CSCI 1470

Eric Ewing

Wednesday, 1/29/25

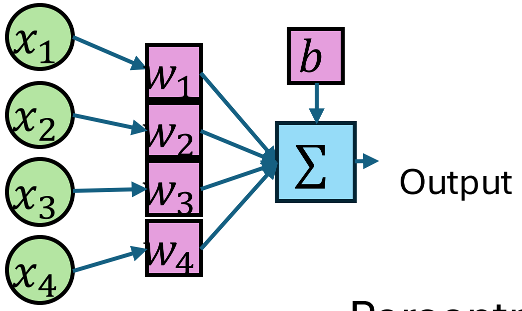
Deep Learning

Day 4: MNIST, Perceptrons, and MLPs

Recap

Linear Regression

- Matrix and Vector Notation
- Closed Form solution for finding optimal parameters



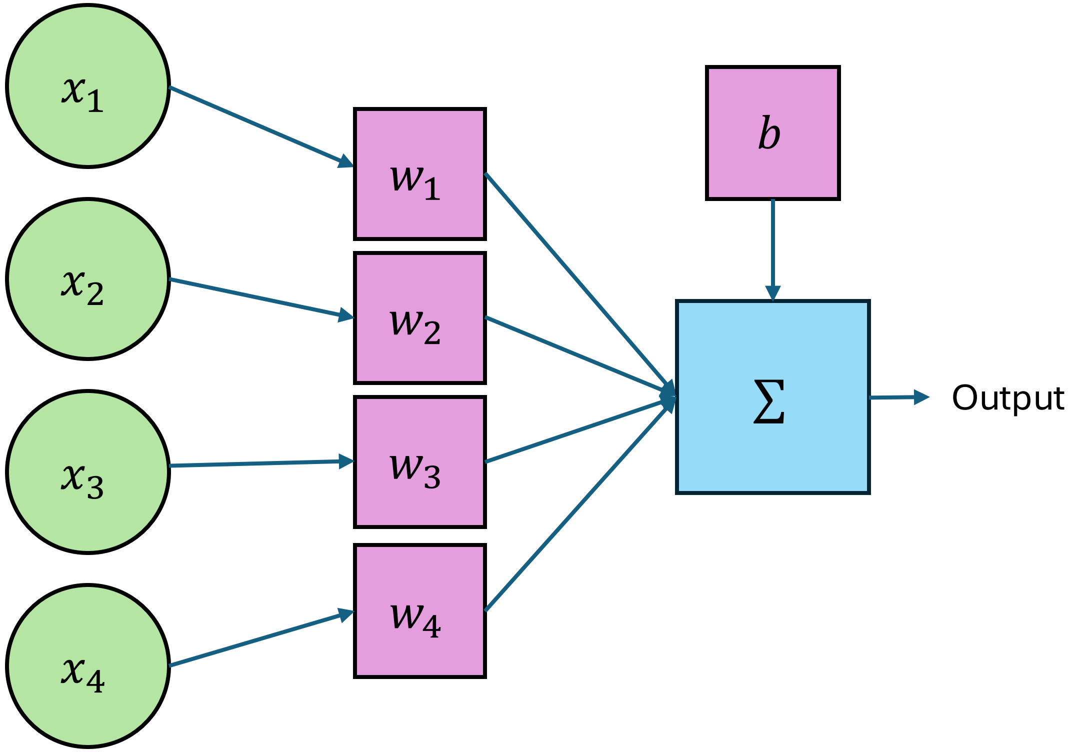
Perceptrons

- natural extension of linear models to binary classification tasks
- Biological inspiration of activation threshold
- Only differ from Linear Regression in terms of activation function

Todays Goals

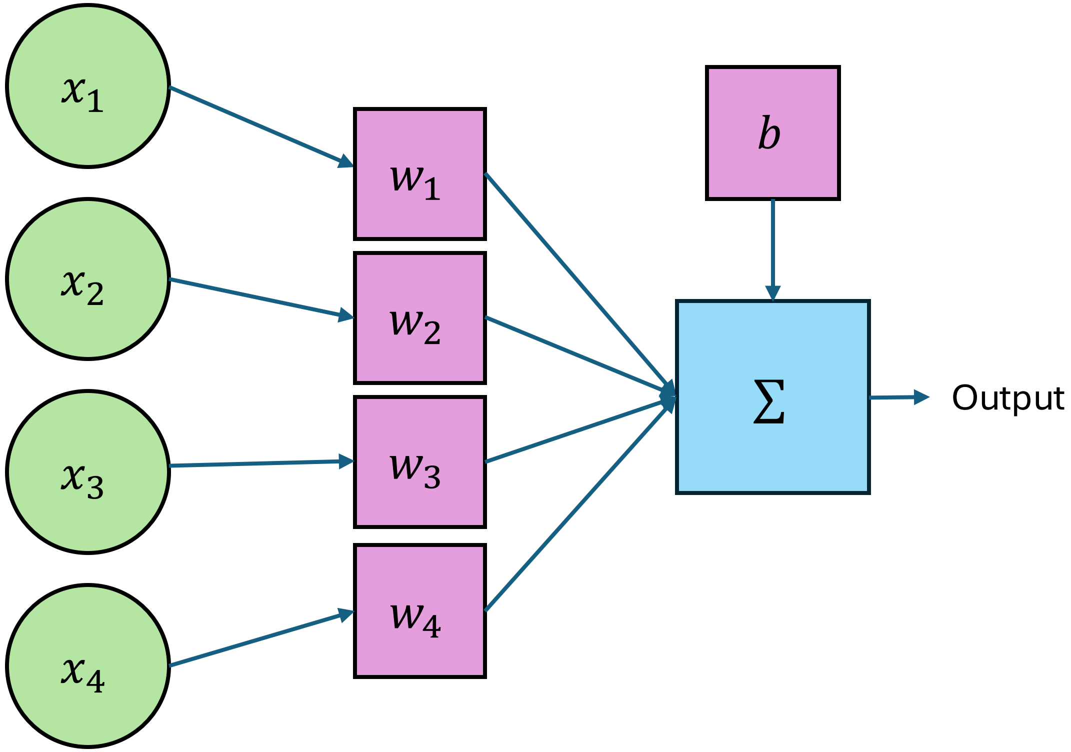
(1) Review Perceptrons and Apply to MNIST
(2) How do we train perceptrons?
(3) What are Perceptrons strengths and weaknesses?
(4) Multi-Layer Perceptrons (aka Neural Networks)

Understanding Perceptron Weights



Understanding Perceptron Weights

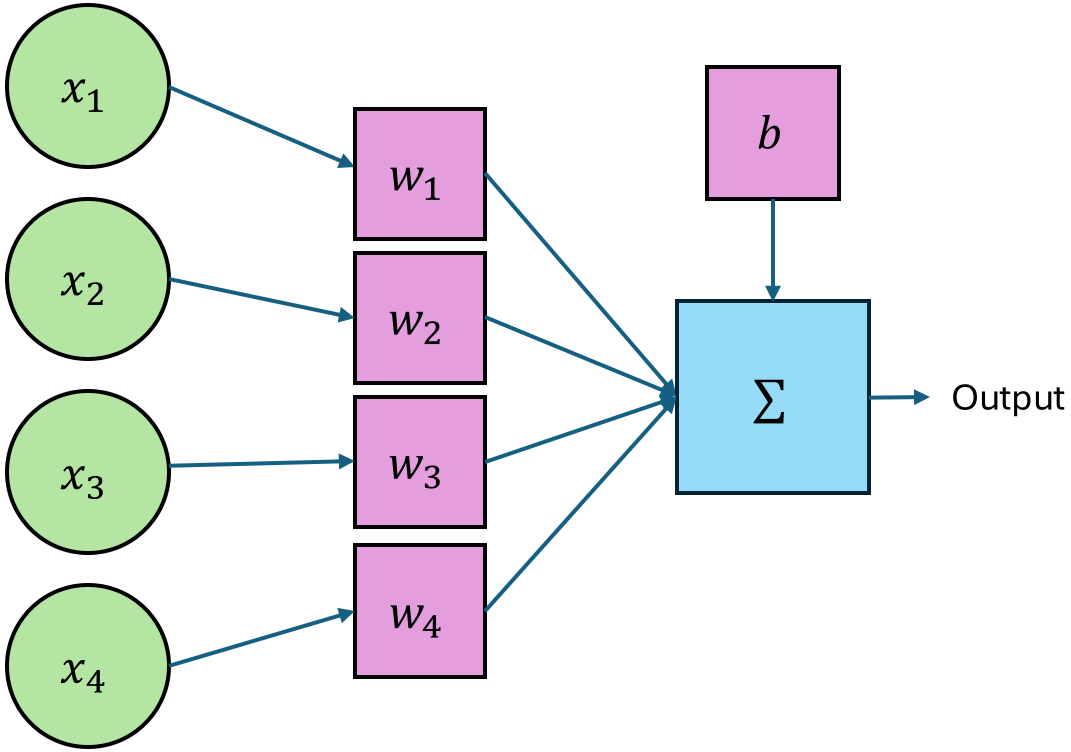
What would it mean for a weight to be 0?



Understanding Perceptron Weights

What would it mean for a weight to be 0?

What would it mean for a weight to be very positive?

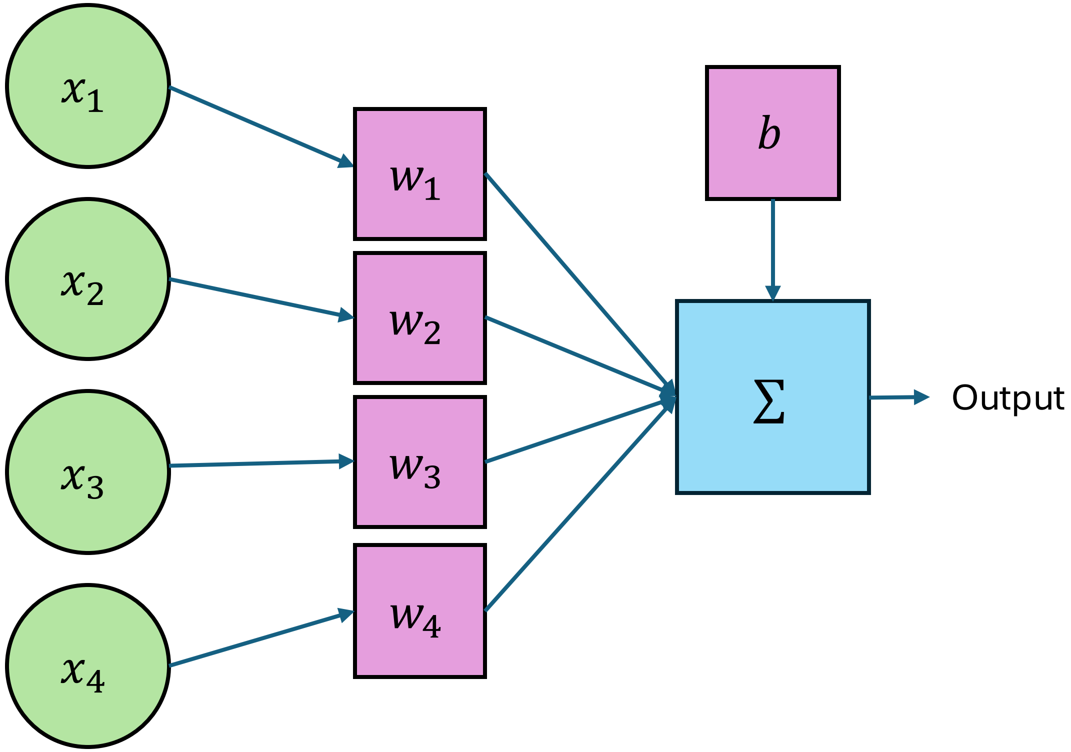


Understanding Perceptron Weights

What would it mean for a weight to be 0?

