UCF Center for Research in Computer Vision

CAP5415 Computer Vision

Yogesh S Rawat

yogesh@ucf.edu

HEC-241

Introduction to Neural Networks

Lecture 5

Where to go from edges?

Image

Gradients, edges, etc.

Features

Recognition

Scenes, places, objects, etc.

Detection

Object, action, etc.

Segmentation

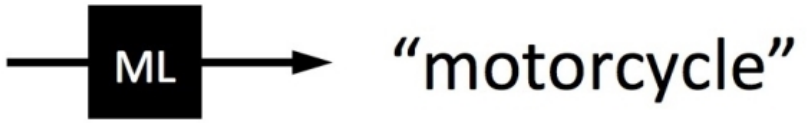
Object, semantic, etc.

Our goal in object classification

Our goal in object classification

Our goal in object classification

Our goal in object classification



Why is this hard?

You see this:

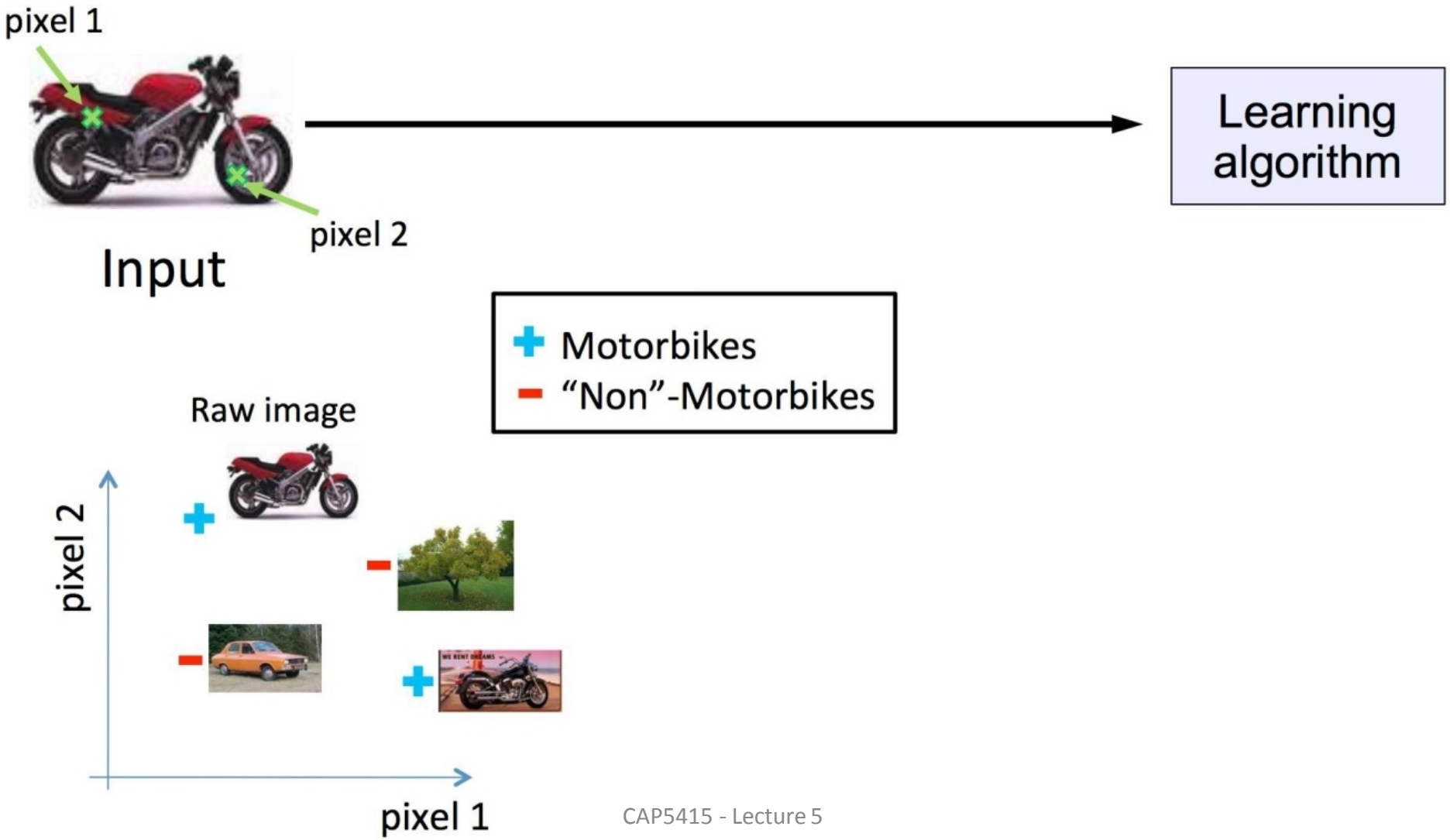
Why is this hard?

You see this:

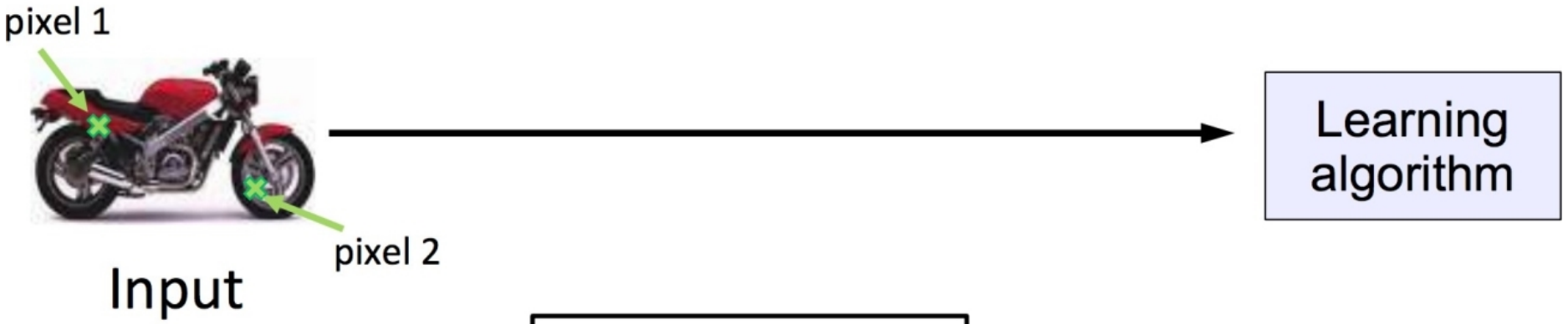
But the camera sees this:

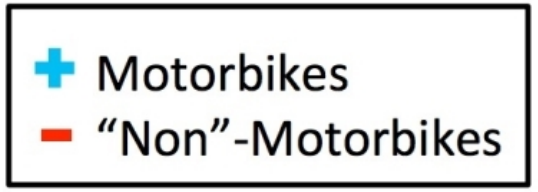
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Pixel-based representation

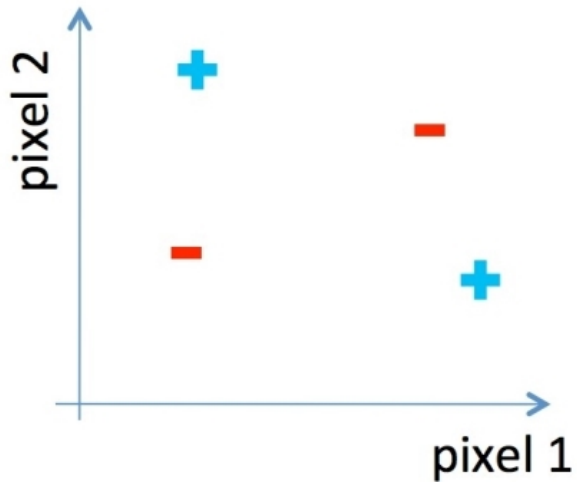


Pixel-based representation

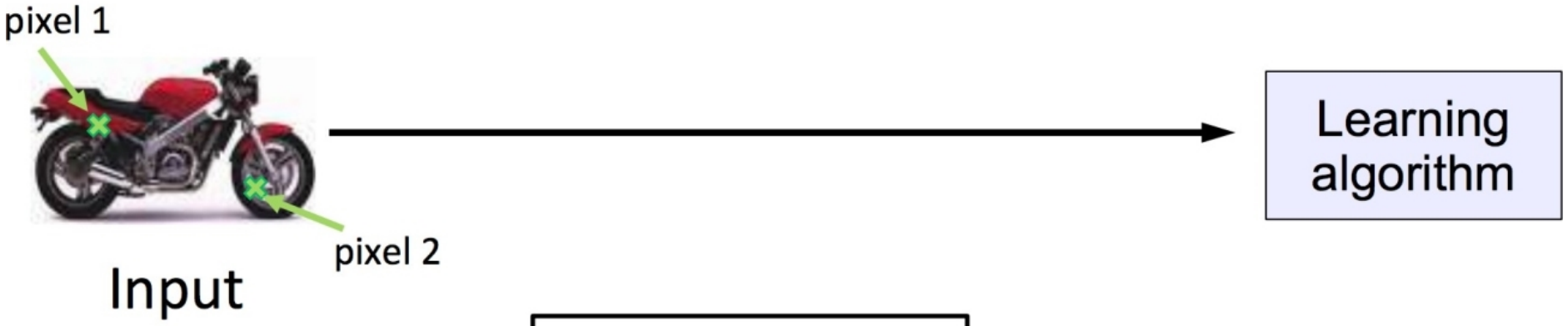


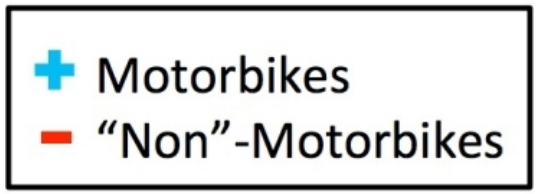


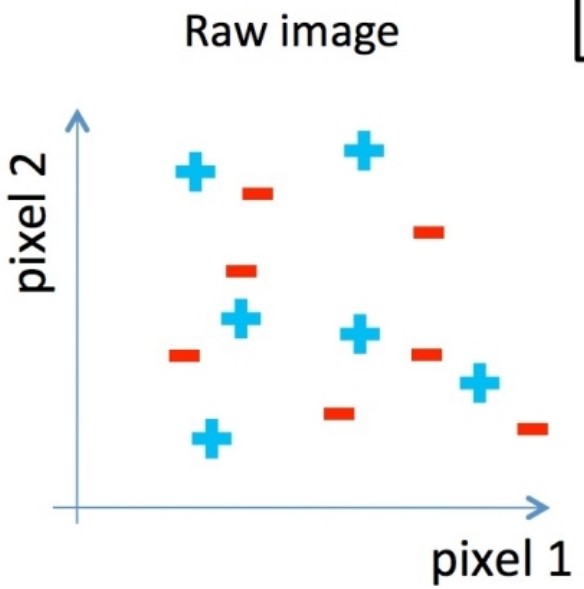
Raw image



Pixel-based representation







What we want

Input

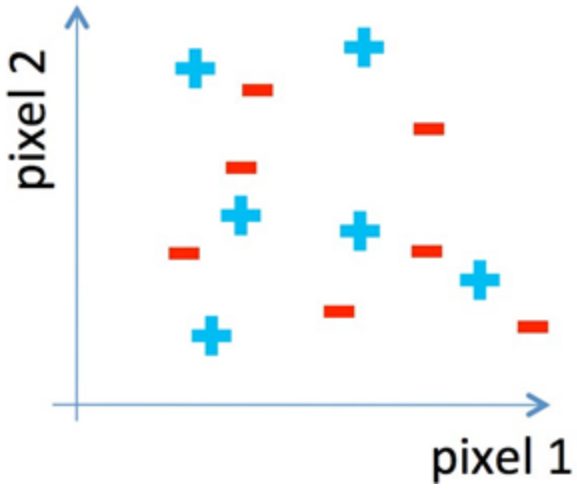
Feature representation

E.g., Does it have Handlebars? Wheels?

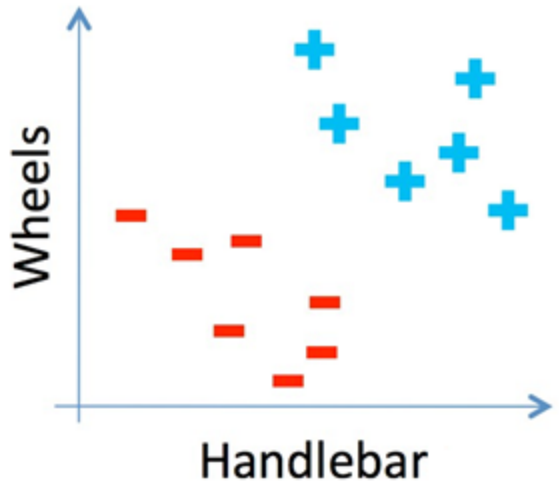
Learning algorithm

Motorbikes
"Non"-Motorbikes

Raw image



Features



What we want

Input

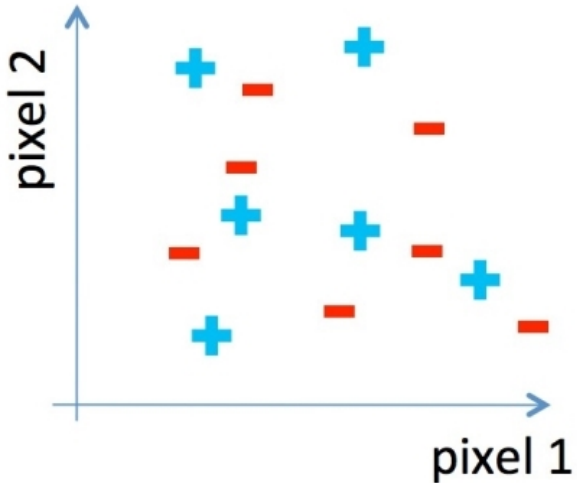
Feature representation

E.g., Does it have Handlebars? Wheels?

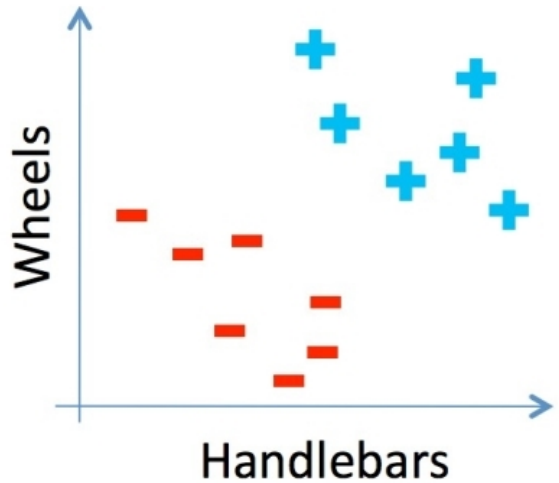
Learning algorithm

Motorbikes
"Non"-Motorbikes

Raw image



Features



Classical feature extraction algorithms:

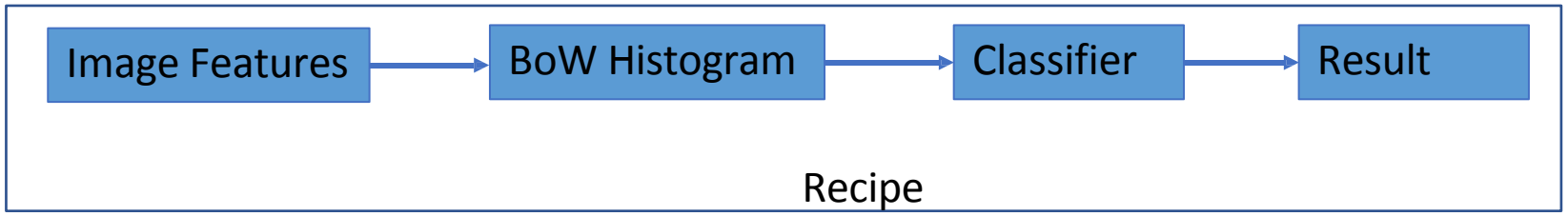
- HoG
- SIFT
- ……

There is a wide range of algorithms that do so for us

Coming up with features is often difficult, time-consuming, and requires expert knowledge.

“Bag-of-Words” (BoW) Histograms

Image



Today

Feature engineering -> Feature learning

Expert knowledge Data

Image classification - **ImageNet**

- Images for each category of WordNet
- 1000 classes
- 14M images
- 100K test

Dataset split

Training Images

- Train classifier

Validation Images

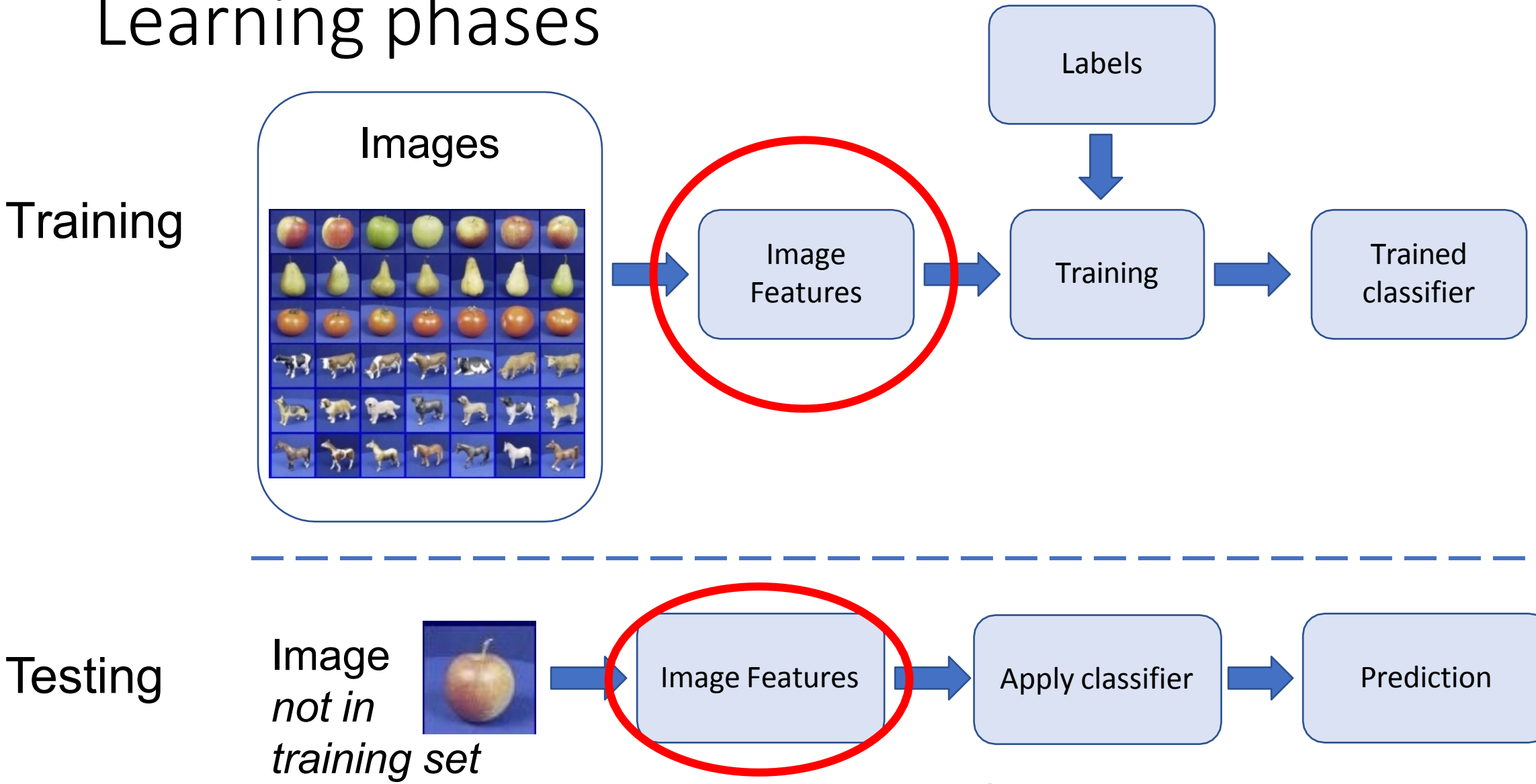
- Measure error
-Tune model hyperparameters

