16-720: Computer Vision

FALL 2017

INSTRUCTORS

SRINIVAS NARASIMHAN AND DAVID HELD

M-W: 4:30p-5:50p

(Slides from Yaser Sheikh)

DEEP LEARNING I

An Introduction to Deep Learning for Visual Computing

“The eye sees only what the mind is prepared to comprehend”

- Henri Bergson

WHAT IS PERCEPTION?

Per·cep·tion: The ability to see, hear, or become aware of something through the senses

Sensation ≠ Perception

Single Sensation, Dual Perception

"My Wife and My Mother-in-law” [Bohring 1930]

CONTEXTUAL PRIMING

The McGurk Effect [McGurk and Donald 1976]

Video Credit: BBC Two

CONTEXTUAL PRIMING

The McGurk Effect [McGurk and Donald 1976]

Video Credit: BBC Two

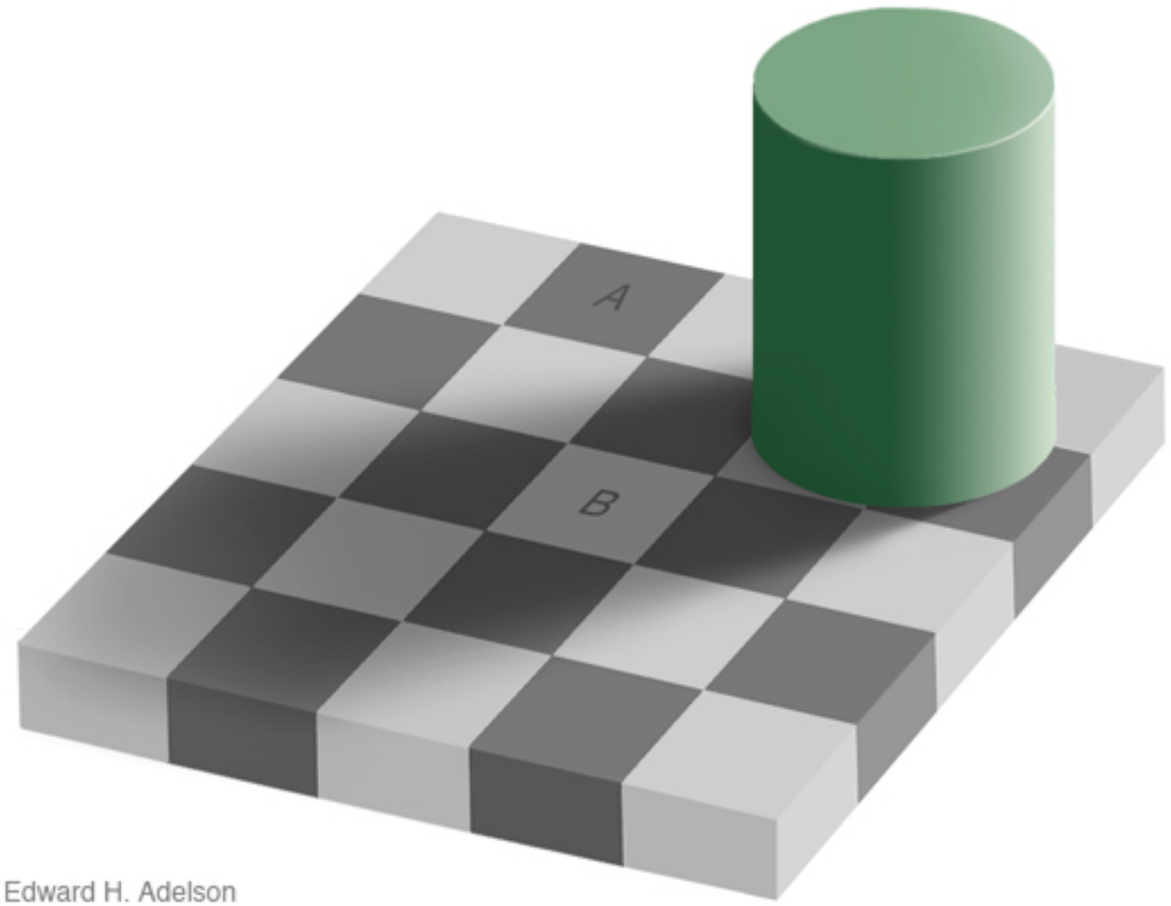
CONTEXTUAL PRIMING

The McGurk Effect [McGurk and Donald 1976]

Video Credit: BBC Two

WHAT IS PERCEPTION?

Optical Illusions are perceptual errors



Edward H. Adelson

Credit: Edward Adelson

WHAT IS PERCEPTION?

Sensation ≠ Perception

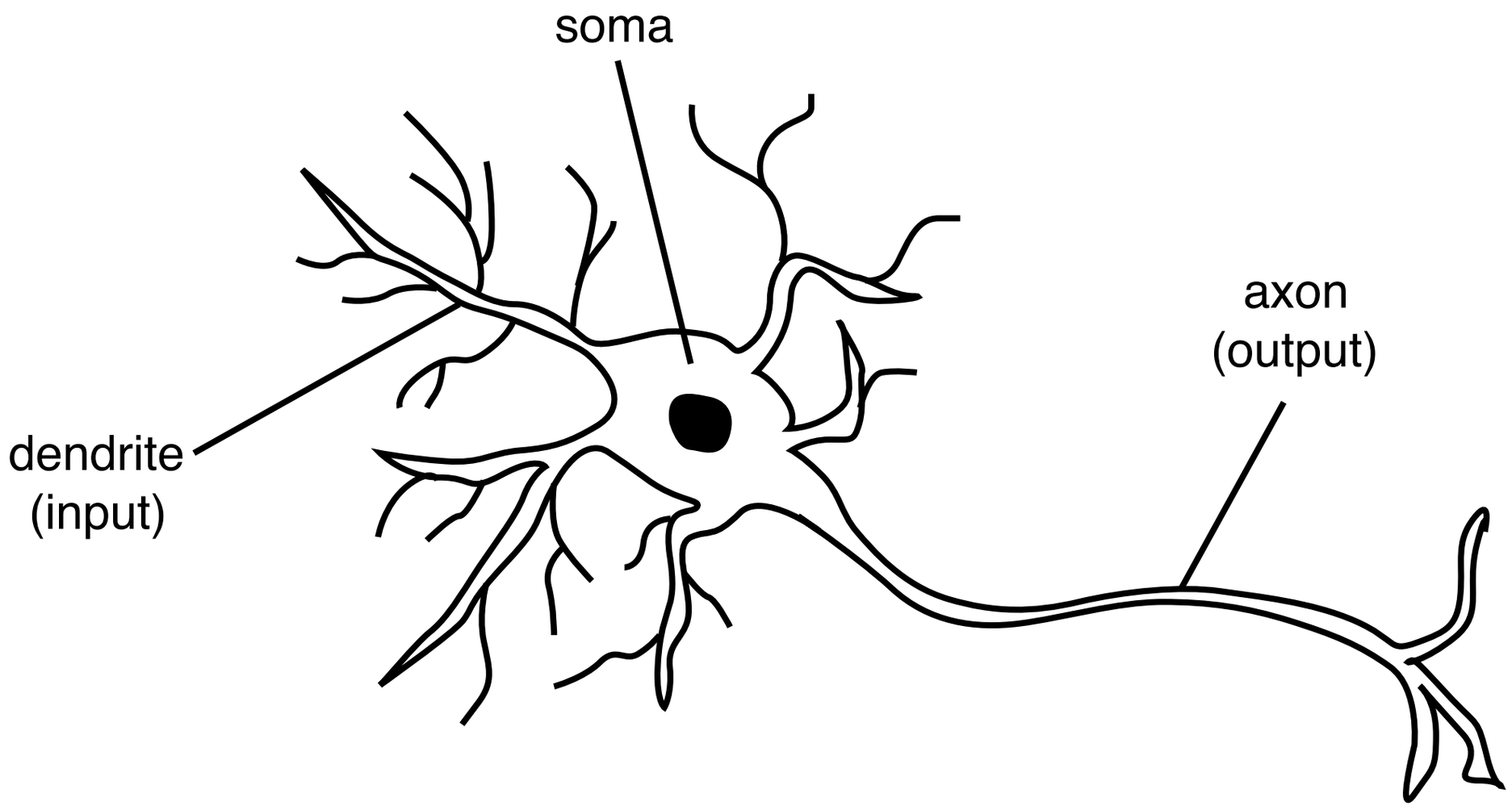
Perception is the processing of sensory information to <u>enable a function</u>

We cannot **program** much of our perception because
we are ignorant about the precise process through which we perceive

