Data Mining & Machine Learning

CS37300
Purdue University

Oct 20, 2023

Neural Networks (introduction)

Tasks Where Neural Network Learning Excels

- Image/Video Recognition

Image by Neurala

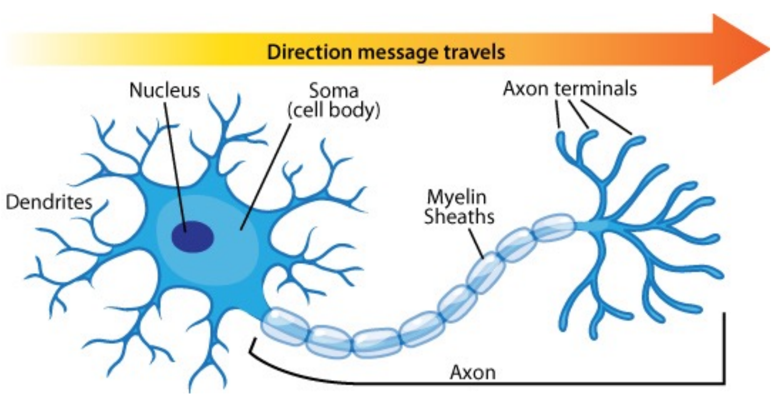
- Speech Recognition

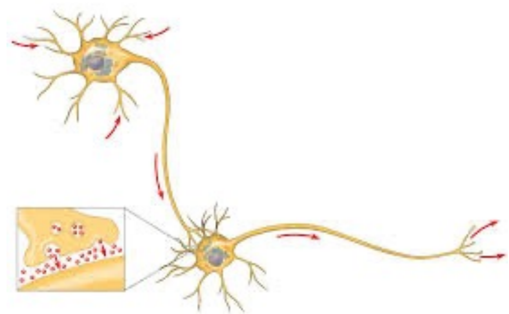
quick brown fox

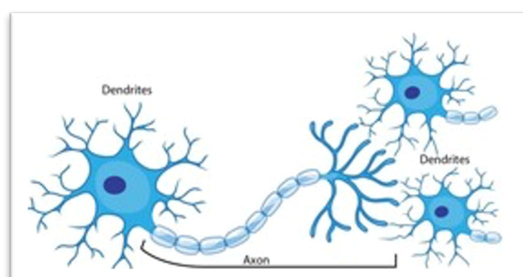
- Text Recognition / Prediction
  - The quick brown fox jumps over the lazy dog

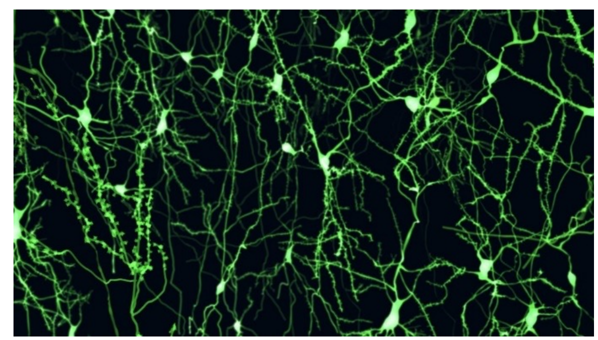
Neurons

- Neuron
  - takes inputs from many other sources (neurotransmitters into dendrites)
  - Each input signal is attenuated by some learned amount
  - If the aggregate of these inputs exceeds some threshold, the neuron “fires”, sending out a signal (neurotransmitters from its axon terminals) to all the other neurons connected to it



