CSCI 1470/2470
Spring 2022

Ritambhara Singh

January 29, 2024
Monday

Deep Learning

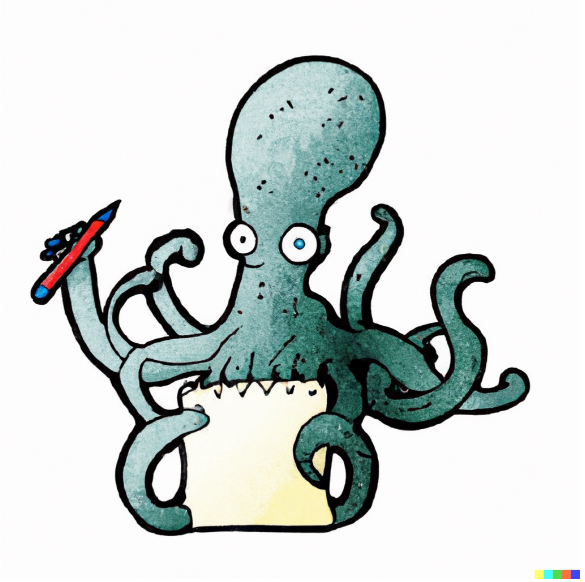
Perceptron

ChatGPT prompt “minimalist landscape painting of a deep underwater scene with a blue tang fish in the bottom right corner”

Recap

How to represent inputs and outputs

- Represent input and output as numbers
- Classification – predicting categorical outputs
- Regression – predicting numerical outputs



Supervised Learning

- Learn a function that approximates the data well
- Get more data!
- Try different models
- Pick a good model

Real world data tends to be complicated!

Input: $\mathbb{X}$

"Temperature" "Stand Hours" "Sunny?"

$x_1^{(1)} = 100.1$ $x_2^{(1)} = 8$ $x_3^{(1)} = 1$

$x_1^{(2)}$

$x_1^{(3}$

.
.
.
.

$x_i^{(k)} = \cdots$

$\mathbb{X} \in \mathbb{R}^3$

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

180.5

115.1

$y^{(k)} = \ldots$

$\mathbb{Y} \in \mathbb{R}$

(Numerical output)

Homework 1 - Diabetes dataset
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

number of samples (n) =442
number of features (p) = 10
output (y): quantified disease progression
Linear regression task

(Image only for explaining concept, not drawn accurately)

Today's goal - Learn about the first component of deep learning model

Perceptron:

(1) Machine Learning problem – Recognizing handwritten digits

(2) Perceptron

(3) Parameters – weights and biases

Handwritten digit recognition

Motivation: ZIP codes

- In 1990s, great increase in documents on paper (mail, checks, books, etc.)
- Motivation for a ZIP code recognizer on real U.S. mail for the postal service!

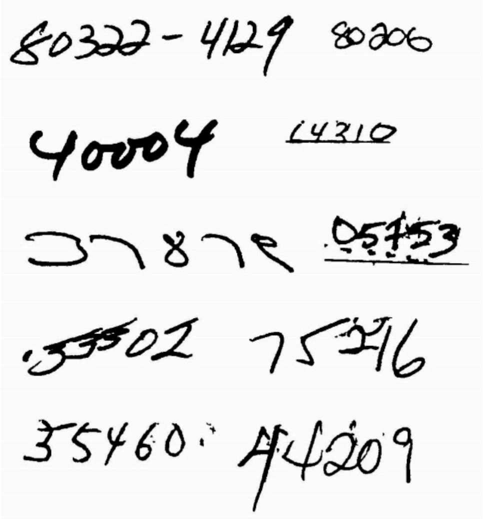


Image from Lecun, Yann, et al.

Our Problem:

Input: $\mathbb{X}$

Function: f

$\boxed{f(\mathbb{X}) \rightarrow \mathbb{Y}}$

Target: $\mathbb{Y}$

Which digit is it?

"3"

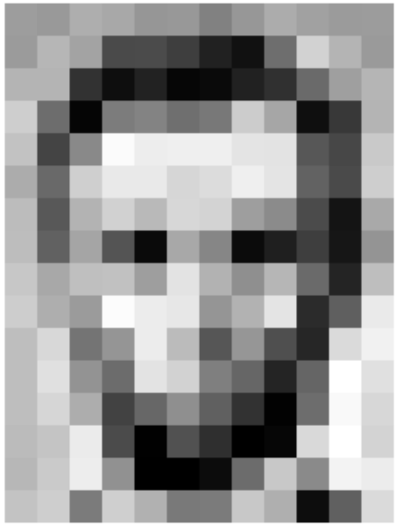
How does a computer know this is a three?

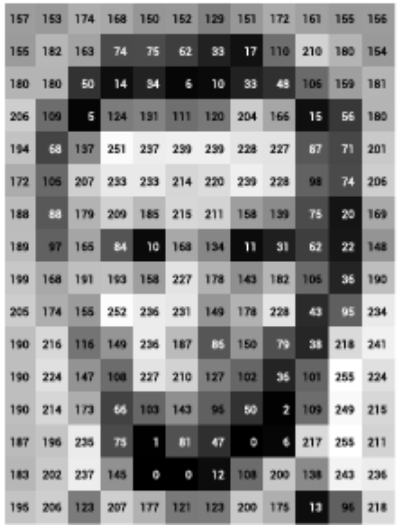
“three”

Representing digits in the computer

- Numbers known as ***pixel values*** (a grid of discrete values that make up an image)

0 is white, 255 is black, and numbers in between are shades of gray

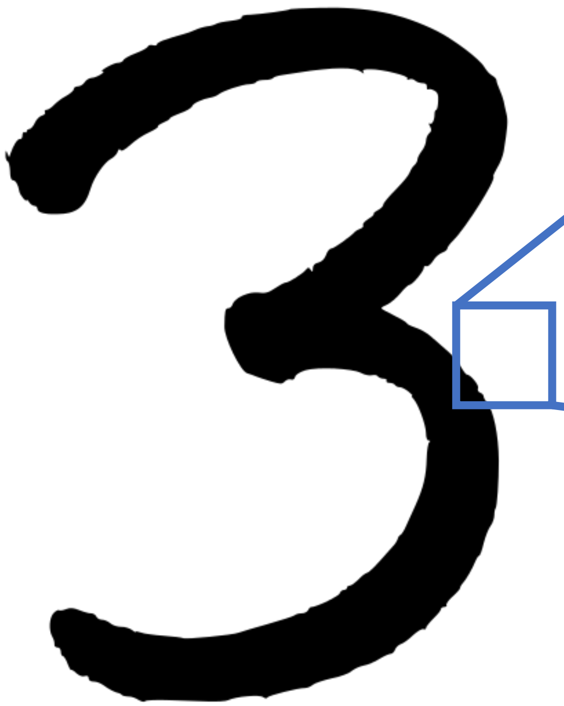




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

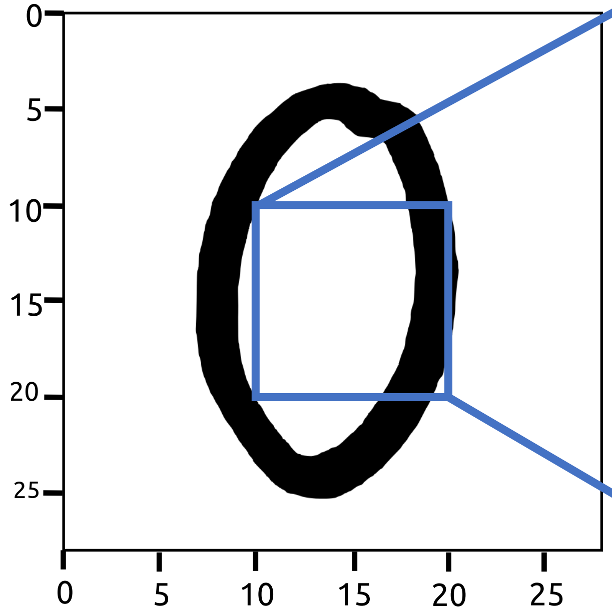
Image from [Stanford]

How is this different from the color image example in the last class?



0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
120	0	0	0	0	0	0	0	0	0	0	0
240	40	0	0	0	0	0	0	0	0	0	0
242	128	0	0	0	0	0	0	0	0	0	0
255	240	10	0	0	0	0	0	0	0	0	0
254	244	120	0	0	0	0	0	0	0	0	0
255	255	121	8	0	0	0	0	0	0	0	0

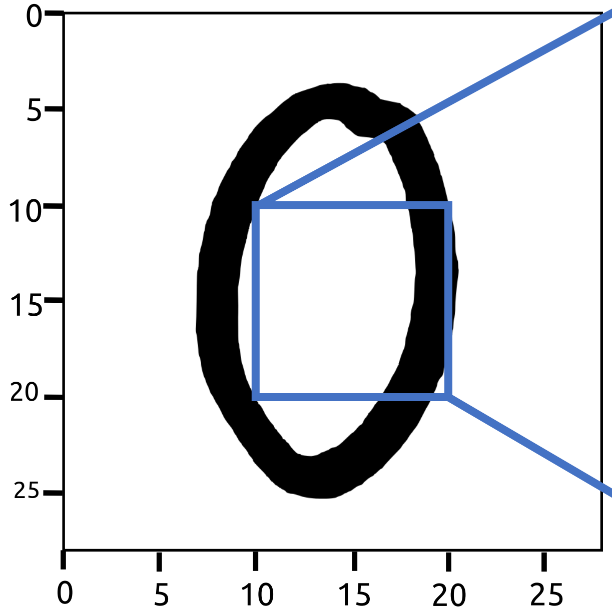
what the computer sees



	10	11	12	13	14	15	16	17	18	19
10	0	0	0	0	0	0	0	170	236	255
11	0	0	0	0	0	0	0	0	250	254
12	0	0	0	0	0	0	0	0	255	255
13	0	0	0	0	0	0	0	0	240	245
14	0	0	0	0	0	0	0	0	245	236
15	0	0	0	0	0	0	0	0	251	253
16	0	0	0	0	0	0	0	124	255	255
17	0	0	0	0	0	0	0	210	254	120
18	0	0	0	0	0	0	0	165	230	0
19	0	0	0	0	0	0	120	154	220	0

