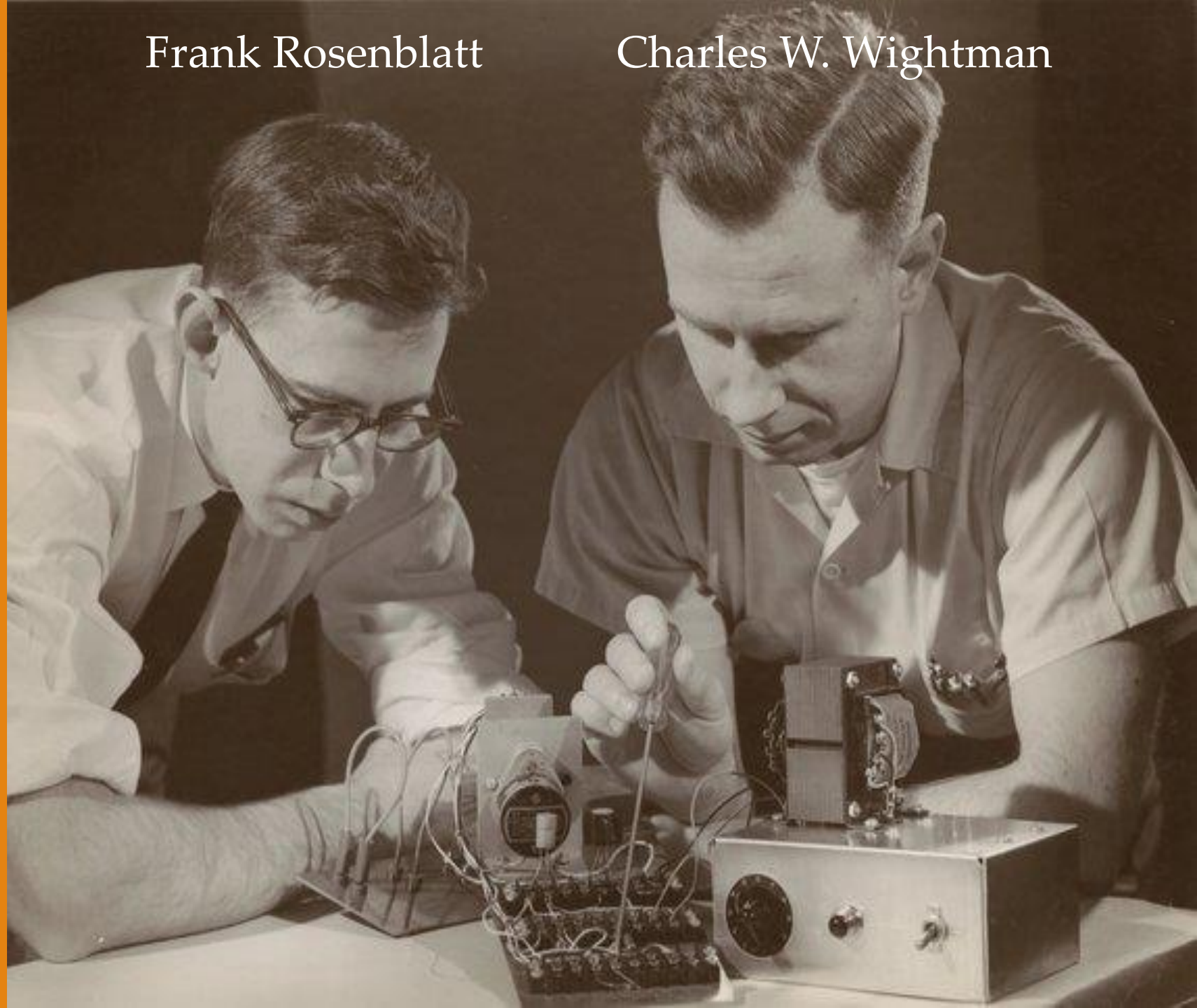
A brief history of neural networks & perceptrons



