Neural networks

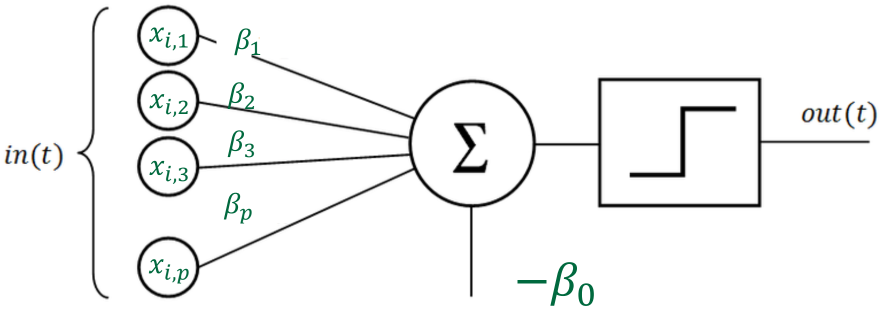
Lecture 15 of “Mathematics and AI”

Outline

1. The multiclass perceptron
2. Deep learning
3. Why do neural networks to learn?
    1. The neuroscientist's answer
    2. The mathematician's answer

The multiclass perceptron

Perceptrons (recap)



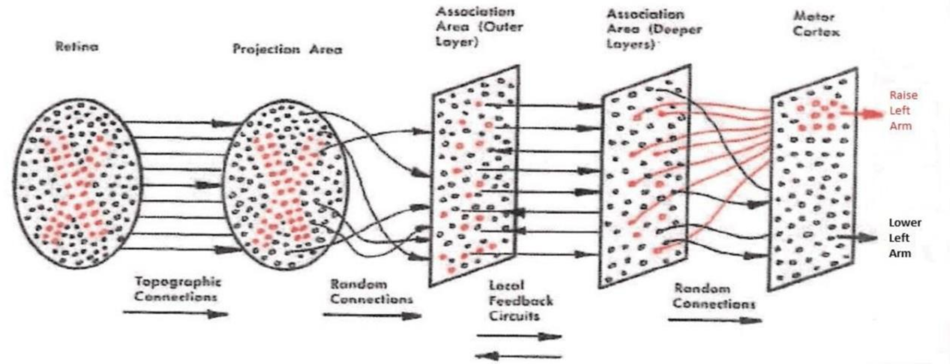


FIG. 1 — Organization of a biological brain. (Red areas indicate active cells, responding to the letter X.)

