Apprendre à jouer

a.k.a. Introduction to Deep and Reinforcement Learning

Mathieu Aubry, LIGM-Imagine

About me

- Researcher in Computer Vision
- At École des Ponts (across the street, shared lab with UPEM)
- Team focus: machine learning and computer vision
- Personal focus: applications to Digital Humanities

Today

1. Introduction to supervised and Deep Learning
   (focus on Computer Vision)

2. Introduction to reinforcement learning
   (focus on learning Pong)

2016 New Yok Times

'The portion of evolution in which animals developed eyes was a big development. Now computers have eyes.'

3. A Deep Explanation of Deep Learning

When Pichai said that Google would henceforth be "A.I. first," he was not just making a claim about his company's business strategy; he was throwing in his company's lot with this long-unworkable idea. Pichai's allocation of resources ensured that people like Dean could ensure that people like Hinton would have, at long last, enough computers and enough data to make a persuasive argument. An average brain has something on the order of 100 billion neurons. Each neuron is connected to up to 10,000 other neurons, which means that the number of synapses is between 100 trillion and 1,000 trillion. For a simple artificial neural network of the sort proposed in the 1940s, the attempt to even try to replicate this was unimaginable. We're still far from the construction of a network of that size, but Google Brain's investment allowed for the creation of artificial neural networks comparable to the brains of mice.

1958 New Yok Times

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly line and which would be conscious of their existence.

In today's demonstration, the "704" was fed two cards, one with squares marked on the left side and the other with squares on the right side.

Learns by Doing

In the first fifty trials, the machine made no distinction between them. It then started registering a "Q" for the left squares and "O" for the right squares.

Dr. Rosenblatt said he could explain why the machine learned only in highly technical terms. But he said the computer had undergone a "self-induced change in the wiring diagram."

The first Perceptron will have about 1,000 electronic "association cells" receiving electrical impulses from an eye-like scanning device with 400 photo-cells. The human brain has 10,000,000,000 responsive cells, including 100,000,000 connections with the eyes.

Slide L. Lazebnik

Supervised learning

- Data $\{(x_1,y_1),...(x_n,y_n)\}$
- Produces a predictor/decision function $\hat{f} : \mathcal{X} \to \mathcal{A}$

- Hypothesis: data sampled from random variable X and Y
- Attempts to minimize the risk $\mathbb{E}(l(f(X), Y))$
- By minimizing the empirical risk $\sum_{i=1}^{n} l(f(x_i), y_i)$

Example: Image classification

- Input x_i are images, i.e. $x_i \in [0,1]^{N \times N}$
- Label $y_i \in \{0,1\}^M$, $y_i^k = 1$ if image *i* is part of class *k*
- Prediction $f(x) \in [0,1]^M$, y_i^k probability image *i* is part of class *k*

$x =$ 

$$y = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \\ 0 \\ \dots \\ 0 \end{pmatrix} \longleftarrow \text{Australian shepherd}$$

- ImageNet: *NxN* ~ 50 K, ~1M number of training examples, 1000 classes

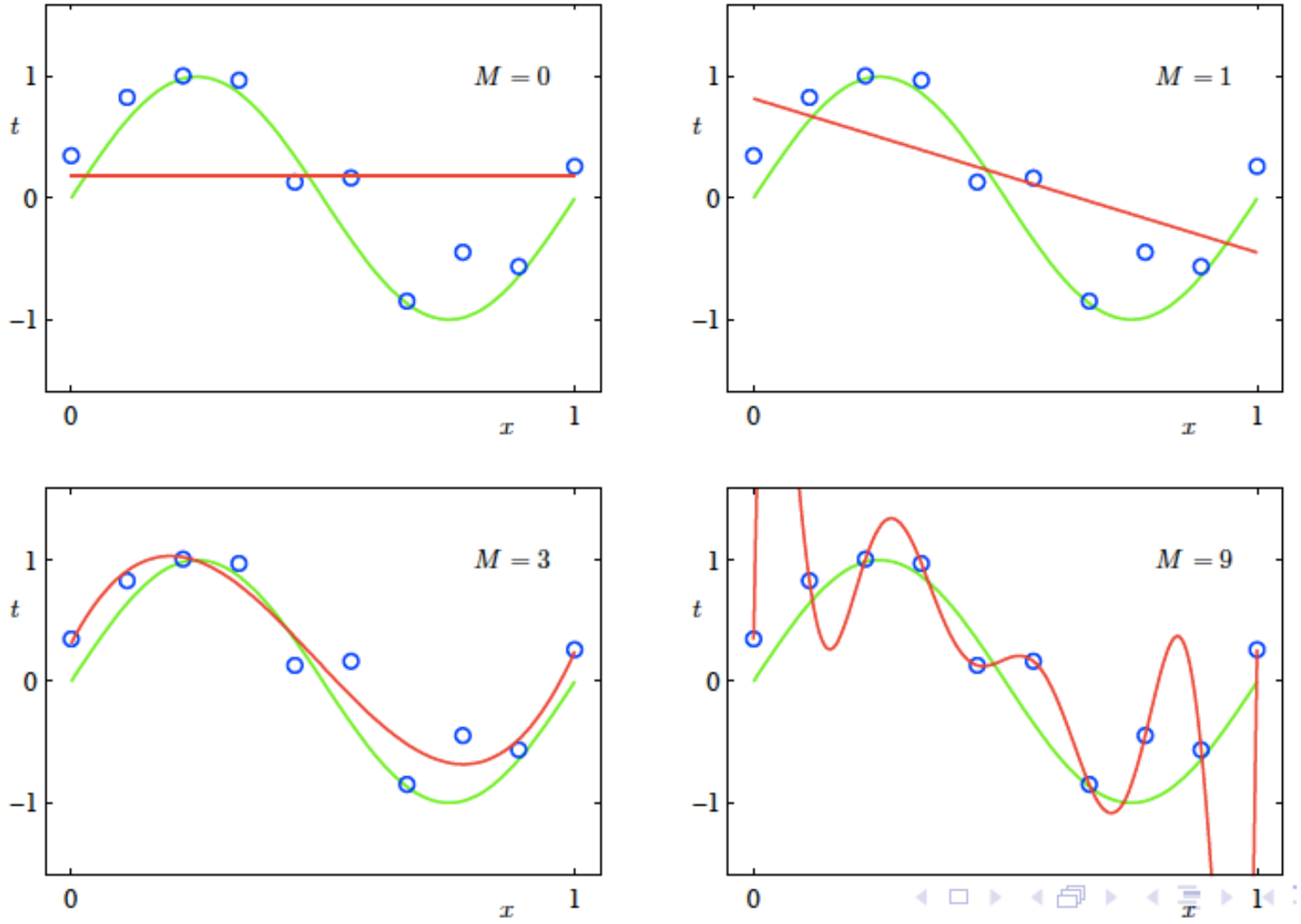
How can we define *f?*

- Using a set of parameters (e.g. linear/polynomial regression)
- Directly using the training data

What are the risks*?*

- Making poor predictions on the training data (underfitting)
- Not generalizing to unseen data (overfitting)

Example: polynomial regression of degree M

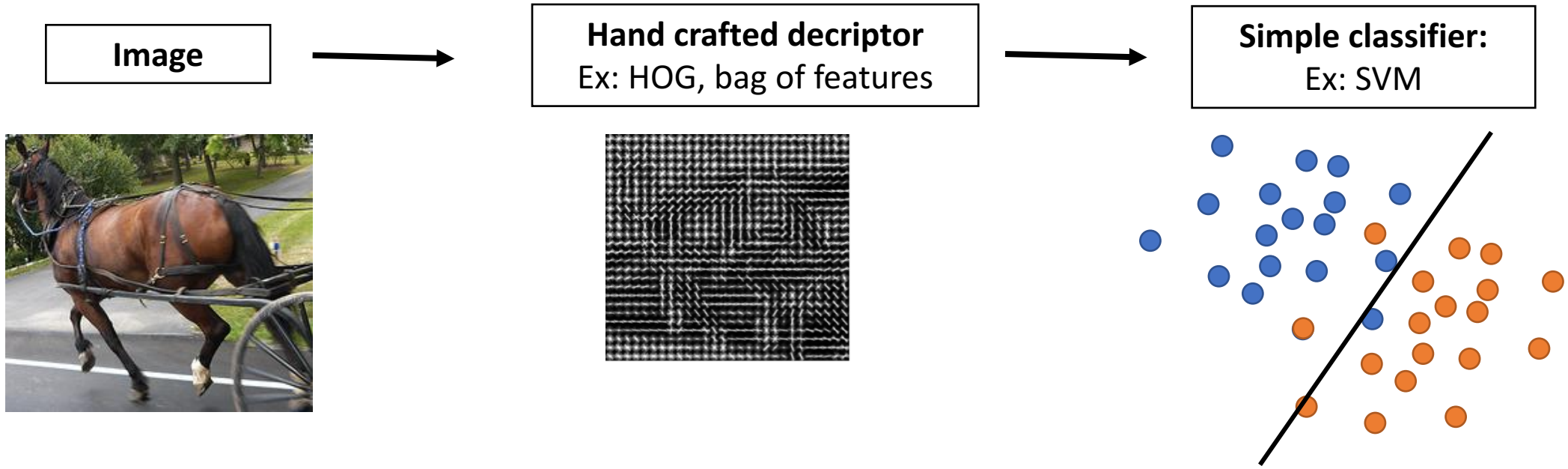


Breaking the curse of dimensionality

- The curse of dimensionality:
  - To approximate a function, you need a number of training examples exponential in the number of dimensions
  - 1M training images is far too little for 100K dimensions