What would it mean for a weight to be very positive?

What would it mean for a weight to be very negative?

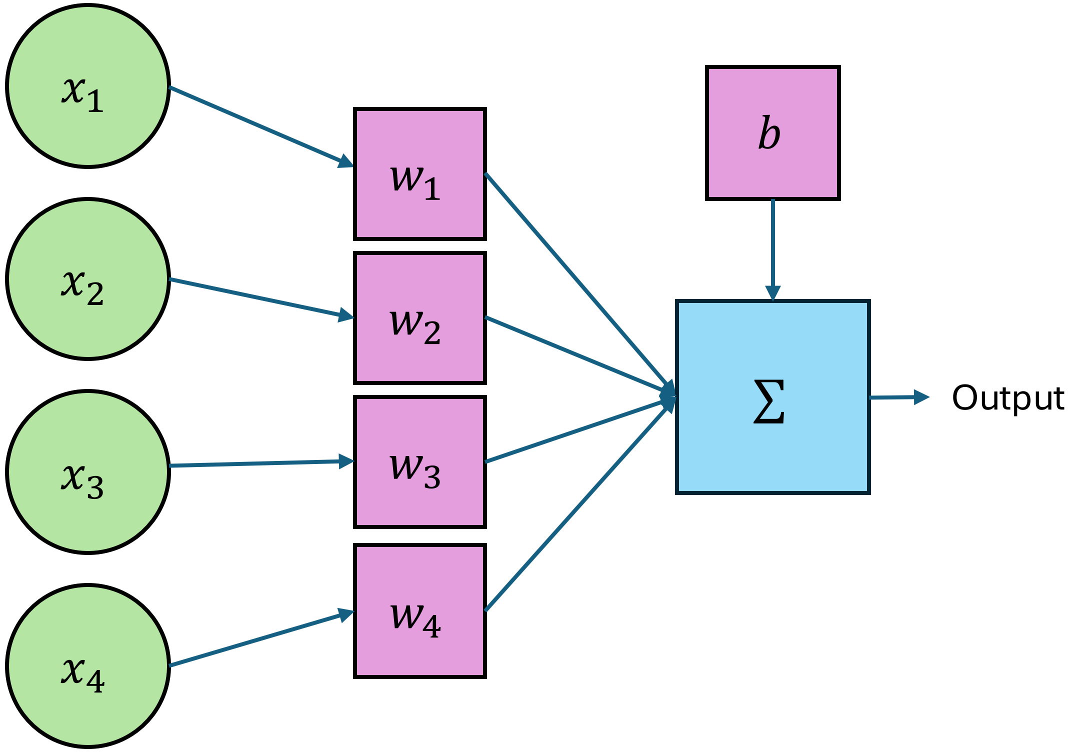


Understanding Perceptron Weights

What would it mean for a weight to be 0?

What would it mean for a weight to be very positive?

What would it mean for a weight to be very negative?



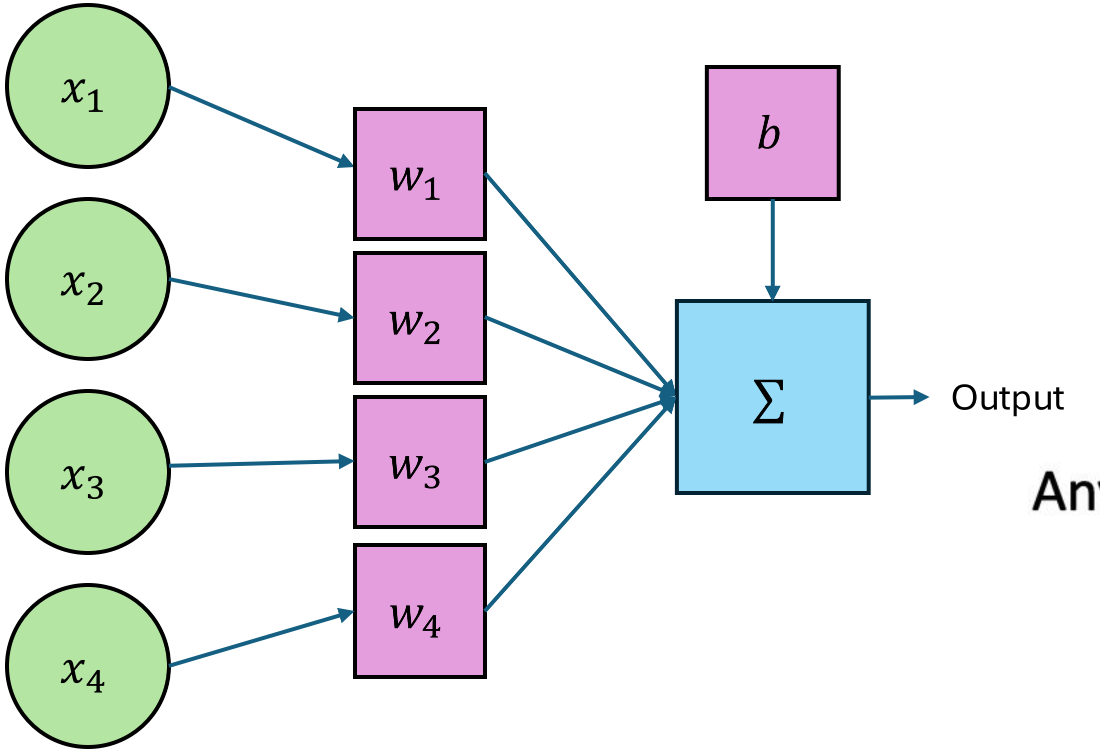
Input features should be in the same range! (e.g., should be between 0 and 1)

Understanding Perceptron Weights

What would it mean for a weight to be 0?

What would it mean for a weight to be very positive?

What would it mean for a weight to be very negative?



Any questions?

How Strong are Linear Separators?

