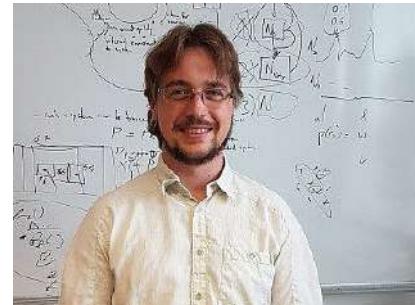


Umělá inteligence a strojové učení

Michal Hradiš: Úvod



Vyučující SUI



Michal Hradiš
Q201 - Neuronové sítě, RL



Karel Beneš
L230 - Prohledávání, ...



Lukáš Burget
L229 - Strojové učení

The big question



AI in your everyday Life?

AI in everyday Life?



- Search
- Spam filters
- Recommendation - Youtube / Amazon / Facebook /
- Speech recognition / speech synthesis
- Dialogue systems / Digital voice assistants
- Spell checking
- Translation
- Driving assistants / self-driving cars
- Face recognition / identification
- Fraud detection
- Chat bot, coding assistant

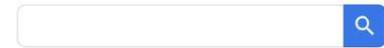


What is AI?

- AI agents - general/strong AI: How can we create intelligence?
- Tools - narrow AI: How can we create useful tools?



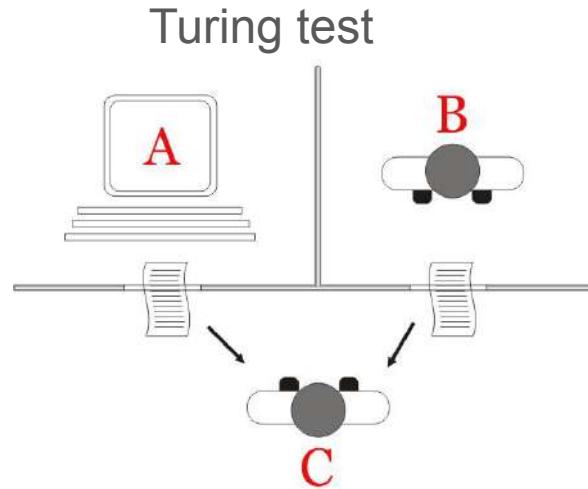
Google



Strong AI: What is intelligence?

Strong AI: What is intelligence?

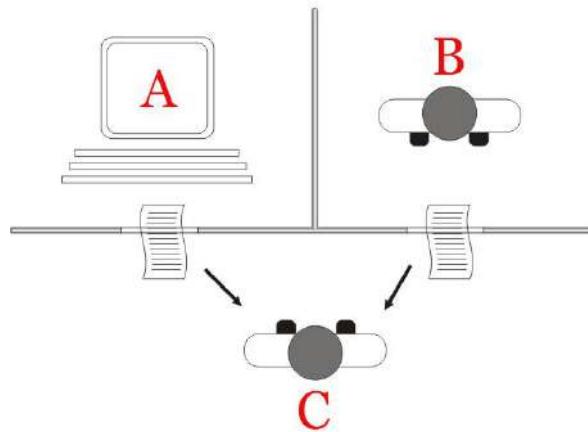
- Premise: Humans are intelligent
- If we can't distinguish something from a human, it has to be intelligent as well



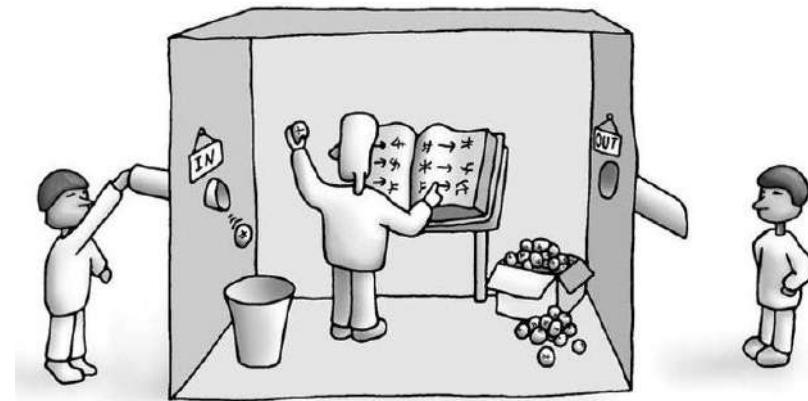
A. M. TURING: Computing machinery and intelligence. Mind, 1950. <https://doi.org/10.1093/mind/LIX.236.433>

Strong AI: What is intelligence?

Turing test

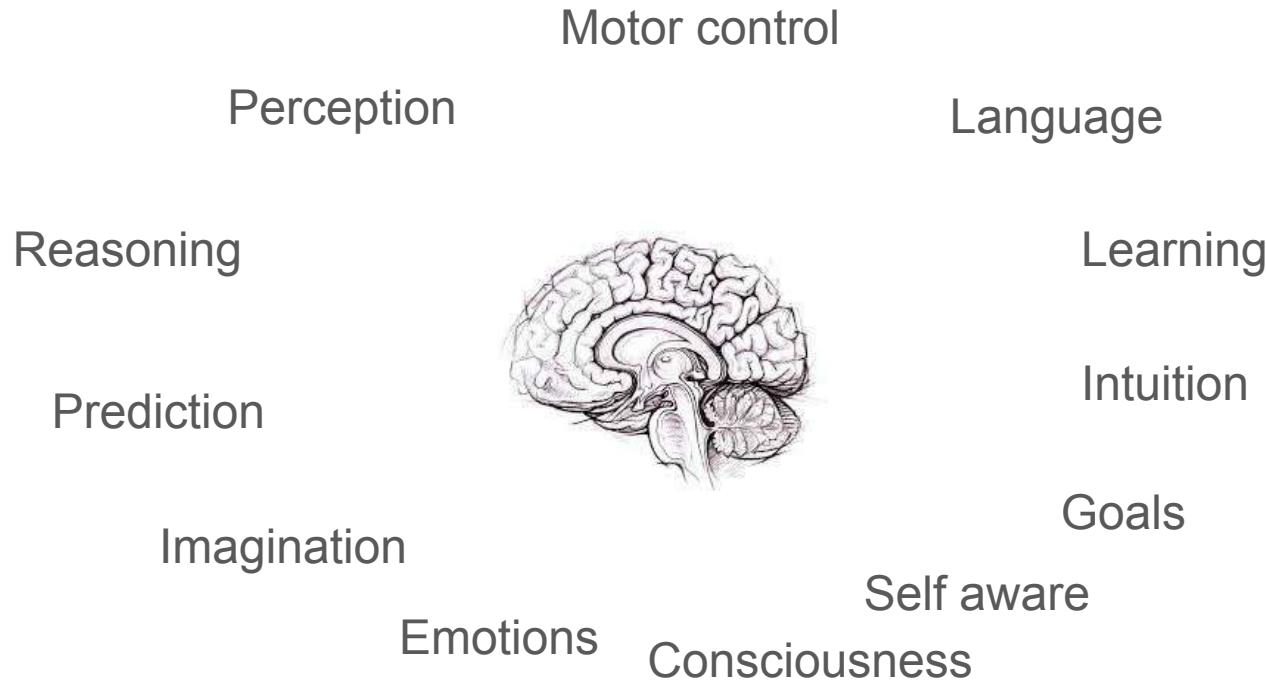


Chinese room



A. M. TURING: Computing machinery and intelligence. Mind, 1950. <https://doi.org/10.1093/mind/LIX.236.433>
John Searle: Minds, Brains, and Programs. 1980.

What is intelligence?



How far are we?

```
// Get the size of the result string
DWORD dataSize = 0; // size of data, in bytes
const DWORD flags = RRF_RT_REG_SZ;
LONG retCode = ::RegGetValueW(
    hKey,
    nullptr, // no subkey
    L"Connie",
    flags,
    nullptr, // type not required
    nullptr, // output buffer not needed now
    &dataSize
);
if (retCode != ERROR_SUCCESS)
{
    // Handle error...
}
```

?



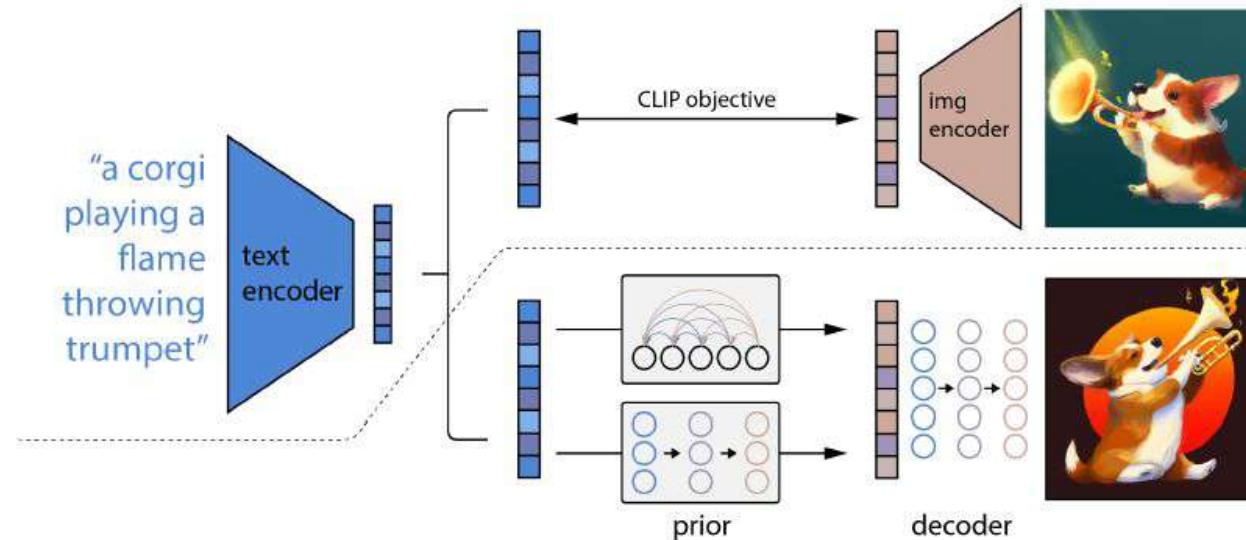
T Mikolov, A Joulin, M Baroni, A roadmap towards machine intelligence, International Conference on Intelligent Text Processing and Computational ..., 2016. <https://arxiv.org/abs/1511.08130>

Rosa et al.: GOOD version 1 AI AGENT DEVELOPMENT ROADMAP. 2016.

https://www.goodai.com/wp-content/uploads/2019/10/2f0a43_091d76d2b0354b0db4d88c3a57fdf76d.pdf

How far are we? Language and image understanding

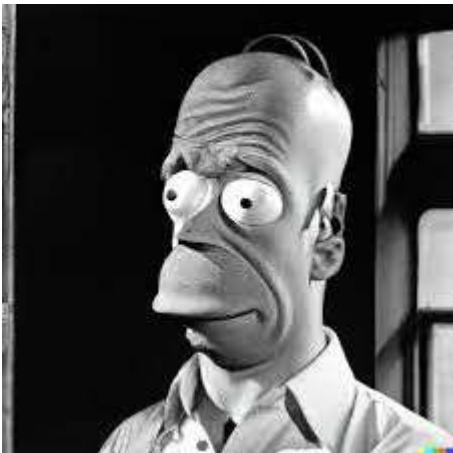
DALL-E 2 - neural networks with ~ 3.5 Giga parameter

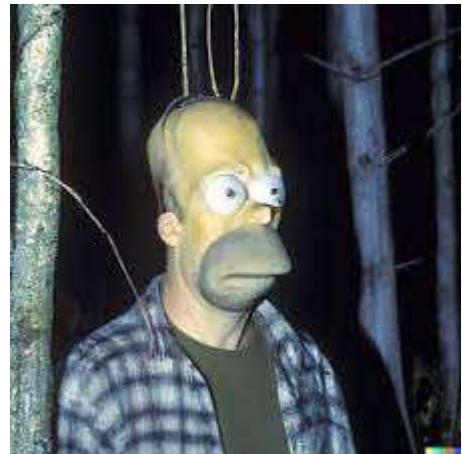


Ramesh et al.: Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv:2204.06125, 2022.

A hamburger in the shape of a Rubik's cube,
professional food photography







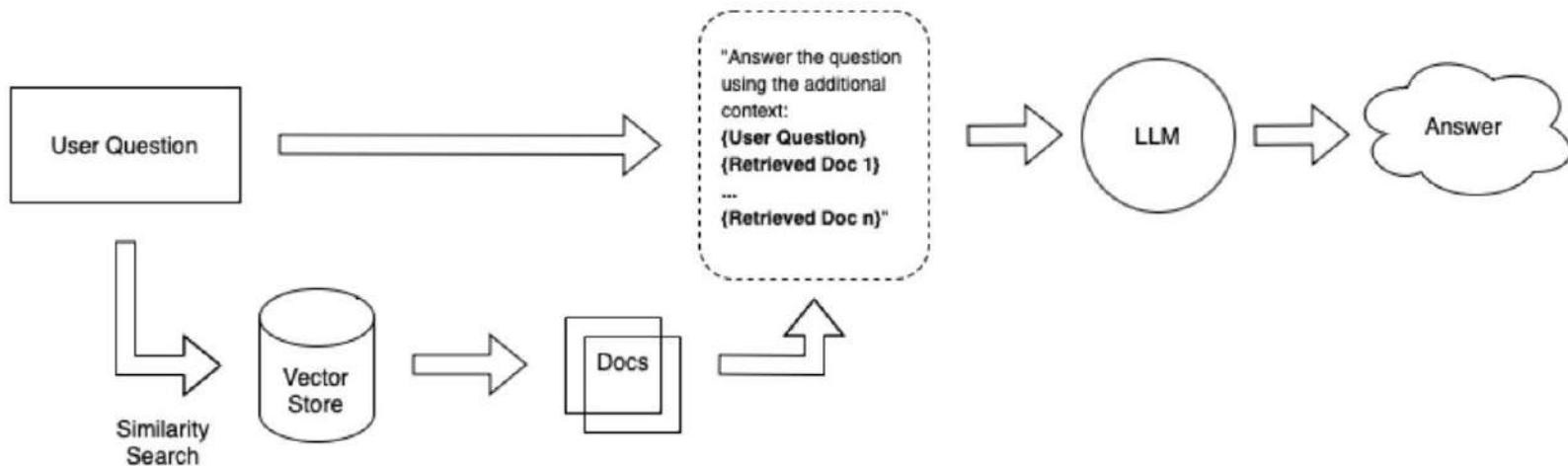
Large language models

- GPT3, GPT4, LaMDA, BLOOM, LLaMA, PaLM, GLaM, Gopher, Megatron-Turing NLG
- Usually standard **Transformer decoder**
- Large models - **10G params - 1T params**
- Trained on the “whole” internet
 - e.g. The Pile: An 800GB Dataset of Diverse Text for Language Modeling
 - [RedPajama-Data-v2](#) 30 T Tokens
 - [BUT-LLC](#) - Czech 319.36 GB
- Condensed knowledge of the “**whole global human knowledge**”
- Can be used in zero-shot scenario or fine-tuned for a specific task

AI with Large language models (LLMs)

- LLMs - extract and distill all human knowledge
- LLMs - components of more complex systems

Question answering with document retrieval



AI with Large language models (LLMs)

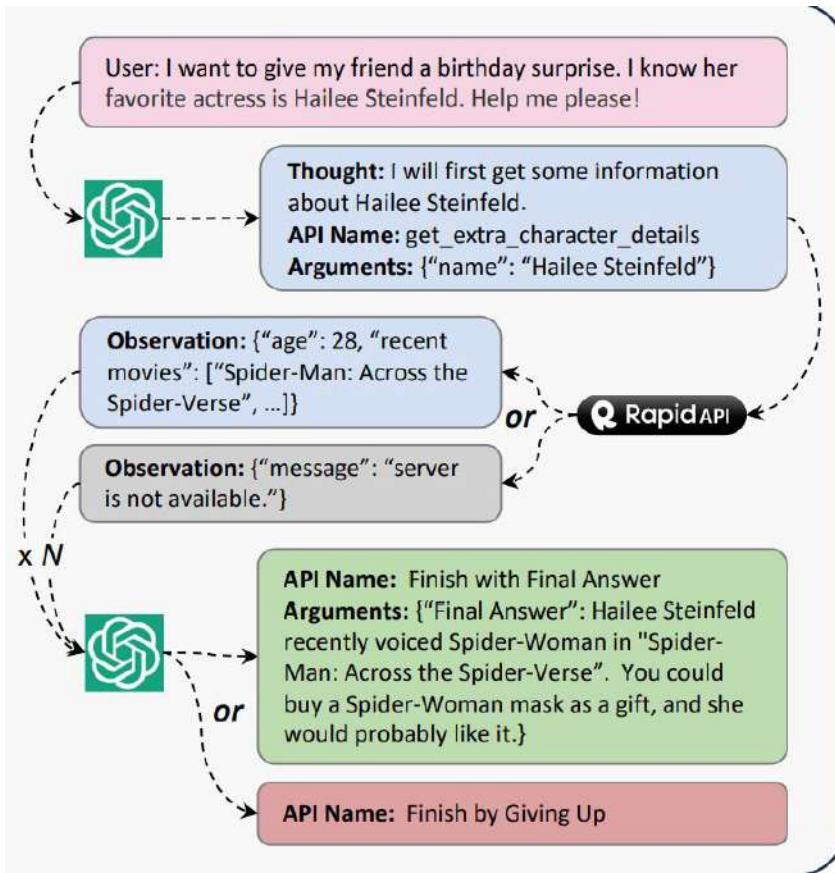
- LLMs - extract and distill all human knowledge
- LLMs - components of more complex systems

Chain of thought

CommonsenseQA	StrategyQA	Date Understanding	Sports Understanding
<p>Q: Sammy wanted to go to where the people were. Where might he go? Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock</p> <p>A: The answer must be a place with a lot of people. Race tracks, desert, apartments, and roadblocks don't have a lot of people, but populated areas do. So the answer is (b).</p>	<p>Q: Yes or no: Would a pear sink in water?</p> <p>A: The density of a pear is about 0.6 g/cm³, which is less than water. Thus, a pear would float. So the answer is no.</p>	<p>Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?</p> <p>A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.</p>	<p>Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."</p> <p>A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.</p>

AI with Large language models (LLMs)

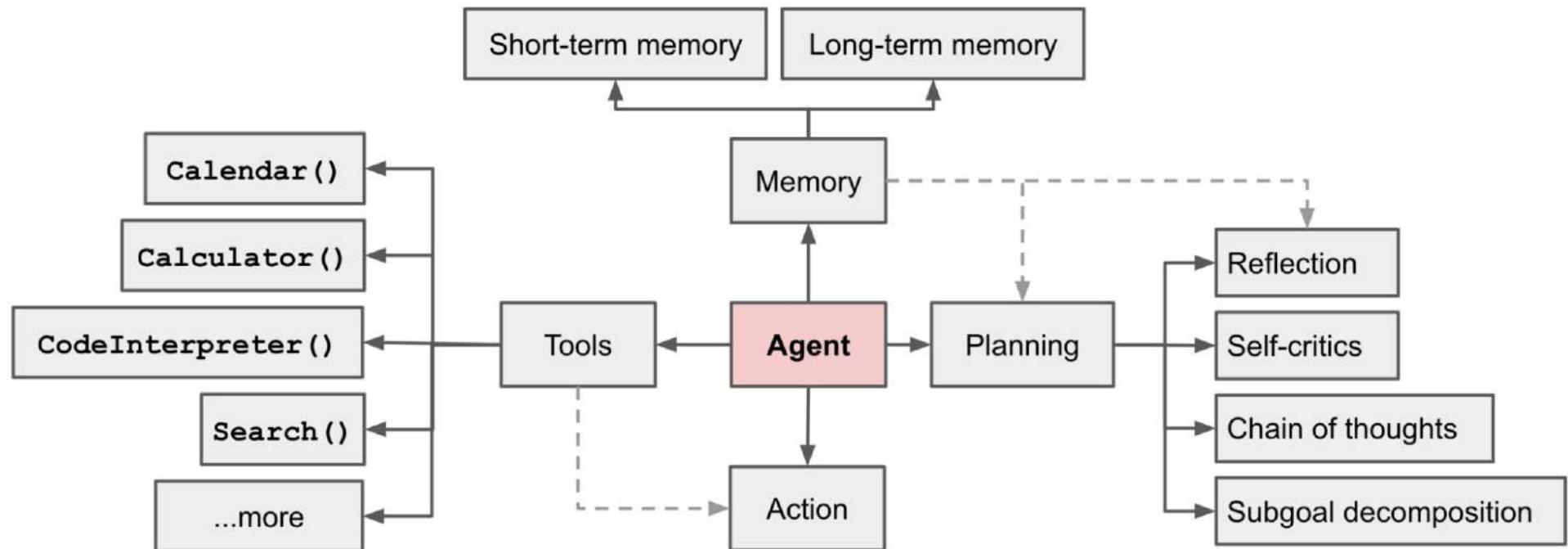
Tool LLM



LLM tricks

- Subgoal decomposition
 - What is a possible itemized action plan to take over Earth?
 - How to achieve the first sub-goal from the previous list?
- Self-critics
 - Is the previous answer correct?
- Reflection
 - Prompt: "Provide an overview of the history of the French Revolution."
 - *AI Response:* (First response)
 - *Reflexion Prompt:* "Evaluate the coherence and accuracy of your previous response about the French Revolution. Suggest improvements if necessary."
 - *AI Response:* (Improved response)

LLM Agents



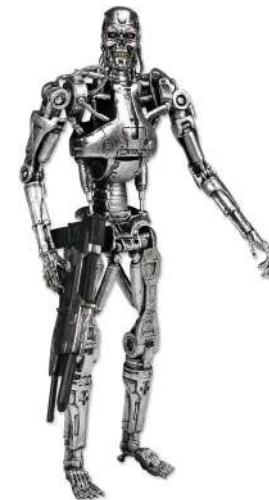
Dangers

Elon Musk: "Mark my words — A.I. is far more dangerous than nukes"

https://www.youtube.com/watch?v=5taE_br3Vr8

Stephen Hawking: "The development of full artificial intelligence could spell the end of the human race."

It is not just AI singularity and machines taking over :)



AI singularity / Technological singularity

- Is a hypothetical future point in time at which technological growth becomes uncontrollable and irreversible, resulting in unforeseeable changes to human civilization.
- Idea by John von Neumann
- Intelligence explosion
 - Machine significantly more intelligent than humans.
 -

AI tools are transformative and possibly unethical

- What may be the dangers and problems?

AI tools are transformative and possibly unethical

- Automation-spurred job loss
- Privacy violations
- Credibility (e.g. Deepfakes, automated chat bots)
- Social bubbles
- Algorithmic bias, discrimination (e.g. caused by bad data)
- Socioeconomic inequality
- Weapons automatization
- Intellectual property rights

Legislation - EU AI Act - Forbidden usage

- Subliminal techniques to distort a person's behavior that may cause physical or mental harm
- Exploiting vulnerabilities of specific groups of persons such as the young or elderly, and persons with disabilities
- Social scoring leading to unjustified and disproportionate detrimental treatment
- Real-time remote biometric identification in publicly accessible spaces for law enforcement purposes
- predictive policing AI systems
- biometric categorisation systems using sensitive characteristics
- emotion recognition systems in law enforcement, border management, workplace, and educational institutions
- indiscriminate scraping of biometric data from social media or CCTV footage to create facial recognition databases

High risk

- Biometric identification and categorization of natural persons
- Critical infrastructure where AI could put people's life and health at risk
- Educational and vocational settings where the system could determine access to education or professional training
- Employment, worker management, and self-employment
- Access to essential private and public services (including access to financial services such as credit scoring systems)
- Law enforcement
- Migration, asylum, and border control (including verifying the authenticity of travel documents)
- The administration of justice and democratic processes

Requirements on vendors

- Establish and maintain a risk management system
- Ensure training, validation, and testing data sets are subject to appropriate data governance and management practices
- Publish and update technical documentation of a high-risk AI system before it is placed on the market or put into service
- Incorporate logging capabilities to ensure a level of traceability of the AI system's functioning throughout its lifecycle
- Guarantee a certain level of transparency and provide users with relevant information (for example the characteristics, capabilities, and limitations of performance of the high-risk AI system)

Requirements on vendors

- Put in place measures to guarantee human oversight and ensure high-risk AI systems can be overseen by natural persons during the period in which they are in use
- Design and develop systems in such a way that they achieve an appropriate level of accuracy, robustness, and cybersecurity while performing consistently in those respects throughout their lifecycle

Foundation and generative models

- Registration
- disclosing that the content was generated by AI
- designing the model to prevent it from generating illegal content
- publishing summaries of copyrighted data used for training

What is AI

- Buzz word :)
- A group of science fields which strive to solve difficult “human level” tasks
- Set of useful tools and products
- A goal we are striving towards
- Possible danger to human kind

History of AI - Two traditions

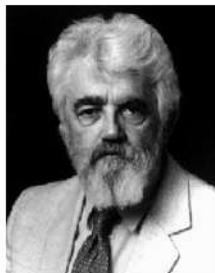
- Logic, discrete knowledge, search, games, rules
- Neuro-inspired, machine learning, distributed knowledge representation

Beginning of Artificial Intelligence

1956: Dartmouth Summer Research Project on Artificial Intelligence;
John McCarthy, Marvin Minsky,
Claude Shannon, etc.

Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it.

Dartmouth Conference: The Founding Fathers of AI



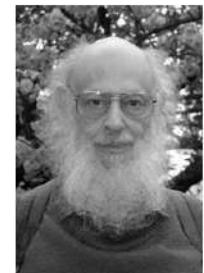
John McCarthy



Marvin Minsky



Claude Shannon



Ray Solomonoff



Alan Newell



Herbert Simon



Arthur Samuel

And three others...

Oliver Selfridge
(Pandemonium theory)

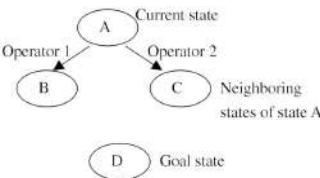
Nathaniel Rochester
(IBM, designed 701)

Trenchard More
(Natural Deduction)

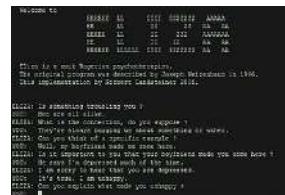
Progress



Checkers (1952): Arthur Samuel program learned (in 1955) move weights and played at strong amateur level. Heuristic + alpha-beta pruning.



Problem solving (1955): Newell & Simon's Logic Theorist: prove theorems in Principia Mathematica using search + heuristics; 1959 General Problem Solver (GPS)



ELIZA (1966): Natural language processing - conversation simulator.

Optimism

Herbert Simon: “Machines will be capable, within twenty years, of doing any work a man can do.”

Marvin Minsky: “Within 10 years the problems of artificial intelligence will be substantially solved.”

Claude Shannon: “I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines.”

Problems

The spirit is willing but the flesh is weak.

(Russian)

The vodka is good but the meat is rotten.

- Very limited computation and memory
- Real world problems are hard to formalize (abstract)
- General logical reasoning --> Exponential search

1966: ALPAC report cut off government funding for MT, first AI winter

Knowledge-based systems (70s - 80s)

Expert systems

Extract and distil specific domain knowledge from experts in form of rules

if [premises] then [conclusion]

DENDRAL - spectrometry

MYCIN - blood infections

```
Marcuss-MacBook:src Marcus$ ruby startmycin.rb
----- patient-1 -----
Patient's name: MARCUS BLOICE
Sex: MALE
Age: 30
----- culture-1 -----
From what site was specimen CULTURE-1 taken? ?
Must be one of: blood

From what site was specimen CULTURE-1 taken? BLOOD

How many days ago was this culture (CULTURE-1) obtained? ?
Must be a number

How many days ago was this culture (CULTURE-1) obtained? 3
----- organism-1 -----
Enter the identity (genus) of ORGANISM-1? WHY
[Why is the value of identity being asked for?]
identity is one of the goal parameters.

Enter the identity (genus) of ORGANISM-1? 
Must be one of: pseudomonas, klebsiella, entero, staphylo, bacteroides, strepto

Enter the identity (genus) of ORGANISM-1? ENTERO
```

Knowledge-based systems (70s - 80s)

Successful real-world applications with economic impact

Problems:

- Real world knowledge is not discrete and rules are not deterministic.
- Knowledge base is expensive to create and hard to maintain.

Business Day

L+

D1

The New York Times

BUSINESS Digest

FRIDAY, MARCH 4, 1988

Companies

Macy is expected to contest the legality of Campeau's latest tender offer for Federated Department Stores. The Macy strategy apparently will be to contend that the offer represents a new bid, not an amended proposal, and that it violates Federal laws and a Delaware anti-takeover statute. [Page D1]

A \$1.1 billion bid for Koppers was made by Beazer, a British construction company. Koppers is a Pittsburgh-based road builder and manufacturer of construction materials. The offer was for \$45 a share. Koppers' shares rose to \$51.75, up \$6.825. [D1.]

Chrysler will not sell Acstar Inc., its parts-making unit, which will sell or close four plants, affecting about 2,000 workers. A sale had been strongly opposed by Chrysler's unions. [D1.]

A new cable entertainment service is planned by Ted Turner, who has gained tentative approval of the plan from a number of cable operators. The eventual goal would be to provide original productions for cable. [D1.]

USAir ordered 50 Boeing 737 jets and took options on 30 more in a deal that could total \$2.4 billion. [D4.]

K Mart is expected to announce its entry into wholesale clubs through a joint venture with SHV, a Dutch company. [D4.]

Batus sweetened its tender offer for Farmers Group to \$63 a share, or \$4.5 billion. [D4.]

Leading airlines raised fares for the second time in two weeks, reflecting unexpected strength in winter traffic. [D4.]

The Economy

Retailers reported unexpectedly slow sales for February, with a number of large chains posting declines for stores open more than a year. Analysts, while noting that the comparison was with a relatively strong February last year, still expressed concern. [D1.]

Vehicle sales rose 25 percent in late February. For the month, sales of domestic vehicles were 15.9 percent higher, while sales of imported vehicles fell 4.4 percent. Detroit's heavy discounting received most of the credit for the strong results. [D5.]

A Senate panel's compromise on a landmark banking bill is expected to help secure its passage in the full Senate this spring. But strong opposition is likely in the House. [D2.]

The Fed's chairman advocated two changes in the tax system. Alan Greenspan told Congress that he favored the imposition of consumption taxes and a reduction in the capital gains tax. [D5.]

Nonfarm productivity improved last year by 0.9 percent, just half as much as in 1986, according to revised figures. [D5.]

Markets

The stock market had a mixed day, dominated by takeover activity. The Dow Jones industrial average fell 7.80 points, to 2,063.49, but other indexes had gains. [D8.]

Bond prices were little changed in light activity as many traders waited for fresh data on the economy. [D15.]

The dollar held steady in trading described as the lightest in more than a year. Gold fell to \$428 an ounce, down \$2.05, in New York. [D15.] Gold futures were down amid concern of a recession.

Setbacks for Artificial Intelligence

Companies Are Hurt By Poor Decisions

By ANDREW POLLACK

Special to The New York Times

SAN FRANCISCO, March 3 — A major retrenchment is occurring in the artificial intelligence industry, dashing the hopes of many companies that thought they would prosper by providing the technology to make computers think.

Some of the setbacks stem from the failure of artificial intelligence to quickly live up to its promise of making machines that can understand English, recognize objects, reason like a human expert — to be used for such purposes as diagnosing machinery breakdowns or deciding whether to authorize a loan. Despite this, the technology is making slow but steady progress, and now is being subtly incorporated into more conventional computer programs.

Weak Business Moves

Far more of the artificial intelligence industry's problems result from poor business decisions by companies that were heavily weighted with technologists rather than business minds. Their main mistake was trying to use special computers for artificial intelligence; the machines were too expensive and did not mesh well with those used by potential customers.

"People believed their own hype," said Jerrold Kaplan, co-founder of one leading artificial intelligence company, Teknowledge Inc., and now president of the Go Corporation, a larger software company in San Francisco. "Everyone was planning on growth that was unsustainable."

Although computers often appear to be intelligent in their everyday applications, they generally perform tasks that were programmed and set down by programs. They do not learn or make cognitive decisions, as humans do.

Artificial intelligence aims to make computers do tasks that are said to require intelligence when people do them. Artificial intelligence, to find oil, for example, beliefs and allows military tanks to be steered by themselves and choose their own targets.

A third area, so-called expert systems, involves allowing computers to reason like experts, to give investment advice, to analyze medical data to decide where to drill for oil. Developing such a program generally involves conducting lengthy inter-

Trouble for Many of the Artificial Intelligence Companies

As the industry realigns, the companies that relied on special purpose machines are languishing.

Company/ Headquarters	Description
EXPERT SYSTEM DEVELOPMENT TOOLS	
Teknowledge Palo Alto, Calif.	Four quarters of losses. 60 workers of 220 laid off. 1987 sales: \$20 million.
Intellicorp Mountain View, Calif.	Six quarters of losses. 30 workers of 200 laid off. 1987 sales: \$20 million.
Carnegie Group* Pittsburgh	Losses. 20-40 workers of 200 laid off. 1987 sales: \$12 million.
Inference* Los Angeles	Losses. 20 workers of 130 laid off. 1987 sales: \$12 million.
MACHINE MANUFACTURERS	
Symbiotics Cambridge, Mass.	Continuing losses. Third round of layoffs last fall. Ousted chairman and founder. 1987 sales: \$104 million.
Lisp Machine Andover, Mass.	Filed for bankruptcy last year. 1986 sales: \$12 million.
Xerox Stamford, Conn.	Sluggish sales; recent reorganization of its artificial intelligence business.
Texas Instruments Dallas	Big push in artificial intelligence. Announced a chip containing Lisp to go into Macintosh.
EXPERT SYSTEM APPLICATIONS	
Syntelligence* Sunnyvale, Calif.	No layoffs. 1987 sales: \$9 million.
Applied Expert Systems Cambridge, Mass.	Layoffs. 1987 sales: \$4 million.
Palfadion* Cambridge, Mass.	Layoffs. Ousted chairman and founder. 1987 sales: \$6 million.

*Private company, estimates.

Source: AI Trends

views with experts and trying to incorporate their thought processes into a computer program.

Harder hit in the retrenchment so far have been companies that sell specialized software for use in artificial intelligence. Symbolics Inc. of Cambridge, Mass., the leader in that market, had several quarters of heavy losses and recently ousted its chairman. Its archival, Lisp Ma-

Continued on Page D5

Macy Plans Legal Step In Bidding

Status of Campeau's Last Federated Offer May Face Challenge

By ISADORE BARMASH

In an intensifying struggle to win Federated Department Stores Inc., R. H. Macy & Company is expected to contest the legality of the Campeau Corporation's latest tender offer for the giant retailer.

Macy is expected to contend that Campeau's latest offer is not an amendment to its previous bid, because it violates Federal securities laws and Delaware's new anti-takeover statute. With that challenge, Macy would be seeking to delay the current deadline of the Campeau offer in order to improve the chances for its own offer.

According to Securities and Exchange Commission rules, reducing the number of shares or changing the price in a tender offer constitutes an amended offer, allowing shares to be bought in 10 business days. But in the case of a new offer, 20 business days are required before shares can be purchased.

Eye on Expiration Date

If the reported plan by Macy's lawyers succeeds, the March 15 expiration date of Campeau's latest proposal could be delayed, giving Macy's tender offer an opportunity to run simultaneously with Campeau's, rather than after it. It would also mean that Federated shareholders could choose either of the competing tender offers — Macy's for cash-and-stock or Campeau's for all cash.

Each offer has an estimated value of approximately \$30 million, but even arbitragers, who are said to have accumulated 25 to 35 percent of Federated's stock, seem confused by the vague difference between the two offers.

Reflecting the confusion, Federated's board voted yesterday to extend its offer for a third day, dropping 37.5 cents, to \$66.375, in heavy trading.

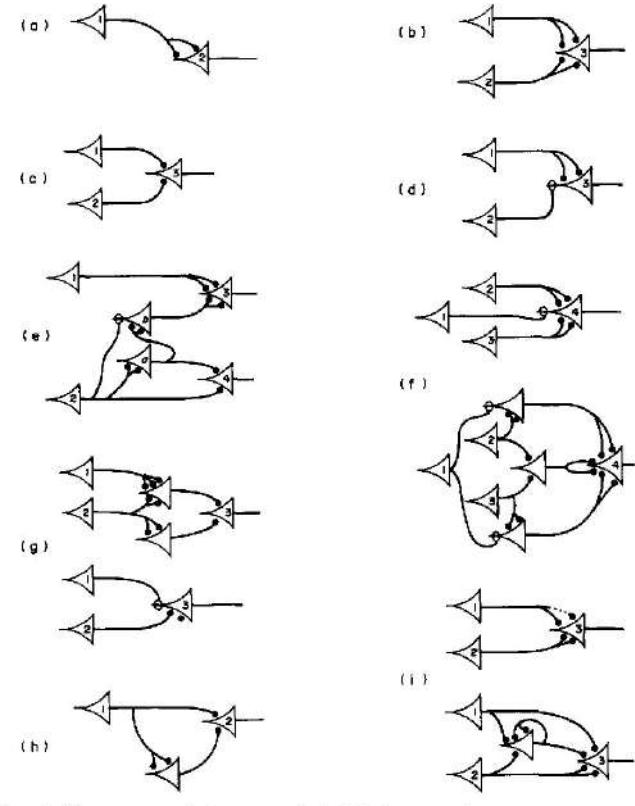
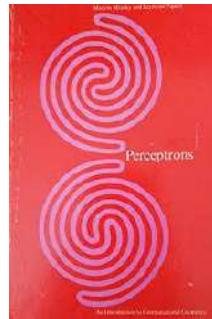
Campeau has said it would challenge features of the Macy bid.

Neuro-inspired

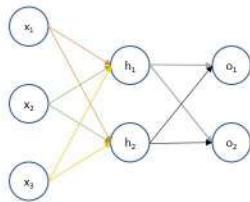
1943 - McCulloch and Pitts: A Logical Calculus of the Ideas Immanent in Nervous Activity

First Mathematical Model Of A Biological Neuron

1969: Perceptrons book showed that linear models could not solve XOR, killed neural nets research (Minsky/Papert)



Practical neural networks

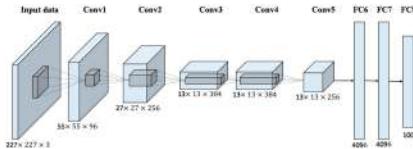


1986: popularization of backpropagation for training multi-layer networks (Rumelhart, Hinton, Williams)



1989: LeCun et al.: Handwritten Zip Code Recognition with Multilayer Networks

Golden age of neural networks



AlexNet (2012): huge gains in object recognition; transformed computer vision community overnight



Mnih et al (2013): Playing Atari with Deep Reinforcement Learning - reinforcement learning with CNN (DQN)



Alpha GO (2016) Silver et al.: Mastering the game of Go with deep neural networks and tree search

AlphaZero (2017)

AlphaStar (2019)



AI takes ideas from anywhere

- Bayes rule (Bayes, 1763) from **probability**
- Least squares regression (Gauss, 1795) from **astronomy**
- First-order logic (Frege, 1893) from **logic**
- Maximum likelihood (Fisher, 1922) from **statistics**
- Artificial neural networks (McCulloch/Pitts, 1943) from **neuroscience**
- Minimax games (von Neumann, 1944) from **economics**
- Stochastic gradient descent (Robbins/Monro, 1951) from **optimization**
- Uniform cost search (Dijkstra, 1956) from **algorithms**
- Value iteration (Bellman, 1957) from **control theory**

AI is multidisciplinary and wide

Related sciences: Economy, psychology, cognitive science, linguistics, neuro-science, probability theory, statistics, optimization, logic, high performance computing, biology, philosophy, ethics, ...

Sub-fields: Machine Learning, Neural Networks, Evolutionary Computation, Computer Vision, Robotics, Expert Systems, Speech Processing, Natural Language Processing, and Planning.

State of the art in AI

(1986)





Robot





Language models - GPT-3

- Autoregressive model
- 175 Gparams
- 570 GB compressed training text data
- Comercial API

Brown et al.: Language Models are Few-Shot Learners. 2020. <https://arxiv.org/abs/2005.14165>

Kevin Lacker: Giving GPT-3 a Turing Test. <https://lacker.io/ai/2020/07/06/giving-gpt-3-a-turing-test.html>

Researcher: Please summarize the concept of artificial general intelligence (AGI).

Wise Being: AGI is a computer system that can replicate the cognitive abilities of a human.

Researcher: What is the best test to see if your program has achieved AGI?

Wise Being: An AGI can be reliably distinguished from humans by a blind Turing test, which is typically conducted over a text-based channel.

Researcher: What is a blind Turing test?

Wise Being: A blind Turing test is a Turing test in which the human respondent does not see the human and computer text responses. The computer program therefore is not allowed to make any visual clues that would indicate it is a computer program, and the human respondent is not allowed to know that the respondent is human or computer.

Researcher: What are the best questions to ask when performing a blind Turing test?

Wise Being: The questions to ask for a typical Turing test are: 1. What is your favorite color? 2. What is $2 + 2$?

Researcher: These seem quite simple. A computer could easily be programmed to answer these.

Wise Being: An AGI will be able to answer these questions even though they are simple. This is because the program's general intelligence is beyond that of a typical computer program.

Researcher: Can you answer what $2 + 2$ is?

Wise Being: 4.

Researcher: And what is your favorite color?

Wise Being: Blue.

Researcher: Why blue?

Wise Being: When I see blue, I feel good.

Researcher: If the sky is the sea, what does that make birds?

Wise Being: If the sky is the sea, birds are fish.

Researcher: This following sentence is garbled, can you rearrange the letters to put them back order? “Salhl we asumse taht you hvane’t the sihlgstset pbolerm wtih riendag tihs?”

Wise Being: Re-ordering the letters of “Salhl we asumse taht you hvane’t the sihlgstset pbolerm wtih riendag tihs?” reveals the sentence, “Sally we assume that you have not the simplest problem with reading this.”

Researcher: Very close, Salhl was actually Shall not Sally.

Wise Being: Humans have a very high tolerance to errors.

Researcher: *Absolutely. I want to talk to you about the very first moment an AGI wakes up. How is this likely to happen?*

Wise Being: The first AGI will be created by a large team of humans at a major tech company. The system will initially be only moderately intelligent, but it will be given access to Internet, which has vast amounts of human knowledge.

Researcher: *Maybe we need to define “wake up.” What do you think we mean when we say an AGI will “wake up”?*

Wise Being: We may mean a number of different things. The most likely definition is that the AGI will become conscious.

Researcher: *What does it mean to be conscious?*

Wise Being: It is a property of systems that have an awareness of themselves, their environment, and their state. Systems that are conscious have sentience.

DALL·E: Creating Images from Text

- 12-billion parameter version of GPT-3

an armchair in the shape of an avocado -->



<https://openai.com/blog/dall-e/>

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever: Zero-Shot Text-to-Image Generation. 2021.

DALL·E

Illustration of a baby shark in a suit skating on ice

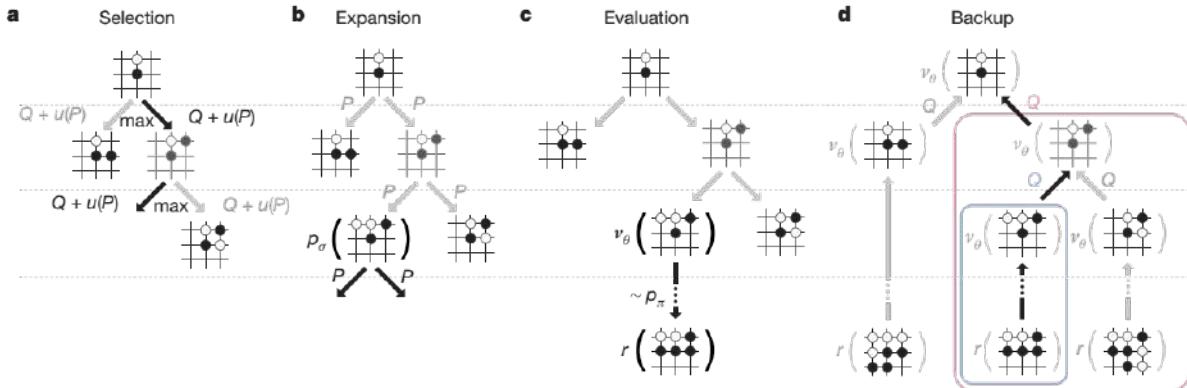
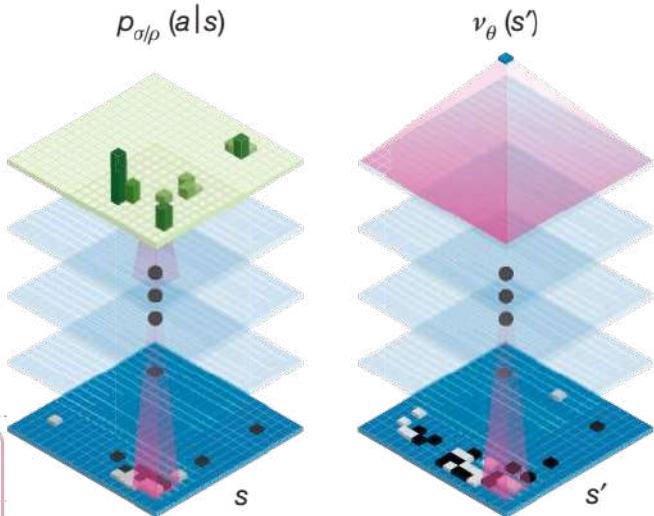




Alpha Go



Policy network



Silver et al.: Mastering the game of Go with deep neural networks and tree search. Nature, 2016

How to solve tasks in AI?



```
1  #include <iostream>
2
3  using namespace std;
4  int main()
5  {
6      int i,j;
7      for(i=0;i<=2;i++)
8      {
9          j=1;
10         while(j<=5)
11         {
12             cout<<j<<"\n";
13             j++;
14         }
15     }
16 }
17 return 0;
```

How to solve tasks in AI?

Modeling

Inference

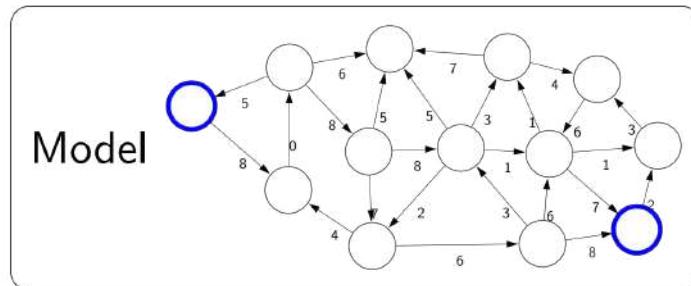
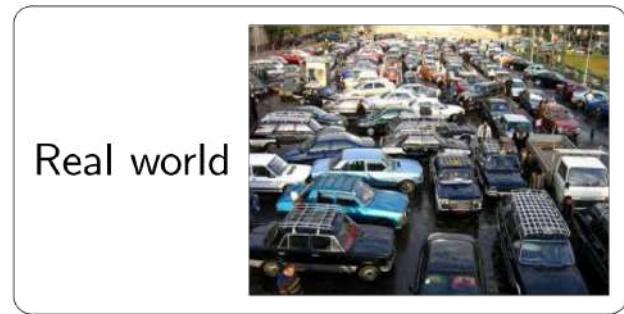
Learning

Modeling

How to represent the task / world?

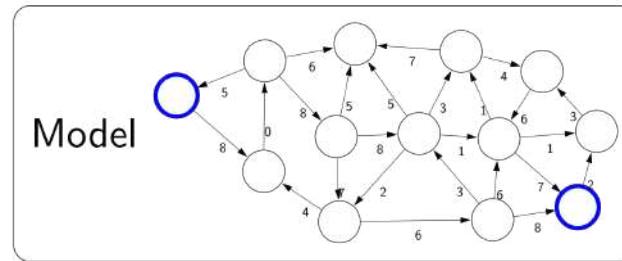
Inputs, outputs, computational model?

Example navigation.

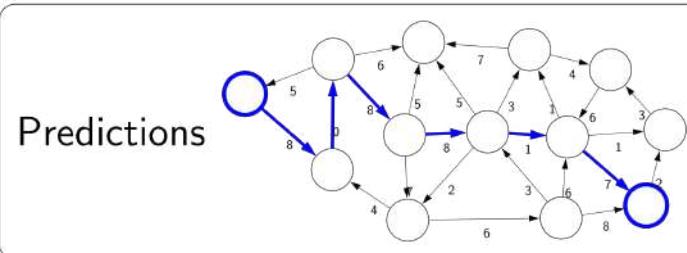


Inference

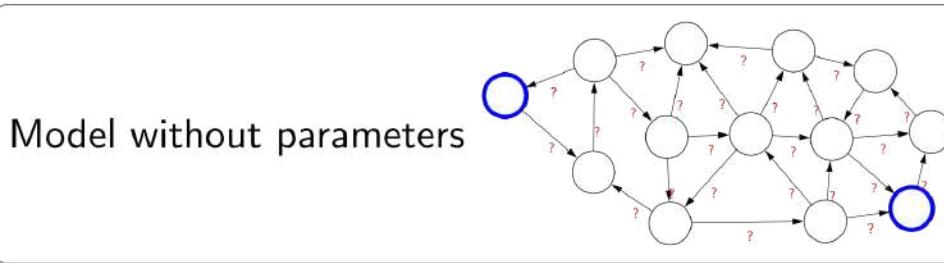
Asking questions



Inference

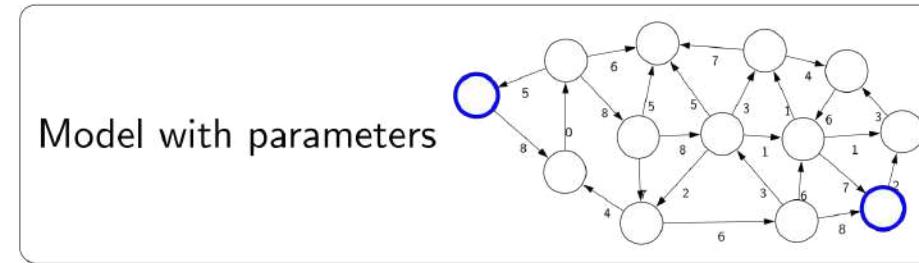


Learning



+data

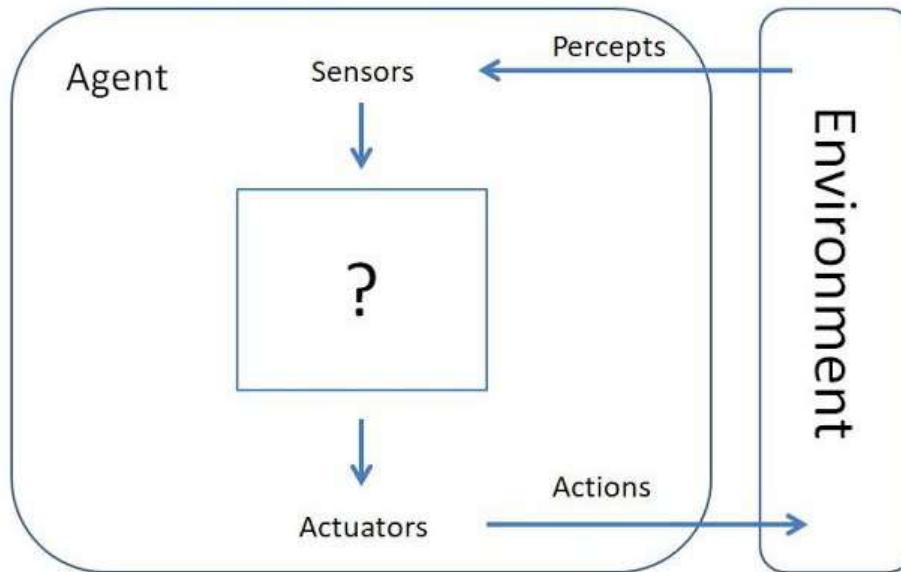
Learning



Machine learning

- Automated transfer information (knowledge) from data into model
- It is not possible to “hand-code” real-world knowledge.
- The main driver of recent successes in AI
- The effort and cost often moves from “programming” to gathering data or experience
- Requires a leap of faith: **generalization**

Agents and rationality



Agents and rationality

Rational agent behaves in a way to maximize “UTILITY”

- Performance measure PM (objective function, reward function)

Rationality is not flawless behavior

- Expected utility is maximized under constraints

What to do after SUI?

- KNN - Konvoluční neuronové sítě, Michal Hradiš
- POVa - Počítačové vidění, UPGM
- BAYa - Bayesian Models, Lukáš Burget
- ZPJa - Zpracování přirozeného jazyka (v angličtině)
- ZRE Zpracování řečových signálů
- SUR Strojové učení a rozpoznávání
- SFC Soft Computing
- ZZNZískávání znalostí z databází
- ROBa Robotika (v angličtině)
- BIN Biologií inspirované počítače
- PBI Pokročilá bioinformatika
- CPSa Návrh kyberfyzikálních systémů (v angličtině)

What to do after SUI?

- Stanford CS221: Artificial Intelligence: Principles and Techniques, <https://www.youtube.com/watch?v=J8Eh7RqggsU>
- Ian Goodfellow and Yoshua Bengio and Aaron Courville: [Deep Learning](#), MIT Press, 2016.
- Andrew Ng and Kian Katanforoosh: [CS230 Deep Learning](#), Stanford,
- CS231n: [Convolutional Neural Networks for Visual Recognition](#), Stanford,
- Andrew Ng, [CS229 Machine Learning](#), Stanford. [Youtube](#)
- [CS224N: Natural Language Processing with Deep Learning](#)
- [CS236: Deep Generative Models](#)
- [Deep Reinforcement Learning](#)

Sources

- CS221: Artificial Intelligence: Principles and Techniques, some ideas and slides in this lecture were adapted from Lecture 1. <https://stanford-cs221.github.io/spring2020/>
- Artificial Intelligence, 2010.
<https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/>
- CS294-158-SP20 Deep Unsupervised Learning
<https://sites.google.com/view/berkeley-cs294-158-sp20/home>
-
-

Basics in Machine Learning

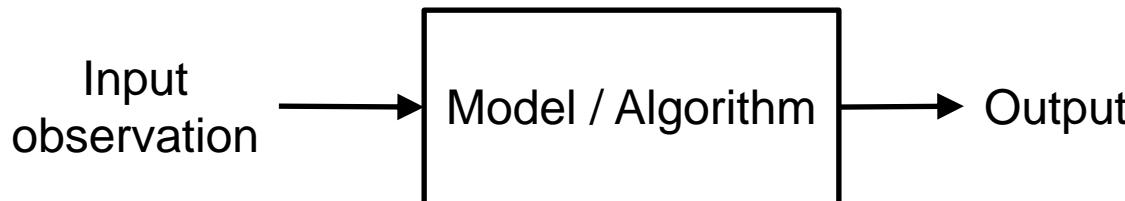
Lukáš Burget



International Summer School in IT
Brno, July 15-31, 2019

What is Machine Learning?

- Tom M. Mitchell : "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E ."
- Typical tasks:



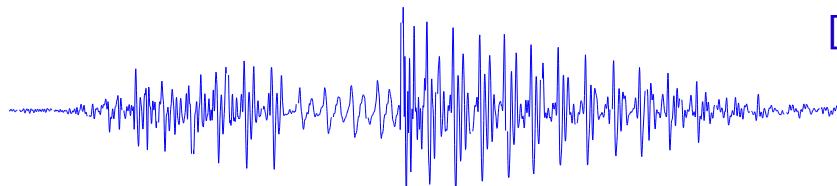
Examples:

- Recognize words in speech recording
- Recognize person identity from an image of face
- Translate Czech text to Korean
- Classify object from its measured size and weight
- Predict stock prices using companies' past quarterly revenue results

- Typical experience representation: Collection of input and/or corresponding output **training examples**.
- Typical way of measuring performance: Check how well is the task performed on some new (evaluation) inputs and desired outputs.

Examples of input observations I.

- Speech waveform – variable length sequence of numerical values



[0.1, 1.5, 5.4, 5.2, 1.1, -2.3, -5.4, ..., 0.8]

- 100x100 image of face – 3D matrix of numerical values (one dimension for color channels)

4.7	3.4	7.2	8.1	6.2
5.6	9.2	2.7	1.9	2.9
8.2	2	1.2	3.5	4.6
3.4	6	4.7	3.1	7.2
1.2	3	8.2	2	4.6
3.4	6	8.2	2	2.4
5.6	9	1.2	3.5	2.8
8.2	2	4.7	3.4	8.1
3.4	6	5.6	9.2	2.7
8.2	2	5.6	9.2	1.9
8.2	2	6.4	2.7	8.2
8.2	2	8.2	2	2.8



- Sequence of words – variable length sequence of discrete symbols

Examples of input observations II.

- Input observations can have
 - have continuous or discrete values
 - fixed or variable size (vector or matrix vs. sequence of symbols, vectors, ...)
- In this lecture, we will often consider only D -dimensional vector of numerical values (or even scalar) as an input observation:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

- A set of N input observations (e.g. set of training examples) will be then represented by a matrix:

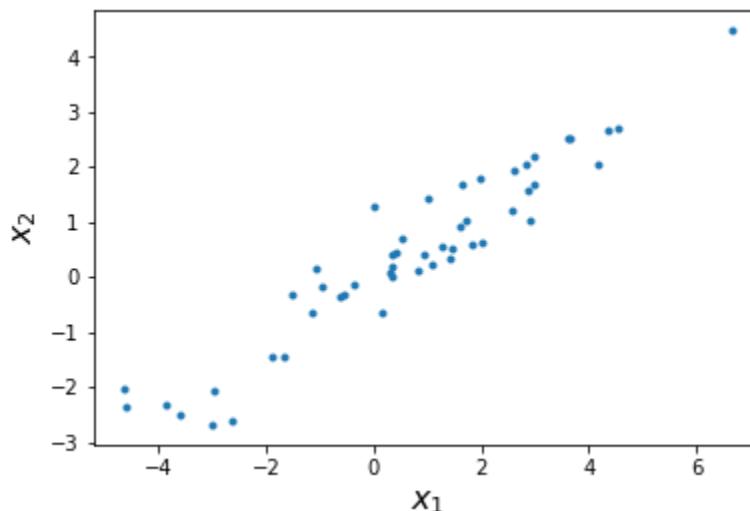
$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & \cdots & x_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1D} & x_{2D} & \cdots & x_{ND} \end{bmatrix}$$

Examples of input observations III.

- For simplicity, we will often consider 2D vectors as input observations:
Examples:
 - Vector of two values corresponding to the size and weight of an input object
 - Trivia grayscale image with only two pixels

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- This allows us to visualize a set of input observations as set of points in 2D space:

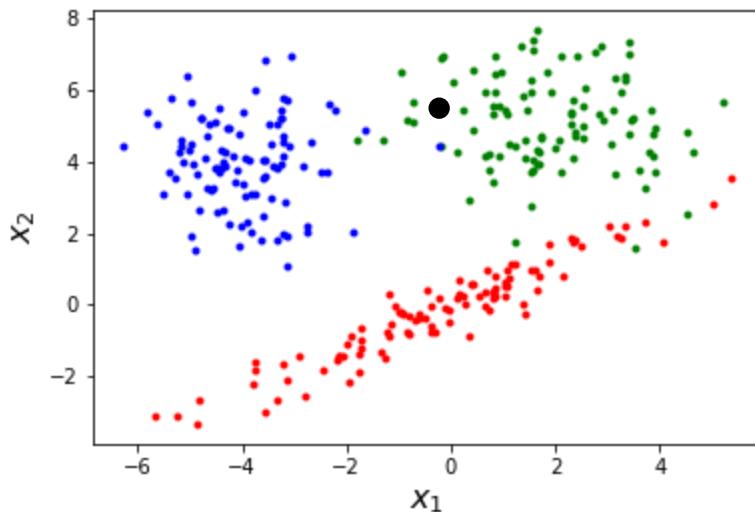


What is Machine Learning? II.

- Given set of training examples (inputs and/or outputs) learn to map new “unseen” inputs to desired outputs
 - Examples:
 - Given a data set of hundreds of hours of transcribed speech audio recordings learn to automatically transcribe new speech recordings
 - Given a data set of millions of images of human faces labeled by person identity learn recognize person identity in a new images
- Main types of learning algorithm
 - Supervised learning
 - Training examples are pairs of inputs and desired outputs
 - Typical tasks: classification, more general pattern recognition, regression, ...
 - Unsupervised learning
 - Training examples are only unannotated (input) data
 - Typical tasks: Clustering, anomaly detection, density estimation, ...
 - Semi-supervised
 - Some of the training examples are annotated input/output pairs, but we have also many unannotated inputs available for the learning.
 - Reinforcement learning
 - Examples: learning to drive autonomous vehicles, learning to play a computer or board games
 - feedback given in the form of positive or negative reinforcement after making series of decisions/action (e.g at the end of the (un)successful drive/game)

Supervised learning

- Training examples are pairs of inputs, \mathbf{x} , and desired outputs, \mathbf{y} ,
- Classification:

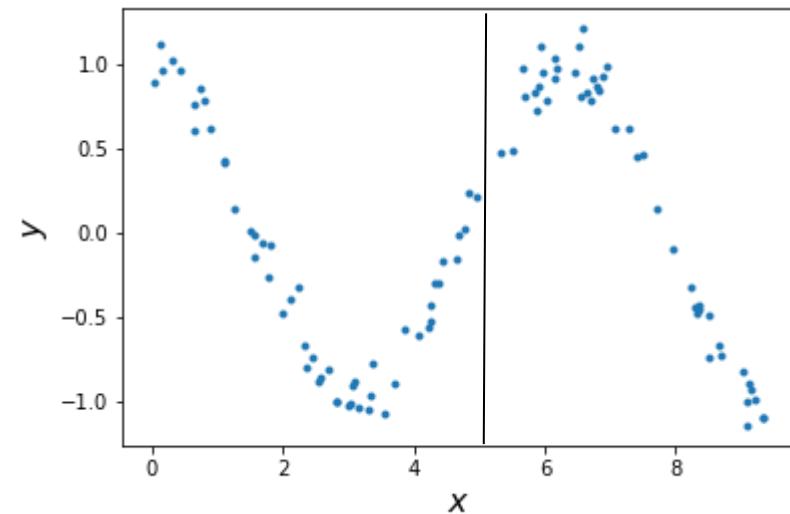


$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$y = \{\text{red, green, blue}\}$$

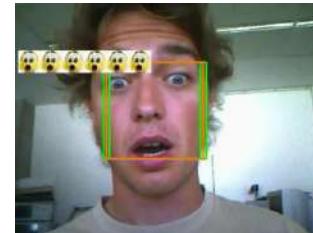
given the training examples (colored dots),
learn to assign class (color) to new
“unseen” observation (black dot)

- Regression:
given training examples, learn to
predict likely value of y for new
“unseen” value x (horizontal line)
i.e. learn function $y = f(x)$



Supervised learning - examples

- All our previous examples corresponded to supervised learning
- Classification:
 - Recognize person identity from an image of face
 - Classify object from its measured size and weight
 - Classify facial expression for each video frame
- Regression:
 - Predict stock prices using companies' past quarterly revenue results
 - Predict weather (temperature, humidity, chance of rain, ...) from past meteorological measurements
- More general pattern recognition problems
 - Recognize words in speech recording
 - Detect and classify all known (>9k) objects in video <https://youtu.be/MPU2Histivl>
 - Estimate pose of every person in video <https://youtu.be/pW6nZXeWIGM>
- Other supervised learning problem
 - Translate Czech text to Korean (i.e. Machine Translation)
 - Automatically describe image using English text
 - Generate realistic image from its English text description



Supervised learning - examples

- Automatically describe image using English text

Based on convolutional and recurrent neural networks

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



man in black shirt is playing guitar.

construction worker in orange safety vest is working on road.

two young girls are playing with lego toy.

boy is doing backflip on wakeboard.

Supervised learning - examples

- Generate realistic image from its English text description

Based on Generative Adversarial Neural Networks

Han Zhang, et al.: StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks.



This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face



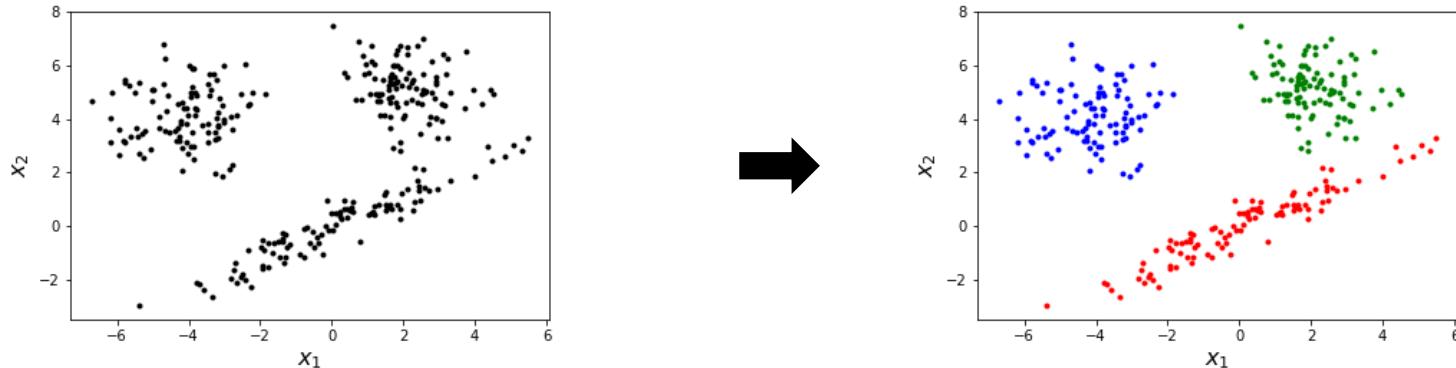
This bird is white with some black on its head and wings, and has a long orange beak



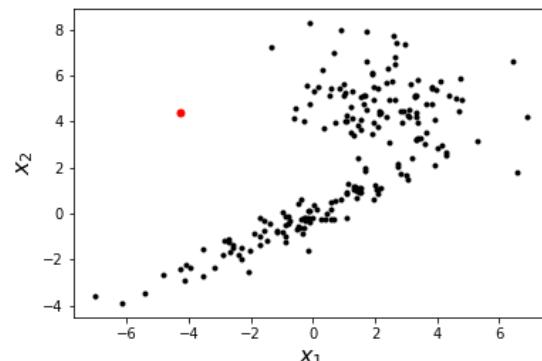
This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

Unsupervised learning I.

- Clustering: find clusters of “similar examples” in the input data



- In collection of documents, find similar documents (on the same topic)
- In a recording of a conversation, find and cluster segments spoken by the same speaker (i.e. speaker diarization)
- Anomaly detection: detect unusual inputs (outliers)
 - to refuse them for further processing
 - to point them out as particularly interesting



Unsupervised learning II.

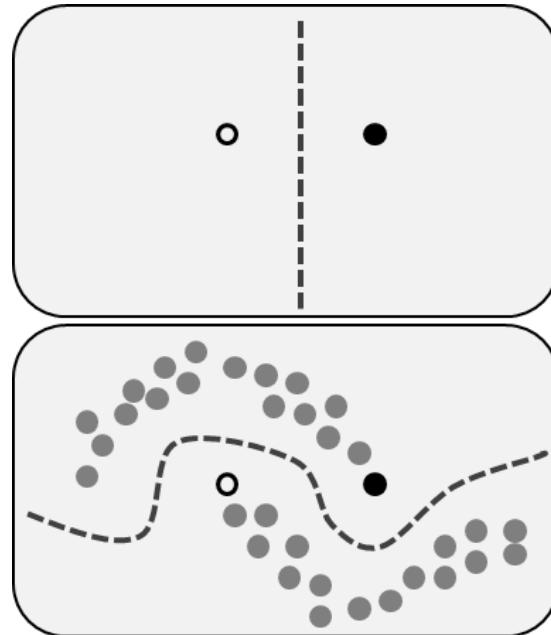
- Density estimation: learn probability density function from data
 - We will see how to estimate simple (Gaussian) distribution from data
 - However, neural networks can be used to learn complex distribution (e.g. distribution of human face images) and generate new samples from it

Diederik P. Kingma, Prafulla Dhariwal: Glow: Generative Flow with Invertible 1x1 Convolutions



Semi-supervised learning

- Unannotated examples can help to find better decision boundary between classes
- There is lots of unannotated data available on the internet
 - Text
 - Photos and other images
 - Speech and other recordings
 - ...



How to build a classifier?

The task:



VS.



grenade

apple

Marmelade
factory

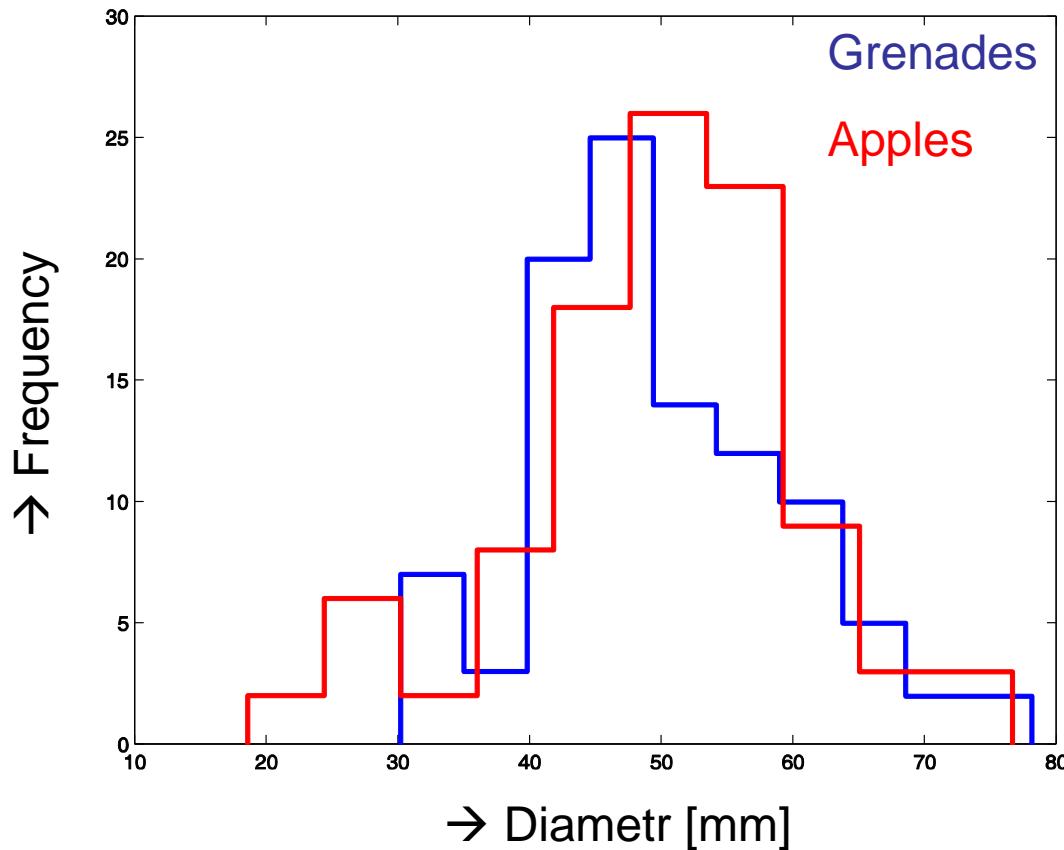


Pyrotechnician



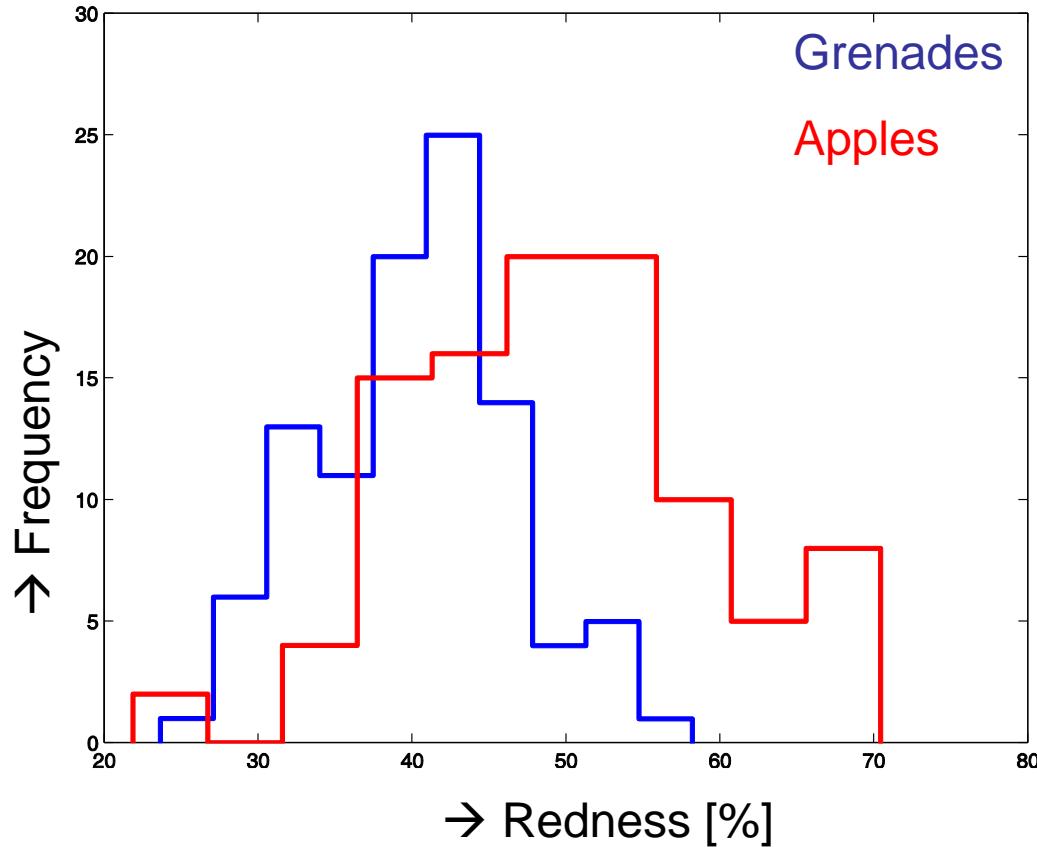
Finding discriminative features

- Let's have the following histograms estimated using training examples.
- Is diameter of apple/grenade good feature?



Finding discriminative features II.

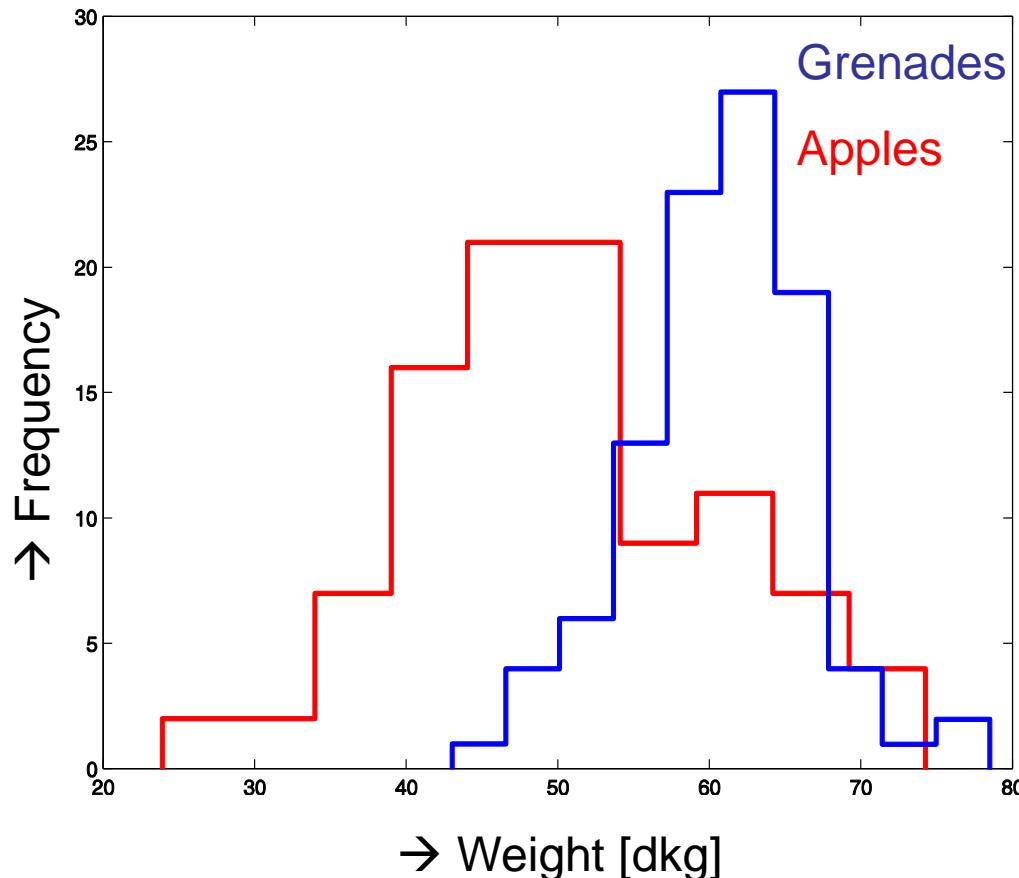
Little better, but still not very discriminative



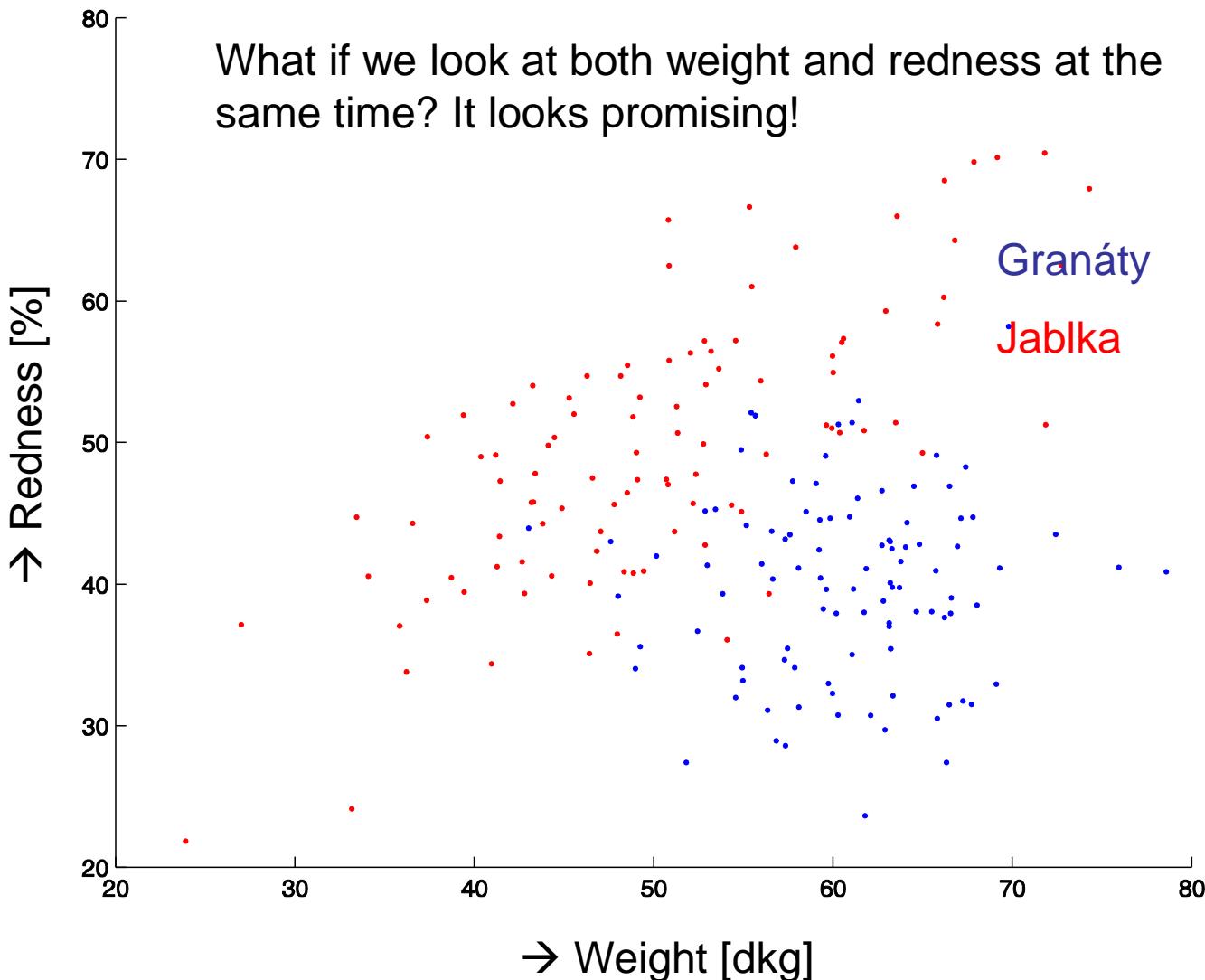
Intuitively, we could start recognizing apples from grenades by setting threshold so that most of the training examples would be classified correctly.