Testing Images

- Secret labels
- Measure error

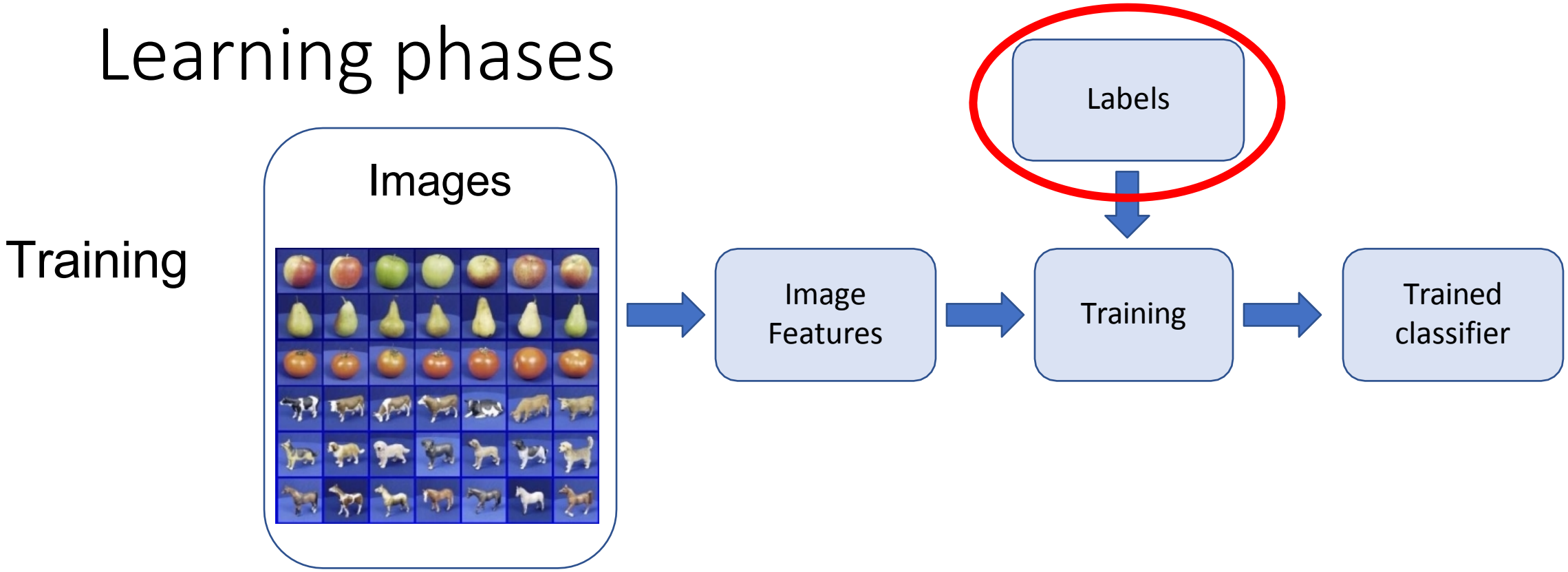
Random train/validate splits = cross validation

Learning phases



Slide credit: D. Hoiem and L. Lazebnik

Learning phases



Slide credit: D. Hoiem and L. Lazebnik

Recognition task and supervision

What are all the possible supervision ('label') *types* to consider?

Contains a motorbike

L. Lazebnik

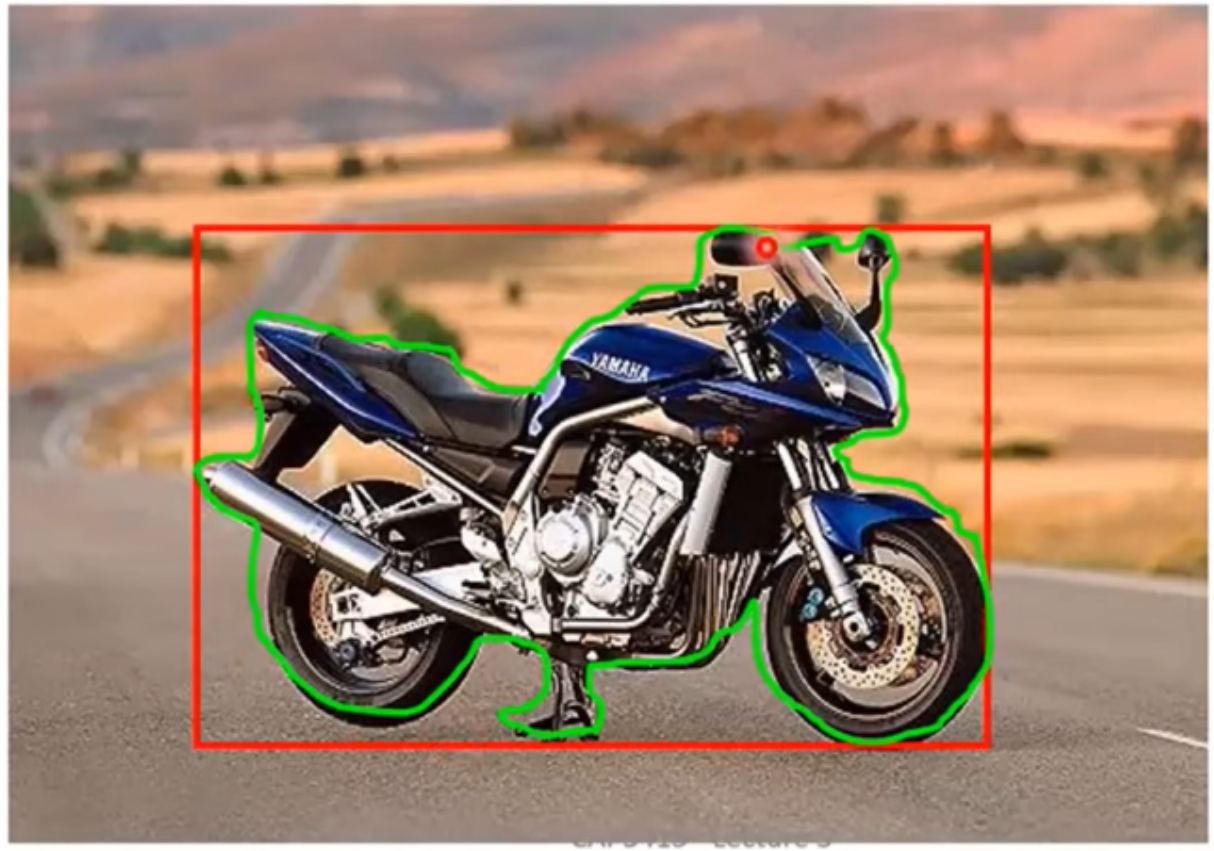
Recognition task and supervision

What are all the possible supervision ('label') *types* to consider?

Contains a motorbike

Recognition task and supervision

What are all the possible supervision ('label') *types* to consider?



Contains a motorbike

Spectrum of supervision

Less

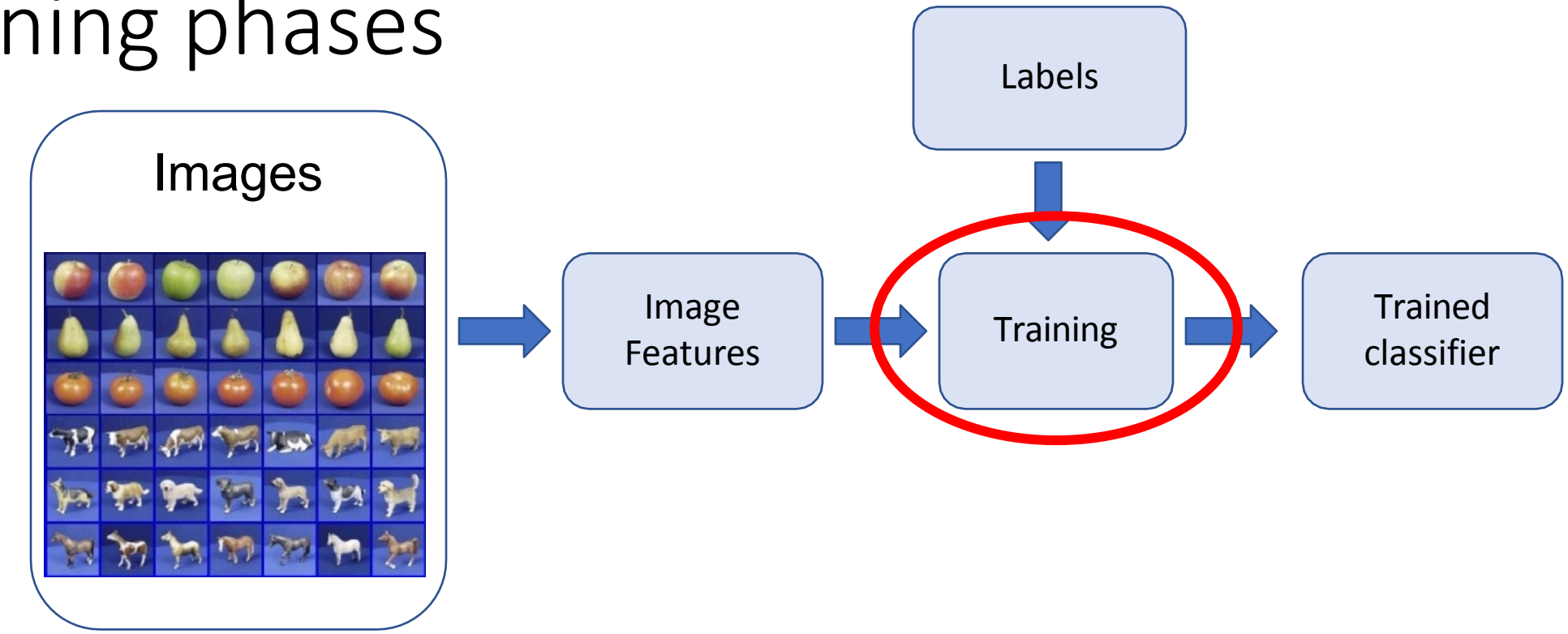
More

e.g., ImageNet

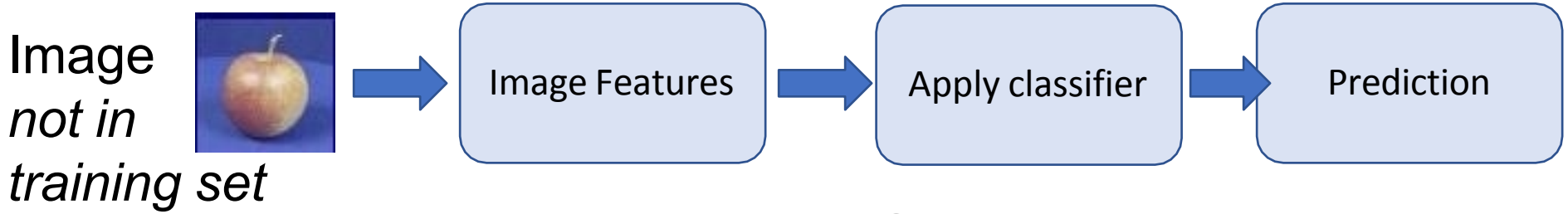
e.g., MS-COCO

Learning phases

Training



Testing



Slide credit: D. Hoiem and L. Lazebnik

The machine learning framework

- Apply a prediction function to a feature representation of the image to get the desired output:

f() = “apple”

f() = “tomato”

f() = “cow”

The machine learning framework

$$f(\mathbf{x}) = y$$

Prediction function or *classifier*

Image feature

Output (label)

Training: Given a *training set* of labeled examples:

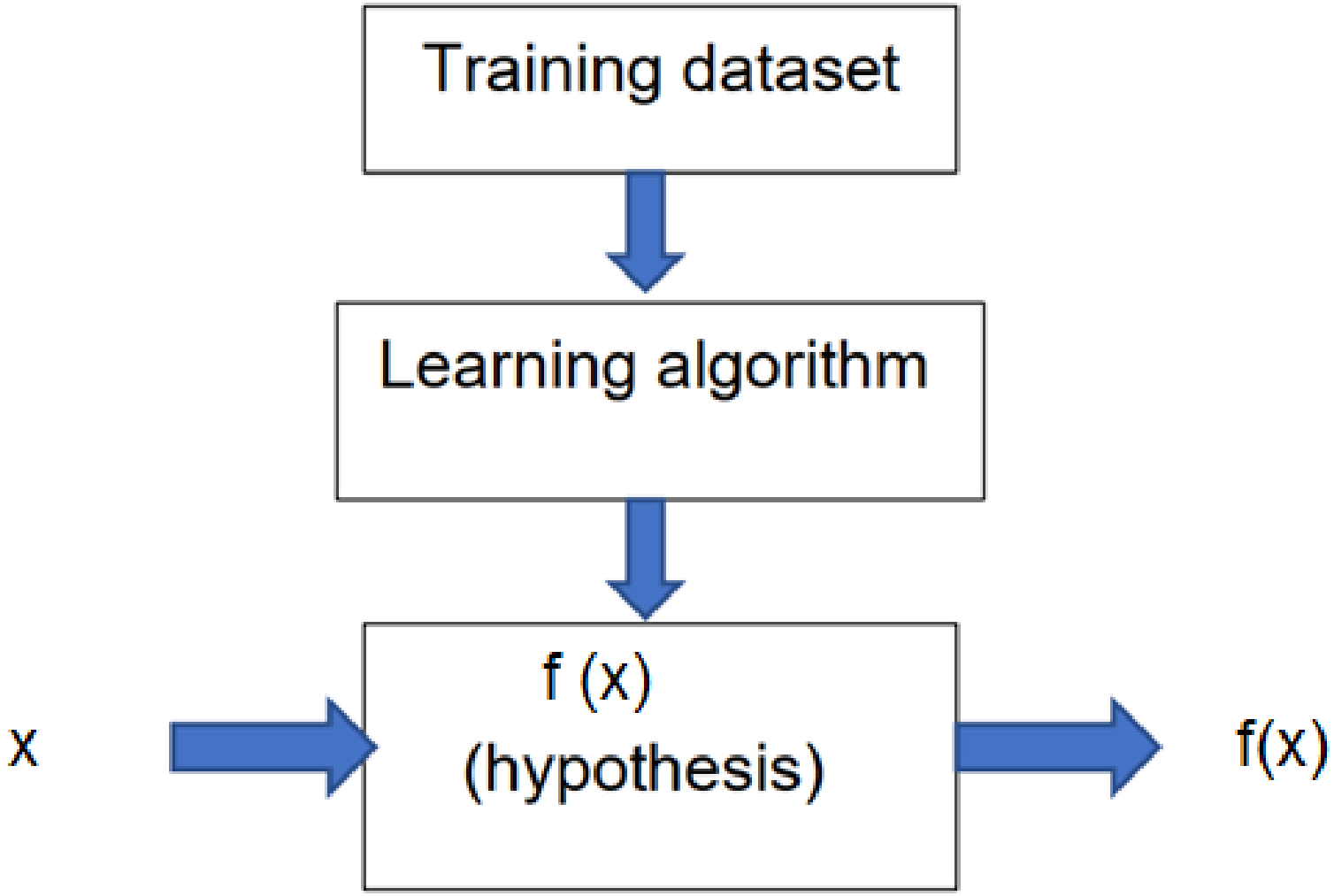
$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

Estimate the prediction function f by minimizing the prediction error on the training set.

Testing: Apply f to an unseen *test example* $\mathbf{x}_u$ and output the predicted value $y_u = f(\mathbf{x}_u)$ to *classify* $\mathbf{x}_u$.