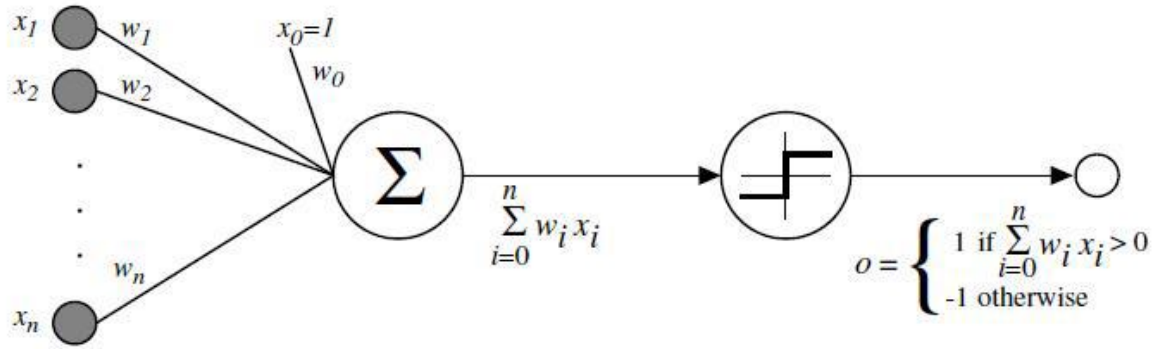
First appearance (roughly)

Perceptrons

- Rosenblatt proposed perceptrons for binary classifications
- A model comprising one weight w_i per input x_i
- Multiply weights with respective inputs and add bias $x_0 = +1$

$$y = \sum_{j=1}^{n} w_j x_j + x_0$$

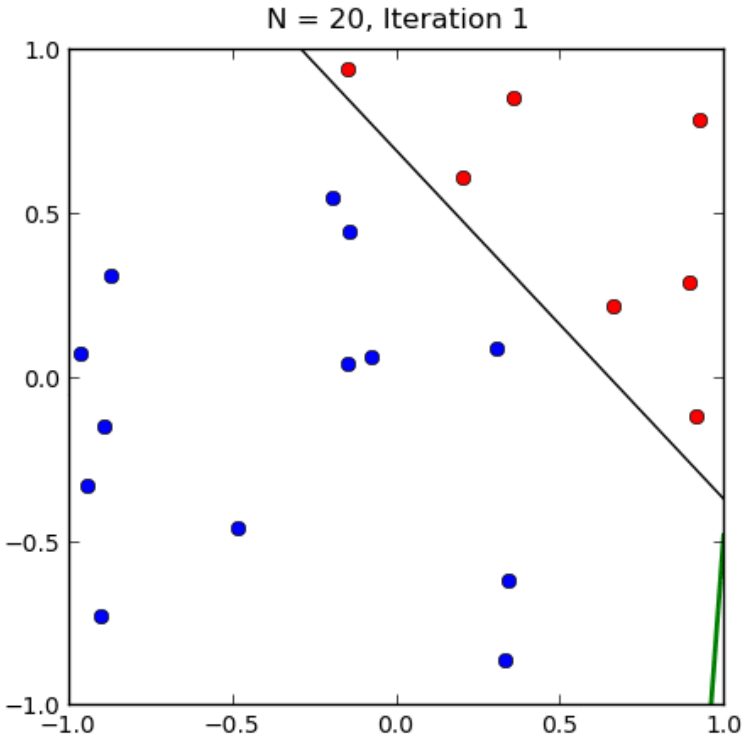
- If score y positive then return 1, otherwise -1



Training a perceptron

- Main innovation: a learning algorithm for perceptrons

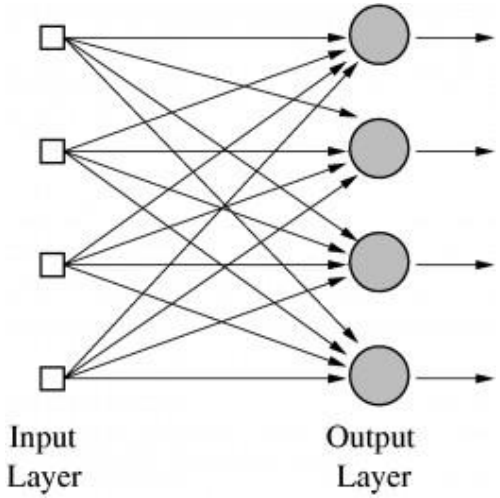
Perceptron learning algorithm	**Comments**
1. Set $w_j \leftarrow$ random	
2. Sample new (x_i, l_i)	New train image, label
3. Compute $y_i = [\![\sum w_i x_{ij} > 0]\!]$	$[\![\cdot]\!]$: indicator function
4. If $y_i < 0, l_i > 0 \rightarrow w_i = w_i + \eta \cdot x_i$	Score too low. Increase weights!
5. If $y_i > 0, l_i < 0 \rightarrow w_i = w_i - \eta \cdot x_i$	Score too high. Decrease weights!
6. Go to 2	Repeat till happy ☺



