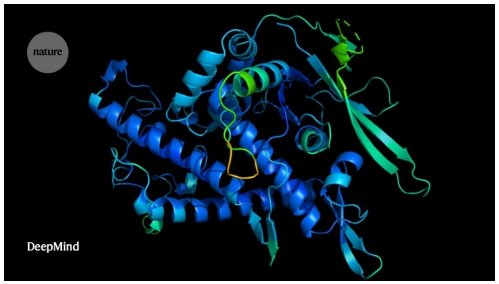
DNN Introduction 1

Instructor: Xuechen Zhang

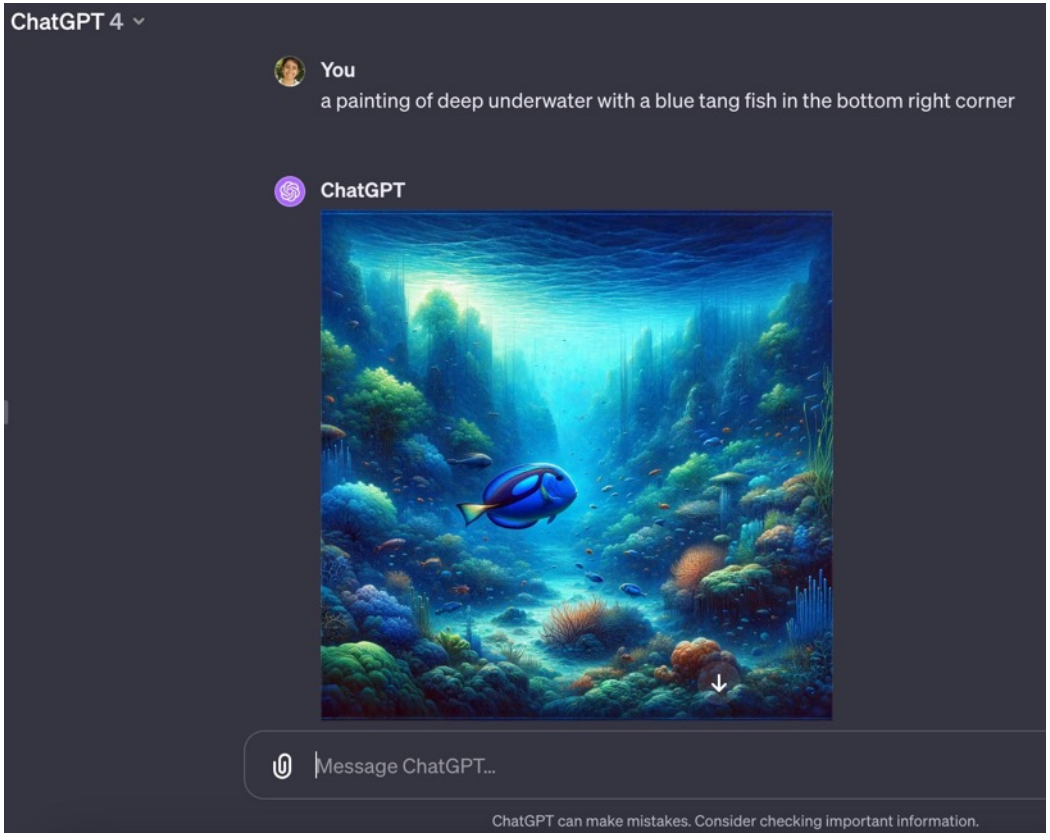
Washington State University Vancouver

Slides adapted from R. Singh@Brown and K. Wong@UTK

You may have heard of "Deep Learning" or "Artificial Intelligence (AI)"



You may have heard of "Deep Learning" or "Artificial Intelligence (AI)"



You may have heard of "Deep Learning" or "Artificial Intelligence (AI)"

Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** June 6, 2019

In review of fatal Arizona crash, U.S. agency says Uber software had flaws

By David Shepardson 4 MIN READ

WASHINGTON (Reuters) - An Uber self-driving test vehicle that struck and killed an Arizona woman in 2018 had software flaws, the National Transportation Safety Board said Tuesday as it disclosed the company's autonomous test vehicles were involved in 37 crashes over the prior 18 months.

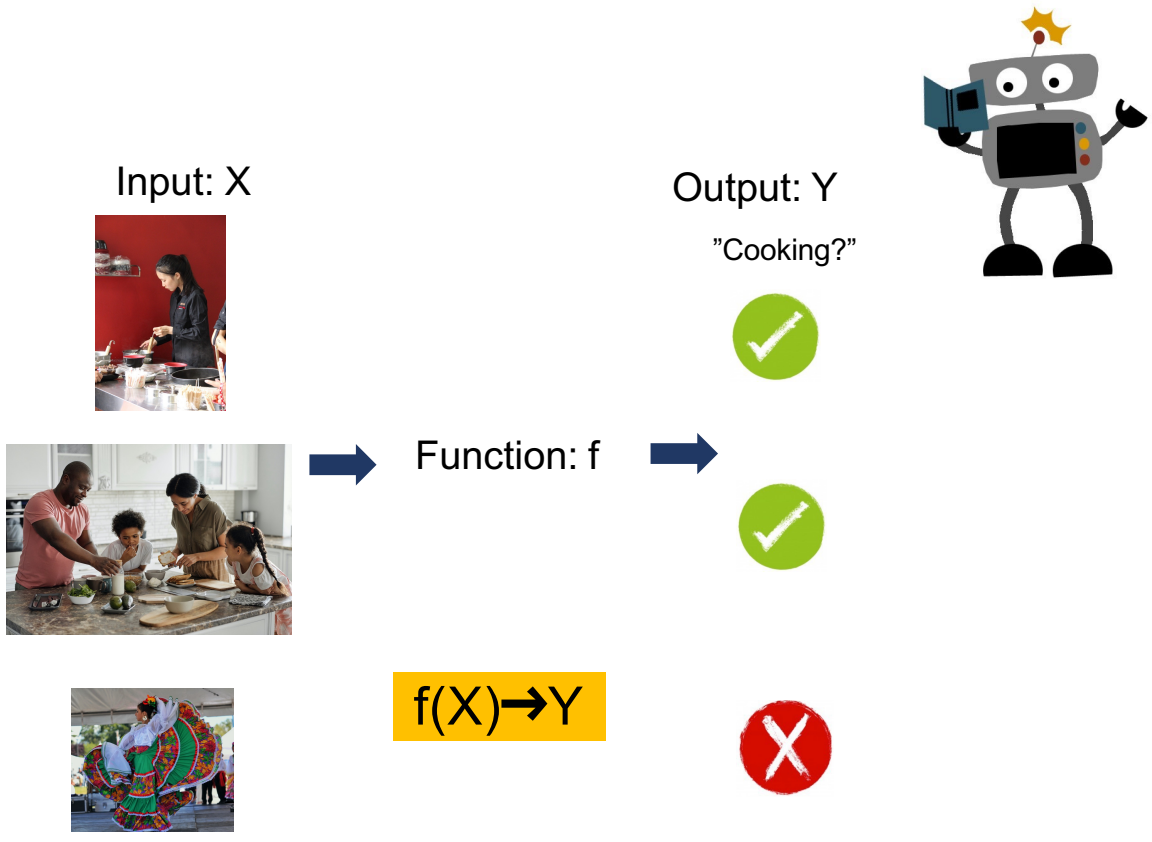
Outline

- **What is deep learning?**
- Perceptron and parameters
- Perceptron learning algorithm and loss function

Next time when you come across “Deep Learning” you will know:

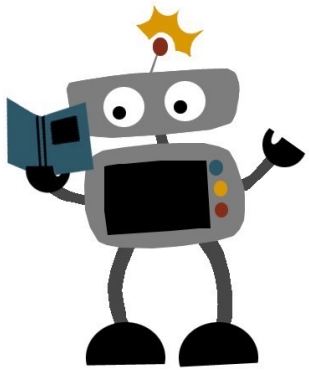
- What is Machine Learning?
- How does it connect to Deep Learning?
- What is Deep Learning?
- What is NOT Deep Learning?

What is Machine Learning?



- A field of study that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959). A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

What is Machine Learning?



Supervised Learning

Input: X

Learned function: f

Output: Y

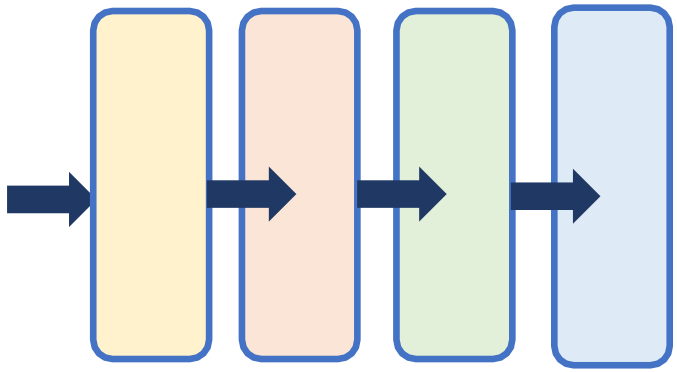
"Cooking?"

f(X)→Y

- A class in ML, dataset has labeled values, use to predict output values associated with new input values.

What is Deep Learning?

Input: X

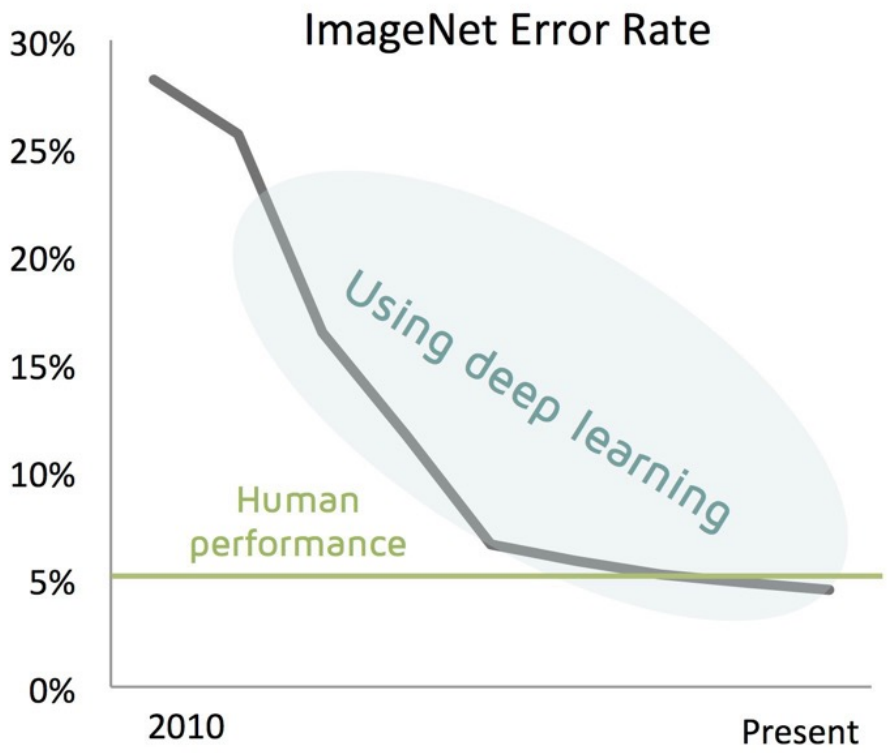


Output: Y

"Cooking?"

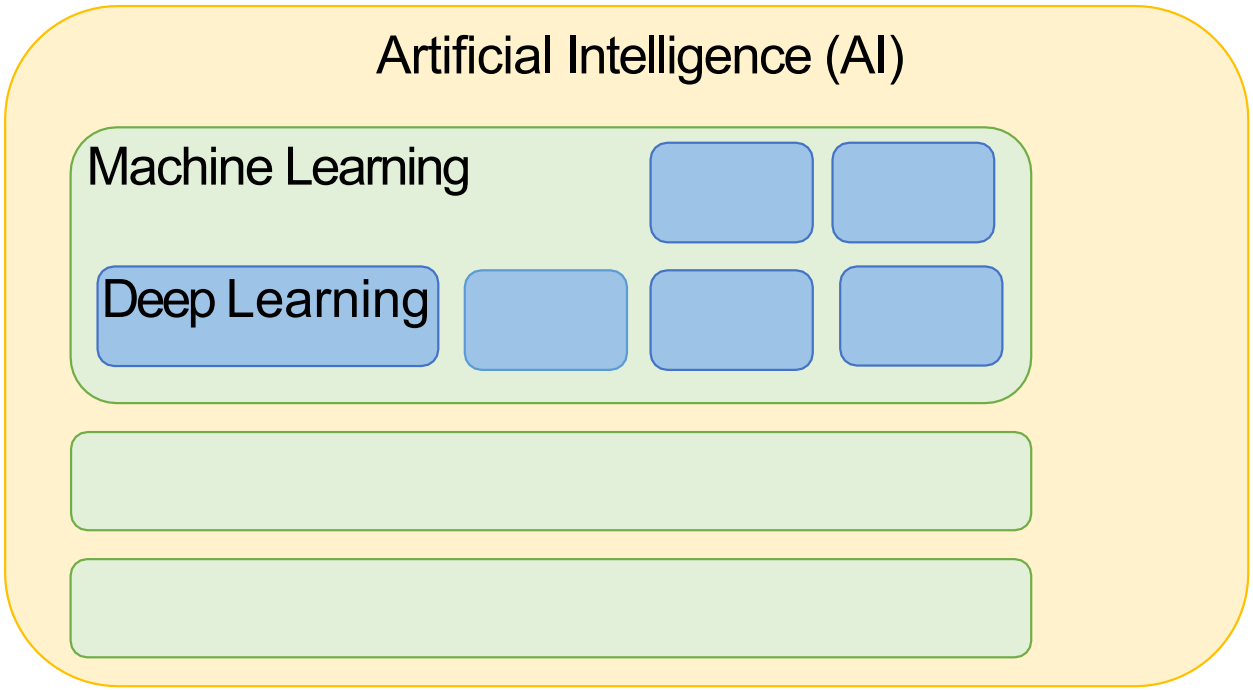
$f_4\ (f_3\ (f_2\ (f_1\ (X)))) \rightarrow Y$

What is Deep Learning?