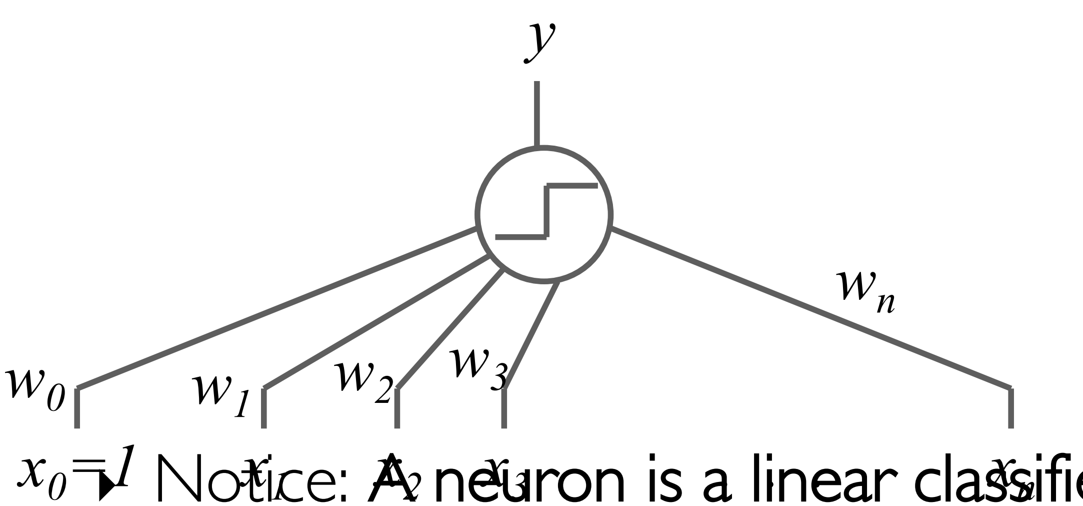






Neurons

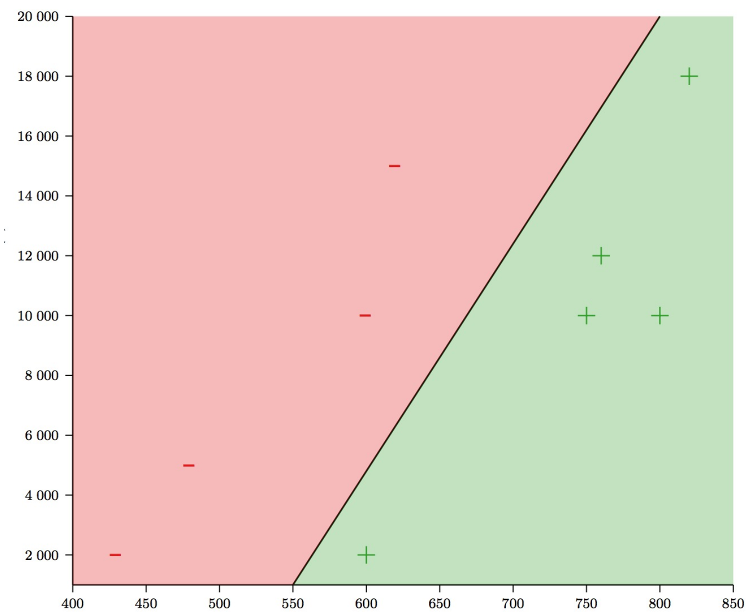
- Abstracting this: A mathematical neuron
  - Takes numerical inputs from other sources (either input units or other neurons)
  - Each input signal is weighted by a learned real value
  - If the sum of weighted inputs exceeds a threshold, neuron outputs 1 (fire), else 0 (don't fire).



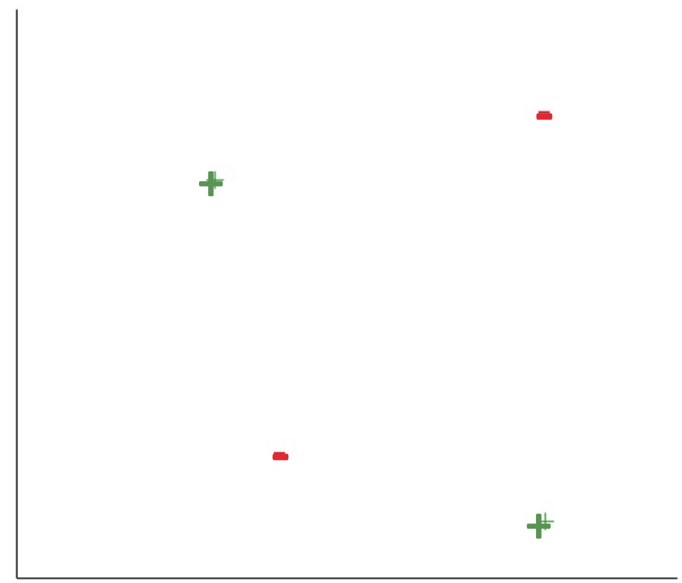
$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- Notice: A neuron is a linear classifier!

Limitation of perceptron

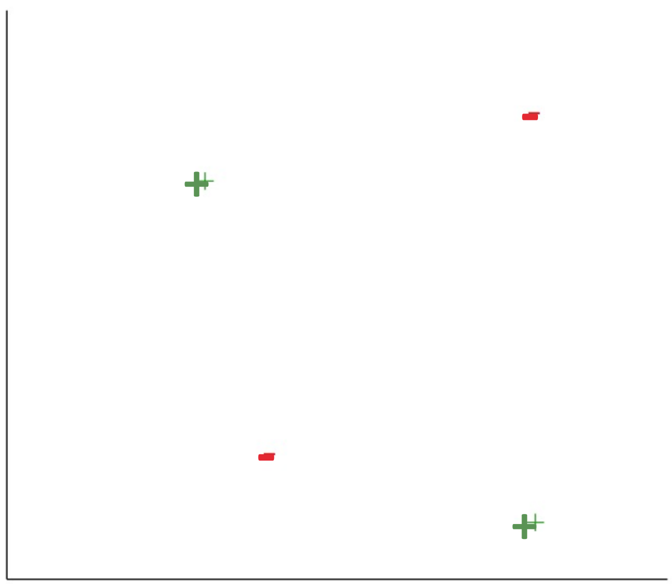


How about this?

