Introduction to Deep Learning

6. Multilayer Perceptron

STAT 157, Spring 2019, UC Berkeley

Mu Li and Alex Smola

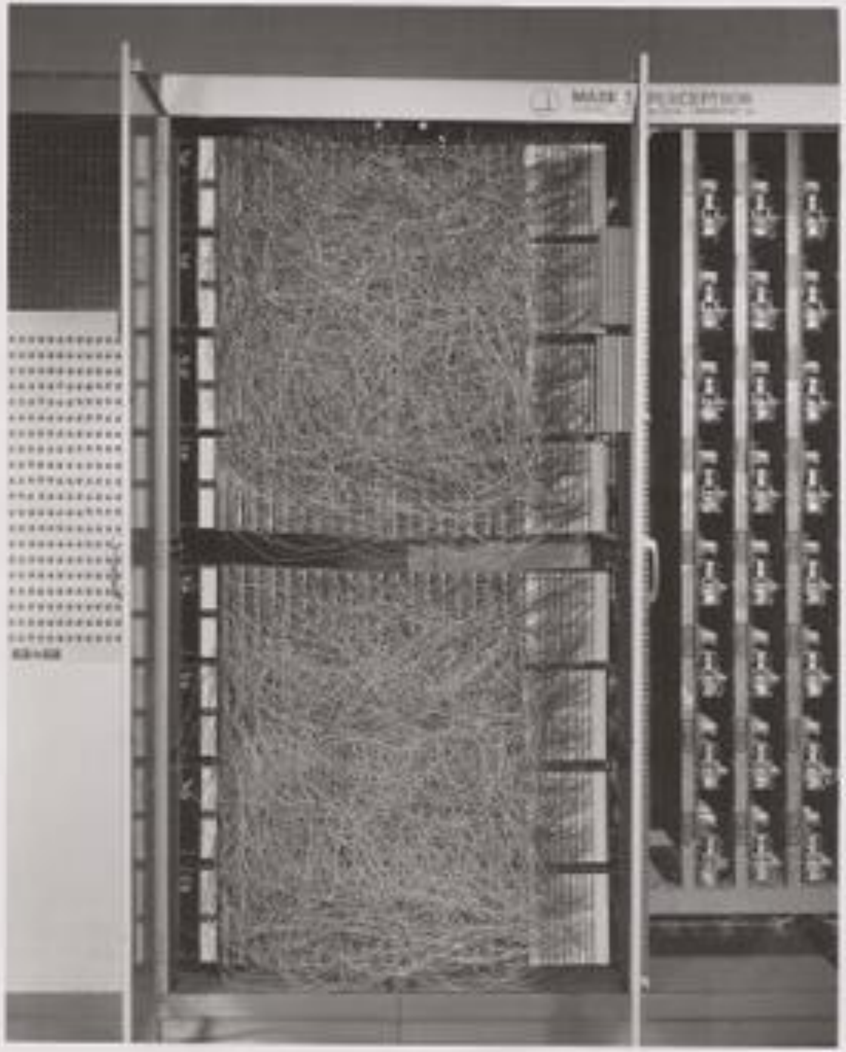
courses.d2l.ai/berkeley-stat-157

Outline

- **Single Layer Perceptron**
  - Decision Boundary
  - XOR is hard
- **Multilayer Perceptron**
  - Layers
  - Nonlinearities
  - Computational Cost

Perceptron

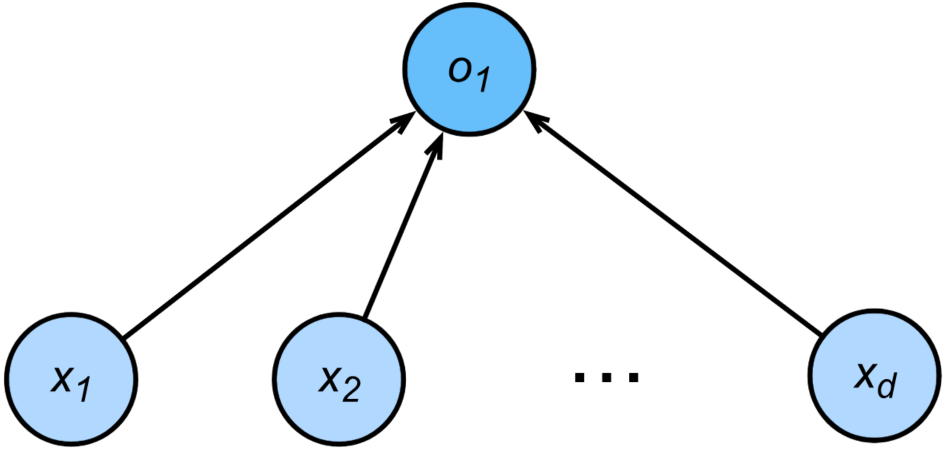
Mark I Perceptron, 1960
(wikipedia.org)



Perceptron

- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \qquad \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