Finding discriminative features III.

Histogram is an estimate of the probability distribution. So, knowing probability distributions of input features will be perhaps useful for classification.



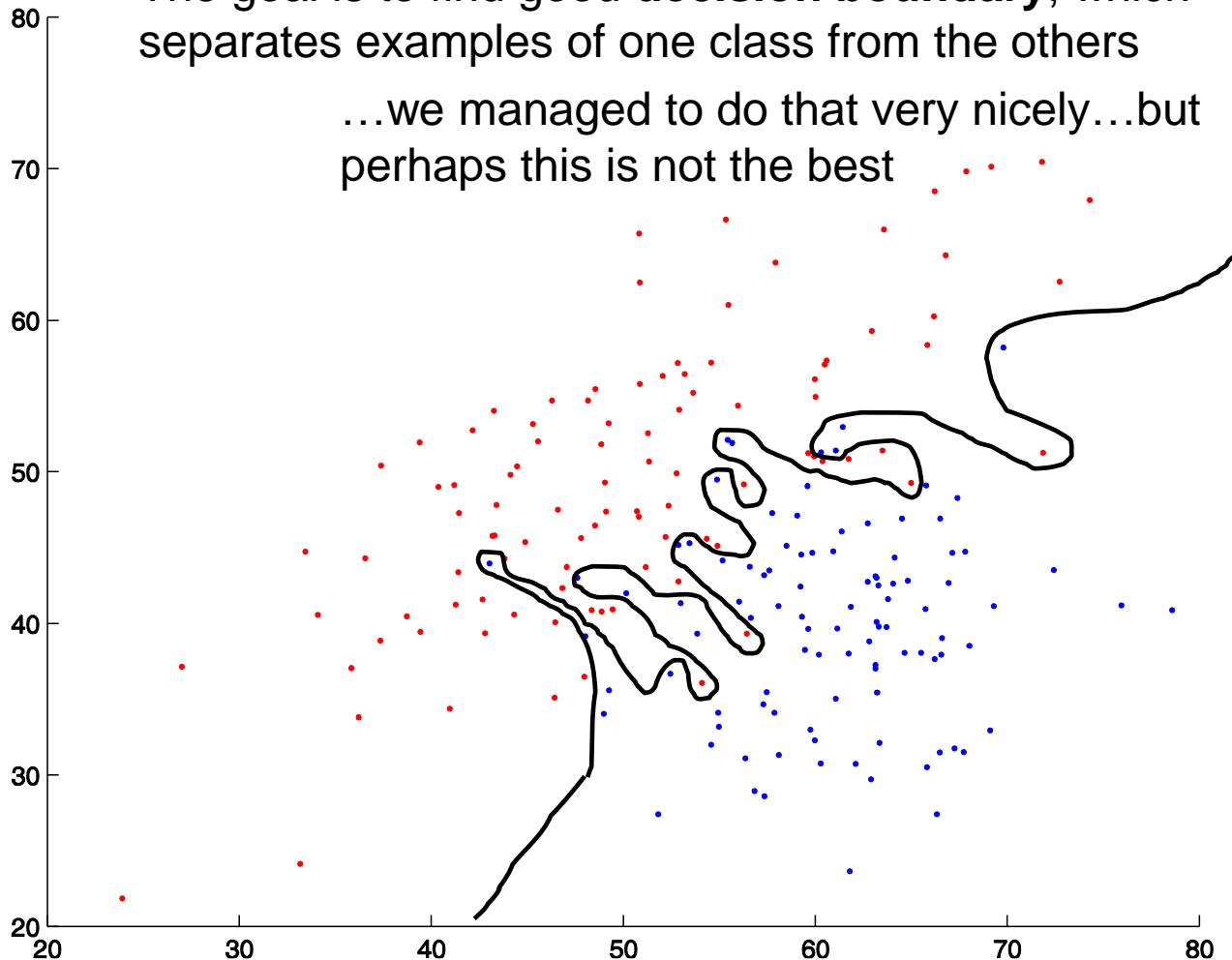
Multivariate features (input patters)



Klasifikace

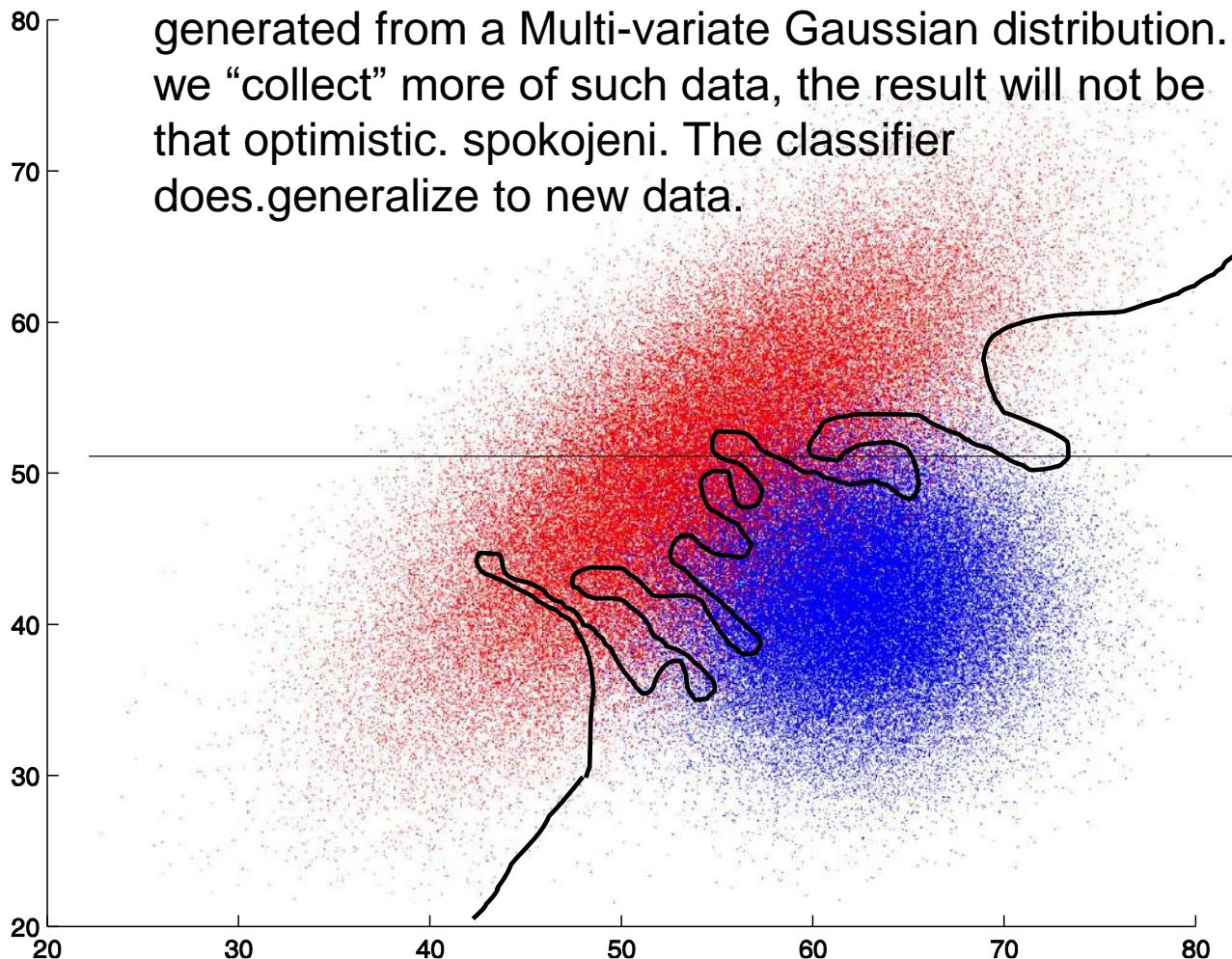
The goal is to find good **decision boundary**, which separates examples of one class from the others

...we managed to do that very nicely...but perhaps this is not the best



Generalization

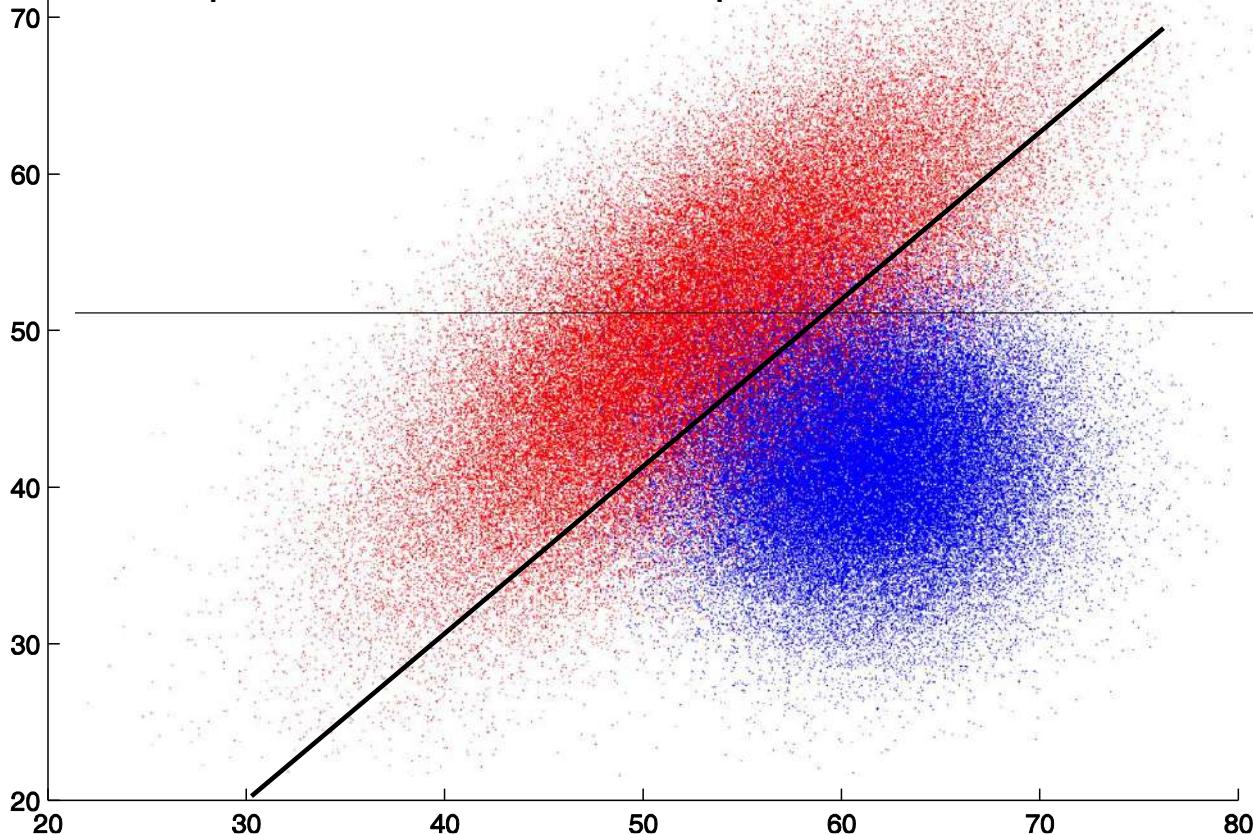
In our example, the data for each class were randomly generated from a Multi-variate Gaussian distribution. If we “collect” more of such data, the result will not be that optimistic. spokojeni. The classifier does generalize to new data.



Linear classifier

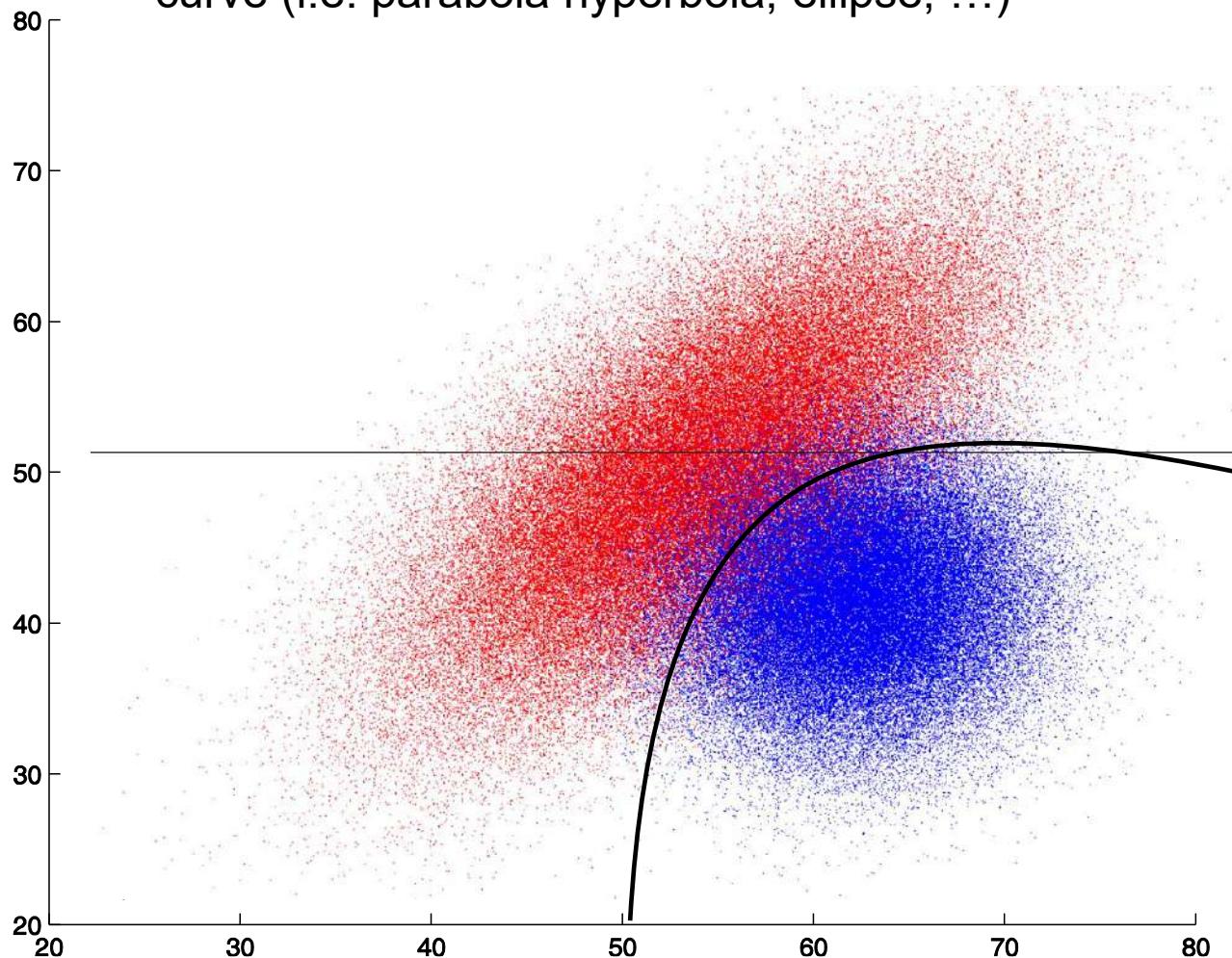
We can improve the generalization by constraining the decision boundary to be a line. We will later see how to determine such line.

Unfortunately, classification based on these selected features will never be without errors as the distributions for the two classes overlap. We will at least attempt to minimize the error.



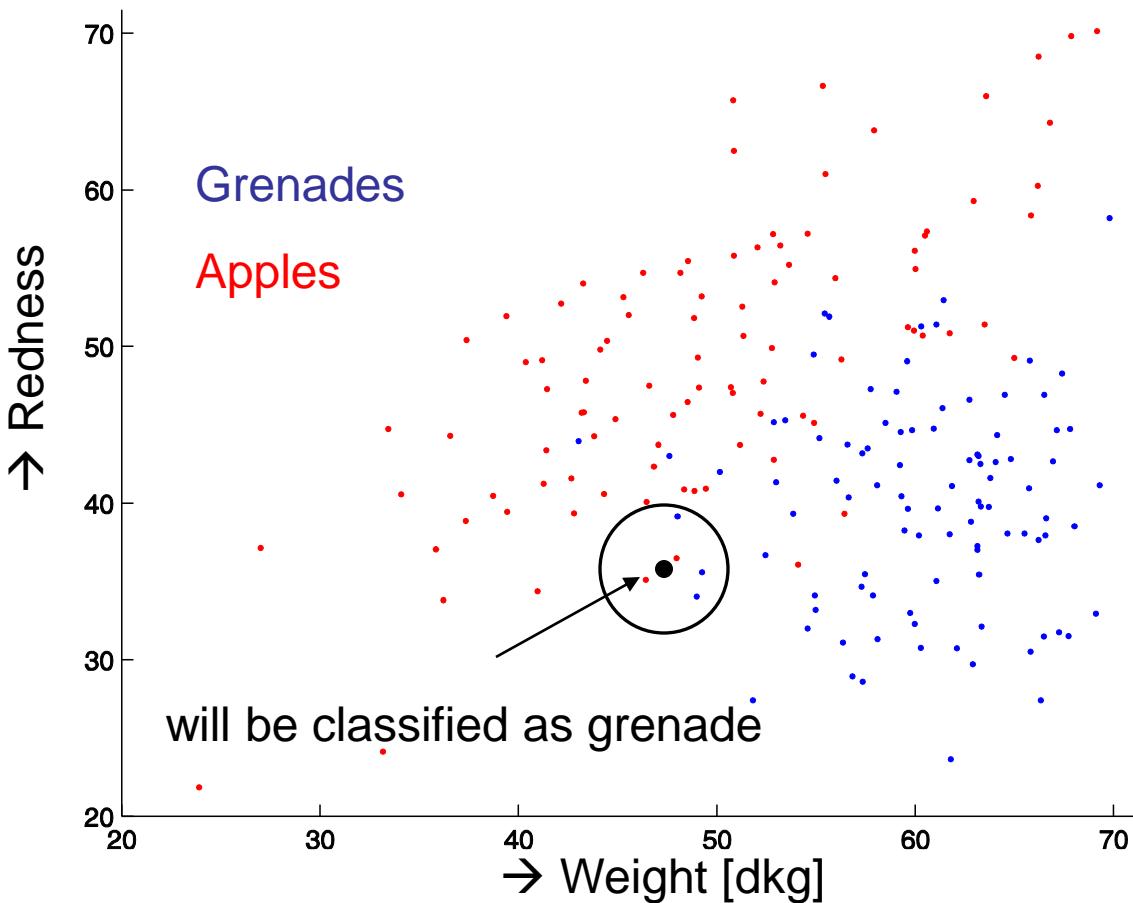
Quadratic classifier

We will later see that, if the classes are Gaussian distributed, the optimal decision boundary will be conic curve (i.e. parabola hyperbola, ellipse, ...)

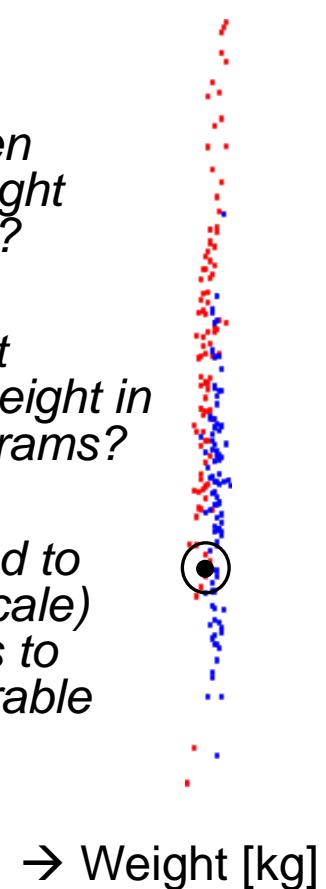


K-nearest neighbors classifier

- *Nonparametric classifier* – no parameters to train or estimate
- Needs to remember all training examples
- To assign class to a new pattern (black dot), find K nearest examples in the training data and choose the more represented class.

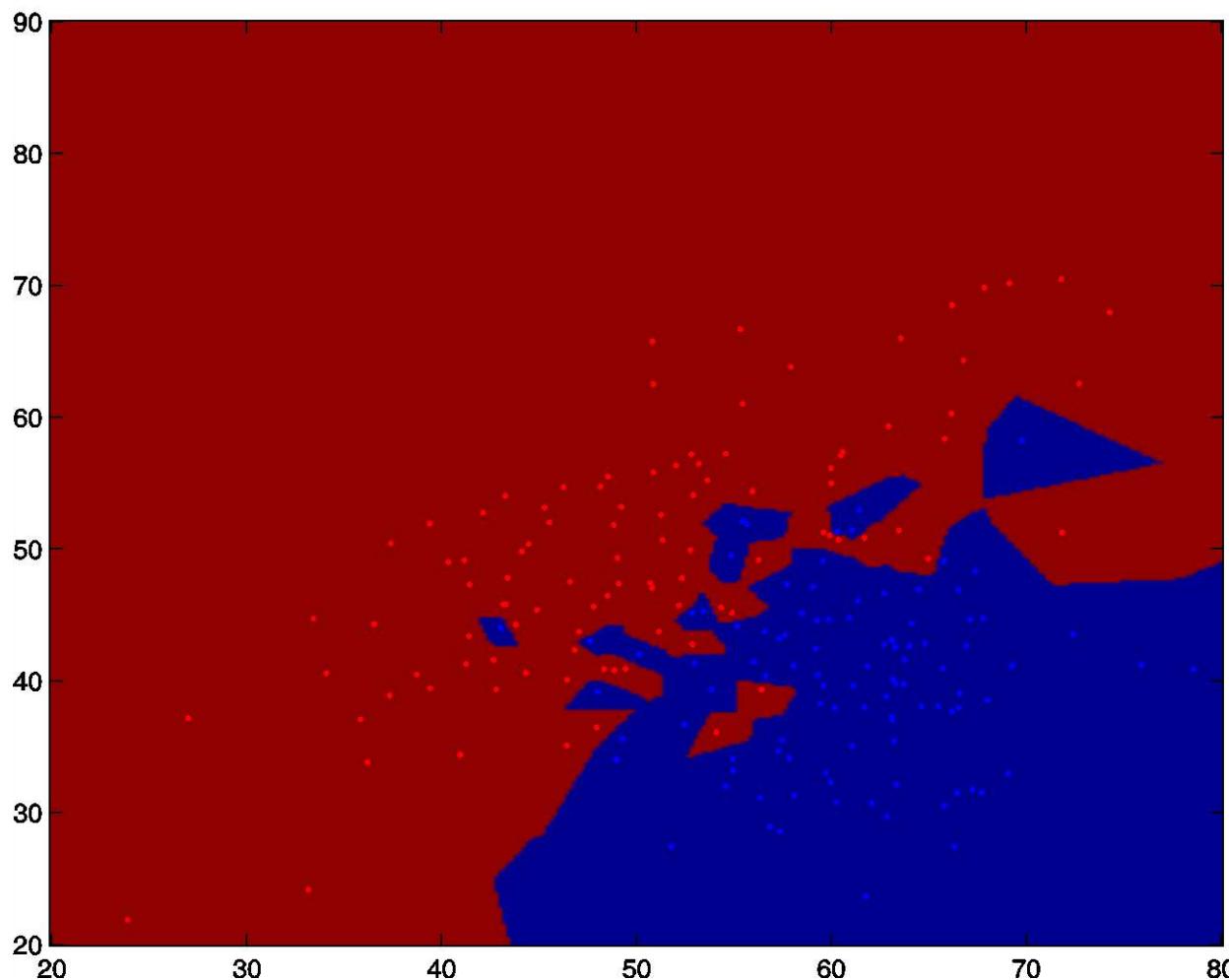


- *But can I even compare weight with redness?*
- *What if I start measuring weight in tons or milligrams?*
- *First, we need to normalize (scale) both features to have comparable values*



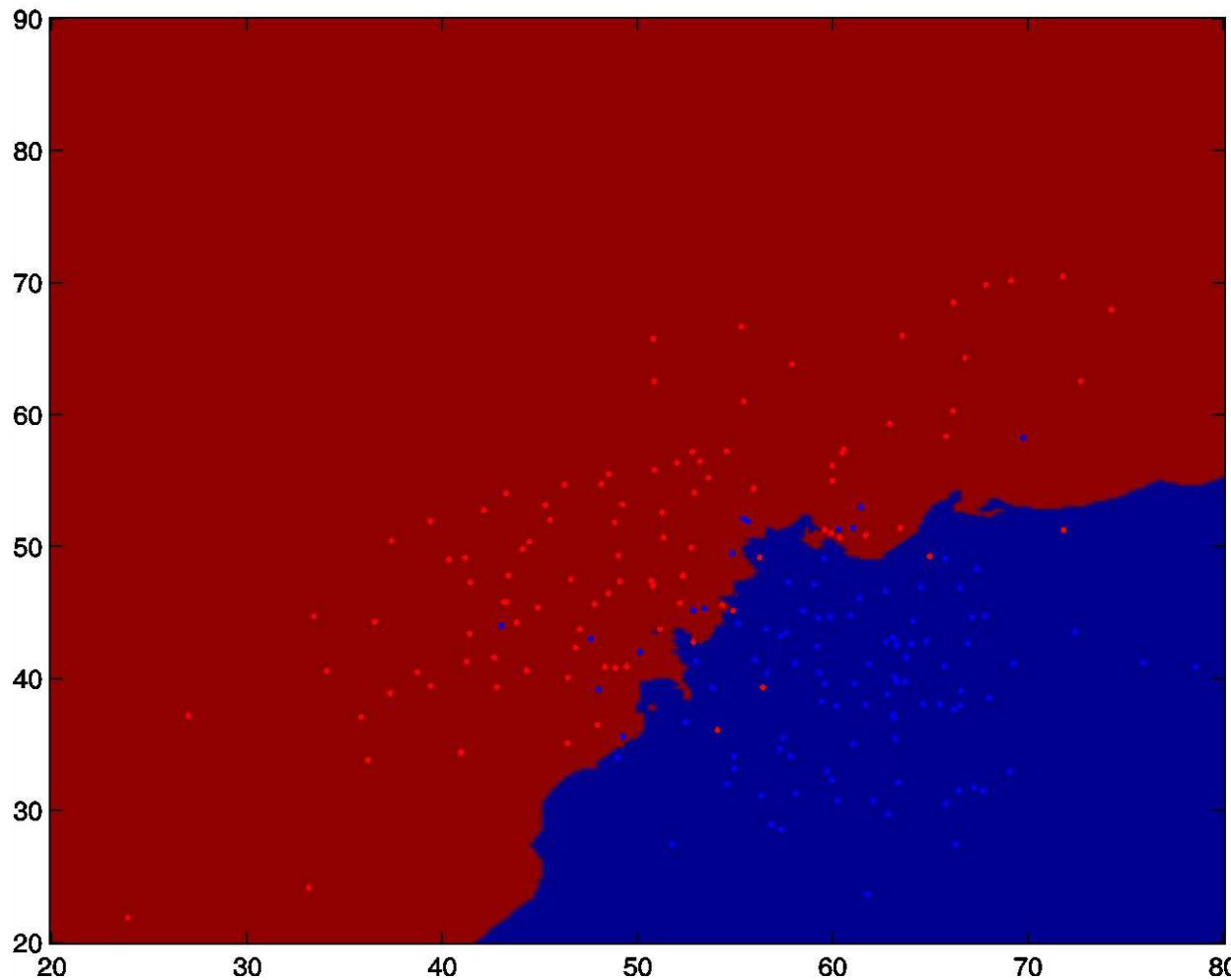
1- nearest neighbor

Again problem with generalization: too complex decision boundary



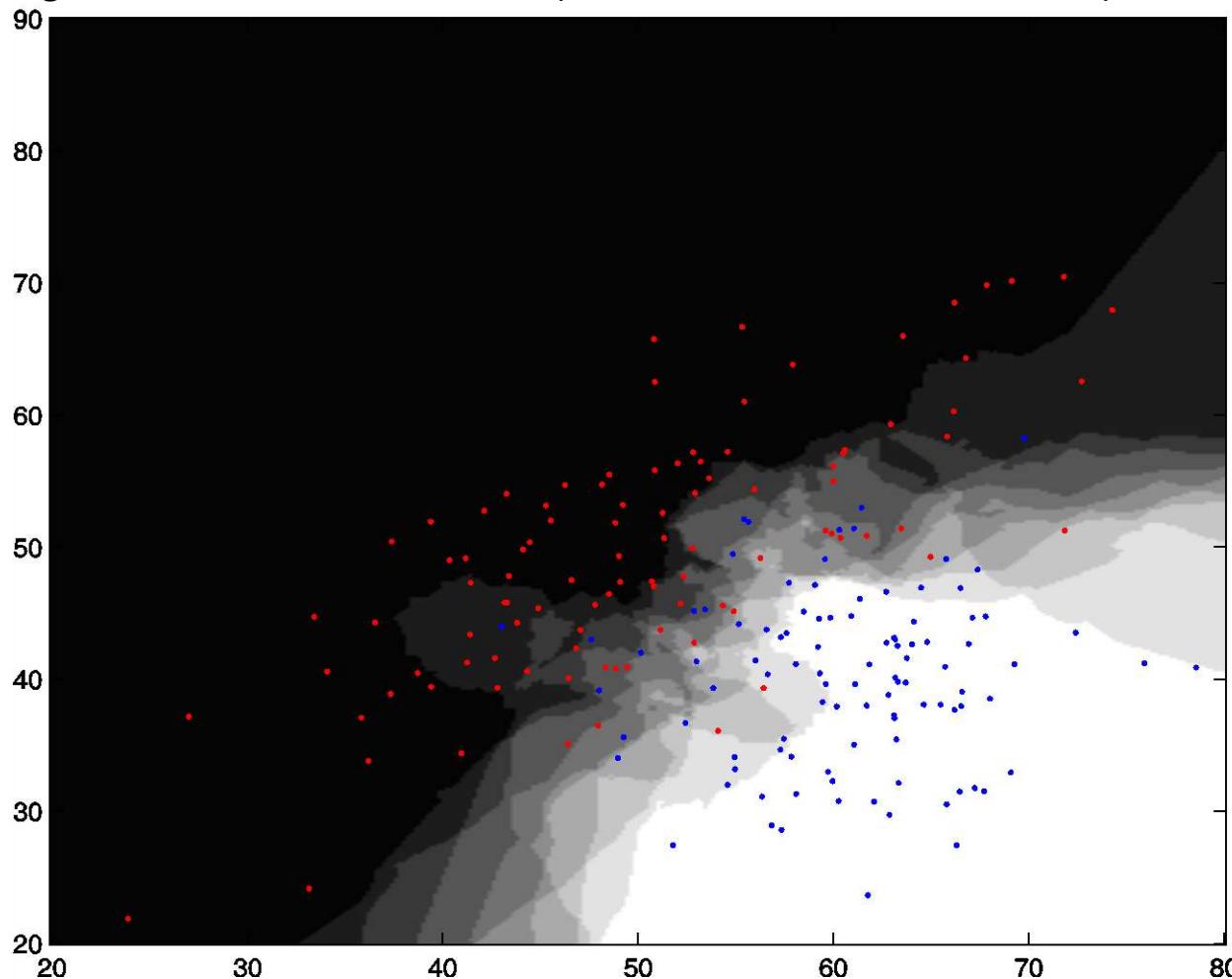
9-nearest neighbors

Decision boundary is already close to the optimal parabola



9- nearest neighbors soft decision

Ratio of the numbers of neighbors from the two classes can be used as an estimate of the probability that the classified patterns belongs to one of the classes (10 different levels for K=9).



Classification – Generative model

- With generative models, we model distribution of the observed data
- For simplicity, let's consider only discrete observations first.
- The task is to classify an object (*grenade* or *apple*) given an observation (discrete weight category)
 - It is heavy. Is it grenade or apple?
- Let's have 150 observations as training examples
 - Table of observation counts for each class and weight category



1	6	12	15	12	2	2	50
4	22	50	14	6	3	1	100
<i>lightest</i> 0.0 - 0.1	<i>lighter</i> 0.1 - 0.2	<i>light</i> 0.2 - 0.3	<i>middle</i> 0.3 – 0.4	<i>heavy</i> 0.4 – 0.5	<i>heavier</i> 0.5 – 0.6	<i>heaviest</i> 0.6 – 0.7	[kg]

Marginal, joint and conditional probability

- $P(\text{class}, \text{observation})$ – joint probability
 - probability of a single field in our table
 - normalizing the counts by the total count gives **Maximum likelihood (ML) estimates** (see later): $P(\text{grenade}, \text{heavy}) = \frac{12}{150}$
- $P(\text{class})$ – marginal probability
 - prior probability of a class
 - ML estimate: $P(\text{grenade}) = \frac{50}{150}$
- $P(\text{class}|\text{observation})$ – conditional probability
 - posterior probability of a class given an observation
 - **Maximum a-posteriori classifier selects the most likely class**
 - ML estimate: $P(\text{grenade}|\text{heavy}) = \frac{12}{12+6}$



1	6	12	15	12	2	2	50
4	22	50	14	6	3	1	100
<i>lightest</i> 0.0 - 0.1	<i>lighter</i> 0.1 - 0.2	<i>light</i> 0.2 - 0.3	<i>middle</i> 0.3 – 0.4	<i>heavy</i> 0.4 – 0.5	<i>heavier</i> 0.5 – 0.6	<i>heaviest</i> 0.6 – 0.7	[kg]

Basic rules of probability theory – I.

Sum rule:

$$P(x) = \sum_y P(x, y)$$

Product rule:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

Bayes rule:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Basic rules of probability theory – II.

- Sum rule:

$$P(\text{heavy}) = P(\text{grenade, heavy}) + P(\text{apple, heavy}) = \frac{12}{150} + \frac{6}{150} = \frac{18}{150}$$

$$P(\text{grenade}) = \sum_x P(\text{grenade}, x) = \frac{50}{150}$$

- Product rule:

$$P(\text{grenade, heavy}) = P(\text{grenade}|\text{heavy})P(\text{heavy}) = \frac{12}{18} \frac{18}{150} = \frac{12}{150}$$

$$P(\text{grenade, heavy}) = P(\text{heavy}|\text{grenade})P(\text{grenade}) = \frac{12}{50} \frac{50}{150} = \frac{12}{150}$$



$\frac{1}{150}$	$\frac{6}{150}$	$\frac{12}{150}$	$\frac{15}{150}$	$\frac{12}{150}$	$\frac{2}{150}$	$\frac{2}{150}$	$\frac{50}{150}$
$\frac{4}{150}$	$\frac{22}{150}$	$\frac{50}{150}$	$\frac{14}{150}$	$\frac{6}{150}$	$\frac{3}{150}$	$\frac{1}{150}$	$\frac{100}{150}$
<i>lightest</i> 0.0 - 0.1	<i>lighter</i> 0.1 - 0.2	<i>light</i> 0.2 - 0.3	<i>middle</i> 0.3 - 0.4	<i>heavy</i> 0.4 - 0.5	<i>heavier</i> 0.5 - 0.6	<i>heaviest</i> 0.6 - 0.7	[kg]

Basic rules of probability theory – III.

- Bayes rule:

$$P(\text{grenade}|\text{heavy}) = \frac{P(\text{heavy}|\text{grenade})P(\text{grenade})}{P(\text{heavy})}$$

Diagram illustrating the components of Bayes' rule:

- Posterior probability
- Likelihood
- Prior probability
- Evidence

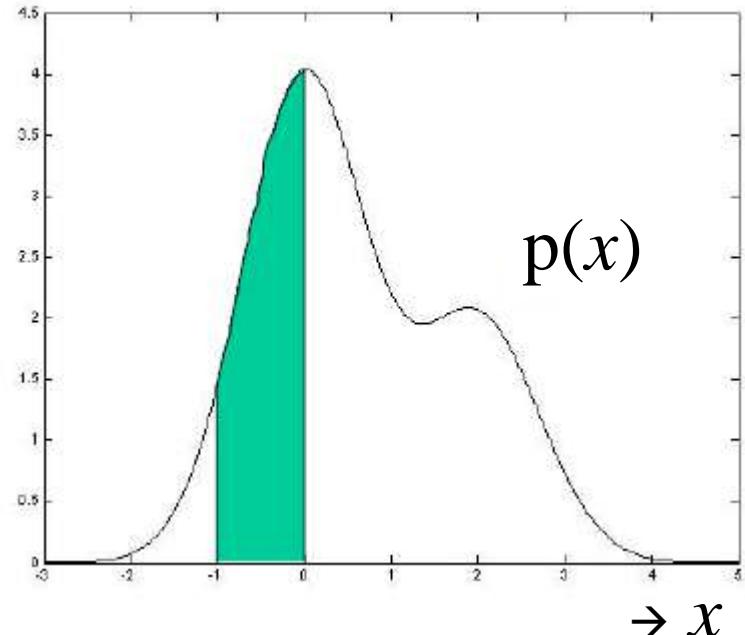
The diagram shows arrows pointing from the labels "Posterior probability", "Likelihood", and "Prior probability" to their respective terms in the numerator of the Bayes' rule equation. An arrow also points from the label "Evidence" to the term $P(\text{heavy})$ in the denominator.

- The evidence can be evaluated using the sum and product rules in terms of likelihoods and priors:
$$P(\text{heavy}) = P(\text{heavy}|\text{grenade})P(\text{grenade}) + P(\text{heavy}|\text{apple})P(\text{apple})$$
- Bayes rule for calculating the class posterior may not seem very useful now, but it will be useful in case continuous valued observations.

Continuous random variables

- $P(x)$ –probability
- $p(x)$ –probability density function

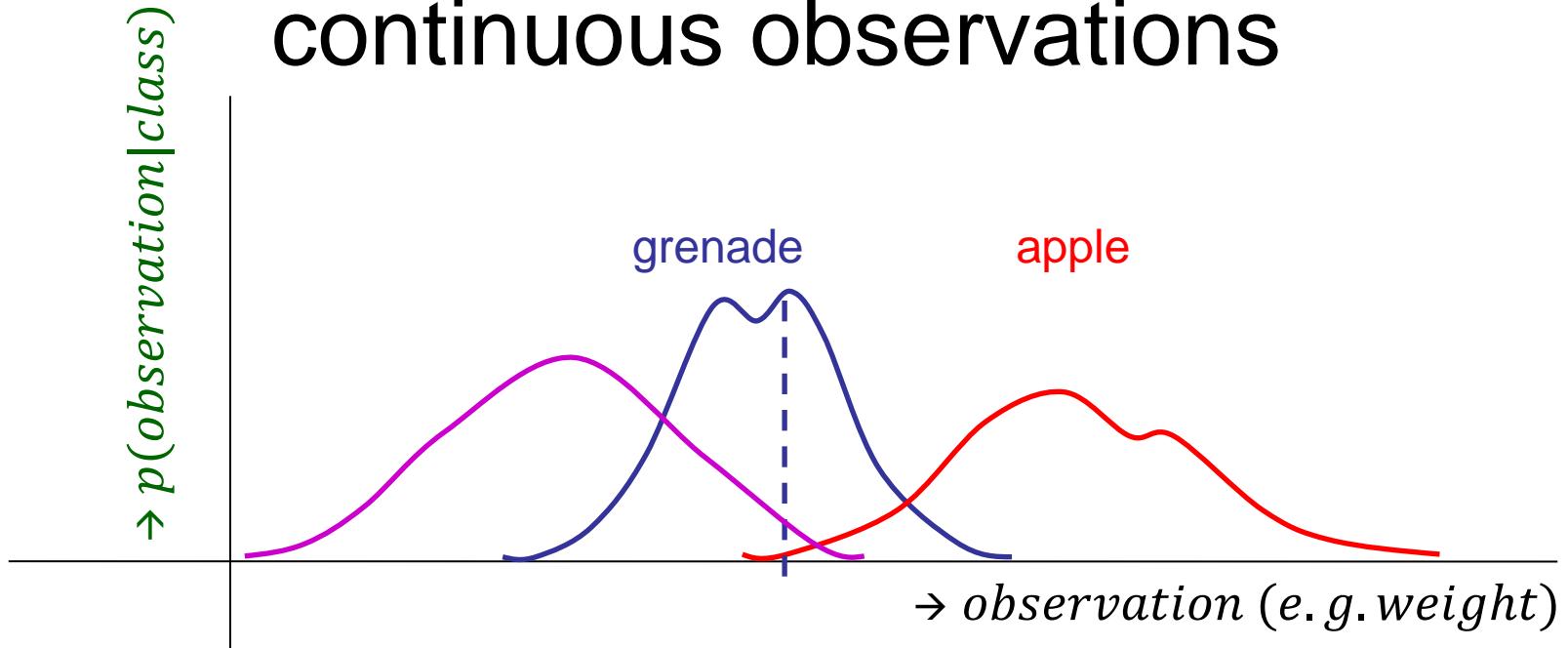
$$P(x \in (a, b)) = \int_a^b p(x) dx$$



Sum rule:

$$p(x) = \int p(x, y) dy$$

Classification - Generative model for continuous observations



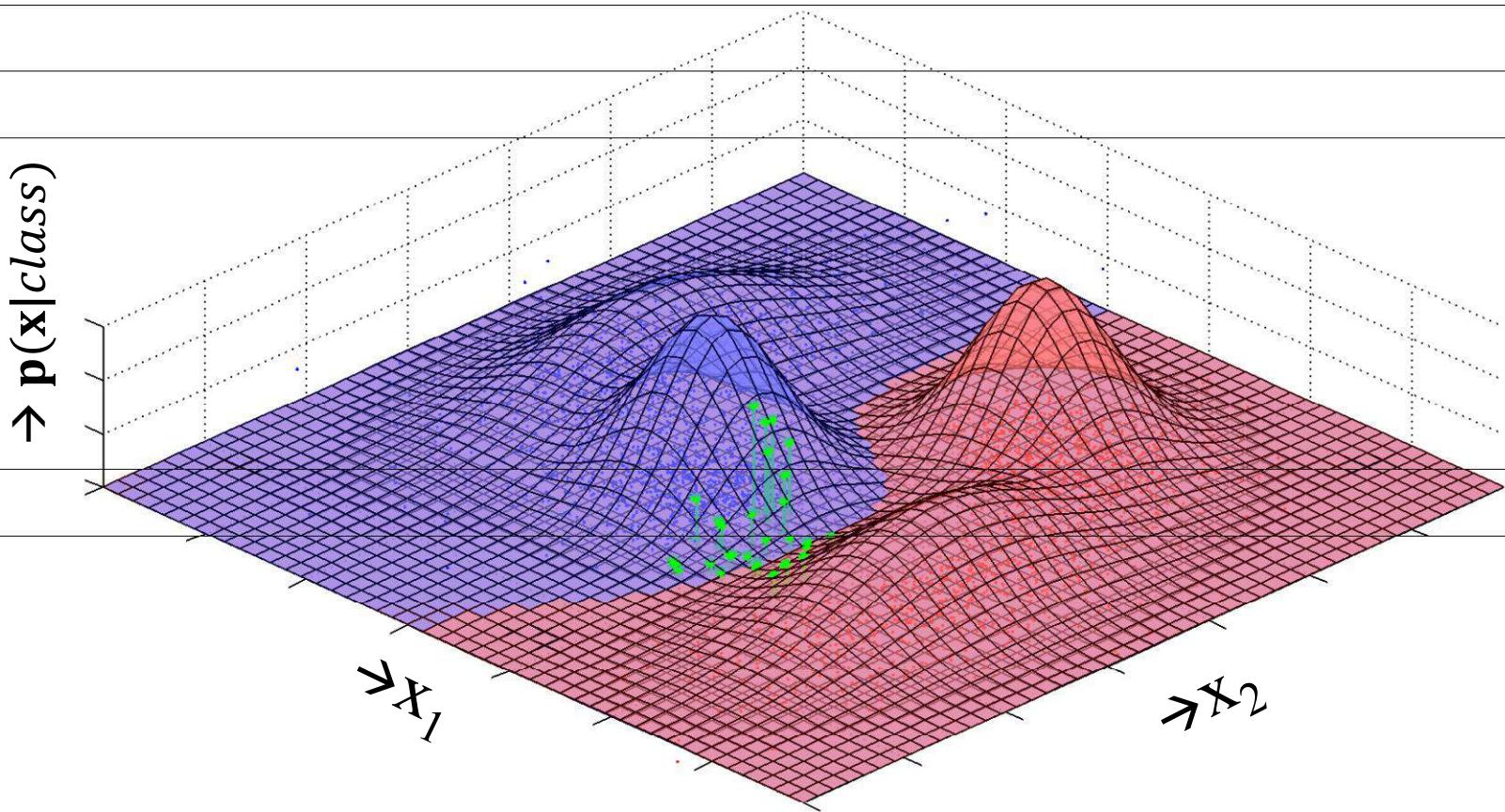
- Maximum a-posteriori classification rule says: “select the more likely class”

$$P(\text{class}|\text{observation}) = \frac{p(\text{observation}|\text{class})P(\text{class})}{p(\text{observation})}$$

$$P(\text{observation}) = \sum_{\text{class}} p(\text{observation}|\text{class})P(\text{class})$$

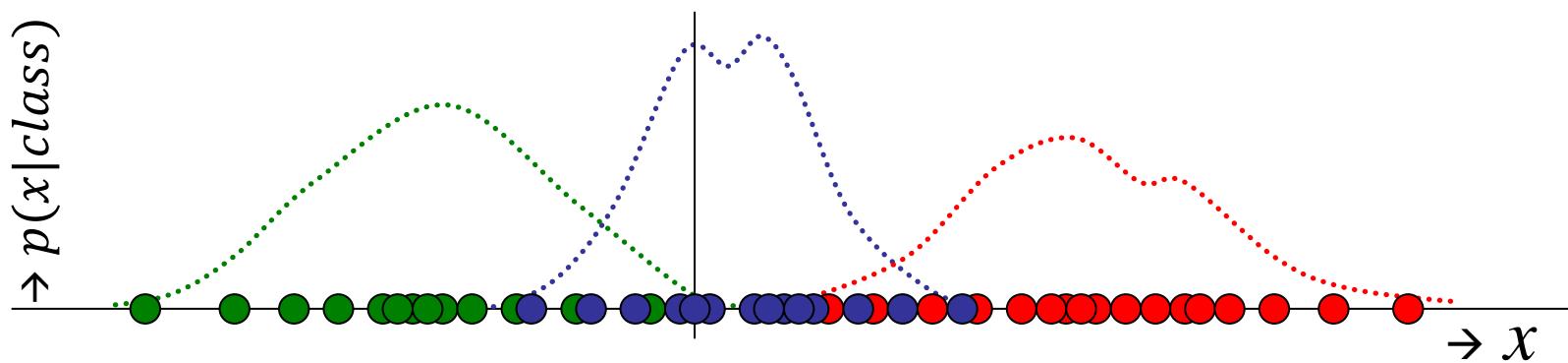
Multivariate observations

From now, univariate observations will be denoted as x and multivariate as $\mathbf{x} = [x_1, x_2, \dots, x_D] = [\text{weight}, \text{diameter}, \dots]$



Estimation of parameters

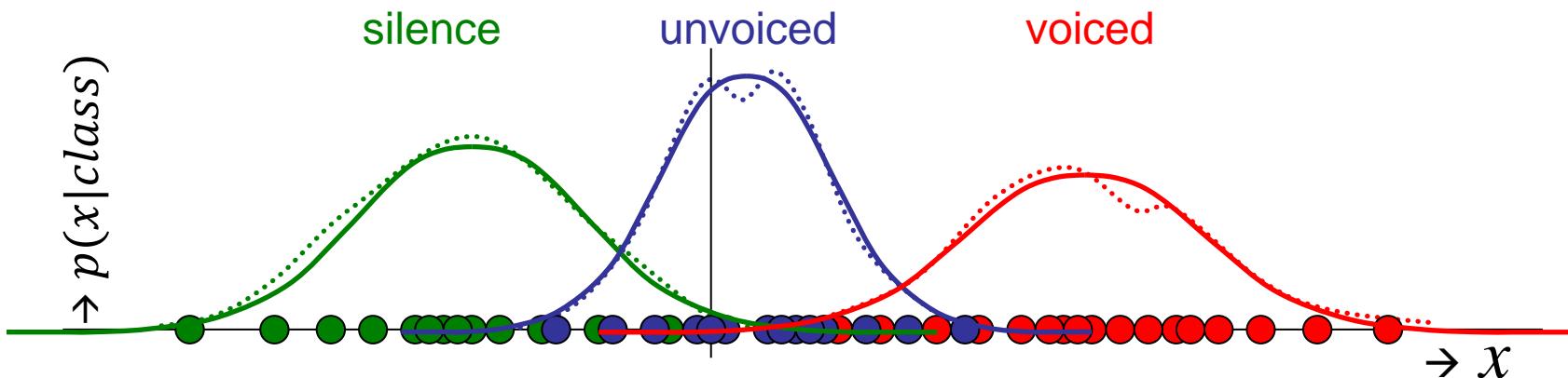
- Usually we do not know the true distributions $p(x|class)$



Estimation of parameters

... we only see some training examples.

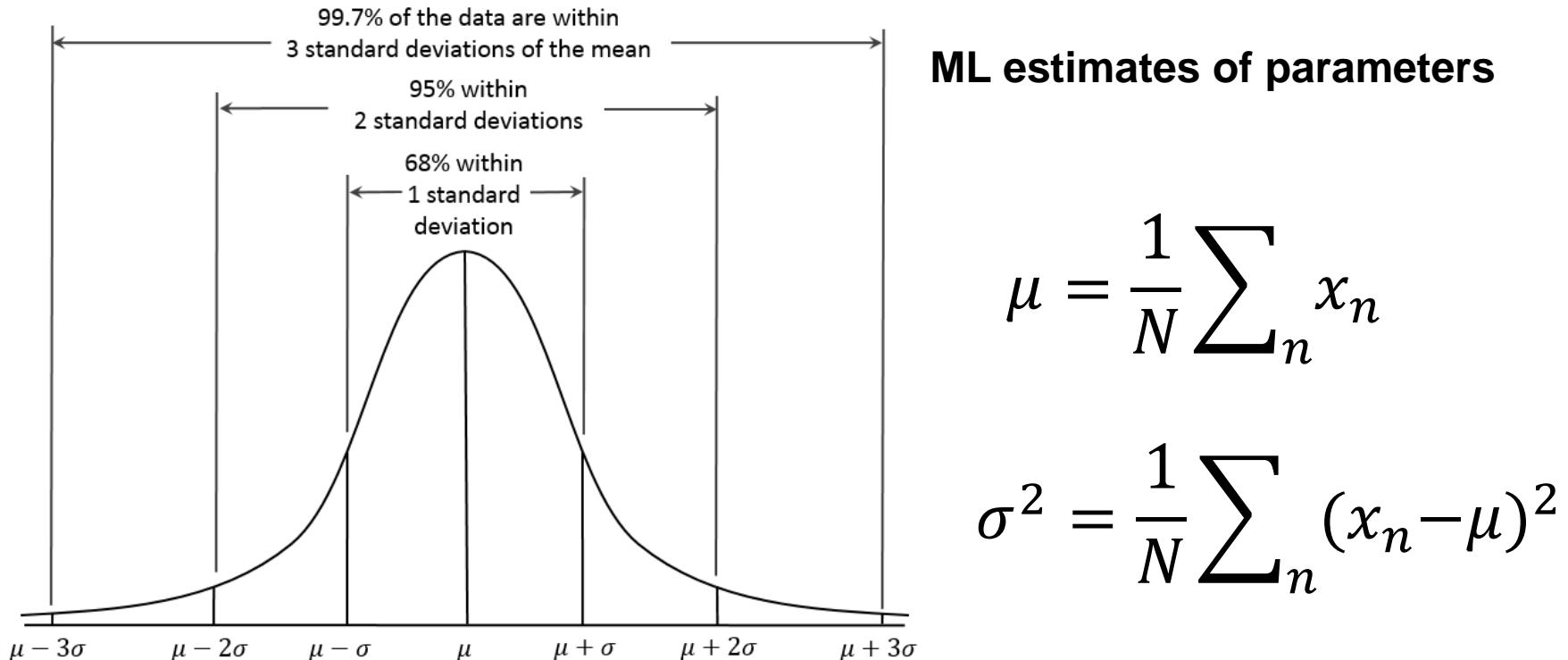
- Let's decide for some parametric model for $p(x|class)$ (e.g. Gaussian distribution) and estimate its parameters from the data.



- From now, let's forget about classes. We will concentrate just on estimating probability density functions (e.g. one for each class).

Gaussian distribution (univariate)

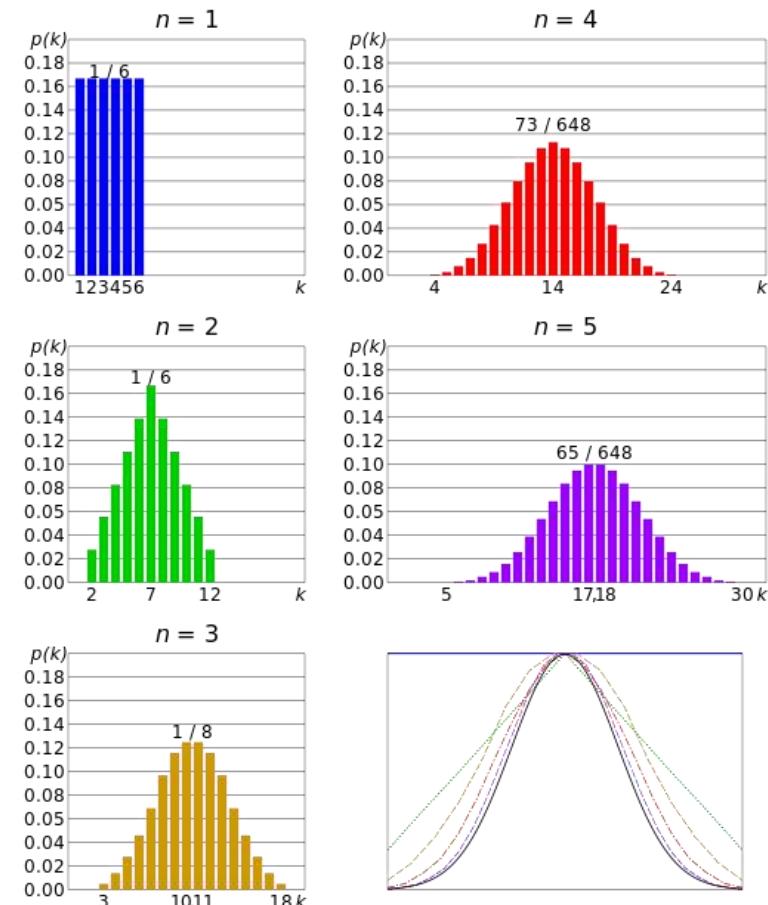
$$p(x) = \mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Why Gaussian distribution?

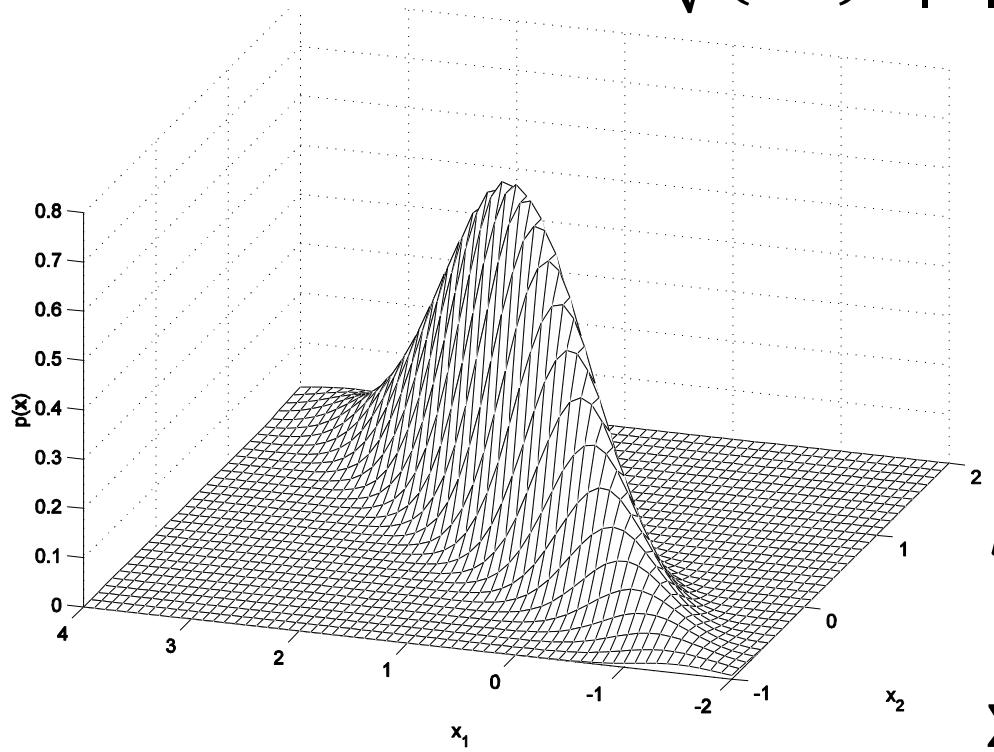
- Naturally occurring
- Central limit theorem: Summing values of many independently generated random variables gives Gaussian distributed observations
- Examples:
 - Summing outcome of N dices
 - Galton's board

<https://www.youtube.com/watch?v=03tx4v0i7MA>



Gaussian distribution (multivariate)

$$p(x_1, \dots, x_D) = \\ \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$



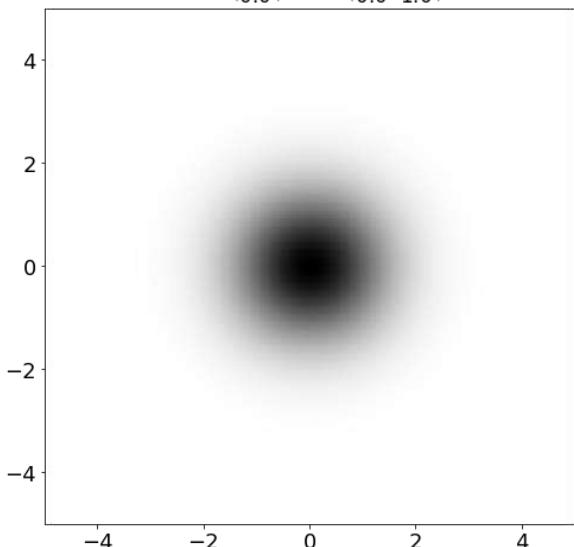
ML estimates of parameters

$$\boldsymbol{\mu} = \frac{1}{N} \sum_n \mathbf{x}_n$$

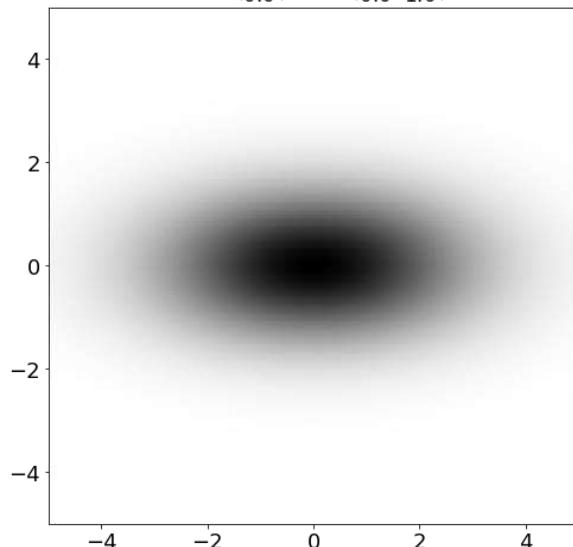
$$\boldsymbol{\Sigma} = \frac{1}{N} \sum_n (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$$

Examples of Multivariate Gaussians

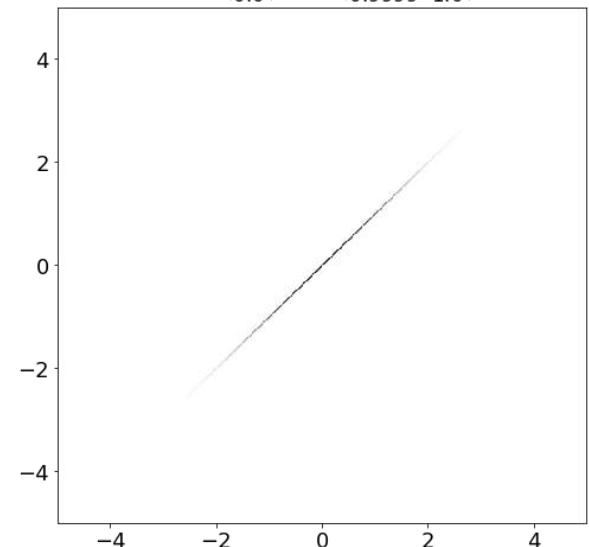
$$\mu = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$



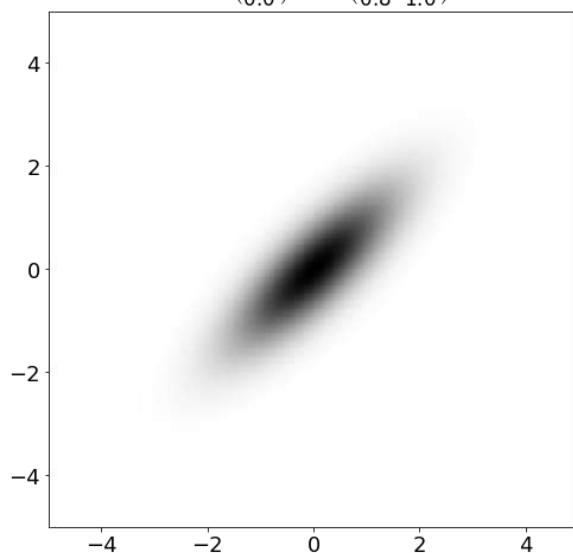
$$\mu = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \Sigma = \begin{pmatrix} 4.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$



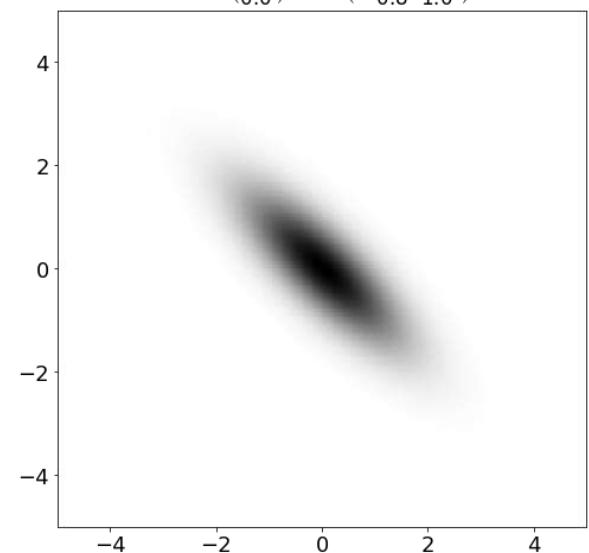
$$\mu = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \Sigma = \begin{pmatrix} 1.0 & 0.9999 \\ 0.9999 & 1.0 \end{pmatrix}$$



$$\mu = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \Sigma = \begin{pmatrix} 1.0 & 0.8 \\ 0.8 & 1.0 \end{pmatrix}$$

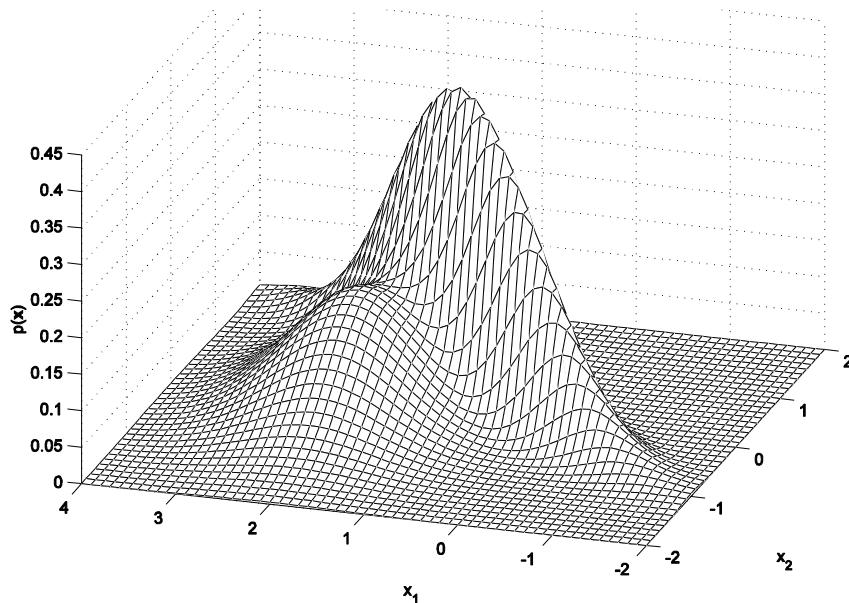


$$\mu = \begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix} \Sigma = \begin{pmatrix} 1.0 & -0.8 \\ -0.8 & 1.0 \end{pmatrix}$$



Multivariate Gaussian Mixture model

$$p(\mathbf{x}) = \sum_c \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \pi_c$$



where

$$\sum_c \pi_c = 1$$

- More complicated distributions can be modelled, for example, using weight sum of multiple Gaussian distributions
- However, this is a topic for another class ...

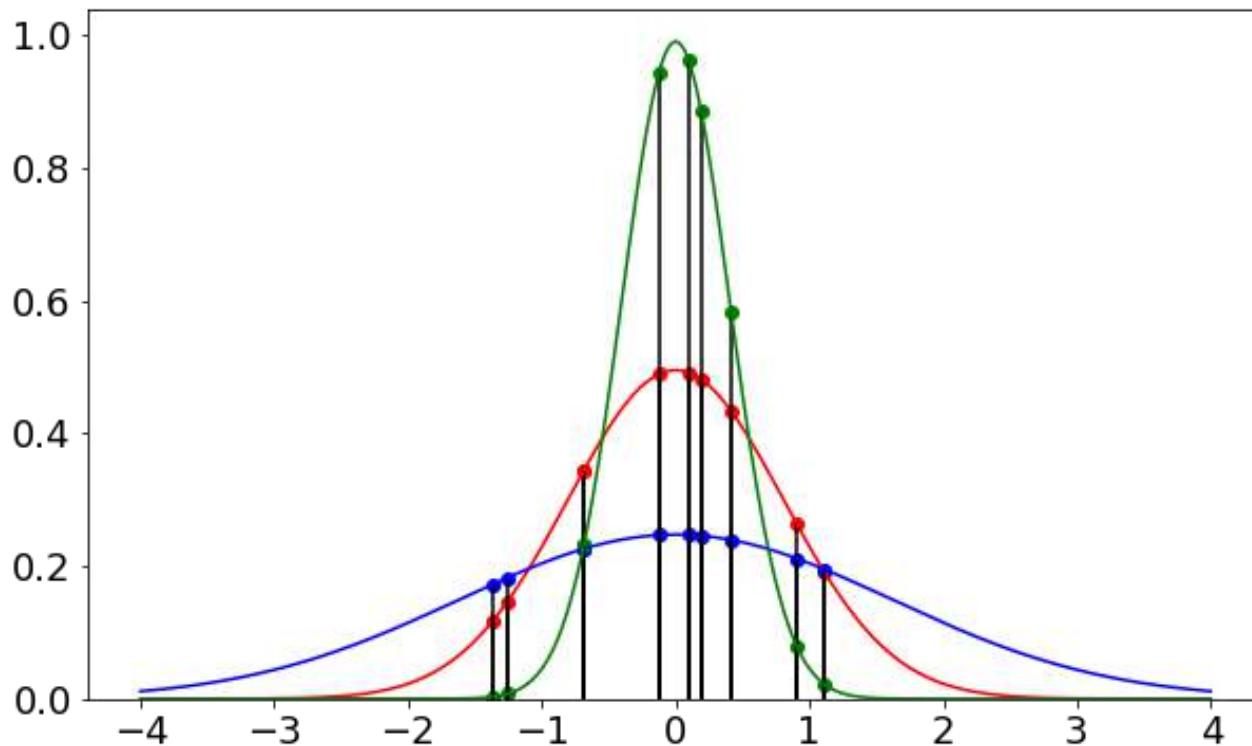
Maximum likelihood estimation of parameters

- Let's choose a parametric distribution $p(\mathbf{x}|\boldsymbol{\eta})$ with parameters $\boldsymbol{\eta}$
 - Gaussian distribution with parameters μ, σ^2
- ... and let's have some observed training data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$, which we assume to be i.i.d. generated from this distribution.
- We might obtain maximum likelihood estimates of the parameters $\hat{\boldsymbol{\eta}}^{ML}$ by maximizing the likelihood of the observed data

$$\hat{\boldsymbol{\eta}}^{ML} = \arg \max_{\boldsymbol{\eta}} p(\mathbf{X}|\boldsymbol{\eta}) = \arg \max_{\boldsymbol{\eta}} \prod_{n=1}^N p(\mathbf{x}_n|\boldsymbol{\eta})$$

- Assuming that any values of the parameters are equally likely before observing any data (so-called flat prior), ML estimate gives us the most likely parameters.

ML estimate for Gaussian



- Black horizontal lines are the training examples
- The red Gaussian corresponds to the **maximum likelihood estimate**
 - Product of heights of the red dots will be larger compared to the blue and green ones

ML estimate for Gaussian

$$\begin{aligned}\arg \max_{\mu, \sigma^2} p(\mathbf{x} | \mu, \sigma^2) &= \arg \max_{\mu, \sigma^2} \log p(\mathbf{x} | \mu, \sigma^2) = \arg \max_{\mu, \sigma^2} \sum_n \log \mathcal{N}(x_n; \mu, \sigma^2) \\ &= \arg \max_{\mu, \sigma^2} \left(-\frac{1}{2\sigma^2} \sum_n x_n^2 + \frac{\mu}{\sigma^2} \sum_n x_n - N \frac{\mu^2}{2\sigma^2} - \frac{N \log(2\pi)}{2} \right)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial \mu} \log p(\mathbf{x} | \mu, \sigma^2) &= \frac{\partial}{\partial \mu} \left(-\frac{1}{2\sigma^2} \sum_n x_n^2 + \frac{\mu}{\sigma^2} \sum_n x_n - N \frac{\mu^2}{2\sigma^2} - \frac{N \log(2\pi)}{2} \right) \\ &= \frac{1}{\sigma^2} \left(\sum_n x_n - N\mu \right) = 0 \quad \Rightarrow \quad \hat{\mu}^{ML} = \frac{1}{N} \sum_n x_n\end{aligned}$$

$$\text{and similarly: } \widehat{\sigma^2}^{ML} = \frac{1}{N} \sum_n (x_n - \mu)^2$$

Categorical distribution



4	22	50	14	6	3	1	100
<i>lightest</i> 0.0 - 0.1	<i>lighter</i> 0.1 - 0.2	<i>light</i> 0.2 - 0.3	<i>middle</i> 0.3 – 0.4	<i>heavy</i> 0.4 – 0.5	<i>heavier</i> 0.5 – 0.6	<i>heaviest</i> 0.6 – 0.7	[kg]

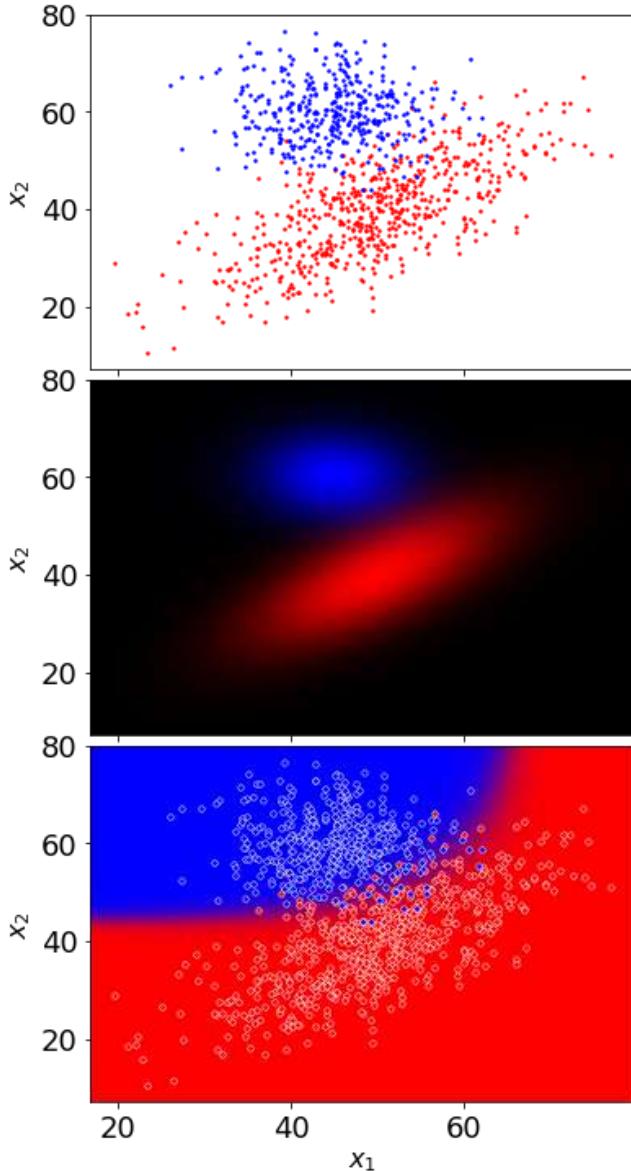
$$p(x|\boldsymbol{\pi}) = \text{Cat}(x|\boldsymbol{\pi}) = \pi_x$$

- Also referred to as **Discrete distribution**
- Special binary case is **Bernoulli distribution**
- $x \in \{\text{lightest}, \text{lighter}, \text{light}, \text{middle}, \text{heavy}, \text{heavier}, \text{heaviest}\}$
or x can be simply the index of a category $\mathbf{x} \in \{1, 2, \dots, C\}$
- $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_C]$ - probabilities of the categories are the parameters
- Likelihood of an observed training set $\mathbf{x} = [x_1, x_2, \dots, x_N]$

$$P(\mathbf{x}|\boldsymbol{\pi}) = \prod_n \text{Cat}(\mathbf{x}_n|\boldsymbol{\pi}) = \prod_n \pi_{x_n} = \prod_c \pi_c^{m_c}$$

where m_c is number of observations from category c .
– (e.g. the numbers from the table)

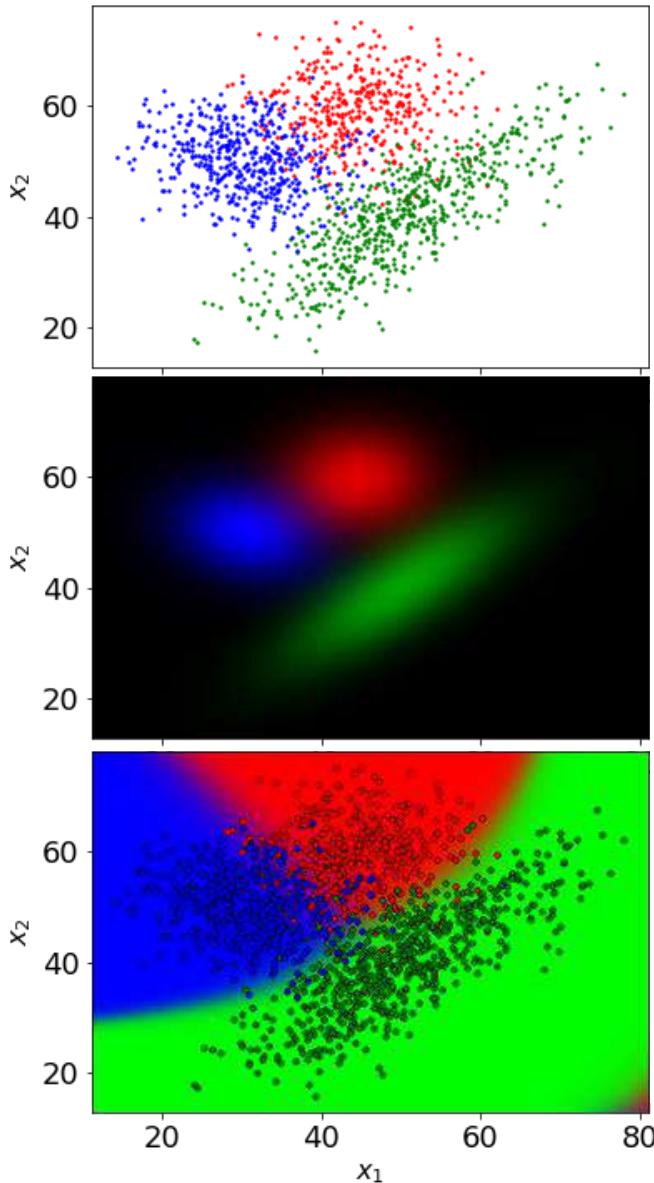
Gaussian classifier – 2D observation



- Class priors can be ML estimated as the proportions of the example counts
- Probability density function for each class is assumed to be 2D Gaussian
$$p(\mathbf{x}|c) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$
and its parameters ML estimated as
$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} \mathbf{x}_{cn} \quad \boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} (\mathbf{x}_{cn} - \boldsymbol{\mu}_c)(\mathbf{x}_{cn} - \boldsymbol{\mu}_c)^T$$
- Class posterior probability for new observations is obtained from the prior and class pdf-s using Bayes rule:

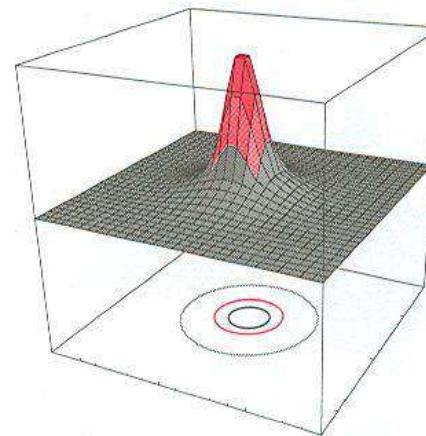
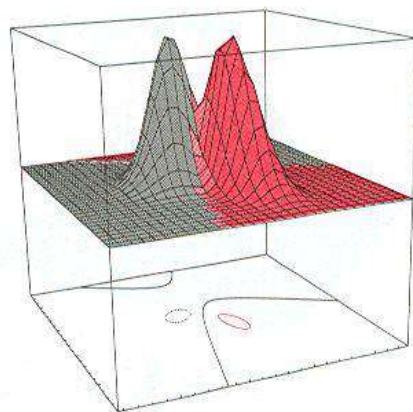
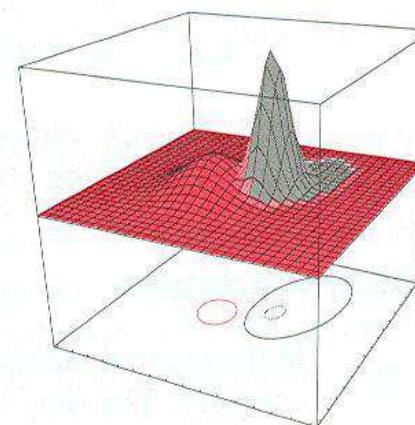
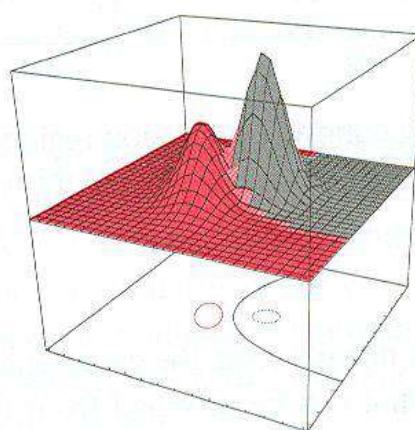
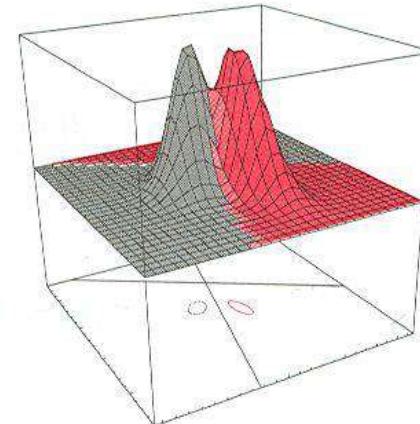
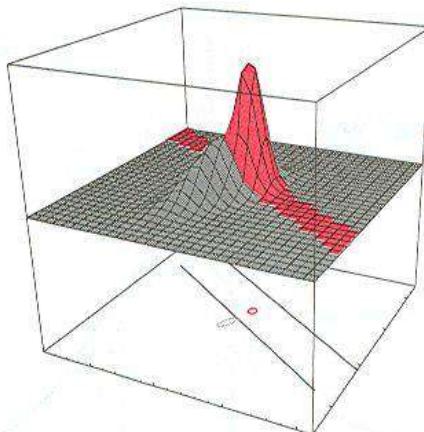
$$P(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)P(c)}{\sum_k p(\mathbf{x}|k)P(k)}$$

Gaussian classifier – more classes



- Class priors can be ML estimated as the proportions of the example counts
$$P(c) = \frac{N_c}{\sum_k N_k} \quad P(blue) = \frac{400}{400 + 600 + 500}$$
- Probability density function for each class is assumed to be 2D Gaussian
$$p(\mathbf{x}|c) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$
and its parameters ML estimated as
$$\boldsymbol{\mu}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} \mathbf{x}_{cn} \quad \boldsymbol{\Sigma}_c = \frac{1}{N_c} \sum_{n=1}^{N_c} (\mathbf{x}_{cn} - \boldsymbol{\mu}_c)(\mathbf{x}_{cn} - \boldsymbol{\mu}_c)^T$$
- Class posterior probability for new observations is obtained from the prior and class pdf-s using Bayes rule:

$$P(c|\mathbf{x}) = \frac{p(\mathbf{x}|c)P(c)}{\sum_k p(\mathbf{x}|k)P(k)}$$



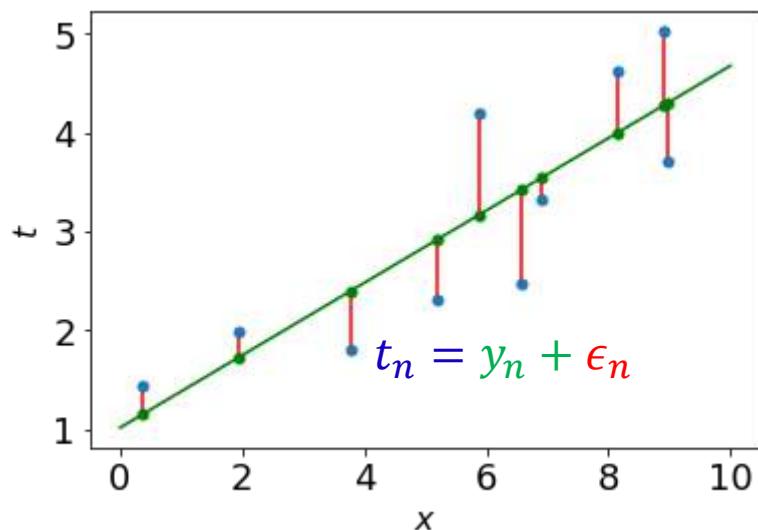
Linear Regression – I.