MACHINE LEARNING FOR COMPUTER VISION

NEURON

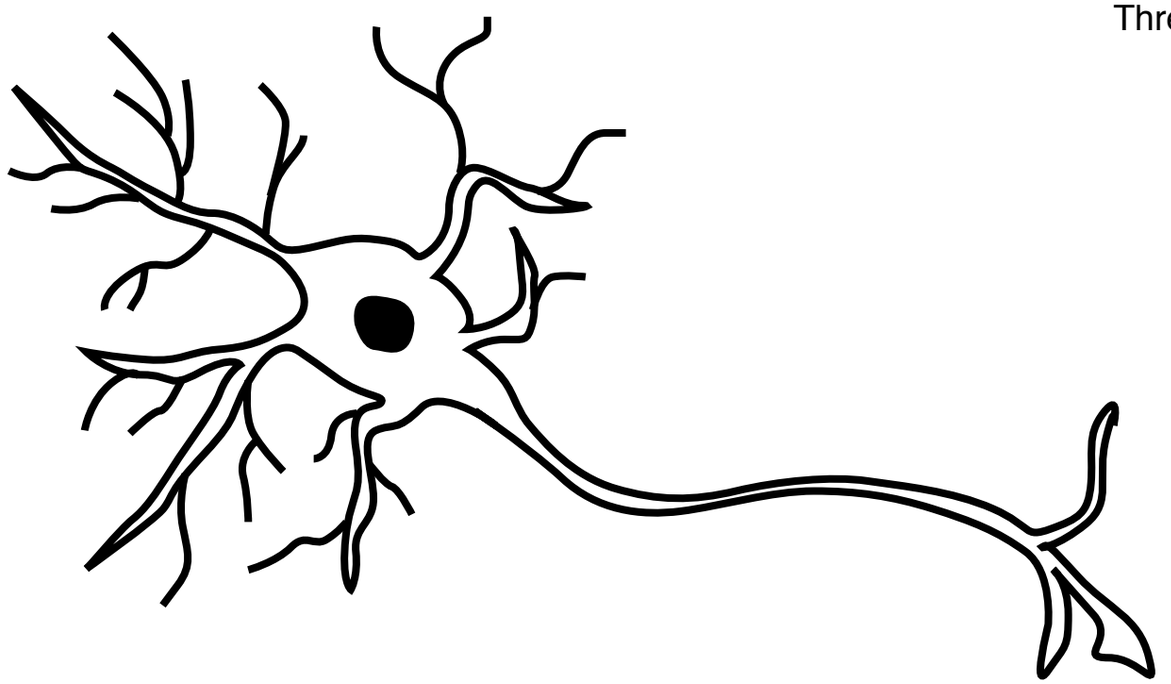
Biological unit of perception

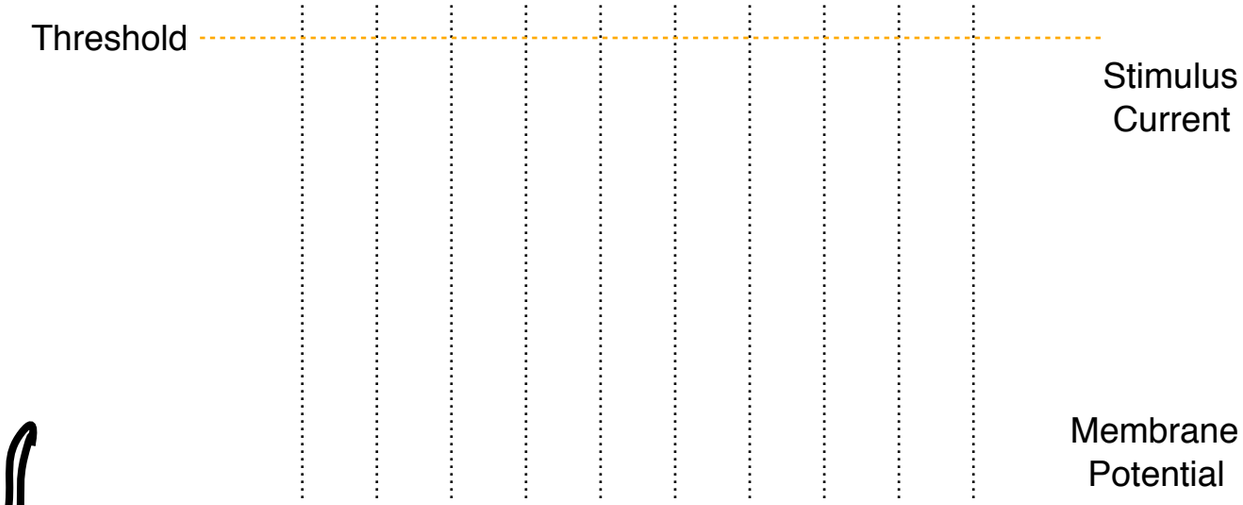


~100 billion neurons in the brain

ACTION POTENTIAL

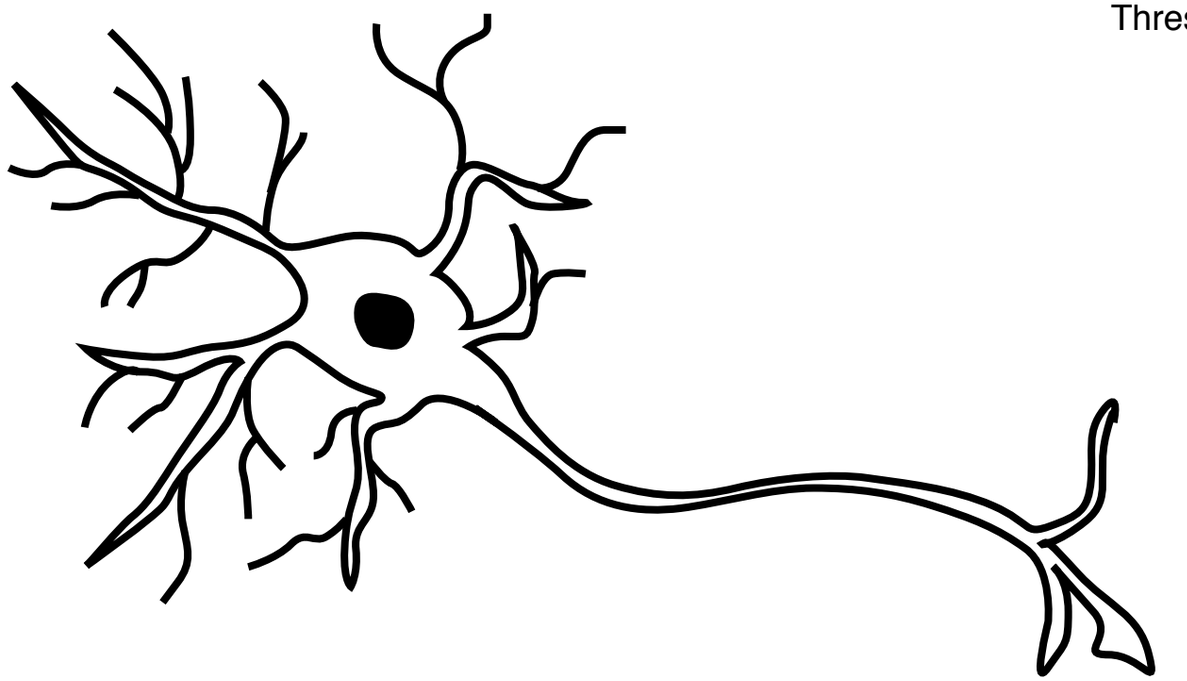
Spike Potentials

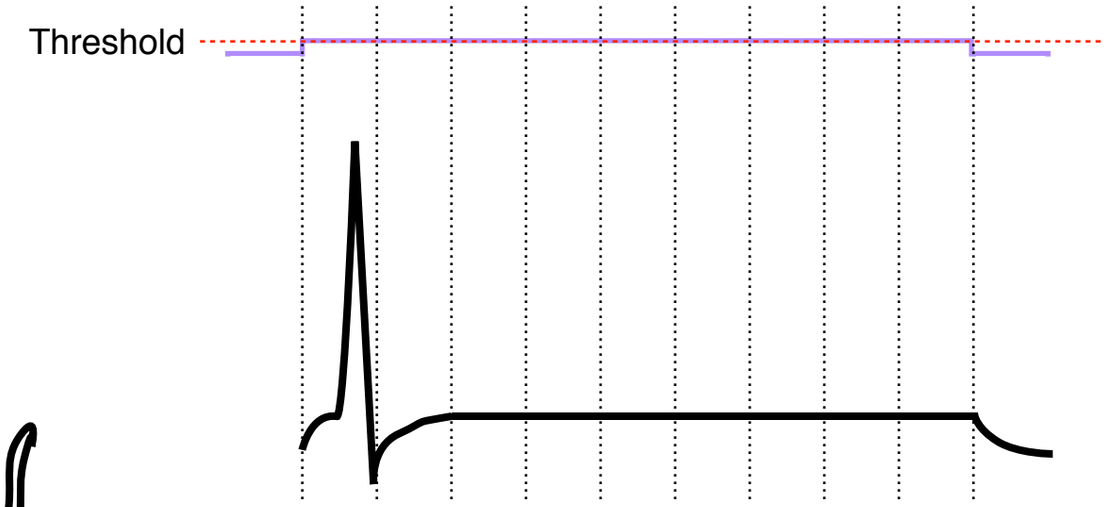




WEAK STIMULUS

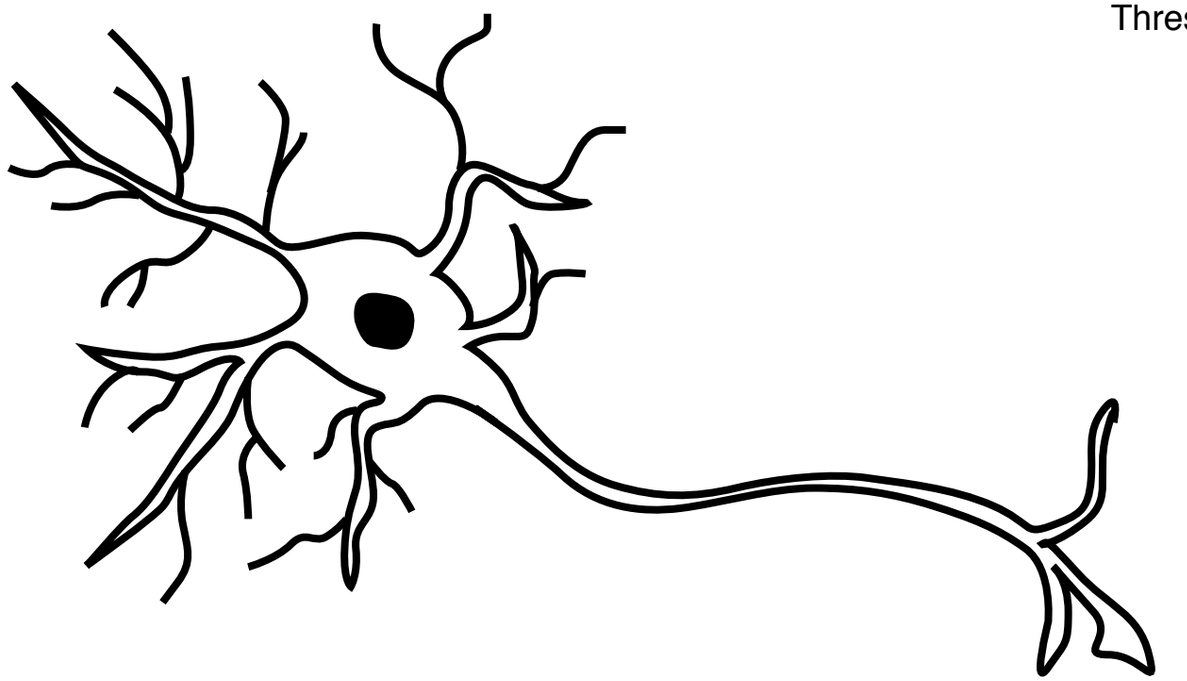
Rate Coding

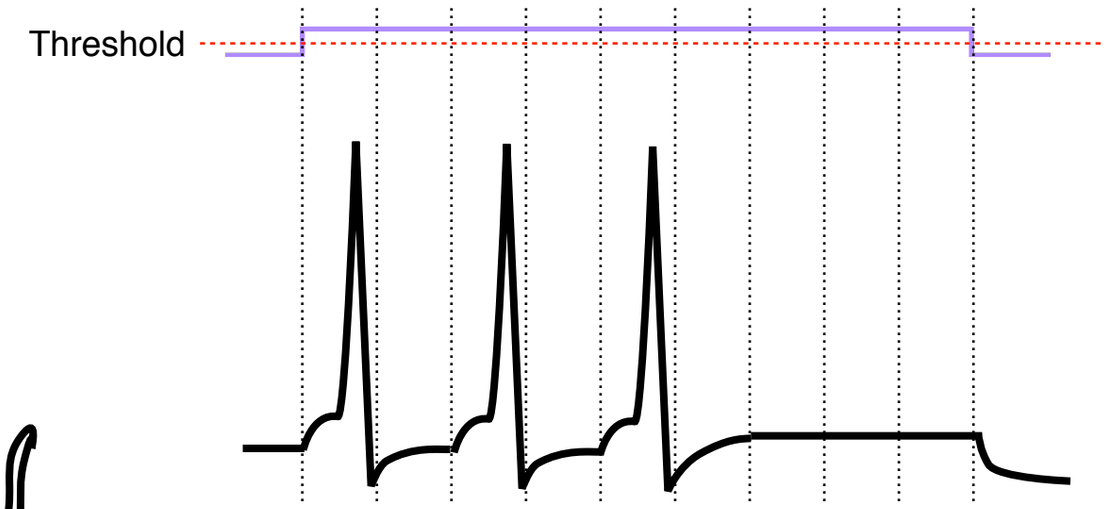




MEDIUM STIMULUS

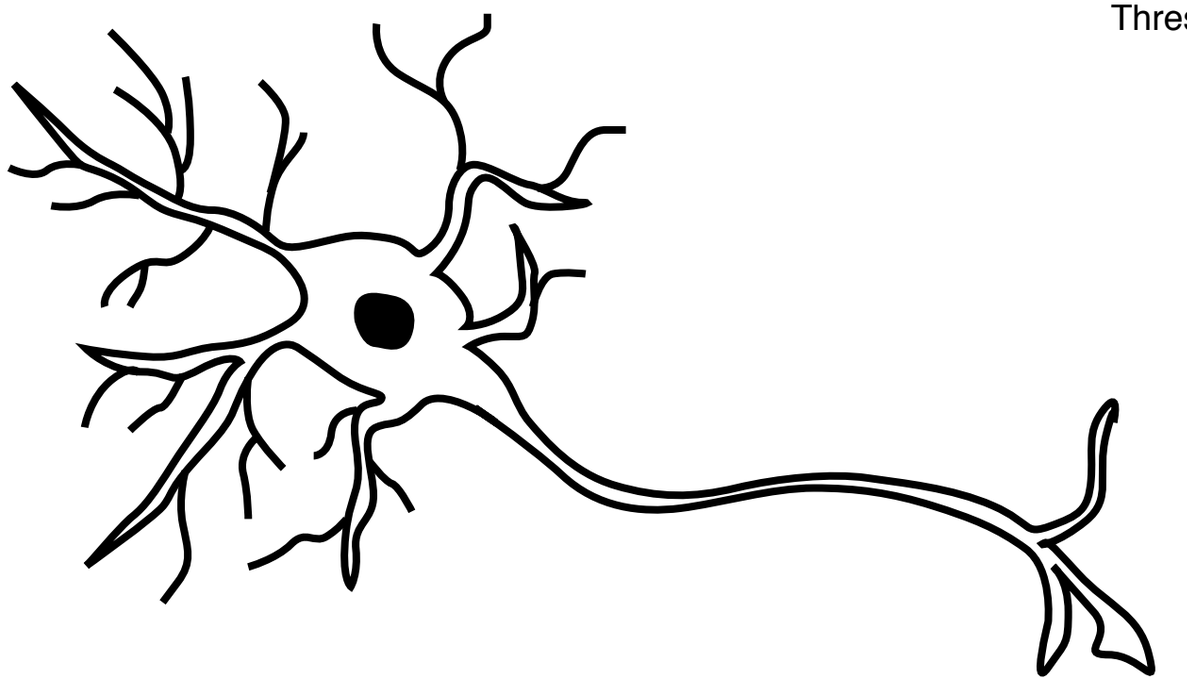
Spike Train

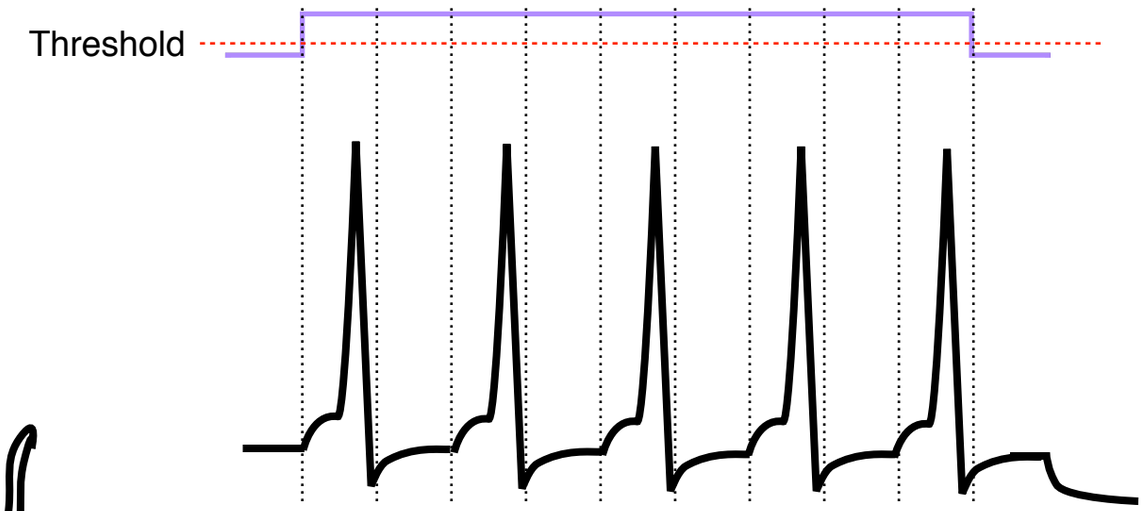




STRONG STIMULUS

Spike Train

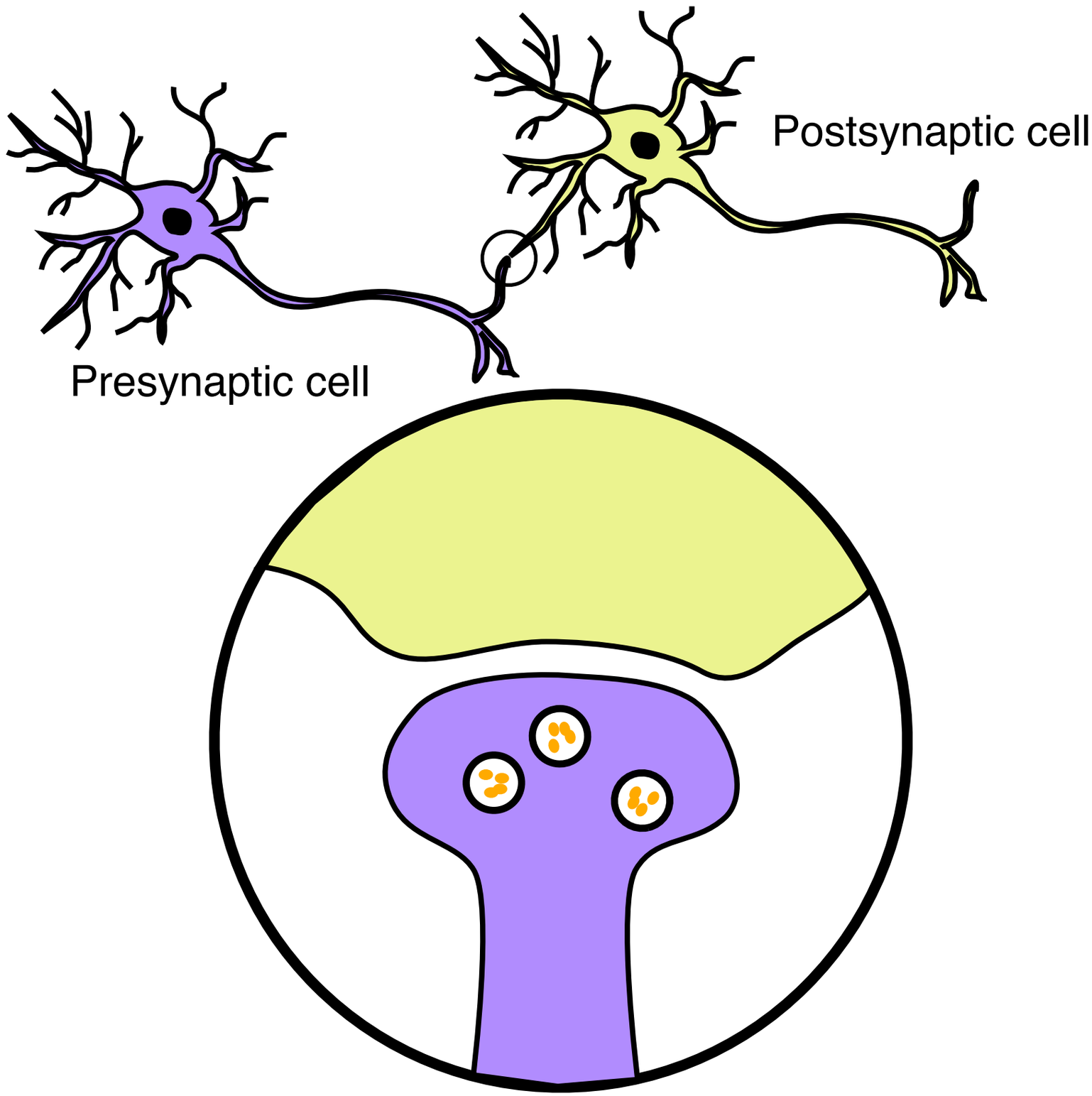




