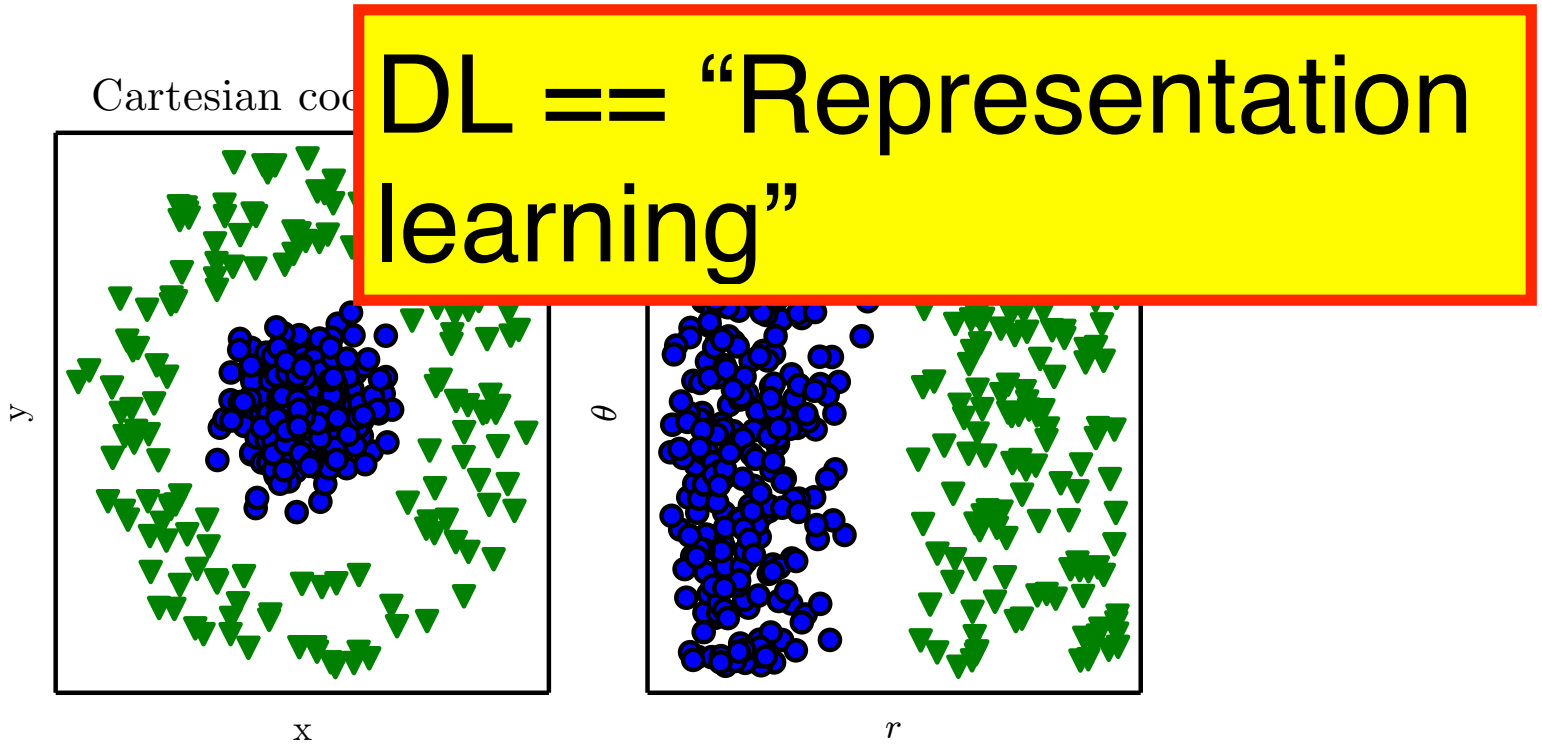
Output is a **nonlinear function** of the inputs with offset:



A gross **oversimplification** of real neurons, but its purpose is to develop understanding of what networks of simple units can do.

Be able to do inference (forward pass) and learning (backward pass) in this model. It IS very easy.

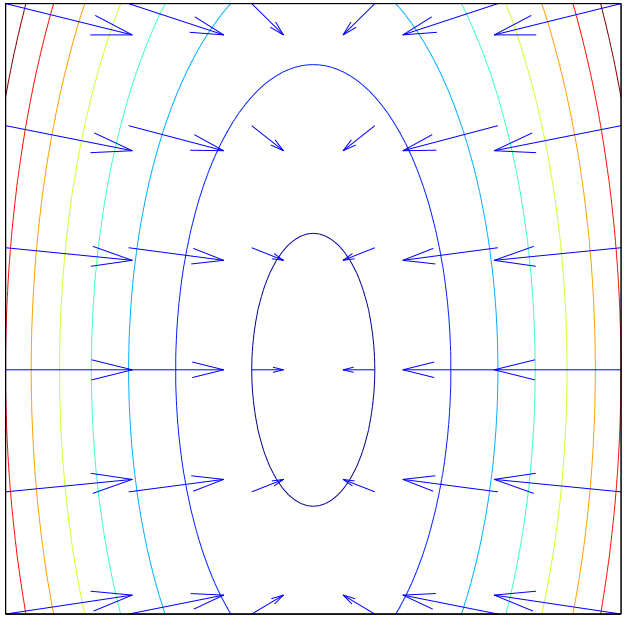
Representation matters



DL == "Representation learning"

- Representation in cartesian coordinates (x and y) is "difficult": Not linearly separable.
- Representation in polar coordinates (r and θ) is "easy".
  But how do we find this representation?

Gradient Descent – in higher dimensions

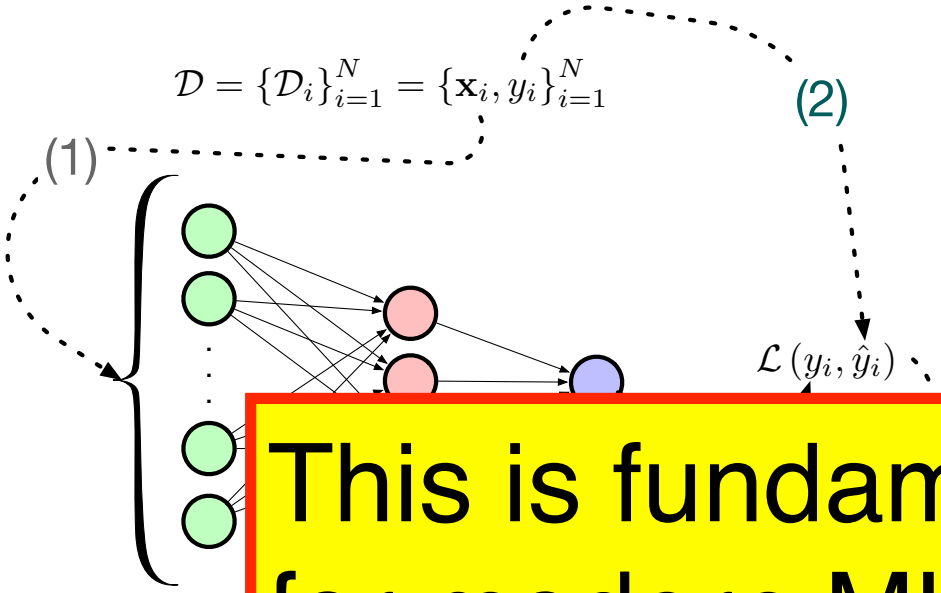


Recall that

- The **gradient** of a surface $\mathcal{L}[\vec{w}]$ is a vector in the direction the curve grows the most (calculated at $\vec{w}$).
- The gradient is calculated as $\nabla\mathcal{L}[\vec{w}] \equiv \left[\frac{\partial\mathcal{L}}{\partial w_0}, \frac{\partial\mathcal{L}}{\partial w_1}, \ldots \frac{\partial\mathcal{L}}{\partial w_n}\right]$.

Training rule: $\Delta\vec{w} = -\eta \cdot \nabla\mathcal{L}[\vec{w}]$, i.e., $\Delta w_i = -\eta \cdot \frac{\partial\mathcal{L}}{\partial w_i}$.