- Solutions:
  - Use a simple predictor, with few parameters, e.g. linear classifier (linear regression, SVM…)
  - Regularize
  - Use a different (better) representation of the data, i.e. a feature

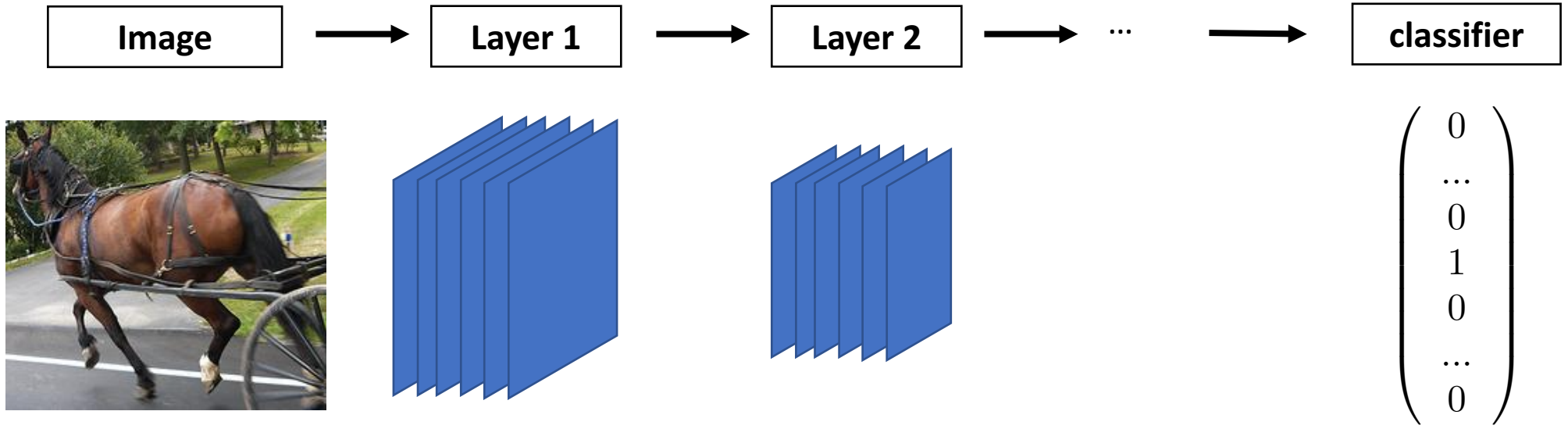
Example: classical vision



- Idea:
  1. Learn intermediate representation
  2. Multiply intermediate representations

Implicit hypothesis: this compositionality is useful for the data we have

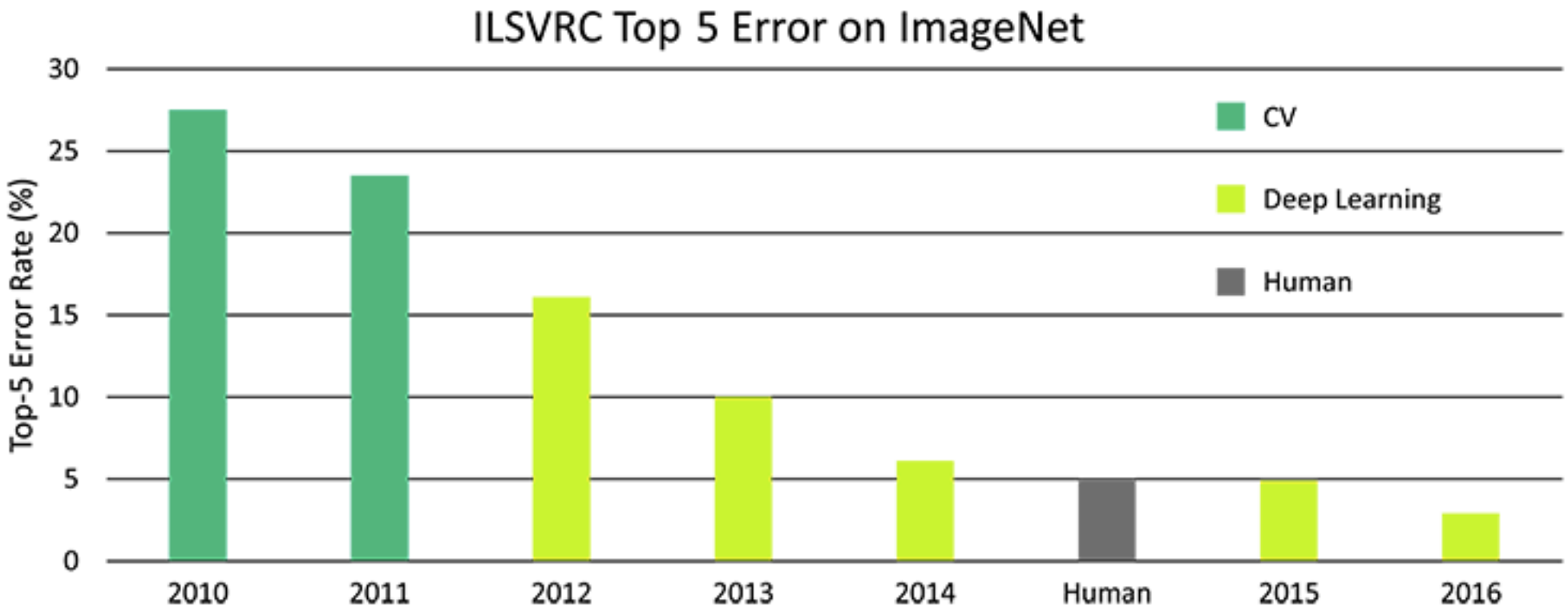
Deep Learning



- Idea:
  1. Learn intermediate representation
  2. Multiply intermediate representations

Implicit hypothesis: this compositionality is useful for the data we have

Results


ILSVRC Top 5 Error on ImageNet
Top-5 Error Rate (%)
30
25
20
15
10
5
0
CV
Deep Learning
Human
2010
2011
2012
2013
2014
Human
2015
2016