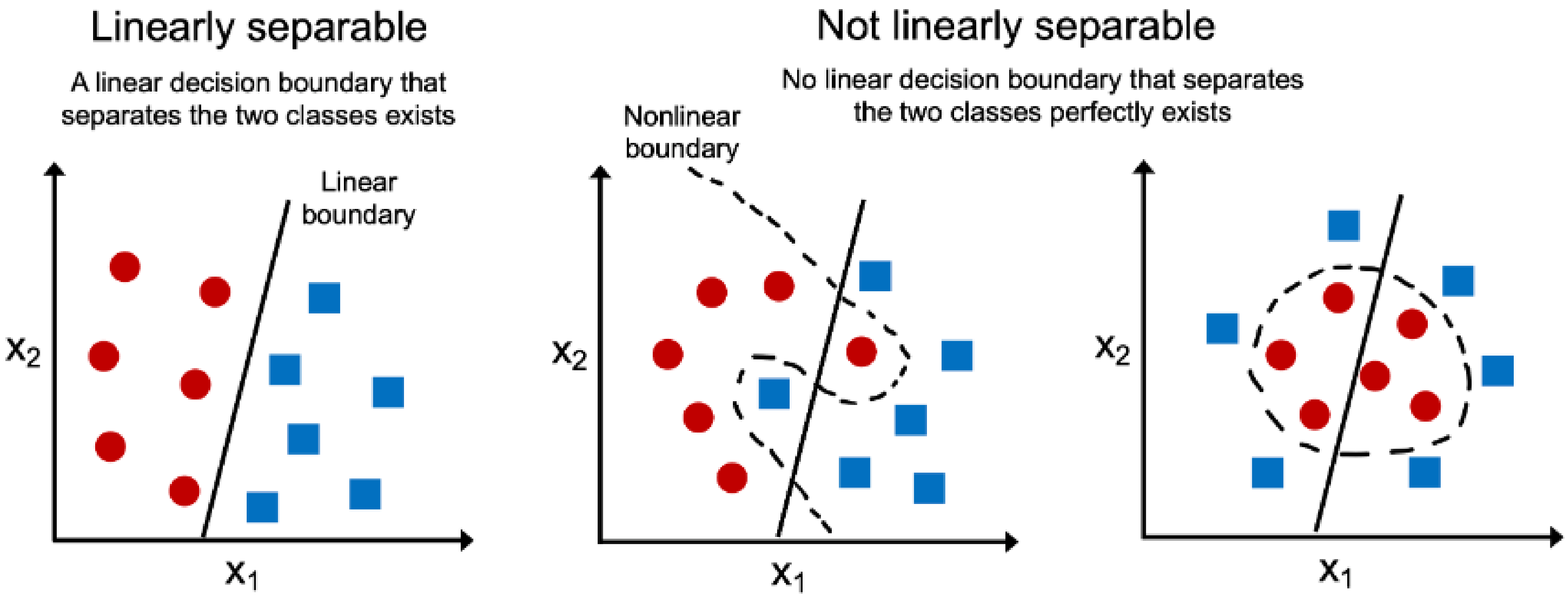


Image courtesy of: https://vitalflux.com/how-know-data-linear-non-linear/

MNIST

The most famous dataset in Deep Learning

Modified **N**ational **I**nstitute of **S**tandards and **T**echnology database

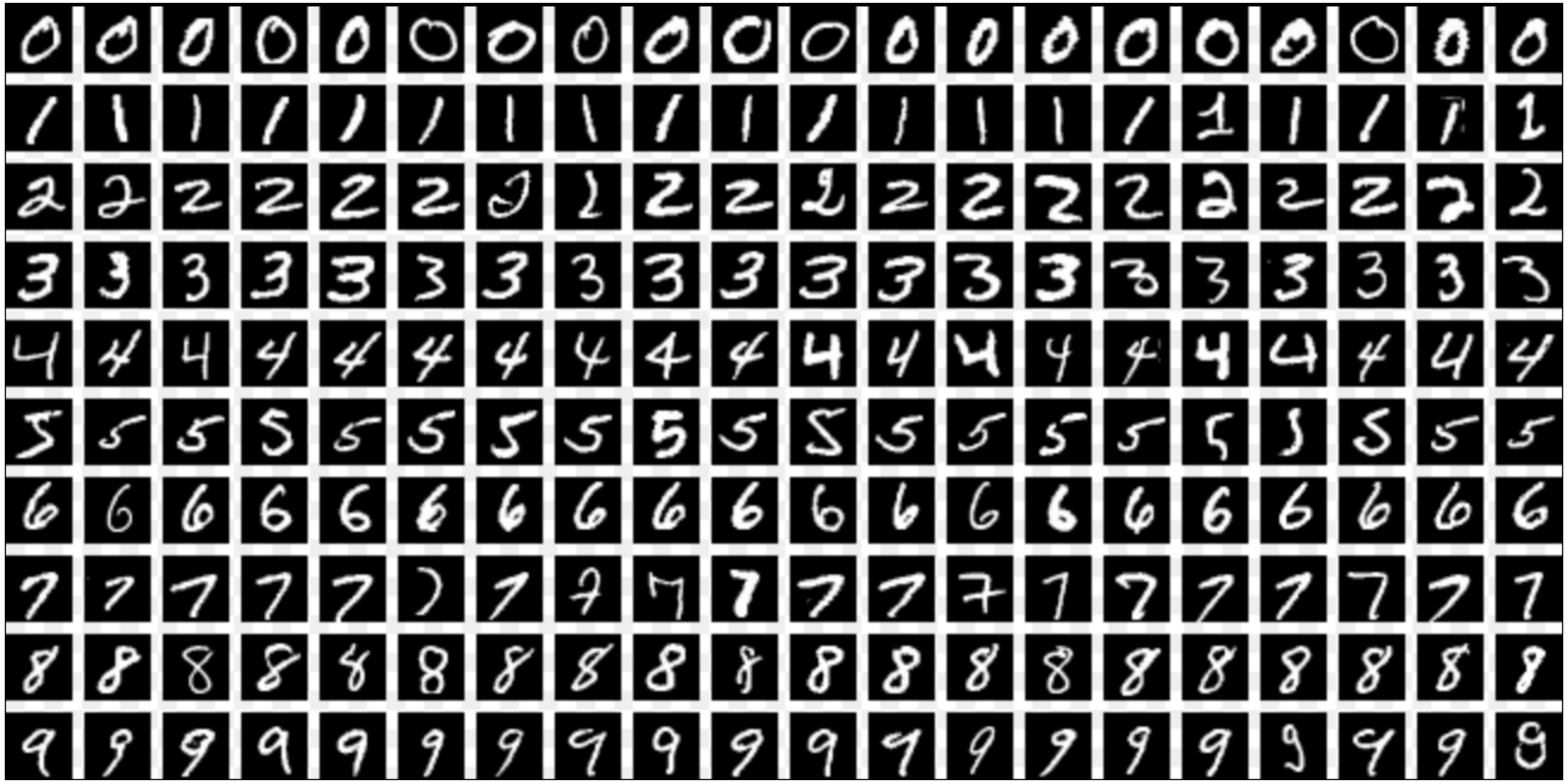
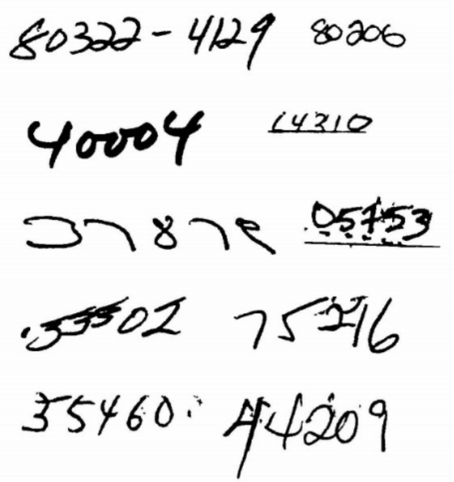


Image courtesy of Wikipedia

Motivation: Zip Code Recognition

- 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!



http://yann.lecun.com/exdb/publis/pdf/matan-92.pdf

Our Problem:

Input: $\mathbb{X}$

Function: f

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

Target: $\mathbb{Y}$

Which digit is it?

"3"

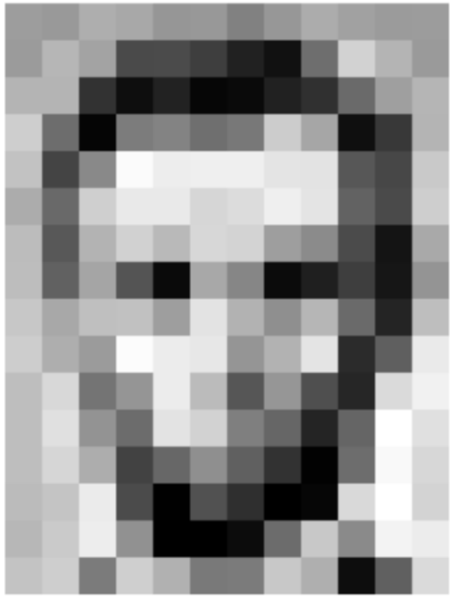
How does a computer know this is a three?

“three”

Representing digits in the computer

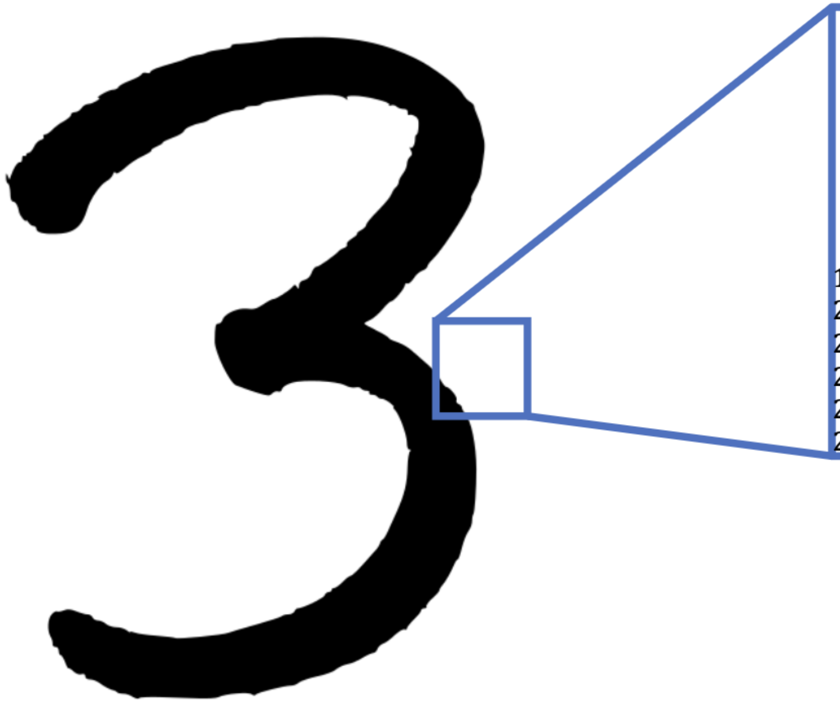
- Numbers known as ***pixel values*** (a grid of discrete values that make up an image)

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

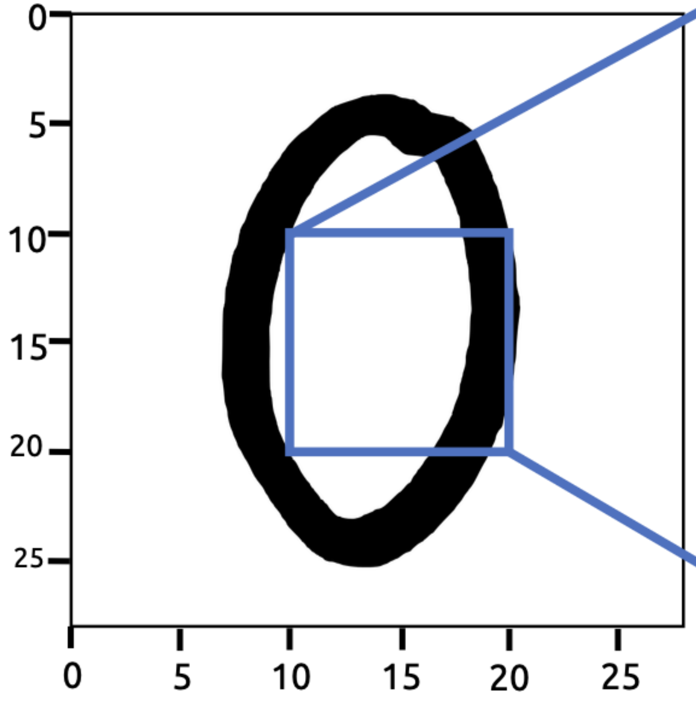
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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