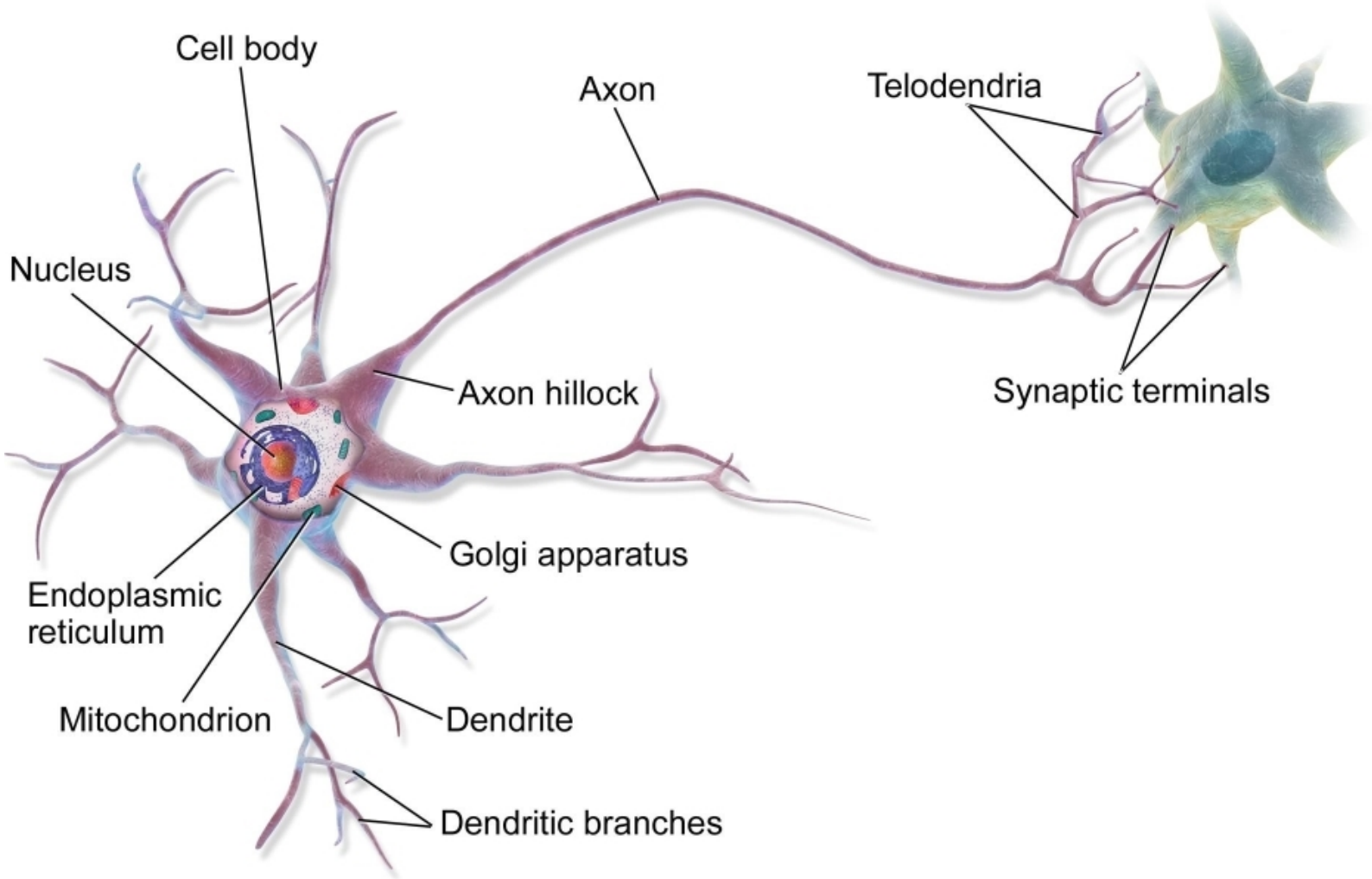
Slide credit: L. Lazebnik

how the supervising algorithm works:

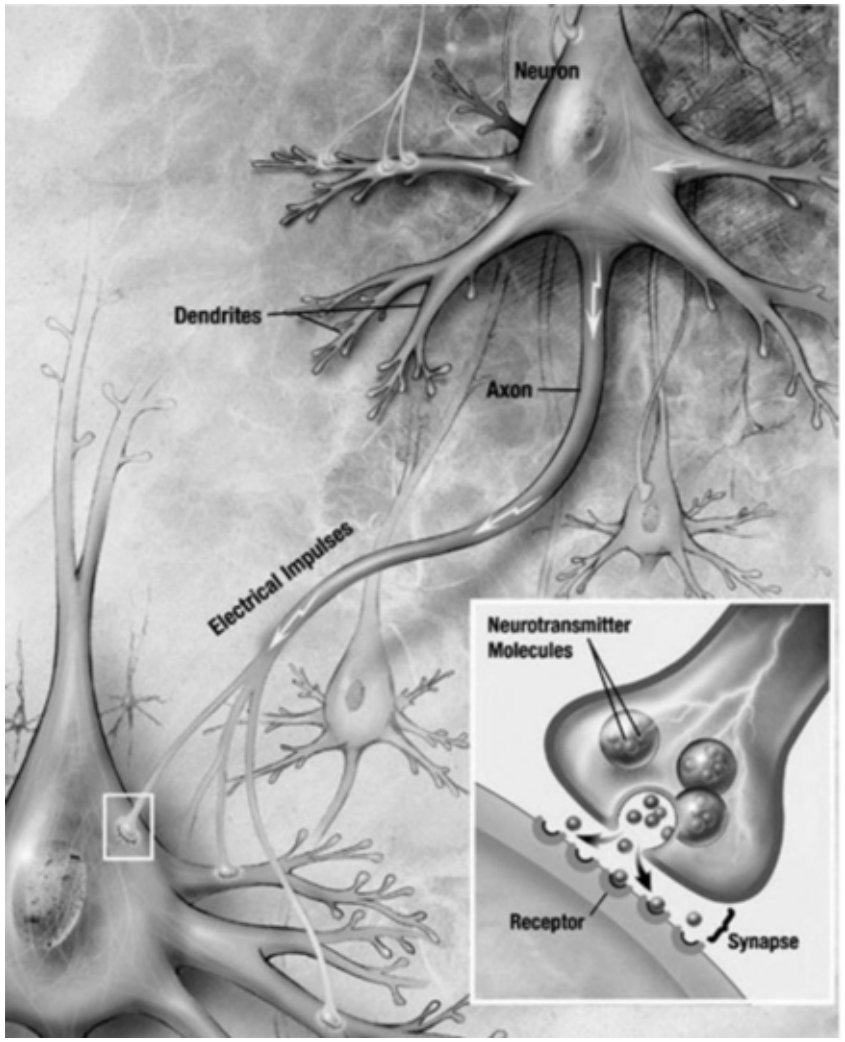


Neural network as a learning framework

Neurons in the Brain



Neurons in the Brain



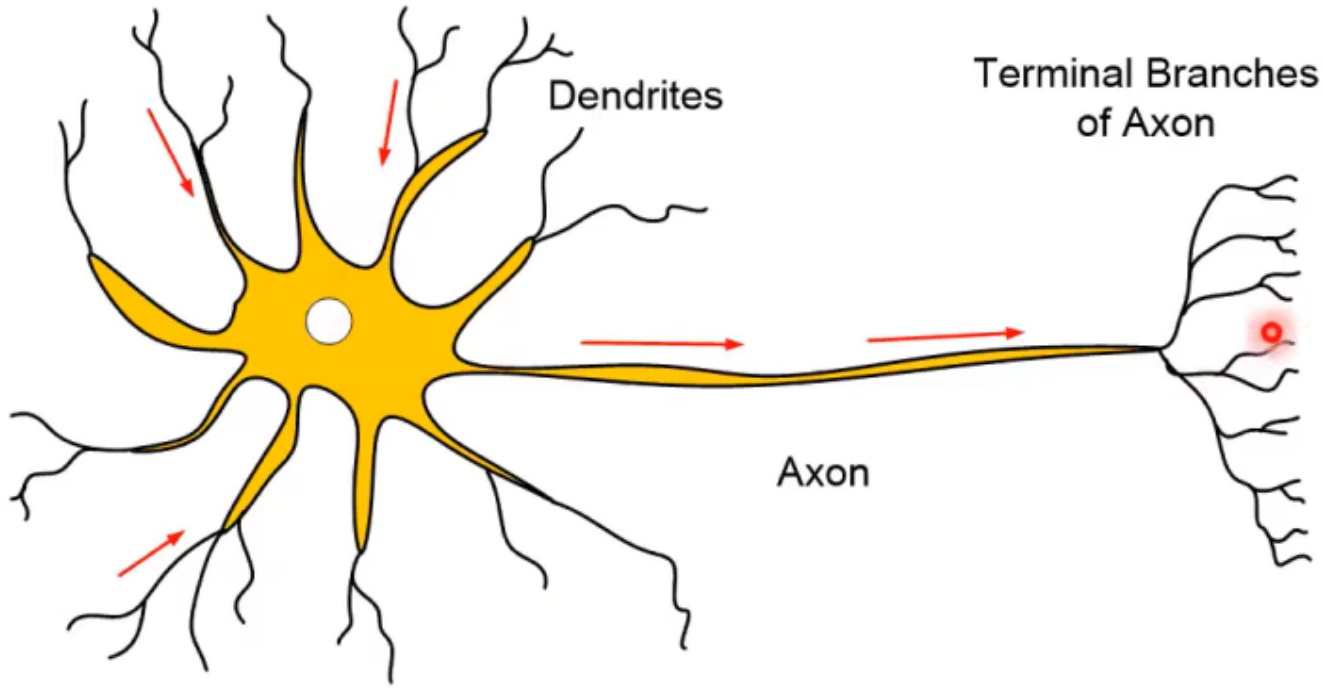
- Brain is composed of **neurons**
- A neuron receives input from other neurons (generally thousands)
- Inputs are approximately **Weighed summed**
- When the input exceeds a threshold the neuron sends an electrical spike that travels down the axon, to the next neuron(s)

Background in Neural Nets (NN)

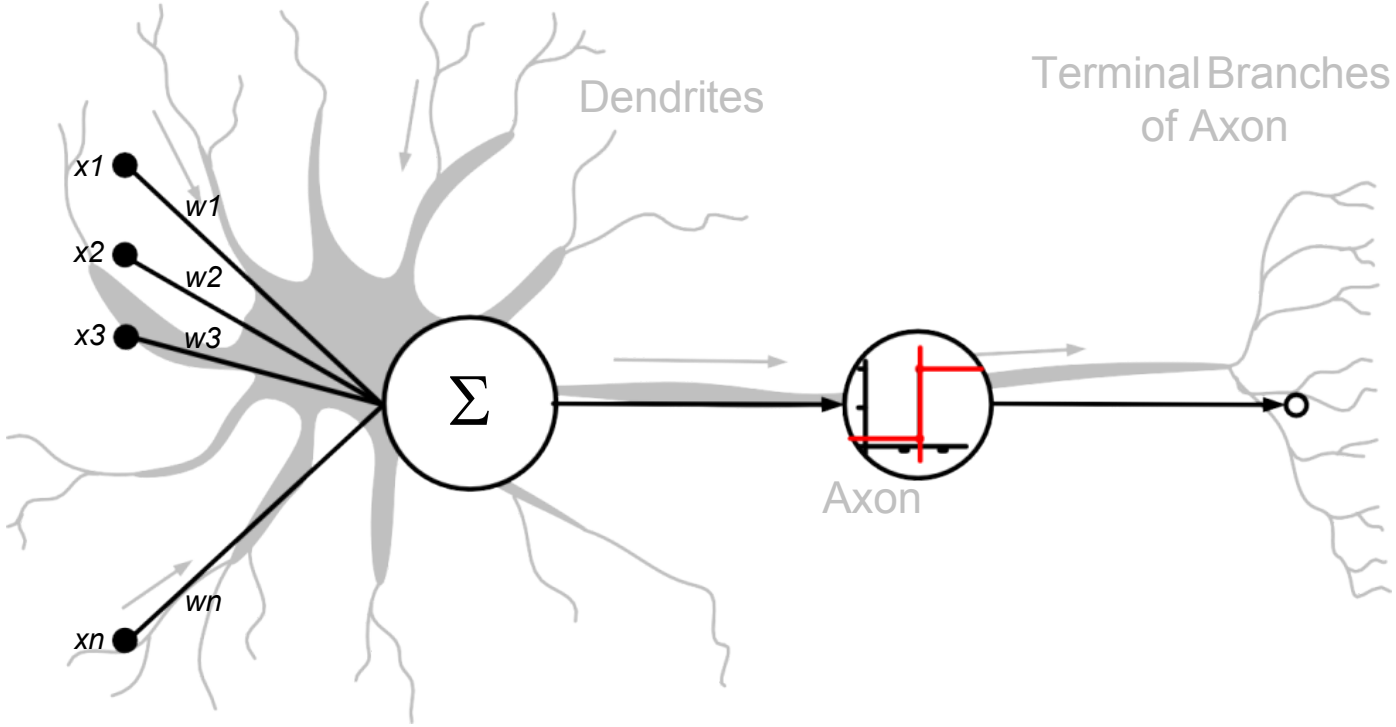
- Neural Nets can be:
  - **Biological** Models
  - **Artificial** Models

- Desire to produce artificial systems
  - capable of sophisticated computations
  - similar to human brain!

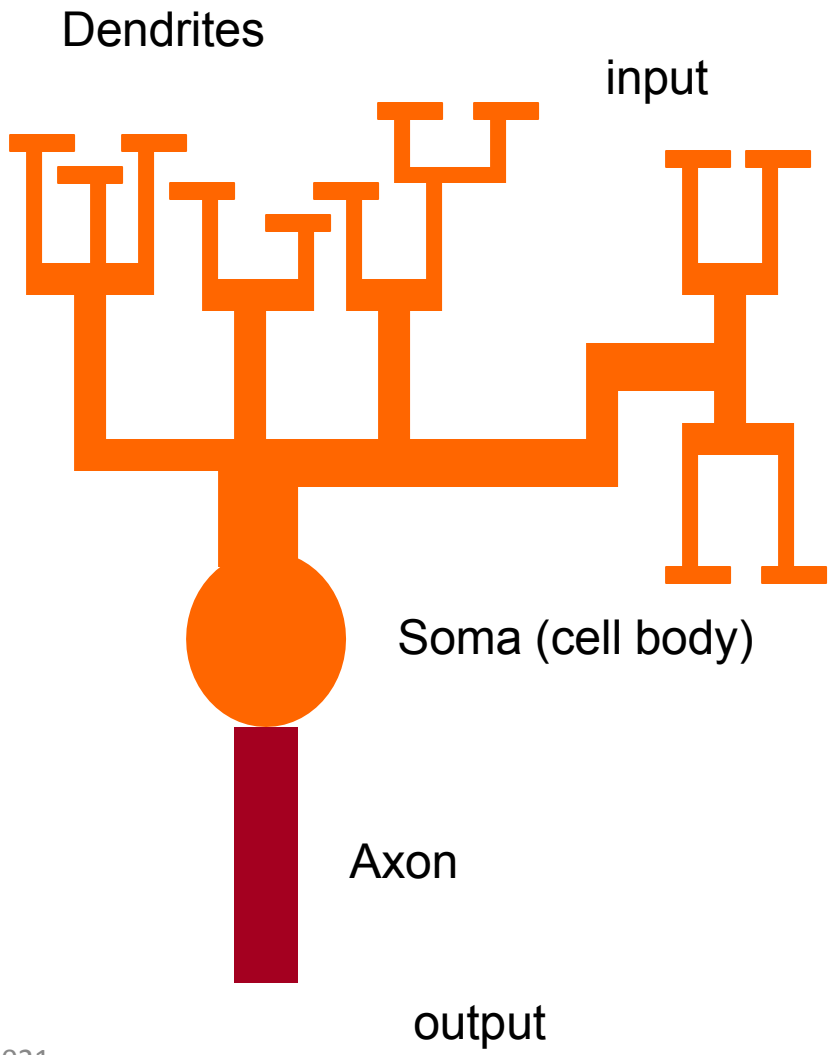
Computational Implementation of the Neural Activation Function