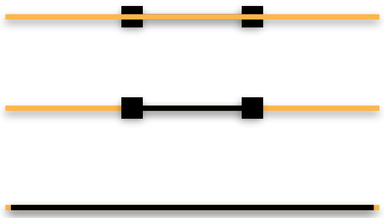


Perceptron

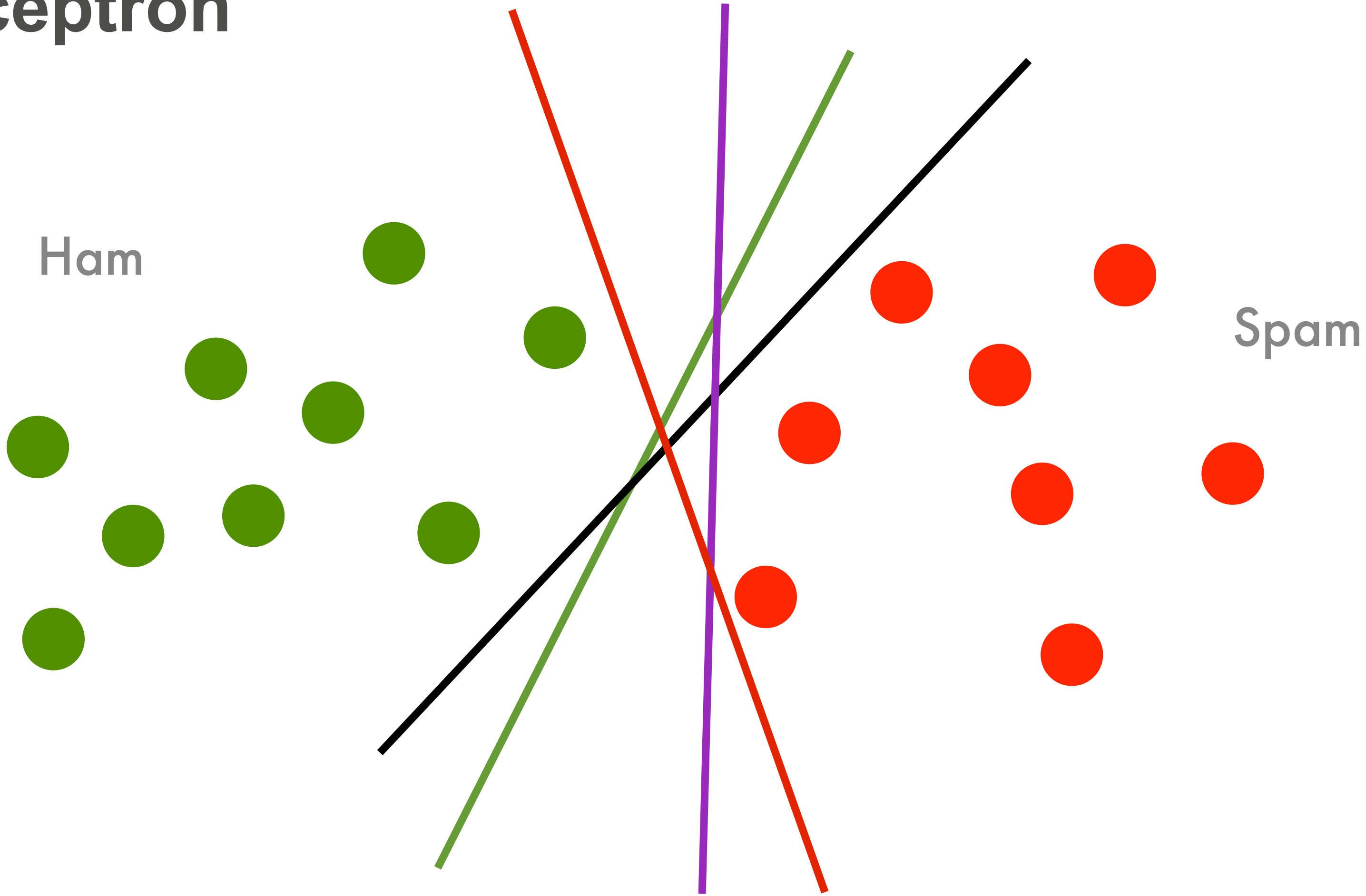
- Given input $\mathbf{x}$, weight $\mathbf{w}$ and bias b, perceptron outputs:

$$o = \sigma\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \qquad \sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

- Binary classification (0 or 1)
  - Vs. scalar real value for regression
  - Vs. probabilities for logistic regression



Perceptron

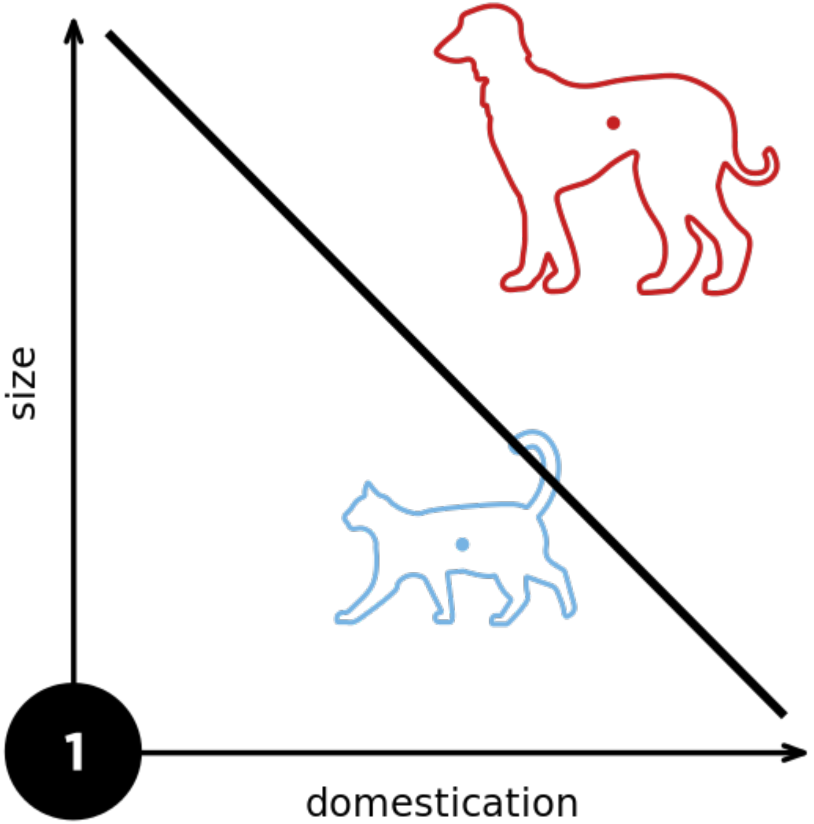


Training the Perceptron

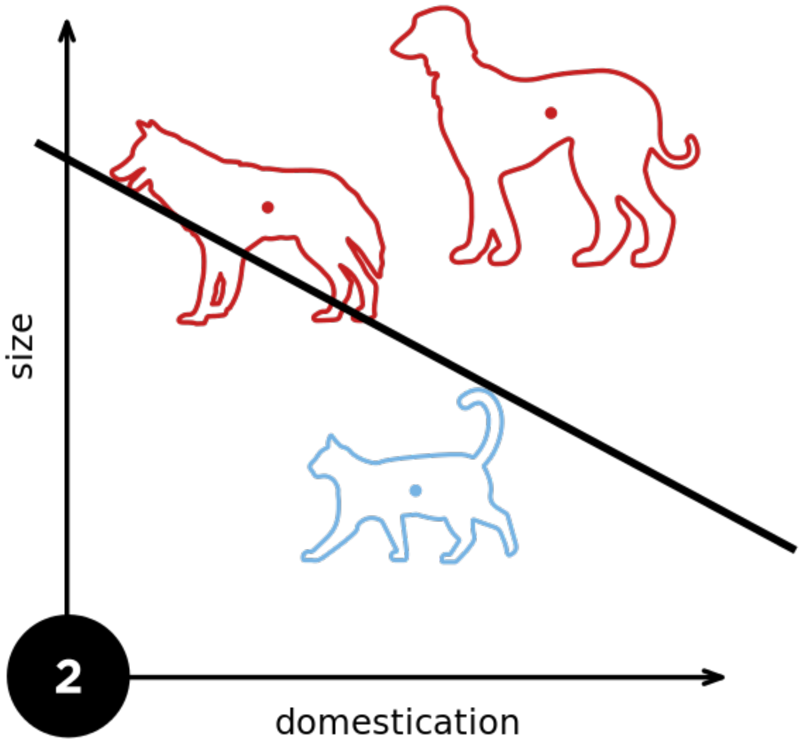
$$
\begin{aligned}
&\textbf{initialize } w = 0 \text{ and } b = 0\\
&\textbf{repeat}\\
&\quad \textbf{if } y_i \left[\langle w, x_i \rangle + b\right] \leq 0 \textbf{ then}\\
&\qquad w \leftarrow w + y_i x_i \text{ and } b \leftarrow b + y_i\\
&\quad \textbf{end if}\\
&\textbf{until } \text{all classified correctly}
\end{aligned}
$$