There are many different types of neurons with widely variant operations

SYNAPSE

Each neuron can have thousands of synapses

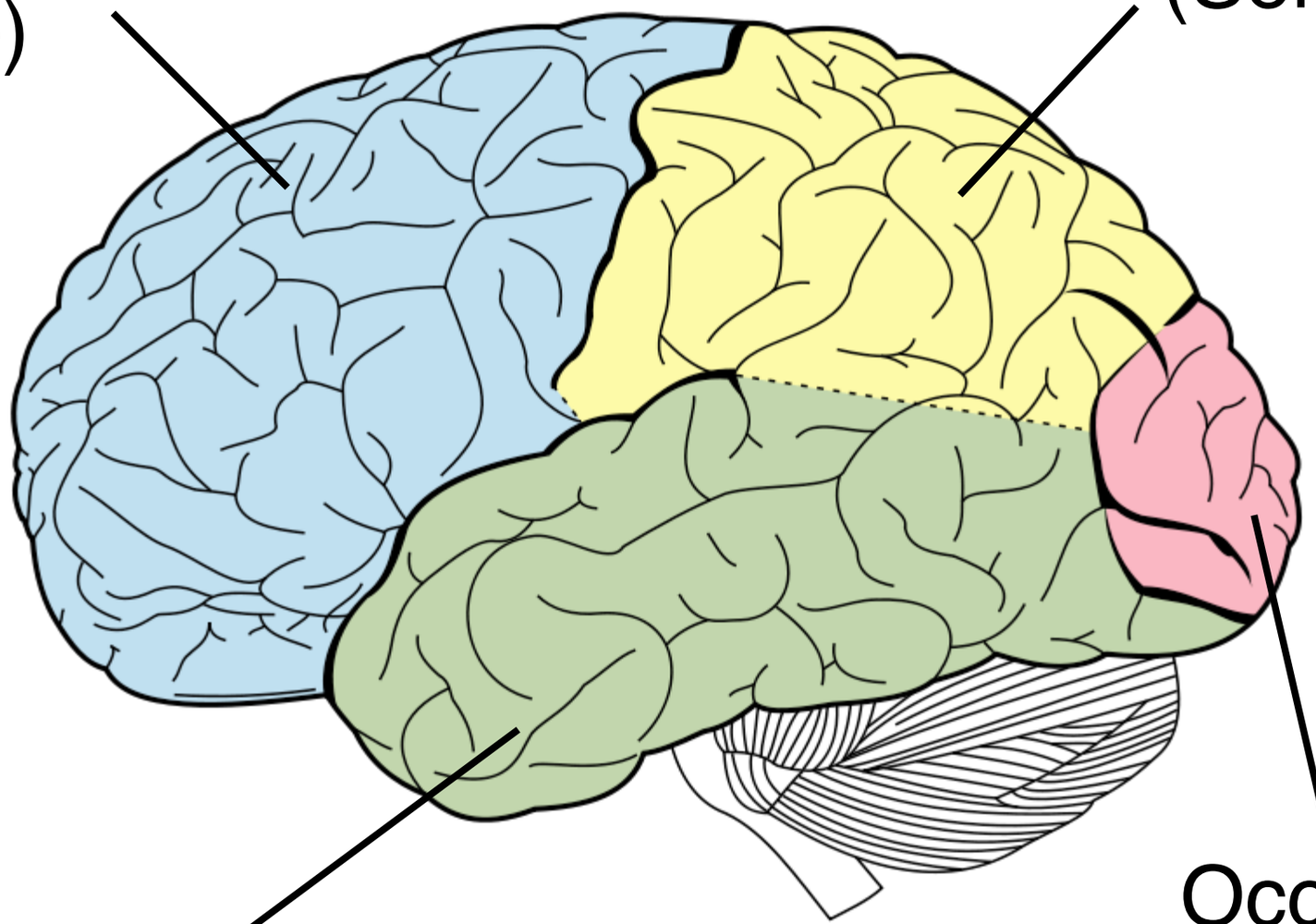


BRAIN

Area Specialization

Frontal Lobe
(Executive)

Parietal Lobe
(Sensing, Spatial)

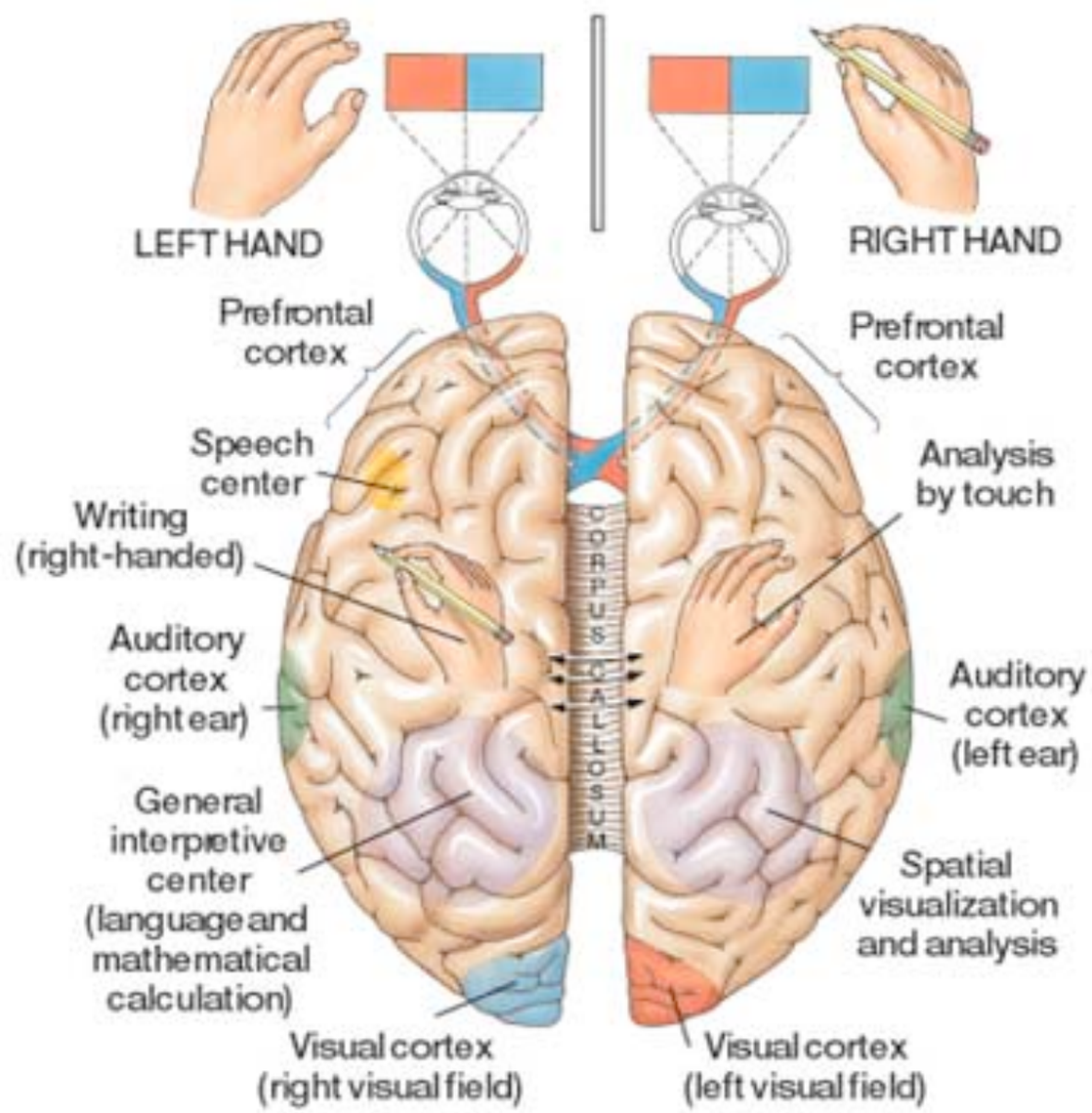


Temporal Lobe
(Audio Sensing, Memory)

Occipital Lobe
(Visual Sensing)

CORPUS CALLOSUM

The information super highway of the brain



200 million fibers processing several billion bits/sec

SPLIT-BRAIN SYNDROME

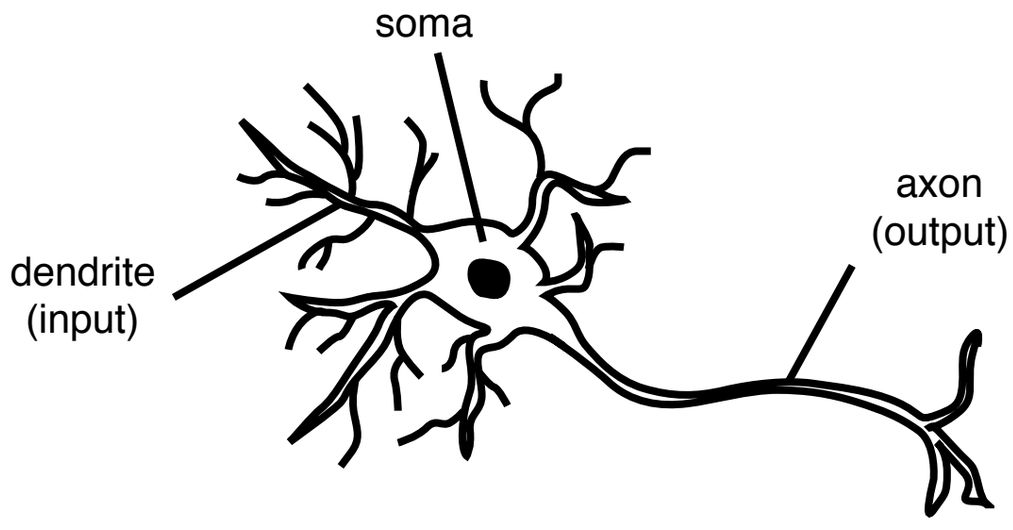
The “Astonishing Hypothesis”