From a single output to many outputs

- Perceptron was originally proposed for binary decisions
- What about multiple decisions, e.g. digit classification?
- Append as many outputs as categories → Neural network

4-way neural network



XOR & 1-layer perceptrons

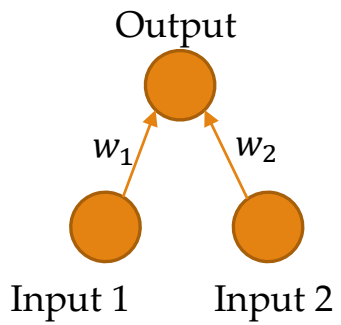
- The original perceptron has trouble with simple non-linear tasks though
- E.g., imagine a NN with two inputs that imitates the "exclusive-or" (XOR)

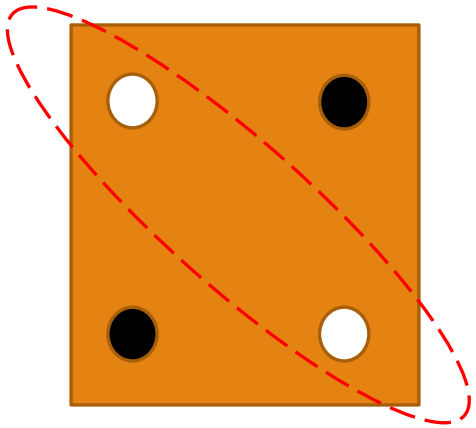
Input 1	Input 2	XOR
1	1	0
1	0	1
0	1	1
0	0	0

$$0 \cdot w_1 + 0 \cdot w_2 < \theta \rightarrow 0 < \theta$$
$$0 \cdot w_1 + 1 \cdot w_2 > \theta \rightarrow w_2 > \theta$$
$$1 \cdot w_1 + 0 \cdot w_2 > \theta \rightarrow w_1 > \theta$$
$$1 \cdot w_1 + 1 \cdot w_2 < \theta \rightarrow w_1 + w_2 < \theta$$