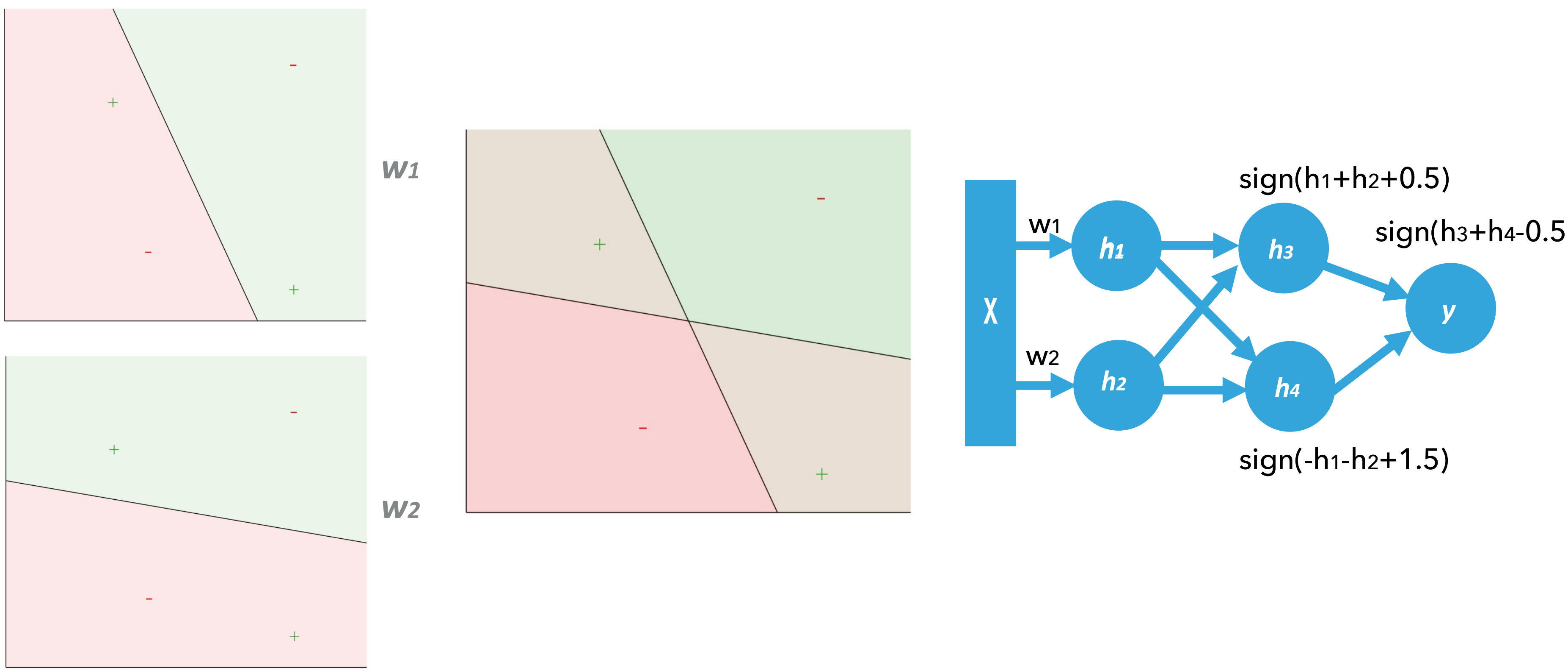


Perceptron is suitable for classifying a set of linearly separable data

From perceptron to multi-layer neural networks



From perceptron to multi-layer neural networks

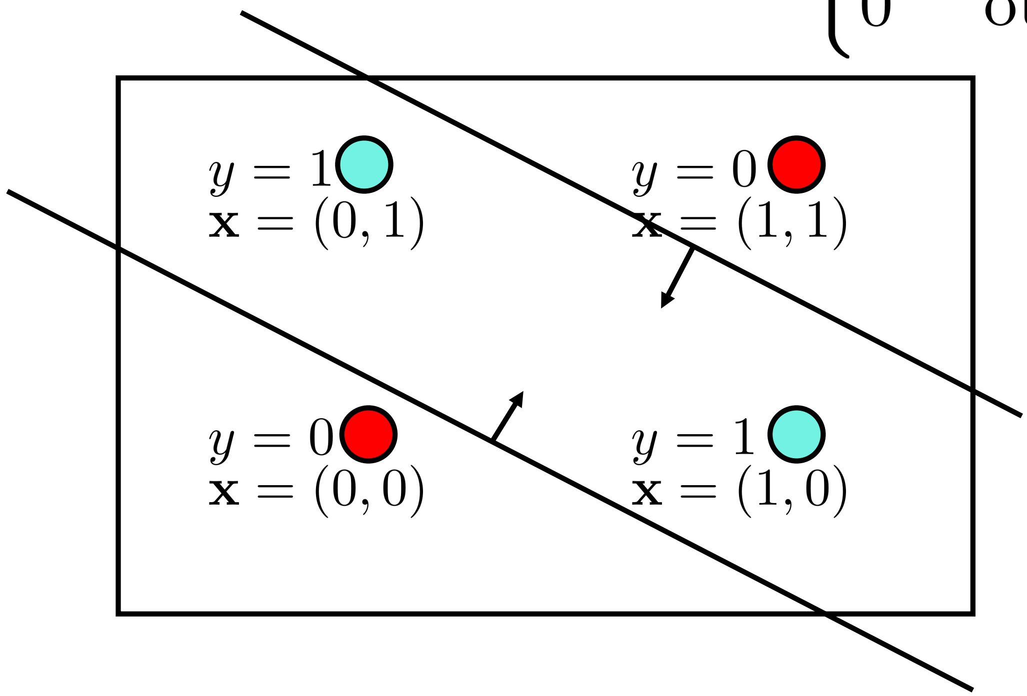


Example: The XOR Problem

Can linear classifiers learn XOR?

$$\text{features: } \mathbf{x} = (x_1, x_2),\ x_1, x_2 \in \{0, 1\}$$

$$f(\mathbf{x}) = \mathrm{XOR}(x_1, x_2) = \begin{cases} 1 & \text{if } x_1 \neq x_2 \\ 0 & \text{otherwise} \end{cases}$$