What is NOT Deep Learning?

Deep Learning is NOT AI



Learn about some basic concepts of machine learning

- How do we represent input/output?
- Learning the function f
- Training a machine learning model
- Learning good models

How do we represent input/output?

Machines work with numbers!

How can we represent input image as numbers?

Input: X

"Model"

Function: f

f(X)→Y

Output: Y

"Cooking?"

How can we represent output labels as numbers?

How do we represent input/output?

Classification

Binary classification

Input: X

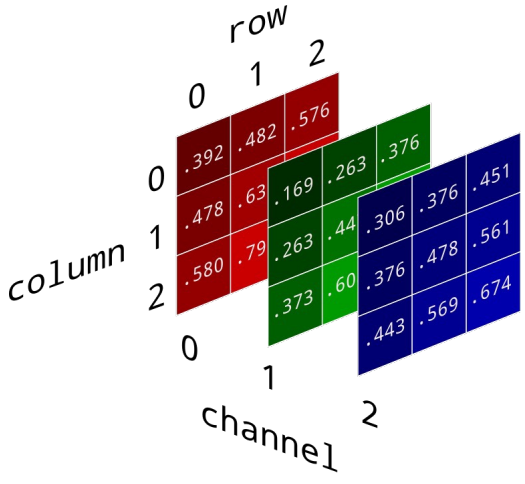
Output: Y

"Cooking?"

Complicated input!

Let's use a simpler example!

$X \in \mathbb{R}^{H*W*3}$



$Y \in \{0,1\}$

(Categorical output)

1 or 0

But first some notations ...

- X: A set of input data
- Y: Associated set of target values (outputs) for supervised learning
- $x^{(k)}$: kth example (input) from a dataset
- $y^{(k)}$: Target (output) associated with $x^{(k)}$ for supervised learning
- R: A set of real numbers

Simpler example: How do we represent input/output?

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

$x^{(1)}$ 100.1 °F

$x^{(2)}$ 80.0 °F

$x^{(3)}$ 30.3 °F

Regression

Function: f

f(X)→Y

Do you see a trend here?

What is different about the output here?

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)}$ $200.0

$y^{(2)}$ $180.5

$y^{(3)}$ $115.1

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

(Numerical output)

Learning function f

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

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

Regression

Function: f

f(X)➔Y

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning function f

Have you seen this equation before?

Regression

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

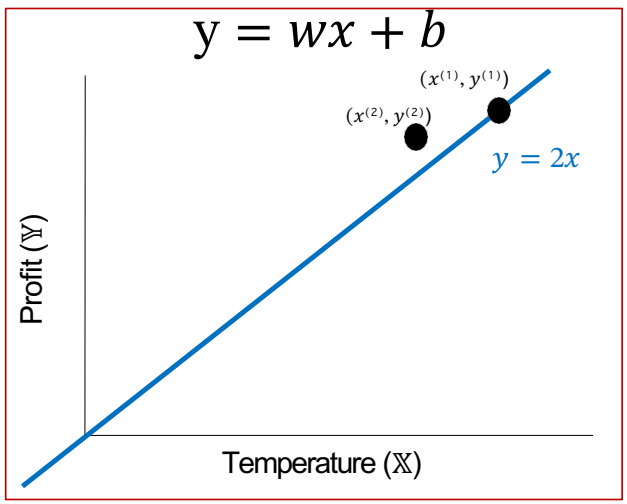
$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

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

Linear function

$y = wx + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning function f

Input: $\mathbb{X}$

"Temperature"

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

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

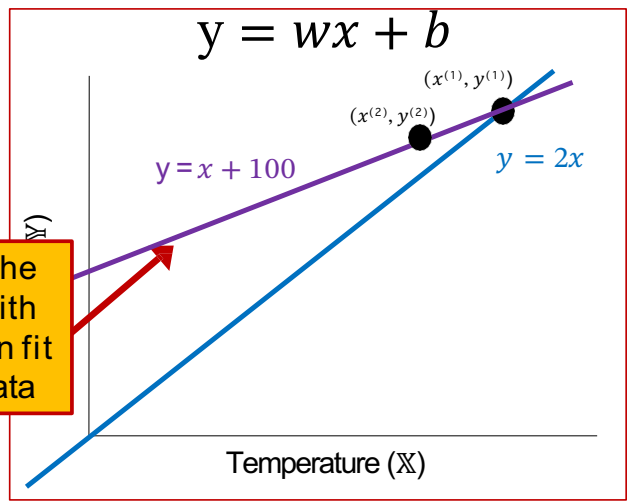
$x^{(3)} =$

Only the line with bias can fit the data

Regression

Linear function

$y = wx + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning function f

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

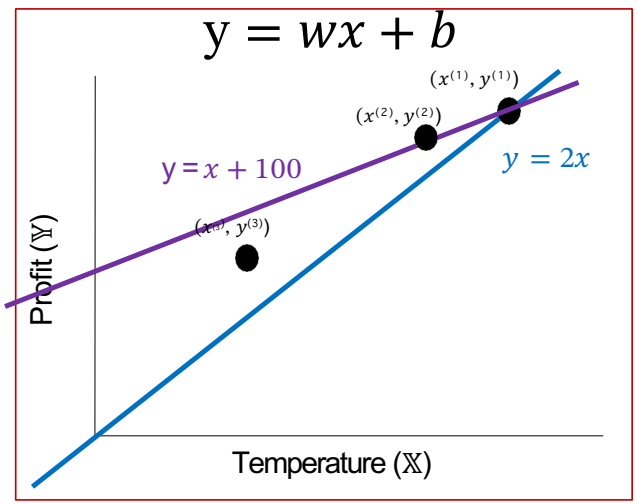
$x^{(3)} = 30.3$

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

Regression

Linear function

$y = wx + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning function f

Input: $\mathbb{X}$

"Temperature"

Regression

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$x^{(1)} = 100.1$

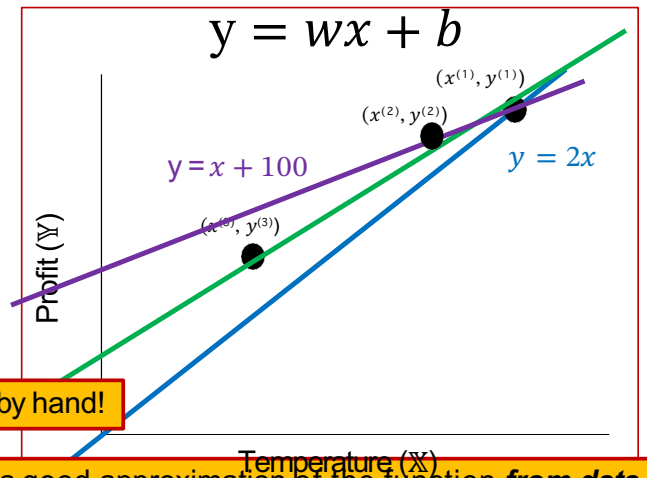
$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

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

Linear function

$y = wx + b$



$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

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

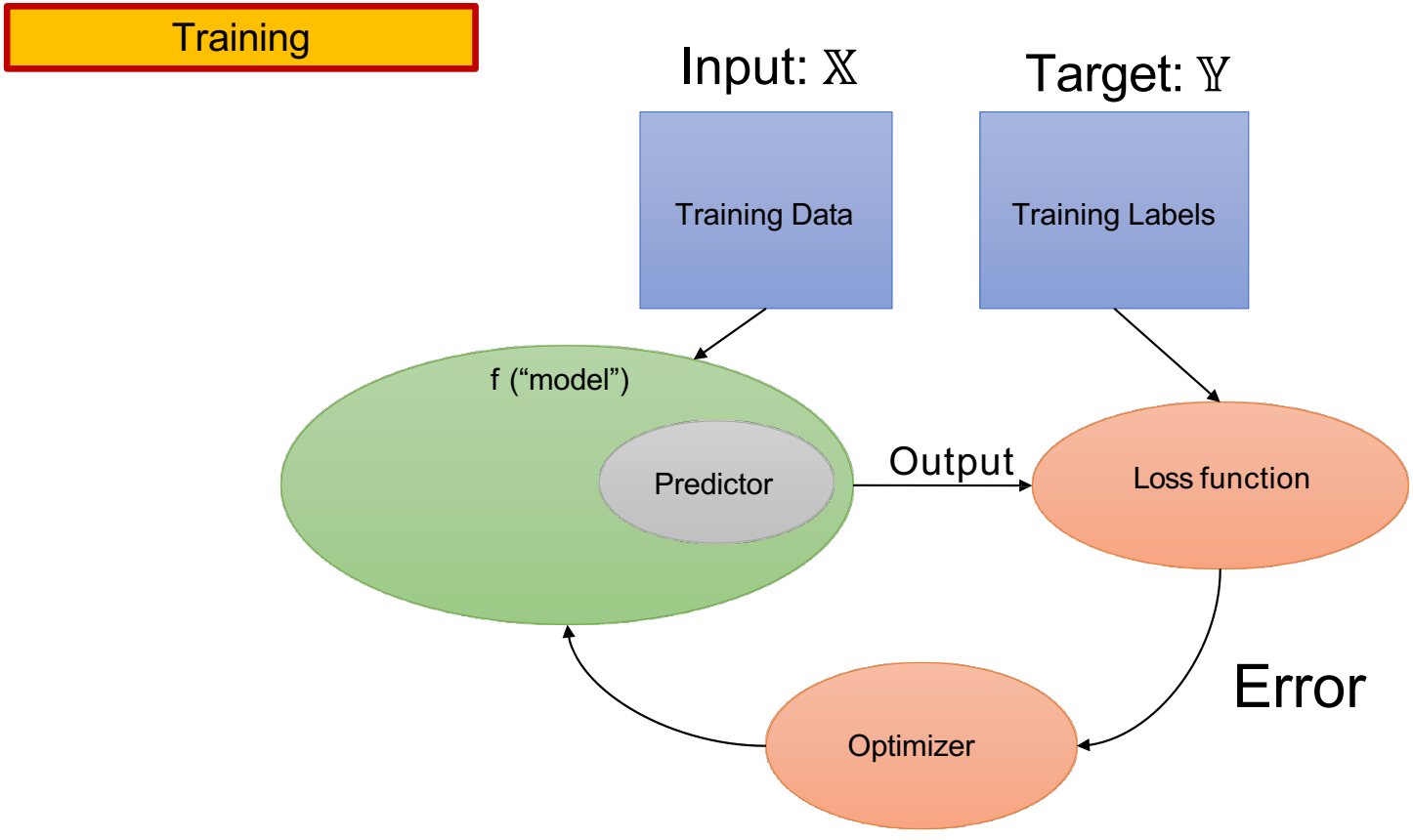
(Numerical output)

Very hard to learn these functions by hand!

Use machine learning to learn a good approximation of the function ***from data***

(Image only for explaining concept, not drawn accurately)

“Classic” Supervised Learning in Machine Learning



Any questions?

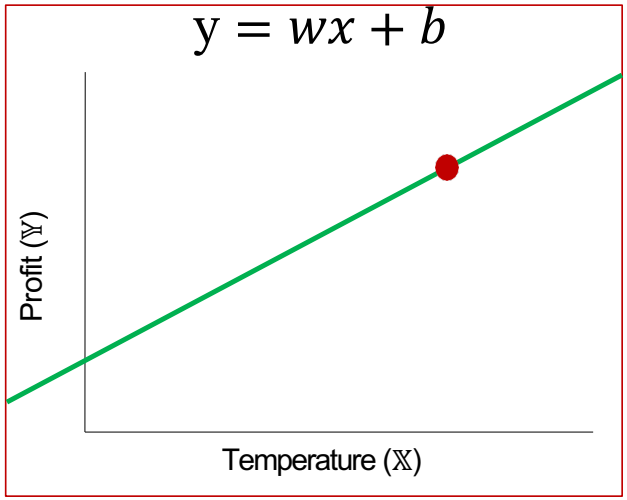
Testing our model

"Temperature"

$x' = 70$

Linear function

$$y = wx + b$$



Prediction

$y' = 175$

"Profit made on selling lemonade"

(Image only for explaining concept, not drawn accurately)

Testing our model

"Temperature"

"Profit made on selling lemonade"

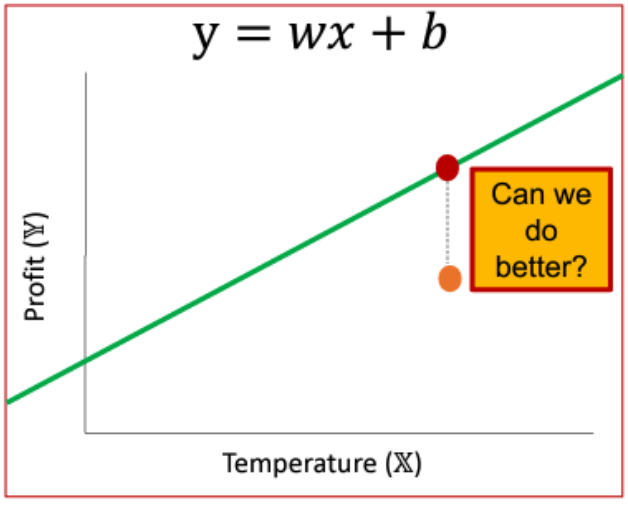
$x' = 70$

$\hat{x} = 70$

Real-world deployment

Linear function

$y = wx + b$



Prediction

$y' = 175$

True observation

$\hat{y} = 140$

(Image only for explaining concept, not drawn accurately)

(An outlier is a data point that differs significantly from other observations)

Can we do better?

- How?

Can we do better?