Gradient-based learning in neural nets



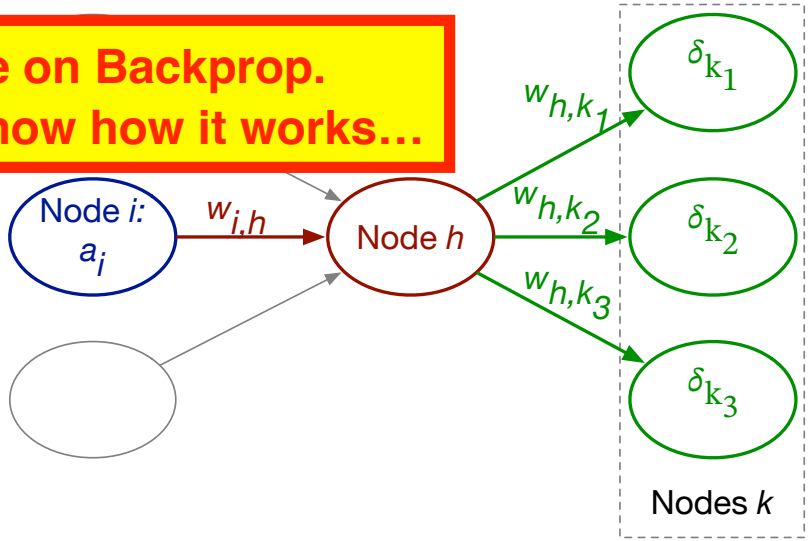
This is fundamental for modern ML.

Steps:

1. Push x_i **forwards** to calculate $\hat{y}_i$.
2. Calculate **loss** based on $\hat{y}_i$ and observed y_i: $\mathcal{L}(y_i, \hat{y}_i)$.
3. Calculate gradients $\nabla_w \mathcal{L}(y_i, \hat{y}_i)$ while moving **backwards**.
4. Update **weights**.

Backprop in yet another picture

We spent a lot of time on Backprop.
I will expect you to know how it works…

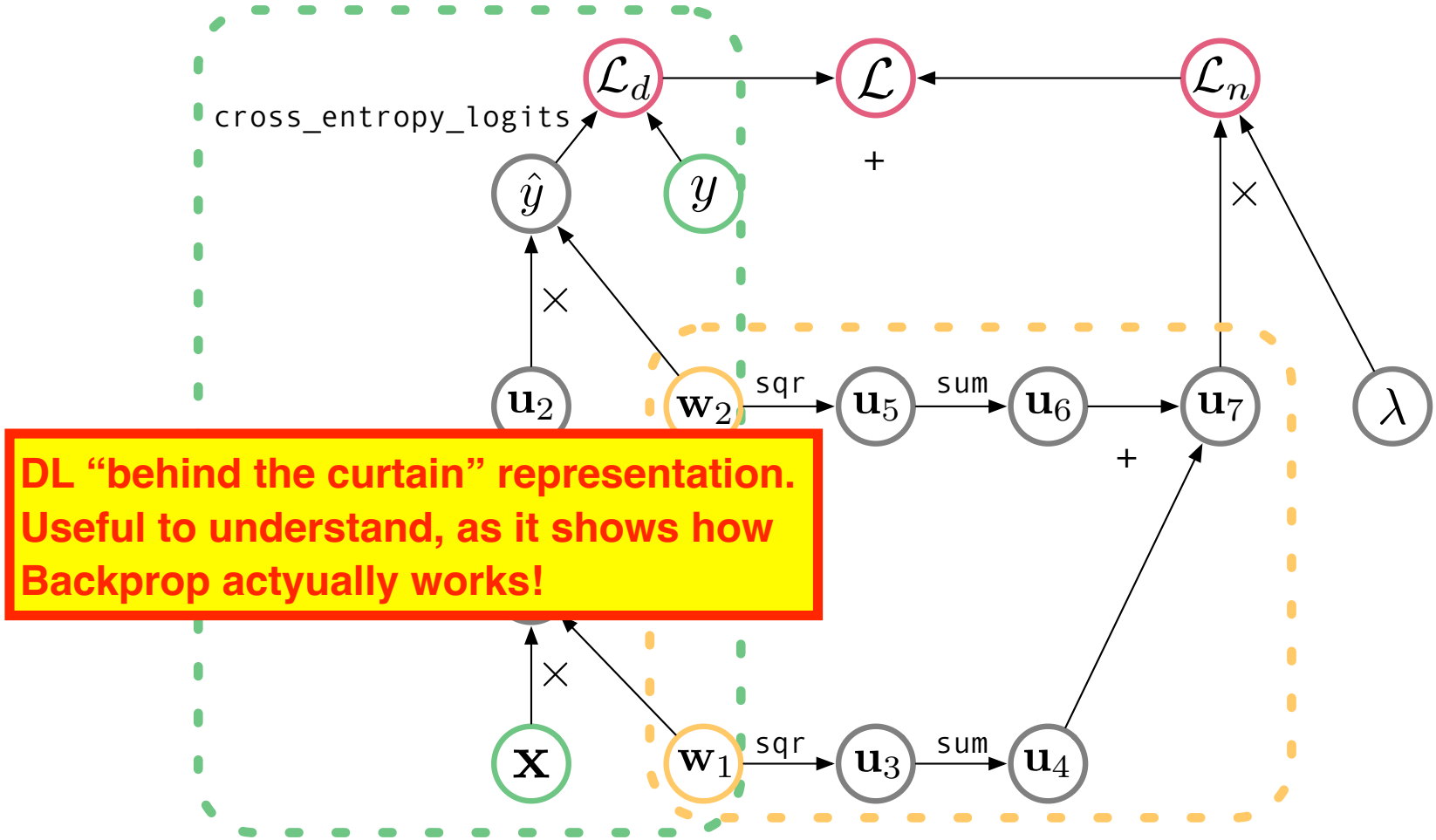


$$\delta_h \leftarrow a_h(1-a_h) \times \sum_k w_{h,k}\,\delta_k \quad \text{and} \quad \Delta w_{i,h} \leftarrow \delta_h \cdot a_i$$

Note the "locality" of the calculations:

- $\Delta w_{i,h}$ only cares about Node h, its parent (Node i), and its children (Nodes k).
- The calculation of $\delta_h = \frac{\partial \mathcal{L}}{\partial \text{in}_h}$ does not care about the parent (Node i), so we can use the same value for all parents!

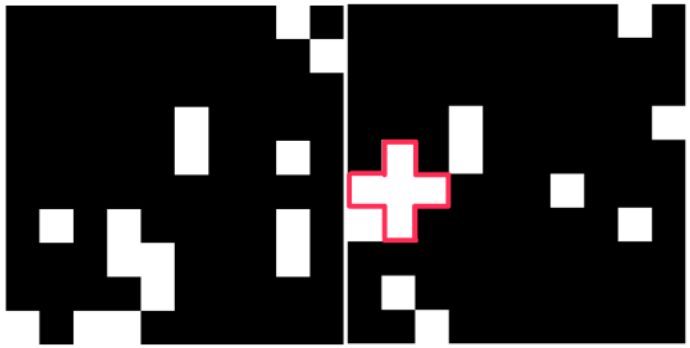
Computational graph for classification



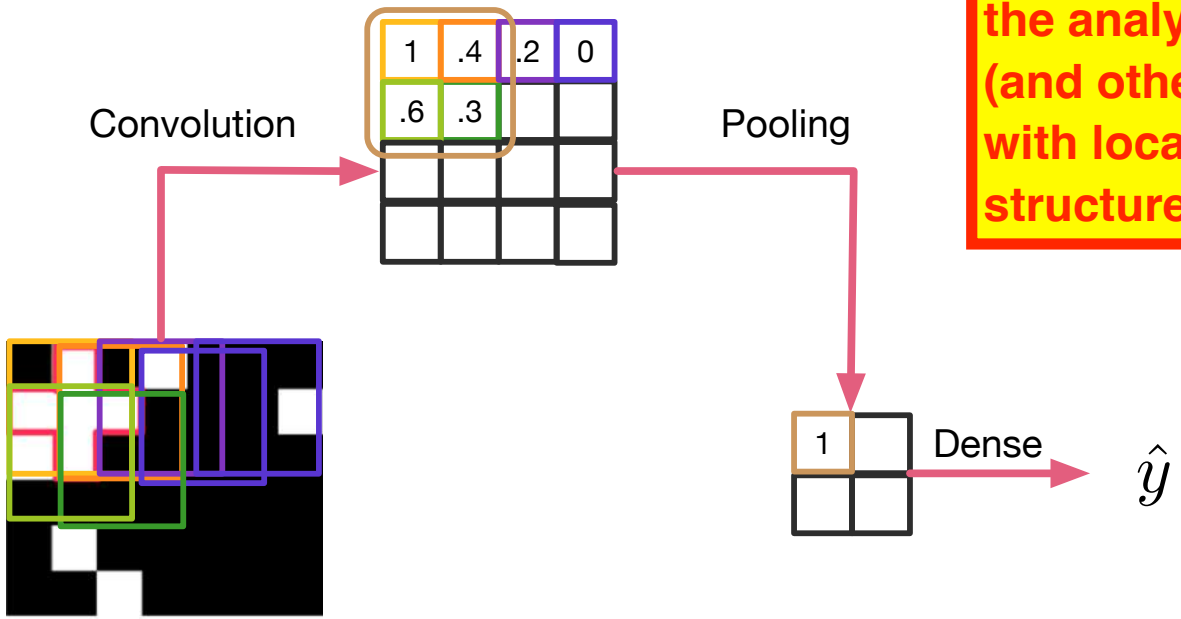
$$\mathcal{L} = \text{cross\_entropy\_logits}(y, \boldsymbol{w}_2^\mathsf{T} \overbrace{\text{relu}(\underbrace{\boldsymbol{w}_1^\mathsf{T}\boldsymbol{x}}_{\boldsymbol{u}_1})}^{\boldsymbol{u}_2} + \overbrace{\lambda \cdot \underbrace{(\|\boldsymbol{w}_1\|_2 + \|\boldsymbol{w}_2\|_2)}_{\boldsymbol{u}_7}}^{\mathcal{L}_n})$$