Equals to SGD (batch size is 1) with the following loss

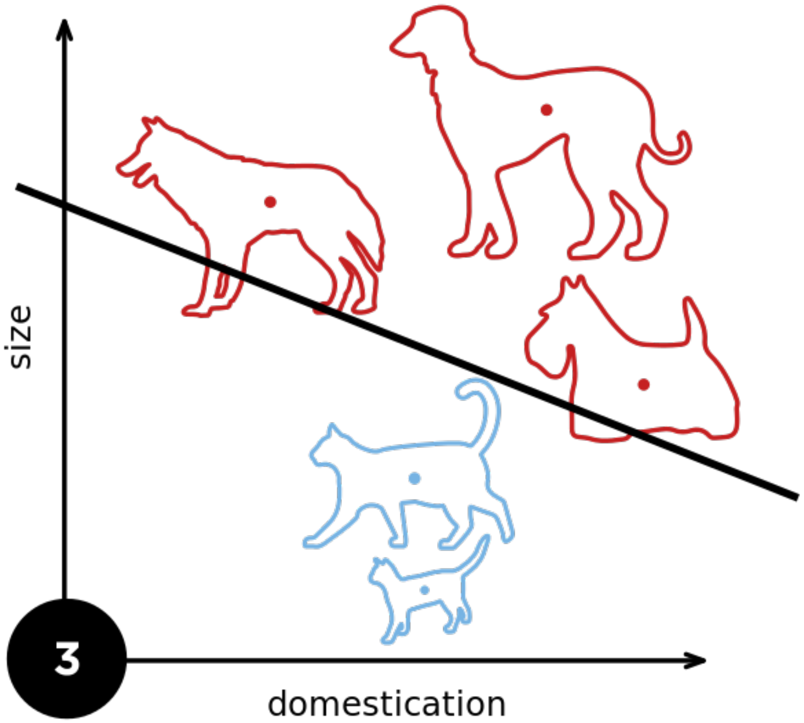
$$\ell(y, \mathbf{x}, \mathbf{w}) = \max(0, -y\langle \mathbf{w}, \mathbf{x} \rangle)$$



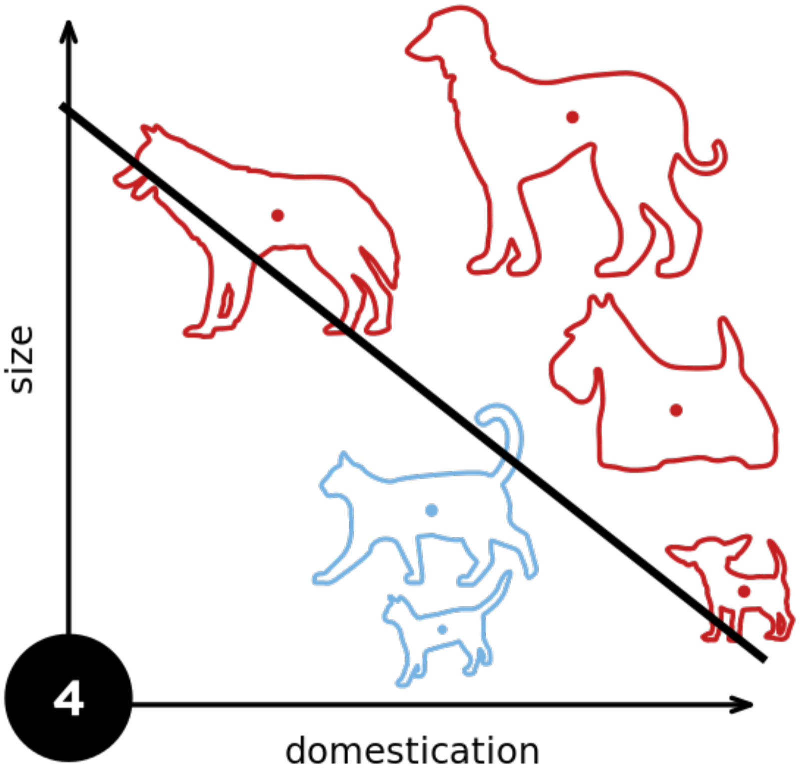
From wikipedia



From wikipedia



From wikipedia



From wikipedia

Convergence Theorem

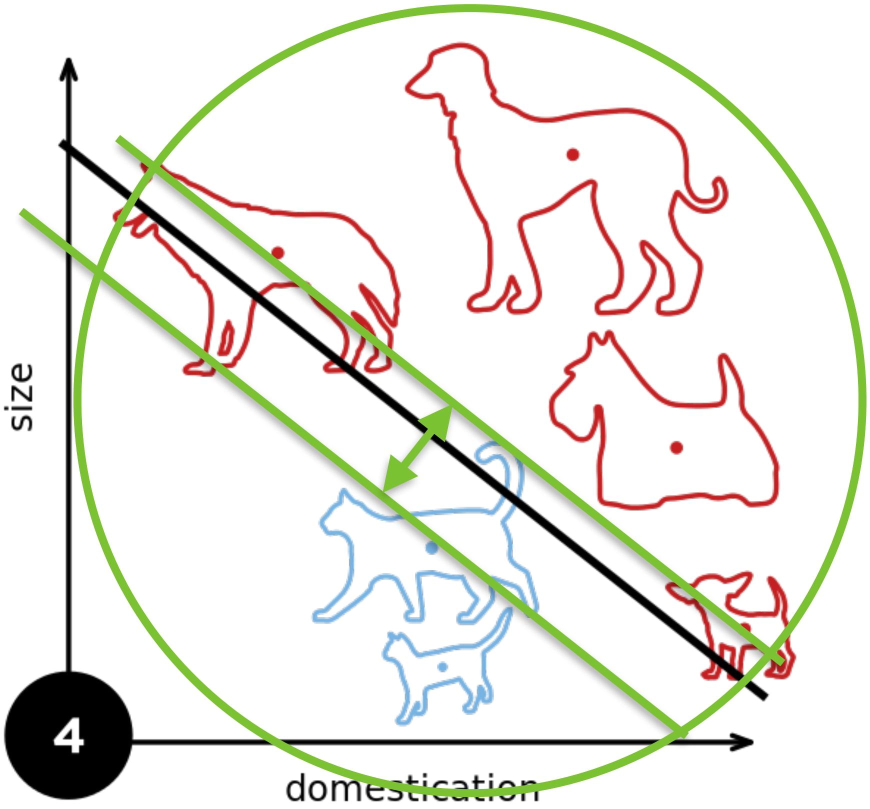
- Radius r enclosing the data
- Margin ρ separating the classes