- How?
- Option 1: Collect more data and retrain
- Option 2: Try a different function
- Option 3: Do both 1 and 2

Learning better models – collect more data

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

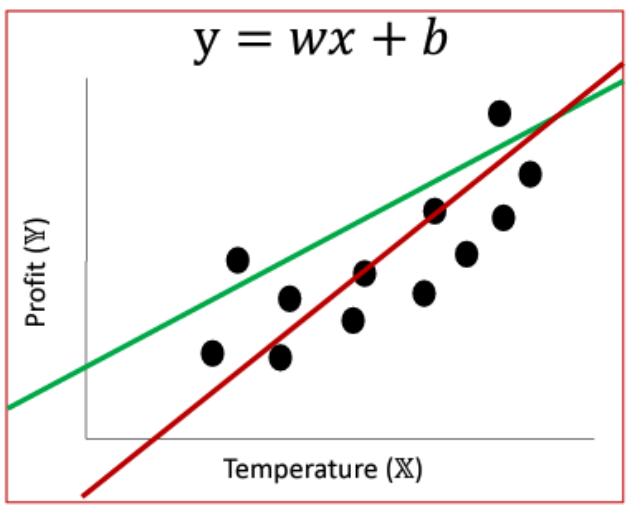
$\vdots$

$x^N = \cdots$

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

Linear function

$y = wx + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

$\vdots$

$y^N = \cdots$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning better models – Try different functions

Input: $\mathbb{X}$

"Temperature"

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

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

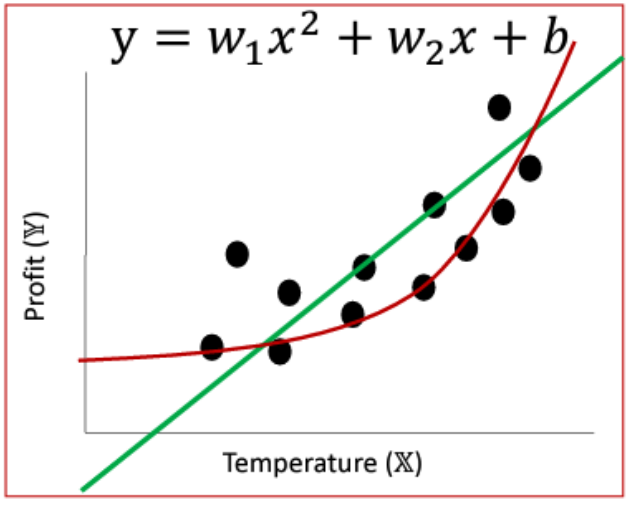
$\cdots$

$x^{N} = \ \ \ldots$

Non-linear function

Polynomial function

$y = w_1 x^2 + w_2 x + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

$\cdots$

$y^{N} = \ \ \ldots$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

How to know which function is the best?

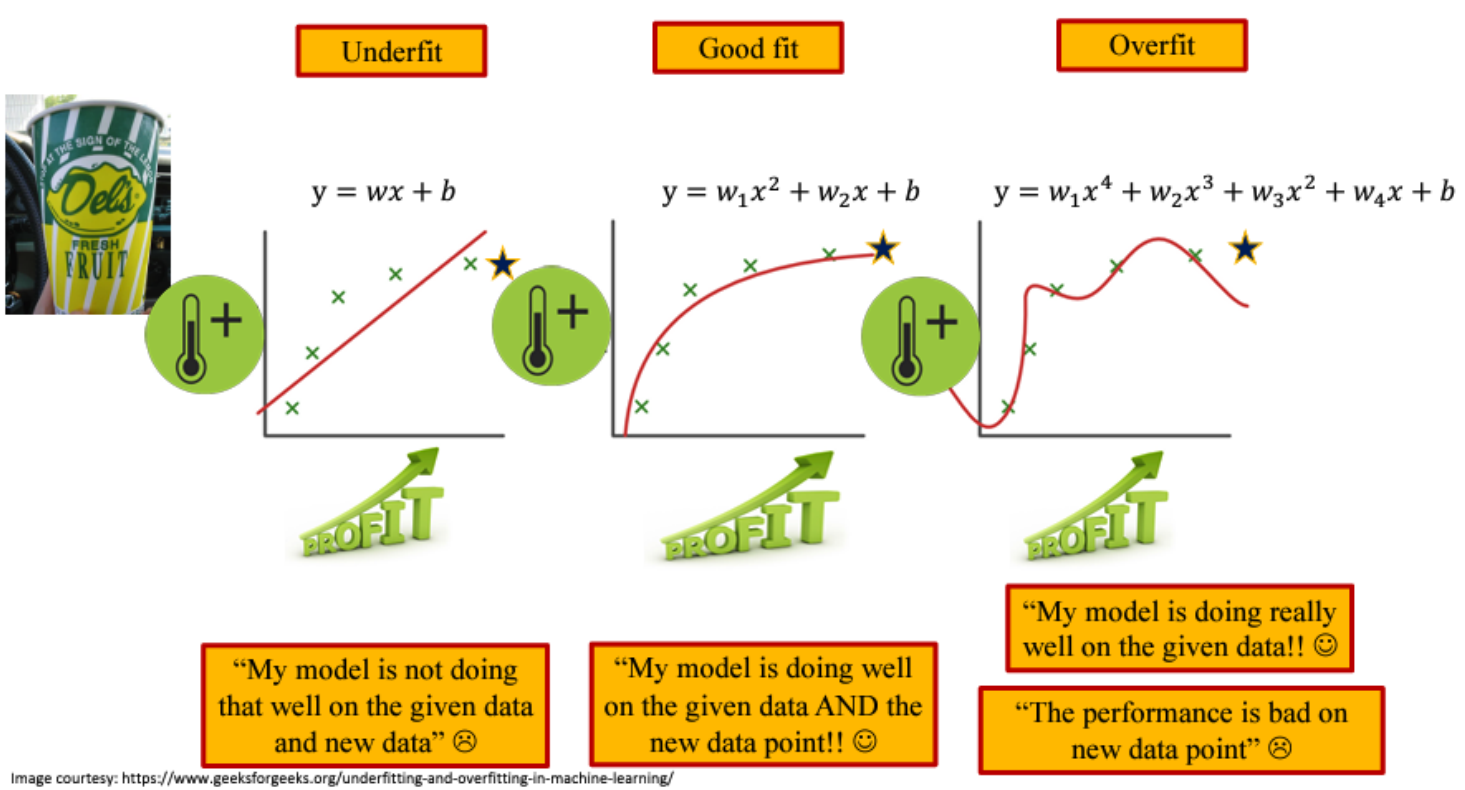


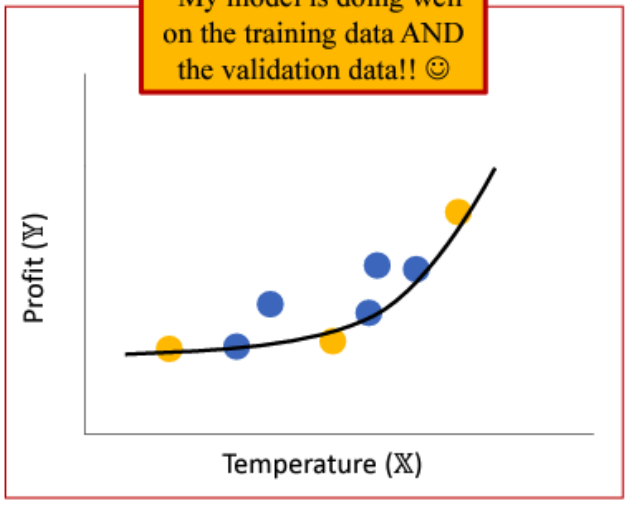
Image courtesy: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

How to train your model?

	Input: $\mathbb{X}$ "Temperature"	Target: $\mathbb{Y}$ "Profit made on selling lemonade"
Training set	$x^{(1)} = 100.1$	$y^{(1)} = 200.0$
	$x^{(2)} = 60.0$	$y^{(2)} = 160.5$
	$x^{(3)} = 30.3$	$y^{(3)} = 115.1$
	.	.
Validation set	.	.
	.	.
	.	.
	.	.
Test set	.	.
	.	.
	$x^N = \ldots$	$y^N = \ldots$

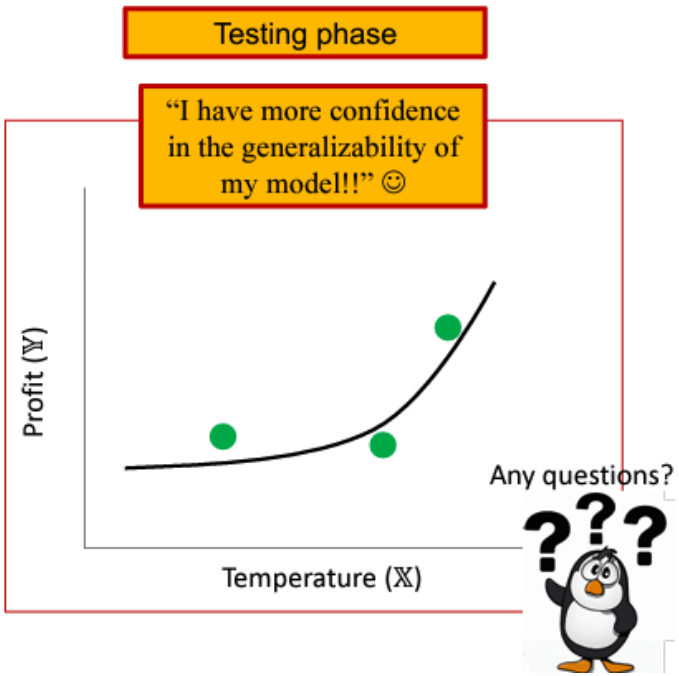
Training phase

"My model is doing well on the training data AND the validation data!! ☺"

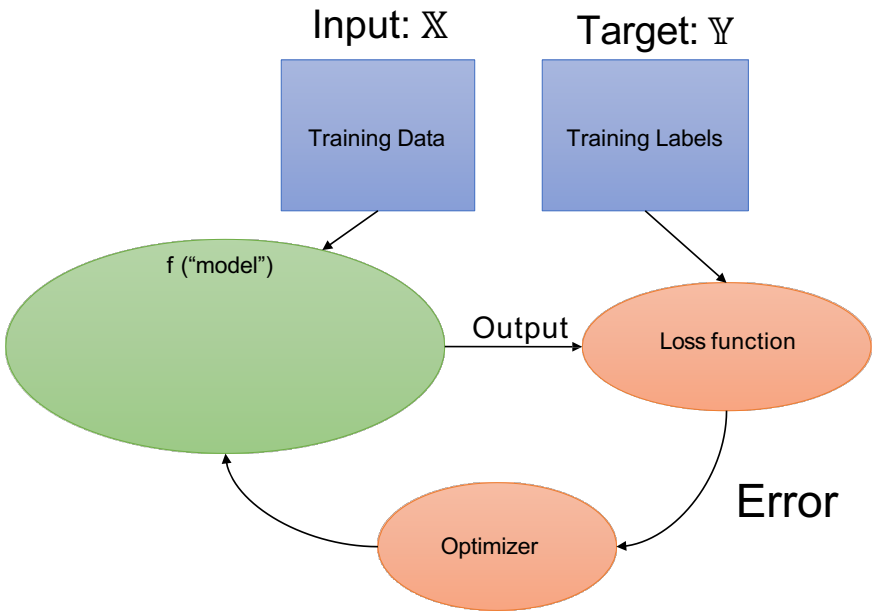


How to train your model?

	Input: $\mathbb{X}$ "Temperature"	Target: $\mathbb{Y}$ "Profit made on selling lemonade"
Training set	$x^{(1)} = 100.1$	$y^{(1)} = 200.0$
	$x^{(2)} = 80.0$	$y^{(2)} = 180.5$
	$x^{(3)} = 30.3$	$y^{(3)} = 115.1$
	.	.
Validation set	.	.
	.	.
	.	.
Test set	.	.
	.	.
	$x^N = \ldots$	$y^N = \ldots$



How to train your model?



Realworld data tends to be complicated!

Input: $\mathbb{X}$

"Temperature"	"Stand Hours"	"Sunny?"
$x_1^{(1)} = 100.1$	$x_2^{(1)} = 8$	$x_3^{(1)} = 1$
$x_1^{(2)} = 80.0$	$x_2^{(2)} = 4$	$x_3^{(2)} = 1$
$x_1^{(3)} = 30.3$	$x_2^{(3)} = 8$	$x_3^{(3)} = 0$
.	.	.
.	.	.
.	.	.
.	.	.
$x_i^{(k)} = \ldots$		

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

Now our function needs to capture the relationships of the combined feature space of the input and the output!

Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

. . . .

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

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Recap

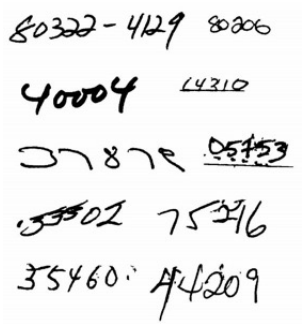
- Represent input and output as numbers
- Classification – predicting categorical outputs
- Regresssion – predicting numerical outputs
- Supervised learning – learn a function that approximates the data well

Outline

- What is deep learning?
- **Perceptron and parameters**
- Perceptron learning algorithm and loss function

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!



Our Problem

Input: $\mathbb{X}$

Function: f

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

Target: $\mathbb{Y}$

Which digit is it?