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

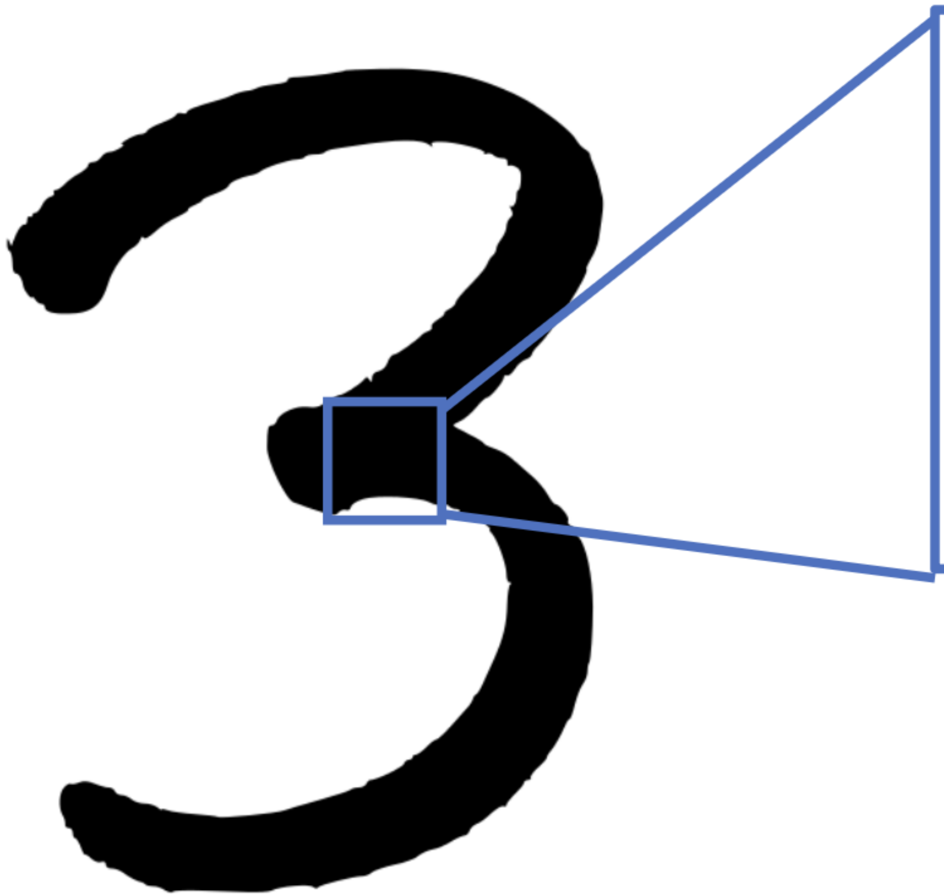
what the computer sees



	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

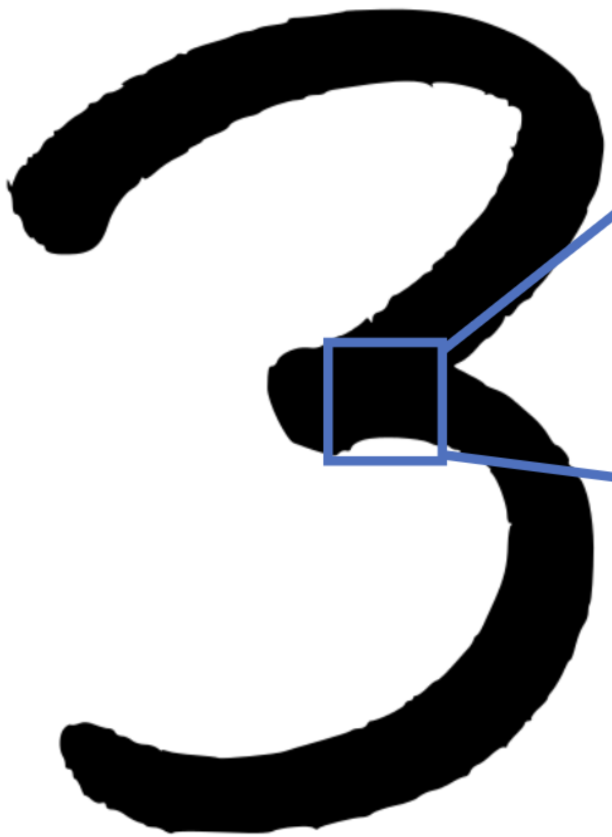
- Pixel in position [15, 15] is light.

Center is typically empty for 0's.
How does this compare with 3's?



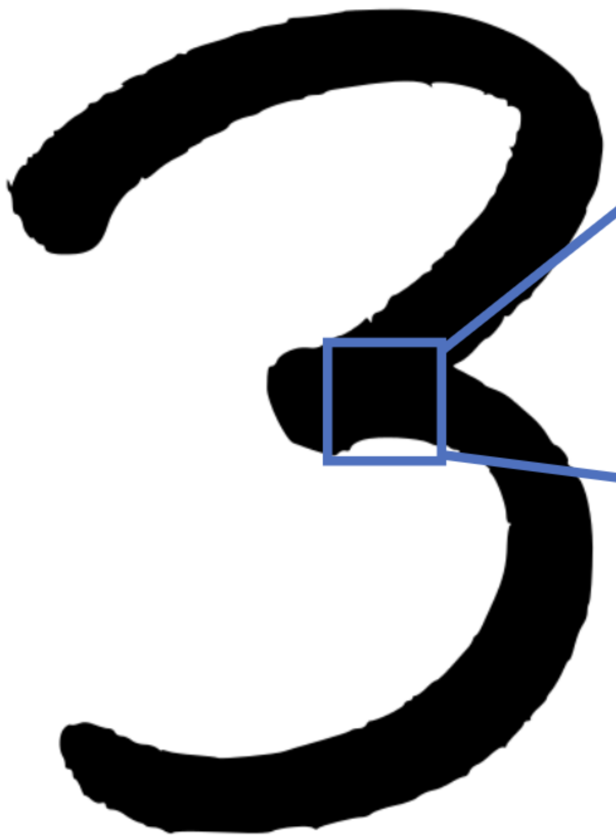
255	255	255	255	255	253	254	245	255
255	255	251	255	255	255	254	235	252
255	252	255	250	255	245	255	253	234
253	255	255	255	251	254	255	255	235
255	255	252	255	249	255	239	243	255
255	250	255	245	255	255	254	244	254
255	255	255	255	249	255	255	255	244
249	255	253	255	233	255	249	245	239
255	255	255	250	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

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

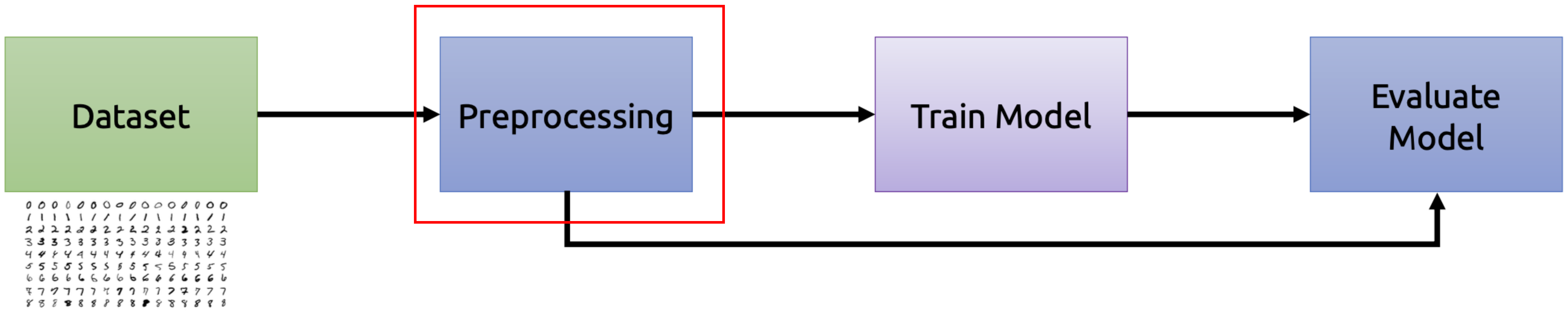
Darker pixels in the middle



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

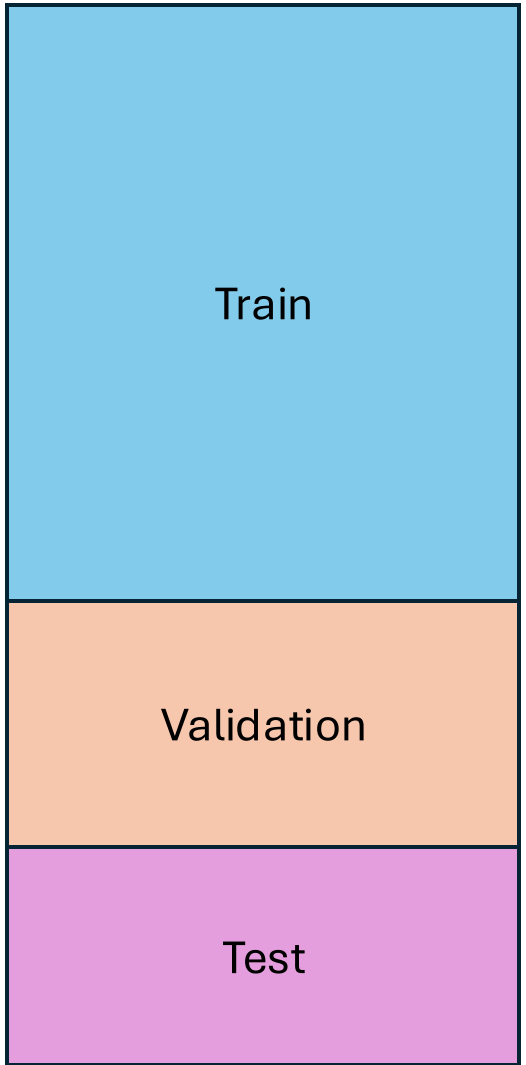
Can we define a set of ***heuristics*** (i.e. rules based on our intuition), to classify digits?

Machine Learning Pipeline for Digit Recognition



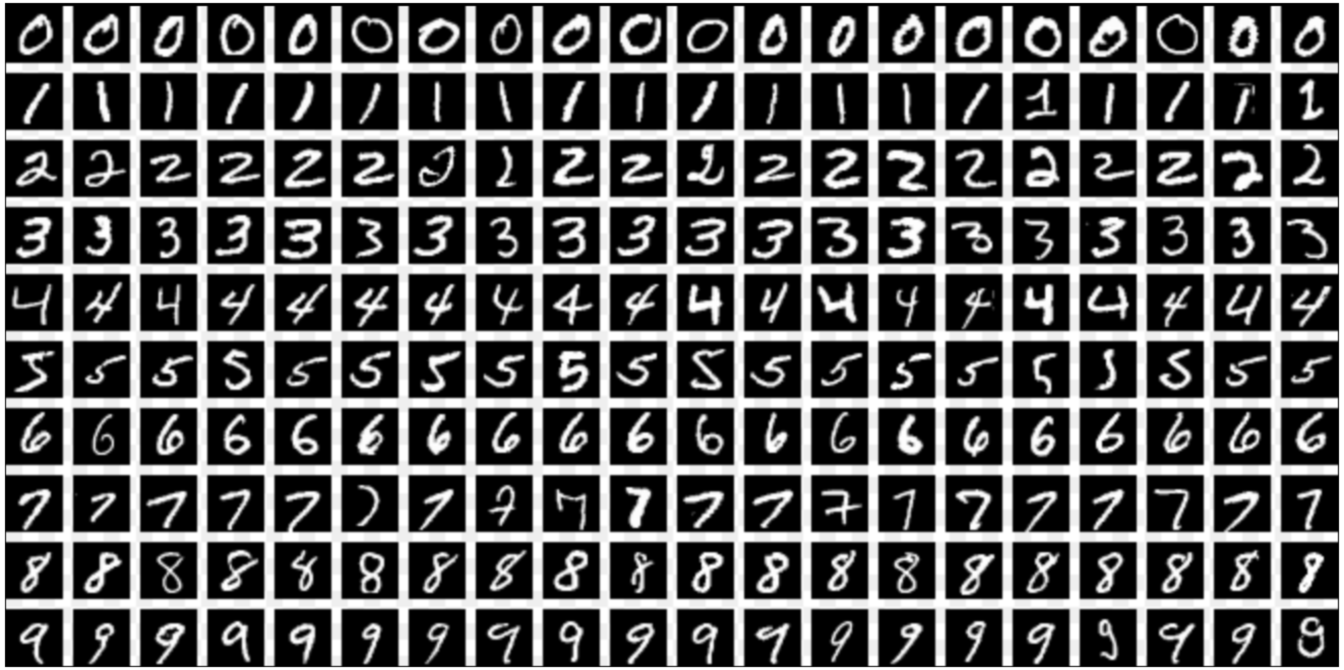
Train, validation, and test sets

- **Training Set**: Used to adjust parameters of 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