“You, your joys and your sorrows, your memories and your ambitions, your sense of personal identity and free will, are in fact no more than the behaviour of a vast assembly of nerve cells and their associated molecules”

—Francis Crick, 1994

MODELING NEURONS

Simplified Idealizations of Neuron Function



Why Idealize?
Need to idealize to further scientific understanding
Need to idealize for mathematical modeling of applied domains

MODELING NEURONS

Linear Neurons

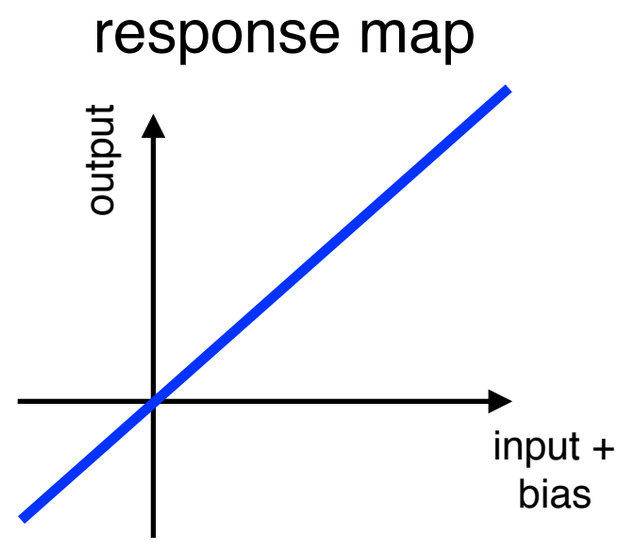
$$y(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + w_0$$

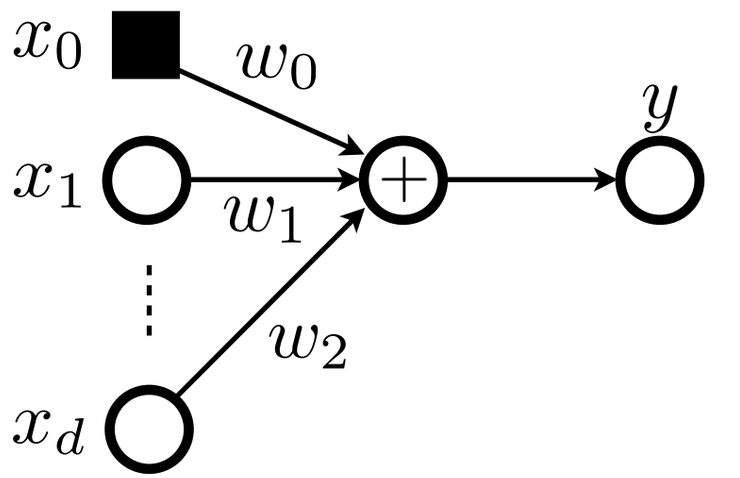
output

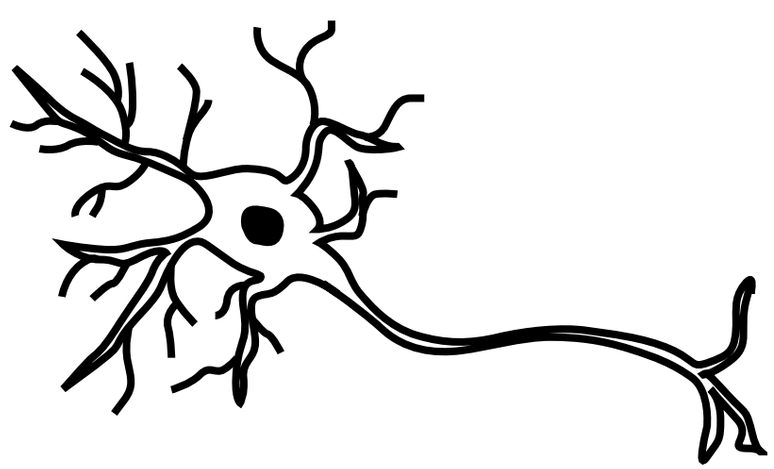
i-th input

i-th weight

bias



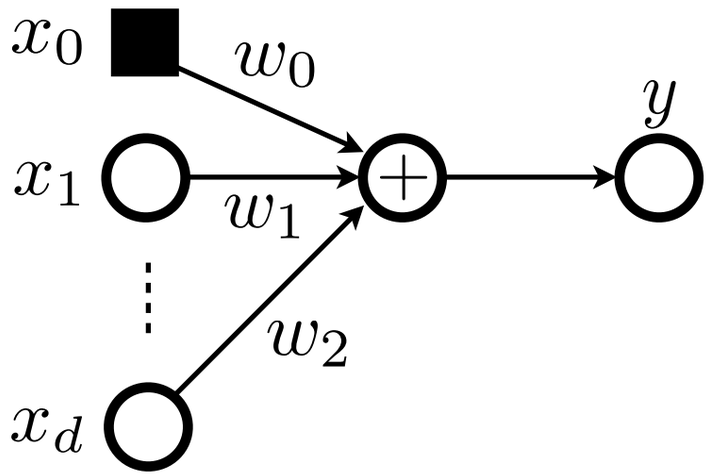




$(x_0 = 1)$

MODELING NEURONS

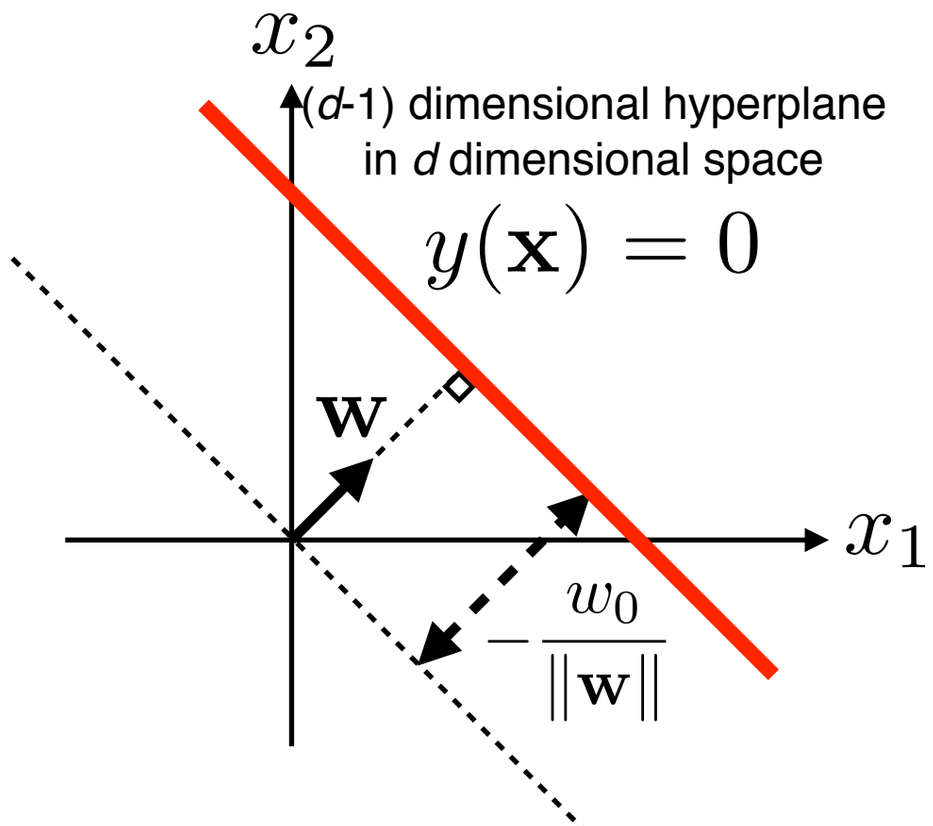
Single Layer Networks: “Shallow” Classifiers



Two category classifier

$$\mathcal{C}_1 : y(\mathbf{x}) > 0$$

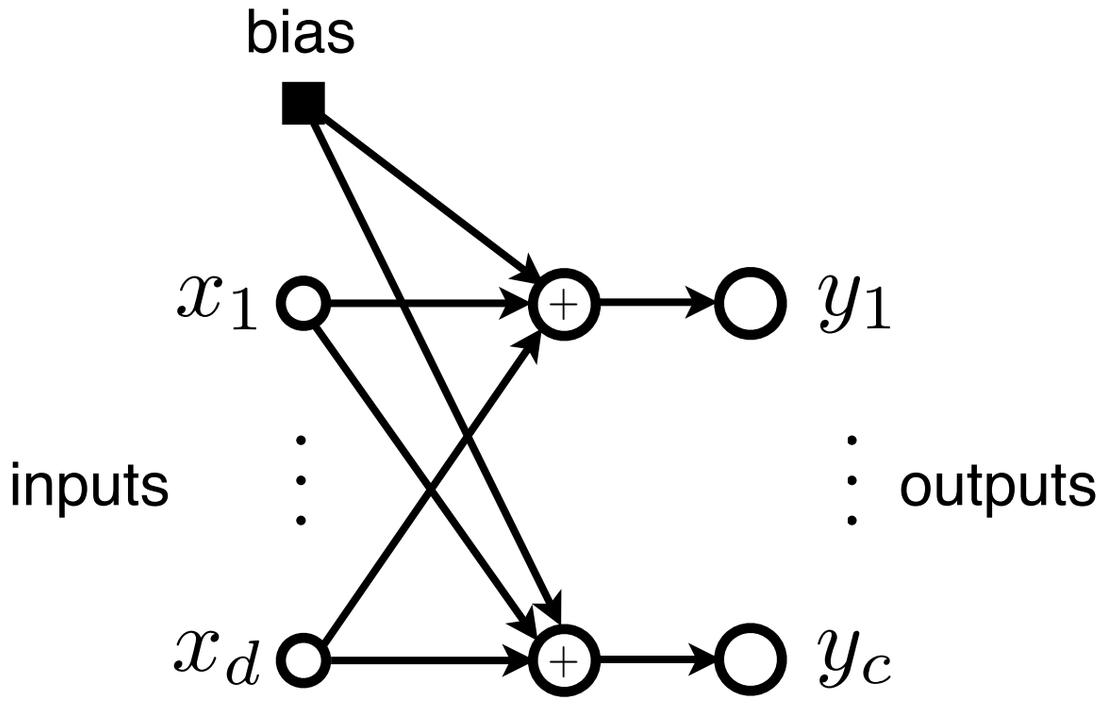
$$\mathcal{C}_2 : y(\mathbf{x}) < 0$$



$$y(x) = \mathbf{w}^T\mathbf{x} + w_0$$

MODELING NEURONS

"Shallow" Learning: Multiple Categories



$$y_k(\mathbf{x}) = \sum_{i=1}^{d} w_{ki} x_i + w_{k0}$$

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

Multi-category classifier

$$\mathcal{C}_i : y_i(\mathbf{x}) > y_j(\mathbf{x}), \forall i \neq j$$

Hyper-plane separating category *k* and *j*

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0$$

MODELING NEURONS

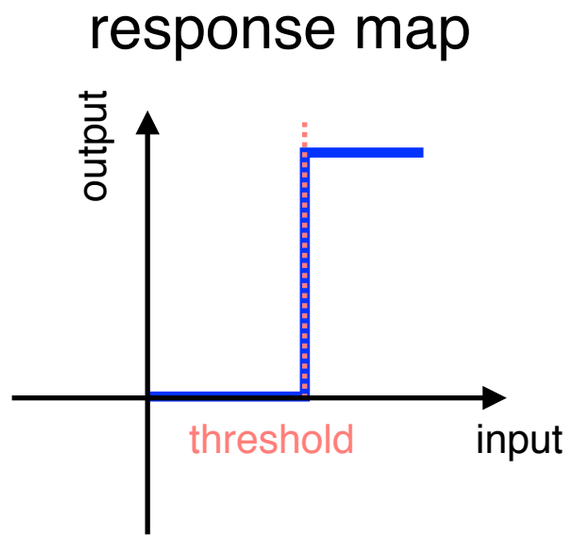
McCulloch-Pitts (1943): Binary Threshold Neurons