- Pixel in position [15, 15] is light.

what the computer sees

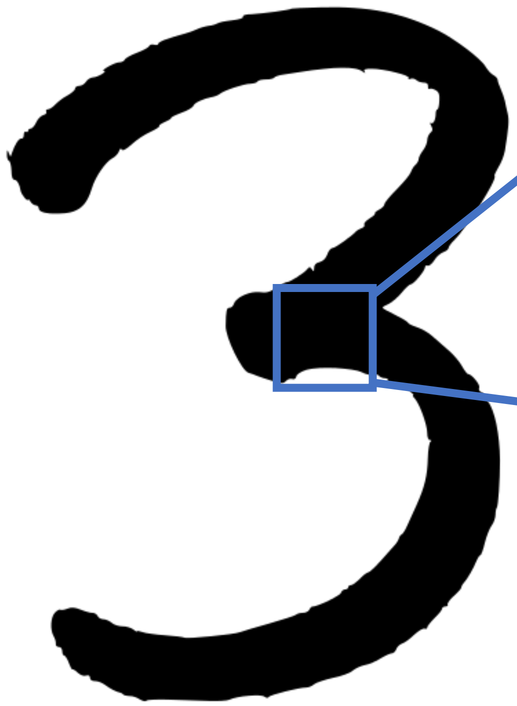


	10	11	12	13	14	15	16	17	18	19
10	0	0	0	0	0	0	0	170	236	255
11	0	0	0	0	0	0	0	0	250	254
12	0	0	0	0	0	0	0	0	255	255
13	0	0	0	0	0	0	0	0	240	245
14	0	0	0	0	0	0	0	0	245	236
15	0	0	0	0	0	0	0	0	251	253
16	0	0	0	0	0	0	0	124	255	255
17	0	0	0	0	0	0	0	210	254	120
18	0	0	0	0	0	0	0	165	230	0
19	0	0	0	0	0	0	120	154	220	0

Often has lighter pixels in the middle!

How does the pattern compare with digit 3?

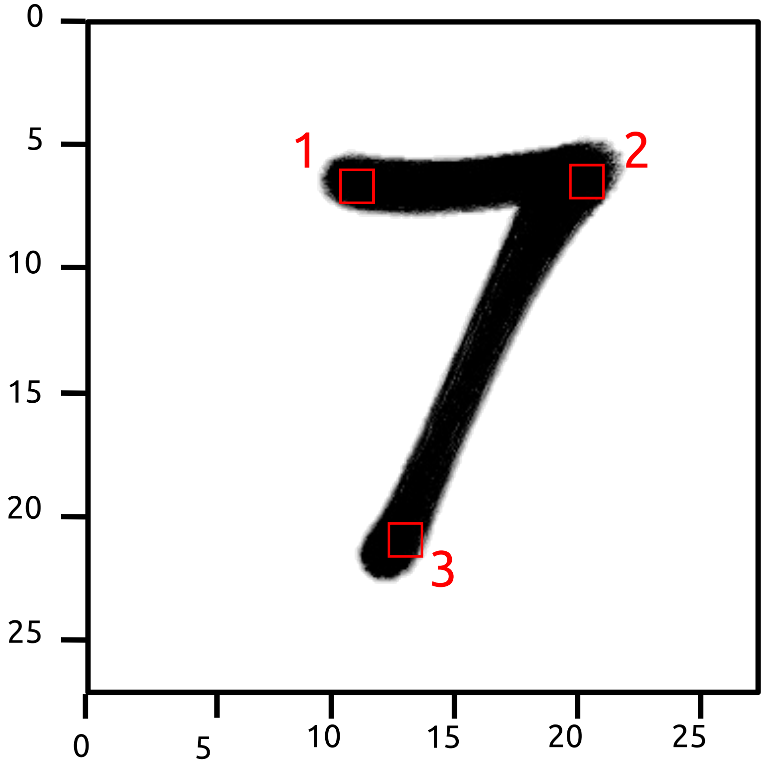
Darker pixels in the middle



255	255	255	255	255	253	254	245	255
255	255	251	255	255	255	254	235	252
255	252	255	250	255	245	255	253	234
253	255	255	255	251	254	255	255	235
255	255	252	255	249	255	239	243	255
255	250	255	245	255	255	254	244	254
255	255	255	255	249	255	255	255	244
249	255	253	255	233	255	249	245	239
255	255	255	250	255	254	251	243	251
245	240	244	240	239	244	255	244	248
242	128	140	150	130	128	110	245	246
240	240	4	5	4	3	2	118	120
240	5	4	2	0	0	0	4	2
0	0	0	0	0	0	0	0	0

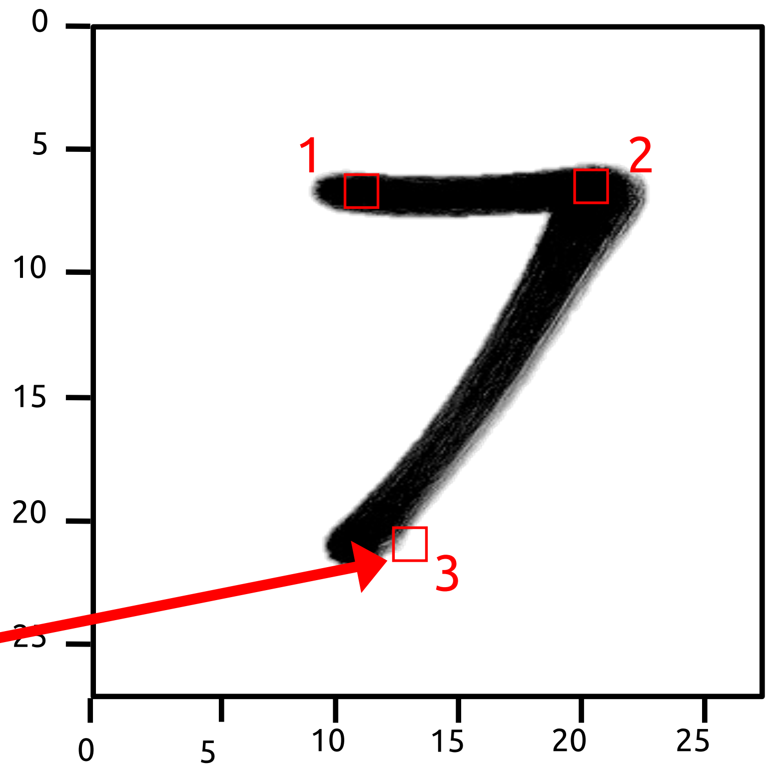
Can we define a set of ***heuristics*** (i.e. rules based on our intuition), to classify digits?

Let’s define some rules (heuristic) for classifying “7”



Digit is a 7 if $P_1 > 128$ and $P_2 > 128$ and $P_3 > 128$

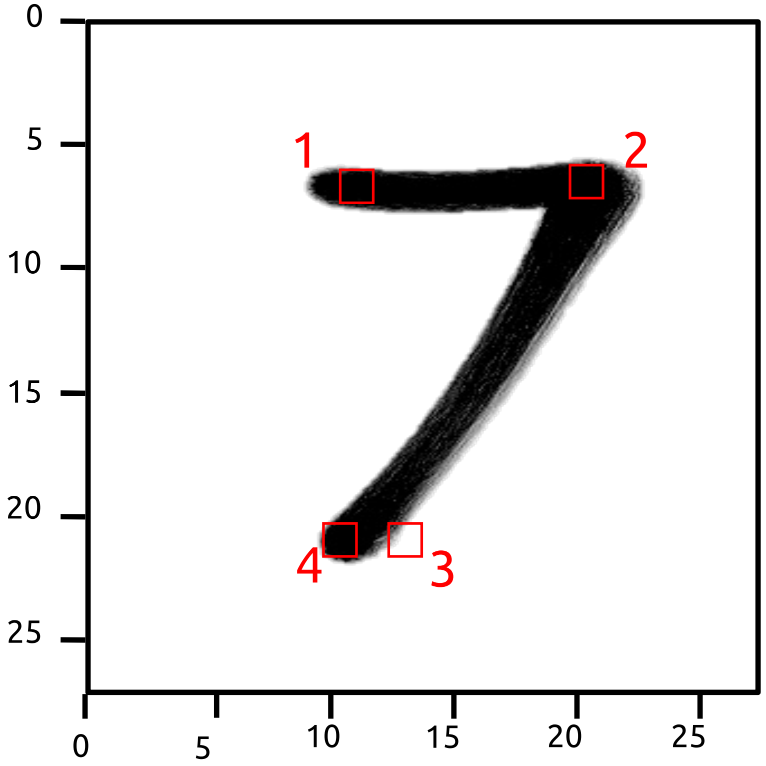
But what if...



pixel 3 is no longer dark!

Slanted digit?