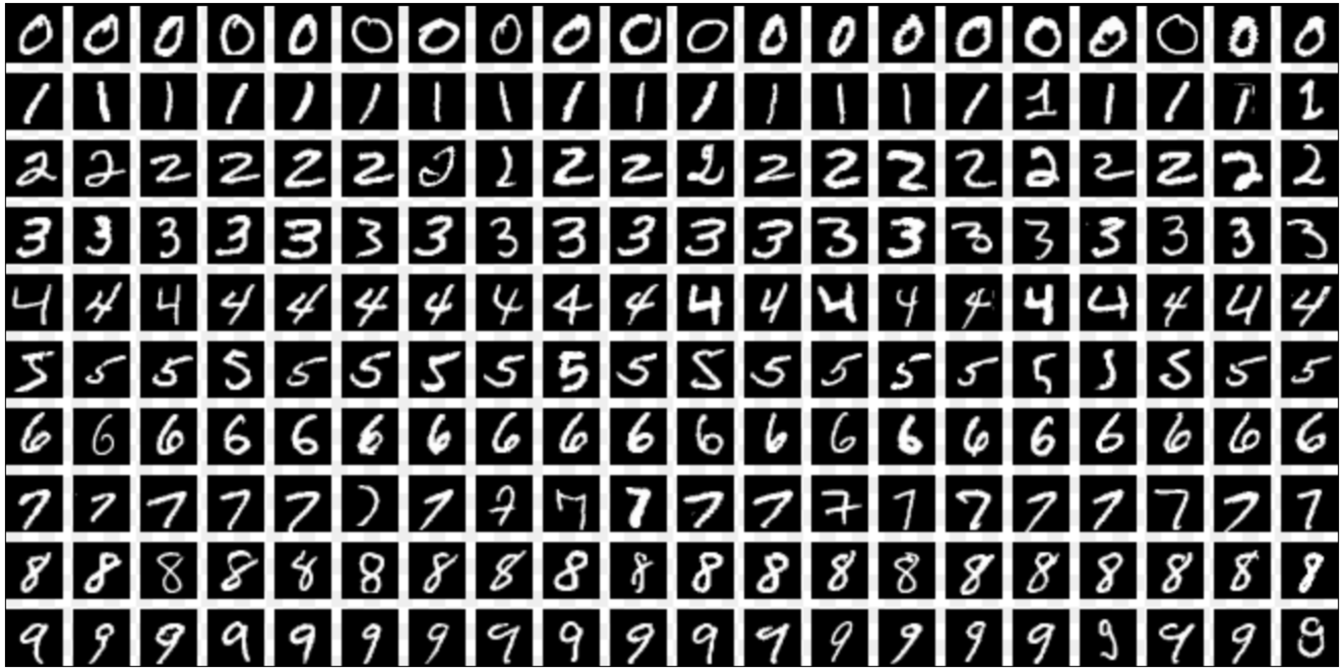
- 60,000 Images in training set
- 10,000 Images in test set
- No explicit validation set



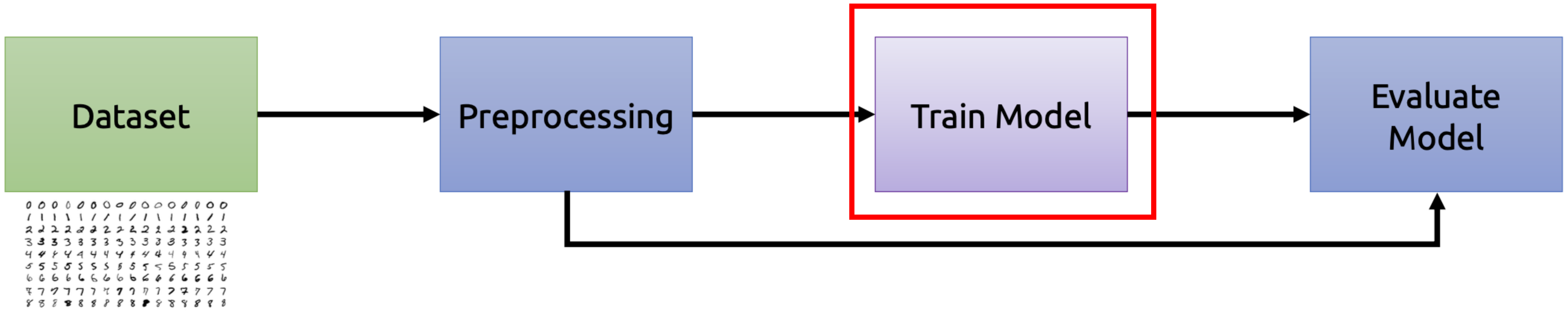
MNIST

- 60,000 Images in training set
- 10,000 Images in test set
- No explicit validation set

What do you suggest we do?



Machine Learning Pipeline for Digit Recognition



Our Problem:

Classifying MNIST digits requires predicting
1 of 10 possible values

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

$x^{(1)} =$ 

28x28 pixels

Function: f

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

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

$x^{(2)} =$ 

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

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

What is our input space?

Target: $\mathbb{Y}$

Pixel Grid

$x^{(1)} =$

28x28 pixels

Function: f

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

Which digit is it?

$y^{(1)} = \text{“2”}$

$x^{(2)} =$

$y^{(2)} = \text{“0”}$

Our Problem:

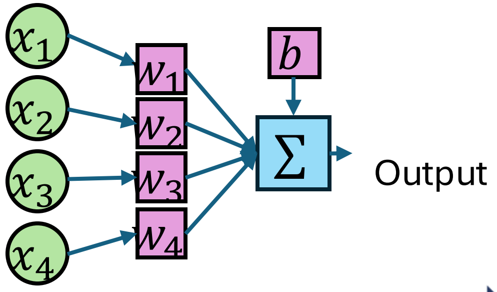
Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

What is our input space?

Target: $\mathbb{Y}$

Pixel Grid



Which digit is it?

$x^{(1)} =$ 

Function: f

$y^{(1)} = \text{“2”}$

28x28 pixels

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

$x^{(2)} =$ 

$y^{(2)} = \text{“0”}$

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

Which digit is it?

$x^{(1)} =$ 

28x28 pixels

Function: f

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

$y^{(1)} =$ “2”

$x^{(2)} =$ 

$y^{(2)} =$ “0”

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

$x^{(1)} =$ 

28x28 pixels

$x^{(2)} =$ 

Function: f

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

Is it digit 2?

$y^{(1)} = 1$ 

$y^{(2)} = 0$ 

The Perceptron Algorithm

Loop Over Dataset (until no weights change)

- For each misclassified example
  - update weights to make better prediction for example

The Perceptron Algorithm

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not change
    1. For each example $x^{(k)}$ with label $y^{(k)}$
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

1. $\boxed{\text{Initialize } \vec{\theta} = \vec{0}}$

Need to start somewhere…
any initial setting will work

2. For N iterations or until $\vec{\theta}$ does not change
    1. For each example $x^{(k)}$ with label $y^{(k)}$
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

N is referred to as **"epochs"**: Number of times the entire dataset is iterated through

1. Initialize $\vec{\theta} = \vec{0}$
2. $\boxed{\text{For N iterations}}$ or until $\vec{\theta}$ does not change
    1. For each example $x^{(k)}$ with label $y^{(k)}$
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not change
    1. $\boxed{\text{For each example } x^{(k)} \text{ with label } y^{(k)}}$ Loop over every example in dataset
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not change
    1. For each example $x^{(k)}$ with label $y^{(k)}$
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + (y^{(k)} - f(x^{(k)})) \cdot x_i^{(k)}$

Look only at examples that are misclassified (i.e., $y^{(k)} \neq f(x^{(k)})$)

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

For every parameter in our perceptron…

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not change
    1. For each example $x^{(k)}$ with label $y^{(k)}$
        1. If $y^{(k)} = f(x^{(k)})$, continue
        2. Else, for all parameters $\theta_i \in \vec{\theta}$ $\boxed{\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

For every parameter in our perceptron…

If $y^{(k)} = 1$ and $f\left(x^{(k)}\right) = 0$ and $x_i^{(k)}$>0…

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not chang
   1. For each example $x^{(k)}$ with label $y^{(k)}$
      1. If $y^{(k)} = f(x^{(k)})$, continue
      2. Else, for all parameters $\theta_i \in \vec{\theta}$ $\boxed{\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not chang
   1. For each example $x^{(k)}$ with label $y^{(k)}$
      1. If $y^{(k)} = f(x^{(k)})$, continue
      2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\boxed{\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}}$

For every parameter in our perceptron…

If $y^{(k)} = 1$ and $f\left(x^{(k)}\right) = 0$ and $x_i^{(k)}$>0…

θ_i increases

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

The Perceptron Algorithm

If no parameters change, then we know that $y^{(k)} = f\left(x^{(k)}\right) \forall k$

1. Initialize $\vec{\theta} = \vec{0}$
2. For N iterations or until $\vec{\theta}$ does not change
   1. For each example $x^{(k)}$ with label $y^{(k)}$
      1. If $y^{(k)} = f(x^{(k)})$, continue
      2. Else, for all parameters $\theta_i \in \vec{\theta}$, $\theta_i = \theta_i + \left(y^{(k)} - f\left(x^{(k)}\right)\right) \cdot x_i^{(k)}$

w: weights
b: bias
θ: parameters (weights and biases), $\vec{\theta} = \{\vec{w} \cup b\}$

$x^{(k)}$: k'th training example, $y^{(k)}$ k'th training label
$x_i^{(k)}$: i'th feature for k'th example

Converting Perceptrons to Multi-Class Classification

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

$x^{(1)} =$ 

28x28 pixels

Function: f

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

Which digit is it?