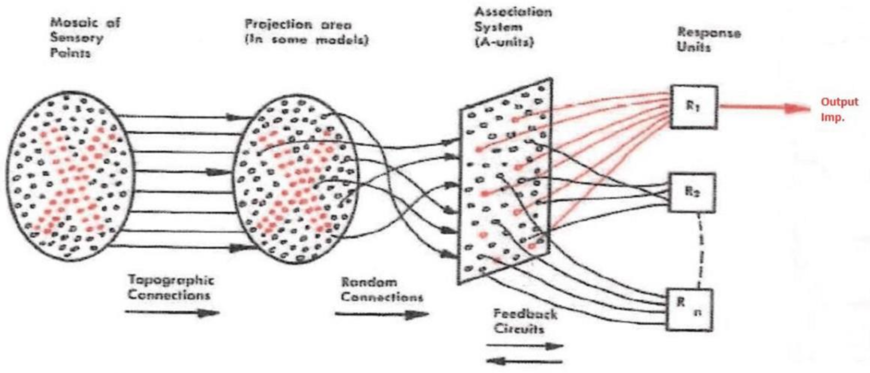
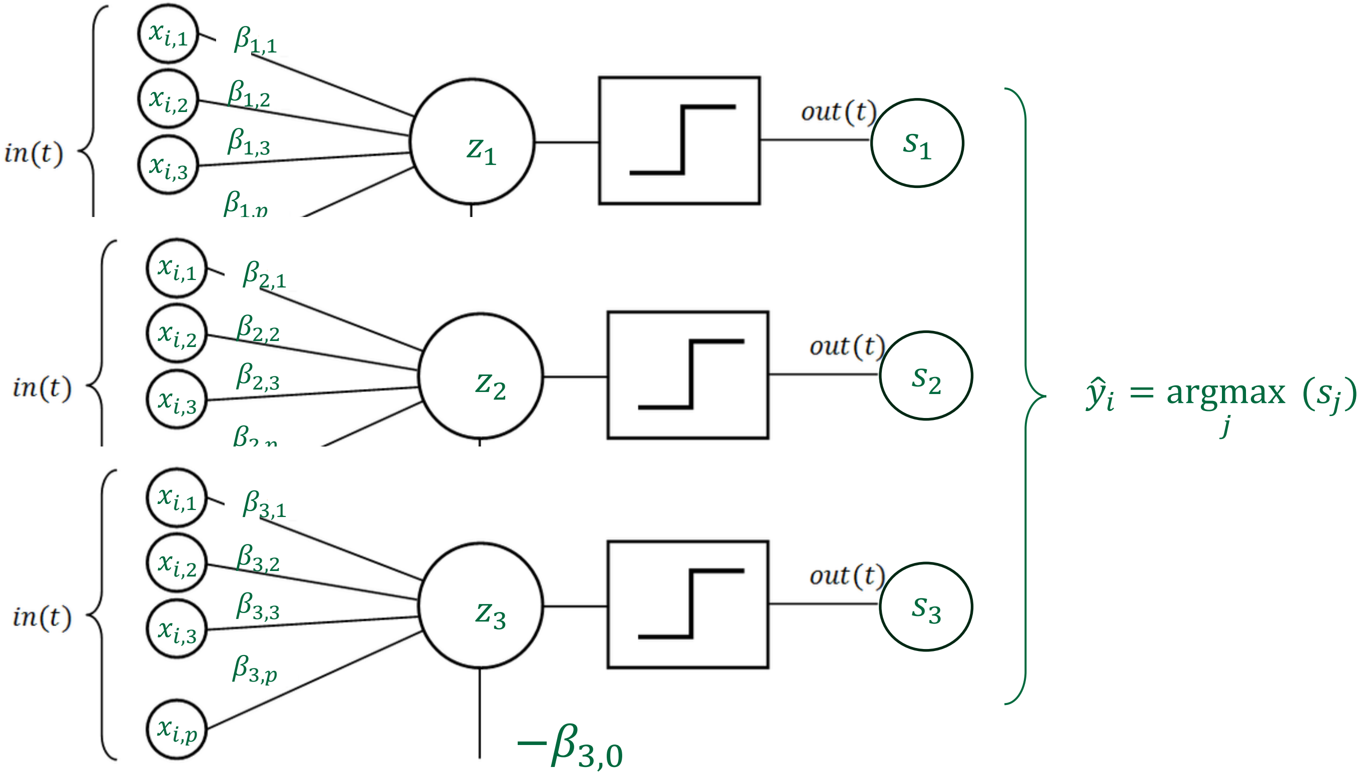


FIG. 2 — Organization of a perceptron.

Image source: https://github.com/jg-fisher/perceptron,
Rosenblatt, F. The Design of an Intelligent Automaton, Research Reviews, Office of Naval Research. Washington, October 1958, 5-13