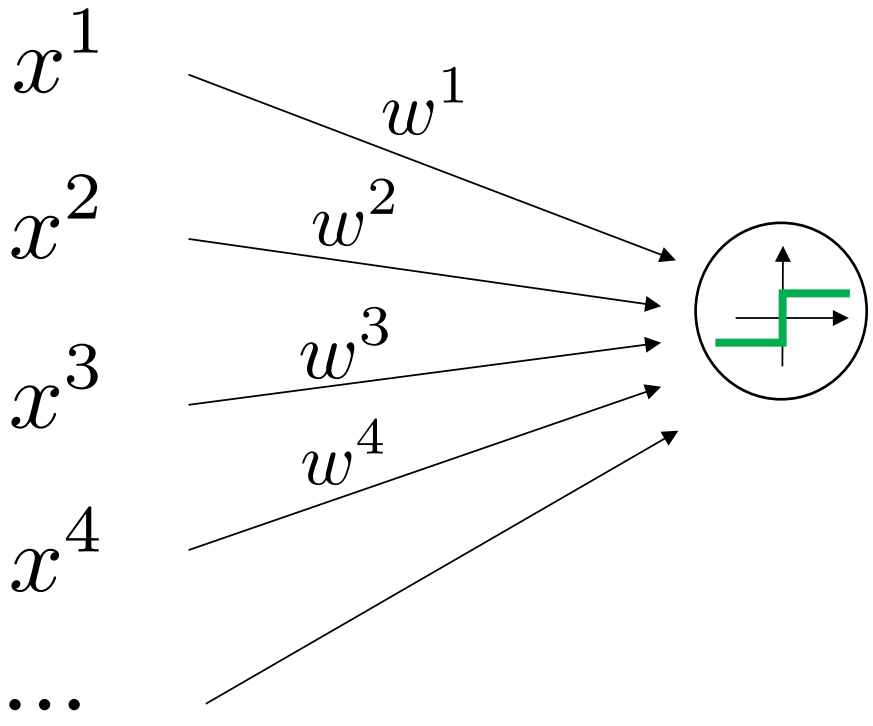
Perceptron

- Frank Rosenblatt, 1957

$$f(x) = sign\left(\sum_i w^i x^i\right)$$

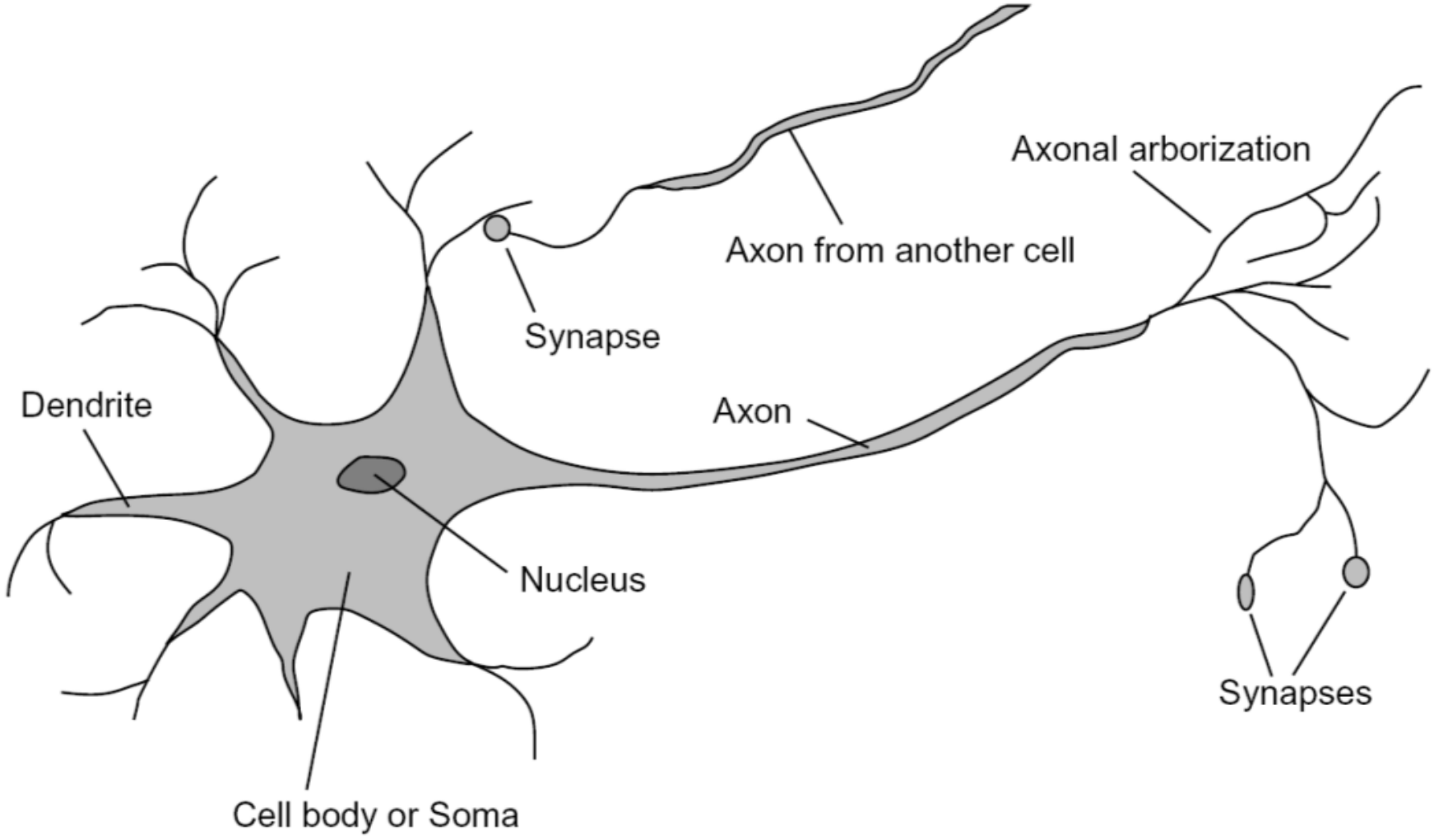
Perceptron

- Frank Rosenblatt, 1957



$$f(x) = sign\left(\sum_i w^i x^i\right)$$