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

$x^{(2)} =$ 

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

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

How do we do this?

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

$x^{(1)} =$ 

28x28 pixels

Function: f

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

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

$x^{(2)} =$ 

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

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

How do we do this?

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

$x^{(1)} =$

Function: f

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

28x28 pixels

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

$x^{(2)} =$

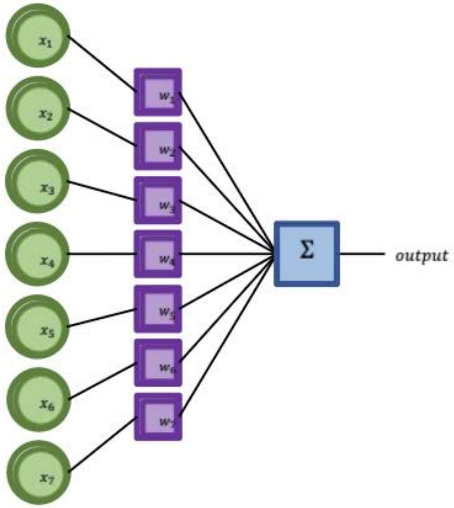
Instead of outputting a binary prediction, make an output for each class.

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

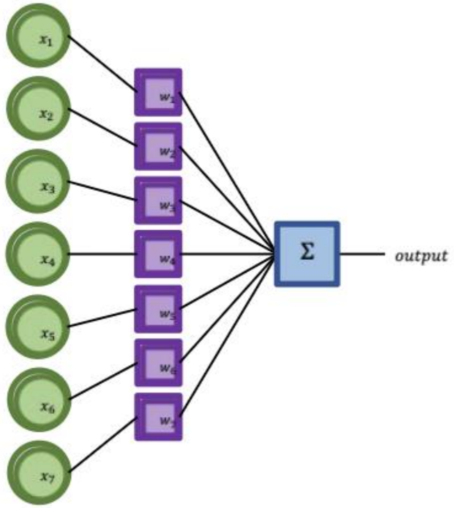
Using Multiple Perceptrons

- We can use *m* perceptrons (where *m* is the number of output classes)
- For MNIST, this would be 10 perceptrons
- Each individual perceptron will need to return a value, our model will return the class with the highest value
  - Here, value refers to the weighted sum before the threshold is applied

Using multiple perceptrons

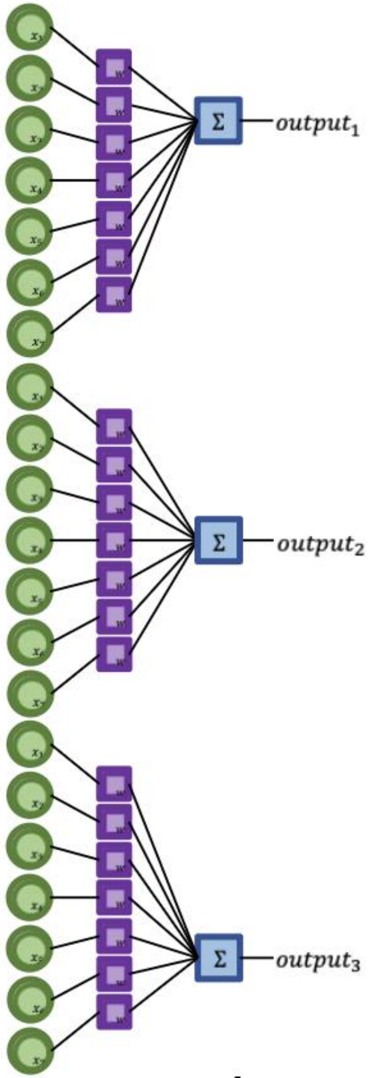


Perceptron for predicting whether handwritten digit is a 0



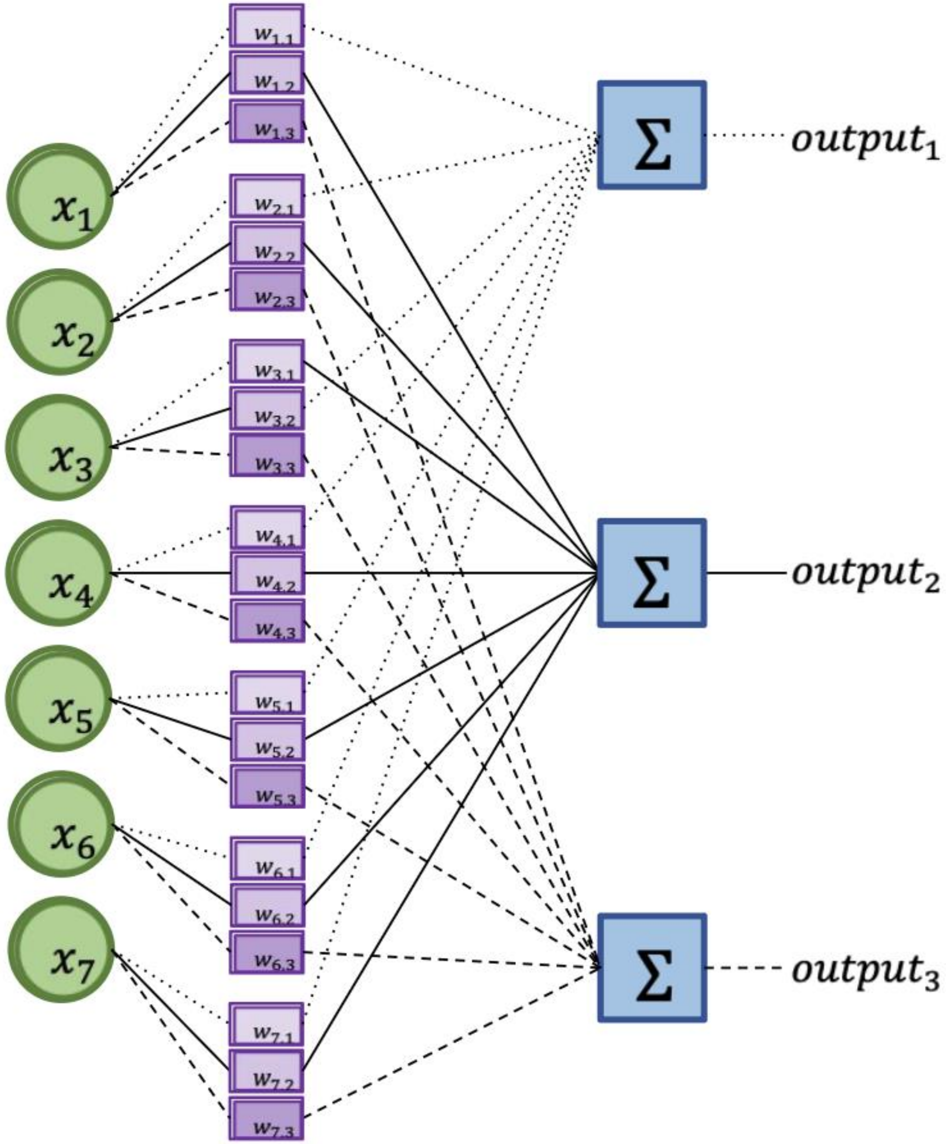
Perceptron for predicting whether handwritten digit is a 9

Multi-class Perceptron



Three separate perceptrons

=



Three perceptrons sharing inputs

So how well does this do?

Perceptrons achieve ~85% accuracy on MNIST

Is this good?

So how well does this do?

Perceptrons achieve ~85% accuracy on MNIST

Is this good?

Can be coded in ~20 minutes, probably achieves better accuracy than whatever else you can do in ~20 minutes...

So how well does this do?

Perceptrons achieve ~85% accuracy on MNIST

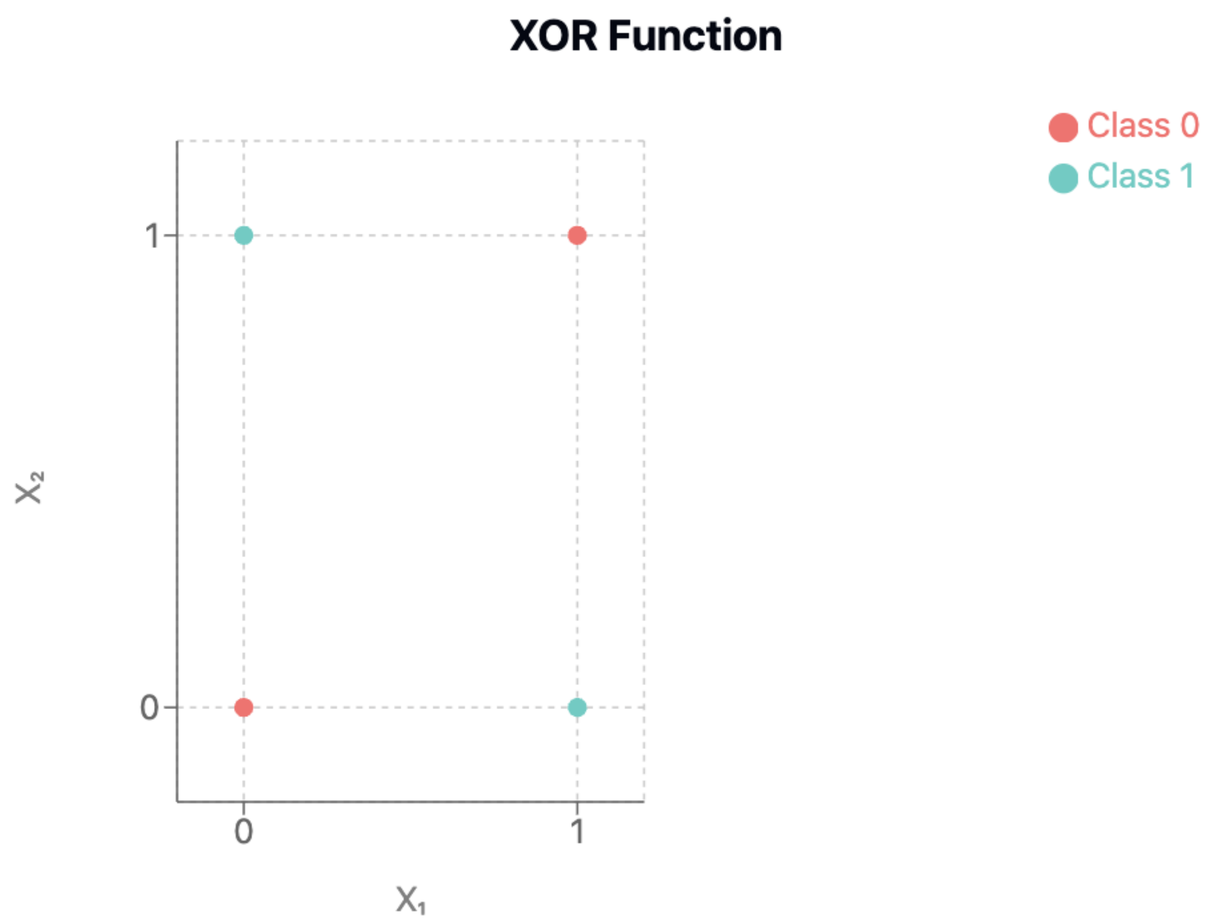
Is this good?

