CSCI 1470

Eric Ewing

Tuesday, 9/9

Deep Learning

Day 2: Linear Regression and Perceptrons

Mt Washington, the tallest mountain in the northeast

Recap from Last Class

Machine Learning:

- Can we learn to approximate a function *f*?
- Deep Learning is machine learning with a specific class of functions (neural networks)

Some Notation

$\mathbb{R}$: The set of real numbers

$v \in \mathbb{R}^d$: A **vector** in dimension d

$V \in \mathbb{R}^{H \times W}$: A **matrix** of dimensions $H \times W$

$V \in \mathbb{R}^{H \times W \times C}$: A **tensor** of dimensions $H \times W \times C$

$\mathbb{X}$: A set of **input** data

$\mathbb{Y}$: A set of target variables (outputs/labels) for supervised learning

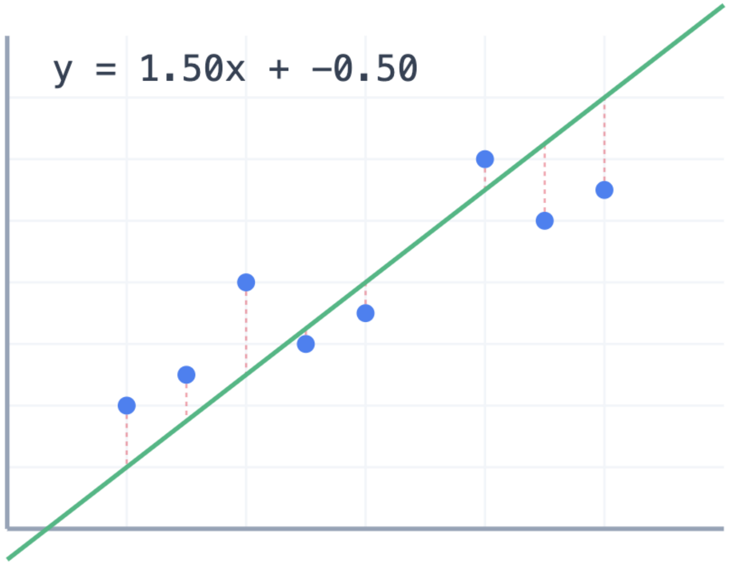
$x^{(k)}$: k'th example (input) from dataset

$y^{(k)}$: k'th example (output) associated with $x^{(k)}$

What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

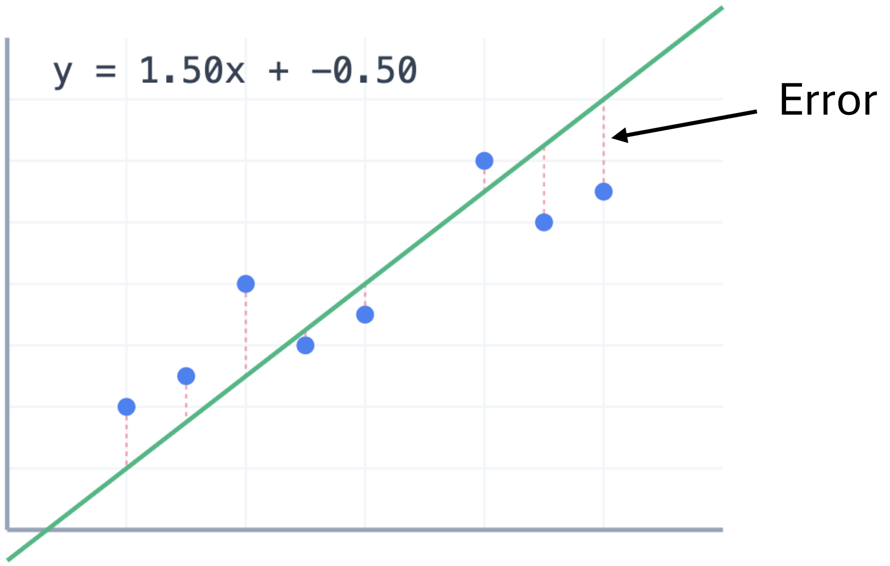
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

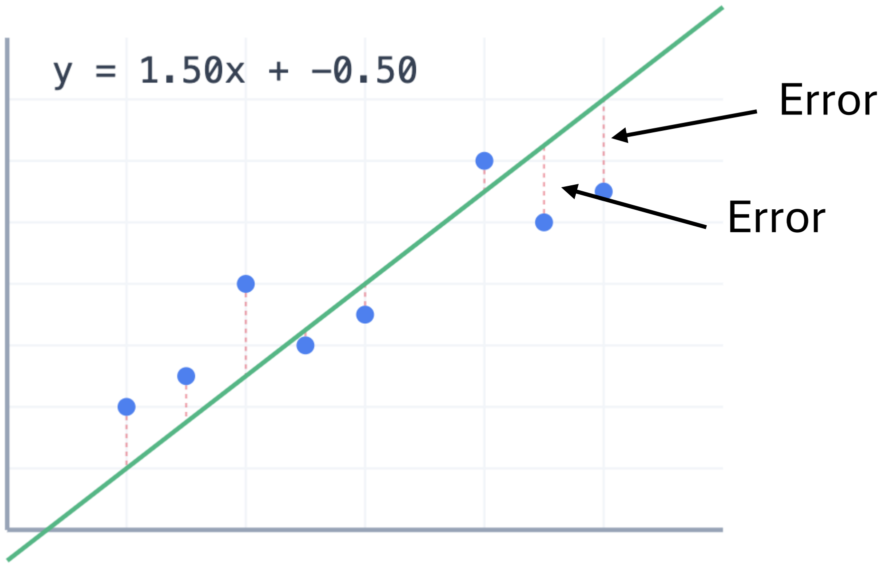
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

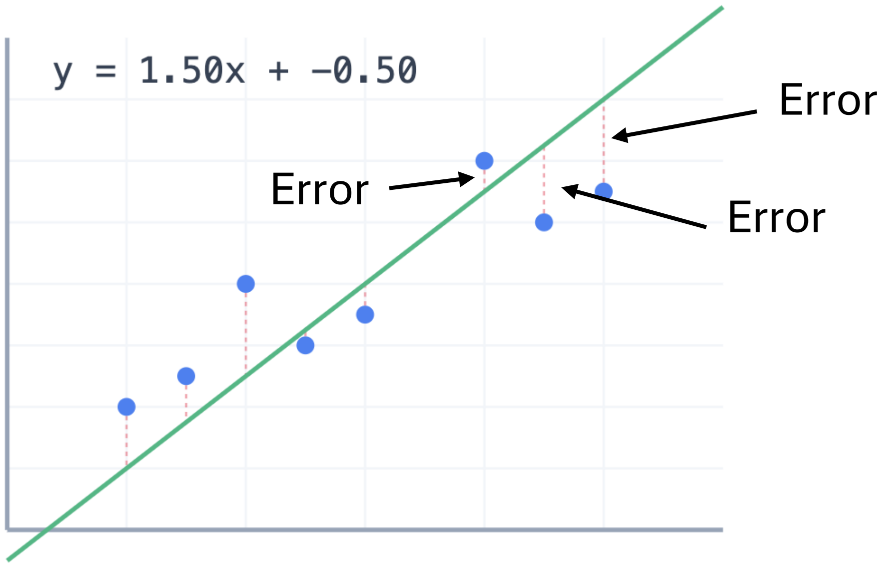
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

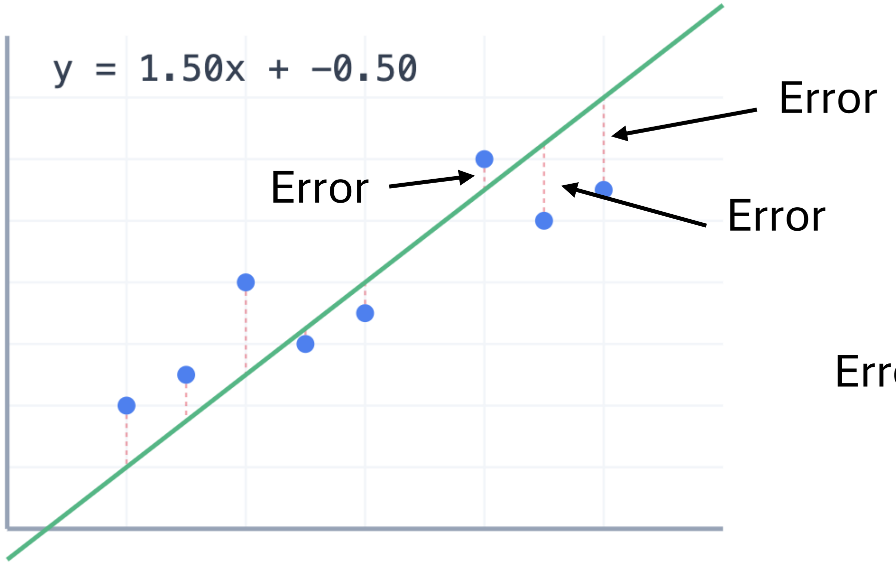
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



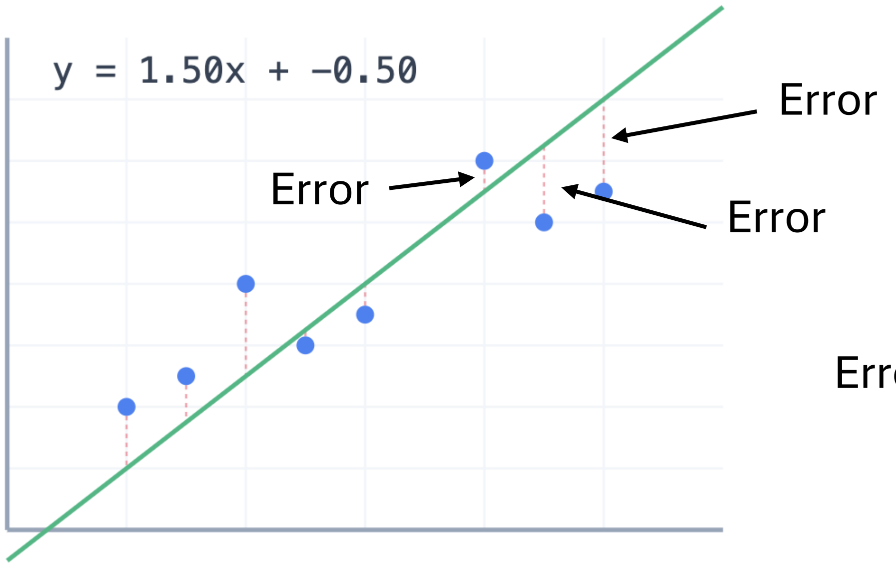
Error = $f\left(x^{(i)}\right) - y^{(i)}$

What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

The standard loss function for Linear Regression is **Mean Squared Error (MSE)**

$$MSE = \frac{\sum_i^n \left(f\left(x^{(i)}\right) - y^{(i)}\right)^2}{n}$$

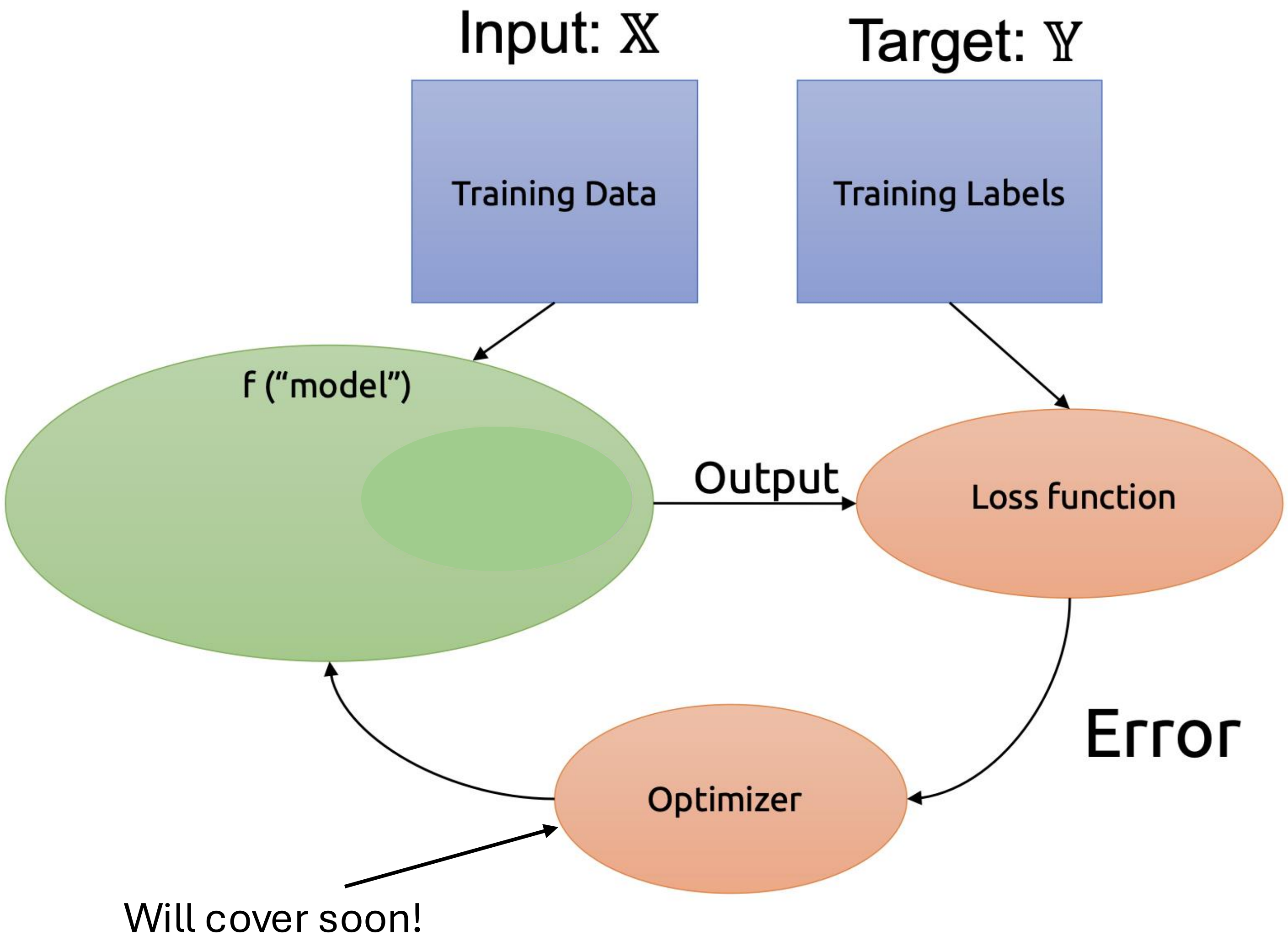


Error = $f\left(x^{(i)}\right) - y^{(i)}$

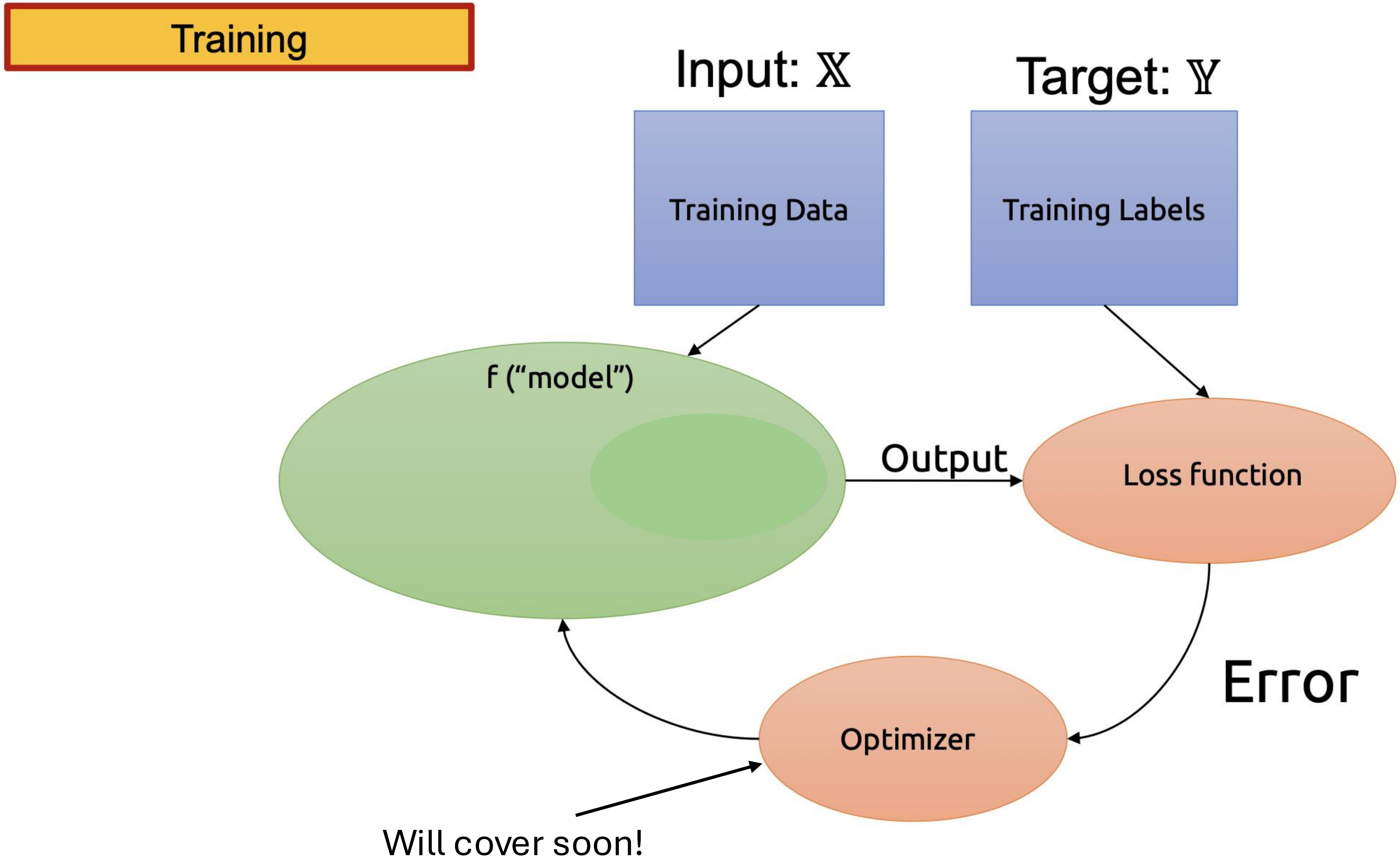
What is the best approximation?

- https://brown-deep-learning.github.io/dl-website-s25/visualizations/visualizations.html

“Classic” Supervised Learning in Machine Learning

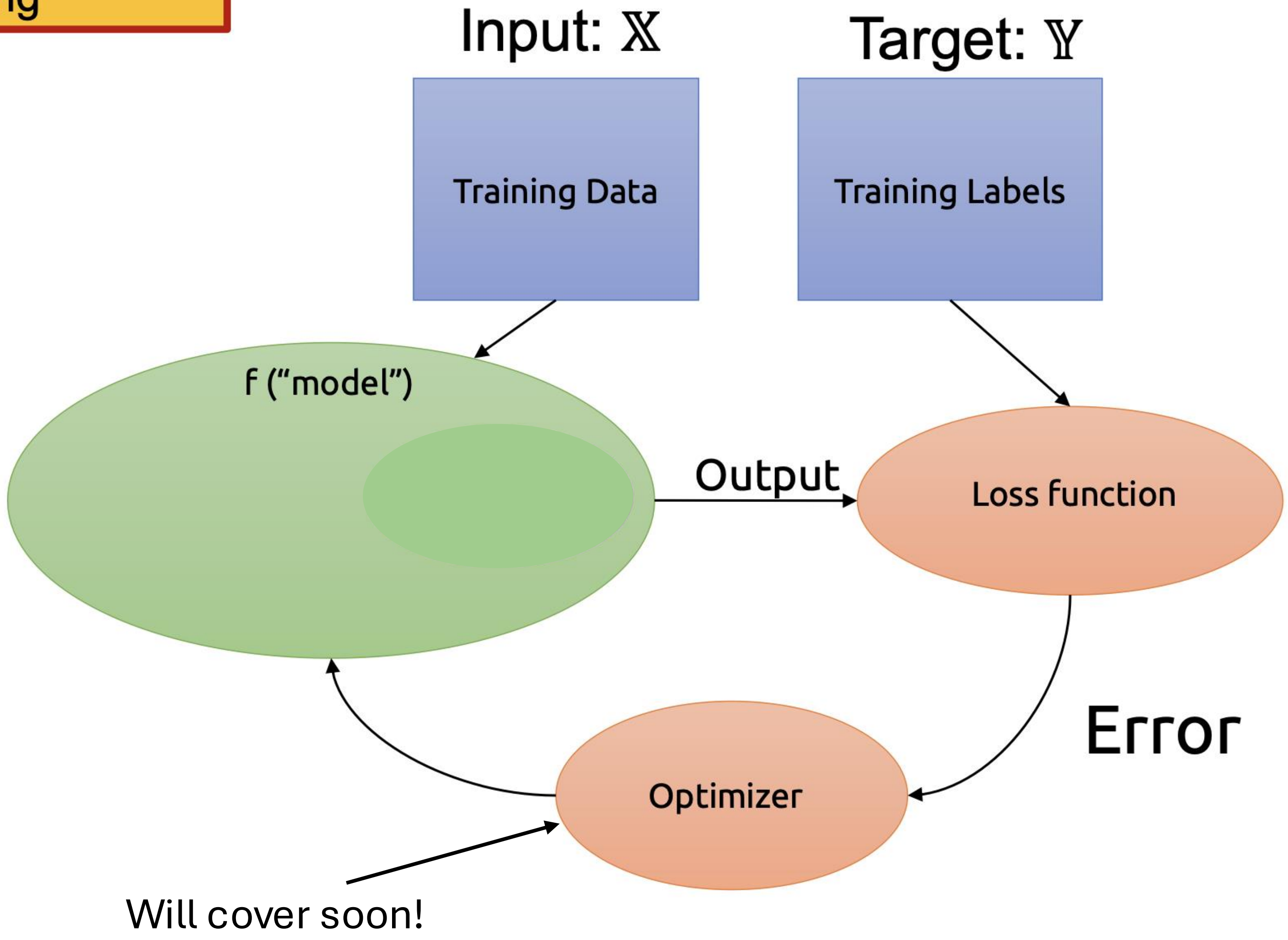


“Classic” Supervised Learning in Machine Learning



“Classic” Supervised Learning in Machine Learning

Training



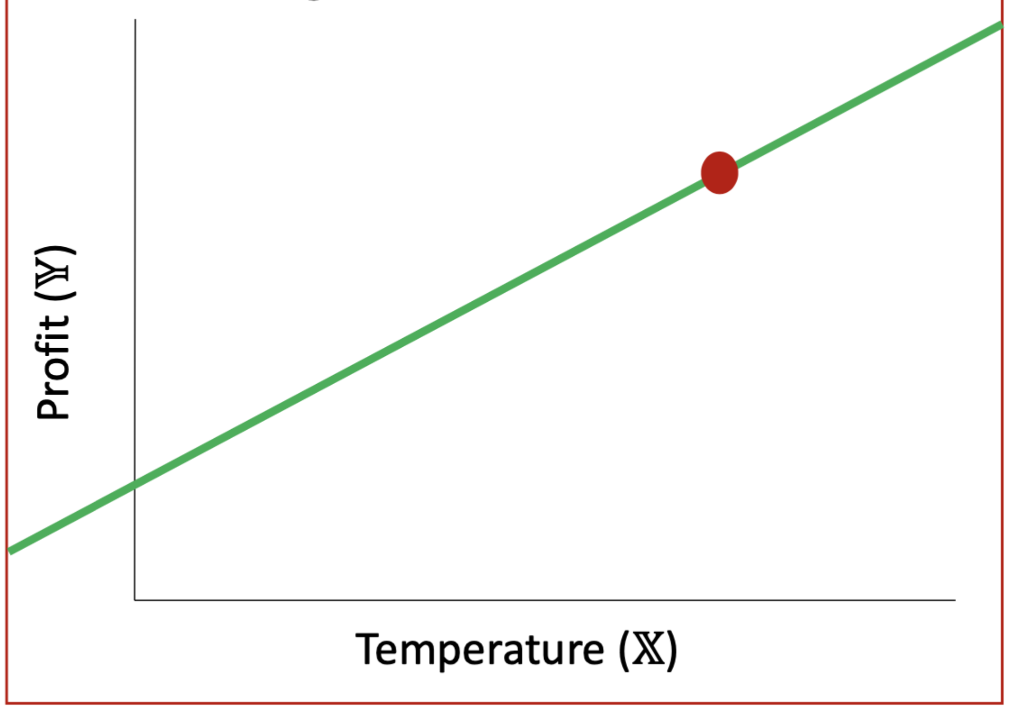
Testing our model

"Temperature"