$$w_1 + w_2 > 2\theta$$
$$w_1 + w_2 < \theta$$

Inconsistent



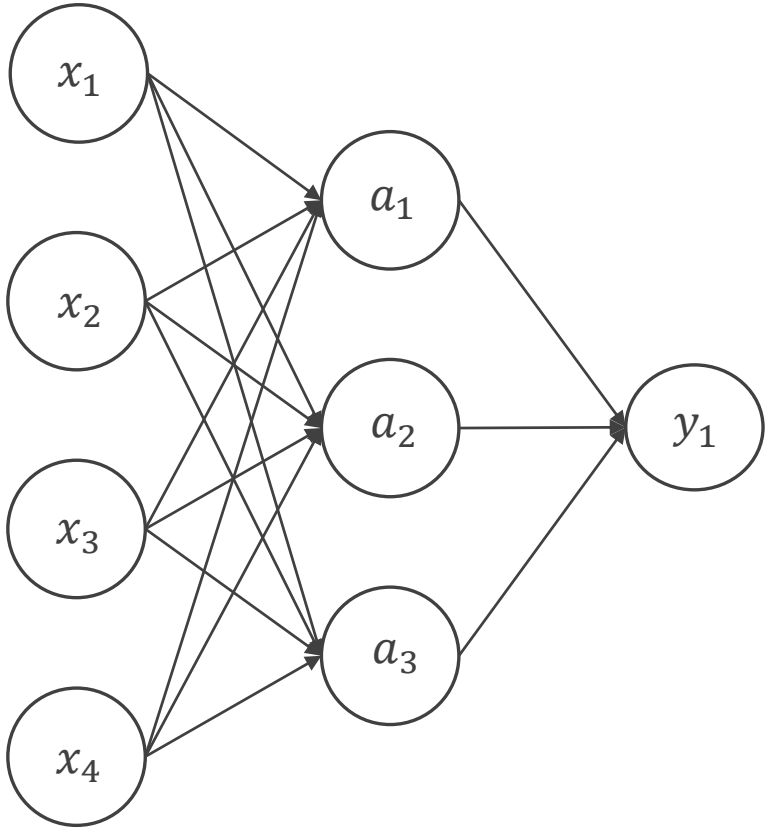


No line can separate the white from the black

Minsky and Papert, "Perceptrons", 1969

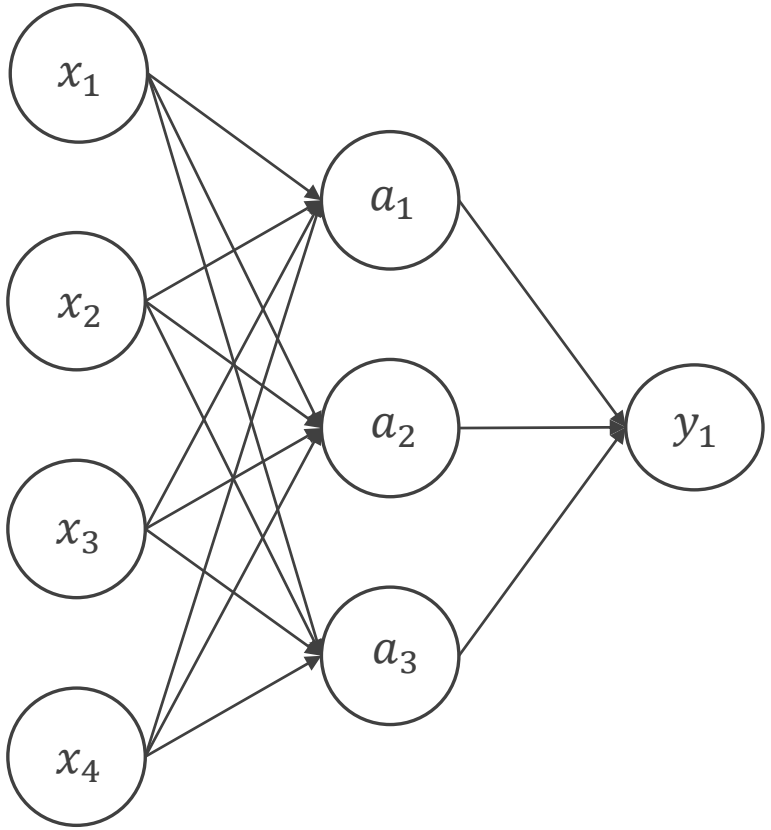
Multi-layer perceptrons to the rescue

- Minsky **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with **1-layer** perceptrons
- Multi-layer perceptrons (MLP) can solve XOR
  - One layer's output is input to the next layer
  - Add nonlinearities between layers, e.g., sigmoids
  - 9 years earlier Minsky built such a multi-layer perceptron
- Problem: how to train a multi-layer perceptron?
- Rosenblatt's algorithm not applicable. Why?