Any line with (0,1),(1,0) classified 1 and (0,0) as 0 must classify (1,1) as 1

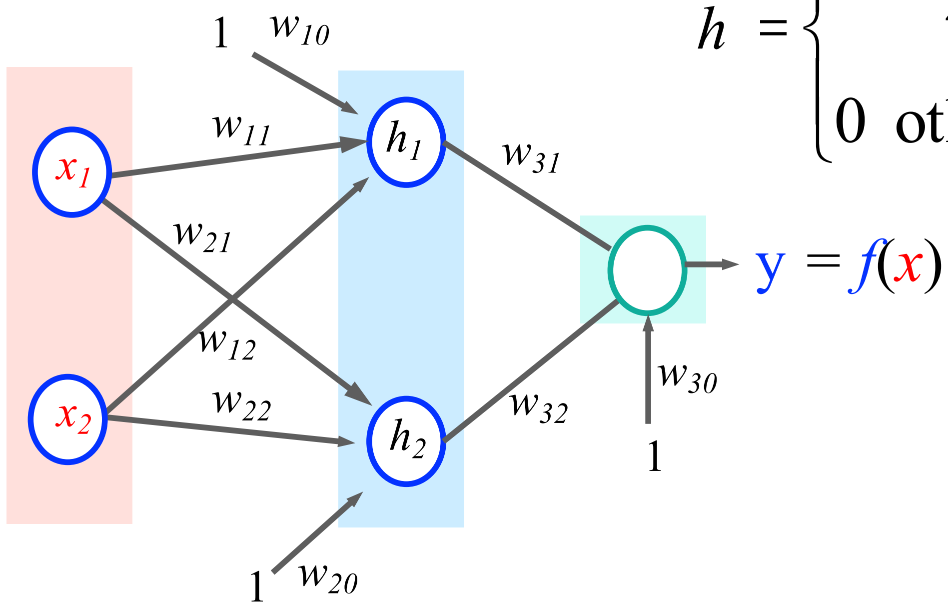
Any line with (0,1),(1,0) classified 1 and (1,1) as 0 must classify (0,0) as 1

Therefore, linear classifiers cannot learn XOR

Example: Solving the XOR Problem

$$h = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Network Topology:
2 hidden nodes
1 output



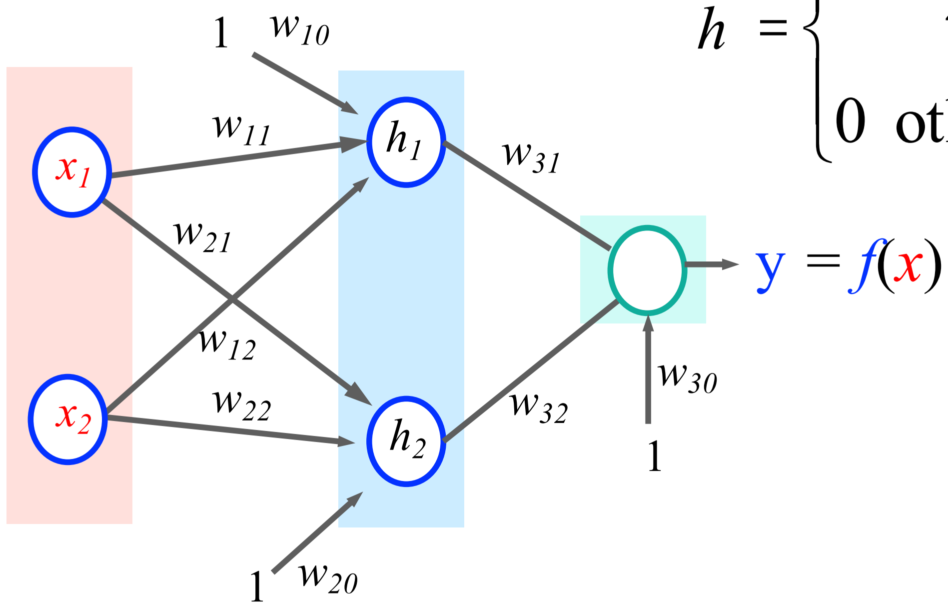
Desired behavior:

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

Example: Solving the XOR Problem

$$h = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Network Topology:
2 hidden nodes
1 output



Desired behavior:

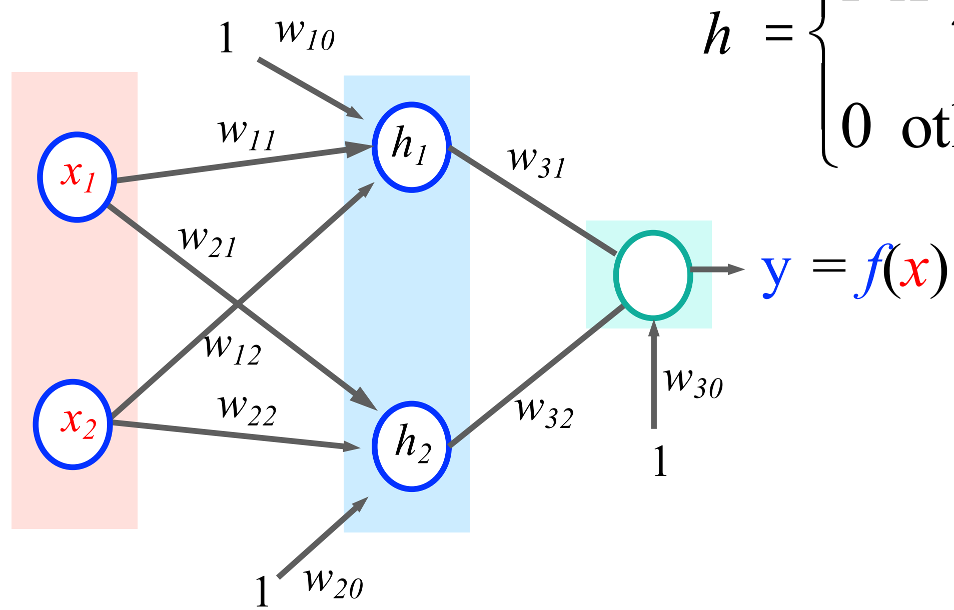
x_1	x_2	h_1	h_2	y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