"Profit made on selling lemonade"

Linear function

$$y = wx + b$$



$x' = 70$

Prediction

$y' = 175$

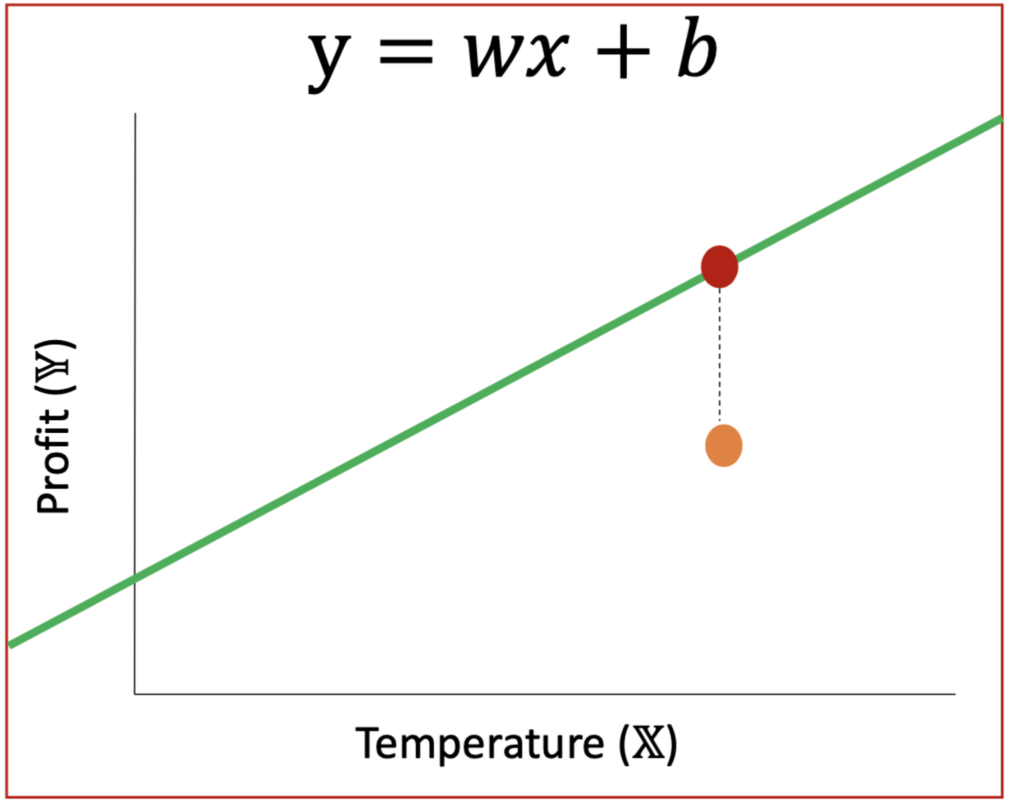
(Image only for explaining concept, not drawn accurately)

Testing our model

"Temperature"

"Profit made on selling lemonade"

Linear function



$x' = 70$

$\hat{x} = 70$

Prediction

$y' = 175$

True observation

$\hat{y} = 140$

(Image only for explaining concept, not drawn accurately)

Learning better models – Collect more data

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

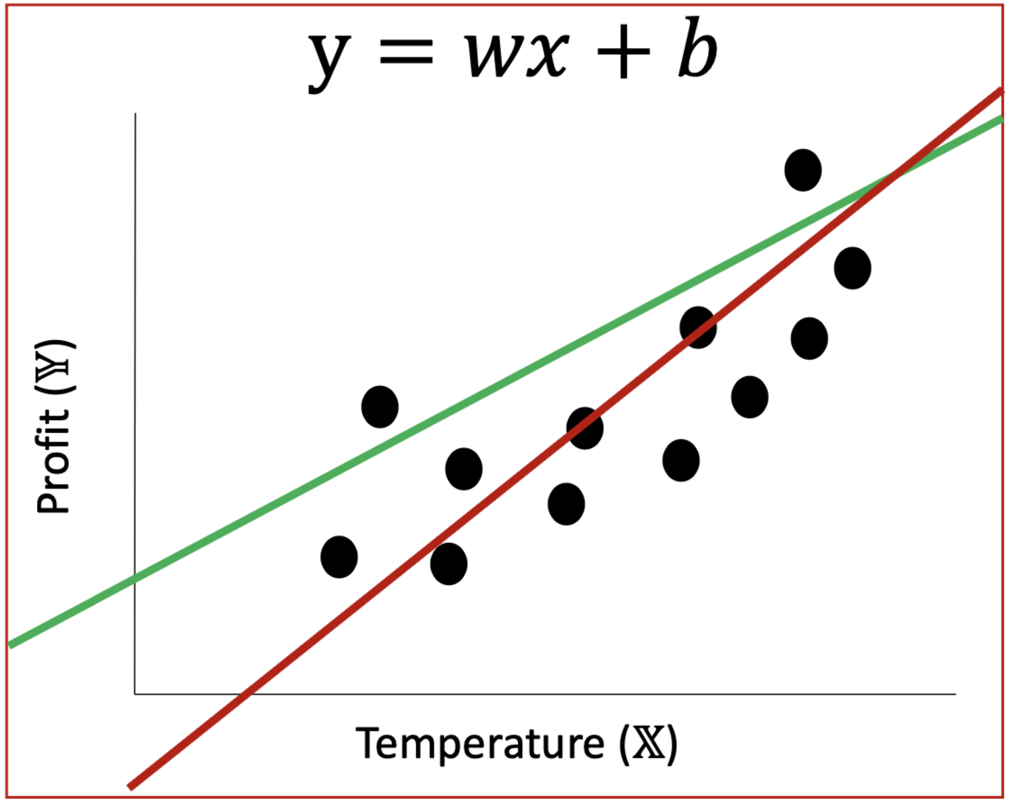
$x^{(3)} = 30.3$

.
.
.
.

$x^{N} = \ldots$

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

Linear function



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

.
.
.
.

$y^{N} = \ldots$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

Learning better models – Try different functions

Input: $\mathbb{X}$

"Temperature"

$x^{(1)} = 100.1$

$x^{(2)} = 80.0$

$x^{(3)} = 30.3$

.

.

.

.

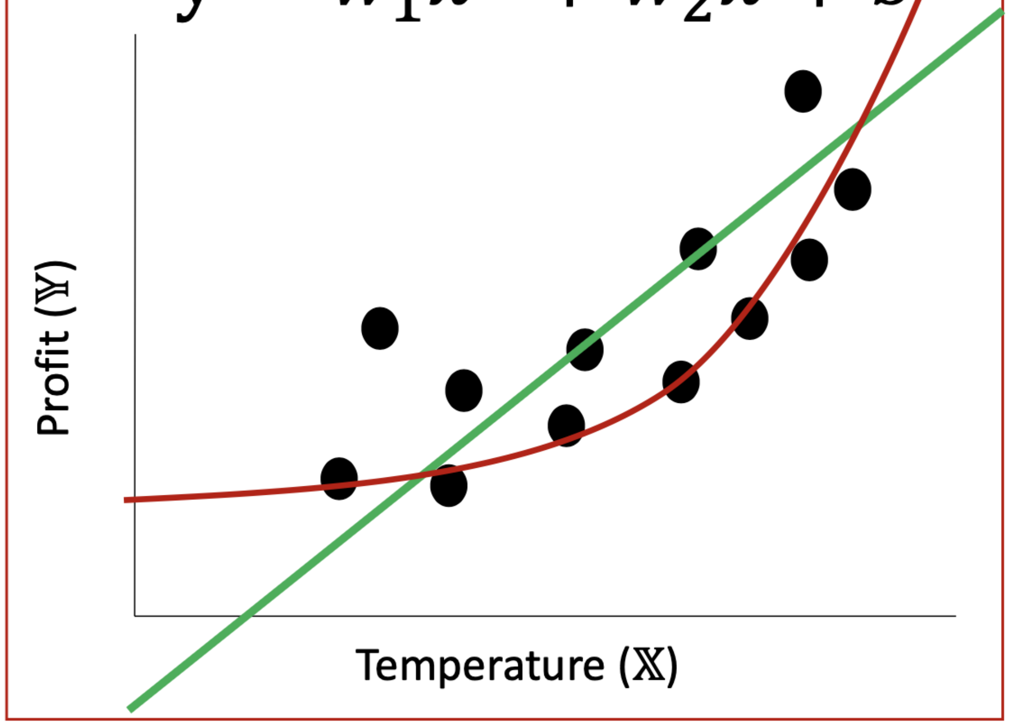
$x^{N} = \quad ...$

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

Non-linear function

Polynomial function

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



Target: $\mathbb{Y}$

"Profit made on selling lemonade"

$y^{(1)} = 200.0$

$y^{(2)} = 180.5$

$y^{(3)} = 115.1$

.

.

.

.

$y^{N} = \quad ...$

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

(Numerical output)

(Image only for explaining concept, not drawn accurately)

How to know which function is the best?

$$y = wx + b$$

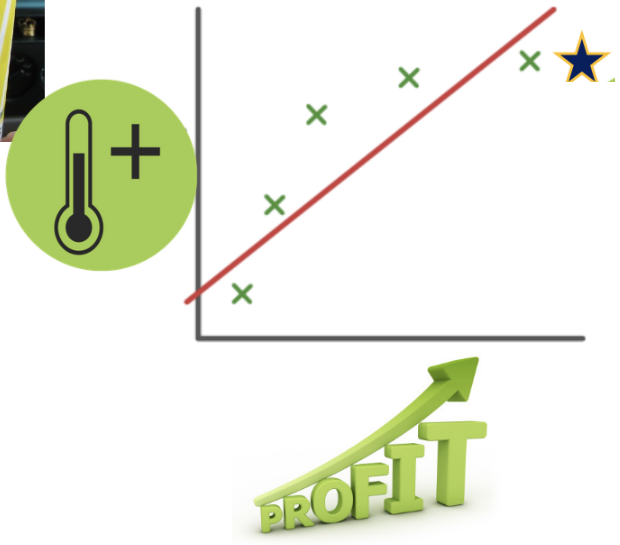
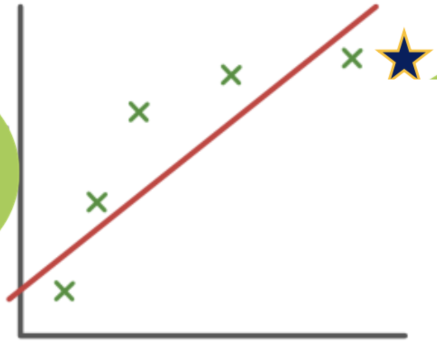


Image courtesy: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

How to know which function is the best?

$$y = wx + b$$

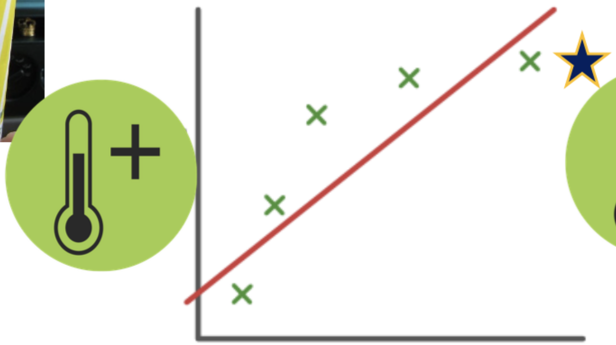


"My model is not doing that well on the given data and new data" ☹

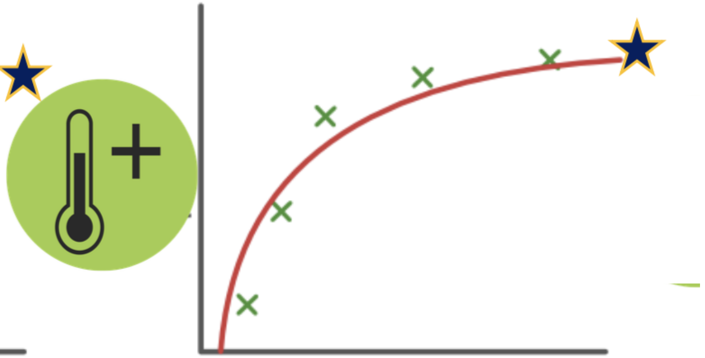
Image courtesy: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

How to know which function is the best?

$y = wx + b$



$y = w_1x^2 + w_2x + b$

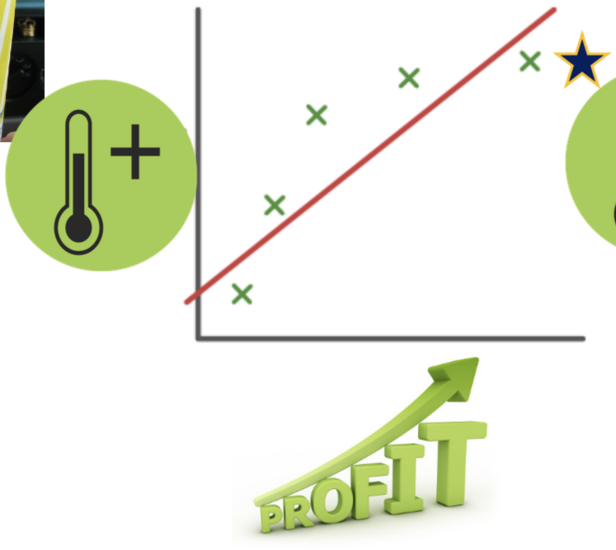


"My model is not doing that well on the given data and new data" ☹

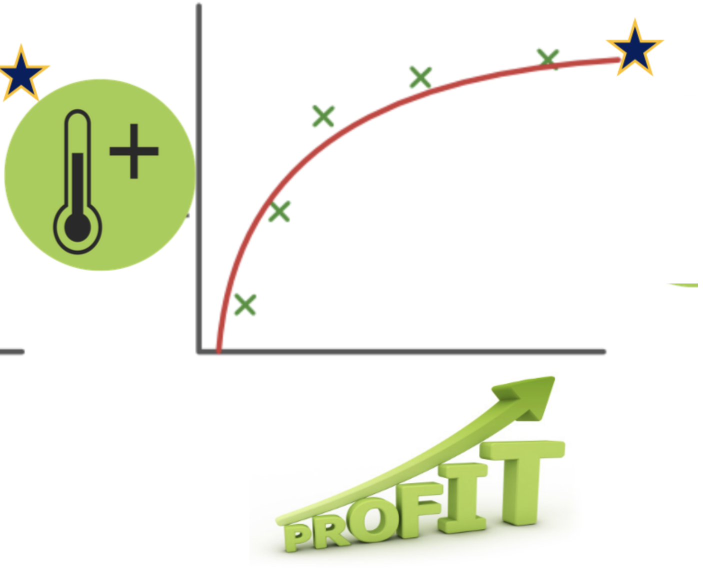
Image courtesy: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

How to know which function is the best?

$y = wx + b$



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



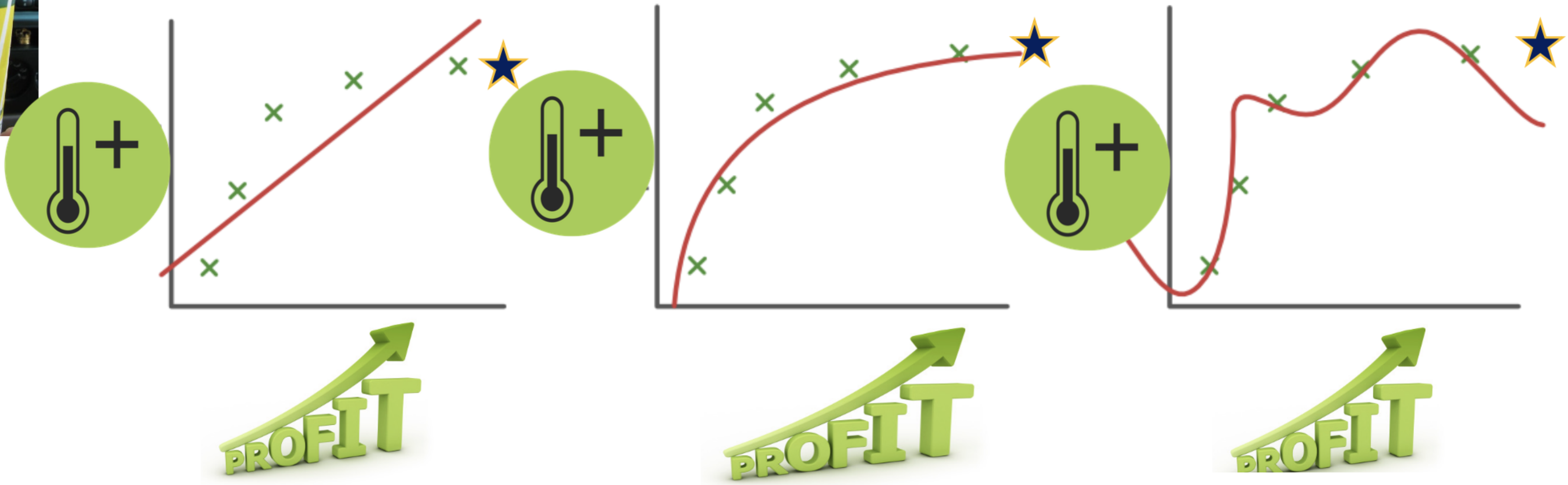
"My model is not doing that well on the given data and new data" ☹

"My model is doing well on the given data AND the new data point!! ☺

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

How to know which function is the best?

$y = wx + b$ $\qquad$ $y = w_1x^2 + w_2x + b$ $\qquad$ $y = w_1x^4 + w_2x^3 + w_3x^2 + w_4x + b$



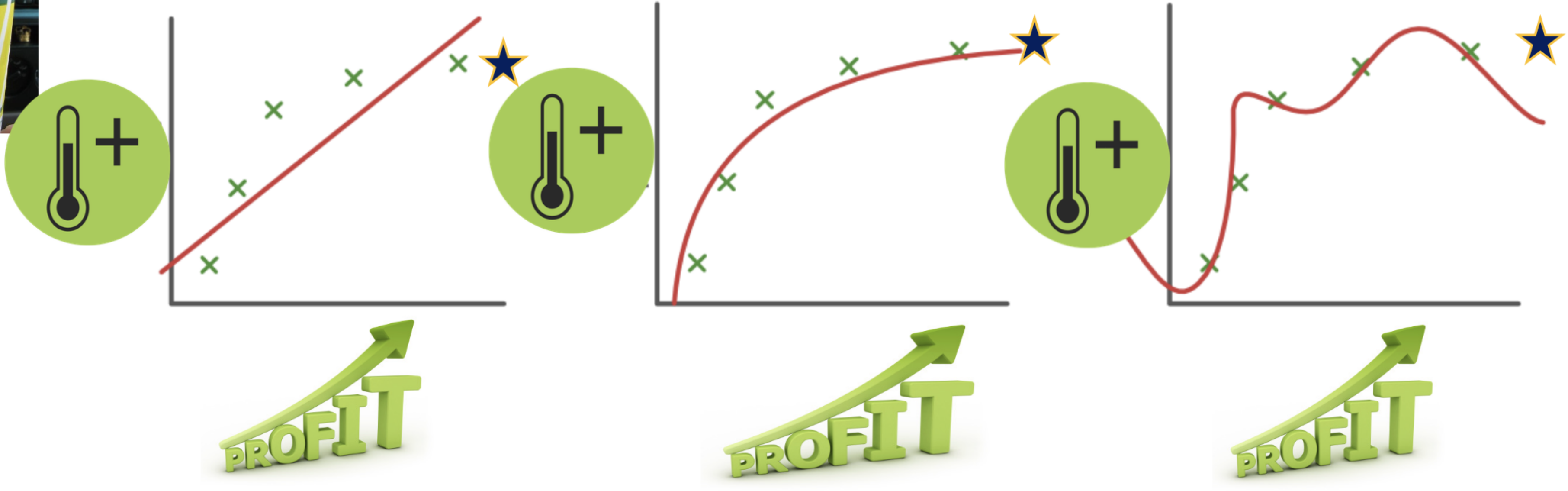
"My model is not doing that well on the given data and new data" ☹

"My model is doing well on the given data AND the new data point!! ☺

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

How to know which function is the best?