Learning convolutions

Data: Random binary images, some with a cross



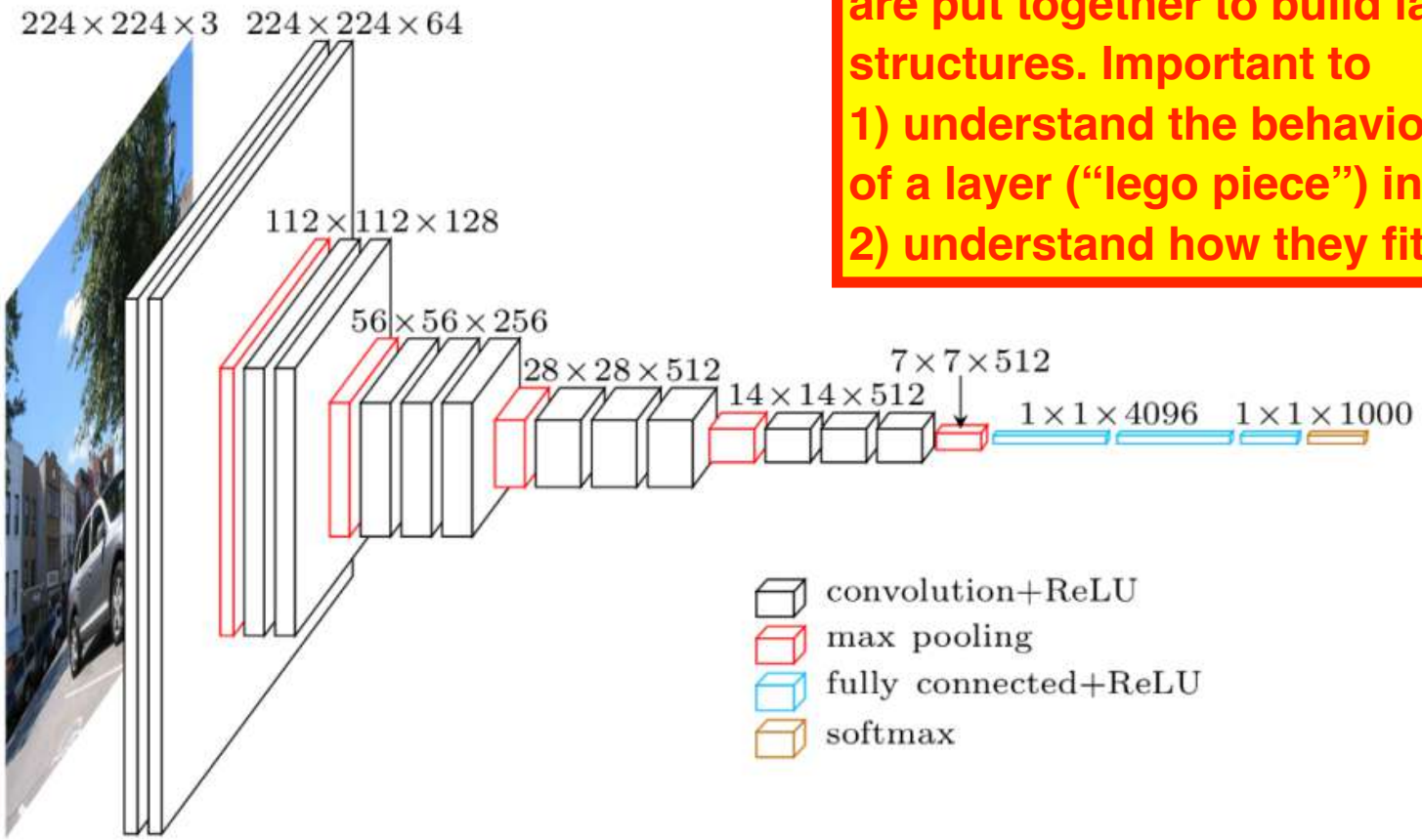
Model:



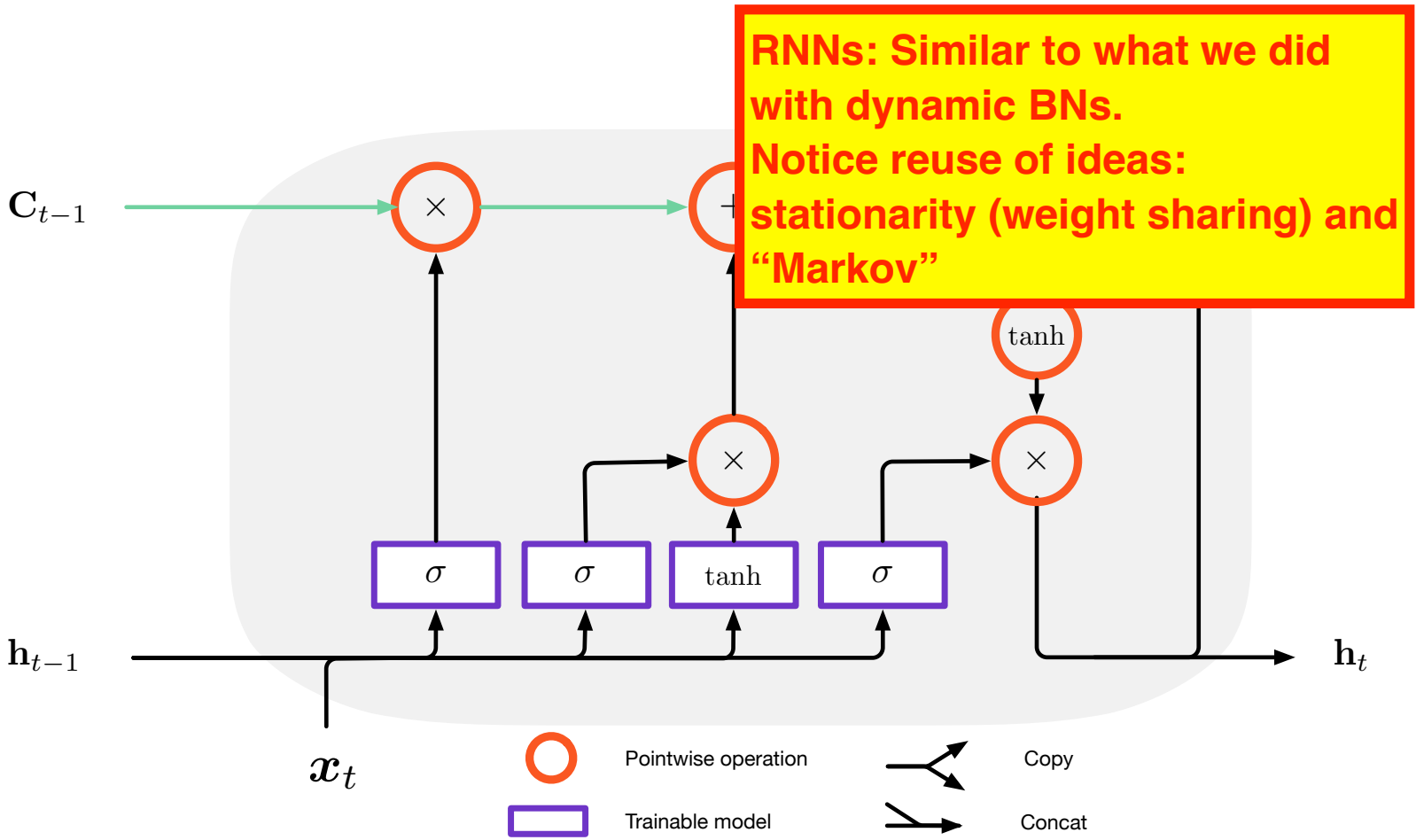
CNNs: The simple yet crucial idea that fuels the analysis of images (and other data-types with local correlation-structure)

Classic CNN model architecture: VGG16 from 2014

Building DNNs are akin to building lego: Smaller pieces are put together to build larger structures. Important to
1) understand the behaviour of a layer ("lego piece") in isolation
2) understand how they fit together!



LSTMs (Hochreiter&Schmidhuber, 1997)



LSTM gradients need not be diminishing
$\rightarrow$ Can learn long-term relationships.