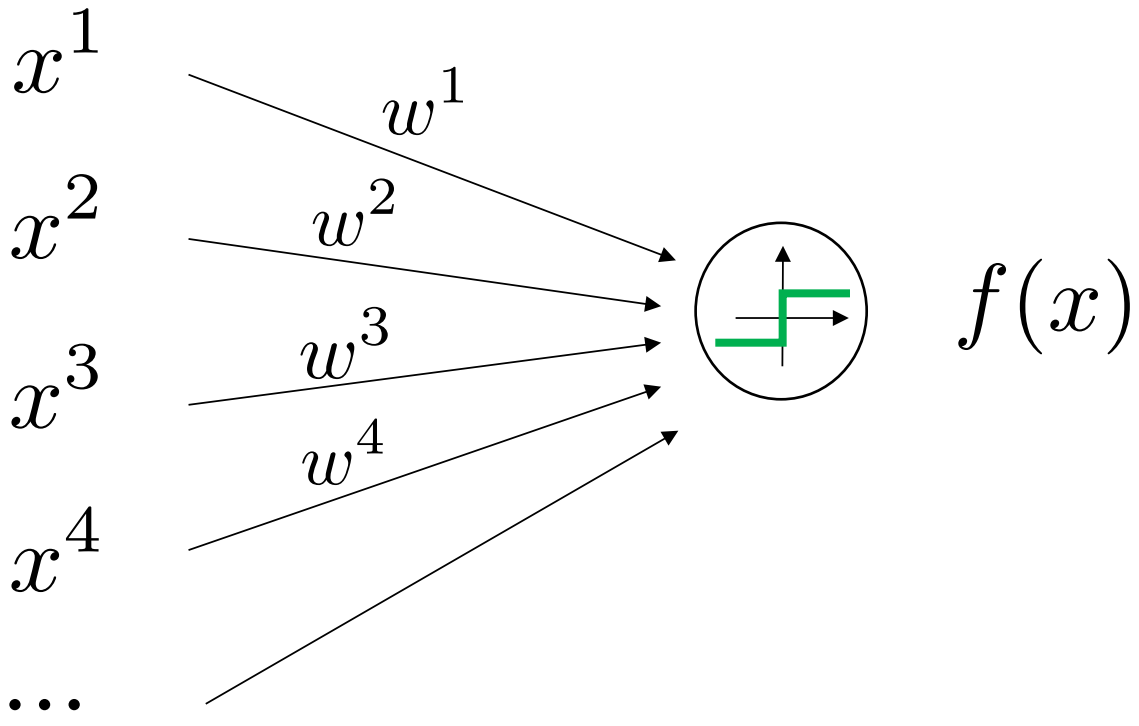
Biological neuron



Slide L. Lazebnik

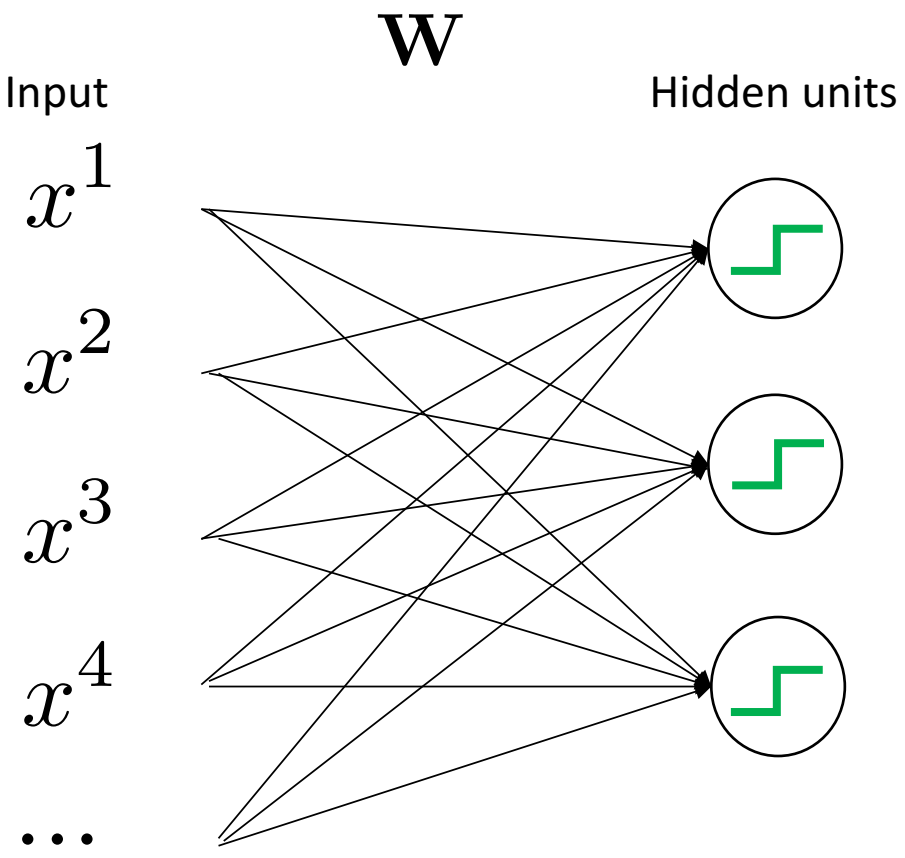
Perceptron

- Frank Rosenblatt, 1957

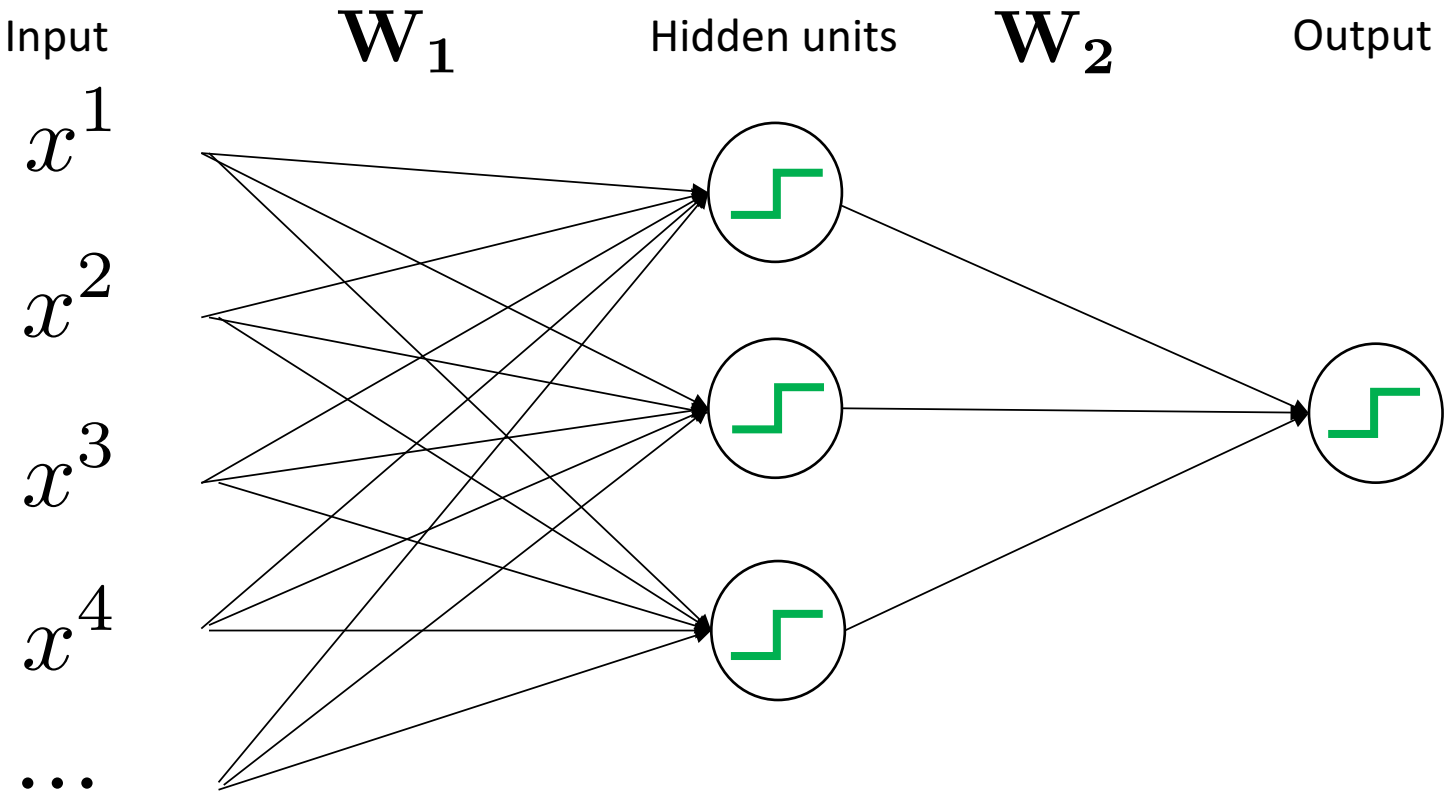


Issue: incapable of performing exclusive OR (Minsky and Papert 1969)