"3"

Our Problem

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

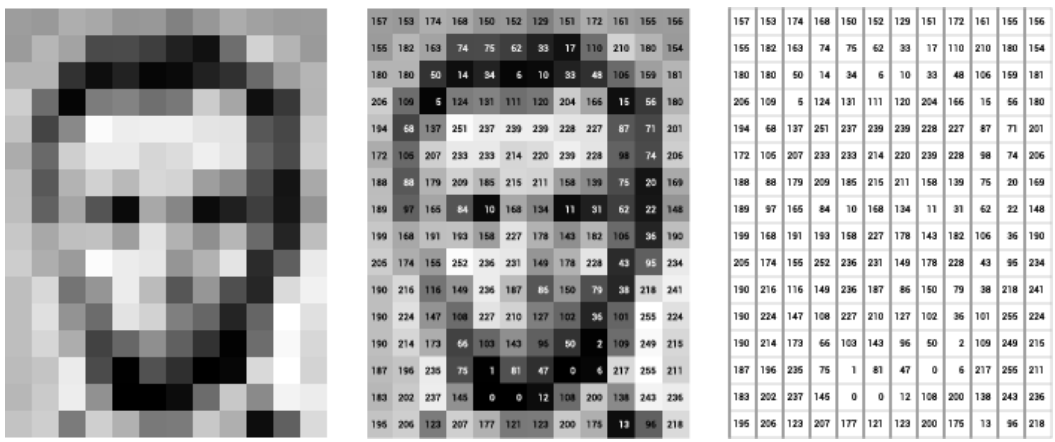


Image from Stanford

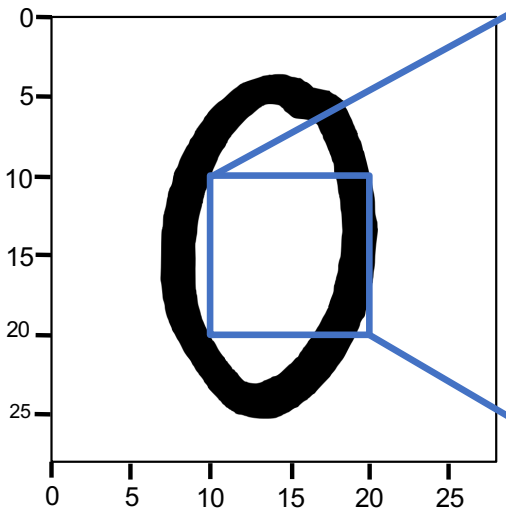
Representing digits in the computer

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

Representing digits in the computer

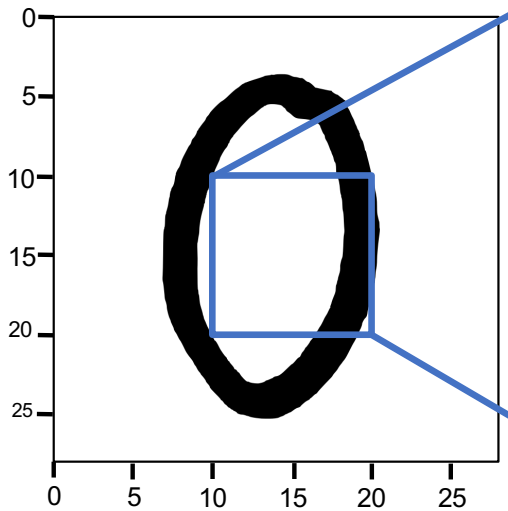


	10	11	12	13	14	15	16	17	18	19
10	0	0	0	0	0	0	0	170	236	255
11	0	0	0	0	0	0	0	0	250	254
12	0	0	0	0	0	0	0	0	255	255
13	0	0	0	0	0	0	0	0	240	245
14	0	0	0	0	0	0	0	0	245	236
15	0	0	0	0	0	0	0	0	251	253
16	0	0	0	0	0	0	0	124	255	255
17	0	0	0	0	0	0	0	210	254	120
18	0	0	0	0	0	0	0	165	230	0
19	0	0	0	0	0	0	120	154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

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

what the computer sees

Representing digits in the computer



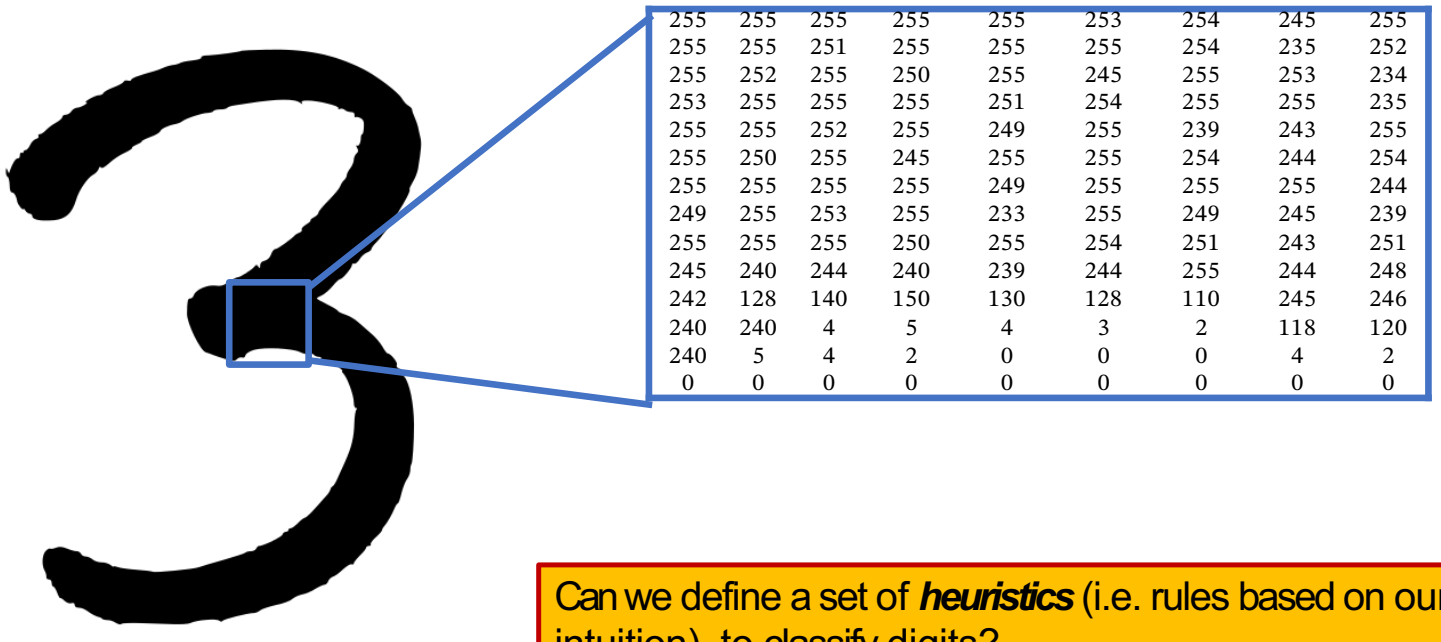
	10	11	12	13	14	15	16	17	18	19
10	0	0	0	0	0	0	0	170	236	255
11	0	0	0	0	0	0	0	0	250	254
12	0	0	0	0	0	0	0	0	255	255
13	0	0	0	0	0	0	0	0	240	245
14	0	0	0	0	0	0	0	0	245	236
15	0	0	0	0	0	0	0	0	251	253
16	0	0	0	0	0	0	0	124	255	255
17	0	0	0	0	0	0	0	210	254	120
18	0	0	0	0	0	0	0	165	230	0
19	0	0	0	0	0	0	120	154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

Often has lighter pixels in the middle!

How does the pattern compare with digit 3?

Representing digits in the computer

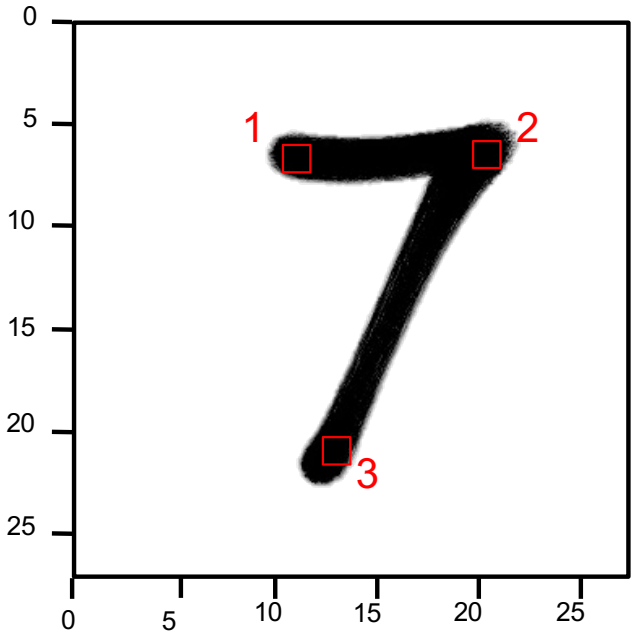
Darker pixels in the middle



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

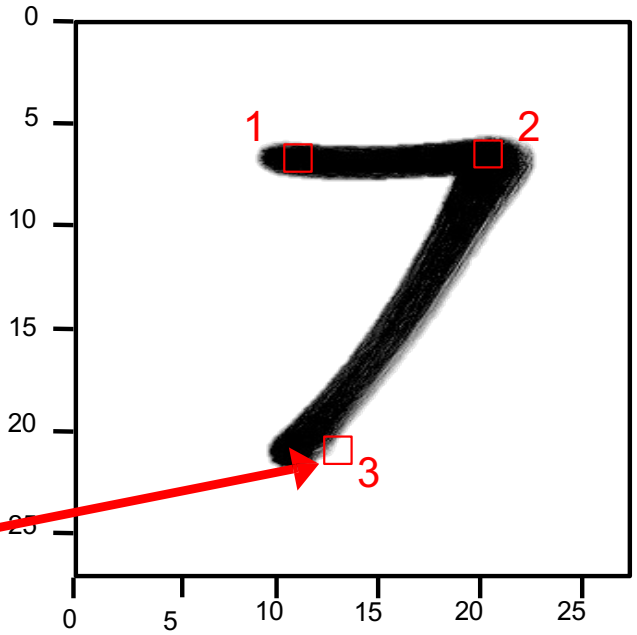
Can we define a set of ***heuristics*** (i.e. rules based on our intuition), to classify digits?

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



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

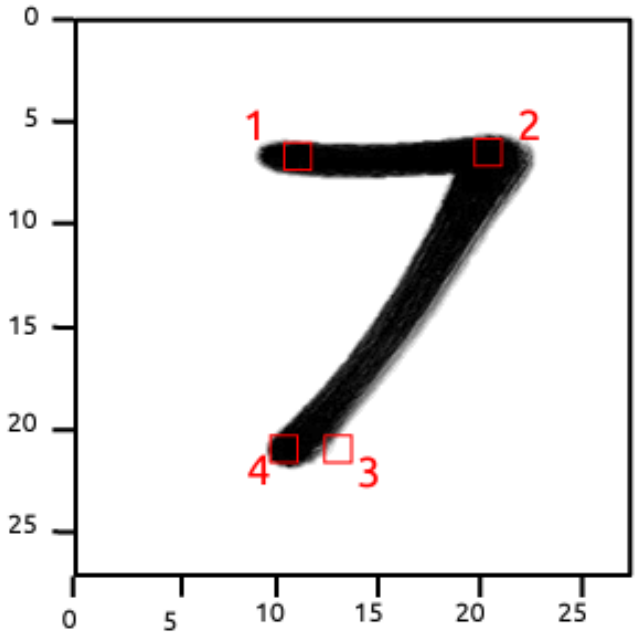
But what if ...



pixel 3 is no longer dark!

Slanted digit?

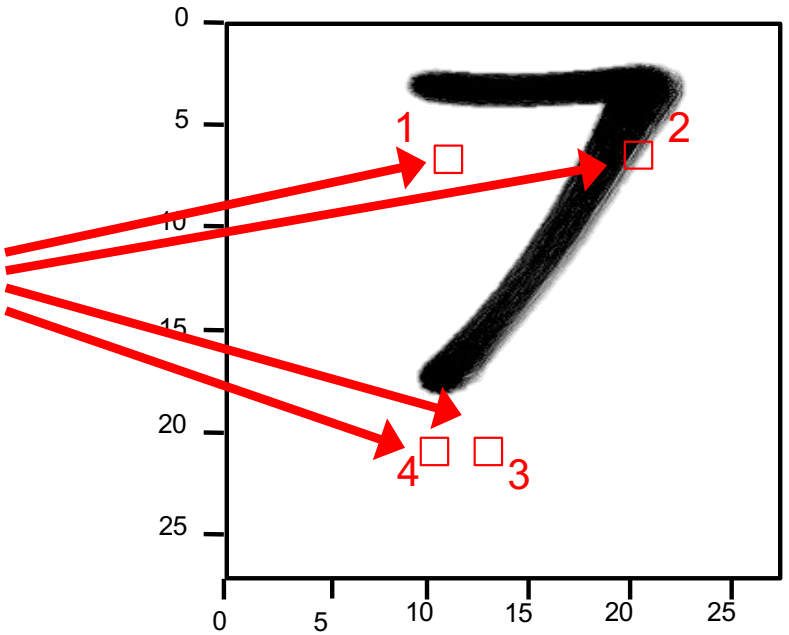
An Improved Heuristic!



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

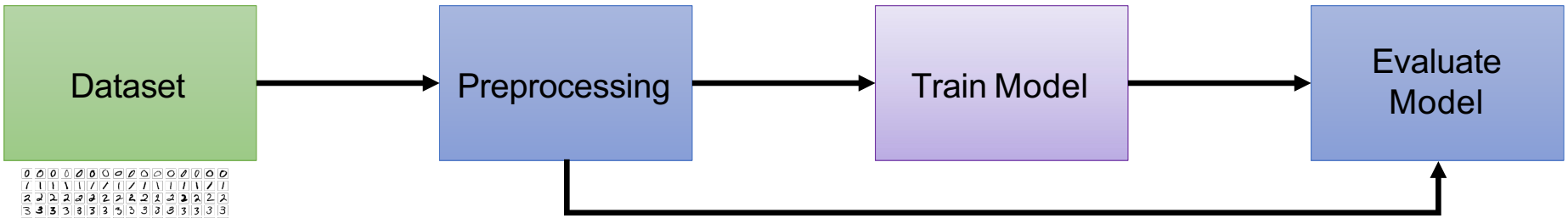


Digit shifted up?

Heuristic...