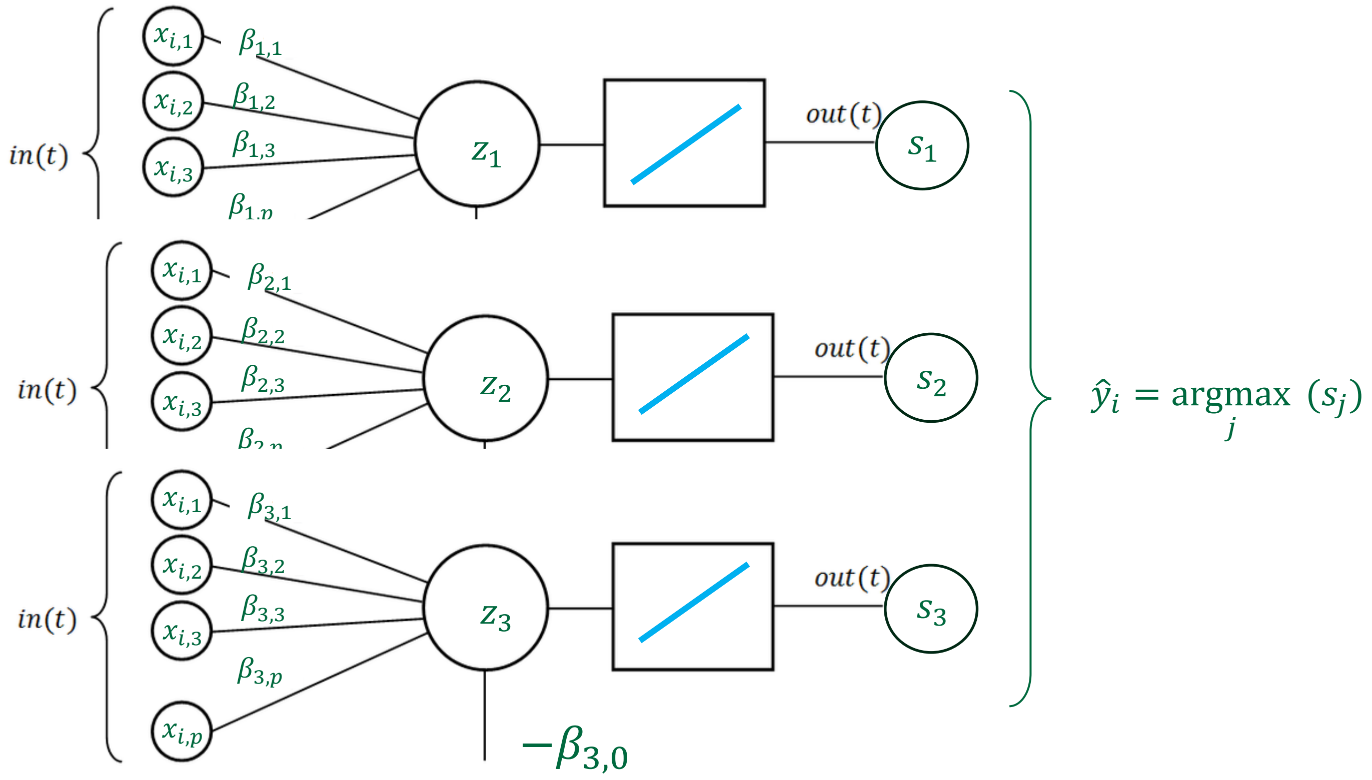
The multiclass perceptron

- Power in numbers!



The linear multiclass perceptron

- Power in numbers!
- Linear activation function
  - enables argmax evaluation
  - yields a linear classifier



The nonlinear multiclass perceptron

- ReLU activation
  - enables argmax evaluation
  - yields a nonlinear classifier
  - decision boundaries are hyperplane segments in each ***orthant*** of the mapped feature space