Can be coded in ~20 minutes, probably achieves better accuracy than whatever else you can do in ~20 minutes...

But 85% is not good enough for the post office

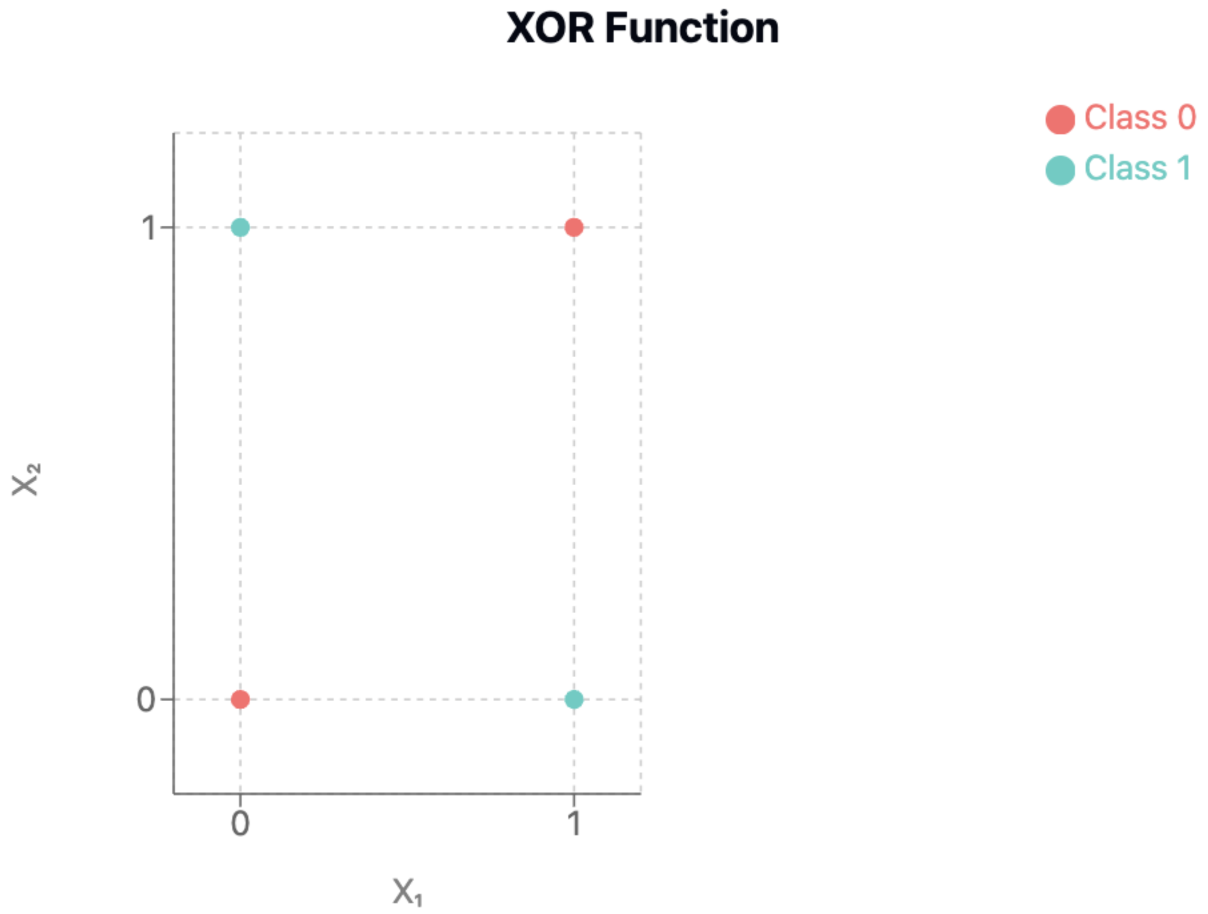
Perceptrons

Are Perceptrons guaranteed to achieve 100% accuracy?



Perceptrons

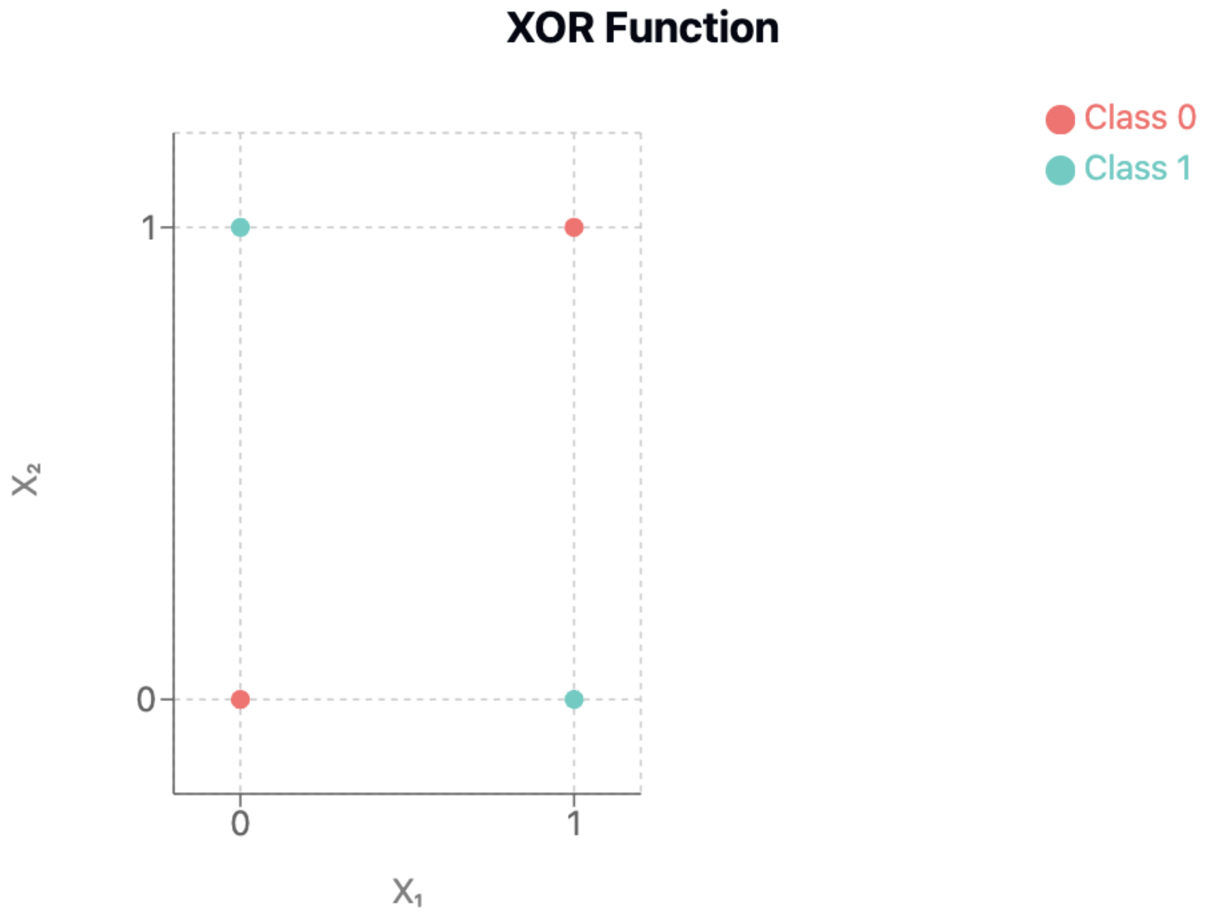
Are Perceptrons guaranteed to achieve 100% accuracy?



Perceptrons

Are Perceptrons guaranteed to achieve 100% accuracy?

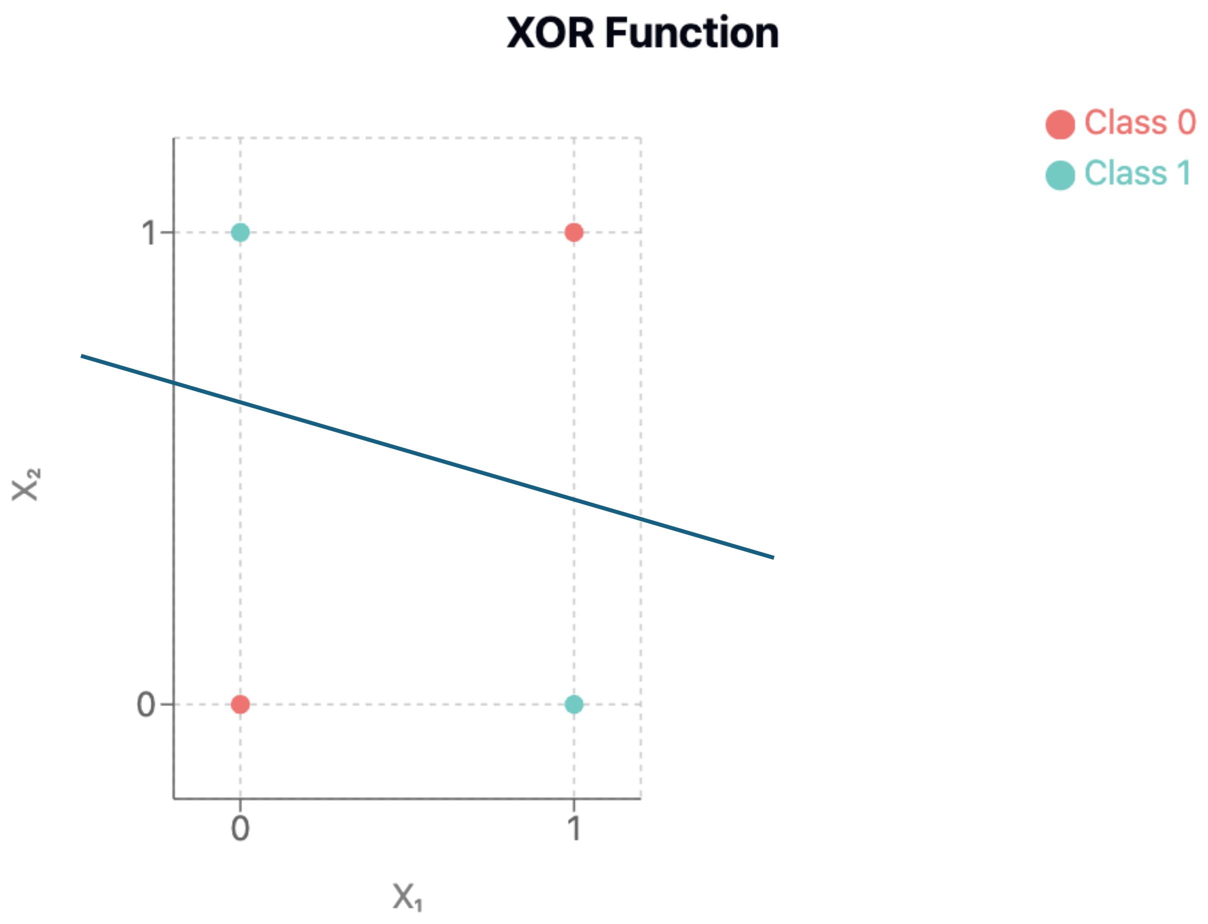
How can you put a linear separator on the plot to separate the two classes?