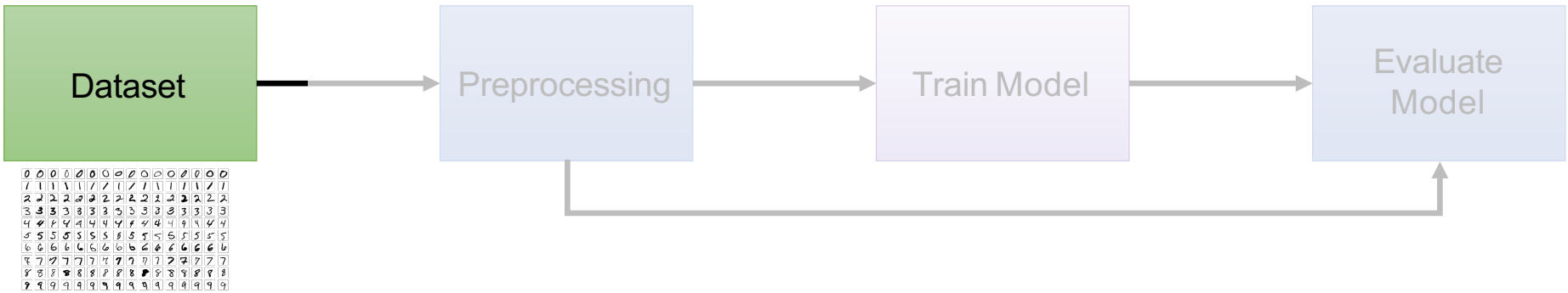
- Not as simple as we think!
- Distortions, overlappings, underlinings, etc.
- Cannot rely on a set of exact rules
- Let's do some machine learning!

Machine Learning Pipeline for Digit Recognition



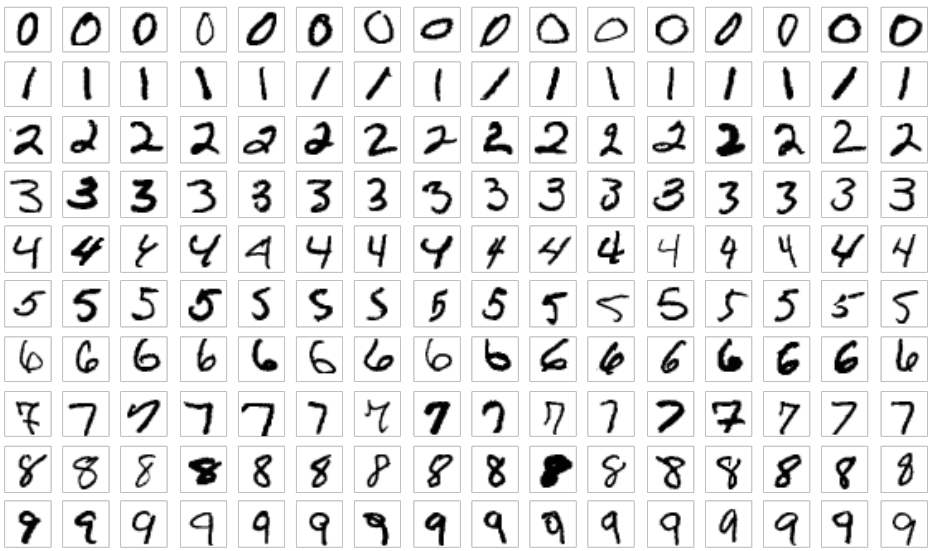
Any questions?

Machine Learning Pipeline for Digit Recognition

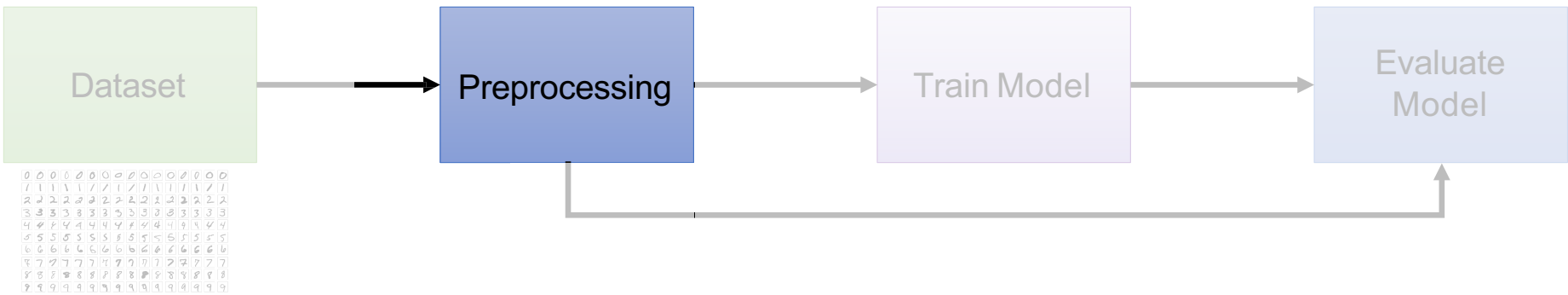


MNIST

- Modified National Institute of Standards and Technology database
- Handwritten digits
- 0 — 9 (10 classes)
- 70,000 images

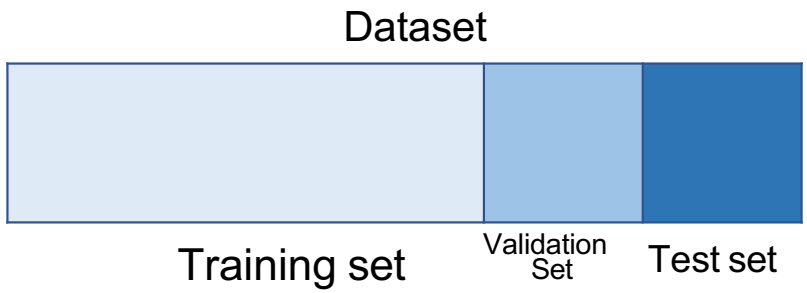


Machine Learning Pipeline for Digit Recognition



Train, validation, and test sets

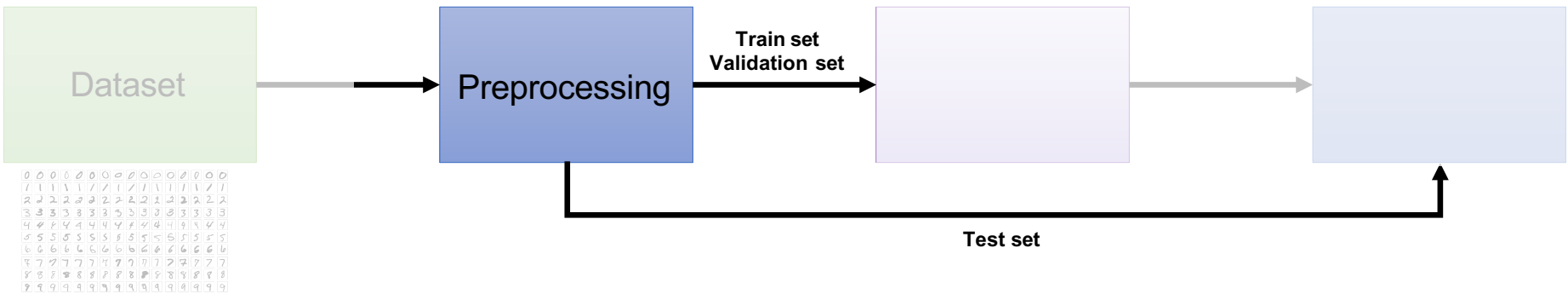
- Train set: used to adjust the parameters of the model
- Validation set: used to test how well we're doing as we develop
  - Prevents overfitting
- Test set: used to evaluate the model once the model is done



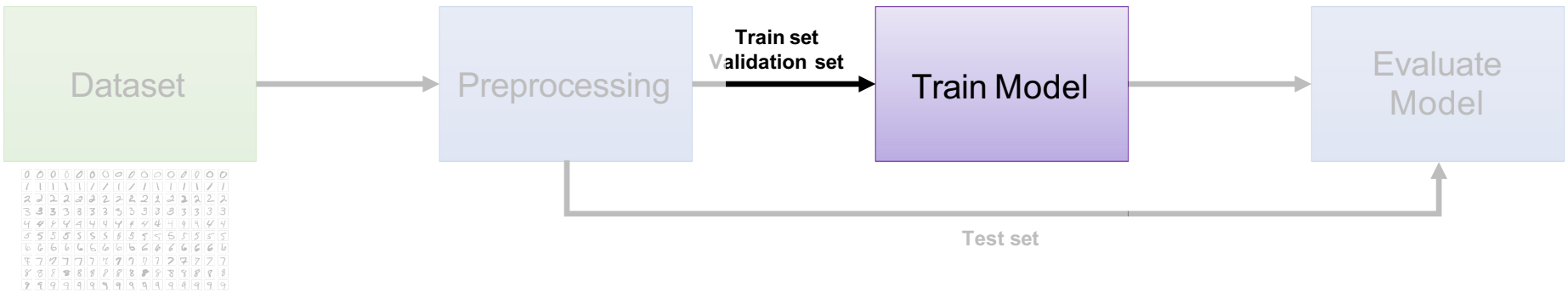
MNIST

- Training set: 60,000 images
- Test set: 10,000 images
- No explicit validation set
- What do you suggest we do here?

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?

What is our prediction task?

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

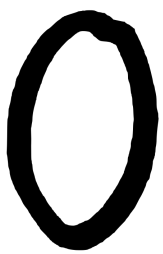
$x^{(1)} =$ 

28x28 pixels

Function: f

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

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

$x^{(2)} =$ 

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

Our simplified problem

What is our input space?

What is our output space?

What is our prediction task?

Input: $\mathbb{X}$

Pixel Grid

$x^{(1)} =$ 

28x28 pixels

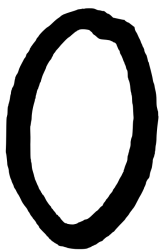
Function: f

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

Target: $\mathbb{Y}$

Is it digit 2?

$y^{(1)} = 1$

$x^{(2)} =$ 

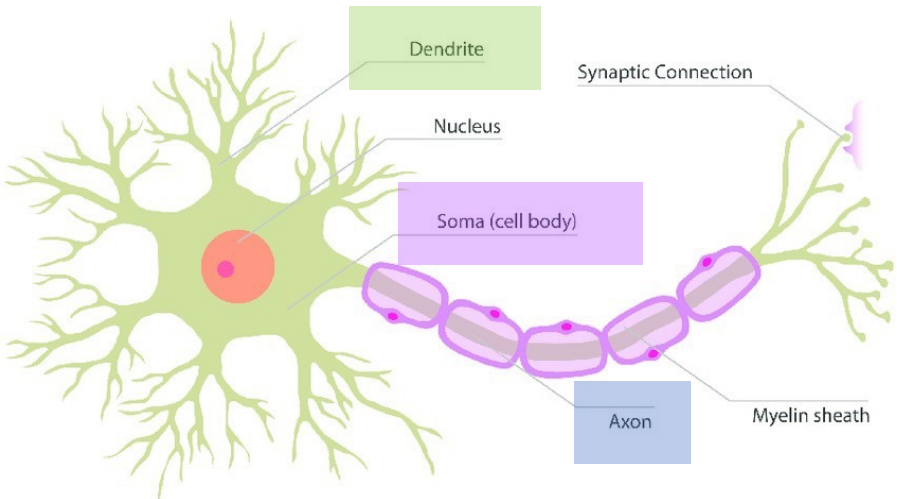
$y^{(2)} = 0$

Learn about the first component of deep learning model

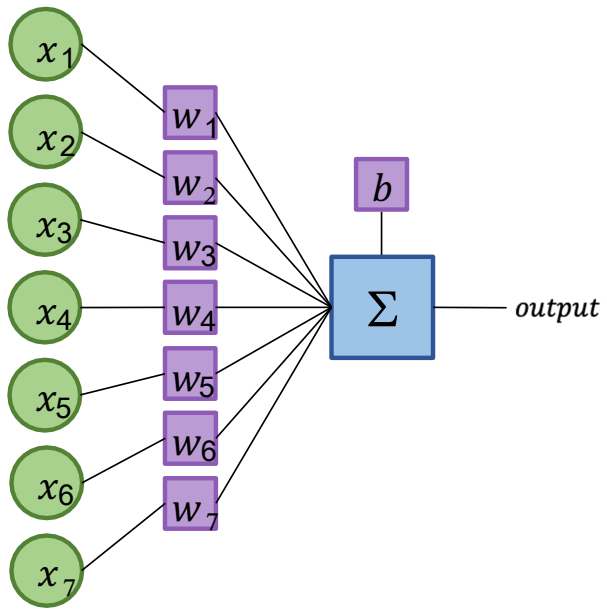
- Perceptron

Biological motivation

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



Biological Neuron



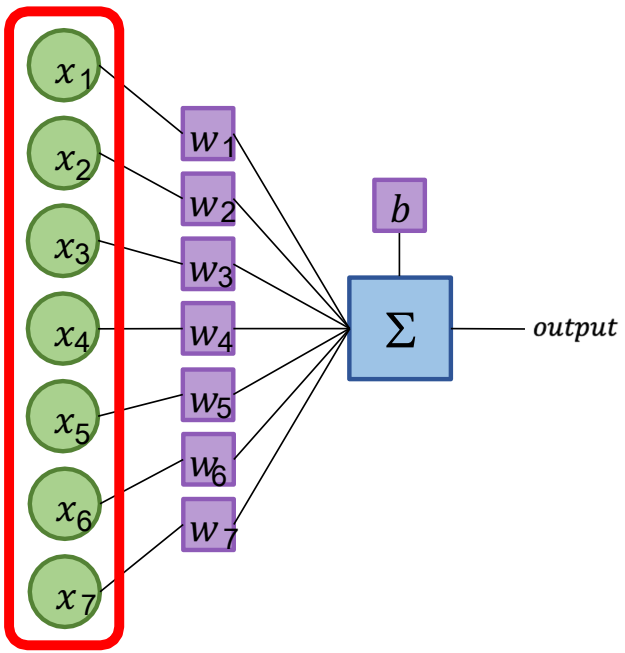
Artificial Neuron (Perceptron)

The Perceptron

- “In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.”