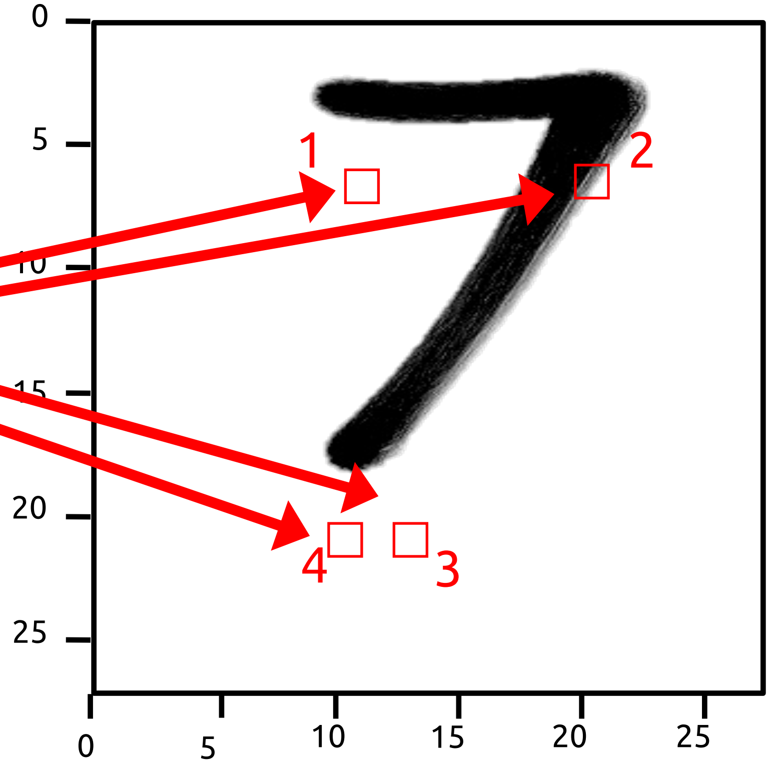
An *Improved* Heuristic!



Digit is a 7 if $P_1 > 128$ and $P_2 > 128$ and $(P_3 > 128$ or $P_4 > 128)$

Not so fast...

the pixel values are completely different



Digit shifted up?

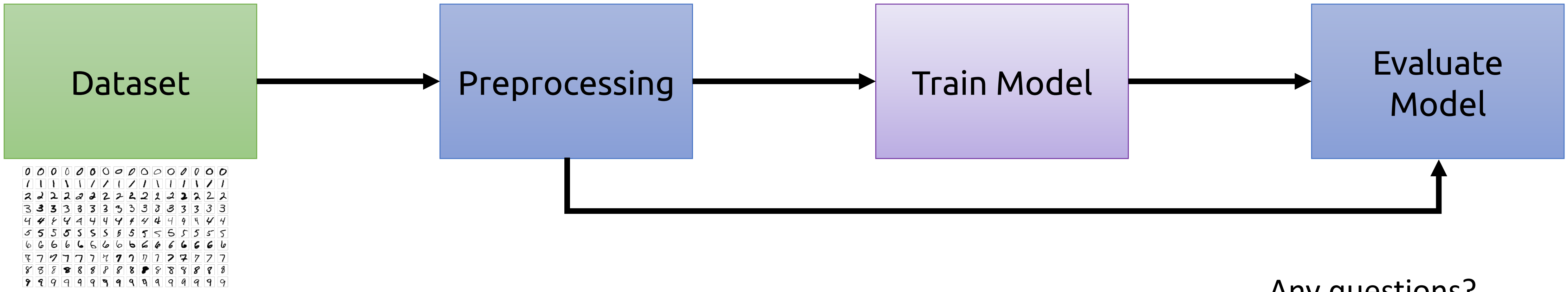
Heuristics...

- Not as simple as we think!
- Distortions, overlappings, underlinings, etc.
- Cannot rely on a set of exact rules

Let's do some machine learning!

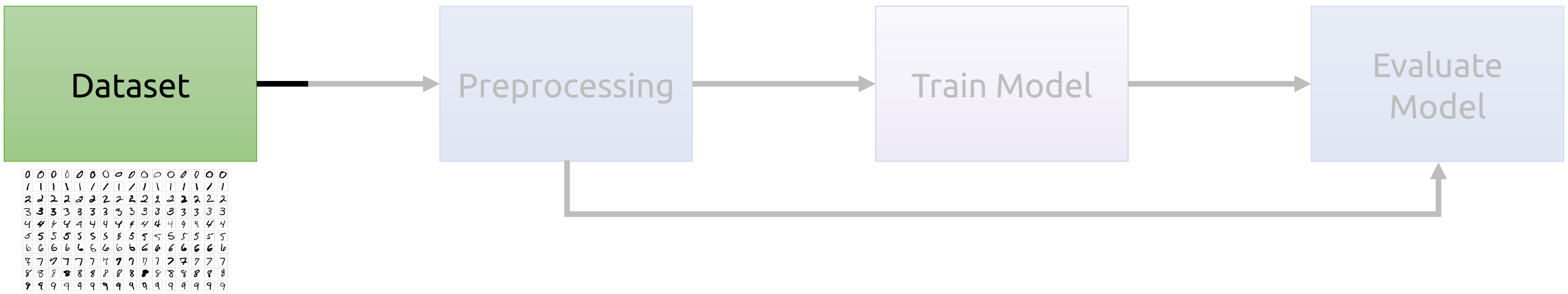
Distorted numbers

Machine Learning Pipeline for Digit Recognition



Any questions?

Machine Learning Pipeline for Digit Recognition



MNIST

- ***M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology* database
- Handwritten digits
- 0 — 9 (10 ***classes***)
- 70,000 images

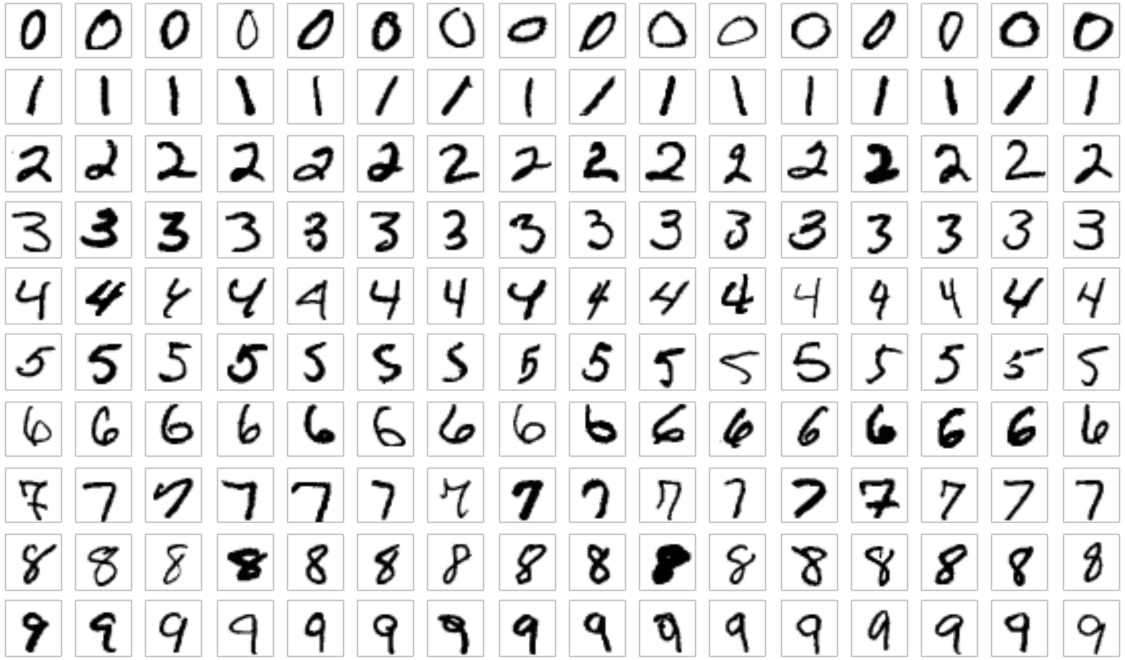
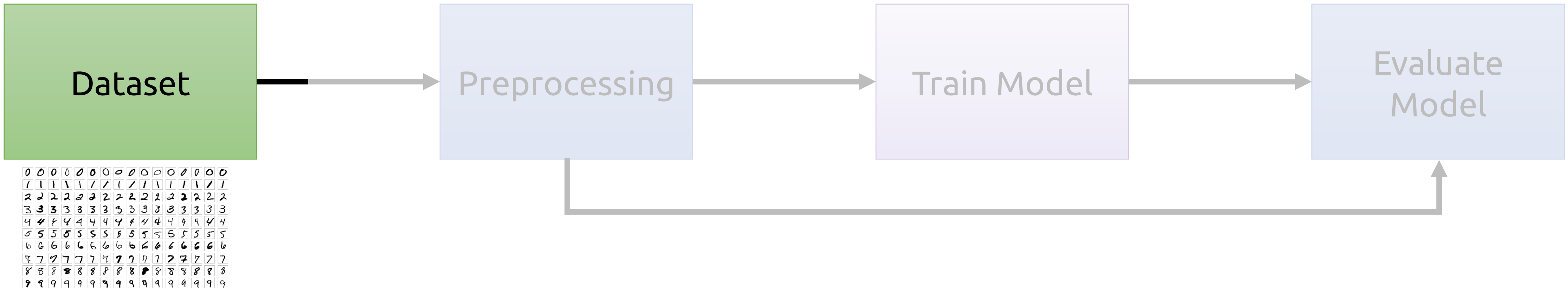
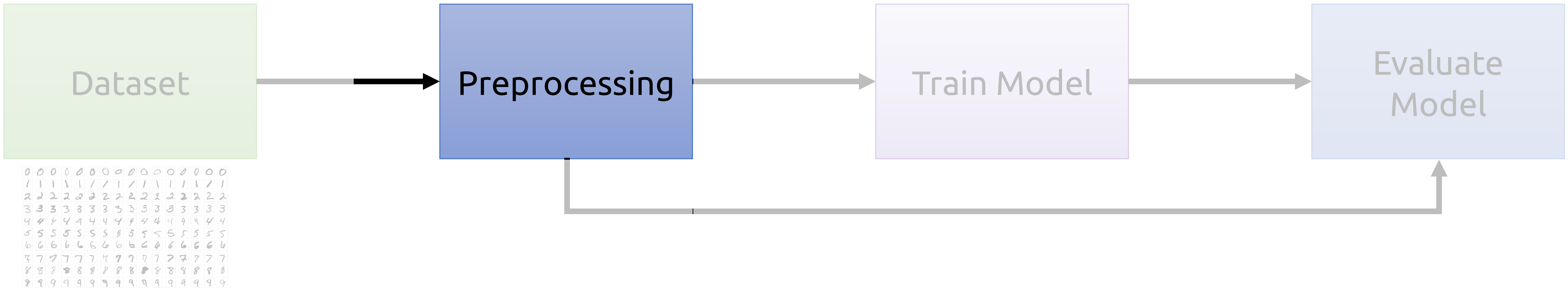