$$y(\mathbf{x}^\top \mathbf{w} + b) \geq \rho$$

for $\|\mathbf{w}\|^2 + b^2 \leq 1$

- Guaranteed that perceptron will converge after

$$\frac{r^2 + 1}{\rho^2} \text{ steps}$$



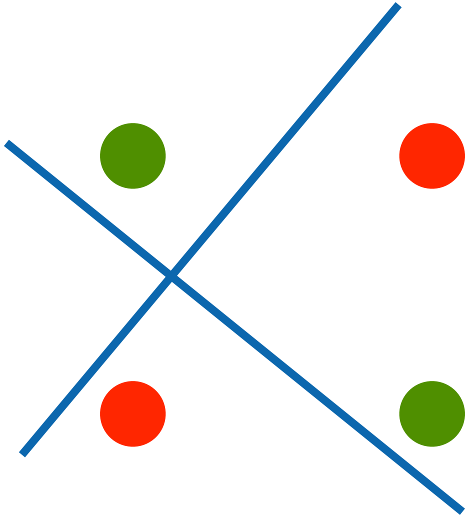
Consequences

- Only need to store errors. This gives a compression bound for perceptron.
- Fails with noisy data

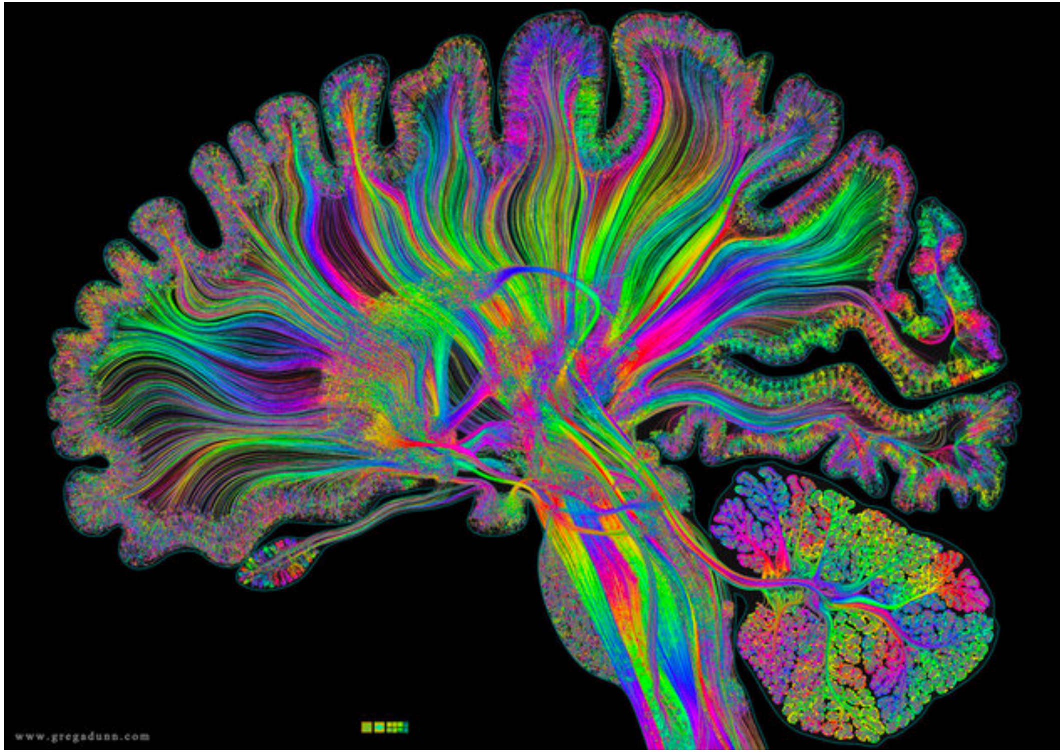
do NOT train your avatar with perceptrons

XOR Problem (Minsky & Papert, 1969)

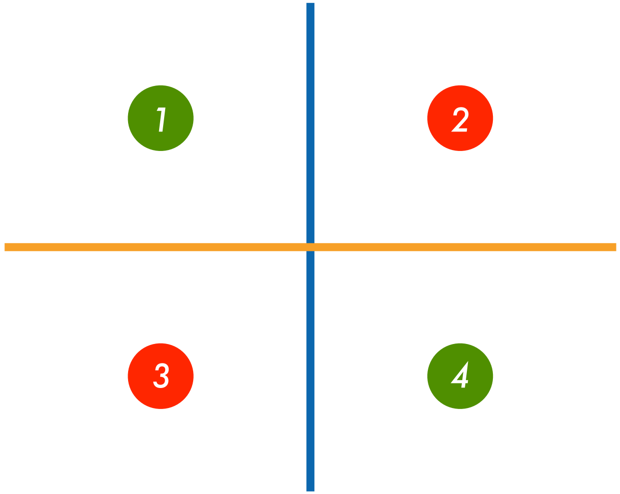
The perceptron cannot learn an XOR function
(neurons can only generate linear separators)