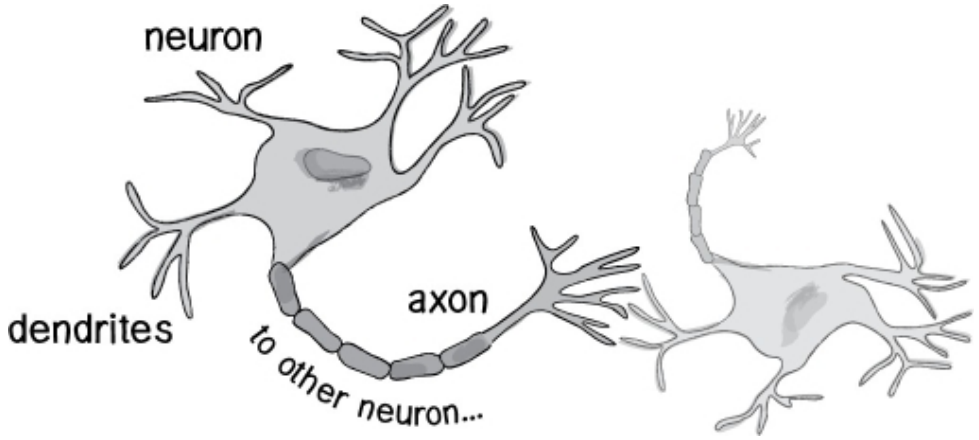


Computational Implementation of the Neural Activation Function

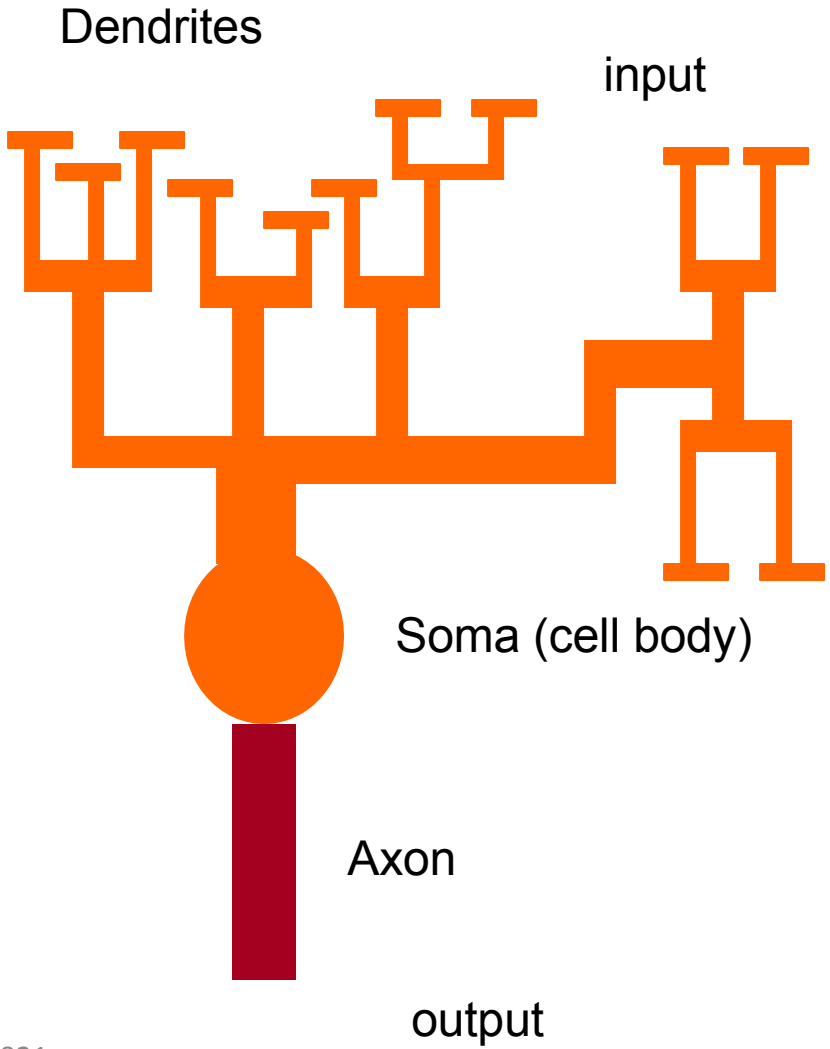


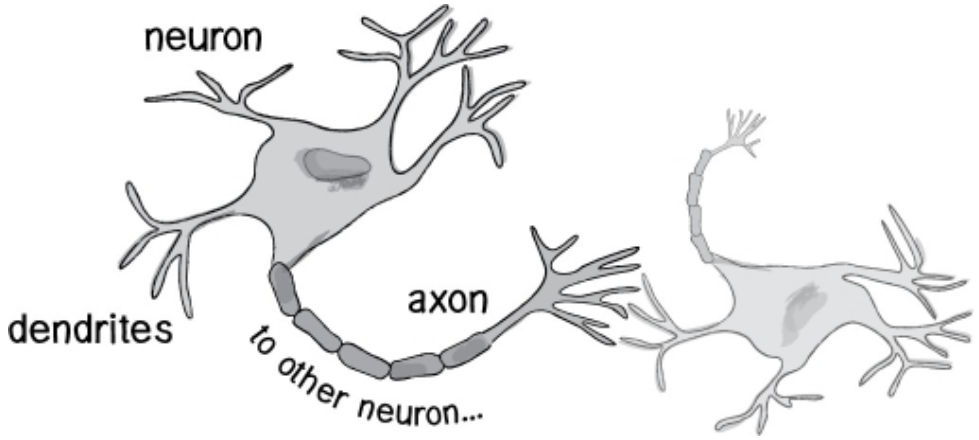
Neural Activation Function





Neural Activation Function

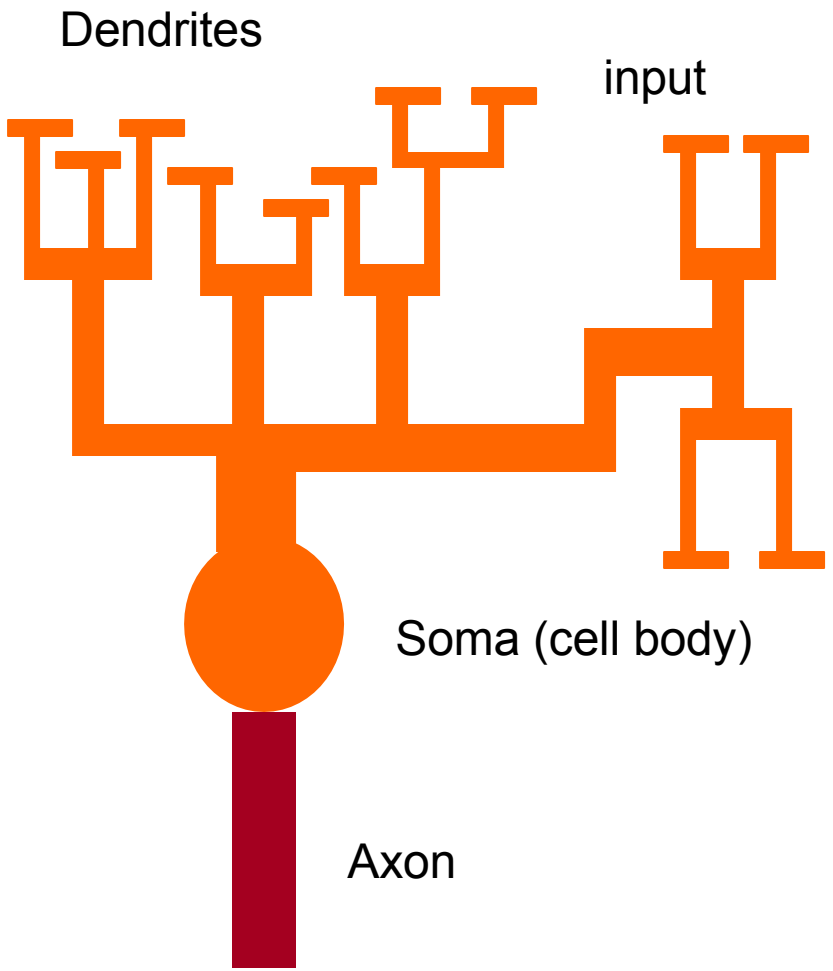


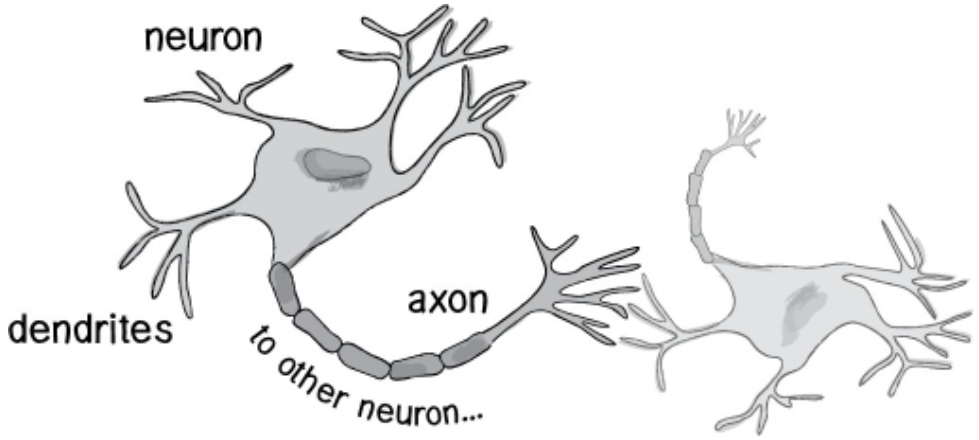


Activation value (sum of individual inputs):

$$n_j = \sum_i x_i w_{ij}$$

Neural Activation Function





Activation value (sum of individual inputs):

$$n_j = \sum_i x_i w_{ij}$$

Activation function (sigmoid / logistic):

$$y_j = \frac{1}{1 + e^{-n_j}}$$

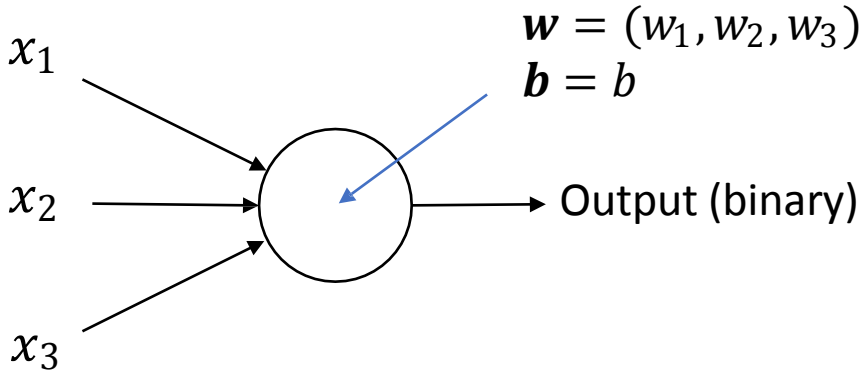
n

Neuron computation = logistic regress computation

Neural Networks

Basic building block for composition is a *perceptron* (Rosenblatt c.1960)

Linear classifier – vector of weights *w* and a 'bias' *b*



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases} \qquad w \cdot x \equiv \sum_j w_j x_j$$

Binary classifying an image

Each pixel of the image would be an input.

So, for a 28 x 28 image, we vectorize:

$$\mathbf{x} = 1 \times 784$$

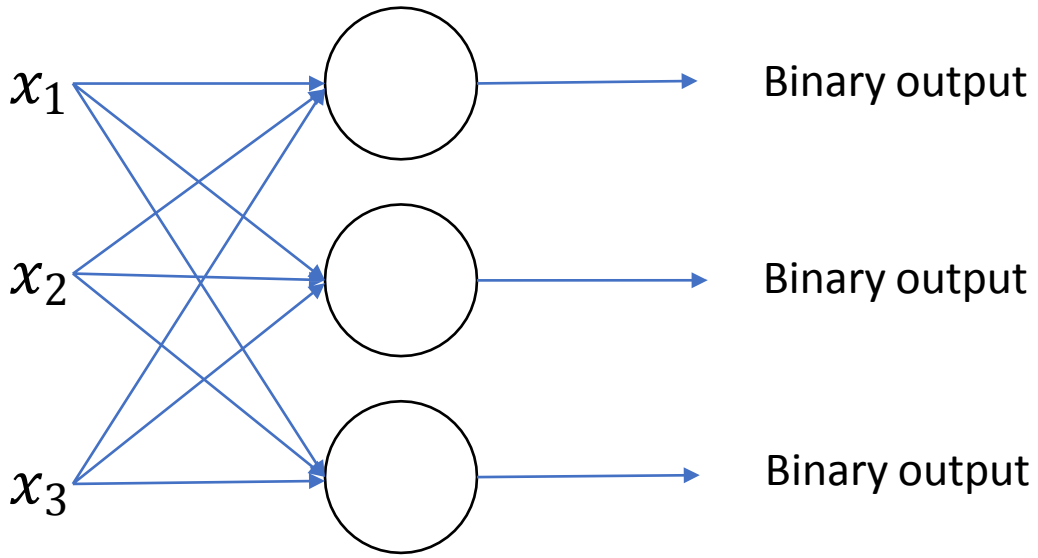
w is a vector of weights for each pixel, 784 x 1

b is a scalar bias per perceptron

Result = **xw** + b -> (1x784) x (784x1) + b = (1x1)+b

Neural Networks - multiclass

Add more perceptrons



Multi-class classifying an image

Each pixel of the image would be an input.

So, for a 28 x 28 image, we vectorize.

$$\mathbf{x} = 1 \times 784$$

W is a matrix of weights for each pixel/each perceptron

$$\mathbf{W} = 784 \times 10 \text{ (10-class classification)}$$

b is a bias *per perceptron* (vector of biases); (1 x 10)

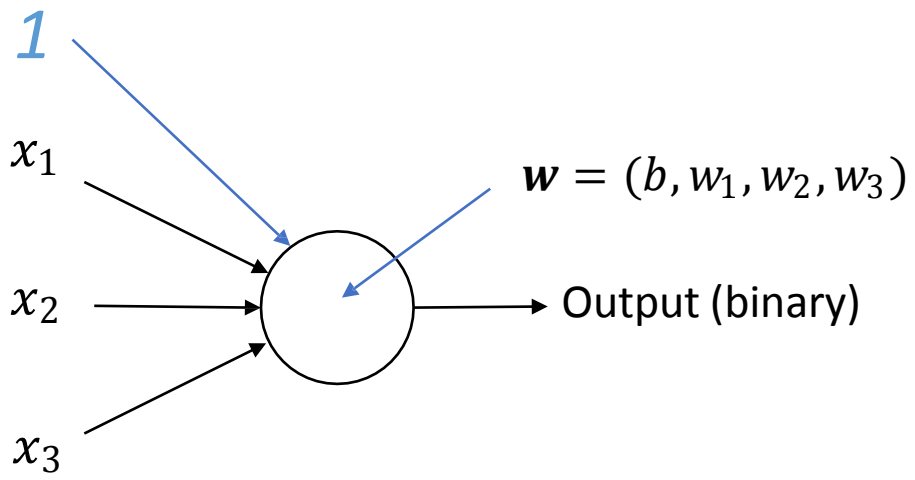
Result = $\mathbf{xW} + \mathbf{b}$ -> (1x784) x (784 x 10) + **b**

-> (1 x 10) + (1 x 10) = output vector

Bias convenience

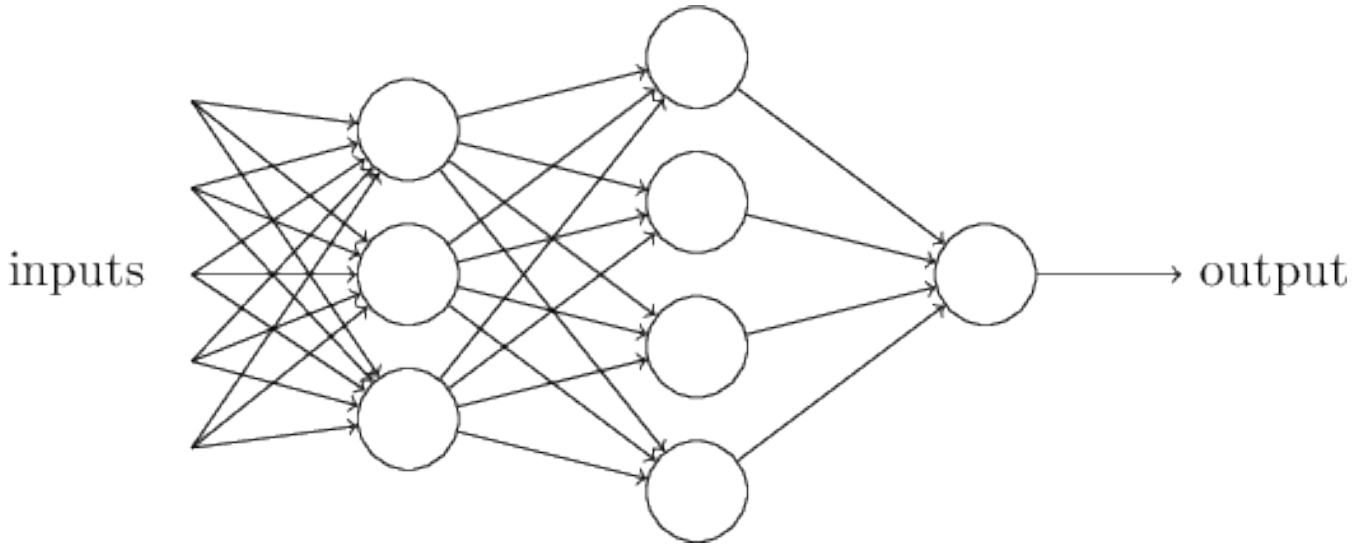
Let's turn this operation into a multiplication only:

- Create a 'fake' feature with value 1 to represent the bias
- Add an extra weight that can vary



$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x \leq 0 \\ 1 & \text{if } w \cdot x > 0 \end{cases} \qquad w \cdot x \equiv \sum_j w_j x_j$$

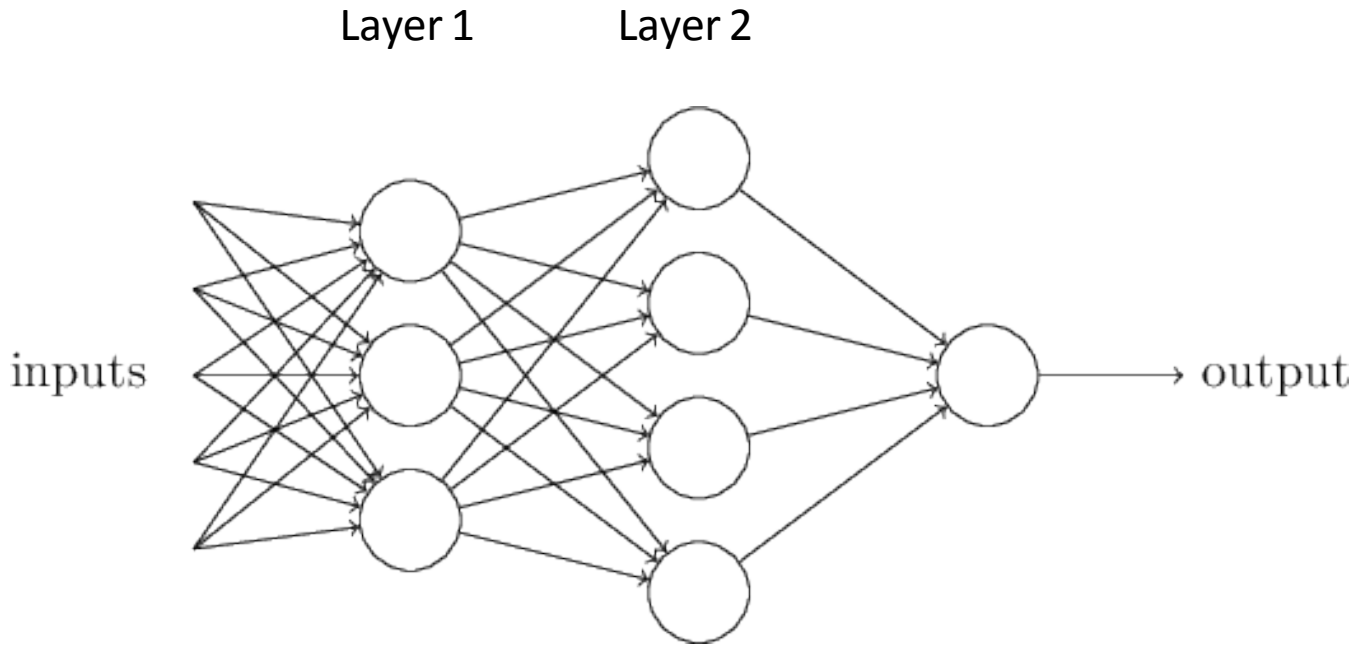
Composition



Attempt to represent complex functions as compositions of smaller functions.
Outputs from one perception are fed into inputs of another perceptron.

Nielsen

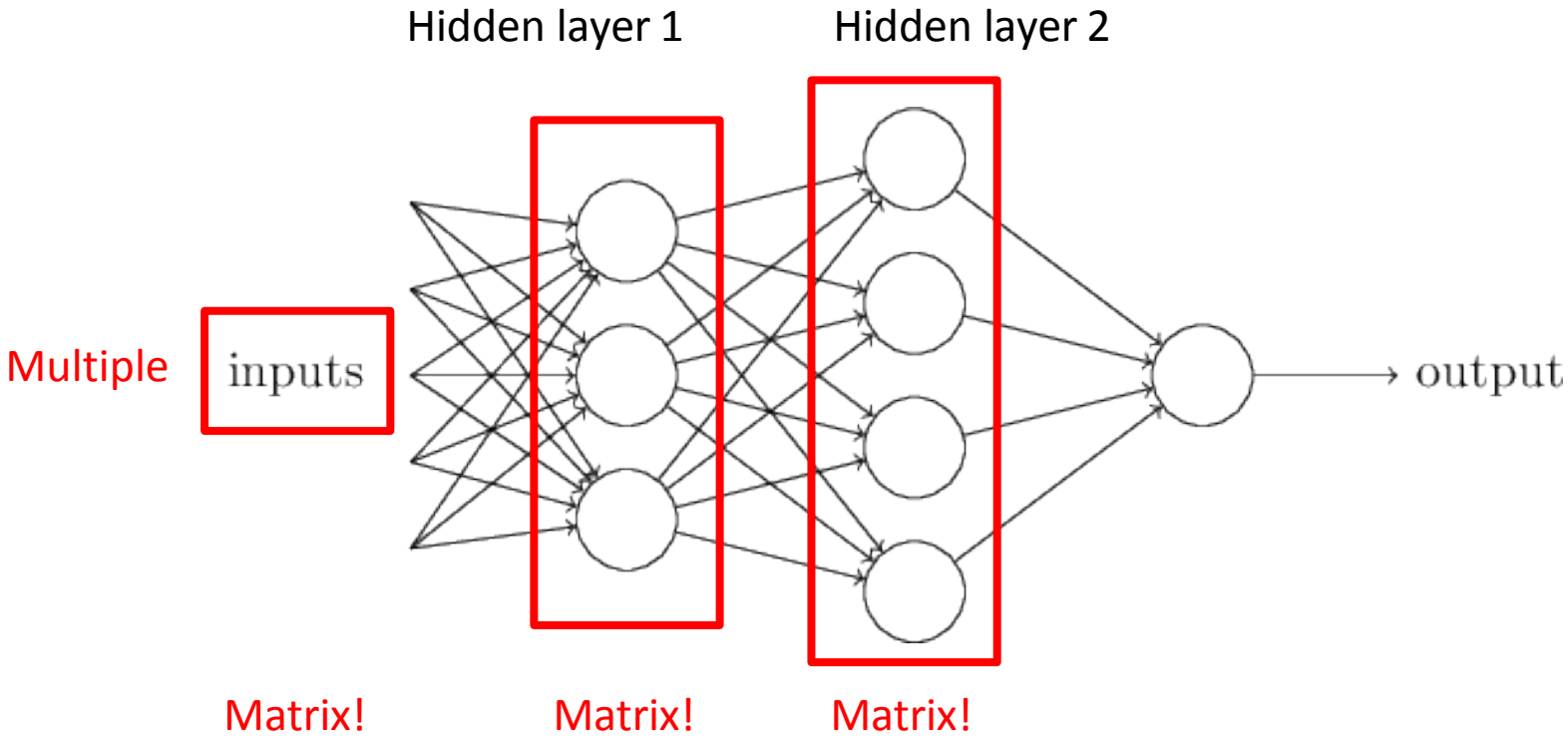
Composition



Sets of layers and the connections (weights) between them define the *network architecture.*

Nielsen

Composition



It's all just matrix multiplication!