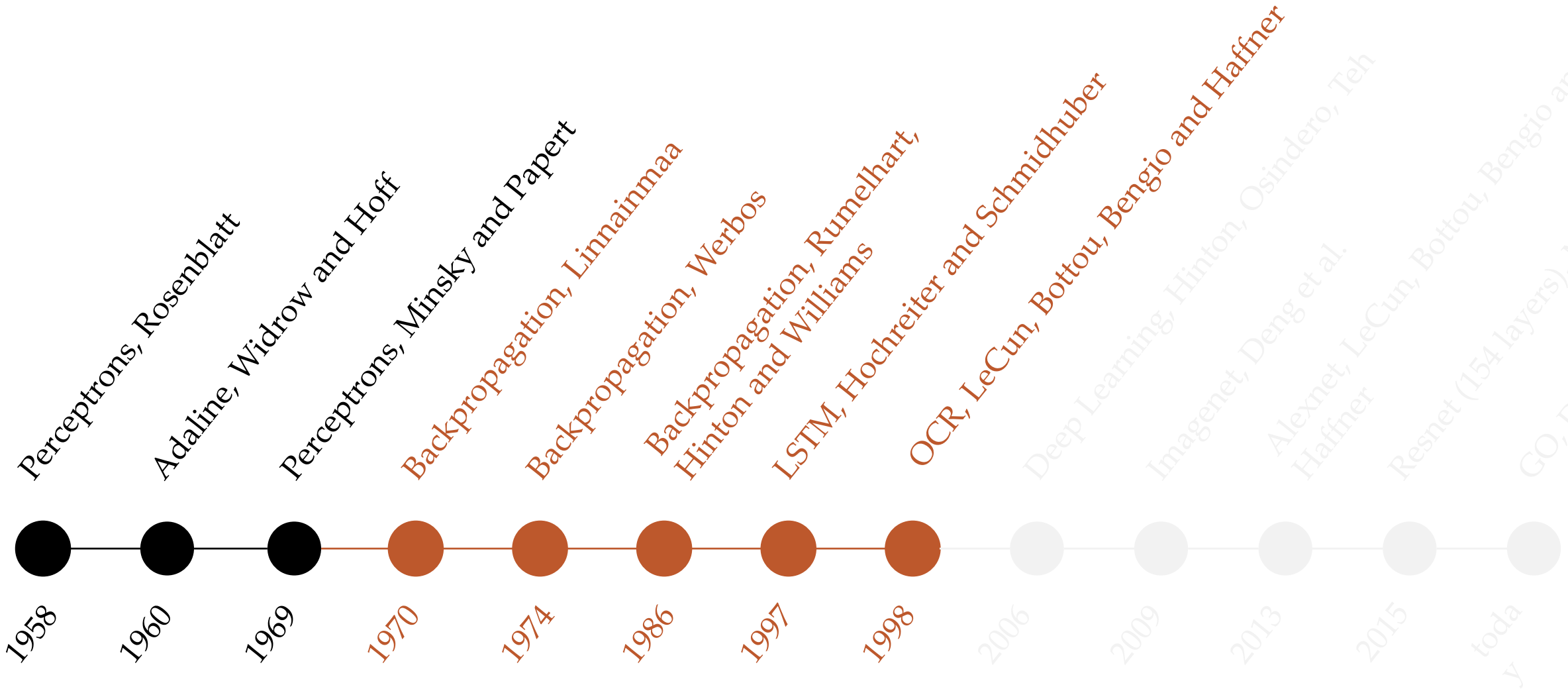


Multi-layer perceptrons to the rescue

- Minksy **never said** XOR cannot be solved by neural networks
  - Only that XOR cannot be solved with **1-layer** perceptrons
- Multi-layer perceptrons (MLP) can solve XOR
  - One layer's output is input to the next layer
  - Add nonlinearities between layers, e.g., sigmoids
  - 9 years earlier Minsky built such a multi-layer perceptron
- Problem: how to train a multi-layer perceptron?
- Rosenblatt's algorithm not applicable. Why?
  - Learning depends on "ground truth" l_i for updating weights
  - For the intermediate neurons a_j there is no "ground truth"
  - The Rosenblatt algorithm cannot train intermediate layers



The "AI winter" despite notable successes



The first "AI winter"

- What everybody thought

"If a perceptron cannot even solve XOR, why bother?"

- Results not as promised (too much hype!)
  → no further funding
  → AI Winter
- Still, significant discoveries were made in this period
  - Backpropagation → Learning algorithm for MLPs
  - Recurrent networks → Neural Networks for infinite sequences

The second "AI winter"

- Concurrently with Backprop and Recurrent Nets, new and promising Machine Learning models were proposed
- Kernel Machines & Graphical Models
  - Similar accuracies with better math and proofs and fewer heuristics
  - Neural networks could not improve beyond a few layers