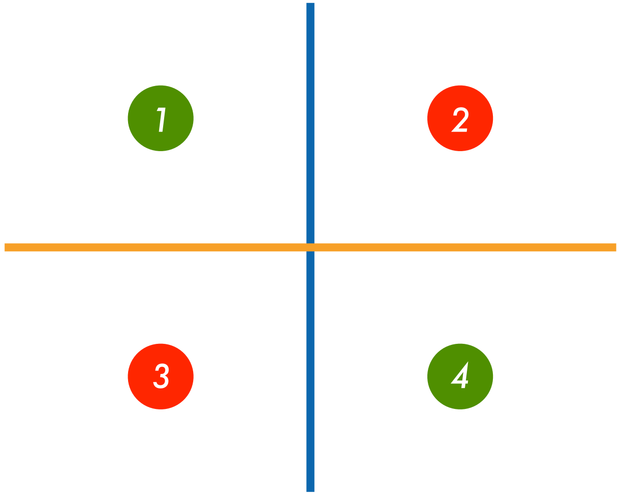
Multilayer Perceptron



Learning XOR

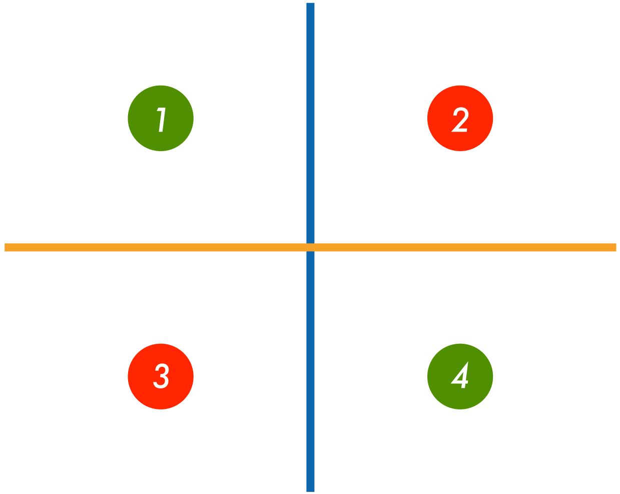


Learning XOR

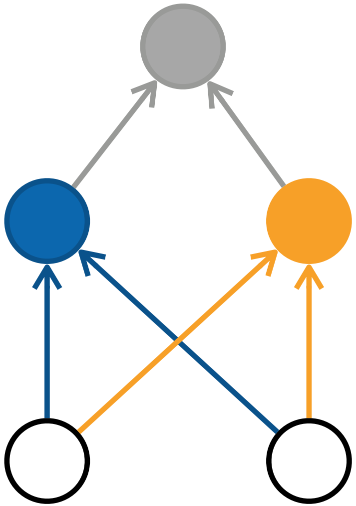


	1	2	3	4
	+	-	+	-
	+	+	-	-
product	+	-	-	+

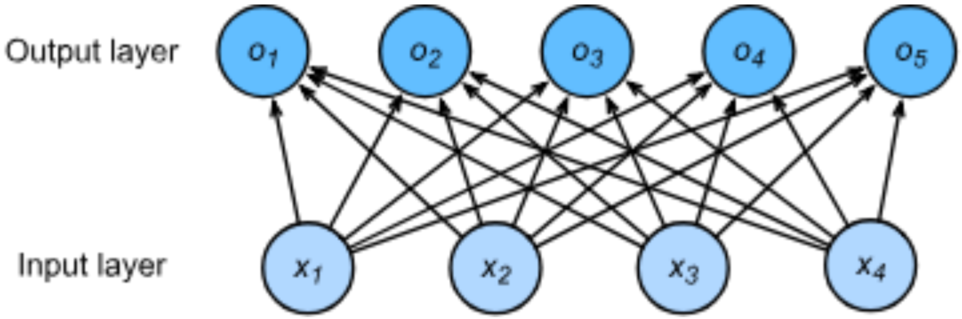
Learning XOR



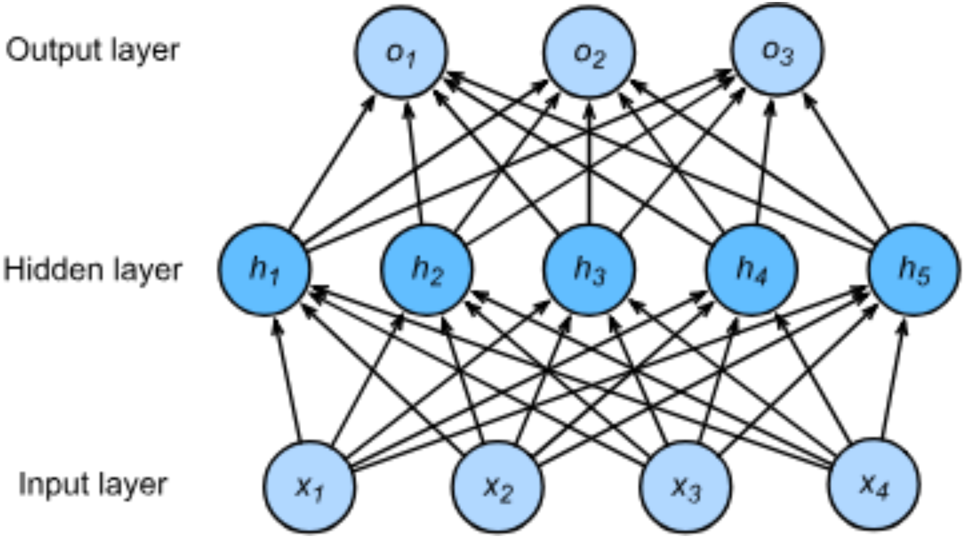
	1	2	3	4
	+	-	+	-
	+	+	-	-
product	+	-	-	+



Single Hidden Layer



Single Hidden Layer



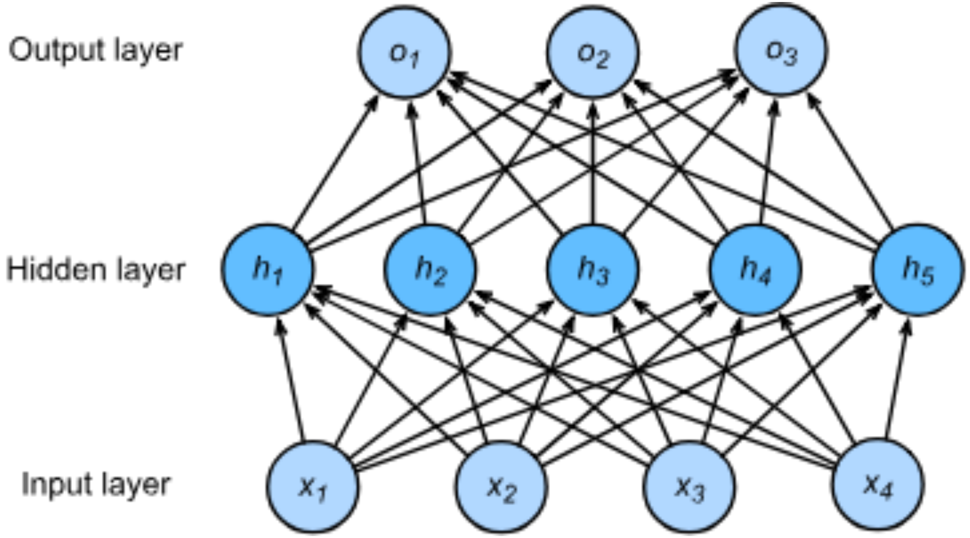
Hyperparameter - size m of hidden layer

Single Hidden Layer

- Input $\mathbf{x} \in \mathbb{R}^n$
- Hidden $\mathbf{W}_1 \in \mathbb{R}^{m \times n}, \mathbf{b}_1 \in \mathbb{R}^m$
- Output $\mathbf{w}_2 \in \mathbb{R}^m, b_2 \in \mathbb{R}$