GPUs -> special hardware for fast/large matrix multiplication.

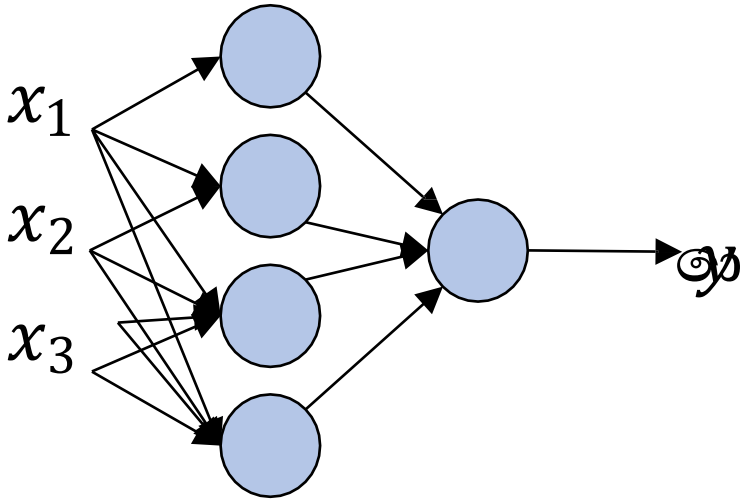
Nielsen

Problem 1 with all linear functions

- We have formed chains of linear functions.
- We know that linear functions can be reduced
  - g = f(h(x))

Our composition of functions is really just a single function : (

Problem 1 with all linear functions



$$\mathbf{z}^{[1]} = \mathbf{W}^{[1]}\,\mathbf{x} + \mathbf{b}^{[1]}$$

$$\mathbf{z}^{[2]} = \mathbf{W}^{[2]}\,\mathbf{z}^{[1]} + \mathbf{b}^{[2]}$$

$$\begin{aligned}\mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\,\mathbf{z}^{[1]} + \mathbf{b}^{[2]} \\ &= \mathbf{W}^{[2]}\,[\mathbf{W}^{[1]}\,\mathbf{x} + \mathbf{b}^{[1]}] + \mathbf{b}^{[2]} \\ &= \mathbf{W}^{[2]}\,\mathbf{W}^{[1]}\,\mathbf{x} + \mathbf{W}^{[2]}\,\mathbf{b}^{[1]} + \mathbf{b}^{[2]} \\ &= \mathbf{W}\,\mathbf{x} + \mathbf{b}\end{aligned}$$

$$y = \mathbf{z}^{[2]} = \mathbf{W}\,\mathbf{x} + \mathbf{b}$$

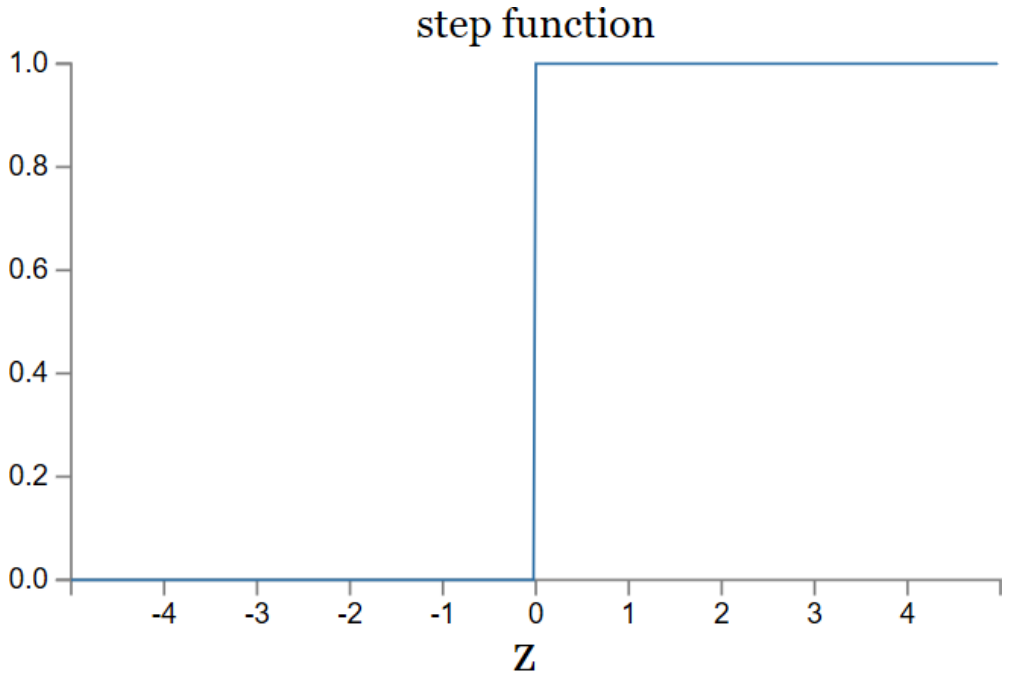
The output is always a linear function of the input!

Problem 2 with all linear functions

Linear classifiers:

small change in input can cause large change in binary output

Activation function for a perceptron: Heaviside function

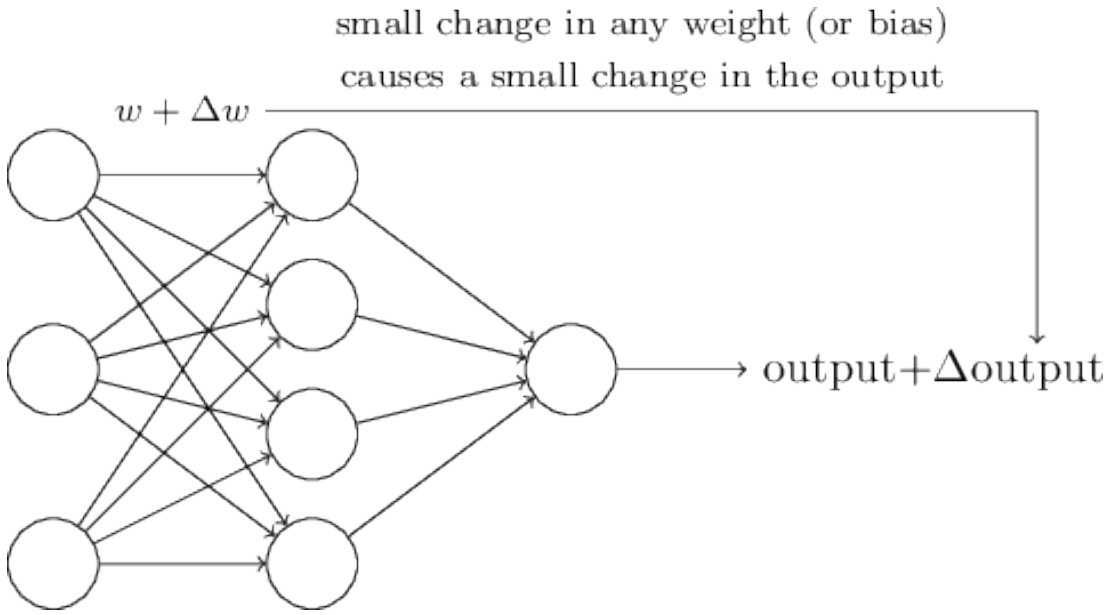


Problem 2 with all linear functions

Linear classifiers:

small change in input can cause large change in binary output.

We want:



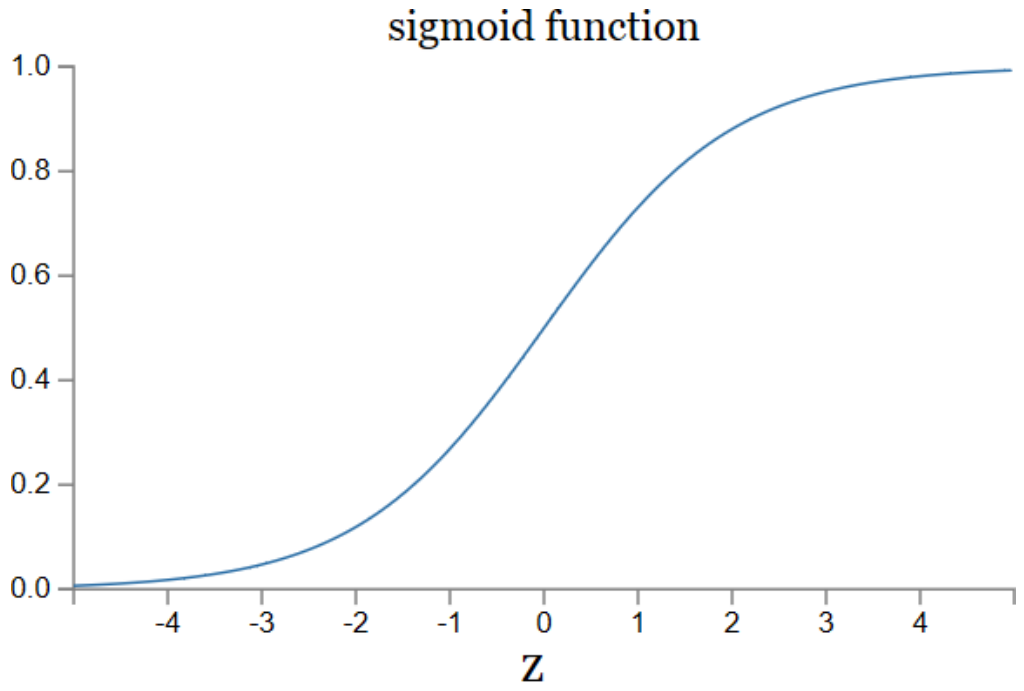
Nielsen

Let's introduce non-linearities

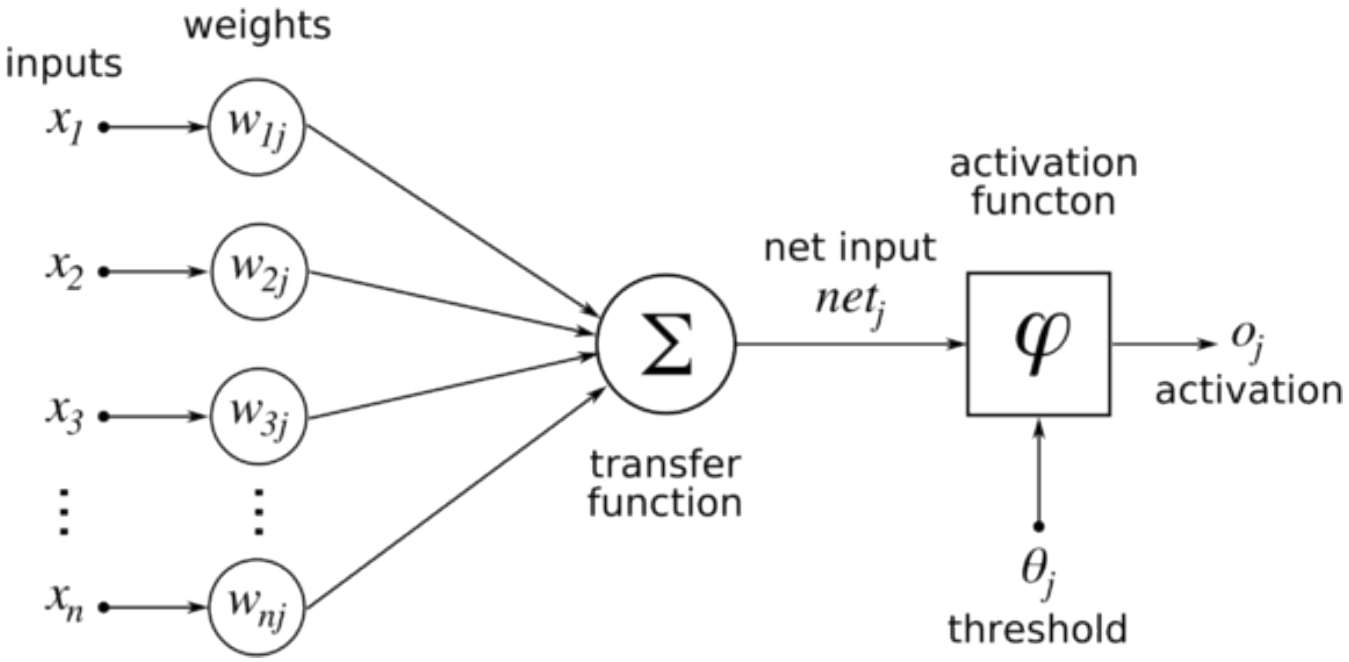
We're going to introduce non-linear functions to transform the features.

$$\sigma(w \cdot x + b)$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}.$$

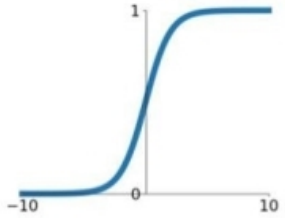


Activation Functions

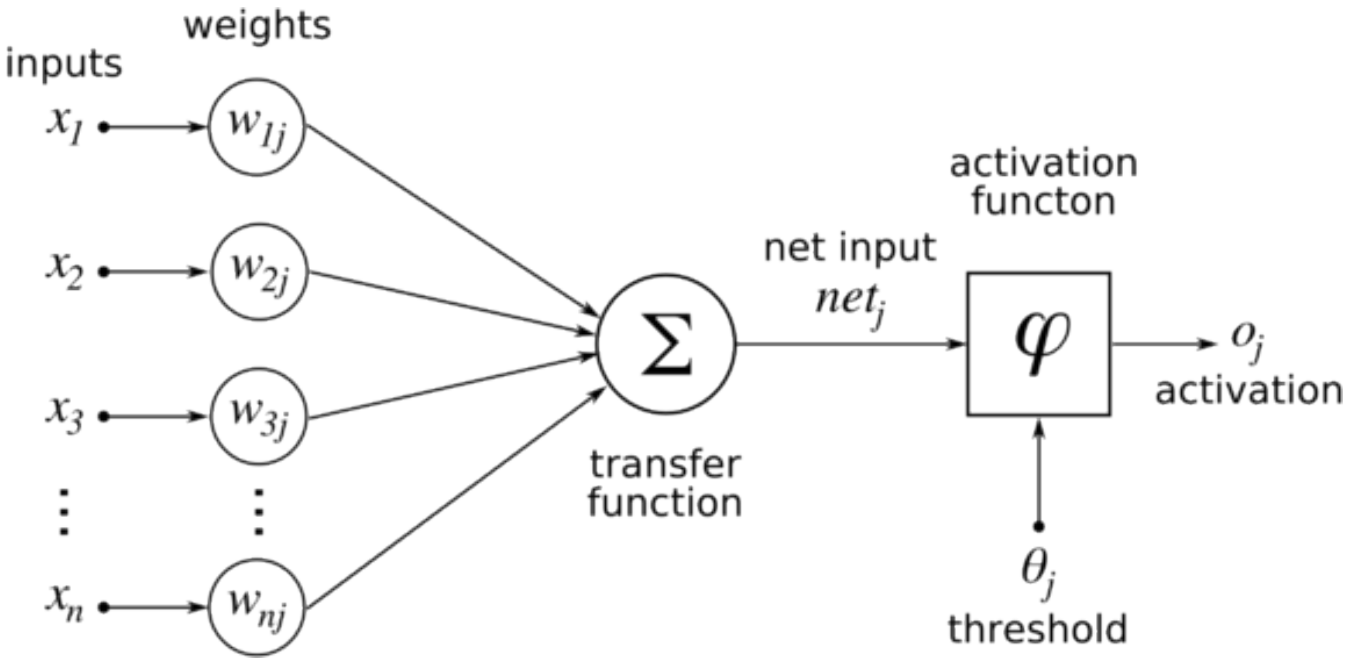


Sigmoid

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

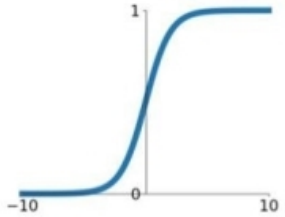


Activation Functions



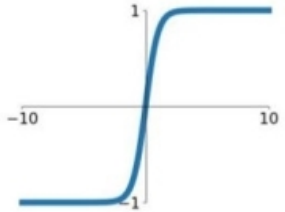
Sigmoid

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

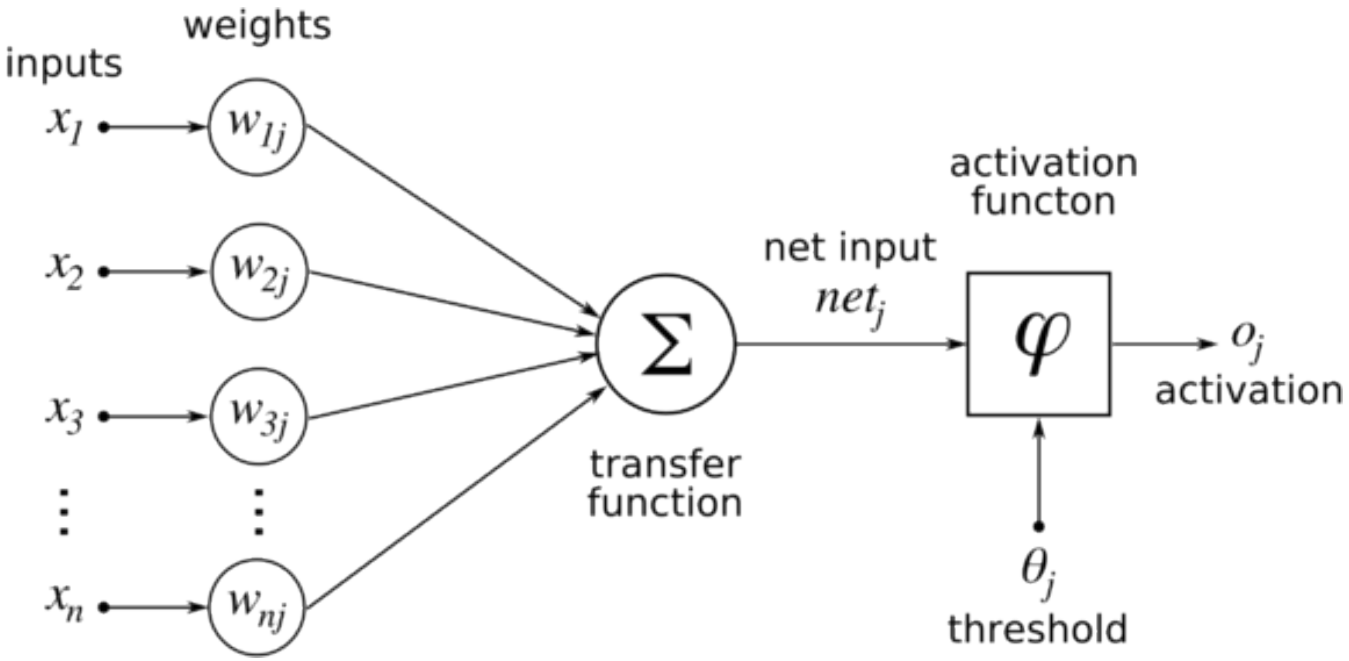


tanh

$$\tanh(x)$$

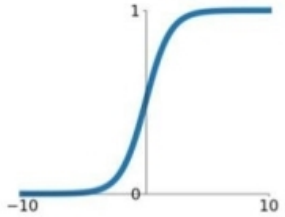


Activation Functions



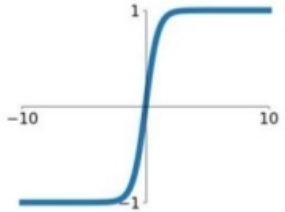
Sigmoid

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



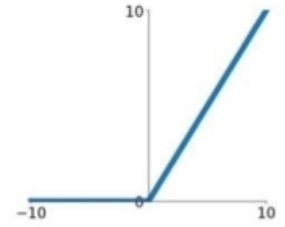
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$