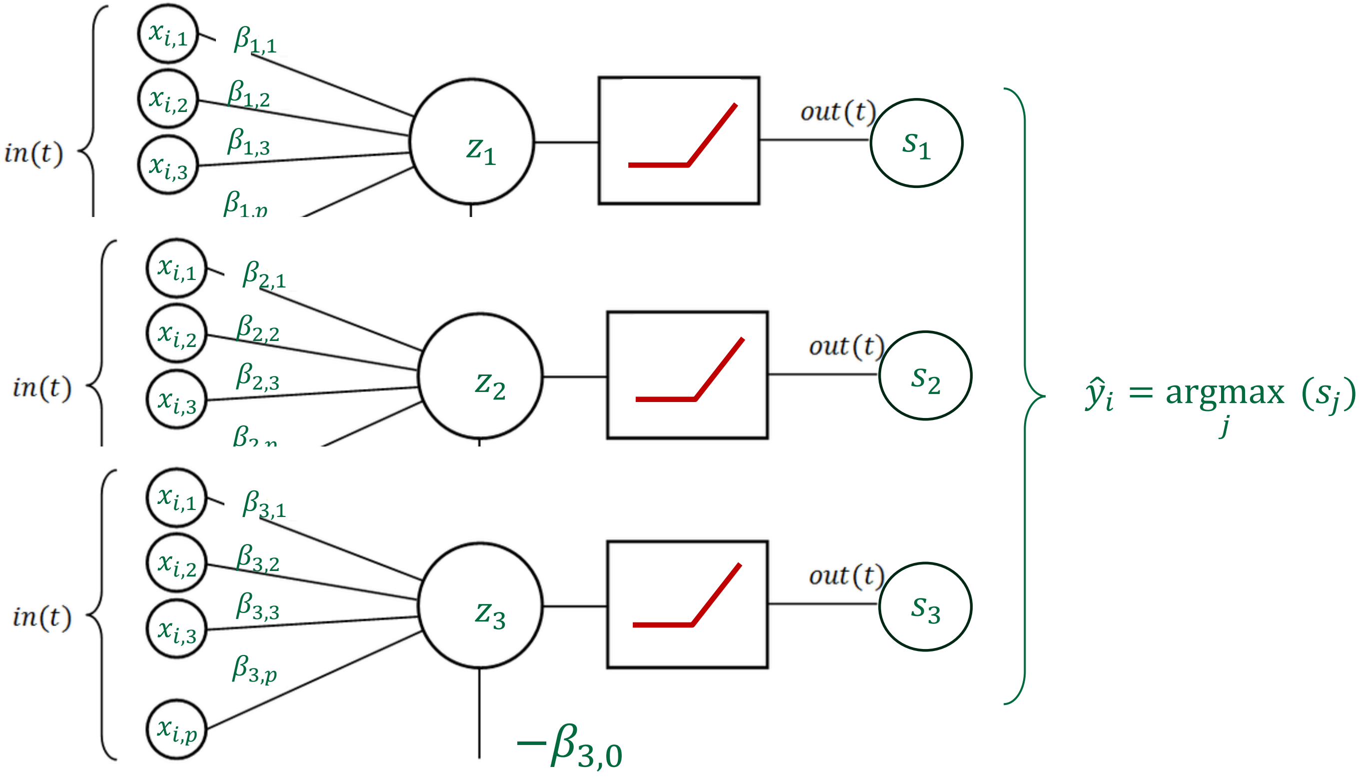
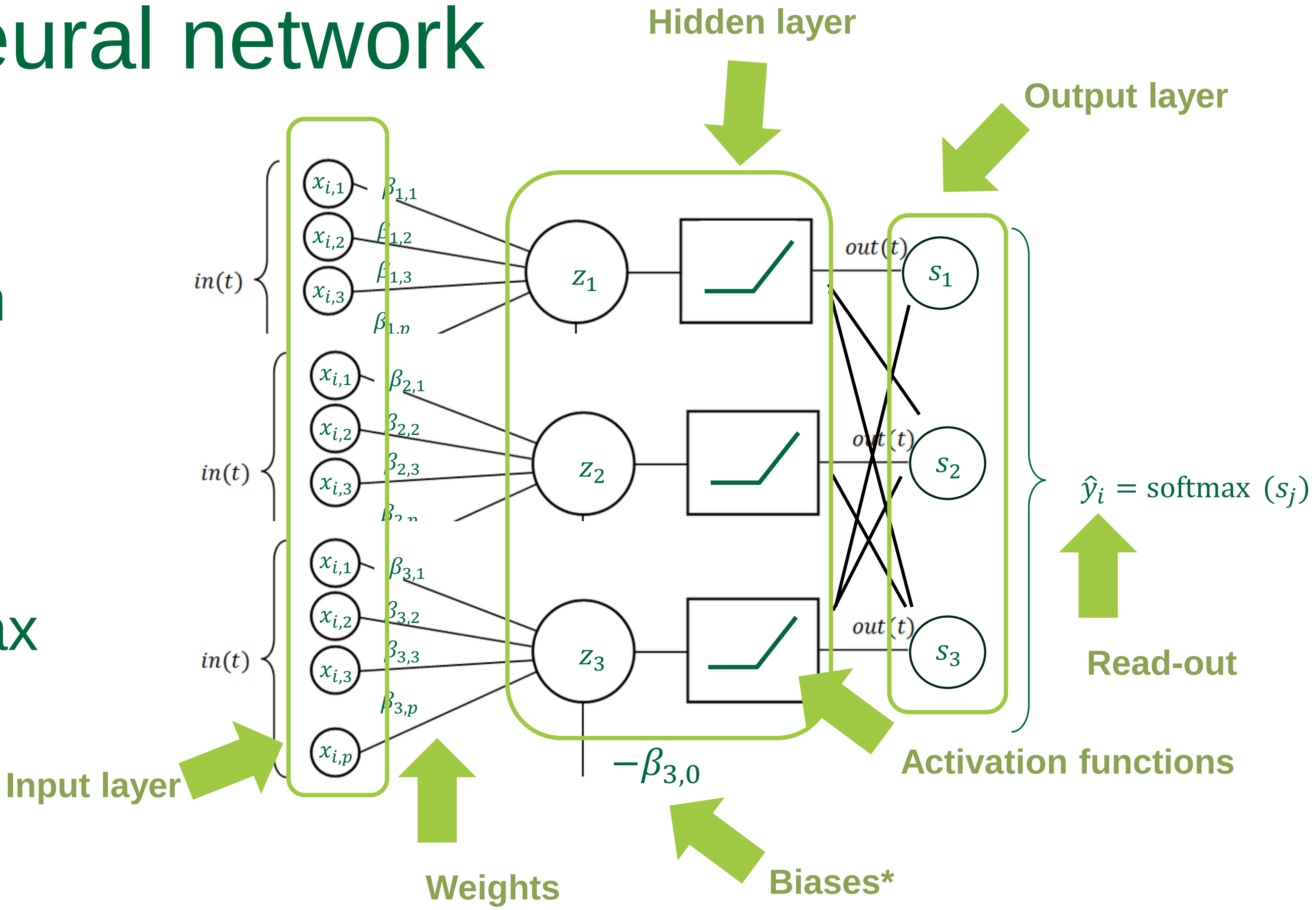