$$\mathbf{h} = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

σ is an element-wise activation function



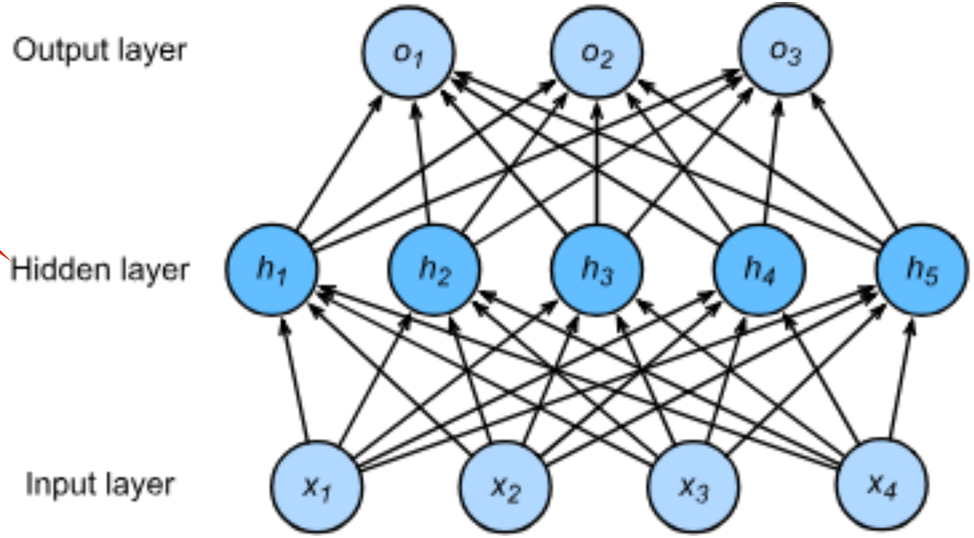
Single Hidden Layer

Why do we need an a nonlinear activation?

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + b_2$$

σ is an element-wise activation function



Single Hidden Layer

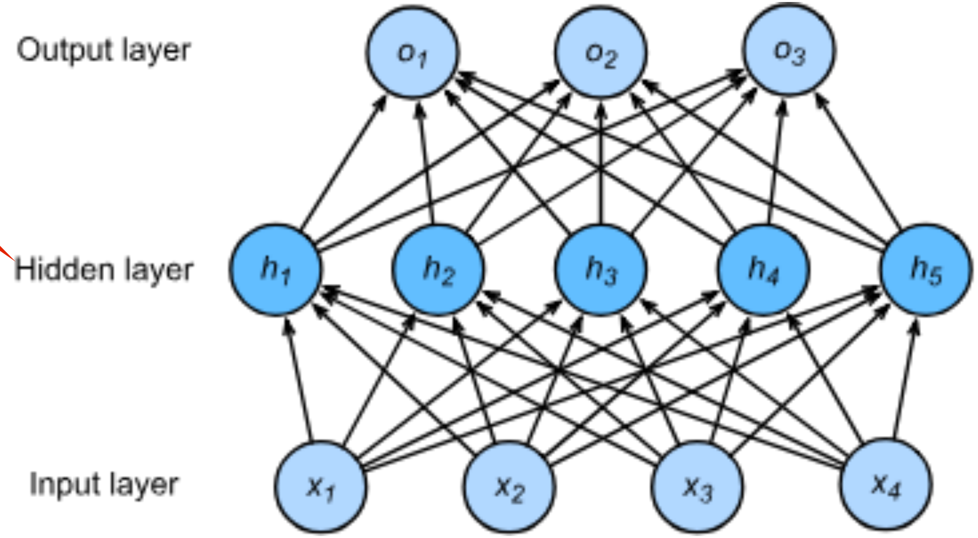
Why do we need an a nonlinear activation?

$$\mathbf{h} = \mathbf{W}_1 \mathbf{x} + \mathbf{b}_1$$

$$\mathbf{o} = \mathbf{w}_2^T \mathbf{h} + b_2$$

hence $o = \mathbf{w}_2^\top \mathbf{W}_1 \mathbf{x} + b'$

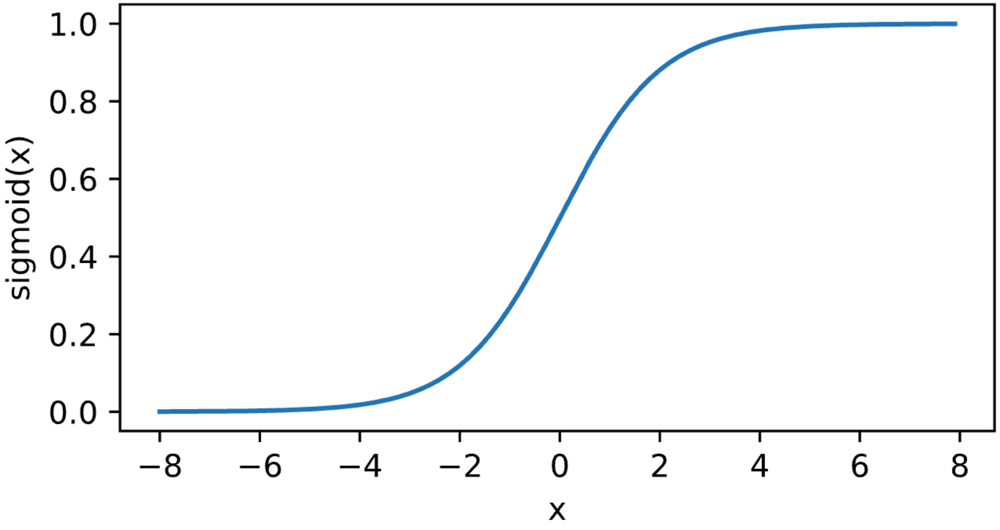
Linear ...



Sigmoid Activation

Map input into (0, 1), a soft version of $\sigma(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

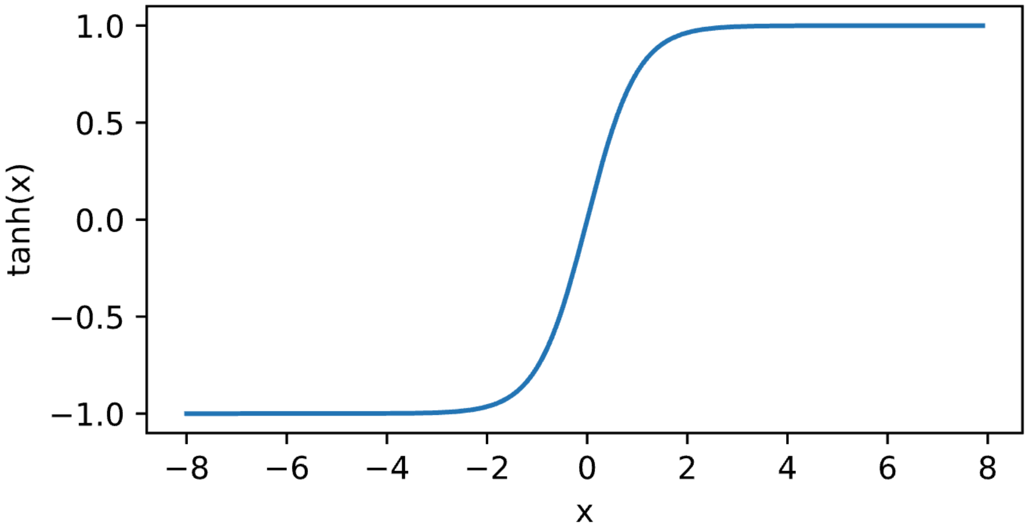
$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}$$



Tanh Activation

Map inputs into (-1, 1)