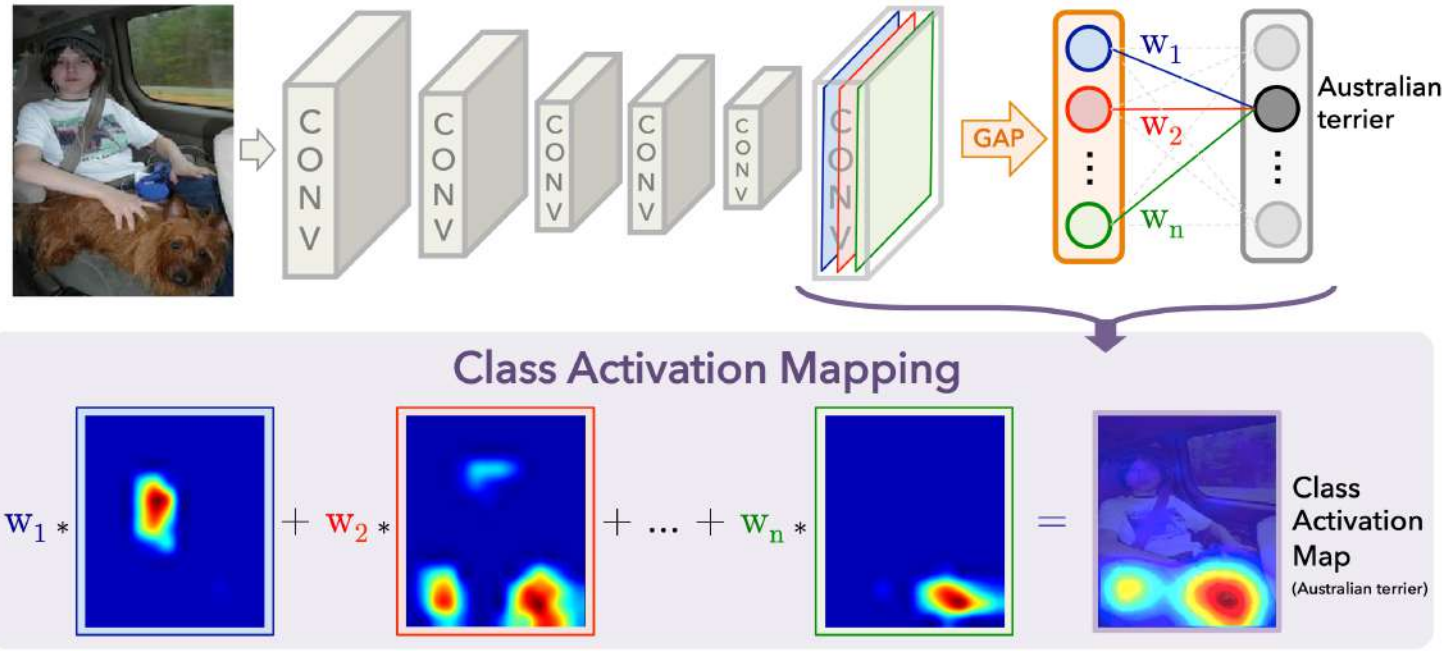
XAI: Understanding what the model does

Early example (2015)

XAI: Understand WHY (but not detailed "how").

This is an example of the general rule that "If in slides, then eligible for exam"

- After last c ions, so each filter is rep
- Learn class d class.
- Use location-based firing combined with filter weights to find class activation map.



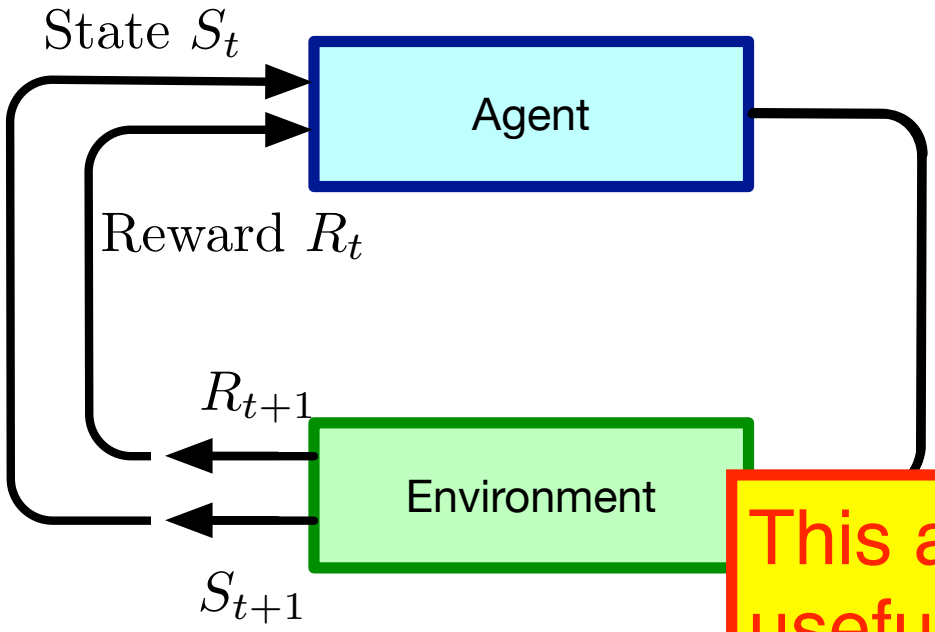
Learning goals for this reinforcement learning-part

Being familiar with:

- **Motivation** for reinforcement learning
- **Relation to MDP** – and what makes RL difficult
- **Learning part**, at least Q-learning for tabular problems
- **DRL** – motivation for general function approximators.
  High-level understanding of DQN and policy-based models.

Note! Topics on these slides are relevant for exam – even if not covered by book curriculum.

RL: Top-level view

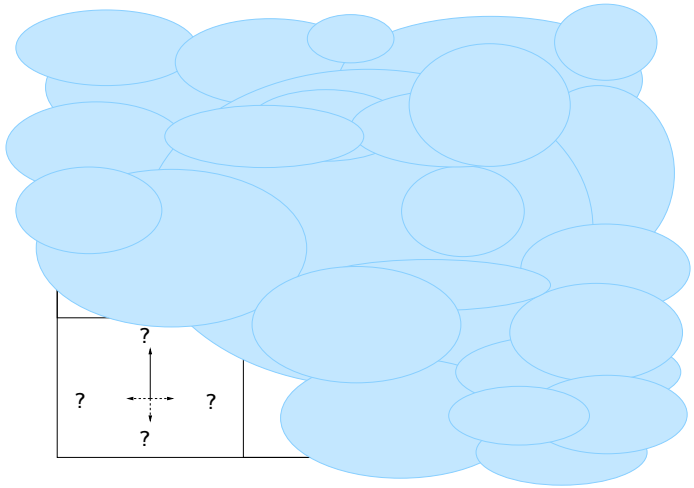


This abstraction level is useful to understand the "mechanics" of RL.

Loop:

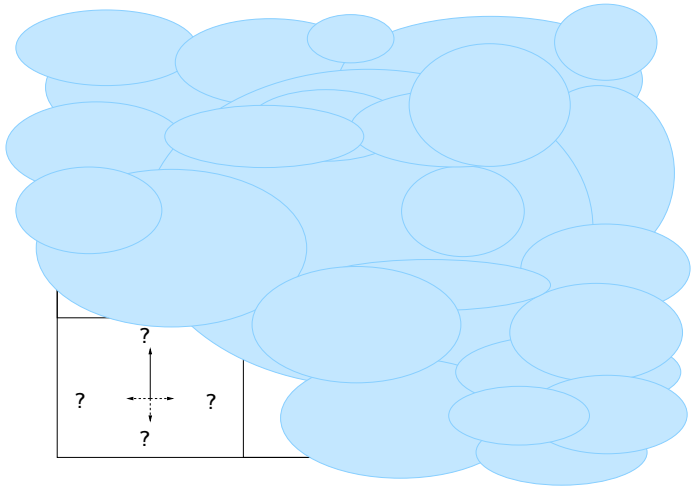
1. The agent receives a percept: Current state and reward
2. The agent updates its understanding of the environment.
3. The agent decides on what to do next
4. The selected action is executed in the environment.
5. The environment produces a new percept (state and reward)

Reinforcement learning in the "maze"



1. The agent gets rewards (**reinforcements**) when entering some states. **Rewards UNKNOWN**
2. **Infinite** time horizon; future rewards **discounted** (factor γ)
3. Model for outcome of actions is **UNKNOWN**, but the agent may spend (unbounded) time on **learning**
4. A **solution** defines action to choose in each state s to maximize expected discounted cumulative reward.