Image source: https://github.com/jg-fisher/perceptron

Deep learning

A feedforward neural network

1. Start with nonlinear multiclass perceptron
2. Let s_k be a linear combination of $\sigma(z_j)$
3. (optional) use softmax instead of argmax



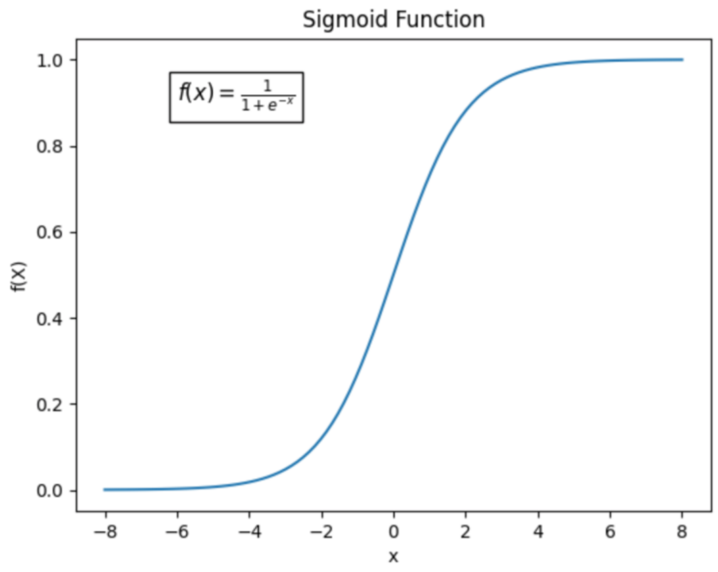
*Unrelated to bias-variance tradeoff!

Image source: https://github.com/jg-fisher/perceptron

Softmax

$$\text{softmax}\ (\vec{s}) = \left(\frac{\exp(s_i)}{\sum_j \exp(s_j)}\right)_i$$

- Bounded: $\text{softmax}\ (\vec{s})\ \in (0,1)$ (logistic/sigmoid shape)
- Continuous and differentiable approximation of the 1-hot encoded arg-max
- Equivalent to Boltzmann distribution (from statistical mechanics) with temperature $T = 1/k$



Boltzmann constant

Image source: https://www.codecademy.com/resources/docs/ai/neural-networks/sigmoid-activation-function

Feed forward neural networks (FNNs) in equations

1. Input features:

$$\vec{x}^{(1)}$$

2. Linear combinations of input features:

$$\vec{w}_j^{(1)} \cdot \vec{x}^{(1)}$$

3. Nonlinear feature mapping:

$$x_j^{(2)} = \sigma\left(\vec{w}_j^{(1)} \cdot \vec{x}^{(1)}\right)$$

4. (in vector notation):

$$\vec{x}^{(2)} = \sigma\left(W^{(1)}\vec{x}^{(1)}\right)$$

5. Linear model(s) on new features:

$$\vec{w}_j^{(2)} \cdot \vec{x}^{(2)}$$

6. Softmax output

$$x_j^{(out)} = \mathrm{softmax}\left(\vec{w}_j^{(2)} \cdot \vec{x}^{(2)}\right)$$