$y = wx + b$ $\qquad$ $y = w_1x^2 + w_2x + b$ $\qquad$ $y = w_1x^4 + w_2x^3 + w_3x^2 + w_4x + b$



"My model is not doing that well on the given data and new data" ☹

"My model is doing well on the given data AND the new data point!! ☺

"My model is doing really well on the given data!! ☺

"The performance is bad on new data point" ☹

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

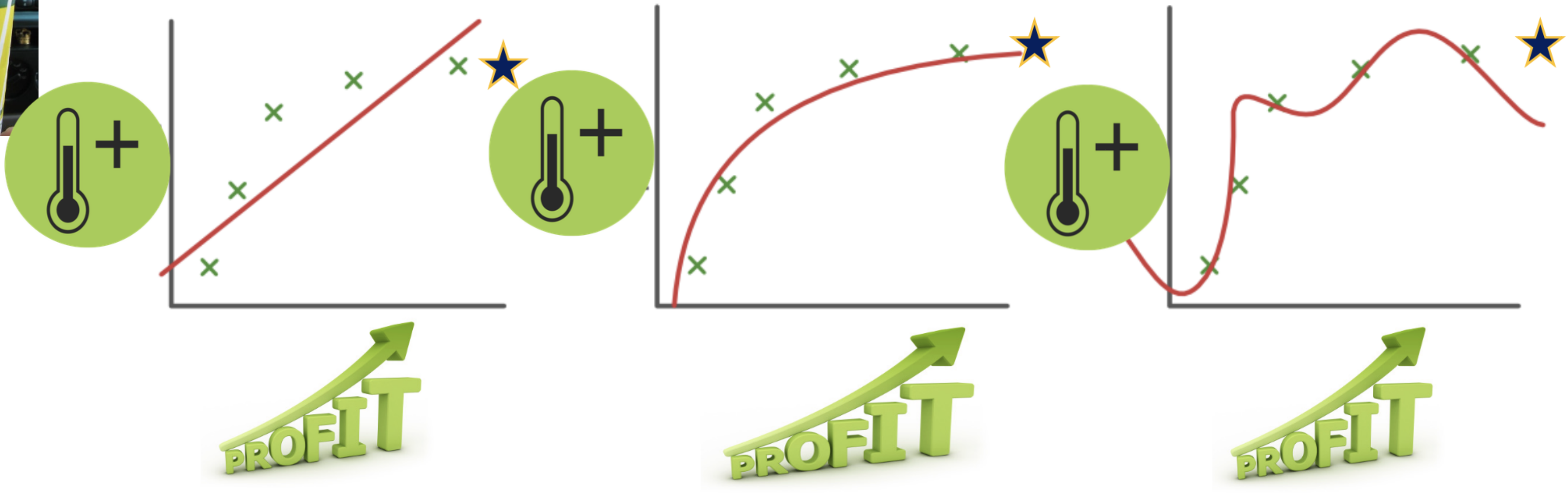
How to know which function is the best?

Underfit

$y = wx + b$

$y = w_1x^2 + w_2x + b$

$y = w_1x^4 + w_2x^3 + w_3x^2 + w_4x + b$



"My model is not doing that well on the given data and new data" ☹

"My model is doing well on the given data AND the new data point!! ☺

"My model is doing really well on the given data!! ☺

"The performance is bad on new data point" ☹

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

How to know which function is the best?

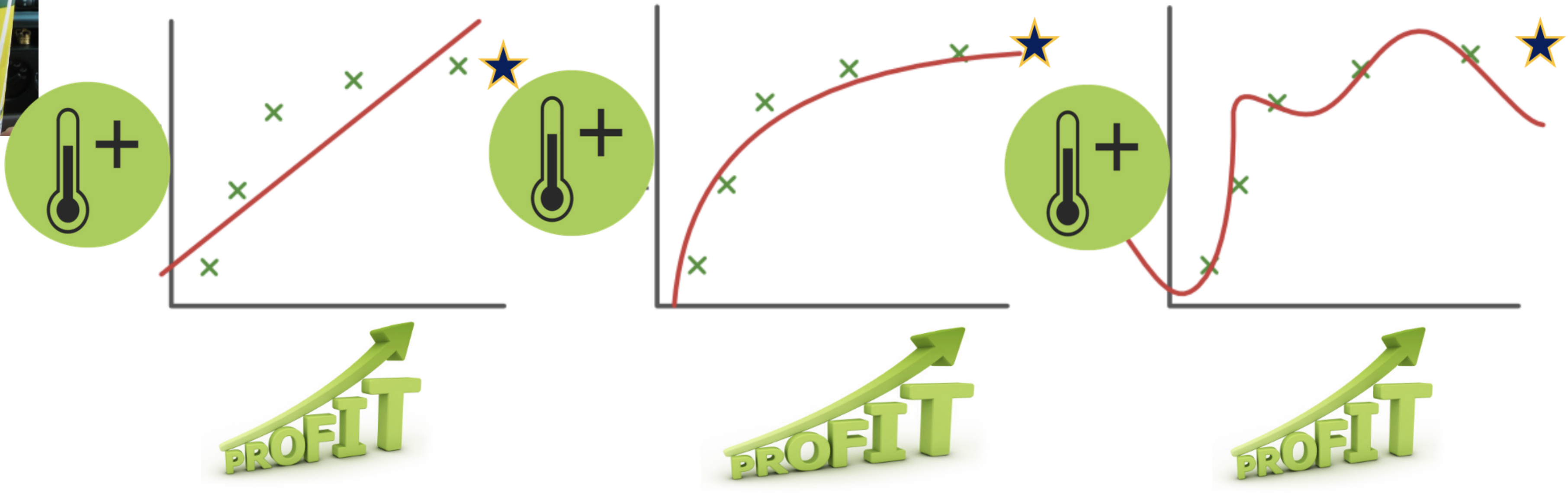
Underfit

Overfit

$y = wx + b$

$y = w_1x^2 + w_2x + b$

$y = w_1x^4 + w_2x^3 + w_3x^2 + w_4x + b$



"My model is not doing that well on the given data and new data" ☹

"My model is doing well on the given data AND the new data point!! ☺

"My model is doing really well on the given data!! ☺

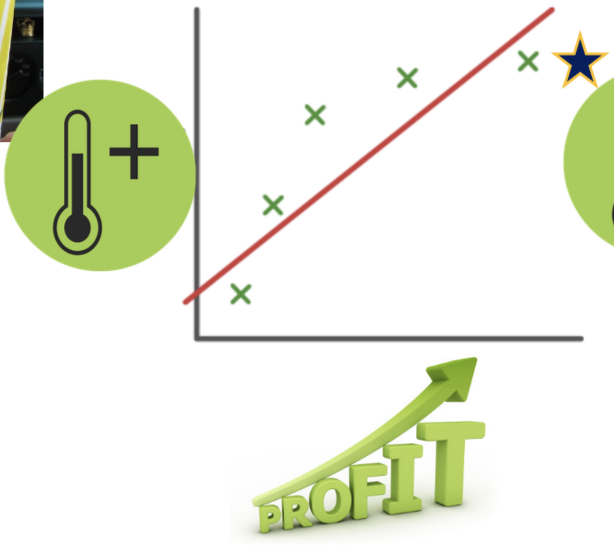
"The performance is bad on new data point" ☹

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

How to know which function is the best?

Underfit

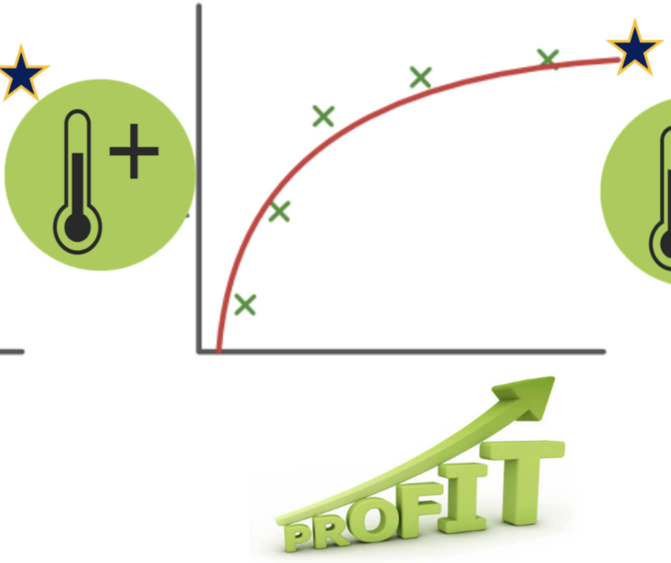
$y = wx + b$



"My model is not doing that well on the given data and new data" ☹

Good fit

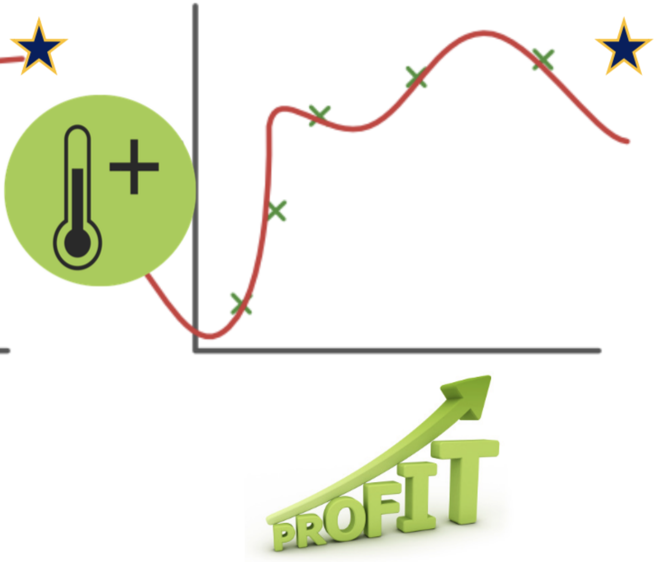
$y = w_1x^2 + w_2x + b$



"My model is doing well on the given data AND the new data point!! ☺

Overfit

$y = w_1x^4 + w_2x^3 + w_3x^2 + w_4x + b$



"My model is doing really well on the given data!! ☺

"The performance is bad on new data point" ☹

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

Model Complexity

- Model complexity refers to… the model’s complexity
  - Polynomial regressions are more complex than linear regressions
- Models with higher complexity can approximate more function types well
- More complex functions also **tend** to overfit

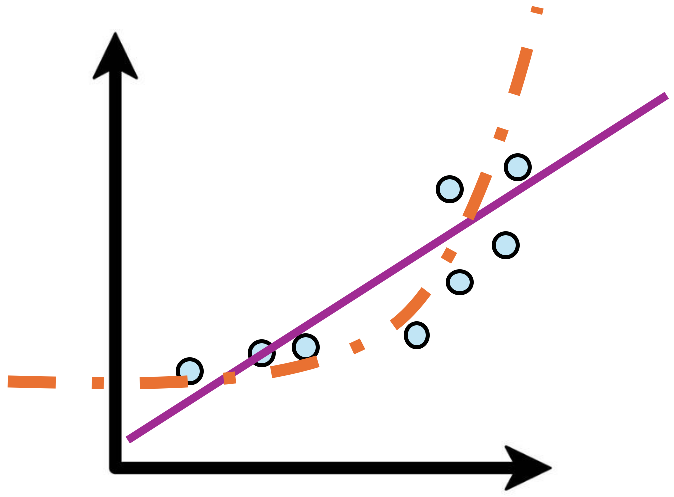
Important Question: A 100 degree polynomial tends to be way overfit. Neural Networks will be even more complex, why do neural networks not overfit?

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
$x^{(5)}$
$x^{(6)}$
$x^{(7)}$
$x^{(8)}$

f_1 or f_2?

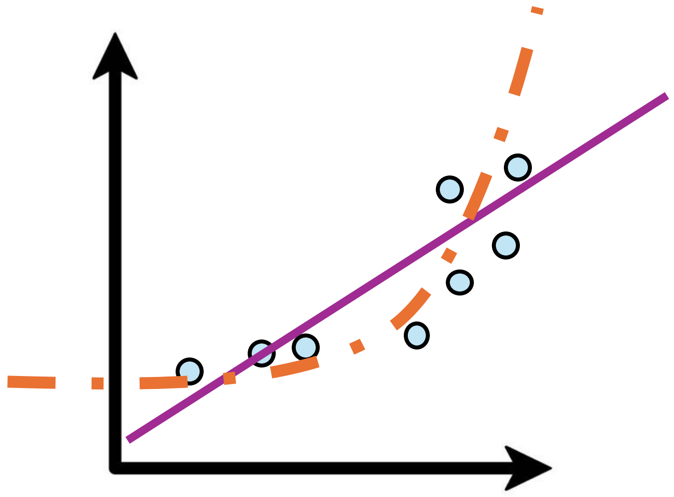


How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
$x^{(5)}$
$x^{(6)}$
$x^{(7)}$
$x^{(8)}$

f_1 or f_2?



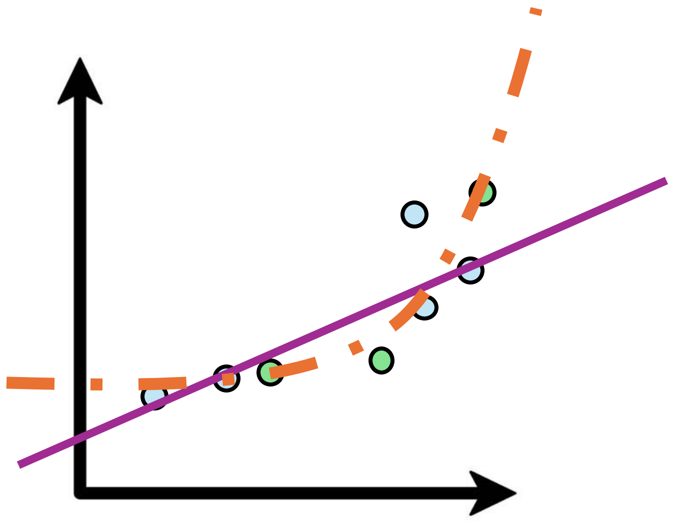
Compare MSE between them?

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$	
$x^{(2)}$	
$x^{(3)}$	Training Set
$x^{(4)}$	
$x^{(5)}$	
$x^{(6)}$	
$x^{(7)}$	Test Set
$x^{(8)}$	

f_1 or f_2?

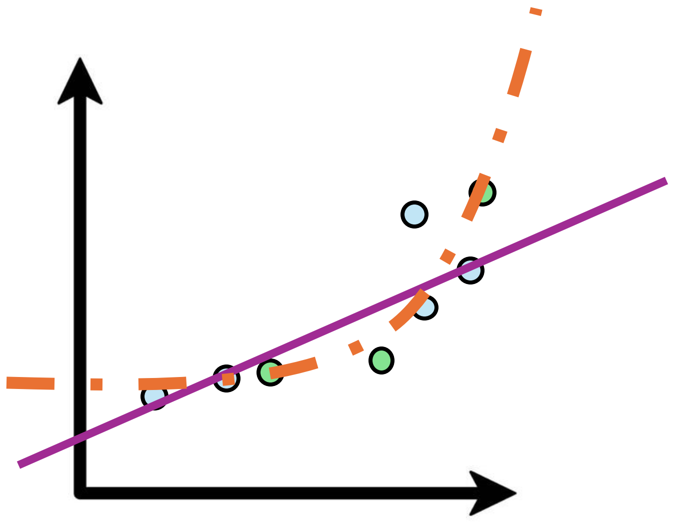


How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$	
$x^{(2)}$	
$x^{(3)}$	Training Set
$x^{(4)}$	
$x^{(5)}$	
$x^{(6)}$	
$x^{(7)}$	Test Set
$x^{(8)}$	

f_1 or f_2?



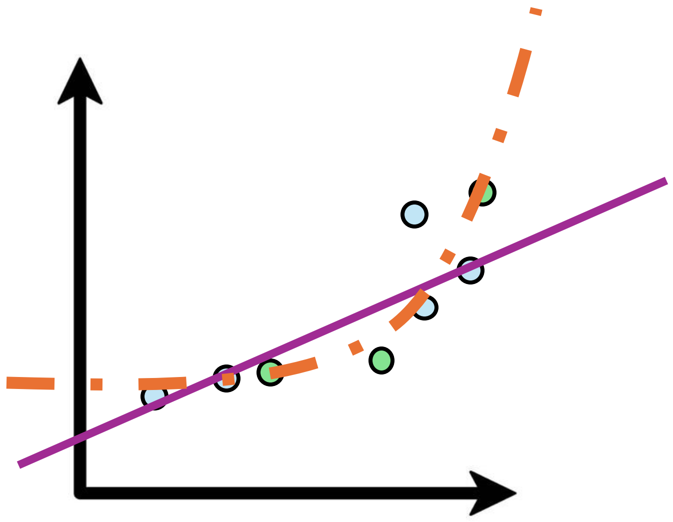
Compare MSE on what data?

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$	
$x^{(2)}$	
$x^{(3)}$	Training Set
$x^{(4)}$	
$x^{(5)}$	
$x^{(6)}$	
$x^{(7)}$	Test Set
$x^{(8)}$	

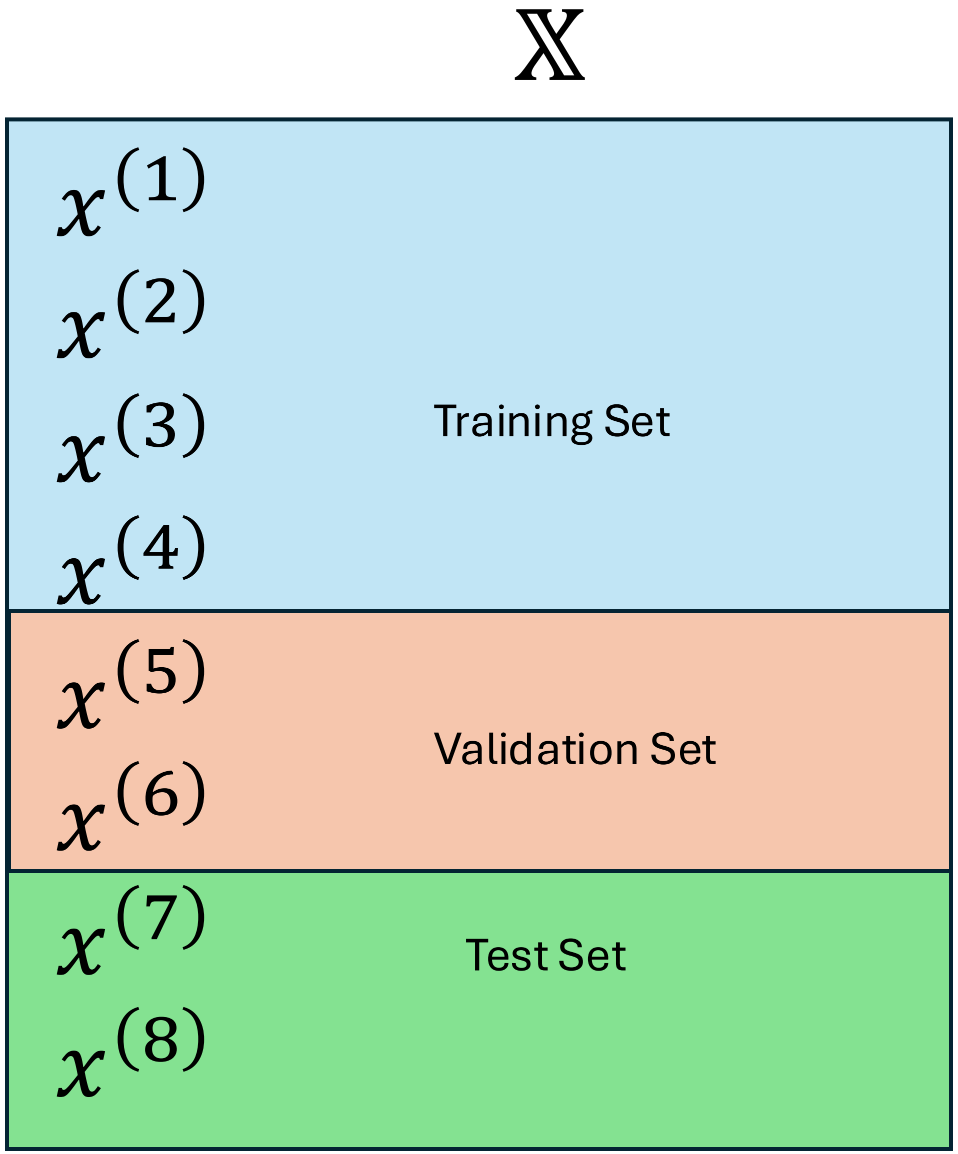
f_1 or f_2?



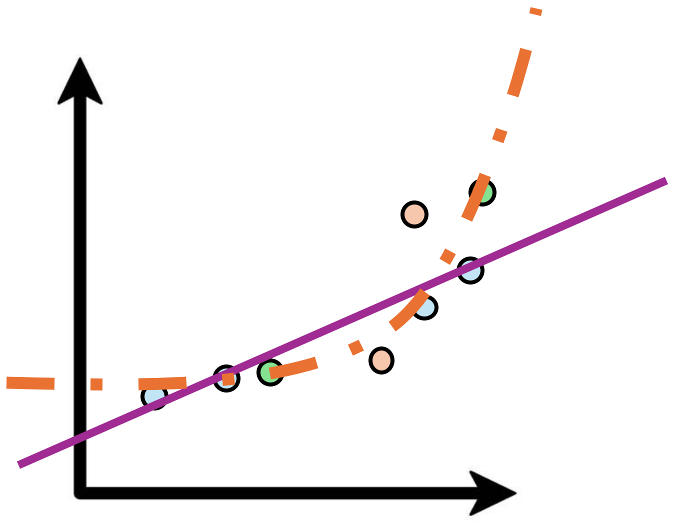
Compare MSE on what data?