$$h_1 = \text{AND},\ h_2 = \text{OR},$$
$$h = (x_1 \text{ OR } x_2) \text{ AND NOT } (x_1 \text{ AND } x_2)$$

Example: Solving the XOR Problem

$$h = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 \text{ otherwise} \end{cases}$$

Network
Topology:
2 hidden nodes
1 output



Desired behavior:

x_1	x_2	h_1	h_2	y
0	0	0	0	0
1	0	0	1	1
0	1	0	1	1
1	1	1	1	0

$h_1 = \text{AND}, \; h_2 = \text{OR},$

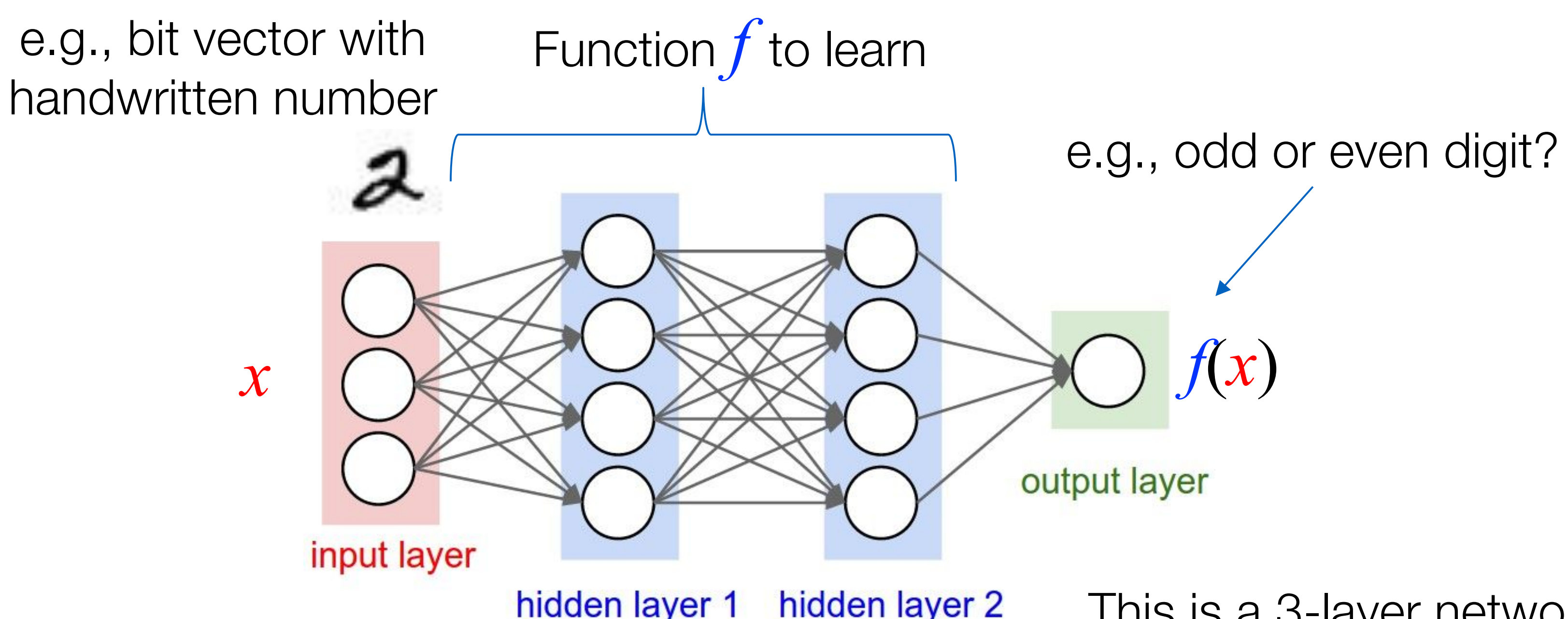
$h = (x_1 \text{ OR } x_2) \text{ AND NOT } (x_1 \text{ AND } x_2)$

Weights:

- h_1=*AND*: $w_{11} = w_{12} = 1$, $w_{10} = -3/2$
- h_2=*OR*: $w_{21} = 1$, $w_{22} = 1$, $w_{20} = -1/2$
- h=*XOR*: $w_{31} = -1$, $w_{32} = 1$, $w_{30} = -1/2$

Neural Network (Multilayer Perceptron)

- More general than linear classifier
- *Input layer* is (integer or real-valued) vector, representing a data point
- *Hidden layers* represent latent (hidden) variables (hard to interpret)
- *Output layer* represents prediction we want to make