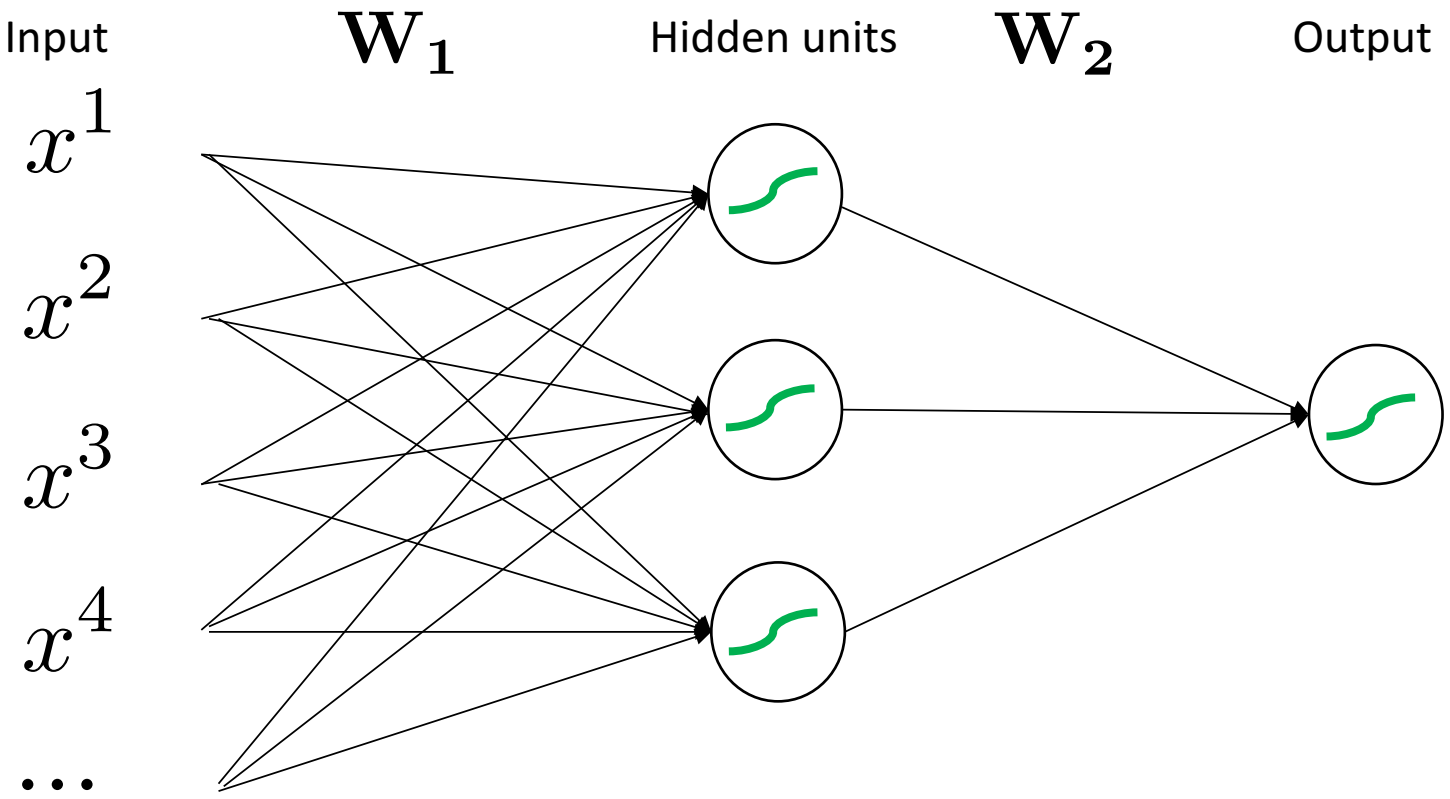
Perceptron



2 layers perceptron

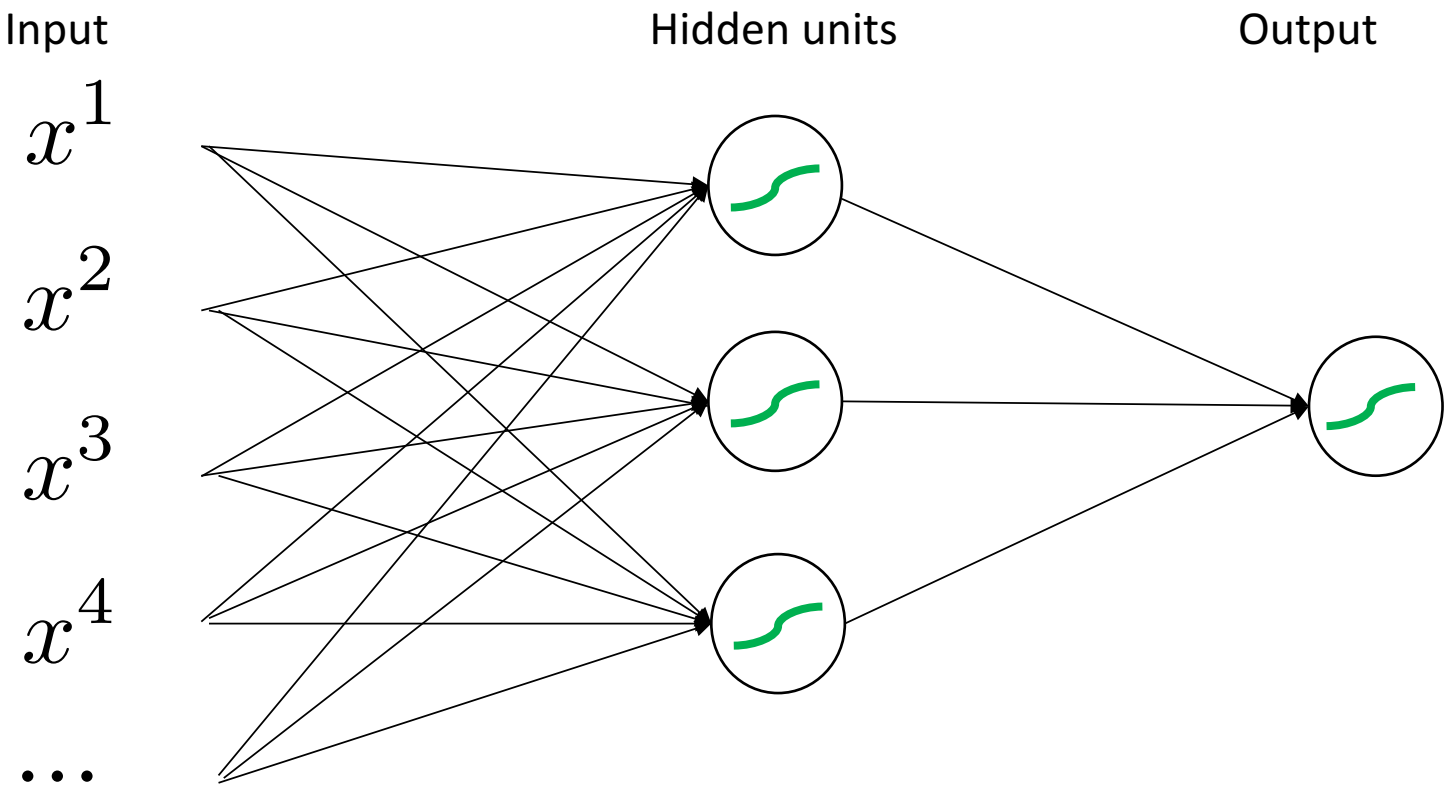


2 layers perceptron



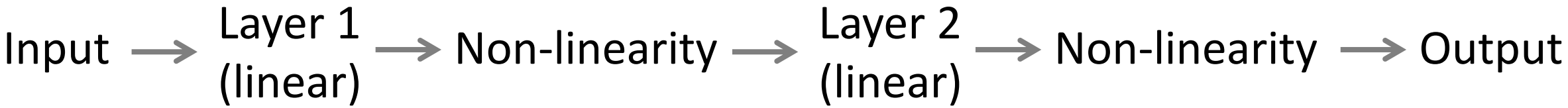
- 2 layers perceptron: universal approximator (Cybenko 1989)
- But potentially needs many neurons -> more layers, deeper networks (e.g. Bengio et al 07, Montufar et al 2014)

Multi-layer perceptron (MLP)



- 2 layers perceptron: universal approximator (Cybenko 1989)
- But potentially needs many neurons -> more layers, deeper networks (e.g. Bengio et al 07, Montufar et al 2014)

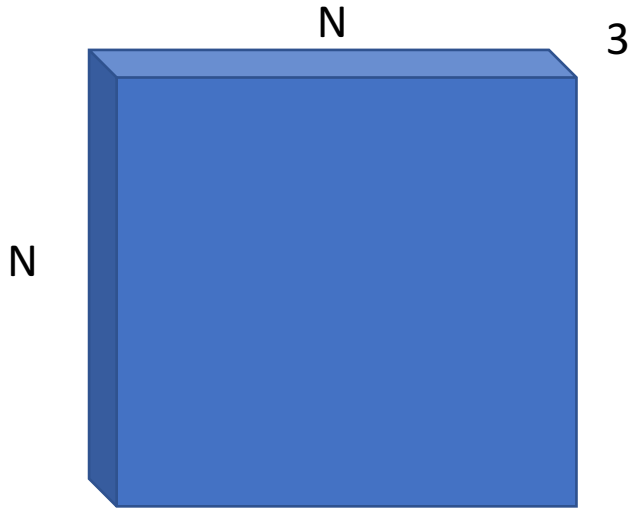
Abstraction



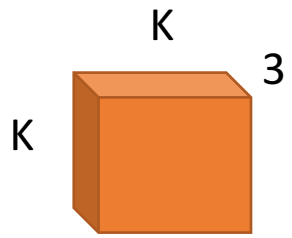
Avoiding overfitting

- Lots of parameters !
  - Regularize (add L2 norm of the weights to the loss, also called weight decay)
  - Enforce structure in the design of the network: convolutions and poolings

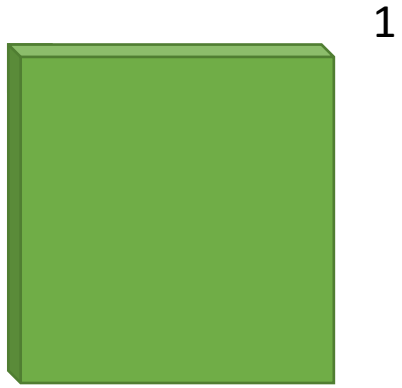
- Convolution for images



Image

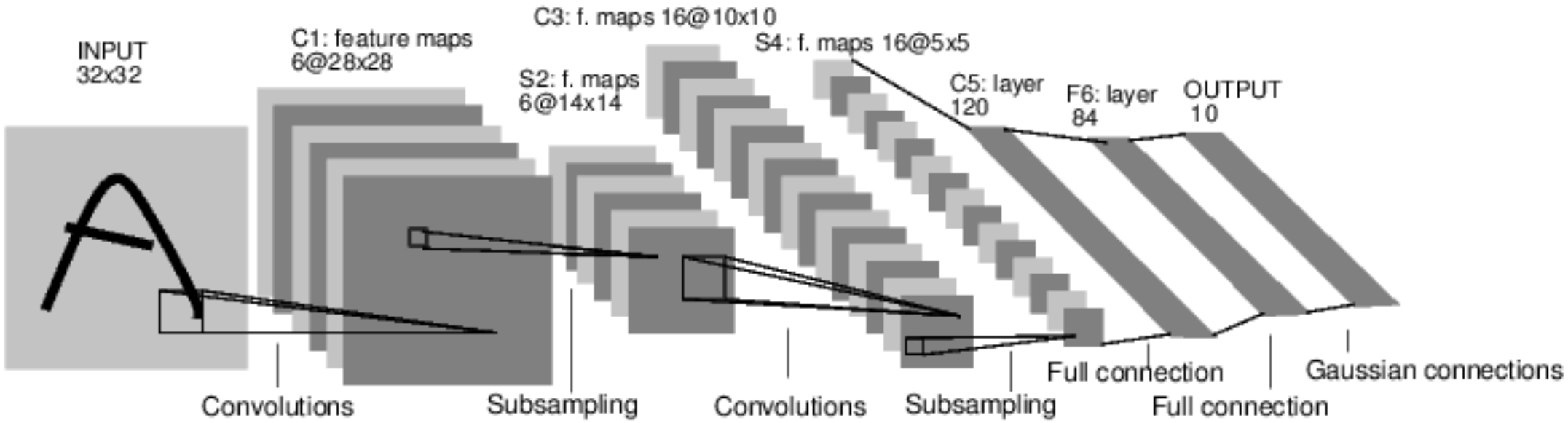


Weights



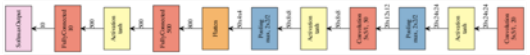
Output

Example: LeNet 5 (1998)



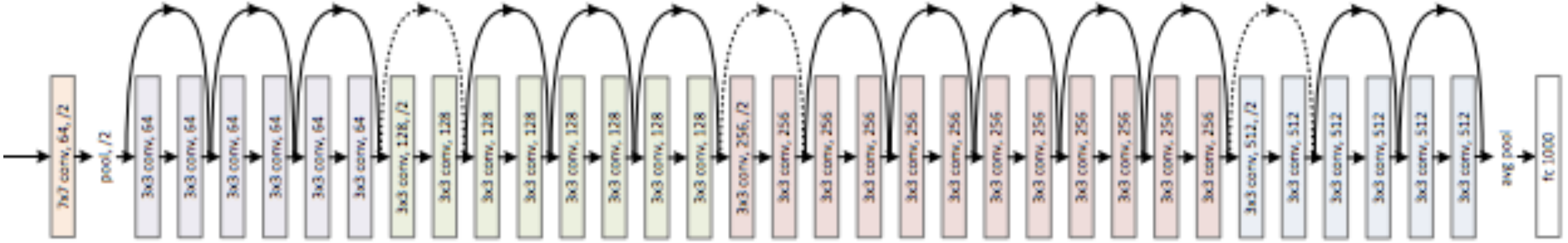
Example: ResNet (2015)

- LeNet 5



- AlexNet

- ResNet – 34 layer version visualized; state of the art 100s layers



Learning

- Non convex!
- Solution: not caring, simply perform SGD (Werbos 1974, LeCun 1998)

Note: every layer needs to be differentiable almost everywhere

- Stochastic Gradient Descend (SGD): same as gradient descent except the gradient is evaluated from a random inputs.