Reinforcement learning in the "maze"



Requirement:

We need something similar to value iteration, but more clever:

1. The agent must **explore** the domain ("maze") on its own
2. The effect of actions and the rewards for each state must be **learned**

Q-learning – deterministic world

We will solve the problem using "Q-learning":

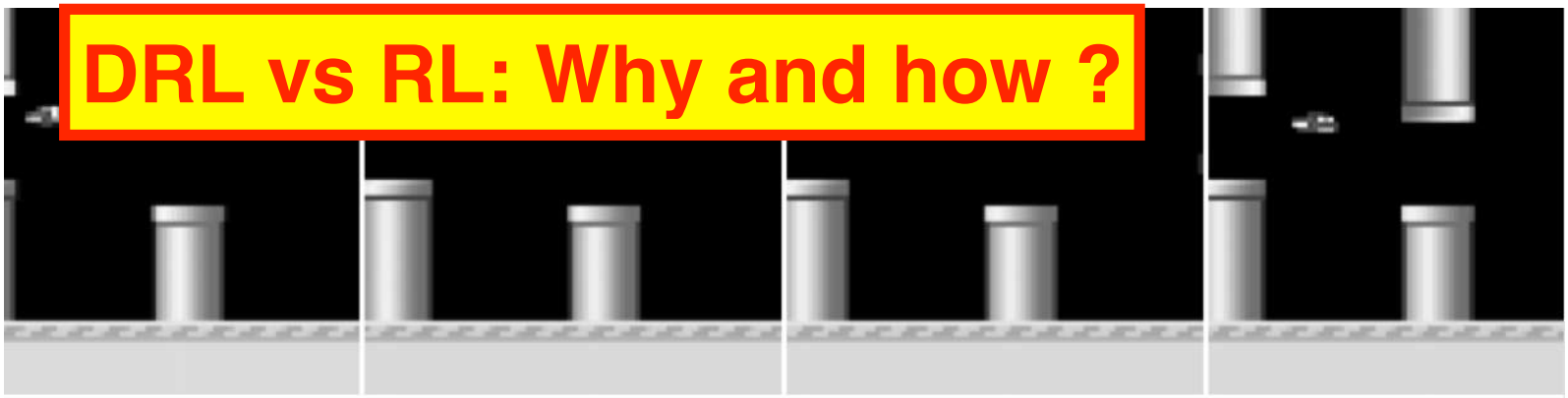
- $Q(a, s)$ is expect discounted cumulative rewards if we start by doing a in state s, and follow **optimal** policy thereafter.
- **Assume for now that** when doing a in a state s the agent always moves to the same state denoted $\delta(a, s)$.

So, $U^*(s) = \max_{a'} Q(a', s)$, and we have

$$\begin{aligned} Q(a, s) &= R(a, s) + \gamma \cdot U^*(\delta(a, s)) \\ &= R(a, s) + \gamma \cdot \max_{a'} Q(a', \delta(a, s)) \end{aligned}$$

Bellman equation: Recognize similarity with "Value Iteration"

Deep Flappy



- **Input:** 4 last screen-grabs from the game:
  - Each screen-grab is a $(\text{width} \times \text{height})$ - matrix of grey-scale values (floats between 0 and 1)
  - We have four of them, giving a tensor of size $(80 \times 80 \times 4)$
- **Model:** Some conv.layers, some dense layers. ϵ-greedy.
- **Output:** Values for $\hat{Q}(a = \text{flap}, s)$ and $\hat{Q}(a = \text{nothing}, s)$.
- **Learning:** Tune weights so that we ensure

$$\left[R(a,s) + \gamma \cdot \max_{a'} \hat{Q}(a', s')\right] - \hat{Q}(a,s) \approx 0.$$

- **Results:** Runs "forever"! (Do `./run_me.command` if time.)

Action selection – Exploitation vs. Exploration

In a state s_0 we should base our action selection on $Q(a, s_0)$:

Greedy: Choose $a = \arg\max_{a'} \hat{Q}(a, s_0)$ (*Yoda disagrees. . .*).

Random: Choose an action on random, all actions equally likely

Mix: With probability p choose an action on random, with probability $1 - p$ be greedy.

Guided: Each action a gets a probability of being chosen proportional to $k^{\hat{Q}(a,s_0)}$ where $k > 1$ is some parameter (typically growing as the agent learns more)

Some examples of possible RL-questions:
* Type of problems, and how to model them
* Assumptions of RL (Markov, observable state, known actionset, ...)
* Differences between this and standard fully observed MDPs ==> Difference between Q-learning and Value Iteration
* Passive vs Active ==> Q-learning vs. action selection
* RL vs DRL

Language

- Language is a tool to
- Formal languages, lik
  semantics. Natural la

Why is NLP hard? Feel for tasks, and what it means to solve them

- Problems w/ analyzing natural language:
  - Ambiguity: "Bank" as financial institution vs. "river bank"
  - Subjectivity: "Football" vs. "soccer"
  - Incorrectness/Inconsistency: "I like to sleeping"
- Natural Language Processing tasks can be, e.g.:
  - Information extraction: Document to nuggets
  - Information retrieval: Information need to document
  - Machine translation: Language to language
  - Question-Answering: Query to nugget
  - Conversational systems: Learning, entertainment
  - Speech recognition: Sound to text
  - Speech synthesis: Text to sound

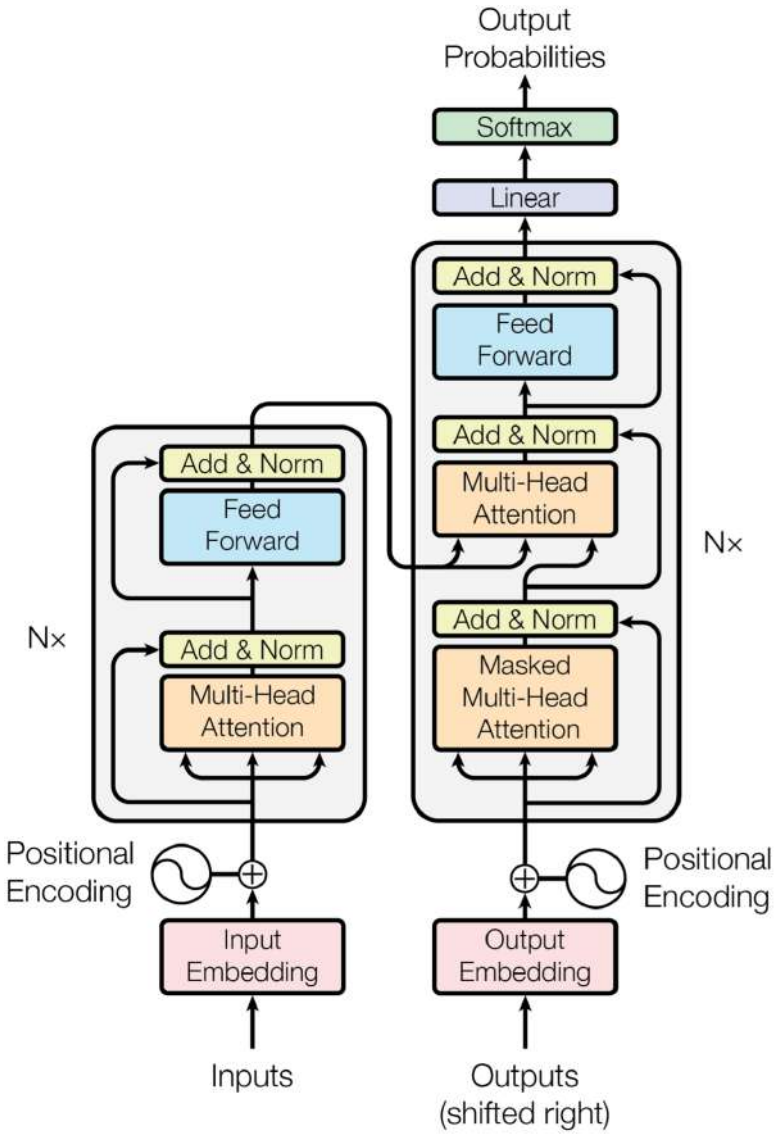
Probabilistic language model

- A **probabilistic language model** defines a probability distribution over (possibly infinite) list of strings
- This is an alternative to **logical** language models, and there are **several advantages**:
  - Can be **trained from data**; learning is simply counting occurrences in a **text corpora** (e.g., www)
  - More **robust**: recognizes that not all speakers agree upon when a sentence is part of a language.
  - Allows **disambiguation**: Use probabilities to say which interpretation is more likely
- Note that spoken languages (in contrast to e.g. programming languages) are "vague" both wrt. syntax and semantics $\Rightarrow$ Logic-based language model may break.

The first big question for deep learning: Representation!