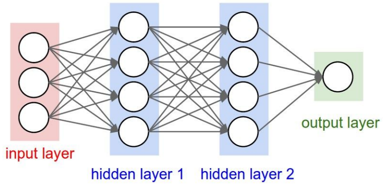


This is a 3-layer network

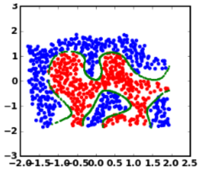
History

- In ML, there were multiple waves of interest in neural nets
  - 1940s-1960s, mathematical models and physical implementations of single neurons (before computers were widely available)

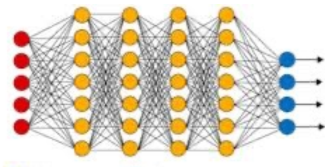
  - 1969 critical article by Minsky & Papert halted most work on neural nets. people focused on statistical and logical approaches for a while instead
  - 1985 A technique for training multilayer neural networks (backpropagation) gave new life to neural networks, and people were again excited about them for a while

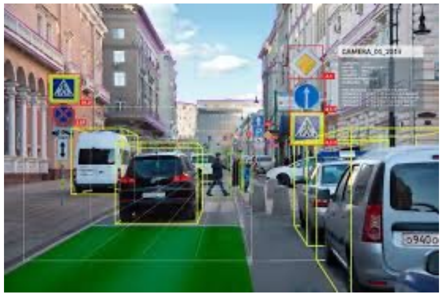


  - 1998 Support Vector Machines / kernel methods arrived, seemed a much simpler, better understood solution than neural networks. (airplane wings vs bird wings debate)



  - 2010s-present New benefits observed for using “deep” neural networks (networks with many layers), leading a renewed wave of interest in neural networks





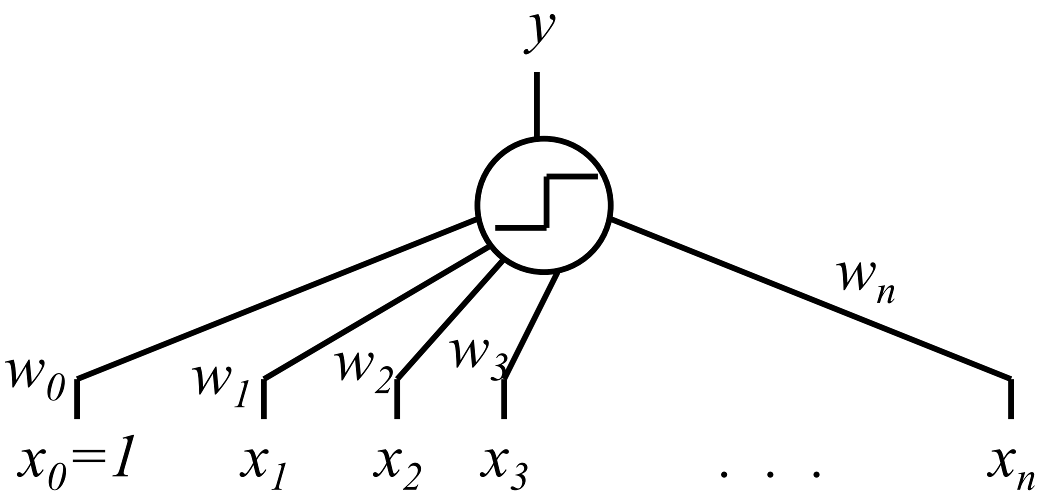
- Every one of these waves comes with lots of hype (neurons are cool)
- Despite the hype, there really are some applications where neural networks have always made sense: e.g., computer vision, speech processing

Neural Nets

- Pro: More general than linear classifiers
  - Not restricted to linear decision boundaries
  - Can also have multiple outputs (vector-valued outputs)
- Con: No simple, guaranteed training procedure
  - Use greedy, hill-climbing procedure to train
  - “Gradient descent”, “Backpropagation”