(in vector notation):

$$\vec{x}^{(out)} = \mathrm{softmax}\left(W^{(2)}\vec{x}^{(2)}\right)$$

Deep learning: Multi-layer FFNs

- A network with m hidden layers:

$$\vec{x}^{(k+1)} = \sigma\left(W^{(k)}\vec{x}^{(k)}\right) \text{ for } k = 1, \ldots m$$

$$\vec{x}^{(out)} = \text{softmax}\left(W^{(m)}\vec{x}^{(m)}\right)$$

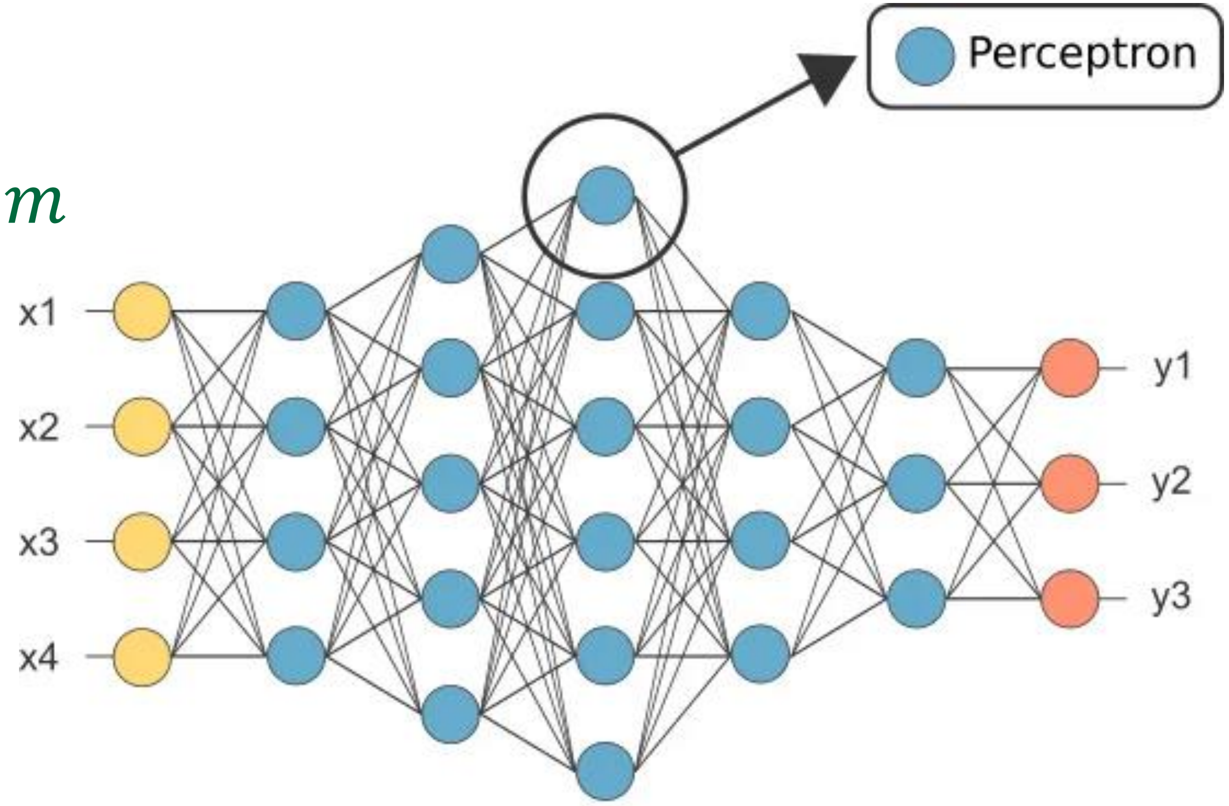


Image source: https://medium.com/@b.terryjack/introduction-to-deep-learning-feed-forward-neural-networks-ffnns-a-k-a-c688d83a309d

Why do neural networks learn?