Input

- Input: a vector of numbers $x = [x_1, x_2, ..., x_n]$



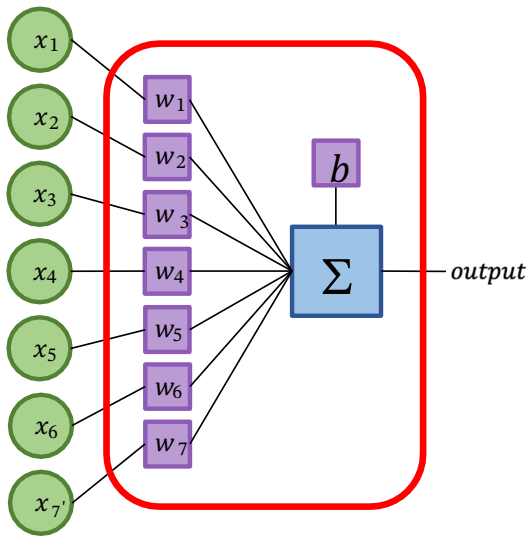
What was x_i for lemonade stand example?

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

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



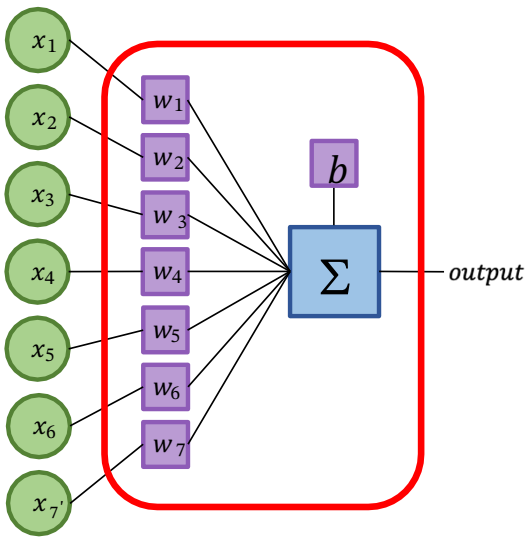
Predicting with a Perceptron

- Multiply each input x_i by its corresponding weight w_i, sum them up.
- Add the bias b.
- If the result value is greater than 0, return 1, otherwise return 0.

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

Parameters

- w and b are parameters of the perceptron.
- Parameters: values we adjust during learning
- Let $\phi = \{w \bigcup b\}$



Parameters

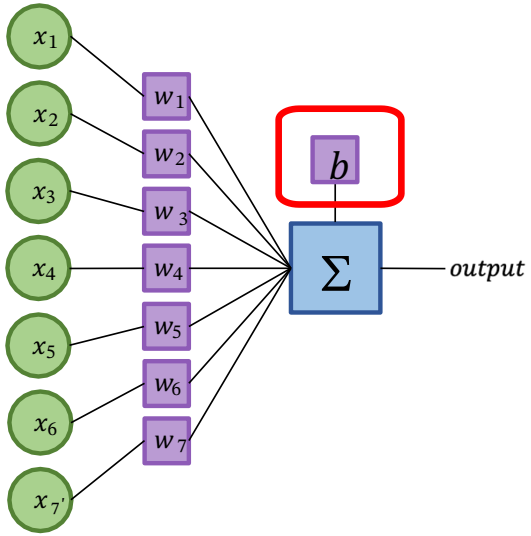
- Weights — the importance of each input to determining the output
  - Weight near 0 imply this input has little influence on the output
  - Negative weight means?
    - Option 1: Increasing input will increase output
    - Option 2: Increasing input will decrease output
    - Option 3: Decreasing input 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?
    - Option 1: Increasing input will increase output
    - Option 2: Increasing input will decrease output
    - Option 3: Decreasing input will decrease output
    - The answer is Option 2.

Parameters

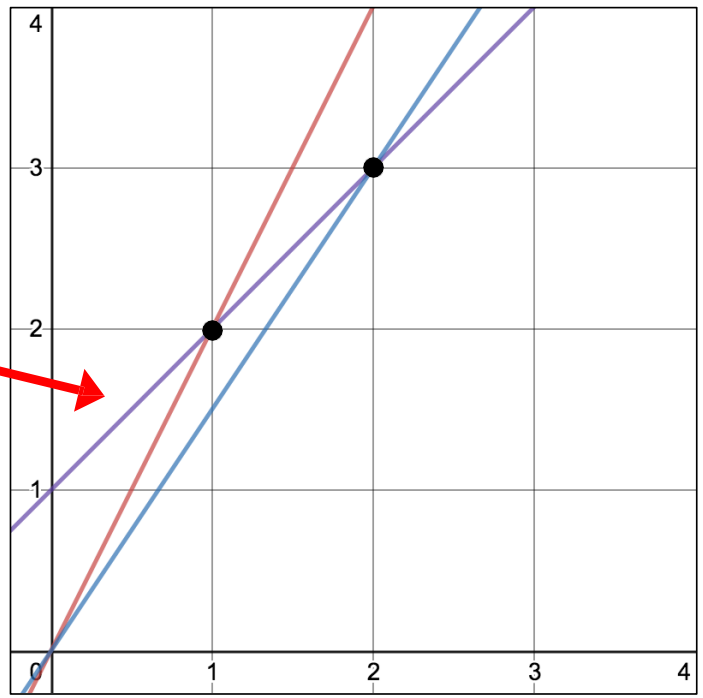
- Bias - What do we need this for?



Bias: Geometric Explanation

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

only the line with bias can fit the data



$$f(x) = 2x$$

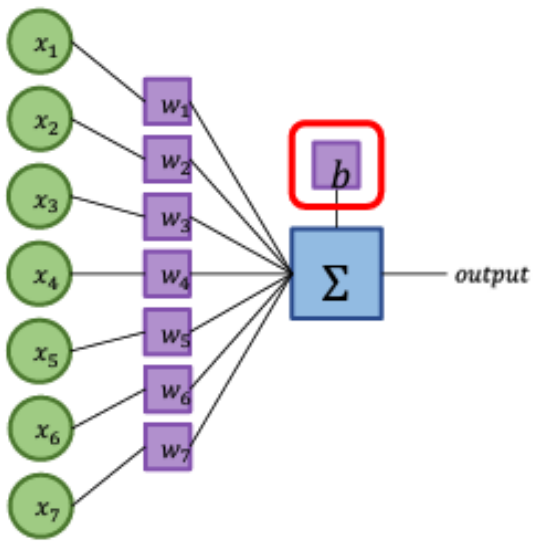
$$f(x) = \frac{3}{2}x$$

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

Bias: Conceptual Explanation

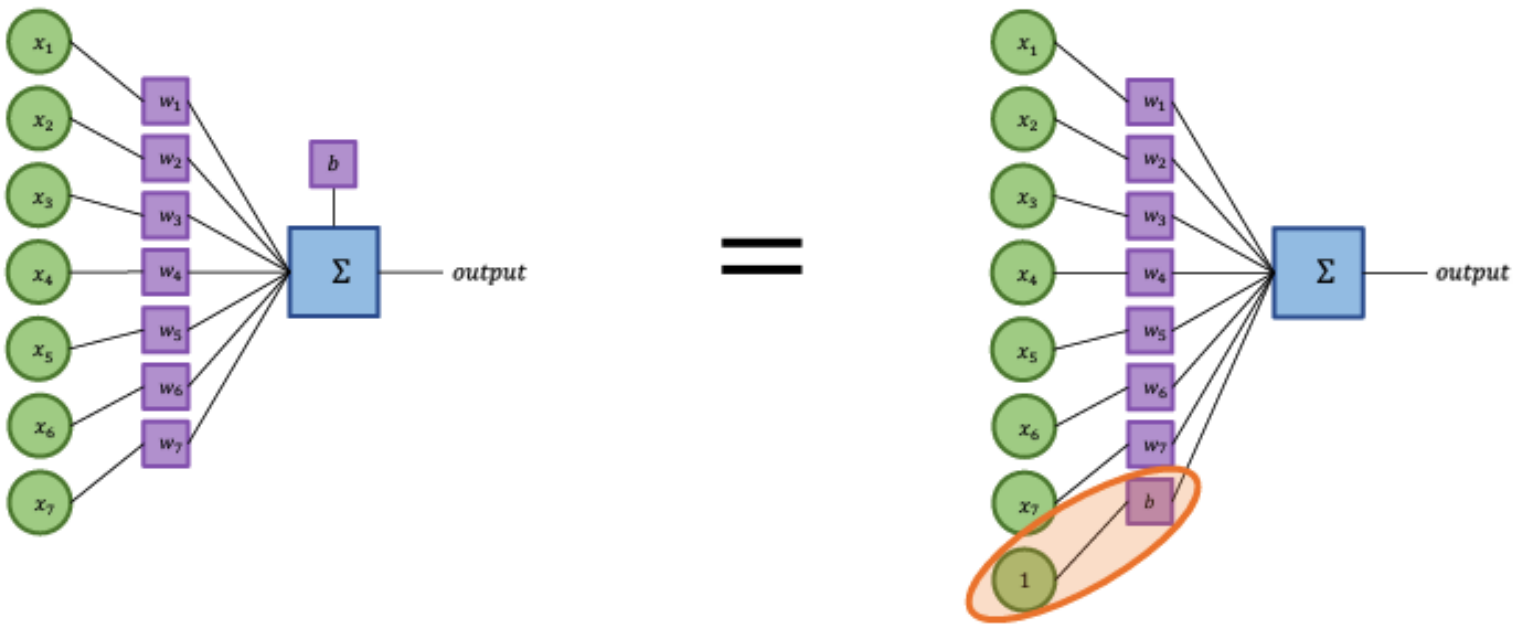
- Bias — the a priori likelihood of the positive class
  - Ensures that even if all inputs are 0, there will be some result value
  - Just because all inputs are 0, it does not mean there are no 1's in the world
  - Maybe there just happen to be more, say, 0's than 1's

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



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

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

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

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

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

Simplifying some notation

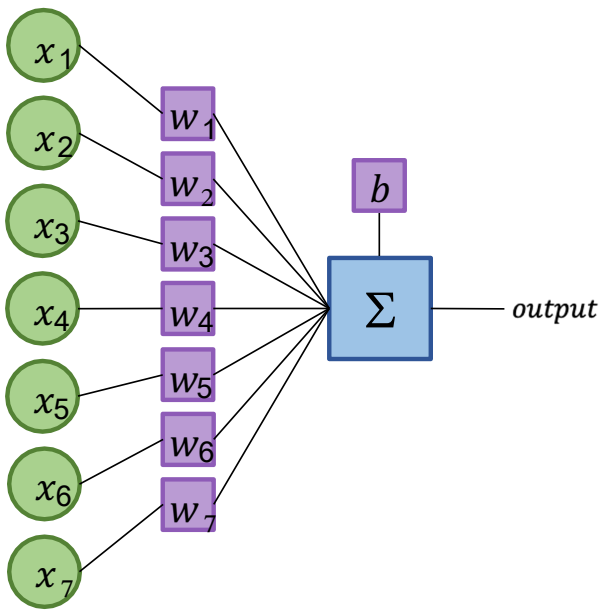
- Recall: the dot product of two vectors of length n is $a \cdot b = \sum_{i=1}^{n} a_i b_i$.
- We can rewrite the perceptron function accordingly:
- In modern deep learning paralance, b+w·x is know as a linear unit.

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

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

A Binary Perceptron from MNIST

- Inputs $[x_1, x_2, ...x_n]$ are all positive
  - n = 784 (28 * 28 pixel values)
- Output is either 0 or 1
  - 0 -> input is not the digit type we are looking for.
  - 1 -> input is the digit type we are looking for.



Outline

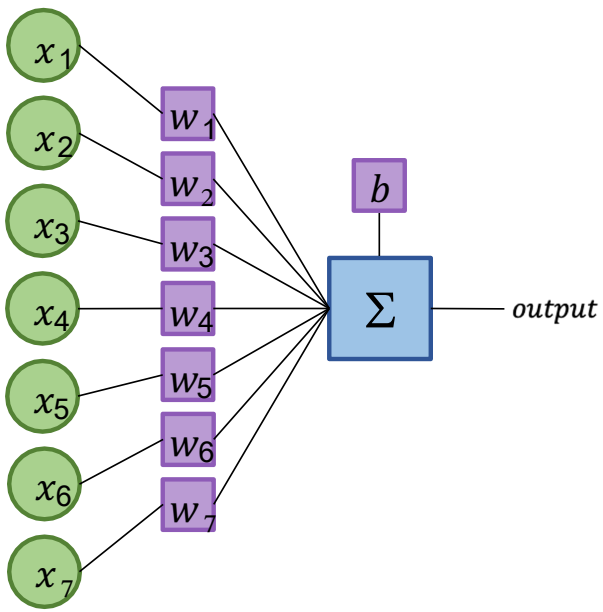
- What is deep learning?
- Perceptron and parameters
- **Perceptron learning algorithm and loss function**

Perceptron and Learn about the Loss Functions

- Perceptron learning algorithm
- Extending perceptron for Multi-class classification
- Loss functions for Regression and Classification

A Binary Perceptron from MNIST

- Inputs $[x_1, x_2, ...x_n]$ are all positive
  - n = 784 (28 * 28 pixel values)
- Output is either 0 or 1
  - 0 -> input is not the digit type we are looking for.
  - 1 -> input is the digit type we are looking for.



Training a perceptron

- 0: set the parameters $\phi = \{w \bigcup b\}$ to 0.
- 1: Iterate over training set several times, feeding in each training example into the model, producing an output, and adjusting the parameters according to whether that output was right or wrong
- 2: Stop once we either get every training example right or after N iterations, a number set by the programmer.
  - N is known as the number of *epochs*, where each epoch is an iteration of going through all data points in the training set
  - As a general rule of thumb, N grows with the number of parameters.