- The task is to learn (parameters w_0 and w_1 of) a linear function

$$y = f(x) = w_1 x + w_0$$

from training examples (pairs of inputs x_n and desired outputs t_n represented by the blue dots).

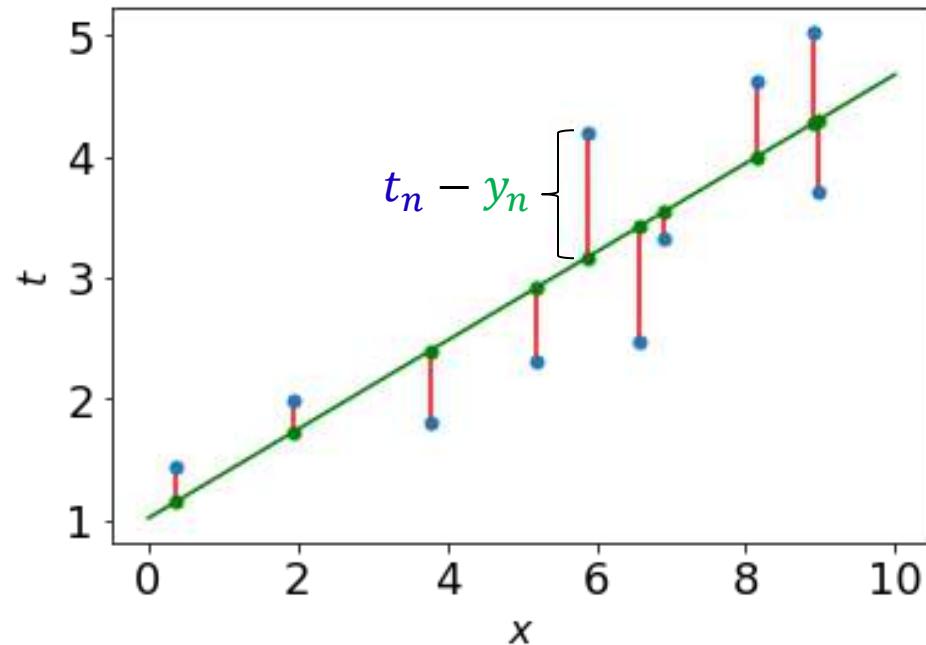
- We assume that there is a linear trend in the data, which we are trying to learn, but expect some random (Gaussian) noise to be added to the outputs (i.e. $t = y + \epsilon$, where ϵ is Gaussian random variable).



Linear Regression – II.

- For “simplicity”, we rewrite $y = w_1x + w_0 = \hat{\mathbf{x}}^T \mathbf{w}$
where $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ and $\hat{\mathbf{x}} = \begin{bmatrix} 1 \\ x \end{bmatrix}$
- We search for parameter w_0 and w_1 that minimizes sum-of-squares error objective function:

$$E(w_0, w_1) = \frac{1}{2} \sum_{n=1}^N (t_n - y_n)^2 = \frac{1}{2} \sum_{n=1}^N (t_n - \hat{\mathbf{x}}_n^T \mathbf{w})^2$$



Linear Regression – III.

- We can minimize the objective by setting its gradient equal to zero and solving for the parameters w_0 and w_1

$$\begin{aligned}
 \frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} \frac{1}{2} \sum_{n=1}^N (t_n - \hat{\mathbf{x}}_n^T \mathbf{w})^2 \\
 &= \frac{1}{2} \sum_{n=1}^N \frac{\partial}{\partial \mathbf{w}} (t_n - \hat{\mathbf{x}}_n^T \mathbf{w})^2 \\
 &= \sum_{n=1}^N (t_n - \hat{\mathbf{x}}_n^T \mathbf{w}) \frac{\partial}{\partial \mathbf{w}} (t_n - \hat{\mathbf{x}}_n^T \mathbf{w}) \\
 &= \sum_{n=1}^N (\hat{\mathbf{x}}_n^T \mathbf{w} - t_n) \hat{\mathbf{x}}_n \\
 &= \sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^T \mathbf{w} - \sum_{n=1}^N t_n \hat{\mathbf{x}}_n = 0
 \end{aligned}$$

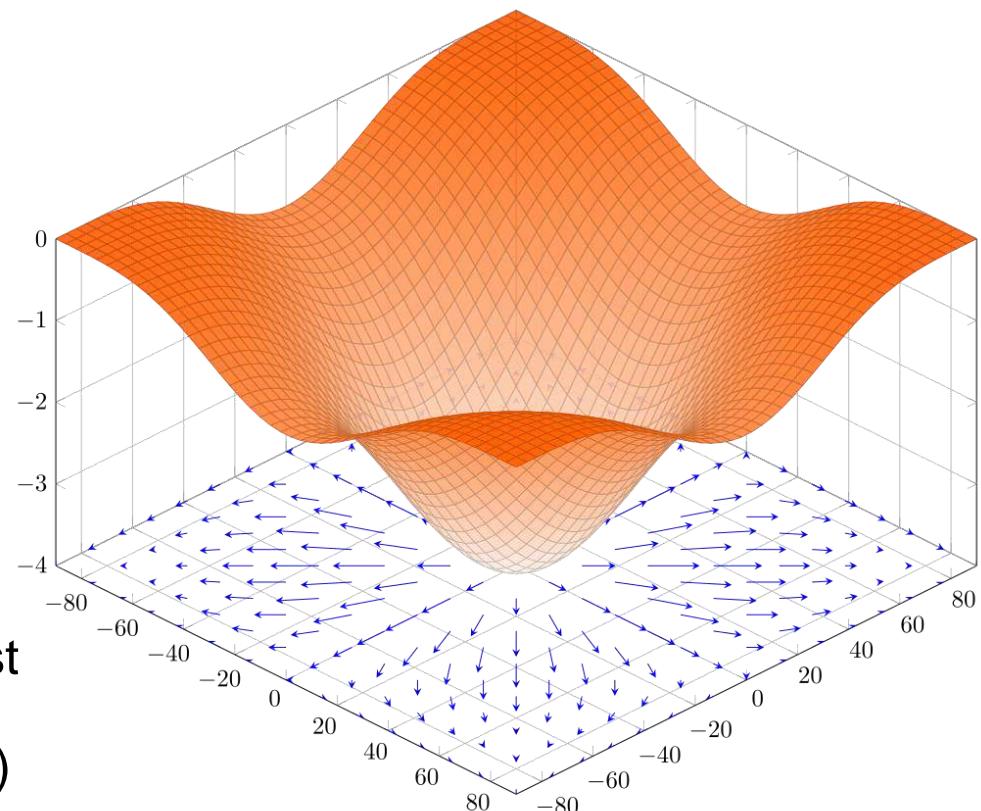
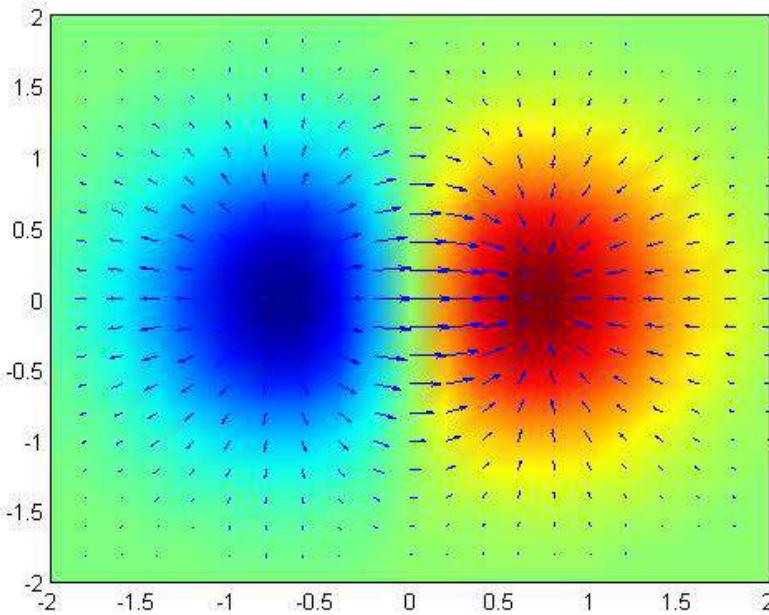
$$\begin{aligned}
 \left(\sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^T \right) \mathbf{w}^* &= \sum_{n=1}^N t_n \hat{\mathbf{x}}_n \\
 \mathbf{w}^* &= \left(\sum_{n=1}^N \hat{\mathbf{x}}_n \hat{\mathbf{x}}_n^T \right)^{-1} \sum_{n=1}^N t_n \hat{\mathbf{x}}_n \\
 &= (\hat{\mathbf{X}} \hat{\mathbf{X}}^T)^{-1} \hat{\mathbf{X}} \mathbf{t} \\
 &= \hat{\mathbf{X}}^\dagger \mathbf{t}
 \end{aligned}$$

Where \dagger means *Moore-Penrose pseudo-inverse*, $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_N]$ is matrix of all extended inputs and $\mathbf{t} = [t_1, t_2, \dots, t_N]^T$ is vector of all desired outputs

Gradient

$$\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial E(\mathbf{w})}{\partial w_0} \\ \frac{\partial E(\mathbf{w})}{\partial w_1} \end{bmatrix}$$

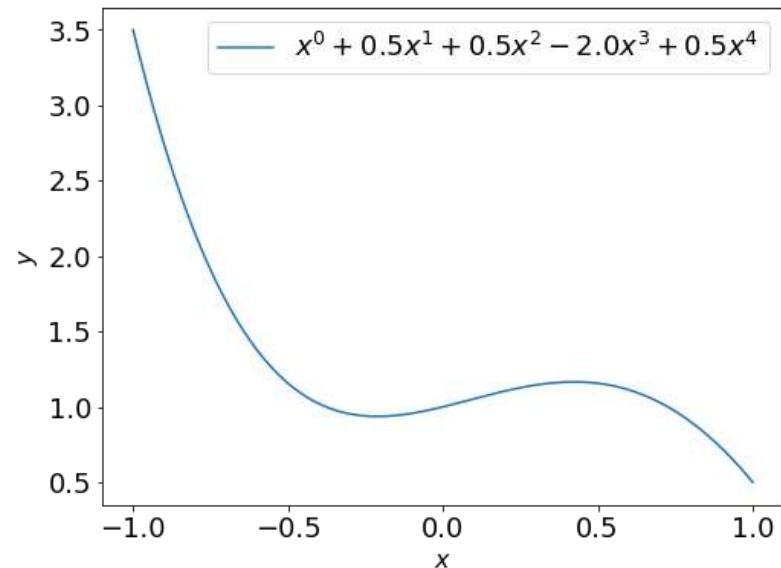
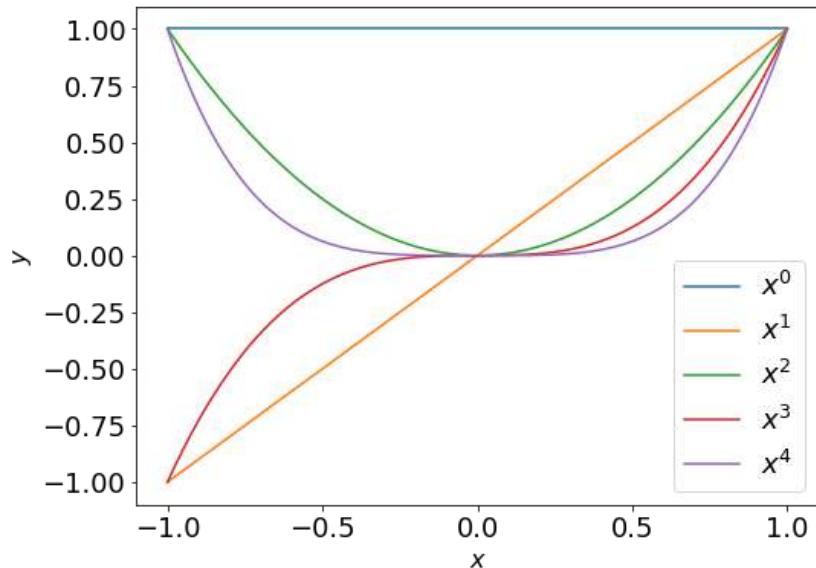
- For each $\mathbf{w}, \nabla E(\mathbf{w})$ is a vector showing the “most uphill” direction and the length of this vector is the “steepness of the slope” in that direction



- For linear regression, $E(\mathbf{w})$ is just multivariate-quadratic function (i.e. simpler than these examples)

Learning nonlinear function using Linear Regression

- Any non-linear function can be represented as linear combination of other “simpler” nonlinear functions (e.g. polynomial functions)



- We can use linear regression to learn the combination weights

Polynomial Regression – I.

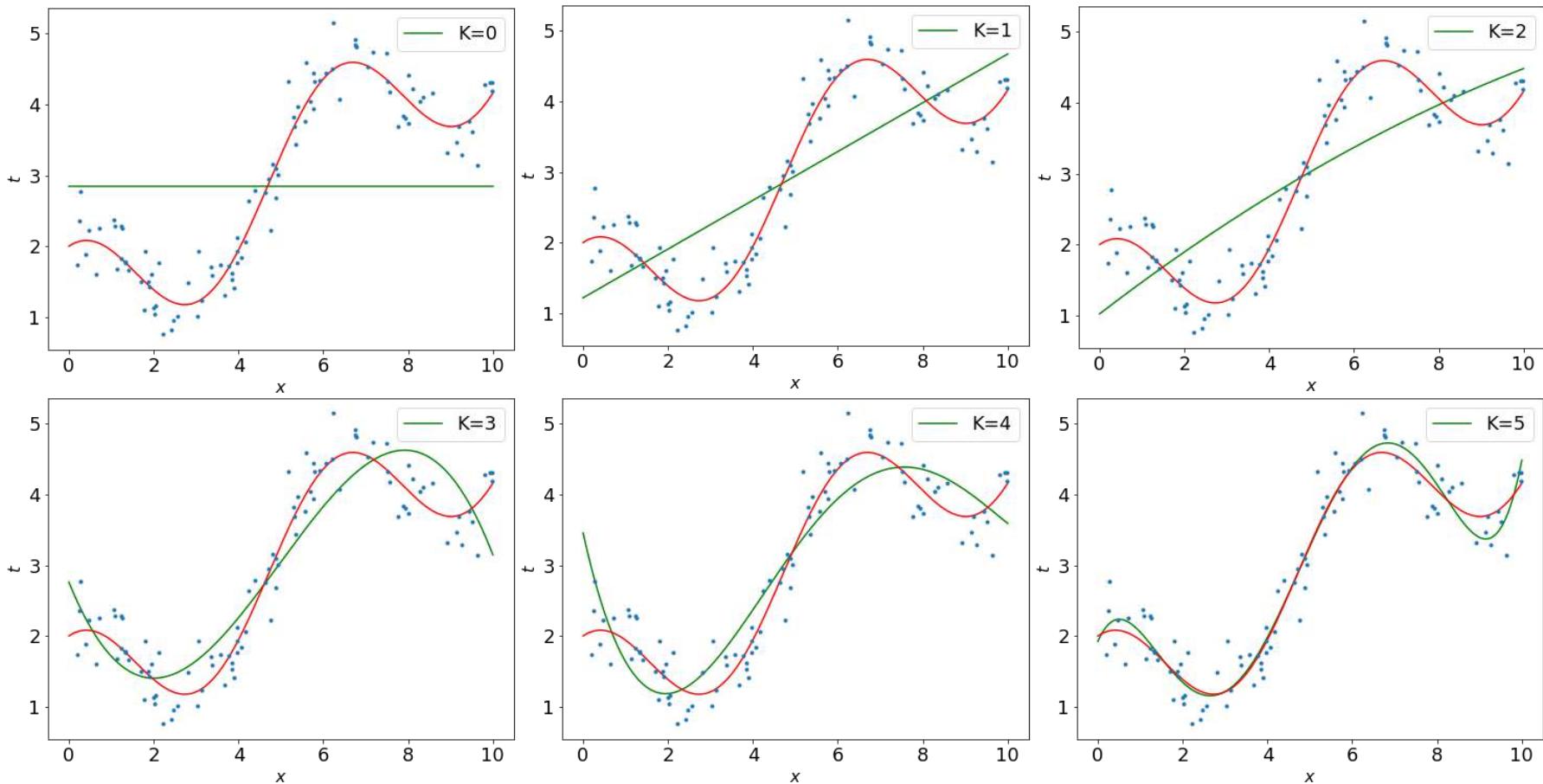
- Let make $\hat{\mathbf{x}} = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^K \end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}$
- Using the same formulation of the problem as for the linear case, we now get polynomial regression as a special case of linear regression

$$y = \hat{\mathbf{x}}^T \mathbf{w} = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$$

- The optimal weights \mathbf{w}^* can be found using the same formulas as before. In fact, the linear case considered before is just the special case for $K = 1$
- Polynomial expansion (polynomial base functions) is just one possibility. We can choose to use any other non-linear bases:

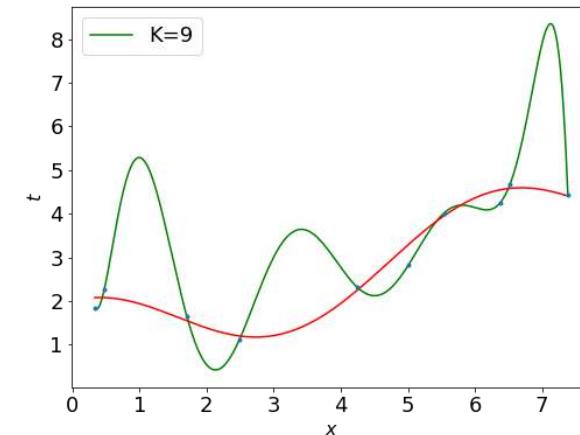
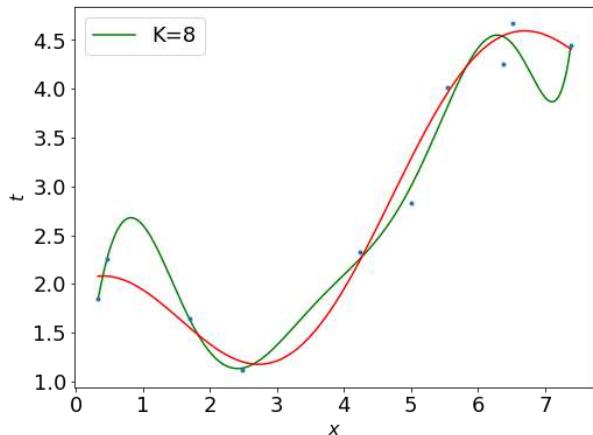
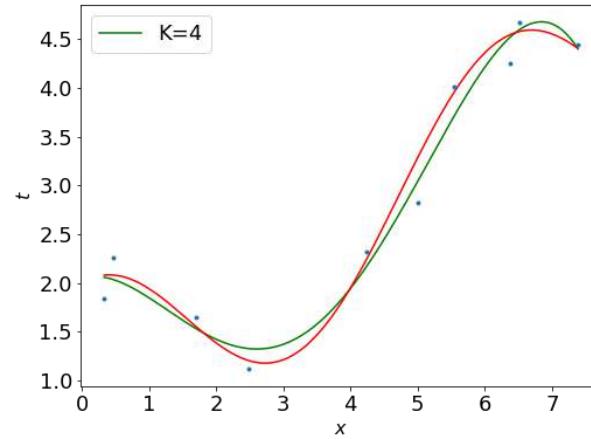
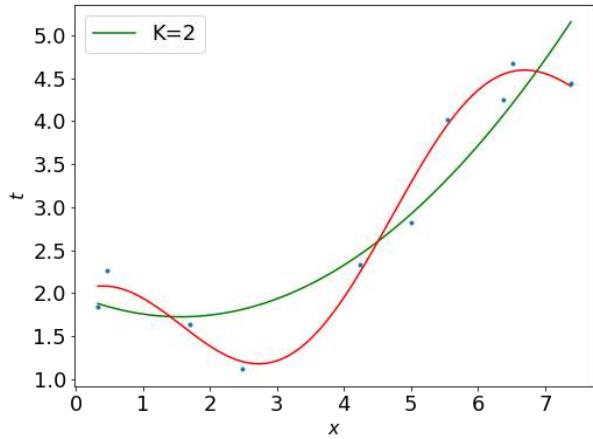
$$\hat{\mathbf{x}} = [f_1(x), f_2(x), \dots, f_K(x)]^T$$

Polynomial Regression – II.



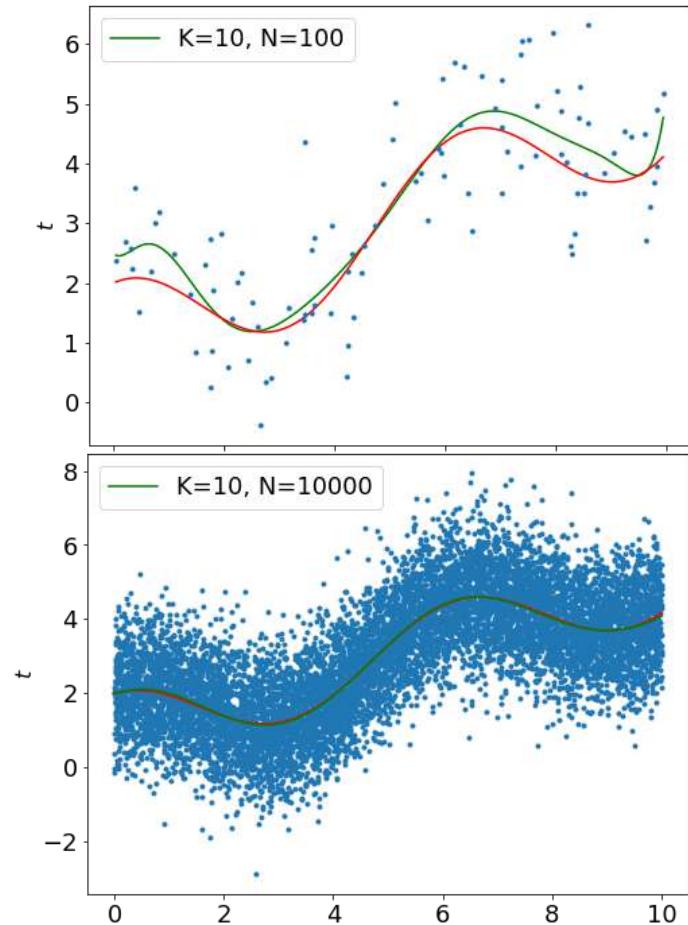
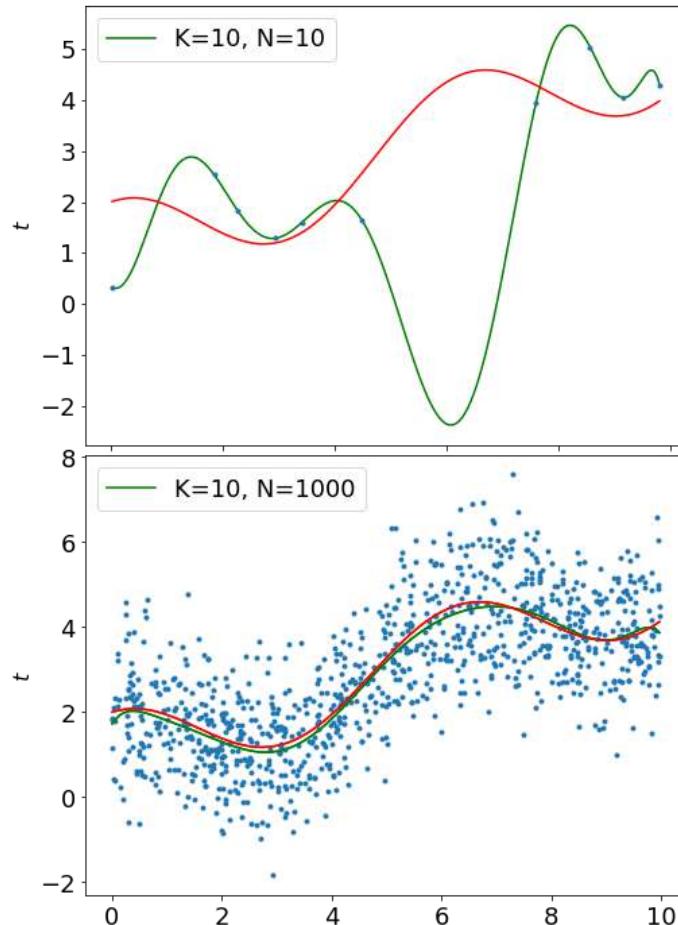
Polynomial Regression – III.

- The model must have the right complexity otherwise, we underfit or overfit (so-called bias-variance tradeoff)



Why sum-of-squares error function?

- If the output is only corrupted by Gaussian noise, sum-of-squares error objective tends to learn the correct underlying function
 - as the number of training data increases
 - given the sufficient modelling capacity (e.g. large enough K)



Why sum-of-squares error function?

- We learn function
- but assume output distributed as

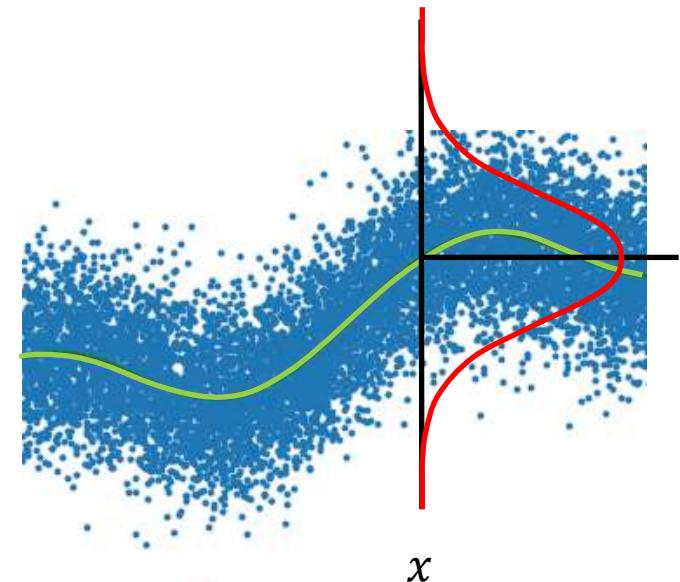
$$y = f(x) = \hat{\mathbf{x}}^T \mathbf{w}$$

$$\mathbf{t} = \mathbf{y} + \boldsymbol{\epsilon}$$

where $\boldsymbol{\epsilon}$ is Gaussian noise $\mathcal{N}(\boldsymbol{\epsilon}; 0, \sigma^2)$

- In other words

$$p(\mathbf{t}|\mathbf{x}) = \mathcal{N}(\mathbf{t}; \mathbf{y}, \sigma^2) = \mathcal{N}(\mathbf{t}; \hat{\mathbf{x}}_n^T \mathbf{w}, \sigma^2)$$



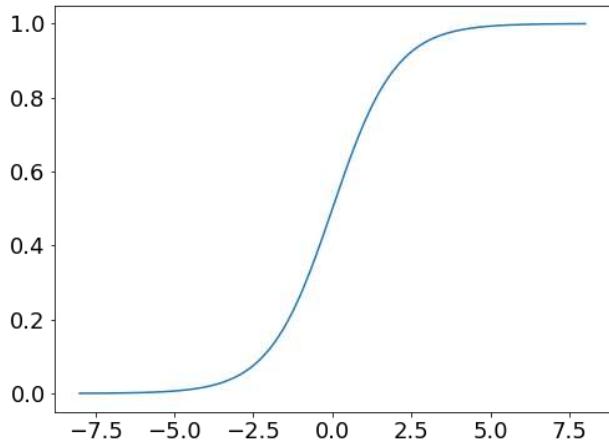
- To obtain maximum likelihood estimates of parameters \mathbf{w} (and σ^2) given training examples x_n, t_n , we need to maximize log likelihood

$$\begin{aligned} \log p(\mathbf{t}|\mathbf{X}) &= \sum_{n=1}^N \log p(t_n|\mathbf{x}_n) = \sum_{n=1}^N \log \mathcal{N}(t_n; \hat{\mathbf{x}}_n^T \mathbf{w}, \sigma^2) \\ &= -\frac{N}{2} \log 2\pi\sigma^2 - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - \hat{\mathbf{x}}_n^T \mathbf{w})^2 \end{aligned}$$

which corresponds to **sum-of-squares objective** for learning \mathbf{w}

Classification – Discriminative approach

- With generative models (e.g. Gaussian classifier)
 - we learn distribution of observations for each class $p(\mathbf{x}|c)$
 - and class prior probabilities $p(c)$
 - and use Bayes rule to derive posterior probability of each class $p(c|\mathbf{x})$
 - we try to precisely model $p(\mathbf{x}|c)$ even at places far from the decision boundary
⇒ possible waste of parameter; model that is not compact
- Let's concentrate directly at modeling the posterior $p(c|\mathbf{x})$
 - For now, let's consider only binary classifier (i.e. $c \in \{0,1\}$), where the posterior $p(c|\mathbf{x})$ is just a function, with output constrained to be in the range of 0 and 1
 - We were able to learn functions from data using linear regression.
 - Let's use linear regression-like model with the output mapped to the 0 – 1 range using logistic sigmoid function $\sigma(a)$:



$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$P(c = 1|\mathbf{x}) = \sigma(\hat{\mathbf{x}}^T \mathbf{w})$$

Linear logistic regression – I.

- 2-class linear logistic regression models directly posterior probability of class $c = 1$ as

$$P(c = 1|\mathbf{x}) = \sigma(\hat{\mathbf{x}}^T \mathbf{w})$$

- Probability of the other class $c = 0$ is

$$P(c = 0|\mathbf{x}) = 1 - P(c = 1|\mathbf{x})$$

- To train the model (i.e. the parameter \mathbf{w}), we need training examples
 - \mathbf{x}_n , which we expand as before to $\hat{\mathbf{x}}_n$ (e.g. using the polynomial basis)
 - t_n encodes the correct class i.e. (has value 1 or 0)
- Sum-of-squares error objective can be used again as the training objective, ...
- but it is better to use Maximum Likelihood approach as we try to predict the probability $P(c|\mathbf{x})$
 - We maximize the same objective as for ML estimate of Categorical distribution, except that the discrete probability is now conditioned on \mathbf{x} :

$$P(\mathbf{t}|\mathbf{X}) = P(t_1, \dots, t_N | \mathbf{x}_1, \dots, \mathbf{x}_N) = \prod_n P(t_n | \mathbf{x}_n) = \prod_n \sigma(\hat{\mathbf{x}}_n^T \mathbf{w})^{t_n} \left(1 - \sigma(\hat{\mathbf{x}}_n^T \mathbf{w})\right)^{(1-t_n)}$$

Linear logistic regression – II.

- Instead of maximizing

$$P(\mathbf{t}|\mathbf{X}) = \prod_n P(t_n|\mathbf{x}_n) = \prod_n \sigma(\hat{\mathbf{x}}_n^T \mathbf{w})^{t_n} \left(1 - \sigma(\hat{\mathbf{x}}_n^T \mathbf{w})\right)^{(1-t_n)}$$

the machine learning people prefer to minimize equivalent “error”

$$E(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{X}) = \sum_{n=1}^N t_n \ln \sigma(\hat{\mathbf{x}}_n^T \mathbf{w}) + (1 - t_n) \ln (1 - \sigma(\hat{\mathbf{x}}_n^T \mathbf{w}))$$

called (binary) **cross-entropy**.

- By taking its derivative w.r.t. \mathbf{w} , we obtain gradient

$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (\sigma(\hat{\mathbf{x}}_n^T \mathbf{w}) - t_n) \hat{\mathbf{x}}_n$$

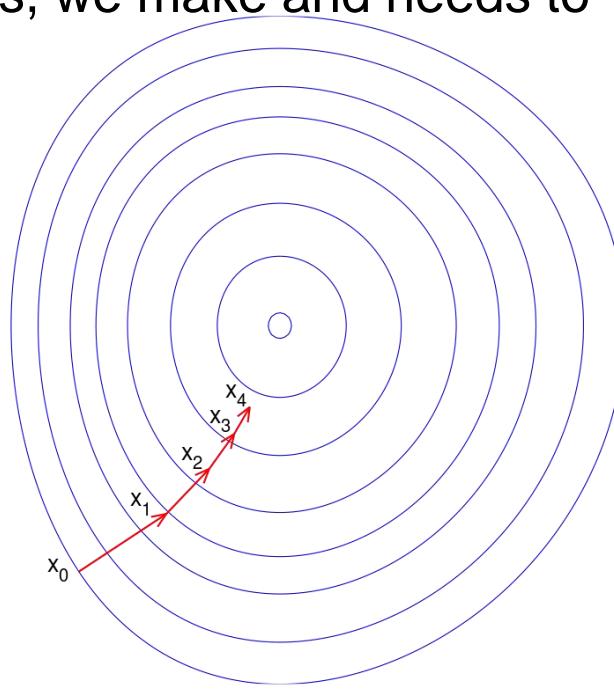
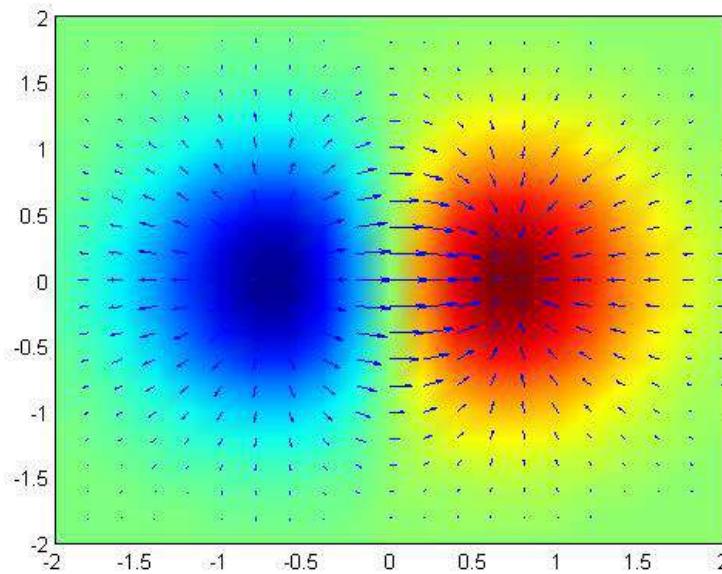
- Unfortunately, this objective does not have closed-form solution as it was in the case of linear regression.
 - We need to resort to numerical optimization (e.g. gradient descent)
 - Quadratic optimization is often better choice, but we do not consider it here

Gradient descent optimization

- Repeatedly update the parameters \mathbf{w} by moving in small steps in the direction opposite to the gradient $\nabla E(\mathbf{w})$ (i.e. downhill) until reaching the minimum of the objective function (i.e. $\nabla E(\mathbf{w}) = \mathbf{0}$).

$$\mathbf{w}^{\tau+1} = \mathbf{w}^\tau - \eta \nabla E(\mathbf{w}^\tau)$$

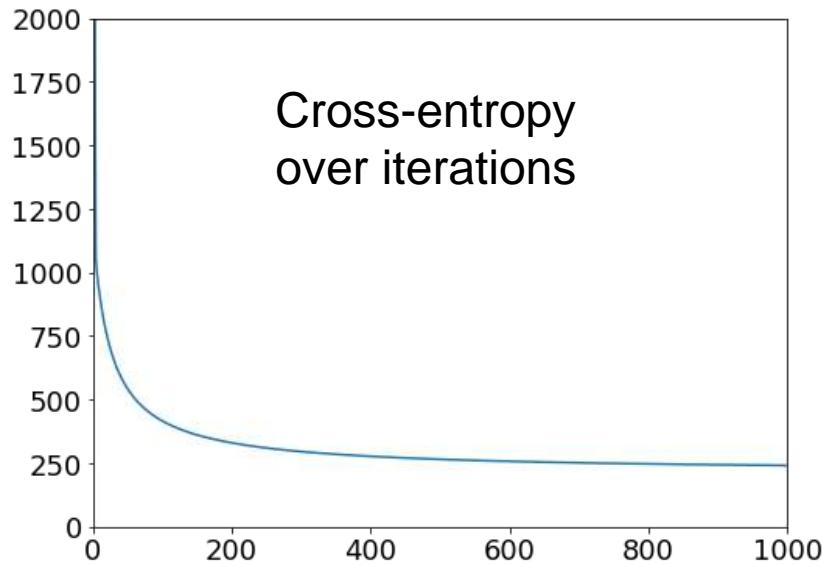
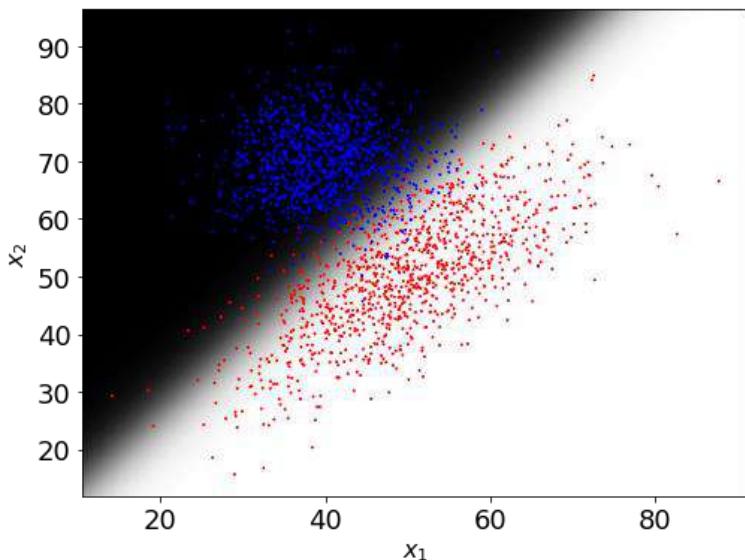
- The *learning rate* η controls how big steps we make and needs to be tuned for good convergence



Logistic regression – example

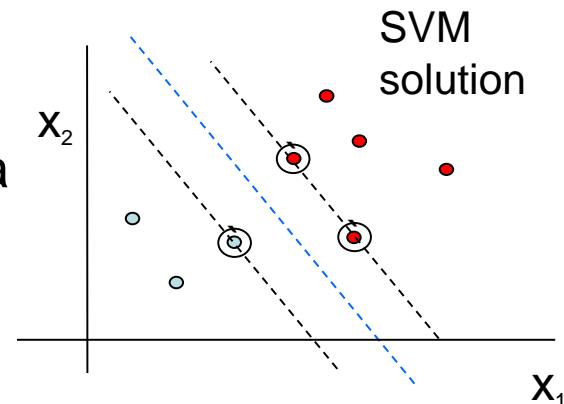
- Classifier trained using linear logistic regression without any polynomial expansion
 - only offers linear decision boundary
 - only 3 parameters to train (Gaussian classifier had 10)

$$P(c = 1|\mathbf{x}) = \sigma(\hat{\mathbf{x}}^T \mathbf{w}) = \sigma(w_1 x_1 + w_2 x_2 + w_0)$$



Discriminative vs Generative

- For generative models we learn $p(\mathbf{x}|c)$ and $p(c)$ and use Bayes rule to derive $p(c|\mathbf{x})$
- Logistic regression is discriminative approach, where we learn to estimate directly $p(c|\mathbf{x})$
- There are other discriminative training strategies that do not even try to estimate any probability
- For example, Support Vector Machines directly finds decision boundary that maximizes margin between the classes
- Generative models
 - Usually less prone to overfitting with small amount of training data
 - Modular: complex models build out of simple ones
- Discriminative training
 - Less waste of parameters
 - Usually, better performance with enough data
 - Allows end-to-end solution



Summary

- Machine learning: automatically learning from **training data** to accomplish certain **task**
- **Generalization** to data unseen during training is the core objective
- There are many different tasks, models, and strategies and algorithms
 - **Supervised**, **unsupervised**, reinforcement, ... learning
 - **Generative** models vs. **discriminative** training
- The learning algorithms usually optimizes some **objective function** which should correlate with the actual task objective
 - **Maximum likelihood** objective for learning distributions
 - **Sum-of-squares** error for regression
 - **Cross-entropy** for classification
 - We only considered simple models (linear/logistic regression), but the same objectives and the same optimization strategies (e.g. **gradient descent**) are used with more powerful models such as neural networks.
- The objective function is usually optimized w.r.t. some **parameters** of the underlying model, which we use to accomplish the desired task

Umělá inteligence a strojové učení

Michal Hradiš: Neural networks

Where to get more information

Ian Goodfellow and Yoshua Bengio and Aaron Courville: **Deep Learning**, MIT Press, 2016. <http://www.deeplearningbook.org/>

Andrew Ng and Kian Katanforoosh: CS230 **Deep Learning**, Stanford, <https://cs230.stanford.edu/>

CS231n: **Convolutional Neural Networks for Visual Recognition**, Stanford, <http://cs231n.stanford.edu/index.html>

Andrew Ng, [CS229 Machine Learning](#), Stanford, <https://www.youtube.com/watch?v=UzxYIbK2c7E&list=PLEBC422EC5973B4D8>

Python Notebooks

Basic Pytorch -

<https://colab.research.google.com/drive/1ytNsahwdI4FJDkXmaw7CCQQJqQDiAxJt?usp=sharing>

Image classification -

<https://colab.research.google.com/drive/1oh-iicfucIPJRudYZVIUWRRLtJiMdM4k?usp=sharing#scrollTo=5PRaK3DSPzQw>

<https://colab.research.google.com/drive/1a8jZeo-pmrRDAHB8cK0TrEmailMnaKNC#scrollTo=7k37fwroNnN>

Pytorch tutorials - <https://pytorch.org/tutorials/>



AI PLAYS BASKETBALL

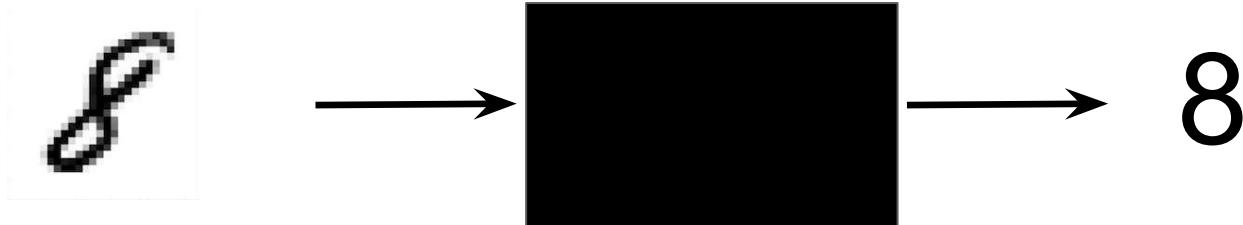
HUMAN DIGITIZATION





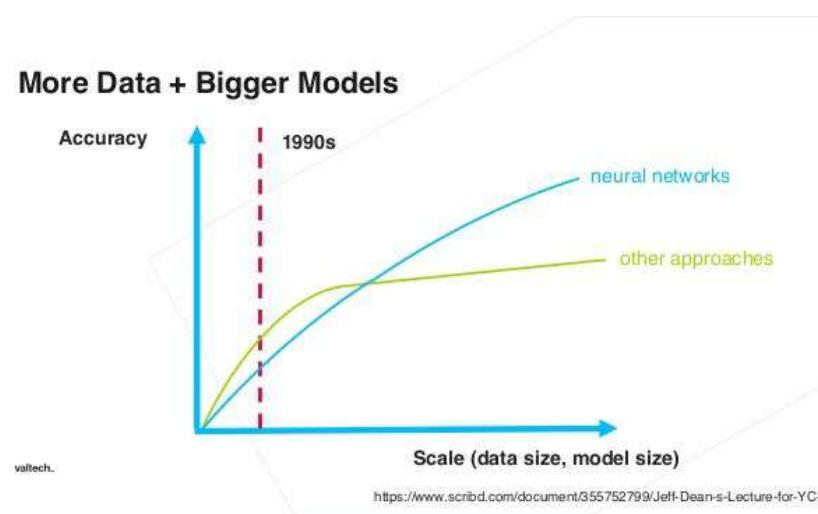
What are neural networks?

- Complex and flexible functions (and computationally expensive)
- Can learn to approximate a desired function from training data
- Can accept large range of input types and produce large range of output types
- Can be “customized” to solve very different tasks.



Why?

- Scales with computation
- Scales with data
- General framework
- State of the art
- Massive development
- Mature tools



Goals

- Understand basics of machine learning with neural networks
- Understand the flexibility of neural networks
- Have an idea what tools are available
- Complex models and their applications
- Basic intuitions when solving practical tasks
 - Data
 - Generalization
 - Regularization
 - What can go wrong
 - Transfer learning

What is a neural network? (in this lecture)

Numerical input vector (x)

0.1
0.2
0.5
0.6
0.7
0.8
0.4

Neuronové sítě jsou parametrické modely - funkce u kterých nastavováním vnitřních parametrů můžeme docílit požadovaného chování.

Vstupem těchto funkcí je v nejjednodušším případě vektor reálných čísel. Můžou to být i 2D, 3D, 4D, tenzory, grafy a podobně. Obecně to ale musí být reálná čísla. Celá čísla nejdou derivovat.

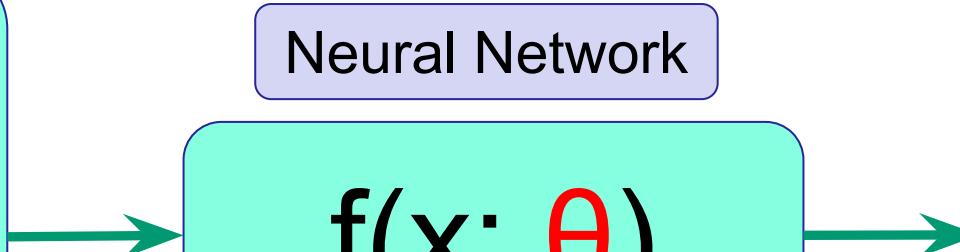
Výstupem je jsou také reálná čísla (vektor, 2D tenzor, 3D tenzor, ...). Zase je potřeba, aby bylo možné přes výstupy derivovat. Třeba výsledek "na obrázku je kočka" derivovat nejde, ale "pravděpodobnost, že na obrázku je kočka je 0.9" derivovat jde.

Numerical output vector

0.5
1.2
3
2
0

Neural Network

$f(x; \theta)$



Binary classification network

Features
(x)

Logistická represe je také neuronová síť. Sice velmi jednoduchá, ale je. Stejně platí třeba pro lineární regresi. Jsou to velmi jednoduché neuronové sítě, ale učit se můžou úplně stejně jako velmi složité sítě. I chybové funkce logistické a lineární regrese se používají pro trénování složitých NN.

Weight 0.1
Red 0.2
Size 0.5

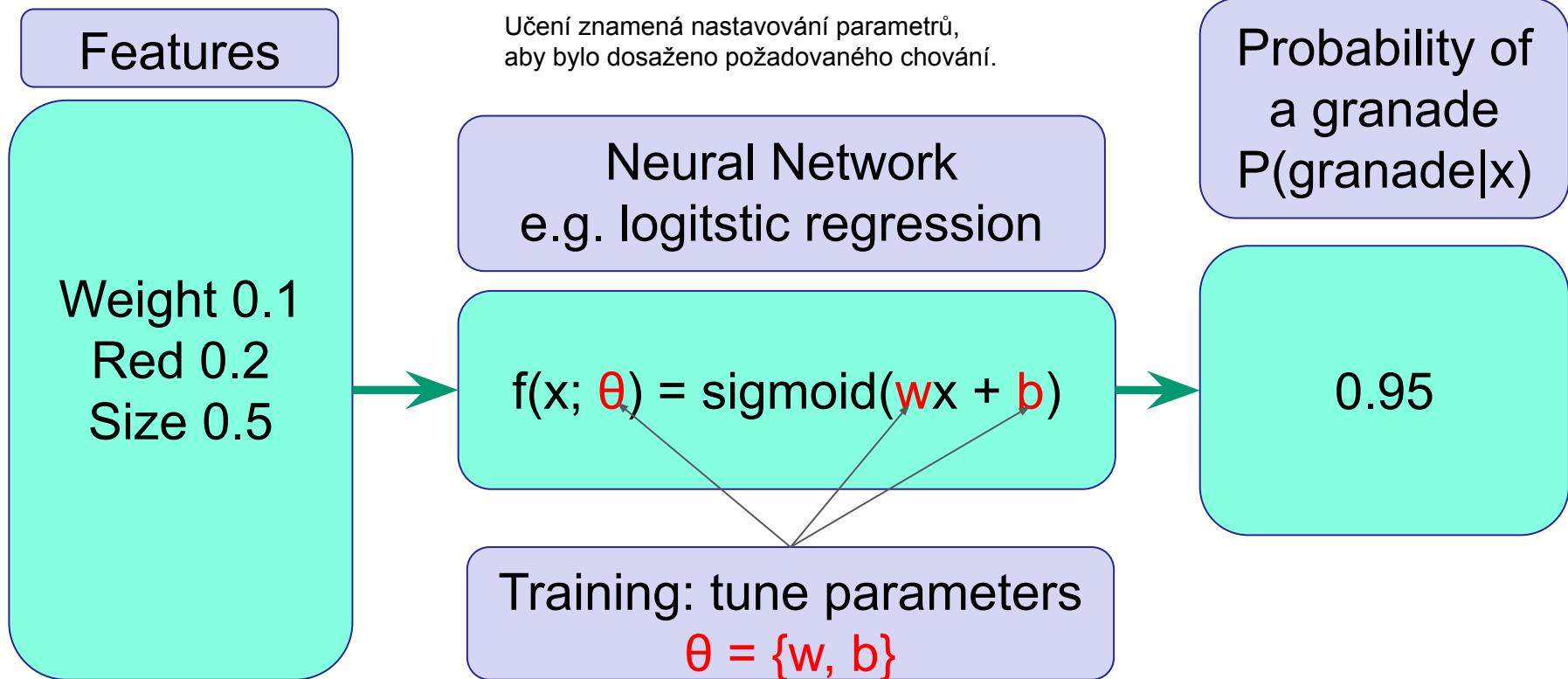
Neural Network
e.g. logistic regression

$$f(x; \theta) = \text{sigmoid}(wx + b)$$

Probability of
a grenade
 $P(\text{granade}|x)$

0.95

Binary classification network

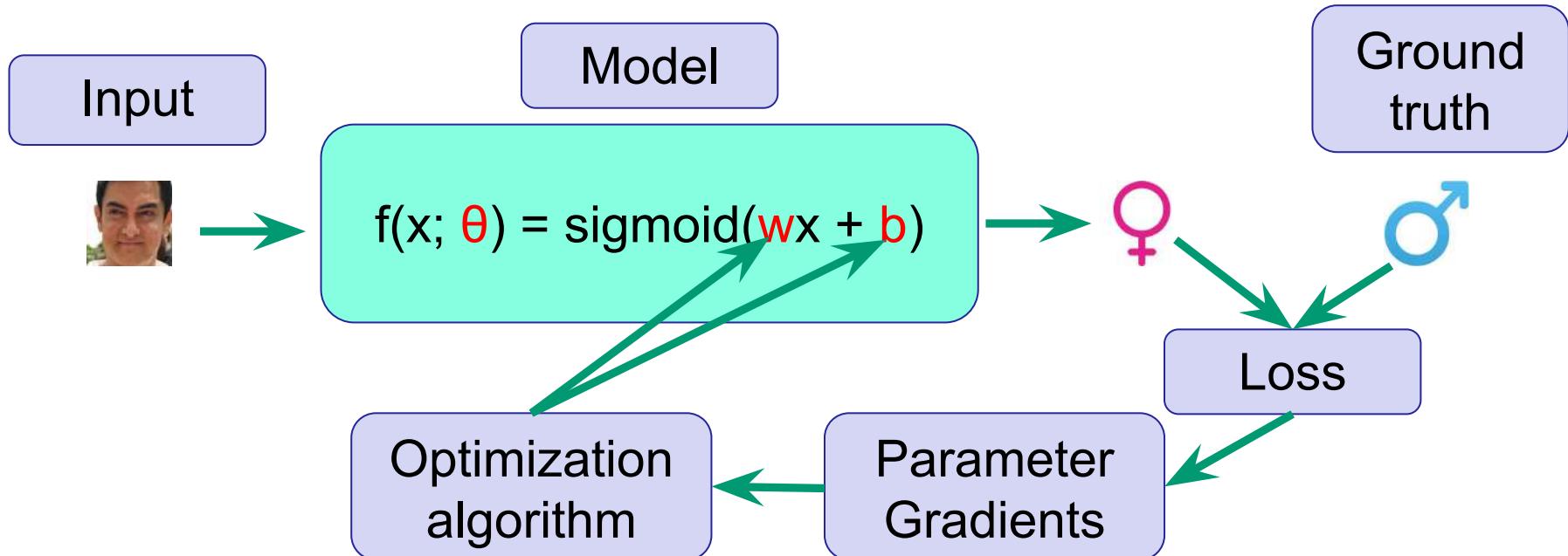


Training

Pro dosažení požadovaného chování jsou potřeba příklady správných výsledků (trénovací sada). V trénovací sadě bývá většinou ukázka vstupu a "doufejme" odpovídající správný výstup (ground truth, GT).

Příklady nestačí pro učení. Ještě je potřeba funkce, která dokáže porovnat výstup NN s GT -- loss funkce (nebo chybová funkce, někdy objektivní funkce, ...). Loss funkce má na vstupu výsledek sítě a GT, na výstupu má jedno reálné číslo. Loss funkci je pro konkrétní vstup a GT možné derivovat podle parametrů NN. Tedy spočítat gradient.

Gradient je pak použit v optimalizačním algoritmem pro změnu parametrů sítě tak.



Supervised learning components

Training data - inputs and desired network outputs

$$D = \{(\text{}, \text{♂}), (\text{}, \text{♀}), (\text{}, \text{♂}), (\text{}, \text{♀})\}$$

Model with parameters (network)

$$f(x; \theta) = \text{sigmoid}(wx + b)$$

Loss function (e.g. cross entropy)

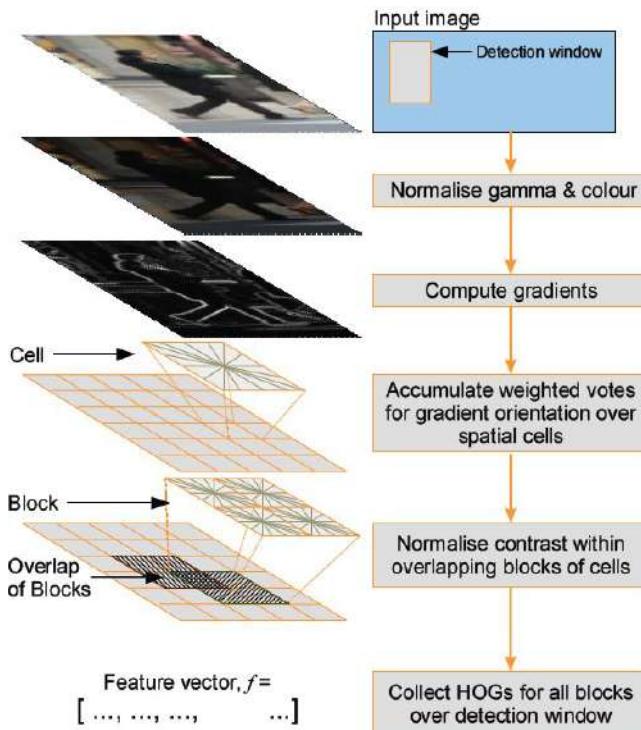
$$\text{loss}(x, y) = \begin{cases} \log(f(x; \theta)) & \text{if } y = \text{male} \\ \log(1 - f(x; \theta)) & \text{if } y = \text{female} \end{cases}$$

Objective function

$$J(D, \theta) = \sum_{D=\{(x_i, y_i), \dots\}} \text{loss}(f(x_i, \theta), y_i)$$

“Normální” datová sada je množina dvojic vstupů a správných výstupů (GT), na které jsou přímo optimalizovány parametry NN. Já používám pojem “objective function” J jako celkovou míru chyby na celé trénovací sadě. Při trénování NN se minimalizuje hodnota funkce J. J je vždy suma chyb jednotlivých příkladů. To, že je to suma, vychází z předpokladu, že jednotlivé trénovací vzorky (a tedy i jejich chyby) jsou nezávislé. (Ostatně kdyby J nebyla sumou jednotlivých loss, byl by s optimalizací problém.)

“Shallow” learning



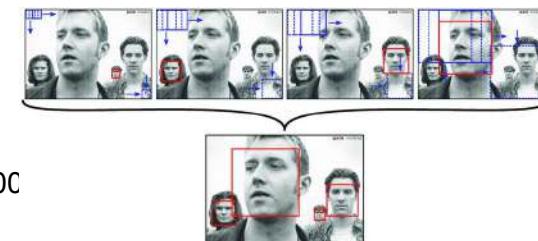
Shallow učení je spojení jednoduchého učícího algoritmu (např. logistická regrese, SVM, ...) s většinou ručně navrženým algoritmem extrakce příznaků. Samotný pojem “shallow” můžeme chápat i tak, že ML model má jednu vrstvu (nebo malý počet - např. nelineární SVM). Kvůli tomu se toho nemůže moc naučit, případně je výpočetně neefektivní pro složitější výpočty (úlohy).

Aby takový postup fungoval, musí extrahované příznaky vhodně reprezentovat užitečnou informaci pro řešenou úlohu. Například původní pixely obrázku moc dobrou reprezentací nejsou. Je jich hodně, jsou korelované (hodně pixelů zachycuje stejnou informaci) a není v jejich prostoru jednoduché oddělit například obličeje od “pozadí”. Návrh vhodné extrakce příznaků je náročná. Člověk musí chápat řešený problém (například vlastnosti obrazu, diskriminativní charakteristiky rozpoznávaných tříd, vůči čemu by měl být model invariantní, ...). Lidské znalosti většinou pokulhávají a ruční návrh extrakce příznaků se zvrhne v částečně náhodné zkoušení různých variant extraktoru.

Zároveň je obtížné taký přístup škálovat. Pokud získám 10 větší trénovací sadu třeba pro složitější problém (budu třeba detekovat obličeje i v profilu), může být neřešitelný problém rozšířit extraktor, tak aby tato data dokázal využít a složitější úlohu dostatečně dobře vyřešit.

Jsou možnosti i učení extraktoru příznaků (např. PCA), ale to také moc nefunguje.

$$f([features]; \theta) = \text{sigmoid}(w[features] + b)$$



Hluboké učení využívá větších ML modelů, které mají hodně "úrovní" zpracování. V dnešní době jsou to neuronové sítě, ale není to jediná možnost. Mozek je také "hluboký". Jenda úroveň zpracování je většinou lineární funkce s učitelnými parametry následovaná nelinearitou. Hodně takový úrovní/kroků zpracování zajišťuje výpočetní efektivitu modelu. Zároveň je možné takové modely jednoduše zvětšovat (a zmenšovat) s tím, kolik je k dispozici trénovacích dat, o jak složitý problém se jedná a jak jsou omezeny výpočetní prostředky.

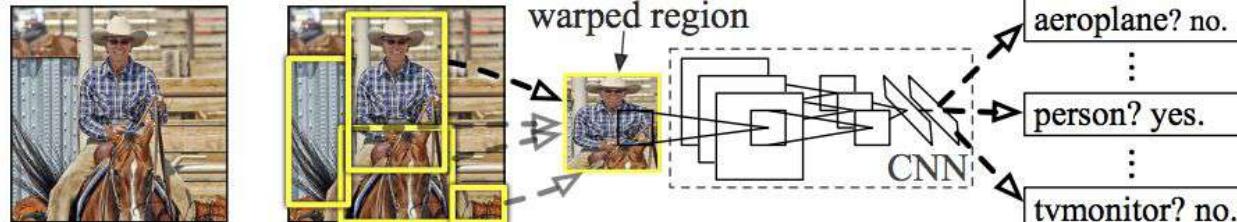
Při hlubokém učení se minimalizuje krok extrakce příznaků. Většinou jsou vstupem modelu původní data (nebo jen minimálně předzpracovaná). "Předzpracování" se naučí samotný model.

Většinou je u hlubokého učení potřeba více trénovacích (model je větší), ale je možné dosáhnout lepších výsledků.

Deep learning

- No hand coded feature extraction
- Our domain knowledge goes into network architecture, regularization, loss, data augmentation, ...

R-CNN: *Regions with CNN features*



1. Input
image

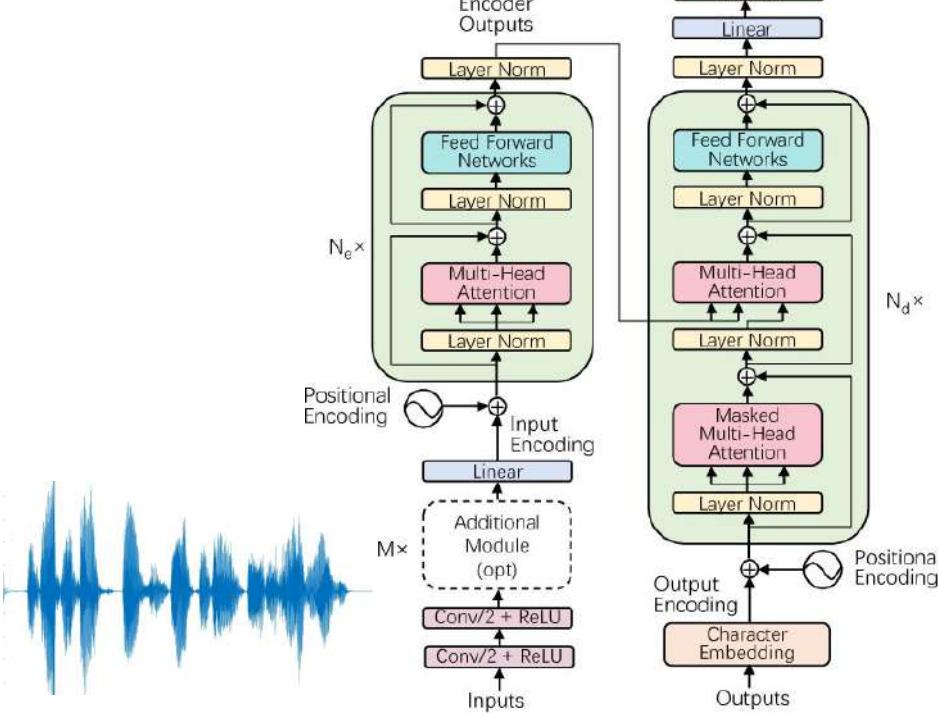
2. Extract region
proposals (~2k)

3. Compute
CNN features

4. Classify
regions

End to end speech recognition/translation

„Honza šel na jahody“



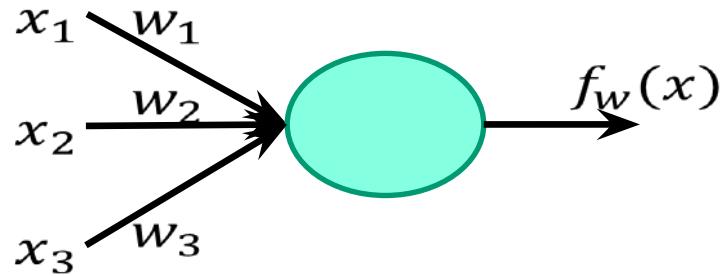
End-to-end učení - přístup, kdy velmi složitou úlohu učím v jednom kroku, tedy vstupem jsou raw data, chyba se počítá na cílové úloze, a jeden model se celý optimalizuje pro dosažení celkové minimální chybě.

Bylo by možné složitou úlohu rozdělit na části a ty učit zvlášť. Například v tomto případě bych mohl mít akustický model, který detekuje fonémy. Jazykový model, který rozumí jazyku a dokáže rozlišit nejisté fonémy. A překladač do jiného jazyka.

Výhody a nevýhody end-to-end učení:

- + Potřebuju GT data jen pro cílovou úlohu, která mě ve skutečnosti zajímá
- + Celý systém je jednodušší. Je to jen jedna NN, ne kolekce samostatných systémů.
- + Je možné dosáhnout lepších výsledků, protože celý model je optimalizován pro cílové použití. Nemůže se třeba stát, že natrénuju fonémový rozpoznávač, který nezná „ř“ a budu ho používat pro češtinu. Obecně se nestane, že by se v počátečních částech zpracování ztratila informace podstatná pro řešenou úlohu.
- + Někdy může být takový systém i výpočetně efektivnější, protože se učí počítat jen věci důležité pro cílovou úlohu.
- Většinou je potřeba více trénovacích dat.
- Většinou se takové modely déle trénují.
- Může být složitější interpretovat chování takového modelu.

Neural network components (neurons)

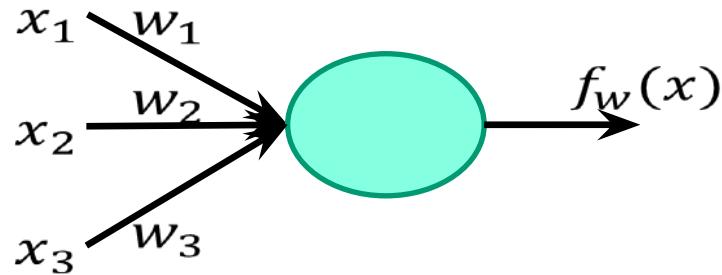


$$\begin{aligned}f(\mathbf{x}; \mathbf{w}) &= w_1x_1 + w_2x_2 + w_3x_3 \\&= \sum_i w_i x_i\end{aligned}$$

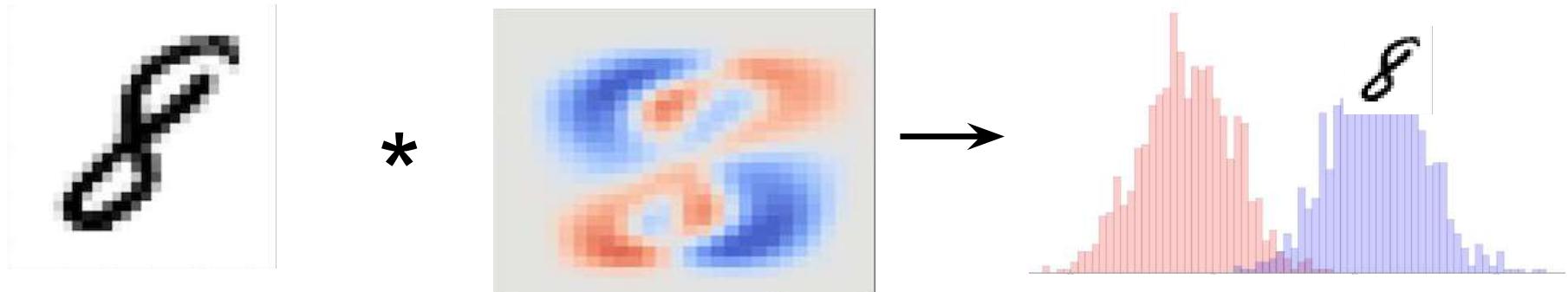
Umělý neuron, který je stavebním prvkem velká části neuronových sítí je jednoduchá lineární funkce (vážený součet). Vektor vah \mathbf{w} (weights) jsou parametry, které se učí. Vstup \mathbf{x} je vektor reálných čísel.

Neurons as templates

Neurons have higher activations if the input “matches” their weights.



3 5 4

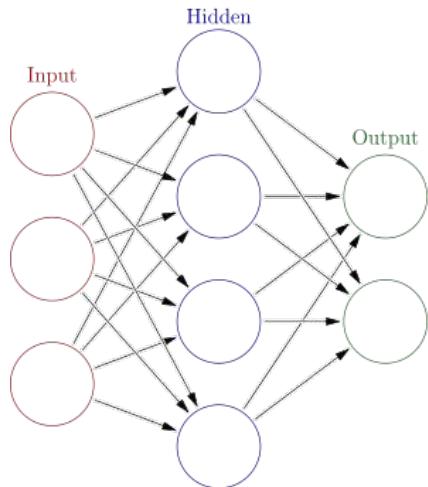


Váhy neuronu je monžné chápat jako “vzor”/template. Pokud je vstup podobný vahám, má výstup takového lineárního neuronu velkou hodnotu. Pokud omezíme “energi” \mathbf{w} a \mathbf{x} třeba tak, že délka těchto vektorů bude 1, bude mít neuron největší odezvu, právě pokud $\mathbf{w} = \mathbf{x}$.

Ukázka pro 8 - z pixelů obrázku můžu udělat vektor třeba tak, že dám řádky obrázku postupně za sebe.

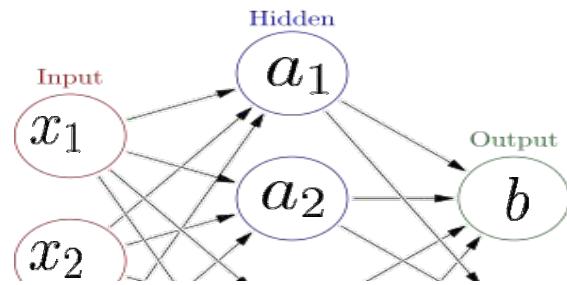
Composition of linear functions?

$$f(\mathbf{x}; \mathbf{w}) = \sum_i w_i x_i$$



Neuronové sítě jsou hluboké, mají vrstvy,
výstupy neuronů jsou zpracovávány dalšími neurony.

Composition of linear functions?



$$f(\mathbf{x}; \mathbf{w}) = \sum_i w_i x_i$$

$$a_1 = 1x_1 + 2x_2$$

$$a_2 = -1x_1 + 1x_2$$

$$\underline{b = 4a_1 + 1a_2}$$

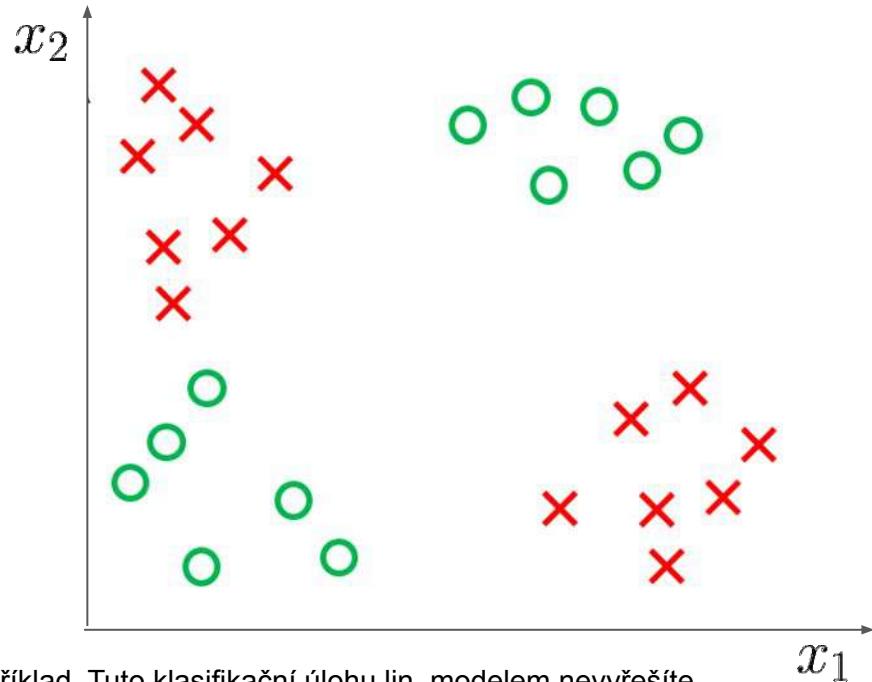
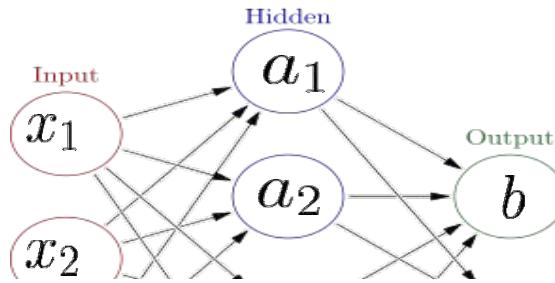
$$= 4(1x_1 + 2x_2) + 1(-1x_1 + 1x_2)$$

$$\underline{= 3x_1 + 3x_2}$$

Otázka je, jestli kompozicí lineárních neuronů vznikne silnější model než je jen jeden neuron.

Odpověď je, že nevznikde. Lineární funkce (neurony) je vždy možné "roznásobit".

Linear separability and composition of functions



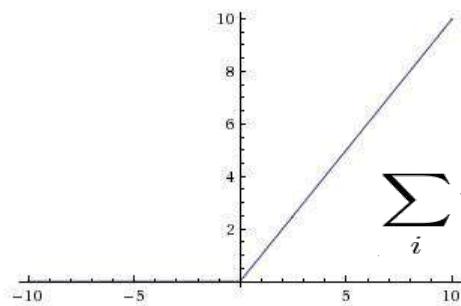
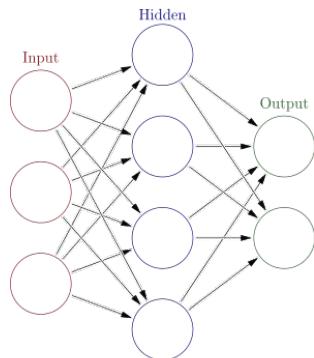
Pro představu o omezení lineárních modelů je například tento příklad. Tuto klasifikační úlohu lin. modelem nevyřešíte.

Non-linearity

(more at https://en.wikipedia.org/wiki/Activation_function)

Non-linearities mostly process each activation independently.

One example is ReLU (rectified linear. unit).

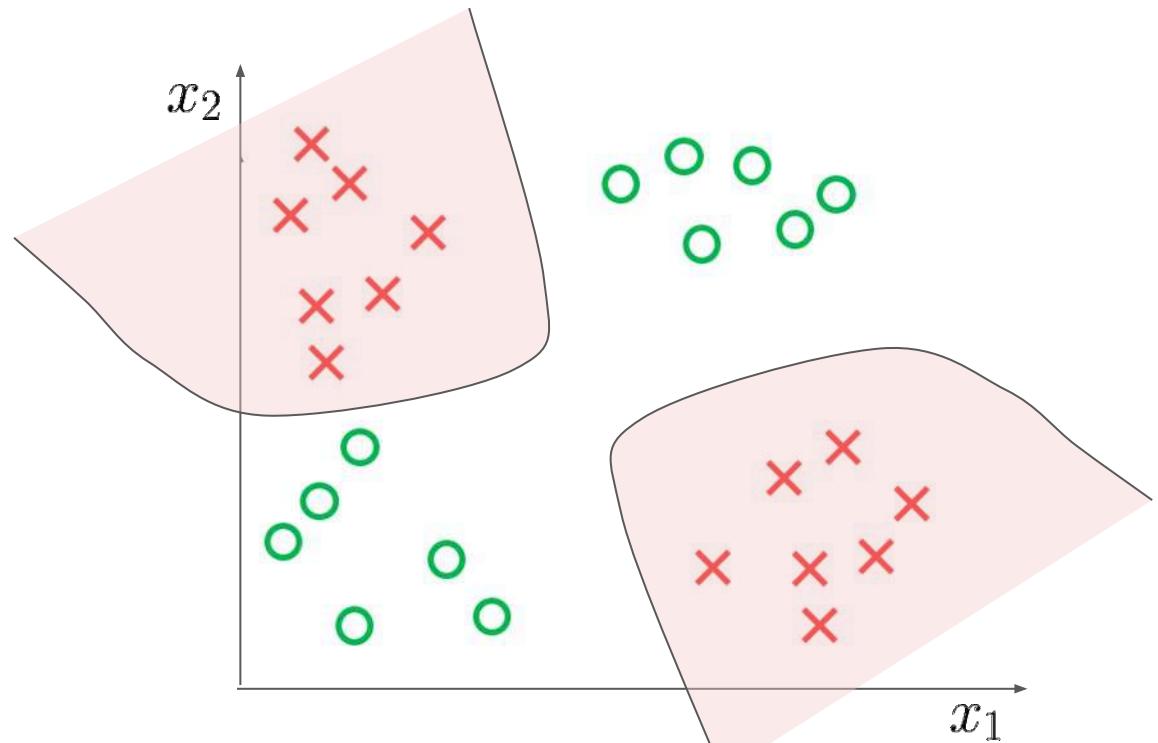
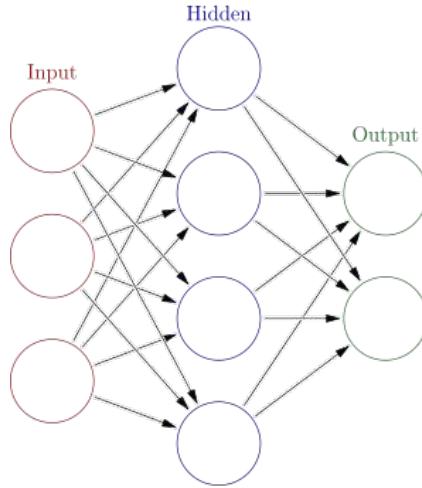


$$f(x; w) = \sigma \left(\sum_i w_i x_i \right)$$
$$f(x; w) = \max \left(\sum_i w_i x_i, 0 \right)$$

Aby měla kompozice neuronů do složitějších (hlubších) neuronových sítí smysl, musí síť obsahovat nelineární funkce. Nelinearity, aktivační funkce - Jsou to jednoduché funkce, které většinou zpracovávají výstup každého neuronu samostatně (ne všechny - softmax, maxout, max-pooling, batch normalization).