Image from Wikipedia

Machine Learning Pipeline for Digit Recognition

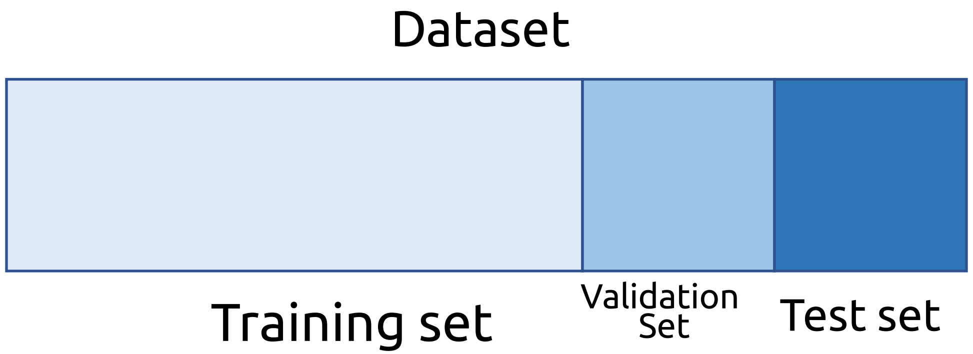


Machine Learning Pipeline for Digit Recognition



Train, validation, and test sets

- ***Train set*** — used to adjust the parameters of the model
- ***Validation set*** — used to test how well we're doing as we develop
  - Prevents ***overfitting***
- ***Test set*** — used to evaluate the model once the model is done



MNIST

- Training set – 60,000 images
- Test set – 10,000 images
- No explicit validation set

What do you suggest we do here?

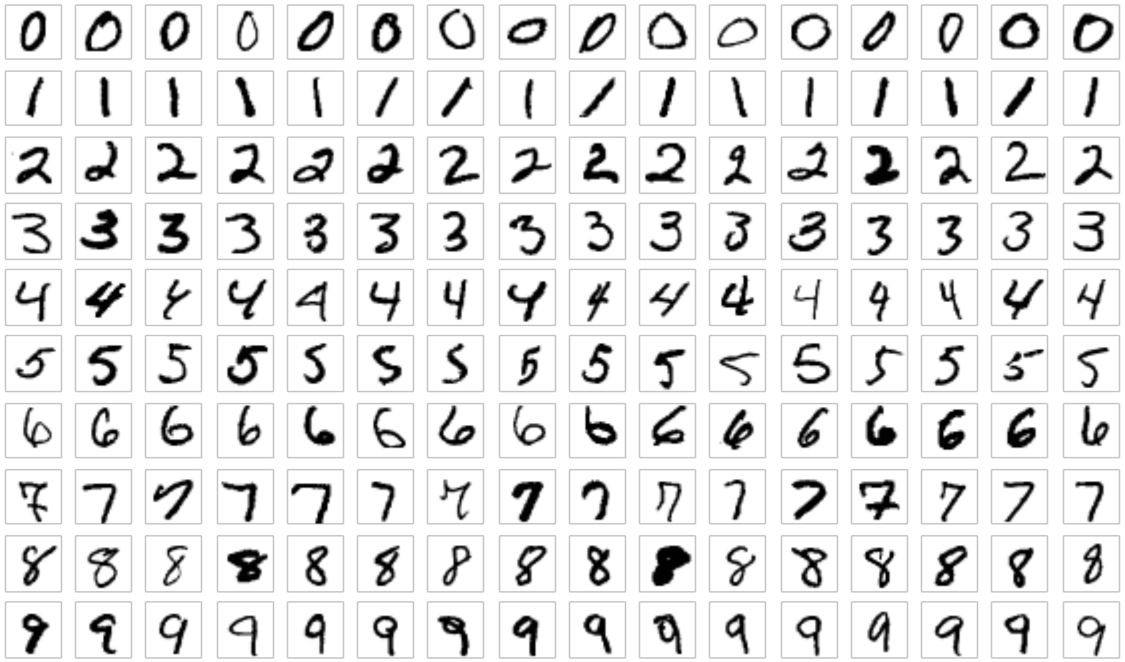
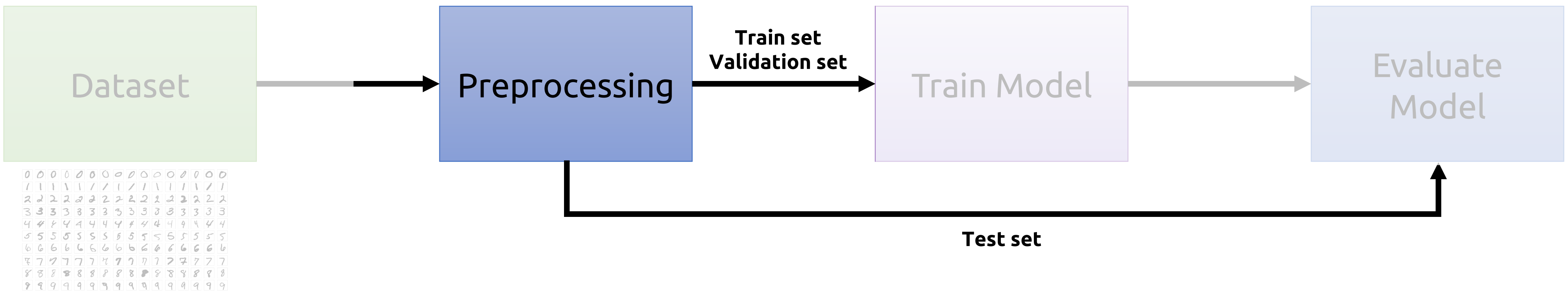
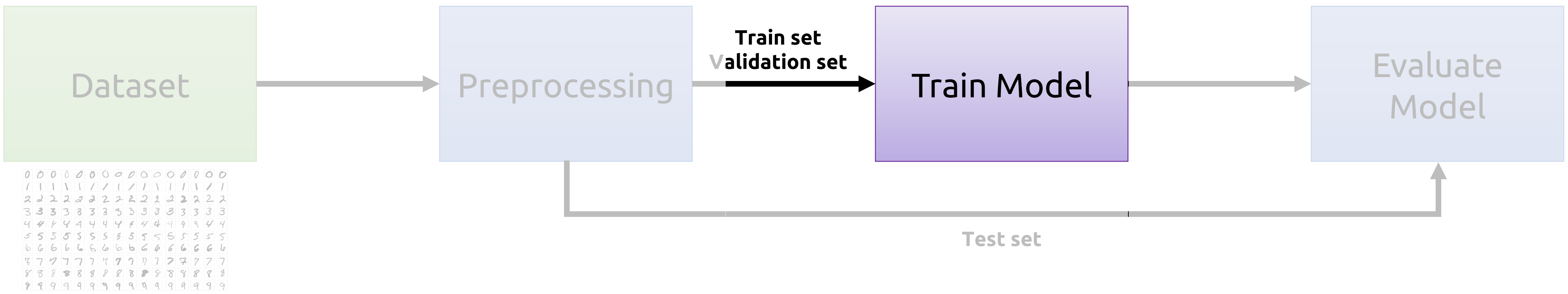


Image from Wikipedia

Machine Learning Pipeline for Digit Recognition



Machine Learning Pipeline for Digit Recognition



Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

What is our input space?

What is our output space?

Target: $\mathbb{Y}$

Pixel Grid

What is our prediction task?

Which digit is it?

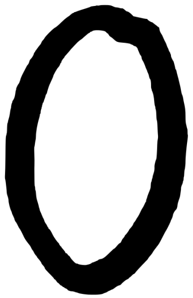
$x^{(1)} =$ 

Function: f

$y^{(1)} =$ "2"

28x28 pixels

$f(\mathbb{X}) \rightarrow \mathbb{Y}$

$x^{(2)} =$ 

$y^{(2)} =$ "0"

Our simplified problem:

What is our input space?

What is our output space?

What is our prediction task?

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

Is it digit 2?