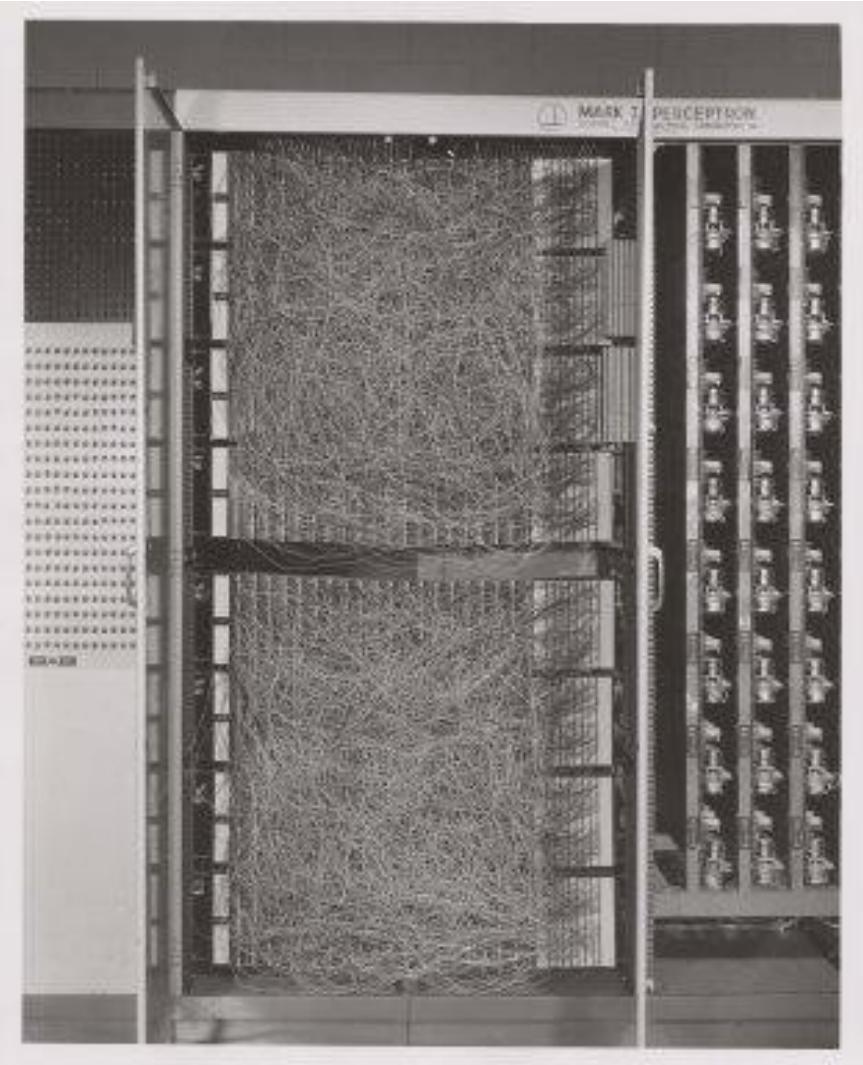


Perceptron model

- Use is grounded in theory
  - Universal approximation theorem (Goodfellow 6.4.1)

- Can represent a NAND circuit, from which any binary function can be built by compositions of NANDs

- With enough parameters, it can approximate any function.



Mark 1 Perceptron
c.1960

20x20 pixel
camera feed

Wikipedia

Perceptron model

If a single-layer network can learn any function…
…given enough parameters…

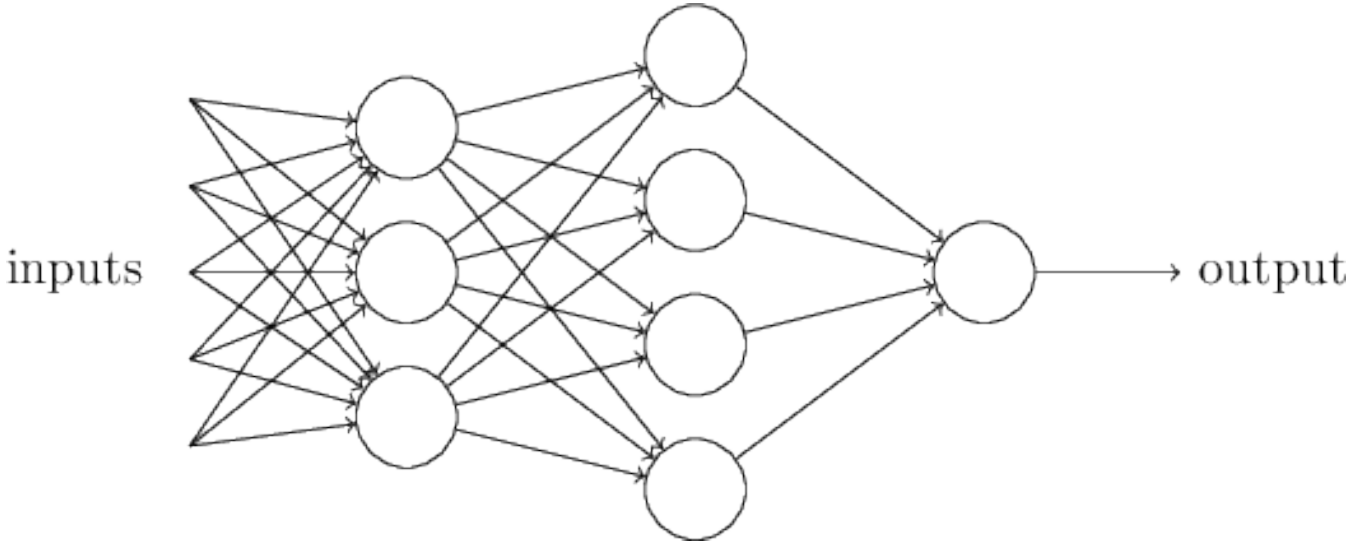
…then why do we go deeper?

Intuitively, composition is efficient because it allows *reuse*.

Empirically, deep networks do a better job than shallow networks at learning such hierarchies of knowledge.

Multi-layer perceptron (MLP)

- …is a '*fully connected*' neural network with non-linear activation functions.



- '*Feed-forward*' neural network

Goals

- Build a classifier which is more powerful at representing complex functions *and* more suited to the learning problem.

- What does this mean?

  1. Assume that the *underlying data generating function* relies on a composition of factors.
  2. Learn a feature representation that is specific to the dataset.

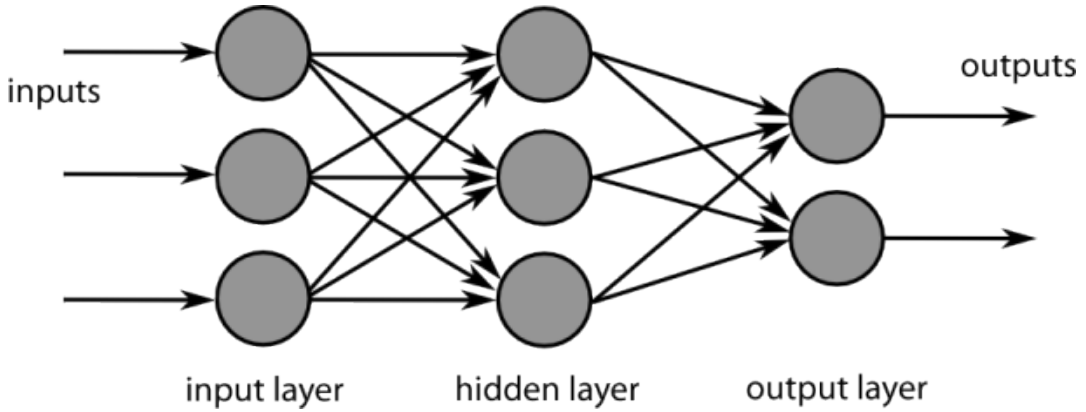
MLP performance on MNIST

- MNIST
  - Dataset of handwritten digits
  - 60K training samples
  - 10K testing samples
- A 6-layer MLP [1]
- Number of neurons
  - 2500, 2000, 1500, 1000, 500, and 10
- Performance - 99.65%

[1] Cireşan, Dan Claudiu, et al. "Deep, big, simple neural nets for handwritten digit recognition." *Neural computation* 22.12 (2010): 3207-3220.

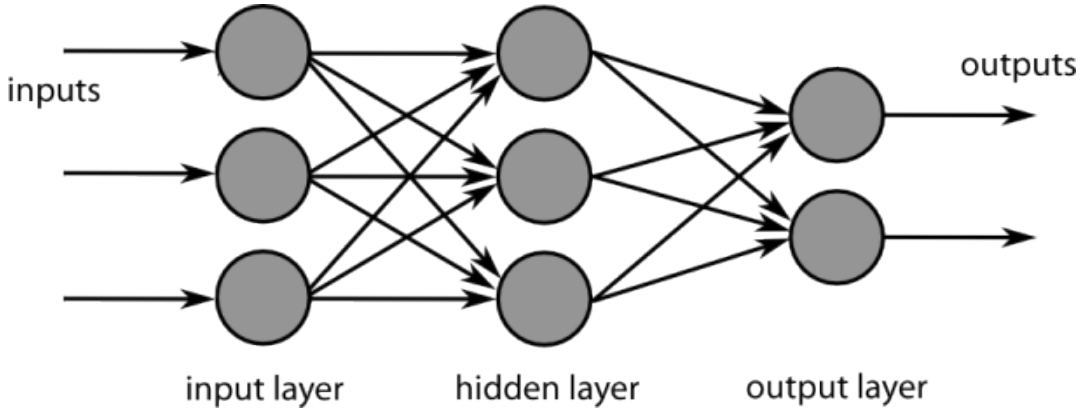
Why we need CNN?

- Neural Network
  - Fully connected layers
- Input
  - Hand-crafted features
  - Pixel values
- Hand-crafted features
  - Limitations
- Image as input
  - Size of feature vector = HxWxC
  - For 256x256 RGB image
    - 196608 dimensions

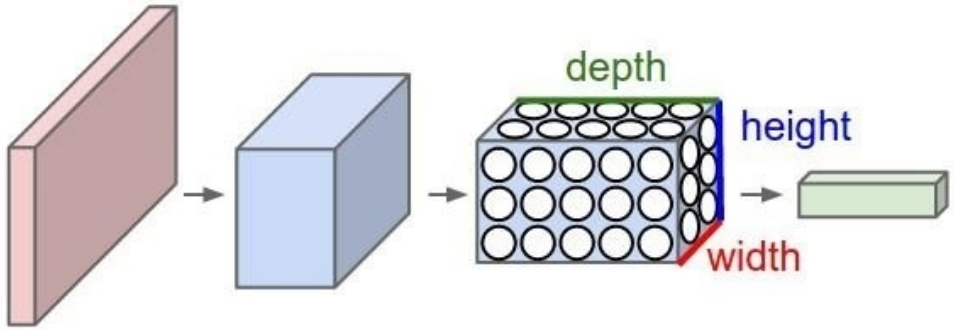


Why we need CNN?

- Neural Network
  - Fully connected layers
- Input
  - Hand-crafted features
  - Pixel values



- CNN - Special type of neural network
  - Operate with volume of data
  - Weight sharing in form of kernels



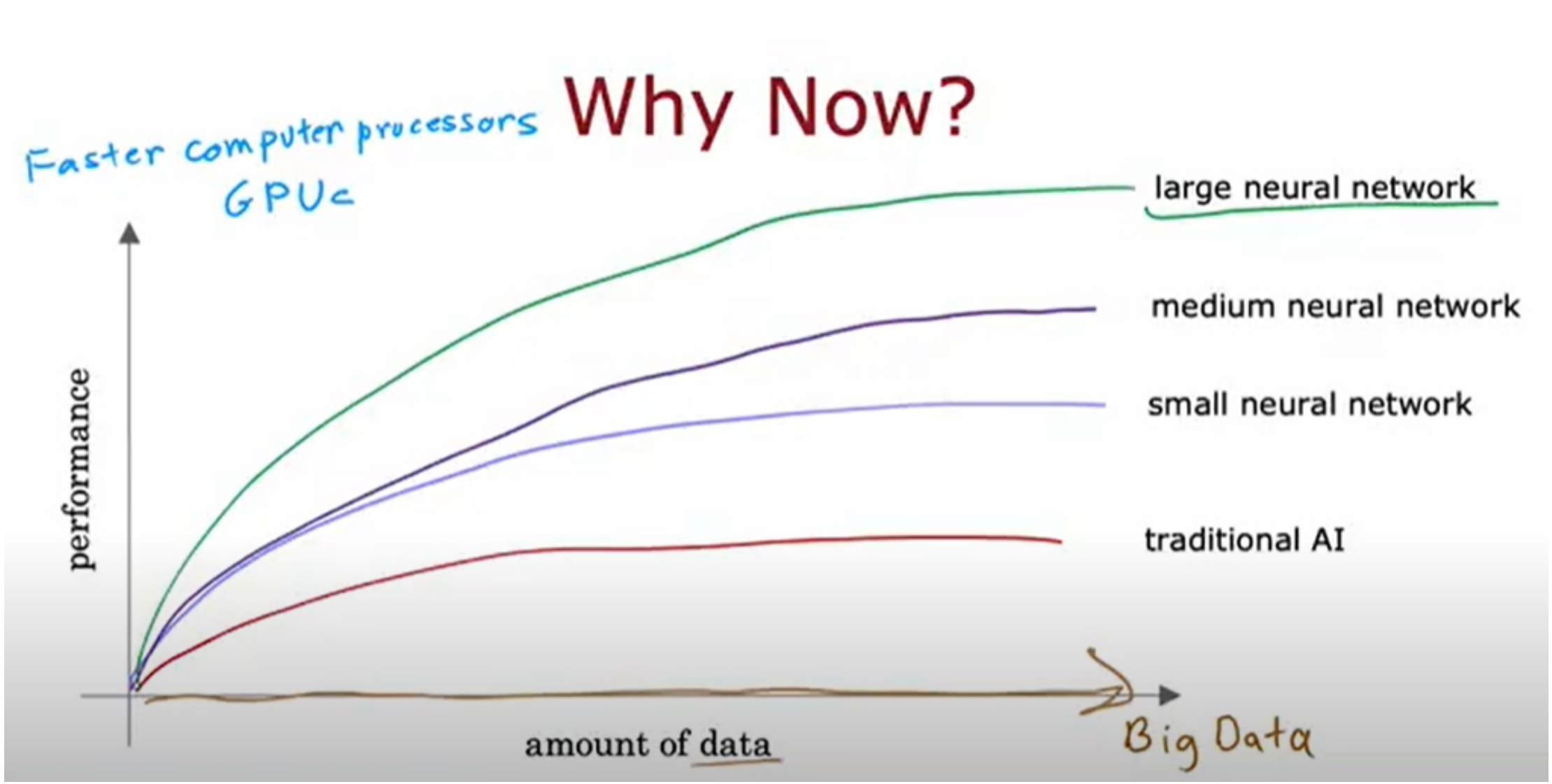
Source: http://cs231n.github.io

Questions?

Sources for this lecture include materials from works by Abhijit Mahalanobis, James Tompkin, Sedat Ozer, and Ulas Bagci

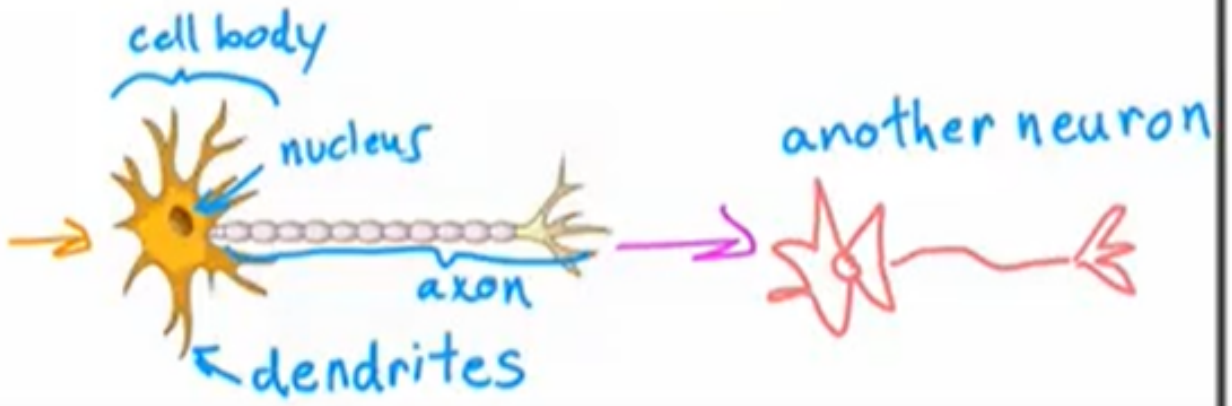

UCF
Center for Research
in Computer Vision

Why Now?