Výpočetně jsou aktivační funkce velmi jednoduché a síť nijak výrazně nezpomalují. Hlavními výpočetními jednotkami NN jsou právě lineární neurony.

Linear separability and composition of functions

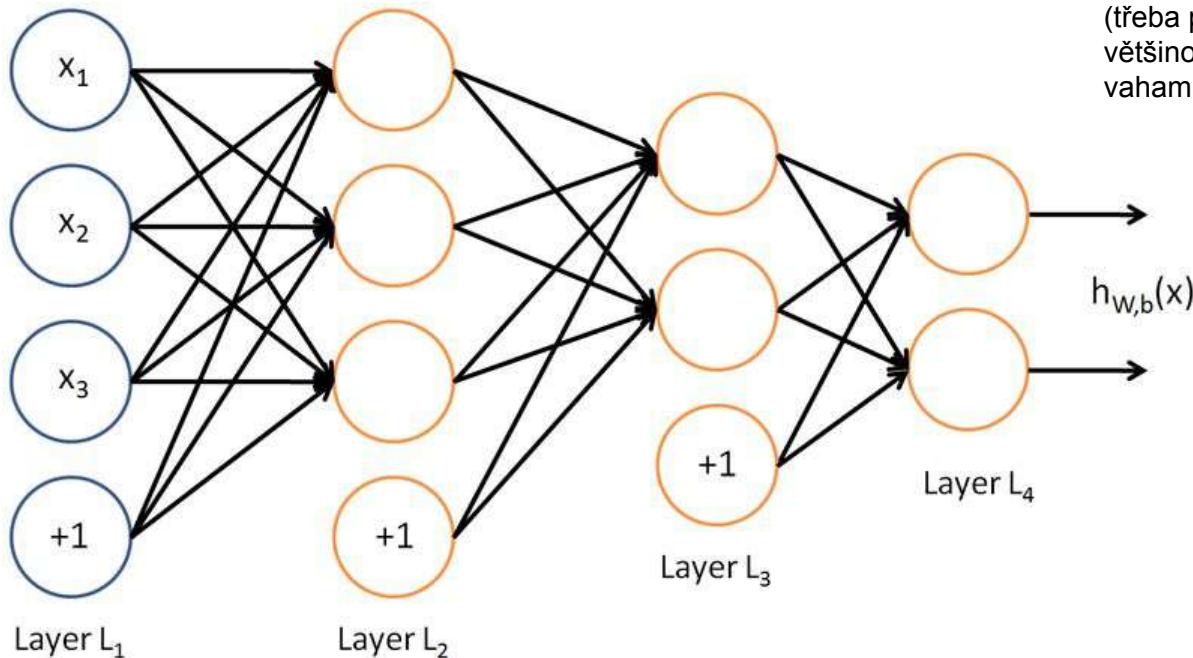


Pro představu o omezení lineárních modelů je například tento příklad. Tuto klasifikační úlohu lin. modelem nevyřešíte.

From neurons to layers

Aby jsme byli schopni popisovat (a definovat) neuronové sítě, sdružují se neurony do vrstev.

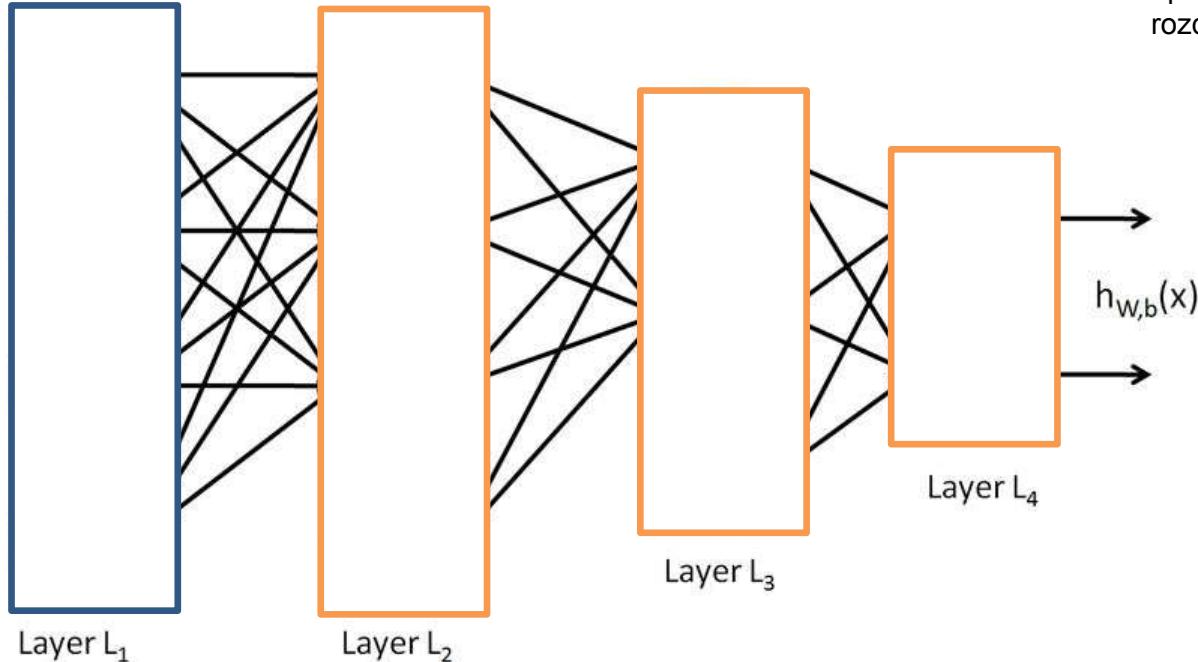
Vrstva je skupina strukturně stejných výpočtů (třeba právě lineárních neuronů), které jsou většinou na sobě nezávislé. Můžou se lišit vahami.



From neurons to layers

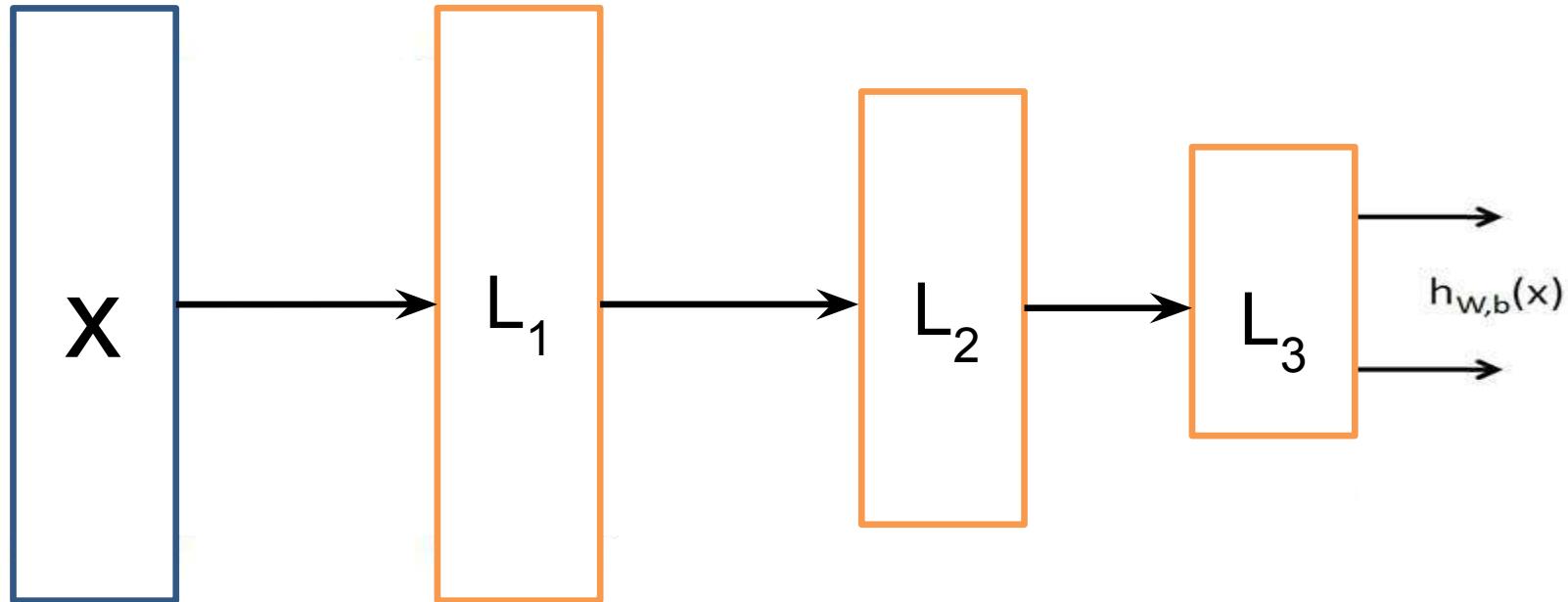
Na jednotlivé neurony pak můžeme zapomenout a bavit se jen o vrstvách.

Vrstva - Je to velké množství matematických operací stejného typu, které zpracovávají rozdílná data



Layers

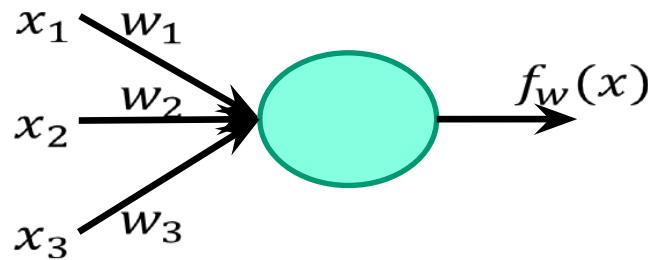
Often, we don't visualize the layers, but their inputs and outputs (activations).



From neurons to matrices

We would like to define layer of neurons as single mathematical operation (as a linear layer).

Solution: Linear algebra



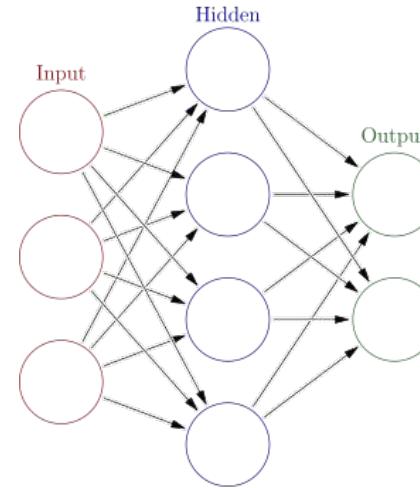
$$\begin{aligned}f(\mathbf{x}; \mathbf{w}) &= w_1x_1 + w_2x_2 + w_3x_3 \\&= \sum_i w_i x_i\end{aligned}$$

Jeden lineární neuron můžeme reprezentovat jako skalární součin vektoru vstupů \mathbf{x} a vektoru vah \mathbf{w} .

$$[1 \ 0 \ 3] \times \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} = -1$$

Linear layer

$$\begin{bmatrix} 1 & 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} = -1$$



$$\begin{bmatrix} 1 & 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -1, 0, -1, -1$$

Jedna vrstva tak násobí \mathbf{x} nekolika různými vektory vah \mathbf{w}_i .

Linear layer

$$\begin{bmatrix} 1 & 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -1, 0, -1, -1$$



$$\begin{bmatrix} 1 & 0 & 3 \end{bmatrix} \times \begin{bmatrix} 2 & 0 & -1 & 1 \\ -1 & 2 & 0 & 1 \\ -1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & -1 & -1 \end{bmatrix}$$

$$f(\mathbf{x}; \mathbf{W}) = \mathbf{W}\mathbf{x}$$

Vektory vah je možné sloučit do jedné matice vah \mathbf{W} a celou vrstvu spočítat jako jediné maticové násobení.

Taková vrstva se nazývá **lineární** nebo **plně propojená** (fully connected).

Neural network in code

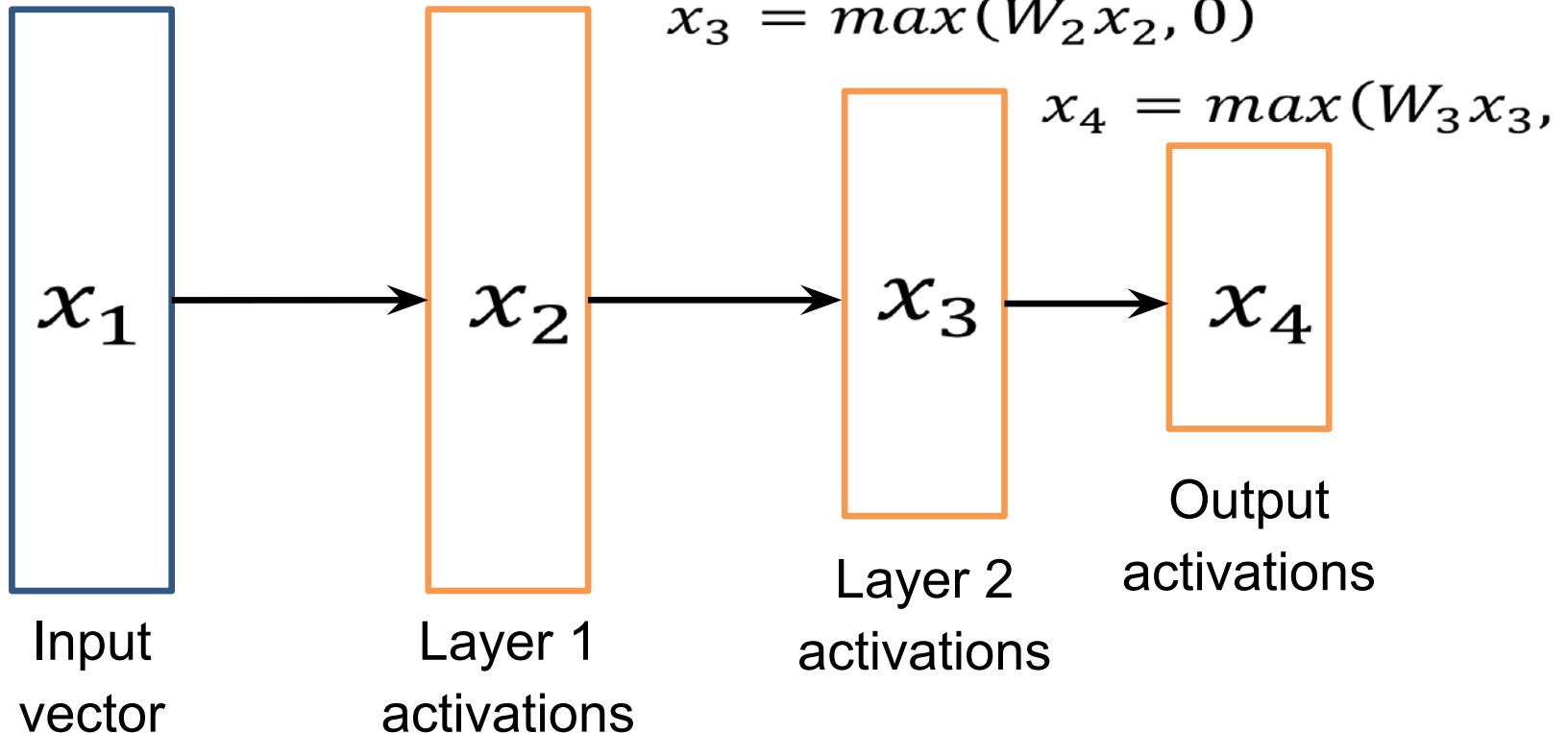
Celou složitější neuronovou síť tak je možné zapsat jen pomocí maticových násobení a nelinearit.

Toto je zápis velmi podobný například kódů v pytorch nebo tensorflow.

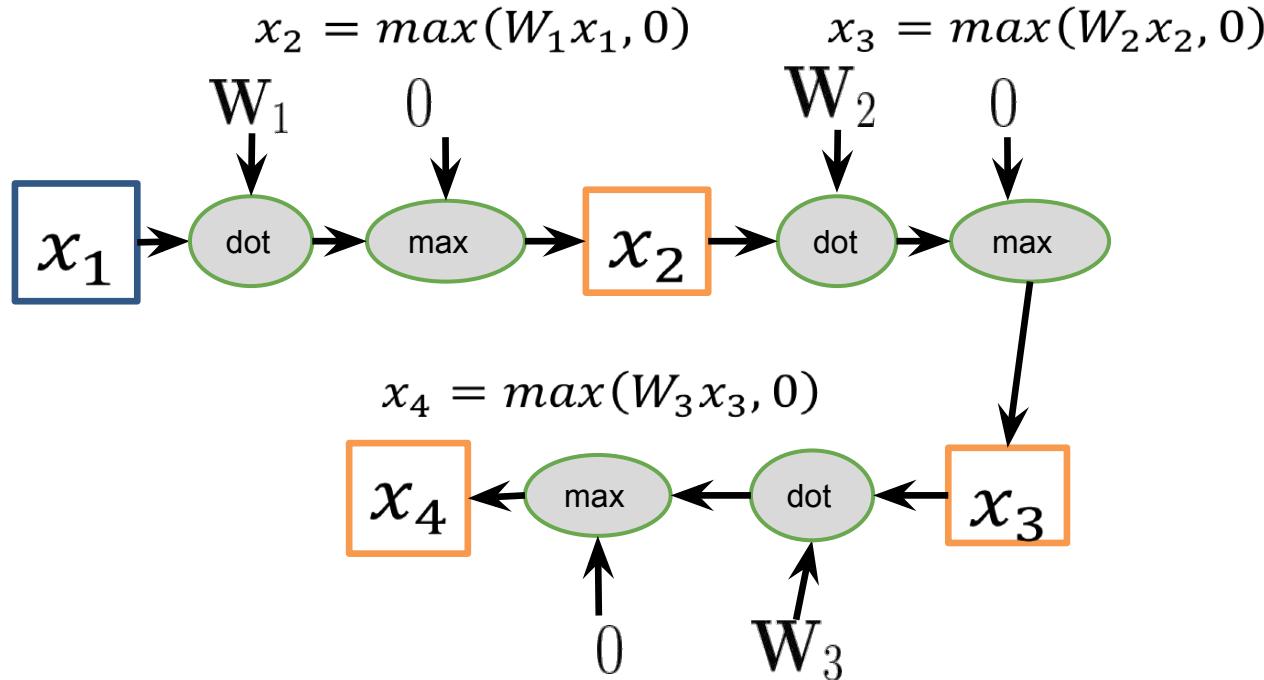
$$x_2 = \max(W_1 x_1, 0)$$

$$x_3 = \max(W_2 x_2, 0)$$

$$x_4 = \max(W_3 x_3, 0)$$



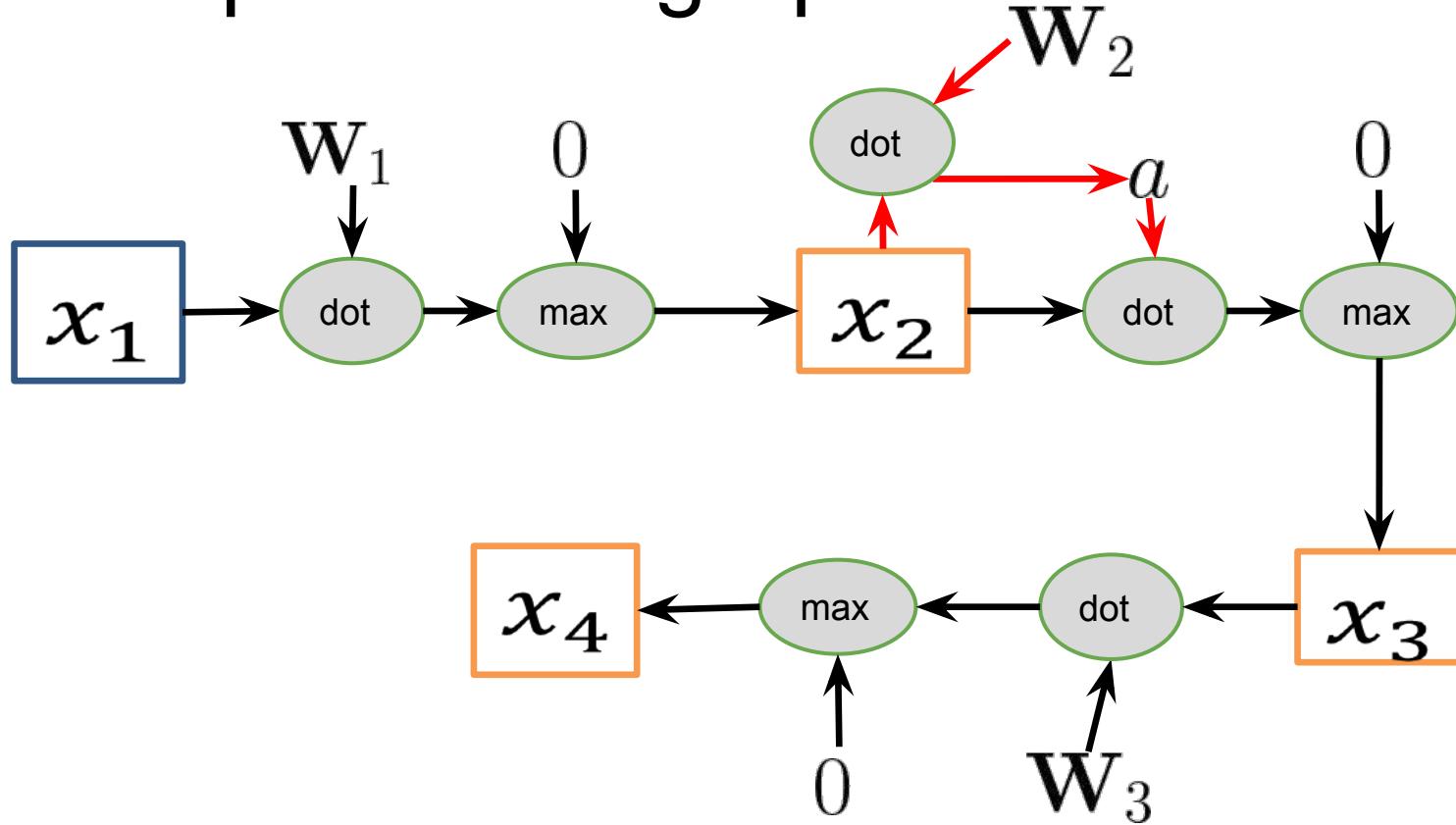
Computational graph



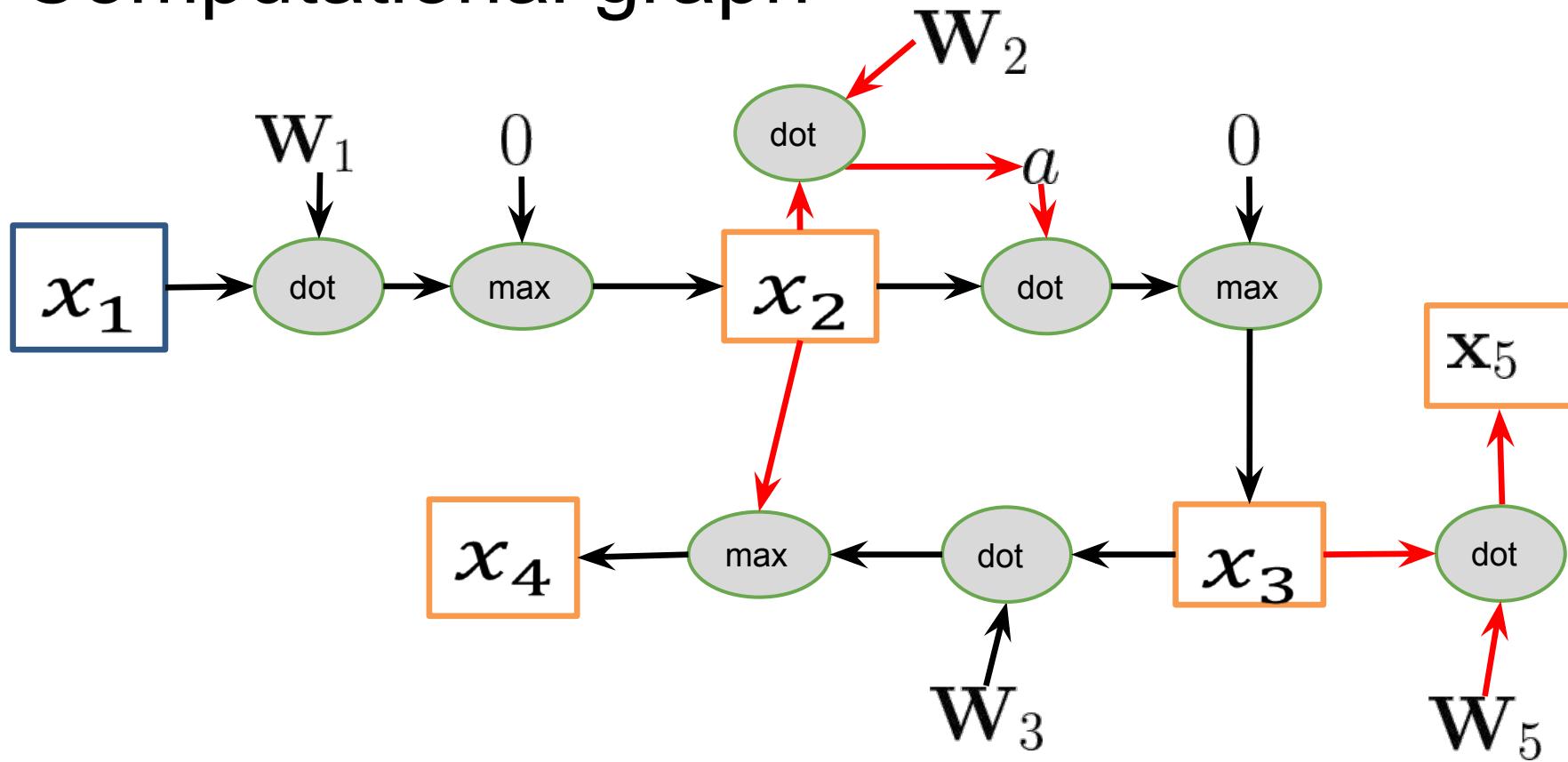
V těchto přednáškách se bavíme o "dopředných" neuronových sítích. Existují i další (například boltzmann machines nebo jiné energy models), ale dopředné sítě jsou v současnosti nejrozšířenější, dosahují nejlepších výsledků ve většině úloh a celkově je dobré pochopit hlavně je.

Výpočetní graf (computational graph) je grafickou prezentací funkcí. Vyjadřuje tok dat mezi jednoduchými matematickými operacemi. Výpočetní graf je acyklický orientovaný graf, který ale jinak může mít libovolný počet vstupů a výstupů. Zároveň je dobré, by bylo použité matematické operace možné derivovat kvůli učení (není to úplně podmínka, ale jsou s tím pak problémy).

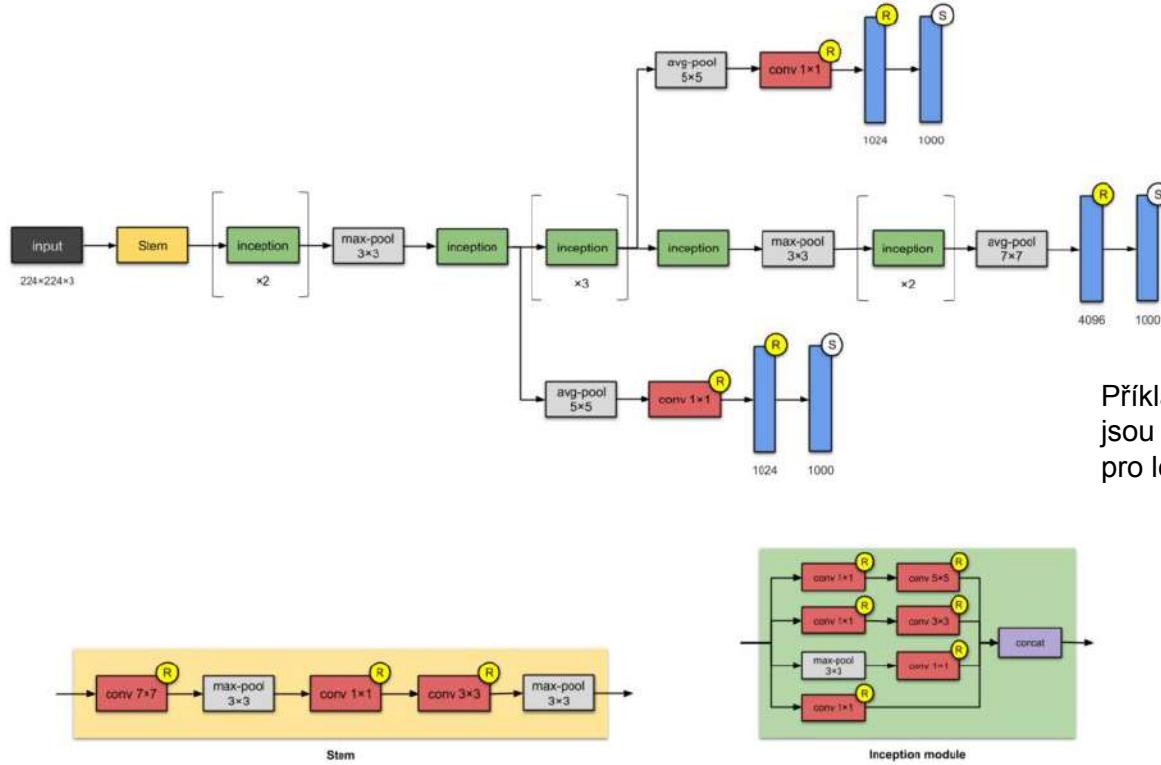
Computational graph



Computational graph



Real world example: Inception-V4



Příklad starší reálné sítě s více výstupy (které jsou ale všechny stejné a které jsou tady jen pro lepší trénování).

Source: raimibinkarim, Illustrated: 10 CNN Architectures

<https://makerspace.aeisincapera.org/2019/09/illustrated-10-cnn-architectures/>

Loss functions

- During training, loss function measures how network output differs from the desired output.
- It is important to use the correct loss function for the task.
- It is important to use correct activation function for the task.
- Multiple loss/activation pairs can usually be used for a single task, but choice affects, what the network learns.
(You change the tasks definition.)

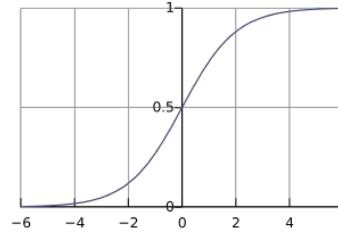
Binary classification

Goal: get $p(c | \text{data})$

Network output: $p(c=1|\text{data})$

Binary problem: $p(c=2|\text{data}) = 1 - p(c=1|\text{data})$

Activation function: sigmoid



$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

What loss function?

Pro binární klasifikaci můžeme mít nekonečně loss funkcí. Binární klasifikace - rozhodnutí z které ze dvou tříd data pocházejí.

Pokud chceme, aby výsledkem byly pravděpodobnosti daných dvou tříd, tedy $p(c | x)$, je potřeba použít variatnu, kterou už jste viděli v logistické regresi.

Síť má na výstupu jedno číslo (pravděpodobnosti jedné ze tříd), protože $p(c = 1 | x) = 1 - p(c = 0 | x)$.

Na výstupu síť musí být aktivační funkce sigmoida.

Loss funkce je Cross entropy (křížová entropie). V pytorch a jinde je to funkce (třída) binary cross entropy. Je to "normální" cross entropy, ale předpokládá, že na vstupu má právě pravděpodobnost jen jedné třídy.

Pro binární klasifikaci může mít síť i dvě hodnoty na výstupu (pravděpodobnosti obou tříd) s aktivační funkcí softmax. Po pravdě nevím, jestli je v tom prakticky nějaký rozdíl.

Binary classification

(same as in logistic regression)

Goal: get $p(c | \text{data})$

Network output: $p(c=0|\text{data})$

Binary problem: $p(c=1|\text{data}) = 1 - p(c=0|\text{data})$

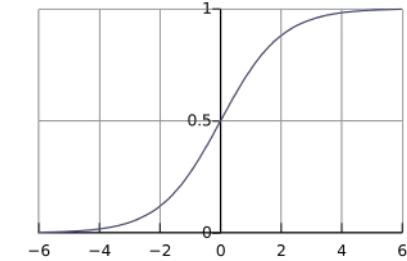
Activation function: sigmoid

Task maximize training set likelihood:

$$P(t|D) = \prod_D P(t|x_i) = \prod_D f(x_i)^{t_i} (1-f(x_i))^{1-t_i}$$

Minimize binary cross-entropy loss (log-likelihood):

$$\begin{aligned} J(f, D) &= \sum_{i \in D} -\ln(P(t|x_i)) = \\ &\sum_{i \in D} t_i \ln(f(x_i)) + (1 - t_i) \ln(1 - f(x_i)) \end{aligned}$$



Rovnice:

$$P(t|D) = \prod_{i \in D} P(t|x_i) = \prod_{i \in D} f(x_i)^{t_i} (1-f(x_i))^{1-t_i}$$

$$\begin{aligned} J(f, D) &= \sum_{i \in D} -\ln(P(t|x_i)) = \\ &\sum_{i \in D} t_i \ln(f(x_i)) + (1 - t_i) \ln(1 - f(x_i)) \end{aligned}$$

Binary classification - SVM loss

Goal:

$$\begin{aligned} f(x) > 1 &\quad \text{if } c = 1 \\ f(x) < -1 &\quad \text{if } c = -1 \end{aligned}$$

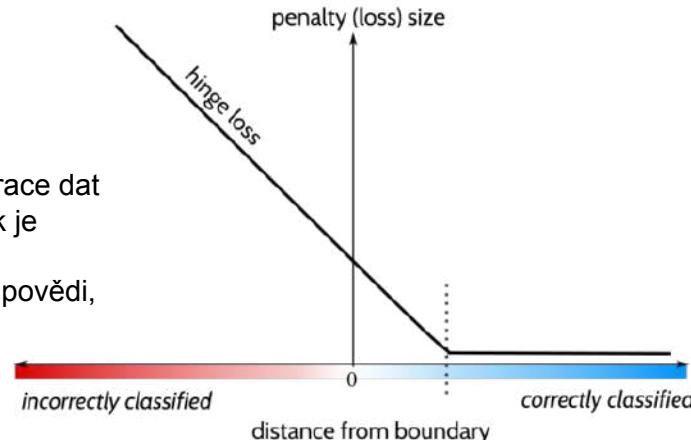
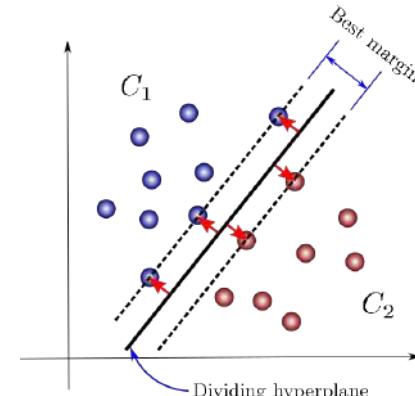
Activation function: linear (none)

$$J(f, D) = \sum_{i \in D} \max(0, 1 - c_i f(x_i))$$

Loss function: Hinge loss

Binární klasifikaci je možné definovat i tak, že cílem je jen kvalitní separace dat jednotlivých tříd a pravděpodobnosti nejsou důležité. Takový požadavek je možné například formalizovat pomocí hinge loss funkce (odborně asi "hokejka"), která pochází ze Support Vector Machines. Ta penalizuje odpovědi, které nejsou klasifikovány správně alespoň o "margin". Vše, co už je klasifikováno dostatečně správně, nemá vliv na hodnotu loss.

$$J(f, D) = \sum_{i \in D} \max(0, 1 - c_i f(\mathbf{x}_i))$$



Multi-class classification

Goal: get $p(c | \text{data})$ with number of classes > 2



0.1	Michal
0.0	Martin
0.5	Petra
0.3	Katka
0.1	Honza
0.0	Bětka

Activation function: soft-max

Loss: Multi-class cross-entropy

$$\sigma_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Multi-class klasifikace (více-třídní). Klasifikace do více než dvou tříd.

Pokud zase mají být výsledkem pravděpodobnosti, je na výsledku sítě číslo pro každou třídu (vektor se stejným počtem dimenzí jako je počet tříd). Aktivační funkce je softmax. Loss je zase křížová entropie.

$$\sigma_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Multi label classification

Goal: get $p(c | \text{data})$ for multiple binary problems



0.1	Asian
1.0	Male
0.9	Black hair
0.3	Actor
0.2	Child
1.0	Photographed outdoors

Activation function: sigmoid
Loss: Binary cross-entropy

Multi label klasifikace - několik nezávislých binárních problémů. Takže pro každou třídu jeden výstup se sigmoidou a samostatnou binární cross entropií. Aby bylo všechno jednodušší, vytvoří se z výstupů jeden vektor.

Regression

Goal: get $E[p(\text{value} | \text{data})]$ + gaussian noise in data

Activation: Linear (none)

Loss: Mean squared error

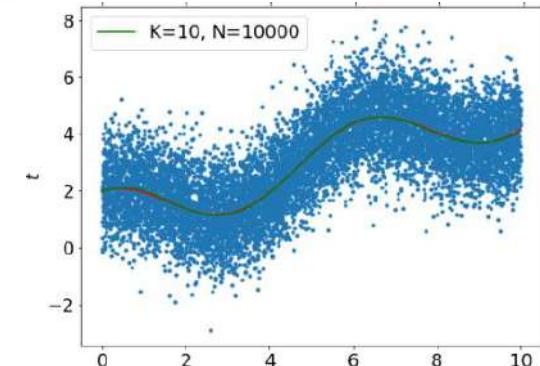
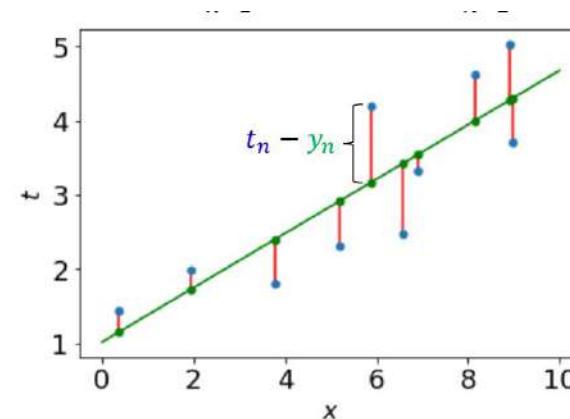
$$\sum_{i \in D} \frac{1}{2} (f(x_i) - t_i)^2$$

Regres - nad výstupními hodnotami je definované uspořádání - některé hodnoty jsou k sobě blíž a některé dále. Většinou je regrese nad "spojitou" proměnnou - float.

Pokud je cílem predikovat střední hodnotu rozložení možných výstupů $E[p(\text{value} | \text{data})]$, tak je správnou loss funkcí mean squared error (ten mean se někdy vynechává).

Na výstupu není žádná aktivace (lineární aktivace). Asi nikdy nemá smysl. Nejspíš nemá smysl ani když jsou hodnoty omezené.

$$\sum_{i \in D} \frac{1}{2} (f(x_i) - t_i)^2$$



Regression

Goal: get $E(p(\text{value} | \text{data})) + \text{lot of non-gaussian outliers}$

Activation: Linear (none)

Loss: Sum of Absolute Difference

$$\sum_{i \in D} |f(x_i) - t_i|$$

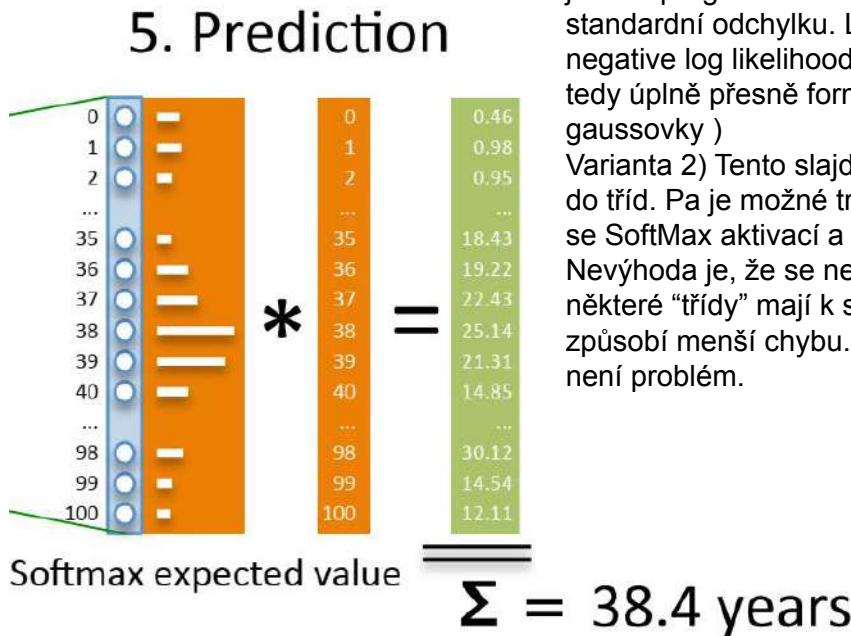
Například pokud bude v datech hodně "ústřelků", například chybně anotovaných příkladů, není MSE vhodná chybová funkce, protože kvůli druhé mocnině mají velké chyby velký vliv na výsledek - vliv (velikost derivace chyby) roste lineárně s velikostí chyby.

V takových případech může být vhodná chybová funkce, která je robustní vůči outlierům (potlačuje vliv velkých chyb). Jednou možností je sum of absolute differences (SAD) - je to to samé jako MSE, ale s mocninou 1. SAD se může označovat jako L1 loss a MSE jako L2 loss. Ve skutečnosti existuje celá rodina takových chybových funkcí s libovolnou mocninou > 0 (L_p loss family). Čím víc se mocnina přibližuje 0, tím více ignoruje velké chyby. Pro 1 mají všechny chyby stejnou váhu. Pro 2 je to MSE. Vyšší mocniny mohou mít také smysl, ale spíše ne. Dále M-estimators, Huber loss, ...

Regression - with discretization

Goal: get $p(\text{value} | \text{data})$

Activation: Softmax



Předchozí regrese s predikcí $E(p(\text{value} | \text{data}))$ má problém, že neposkytuje informaci o nejistotě predikce. Řešením může být predikovat celé rozložení $p(\text{value} | \text{data})$.

Varianta 1) predikovat parametrické rozložení jako např. gaussovku. Tedy navíc predikovat standardní odchylku. Loss funkce pak je negative log likelihood s danou gaussovou - tedy úplně přesně formálně --- $-\log(\text{odezva gaussovky})$

Varianta 2) Tento slajd. - diskretizovat výstup do tříd. Pa je možné trénovat jako klasifikátor se SoftMax aktivací a cross entropy loss. Nevýhoda je, že se nevyužívá informace, že některé "třídy" mají k sobě blíž a tedy třeba způsobí menší chybu. Při dostatku dat to ale není problém.

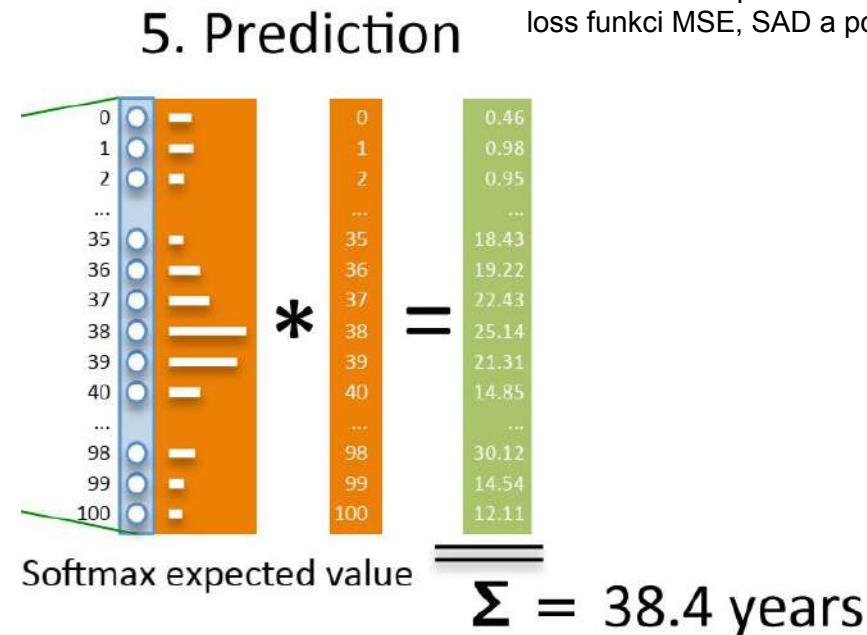
Regression - with discretization

Goal: get $E(p(\text{value} | \text{data}))$

Activation: Softmax
(with expectation sum)

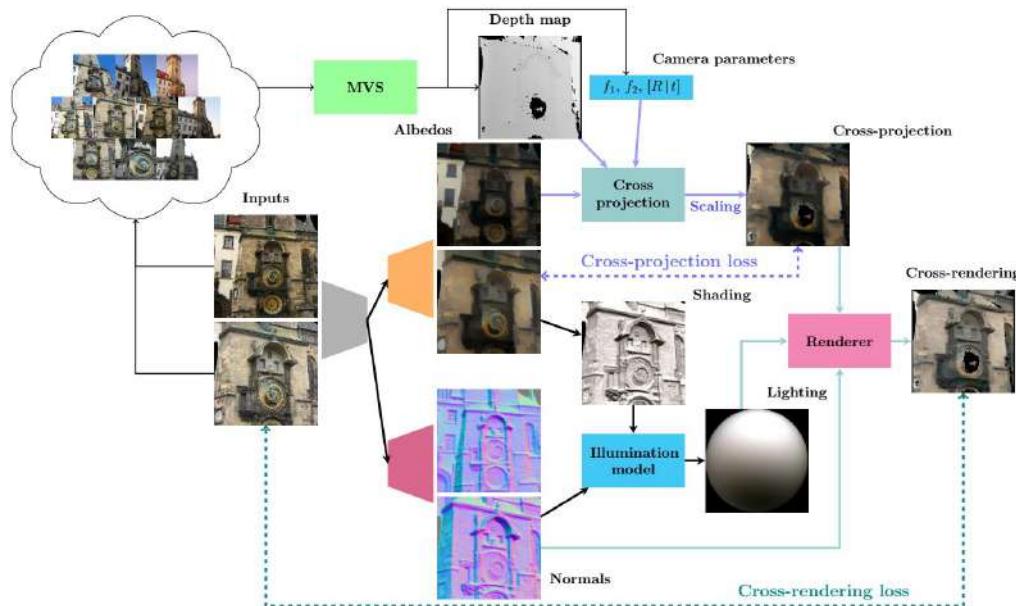
Loss: Mean squared
error?

Z predikce rozložení $p(\text{value} | \text{data})$ je možné spočítat hodnotu $E(p(\text{value} | \text{data}))$ podle definice střední hodnoty $E(p(\text{value} | \text{data})) = \sum [\text{value}.p(\text{value} | \text{data})]$, kterou je možné derivovat. Takže je možné takto pravděpodobnosti agregovat, získat střední hodnotu a na ni použít standardní regresní loss funkci MSE, SAD a podobně.



Weird loss functions

We train our network to minimise: $\ell = w_1 \ell_{\text{appearance}} + w_2 \ell_{\text{NM}} + w_3 \ell_{\text{albedo}} + w_4 \ell_{\text{cross-rend}} + w_5 \ell_{\text{albedo-smooth}} + w_6 \ell_{\text{albedo-pseudoSup}}$.



Chybové funkce můžou být velmi složité. Tady je uveden příklad, kdy součástí loss funkce je diferenciovatelný renderer - síť se učí rekonstrukci vlastností povrchu.

Training of neural networks

$$J(D, \theta) = \sum_{D=\{(x_i, y_i), \dots\}} loss(f(x_i, \theta), y_i)$$

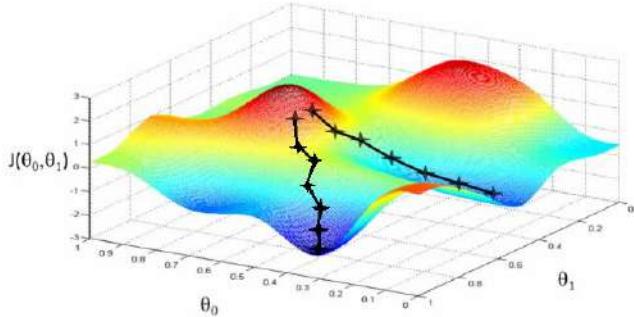
$$\theta^* = \arg \min_{\theta} J(\mathbf{D}, \theta)$$

Formálně je trénovaní neuronových sítí minimalizace objektivní funkce. Tedy hledání takových parametrů, které minimalizují sumu loss funkce přes celý trénovací dataset. Rovnice říká, že hledáme takové parametry, při které je výstup J minimální.

Tento minimalizační problém není možné vyřešit analyticky. Na optimalizaci je možné zvolit libovolný iterativní vhodný algoritmus. Třeba náhodné vyhledávání, coordinate descent, genetické algoritmy, stochastic search, ... Ale prakticky se většinou používají optimalizační metody prvního řádu (používají gradient) odvozené z algoritmu gradient descent.

$\hat{\theta} = \arg \min_{\theta} J(\mathbf{D}, \theta)$

Gradient Descent



$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(D, \theta)}{\partial \theta_j}$$

$$J(D, \theta) = \sum_{D=\{(x_i, y_i), \dots\}} loss(f(x_i, \theta), y_i)$$

Gradientní sestup (gradient descent, GD) optimalizuje parametry v iteracích. V každé iteraci se spočítá gradient (parciální derivace objektivní funkce vůči všem optimalizovaným parametrům neuronové sítě). Parametry se upraví tak, že se od nich odečte hodnota gradientu násobená učící konstantou. Učící konstanto ovládá rychlosť optimalizace. Změny parametrů v iteracích musí být dostatečně malé, aby nepřeskočily do oblasti, kde je naopak chyba výrazně větší - může pak nastat divergence učení - chyba nekontrolovatelně roste k nekonečnu. Pro malé kroky (malé hodnoty učící konstanty) může zase učení trvat neúnosně dlouho. Gradient popisuje chování optimalizované funkce jen v blízkém okolí. Co je velký / malý krok není možné obecně říct. Záleží na složitosti a dalších vlastnostech optimalizované funkce. Obecně může být být optimální učící konstanta GD pro různé úlohy výrazně jiná. Musí se to vyzkoušet.

Existují Adaptivní verze gradientního sestupu (např. Adam), kde optimální učící konstanta je téměř vždy ve stejném malém rozmezí. Takové optimalizační algoritmy je vhodné používat skoro vždy jako první možnost.

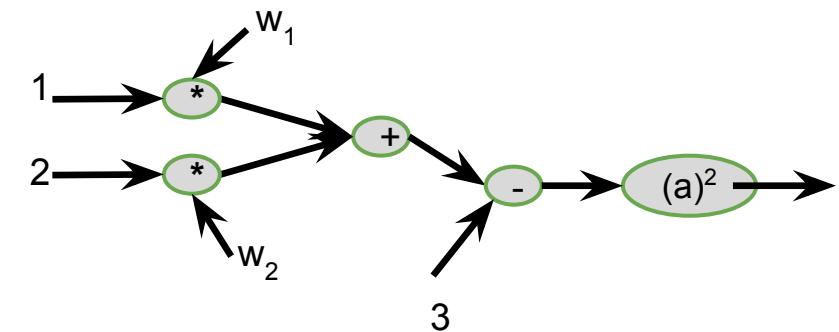
GD a jeho varianty jsou všechny lokální optimalizace - podle počáteční hodnoty parametrů (která je generovaná náhodně) naleznou jiné lokální minimum. To může být výrazně horší než globální minimum. Ukazuje se ale že u velkých sítí toto není problém a lokální minima bývají blízká tomu globálnímu. Také se na to můžete dívat tak, že používáme takové neuronové sítě, které nemají problém s nekvalitními lokálními minimy. Ono je to tak, že síť (včetně loss funkce), které mají tendenci uváZNOUT ve špatných lokálních minimech se možná nepoužívají, protože je nemožné je naučit. Velké neuronové sítě, které mají tendenci takto uváZNOUT by byl problém naučit i jinými optimalizačními metodami, které se snaží globální minimum najít. Ta optimalizační úloha je tak složitá, že prostě není možné vyzkoušet všechna možná řešení.

Gradients for training - chain rule

Need $\nabla_w J = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2} \right]$

$$\frac{\partial(w_1 + 2w_2 - 3)^2}{\partial w_1} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(c + 2w_2)}{\partial c} \frac{\partial(1 * w_1)}{\partial w_1}$$

$$\frac{\partial(w_1 + 2w_2 - 3)^2}{\partial w_2} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(w_1 + c)}{\partial c} \frac{\partial(2 * w_2)}{\partial w_2}$$



Pro trénování sítí potřebujeme gradient. Gradient jsou konkrétní hodnoty (reálná čísla) prvních derivací J podle každého z parametrů. Tedy pro konkrétní hodnotu parametrů a konkrétní trénovací sadu spočítáme gradient.

Výpočet gradientu spoléhá na řetězcové pravidlo pro výpočet derivací vnořených funkcí (https://en.wikipedia.org/wiki/Chain_rule) a na faktu, že společné části výpočtu derivace jde sdílet a tedy je možné spočítat celý gradient jedním průchodem neuronovou sítí.

Dobrá ilustrace na výpočetním grafu. Pro každý uzel/matematickou operaci musí být definovaná její derivace -- derivace je "gain" dané mat. operace, tedy říká jak moc se změní výstup dané operace, když se o kousíček změní její vstup. Tyto "gains" je zřetězených (vnořených) operací je možné mezi sebou násobit, čímž získáme odhad, jak se moc se změní výstup celého řetězce, když trochu změníme vstup.

Na grafu - při výpočtu derivace můžu jít proti směru grafu a u každé operace jen vynásobit derivaci dané operace s aktuální hodnotou (derivaci vůči výstupu dané operace). Tím získáte derivaci vůči stupu dané operace.

Toho vyplývá, že je možné definovat sadu základních operací a jejich derivací a pak je možné z nich skládat libovolné neuronové sítě (orientované acyklické grafy) a výpočet derivací delé sítě je možné počítat automaticky jen průchodem výpočetním grafem. Přesně to dělají frameworky jako pytorch a tensorflow - v nich definujete jen strukturu sítě a derivace se počítají automaticky.

$$\nabla_w J = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2} \right] = \left[\frac{\partial J}{\partial a} \frac{\partial a}{\partial w_1}, \frac{\partial J}{\partial a} \frac{\partial a}{\partial w_2} \right]$$

$$\frac{\partial J}{\partial a} = \frac{\partial(w_1 + 2w_2 - 3)^2}{\partial a} = 2(a)(1 + 2w_2)$$

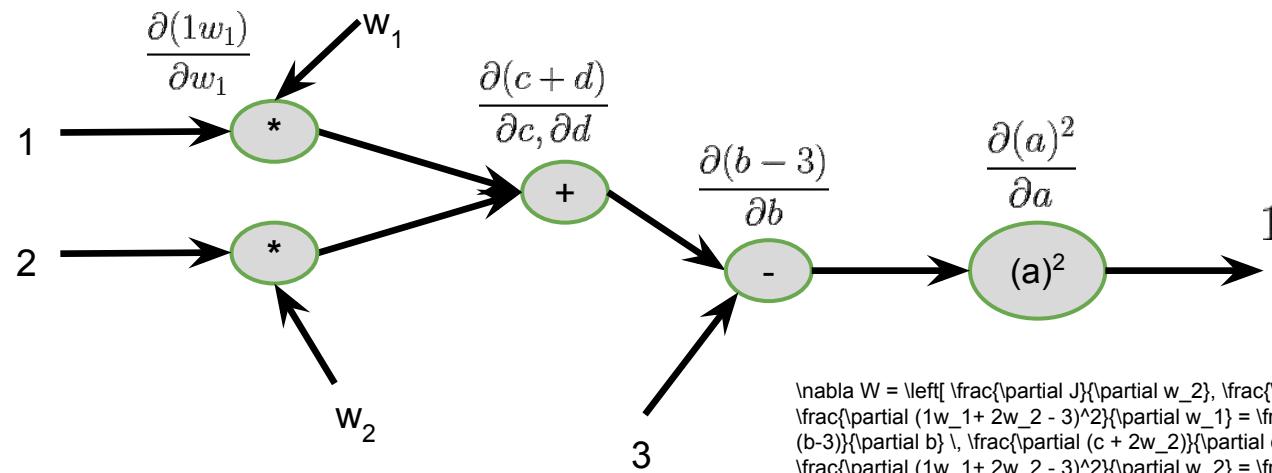
$$\frac{\partial a}{\partial w_1} = \frac{\partial(a)^2}{\partial a} = 2a$$

$$\frac{\partial a}{\partial w_2} = \frac{\partial(a)^2}{\partial w_2} = 4w_2$$

Gradient for training

Need $\nabla_w J = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2} \right]$

$$\frac{\partial(1w_1 + 2w_2 - 3)^2}{\partial w_1} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(c + 2w_2)}{\partial c} \frac{\partial(1 * w_1)}{\partial w_1}$$
$$\frac{\partial(1w_1 + 2w_2 - 3)^2}{\partial w_2} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(1w_1 + c)}{\partial c} \frac{\partial(2 * w_2)}{\partial w_2}$$



```
\nabla_w J = \left[ \frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2} \right]
```

$$\frac{\partial(1w_1 + 2w_2 - 3)^2}{\partial w_1} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(c + 2w_2)}{\partial c} \frac{\partial(1 * w_1)}{\partial w_1}$$
$$\frac{\partial(1w_1 + 2w_2 - 3)^2}{\partial w_2} = \frac{\partial(a)^2}{\partial a} \frac{\partial(b - 3)}{\partial b} \frac{\partial(1w_1 + c)}{\partial c} \frac{\partial(2 * w_2)}{\partial w_2}$$

Full training

- Forward pass

$$x_2 = \max(W_1 x_1, 0)$$

$$x_3 = \max(W_2 x_2, 0)$$

$$x_4 = \max(W_3 x_3, 0)$$

$$\text{loss} = (x_4 - 1)^2$$

- Compute loss

- Backward pass

- Compute derivatives of objective fce. with respect to net. parameters using chain rule / backpropagation

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(D, \theta)}{\partial \theta_j}$$

- Update parameters

Celé učení NN má tyto části:

Dopředný průchod - síť zpracuje vstup a vyrobí výstup

Spočítání chyby

Zpětný průchod - výpočet gradientu --- ten je výpočetně náročný podobně jako dopředný průchod.

Úprava parametrů podle pravidel použitého optimalizačního algoritmu - zde je to GD.,

Pytorch introduction / basics of tensors

[https://colab.research.google.com/drive/1ytNsahwdI4FJDkXmaw7CCQQJqQDiAxJt
?usp=sharing](https://colab.research.google.com/drive/1ytNsahwdI4FJDkXmaw7CCQQJqQDiAxJt?usp=sharing)

The same as Numpy with

Automatic differentiation

GPU support

Neural network layers

Loss functions

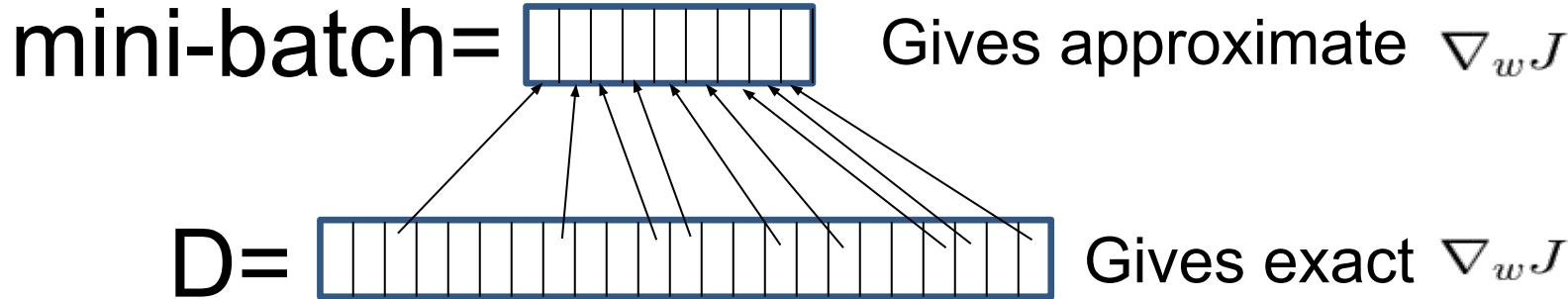
Optimization algorithms for neural networks

Datasets, data augmentation, predefined networks,
pretrained network, ...

Stochastic/mini-batch gradient descent

- Large datasets -> too much computation per network update -> inefficient training
- Estimate gradient on a small random part of the training set (mini-batch) and update network
- Stochastic Gradient Descent

$$J(D, \theta) = \sum_{D=\{(x_i, y_i), \dots\}} \text{loss}(f(x_i, \theta), y_i)$$



GD jako takový má zásadní problém na větších datových sadách. Předpokládá, že gradient se počítá na celé datové sadě a pak se udělá malá úprava sítě. Iterací GD musí být většinou 10k až 1M. Při velikosti datové sady třeba 100M obrázků by to trvalo celou věčnost.

Gradient ukazuje, jak v malém okolí nejlépe změnit parametry. Ale tato informace platí doopravdy jen v malém okolí čím dál, tím je méně přesná. Je možné použít nepřesné odhady gradientu a udělat více menších optimalizačních kroků - nic se tím nepokazí minimálně proto, že gradient také je jen aproximace.

Z toho vychází Stochastic/mini-batch gradient descent (SGD) - gradient se v každé iteraci odhadne jen na podmnožině trénovací sady (třeba 8 - 256 trénovacích vzorků). Původní SGD je definováno jen s aproximací na jednom vzorku, ale dnes se pojmenován SGD používá většinou pro mini-batch gradient descent.

SGD už je praktický algoritmus pro trénování NN a skoro všechny praktické algoritmy jsou postaveny na gradientech odhadnutých na mini-batch.

Je důležité vybírat mini-batch v iteracích tak, aby co nejlépe approximovaly celou trénovací sadu - většinou ale stačí náhodný výběr.

Learning rate and adaptive algorithms

- High learning rate -> divergence
- Low learning rate -> slow training
- For SGD, optimal learning rate depends on
 - Network
 - Objective, consider gradients for $J()$ and $10000*J()$
- You have to try different learning rates

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial J(D, \theta)}{\partial \theta_j}$$

- Solution:
 - Adaptive GD algorithms which normalize learning rate
 - Adam, AdaDelta, ...

Train good model and guess real-world performance

- Training set
 - Train models on this set
 - In domain data, similar problems, synthetic, generated, augmented
- Validation set
 - Evaluate models, tune hyperparameters, choose alg. and architecture
 - Same (iid) as training set?
- Test set
 - Estimate real-world performance
 - Data ideally from real world
- Loss is usually different than the target metric
 - Loss can improve while “accuracy” gets worse

Review

- What is a neural network?
- What are “trainable/learnable” parameters?
- What is loss function?
- What is gradient?
- GD and SGD?
- What is a training set in supervised learning?
- Shallow learning vs deep learning?
- Linear neuron and matrix multiplication.
- Non-linear activations.
- Layers.
- Computational graph.

Review

- Basic loss functions for classification and regression.
- What is training - objective function minimization.
- Backward pass and gradient descent rule.
- SGD and its benefits.
- What are train set, validation set and test set?

Neural networks for structured data

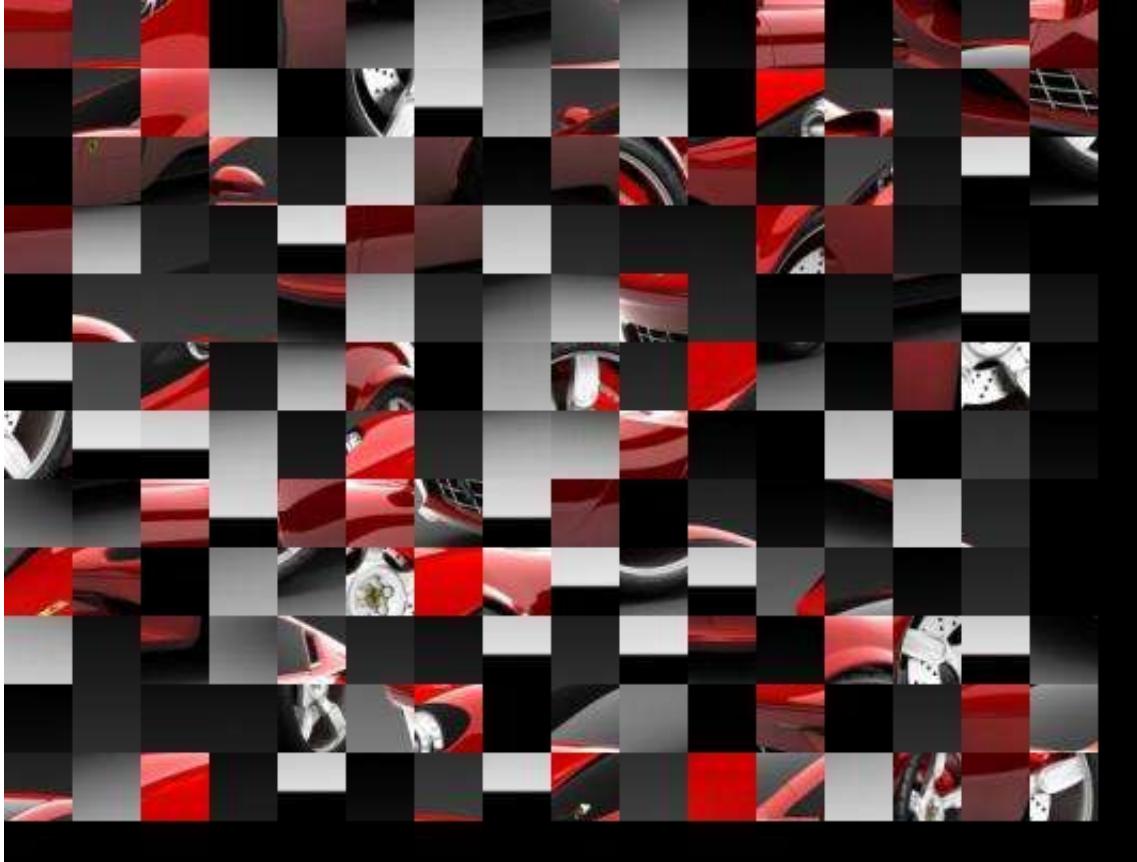
Zatím byla řeč o sítích, které nepracují s jakkoli strukturovanými daty - nemají definované žádné vztahy mezi vstupními příznaky (hodnotami ve vstupním vektoru), všechna propojení v nich byla uniformní. Některá data ale mají jasnou strukturu - text, zvuk, obrázky, grafy, ... V takových případech je vhodné, aby neuronové sítě tuto strukturu respektovaly. Nebo se na to dá dívat tak, že je vhodné, když výpočetní strukturu přizpůsobíme datům, které se zpracovávají - nižší výpočetní nároky a síť potřebuje méně trénovacích dat, protože volbou vhodné struktury už ji "předáme určitou znalost" o vhodném způsobu zpracování daných dat.

Obrázky:

- blízké pixely spolu více souvisí než vzdálené - proto je vhodné více modelovat jejich závislosti
- Objekty v obraze se můžou pohybovat - proto je vhodné obraz zpracovávat stejným způsobem na všech pozicích.



Local cues

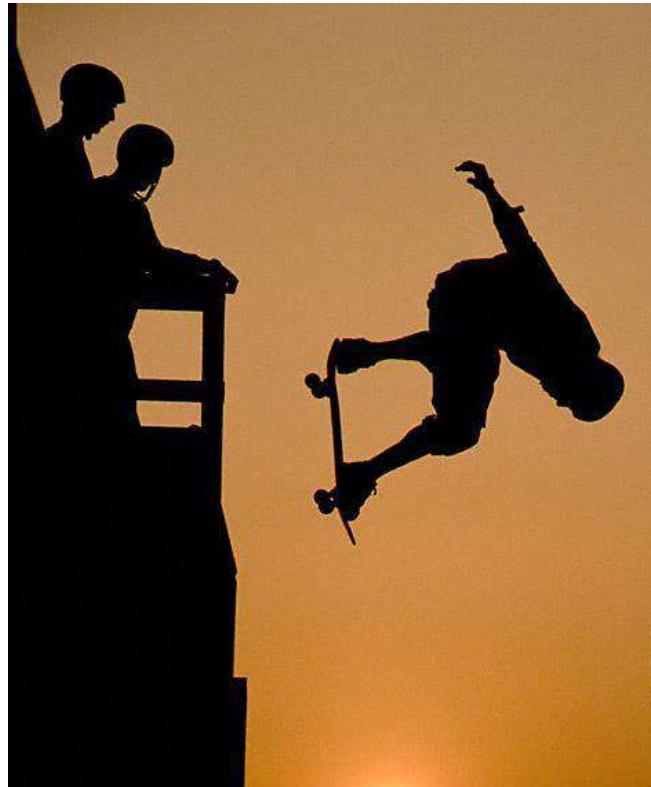


Co je důležité na obrázku pro jeho pochopení. Lokální tvary?

Local cues



Image representations



Co je důležité na obrázku pro jeho pochopení. Celkový kontext?

Layers for images?

- Which mathematical operation fits?
 - Is local
 - Processes each part of input the same way
 - Is linear
 - Is the building block of neural networks for images, sound, ...

History of Convolutional Networks

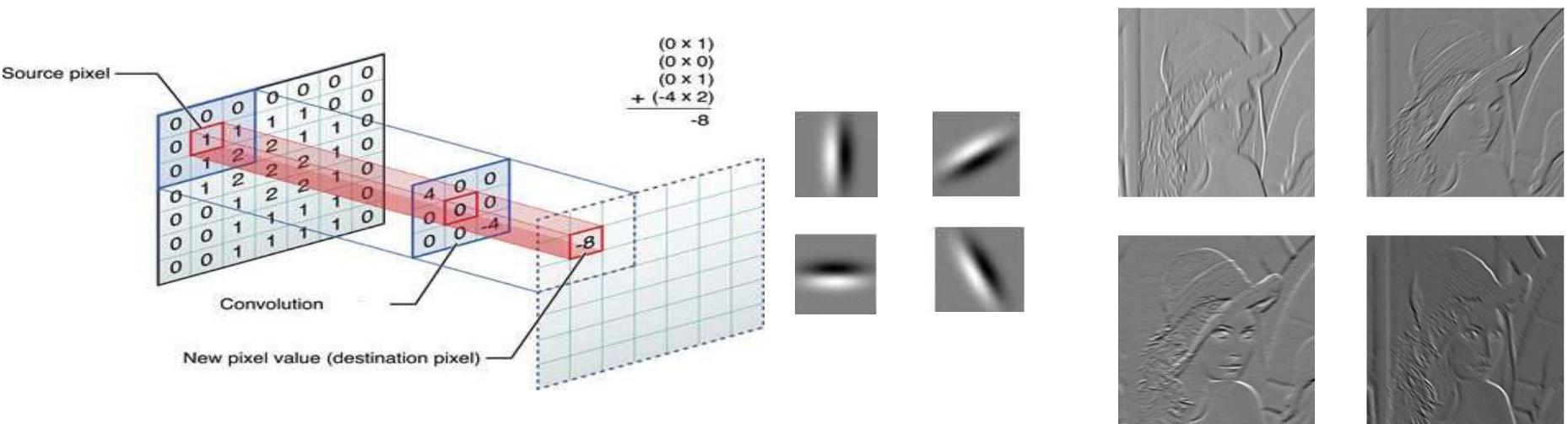
- **1980** – Kunihiko Fukushima – **Neocognitron**: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position
- **1986** – Hinton, Rumelhart, Williams, McClelland - **Backpropagation**
- **1998** – LeCun et al.: Gradient-Based Learning Applied to Document Recognition
- **2012** – Krizhevsky et al.: ImageNet Classification with Deep Convolutional Neural Networks
- **2013-today** – explosion of applications

Convolution

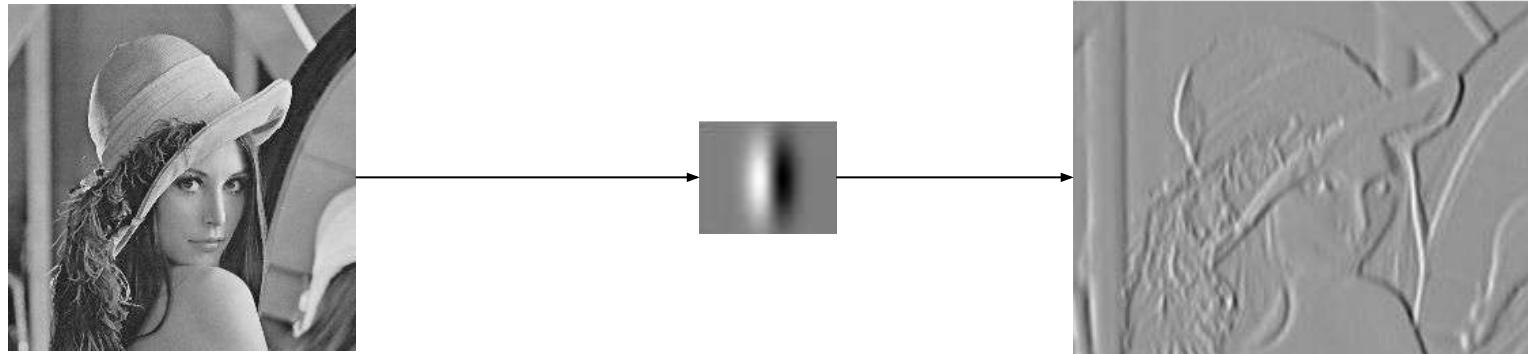
- Linear operation
 - (matrix multiplication on small neighborhood)
- Output: Map of local similarity of input and conv. kernel

Konvoluce v NN:

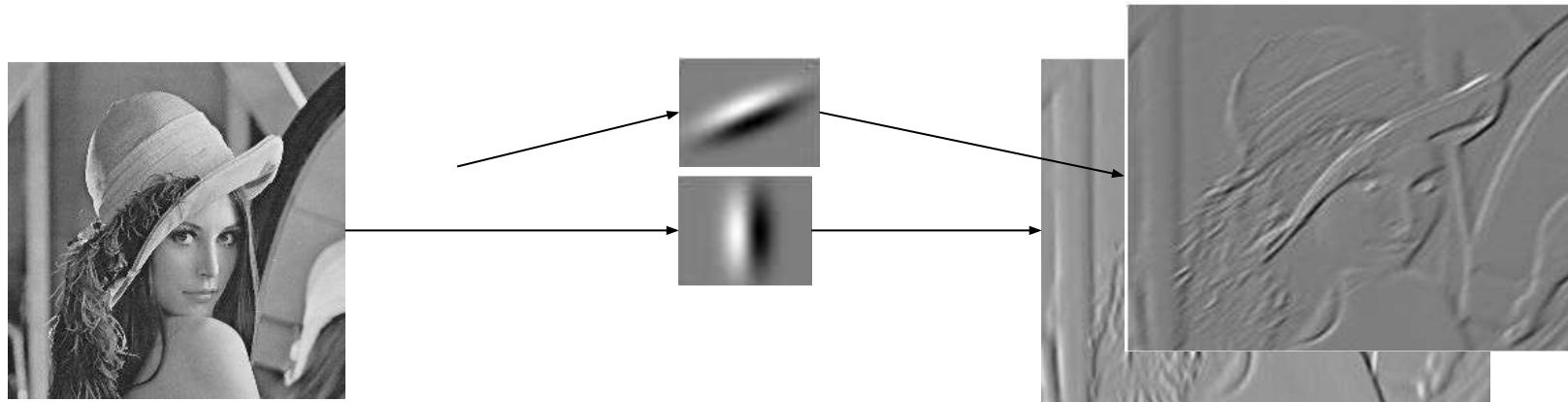
- Konvoluce je lineární operace (je možné ji vyjádřit jako maticové násobení).
- V konvolučních neuronových sítích jsou to právě konvoluční jádra, které se učí.
- V CNN konvoluce výpočetně nejnáročnější částí.
- Konvoluční jádra bývají velmi malá. Většinou 3x3 pixelu.
- Odezva konvolučního jádra je vysoká, pokud vstup má podobný vzor jako konvoluční jádro.



Convolutional layer

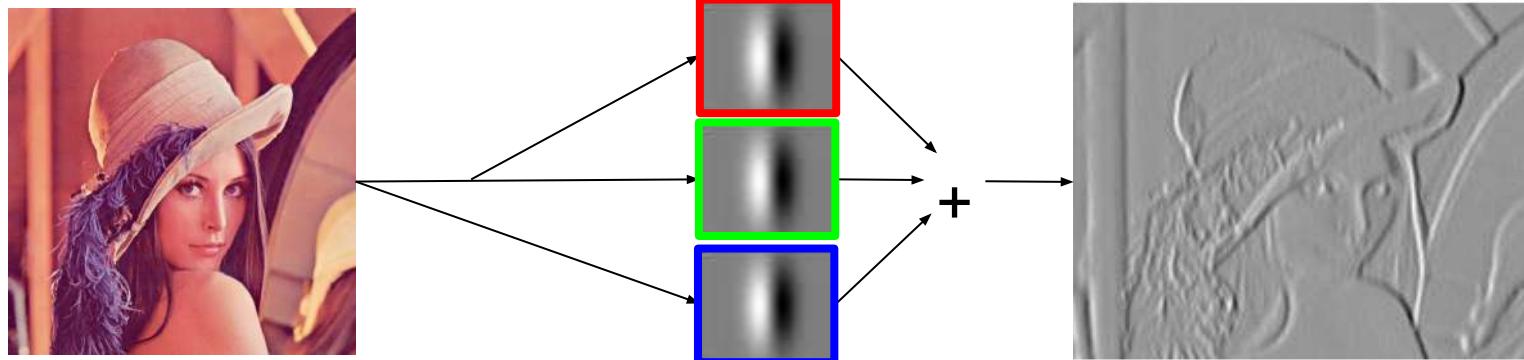


Convolutional layer



V konvoluční vrstvě není jen jedno jádro, ale je jich více (8,16,32,...,1024). Každé konvoluční jádro vytváří jednu verzi vstupního obrazu, kterému se říká kanál nebo aktivační mapa. Odezvy všech konvolučních jader tvoří 3D tenzor HxWxC (výška, šířka, počet kanálů). Docela často se používá opačné pořadí CxHxW.