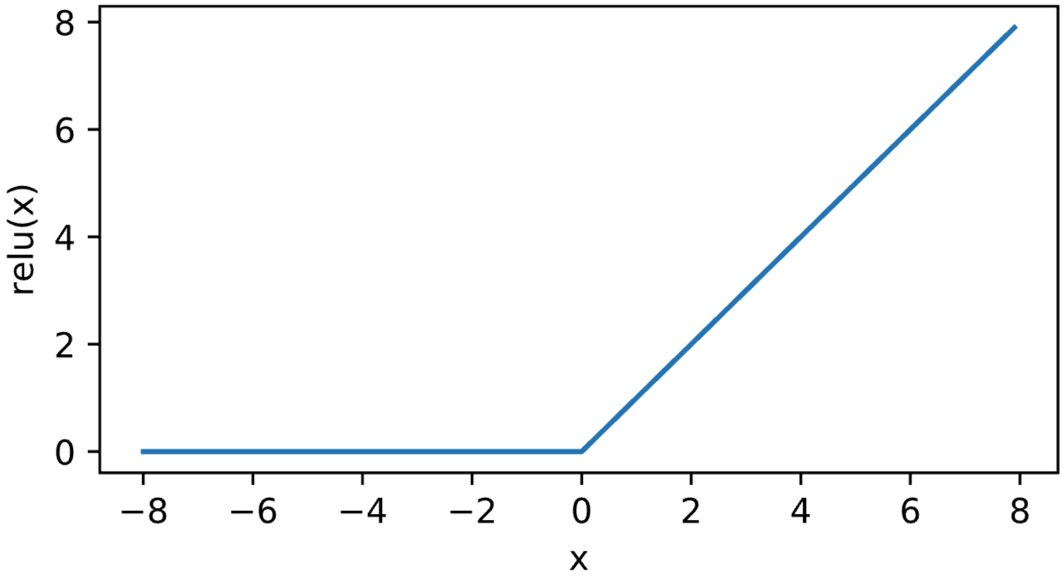
$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$



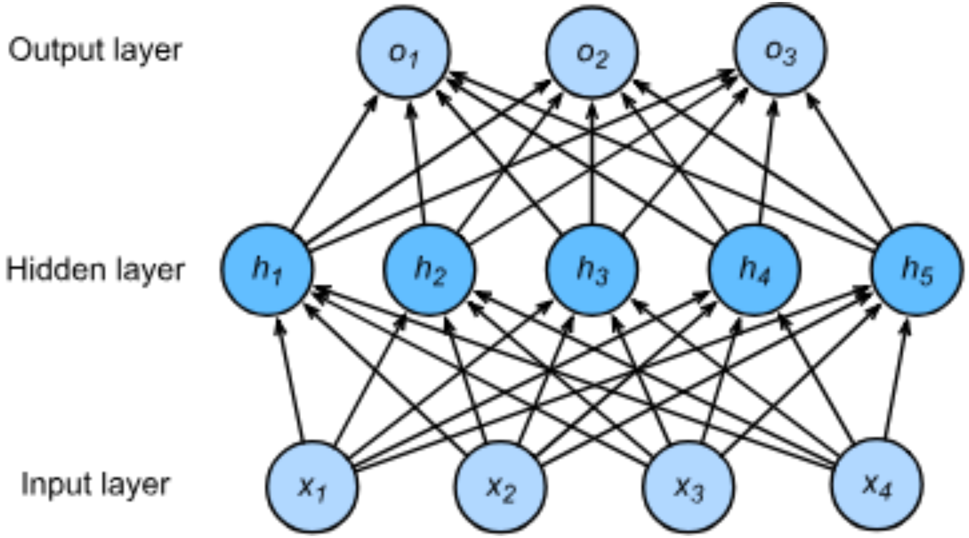
ReLU Activation

ReLU: rectified linear unit

$$\mathrm{ReLU}(x) = \max(x, 0)$$

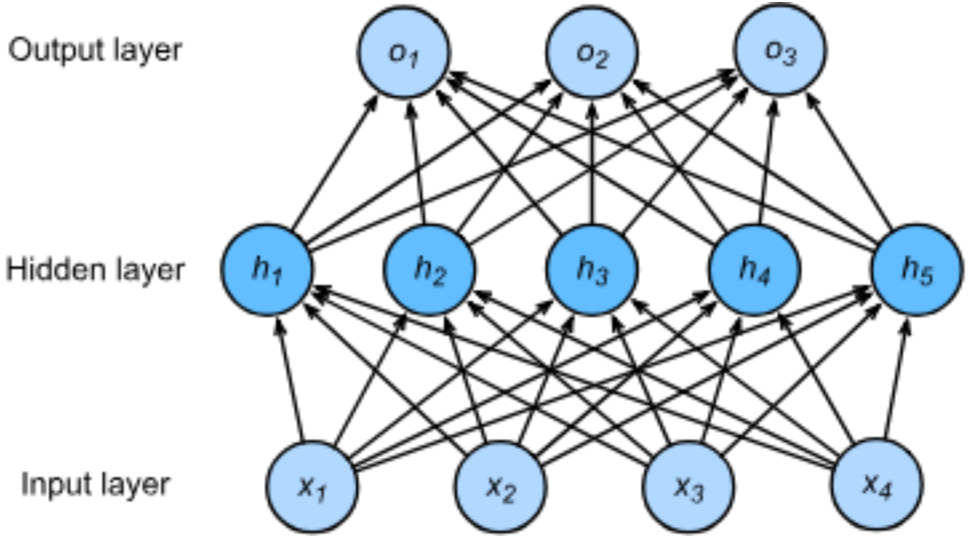


Multiclass Classification



Multiclass Classification

$$y_1, y_2, \ldots, y_k = \text{softmax}(o_1, o_2, \ldots, o_k)$$



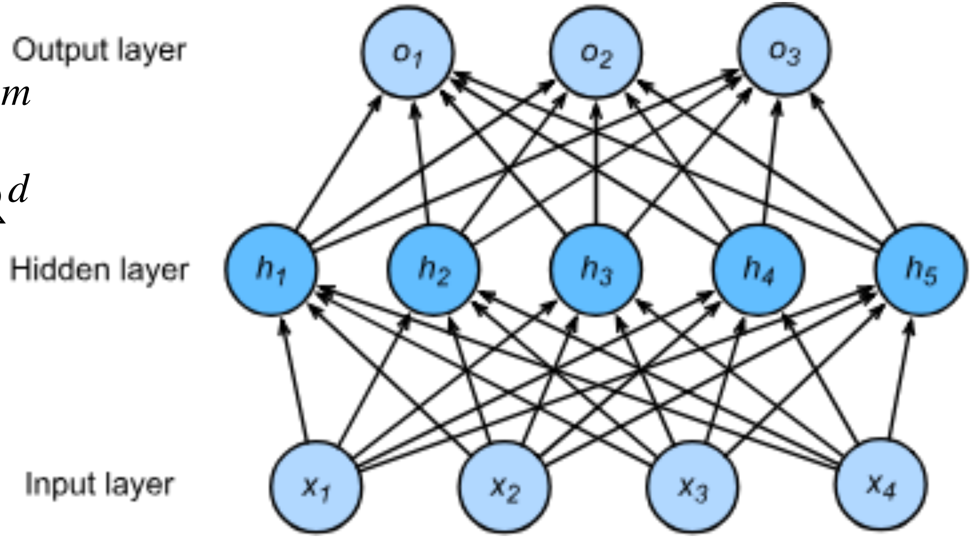
Multiclass Classification

- Input $\mathbf{x} \in \mathbb{R}^n$
- Hidden $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$ and $\mathbf{b}_1 \in \mathbb{R}^m$
- Output $\mathbf{W}_2 \in \mathbb{R}^{m \times d}$ and $\mathbf{b}_2 \in \mathbb{R}^d$