When might we want to overfit?

How to know which function is the best?

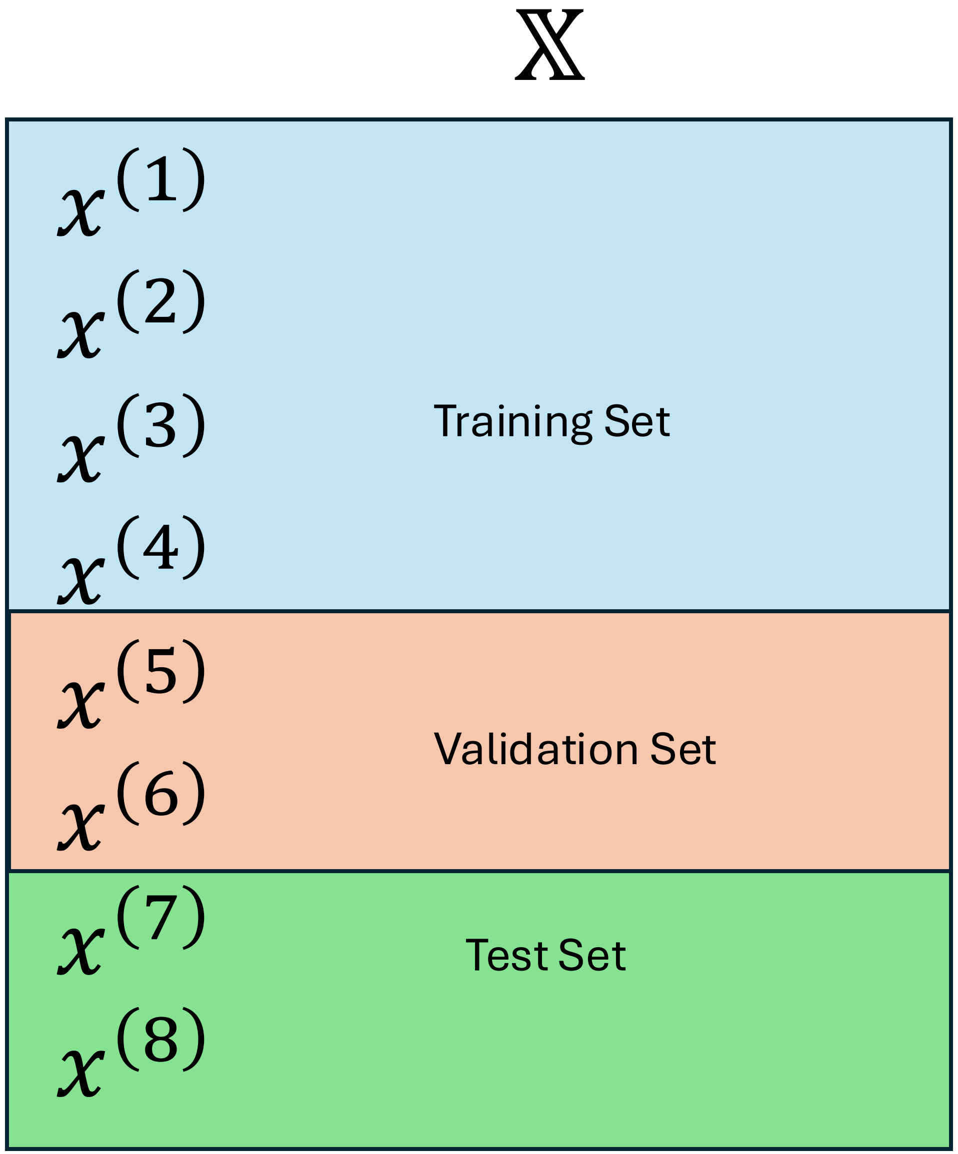


f_1 or f_2?



1. Train model on training set
2. Validate performance on validation set
3. Report results on test set

How to know which function is the best?



In this class

1. Train model on provided training data
2. Validate your model locally with validation set
3. Submit to Gradescope and we have a separate test set

In real world

1. Train model on provided training data
2. Validate your model locally with validation set
3. Deploy your model to real world and track performance

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$	Training Set
$x^{(5)}$ $x^{(6)}$	Validation Set
$x^{(7)}$ $x^{(8)}$	Test Set

In this class

1. Train model on provided training data
2. Validate your model locally with validation set
3. Submit to Gradescope and we have a separate test set

In real world

1. Train model on provided training data
2. Validate your model locally with validation set
3. Deploy your model to real world and track performance

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	$200
$x^{(2)}$	80	No	Mon	$91
$x^{(3)}$	62	No	Wed	$54

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	$200
$x^{(2)}$	80	No	Mon	$91
$x^{(3)}$	62	No	Wed	$54

How can we represent binary variables?

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	$200
$x^{(2)}$	80	No	Mon	$91
$x^{(3)}$	62	No	Wed	$54

How can we represent binary variables?

$$x_2^{(k)} \in \{0, 1\}$$

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	$200
$x^{(2)}$	80	No	Mon	$91
$x^{(3)}$	62	No	Wed	$54

How can we represent binary variables?

How can we represent categorical variables?

$$x_2^{(k)} \in \{0, 1\}$$

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	\$200
$x^{(2)}$	80	No	Mon	\$91
$x^{(3)}$	62	No	Wed	\$54

How can we represent binary variables?

How can we represent categorical variables?

Idea 1: Mon=0, Tue=1, Wed.=2

$$x_2^{(k)} \in \{0, 1\}$$

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	$200
$x^{(2)}$	80	No	Mon	$91
$x^{(3)}$	62	No	Wed	$54

How can we represent binary variables?

How can we represent categorical variables?

$$x_2^{(k)} \in \{0, 1\}$$

Idea 1: Mon=0, Tue=1, Wed.=2

The problem: Is Wednesday being 2x Tuesday meaningful?
Why use this ordering and not a random ordering?

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	\$200
$x^{(2)}$	80	No	Mon	\$91
$x^{(3)}$	62	No	Wed	\$54

How can we represent binary variables?

$$x_2^{(k)} \in \{0, 1\}$$

How can we represent categorical variables?

Idea 1: Mon=0, Tue=1, Wed.=2

The problem: Is Wednesday being 2x Tuesday meaningful?
Why use this ordering and not a random ordering?

Idea 2: Use a series of binary variables
If day==Mon, x_4=1, else 0
If day==Tue, x_5=1, else 0
...

Other ways to improve performance

Collect additional information

	x_1 Temperature	x_2 Sunny?	x_3 Day of Week	y Profit
$x^{(1)}$	90	Yes	Sat	\$200
$x^{(2)}$	80	No	Mon	\$91
$x^{(3)}$	62	No	Wed	\$54

How can we represent binary variables?

$x_2^{(k)} \in \{0, 1\}$

How can we represent categorical variables?

Idea 1: Mon=0, Tue=1, Wed.=2

The problem: Is Wednesday being 2x Tuesday meaningful?
Why use this ordering and not a random ordering?

Idea 2: Use a series of binary variables
If day==Mon, x_4=1, else 0
If day==Tue, x_5=1, else 0
…

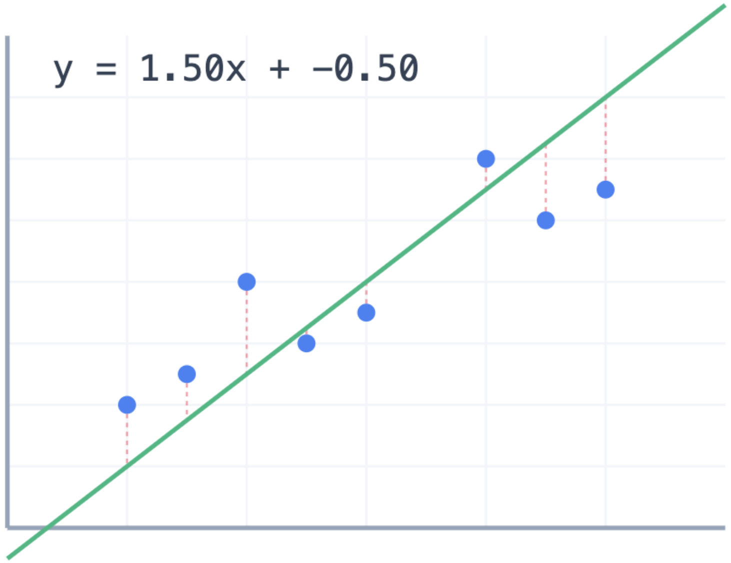
"One-Hot Vector": Turn categorical variables into a vector of binary variables

Weekly “Participation Quiz”

Available on Gradescope, closes at 11:59pm the day of class, but we also provide time in class to complete it.

Linear Regression

$$y = mx + b$$

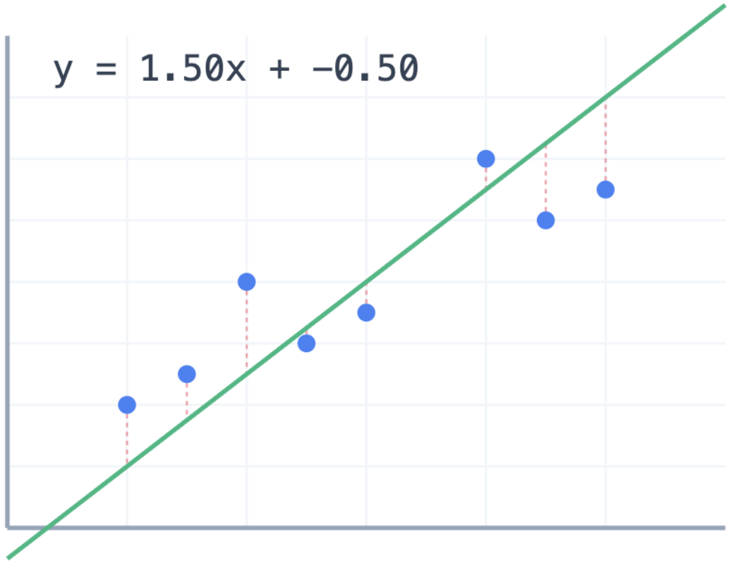


With 1 input feature, 2 ***parameters***

- m (slope)
- b (bias)

Linear Regression

$$y = mx + b$$



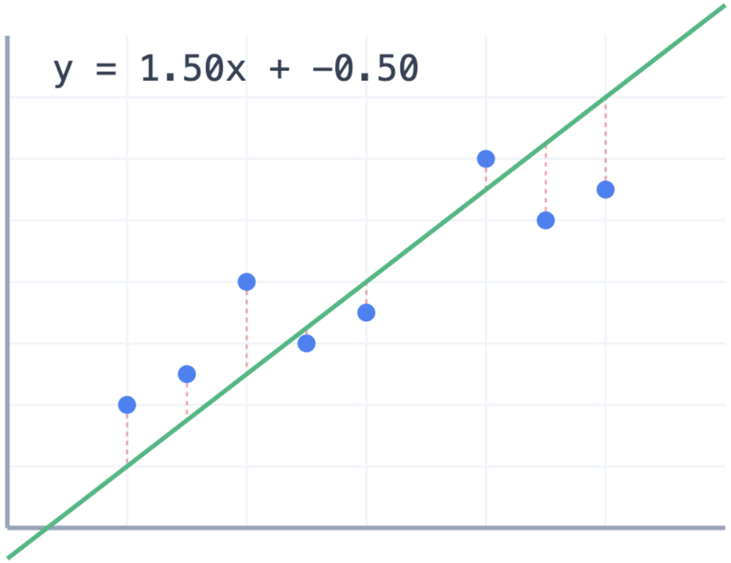
With 1 input feature, 2 ***parameters***

- m (slope)
- b (bias)

	Input Features			Output Target
	x_1	x_2	x_3	y
	Temperature	Sunny?	Day of Week	Profit
$x^{(1)}$	90	Yes	Sat	\$200
$x^{(2)}$	80	No	Mon	\$91
$x^{(3)}$	62	No	Wed	\$54

Linear Regression

$$y = mx + b$$



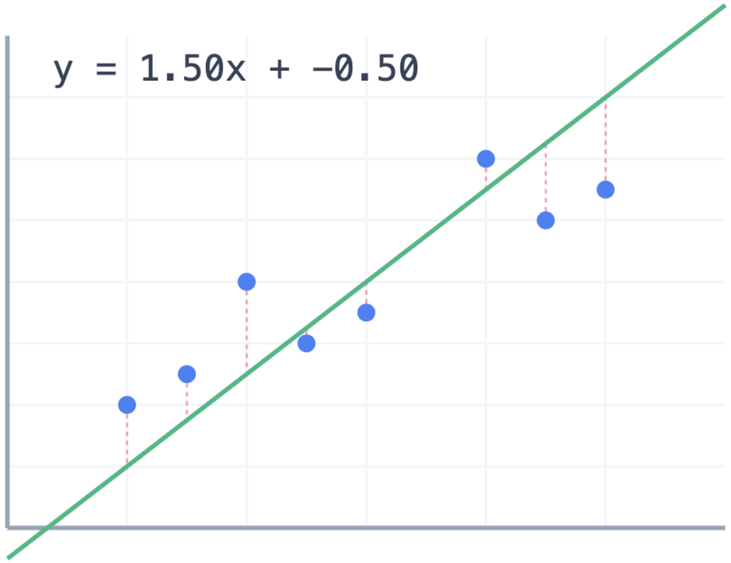
With 1 input feature, 2 ***parameters***

- m (slope)
- b (bias)

	Input Features				Output Target
	x_1	x_2	x_3	x_4	y
	Temperature	Sunny?	Day of Week	Constant	Profit
$x^{(1)}$	90	Yes	Sat	1	\$200
$x^{(2)}$	80	No	Mon	1	\$91
$x^{(3)}$	62	No	Wed	1	\$54

Linear Regression

$$y = mx + b$$



With 1 input feature, 2 ***parameters***

- m (slope)
- b (bias)

	Input Features				Output Target
	x_1	x_2	x_3	x_4	y
	Temperature	Sunny?	Day of Week	Constant	Profit
$x^{(1)}$	90	Yes	Sat	1	\$200
$x^{(2)}$	80	No	Mon	1	\$91
$x^{(3)}$	62	No	Wed	1	\$54

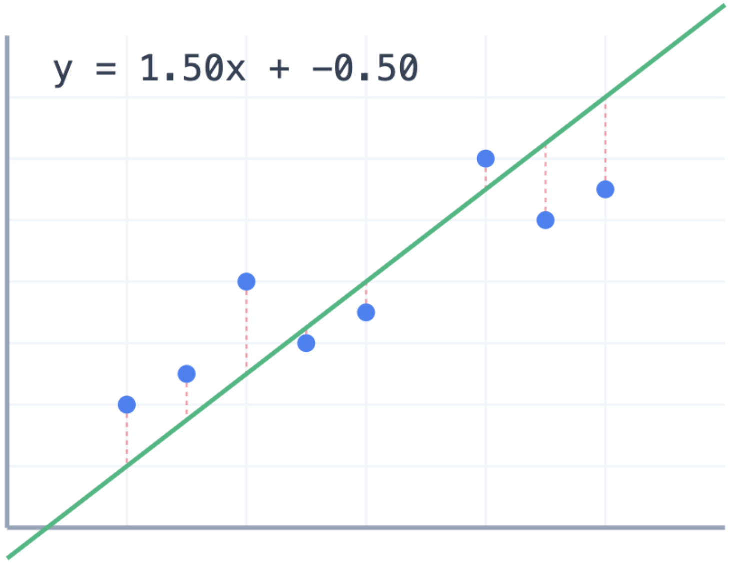
With multiple input features:

- Need a weight parameter w_i for each feature $\mathrm{x_i}$
- $y = x_1^{(i)} \cdot w_1 + x_2^{(i)} \cdot w_2^{(i)} + \cdots + x_d^{(i)} \cdot w_d$
- Can be rewritten: $y = \vec{x} \cdot \vec{w}$

Linear Regression

How do we find optimal parameter values?

$$y = mx + b$$



With 1 input feature, 2 ***parameters***

- m (slope)
- b (bias)

	Input Features				Output Target
	x_1	x_2	x_3	x_4	y
	Temperature	Sunny?	Day of Week	Constant	Profit
$x^{(1)}$	90	Yes	Sat	1	$200
$x^{(2)}$	80	No	Mon	1	$91
$x^{(3)}$	62	No	Wed	1	$54

With multiple input features:

- Need a weight parameter w_i for each feature $\mathrm{x_i}$
- $y = x_1^{(i)} \cdot w_1 + x_2^{(i)} \cdot w_2^{(i)} + \cdots + x_d^{(i)} \cdot w_d$
- Can be rewritten: $y = \vec{x} \cdot \vec{w}$

Option 1: Closed Form Solution

Goal: Minimize *Loss* function

Process:

- Find derivative (or gradient) of loss function
- Set derivative to 0
- Solve for parameters

Option 1: Closed Form Solution

Goal: Minimize *Loss* function

MSE (Mean Squared Error)

Process:

- Find derivative (or gradient) of loss function
- Set derivative to 0
- Solve for parameters

Option 1: Closed Form Solution

Goal: Minimize *Loss* function

MSE (Mean Squared Error)

Process:

- Find derivative (or gradient) of loss function
- Set derivative to 0
- Solve for parameters

Generalization of derivatives to functions with multiple inputs

Option 1: Closed Form Solution

Goal: Minimize *Loss* function

MSE (Mean Squared Error)

Process:

- Find derivative (or gradient) of loss function
- Set derivative to 0
- Solve for parameters

Generalization of derivatives to functions with multiple inputs

Is this guaranteed to find the global best parameter settings?

Option 1: Closed Form Solution

Goal: Minimize *Loss* function

MSE (Mean Squared Error)

Process:

- Find derivative (or gradient) of loss function
- Set derivative to 0
- Solve for parameters

Generalization of derivatives to functions with multiple inputs

Is this guaranteed to find the global best parameter settings?

weight vector $w \in \mathbb{R}^d$

Gradients

The gradient of a function *f* is a vector of partial derivatives:

$$\nabla f_\theta = [\frac{\partial f}{\theta_1}, \frac{\partial f}{\theta_2}, \frac{\partial f}{\theta_3}, \ldots, \frac{\partial f}{\theta_d}]$$

Gradients

The gradient of a function *f* is a vector of partial derivatives:

$$\nabla f_\theta = [\frac{\partial f}{\theta_1}, \frac{\partial f}{\theta_2}, \frac{\partial f}{\theta_3}, \ldots, \frac{\partial f}{\theta_d}]$$

For a linear regression model with one input variable what dimension is ∇f_θ in?

$\nabla f_\theta \in \mathbb{R}^?$

Gradients

The gradient of a function *f* is a vector of partial derivatives:

$$\nabla f_\theta = [\frac{\partial f}{\theta_1}, \frac{\partial f}{\theta_2}, \frac{\partial f}{\theta_3}, \dots, \frac{\partial f}{\theta_d}]$$

For a linear regression model with one input variable what dimension is ∇f_θ in?

$\nabla f_\theta \in \mathbb{R}^?$

∇f_w tells us what happens to *f* with small adjustments to each parameter *w*

Option 1: Closed Form Solution

Option 1: Closed Form Solution

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T \ \vec{x})\hat{}2}{n}$$

Option 1: Closed Form Solution

Matrix notation will make our lives easy!
$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$
Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

$$\mathcal{L} = \ MSE = \frac{\sum_i^n (y_i - \vec{w}^T \ \vec{x})^\wedge 2}{n}$$

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T \vec{x})\hat{}2}{n}$$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T \vec{x})^\wedge 2}{n}$$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T \vec{x})\text{^}2}{n}$$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T \vec{x})^\wedge 2}{n}$$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

$$\mathcal{L} = \frac{(\mathbb{y} - \mathbb{X}\vec{w})^T (\mathbb{y} - \mathbb{X}\vec{w})}{n}$$

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

Option 1: Closed Form Solution

Matrix notation will make our lives easy!
$\mathbb{X} \in \mathbb{R}^{n\times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$
Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n\times 1}$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T\ \vec{x})^\wedge 2}{n}$$

$$\mathcal{L} = \frac{(\mathbb{y} - \mathbb{X}\vec{w})^T(\mathbb{y} - \mathbb{X}\vec{w})}{n}$$

$$\mathcal{L} = \frac{\mathbb{y}^T\mathbb{y} - \mathbb{y}^T\mathbb{X}\vec{w} - \overrightarrow{w^T}\mathbb{X}^T\mathbb{y} + \vec{w}^T\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T\ \vec{x})^\wedge 2}{n}$$

$$\mathcal{L} = \frac{(\mathbb{y} - \mathbb{X}\vec{w})^T(\mathbb{y} - \mathbb{X}\vec{w})}{n}$$

$$\mathcal{L} = \frac{\mathbb{y}^T\mathbb{y} - \mathbb{y}^T\mathbb{X}\vec{w} - \overrightarrow{w^T}\mathbb{X}^T\mathbb{y} + \vec{w}^T\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

$$\nabla\mathcal{L}_w = \frac{-2\mathbb{X}^T\mathbb{y} + 2\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

$$\mathcal{L} = MSE = \frac{\sum_{i}^{n}(y_i - \vec{w}^T\,\vec{x})^\wedge 2}{n}$$

$$\mathcal{L} = \frac{(\mathbb{y} - \mathbb{X}\vec{w})^T(\mathbb{y} - \mathbb{X}\vec{w})}{n}$$

$$\mathcal{L} = \frac{\mathbb{y}^T\mathbb{y} - \mathbb{y}^T\mathbb{X}\vec{w} - \overrightarrow{w^T}\mathbb{X}^T\mathbb{y} + \vec{w}^T\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

$$\nabla\mathcal{L}_w = \frac{-2\mathbb{X}^T\mathbb{y} + 2\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

$$0 = -\mathbb{X}^T\mathbb{y} + \mathbb{X}^T\mathbb{X}\vec{w}$$

Option 1: Closed Form Solution

Matrix notation will make our lives easy!