- How to evaluate the gradient?
  - Short answer: since everything is differentiable, just use chain rule
  - Long answer: Back Propagation algorithm

Back Propagation

Two step process:

- Forward pass: compute the X_i by

$$X_i = F_i(X_{i-1}, W_i)$$

- Backward: compute the $\frac{\partial E}{\partial W_i}$ and $\frac{\partial E}{\partial X_i}$ by

$$\frac{\partial E}{\partial W_i} = \frac{\partial E}{\partial X_i}\frac{\partial F_i(X_{i-1}, W_i)}{\partial W_i}$$

$$\frac{\partial E}{\partial X_{i-1}} = \frac{\partial E}{\partial X_i}\frac{\partial F_i(X_{i-1}, W_i)}{\partial X_{i-1}}$$

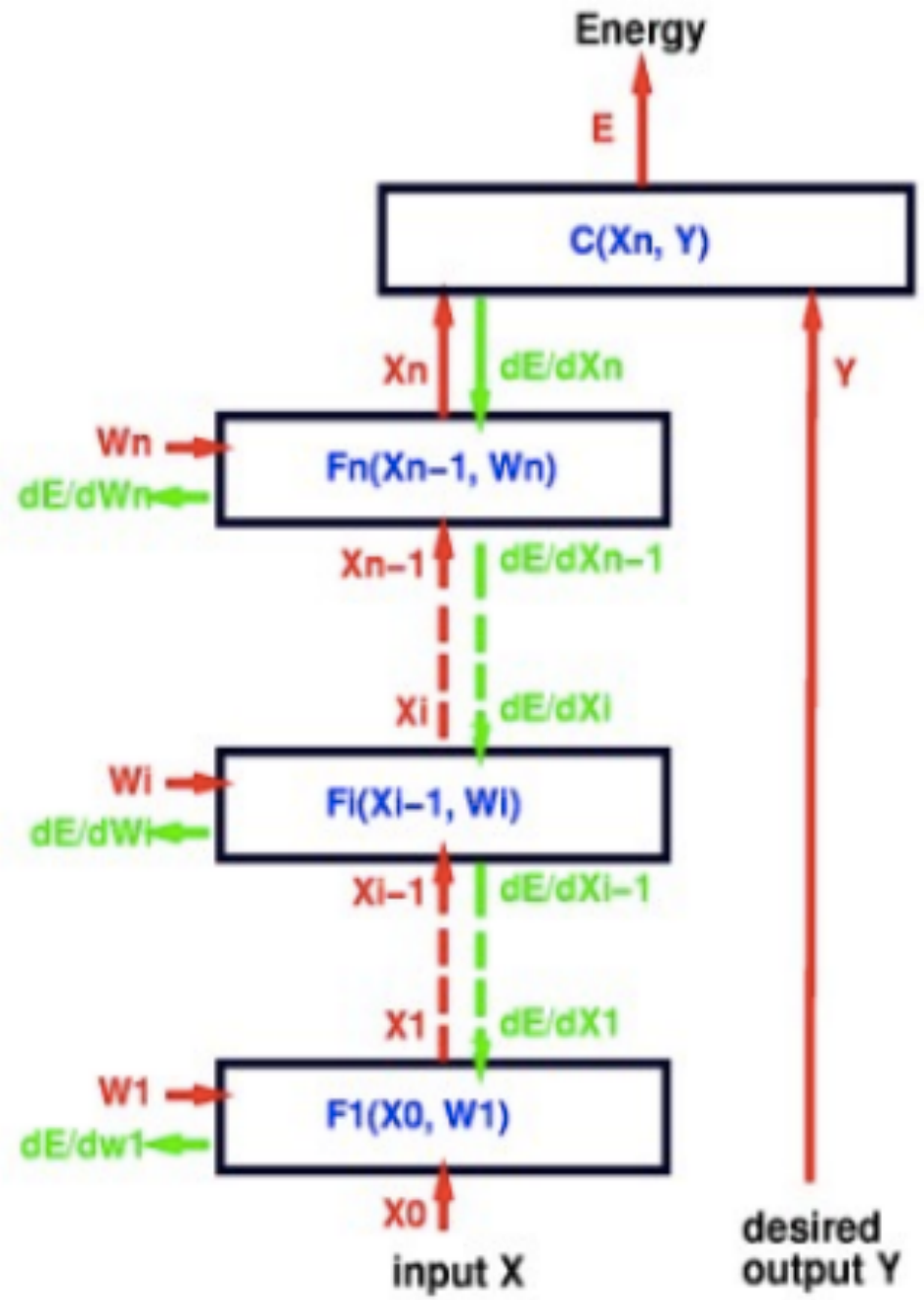
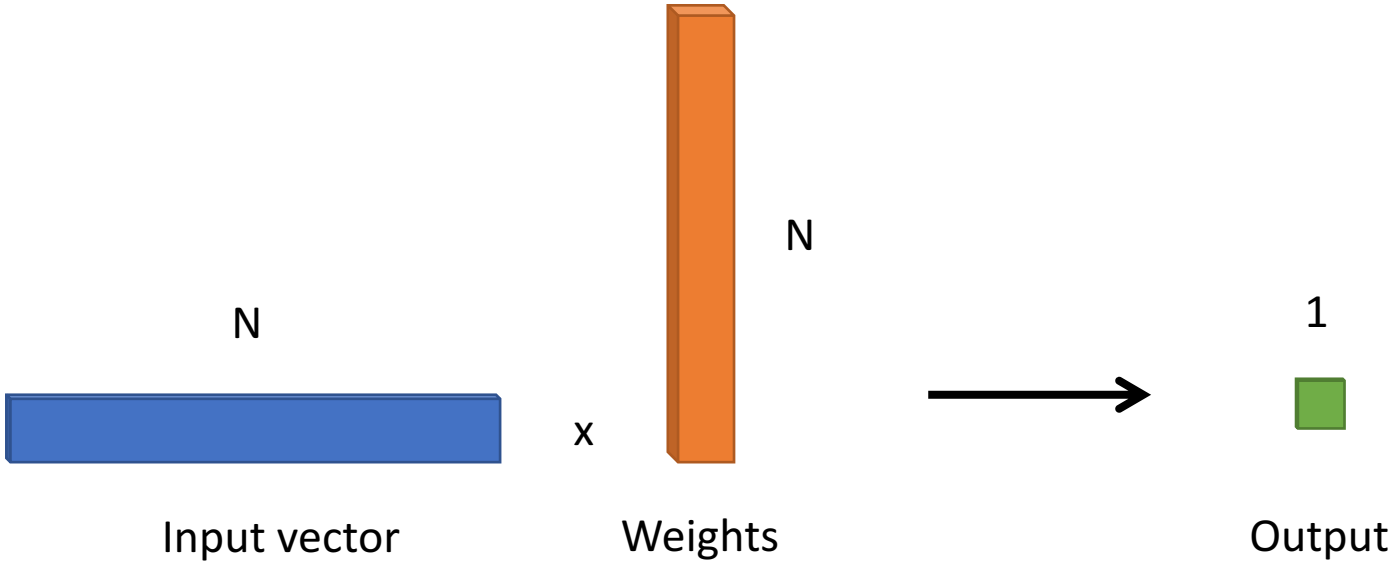