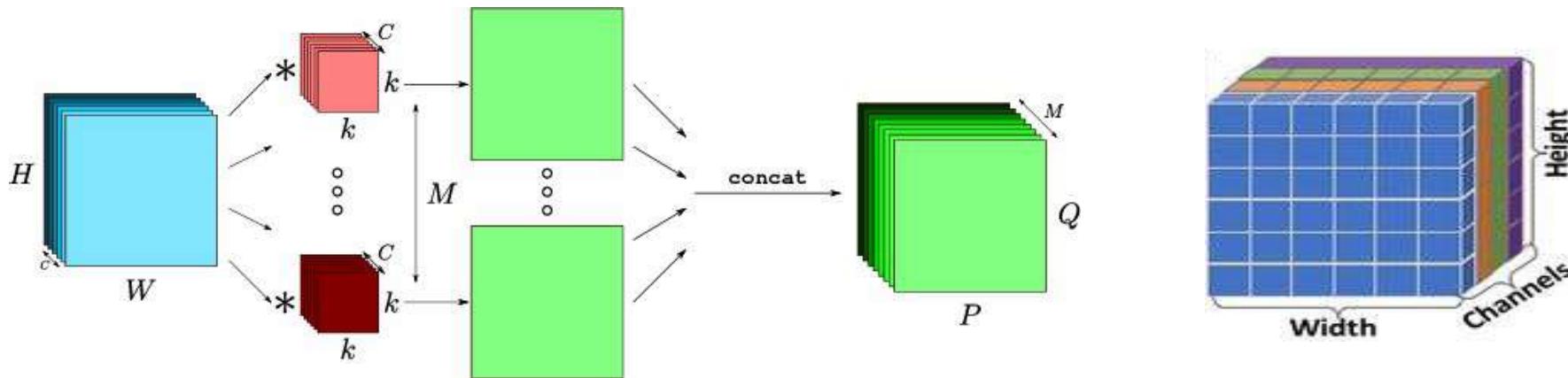
Convolutional layer



Pokud má vstup konvoluční vrstvy více kanálů, musejí mít jádra více kanálů. Pokud je vstup $64 \times 64 \times 16$, jádra můžou být např. $3 \times 3 \times 16$ (každé 144 hodnot).

Convolution

- Channels -> activation map



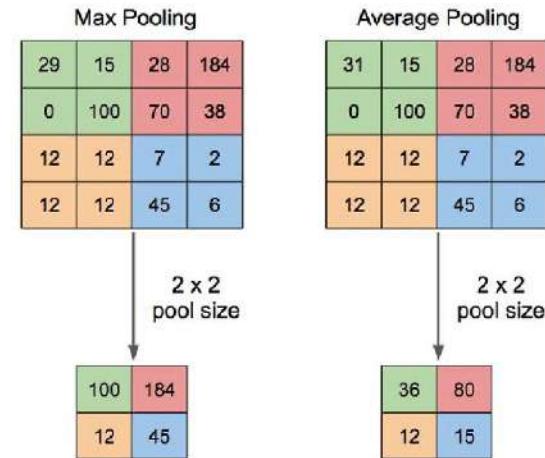
Shrnutí konvoluční vrstvy - pěkné vizualizace a vysvětlení jsou na <https://www.youtube.com/@animatedai/videos>

Konvoluční vrstva v "základním" nastavení nemění velikost obrázku a má jádra se stejným počtem kanálů jako má vstup.

Na zkoušce bývají otázky na výpočetní složitost konvoluční vrstvy a počet parametrů vrstvy.

Pooling layer

- **Downsampling**
 - Local pixels are aggregated into a single pixel
 - Spatial resolution is reduced, channels stay the same
 - Reduction by AVERAGE or MAX functions.
 - MAX-pooling directly encourages translation invariance



Pooling vrstvy zmenšují velikost "obrazu", podvzorkovávají ho, zmenšují prostorové souřadnice. Mají parametr krok (kolikrát bude výstup menší) a velikost okolí (obdoba velikosti jádra u konv. vrstvy) určuje jak velké okolí se bude agregovat. Často krok = velikost okolí.

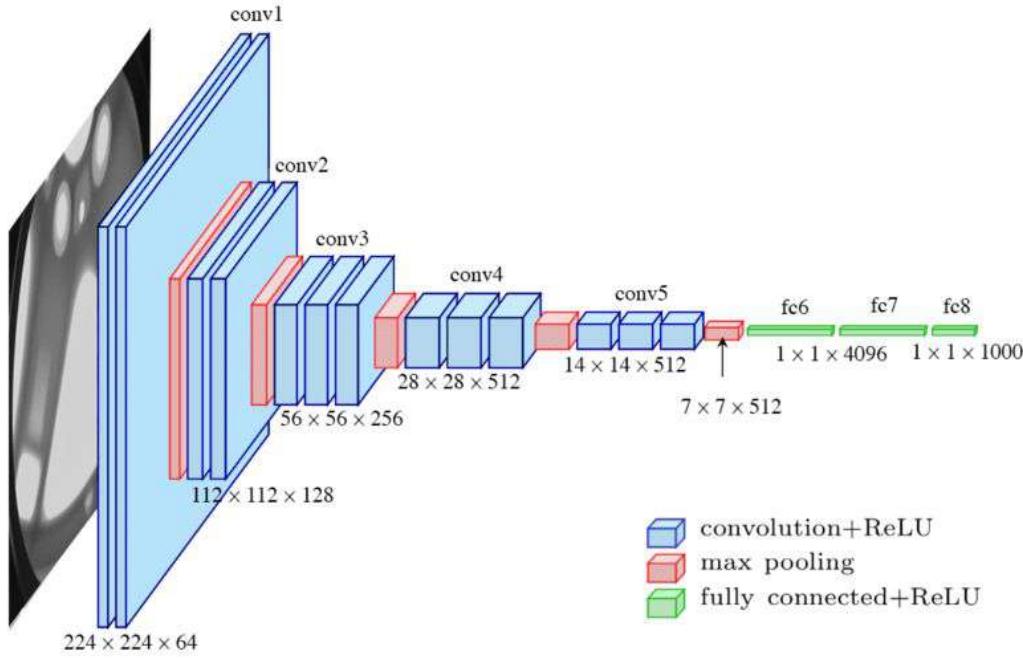
Pooling vrstvy nemění počet kanálů a každý kanál vstupu zpracují samostatně.

Max-pooling nahradí hodnoty z okolí jejich maximální hodnotou, average-pooling používá průměr.

Max-pooling do jisté míry do sítě explicitně vnáší robustnost vůči malým deformacím (posunutím), protože je má jedno, ze které pozice v agregovaném okolí pochází maximální hodnota.

V sítích se většinou používá max-pooling kromě posledních částí sítě, kde average-pooling často slouží k agregaci informace z celého obrázku (průměrné příznaky agregované přes celý vstup).

Network architecture (VGG net)



Architektury sítí pro klasifikaci mají skoro vždy tvar jako nahoře. Vstupem je "velký" obrázek a na výstupu potřebuje jedno rozhodnutí pro celý tento obrázek. Je potřeba postupně agregovat informaci z celého vstupu.

Na začátku mívají síť velké rozlišení a konvoluční vrstvy s menším počtem kanálů. Postupně se snižuje prostorové rozlišení a naopak se zvyšuje počet kanálů.

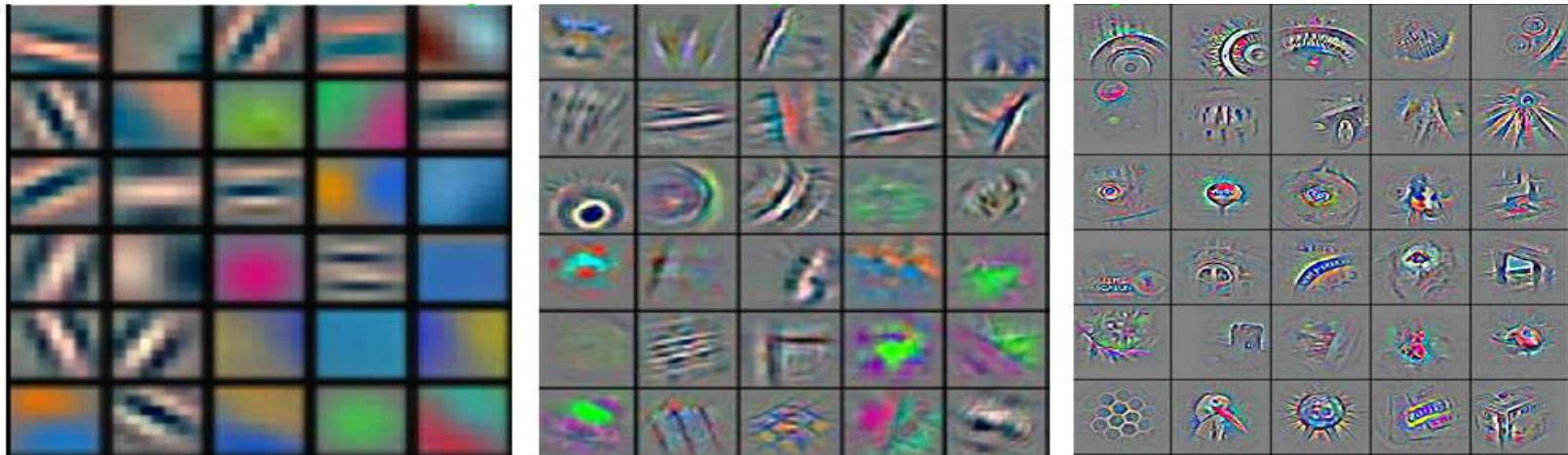
Konvoluce 3×3 na nízkém rozlišení efektivně zpracovávají velkou část vstupu.

V nízkém rozlišení jeden "pixel" (reprezentovaný vektorem reálných čísel s dimenzí rovnou počtu kanálů) reprezentuje velkou část vstupu, která může obsahovat velké množství složitých vzorů - na jejich zachycení/popis je potřeba více čísel než na začátku sítě, proto se počet kanálů zvyšuje.

Síť končí po redukci na jeden "pixel", který závisí na celém vstupu.

U konvolučních sítí se používá pojem "receptive field", což je oblast, ze které může určitá aktivace/pixel čerpat informace, neboli oblast vstupu, která může ovlivnit danou aktivaci. Zkuste si spočítat, pro různé vrstvy této sítě velikost receptive field. Všechny konvoluce jsou 3×3 a max pooling má krok i velikost okolí 2×2 .

Hierarchy of features (and human visual system analogy)



Vizualizace vzorů na jaké nejvíce reagují "neurony" / filtry v různých úrovních sítě - AlexNet (2012).

Na začátku - zachycují malý patch obrázku - jednoduché filtry optimální pro zpracování obrázku - šedotónové vysokofrekvenční (orientované hrany) + nízkofrekvenční barevné přechody.

Hlouběji - větší patch, složitější tvary, které už můžeme i identifikovat jako části objektů.

Ještě hlouběji - sémanticky identifikovatelné části objektů

Very Deep Convolutional Networks

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

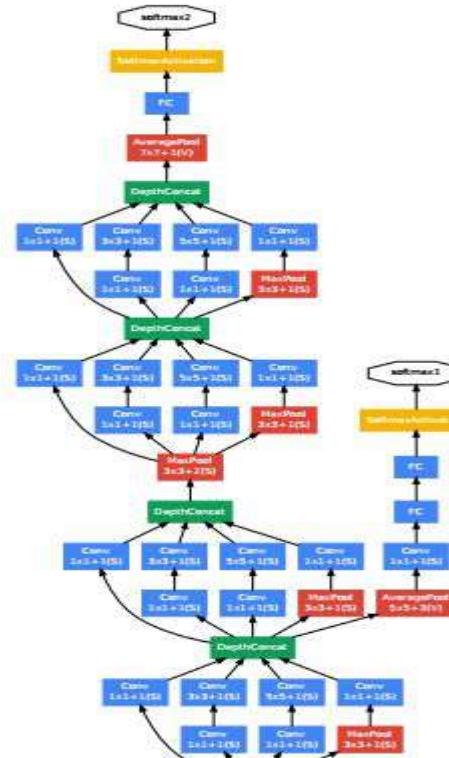
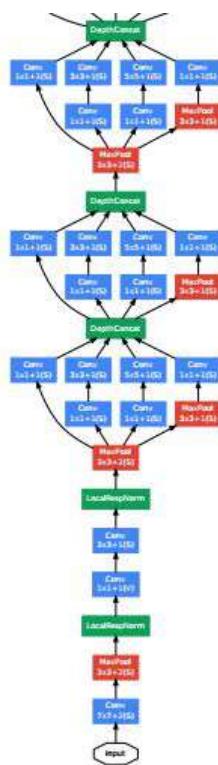
Rozepsané vrstvy různých verzí VGG sítí - formát conv[kernel_size]-[kernel_count] jako ilustrace možných sítí. VGG dnes není nejlepší architektura, ale stále se používá.

FC-[channel_count] - plně propojená - fully connected - už pracuje nad vektorem / obrázkem s rozlišením 1x1.

Promyslete si, kolik je kde parametrů a jak jsou výpočetně náročné různé části sítě. PS: První FC vrstva má na vstupu asi vektor o velikosti $7 \times 7 \times 512$, protože 7×7 je velikost po konvoluční části.

Karen Simonyan, Andrew Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition,

GoogLeNet



Szegedy et al.: Going deeper with convolutions

Další příklad architektury klasifikační sítě - GoogLeNet.

Podobné jako VGG, ale konvoluční vrstvy jsou nahrazeny několika vrstvami s různě velkými jádry pro optimalizaci výpočetní náročnosti a velikosti zpracovaného okolí.

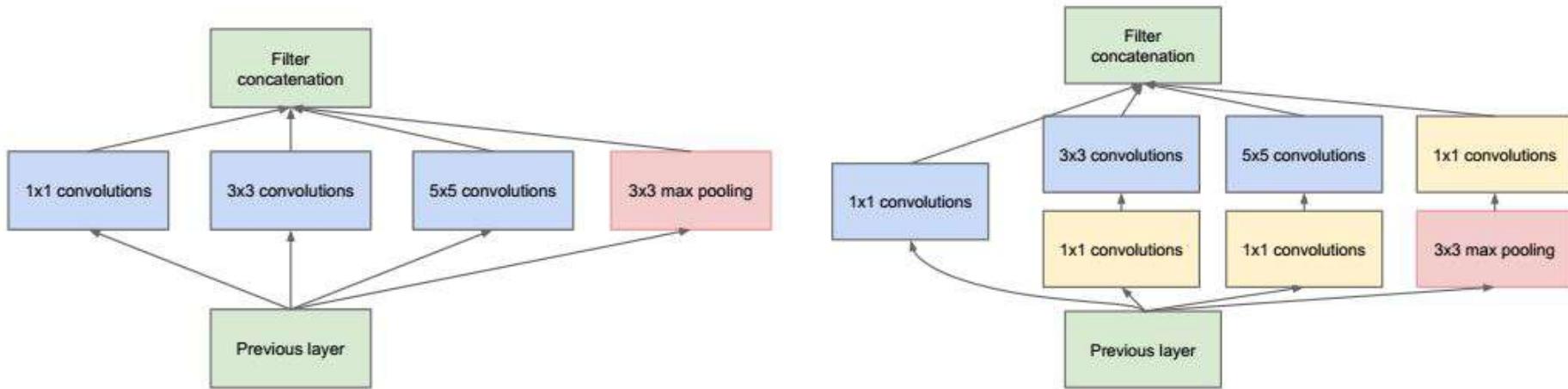
Více výpočtů použito pro zpracování lokální informace, ale stále jsou tam i jádra 5x5.

GoogLeNet - Inception Layer

- Winner of ILSVRC 2014

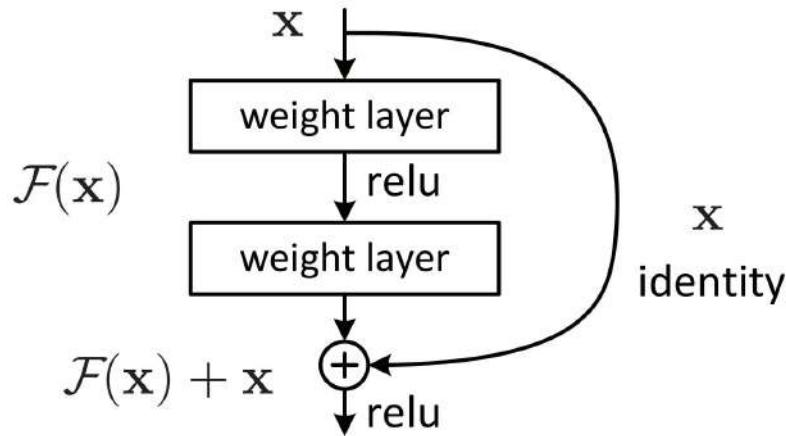
Ukázka možné struktury bloků v GoogLeNet. PS: jádra 1x1 jsou 25x menší a rychlejší než jádra 5x5 při stejném počtu kanálů.

Žluté 1x1 konvoluce vpravo jsou použity pro snížení počtu kanálů a tedy snížení výpočetní náročnosti následující vrstvy s větším filtrem.



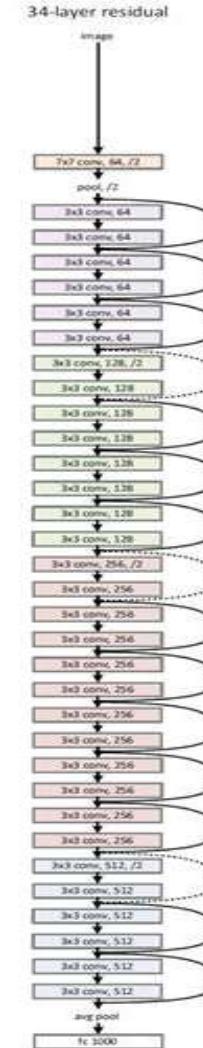
Szegedy et al.: Going deeper with convolutions

Residual networks

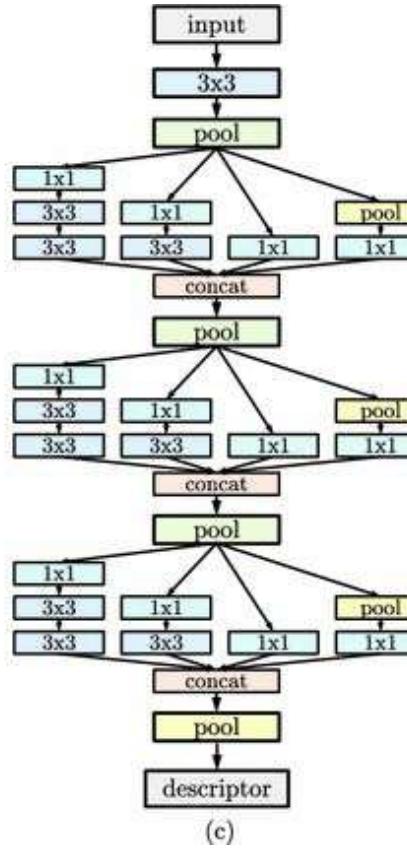
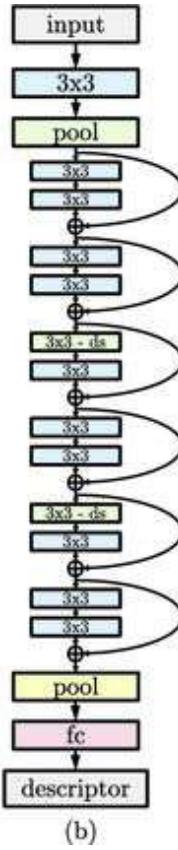
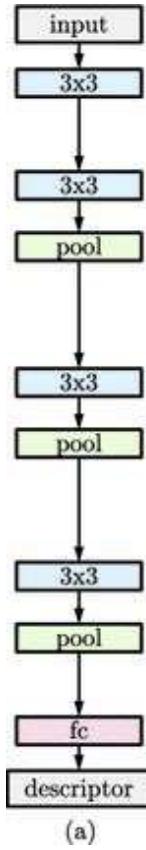


Předchozí sítě mají limit počtu vrstev (hloubky) při kterém už začíná být problém je natrénovat. Každá vrstva po inicializaci úplně náhodně promíchá data při průchodu touto sítí. To platí o dopředném průchodu, kdy se "ztrácí" informace o vstupech, stejně tak se při zpětném průchodu "ztrácí" informace z gradientů (spočítaný gradient je pak jenom šum a síť se zlepšuje jen pomalu při optimailzaci).

Pro trénování hlubokých sítí je potřeba, aby se informace mohla lépe šířit sítí. ResNets k tomu využívají "reziduální" zapojení vrstev, kde je přímá linka, kterou se informace může šířit a konvoluční vrstvy jsou zapojeny "bokem" a jejich výstup se přičítá k původní informaci.



Combinations



Je možné kombinovat různé myšlenky architektur.

Základních (obecných) architektur sítí existuje hodně.

Můžete se podívat třeba na připravené a předtrénované sítě v torchvision.models

<https://pytorch.org/vision/stable/models.html>

Generalization

- **Good model structure**
- Multi-task learning
- Pre-training
- Domain adaptation
- More data - augmentation, synthetic data
- Model size

Regularization and overfitting

- Deep convolutional networks overfit surprisingly late
- Early stopping (when training)
- Weight regularization (weight decay)
 - does not help much (with ReLU)
- Activation regularization

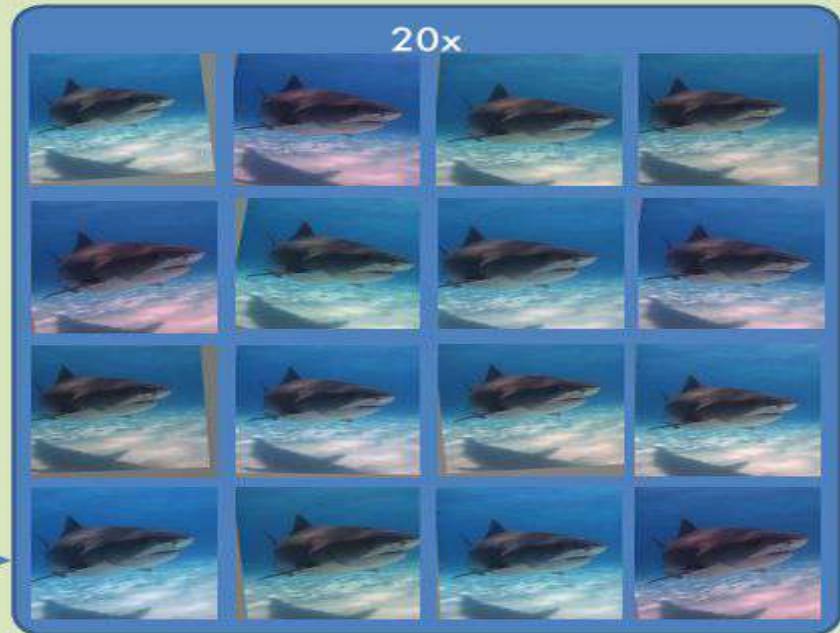
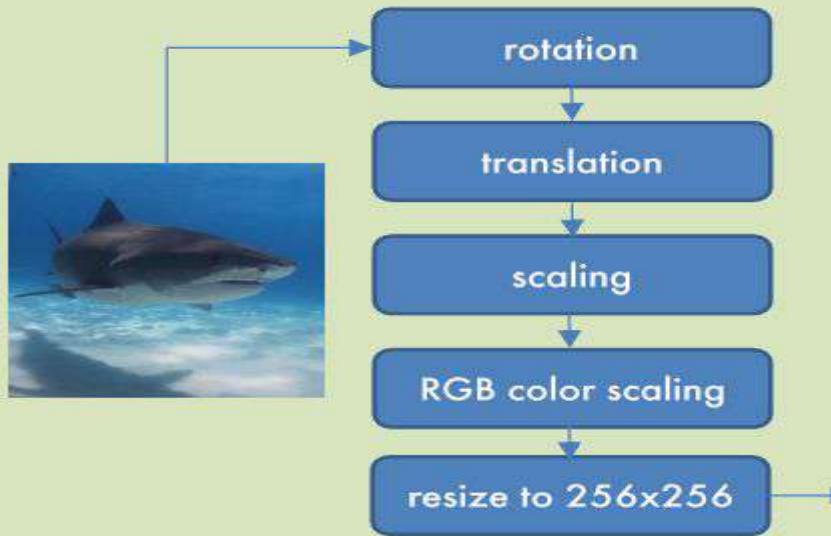
$$J(D, \theta) = \left(\sum_D loss(f_\theta(x_i), y_i) \right) + \lambda \|\theta\|^2 + \beta \|activation\|^2$$

- **Dropout** helps a lot
 - Sets random activations to 0

$$J(D, \theta) = \left(\sum_D loss(f_\theta(x_i), y_i) \right) + \lambda \|\theta\|^2$$

Data augmentation

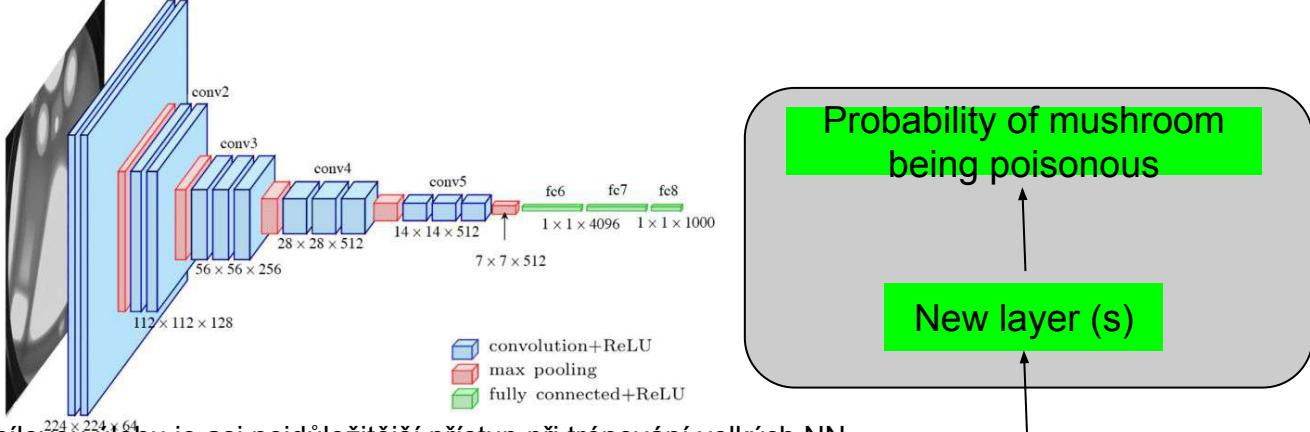
Image transformations



Augmentace dat - použití umělých náhodných úprav dat pro vytvoření nových verzí. Je to zadarmo a má smysl augmentaci použít vždy. Otázka je, co jsou vhodné augmentace a jaká může být jejich "síla" - to ale záleží na konkrétní úloze (podívat se do literatury a vyzkoušet různé varianty). Augmentace, nesmí zničit důležitou informaci z dat - třeba náhodně rotovat obrázek při klasifikaci, jestli je fotografie vyfocená "optimálně".

Finetuning

Network pre-trained on a large general dataset (ImageNet)



Předtrénování a následné doladění sítí na cílovou úlohu je asi nejdůležitější přístup při trénování velkých NN.

Předtrénování:

- Na velké datové sadě se složitou úlohou se předtrénuje velká NN. To může trvat až týdny a vyžadovat i stovky GPU.
- Používá se například datová klasifikační sada ImageNet (10M obrázků 10k tříd)
- Autoři pak předtrénované sítě uvolňují veřejnosti -- například torchvision.models <https://pytorch.org/vision/stable/models.html>
- Takto se síť naučí efektivně extrahat obecně užitečnou informaci z obrazu. Podobně se člověk naučí vidět a pak s touto schopností se teprve učí konkrétní úlohy mnohem rychleji.
- Pro tyto základní konvoluční sítě se používá pojem **backbone**.

Dolandění:

- Malá datová sada pro cílovou úlohu (můžou to být v extrémním případě jen jednotky příkladů, ale většinou stovky nebo tisíce). Existují pojmy one-shot-learning a few-shot-learning
- Pro klasifikaci se vymění poslední nebo několik koncových vrstev sítě. Poslední vrstva pak má počet výstupů shodný s počtem tříd v cílové úloze.
- Je možné doučit jen nové vrstvy, ale dotrénování (dolandění, finetuning) celé sítě většinou poskytuje lepší výsledky.
- Takové předtrénované "backbone" jsou základem i jiných sítí: detekce objektů, segmentace, instance segmetace, optical flow, identifikace lidí podle obličeje, ...

Active learning

Make annotations effective.

Use model to select hard examples,
annotate, retrain.

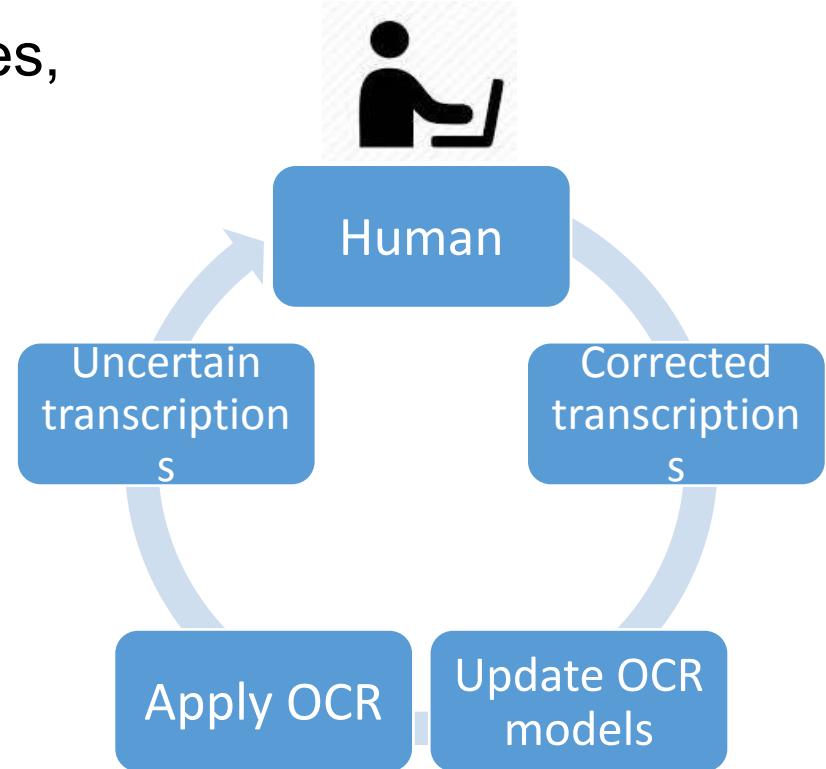
Use prediction confidence.



Aktivní učení je způsob jak anotovat co nejužitečnější data (utratit peníze za anotaci užitečných dat).

Data se vybírají opakovaně například podle nejistoty již natrénované sítě - například podle odhadu pravděpodobnosti při klasifikaci.

Jsou možné další způsoby výběru třeba podle podobnosti obsahu "obrázků".



Weird learning

Semi supervised learning

Active learning

Self-supervised learning

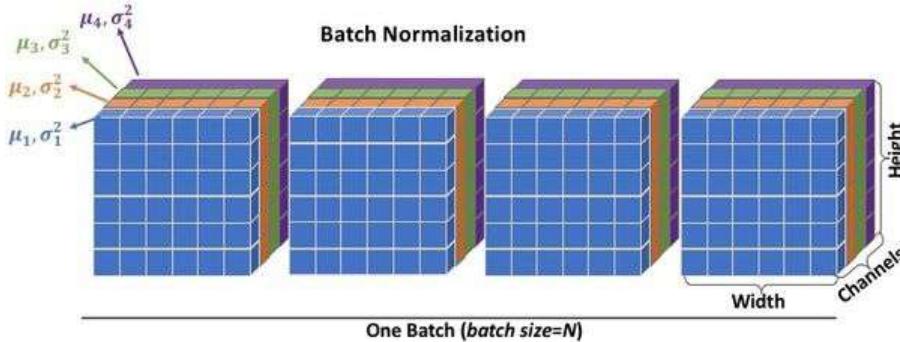
Transfer learning

Knowledge distillation

Domain adaptation

Self training

Batch normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Facial alignment

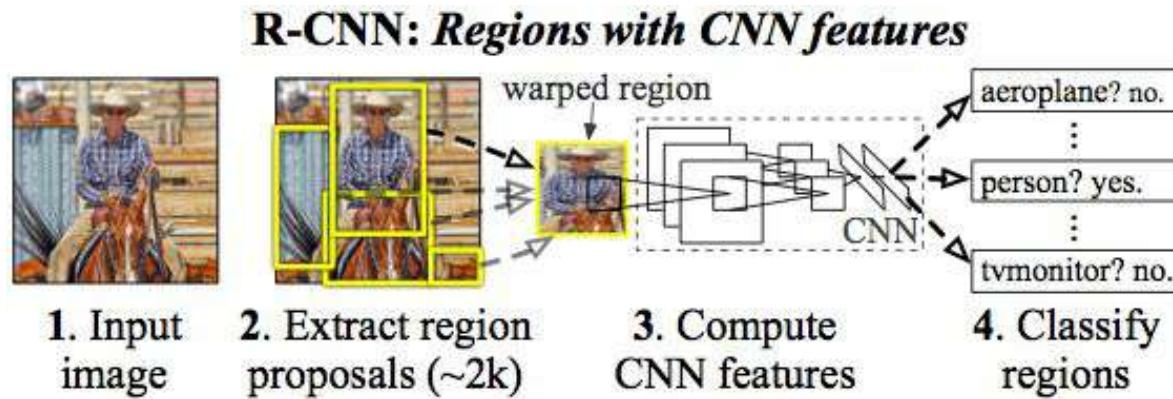
- Input - Facial crop
- Output: x,y coordinates -> regression loss (e.g. MSE)



Úloha s obrázky je mnohem více. Třeba výměnou výstupní klasifikační vrstvy za linární je možné odhadovat x,y pozice bodů a řešit tak lokalizaci obličeje - regresní úloha.

Object detection

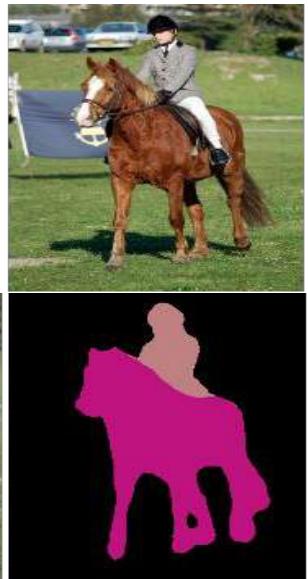
- R-CNN



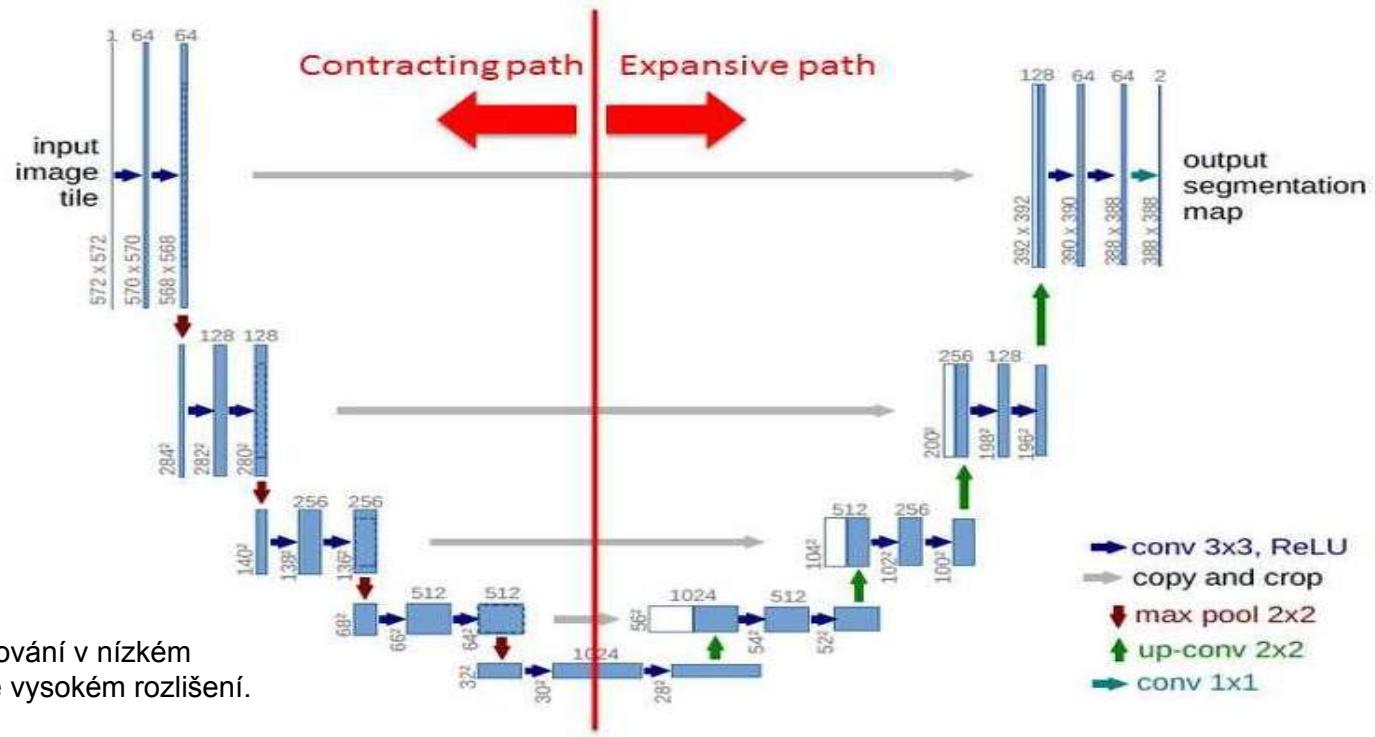
Detekce objektů jako klasifikace výřezů. R-CNN je staré, blbé, pomalé. Rozumné detektory jsou v KNN, POVa, nebo o dva slajdy dál.

Girshick, R., Donahue, J., Darrell, T., & Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation"

Fully Convolutional Networks: FCN



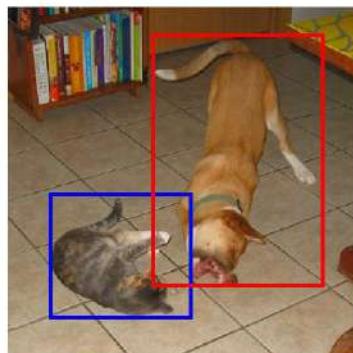
Network Architecture



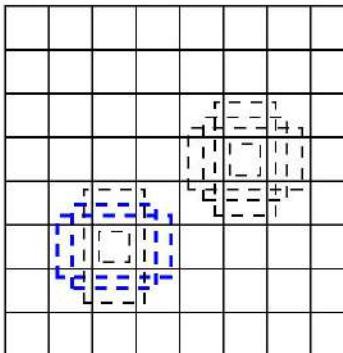
Segmentace - potřeba zpracování v nízkém rozlišení a zároveň výstup ve vysokém rozlišení.

Convolutional object detection

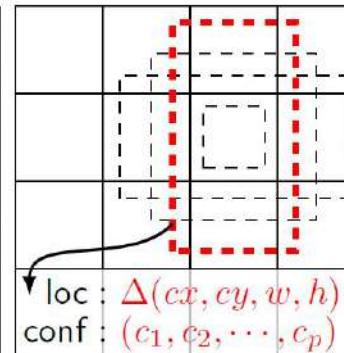
- Output per pixel:
 - For each anchor (base bounding box)
 - Object class (e.g. softmax)
 - Precise bounding box position (regression x, y, w, h)



(a) Image with GT boxes



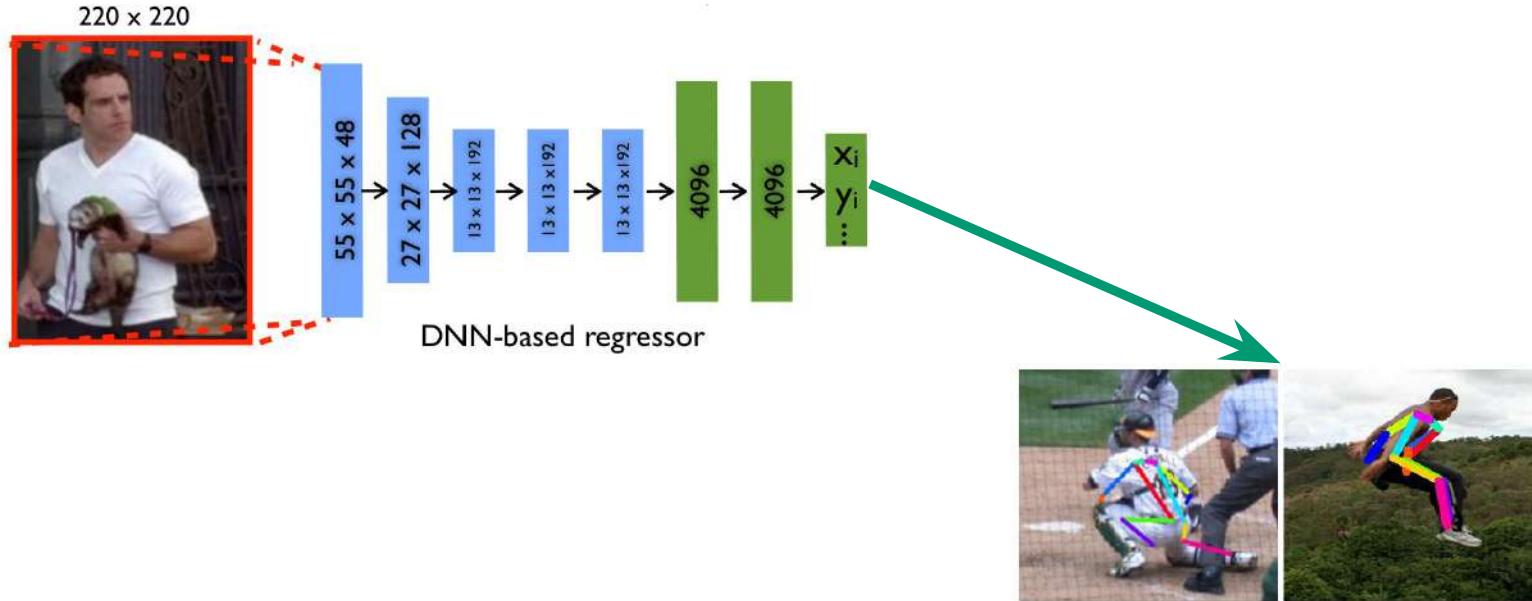
(b) 8×8 feature map



(c) 4×4 feature map

Detektor objektů může mít podobnou strukturu jako předchozí síť pro segmentaci, ale výstup v nižším rozlišení a v každém pixelu pro různé výřezy centrovány v daném pixelu odhadovat, jestli se v nich nachází objekt. Kombinuje klasifikační výstup pro rozhodnutí, jestli je ve výřezu objekt s regresí, protože síť zároveň odhaduje změnu pozice daného výřezu, aby co nejlépe ohraňoval objekt.

Pose estimation



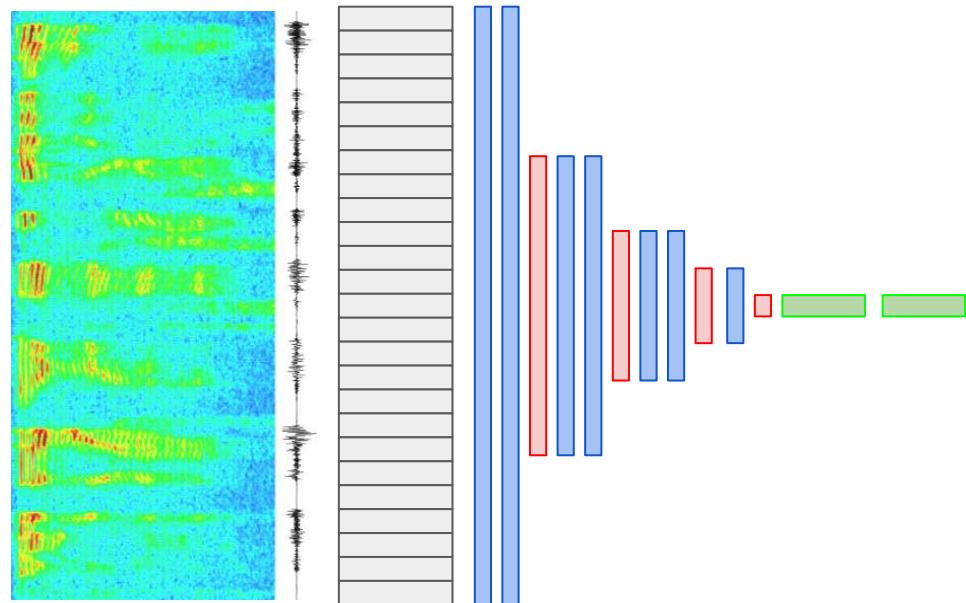
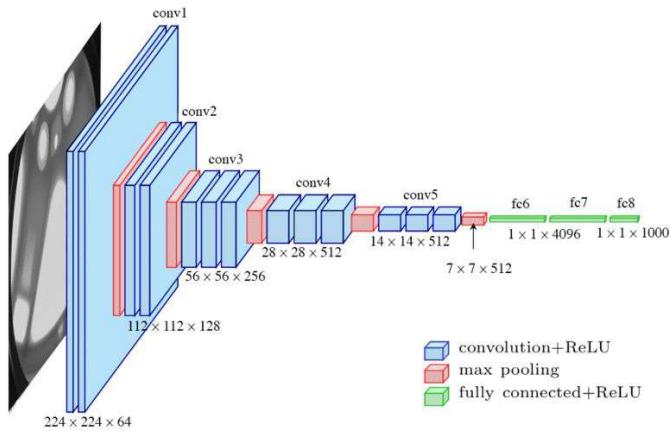
Toshev et al.: DeepPose: Human Pose Estimation via Deep Neural Networks. CVPR 2014

Umělá inteligence a strojové učení

Michal Hradiš: sequences and language

Teaser: ChatGPT live

Sequences - similar to images - just 1D



Sekvence jsou 1D struktury a nic nebrání je zpracovávat stejně jako obrázky - 1D konvoluce, 1D pooling vrstvy, stejné aktivace,

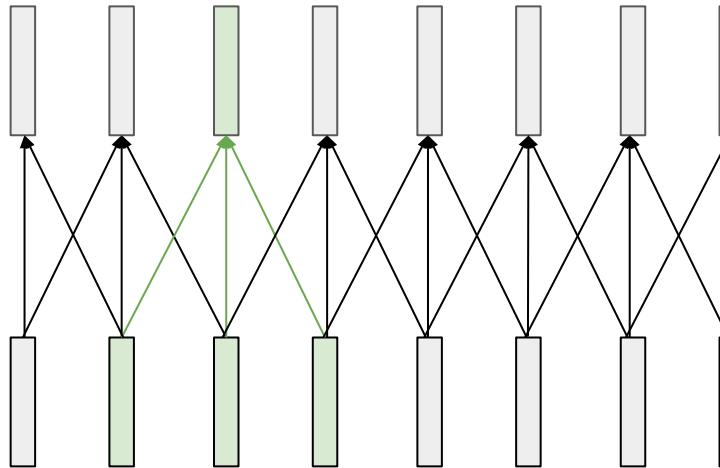
...
Klasifikační síť pro zvuky by klidně mohla být podobná jako VGG síť pro obrázky akorát v 1D. Třeba pro klasifikaci jazyka.

1D conv.

1D pooling

Fully connected

Conv. layers



Existují ale vrstvy, které jsou specificky navržené pro zpracování sekvencí. Není možné tvrdit, že jsou vždy lepší než konvoluce. Mají jiné vlastnosti a jejich vhodnost záleží na konkrétní úloze a datech.

Nyní rekurentní vrstvy, pozděj i attention vrstvy.

Rekurentní vrstva je sekvence "buňek", které mají realizují stejný výpočet, sdílejí stejné parametry (váhy) a při zpracování sekvence je výstup předchozím vstupem následující buňky.

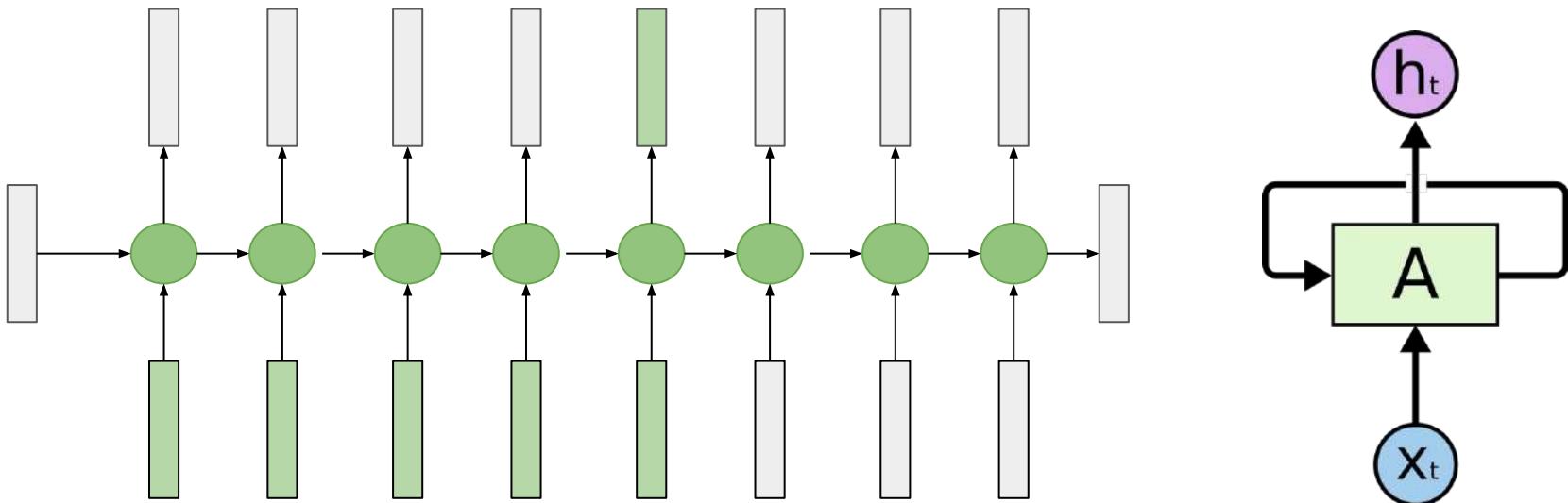
Tedy v každém "časovém" kroku buňka rekurentní sítě zpracuje 2 vektory - datový vstup a výstup předchozí buňky (stav). Protože každá buňka vyprodukuje vektor výstupu, je výstupem takové vrstvy sekvence vektorů stejně dlouhá jako je vstupní sekvence vektorů (to je přesně stejně jako u konvoluční vrstvy).

Navíc mají rekurentní vrstvy počáteční stav, který se někdy učí, někdy se nechá konstantní nebo může být výsledkem nějakého předchozího výpočtu. Produkují také koncový stav, který u některých typů rekurentních vrstev je stejný jako výstup poslední buňky u některých jsou rozdílné.

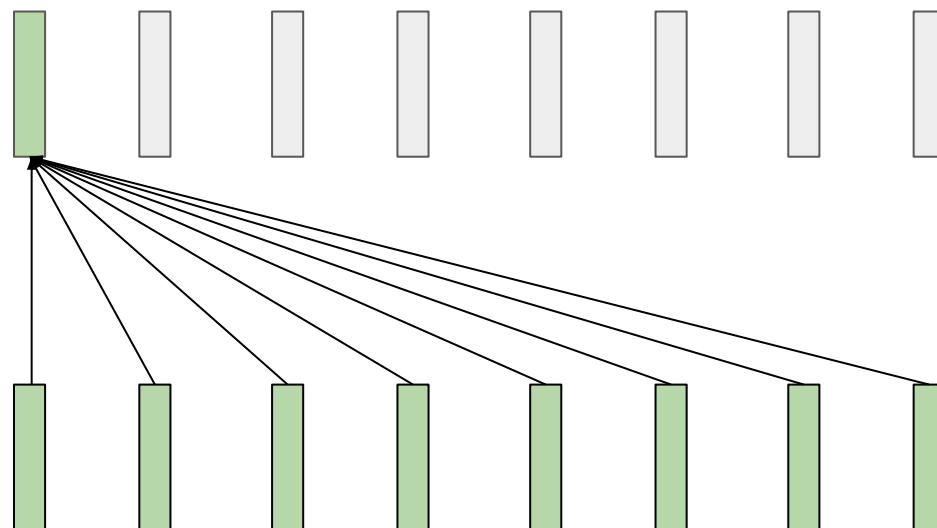
Důležité je propojení v takovéto vrstvě (síti). Výstup v určitém místě sekvence závisí na všech předchozích vstupech a počátečním stavu. To je rozdíl oproti konvolucím, kde závisí na přesně definovaném okolí daném velikostí konvolučních jader.

Nevýhodou může být, že se rekurentní vrstva musí počítat sekvenčně - jedna buňka po druhé. Oproti tomu konvoluce se počítá nezávisle a může paralelně. To může být "problém" na silně paralelním HW, pokud je výpočet jedné buňky poměrně levný.

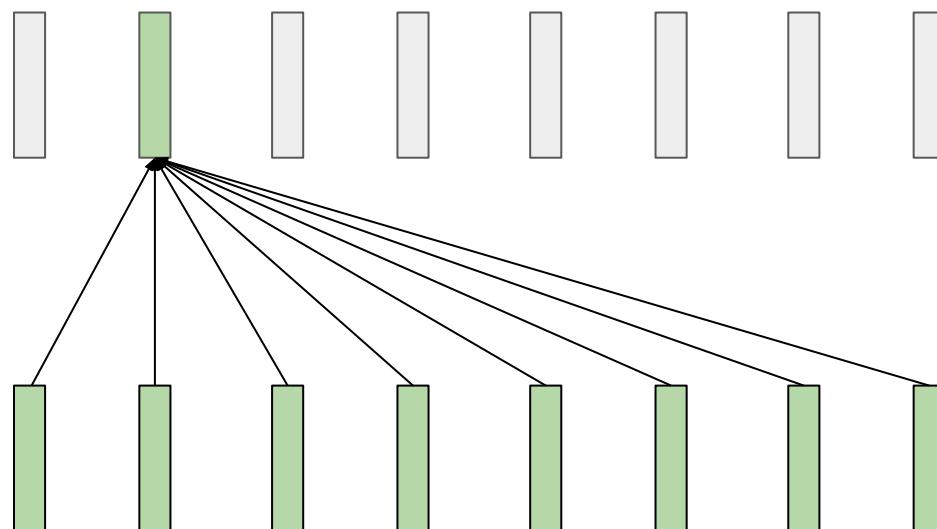
Recurrent layers



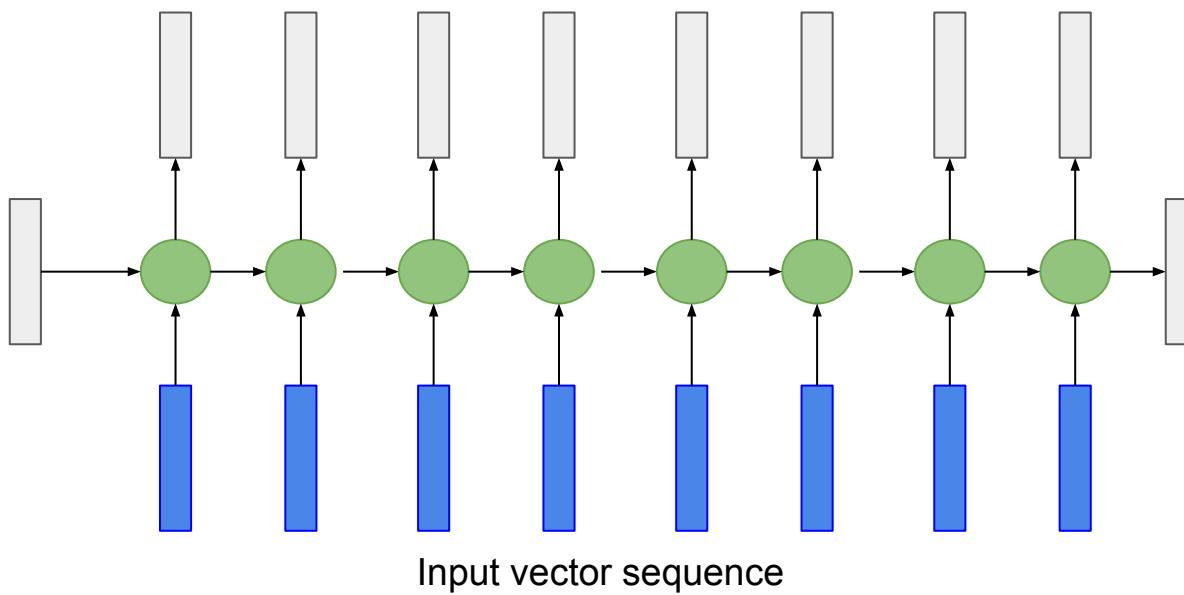
Attention layers



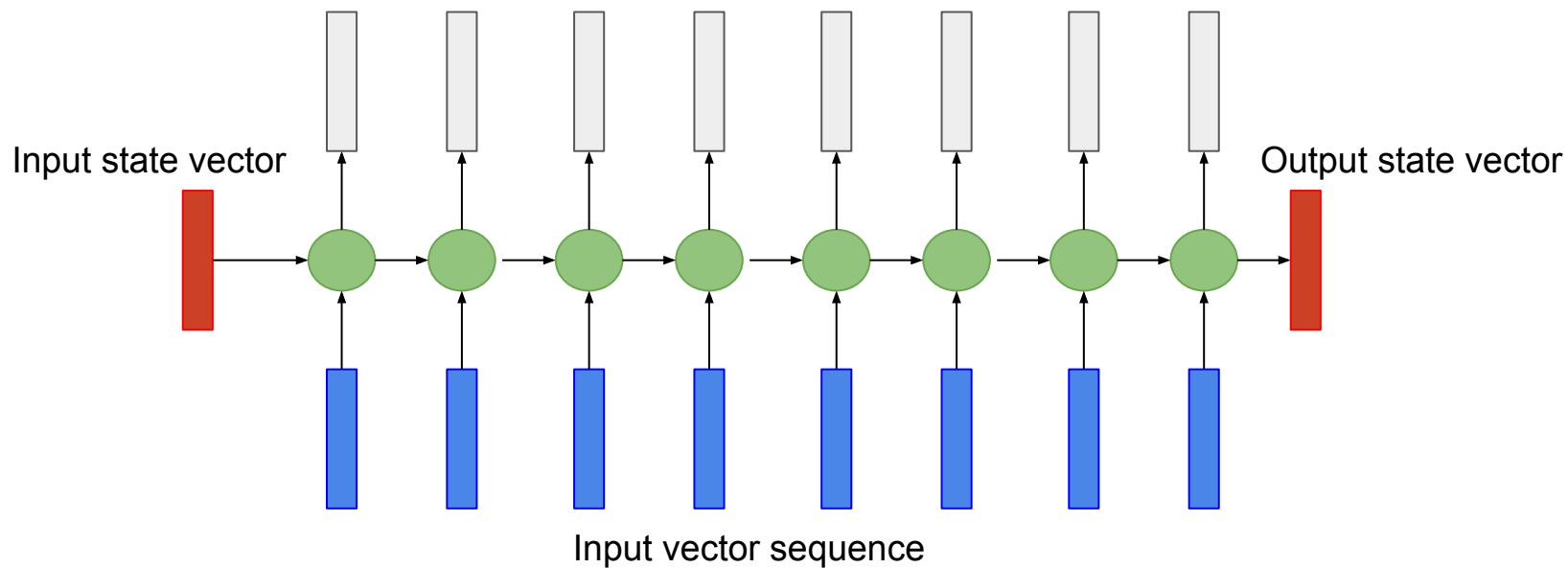
Attention layers



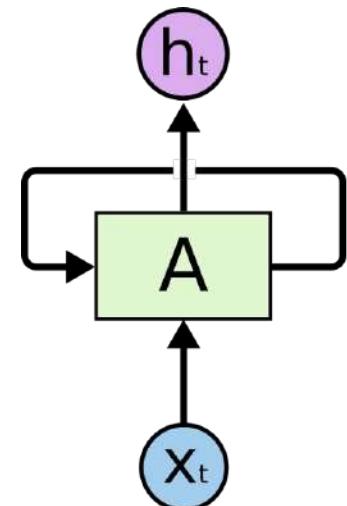
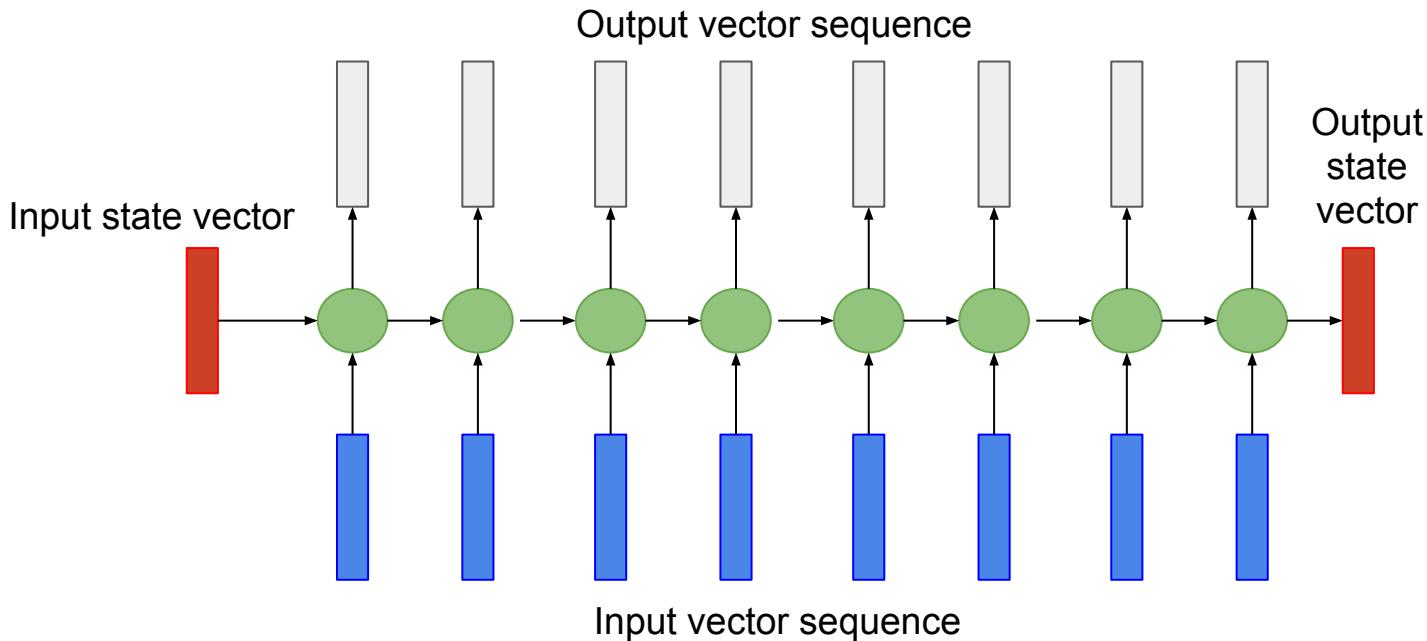
Recurrent layers



Recurrent layers



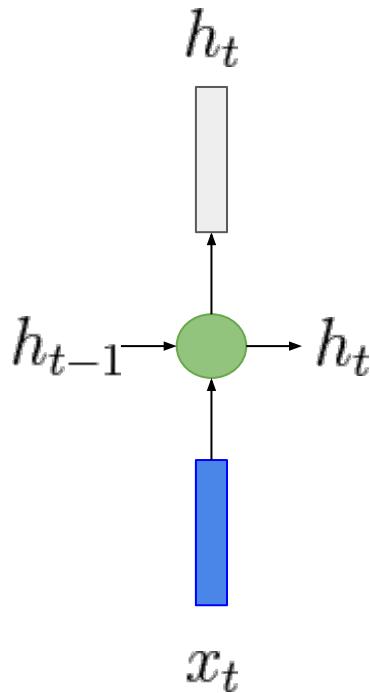
Recurrent layers



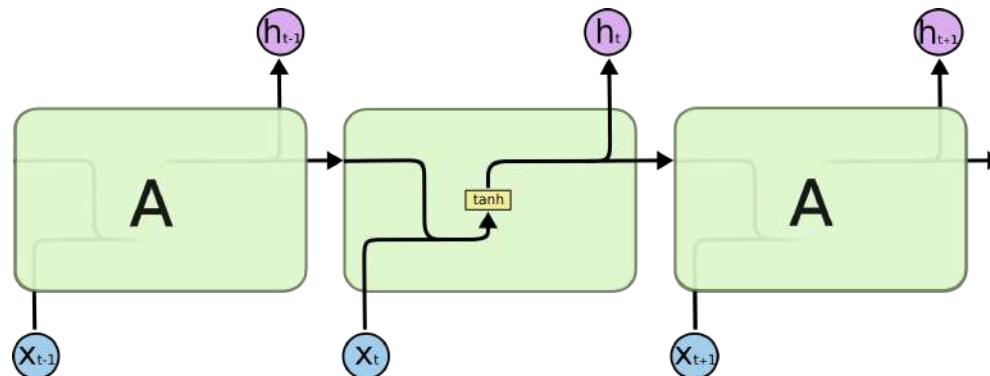
Vanila RNN

Nejjednodušší rekurentní síť (Vanila RNN) jen konkatenuje vektor předchozího stavu h a vstup x, vynásobí je maticí vah a použije tanh nelinearity. To je to samé jako u plně propojených sítí.

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$



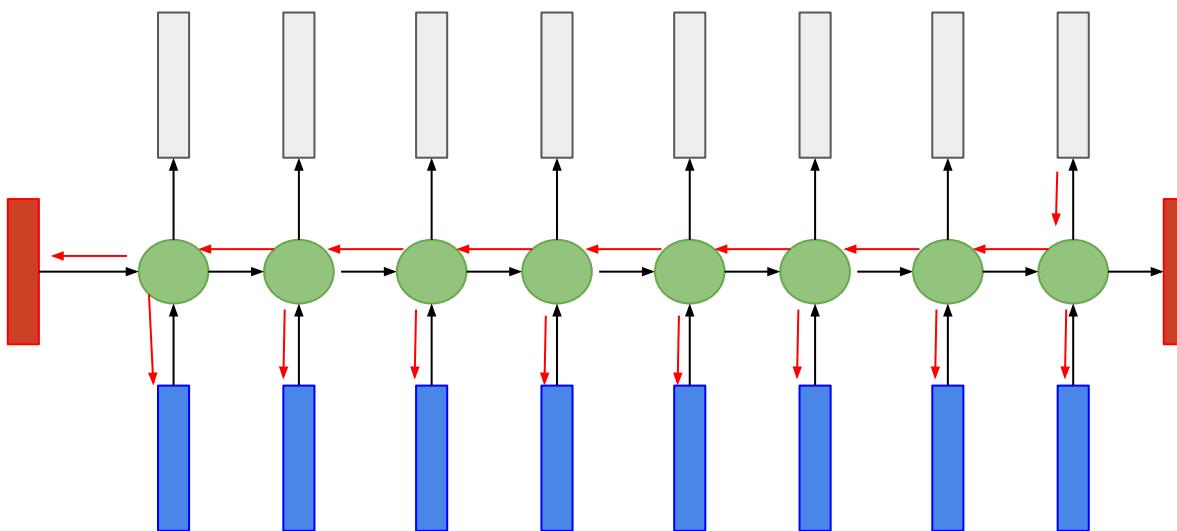
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$



Christopher Olah: Understanding LSTM Networks.

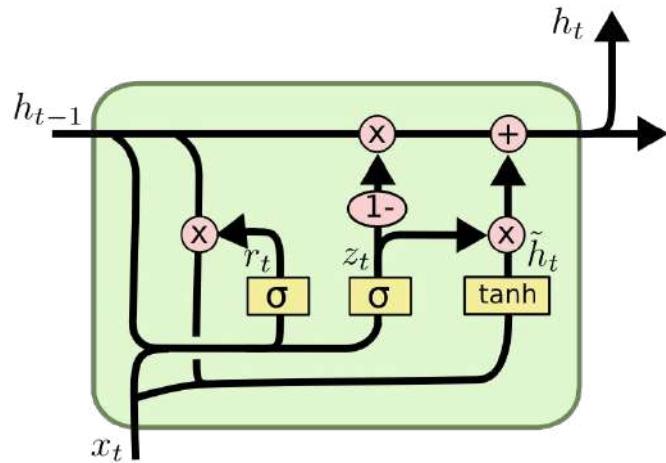
<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Recurrent layers - training



Trénování rekurentních sítí se v ničem neliší od jiných sítí. Pořád se jedná o dopředné sítě, je to funkce, kterou je možné prezentovat acyklickým orientovaným grafem. Jde možné standardním způsobem automaticky (error backpropagation) spočítat gradient a použít standardní optimalizační algoritmy jako u jiných sítí.

GRU - Gated Recurrent Unit // nebo LSTM



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

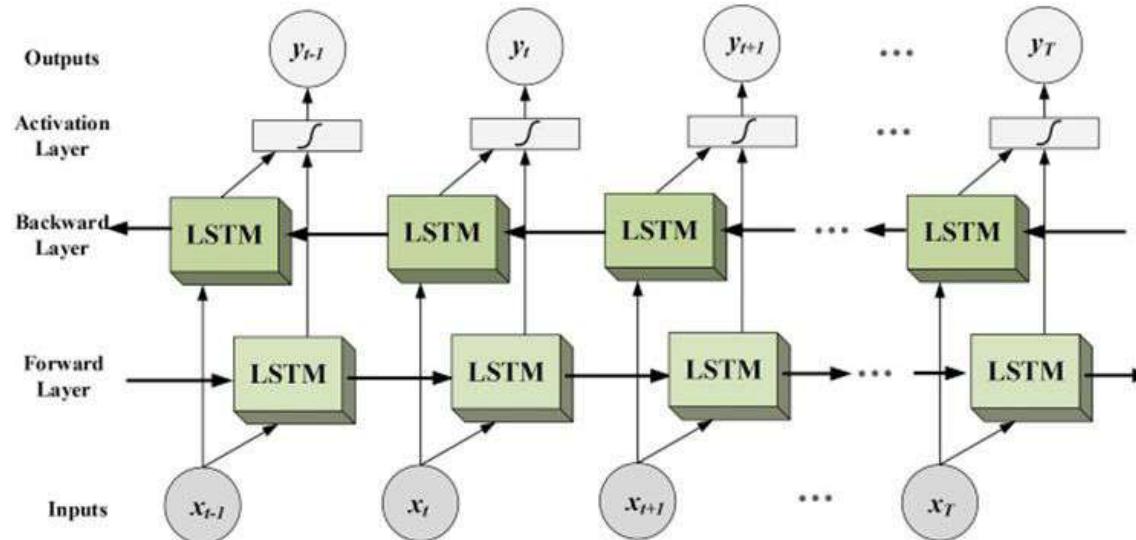
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU je jednodušší verze LSTM.

Christopher Olah: Understanding LSTM Networks.
<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Bidirectional recurrent layer

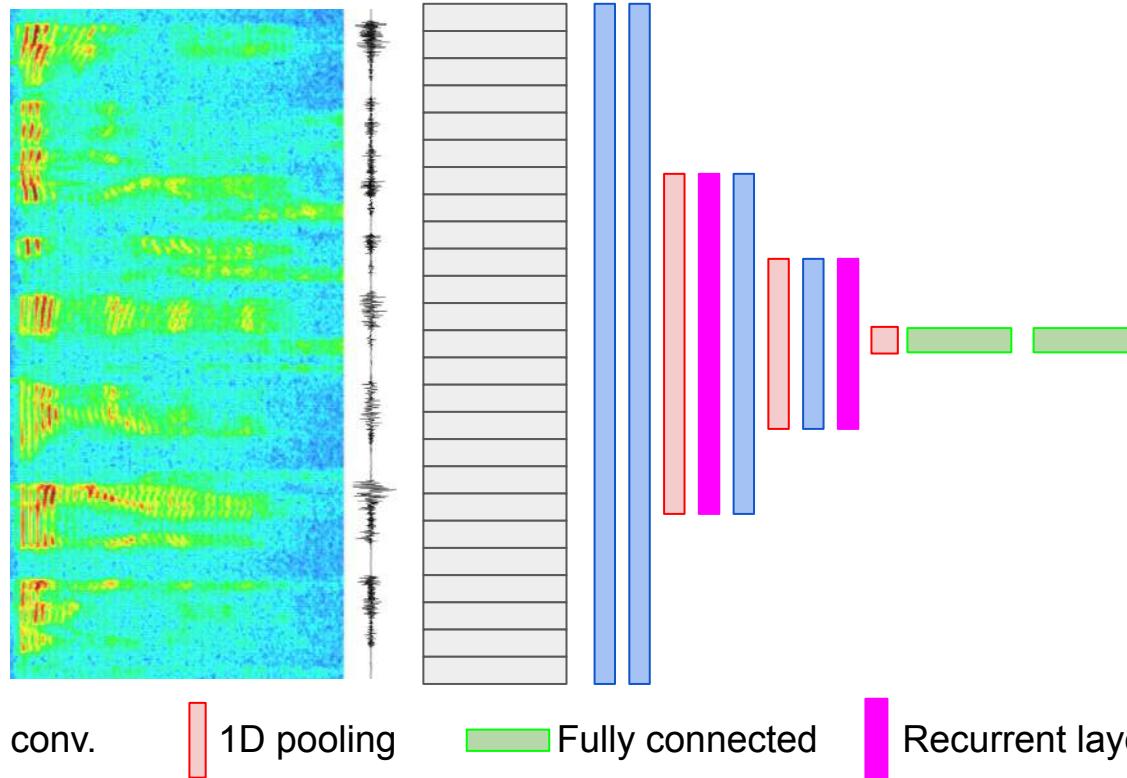


Obousměrné rekurentní vrstvy kombinují dvě "normální" rekurentní vrstvy. Jedna zpracuje sekvenci jedním směrem a druhá obráceně. Výstupy v každé pozici se konkatenují.

Taková struktura má smysl vždy, pokud síť nemusí být "kauzální" - třeba při real time detekci slov v řeči se nemůže dívat do budoucna, protože budoucnost ještě neexistuje - oproti tomu při zpracování hotových nahráve můžu.

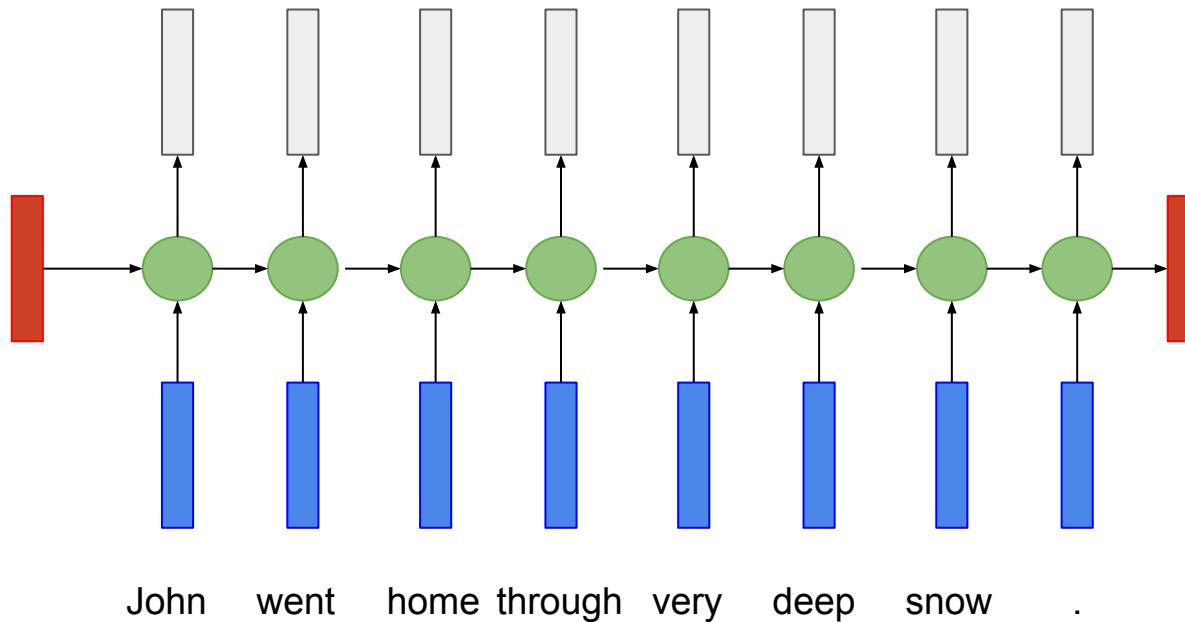
Mix convolution and recurrent layers

Ano, konvoluční a rekurentní vrstvy je možné takto kombinovat bez jakýchkoliv problémů.



Text as input?

Vstupem sítí musí být vektory reálných čísel. To text prostě není a musí se nejdřív vhodně transformovat.



One-hot-encoding

$$\begin{array}{l} x_t \\ \hline \text{John} \\ \begin{matrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \end{array} \quad Wx_t = \begin{bmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.2 & 1.2 & 0.2 & 1.0 & 0.0 & 0.2 & 0.2 \\ 0.8 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.2 & 1.2 & 0.2 & 1.0 & 0.0 & 0.2 & 0.2 \\ 0.8 & 0.2 & 0.1 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 1.2 \\ 0.2 \\ 0.2 \\ 1.2 \\ 0.1 \end{bmatrix}$$

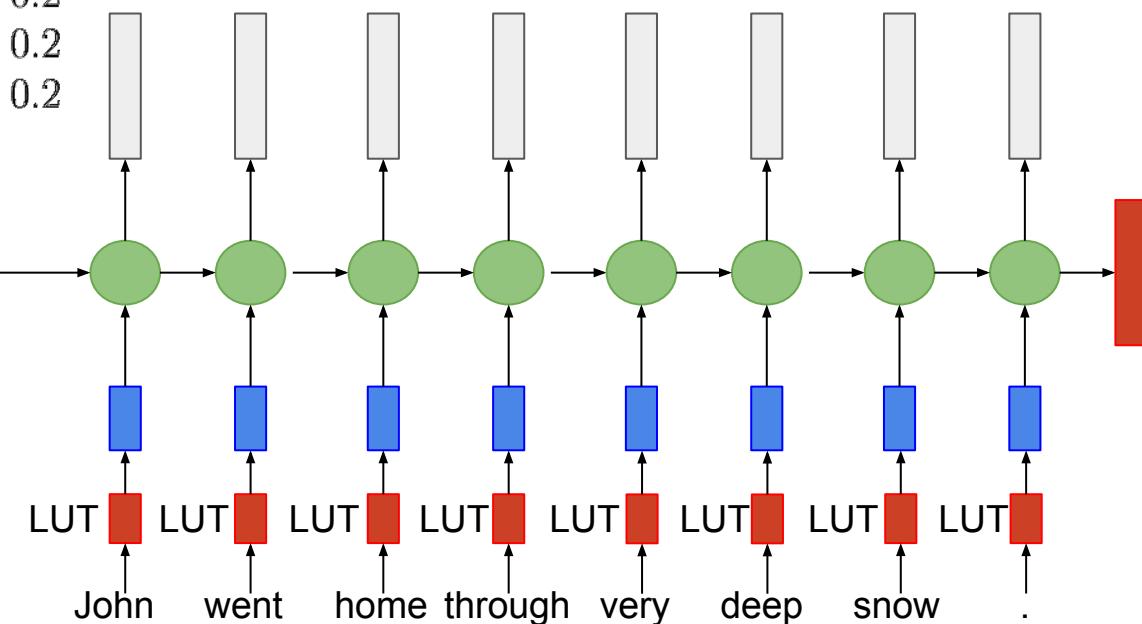
Jeden ze způsobů je nejdřív text rozsekat například na slova, každému slovu přiřadit unikátní ID a pak dané slovo reprezentovat vektorem, kde jsou samé nuly a právě jedna jednička na pozici ID daného slova.

Násobení matice s takovým vektorem vybere (zkopíruje) jeden sloupec (nebo řádek při opačném zápisu) z násobené matice.