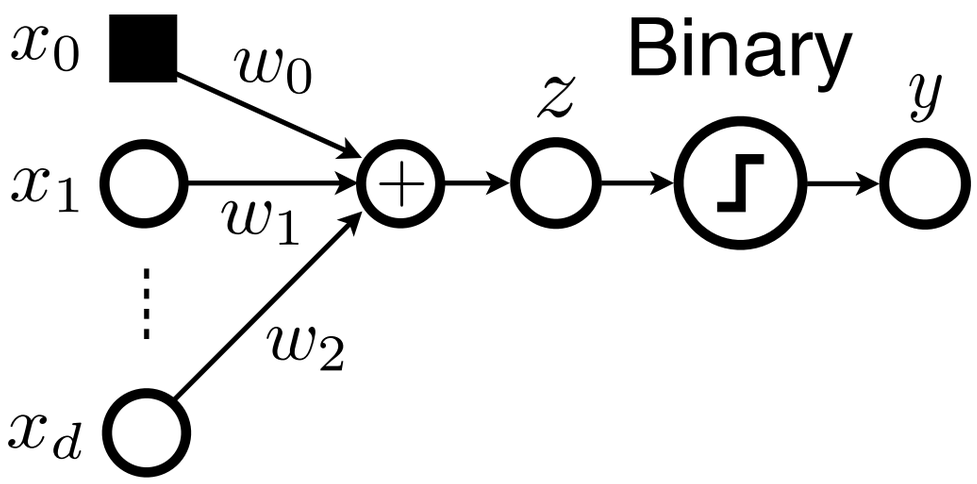
$$z = \sum_{i=1}^{d} w_i x_i$$

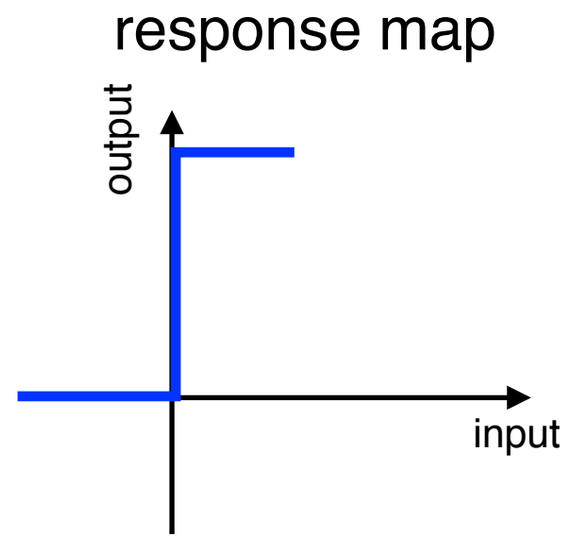
$$y = \begin{cases} 1, & \text{if } z \geq -w_0 \\ 0, & \text{otherwise} \end{cases}$$

$$z = \sum_{i=0}^{d} w_i x_i \qquad (x_0 = 1)$$

$$y = \begin{cases} 1, & \text{if } z \geq 0 \\ 0, & \text{otherwise} \end{cases}$$







MODELING NEURONS

Rectified Linear Neurons (ReLU)

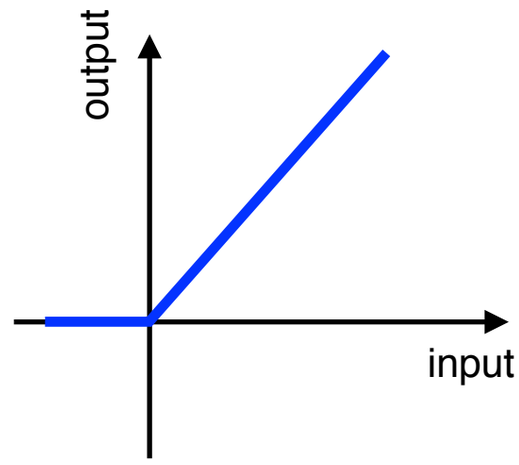
$$z = \sum_{i=0}^{d} w_i x_i \qquad y = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

ReLU: $y = \max(0, z)$

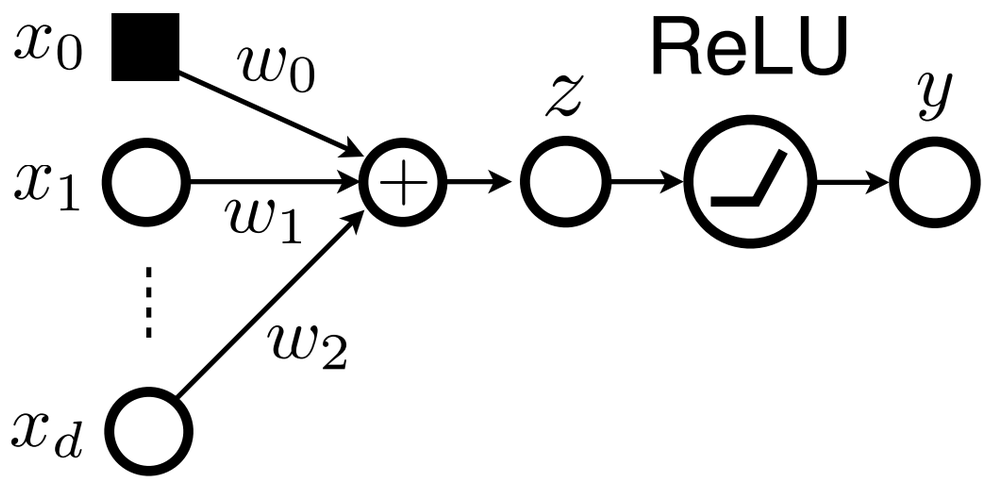
Noisy ReLu: $y = \max(0, z + \epsilon) \quad \epsilon \sim \mathcal{N}(0, \sigma)$

Leaky ReLu: $y = \begin{cases} z, & \text{if } z > 0 \\ az, & \text{otherwise} \end{cases}$

response map



Note: Output is a nonlinear function of input,
but is linear above zero

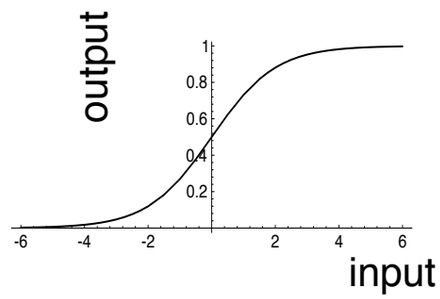


MODELING NEURONS

Sigmoid Neurons

$$z = \sum_{i=0}^{d} w_i x_i \qquad y = \frac{1}{1 + e^{-z}}$$

response map

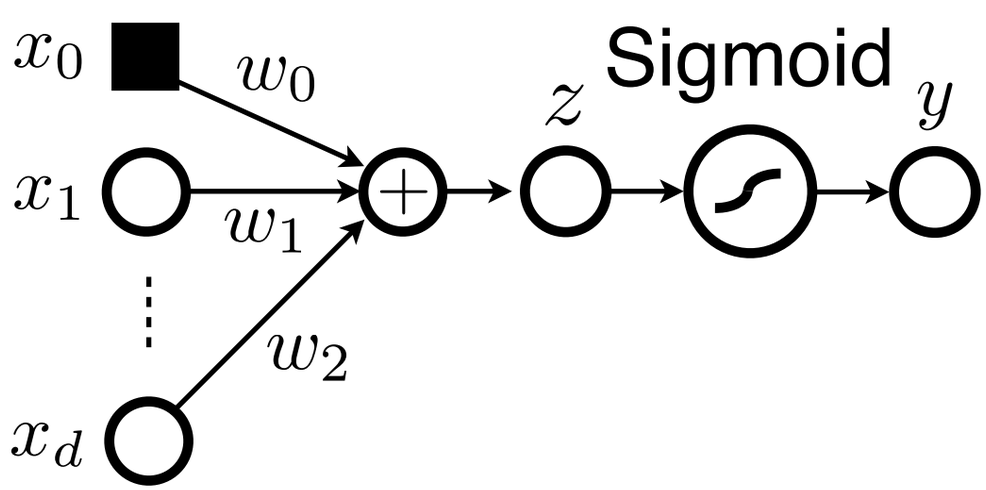


Logistic Function

$$y = \sigma(z)$$

$$y = \sigma\Big(\sum_{i=0}^{d} w_i x_i\Big) \qquad \frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$$

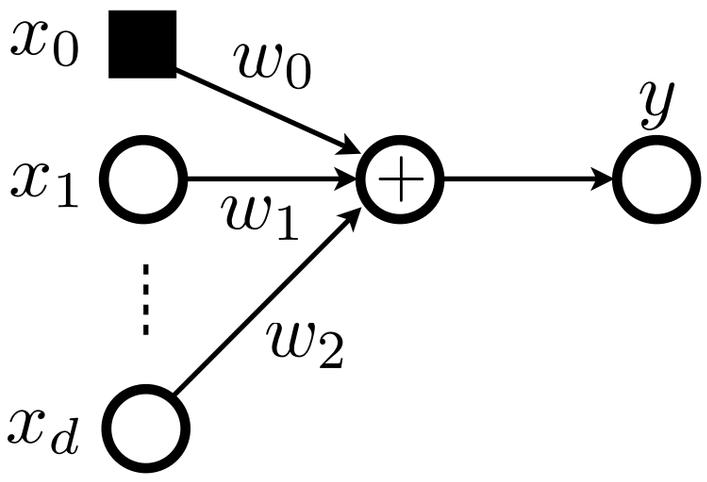
Note: Real valued smooth output.



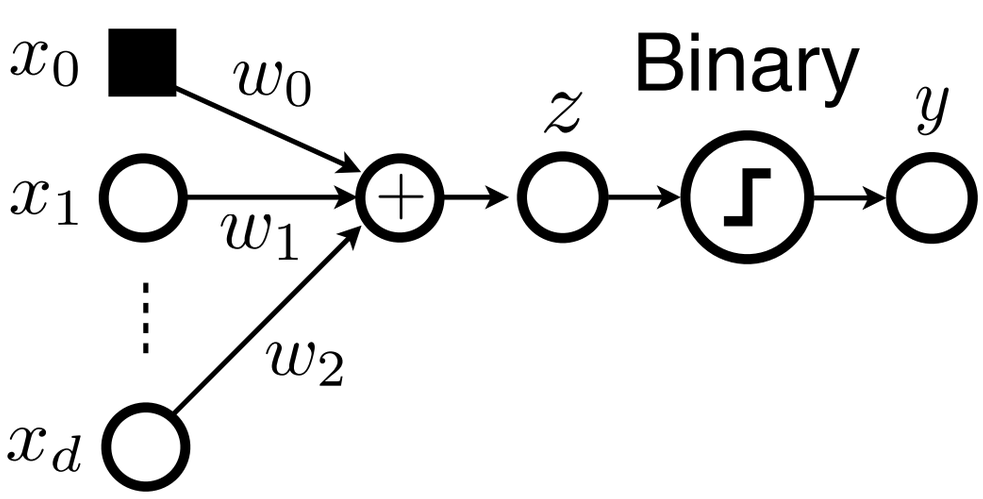
MODELING NEURONS

Perceptron Architecture

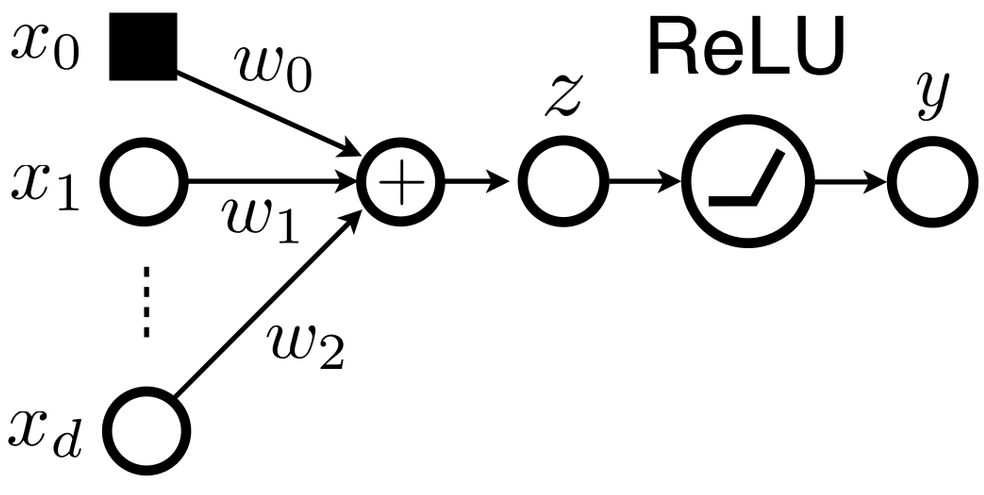
Linear Neuron



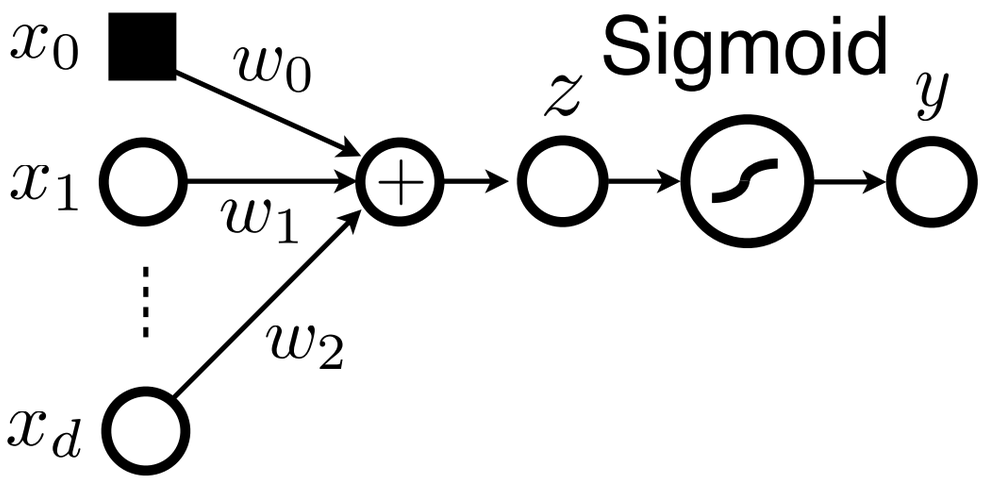
Binary Threshold Neuron



Rectified Linear Neuron



Sigmoid Neuron



MODELING NEURONS

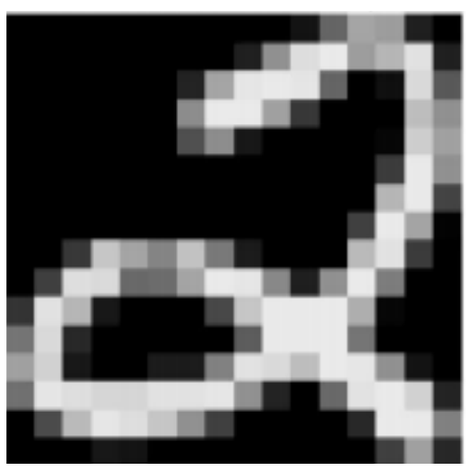
Supervised Learning Example: 2-Category Image Classification