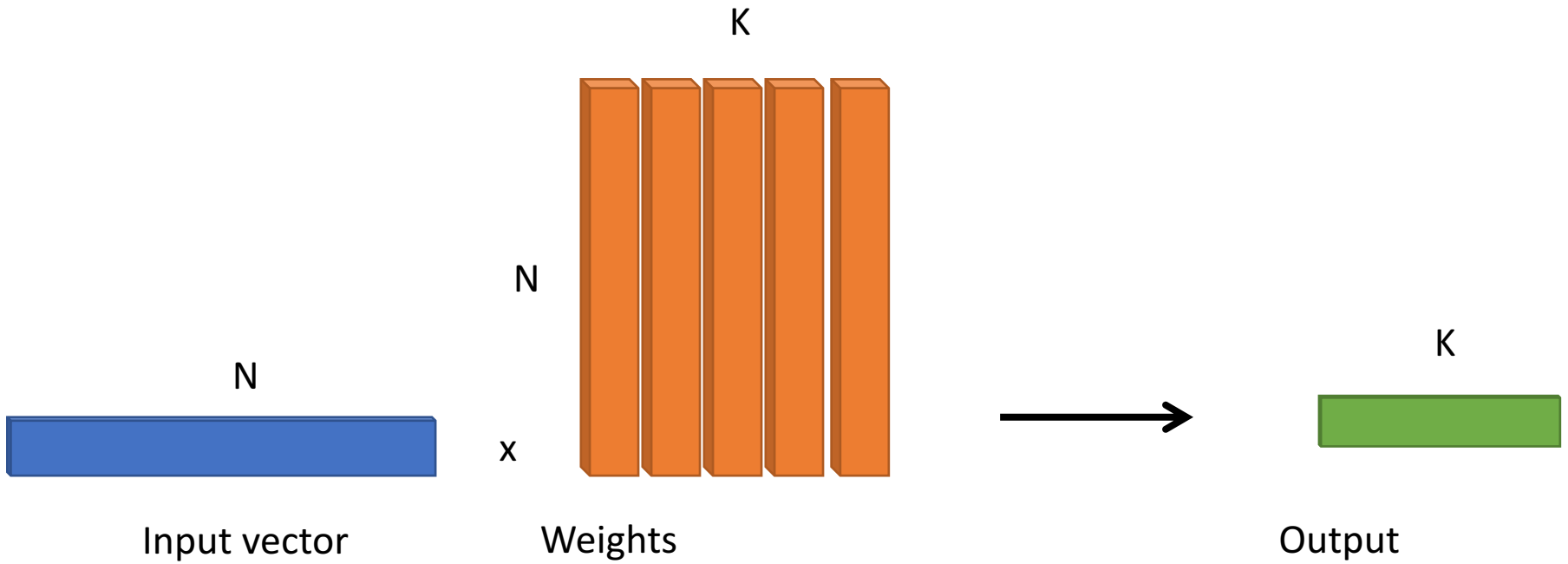
Diagram from Y. LeCun

What are the different layers?

Linear



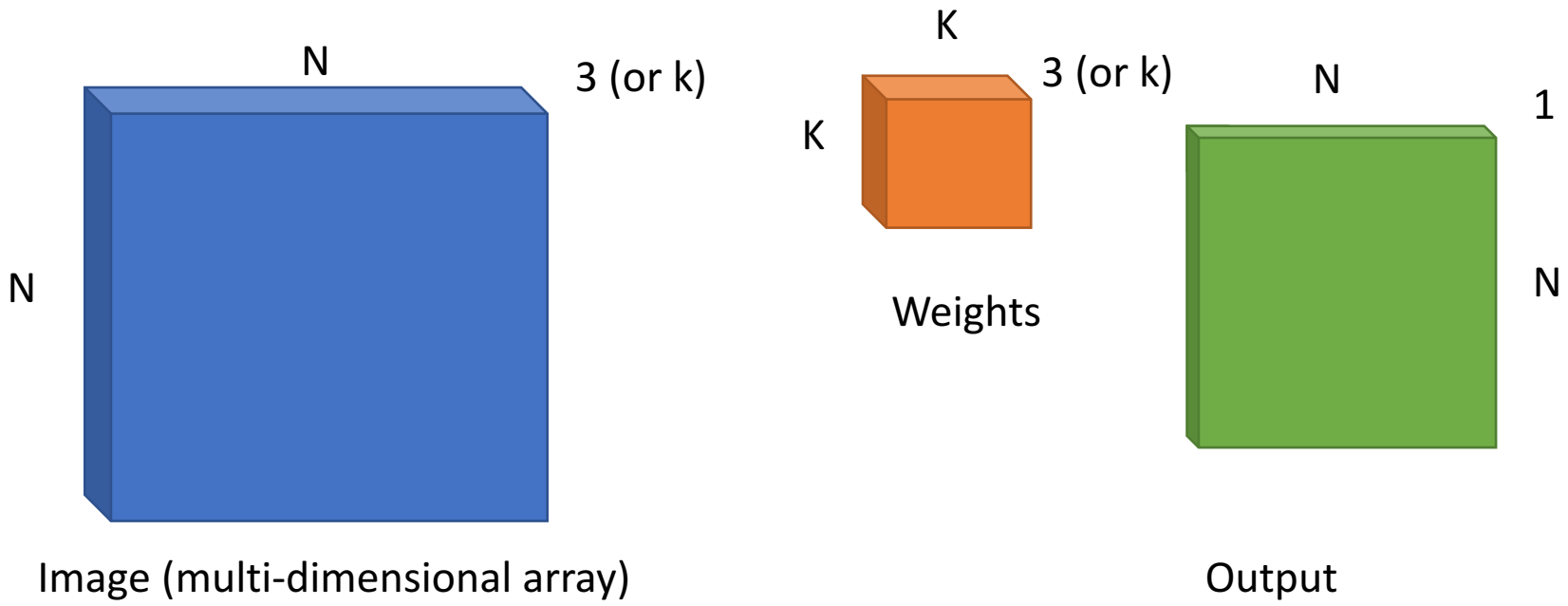
Linear



- Issue: lots of parameters

Convolution

- Sharing the weights / using the structure of the data



- Zero padding to keep the same size.
- Relation with linear layers -> ”fully convolutional” approach

Pooling

- Goal: aggregating information, gain invariance
- Example:
  - Max pooling: $z \rightarrow \max_i(z^i)$

    Ex: Image classification with fully convolutional approach

  - Average pooling: $z \rightarrow \frac{1}{N}\sum_{i=1}^{N} z^i$

Non linearities

- Sigmoid, tanh, ReLu, ”leaky” ReLu ($max(x, \epsilon x)$)
- In practice, all of them seem to work, but can make the networks harder to train.
- Lots of success with ReLu:
  - Avoid extremely small derivatives (e.g. of a sigmoid)
  - Leads to sparse outputs
  - Very simple derivative

Reshaping, concatenating, upsampling…

- Reshaping: e.g. need to interpret an array as a vector or vice-versa
- Concatenating: e.g. a network first treating two images independently then together
- Upsampling: e.g. a generative network

-> very simple derivative, but very useful in practice

Many more

- As long as you can compute forward and backward (fast)
- E.g. normalization, cropping…

- NN can be seen just as a general modular framework, you can define new modules, organize them etc.

Practical problems

- Data size:
  get lots of memory, SSD, use efficient database structure
- Speed: use GPUs, parallel data loading
- Network size: get lots of memory on your GPU or/and use several GPUs

Good news: you don’t have to do it!

Many ready to use and efficient frameworks are available (Pytorch, Torch, TensorFlow, MatConvNet, Caffe, …)

Outline

1. Introduction to supervised and Deep Learning
   (focus on Computer Vision)

2. Introduction to reinforcement learning
   (focus on learning Pong)

Disclaimer

- Not my main research area, but ”simple” and fun application of NN to playing games.

- Inspiration for these slides :
  - Online book from Sutton and Barto, Reinforcement Learning: An Introduction
  - Andrej Karpathy blog post http://karpathy.github.io/2016/05/31/rl/

- Atari games with RL seminal paper: Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. Human-level control through deep reinforcement learning. *Nature*, 2015

Reinforcement learning

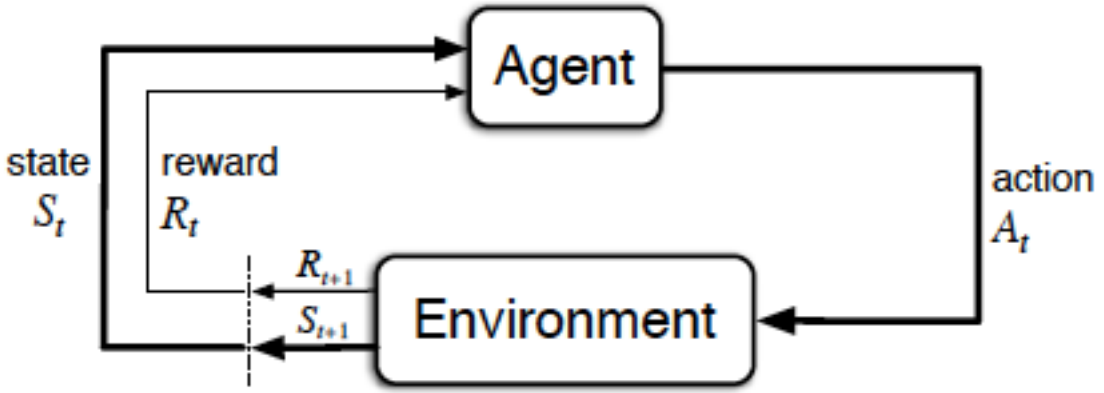
- No prior data / ideal behavior available
  -> different from supervised learning

- Your algorithm has to choose a set of actions, that result (in a non deterministic manner) in a final reward.
  -> optimize a reward, different from unsupervised learning

- It’s typically impossible to explore exhaustively the set of actions.