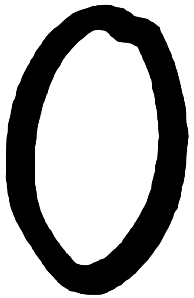
$x^{(1)} =$ 

28x28 pixels

Function: f

$\boxed{f(\mathbb{X}) \rightarrow \mathbb{Y}}$

$y^{(1)} = 1$

$x^{(2)} =$ 

$y^{(2)} = 0$

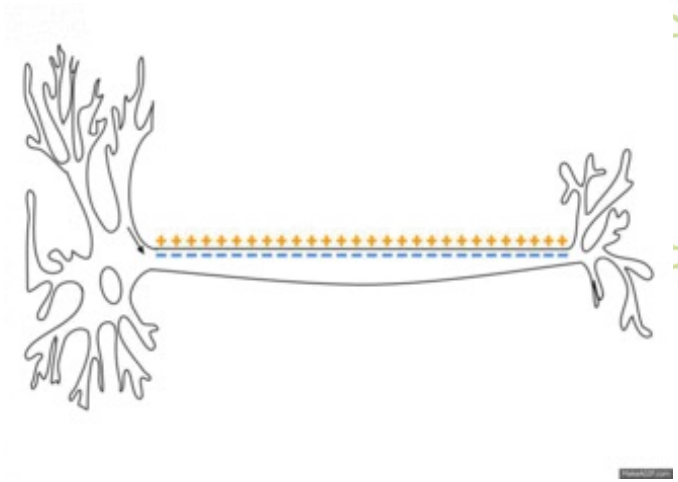
Perceptron

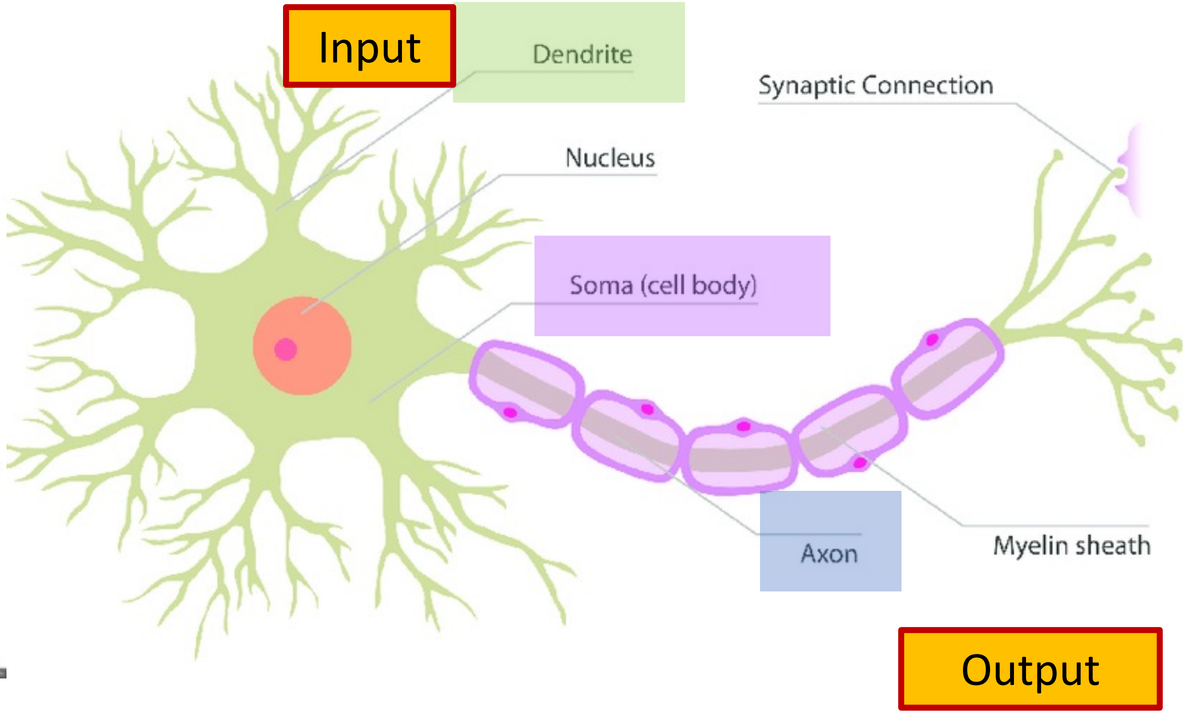
(Our first deep learning model unit)

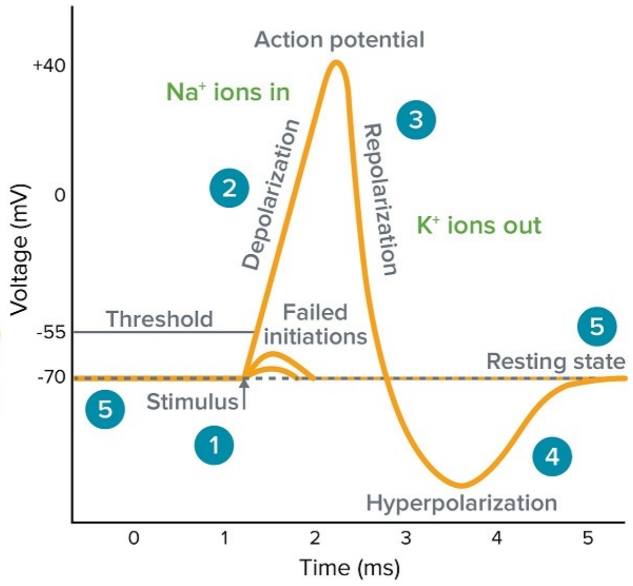
Biological motivation

- Loosely inspired by neurons, basic working unit of the brain
- Serve to transmit information between cells

Inputs are membrane depolarizations!



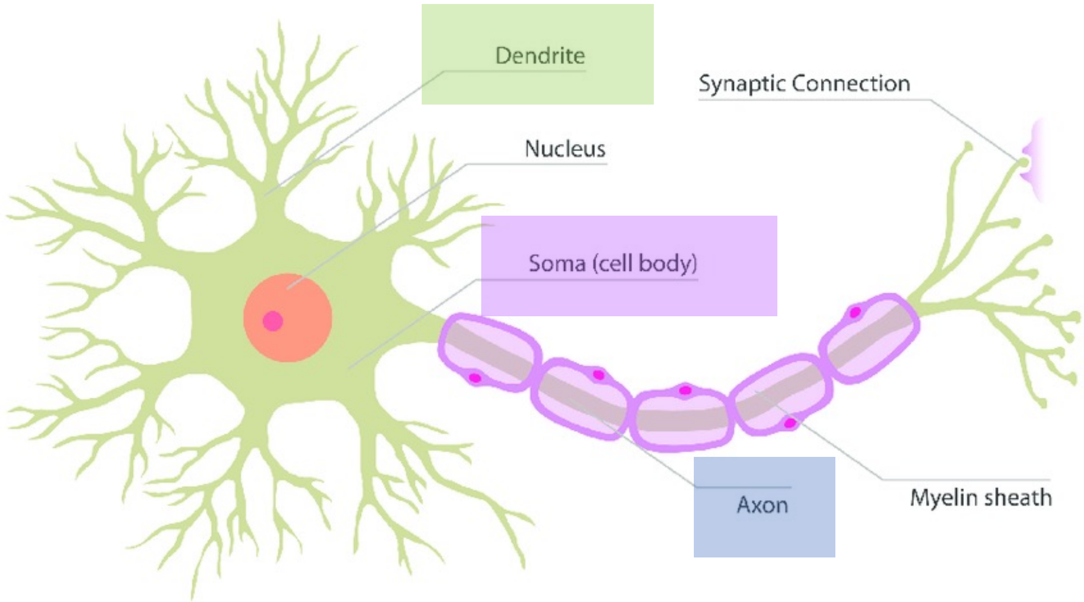


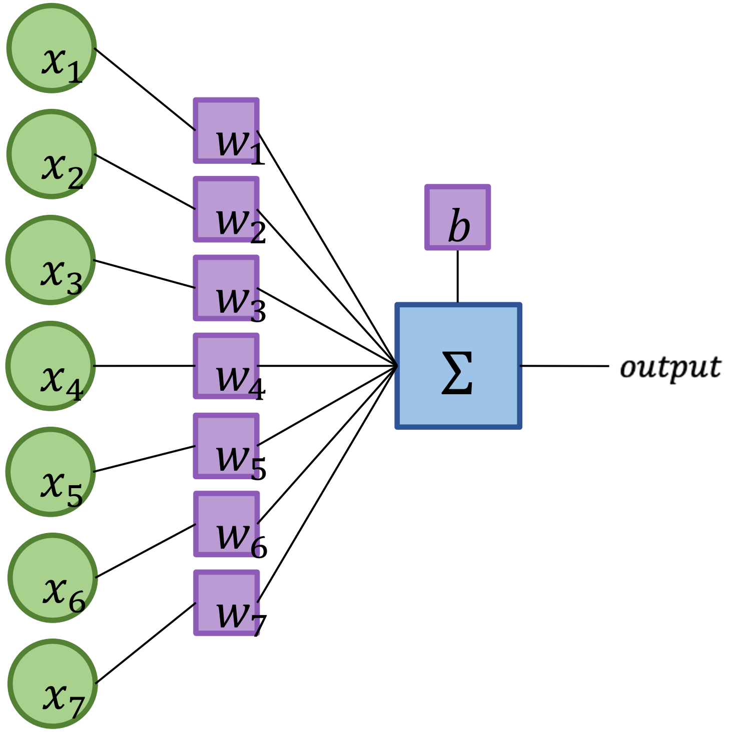


(Depolarization is a change within a cell, during which the cell undergoes a shift in electric charge distribution)
https://en.wikipedia.org/wiki/Depolarization

Image from here

The Perceptron



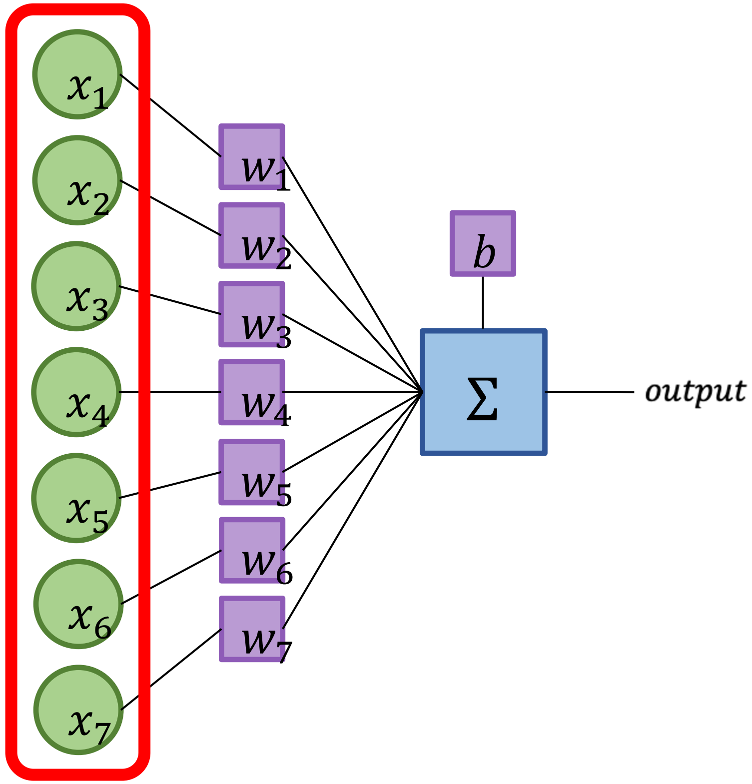


Biological Neuron

Artificial Neuron (Perceptron)

Input

- Input: a vector of numbers $\mathrm{x} = [x_1, x_2, \dots x_n]$



What was x_i for lemonade stand example?