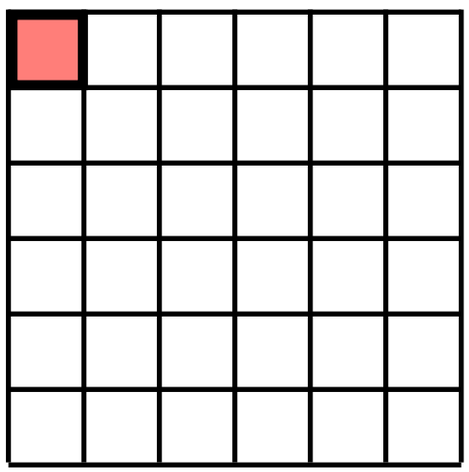
Training Pair: $(\mathbf{x}_k, y_k)_{k=1}^K$

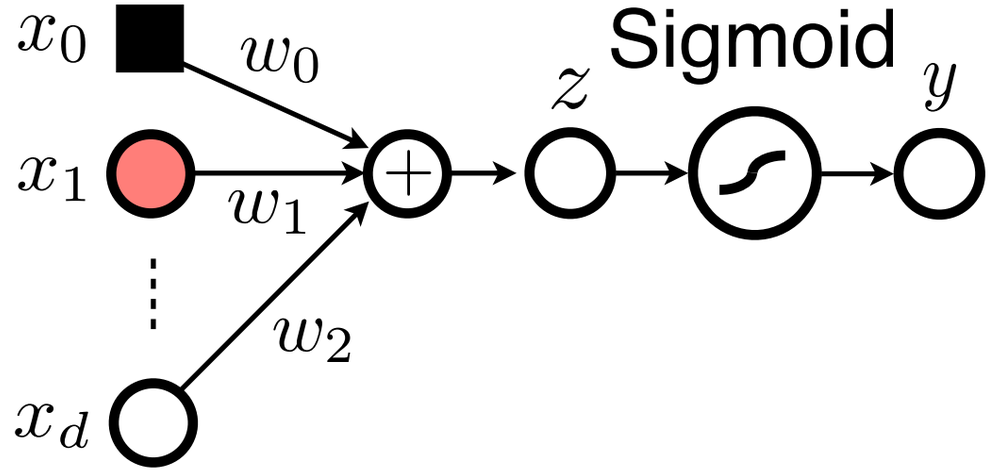
Input: $\mathbf{x}_k \in \mathbb{R}^d$

Output: $\mathbf{y}_k \in \mathbb{R}$

NN Classifier: $y = \sigma\left(\sum_{i=0}^{d} w_i x_{ki}\right) = h(\mathbf{x}_k; \mathbf{w})$







Loss Function: $E(\mathbf{w}) := \sum_{k=1}^{K} \left(h(\mathbf{x}_k; \mathbf{w}) - y_k\right)^2$

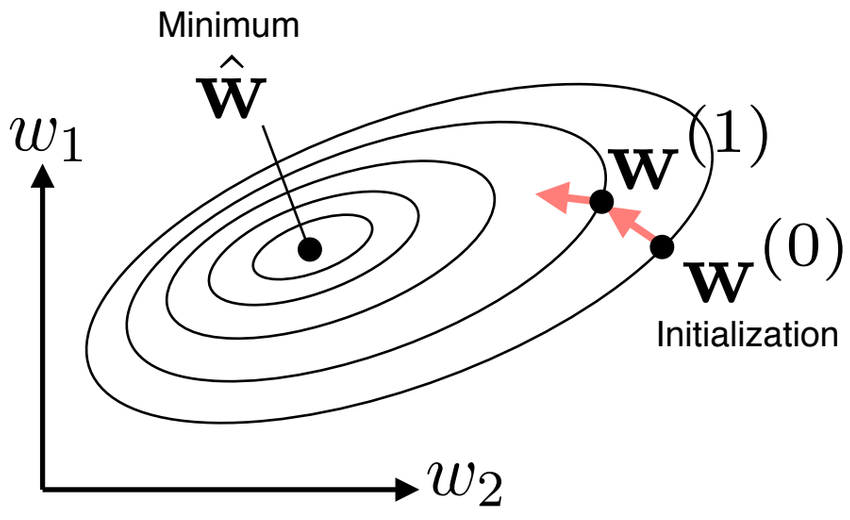
What are the right weights?

TRAINING A NEURAL NETWORK

Gradient Descent: Take a Step Proportional to the Negative of the Gradient Direction

$$y = \sigma\Big(\sum_{i=0}^{d} w_i x_{ki}\Big) = h(\mathbf{x}_k; \mathbf{w}) \qquad E(\mathbf{w}) := \sum_{k=1}^{K} \Big(h(\mathbf{x}_k; \mathbf{w}) - y_k\Big)^2$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w})$$



Learning Rate

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}^{(t)}}$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \left(\frac{\partial E(\mathbf{w})}{\partial w_0} \quad \frac{\partial E(\mathbf{w})}{\partial w_1} \quad \cdots \quad \frac{\partial E(\mathbf{w})}{\partial w_d}\right)$$ gradient

TRAINING

Optimizing Weights for a Single Layer Neural Network

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \left(\sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) - y_k \right)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial E(\mathbf{w})}{\partial w_0} & \frac{\partial E(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial E(\mathbf{w})}{\partial w_d} \end{pmatrix}$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = \frac{\partial}{\partial w_j} \sum_{k=1}^{K} (h(\mathbf{x}_k) - y_k)^2 \qquad 0 \le j \le d$$

$$= \sum_{k=1}^{K} 2(h(\mathbf{x}_k) - y_k) \frac{\partial}{\partial w_j} (h(\mathbf{x}_k) - y_k) \qquad \text{Chain Rule}$$

$$= \sum_{k=1}^{K} 2(h(\mathbf{x}_k) - y_k) \frac{\partial}{\partial w_j} \sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big)$$

$$= \sum_{k=1}^{K} 2(h(\mathbf{x}_k) - y_k) \sigma'\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) \frac{\partial}{\partial w_j} \sum_{i=0}^{d} w_i x_{k,i}$$

$$= \sum_{k=1}^{K} 2(h(\mathbf{x}_k) - y_k) \sigma'\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) x_{k,j}$$

Partial derivate of *E(w)* w.r.t one weight

TRAINING

Example: Logistic Activation

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{k=1}^{K} \left(\sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) - y_k \right)^2 \qquad h(\mathbf{x}_k) = \sigma\Big(\sum_{i=0}^{d} w_i x_i \Big)$$

w.r.t one weight

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = \sum_{k=1}^{K} 2(h(\mathbf{x}_k) - y_k)\sigma'\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) x_{k,j} \qquad 0 \le j \le d$$

$$= \sum_{k=1}^{K} 2\left(\sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) - y_k \right) \sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) \left(1 - \sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) \right) x_{k,j}$$

gradient

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial E(\mathbf{w})}{\partial w_0} & \frac{\partial E(\mathbf{w})}{\partial w_1} & \cdots & \frac{\partial E(\mathbf{w})}{\partial w_d} \end{pmatrix}$$

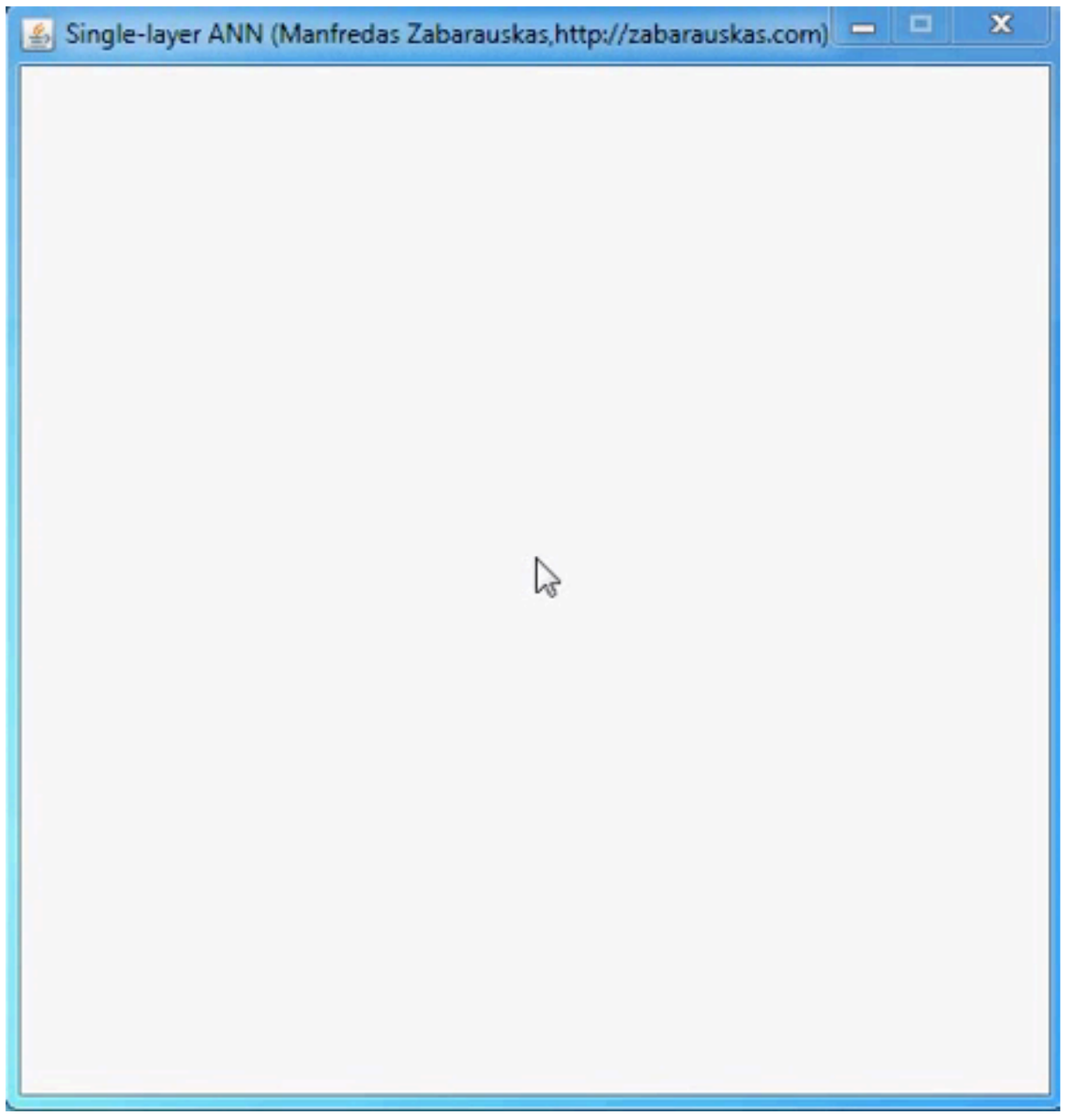
$$= 2 \sum_{k=1}^{K} \left(\sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) - y_k \right) \sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) \left(1 - \sigma\Big(\sum_{i=0}^{d} w_i x_{k,i} \Big) \right) \mathbf{x}_k$$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \left. \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \right|_{\mathbf{w}^{(t)}}$$

$$\mathbf{w}^{(t+1)} := \mathbf{w}^{(t)} - 2\eta \sum_{k=1}^{K} (h(\mathbf{x}_k) - y_k) h(\mathbf{x}_k)(1 - h(\mathbf{x}_k)) \mathbf{x}_k \Big|_{\mathbf{w}^{(t)}}$$