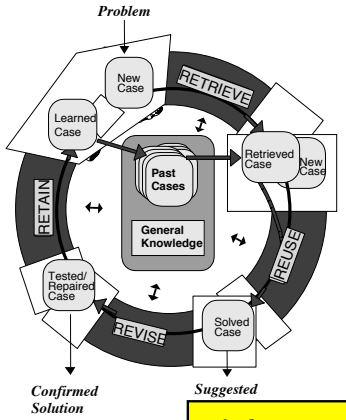
- A sentence is a **sequence** of **words** (or "tokens")
  - We are not afraid of the sequence-part: We have **RNNs**
- ... but **what about each word**?
  - If we have B words in the vocabulary, why not give them unique index-values?
  - ... or use one-hot encoding (binary vector of length B)? And maybe extend to something encoding n-grams, too?
- We will do better using **word-embeddings**:
  - For each word we define a high-dim. **vector-representation** for the word. This is a d-dimensional vector of real values, that hopefully somehow encodes the **semantics** of the word.
  - The representation is used **instead of** one-hot or whatever.
  - We **learn** the representation from massive data-sets.

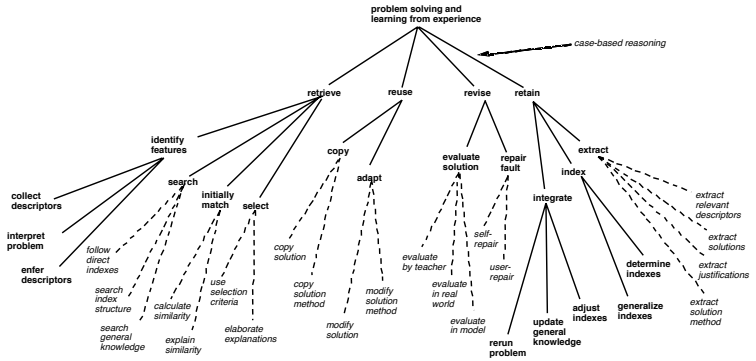
The Transformer architecture



- Google (2017): *"Attention Is All You Need"* describes an encoder-decoder model that feeds off these relations.
- The **attention**-module extremely important in NLP (and other DL applications).
- Use **positional encoding** to help encode sequential structure of data.
- Modelling **heavily optimized** for parallell computation on GPUs.

The CBR Cycle





* Instansbased learning vs "standard" learning
* Instancebased vs. CBR
* The CBR cycle

— and whatever Kerstin told you about the papers

Case-based approaches

- Instance-based reasoning/learning
- Memory-based reasoning/learning
- Case-based reasoning/learning (typical)
- Analogical reasoning/learning

Instance-based methods

- Motivated by classical machine learning research
- Addresses classification tasks
- A concept (class) is defined by its set of exemplars:
  Concept space = Instance space + Similarity metric
- Representation is attribute-value pairs
- Knowledge-poor method
- 'IBL' framework (Kibler&Aha) contains
  - Similarity function
  - Classification function
  - Concept decsription updater

What we are thinking about when making the exam . . .

- Avoid "just" **memorizing the book**:
  - **I don't like to ask for "clean facts":** Less of *"Do you remember . . . ?"* and more of *"Do you understand . . . ?"*
  - **Challenge understanding:** Rather *"How can you solve . . . ?"* than *"Use [XXX] to solve . . . ?"*
- Fairness for **everyone:** I like a good number of A's, but aim for a good spread! (Typically some also fail the course.)
  - **Zero-expectation randomness:** I typically give negative points for wrong answers in True/False questions. Otherwise, you will get 50% score with *no* knowledge.
  - **Lots of questions:** By asking *many* questions we can better cover the full curriculum. If someone gambles on not reading a chapter, I want that gamble to fail.

TDT4171 - Artificial Intelligence Methods Final Exam Spring 2024

Introduction

- The exam contains 9 tasks with several subtasks in each task.
- Each subtask is worth a certain number of % points, adding up to the number of points specified for the task and 100% for the complete exam.
- If the description of the task is ambiguous, specify your assumptions and solve the task according to these assumptions.
- To get maximum score, the answer should be correct, complete, well-justified and precise. Partial points will be given for answers that are somewhat lacking in content or quality.
- In several tasks, we specified a range for how many sentences should be used for the answer. This is a recommended number of sentences to give you an idea about the expected level of detail. Minor deviations from this range are acceptable. If your answer is significantly longer, consider summarizing the main points in fewer sentences. If your answer is shorter, consider expanding it a bit.

1 [6%] AI foundation

1.1 [3%]

W
in

S

G

A

Exam made by: Not me.
Still fairly similar to what you will get… (…maybe…)!

I do not focus on how to solve the exam questions here, but rather on why we ask types of questions, and when relevant how we grade your answers.

1.2 [3%]

List any five ethical concerns / misuses of AI.

Solution

1. Lethal autonomous weapons
2. Large scale surveillance and privacy.
3. Fairness and bias
4. Lack of transparency
5. Work automation, people loosing jobs
6. Concentration of wealth

Grading

There can be variations, e.g. more specific concerns. Give % proportional to how many items are mentioned.

2 [10%] Uncertainty and Bayesian networks

2.1 [3%]

Given the following joint probability distribution:

	toothache		¬ toothache	
	catch	¬ catch	catch	¬ catch
cavity	.13	.03	.07	.01
¬ cavity	.01	.06	.12	.57

Compute the following probabilities:

1. $P(toothache)$
2. $P(cavity \vee toothache)$
3. $P(\neg cavity \mid toothache)$

Round your answers to two decimals, e.g. 1.236 should be rounded to 1.24

Solution

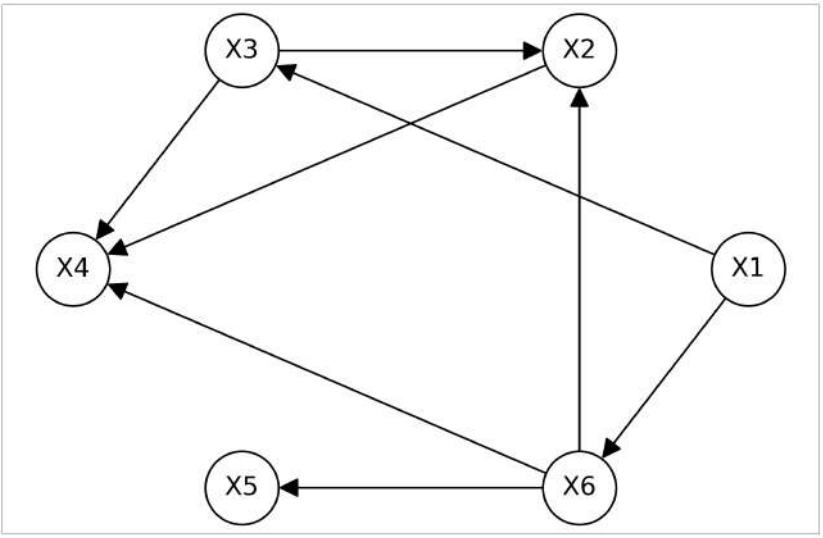
1. $P(toothache) = 0.13 + 0.03 + 0.01 + 0.06 = 0.23$
2. $P(cavity \vee toothache) = 0.13 + 0.03 + 0.01 + 0.06 + 0.07 + 0.01 = 0.31$
3. $P(\neg cavity \mid toothache) = P(\neg cavity \wedge toothache)/P(toothache) = (0.01 + 0.06)/(0.13 + 0.03 + 0.01 + 0.06) = 0.30$

Grading

Automatic

2.2 [7%]

Given the following Bayesian network:



Answer these Yes/No questions. +1% for correct and -1% for incorrect, 0% points for no answer. +2% for all correct.

1. Is X1 independent of X4?
2. Is X3 independent of X6 given X2?
3. Is X1 independent of X5 given {X4, X6}?
4. Is X1 independent of X2 given {X4, X5, X6}?
5. Is X3 independent of X5 given {X1, X2, X4, X6}?

Solution

1. No
2. No
3. Yes
4. No
5. Yes

Grading

Automatic

3 [16%] Decision networks

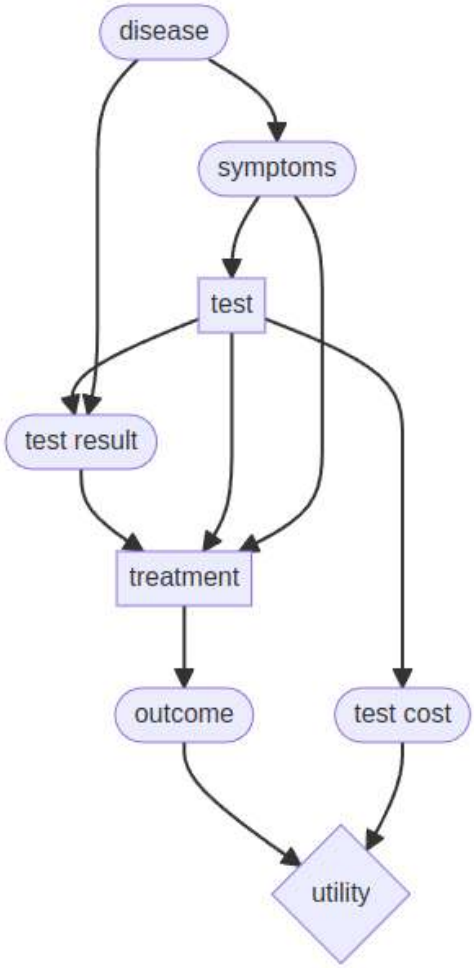
3.1 [5%]

Consider a scenario where a doctor is diagnosing a patient with some symptoms. The doctor needs to make two decisions: (1) what kind of test to perform and (2) what treatment to prescribe.