Perceptrons

Are Perceptrons guaranteed to achieve 100% accuracy?

How can you put a linear separator on the plot to separate the two classes?

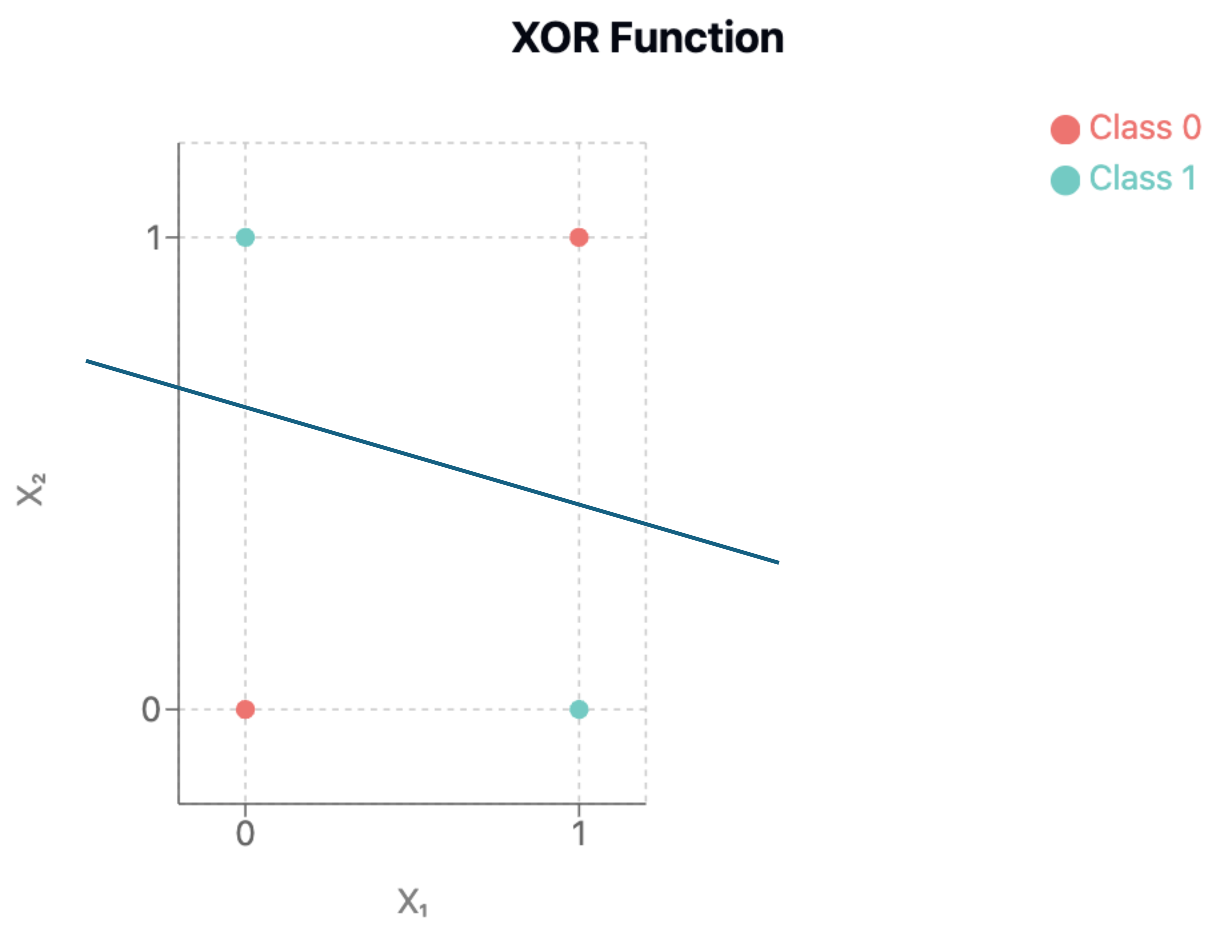


Perceptrons

Are Perceptrons guaranteed to achieve 100% accuracy?

How can you put a linear separator on the plot to separate the two classes?

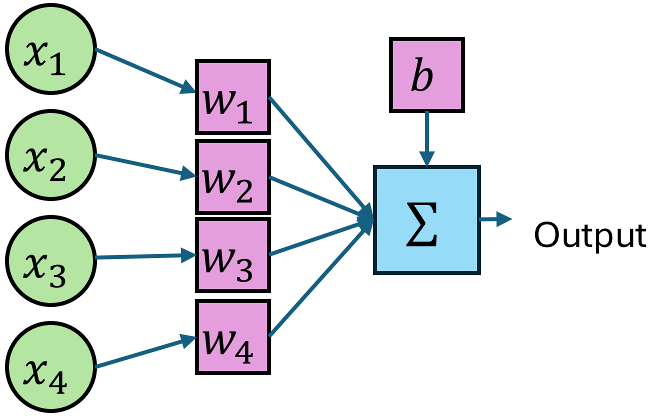
There are simple functions that perceptrons can't learn!

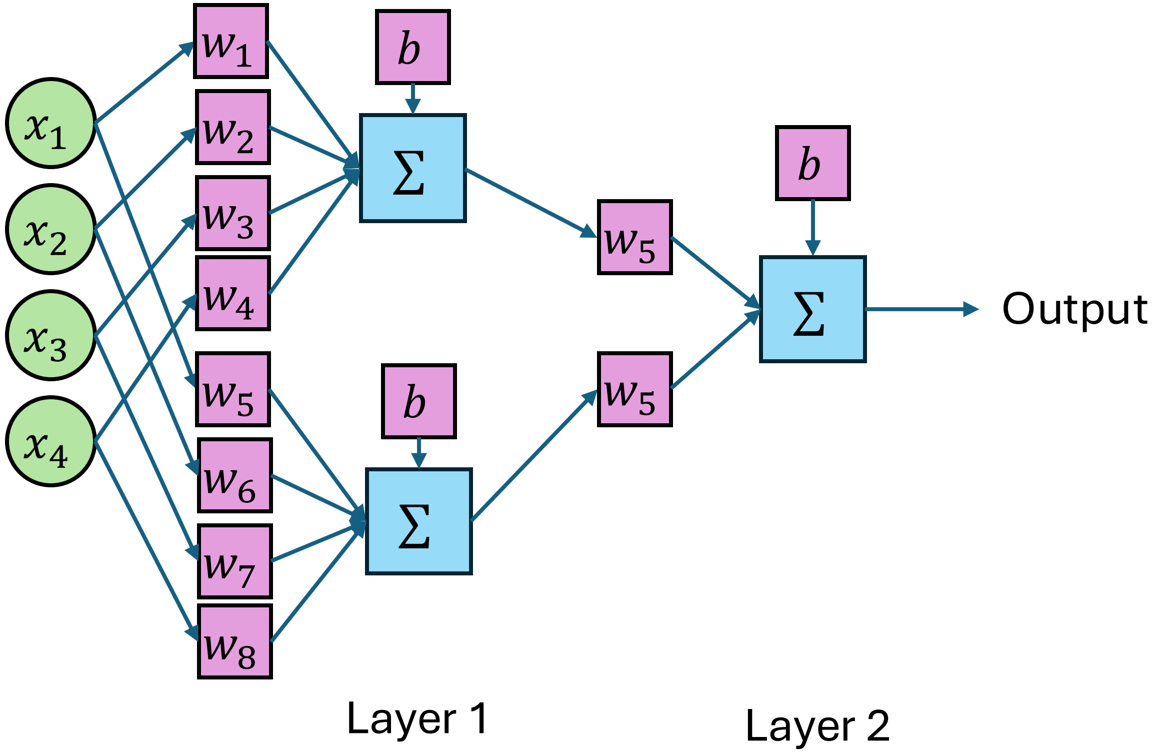


The Solution:

Multi-Layer Perceptron (MLP, Neural Network)

Perceptron





Perceptrons: Marvin Minsky and Seymor Papert

- Published in 1969 and very pessimistic of “connectionism”
- Limited funding for neural networks research in the 1970s
  - (First AI winter)
- 1980s – revival of neural networks research
  - “Invention” of backpropagation, needed for efficient training of neural networks
- 1987 – collapse of LISP machine market and abandonment of expert systems
  - (Second AI winter)

Remaining Questions (for next time)

- We trained perceptrons with a special algorithm for binary classification. How does that change when we have multiple outputs or multiple layers?
- Multi-layer perceptrons can achieve better performance on MNIST and can work with non-linear separable data. Is there anything they can't learn?

Recap

First weekly quiz is up on Gradescope (as of 12:40pm), and due in 24 hours (1pm Thursday)

MNIST image data is a testbed for multi-class classification

Perceptrons have binary outputs (0 or 1), how do we do multi-class classification?

With a different perceptron output for each class! (and using continuous output value not binary)

Perceptrons can only correctly classify linearly separable data, how can we learn more complex functions?

Using multi-layer perceptrons (MLPs)!