Markov Decision Process

- Formalization: Markov decision process, relate sensation, action and goal

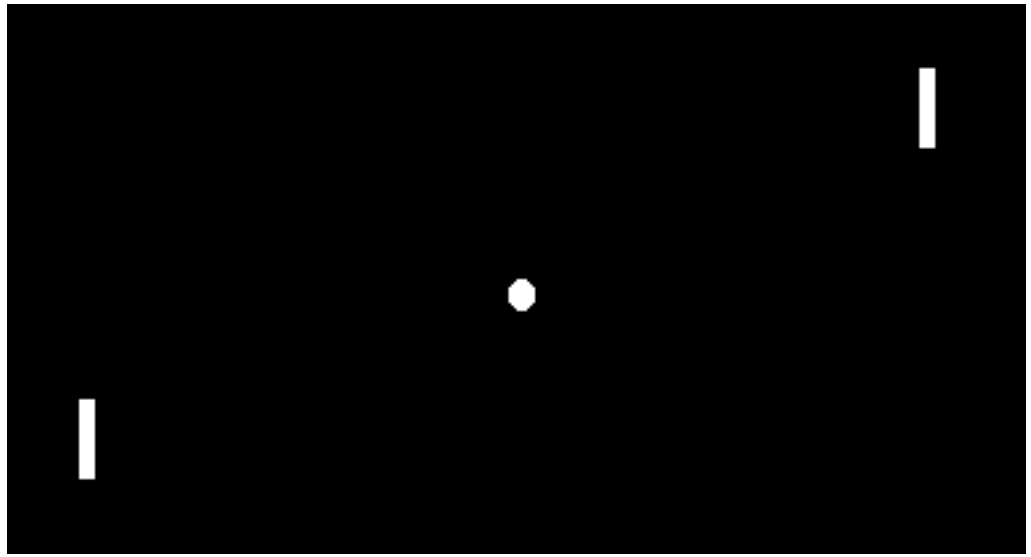


From Sutton and Barto, 2018

Figure 3.1: The agent–environment interaction in a Markov decision process.

- RL objects: environment, state, action, agent, policy, reward signal, value function, environment model

Example: pong



Gif from http://karpathy.github.io/2016/05/31/rl/

Reinforcement learning

- Goal: minimize regret (difference with optimal policy)
- Trial and error search, exploration / exploitation trade-off
- Often stochastic environment/rewards
- How to map situations to actions?

Policy Gradient (PG)

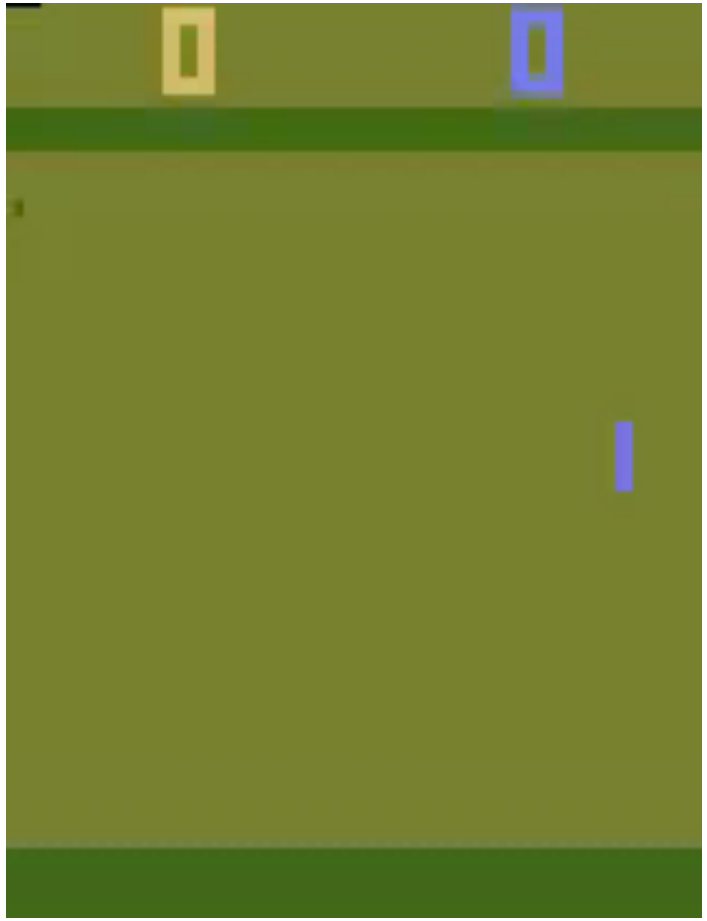
One possible approach:

- We use a policy network (a standard MLP) that predicts a probability for each action (e.g. going up in pong)

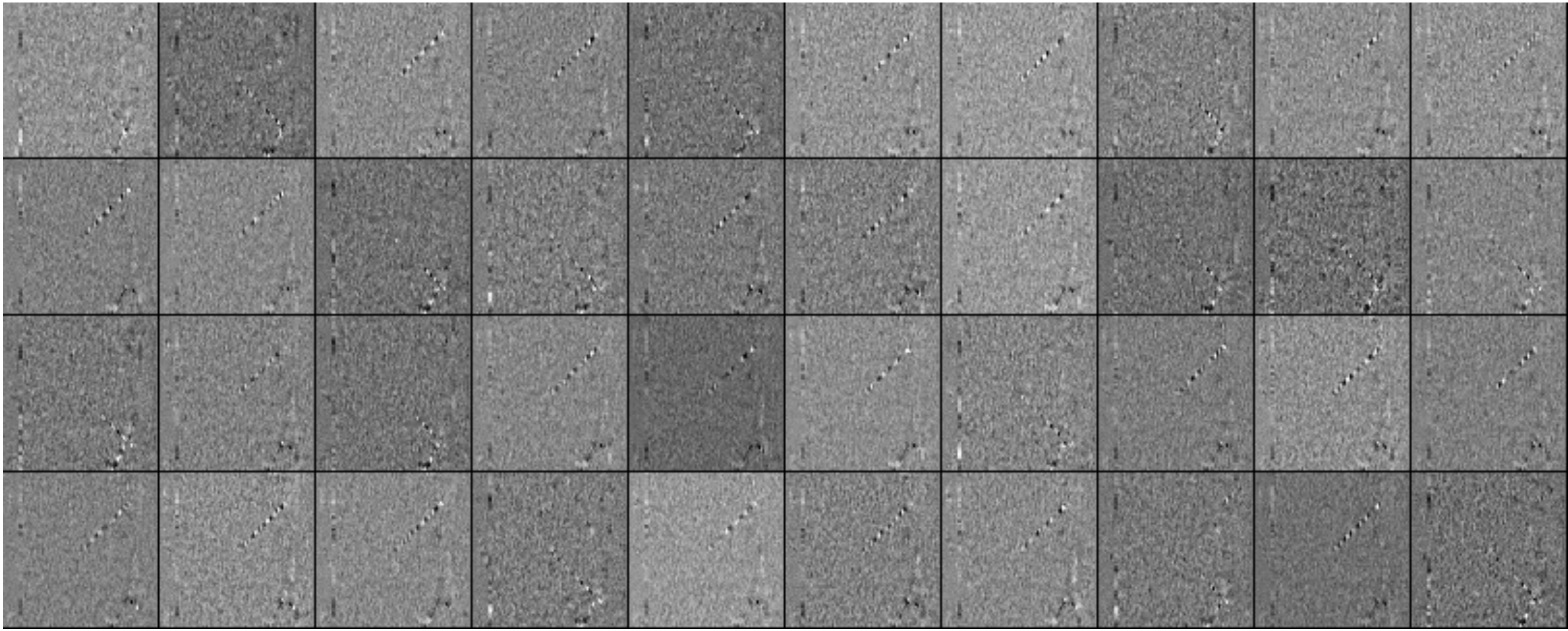
- To train it, we sample actions from the current policy, accumulate the gradients, and do a gradient step depending on the final reward (winning or loosing the game in Pong)

- It will work … if you have a reasonnable probability to win by chance

Example Pong



Learnt weights



Conclusion

- NN and particularly CNN work remarkably well for high dimensional problems, despite their simplicity, when enough data is available
- RL works well when there is a good probability to succeed by chance (e.g. better than human / classical AI on most Atari games)
- Fast moving area!