Construct the decision network for this scenario. Include nodes for disease, symptoms, test, test cost, test results, treatment, treatment outcome and utility. Use ovals for chance nodes, rectangles for decision nodes and diamonds for utility nodes.

Draw the structure (the directed acyclic graph) of this model on **paper**. You are not asked to provide conditional probability tables. However, make your model so that it would be as easy as possible for a domain expert to provide the required probabilities.

Solution



Some other variations are acceptable as well. Mostly looking at the overall understanding of how the decision network can be structured.

Grading

- Full points for all nodes with the correct type and edges with the correct direction. Missing or adding one, two edges is ok with some explanation.
- Partial points if most of the network is correct.
- No points if the student doesn't know what a decision network is.

3.2 [4%]

Given 3.1, specify the conditional probability table for the node representing test result. Possible values for disease: flu and allergy. Possible values for test: flu test and allergy test. Possible values for test results: positive and negative.

Solution

disease	test	test result	probability, %
flu	flu	positive	70
flu	flu	negative	30
flu	allergy	positive	30
flu	allergy	negative	70
allergy	flu	positive	10
allergy	flu	negative	90
allergy	allergy	positive	80
allergy	allergy	negative	20

Alternative without negative test results, i.e. (100 - positive) with the same disease and test values.

disease	test	test result	probability, %
flu	flu	positive	70
flu	allergy	positive	30
allergy	flu	positive	10
allergy	allergy	positive	80

Grading

We are checking that the student understand what conditional probability table is. Probability of positive result for a disease with the corresponding test should be higher than when using another test. Important that rows with the same disease and test values should sum up to 100%. Specific numbers are not important.

3.3 [3%]

Design utility for the network in 3.1 The utility should be based on outcome and test cost. You may use an additive value function. Possible values for the outcome: No symptoms, fewer symptoms, same symptoms. Possible values for the test cost: high, moderate, low.

Solution

Assign utility values to variables.

Outcome: 1. No symptoms: 100 2. Fewer symptoms: 50 3. Same symptoms: 0

Test cost:

1. High: -100
2. Moderate: -50
3. Low: -10

U(outcome, cost) = outcome * 3 + cost

Grading

The students are expected to know how to define simple utility. Concrete numbers don't matter, just needs to be something that makes sense.

3.4 [4%]

Specify the general algorithm (briefly describe steps) for evaluating decision networks, i.e. how they are used for making decisions. You don't need to explain how to calculate posterior probabilities.

Solution

1. Set the evidence variables for the current state.
2. For each possible value of the decision node
    1. Set the decision node to that value.
    2. Calculate the posterior probabilities for the parent nodes of the utility node.
    3. Calculate the resulting utility for the action.
3. Return the action with the highest utility

Grading

- Full points for including all the steps.
- Partial points if some steps are missed.
- No points if the algorithm is mostly wrong.

4 [20%] Probabilistic reasoning over time - Hidden Markov model

One of the challenges with solar energy production in the Nordic countries is snow covering solar panels. Solar panels can still produce some energy if a snow layer is thin or only part of the panel is covered. There are no sensors that can measure it directly, so we have to infer whether the panel is covered based on measured production.

Consider the following probabilities:

- Given that the panel is covered with snow, the probability that a solar panel produces energy is 0.1.
- Given that the panel is **not** covered with snow, the probability that a solar panel produces energy is 0.7.

- Given that the panel is covered with snow during the current hour, the probability that the panel will be covered with snow for the next hour is 0.9.
- Given that the panel is **not** covered with snow during the current hour, the probability that it will **not** be covered with snow for the next hour is 0.8.
- The prior probability for the panel being covered with snow is 0.3.

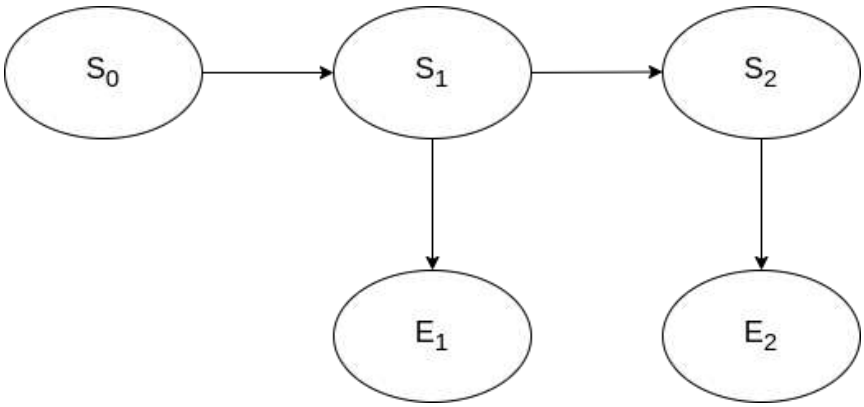
4.1 [8%]

Specify a hidden Markov model representing the problem above. You need to specify the structure and the probability tables. The structure should include steps (hours) 0, 1 and 2. Draw the model and tables on **paper**.

Use the following notation:

- S_t - random variable for the panel covered with snow at step t
- s_t - value for S_t at step t
- E_t - random variable for the panel producing energy at step t
- e_t - value for E_t at step t

Solution



Transition model:

S_{t-1}	$P(S_t \mid S_{t-1})$
t	0.9
f	0.2

or

S_{t-1}	S_t	$P(S_t \mid S_{t-1})$
t	t	0.9
t	f	0.1
f	t	0.2
f	f	0.8

or

S_{t-1}	$P(S_t = s_t \mid S_{t-1})$	$P(S_t = \neg s_t \mid S_{t-1})$
t	0.9	0.1
f	0.2	0.8

Observation model:

S_t	$P(E_t \mid S_t)$
t	0.1
f	0.7

or

S_t	E_t	$P(E_t \mid S_t)$
t	t	0.1
t	f	0.9
f	t	0.7
f	f	0.3

or

S_t	$P(E_t = e_t \mid S_t)$	$P(E_t = \neg e_t \mid S_t)$
t	0.1	0.9
f	0.7	0.3

I have also seen several other variations for students that are considered correct.

4.2 [3%]

Given the hidden Markov model from 4.1, are these statements true or false? +0.5% for correct and -0.5% for incorrect, 0% points for no answer.

1. $P(S_t|E_t)$ is called the observation model.

2. First-order Markov assumption can be expressed as $P(S_t|S_{0:t-1}) = P(S_t|S_{t-1})$
3. Filtering is the task of computing $P(S_k|e_{1:t})$ where t is the most recent time step and $0 \leq k < t$.
4. Prediction is the task of computing $P(S_{t+k}|e_{1:t})$ where t is the most recent time step and $k > 0$.
5. Smoothing is the task of computing $P(S_t|e_{1:t})$ where t is the most recent time step.
6. Most likely explanation is the task of computing $P(S_{1:t}|e_{1:t})$ where t is the most recent time step.

Solution

1. False
2. True
3. False
4. True
5. False
6. False

Grading

Automatic

4.3 [9%]

Compute the probability of the panel being covered with snow at the second hour given that our measurements show energy production at hour 1 and no production at hour 2. Show your computations step by step for steps 0, 1 and 2. Show computations in symbolic form (variables, probabilities and distributions) before replacing them with numbers. In you computations, round intermediate and final results to two decimals.

Solution

These computations almost exactly follow the example at page 485 and lecture 4 but with different symbols and numbers.

Hour 0

No energy measurements available, we only have prior beliefs:

$$P(S_0) = \langle 0.3, 0.7 \rangle$$

Hour 1

We measured some energy production, so $e_1 = t$

Predict from t_0 to t_1:

$$P(S_1) = \sum_{s_0} P(S_1 \mid s_0)p(s_0) = \langle 0.9, 0.1 \rangle \times 0.3 + \langle 0.2, 0.8 \rangle \times 0.7 = \langle 0.41, 0.59 \rangle$$

Update given $e_1 = t$.

$P(S_1 \mid e_1) = \alpha P(e_1 \mid S_1)P(S_1) = \alpha\langle 0.1, 0.7\rangle\langle 0.41, 0.59\rangle = \alpha\langle 0.04, 0.41\rangle = \langle 0.09, 0.91\rangle$

α is a normalization constant.