$\mathbb{X} \in \mathbb{R}^{n \times d}, \mathbb{y} \in \mathbb{R}^{n}, \vec{\mathrm{w}} \in \mathbb{R}^{d}$

Vectors are assumed to be column vectors, i.e., $\mathbb{y} \in \mathbb{R}^{n \times 1}$

Is this a legal operation: $\mathbb{y} - \vec{w}\mathbb{X}$?

Is this a legal operation: $(\mathbb{y} - \mathbb{X}\,\vec{w})(\mathbb{y} - \mathbb{X}\vec{w})$?

Shape errors are the most common errors you will face when starting deep learning

$$\mathcal{L} = MSE = \frac{\sum_i^n (y_i - \vec{w}^T\ \vec{x})\text{^}2}{n}$$

$$\mathcal{L} = \frac{(\mathbb{y} - \mathbb{X}\vec{w})^T(\mathbb{y} - \mathbb{X}\vec{w})}{n}$$

$$\mathcal{L} = \frac{\mathbb{y}^T\mathbb{y} - \mathbb{y}^T\mathbb{X}\vec{w} - \overrightarrow{w^T}\mathbb{X}^T\mathbb{y} + \vec{w}^T\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

$$\nabla\mathcal{L}_w = \frac{-2\mathbb{X}^T\mathbb{y} + 2\mathbb{X}^T\mathbb{X}\vec{w}}{n}$$

$$0 = -\mathbb{X}^T\mathbb{y} + \mathbb{X}^T\mathbb{X}\vec{w}$$

$$(\mathbb{X}^T\mathbb{X})^{-1}(\mathbb{X}^T\mathbb{y}) = \vec{w}$$

Closed Form Solution

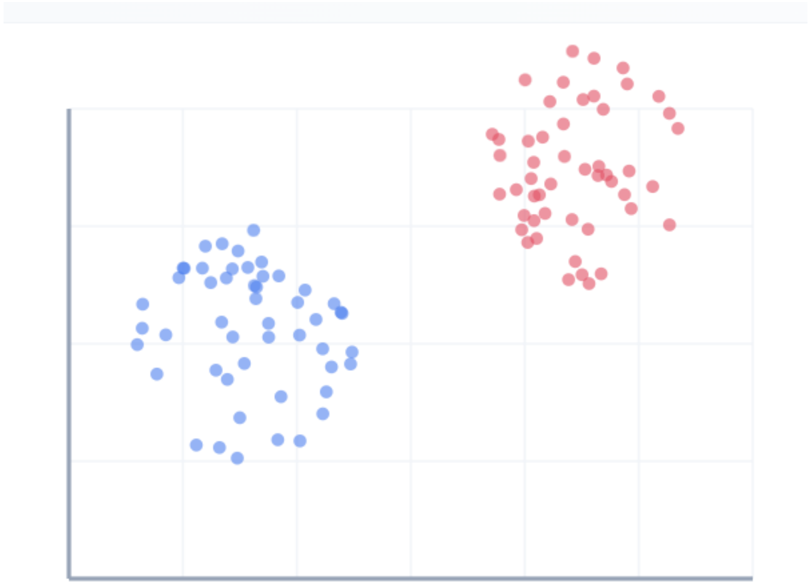
Advantages:

- Simple/fast to implement

Disadvantages:

- Need to invert: $(\mathbb{X}\mathbb{X}^T)^{-1}$
- Matrix inversion is $O(n^3)$
- $(\mathbb{X}\mathbb{X}^T)$ May not be invertible
- Doesn't necessarily exist for other models

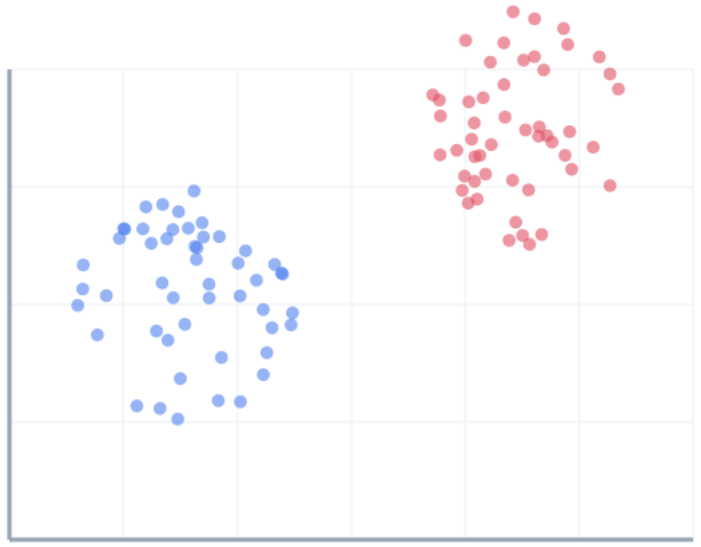
A Linear Classification Model



A Linear Classification Model

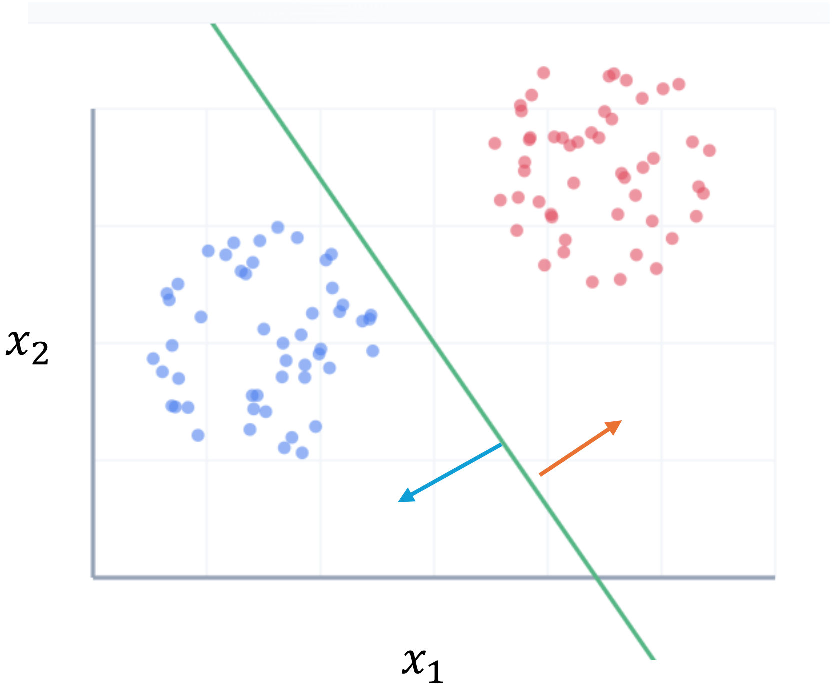
Linear Regression is a linear model for *regression*.

What's a natural way to make a linear *classifier*?



A Classifier

Everything above the line (or hyperplane in >2D) is classified as 1, everything below the line as 0

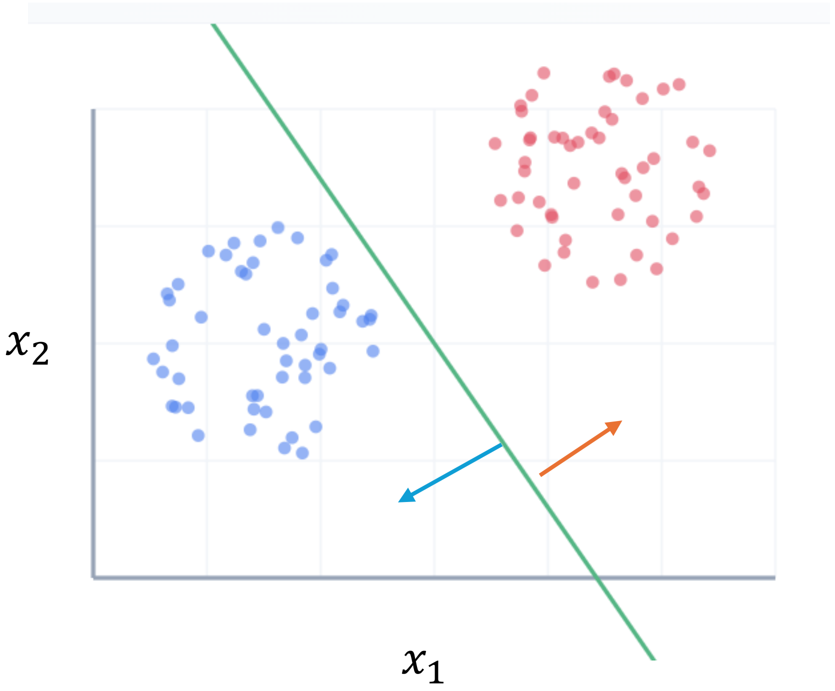


$$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

A Classifier

Everything above the line (or hyperplane in >2D) is classified as 1, everything below the line as 0

How can you tell if a point is above or below the line?



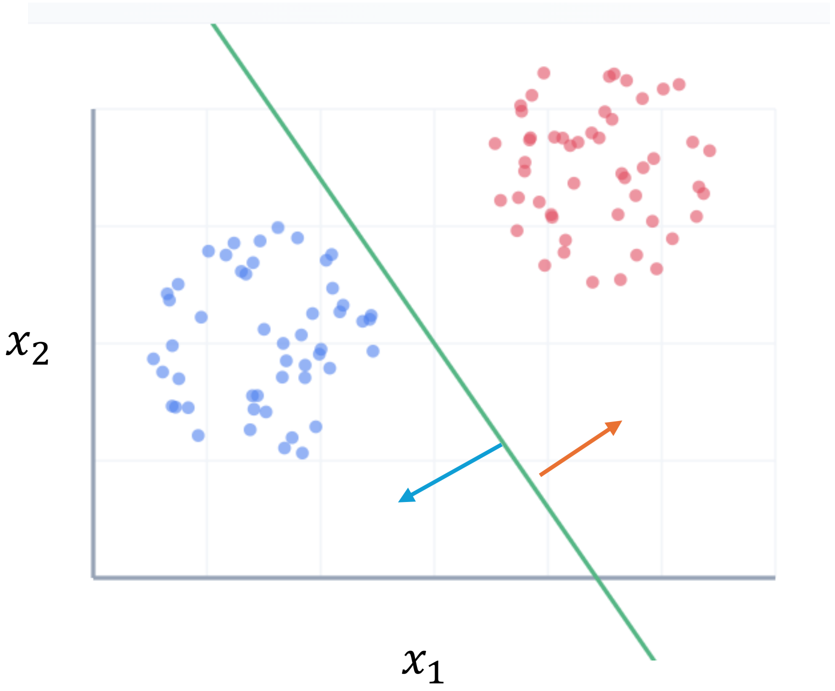
$$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

A Classifier

Everything above the line (or hyperplane in >2D) is classified as 1, everything below the line as 0

How can you tell if a point is above or below the line?

If $\hat{y} = 0$, the point is **on** the line,
If $\hat{y} > 0$, the point is "**above**" the line,
If $\hat{y} < 0$, the point is "**below**" the line



$$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

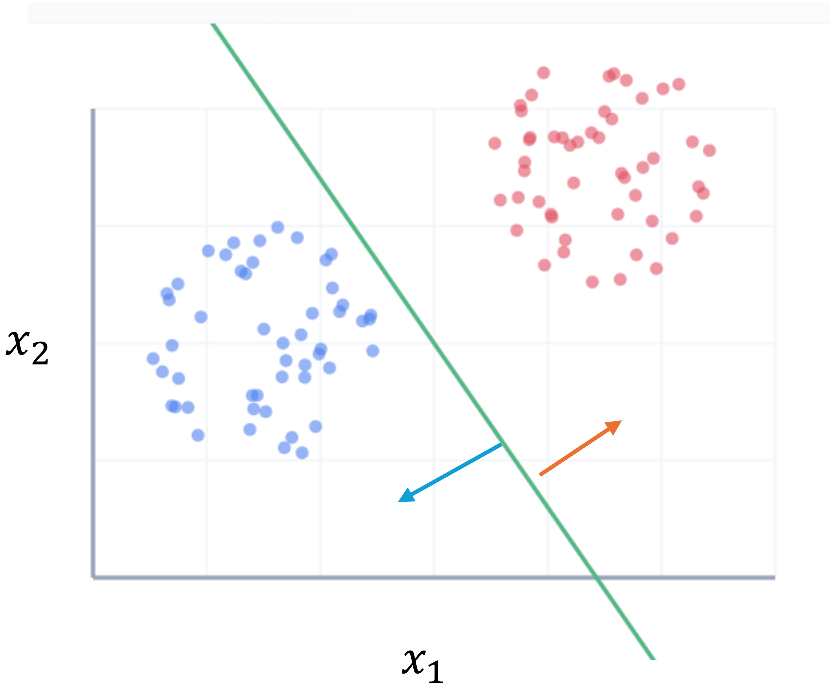
A Classifier

Everything above the line (or hyperplane in >2D) is classified as 1, everything below the line as 0

How can you tell if a point is above or below the line?

If $\hat{y} = 0$, the point is **on** the line,
If $\hat{y} > 0$, the point is "**above**" the line,
If $\hat{y} < 0$, the point is "**below**" the line

If $\hat{y} > 0$, predict 1.
If $\hat{y} \leq 0$, predict 0.



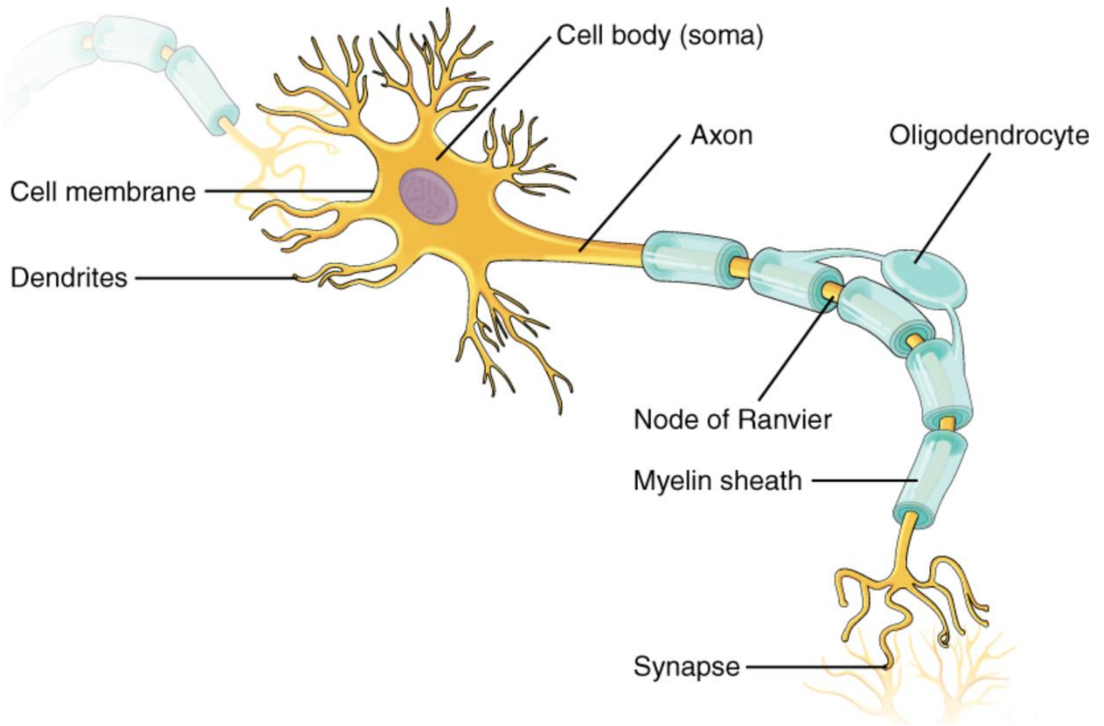
$$\hat{y} = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

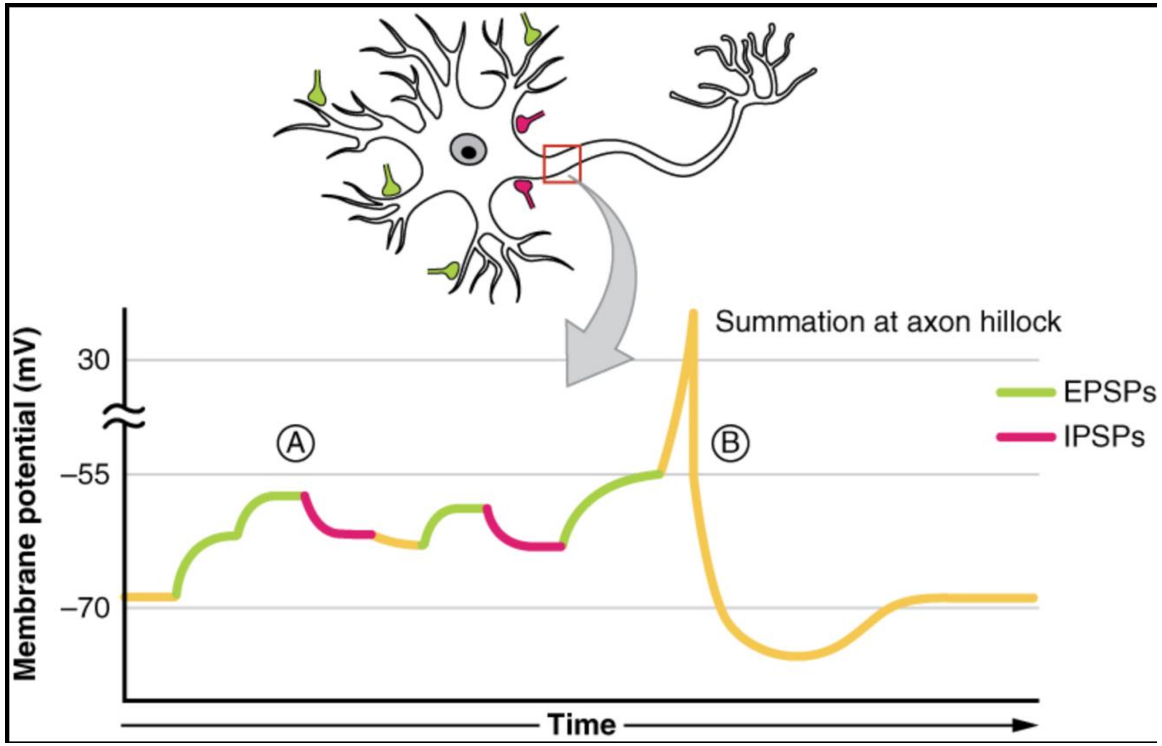
Perceptrons: A Linear Classifier

(Our first building block of Deep Learning)