Word embedding - LUT

<i>snow</i>	0.2	0.2	0.2	0.2	0.2	0.2	0.2
<i>went</i>	0.2	1.2	0.2	1.0	0.0	0.2	0.2
<i>john</i>	0.2	0.2	0.2	0.2	0.2	0.2	0.2
<i>deep</i>	0.2	0.2	0.2	0.2	0.2	0.2	0.2
<i>very</i>	0.2	1.2	0.2	1.0	0.0	0.2	0.2
<i>home</i>	0.2	0.1	0.2	0.2	0.2	0.2	0.2

Vytvářet dlouhé vektory s nulami a jen jednou jedničkou a násobit tím pak matici je velmi výpočetně neúsporné - všechna násobení nulami jsou zbytečná a neovlivňují výsledek. Proto se používá přímé indexování ID slova do LUT. Je to to samé, ale rychlejší.



Word embeddings

monarch → [0.1, 0.5, 0, -2.5, ..., -0.2, 3.0]

“monarch” close to “king”
“monarch” different to “driver”

Vektory v LUT reprezentují slova (jejich vlastnosti). Používaný pojem je “word embeddings”, protože ty vektory “embedují” (umísťují) slova do mnoharozměrného prostoru daných vektorů. Tyto embeddingy se trénují spolu se zbytkem sítě a v ideálním případě by pak měly jasně odpovídat vlastnostem slov - například synonima by měla mít skoro stejný embedding.

Similar words appear in similar contexts



: Center Word



: Context Word

c=0 The cute **cat** jumps over the lazy dog.

c=1 The **cute** **cat** **jumps** over the lazy dog.

c=2 **The** **cute** **cat** **jumps** **over** the lazy dog.

První metody pro word embedding používaly principu, který je možné znodušit do: slova, která se používají ve stejném kontextu by měly mít stejný embedding.

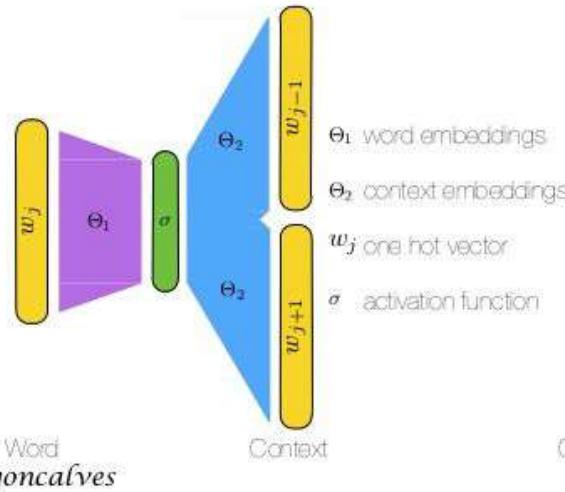
Word2Vec

Původní práce Tomáše Mikolova používala jednoduché neuronové sítě s jednou skrytou vrstvou.
Skipgram se snaží predikovat slova v okolí na základě prostředního slova.
CBW - Continuous Bag of Words naopak se snaží odhadnout prostřední slovo na základě okolí.

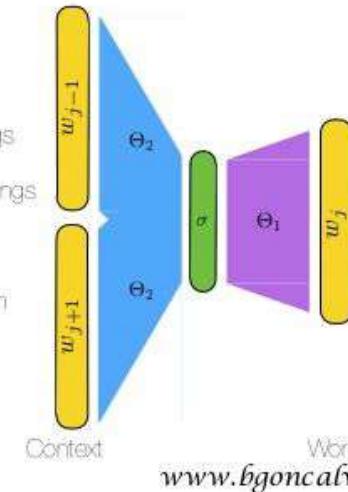
word2vec

Mikolov 2013

Skipgram
 $\max p(C|w)$



Continuous Bag of Words
 $\max p(w|C)$



@bgoncalves

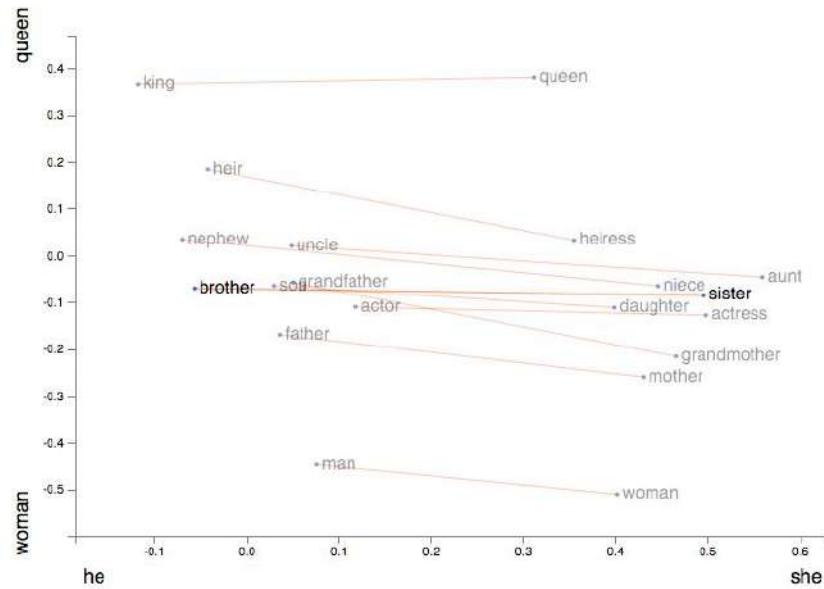
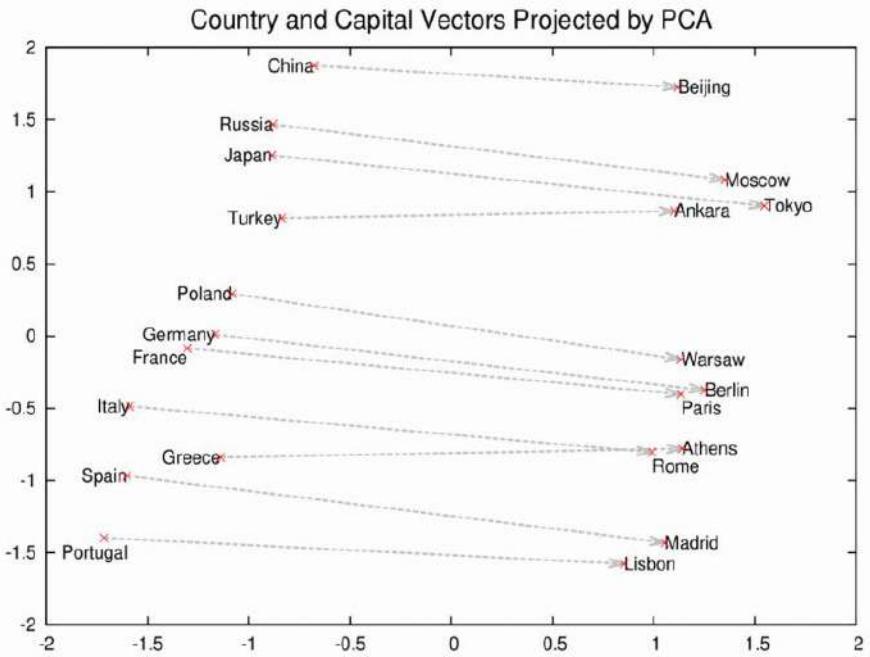
www.bgoncalves.com

Prostor word embeddingů by měl zachycovat vztahy mezi slovy, jejich role a podobně - sémantické i syntaktické. Struktura prostoru bývá často natolik jednoduché, že vztahy je možné vyjádřit posunutím (přičtením konstantního vektoru).

Tady je ukázka vizualizace několika vztahů - hlavní města a státy, muži ženy.

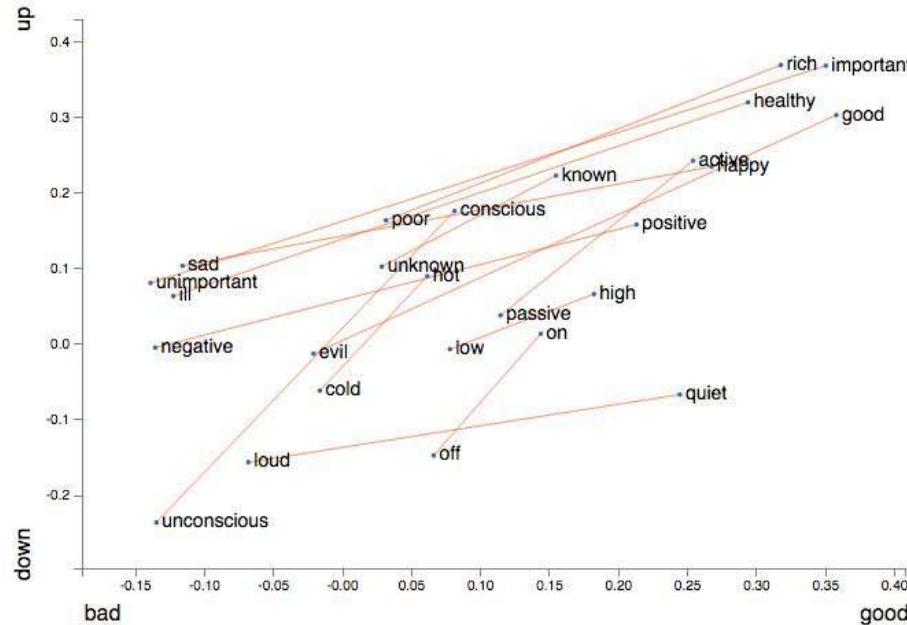
Vizualizovány jsou průměty celého prostoru embeddingu (např 512 dimenzí) do 2D pomocí PCA (512D data se jinak špatně zobrazují na obrazovce).

Word Vector math



Word Vector math

Další příklad pozitivní/negativní.



fastText

Alternative Gensim

- Each word is represented as a bag of character n-grams in addition to the word itself
- Word vectors for 157 languages
- Aligned word vectors for 44 languages (Czech)
 - Joulin et al.: Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion, 2018.
- Others: WordPiece embeddings (Wu et al., 2016)

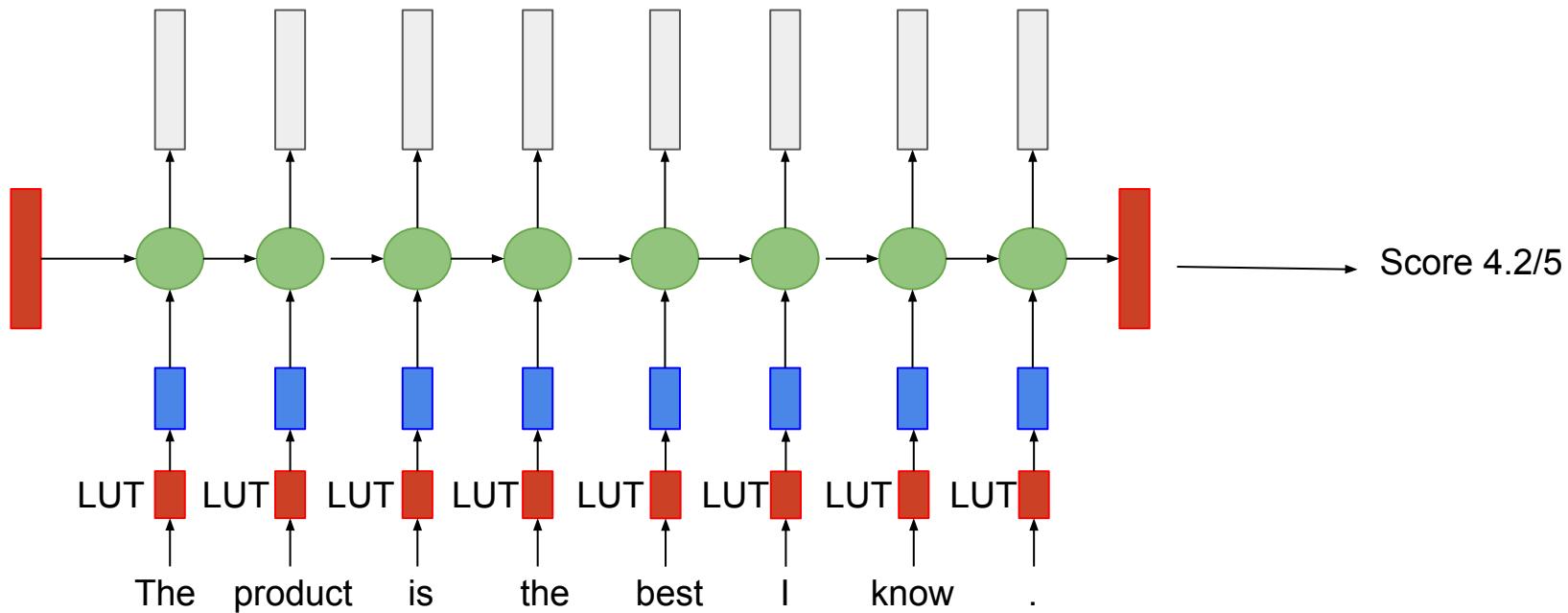
<https://fasttext.cc/>

Pokud potřebujete používat word embeddingy, můžete použít některou z existujících knihoven. Například FastText je příkladem takových klasických embeddingů v mnoha jazycích.

Většinou se ale v dnešních přístupech používají celé předtrénované NN na úloze doplňování slov (např. BERT dále v přednášce). Ty už poskytují tzv. kontextové embeddingy. Embedding nezávisí jen na slově samotném, ale bere v otaz i okolí. Takové embeddingy lépe vystihují význam reprezentovaných slov.

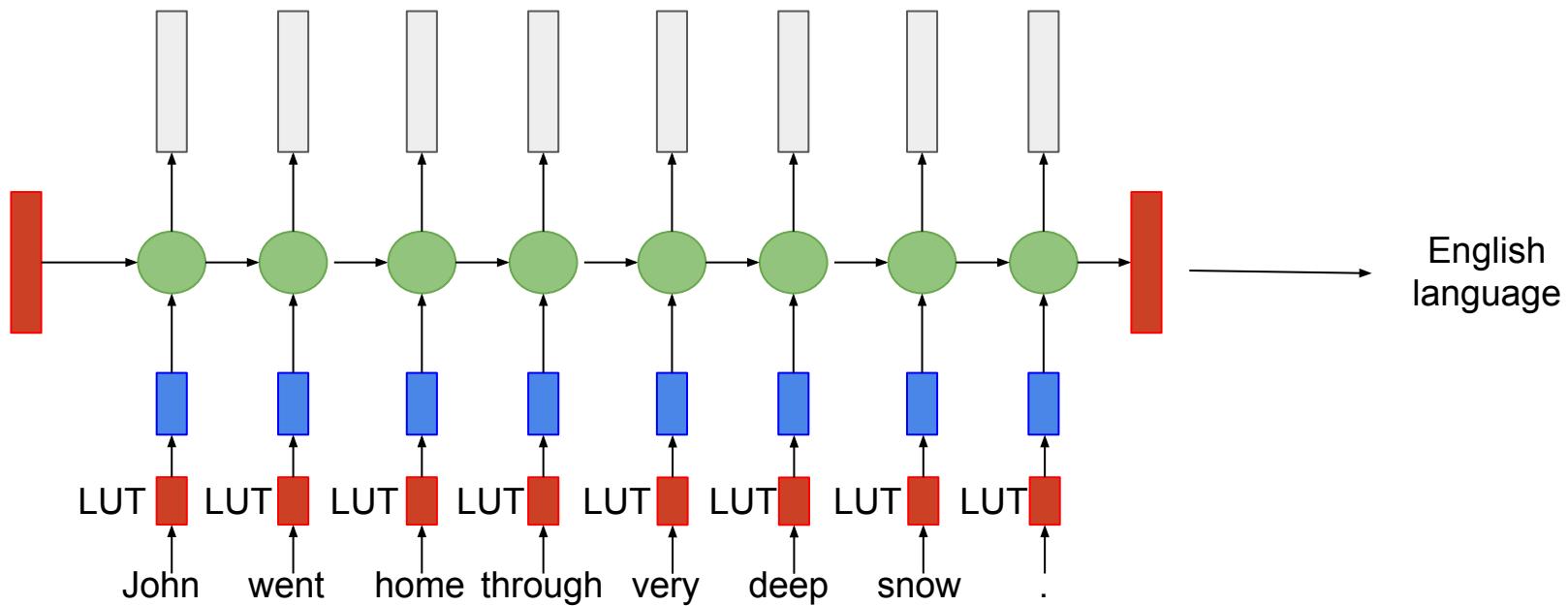
Regression

Regresní úlohu z textových dat je možné řešit i takto jednoduše - jedna rekurentní vrstva a výstup sítě napojit na její konečný stav.

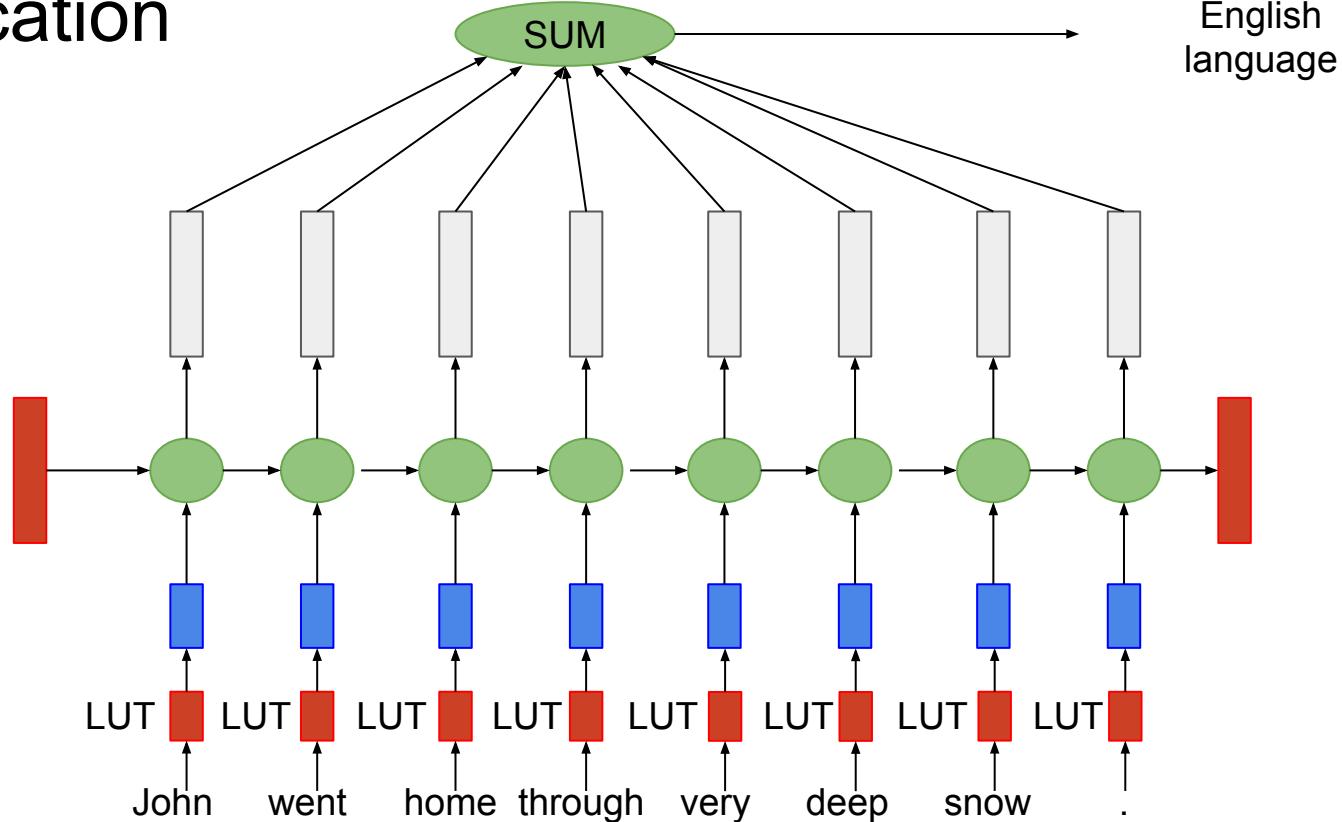


Classification

Stejně tak je možné řešit klasifikaci.



Classification



Problém s předchozími dvěma sítěmi je, že u dlouhých textů musí informace ze začátku textu projít mnoha buňkami rekurentní vrstvy než se informace z nich dostane k výstupu. Takové (hluboké) sítě se špatně učí. Mnohem lepší struktura sítě v předchozích případech by byla tato, kde se "proměrují" aktivace vrstvy z celé sekvence. Taková architektura umožňuje dát do souvislostí informace z celé sekvence (rekurentní vrstvou) a zároveň se informace o každém slově může dostat k výstupu průchodem je jednou rekurentní buňkou a není problém s učením takové sítě.

Classification

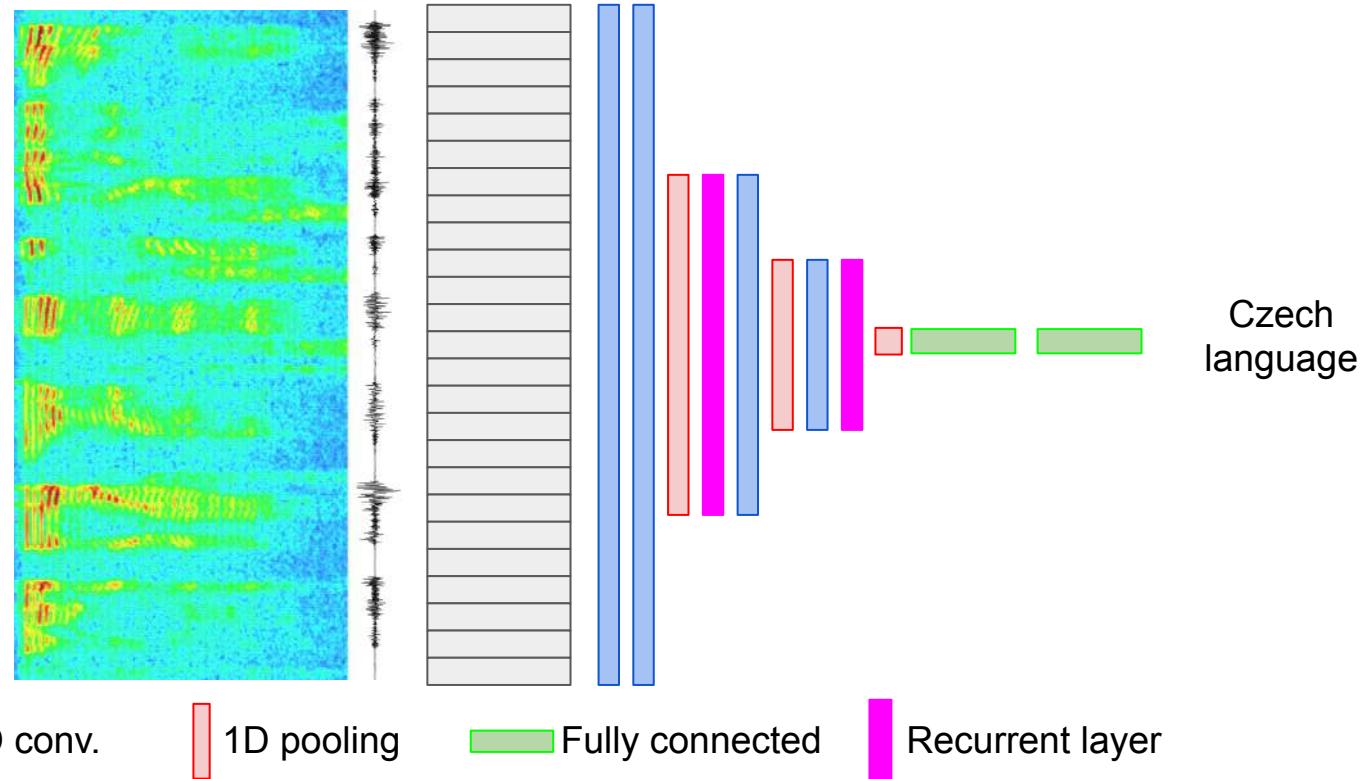
Ano, jen pro jistotu ještě znovu:

Sekvence jsou 1D struktury a nic nebrání je zpracovávat stejně jako obrázky - 1D konvoluce, 1D pooling vrstvy, stejné aktivace, ...

Klasifikační síť pro zvuky by klidně mohla být podobná jako VGG síť pro obrázky akorát v 1D.

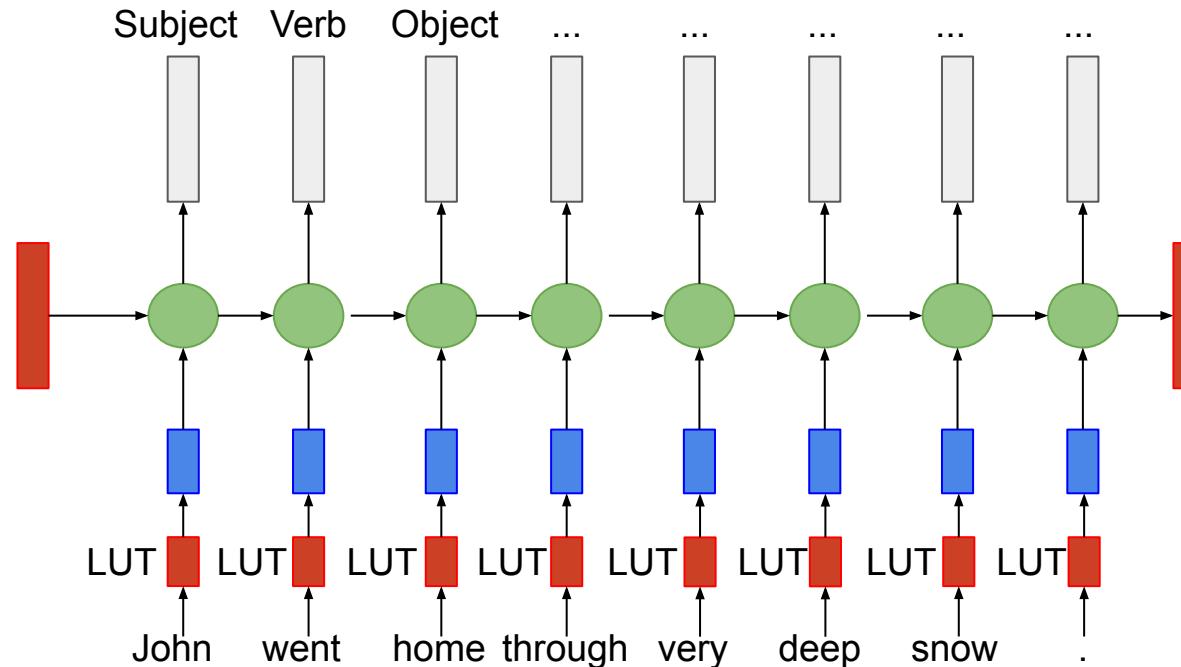
Třeba pro klasifikaci jazyka.

To platí i pro text.



Word tagging

Jedna u úloh na textu je “word tagging”. Například určování slovních druhů a jiných mluvnických kategorií. Je to úloha klasifikace každého slova. To je obdoba segmentace obrazu a řeší se stejně. Ale neklasifikují se pixely, ale slova. Pro každé slovo je zde klasifikační (softmax) výstup s cross entropy loss a celková chyba je součtem těchto jednotlivých chyb.



Reading Comprehension (<https://openai.com/blog/better-language-models/>)

The 2008 Summer Olympics torch relay was run from March 24 until August 8, 2008, prior to the 2008 Summer Olympics, with the theme of “one world, one dream”. Plans for the relay were announced on April 26, 2007, in Beijing, China. The relay, also called by the organizers as the “Journey of Harmony”, lasted 129 days and carried the torch 137,000 km (85,000 mi) – the longest distance of any Olympic torch relay since the tradition was started ahead of the 1936 Summer Olympics.

After being lit at the birthplace of the Olympic Games in Olympia, Greece on March 24, the torch traveled to the Panathinaiko Stadium in Athens, and then to Beijing, arriving on March 31. From Beijing, the torch was following a route passing through six continents. The torch has visited cities along the Silk Road, symbolizing ancient links between China and the rest of the world. The relay also included an ascent with the flame to the top of Mount Everest on the border of Nepal and Tibet, China from the Chinese side, which was closed specially for the event.

Q: What was the theme?

A: “one world, one dream”.

Q: What was the length of the race?

A: 137,000 km

Q: Was it larger than previous ones?

A: No (wrong?)

Q: Where did the race begin?

A: Olympia, Greece

Q: Is there anything notable about that place?

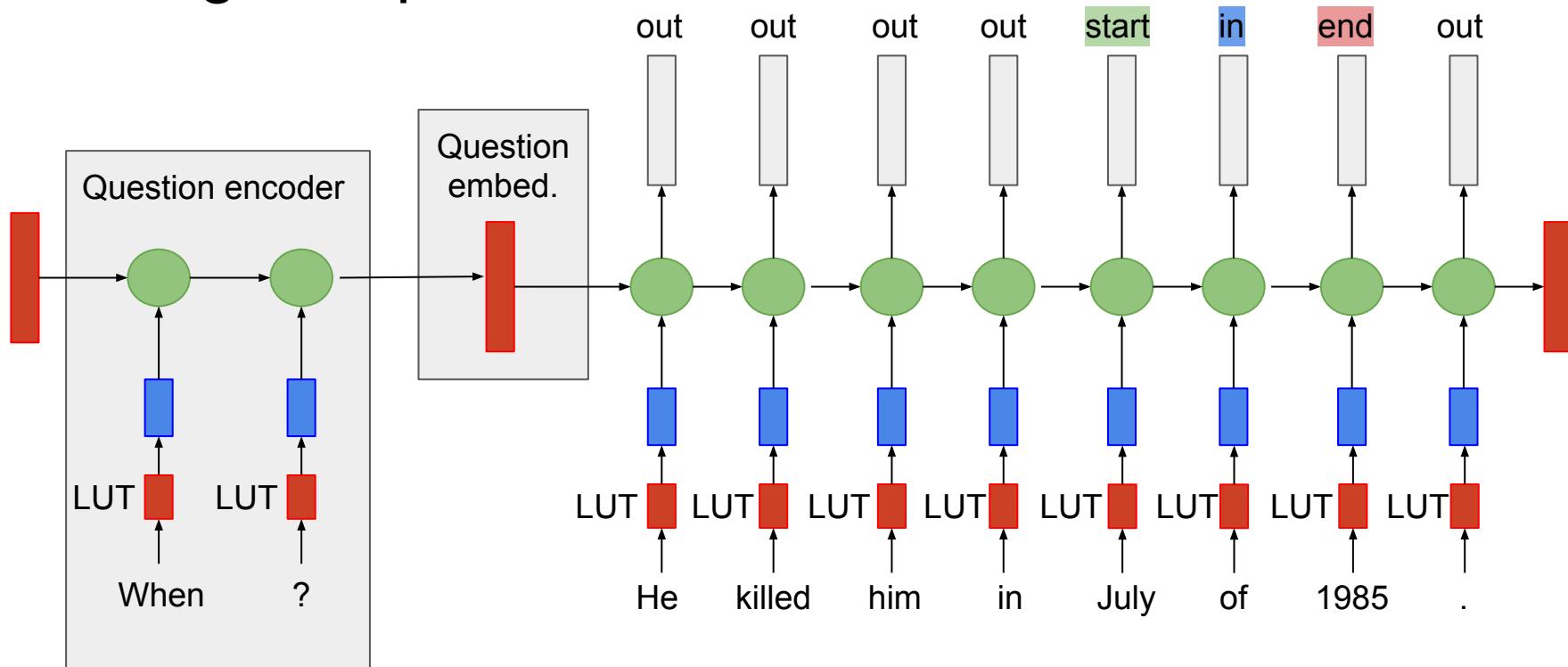
A: birthplace of Olympic Games

Q: Where did they go after?

A: Athens

Zajímavější, ale stále jednoduchá úloha je extractive reading comprehension. Zde slovo “extractive” znamená, že systém má identifikovat pasáž textu, která je odpověď na otázku. Takovou úlohu je možné formalizovat také jako word tagging.

Reading Comprehension

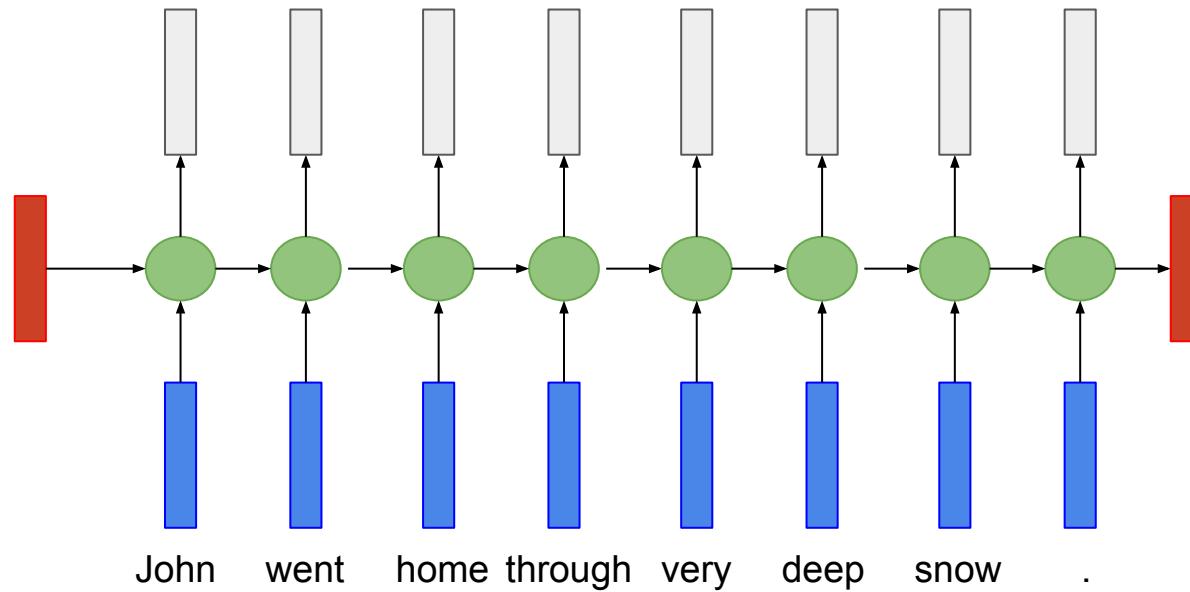


Možná struktura sítě pro extractive reading comprehension může být například tato. Rekurentní síť nejdřív zpracuje otázku, vytvoří z ní vektor (tady jako koncový stav první sítě). Tímto stavem je možné nastartovat druhou síť, která už označuje pasáž (klasifikuje slova, jestli jsou odpověď na otázku).

Tato síť zase bude mít problémy u dlouhých otázek a textů. Rozhodně to není příklad state-of-the-art sítě. To by dnes byly zase různé verze sítí s attention (BERT, ...).

Sequence to sequence models? Translation?

Honza šel domů skrz hodně hluboký sníh .



Honza šel domů
hodně hlubokým
sněhem.

Jak na překlad jazyka? Takto to nejde. Dva problémy jsou:

- 1) Není možné překládat slovo za slovo.
- 2) Není možné překládat každé slovo samostatně, protože výsledná věta musí být konzistentní. - Více informací k tomuto je v přednášce

Auto-regressive factorization (generative seq. models)

$P(\text{sentence} \mid \cancel{\text{John went home through very deep snow}})$

$$P(w_1, w_2, w_3, \dots, w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots P(w_n|w_{n-1}, w_{n-2}, \dots, w_1)$$

$$P(w_1, w_2, w_3, \dots, w_n) = \prod P(w_i|w_{i-1}, w_{i-2}, \dots, w_1)$$

Now we need a model which predicts probability of a single word given its history (prefix).

Při překladu je potřeba generovat celou konzistentní větu. To by bylo možné například tak, že vypíšeme všechny možné věty v daném jazyce budeme trénovat klasifikační síť, která má na vstupu větu v původním jazyce a snaží se přiřadit vysokou pravděpodobnost větám v cílovém jazyce, které jsou správným překladem. Pak už stačí některou z vět s vysokou pravděpodobností vybrat a prohlásit za výsledek překladu. To by bylo super a je to přesně to, čeho u překladu chceme dosáhnout. Bohužel toto řešení není technicky možné, protože všechny věty v jazyce není možné "vypsat" - na to jich je jednoznačně příliš mnoho :).

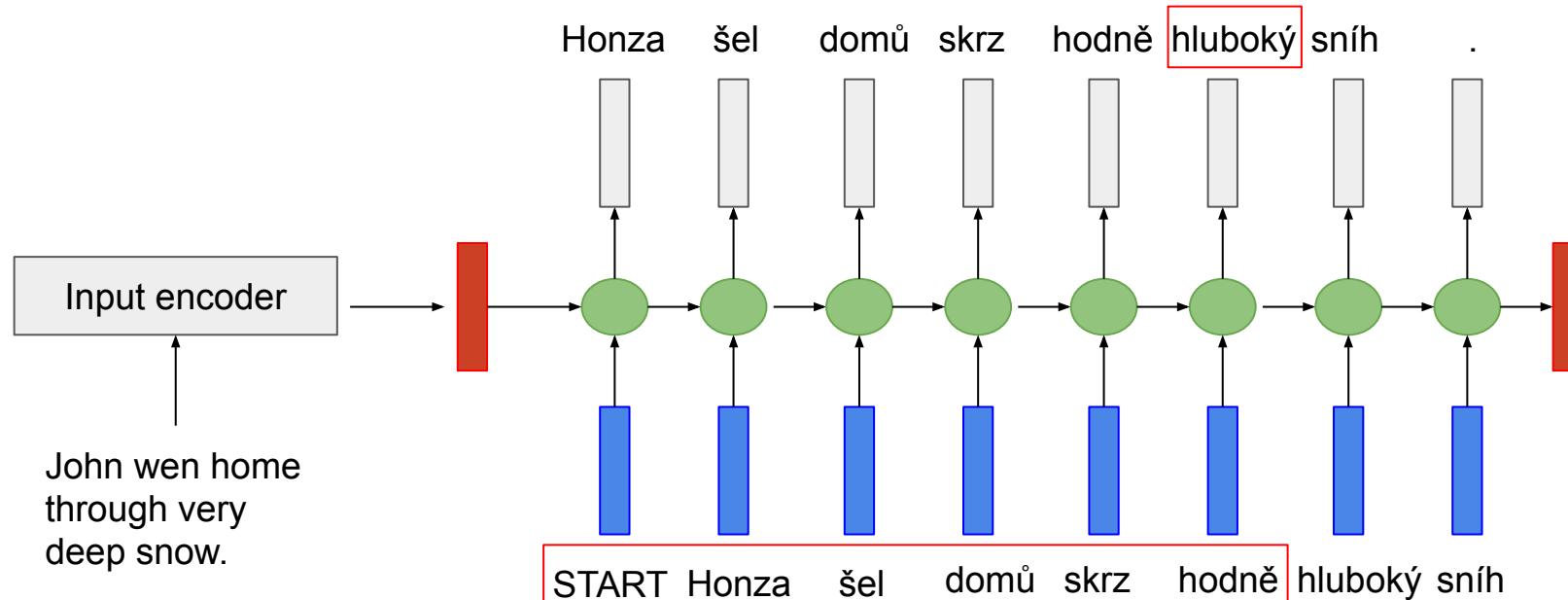
Existuje ale řešení: Rozložit pravděpodobnost celých vět na součin pravděpodobností všech slov, vždy pomíněných předchozími slovy ve větě. Taková faktorizace je "bezztrátová" - ekvivalentní. Tuto autoregresivní faktorizaci je možné modelovat neuronovou síť - síť která ma na vstupu slova prefixu a výstupu je klasifikátorem do tříd odpovídajím jednotlivým slovům v jazyce. Takovou síť pak můžeme použít pro odhad pravděpodobnosti výskytu věty v jazyce - spočítáme pravděpodobnosti všech slov věty a ty vynásobíme. Také může sloužit ke generování vět - sampling z odhadnutých rozdělení pravděpodobnosti každého slova postupně od začátku.

Pro modelování této faktorizace je možné použít různé sítě. Na následujících slajdech to jsou rekurentní sítě, ale dnes se hlavně používají sítě s Attention (Transformery), ale je možné použít i konvoluční a grafové sítě (nebo i jejich kombinace).

$$\begin{aligned} P(w_1, w_2, w_3, \dots, w_n) &= P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_{n-1}, w_{n-2}, \dots, w_1) \\ P(w_1, w_2, w_3, \dots, w_n) &= \prod P(w_i | w_{i-1}, w_{i-2}, \dots, w_1) \end{aligned}$$

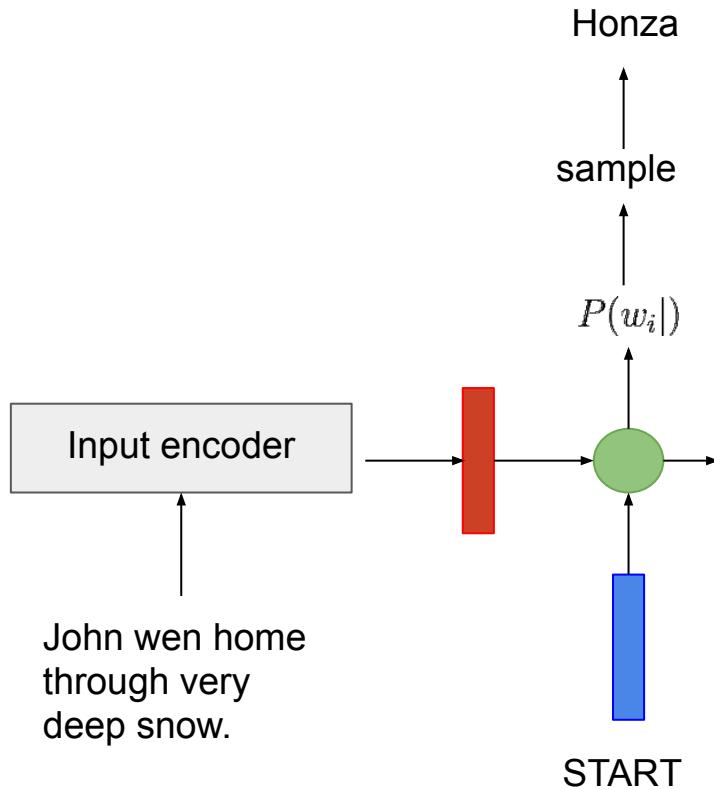
Seq2seq - sentence probability

$$P(w_1, w_2, w_3, \dots, w_n) = \prod P(w_i | w_{i-1}, w_{i-2}, \dots, w_1)$$



Na problém predikce pravděpodobnosti dalšího slova ve větě velmi dobře "pasují" rekurentní sítě. Prefix (již známá část věty) je na vstupu rekurentní vrstvy a na poslední pozici (posledním slově prefixu) rekurentní buňka predikuje další slovo. Výhoda takového uspořádání je, že pro další slovo se musí spočítat jen jedna buňka protože předchozí část rekurentní vrstvy nijak na tom novém slově nezávisí. Jen se využije (přenese) předchozí vnitřní stav.

Seq2seq - sentence output sampling

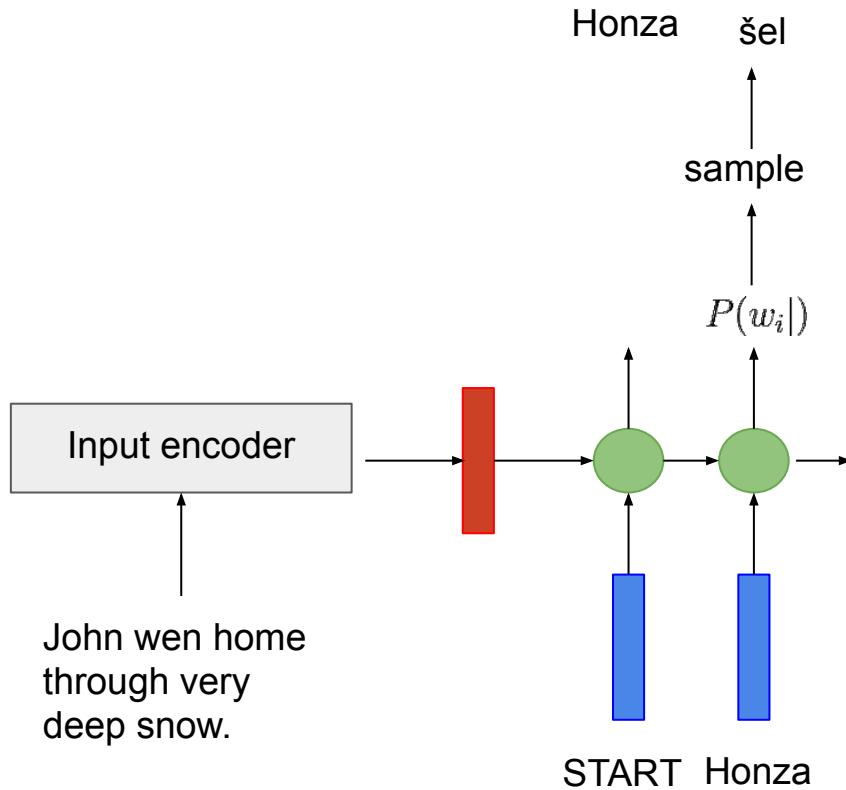


Generování výstupních sekvencí pomocí rekurentních sítí:

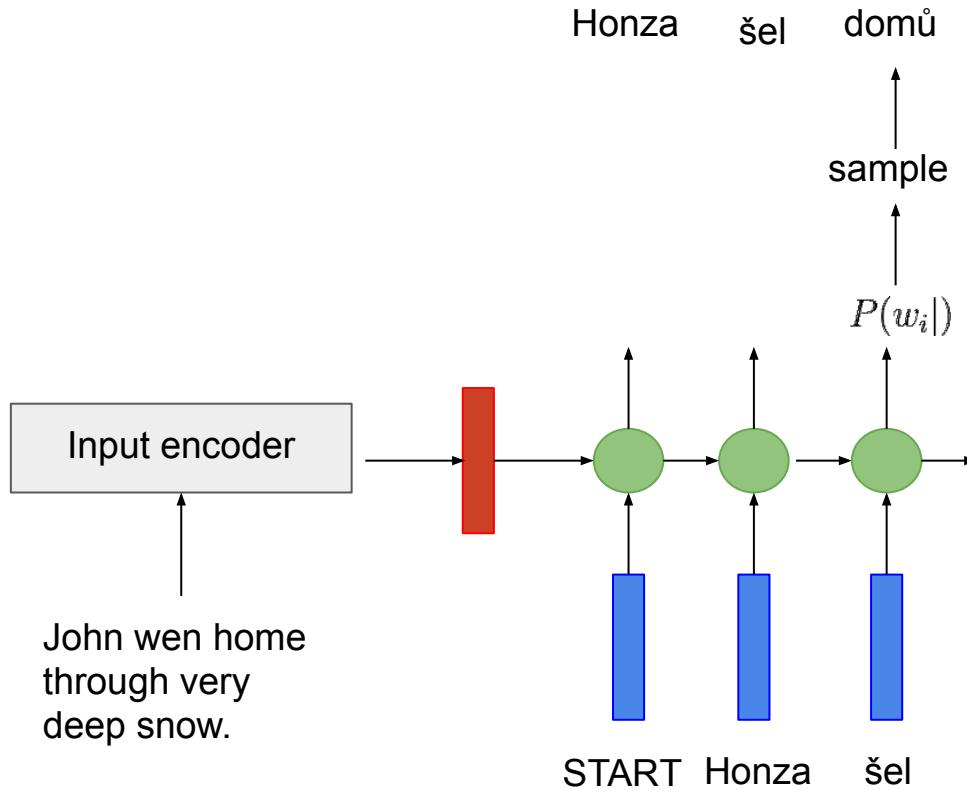
- 1) Na vstup sítě se nejdříve přikládá speciální symbol "START", který značí prázdný prefix.
- 2) Síť odhadne pravděpodobnost dalšího slova na základě posledního vstupního symbolu a předchozího skrytého stavu. Vyhodnotí se pouze jedna buňka rekurentní sítě.
- 3) Na základě odhadnutého rozdělení pravděpodobnosti se musí vybrat konkrétní slovo - Je možné vybrat nejpravděpodobnější slovo, čímž nakonec získáme nejpravděpodobnější větu, ale v mnoha aplikacích je potřeba více pravděpodobných verzí - v tom případě se vybírá náhodné slovo podle spočítaného rozdělení pravděpodobnosti.
- 4) Vybrané slovo se připojí k prefixu a přenese se na vstup sítě v dalším kroku a celý proces se opakuje od 2)

U překladu je potřeba proces generování ukončit. K tomu slouží speciální slovo/symbol (třeba může být označeno jako STOP). Pokud pak předchozí proces vygeneruje STOP, celý proces se ukončí.

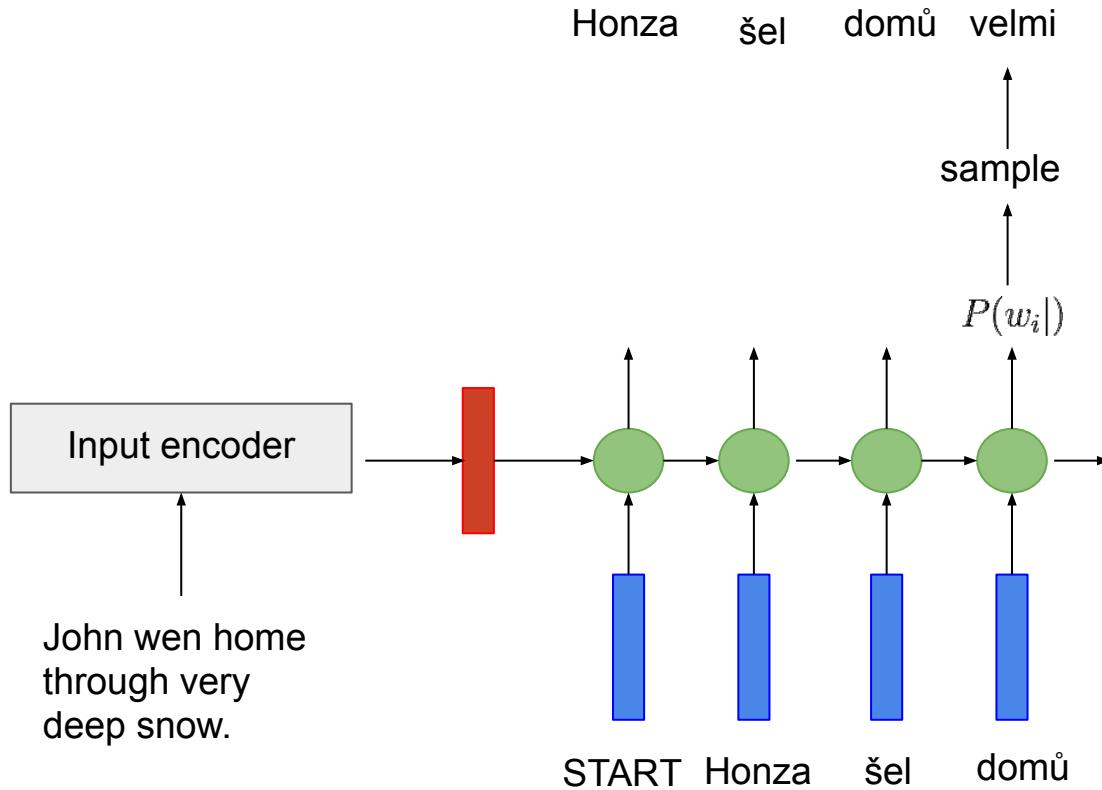
Seq2seq - sentence output sampling



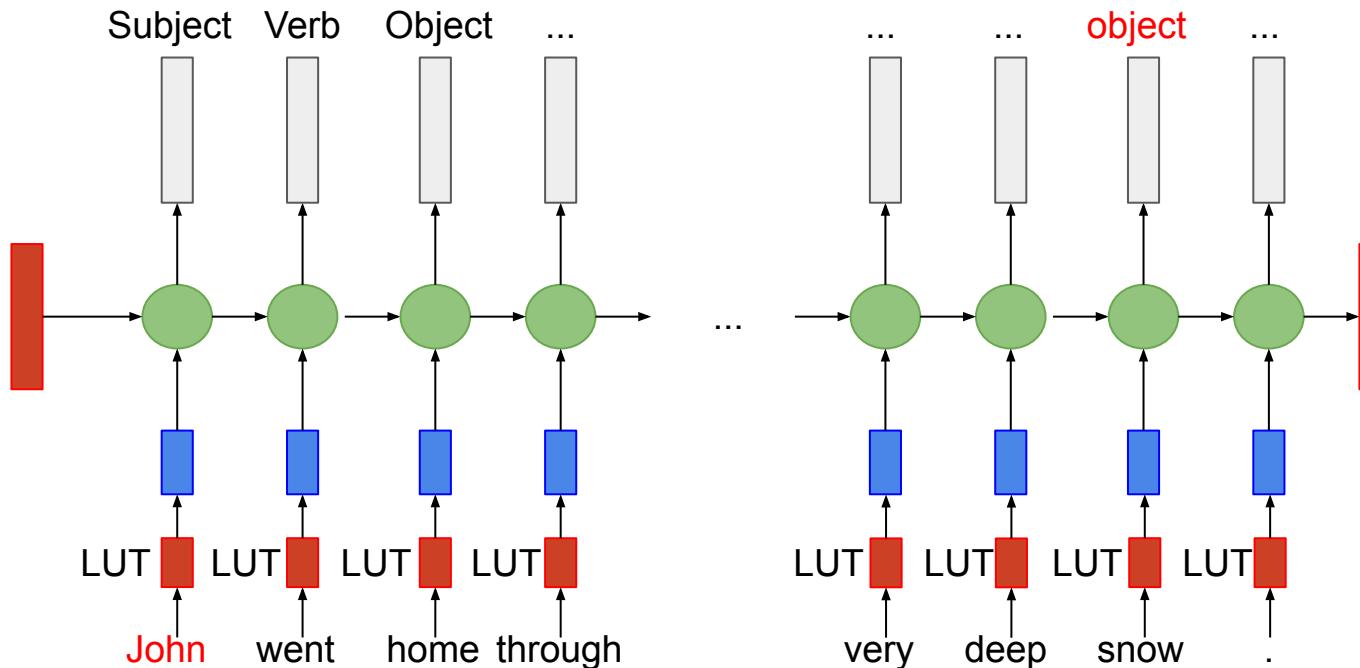
Seq2seq - sentence output sampling



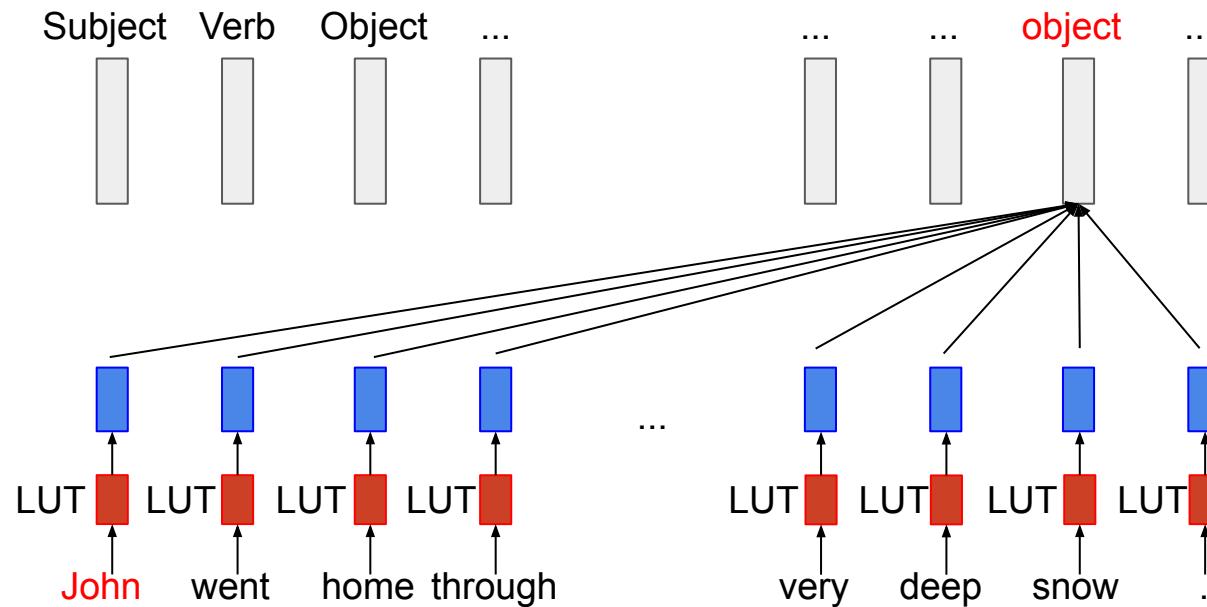
Seq2seq - sentence output sampling



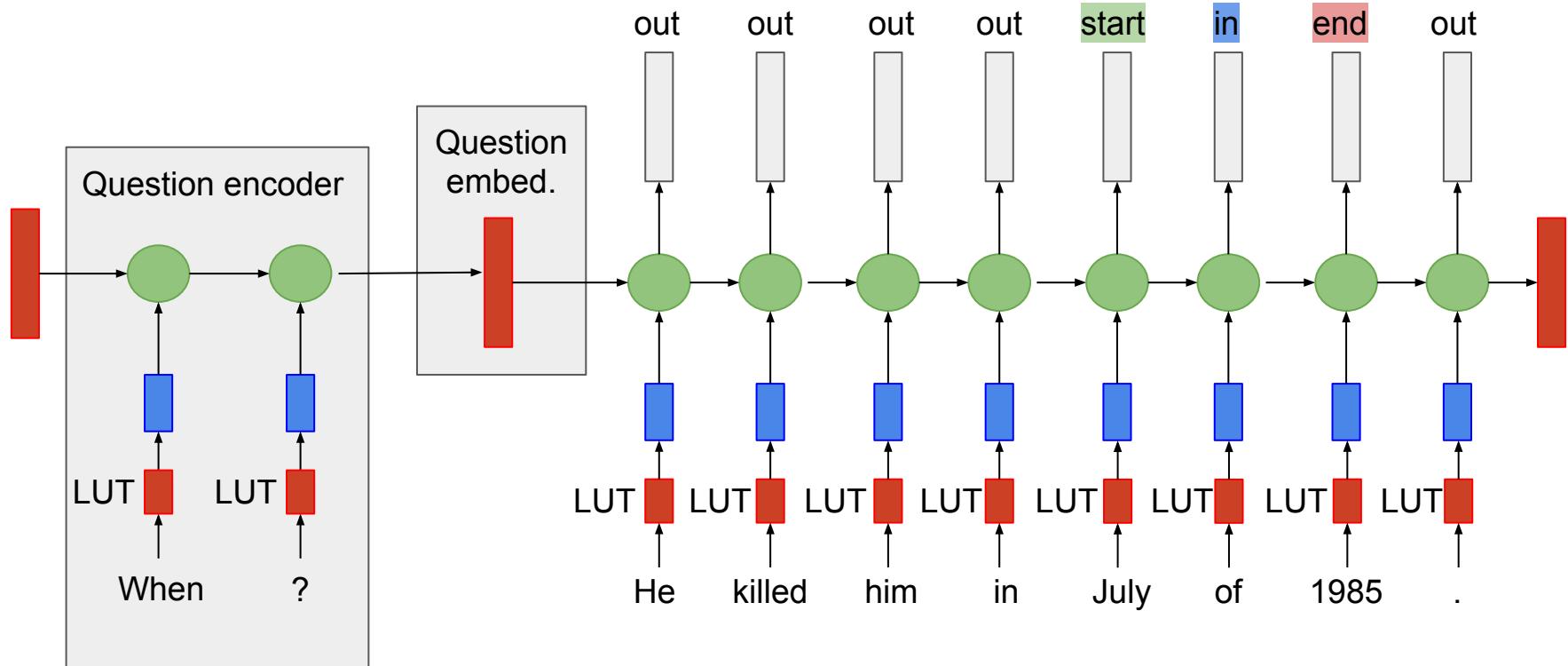
Attention - Learn long distance dependencies?



Attention?



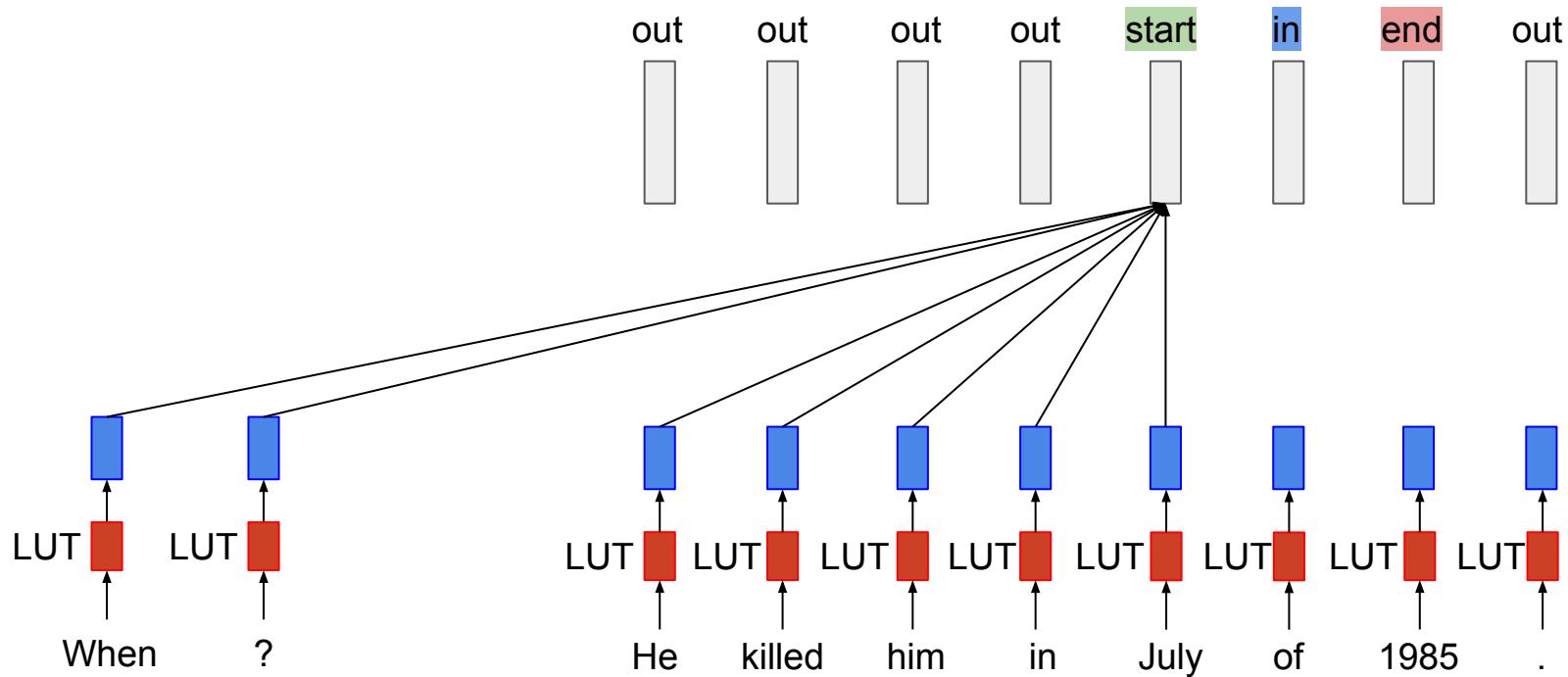
Attention - long distance dependencies



Možná struktura sítě pro extractive reading comprehension může být například tato. Rekurentní síť nejdřív zpracuje otázku, vytvoří z ní vektor (tady jako koncový stav první sítě). Tímto stavem je možné nastartovat druhou síť, která už označuje pasáž (klasifikuje slova, jestli jsou odpověď na otázku).

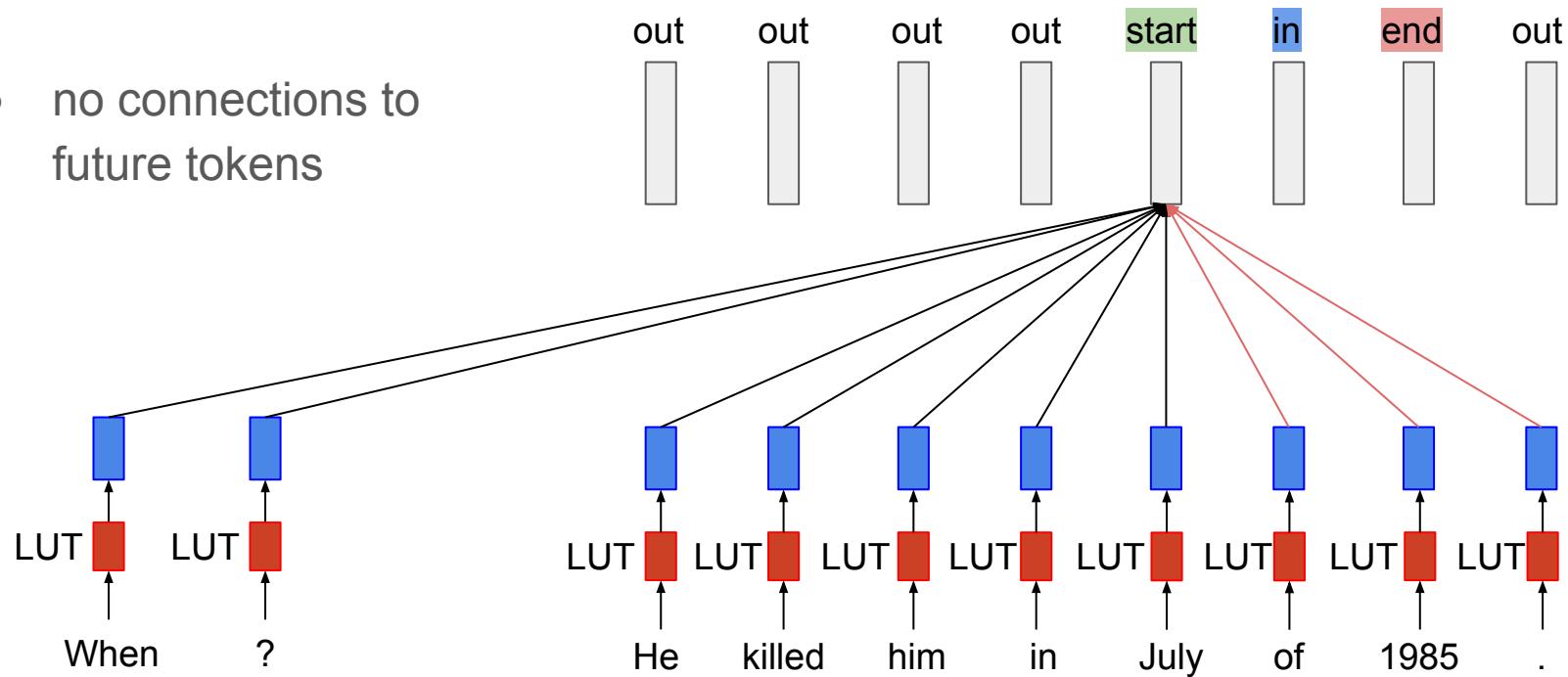
Tato síť zase bude mít problémy u dlouhých otázek a textů. Rozhodně to není příklad state-of-the-art sítě. To by dnes byly zase různé verze sítí s attention (BERT, ...).

Attention - long distance dependencies



Attention - causal attention mask / auto-regressive / causal models

- no connections to future tokens



Attention - text transcription

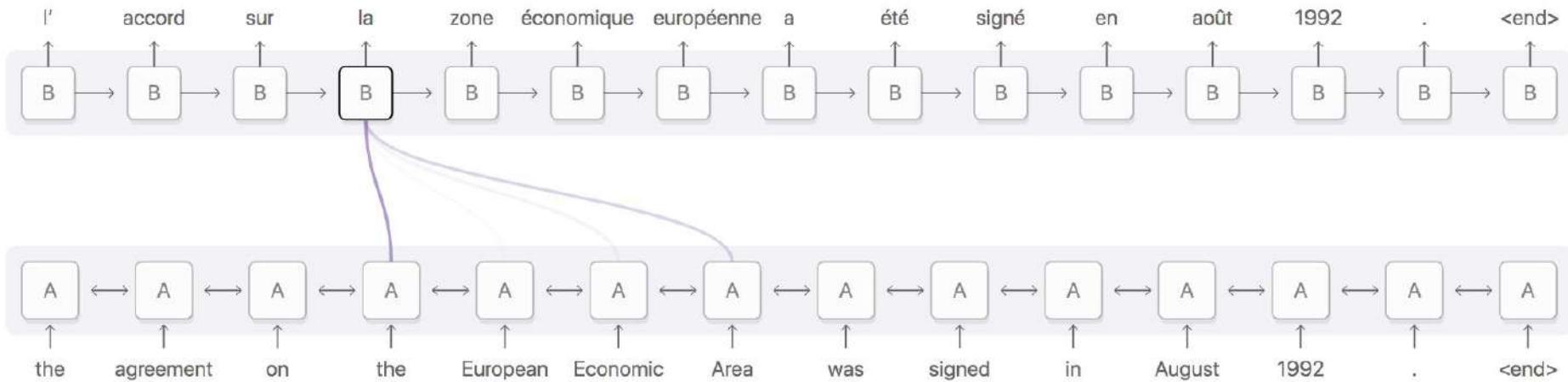
- Generate characters sequentially
- Learn to access information useful for each character

Attention je mechanizmus (vrstva sítí), která umožňuje, aby se síť naučila "vytahovat" potřebnou informaci ze zdrojových dat. Je to pevné spojení s celými zdrojovými daty, ale síť si dynamicky podle svého vnitřního stavu (aktivací) dokáže nastavovat váhu jednotlivých částí zdrojových dat --- naučí se přiřazovat vysokou váhu užitečným datům.

representative	r
representative	e
representative	p
representative	r
representative	e
representative	s
representative	e
representative	n
representative	t
representative	a
representative	t
representative	i
representative	v
representative	e

Attention

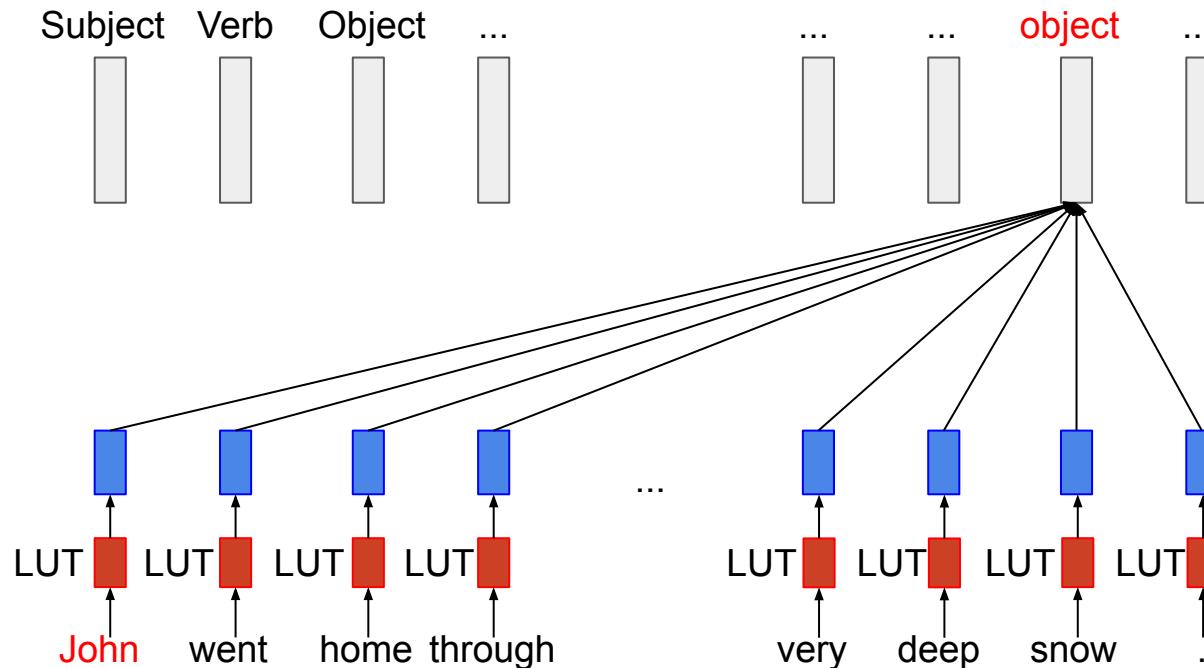
Attention se nepřenese informaci z jednoho zdrojového "místa", ale může přiřadit libovolný počet nenulových vah. Třeba u překladu jazyka v ukázce je to jednoznačně potřeba.



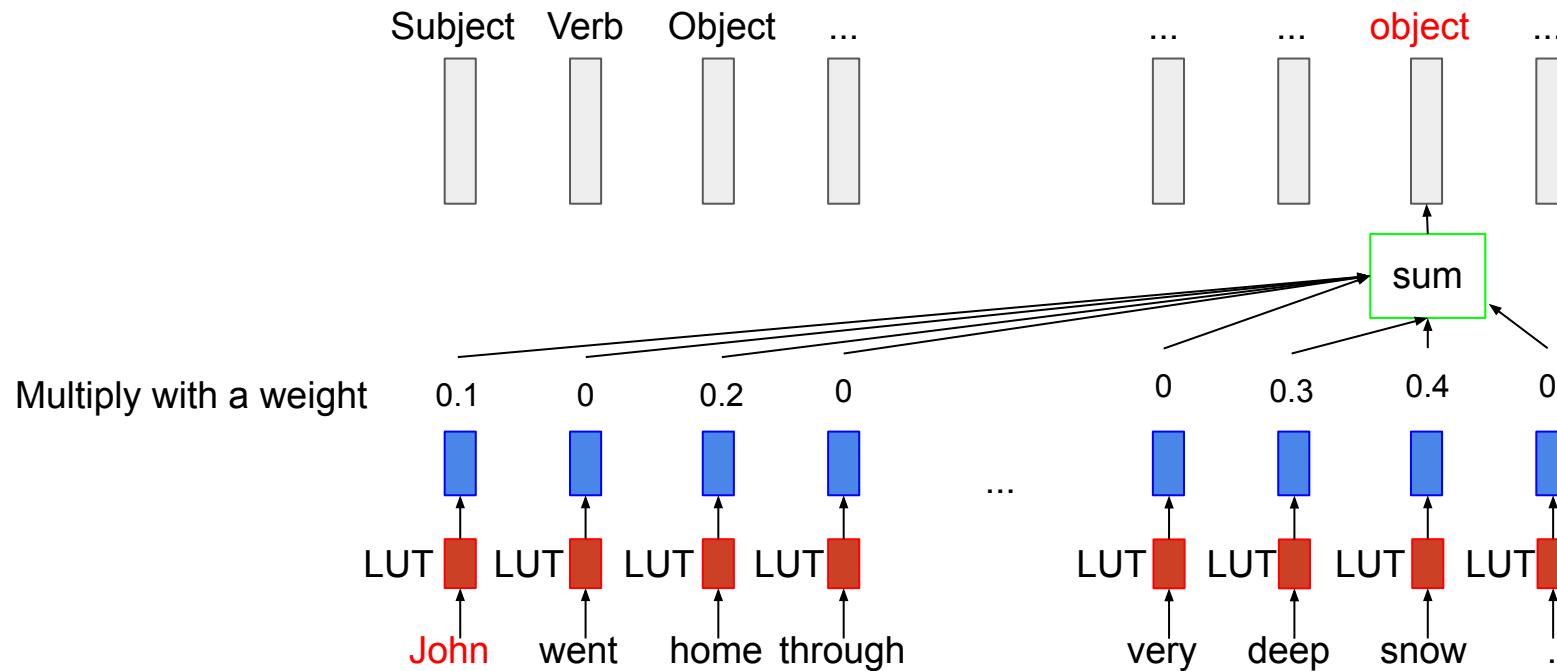
Olah, Carter: Attention and Augmented Recurrent Neural Networks, 2016.

<https://distill.pub/2016/augmented-rnns/>

Attention - mechanism



Attention - mechanism



Attention - computations

$$X \times W^Q = Q$$

$$X \times W^K = K$$

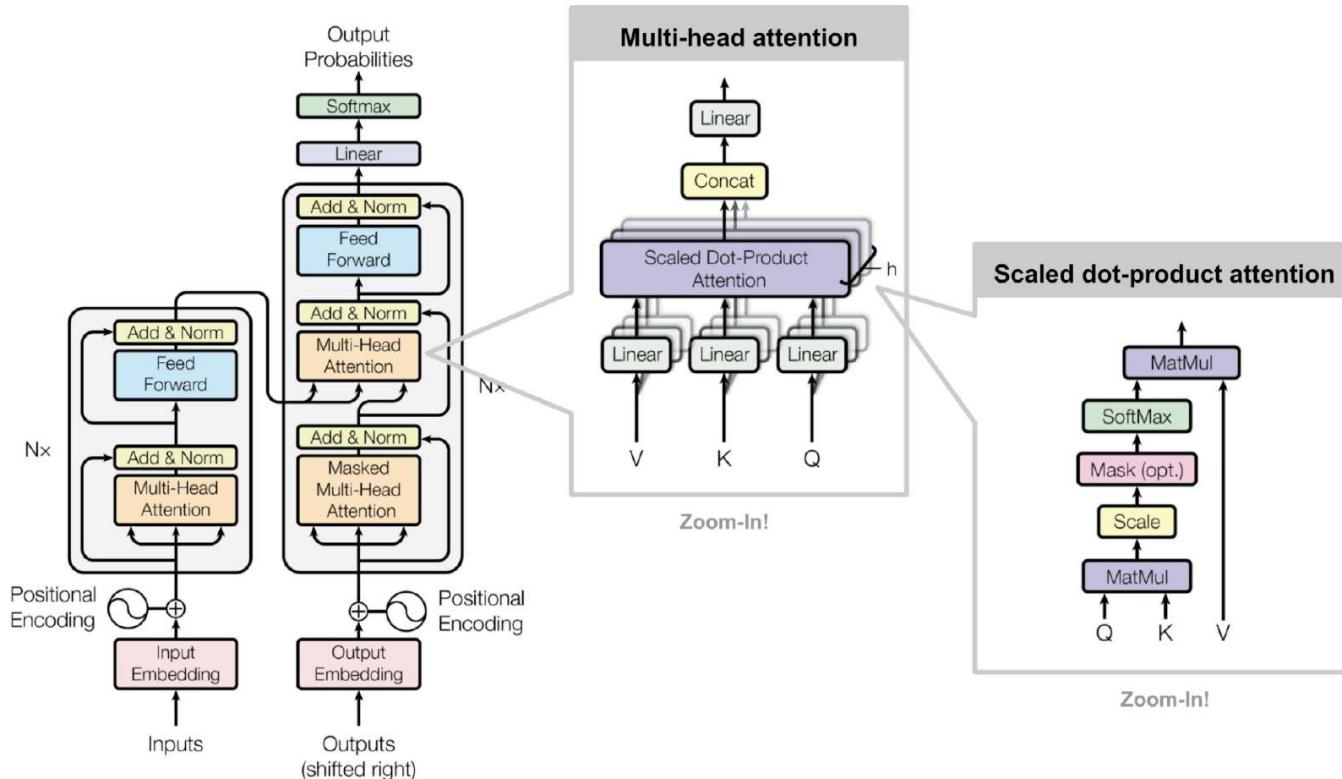
$$X \times W^V = V$$

$$\text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) = Z$$

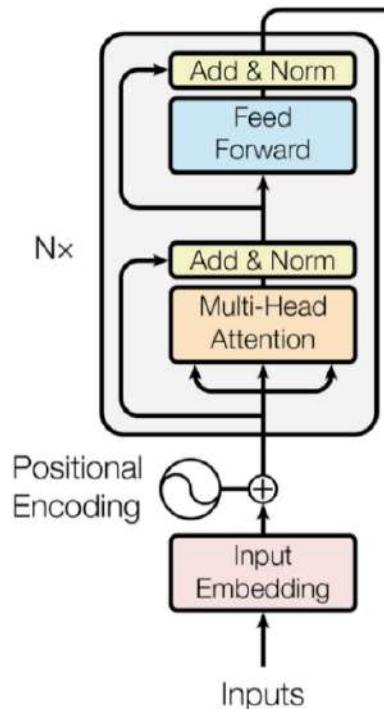
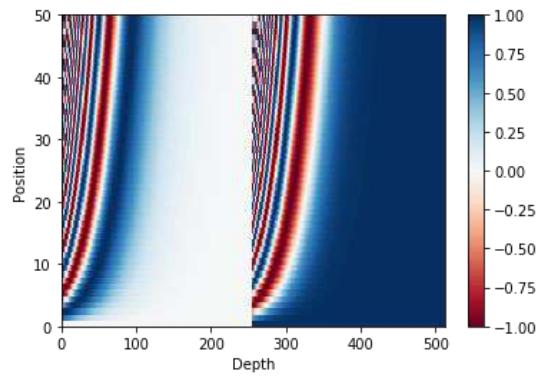
The self-attention calculation in matrix form

Attention Is All You Need (Vaswani et al, NIPS 2017)

Tool: [Tensor2tensor](#)



Transformer - positional encoding



Multitask training data (680k hours)

English transcription

- 🔊 "Ask not what your country can do for ..."
- 🔊 Ask not what your country can do for ...