Hour 2

We measured no energy production, so $e_2 = f$

Predict from t_1 to t_2:

$P(S_2 \mid e_1) = \sum_{s_1} P(S_2 \mid s_1)p(s_1 \mid e_1) = \langle 0.9, 0.1\rangle \times 0.09 + \langle 0.2, 0.8\rangle \times 0.91 = \langle 0.26, 0.74\rangle$

Update given $e_2 = f$:

$P(S_2 \mid e_1, e_2) = \alpha P(e_2 \mid S_2)P(S_2 \mid e_1) = \alpha\langle 0.9, 0.3\rangle\langle 0.26, 0.74\rangle = \alpha\langle 0.23, 0.22\rangle = \langle 0.51, 0.49\rangle$

Grading

- Full points for correct or almost correct computation flow even if the numbers are a bit wrong. Check that the distributions are at least somewhat close to the solution.
- Partial points if the student made major mistakes in the computation in symbolic level.
- No points if the flow is completely incorrect.

5 [10%] Artificial neural networks - Gradient descent

Given the Gradient Descent algorithm for the perceptron:

1. Initialize each w_i to some small random value
2. Until the termination condition is met:
    1. Initialize: $\Delta w_i \leftarrow 0$
    2. For each $\langle \vec{x}, t\rangle$ in D:
        - Input $\vec{x}$ to the unit and compute the output o
        - For each w_i: $\Delta w_i \leftarrow \Delta w_i - \eta \cdot 2(-x_i(t - o))$
    3. For each w_i: $w_i \leftarrow w_i + \Delta w_i$

5.1 [5%]

Explain what the following symbols mean and their role in this algorithm: D, $\vec{x}$, t, w, η. Answer with 1-2 sentences per symbol.

Solution

- D - training data set containing pairs with feature vectors and target values
- $\vec{x}$ - feature vectors, characteristic of a training instance, provided as input to the perceptron.

"Answer with up to X sentences" is a way for us to get free-form feedback. It can be good to "practice" — try to summarize important concepts for yourself in a clear "bullet-list" way. Almost any topic can be formulated in this way, as we see in this question-set.

Grading: Coverage of the main points in the solution. Still, don't feel like you HAVE to write the max no. sentences. Lots of fluffy or irrelevant text is not helping your cause, and we often see students exposing lack of knowledge when trying to show off with tangential (at best) information. If you find yourself in that situation, you have spent time (written more) in order to LOOSE points.

- This term is derived from the partial derivative of the squared error function with respect to each weight w_i.
- Using the gradient this way makes sense because it indicates the direction and magnitude of the steepest ascent in the error function's value.
- By subtracting this gradient term from the current weights moves the weights in the direction that minimizes the error, improving the model's accuracy iteratively.

6 Deep learning [10%]

6.1 [2%]

What is "deep" in deep learning, and why does it help? Answer with 3-10 sentences.

Solution

- "Deep" in deep learning refers to the use of multiple layers of neurons in a neural network, forming a deep architecture as opposed to a shallow one with only a few layers.
- This depth allows the model to learn and represent complex features and patterns at various levels of abstraction.
- Each layer captures higher-level features built upon the lower-level features extracted by previous layers.
- This enables deep learning models to identify patterns that shallow models can't.

6.2 [4%]

What is overfitting, and how can we avoid it in deep learning? Answer with 3-10 sentences.

Solution

Overfitting occurs when a deep learning model learns the training data too well, including its noise and outliers, resulting in poor generalization to new, unseen data. To avoid overfitting, several techniques can be employed: using a larger and more diverse dataset, implementing regularization methods like L1 or L2

regularization, applying dropout to randomly deactivate neurons during training, and utilizing early stopping to halt training when performance on a validation set begins to degrade. Data augmentation and cross-validation can also help in creating a more robust model that generalizes better.

6.3 [4%]

Explain why convolution networks work better for image data than a multilayer perceptron. Answer with 3-10 sentences.

Solution

Unlike MLP, CNNs leverage the spatial structure of images through convolutional layers. These layers apply convolutional filters that slide across the image detecting patterns such as edges, textures, and more complex structures at different locations. The same set of filters is applied across the whole image with shared weights. This reduces the number of parameters, making the model more efficient, faster to learn. CNNs also use pooling layers to reduce the spatial dimensions and amplifying the most prominent signals. This also makes the model more computationally efficient and less prone to overfitting.

7 [8%] Natural language processing

7.1 [4%]

In natural language processing, what are word embeddings, and why are they better for many NLP tasks than one-hot encoding? Answer with 2-5 sentences.

Solution

In natural language processing, word embeddings are dense vector representations that capture the semantic meanings of words by placing similar words closer together in the vector space. They are better than one-hot encoding because they provide meaningful relationships between words and reduce the dimensionality, improving the performance and efficiency of NLP models.

7.2 [4%]

How would you use word embeddings to classify emails as spam, not spam? Here you only need to outline the general idea. Answer with 2-5 sentences.

Solution

1. Convert the words in each email into their corresponding word embedding vectors using a pre-trained embedding model.
2. Aggregate these word embeddings to form a single fixed-size representation for each email, such as by averaging the embeddings.
3. Feed these aggregated embeddings into a classification model, such as neural network, which is trained to distinguish between spam and not spam emails based on these features.

4. Finally, the trained model can classify new emails based on their aggregated word embeddings.

8 [10%] Reinforcement learning

8.1 [2%]

What does "reinforcement" stand for in reinforcement learning? How is it different from supervised and unsupervised learning? Answer with 2-5 sentences.

Solution

Reinforcement refers to the occasional rewards the agent receives from the environment based on the actions taken. This reinforces behaviors that lead to higher rewards, guiding the agent to learn optimal strategies over time. Unlike supervised learning, which relies on labeled data, and unsupervised learning, which finds patterns in data without labels, reinforcement learning focuses on interaction with an environment to maximize cumulative rewards over multiple steps, making it particularly suitable for sequential decision-making tasks.

8.2 [4%]

How is reinforcement learning (RL) different from solving a Markov decision process (MDP)? Answer with 2-5 sentences.

Solution

- RL is used when the agent must learn the optimal policy through interaction with an environment. In contrast, we can find an optimal policy for MDP without interacting with an environment.
- In RL the agent doesn't know the transition probabilities and reward functions a priori. In contrast, when solving an MDP these dynamics are fully known.
- RL relies on exploration and exploitation to learn the optimal actions. In contrast, solving an MDP involves no exploration.

8.3 [4%]

Explain how deep learning is combined with reinforcement learning to play Flappy Bird. If you don't know Flappy Bird use Mario or some other simple arcade game as an example. You don't need to explain the complete architecture or details of the learning algorithm. Focus on how deep learning is integrated into reinforcement learning. Answer with 3-10 sentences.

Solution

For Flappy Bird, we can use Deep Q-Learning.

- Deep neural network is used to approximate the Q-function.

- Q-function estimates the expected rewards for taking certain actions in given states.
- The game's current state is represented by pixel colors in several consequent game screenshot.
- The state is fed into a convolutional neural network that outputs a vector of values containing q-values for each action.
- During testing we select an action with highest q-value.
- Training is similar to standard q-learning, but q-function is learned by updating weights in a deep neural network.

9 [10%] Case-based reasoning

9.1 [5%]

Explain each step in CBR cycle. 1-2 sentences per step.

Solution

1. Retrieve: the most similar case or cases: The case(s) with the most similar proble
2. Reuse: the ret
3. Revise proble
4. Retain sisting of the presented problem description and the adapted solution description as a new experience in the case base

This question is overused, but nevertheless it still re-appears regularly.

9.2 [5%]

Find a problem/application where case-based reasoning would perform better than deep learning? Explain why it would be better for this problem. Answer with 3-10 sentences.

Solution

- An diag
- In t mod
- CBR with
- CBR for
- Eac which is important for rare diseases.
- Reasoning in CBR is much easier to explain, verify and correct manually, which makes for a better decision-support system.

This is a classic:
Show that you know the tools in your toolbox by understanding which tool works best for which task.

- CBR doesn't require computation resources for training large models as in deep learning.

Summary of the summary

1. Lots of stuff to learn in this course – and afterwards.
   - I hope to see many of you again in a year or two.
2. Look at the slides. If something is on the slides it is part of the syllabus. (If it is not, it is **still** part of the syllabus. . .)
3. Remember to read the two papers on CBR.
4. At the the exam, do make sure to read, and understand, the full set, including how points are awarded.
   - Don't be afraid of multiple-choice – even if there may be lost points for wrong answers!
5. I *may* get technical, e.g., ask what is $P(X = 1|Y = 0, Z = 0)$.
6. Help each-other on Piazza during your preparations.
   - We are also here to help, and may (within reason) answer *"well-researched"* questions on Piazza if no students are able to.
7. *New this year:* No communication regarding errors in the question-set (if any). If you think something is wrong, then fix as you see fit, **tell us** what you did & why, solve, and move on.
8. **Good luck!**