The neuroscientist's answer

- Many parallels to the human visual system suggest that FNNs could be good models for computer vision
- Other architectures may be better for other tasks (RNNs, reservoir computing)

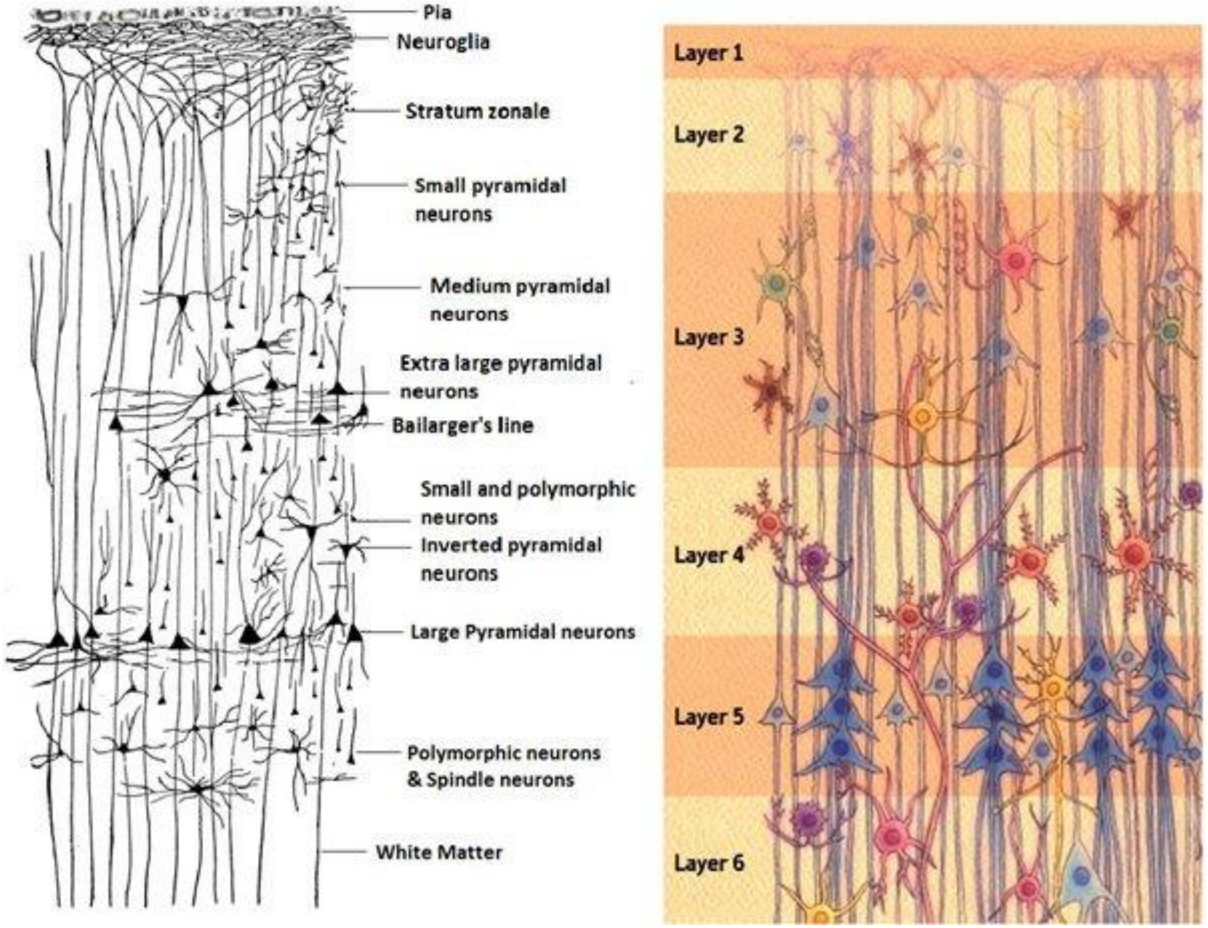


Image source: https://www.quora.com/How-many-layers-of-neurons-are-in-the-cortex-of-the-human-brain

The mathematician's answer

- Model expressiveness:
  - A model with features mapped to the basis of a function space can express any function in that function space (Taylor, Fourier, etc.)
  - A model with a kernel K can express any function in a corresponding RHKS
  - In principle: infinite basis, but finite non-zero coefficients to get to 0 training error (representer theorem)

Image source: https://blog.gopenai.com/mastering-multiclass-classification-with-single-layer-perceptrons-matlab-implementation-from-d8d30e979dec

The mathematician's answer

- What can neural networks "express" (i.e., model)?
- Universal approximation theorem:

Universal approximation theorem — Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$$

where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$

Source: Wikipedia