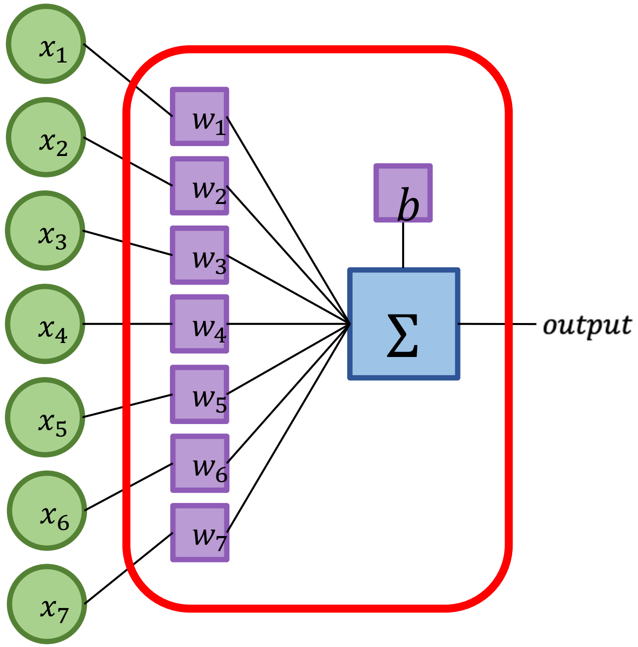
What is x_i MINIST image?

x is represented by a 28 * 28 matrix of pixel values, flattened into a one-dimensional vector (size 784)
(more on this later)

Predicting with a Perceptron

1. Multiply each input x_i by its corresponding weight w_i, sum them up.
2. Add the bias b

Does this look familiar?

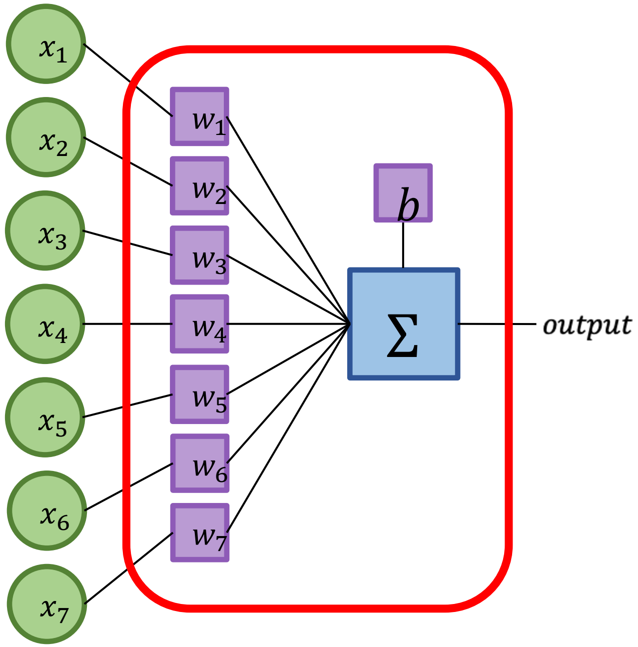


Predicting with a Perceptron

1. Multiply each input x_i by its corresponding weight w_i, sum them up.
2. Add the bias b

3. If the result value is greater than 0, return 1, otherwise return 0

$$f_\Phi(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$



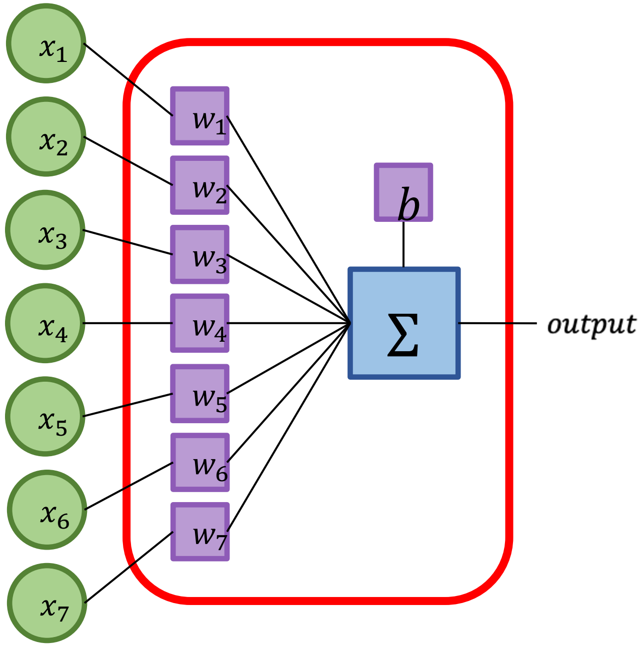
How is perceptron different from linear regression?

Threshold value = 0

Performs binary classification!

Parameters

- w and b are parameters of the perceptron
  - Parameters: values we adjust during learning
  - Let $\Phi = \{w \cup b\}$ (the set of all parameters)

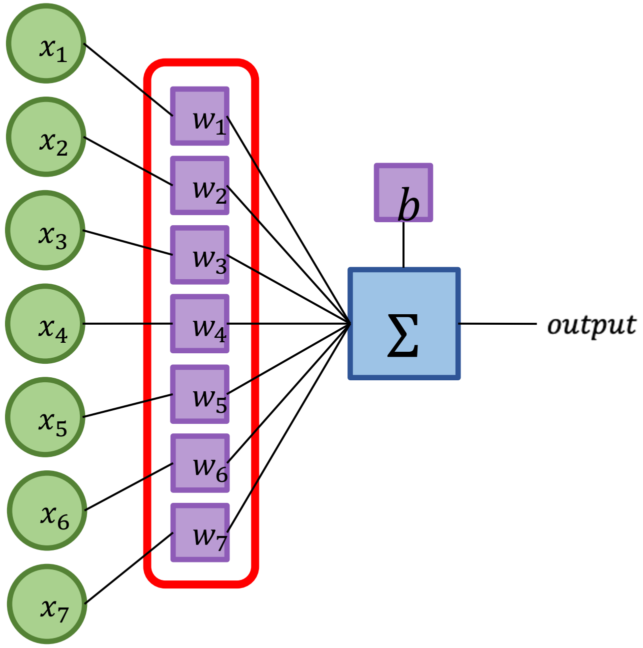


Parameters

Join at menti.com | use code 3340 9640

- ***Weights*** — the importance of each input to determining the output
  - Weight near 0 imply this input has little influence on the output
  - Negative weight means?

$$f_\Phi(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$



Option 1: Increasing input will increase output

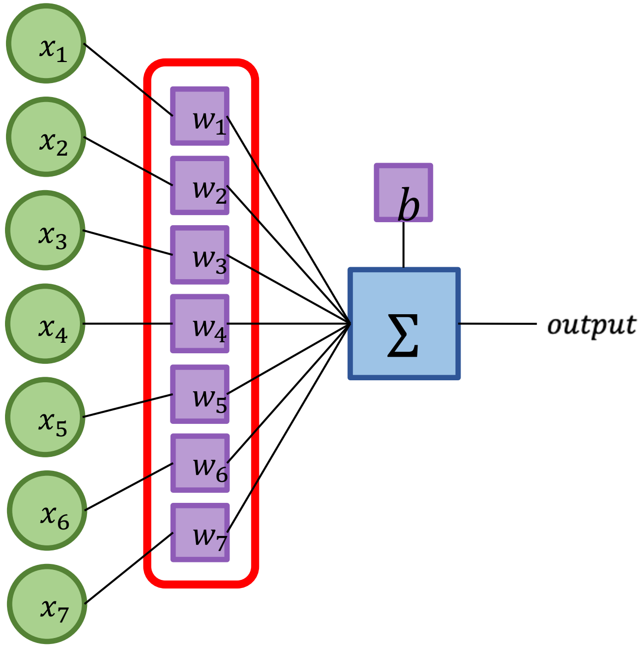
Option 2: Increasing input will decrease output

Option 3: Decreasing input will decrease output

Parameters

- ***Weights*** — the importance of each input to determining the output
  - Weight near 0 imply this input has little influence on the output
  - Negative weight means increasing the input will decrease the output

$$f_{\Phi}(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$

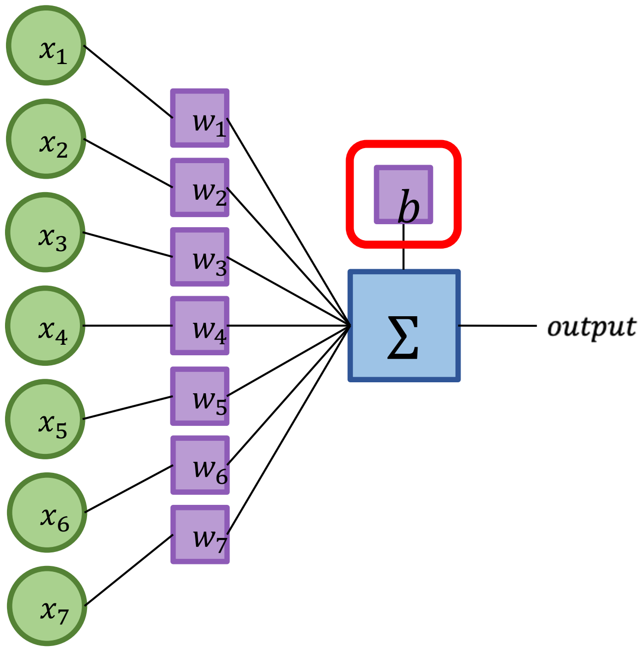


Any questions?