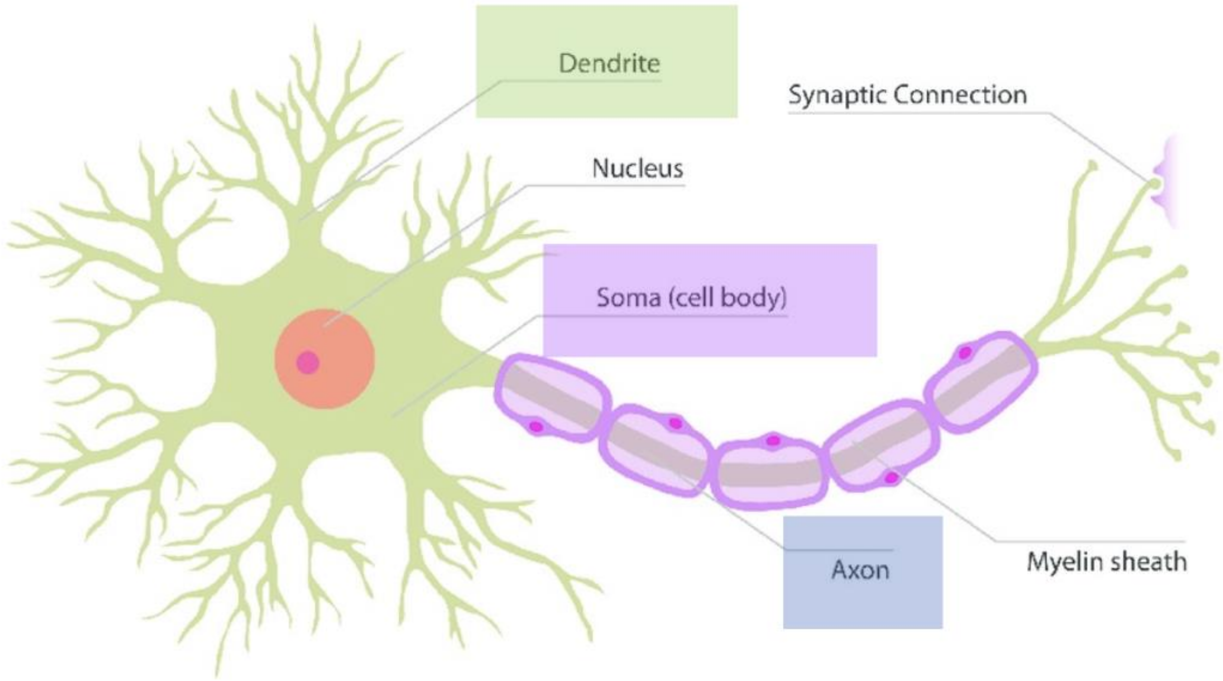
Biological Motivation

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

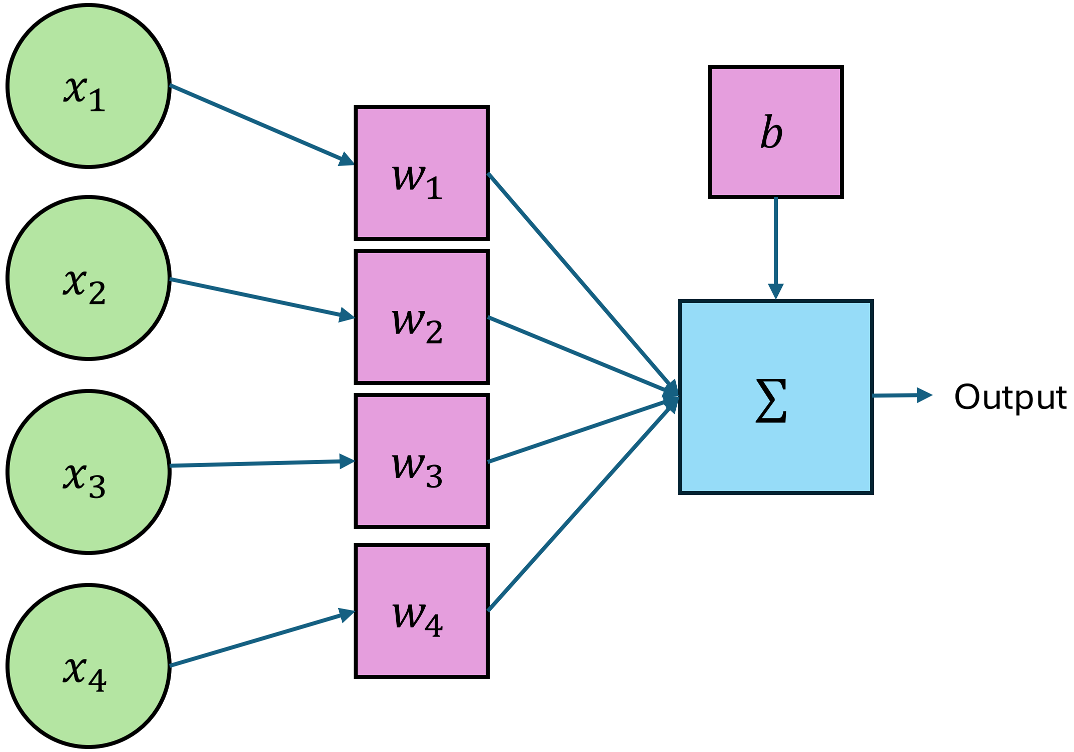




The Perceptron



Biological Neuron

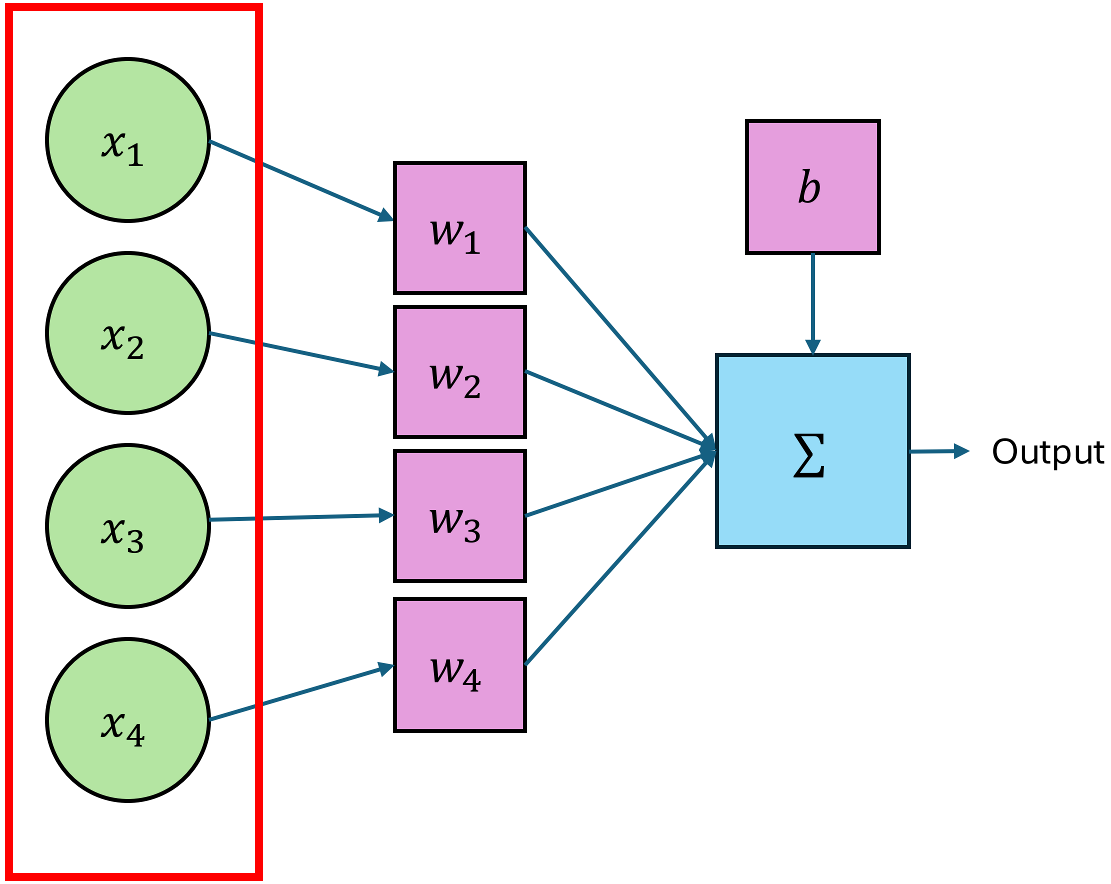


Artificial Neuron (Perceptron)

Inputs

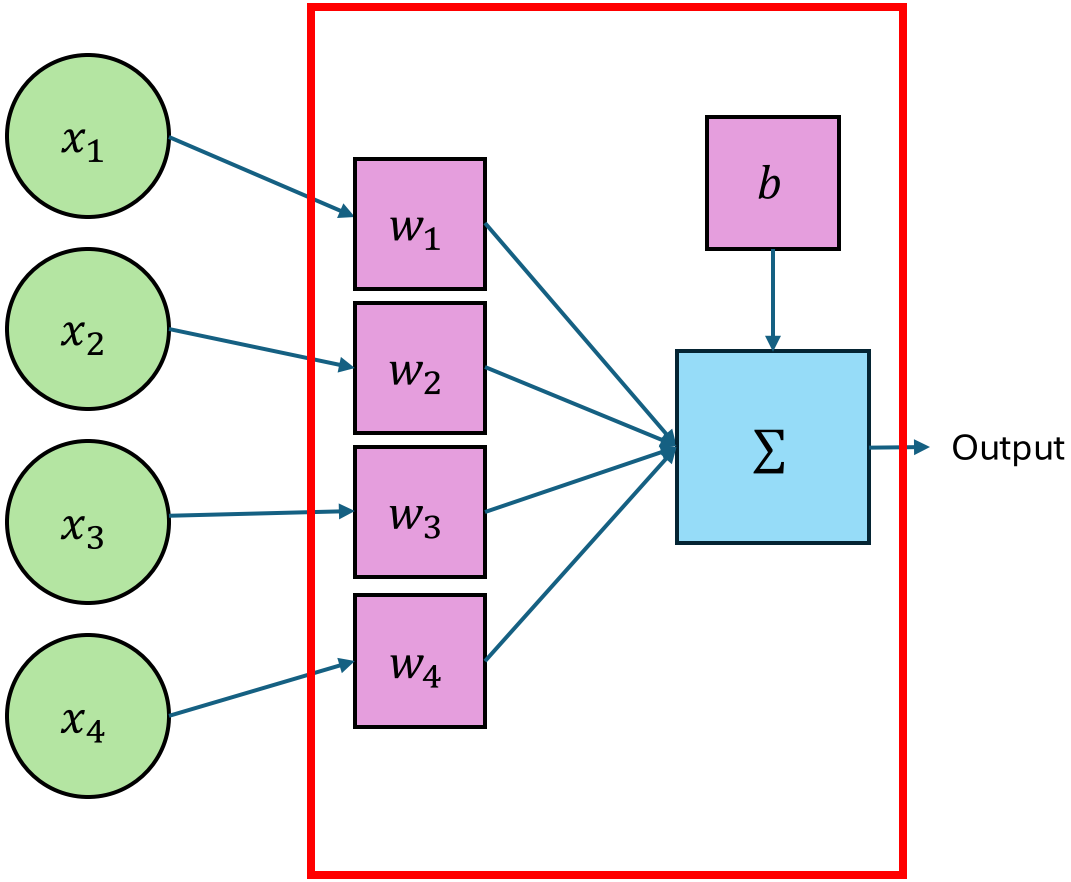
Inputs are $\vec{x} = [x_1, x_2, \dots, x_d]$

Features of the data



Predicting with a Perceptron

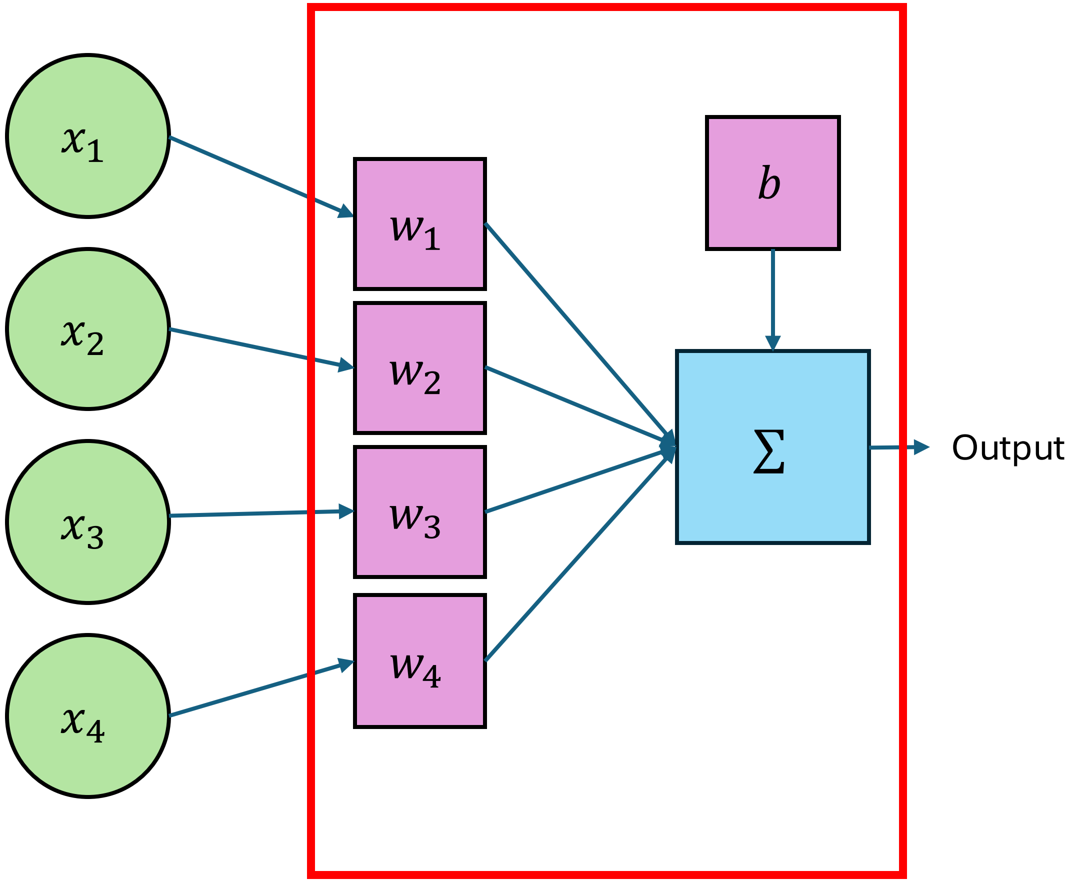
1. Take each of the inputs and multiply by corresponding weight
2. Sum the results, add bias term



Predicting with a Perceptron

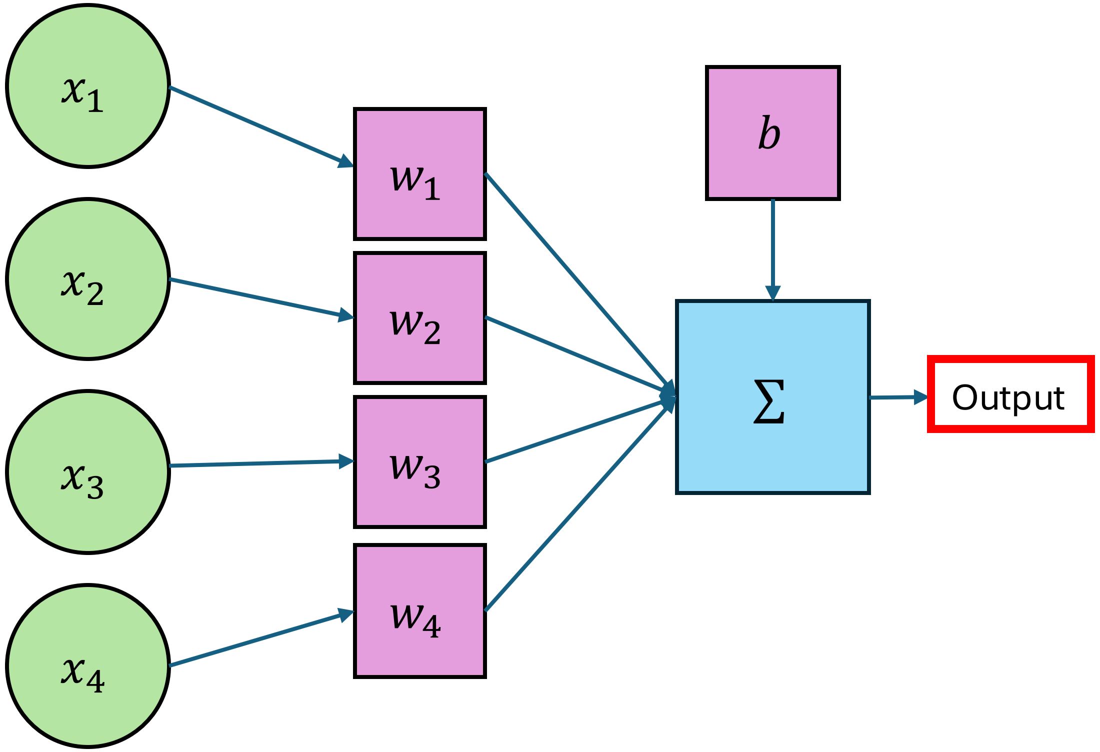
1. Take each of the inputs and multiply by corresponding weight
2. Sum the results, add bias term

Until here, a Perceptron and Linear Regression are equivalent



Predicting with a Perceptron

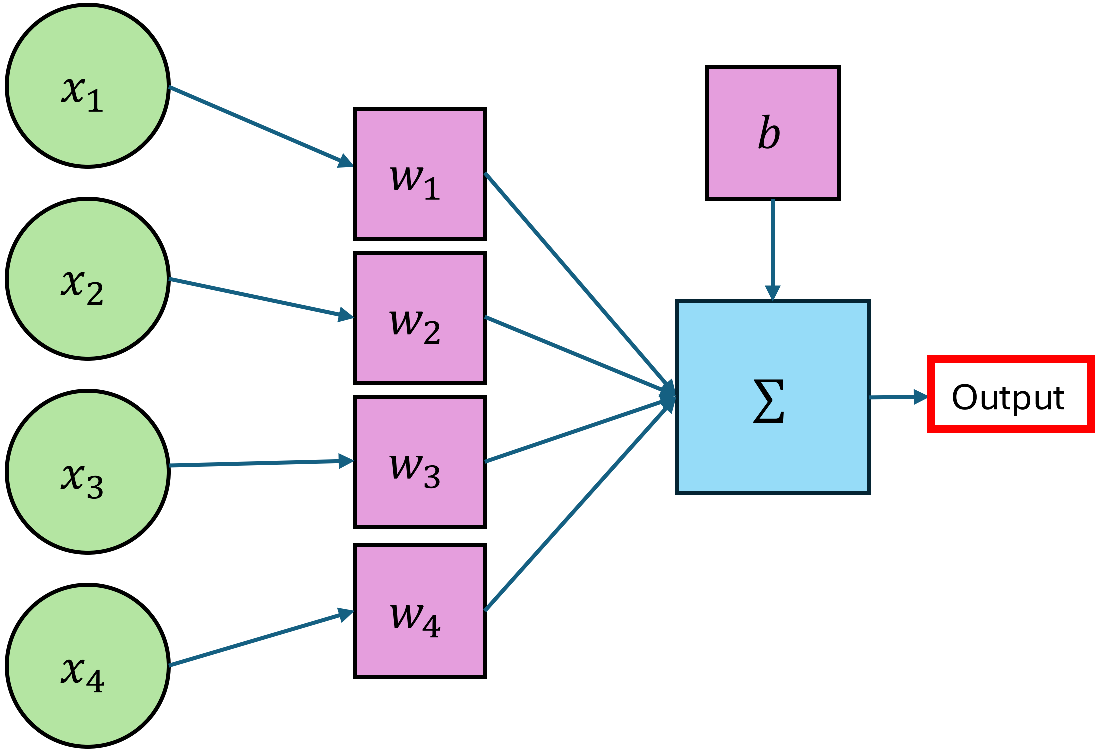
1. Take each of the inputs and multiply by corresponding weight
2. Sum the results, add bias term
3. If output is above 0, return 1, otherwise return 0



Predicting with a Perceptron

1. Take each of the inputs and multiply by corresponding weight
2. Sum the results, add bias term
3. If output is above 0, return 1, otherwise return 0

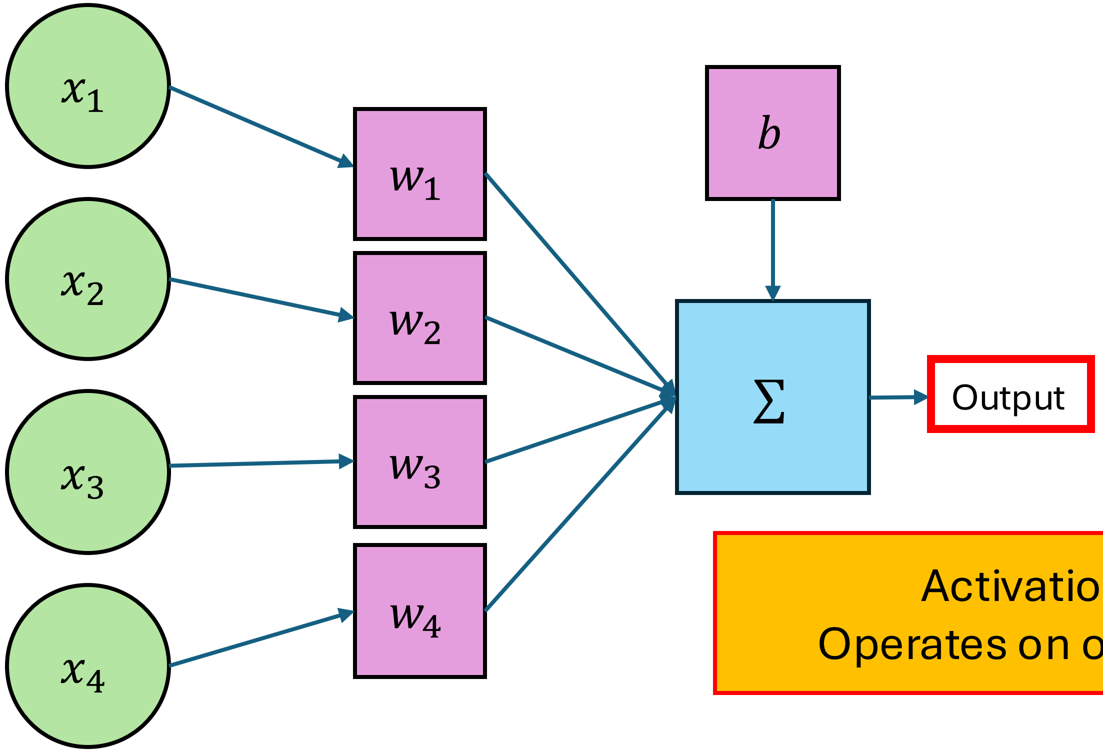
Activation Function
(many more to come)



Predicting with a Perceptron

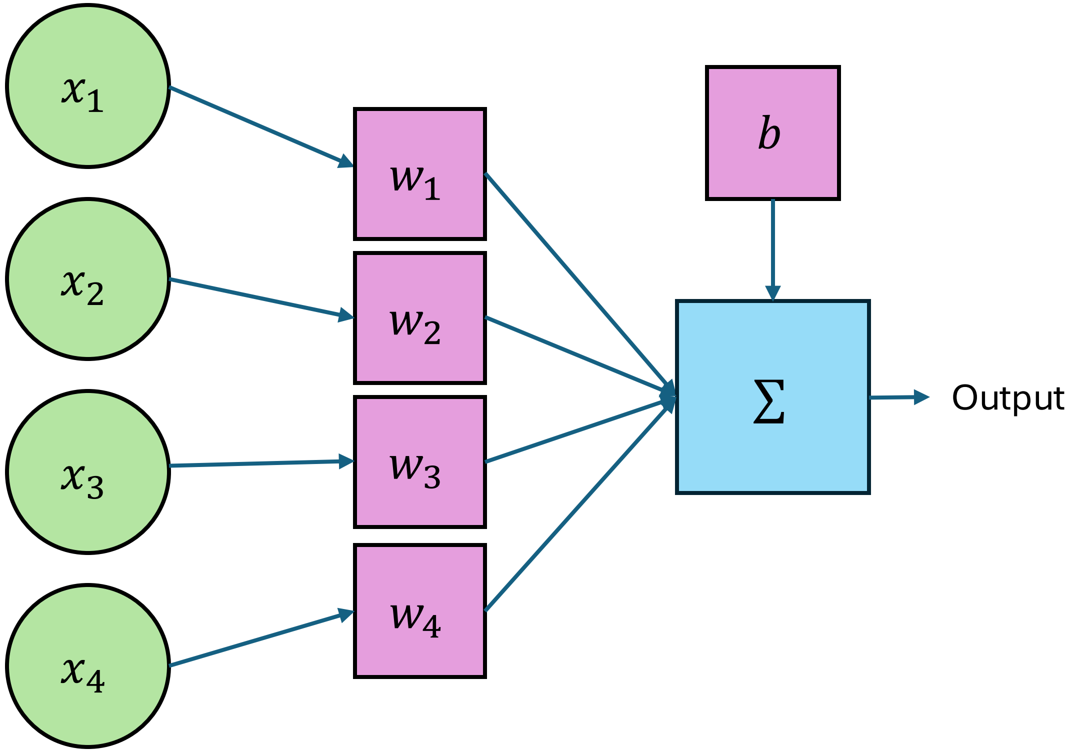
1. Take each of the inputs and multiply by corresponding weight
2. Sum the results, add bias term
3. If output is above 0, return 1, otherwise return 0