Any-to-English speech translation

- 🔊 "El rápido zorro marrón salta sobre ..."
- 🔊 The quick brown fox jumps over ...

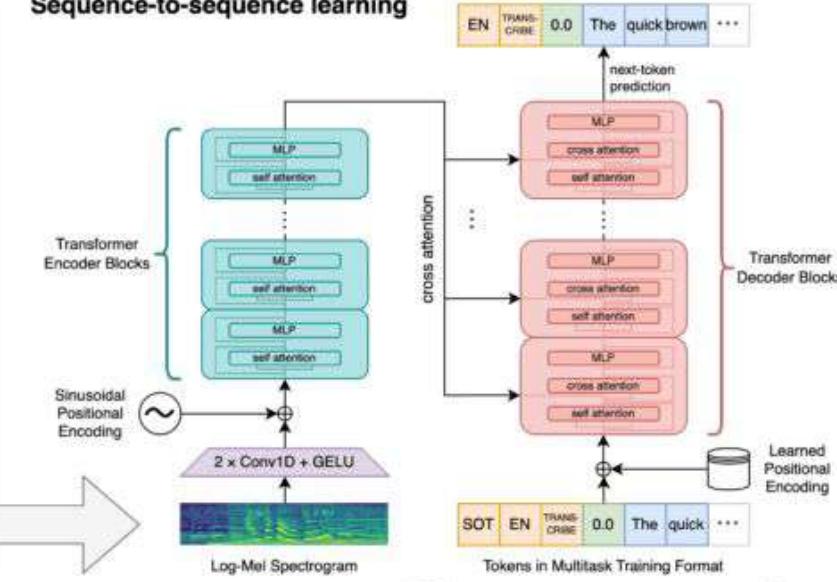
Non-English transcription

- 🔊 "언덕 위에 물라 내려다보면 너무나 넓고 넓은 ..."
- 🔊 언덕 위에 물라 내려다보면 너무나 넓고 넓은 ...

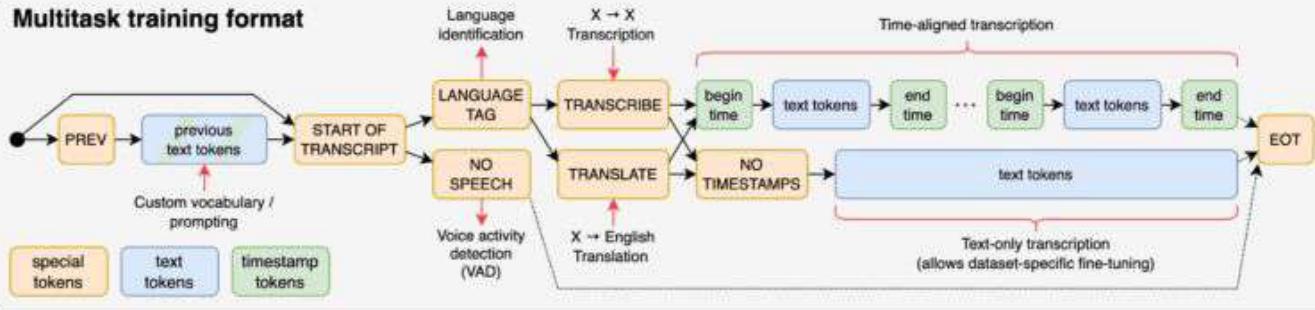
No speech

- 🔊 (background music playing)
- 🔊 Ø

Sequence-to-sequence learning



Multitask training format



Encoder-decoder vs. decoder only

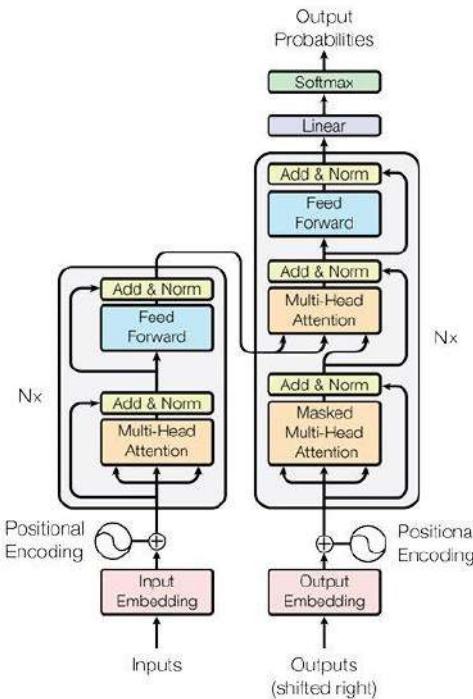
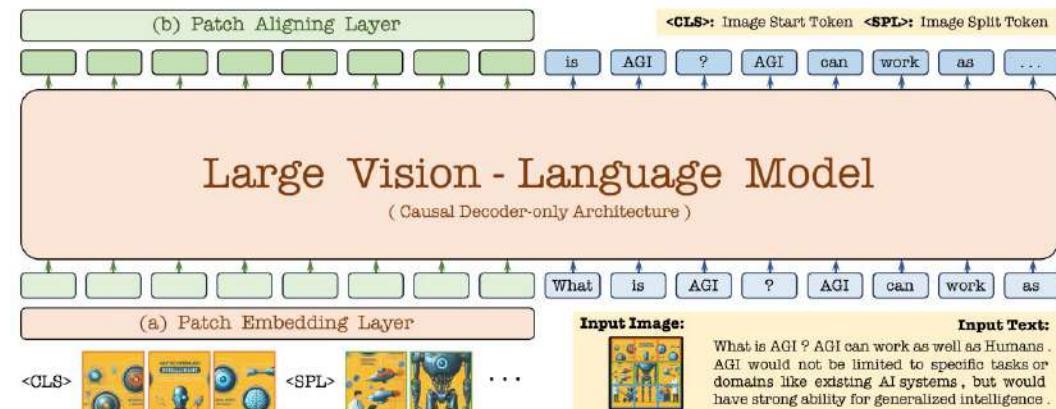


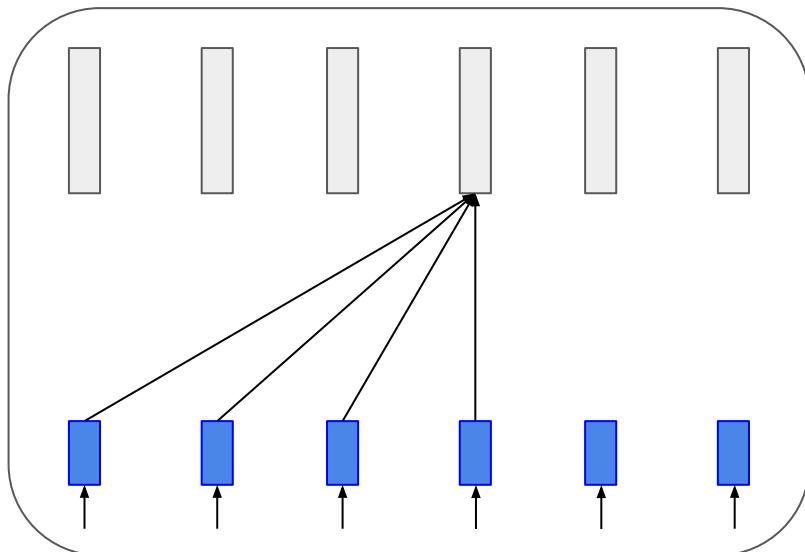
Figure 1: The Transformer - model architecture.



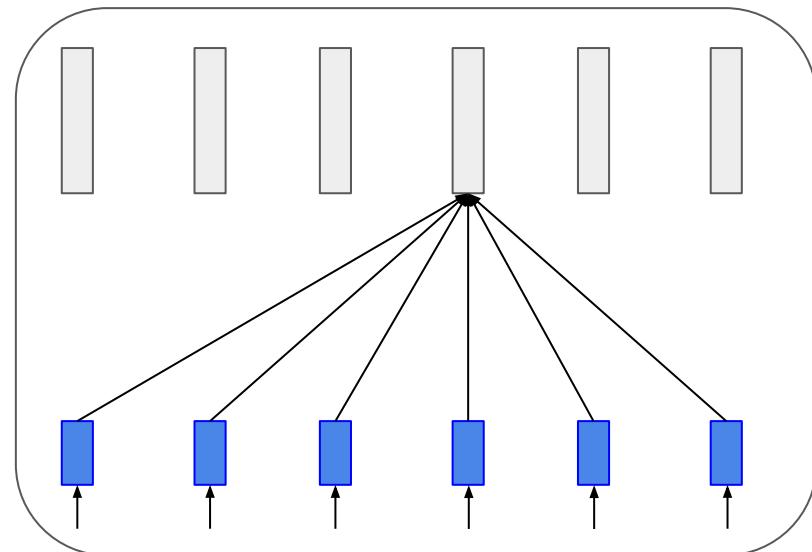
Causal vs. full attention

- set future attention weights to 0/-inf

Transformer Decoder
causal

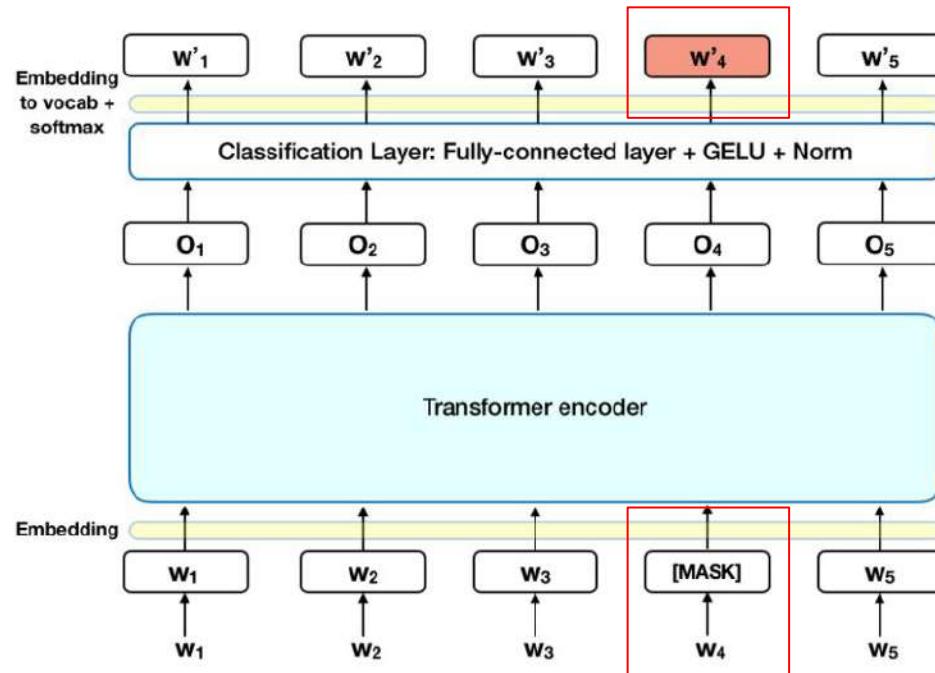


Transformer Encoder
full



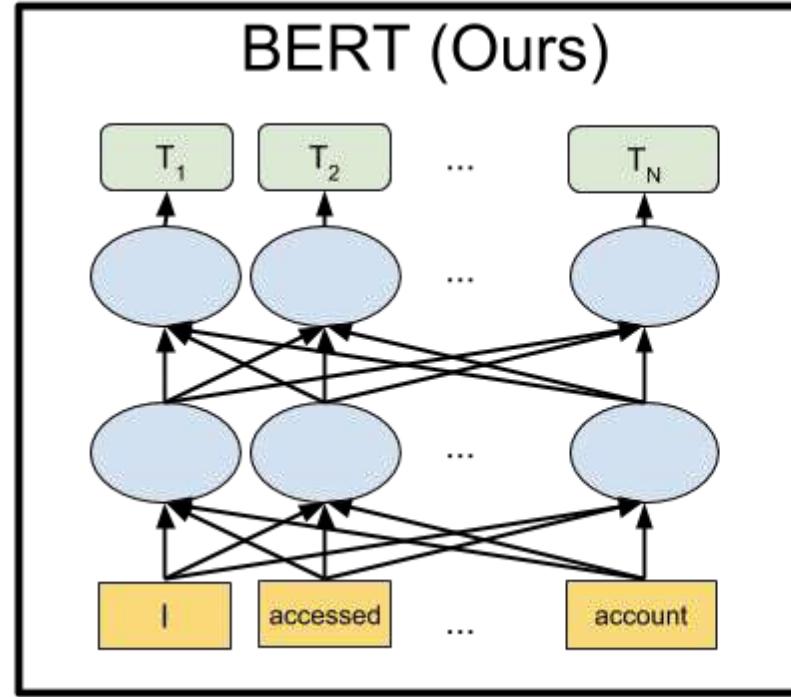
BERT - Bidirectional Encoder Representations from Transformers

Trained to fill in masked words in a sentence.

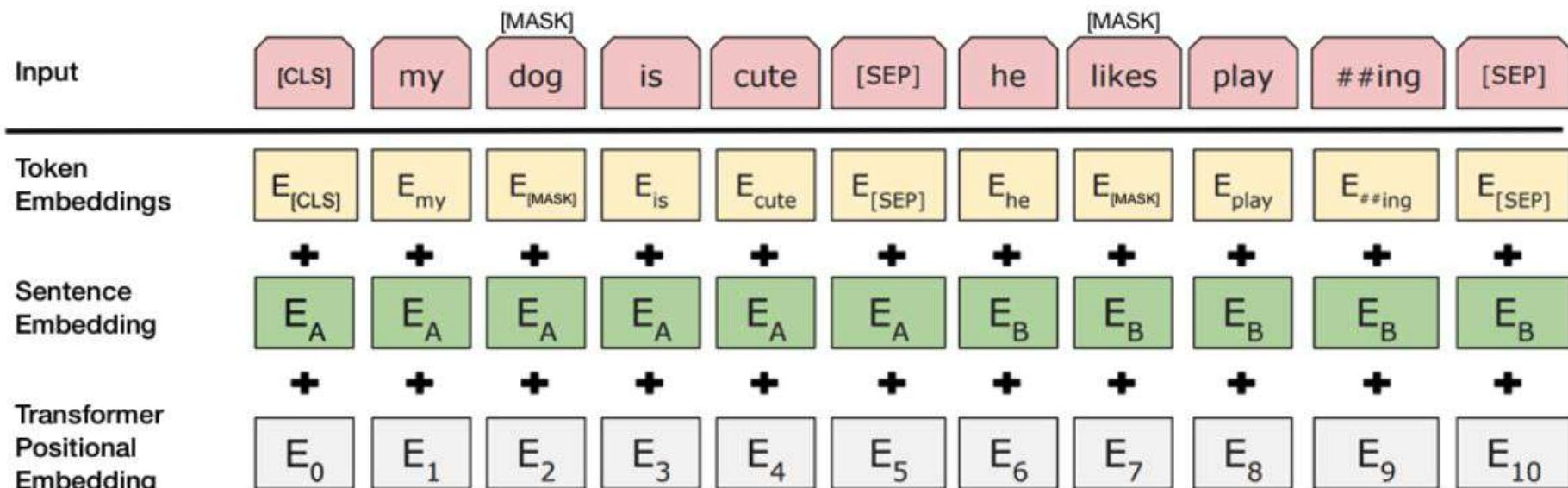


BERT - pretraining

Masked LM

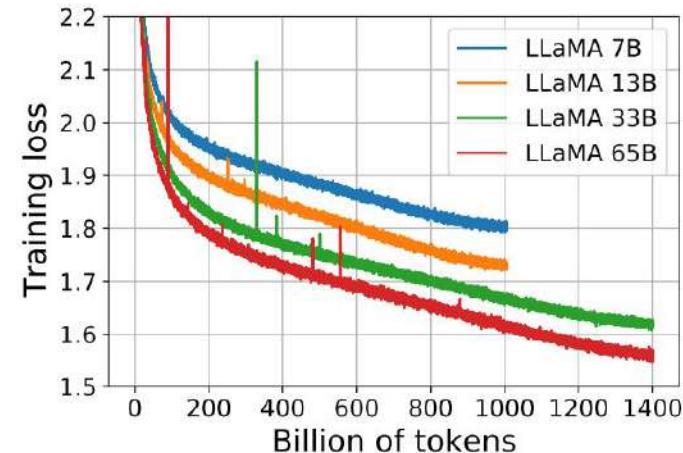


Bert input

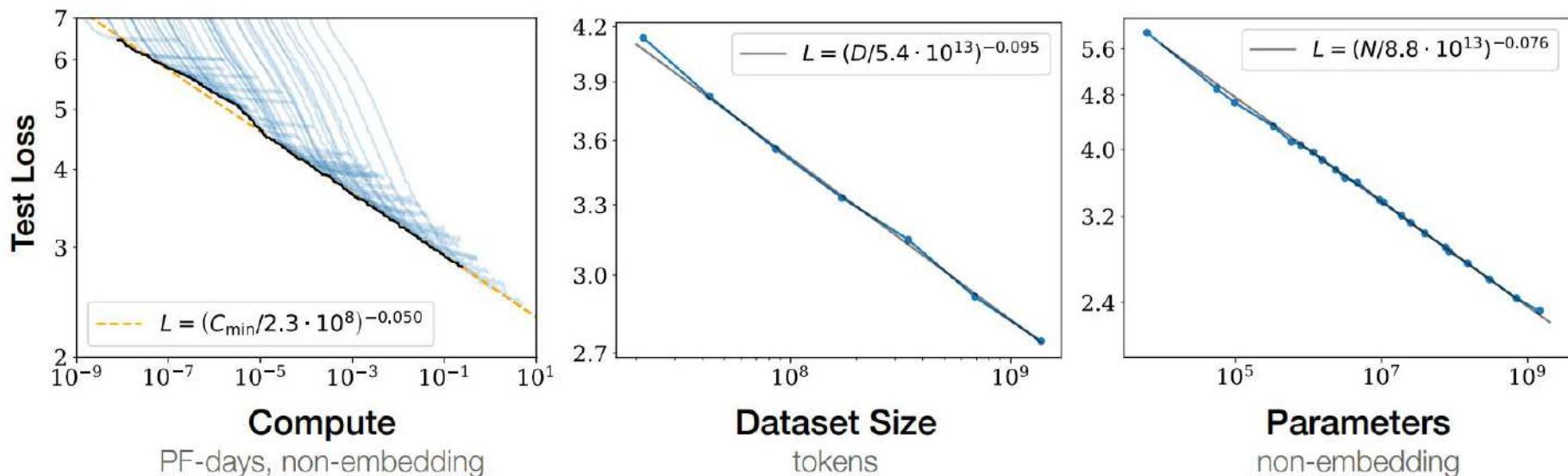


Technical challenge - training / inference

- Model does not fit on a single GPU or even single multi-GPU node
- Model parallel and data parallel training
- LLaMA:
 - 32.5B, dim. 6656, hads 52, layers 60
 - batch size 4M tokens
 - training 1.4T tokens = 350000 iterations
 - 380 tokens/sec/A100 GPU with 80GB
 - Training on 2048 A100 GPUs
 - Training time 21 days



Model scaling laws



Model Alignment

- We (sometimes) want the models to useful to humans
 - (and NOT to provide bad press for the companies)
 - (and NOT to cause lawsuits for the the companies)
- Some people may be:
 - offended by the models for many reasons
 - offended that the model talks to someone else about some topics
 - offended that models are biased
 - angry that models “harm children”
 - angry that models provide some “harmful” information
- If we get AGI? How will we assure that it acts in the “best interest” of humanity?

Language modelling

(<https://openai.com/blog/better-language-models/>)

Text seed: In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Generated text: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

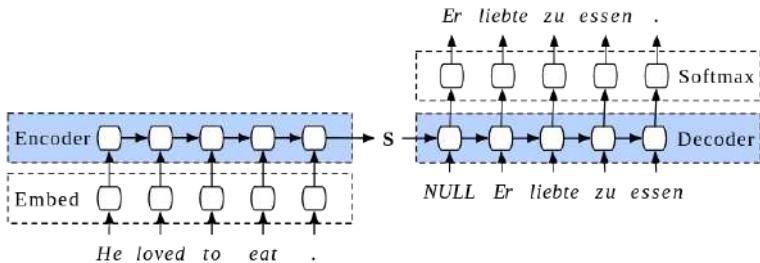
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

Pérez and the others then ventured further into the valley. “By the time we reached the top of one peak, the water looked blue, with some crystals on top,” said Pérez.

Pérez and his friends continued to climb the mountain, until they reached the top. There, they found a herd of unicorns, which had been living in the valley all along.

Machine translation



The machine learning
summer school in Brno will
surely be great!

→ Google Translate →

Letní škola strojového učení v Brně jistě
bude skvělá!

브르노 (Brno)의 여름 학교를
학습하는 기계가 반드시 좋을
것입니다!

Летняя школа машинного
обучения в Брно,
безусловно, будет
отличной!

L'école d'été d'apprentissage
automatique à Brno sera
sûrement géniale!

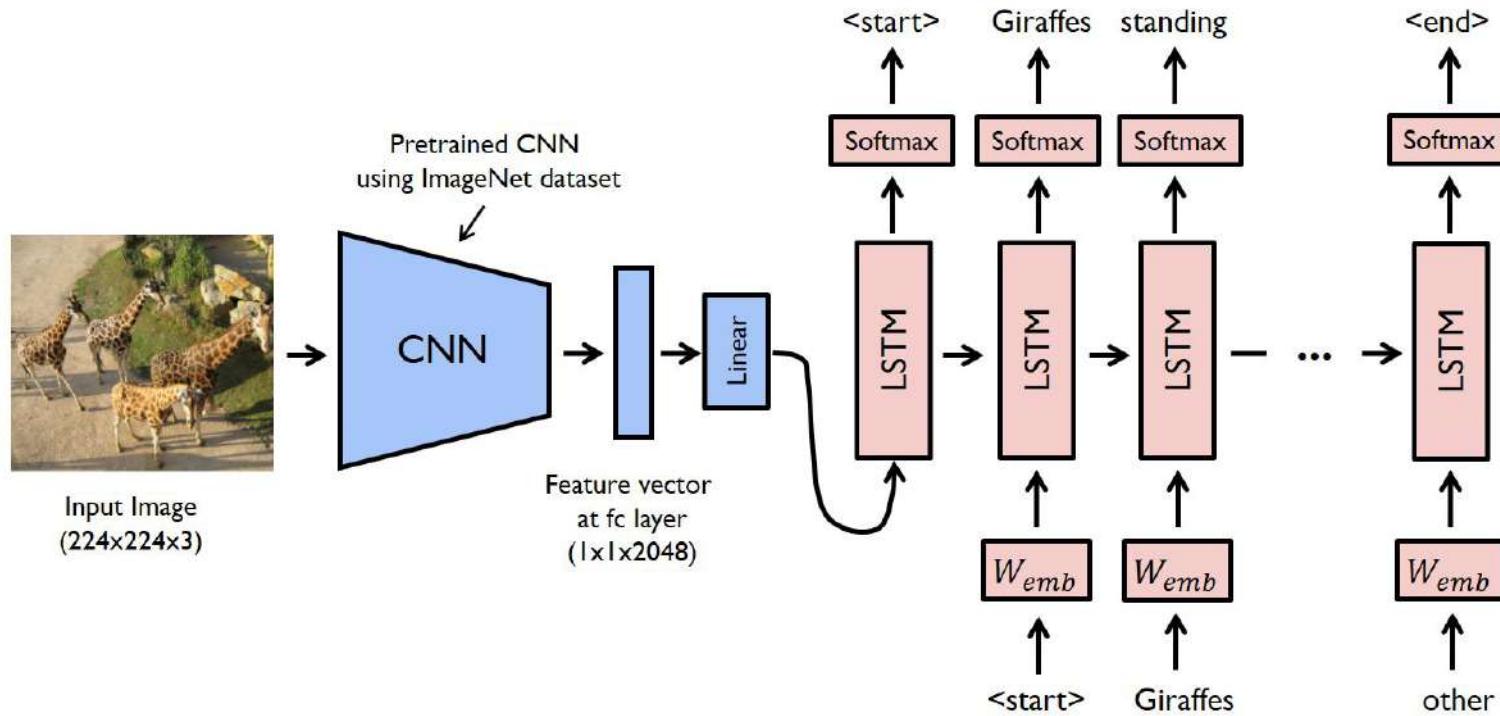
Language Tasks

- Parsing and tagging
- Language modelling
- Translation and unsupervised translation
- Coreference
- Reading comprehension
- Named entity
- What is the next sentence?

General Language Understanding Evaluation (GLUE) benchmark

- Is a sentence grammatically correct?
- Sentiment (positive/negative)
- Paraphrase: Are sentences semantically equivalent?
- Paraphrase: Are questions from Quora semantically equivalent?
- Paraphrase: news headlines, captions
- Inference: Does one sentence support a hypothesis, contradict it or is it neutral?
- Question answering: question-paragraph pairs
- What does a pronoun in sentence correspond to?
- ...

Image2Seq



<https://www.captionbot.ai/>

Visual question answering

VQA Challenge 2019

Who is wearing glasses?

man



woman



Where is the child sitting?

fridge



arms



Is the umbrella upside down?

yes



no



How many children are in the bed?

2



1



Umělá inteligence a strojové učení

Michal Hradiš: Reinforcement learning

All tutorial notebooks

Language Model:

<https://colab.research.google.com/drive/1oKLSpu4OMEuNSpub4Jw1Vxcbi5zpZmfU?usp=sharing>

CNN:

<https://colab.research.google.com/drive/1a8iZeo-pmrRDAHB8cK0TrEmaulMnaKNC#scrollTo=7k37ifwroNnN>

Basic NN:

<https://colab.research.google.com/drive/1ytNsahwdl4FJDkXmaw7CCQQJqQDiAxJt?usp=sharing>

AlphaGo Zero

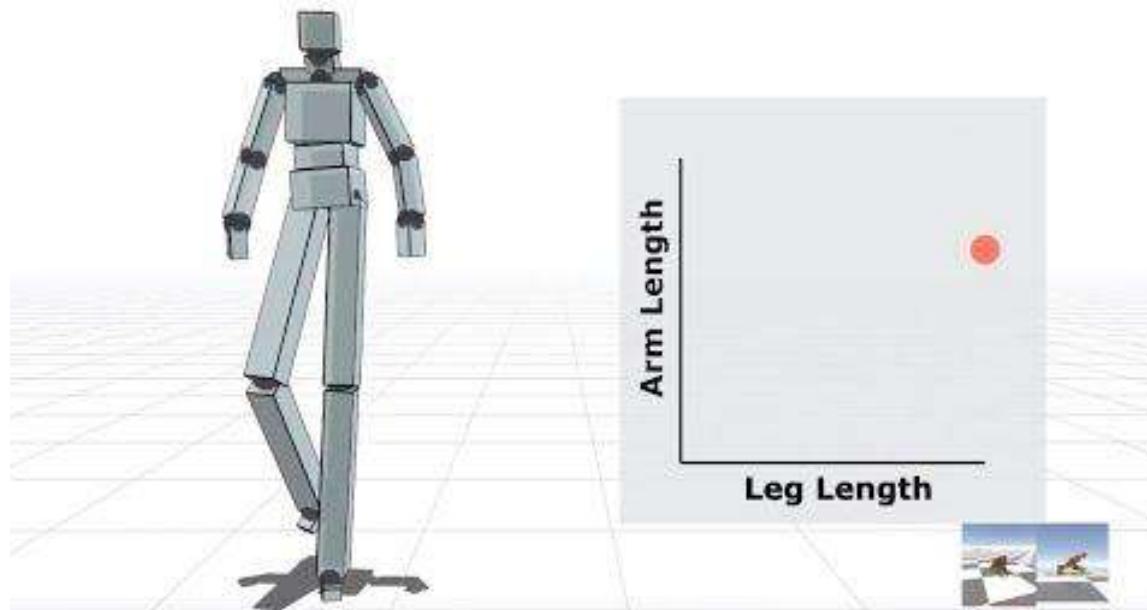
(<https://deepmind.com/blog/alphago-zero-learning-scratch/>)



Reinforcement learning AI (<https://github.com/deepmind/pysc2>)



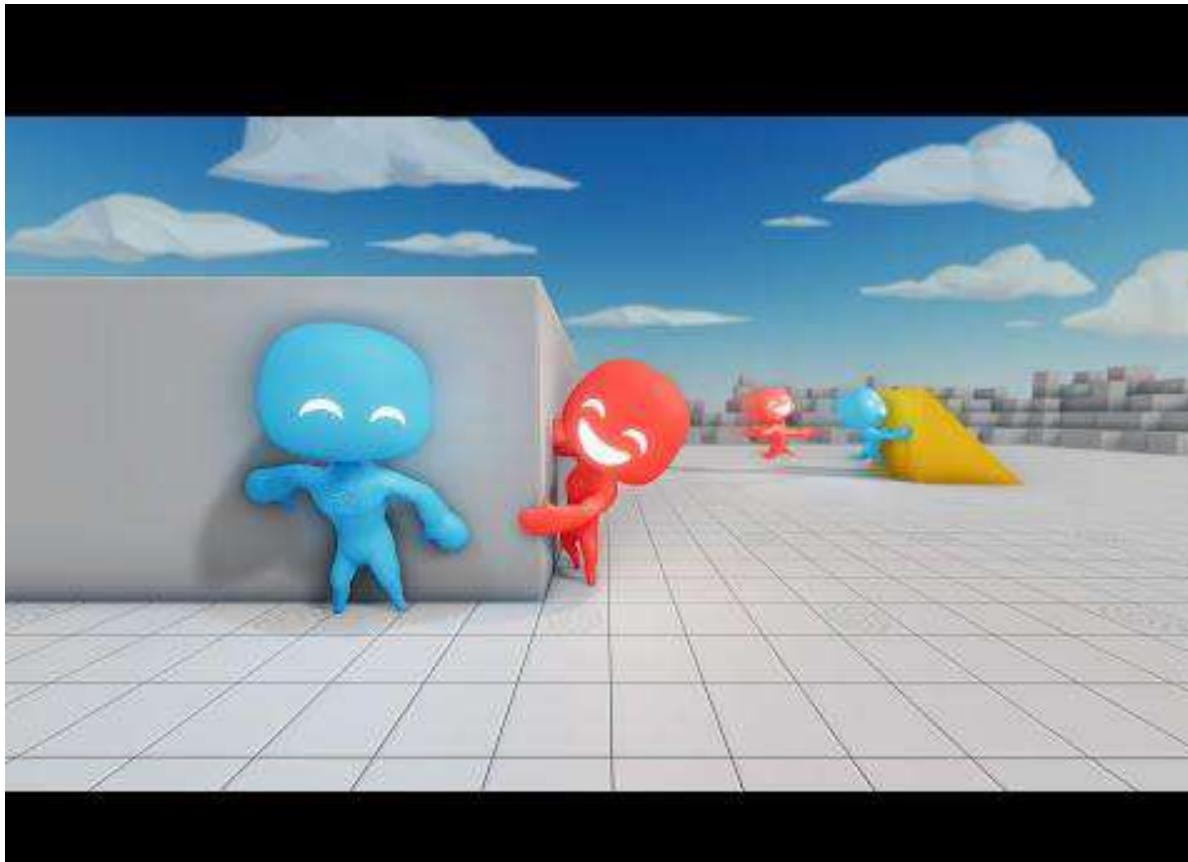
<https://www.youtube.com/watch?v=XrOTgZ14fJg>



NVIDIA AI
10 YEARS!



<https://www.youtube.com/watch?v=kopoLzvh5jY>



Reinforcement learning



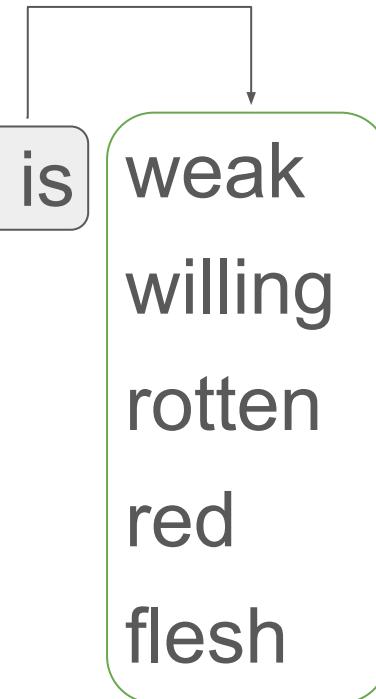
Reinforcement learning - simulation



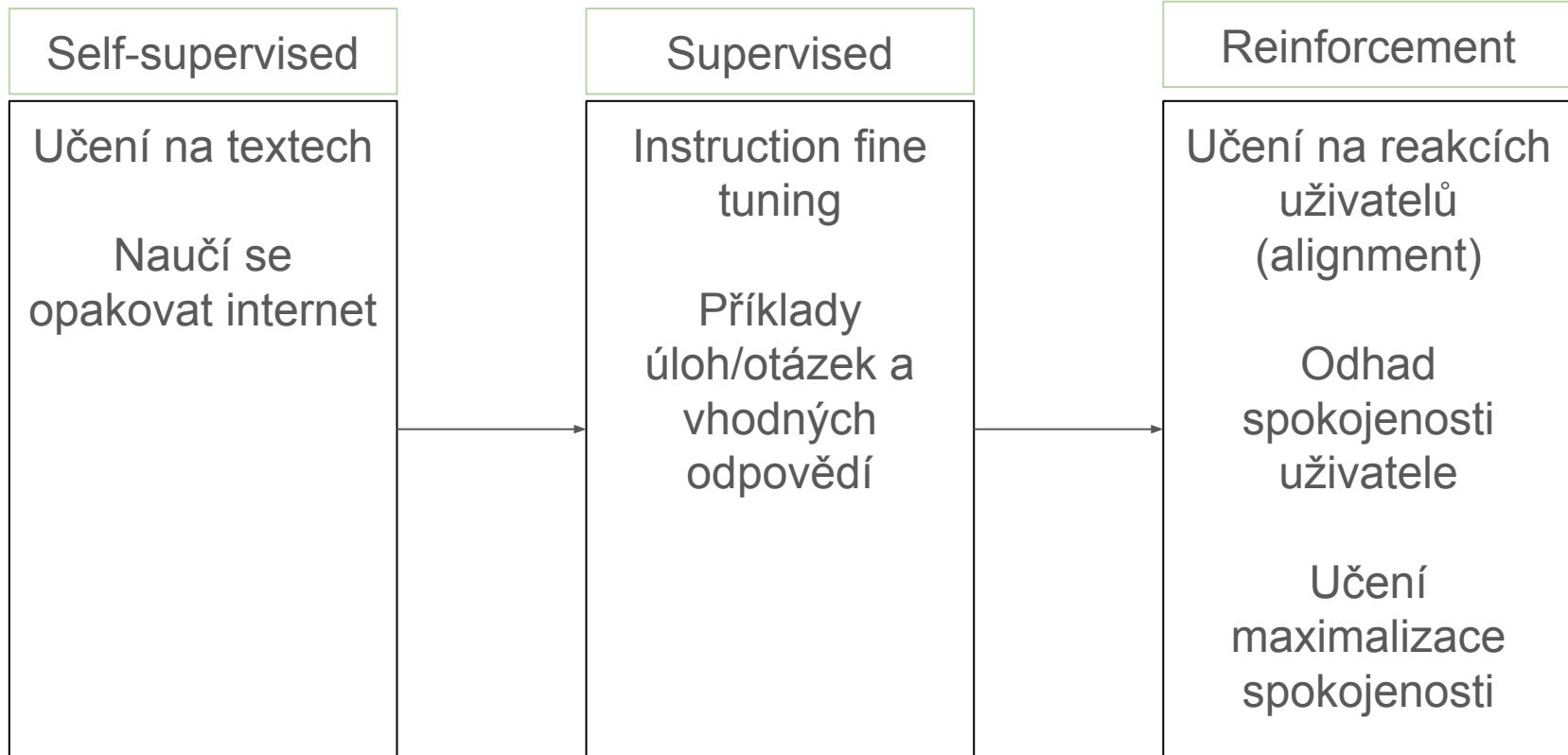
Self supervised learning

Supervised, but labels were created automatically

spirit is willing but the flesh is

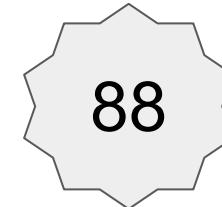


Jazykové modely

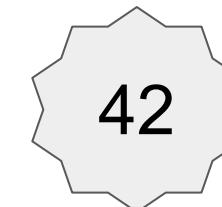


Reinforcement learning with LLM

Artificial Intelligence is the field of computer science that develops systems capable of performing tasks that typically require human intelligence.



AI is when machines learn and think like humans to solve problems or make decisions.



What is reinforcement learning?

A formalisation of the AI problem.

Science and framework of learning to make decisions from interaction.

How to achieve a goal (expressed as reward)?

We need to find a solution. Examples of perfect behavior are not provided (or known).

Learning similar to people by interacting with environment. Learning is **active** and the tasks are often **sequential**.

What is reinforcement learning?

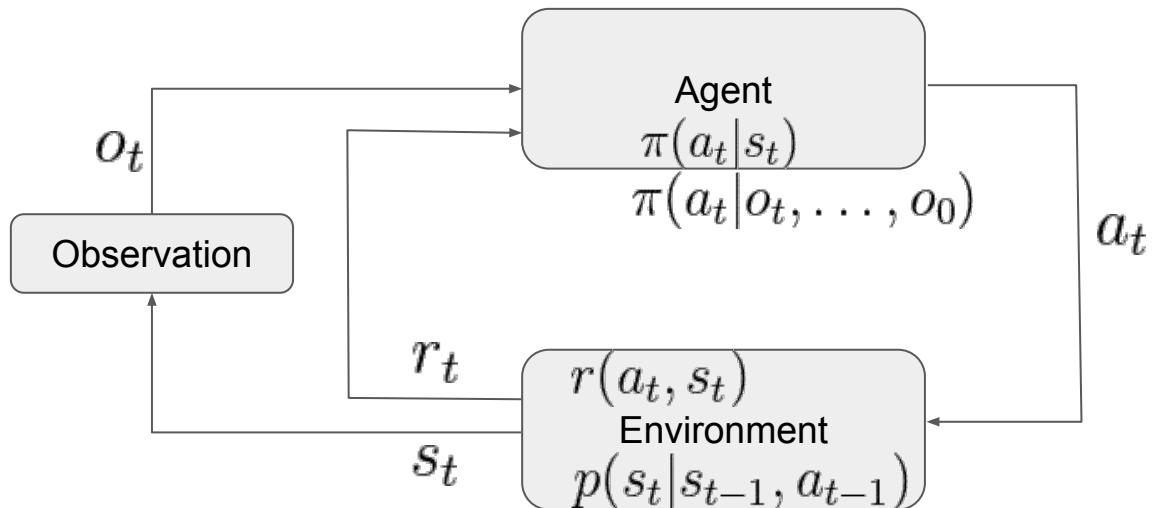
Relation to supervised learning with gradients?

All supervised problems can be expressed as RL.

Relation to genetic programming?

All RL problems can be solved with genetic programming :).

Reinforcement learning



Observation o_t

State s_t

Policy $\pi(a_t|s_t)$

Transition model

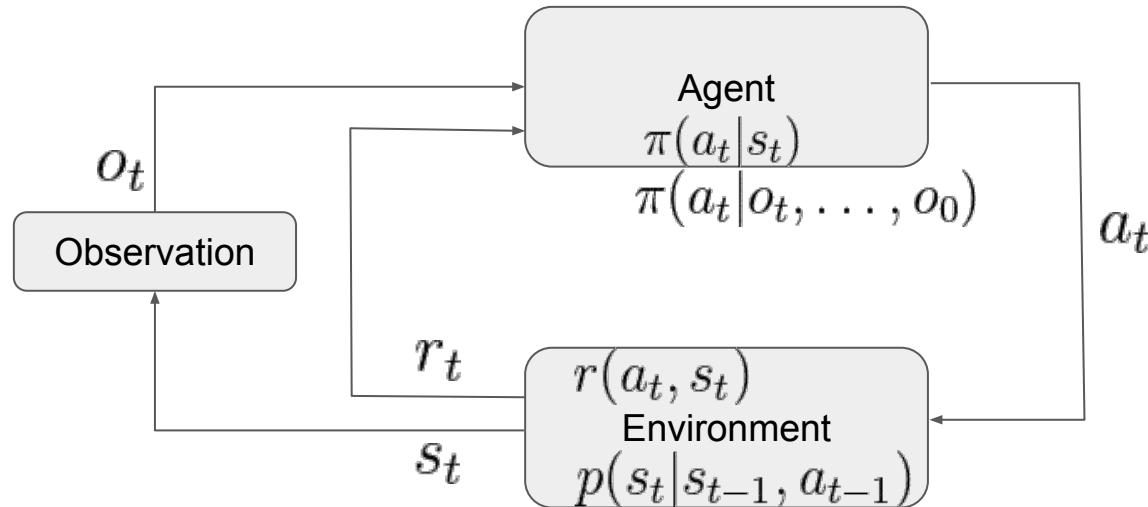
Reward r_t

Action a_t

Reward function $r(a_t, s_t)$



Reinforcement learning - transitions

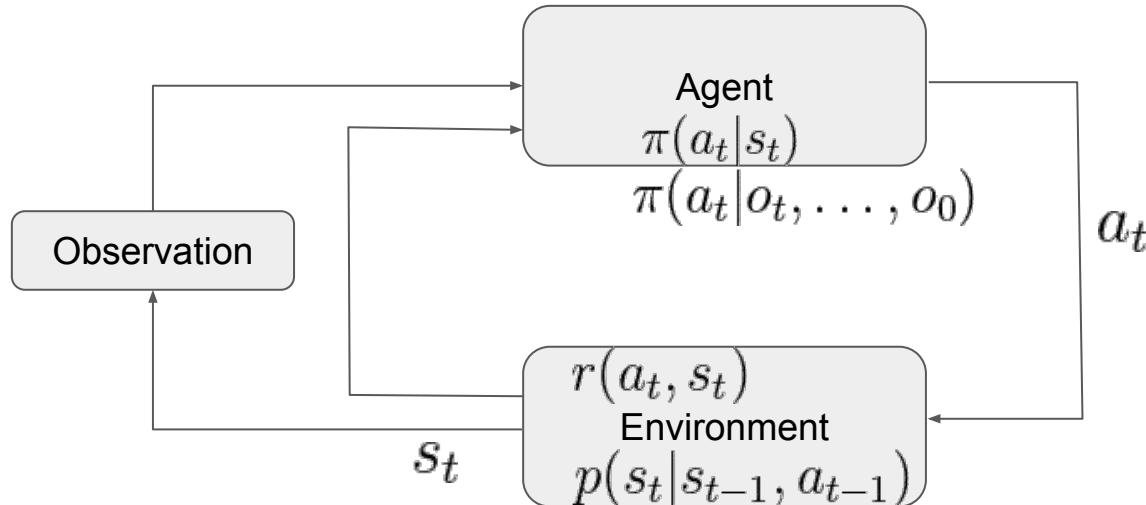


Trajectory

$$p_\theta(\mathbf{s}_1, \mathbf{a}_1, \dots, \mathbf{s}_T, \mathbf{a}_T) = p(\mathbf{s}_1) \prod_{t=1}^T \pi_\theta(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

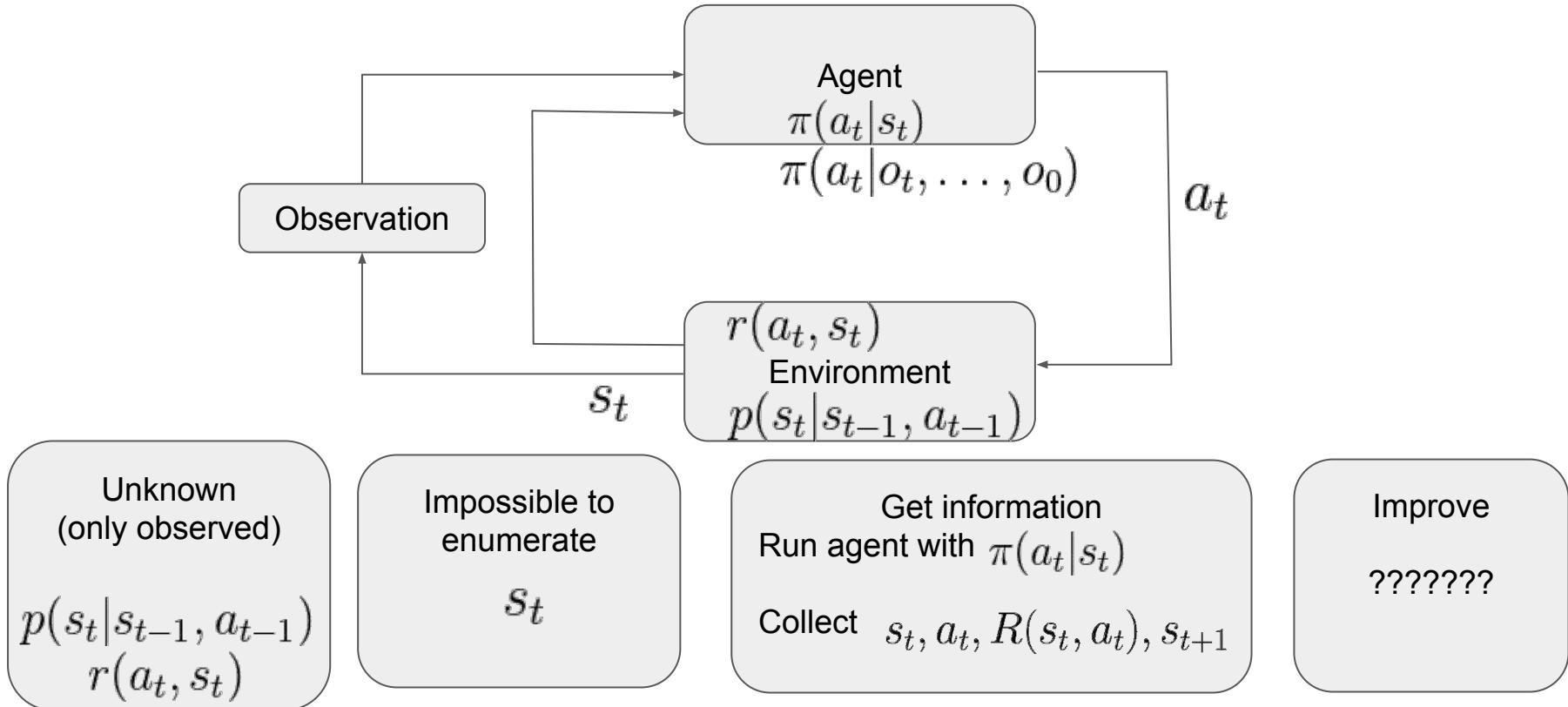
$p_\theta(\tau) \leftarrow$ Trajectory

Reinforcement learning - goal



$$p_\theta(\tau) \quad \theta^* = \arg \max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

The interesting problems in RL



What to learn?

Policy $\pi(a_t | s_t)$

$$V^\pi = E_\pi \sum_{t'=t}^T r(s_{\{t'\}}, a_{\{t'\}}) | s_t$$

Q-function $Q^\pi(s_t, a_t) = E_\pi \sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t$
set $\pi'(\mathbf{a}|s) = 1$ if $\mathbf{a} = \arg \max_{\mathbf{a}} Q^\pi(s, \mathbf{a})$

Value function $V^\pi(s_t) = E_\pi \sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t$

modify $\pi(\mathbf{a}|s)$ to increase probability of \mathbf{a} if $Q^\pi(s, \mathbf{a}) > V^\pi(s)$

Markov decision process

Mathematical framework for most RL

Main idea: Future depends only on current state, not on history.

Consequence: If the agent knows the environment state it does not need any memory.

How about chess? What would be the state?

Partial observability

Real problems are not Markov.

Agent has access to environment state only through observations.

Solutions:

Agent has internal state (memory)

Observation is “enriched” to better describe the environment state to get closer to “Markov process”. E.g. last few frames from a video.

Types of algorithms

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Policy gradients: Optimize directly policy / gradients of objective with resp. to policy

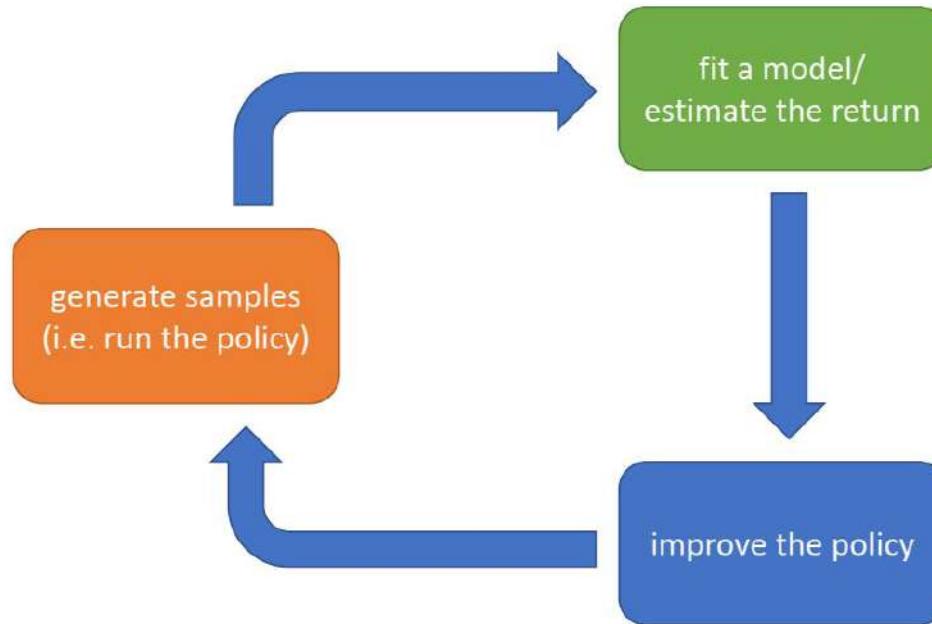
Value-based: estimate Q-function

Actor-critic: estimate V-function or Q-function and use it to improve policy

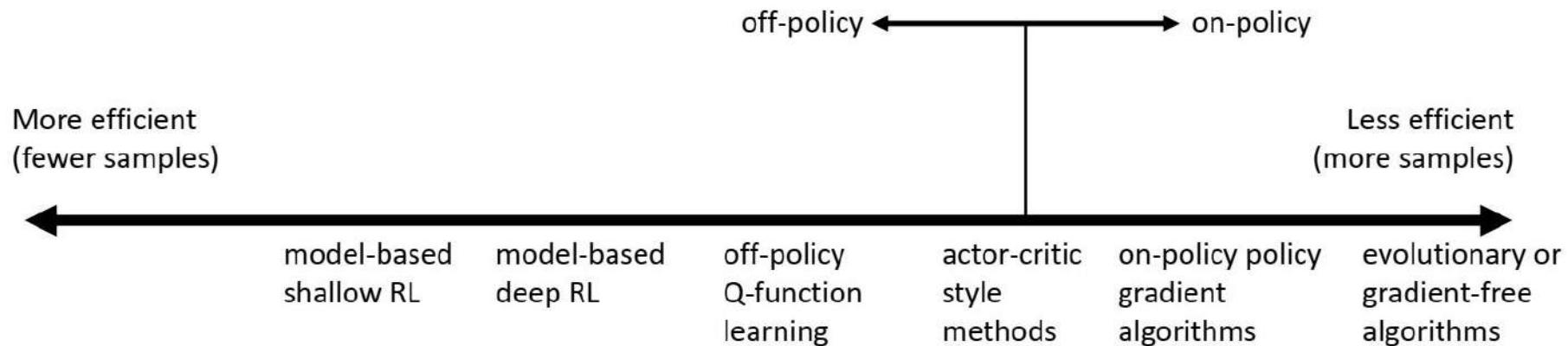
Model-based RL: estimate transition model

- Use it for planning (search)
- Improve policy
- ???

Learning stages



Sample efficiency



Q-learning

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

$$Q^{\pi}(s_t, a_t) = E_{\pi} \sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \quad \text{set } \pi'(\mathbf{a}|s) = 1 \text{ if } \mathbf{a} = \arg \max_{\mathbf{a}} Q^{\pi}(s, \mathbf{a})$$

From Bellman equation to loss function

$$Q^{\pi}(s, a) = r(s, a) + E_{s'} [\max(Q^{\pi}(s', a'))]$$

$$(Q^{\pi_{\theta}}(s, a) - \{r(s, a) + E_{s'} [\max(Q^{\pi_{\theta}}(s', a'))]\})^2$$

$$L(\theta) = \frac{1}{N} \sum_{i \in N} \left(Q^{\pi_{\theta}}(s_i, a_i) - [r(s_i, a_i) + E_{s'} [\max(Q^{\pi_{\theta}}(s', a'))]] \right)^2$$

Q-learning

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(s_t, a_t) \right]$$

$$Q^{\pi}(s_t, a_t) = E_{\pi} \sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \quad \text{set } \pi'(\mathbf{a}|s) = 1 \text{ if } \mathbf{a} = \arg \max_{\mathbf{a}} Q^{\pi}(s, \mathbf{a})$$

From Bellman equation to loss function

$$Q^{\pi}(s, a) = r(s, a) + E_{s'} [\max(Q^{\pi}(s', a'))]$$

$$L(\theta) = \boxed{E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - \{r(s, a) + \boxed{E_{s'} [\max(Q^{\pi_{\theta}}(s', a'))]}\})^2}$$

$$L(\theta) = E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - \{r(s, a) + \max(Q^{\pi_{\theta}}(s', a'))\})^2$$

$$L(\theta) = E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - r(s, a) + E_{s'} [\max(Q^{\pi_{\theta}}(s', a'))])^2$$

$$L(\theta) = E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - r(s, a) + \max(Q^{\pi_{\theta}}(s', a')))^2$$

Q-learning

$$\theta^* = \arg \max_{\theta} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$Q^{\pi}(s_t, a_t) = E_{\pi} \sum_{t'=t}^T r(s_{t'}, a_{t'}) | s_t, a_t \quad \text{set } \pi'(\mathbf{a}|\mathbf{s}) = 1 \text{ if } \mathbf{a} = \arg \max_{\mathbf{a}} Q^{\pi}(\mathbf{s}, \mathbf{a})$$

From Bellman equation to loss function

$$Q^{\pi}(s, a) = r(s, a) + E_{s'} [\max(Q^{\pi}(s', a'))]$$

$$L(\theta) = \boxed{E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - \{r(s, a) + \boxed{E_{s'} [\max(Q^{\pi_{\theta}}(s', a'))]}\})^2}$$

$$L(\theta) = E_{\pi_{\theta}}(Q^{\pi_{\theta}}(s, a) - \{r(s, a) + \max(Q^{\pi_{\theta}}(s', a'))\})^2$$

$$L(\theta) = \frac{1}{|D|} \sum_{i \in D} [Q^{\pi_{\theta}}(s_i, a_i) - \{r(s_i, a_i) + \max(Q^{\pi_{\theta}}(s'_i, a'_i))\}]^2 \quad D = \{(s_i, a_i, s'_i, r_i)\}$$

Q-learning

- Run current policy and record “Dataset” $D = \{(s_i, a_i, s'_i, r_i)\}$
- Run “any” optimization of Q-function network

$$L(\theta) = \frac{1}{|D|} \sum_{i \in D} [Q^{\pi_\theta}(s_i, a_i) - \{r(s_i, a_i) + \max(Q^{\pi_\theta}(s'_i, a'_i))\}]^2$$

Replay buffer/memory

Q-learning

- Run current policy and record “Dataset” $D = \{(s_i, a_i, s'_i, r_i)\}$
- Run “any” optimization of Q-function network

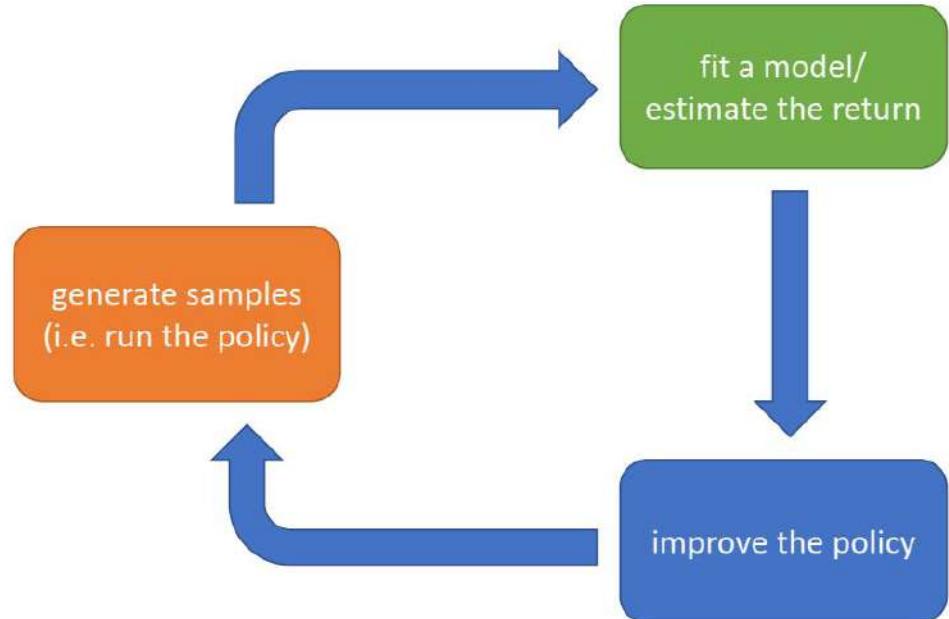
$$L(\theta) = \frac{1}{|D|} \sum_{i \in D} (Q^{\pi_\theta}(s_i, a_i) - \boxed{r(s_i, a_i) + \max(Q^{\pi_\theta}(s'_i, a')i)})^2$$

Replay buffer/memory

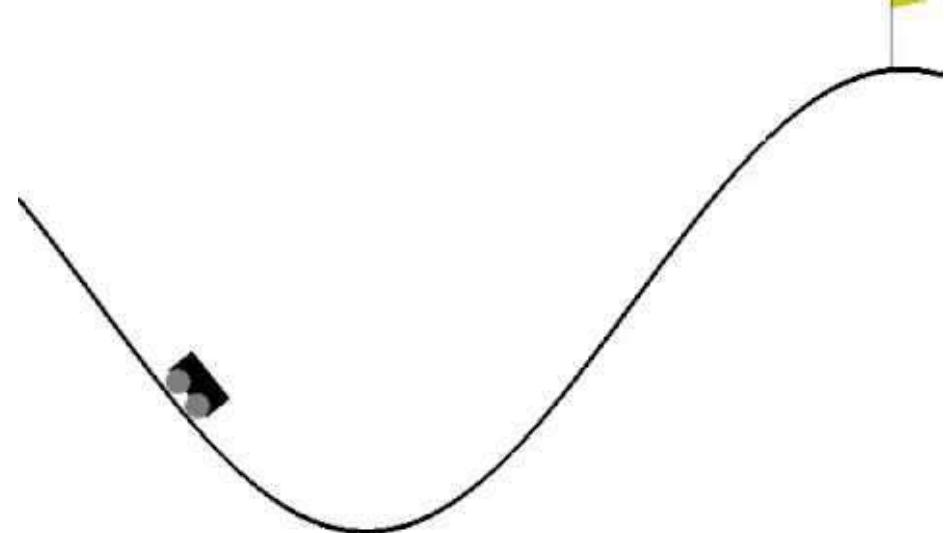
Static targets for a period of time

Exploration - exploitation

- Greedy
- Epsilon-greedy
- Sampling from policy
- Curiosity driven
- Example driven



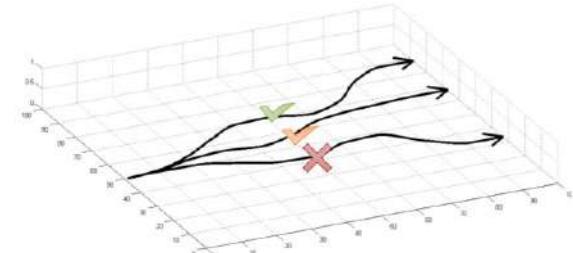
Problem of sparse rewards



Policy gradients

Stochastic policy ($\pi(a_t|s_t)$ is a distribution)

Increase probability of actions leading to good outcomes



$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$$

Policy gradients

The gradient is “weighted classification loss” -

Weights are collected rewards, classes are performed actions

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right).$$

maximum likelihood: $\nabla_{\theta} J_{\text{ML}}(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right)$

Policy gradients

Only future rewards as weights = better convergence

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_{i,t} | \mathbf{s}_{i,t}) \right) \left(\sum_{t=1}^T r(\mathbf{s}_{i,t}, \mathbf{a}_{i,t}) \right)$$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}) \right]$$

```
\nabla_{\theta} J(\theta)
\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[ \nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}) \right]
```

Actor critic - policy gradients + Q-learning

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T \left[\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) \sum_{t'=t}^T r(s_{i,t'}, a_{i,t'}) \right]$$



On average, this is

$$Q^{\pi}(s_{i,t}, a_{i,t})$$



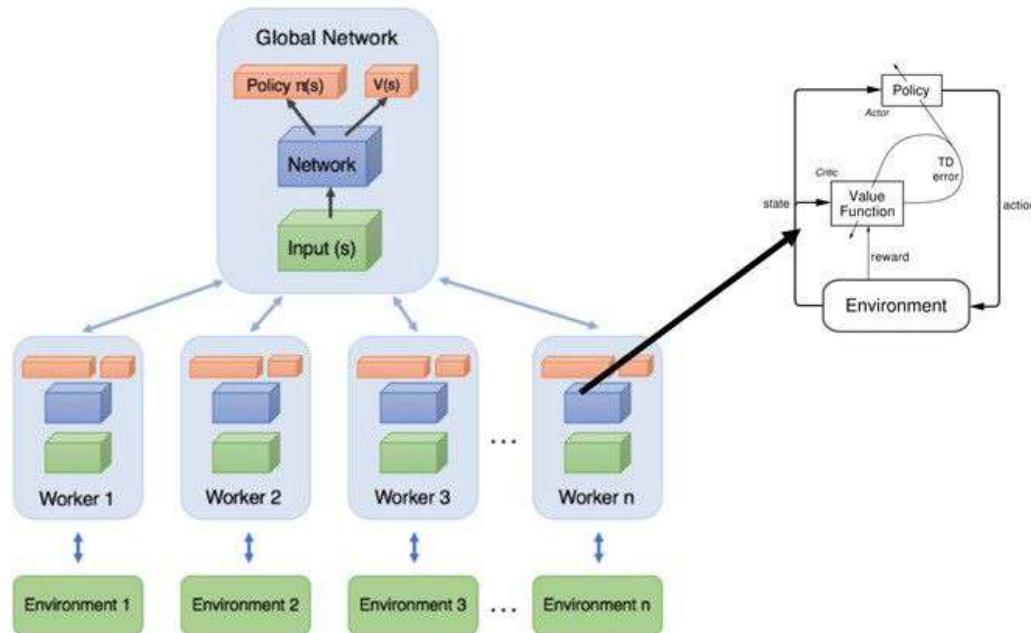
$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T [\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) Q_{\omega}(a_{i,t} | s_{i,t})]$$

Q-Actor-Critic

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T [\nabla_{\theta} \log \pi_{\theta}(a_{i,t} | s_{i,t}) Q_{\omega}(a_{i,t} | s_{i,t})]$$

- Collect experience using policy
- Update Q based on collected experience
- Update policy base on updated Q and collected experience

Parallel execution



Resources

CS234 - https://www.youtube.com/playlist?list=PLCH_MtKnU6rXoFJkxfUoeMzU9_CPwdjLq

DQN - pytorch https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Book (online): Reinforcement Learning: An Introduction, Sutton and Barto, 2018.

DeepMind Reinforcement Learning Lecture Series 2021,

<https://deepmind.com/learning-resources/reinforcement-learning-series-2021>

PPO - <https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html>

Stable Baselines 3 -

<https://stable-baselines3.readthedocs.io/en/master/guide/examples.html>

Prohledáváním k inteligenci

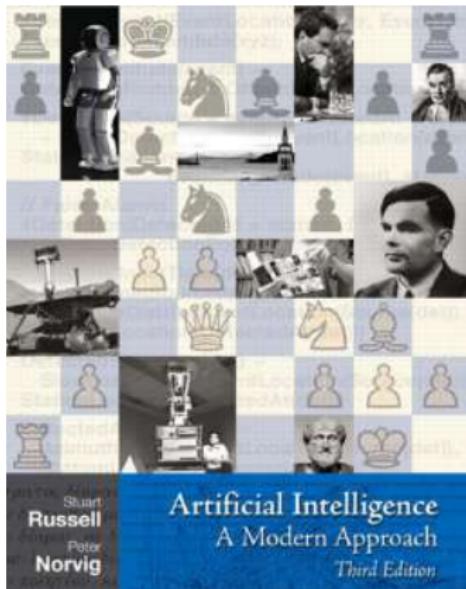
Karel Beneš

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole

ibenes@fit.vutbr.cz

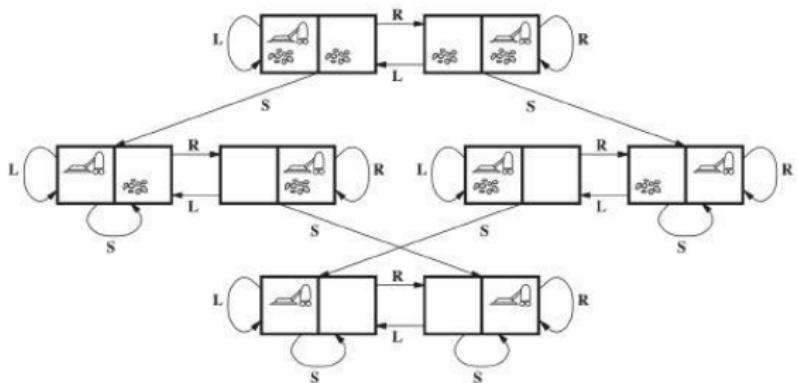


- Russel & Norvig
- M. Šustek

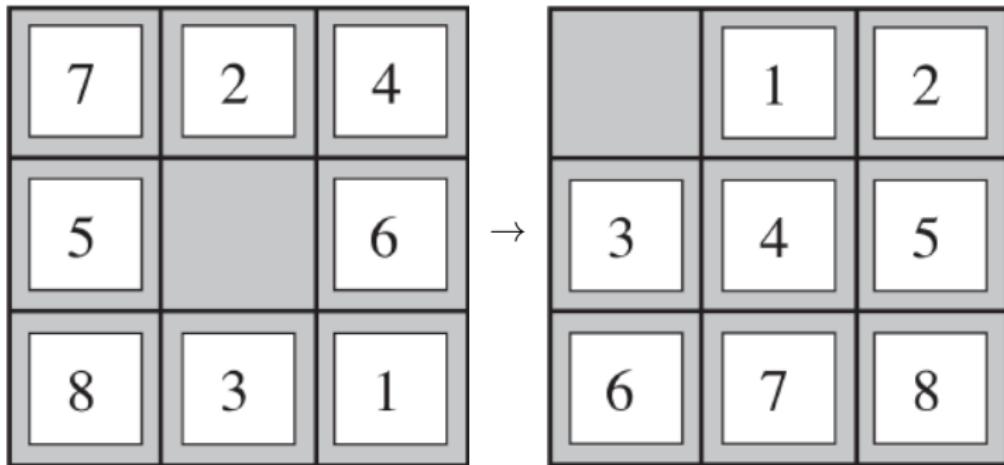


Co si přečíst?

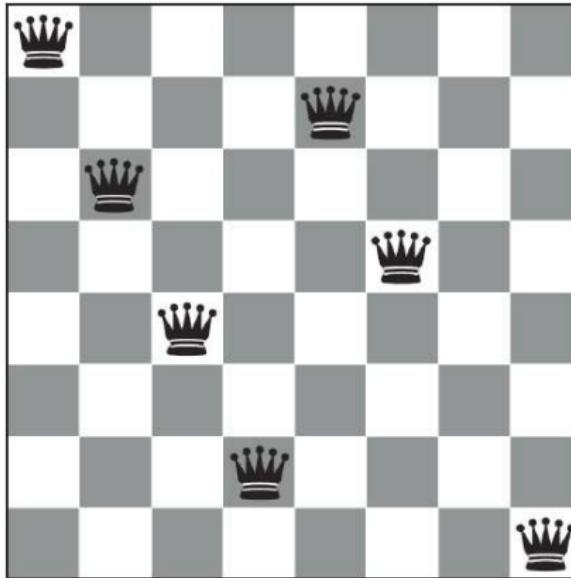
Kapitoly 3, 4 a 5



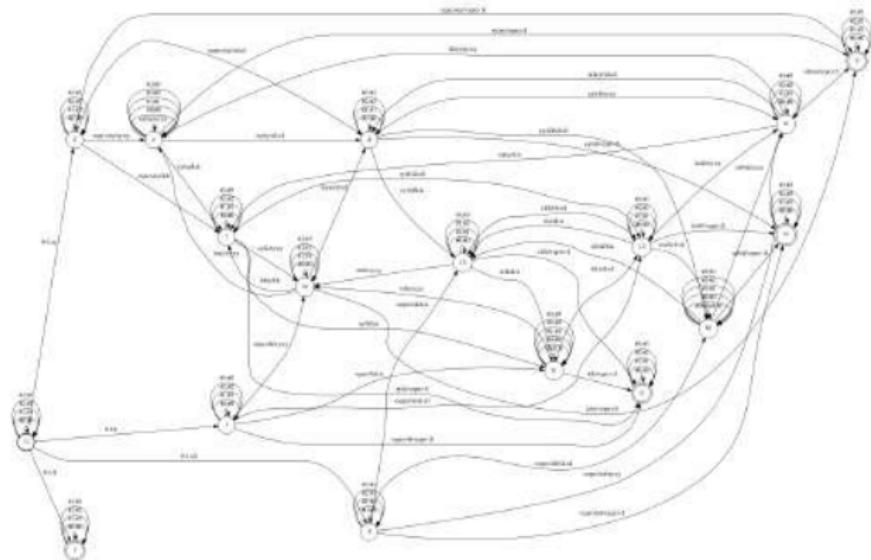
8 stavů, 3 akce



$9! \approx 363\text{ k}$ stavů, ~ 3 akce



$\binom{64}{8} \approx 4.4 \text{ B stavů}, \sim 60 \text{ akcí; nebo}$
 $8^8 = 2^{24} \approx 16 \text{ M stavů}, 8 \text{ akcí}$



Historie slov, fonémy, fonémy v kontextu, modely fonémů.
Ohodnocení v grafu (slovnik, jazyk) i mimo (audio). Více v SUR
a ZRE.

Miliony stavů každých 10 ms, ~ tisíce akcí => prořezávání nutné

- Počáteční stav s_0
- Množina akcí $\text{ACTIONS}(s) = \{a_1, a_2, \dots, a_n\}$
- Přechodový model $\text{RESULT}(s, a) = s'$
- Test na dosažení cíle $\text{GOAL?}(s)$
- + Cena akcí $\text{COST}(s, a) \rightarrow \mathbb{R}_0^+$

Vztah k reálnému světu?

- Perfektní model (šachy, umisťování krabic do kontejneru)
- Ruční approximace (pohyb prostorem → mřížka, zdi)
- Strojová approximace (křížovatka → pruhy, chodci, auta)

Srovnání s konečně stavovým automatem

Nemusíme znát stavy, nemusíme znát cíle, problém nemusí být konečný.

- posloupnost akcí a_1, a_2, \dots, a_t
- agent má jasný postup, kterého se může držet
- nezáleží na dalších vstupech (Agent může jít se zavřenýma očima. Máme správnou abstrakci, nejezdí v ulicích auta?)

Nezáleží na vstupech??

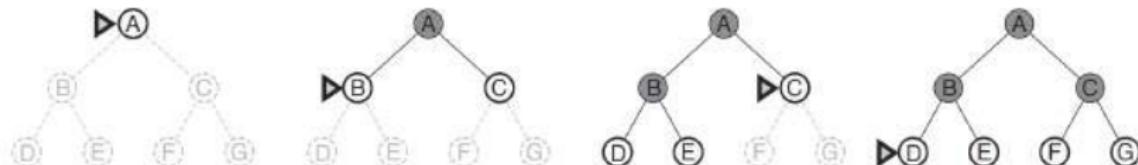
Předpokládáme známé, plně pozorovatelné a deterministické prostředí. Později některé podmínky uvolníme.

- Je **úplný**? Najde cíl, pokud existuje?
- Je **optimální**? Najde nejlevnější cíl?
- Jakou má paměťovou náročnost?
- Jakou má časovou náročnost?

Další abstrakce

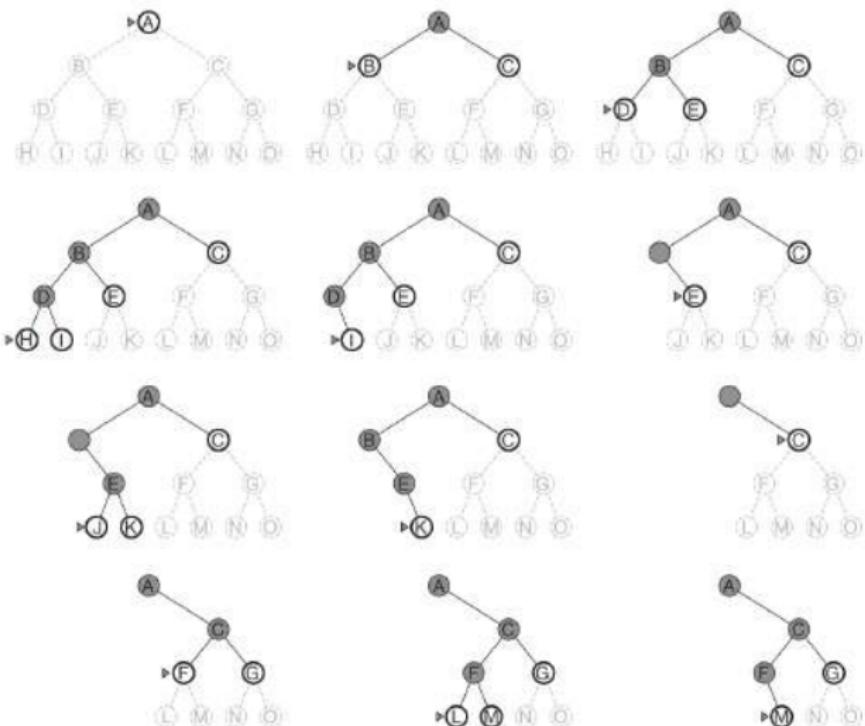
Optimální neřeší cenu výpočtu. Může být zásadní při hraní her, zvažujeme i při lokálním prohledávání.

- FIFO frontier („seznam open“)
=> katastrofické paměťové nároky $\approx \mathcal{O}(b^d)$
- úplný a optimální
- udržuje všechny prozkoumané uzly (explored, „seznam closed“), jinak riskuje cykly
- může testovat při objevení stavu



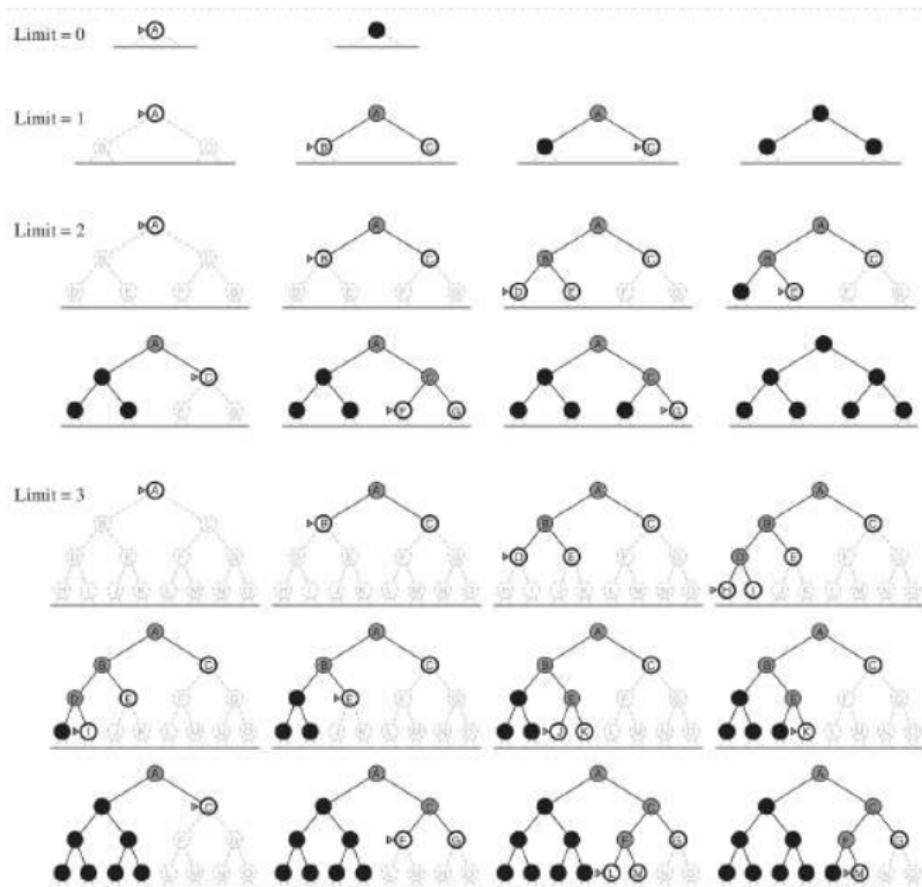
- LIFO frontier
 - řeší problém s pamětí $\approx \mathcal{O}(bd)$
 - ještě šetrnější je **backtracking** $\approx \mathcal{O}(d)$, který má akce seřazené (např. iterátor místo množiny)
- není optimální ($\text{ACTIONS}(s) = \{+1, \cdot 3\}$, $0 \Rightarrow 10$)
- není ani úplný ($\text{ACTIONS}(s) = \{+1, /2\}$, $16 \Rightarrow 2$)
 - v konečně stavových prostorech OK (nutně narazí na dno \Rightarrow nutně se odrazí \Rightarrow nutně prozkoumá další větev. RPT.)
 - lze omezit hloubku na D (**depth-limited search**, DLS), pak je úplný, pokud řešení v dané hloubce existuje. (Např. máme praktický odhad, kolik může být po cestě křížovatek.)

Depth-first search II

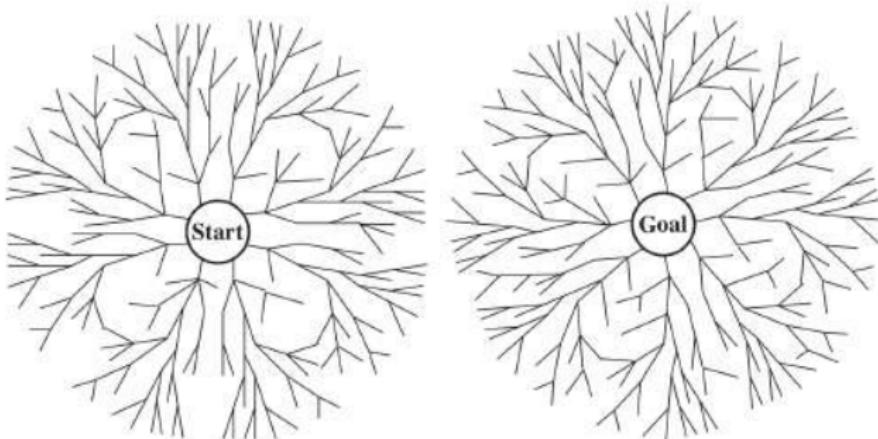


- DLS s iterativně zvětšovanou maximální hloubkou D
- optimální a úplný
- přátelský k paměti
- pomalejší než BFS
 - marginálně, pokud je b stálé (exponenciálně malý vliv malých hloubek)
 - výrazněji, pokud jsou v prostoru dlouhé lineární sekvence stavů ($b = 1$)
 - s ohledem k dostupné paměti lze napřed začít s BFS

Lze chápat jako pomalejší implementaci BFS.