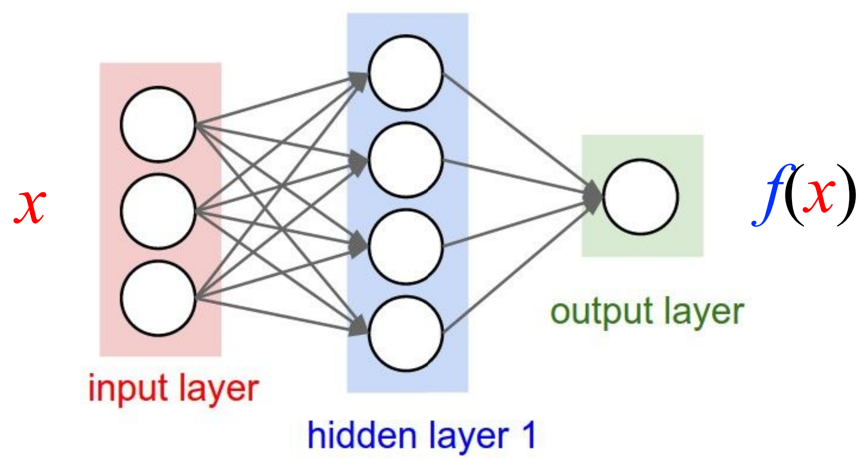
Continuous Activation Functions

Problems with Step Function Activation



$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

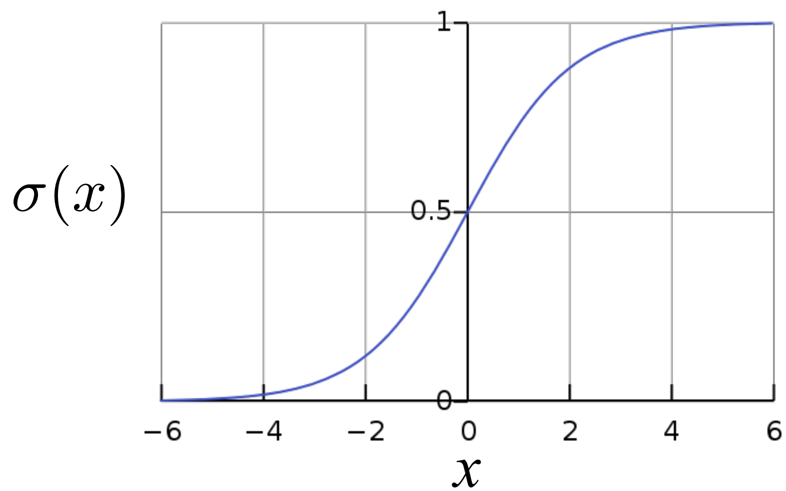
- The problem with this is that, even in a 2-layer network (one hidden layer), it is (provably) computationally hard to tune the parameters to fit some data sets



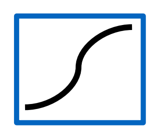
- Solution: relax step function to a continuous activation function

Logistic (neuron) Activation (non-linear filter)

- One common choice of activation function is the logistic function (aka, sigmoid function):
- If input is $x = \mathbf{w}^T\mathbf{x}$, the output will look like a probability $\sigma(\mathbf{w}^T\mathbf{x}) \in [0, 1]$
- $\mathrm{p}(\mathrm{y} = 1 \mid \mathbf{x}; \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}) \in [0, 1]$

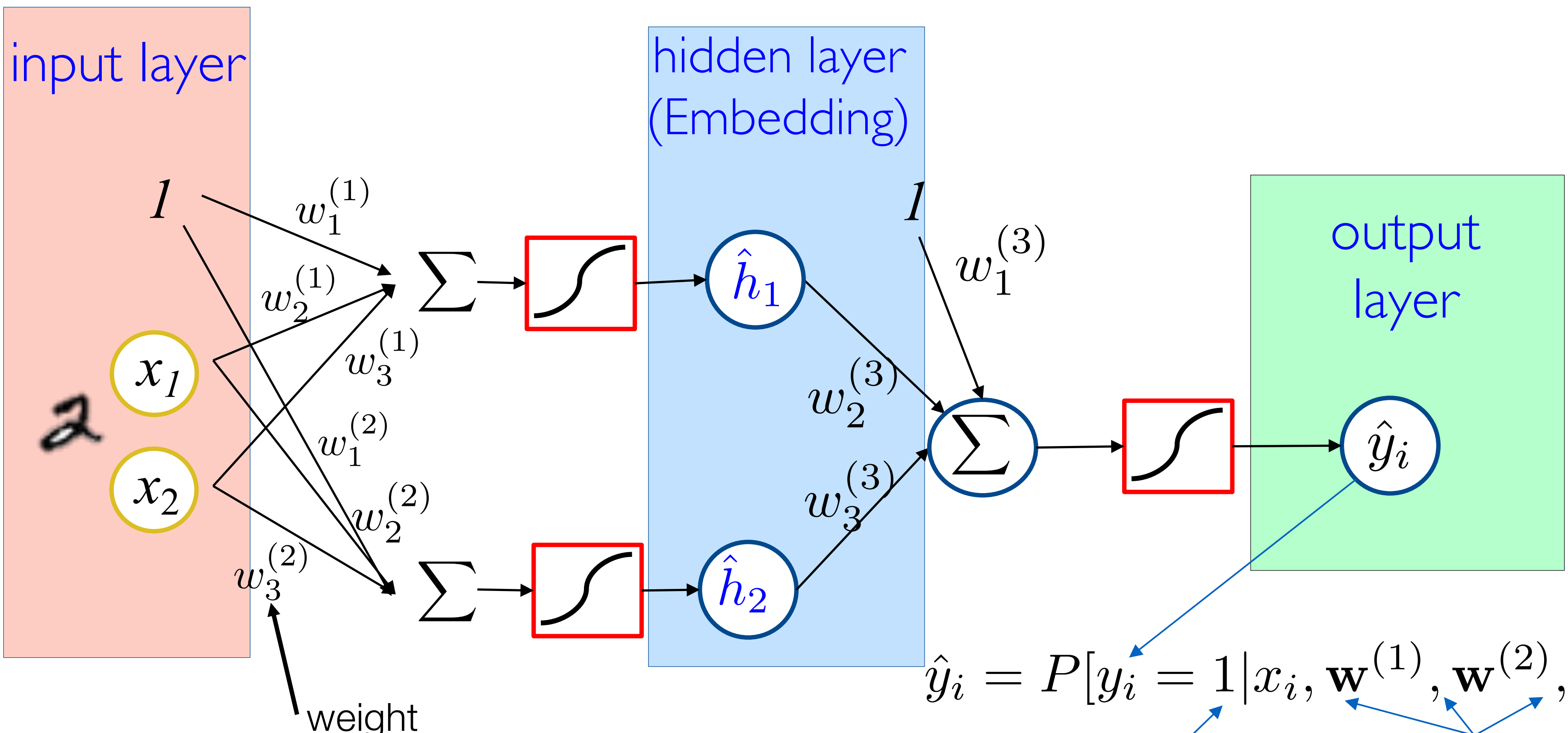


$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- We will represent the logistic function with the symbol: 