Activation Function
(many more to come)



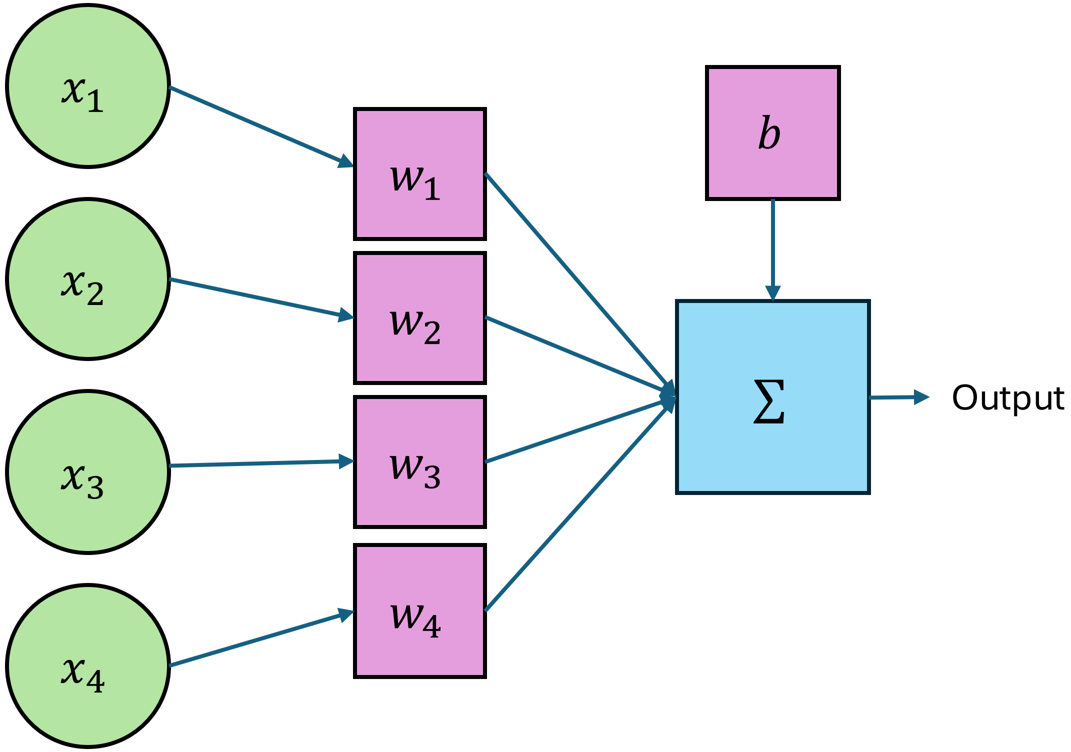
Activation Function
Operates on output of neuron

Understanding Weights



Understanding Weights

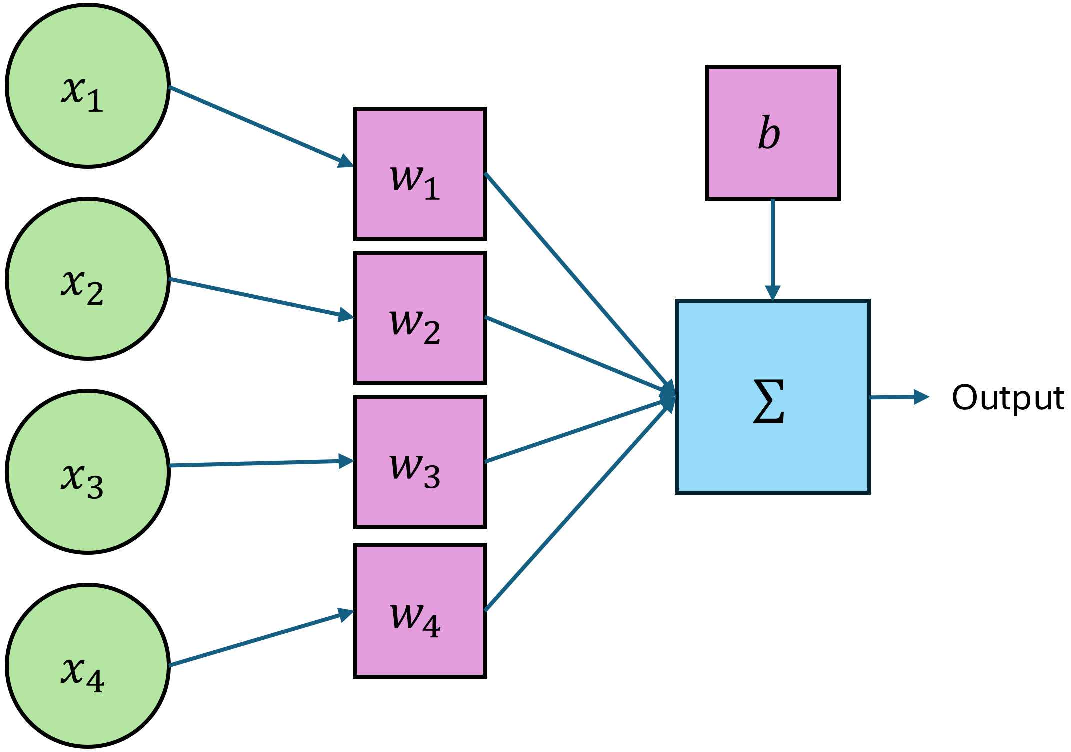
What would it mean for a weight to be 0?



Understanding Weights

What would it mean for a weight to be 0?

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

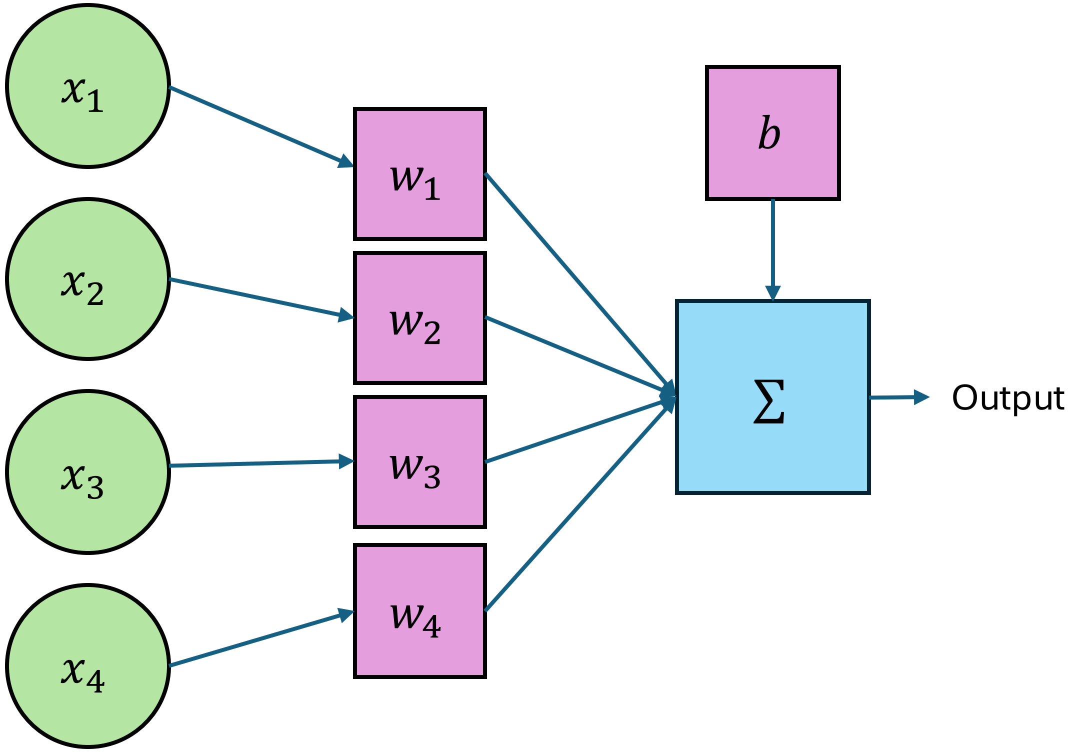


Understanding Weights

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?

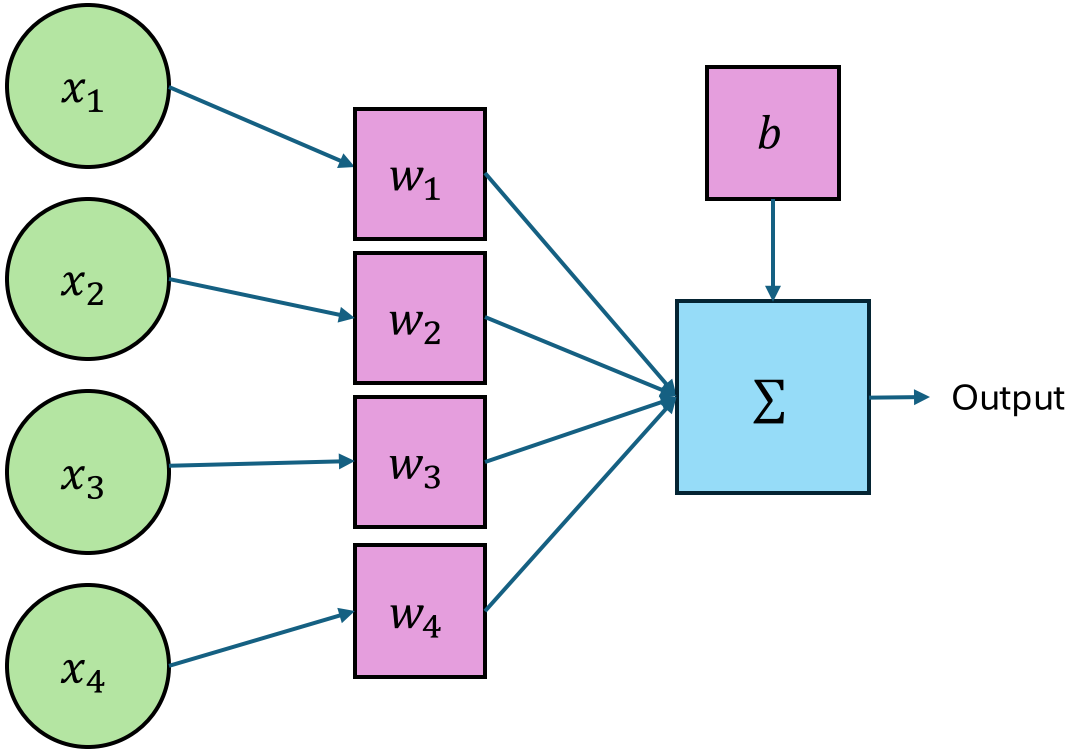


Understanding Weights

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?



How Strong are Linear Separators?

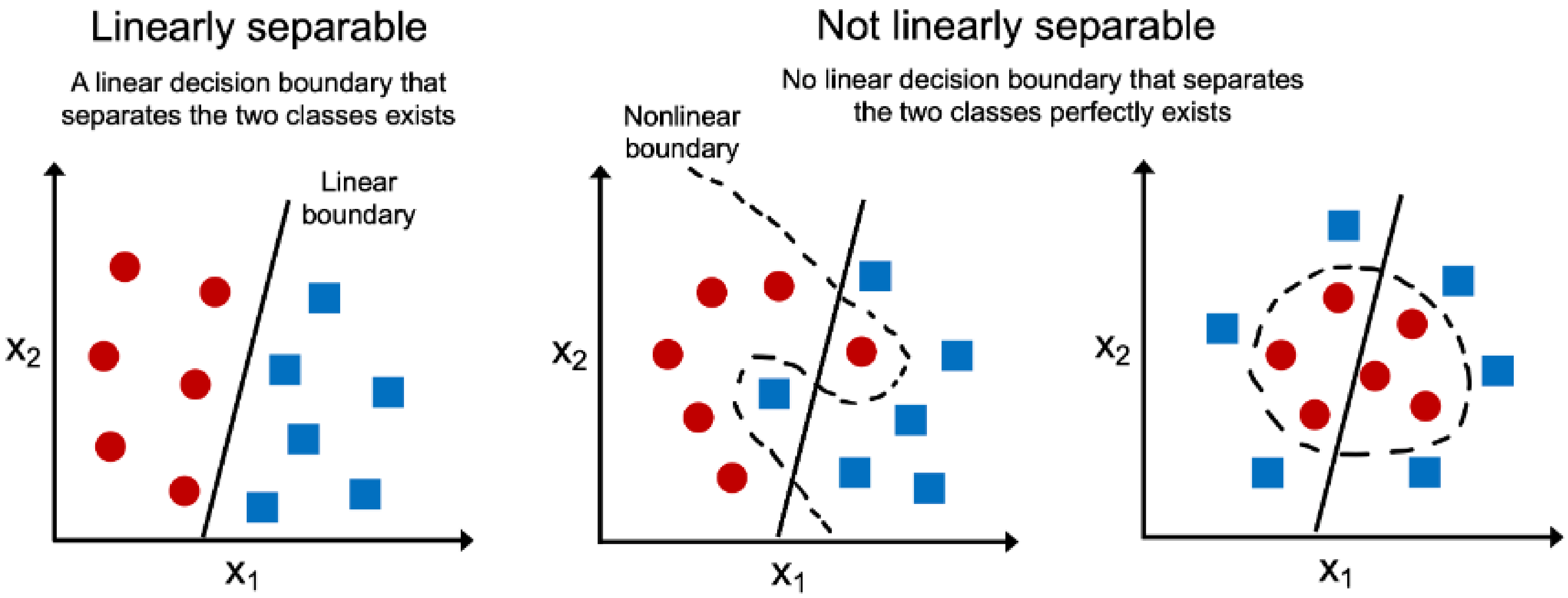


Image courtesy of: https://vitalflux.com/how-know-data-linear-non-linear/

MNIST

The most famous dataset in Deep Learning

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

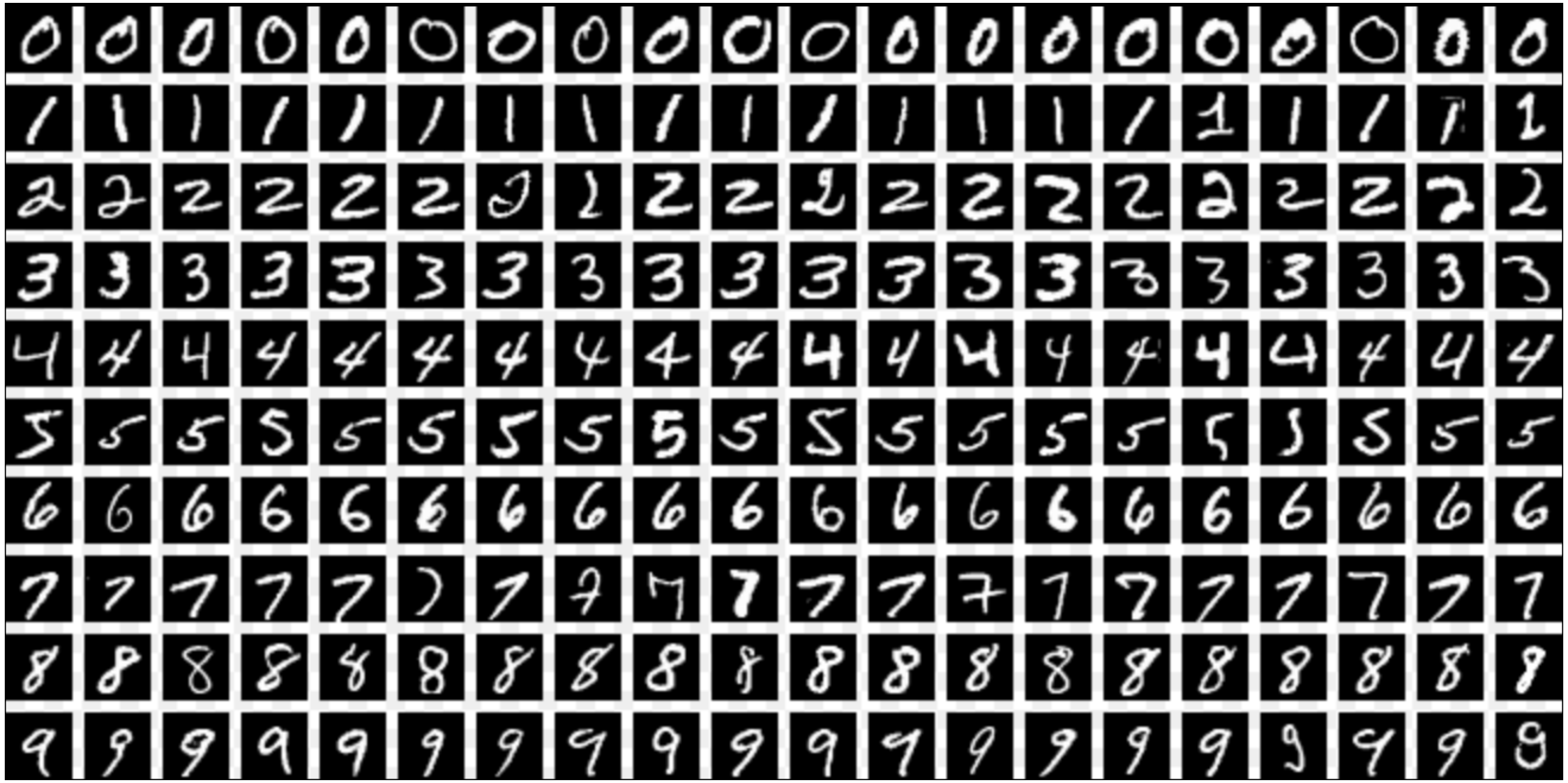
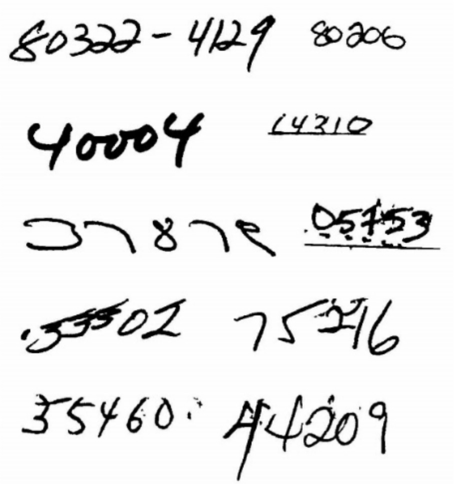


Image courtesy of Wikipedia

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”

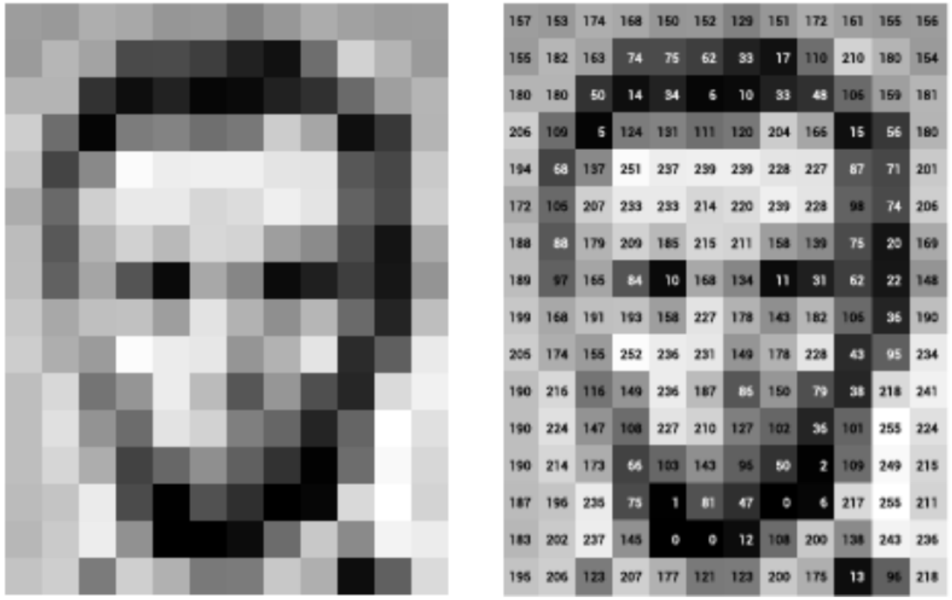
How Does a Computer know this is a three?