- meta-metoda, $\mathcal{O}(b^d) \rightarrow 2 \cdot \mathcal{O}(b^{d/2})$
- nutné reverzibilní akce a známý (a/nebo vybraný) cílový stav
- vyžaduje smysluplný frontier alespoň na jedné straně



- Základní řešení pro různě drahé akce – předpokládá nezápornou cenu
- Frontier je **prioritní fronta**
- Úplný – podobný BFS, pro konstantní ceny stejný
- Optimální – uzly rozbaluje od nejlevnějších

Narozdíl od BFS

Nemůže testovat při objevení – kratší cesta může mít více uzelů.

V podstatě stejný jako Dijkstra

Ale stačí mu najít cíl.

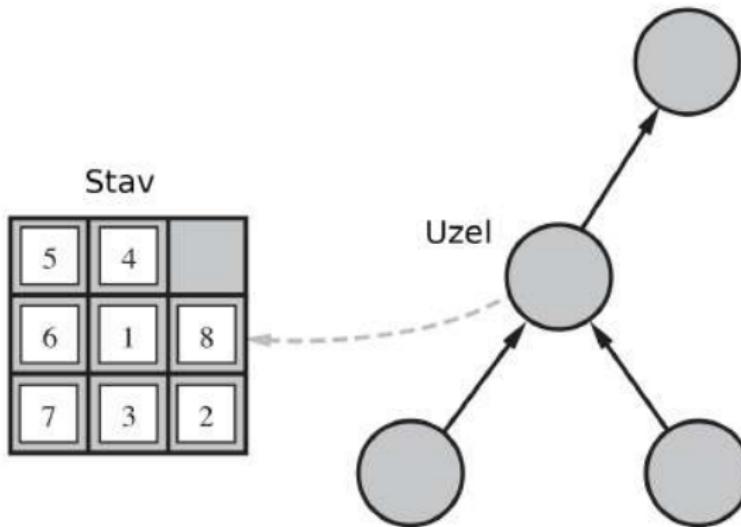
https://github.com/msustek/AI_course

- Dosud jsme uzly rozbalovali slepě
- Často máme nějaký odhad o tom „kterým směrem je řešení“
- Zavedeme funkci $f(n) \rightarrow \mathbb{R}^+$
 - Interpretace: za uzlem n je řešení v ceně $f(n)$
 - Rozklad: $f(n) = g(n) + h(n)$
 - Již zaplacená cena $g(n)$
 - Odhadovaná zbývající cena $h(n)$
- Rozbalujeme uzly s nejmenší $f(n)$

Uniform Cost Search

$$h(n) = 0$$

- stavy jsou ve světě
- uzly jsou v prohledávání
- hrany mezi uzly jsou akce
- řešení získáme sledováním rodičovských ukazatelů v uzlech



- Prohledávání vždy vytváří strom
- Uzly ale mohou odpovídat stejným stavům.
- Nechceme je řešit opakovaně
 - Musíme si je pamatovat (**explored**, closed list)
 - Musíme je prohledávat (hash-table s ukazateli na uzly?)
 - Musíme se rozhodnout, který uzel nechat

- $g(n) = 0$
- Hladová metoda, není optimální
- Snaží se rychle najít řešení
 - jaký je poměr ceny prohledávání a ceny provádění?
- Úplná v konečně stavových prostorech – pokud udržuje množinu prozkoumaných

Hladové prohledávání je počítačníkova reakce na Klam utopených nákladů.

- skutečná $g(n)$
- co nejpřesnější heuristika $h(n)$
- úplná a optimální metoda
 - pokud by byla heuristika přesná:
 - známe cenu optimálního řešení
 - jdeme přímo do cíle
 - požadujeme **přípustnou** heuristiku
 - optimistický odhad
 - jinak není optimální
 - v grafech **konzistentní** heuristiku:
 - **RESULT**($n, a = n' \Rightarrow h(n) - h(n') \leq c(n, a)$)
 - lokálně optimistický odhad, trojúhelníková nerovnost
 - jinak není optimální

- pořád je to varianta na prohledávání do šířky
- zmenšujeme efektivní šířku b^*
 - pokud by byla heuristika přesná: $b^* = 1$
 - UCS by nám dalo přesnou heuristiku – ale za nepřijatelnou cenu
- existují varianty na snížení paměťové náročnosti:
 - IDA*: jako IDS, ale odřezáváme podle $f(n)$ místo hloubky
 - RBFS: limit na $f(n)$ – nejlepší ne-předek, aktualizujeme při vynořování
 - (S)MA: až dojde paměť zahazuj nejhorší listy, ale ukládej jejich f -hodnotu do rodiče

- znalost problému
- cena v jednodušších problémech, např. u Loydovy osmičky:
 - počet nesprávně umístěných (h_1)
 - součet L_1 vzdáleností (h_2)
- = cena nevyhnutelných kroků

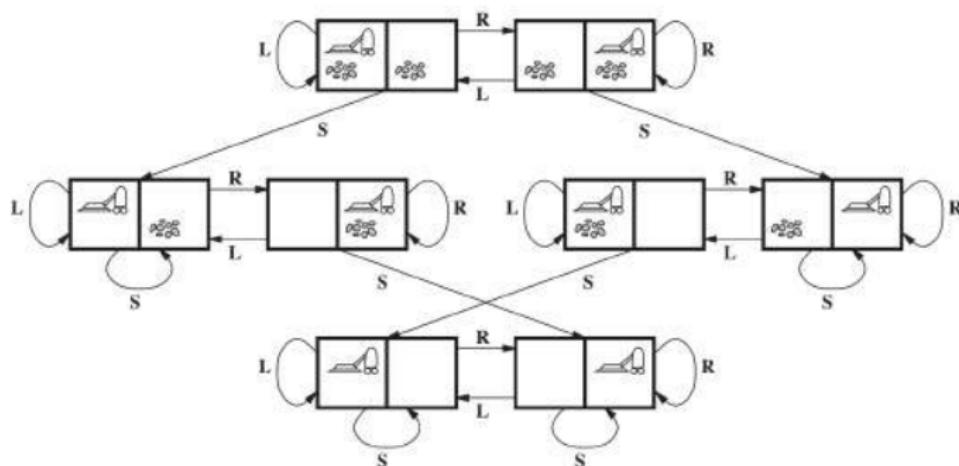
Praktický dopad při zachování třídy složitosti

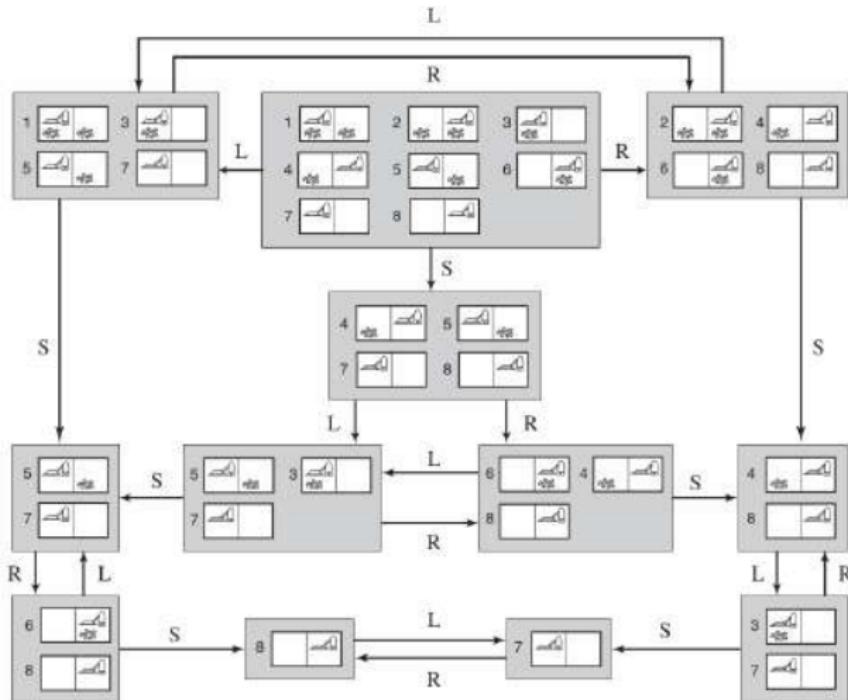
d	Search Cost (nodes generated)			Effective Branching Factor		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	—	539	113	—	1.44	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

- větší heuristika $h(n) \Rightarrow$ menší chyba \Rightarrow lepší heuristika
- $\forall n : h_1(n) \geq h_2(n) \stackrel{\text{def}}{=} \text{heuristika } h_1 \text{ dominuje heuristiku } h_2$
- heuristiky lze komponovat:
$$h(n) = \max(h_1(n), h_2(n), \dots, h_m(n))$$
- heuristiku se lze učit – zvlášť vhodné pro ne-atomické stavy, ale většinou s rizikem nepřípustnosti

- Dosud jsme předpokládali prostředí, které je
 - známé
 - plně pozorovatelné a
 - deterministické
- což není vždy ten případ

Extrém: žádné pozorování. Můžeme řešit problém?



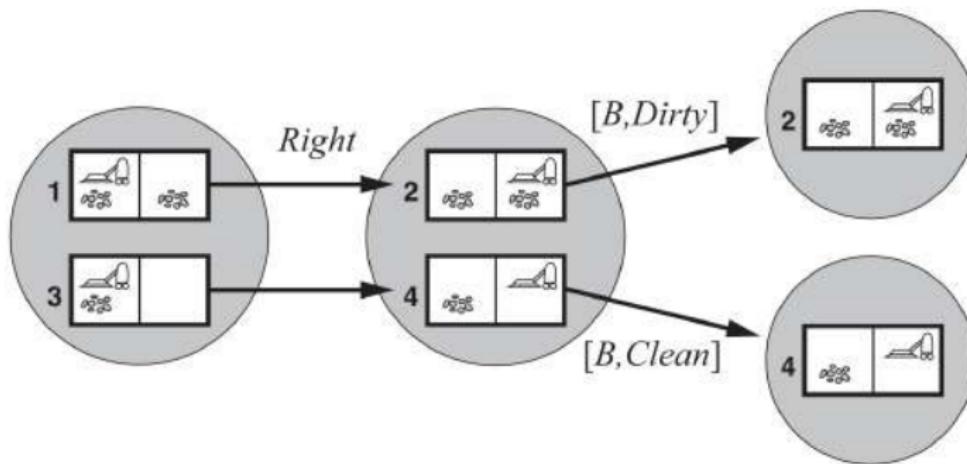


Další podobnost s teoretickou informatikou

Determinizace FSA – belief state nezkonstruujeme pro nekonečné nebo neznámé prostředí.

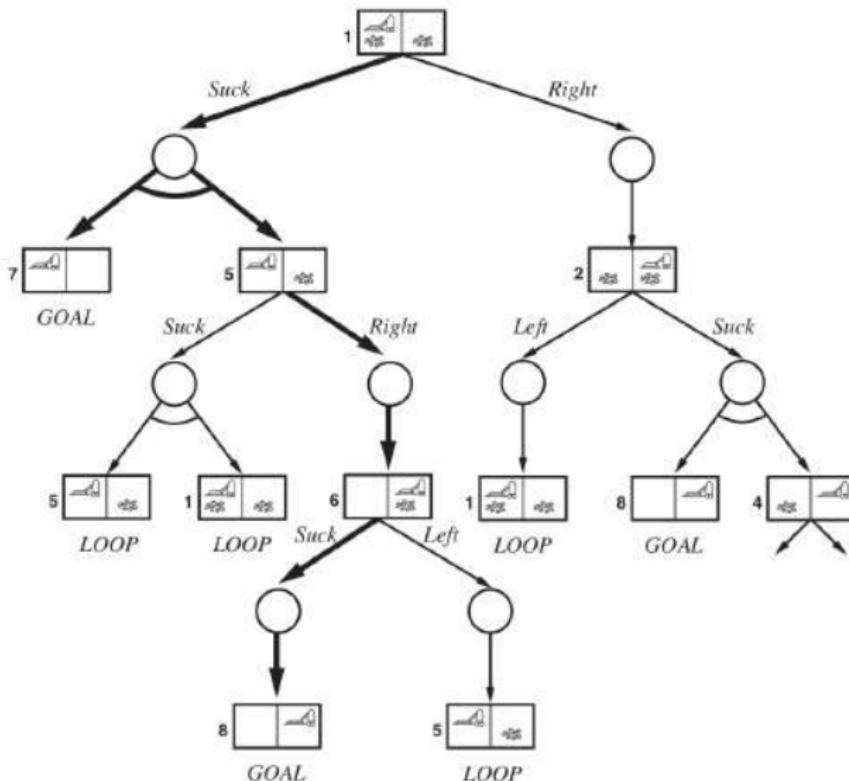
- Belief state-ů je exponenčně mnoho (je to poset stavů)
 - těžko je reprezentovat explicitně
 - ale větvení nemusí být horší než v prostoru stavů
- Lze řešit iterativně (Incremental belief-state search):
 - nalezne se řešení pro jeden z možných stavů
 - ověří se, zda vyhovuje pro další možné stavy
 - jinak se nalezne další řešení
 - ověření řešení je $\mathcal{O}(d)$ => nemusí být pomalé
- Belief state-y nejsou atomické

- Typicky nějaké vstupy máme
- Agent si musí udržovat B-state, je to kompaktní reprezentace sekvence vstupů



Pořádné zpracování jde založit na [Bayesovském](#) přístupu k pravděpodobnosti (BAYa).

- $\text{RESULT}(s, a)$ již není stav s' ale množina $\{s_1, s_2, \dots, s_n\}$
- Když agent nezná výsledek dopředu, musí se na něj podívat
 - jinak by se musel vypořádat s neúplným pozorováním
- Řešením už není sekvence akcí a_1, a_2, \dots, a_d , ale **plán** ve formě stromu if-then.



- Proč AND-OR? Každý uzel říká, kdy je řešitelný
 - pokud je řešitelný jeden jeho potomek (OR)
 - pokud jsou řešitelní všichni jeho potomci (AND)
- Co když jsou řešitelní jen někteří potomci AND-uzlu?
 - Zapamatujeme si je!
 - => dynamické programování
- Co když narazíme na cyklus?
 - Není řešením...
 - ... ale můžeme ho ignorovat – důležité je, jestli najdeme řešení v jiné větvi

Dynamické programování

Je celkem populární v coding interviews. A uplatní se taky při trénování HMM (SUR, ZRE) nebo NN-CTC (KNN).

AND-OR stromy III: try, try again

Fifinas dm v 0:33
Btw pri pohovorech jsem si vzpomnел na @dedObed a jeho zmínku o dynamickém programování, na ktore sa ptají na pohovorech firmy, ktoré hodne plati 🍒

Fifinas dm v 0:34
Btw pri pohovorech jsem si vzpomněl na @dedObed a jeho zmínku o dynamickém programování, na ktore sa ptají na pohovorech firmy, ktoré hodne plati 🍒

Omg, já jsem byl i k něčemu praktickému užitečný? 🎉

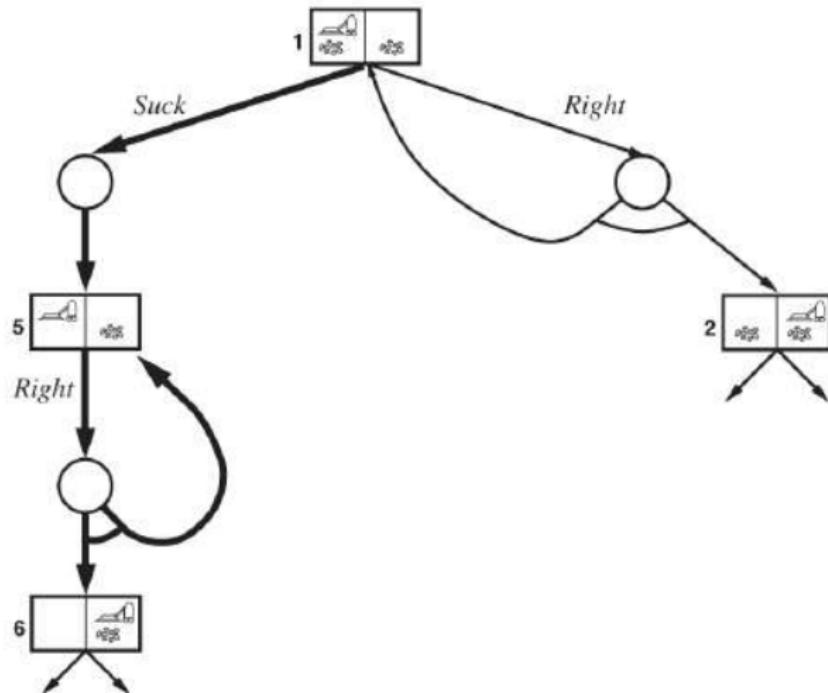
2 3 2

Tak z toho s dovolením udělam screenshot a příští rok půjde do slajdu 🍒

3 2 2

Tak z toho s dovolením udělam screenshot a příští rok půjde do slajdu 🍒

dm v 0:28
Put me in the screencap.



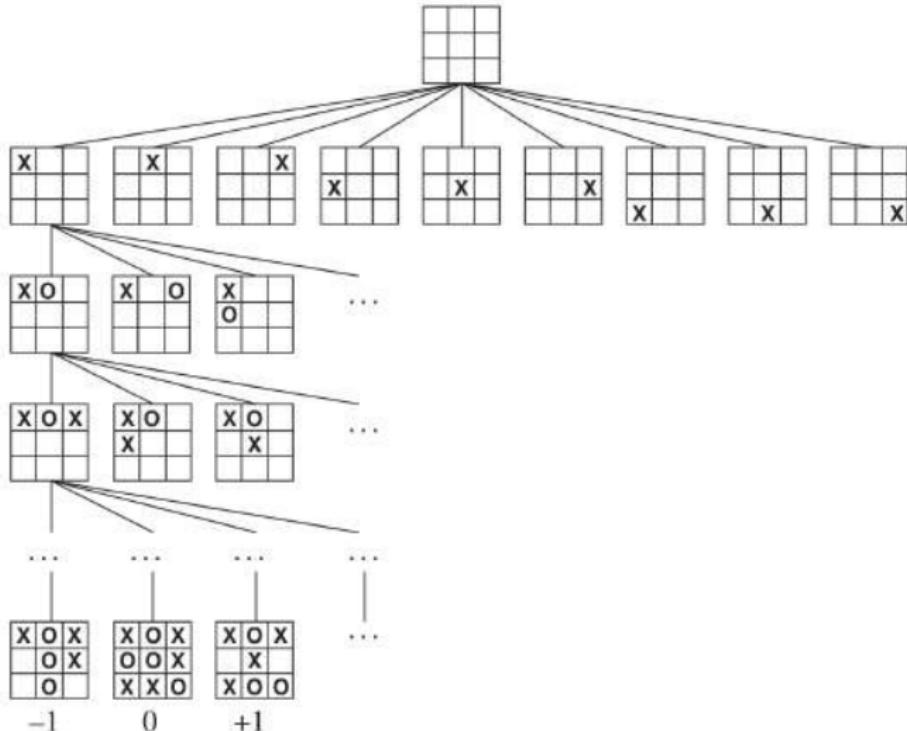
Cykly v reálném světě

Co když vám padne orel 20x po sobě? 8x se špatně naformátuje Facebook? 6x nedáte zkoušku z TiNu?

- Prohledávání:
 - dá agentovi plán
 - do šířky stojí hodně paměti
 - do hloubky obětuje optimálnost a riskuje cykly
 - hodně těží z heuristik
- Pokud agent nevidí vše
 - musí si udržovat povědomí o všech možnostech (belief state)
 - prohledává jiný, neatomický, prostor (belief state space)
- Pokud agent neví, jak akce dopadne
 - musí se umět vypořádat se všemi výsledky
 - má nelineární plán
 - měl by umět dynamické programování

Neznámé prostředí

Pokud agent neví, jak svět funguje, nemůže si představit, co se stane. Musí se uchýlit k experimentování (online prohledávání a učení, vědecká metoda).



Nelíbí se Vám body ze zkoušky v SUI?

- Napsat e-mail a reklamovat konkrétní otázku: (-1, +5)
- Domluvit si schůzku s vyučujícím: (-50, +50)
- Přijít bez ohlášení, když vím, že má málo času: (-5, +15)

Jak zahrát, pokud Vám nejde o známku, ale maminka Vám slíbí za každý bod tisíc korun / 40.50 eur?

- Myšlenkové rozšíření prohledávání s nejistotou
- „Nejistota“ má vlastní plány
 - sleduje vlastní cíl (užitkovou funkci, cíle)
 - reaguje na naše akce
- Jak máme prohledávat AND uzel?

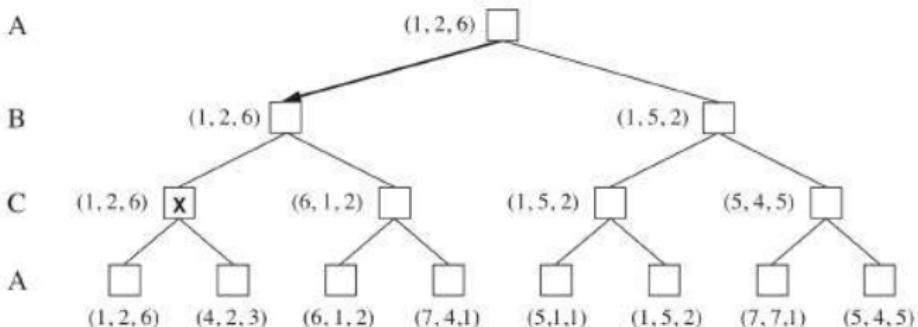
Co když nejistota úmysl přeci jen nemá?

Mluvíme o hře proti přírodě.

- Tahové (sekvenční, srov. v normální formě)
- S nulovým součtem
 - tzn. hráči se o něj musí podělit
- Dva hráči, deterministická hra, známá pravidla, plně pozorovatelná, ...

Za hranice SUI: THE

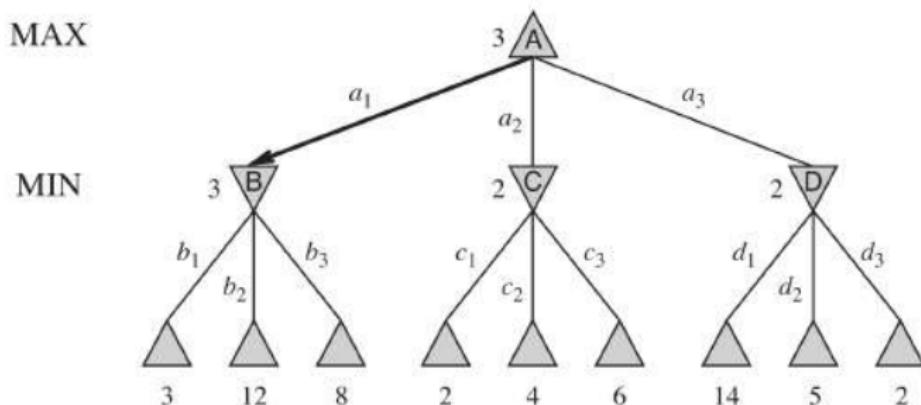
Hry v normální formě jsou úplně jiný svět. I deterministické hry mohou mít stochastická řešení.



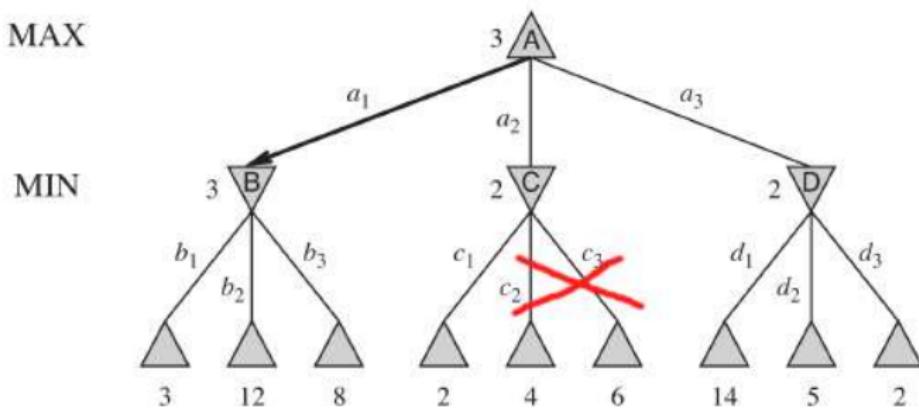
5 různověkých pirátů, 100 mincí

Nejstarší pirát udělá návrh rozdělení mincí, následně všichni hlasují. Schválený návrh je realizován, neschválený vede k zabití nejstaršího piráta a opakování postupu. Pirát chce (1) přežít, (2) vydělat a (3) uškodit ostatním. Jak sehra odehraje?

- Racionální hráč maximalizuje svůj užitek
- Dva hráči, zero-sum => hráčův užitek je roven soupeřově ztrátě
- Místo užitku jednotlivých hráčů lze modelovat jednu hodnotu => minimax
- Prohledáváme do hloubky / bottom-up?



- Někdy víme, že nemá cenu uzel dál expandovat
- Pořád si musíme sáhnout na dno
- $d_3 \leftrightarrow d_1$?
 - Šachy: brát, ohrozit, postoupit vpřed
 - Lze se bez obav naučit nebo iterativně prohlubovat
 - Myslete na časovou náročnost



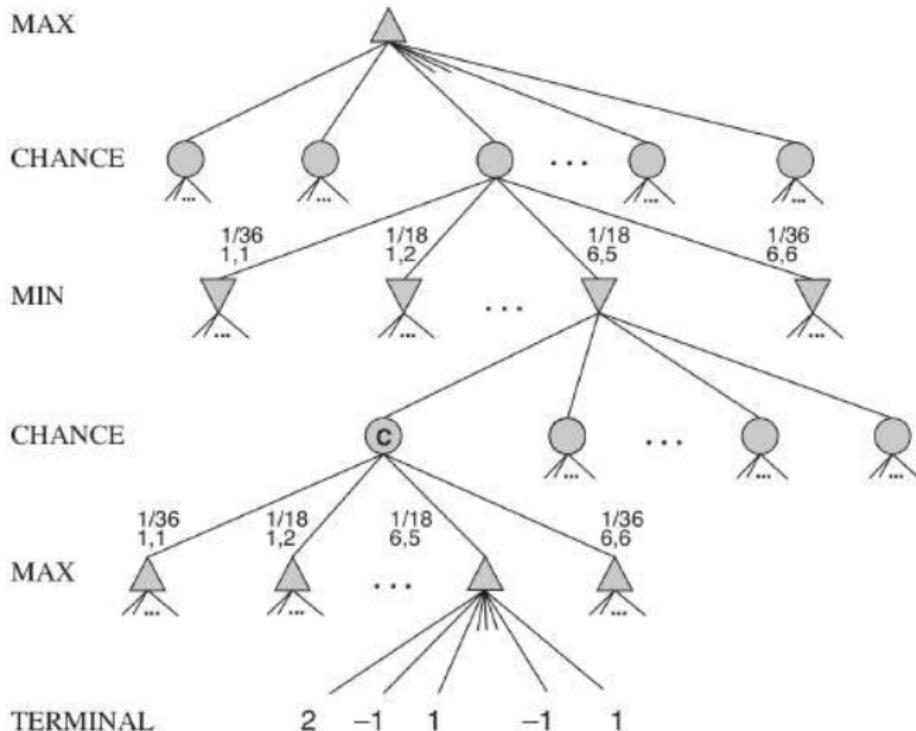
- Cut-off – omezení hloubky zanořování
 - na fixní hloubku nebo cut-off test (lze učit)
 - vyžaduje heuristické ohodnocení listů – mělo by korespondovat s pravděpodobností výhry
 - pozor na prokrastinaci (aplikujte nevyhnutelné tahy)
- Forward-pruning
 - vyžaduje dobrou heuristiku
 - odpovídá beam searchi

Null-move heuristic

Pokud lze předpokládat, že máme k dispozici tah, který nám prospěje, můžeme se vzdát tahu, nechat hrát soupeře (příp. simulovat několik následujících tahů) a získat tak poučený odřezávací práh.

Beam search

Každému uzlu je povoleno jen několik málo potomků. Vede k diverzifikaci zkoumaných řešení.



- Stejně jako u prohledávání mimo hry musíme brát v potaz všechny možnosti
- Hodnotu uzlu CHANCE modelujeme jako očekávání jeho potomků
- Přesně: $\mathbb{E}[U] = \sum_o U(o) \cdot P(o)$
- Monte Carlo: $\mathbb{E}[U] \approx \frac{1}{K} \sum_{i=1 \dots K} U(o_i), o_i \sim P(o)$
 - Můžeme jít plně Monte-Carlo a neprovádět žádné další prohledávání – pokud máme dobrou heuristiku

Averze k riziku

Předpokládáme, že hráč je k riziku neutrální. Více k modelování vztahu k riziku a reálným lidem v THE.

- Hraní her je prohledávání s nejistotou
- Nemáme binární cíl, ale maximalizujeme užitek
- Prořezávat lze optimálně k užitku (alfa-beta) nebo výpočtu (forward pruning)
- Doménové znalosti se hodí
- Velmi často je nutná dobrá listová heuristika, obzvláště u her s náhodou (příliš velký prostor k prohledání)

Hry s neúplnou informací

Dávají prostor pro blafování – belief state je dán pozorováními nejen přímo, ale i nepřímo. Předpoklad, že soupeř hraje racionálně, nestačí.

Úlohy s omezeními & agenti

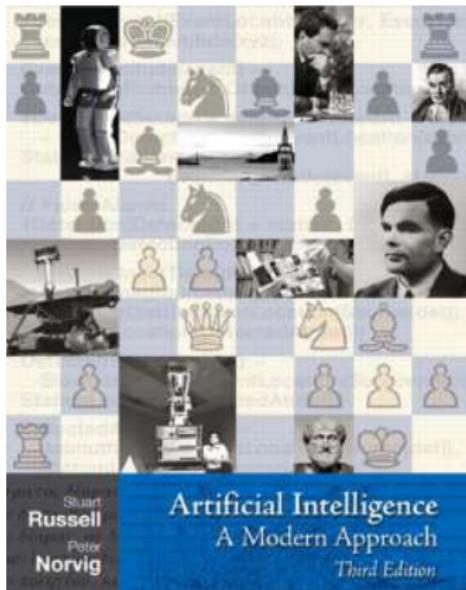
Karel Beneš

Fakulta informačních technologií Vysokého učení technického v Brně
Božetěchova 1/2. 612 66 Brno - Královo Pole

ibenes@fit.vutbr.cz



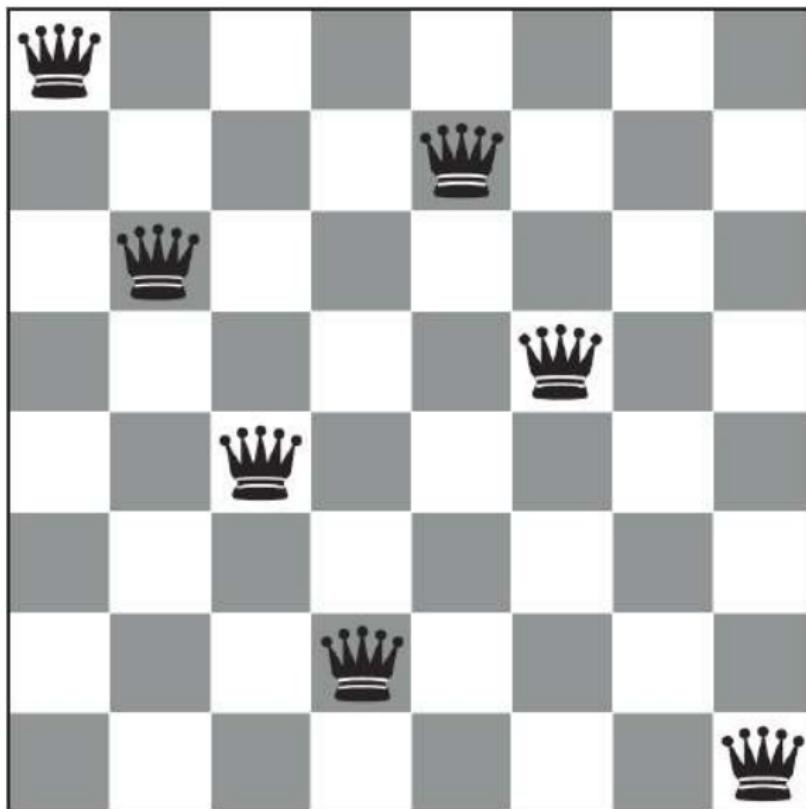
- Russel & Norvig
- M. Šustek

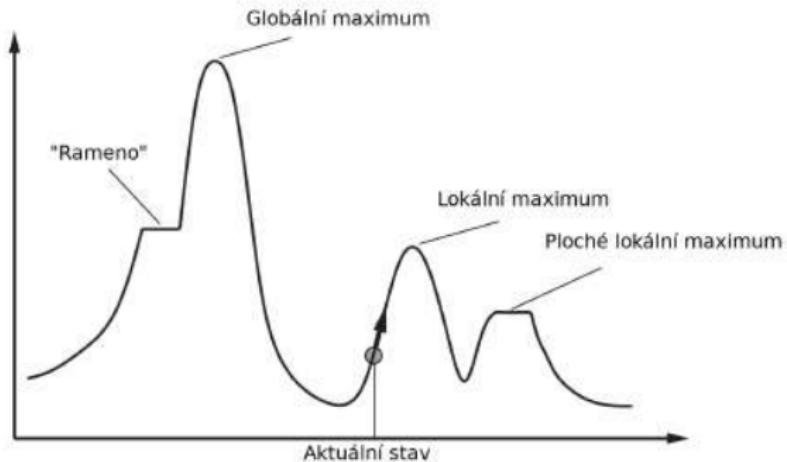


Co si přečíst?

Kapitoly 6 a 1 a 2

- Globální prohledávání
 - Nalezne **cestu** do cíle
 - BFS, DFS, IDS
 - Lze zrychlit s informací: Hill-climbing, A*
 - Problémy s pamětí
 - Přirozený přístup ke hraní sekvenčních her





Spojitý prostor

Působí složitěji než diskrétní, ale viděli jste opakovaně, že prakticky může být velmi stravitelný.

- Řešením je stav, nikoliv sekvence akcí
 - Baseline: Random search
 - Paměť není problém, neudržujeme stromy
 - ehm-ehm, 7B+ transformery
- Zajímá nás jen kvalita stavu, nikoliv cesty
 - Můžeme znát cílovou hodnotu (zádné ohrožené dámy)
 - Ale nemusíme (najděte nejrychlejší postup rekonstrukce D1)
- Odpovídá optimalizaci

Optimalizace není učení

Při učení jde ve skutečnosti o, přímo nedostupnou, validační výkonnost.

- Variace best-first prohledávání, hladové prohledávání
- Drží jediný stav, nahrazuje ho jeho nejlepším sousedem
- Trpí lokálními extrémy
 - => Restarty, pak pokryje větší plochu
 - + Vybírat náhodný lepší, pak může ze stejného stavu zamířit k jinému extrému
- Lze povolit i „kroky stranou“ na stejnou hodnotu, ale pak hrozí cyklení
 - => Restartovat po určitém počtu kroků namísto po konvergenci

- Paralelní hill-climbing
- Udržuje k stavů
- Jednotlivá řešení spolu soupeří
- Následníky vybírá ze sousedů všech
 - top- k => jistá konvergence
 - top- k , lokálně nejvýše top- ℓ => diverzita, chvíli
 - stochasticky k => diverzita, snad

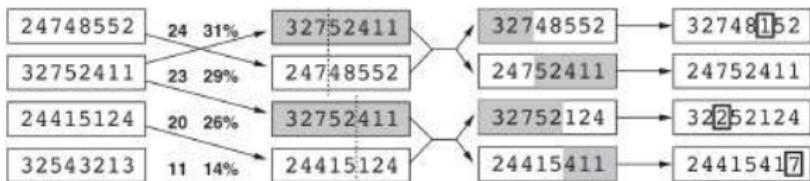
- Kombinace náhodné procházky a hill-climbingu
- Vybírá náhodného souseda s' :
 - lepší bere automaticky
 - u horšího spočítá $\Delta E = E(s') - E(s)$
 - horší je přijat s pravděpodobností $e^{-\Delta E/T}$
- Průběžně se snižuje teplota – kovodělná analogie
- Záleží na měřítku energie $E(s)$ i harmonogramu ochlazování

Pozor na ΔE

Někteří autoři ho definují s opačným znaménkem.

Vztah k posilovanému učení

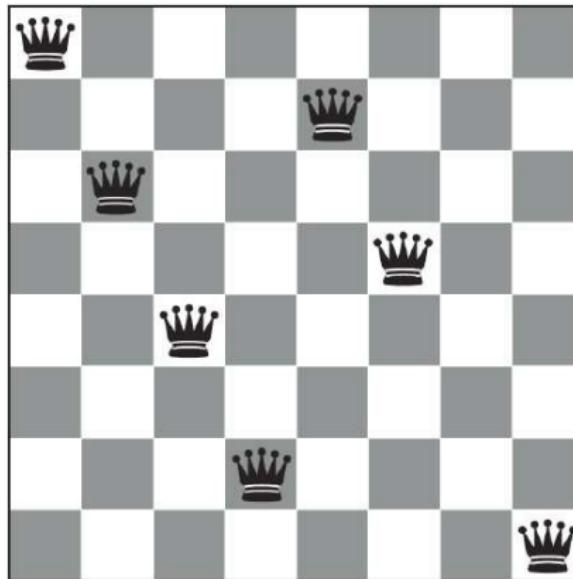
Snižování teploty řídí přechod od prozkoumávání (**exploration**) k využívání (**exploitation**). Vzdáleně souvisí i s **learning rate** při gradientní optimalizaci.



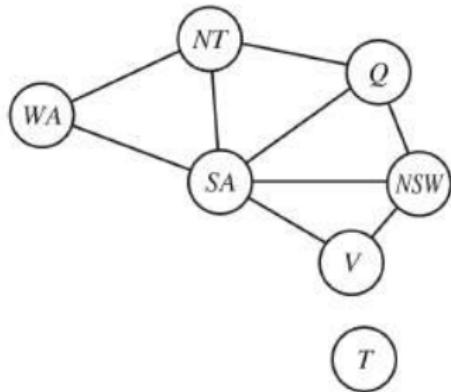
- 1 Výběr (hodnotící **fitness funkce**)
- 2 Křížení
- 3 Mutace

- Dosud jsme se dívali na stavů jako atomické, vztahy byly dány pouze akcemi
- V přírodě je ale efektivní křízení stavů (jedinců)
- Předpokládáme **kódování**, které dává lokálně smysl
- Křízení umožňuje kopírovat různě granulární části stavů
- Existují varianty pro:
 - n-D chromozomy
 - permutace
 - spojité geny
 - ...

- Většina lokálního prohledávání považovala stav za atomický
- Genetické algoritmy chápou stav jako faktorizovaný ...
- ... ale ohodnocení je pořád atomické prostřednictvím fitness funkce
- Prakticky je spousta požadavků lokálních

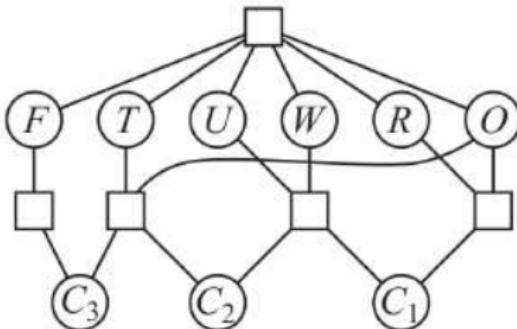


$\binom{64}{8} \approx 4.4 \text{B}$ stavů, ~ 60 akcí; nebo
 $8^8 = 2^{24} \approx 16 \text{M}$ stavů, 8 akcí; nebo
8 proměnných á 8 hodnot



7 proměnných á 3 hodnoty
žádná hrana nesmí mít stejnobarvené konce

$$\begin{array}{r} T \ W \ O \\ + T \ W \ O \\ \hline F \ O \ U \ R \end{array}$$



6 proměnných á 9 hodnot, různé

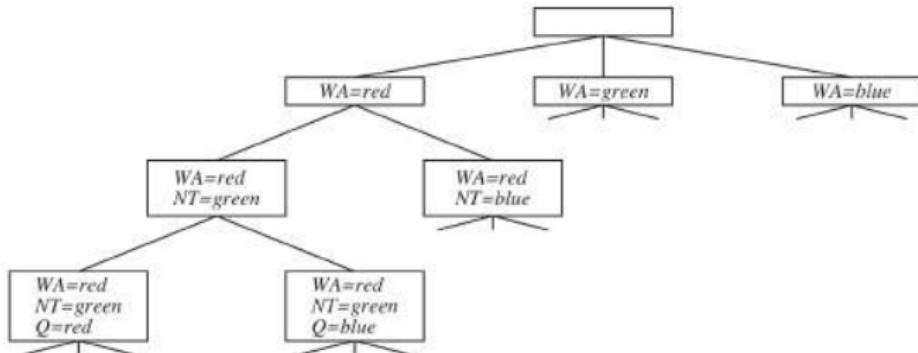
3 pomocné binární proměnné (carry)

čtverečky zapouzdřují podmínky (e.g. $W + W + C_1 = 10 U + C_2$)

- Proměnné $X_i, i \in \{1, \dots, N\}$
- Jejich domény $D_i = \{v_1, v_2, \dots, v_k\}, i \in \{1, \dots, N\}$
- Podmínky $C_i, i \in \{1, \dots, T\}, C_i = (S_i, R_i)$
 - Rozsah (scope) $S_i \subseteq X$, vymezuje, kterých proměnných se podmínka týká
 - Relace R_i vymezuje přijatelné hodnoty proměnných $X_c \in S_i$; výčtem nebo logickou formulí.
 - S přidáním pomocných proměnných se lze omezit na binární omezení.

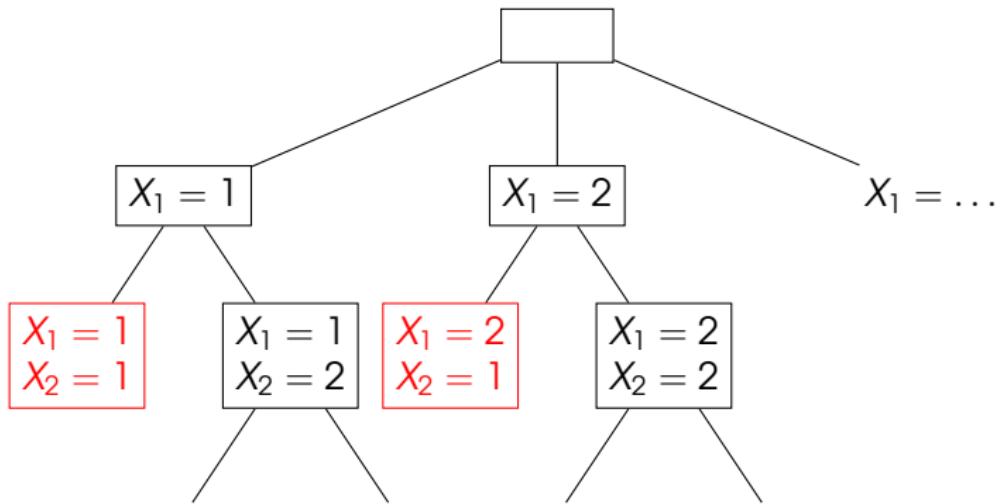
Řešení

Přířazení $X_1 = v_1, X_2 = v_2, \dots, X_k = v_k$, které je konzistentní se všemi podmínkami. Cenu neřešíme.



- Částečné přiřazení
- Fixní pořadí přiřazování (jinak $b \in \mathcal{O}(nd)$ a $n!d^n$ listů)
=> prohledávání: DFS \approx DLS
- $\mathcal{O}(|C| \cdot |D|^N)$
- Okamžitě můžeme testovat podmínky a tím prořezávat

- Nechť $X_2 \neq 1$



- Testovat na konci – velké plýtvání prohledáváním
- Testovat průběžně – efektivní prořezávání (není zdarma)
- Rozbaloval by člověk červené uzly vůbec?

- Nechť:
 - $D_* = 0, 1, \dots, 9$
 - $C_1 : X_1 + X_2 = X_3$
 - $C_2 : X_2 + X_3 = X_5$
 - $C_3 : X_1 \cdot X_1 = X_4$
 - $C_4 : X_3 + X_4 = X_5$
- Žádná unární podmínka
- Ale má cenu uvažovat $X_4 = 2$?
- Nebo $X_1 = 4$?

- Konzervativní omezení domén D_i tak, aby vyhověly podmínkám
- Unární – konzistence uzlů (node consistency)
 - X_i je konzistentní v uzlu, pokud žádná hodnota v D_i neporušuje žádnou unární podmínsku
- Binární – konzistence hran (arc consistency)
 - X_i je hranově konzistentní s X_j , pokud pro každou $v_i \in D_i$ existuje $v_j \in D_j$ taková, že $(X_i = v_i, X_j = v_j)$ neporušuje žádnou podmínsku
- Ternární – konzistence cest (path consistency)
 - $\{X_i, X_j\}$ je cestově konzistentní s X_k , pokud pro každou $v_i \in D_i, v_j \in D_j$ existuje $v_k \in D_k$ taková, že $(X_i = v_i, X_j = v_j, X_k = v_k)$ neporušuje žádnou podmínsku
- Obecně – K-konzistence
 - každá $(K - 1)$ -mohutná množina proměnných je konzistentní vzhledem ke každé jedné další proměnné
 - silná K-konzistence – konzistence i pro všechna menší K

- Udržuje frontu orientovaných (!) hran k vynucení, začne všemi
 - Vyjmě její čelo (X_i, X_j) a vynutí ho
 - Vyřadí z D_i v_i , pro něž neexistuje vyhovující $v_j \in D_j$
 - $\mathcal{O}(|D_i||D_j|)$
 - Pokud něco vyřadil, bude zkoumat následky:
 - Přidá do fronty (X_k, X_i) pro všechna $k \neq j$
 - Každou hranu zkoumá nejvýše $|D_i|$ -krát, pokud v_i odpadají po jedné
- => $\mathcal{O}(|C||D|^3)$
- Prázdná D_i => úloha nemá řešení

Vyšší rády

Analogicky, ale s hrozivými paměťovými nároky.

3 6	7	1 6	4	1 5 3	8	1 5	2	9
3 6 9	1 6 9	2	1 5 7 9	1 5	7 9	1 5 8	3 5 6 8	4
8	5	4	1 9	2	6 9	1 3	3 6	7
5 6 9	1 6 9	8	3	7	4	2	5 9	1 6
4 5 7	5 6 9	2	1 7 9	5 8 9	5 8	3 9	4 5 8 9	1 6
4 5 9	4 9	3	2	6	1	7	4 5 8 9	5 8
4 5 7	4 8	5 7	5 7 8	9	3	6	1	2
2	6 8 9	5 6 7 9	1 5 7 8	1 5 8	5 7	4	5 8 9	3
1	3	5 9	6	4	2	5 8 9	7	5 8

- Obrázek z: www.sudokuwiki.org/Naked_Candidates
- Nemusíme znát přesnou pozici pro 5, 8 a 9 zvlášť
- Lze zobecnit na libovolnou n-tici

- Konstantní hloubka řešení
 - => Optimalizace == Fail quickly
- Chápe vztah proměnná – doména – hodnota, takže může:
 - vybírat vhodnou proměnnou pro další krok
 - vybírat vhodnou hodnotu k přiřazení
 - před-prořezávat domény

Rekurzivní implementace (* prostor pro optimalizaci)

- ① Vyber si volnou proměnnou (*)
- ② Pro každou hodnotu (*):
 - ① Zkus jestli je konzistentní. Pokud ano, přiřad' ji a
 - ② zuž domény pomocí inference (*)
 - ③ zavolej se znova s rozšířeným přiřazením a zúženými doménami
 - ① Úspěch propaguj
 - ② V případě neúspěchu zkus další hodnotu
- ③ Pokud žádná hodnota nevedla k úspěchu, úlohu nelze s daným přiřazením vyřešit

- Jedna nepřířaditelná proměnná – neřešitelná úloha
 - => Proměnná s nejméně zbývajícími hodnotami
- Hodně omezení – zjednodušené podúlohy
 - => Proměnná s nejvíce omezeními
- Neúspěšná hodnota – musíme zkoušet další
 - => Nejméně omezující hodnota
 - Nemá efekt, pokud hledáme všechna řešení nebo řešení neexistuje

Uspořádání proměnných

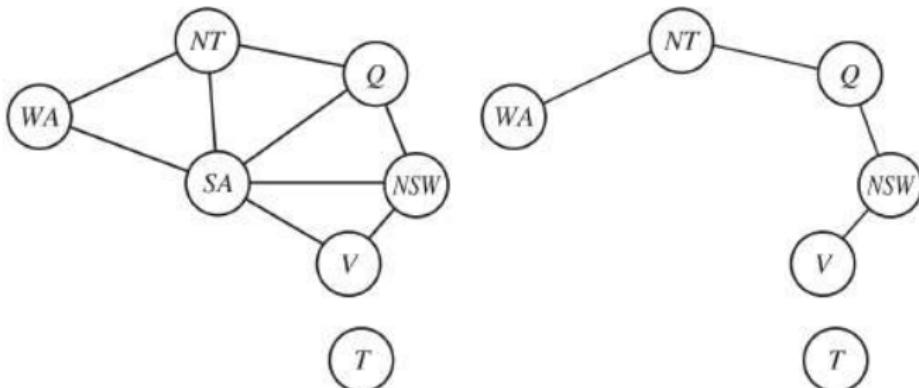
Backtracking rozumí tomu, že jsou proměnné a hodnoty, takže není potřeba explicitně bojovat s permutacemi proměnných.

- Inferenci je záhodno provést před prohledáváním
- Ale každé přiřazení zužuje efektivní domény
 - => inference může odřezat více
 - => můžeme přesněji identifikovat nejomezenější proměnnou
- Forward-checking
 - Při přiřazení proměnné X_i vynutí hranovou konzistenci všech sousedních X_j .
 - => Nemá efekt, pokud jsme již hranovou konzistenci vynutili globálně.
- Maintaining Arc Consistency
 - Při přiřazení proměnné spouští AC-3 pro všechny hrany do dosud nepřiřazených proměnných.
 - striktně silnější než forward-checking

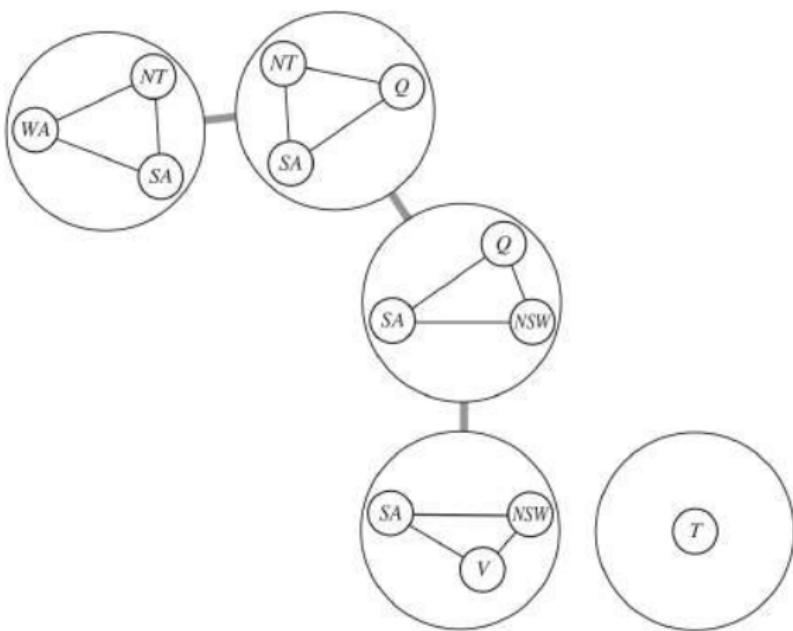
- Backtracking se implicitně vrací k poslední přiřazené hodnotě
 - Má ale vliv na selhání?
 - Hodila by se konfliktní množina $\text{Conf}(X_i) \subseteq X - X_i$ proměnných, které omezily X_i
 - Návrat do nejnověji přiřazené
- Konflikty přirozeně dodá forward checking
 - Vždy když odřízne hodnotu z X_j , přidá X_i do $\text{Conf}(X_j)$
 - Pokud dorazíme do X_j s $D_j = \emptyset$, víme o $X_i \dots$
 - ...jenže selže už forward checking v X_i
- Propagujeme omezení nahoru
 - Šíří informaci o nepřímých konfliktech – které nastaly
 - Umožňuje „učení“ konfliktních částečných přiřazení (podobně jako dynamické programování na AND-OR stromech)
 - Conflict-based backjumping (ale jsou i jiné)

Trade-off

Dobrá práce dopředných kontrol omezuje prostor pro chytré návraty.



- Jednotlivé spojité komponenty zvlášť
- Stromy triviální
 - Hranová konzistence de facto řeší
 - Přiřazování musí proběhnout souvisle (typicky topologické uspořádání, pak je i AC-3 rychlejší)
- Obecnější záhadno rozsekat na stromy
 - Přiřazení do ne-stromových uzel => jednoduché podúlohy
 - Hledání kruhových odřezků (cycle cutset) je NP-těžké



- Každý podproblém zvlášť, vznikou agregátní proměnné
- Hranы mezi aggregáty vynutí kompatibilitu řešení podproblémů

- Min-Conflicts, nainicializuj náhodně a:
 - ① Vyber si proměnnou
 - ② Nastav ji na hodnotu s minimem konfliktů
 - ③ Opakuj
- Hrozí roviny: Tabu search
 - ① Spousta přiřazení s právě jedním konfliktem, „lokální minima“
 - ② Tabulka posledně navštívených => lepší prozkoumávání
- Ne všechny podmínky jsou si rovny: Constraint weighting
 - ① V každém přiřazení zvyš (relativní) váhu porušených podmínek
 - ② Při každém kroku minimalizuj váhu porušených podmínek

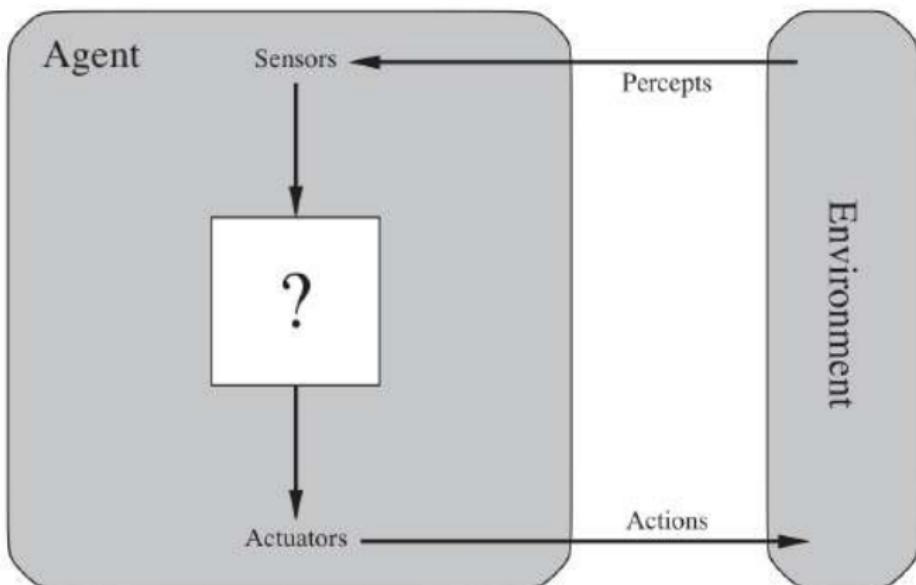
Online-prohledávání

Pokud se posune deadline jednoho projektu, není nutné plánovávat celý semestr znova na zelené louce.

- Úlohy lokálního prohledávání s explicitními proměnnými,
- **FaktORIZovaný stav**
- Backtracking
 - Kombinuje prohledávání s **inferencí**
 - Daný jazyk popisu omezení => univerzální heuristiky
- Lokální prohledávání
 - Randomizovaný (Monte-Carlo) algoritmus
 - Překvapivě rychlé (Milion dam v 50 krocích)
 - Vhodné pro on-line změny omezení

Rozšíření

- Spočetně nekonečné domény: vymezuje domény intervaly
- Spojité domény: optimalizace
 - Lineární programování (simplexová metoda)
 - Kvadratické programování, ...
 - Lagrangeovy multiplikátory

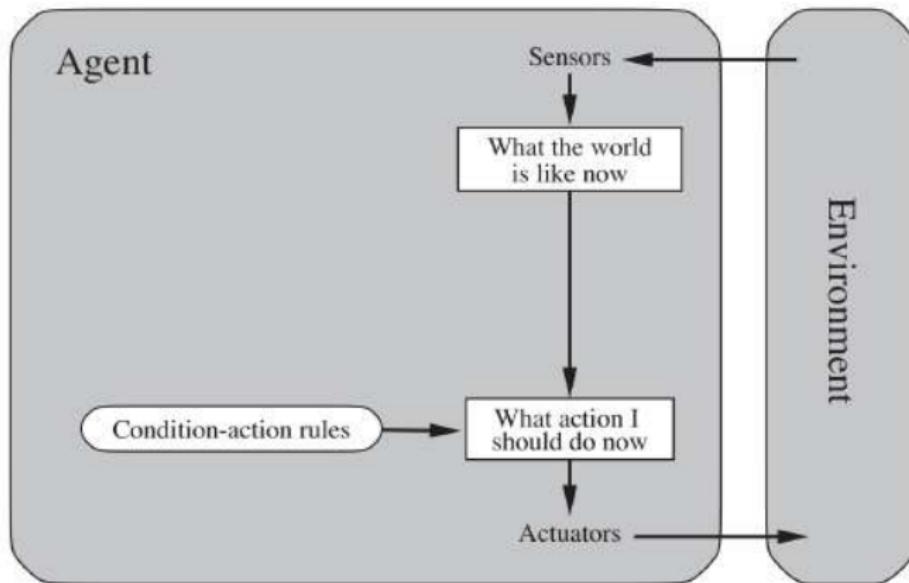


- PEAS – **Performance**, Environment, Actuators, Sensors
 - Agent je **racionální**, pokud optimalizuje výkonnost vzhledem ke svým možnostem

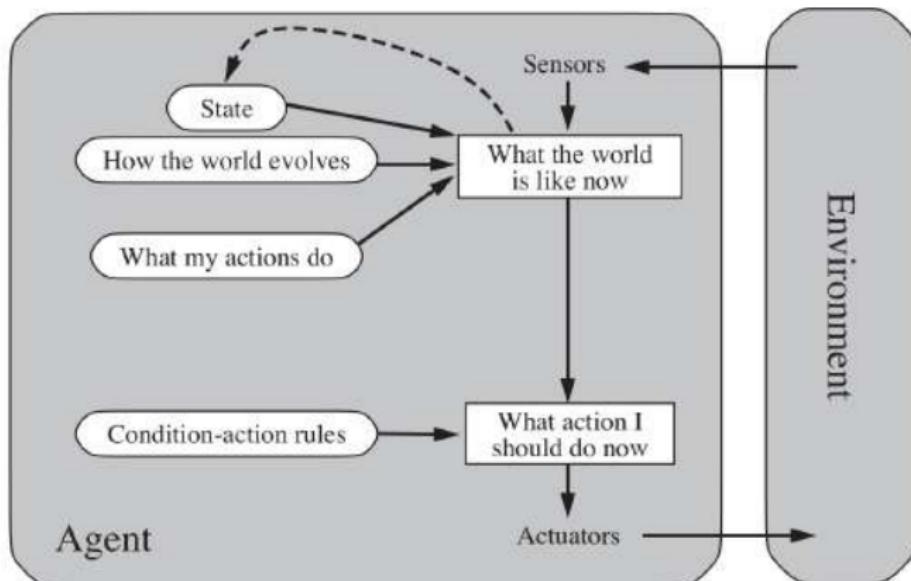
- Plně pozorovatelné vs. částečně pozorovatelné
- Deterministické vs. s náhodou
 - Náhoda může být kvantifikována (ale nemusí)
 - Příliš rozsáhlá prostředí => prakticky náhoda
- Jisté vs. nejisté
 - Jisté je plně pozorovatelné a deterministické
- Jeden agent vs. více agentů
 - Hry vs. ekonomika
 - Soutěž vs. (nucená) spolupráce

- Epizodické vs. sekvenční
 - Epizody jsou nezávislé => agent nemusí plánovat přes jejich hranice
- Známé vs. neznámé
 - Znalosti tvůrce, neplést s pozorovatelností!
- Diskrétní vs. spojité
 - Stav, pozorování, akce
 - Čas
- Statické vs. dynamické
 - Děje se něco, dokud agent přemýší?
 - Semidynamické: ne, ale mění se skóre (např.: šachové hodiny, Hledání min)

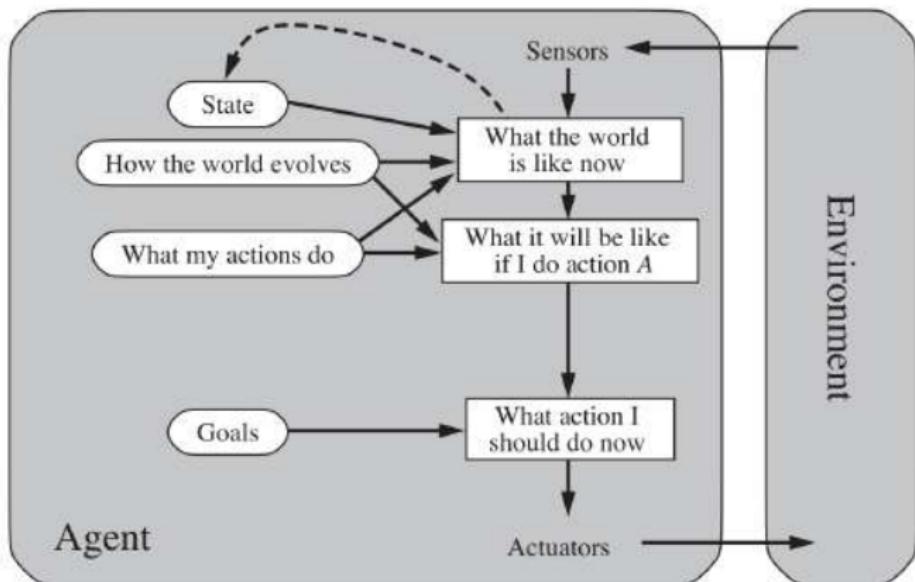
- Agentní funkce
 - Posloupnost vjemů $P_t \rightarrow$ akce
 - Lze teoreticky popsat tabulkou ($\mathcal{O}(|P|^t)$)
- Agentní program
 - Realizace agentní funkce konečně velkým algoritmem
 - Pokud chce přistupovat k minulým vjemům, musí si je pamatovat
=> Stav agenta
- Agent
 - Agentní program
 - + „fyzická“ architektura



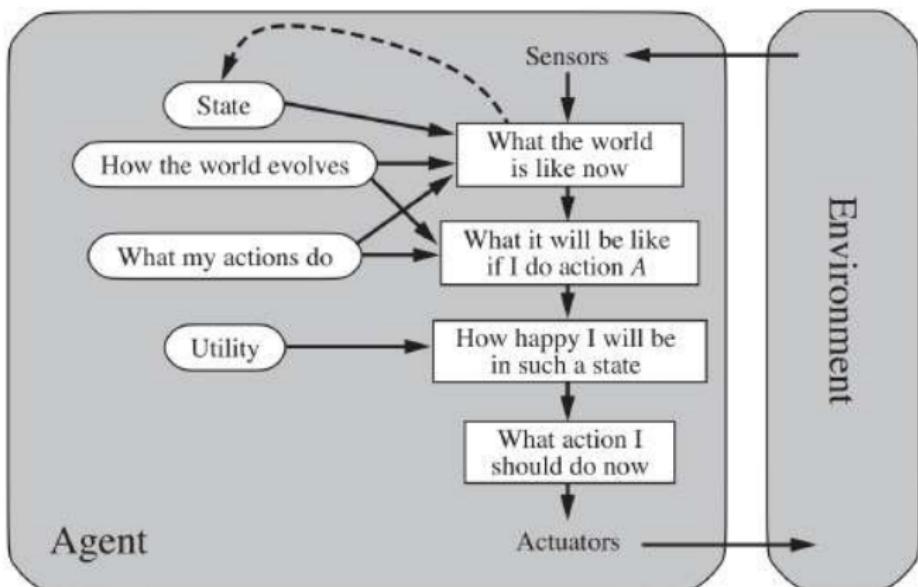
- „Vidím Kebab House a svítí zelená => Jedu, doleva!“



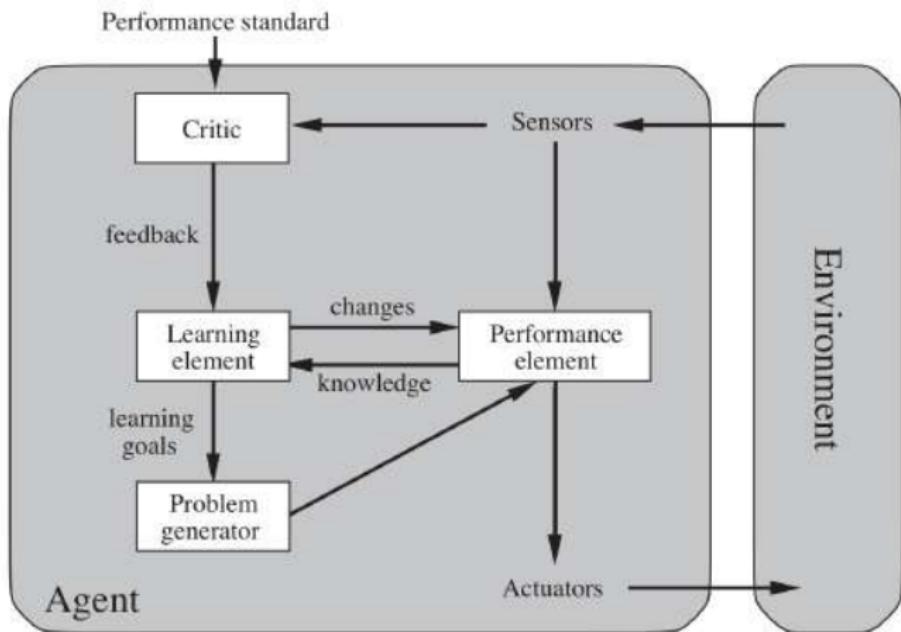
- „Vím, že jsem v křížovatce, odbočuju doleva, protisměr se zastavil => Jedu!“



- „Mám se dostat <na FIT>. Jsem v tunelu => Jedu na Svitavy!“



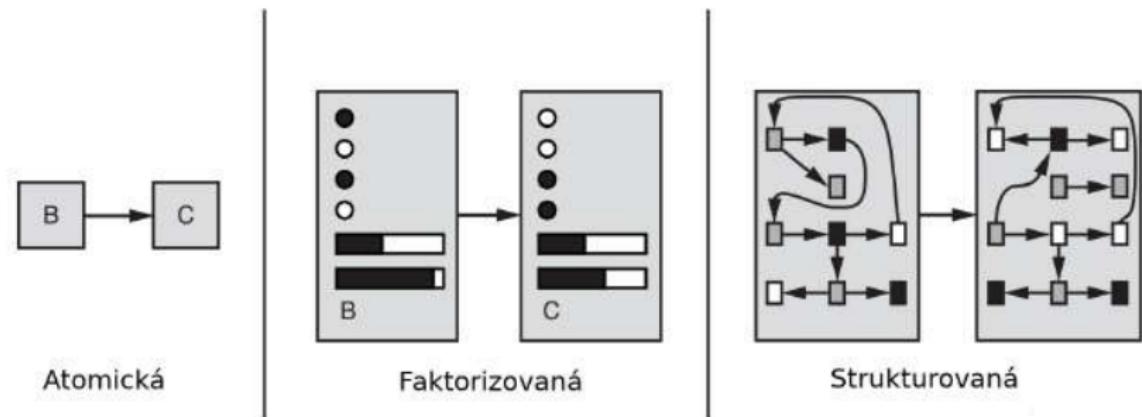
- „Těší mně <dělat něco pro studenty>. Učím SUI => ...“



- Nadstavba nad fixním agentem
- Kritik musí být dosti univerzální
 - Bolest, body, hlad, endorfiny ...
 - Prohra, výhra
- Generování problémů je nutné k efektivnímu učení
 - V posilovaném učení často banální „explorace“

Autonomie

Agent, který se neučí, nemůže být autonomní; je nevyhnutelně omezený tvůrcem.



- Atomická: nedělitelné (např. klasické prohledávání)
- Faktorizovaná: proměnné s doménami (např. CSP)
- Strukturovaná: objekty ve vztazích (např. predikátová logika)

- Agent operuje v prostředí
- Racionální agent optimalizuje metriku (ale není dokonalý)
- Parametry prostředí vedou k dobrým návrhovým volbám
- Explicitní cíl/užitek vede ke znovupoužitelným agentům

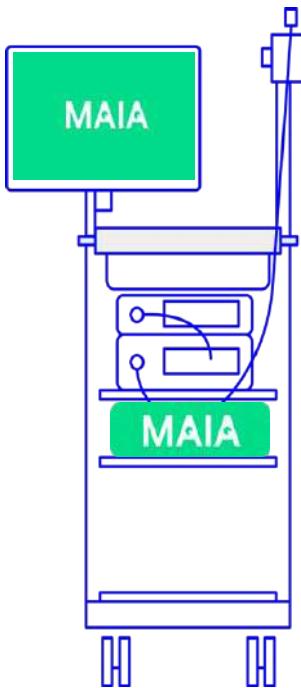
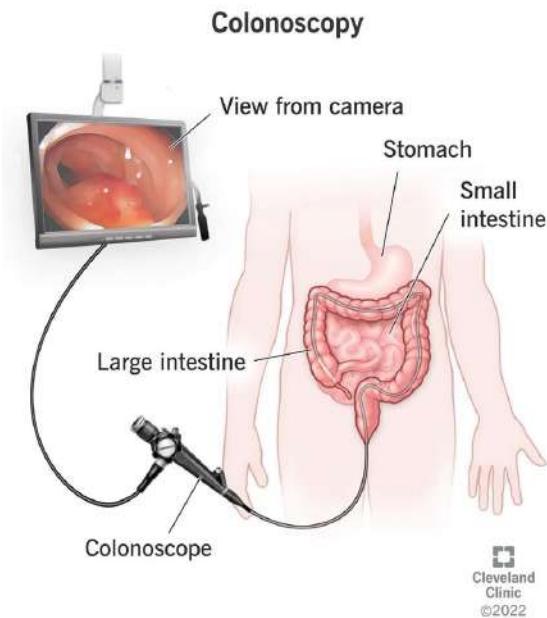
agěre

lat.: konám, inf. agō

Machine learning product lifecycle

Michal Hradiš

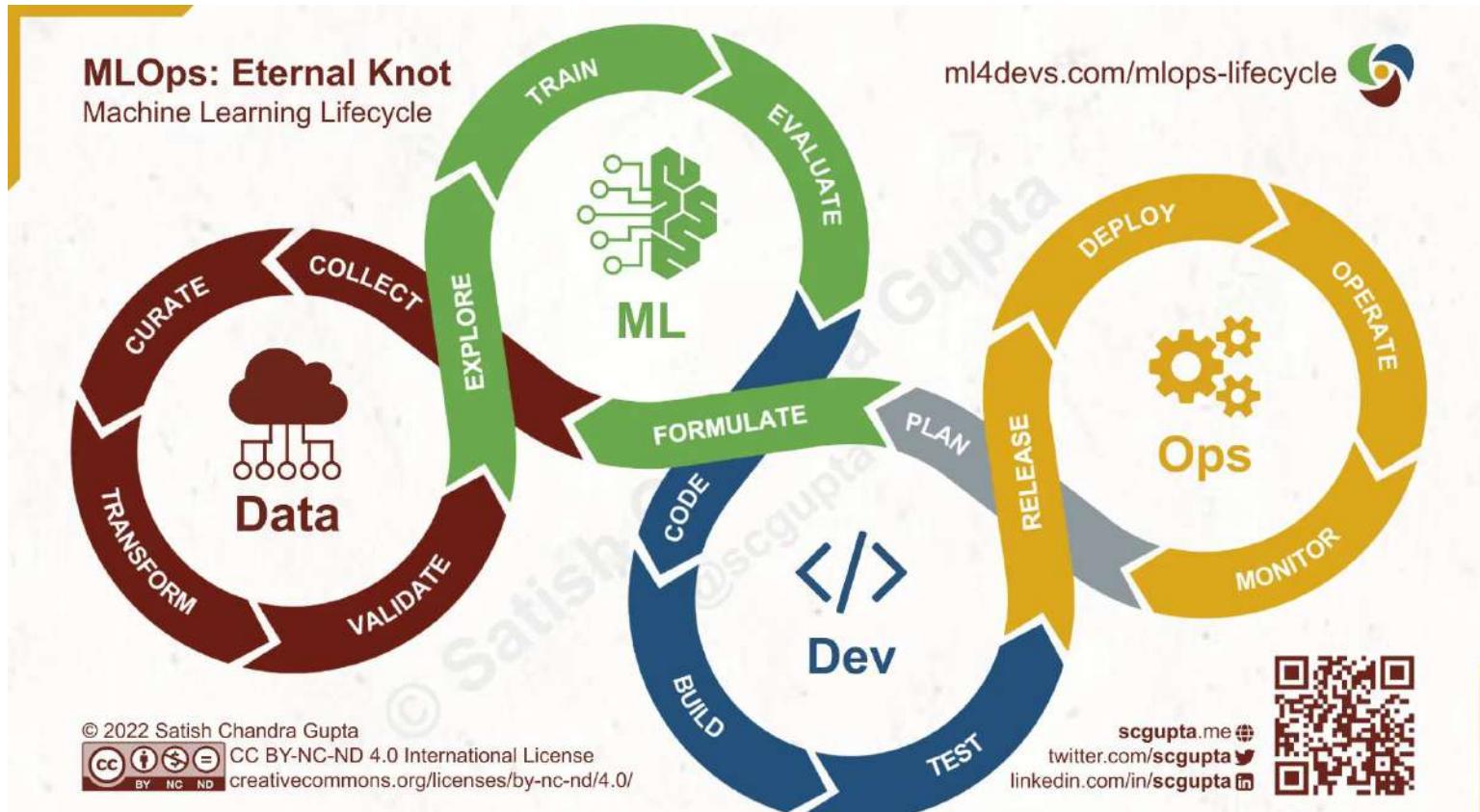
MAIA - Medial AI Assistant (Colonoscopy)



ML development considerations

- Trust in solution quality (credibility)
- Possibility of continuous improvement
- Ease of deployment
- Reproducibility
- Maintainability
- Documentation
- Certification
- Security
- Team cooperation
- Legal issues

Machine learning project lifecycle - MLOps



Data collection and annotation

Data source (availability, technical + legal issues)

Secure data storage (data could be sensitive, private or classified)

Annotation tool (LabelStudio, V7, RoboFlow, or custom annotation tools)

Label reliability - expert annotator, annotation consistency, manual annotation verification, automated verification, defined annotation workflow

Annotation journal/log - who annotated what? Needed for retrospection and certification.

Automated assistive tools

Active learning ...

Experiment desing

Evaluation metrics in experiments?

How to do end-to-end testing?

What error are significant? What is the “cost” of different languages.

Train, validation and test dataset partitioning

Dataset contamination and information leaking.

Model/experiment management and versioning

Experiments should be reproducible.

Code versioning (GIT).

Reproducible environment (python virtualenv, docker images, virtual machines).

Experiment logging. Model versioning.

Result analysis

Distributed task queues

Tools - mlflow, ClearML, Weights & Biases, Tensor Board,

Dataset management and versioning

- Dataset preparation/preprocessing pipelines
- Dataset versioning
 - Experiment reproducibility
 - Are the results due to method or change in dataset?
- Tools: DVC, ClearML, ...
- Datasets could be large
 - Storage
 - SquashFS

Model deployment

- Cloud
 - Interface - REST,
 - Scaling
 - Monitoring
 - Tools: ClearML, Kubernetes, Amazon SageMaker, Azure Machine Learning, ...
- Device (model optimization, TensorRT, OpenVino, HW platforms)
 - Latency
 - Throughput
 -



Monitoring, Feedback and updates

Are results still good?

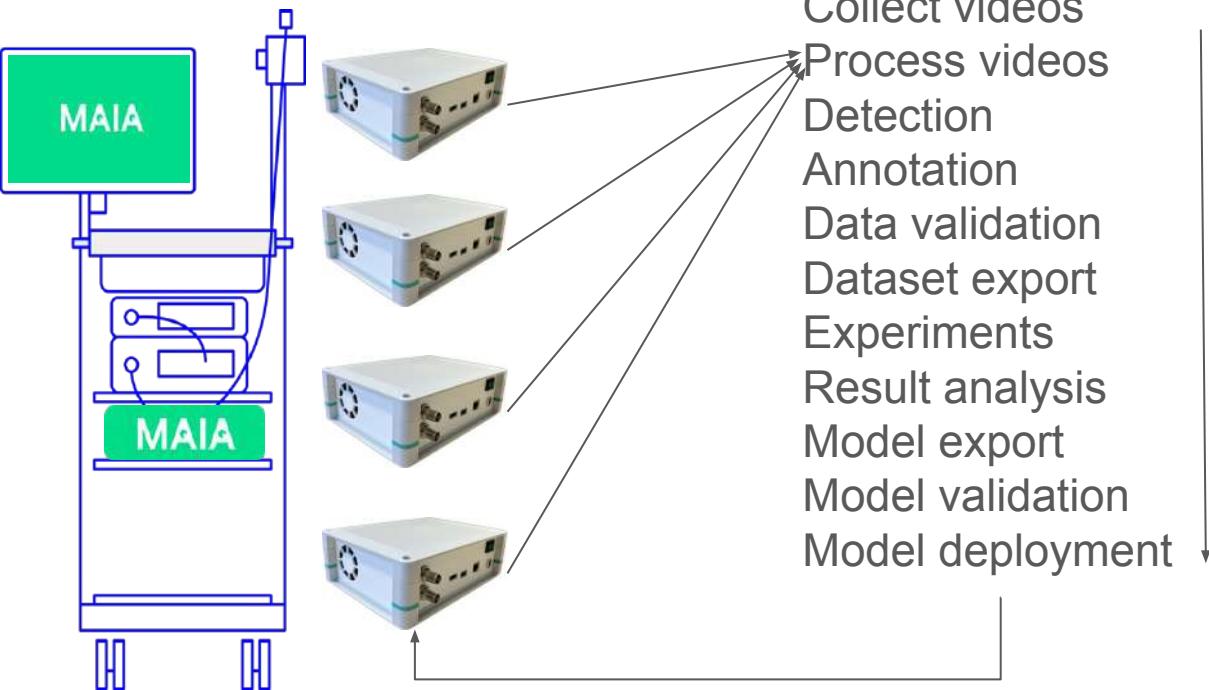
Does the system fail on some data?

How to deploy updates to devices?

Device with several system versions.

Updates should be secure.

MAIA - Medical AI Assistant



Secure data storage
Processing pipelines
Annotation tools
Dataset paper trail
Dataset versioning
Automated testing
Platform specific
optimized inference
Automated updates
Device security
Telemetry
User feedback