Faster computer processors
GPUs
large neural network
medium neural network
small neural network
traditional AI
performance
amount of data
Big Data

Biological neuron

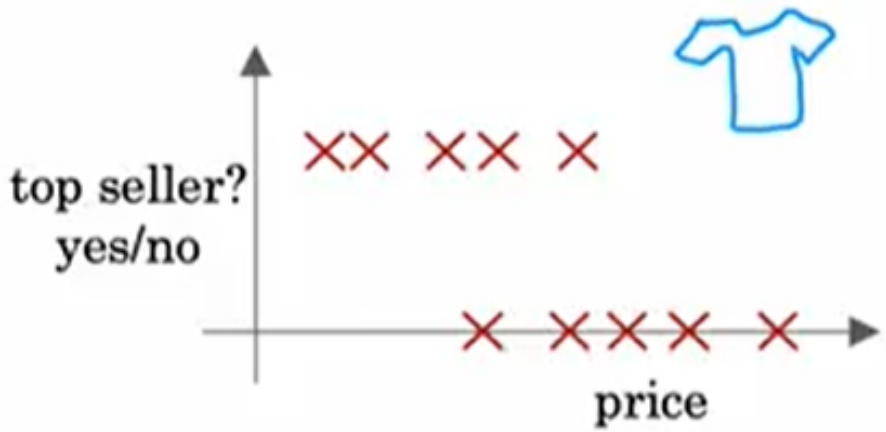
inputs outputs



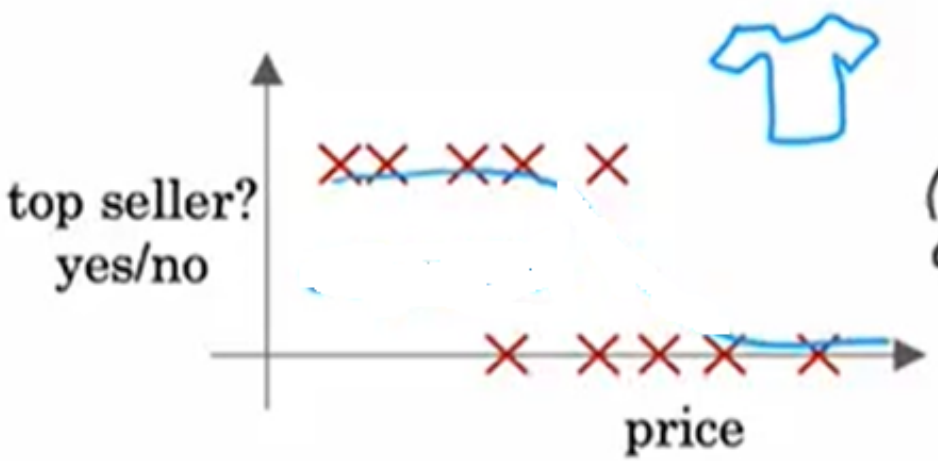
Simplified mathematical model of a neuron

