INF367 25H: Selected Topics in Artificial Intelligence

Diamonds and Rust in the AI Treasure Chest

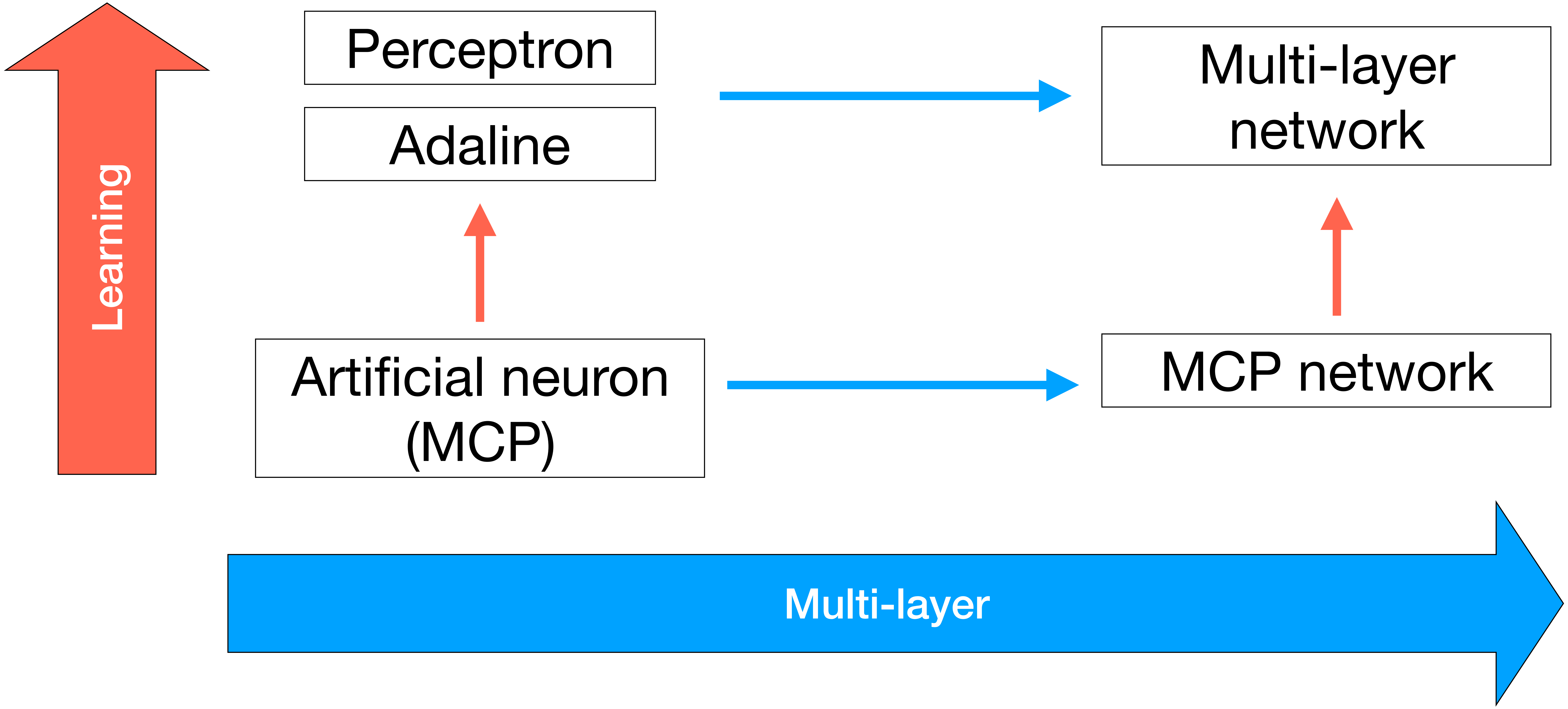
Dario Garigliotti - UiB - 13.10.2025

Plan for today

- Recap from last lecture
- Hopfield network
- Cybenko’s universal approximation

Context before
Chapter 8

Context


Learning
Perceptron
Adaline
Multi-layer network
Artificial neuron (MCP)
MCP network
Multi-layer

Hopfield network

Hopfield network

- Most biologically-inspired systems for information processing are supposed to have these properties:
  - Patterns are stored in a set of units, e.g. neurons
  - Recovery of patterns is robust to noise
  - The neural activity is mostly binary
  - Information processing is distributed and modular

Hopfield network

- Hopfield network takes inspiration from physics
  - And, like the perceptron, from artificial neurons and Hebbian learning
- It seeks to approach some brain-related aspect, e.g., associative memory
- It proposes a network to learn “memories” (patterns), as local minima of some energy function, that are retrievable upon feeding a new data instance to it
  - Such a prediction request perturbs the network, increasing the energy, that must be lowered back to stability reaching the closest pattern as minimum

Cybenko’s universal approximation

Cybenko’s universal approximation

- A theorem, proved by contradiction, …
  - About the existence of a neural network
    - with a single hidden layer
    - such that with a sufficient number of hidden neurons
    - can approximate any true input-output mapping function *f(x)*
- Intuition: approximate *f(x)* as a sum of contributions of (as many as needed) non-linearly activated output neurons

Learner vs Predictor/Model

- Some phenomenon *ph* in the real world *RW*
- Some data distribution $\cal{Distrib}$ underlying *ph* according to the true function f that “maps input into output” in *RW*
- A dataset $D \sim \cal{Distrib}$, $D = \{(x_i, y_i) : i = 1..n\}$
- An approximator h^* in $H = \{h : h \text{ approximates } f\}$ = set of models or predictors
- In relation with H, a learner L learns h’s from D: $L(D)=h$
    - E.g. H1 = {decision trees…}, H2 = {decision forests…}, H3 = {neural nets…}