Parameters

- ***Bias*** — What do we need this for?

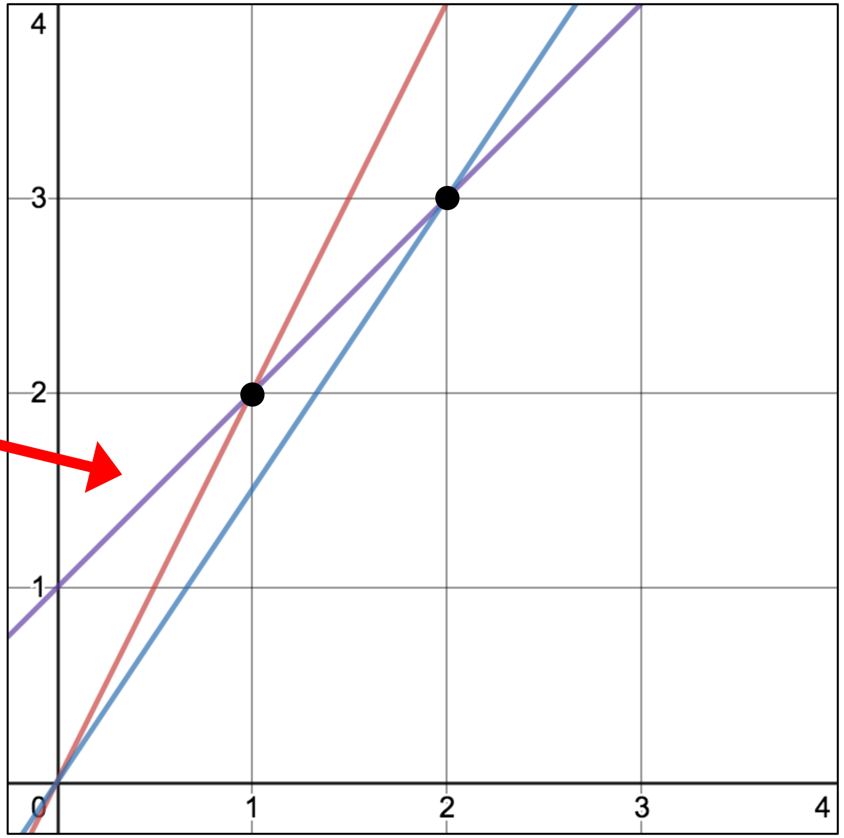
$$f_\Phi(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$



Bias: Geometric Explanation

- the bias is essentially the **b** term in **y = mx+b**

only the line with bias can fit the data



$$f(x) = 2x$$

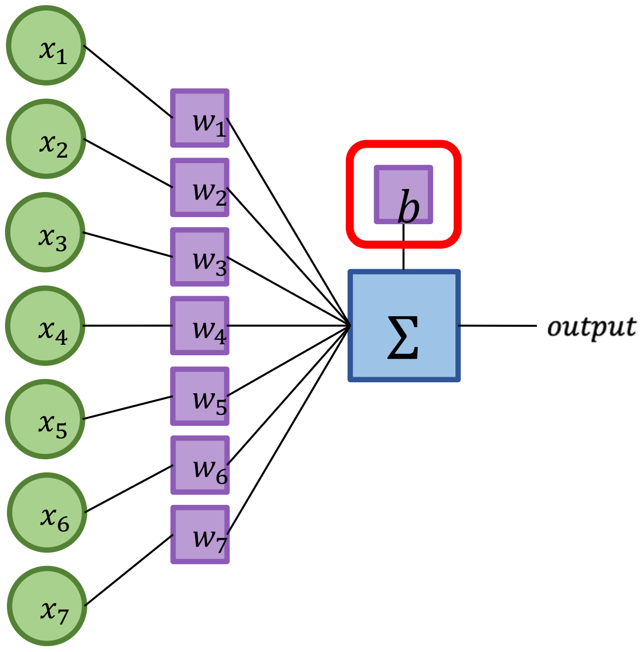
$$f(x) = \frac{3}{2}x$$

$$f(x) = x + 1$$

Bias: Conceptual Explanation

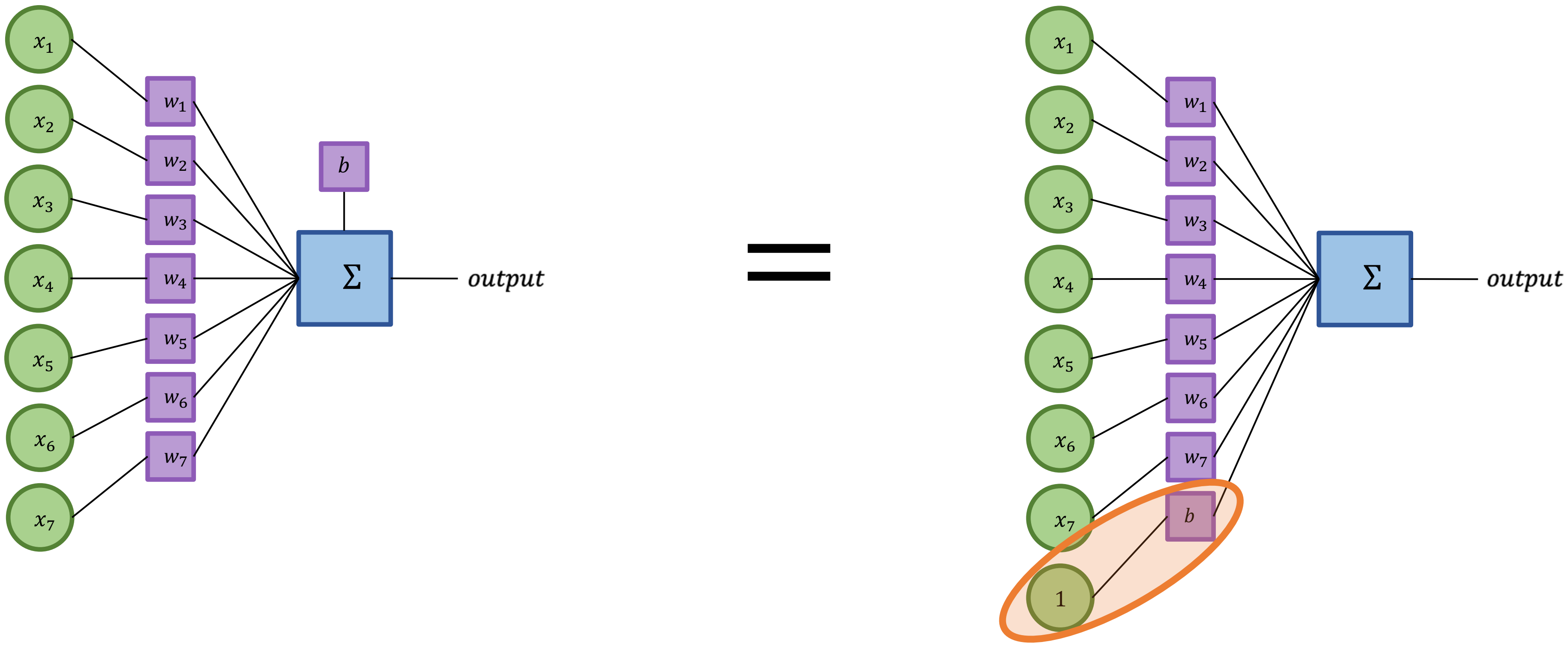
- ***Bias*** — the *a priori* likelihood of the positive class
  - Ensures that even if all inputs are 0, there will be some result value
  - Just because all inputs are 0, it does not mean there are no 1's in the world
  - Maybe there just happen to be more, say, 0's than 1's

$$f_\Phi(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$



Bias as special type of weight

- Another way to think of bias is to represent it as an extra weight for an input/feature that is always 1



Bias as special type of weight

- Another way to think of bias is to represent it as an extra weight for an input/feature that is always 1

$$[x_0, x_1, x_2, \dots x_n] \cdot [w_0, w_1, w_2, \dots w_n] + b$$

$$= [x_0, x_1, x_2, \dots x_n, 1] \cdot [w_0, w_1, w_2, \dots w_n, b]$$

Recall

$\mathbf{a}=[a_1, a_2, \ \ldots, a_n]$ and $\mathbf{b}=[b_1, b_2, \ \ldots, b_n]$ with vector space n,

the dot product is

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Courtesy: MathType

Simplifying some notation...