inputs outputs

Demand Prediction



Demand Prediction



$x = price$ input

$a = f(x) = \frac{1}{1 + e^{-(wx+b)}}$ output

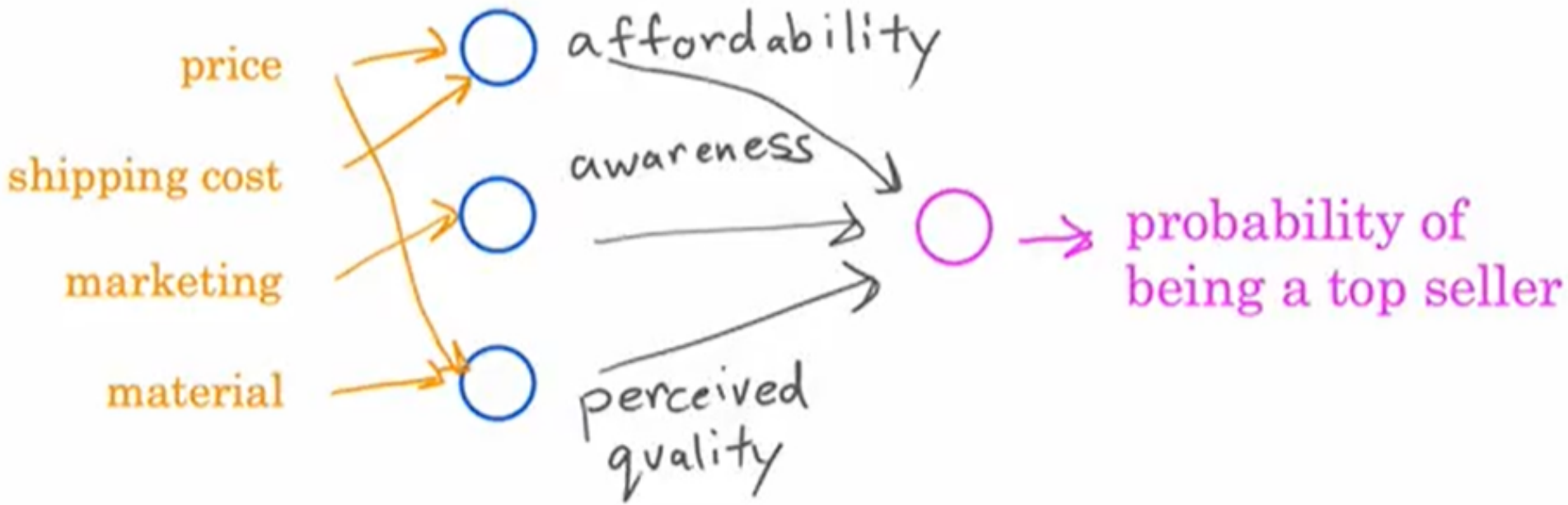
activation

x → price

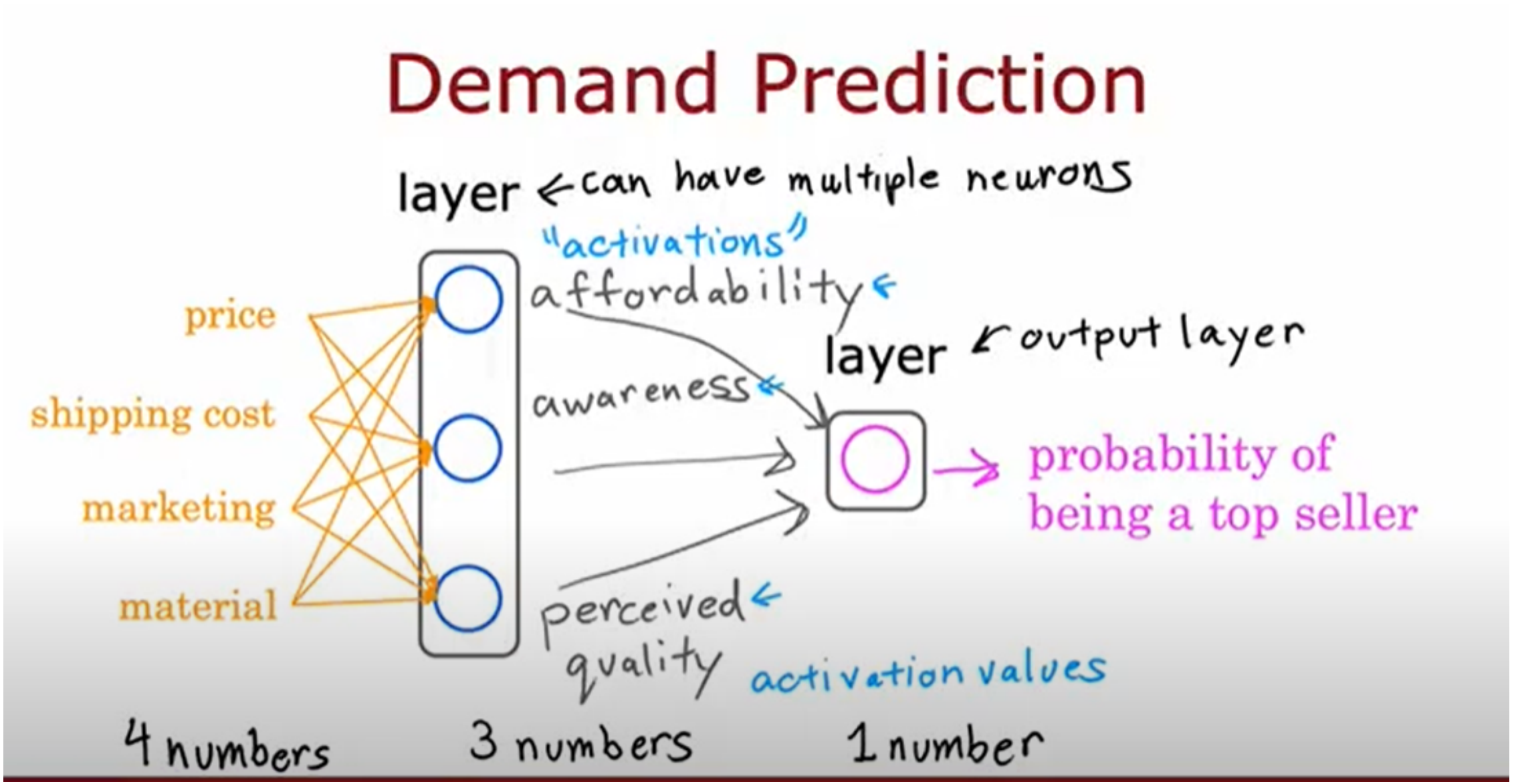
a → probability of being top seller

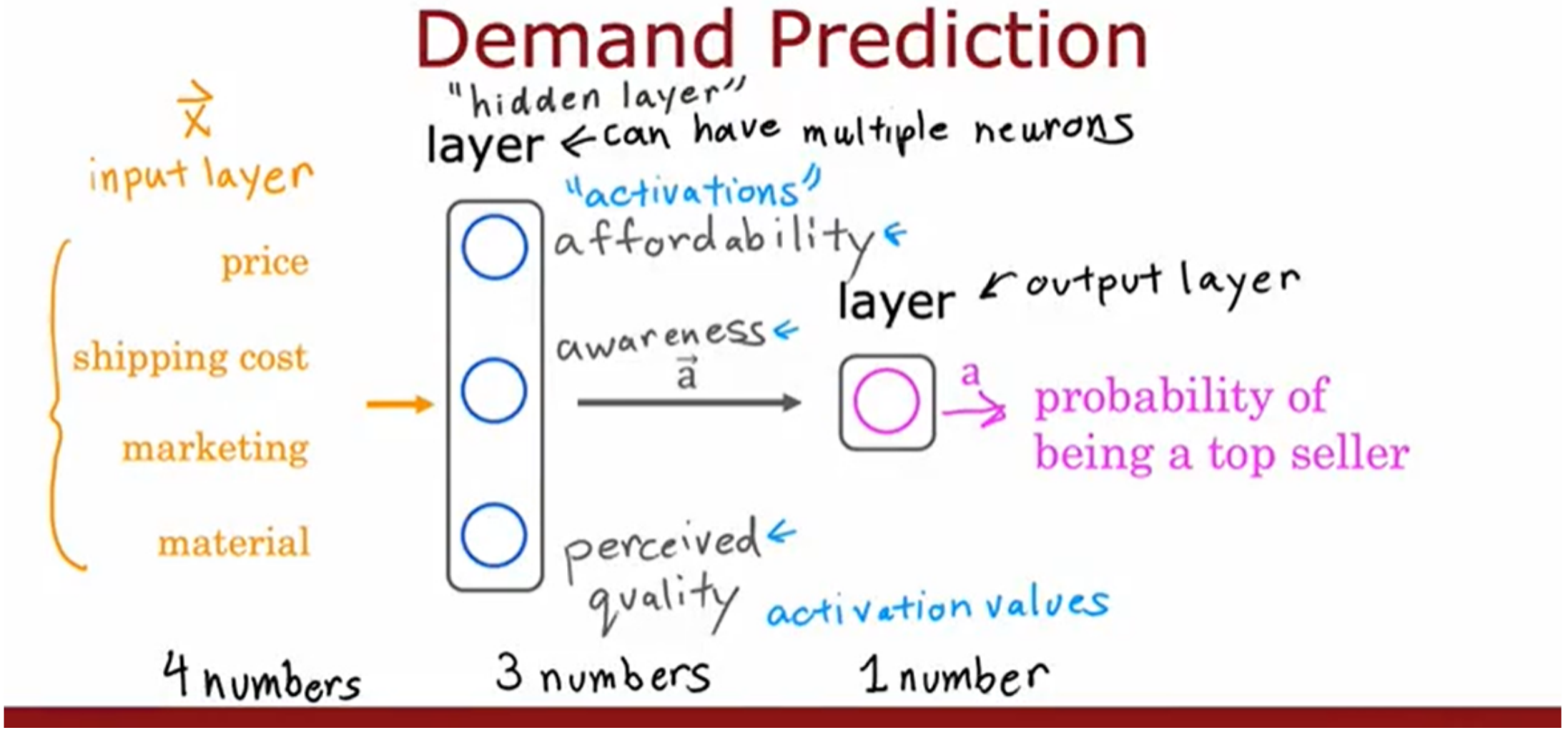
neuron

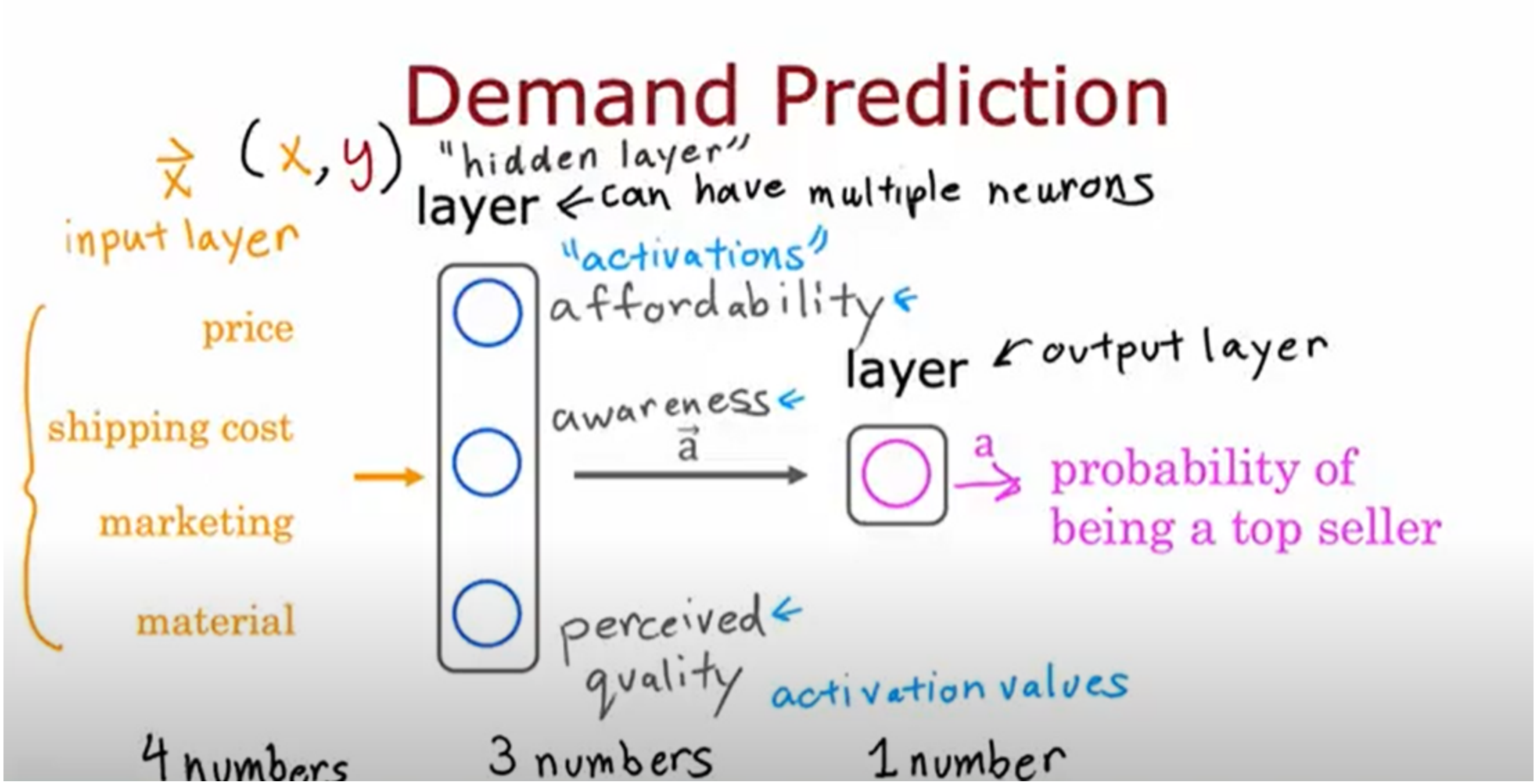
Demand Prediction



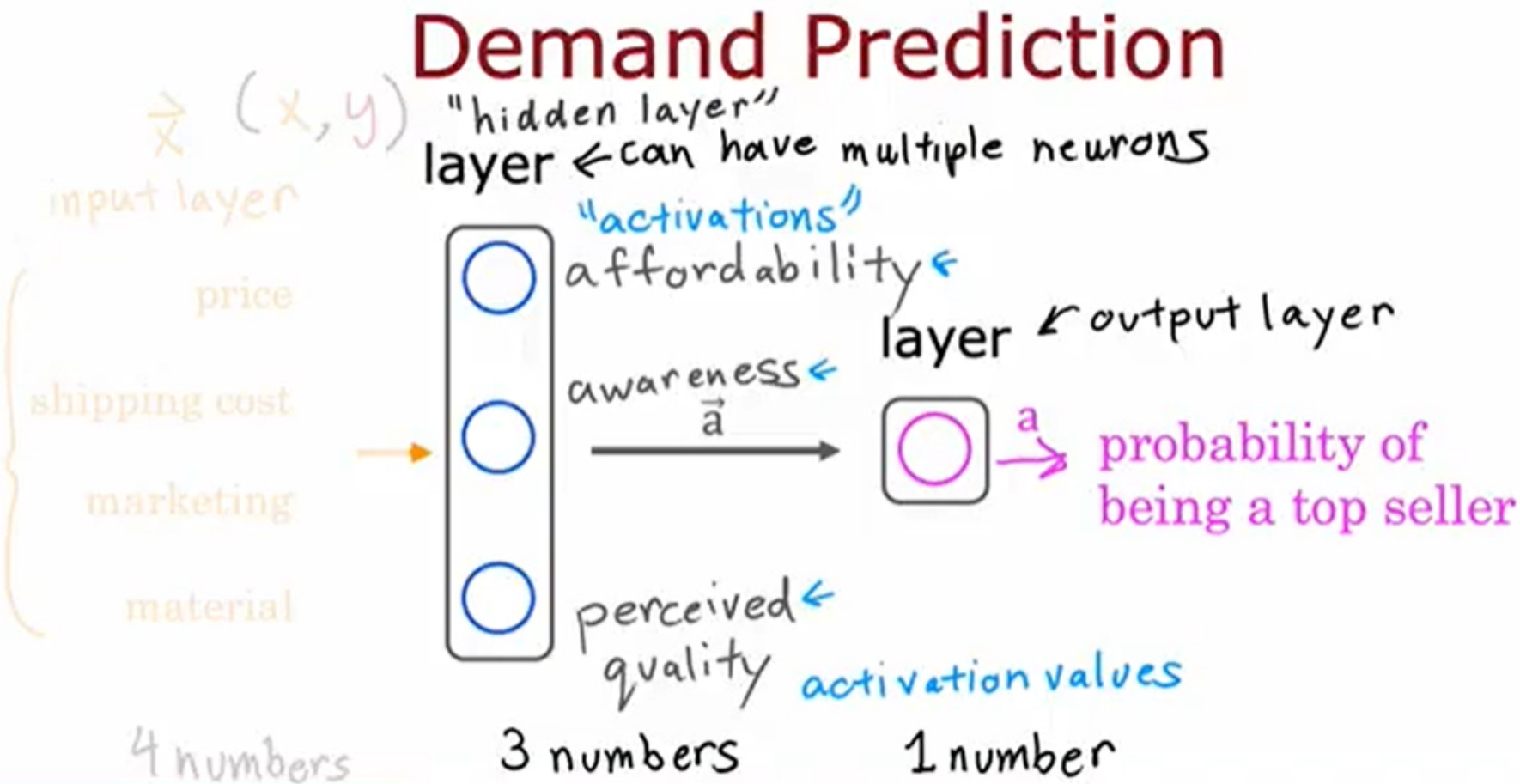
Demand Prediction

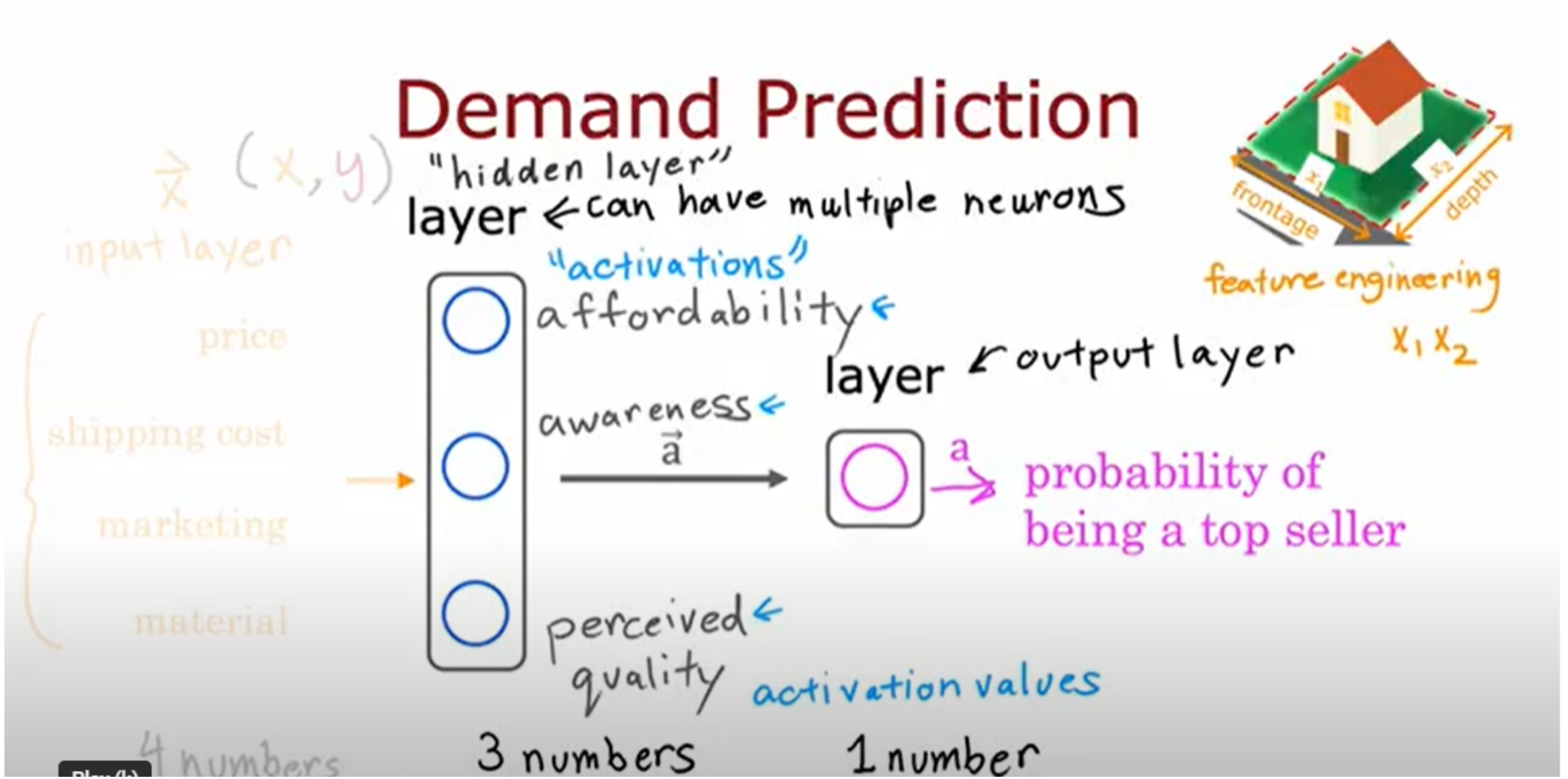



Demand Prediction
"hidden layer"
layer ← can have multiple neurons
$\vec{x}$
input layer
price
shipping cost
marketing
material
"activations"
affordability ←
awareness ←
$\vec{a}$
perceived quality ←
layer ← output layer
a
probability of being a top seller
activation values
4 numbers
3 numbers
1 number

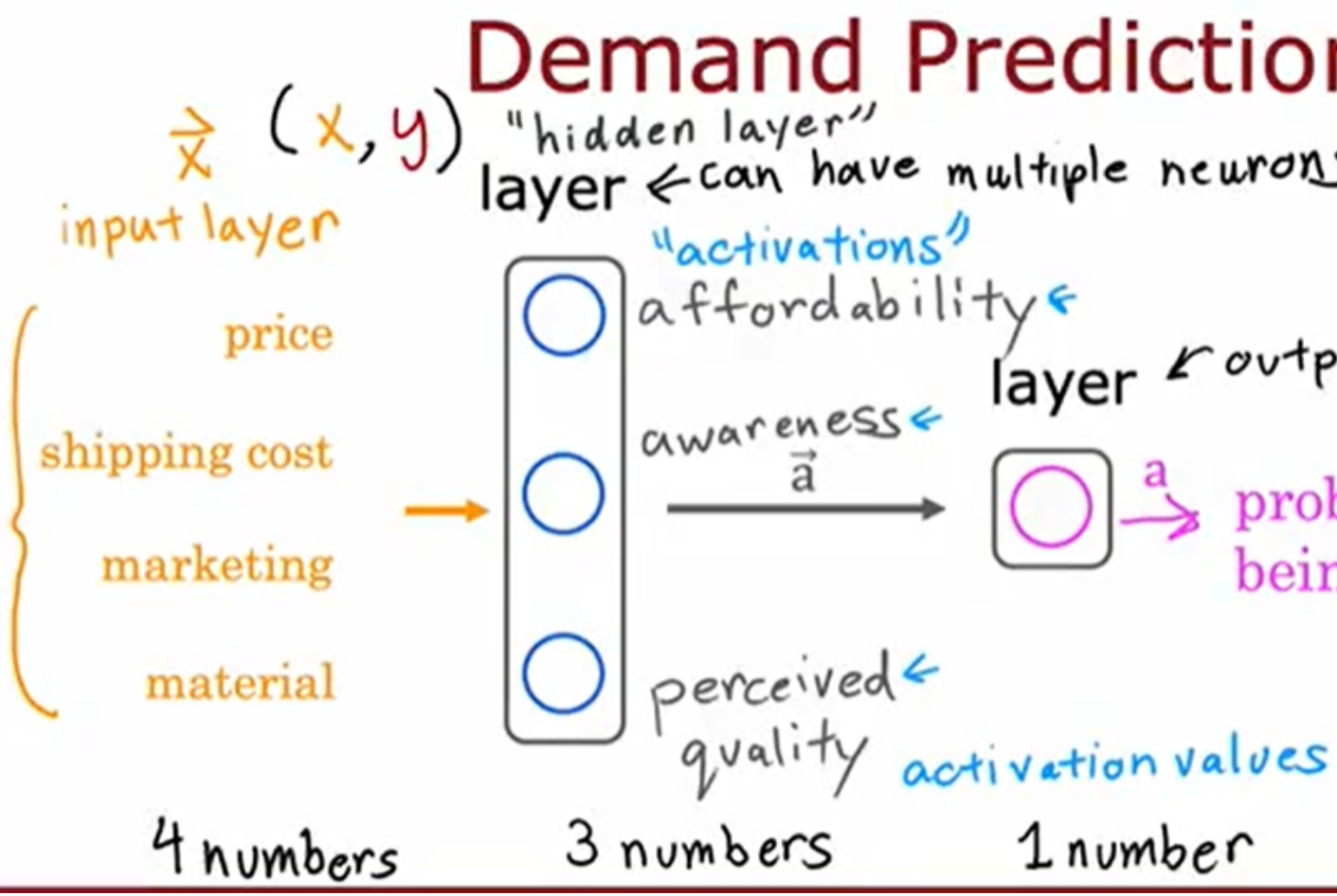

Demand Prediction
$\vec{x}$ (x, y)
"hidden layer"
layer ← can have multiple neurons
input layer
"activations"
affordability ←
price
shipping cost
marketing
material
awareness ←
perceived quality ←
$\vec{a}$
layer ← output layer
a
probability of being a top seller
activation values
4 numbers
3 numbers
1 number

Demand Prediction




Demand Prediction
"hidden layer"
layer ← can have multiple neurons
"activations"
affordability ←
awareness ←
perceived quality ←
$\vec{a}$
layer ← output layer
a
probability of being a top seller
activation values
3 numbers
1 number
feature engineering
x_1 x_2
frontage
depth
$\vec{x}$ (x, y)
input layer
price
shipping cost
marketing
material
4 numbers

Demand Prediction



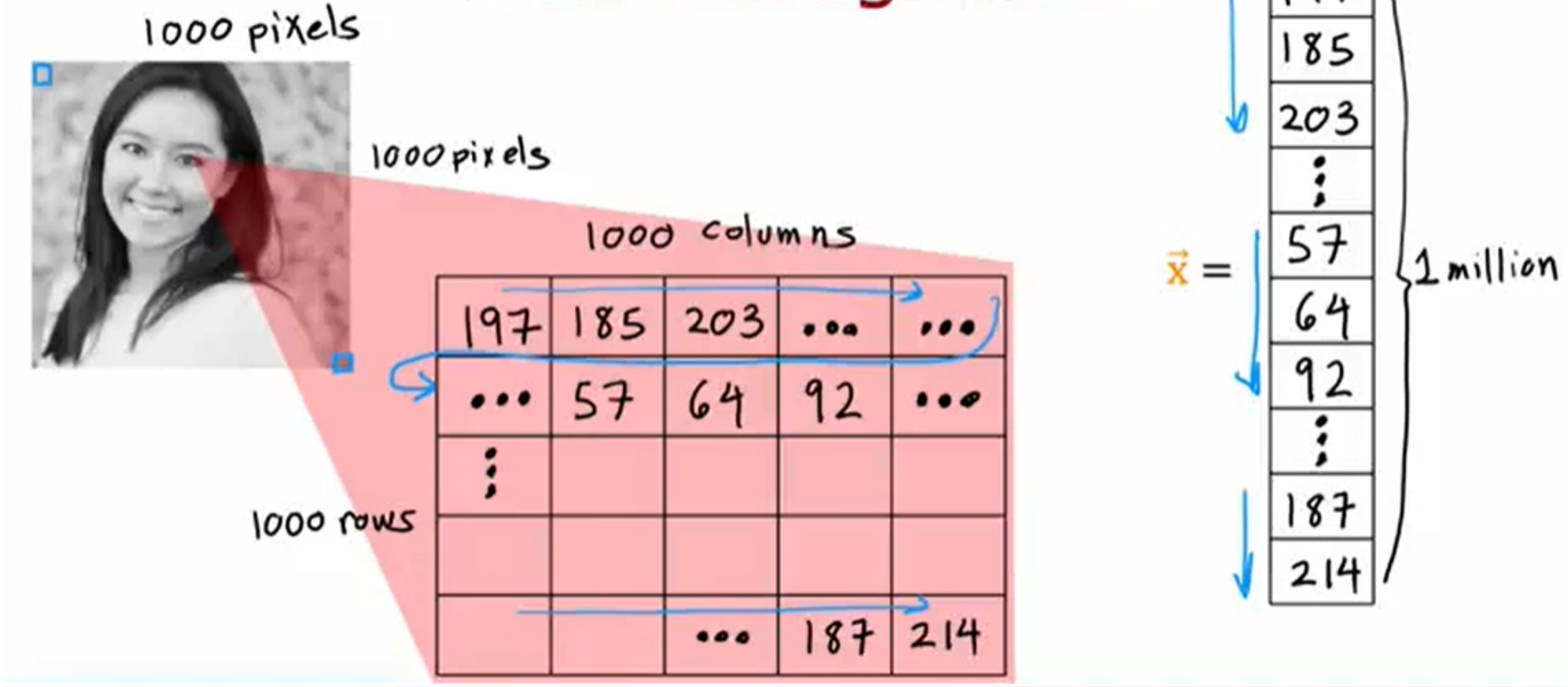
feature engineering

x_1 x_2

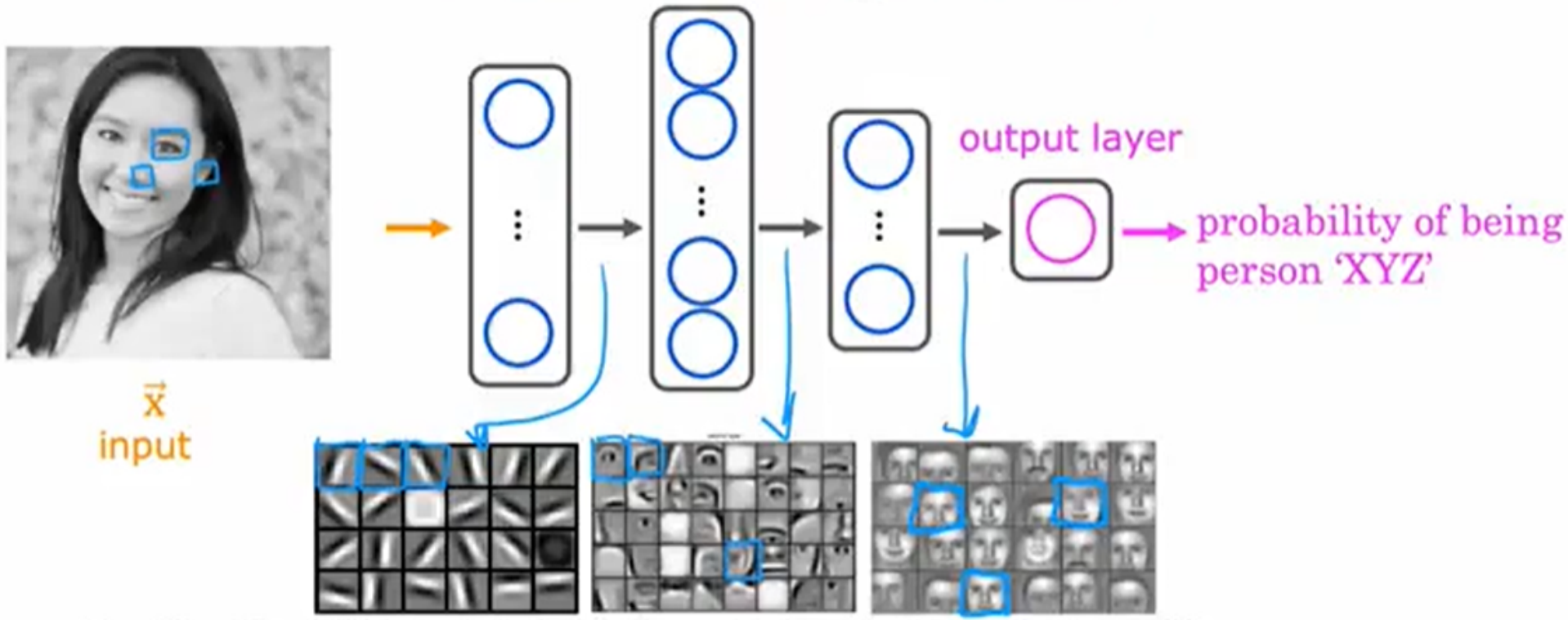
Neural Networks Intuition

Example: Recognizing Images

Face recognition

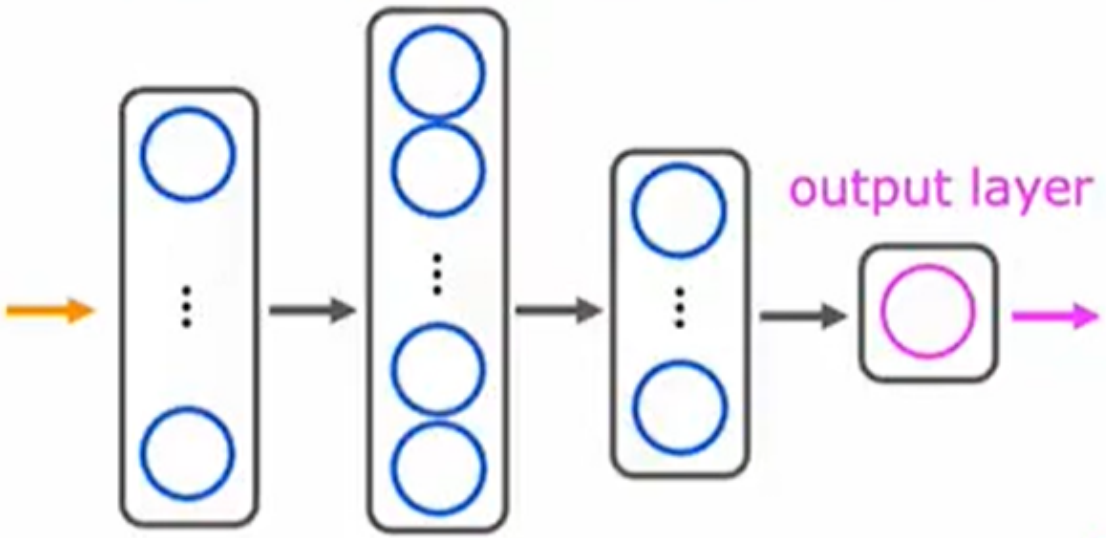


Face recognition



source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

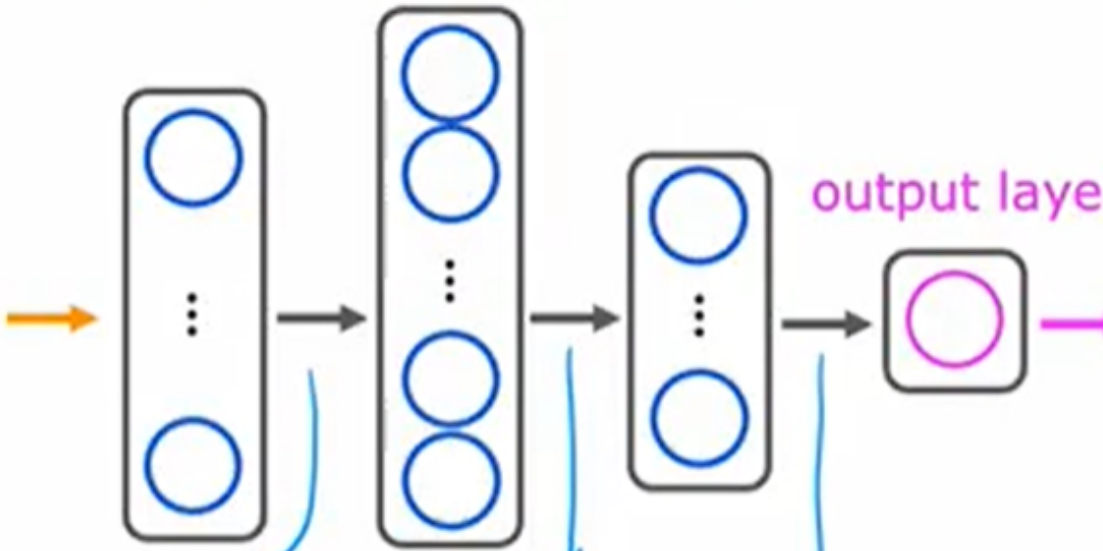
Car classification



activations are higher level features

source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations
by Honglak Lee, Roger Grosse, Rangaanath Andrew Y. Ng

Car classification



activations are higher level features

source: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations by Honglak Lee, Roger Grosse, Ranganath Andrew Y. Ng

Learn XNOR function using 2 layers NN