The Perceptron Learning Algorithm

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
- $\mathbf{x}^k$ = k$^{\text{th}}$ training example
- y^k = label for the k$^{\text{th}}$example
- w_i = weight for the i$^{\text{th}}$ input where $i \leq n$
- n = number of pixels per image
- x_i^k = i$^{\text{th}}$ input of the example where $i \leq n$

The Perceptron Learning Algorithm

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 $w_i, \Delta w_i = \left(y^k - f(\mathbf{x}^k)\right) x_i^k$

- If the output of our model matches the label, we continue
- If the correct label is 1, and our output is 1, $1 - 1 = 0$
- If the correct label is 0, and our output is 0, $0 - 0 = 0$

The Perceptron Learning Algorithm

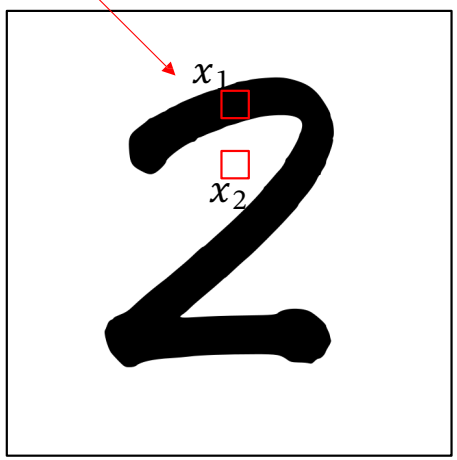
1. set w's to 0.
2. for N iterations, or until the weights do not change:
   a) for each training example $\mathbf{x}^k$ with label y^k
      i. if $y^k - f(\mathbf{x}^k) = 0$ continue
      ii. **else for all weights w_i, $\Delta w_i = \left(y^k - f(\mathbf{x}^k)\right) x_i^k$**

- If our label y^k is a 1, and our model's output is a 0, we update the i^{th} weight by:
  - $(1 - 0) \cdot x_i^k = x_i^k$
  - Output was 0 and should have been 1, so make the output more positive
- If our label y^k is a 0, and our model's output is a 1, we update the i^{th} weight by:
  - $(0 - 1) \cdot x_i^k = -x_i^k$
  - Output was 1 and should have been 0, so make the output more negative

Example: Predict whether a digit is a “2”

- Predict whether a digit is a “2”

Just look at the effect of these two pixels

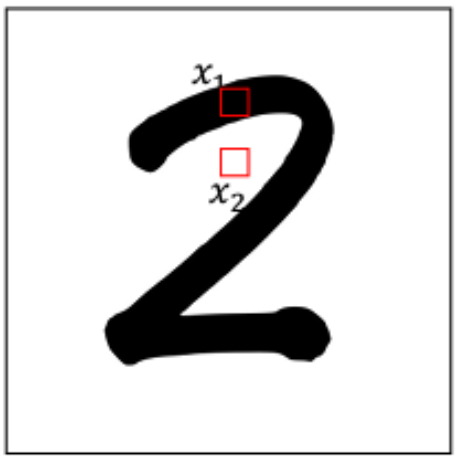


$$x_1 = 0.8$$
$$x_2 = 0$$

Example: Predict whether a digit is a "2"

- Start off training with all parameters as 0, so $w_1 = 0$, $w_2 = 0$, and $b = 0$
- $f(x) = (w_1 \cdot x_1 + w_2 \cdot x_2 + b.1)$
- $f(x) = (0 \cdot 0.8 + 0 \cdot 0 + 0 \cdot 1) = 0$
  - Return 0 because value is not greater than 0
- Predict that it is not a 2!
- Correct answer: it is a 2...
- Parameter update:
  - $\Delta w_1 = (1 - 0) \cdot 0.8 = 0.8$
  - $\Delta w_2 = (1 - 0) \cdot 0 = 0$
  - $\Delta b = (1 - 0) \cdot 1 = 1$
- Now
  - $w_1 = 0.8$
  - $w_2 = 0$
  - $b = 1$

True label = 1

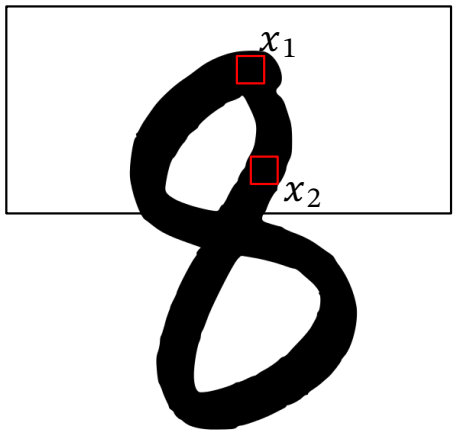


$x_1 = 0.8$
$x_2 = 0$

Example: Predict whether a digit is a “2”

Remember the starting weights are now:

$w_1 = 0.8$
$w_2 = 0$
$b = 1$



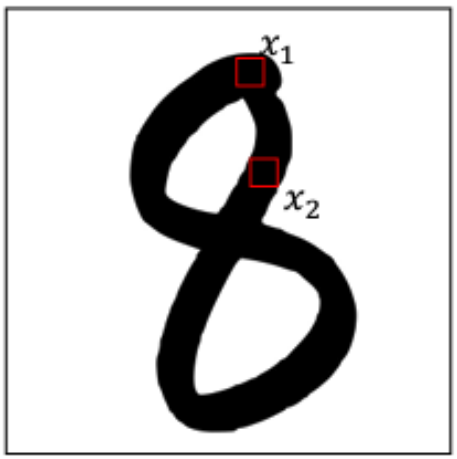
$x_1 = 0.9$
$x_2 = 0.9$

Example: Predict whether a digit is a "2"

- At end of last iteration:
  - $w_1 = 0.8, w_2 = 0$, and $b = 1$
- $f(x) = (w_1 \cdot x_1 + w_2 \cdot x_2 + b.1)$
- $f(x) = (0.8 \cdot 0.9 + 0 \cdot 0.9 + 1 \cdot 1) > 0$
  - Return 1 because value is greater than 0
- Predict that it is a 2!
- Correct answer: it is not a 2…
- Parameter update:
  - $\Delta w_1 = (0 - 1) \cdot 0.9 = -0.9$
  - $\Delta w_2 = (0 - 1) \cdot 0.9 = -0.9$
  - $\Delta b = (0 - 1) \cdot 1 = -1$
- Now
  - $w_1 = 0.8 - 0.9 = -0.1$
  - $w_2 = 0 - 0.9 = -0.9$
  - $b = 1 - 1 = 0$

Any questions?

True label = 0



$x_1 = 0.9$
$x_2 = 0.9$

Perceptron and Learn about the Loss Functions

- Perceptron learning algorithm
- Extending perceptron for Multi-class classification
- Loss functions for Regression and Classification

Bringing back the complexity

Input: $\mathbb{X}$

Classifying MNIST digits requires predicting 1 of 10 possible values

Target: $\mathbb{Y}$

Pixel Grid

$x^{(1)} =$ 

28x28 pixels

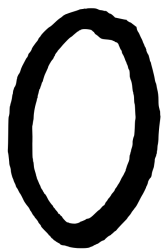
Function: f

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

Which digit is it?

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

How do we do that?

$x^{(2)} =$ 

Rather than predicting whether a handwritten digit is of a certain class, we predict the class it is most likely in

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

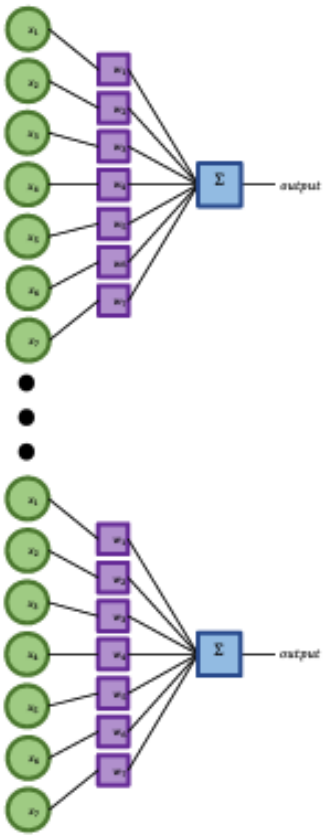
Using multiple perceptrons

- We can extend perceptrons to multi-class problems by creating m perceptrons, where m = the number of classes

Using multiple perceptrons

- We can extend perceptrons to multi-class problems by creating m perceptrons, where m = the number of classes
- For MNIST, we would have 10 perceptrons
- Each individual perceptron returns a value, so our model will return the class whose perceptron value is the highest.
  - Here, “perceptron value” refers to the value of the weighted sum before being thresholded.

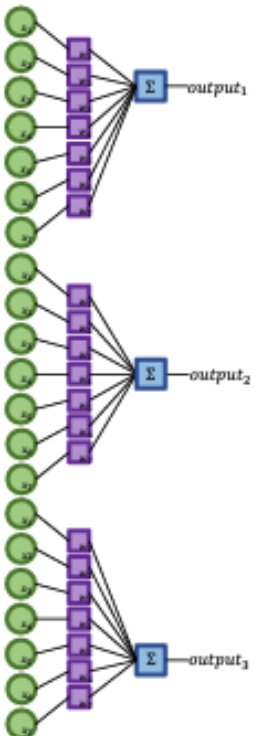
Using multiple perceptrons



Perceptron for predicting whether handwritten digit is a 0

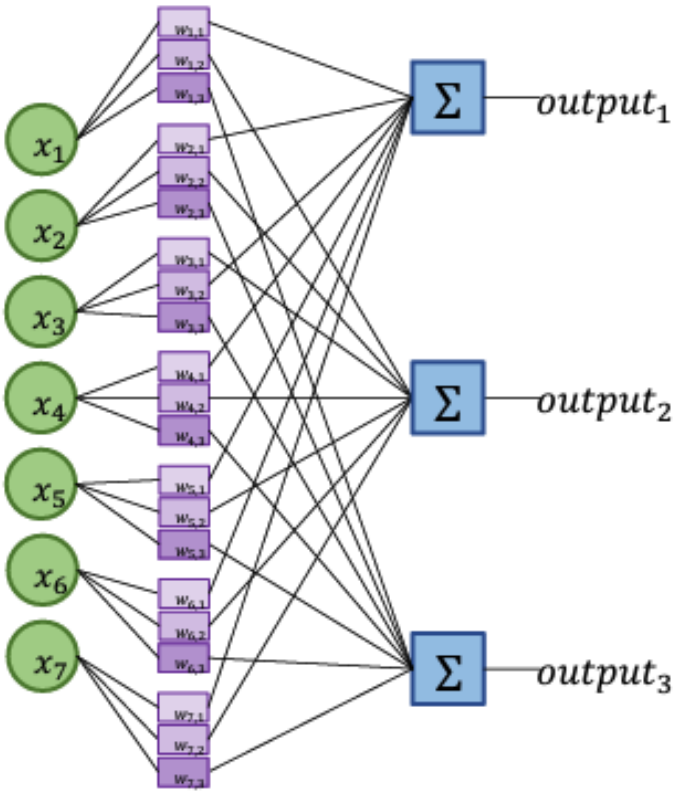
Perceptron for predicting whether handwritten digit is a 9

Multi-class Perceptron



Three separate perceptrons

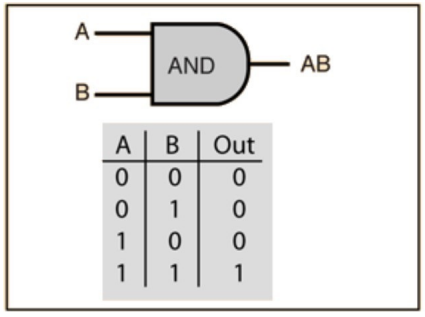
Multi-class Perceptron



Three perceptrons sharing inputs

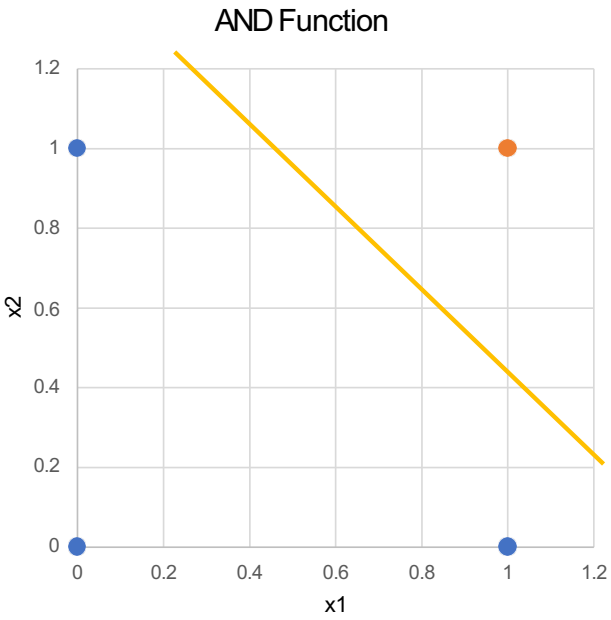
AND Function

- Perceptrons work well with linearly separable data.



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate



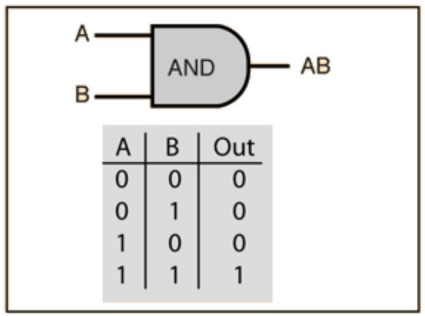
Output = 0

Output = 1

$$w_5 \cdot x_5 + w_6 \cdot x_6 + b > 0\ ?$$

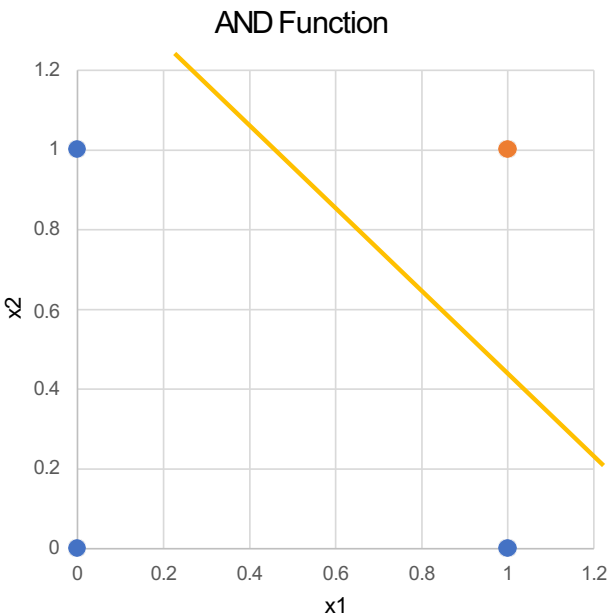
Linear decision boundary

OR Function



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate



Output = 0

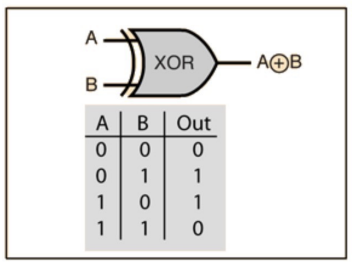
Output = 1

$w_5 \cdot x_5 + w_6 \cdot x_6 + b > 0$?