Neural Network Definitions

Neural networks are a combination of linear and non-linear functions

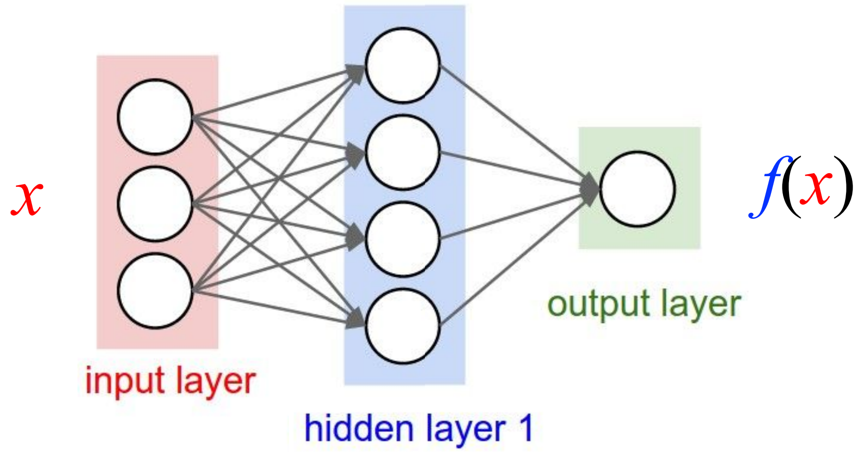


$$\hat{y}_i = P[y_i = 1 | x_i, \mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \mathbf{w}^{(3)}$$

Digit is odd = 1
Digit is even = 0

Need to learn parameter weights that best predict output

Neural Networks as Nonparametric Learners



- Even for 2-layer networks:
  As number of hidden units grows (called the width), the model becomes rich enough to fit any dataset and learn any function

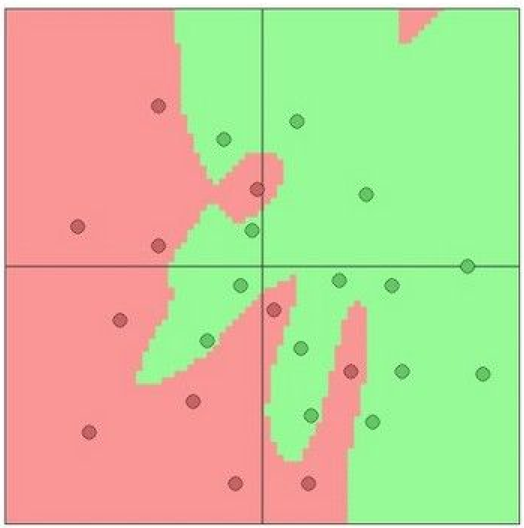
- Can think of number of hidden units as controlling model complexity (analogous to K in KNN, depth in decision trees, bandwidth in Gaussian kernel SVM)

- We can choose the number of hidden units using cross validation

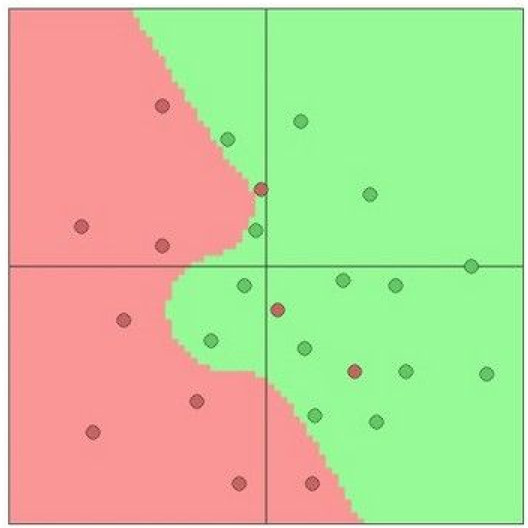
Neural Network Complexity

- More complex neural network models require care not to overfit the training data

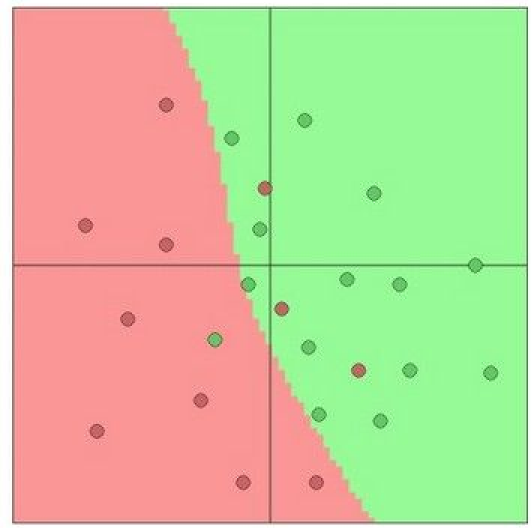
Neural net 1



Neural net 2



Neural net 3



Next Class: Searching for good neural network parameters (optimization)