TRAINING

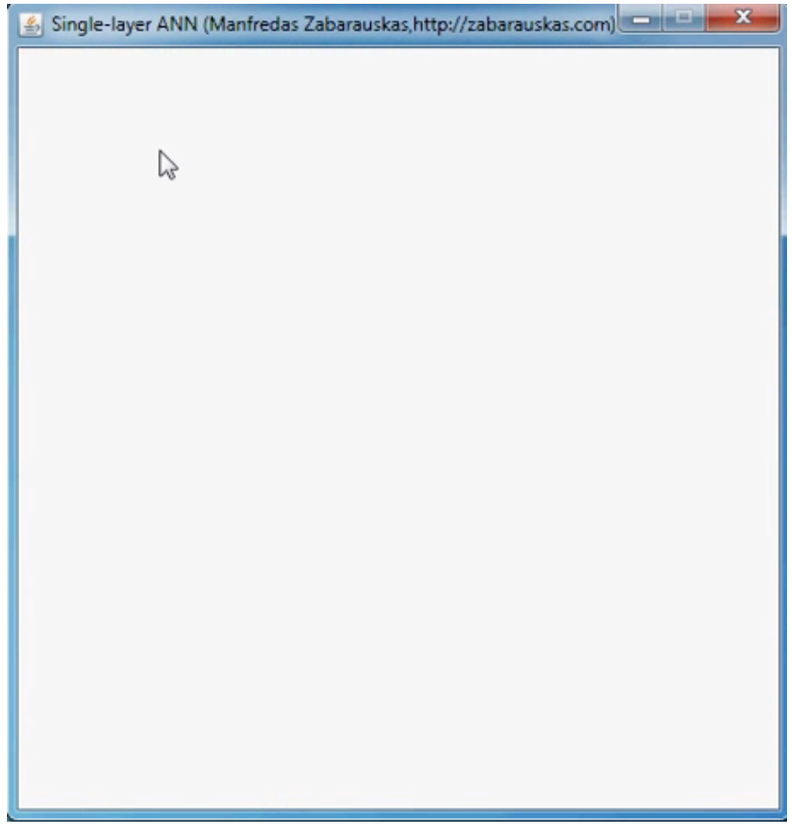
Optimizing the weights of a Single-Layer Neural Network



Visualization Credit: Manfred Zabarauskas

LIMITATIONS OF THE PERCEPTRON

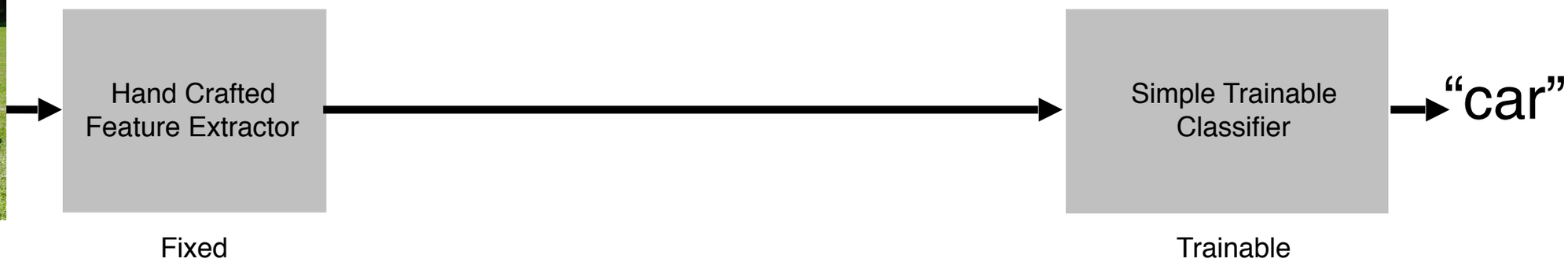
Linear Separability



Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

LIMITATIONS OF THE PERCEPTRON

Linear Separability



If you craft enough good features, perceptrons can be powerful.

Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

“COMPLICATED” CLASSIFIERS

Linear Combination of Functions

Given a library of simple function

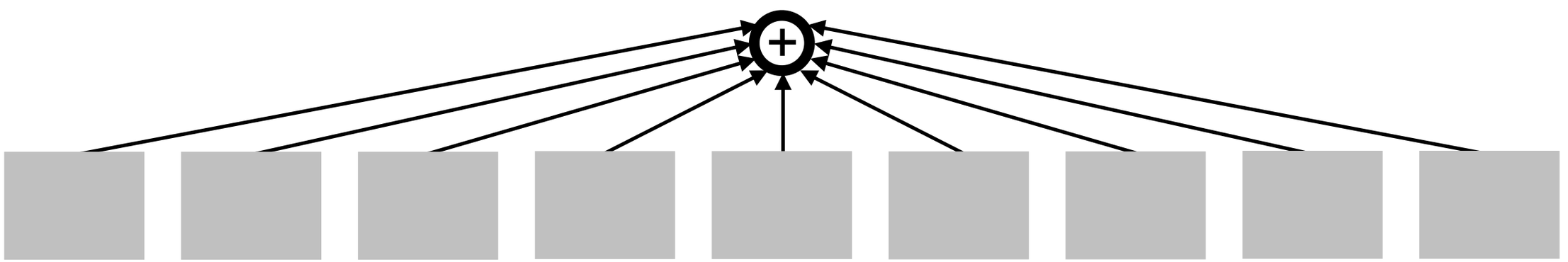
$$\sin(x)$$
$$\log(x)$$
$$x^4$$
$$\tanh(x)$$
$$\vdots$$

Compose into a “complicated” function

IDEA 1: Linear Combination

Boosting, Kernels, …

$$f(x) = \sum_i \alpha_i g_i(x)$$



Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

“COMPLICATED” CLASSIFIERS

Composition: Functions of Functions

Given a library of simple function

$$\sin(x)$$
$$\log(x)$$
$$x^4$$
$$\tanh(x)$$
$$\vdots$$

Compose into a “complicated” function

IDEA 2: Compositionality
Deep Learning

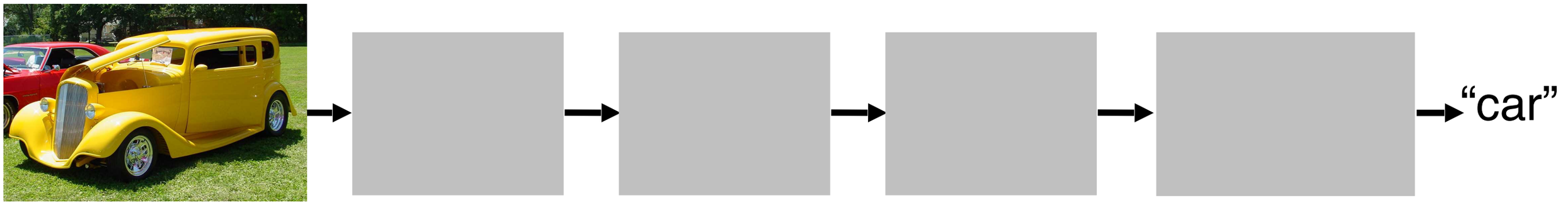
$$f(x) = g_1(g_2(\cdots g_n(x)))$$

$$f(x) = \sin(\log(\cdots \tanh(x))^3))$$

Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

HIERARCHICAL COMPOSITIONALITY

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

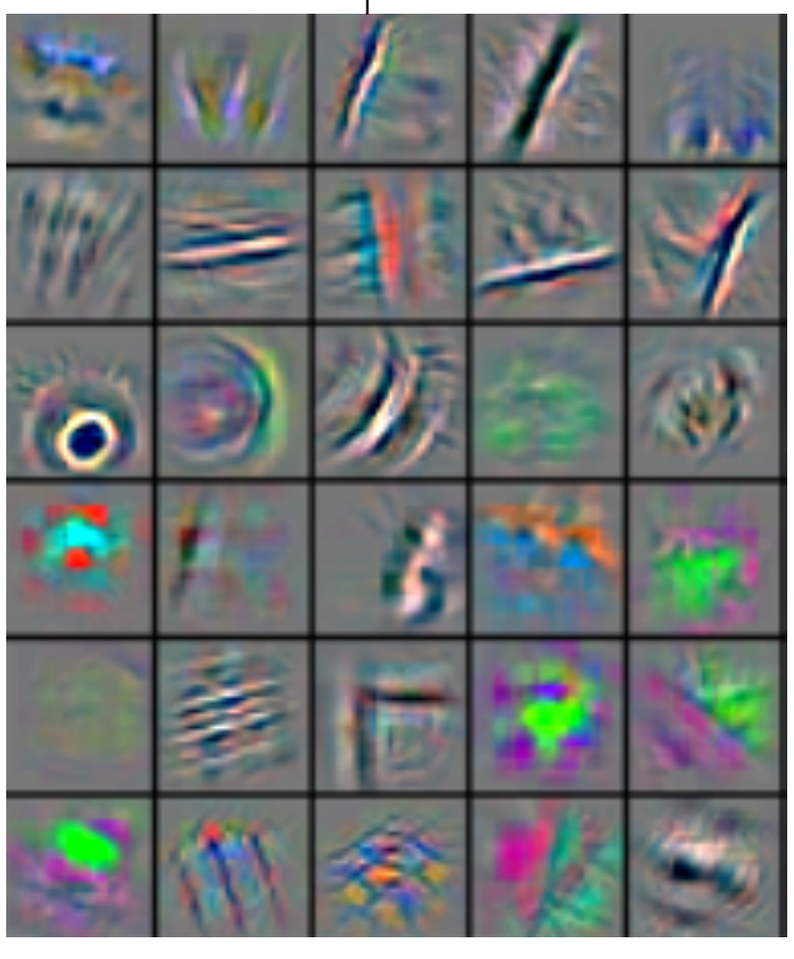


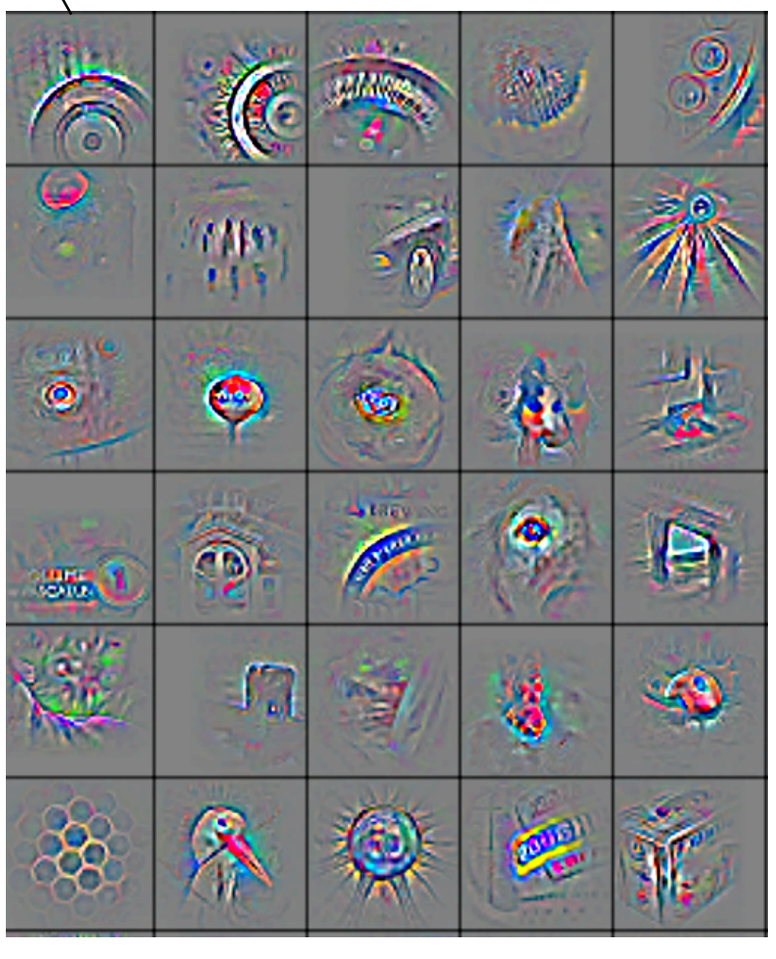
Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

HIERARCHICAL COMPOSITIONALITY

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Low-Level Feature → Mid-Level Feature → High-Level Feature → Trainable Classifier → “car”





Based on a slide by Marc'Aurelio Ranzato, Yann

HIERARCHICAL COMPOSITIONALITY

Shallow vs Deep Learning

“Shallow” Learning

Hand Crafted Feature Extractor

Fixed

Simple Trainable Classifier

Trainable

“car”

“Deep” Learning

Trainable Feature Transform

Trainable Feature Transform

Trainable Feature Transform

Trainable Feature Transform

“car”