Linear decision boundary

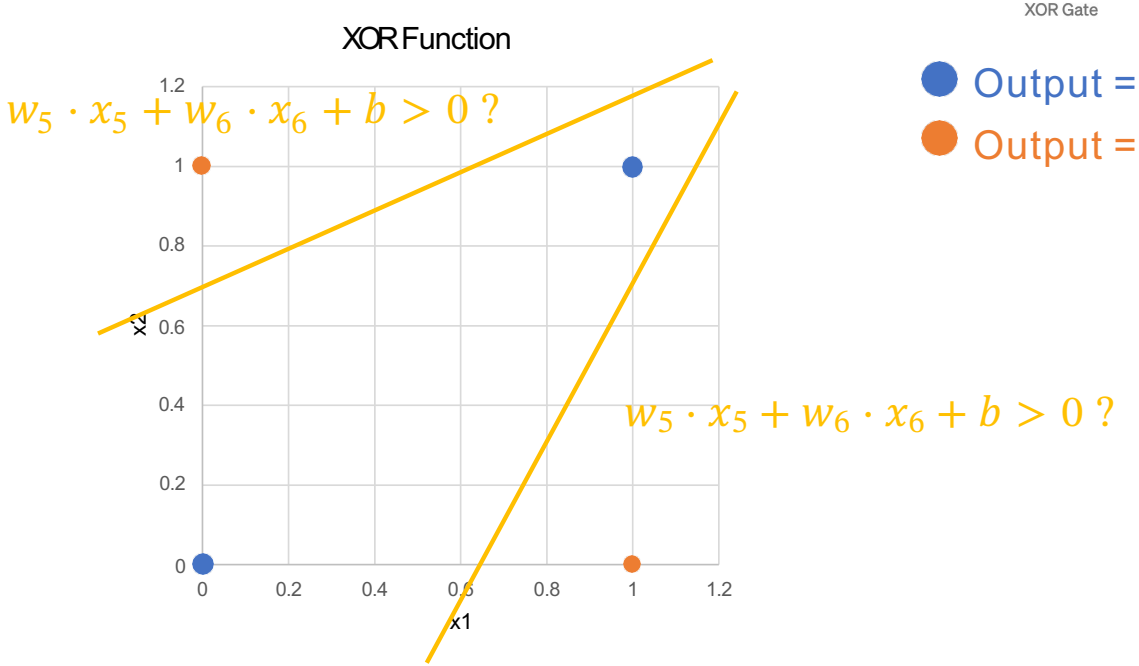
XOR Function

- Complicated data would need a complicated function!
- Need two linear decision boundaries to represent this function.



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

XOR Gate



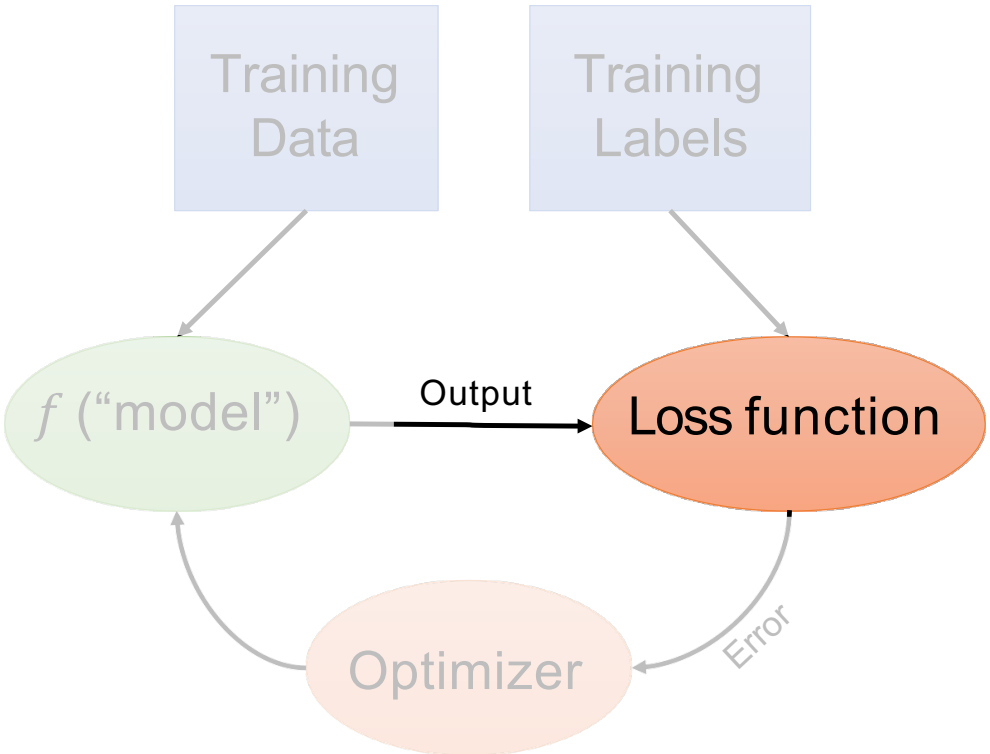
Perceptron and Learn about the Loss Functions

- Perceptron learning algorithm
- Extending perceptron for Multi-class classification
- Loss functions for Regression and Classification

How do we train multi-layer networks?

- Unfortunately, the perceptron algorithm doesn't generalize beyond the one-layer case...
- We need a new algorithm...

A critical ingredient for our new approach: Loss functions



What is a Loss Function?

- A function L which measures how “wrong” a network is
- L is computed by comparing two values (predicted and true) that shows which is better
- Evaluate L and update parameters to decrease L, making the network “less wrong”

Recap – regression task

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

Recap – Learning function f

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 60.0$

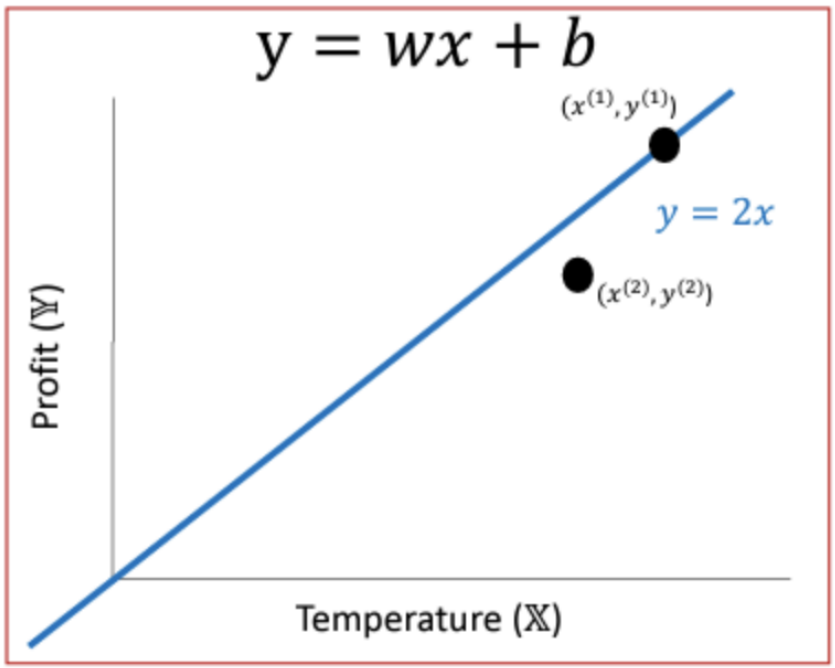
$x^{(3)} = 30.3$

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

What could be our loss function?

Linear function

$y = wx + b$



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 160.5$

$y^{(3)} = 145.1$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Mean Squared Error (MSE)

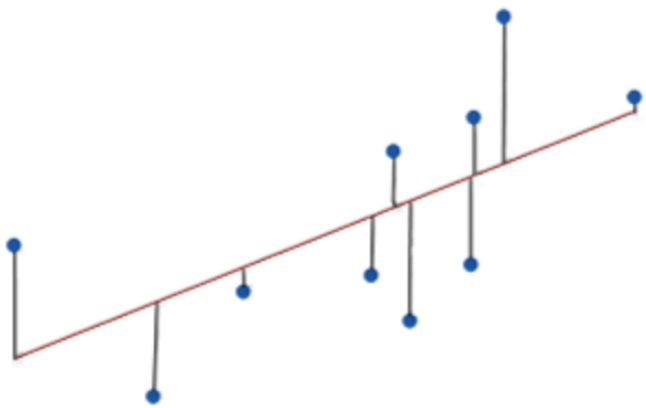
- Average squared residual (residual: difference between predicted and true value)
- Decreasing the MSE = the model has less error = data points fall closer to the regression line

$$MSE = \frac{\sum_{k=1}^{n}(y^k - \hat{y}^k)^2}{n}$$

y^k : *true output value*

$\hat{y}^k$: *predicted output value*

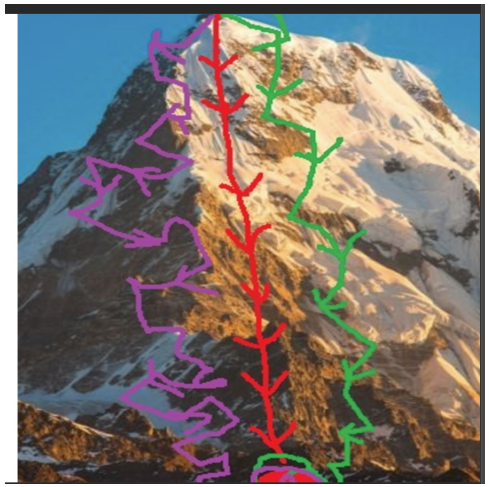
n: *number of samples*

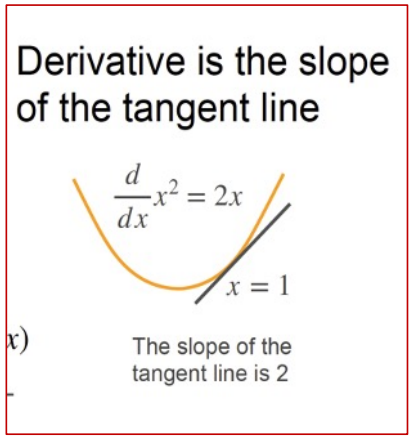


MSE is the average squared distance between the observed and predicted values

Mean Squared Error (MSE)

- Derivative is the slope of the tangent line





Recap: Binary classification

Input: $\mathbb{X}$

What is a good loss for our binary classification?

Target: $\mathbb{Y}$

Pixel Grid

We want our network to produce the right answer with high probability

Is it digit 2?

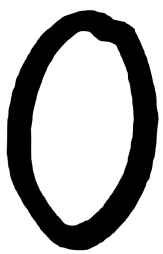
$x^{(1)} =$ 

28x28 pixels

Function: f

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

1. Make the network output a probability for class 1 (a value between 0 and 1)

$x^{(2)} =$ 

2. Use this probability to compute a loss

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

Entropy

- Entropy represents the surprise we would expect per coin toss if we flipped this coin a bunch of times.
- Suprise: $log_2(\frac{1}{(p(x)}$

	Heads	Tails
Probability p(x)	0.9	0.1
Surprise	0.15	3.32

- Entropy is the Expected Value of the Surprise.
- E(Surprise) = (0.9 * 0.15) + (0.1 * 3.32) = 0.47
- E(Surprise) = $\sum x * P(X = x)$, where x is the specific value for surprise and P(X=x) is the probablity of observing that specific value for Surprise.

Entropy

- Entropy = $\sum log(\frac{1}{(p(x)}) p(x)$
  - $log(\frac{1}{(p(x)})$ is the surprise.
  - $p(x)$ is the probability of the surprise.
- Entropy = $\sum p(x) log(\frac{1}{(p(x)}) = \sum p(x)[log(1) - log(p(x))]$
- Entropy = $-\sum p(x) log(p(x))$

Cross Entropy Loss (for Binary classification)

- y = true label of class (0 or 1)
- p = predicted probability of class 1
- Cross Entropy Loss = -(y * log(p) + (1-y) * log(1-p))

Recap: Multi-class classification

Input: $\mathbb{X}$

Classifying MNIST digits requires predicting 1 of 10 possible values

Target: $\mathbb{Y}$

Pixel Grid

$x^{(1)} =$ 

28x28 pixels

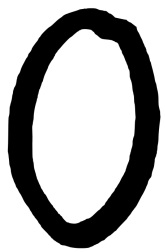
Function: f

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

Which digit is it?

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

How do we do that?

$x^{(2)} =$ 

Rather than predicting whether a handwritten digit is of a certain class, we predict the class it is most likely in

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

Cross Entropy Loss (for Multi-class classification

- Loss = $-\sum_{j=1}^{m} y_j * log(p_j)$

p	Classes (m)	y
0.3	"0"	0
0.2	"1"	0
0.5	"2"	1

We want model to assign high probability to the true class and low to others

Recap

- Perceptron training with working examples
- Multi-class classification
- MSE loss for regression
- Cross entropy loss for binary and multi-class classification