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + \mathbf{b}_2$$

$$\mathbf{y} = \mathrm{softmax}(o)$$



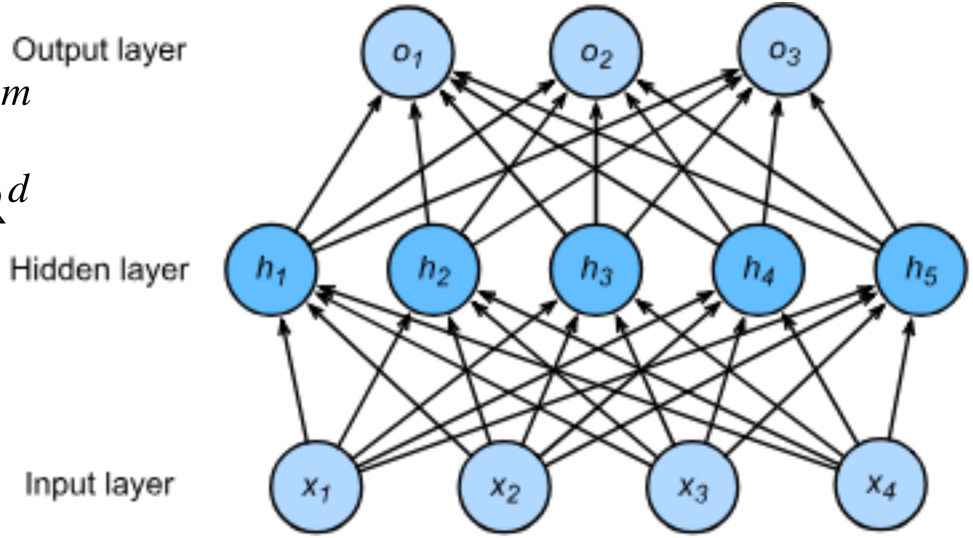
Multiclass Classification

- Input $\mathbf{x} \in \mathbb{R}^n$
- Hidden $\mathbf{W}_1 \in \mathbb{R}^{m \times n}$ and $\mathbf{b}_1 \in \mathbb{R}^m$
- Output $\mathbf{W}_2 \in \mathbb{R}^{m \times d}$ and $\mathbf{b}_2 \in \mathbb{R}^d$

$$\mathbf{h} = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{o} = \mathbf{w}_2^T\mathbf{h} + \mathbf{b}_2$$

$$\mathbf{y} = \mathrm{softmax}(o)$$

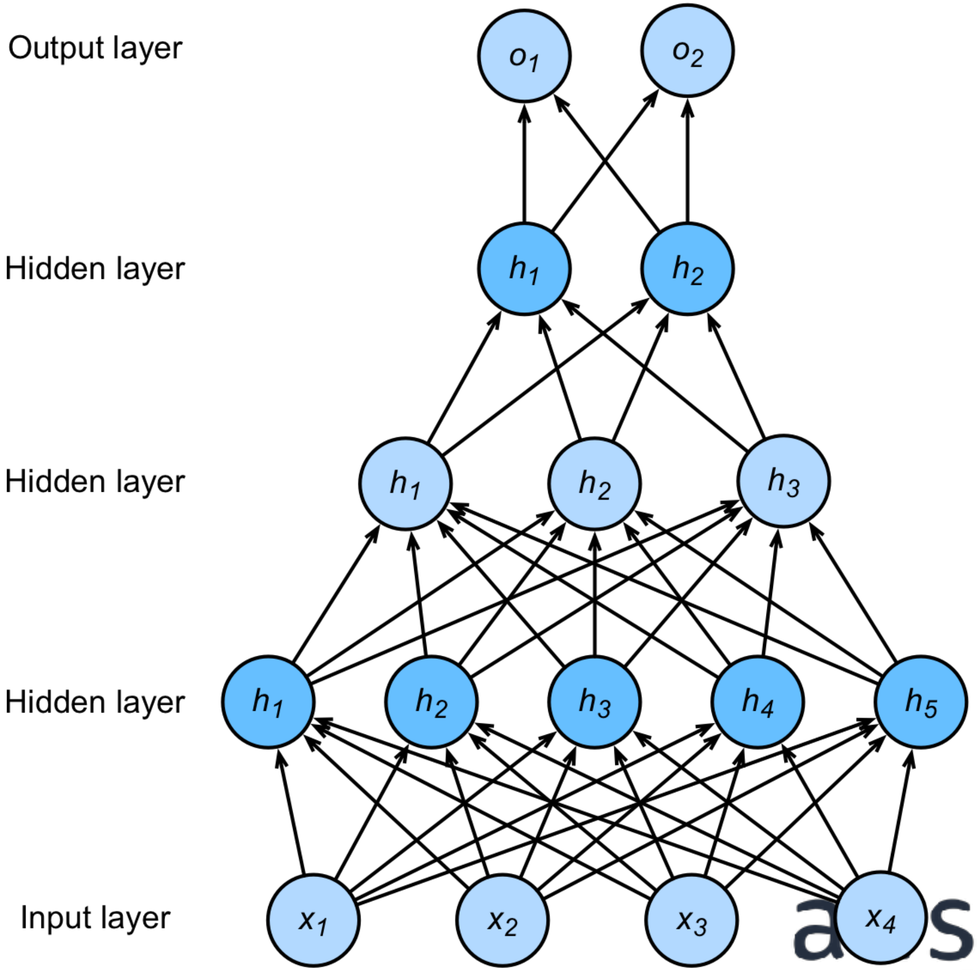


Multiple Hidden Layers

$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$
$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$
$$\mathbf{o} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$

Hyper-parameters

- # of hidden layers
- Hidden size for each layer



Summary

- Perceptron
    - Simple updates
    - Limited function complexity
- Multilayer Perceptron
    - Multiple layers add more complexity
    - Nonlinearity is needed
    - Simple composition (but architecture search needed)