Learned Internal Representation

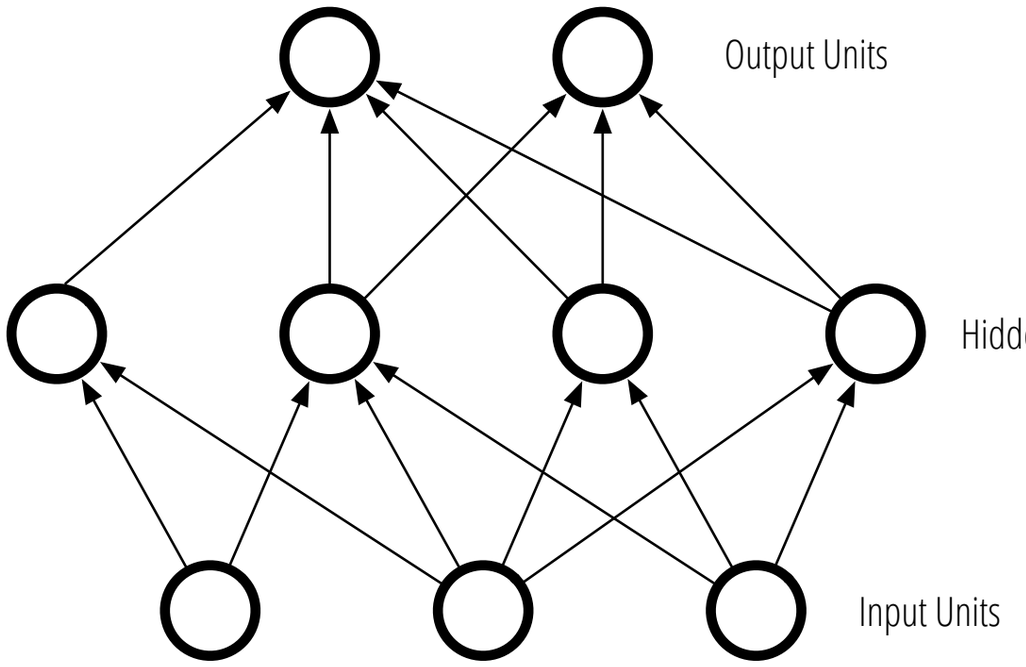
Learned Internal Representation

Learned Internal Representation

Based on a slide by Marc'Aurelio Ranzato, Yann LeCun

NEURAL NETWORK ARCHITECTURES

Feed-forward Neural Networks

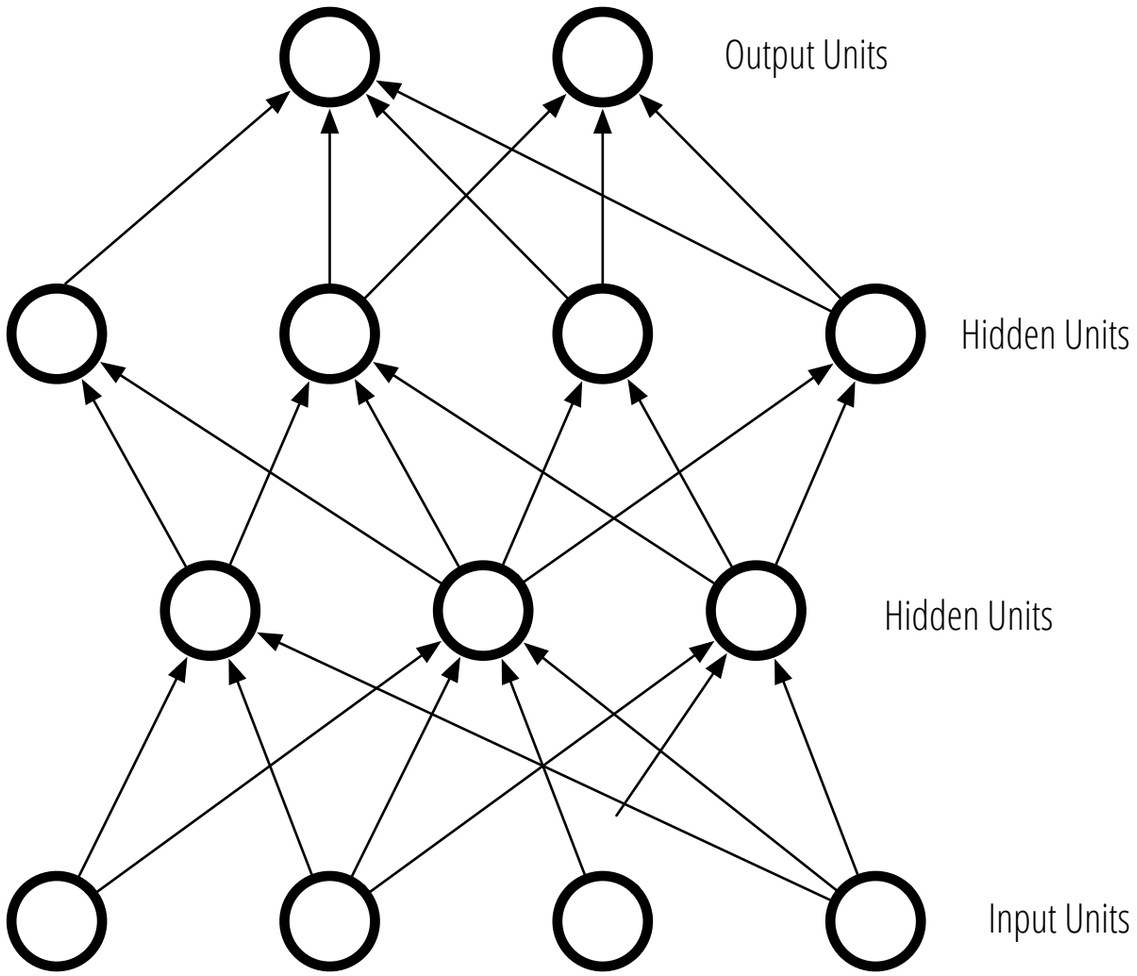


Information enters through the input units and flows in one direction, through hidden layers, to the output

Each hidden layer changes similarity relationships, producing a new representation of the input

NEURAL NETWORK ARCHITECTURES

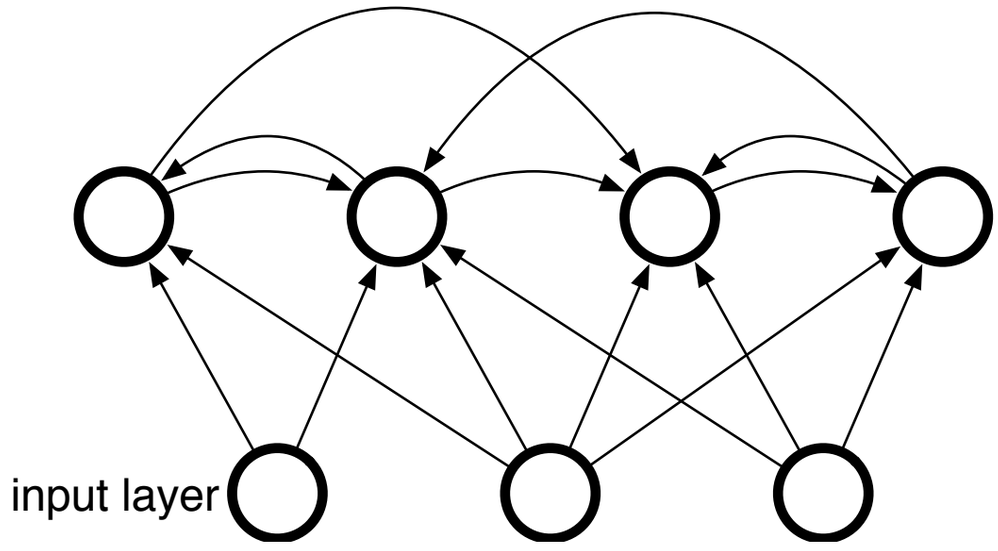
“Deep” Feed-forward Neural Networks



Networks are considered “deep” if they have more than one hidden layer

NEURAL NETWORK ARCHITECTURES

Recurrent Neural Networks

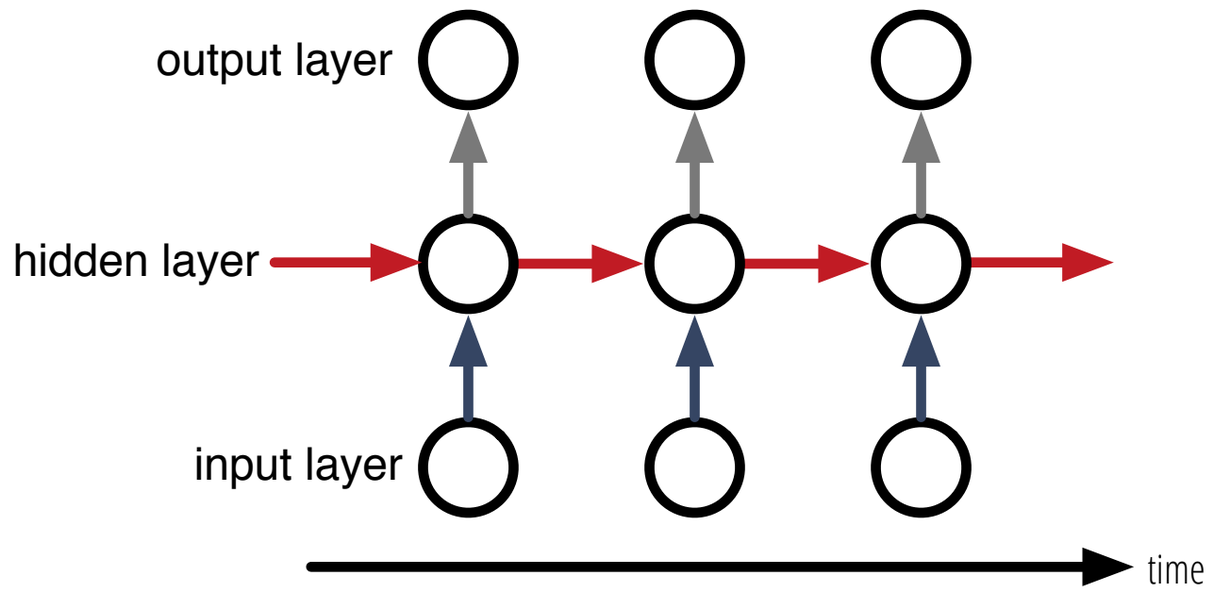


Recurrent Networks have directed cycles in their topology—a form of memory

Recurrent Networks are naturally “deep”; multiple hidden layers are a special case

NEURAL NETWORK ARCHITECTURES

Recurrent Neural Networks are a natural model for time series data

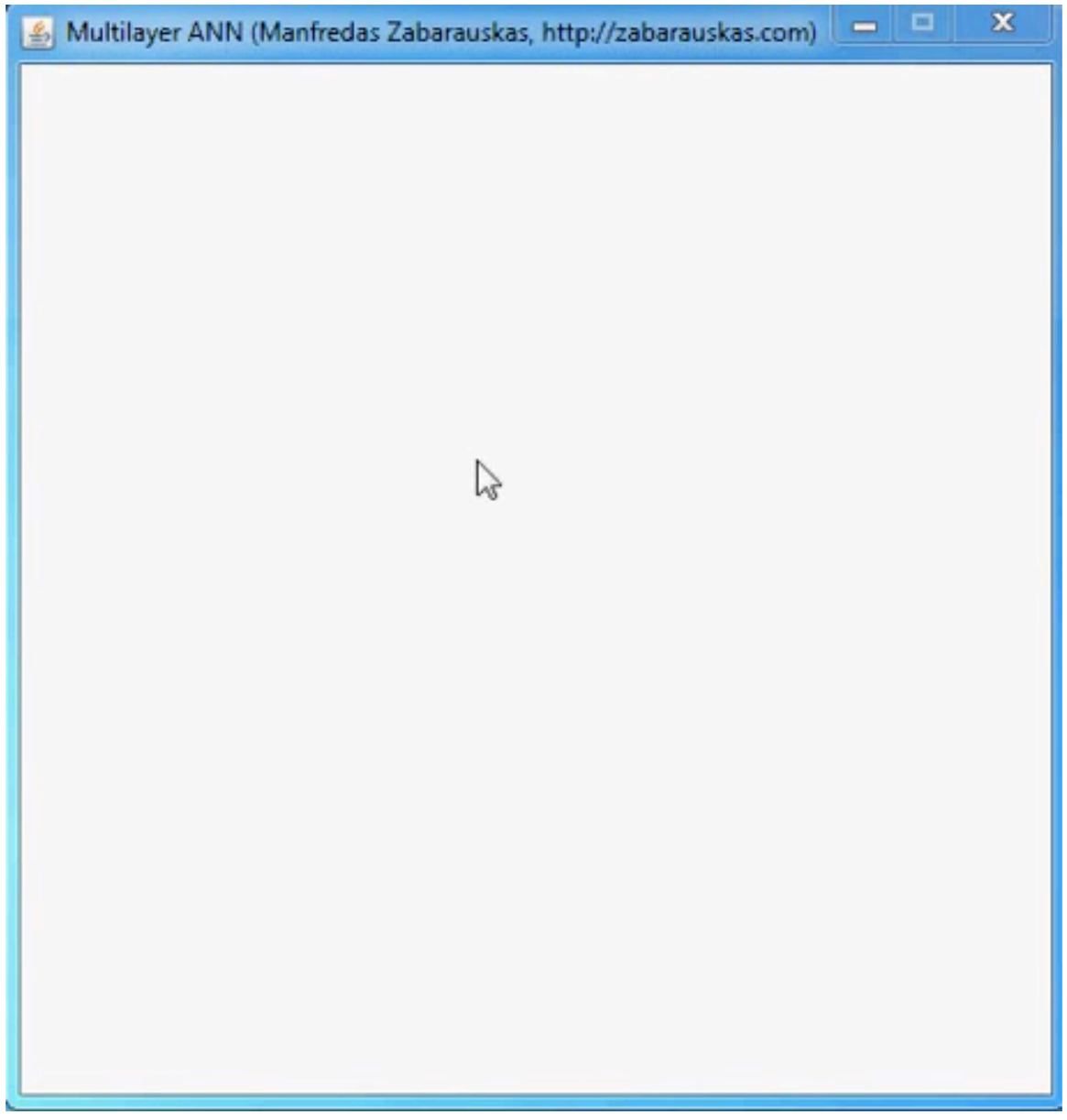


Input at each time slice; output at each time slice; very deep in time with a hidden layer at each time slice

Same weights at each time slice; they can have a long memory

TRAINING A MULTILAYER PERCEPTRON

Optimizing Weights for a Multi-Layer Neural Network



Visualization Credit: Manfred Zabarau