- Recall: the dot product of two vectors of length n is $\mathbf{a} \cdot \mathbf{b}$ = $\sum_{i=1}^{n} a_i b_i$
- We can rewrite the perceptron function accordingly:

$$f_{\Phi}(x) = \begin{cases} 1, & if\ b + \sum_{i=0}^{n} w_i\, x_i > 0 \\ 0, & otherwise \end{cases}$$

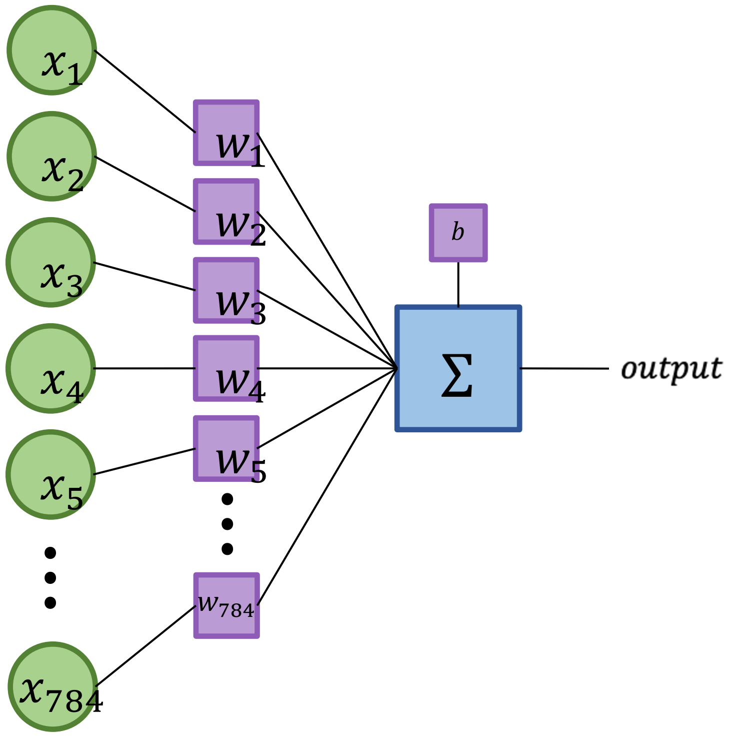
$$f_{\Phi}(x) = \begin{cases} 1, & if\ b + \mathbf{w} \cdot \mathbf{x} > 0 \\ 0, & otherwise \end{cases}$$

Any questions?

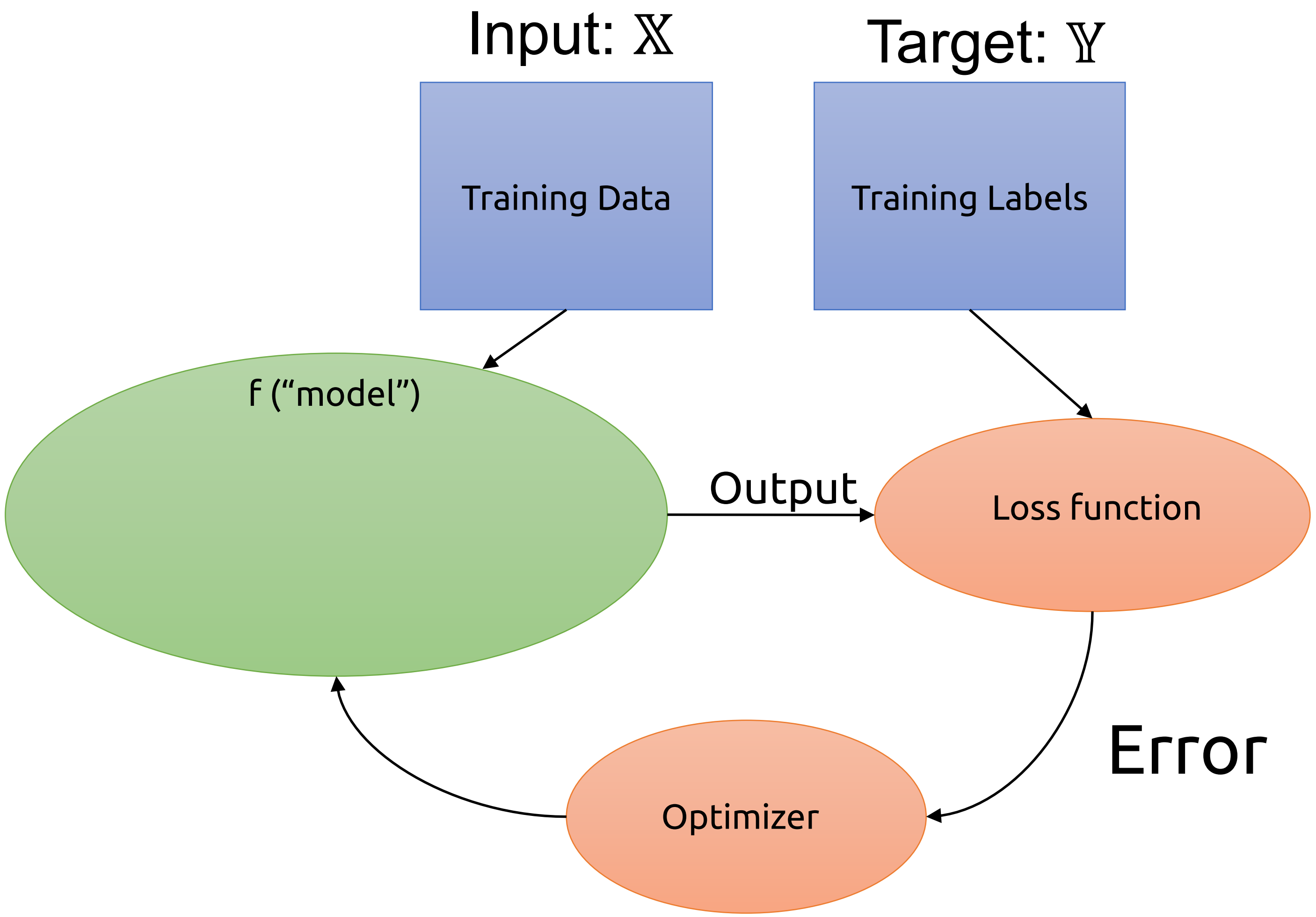
- In modern deep learning parlance, $b + \mathbf{w} \cdot \mathbf{x}$ is known as a ***linear unit***

A Binary Perceptron for MNIST

- *Inputs* $[x_1, x_2, \dots x_n]$ are all positive
  - $n = 784$ (28 ∗ 28 pixel values)
- *output* is either 0 or 1
  - 0 → input is not the digit type we're looking for
  - 1 → input ***is*** the digit type we're looking for



Training a perceptron (Next Class)



The Perceptron Learning Algorithm (Next class)

1. set w's to 0.
2. for N iterations, or until the weights do not change:
   a) for each training example $\mathbf{x}^k$ with label y^k
      i. if $y^k - f(\mathbf{x}^k) = 0$ continue
      ii. else for all weights w_i, $\Delta w_i = \left(y^k - f(\mathbf{x}^k)\right) x_i^k$

- b = bias
- w = weights
- N = maximum number of training iterations
- $\mathbf{x}^k$ = $\mathrm{k^{th}}$ training example
- y^k = label for the $\mathrm{k^{th}}$ example
- w_i = weight for the $\mathrm{i^{th}}$ input where $i \leq n$
- n = number of pixels per image
- x_i^k = $\mathrm{i^{th}}$ input of the example where $i \leq n$

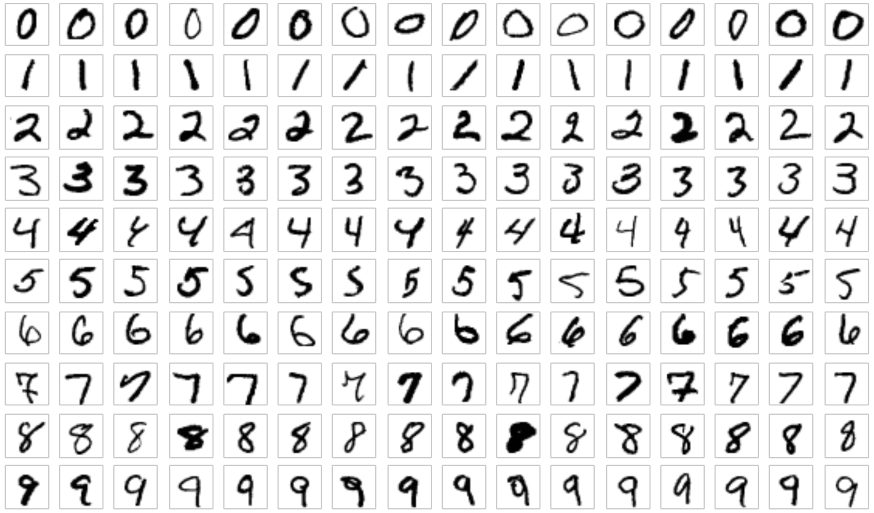
Recap

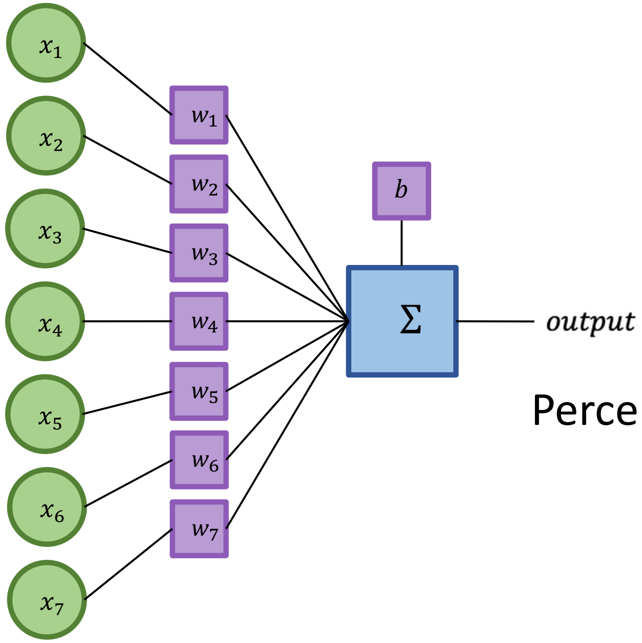
MNIST Data

Representing handwritten digits

Handwritten digit prediction task

Machine learning pipeline





Perceptron

Biological motivation

Perceptron equation

Perceptron parameters and training