“three”

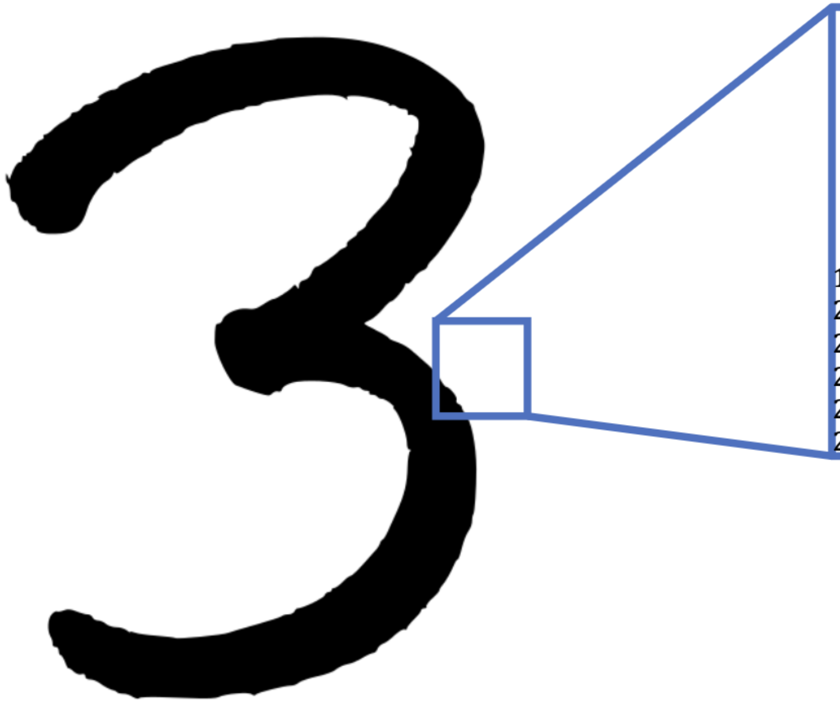
Representing digits in the computer

- 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

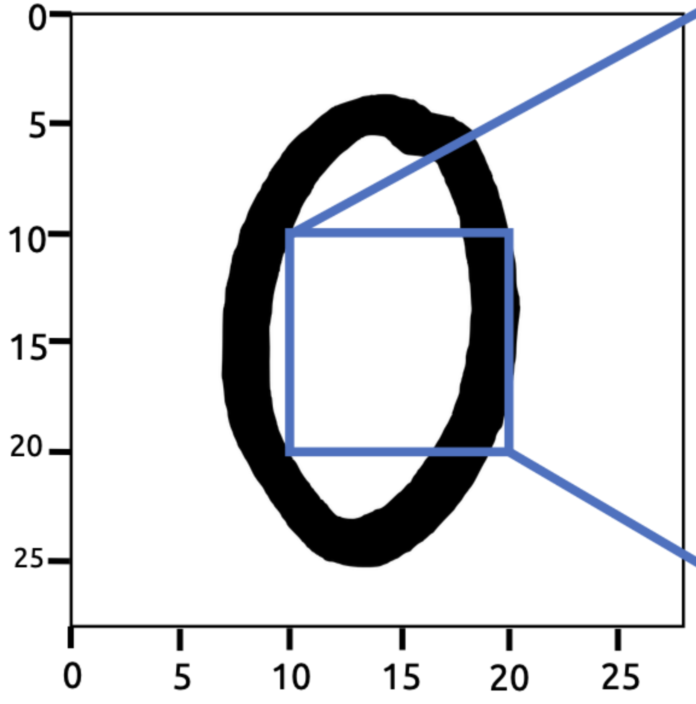


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139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



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

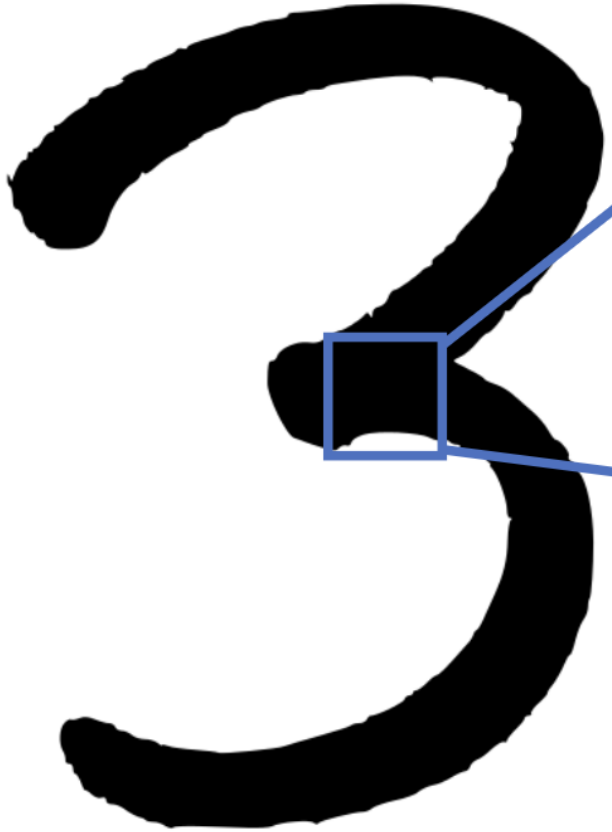


	10	11	12	13	14	15	16	17	18	19
10	0	0	0	0	0	0	0	170	236	255
11	0	0	0	0	0	0	0	0	250	254
12	0	0	0	0	0	0	0	0	255	255
13	0	0	0	0	0	0	0	0	240	245
14	0	0	0	0	0	0	0	0	245	236
15	0	0	0	0	0	0	0	0	251	253
16	0	0	0	0	0	0	0	124	255	255
17	0	0	0	0	0	0	0	210	254	120
18	0	0	0	0	0	0	0	165	230	0
19	0	0	0	0	0	0	120	154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

what the computer sees

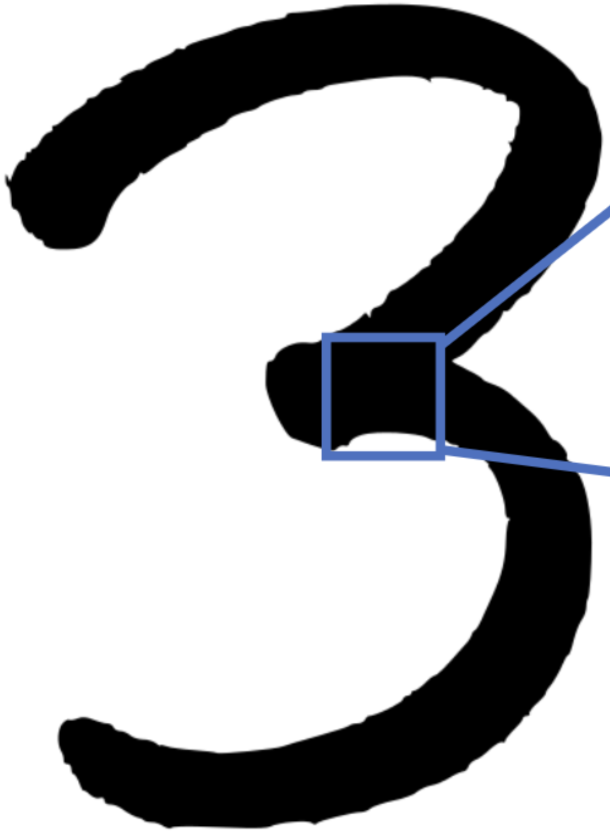
- Pixel in position [15, 15] is light.

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



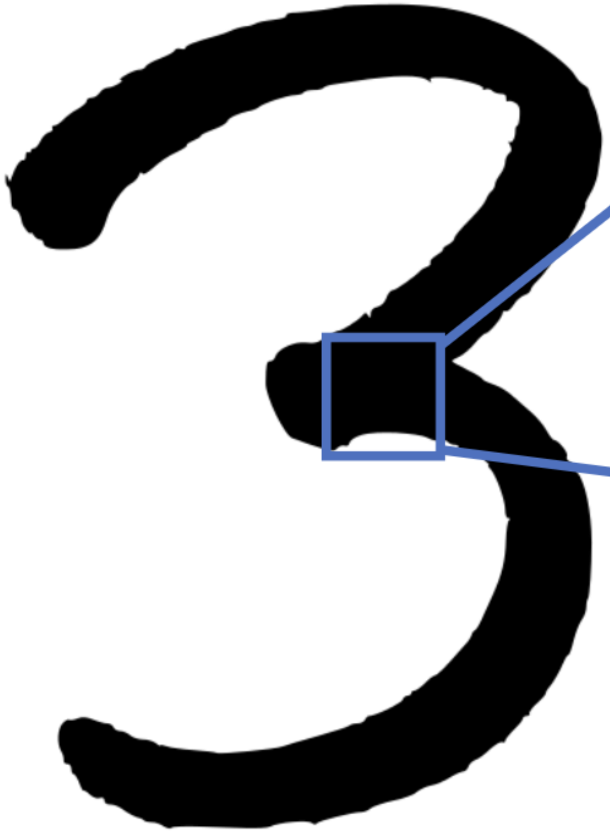
255	255	255	255	255	253	254	245	255
255	255	251	255	255	255	254	235	252
255	252	255	250	255	245	255	253	234
253	255	255	255	251	254	255	255	235
255	255	252	255	249	255	239	243	255
255	250	255	245	255	255	254	244	254
255	255	255	255	249	255	255	255	244
249	255	253	255	233	255	249	245	239
255	255	255	250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

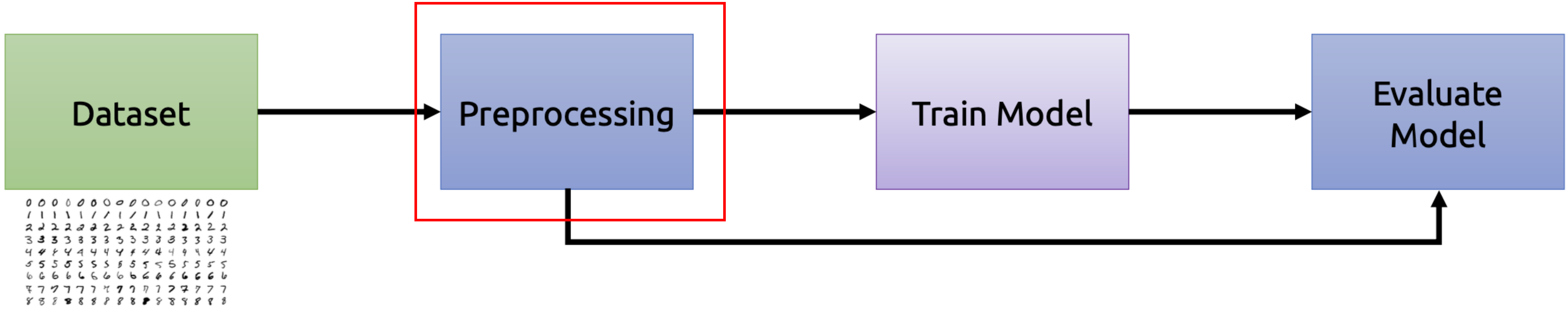
Darker pixels in the middle



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

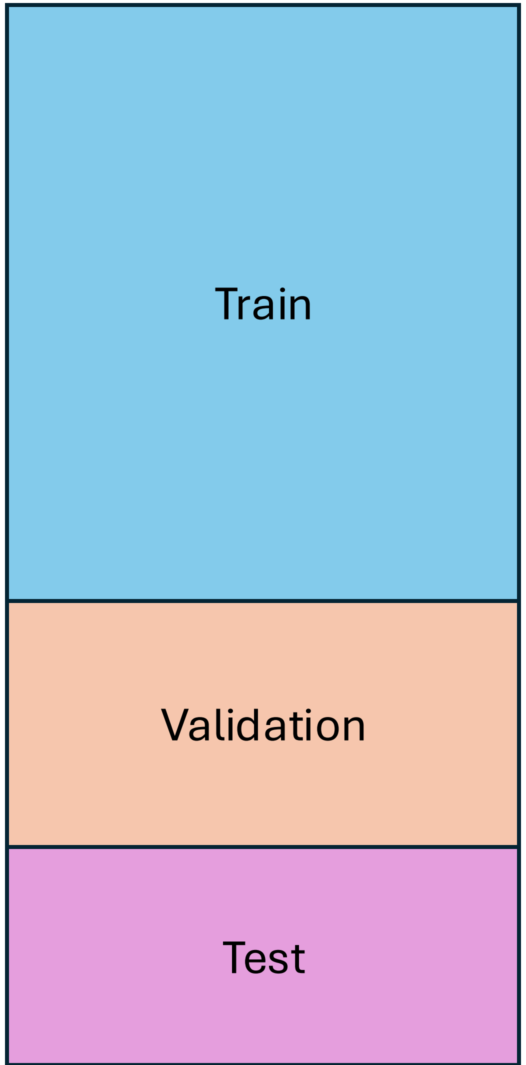
Can we define a set of ***heuristics*** (i.e. rules based on our intuition), to classify digits?

Machine Learning Pipeline for Digit Recognition



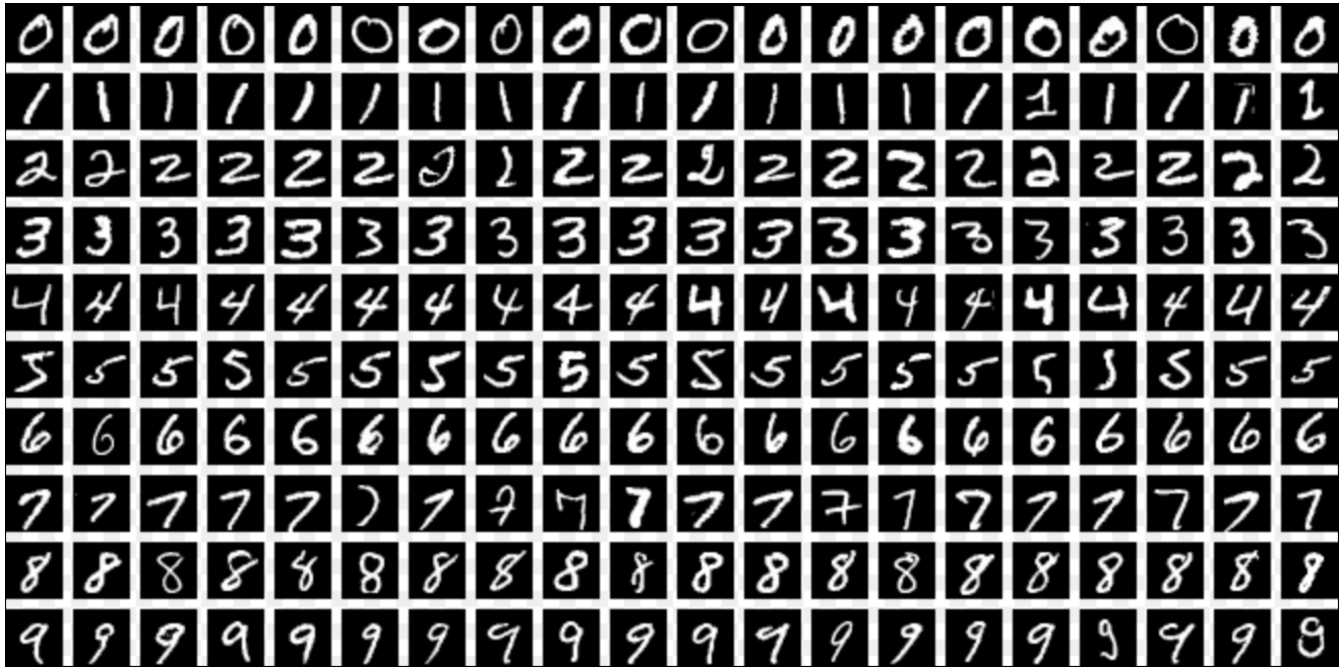
Train, validation, and test sets

- **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

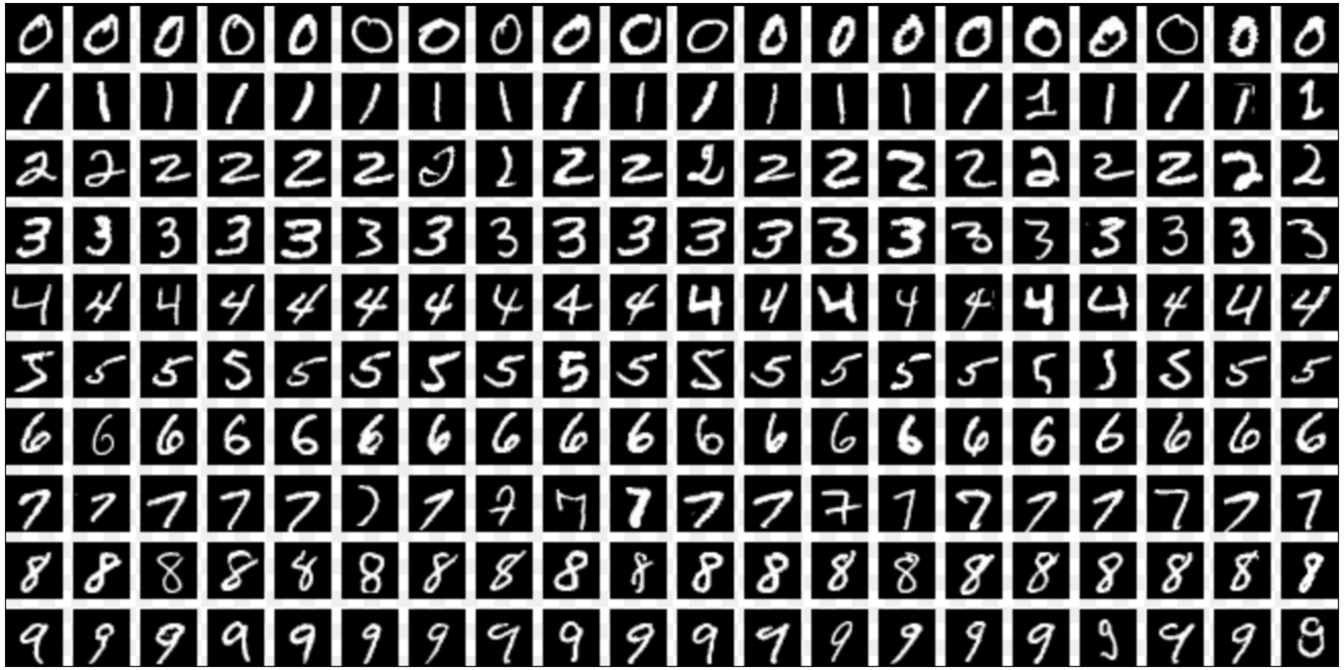
- 60,000 Images in training set
- 10,000 Images in test set
- No explicit validation set



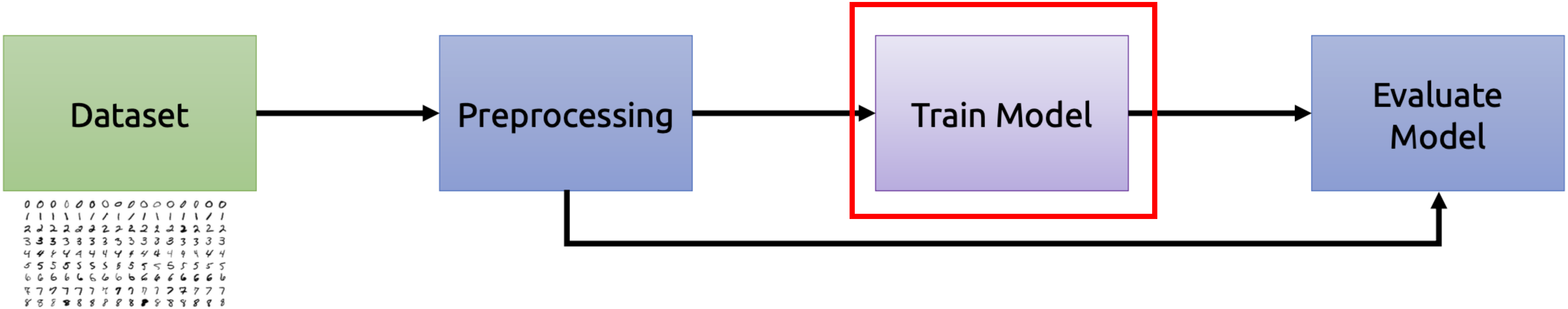
MNIST

- 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

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

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

$x^{(2)} =$ 

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

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

What is our input space?

Input: $\mathbb{X}$

Target: $\mathbb{Y}$

Pixel Grid

Which digit is it?

$x^{(1)} =$ 

28x28 pixels

Function: f

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

$y^{(1)} = \text{“2”}$

$x^{(2)} =$ 

$y^{(2)} = \text{“0”}$

Our Problem:

Classifying MNIST digits requires predicting 1 of 10 possible values

Input: $\mathbb{X}$

What is our input space?

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?

What is our prediction task?

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 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

$x^{(2)} =$ 

Function: f

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

Target: $\mathbb{Y}$

Is it digit 2?

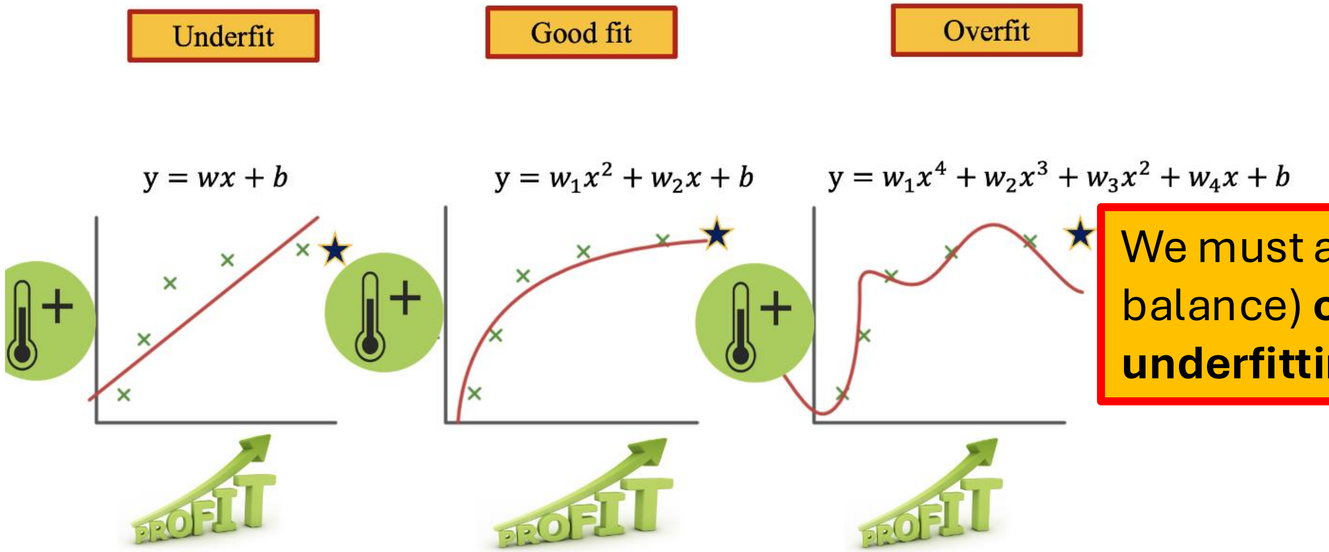
$y^{(1)} = 1$ 

$y^{(2)} = 0$ 

MNIST Results

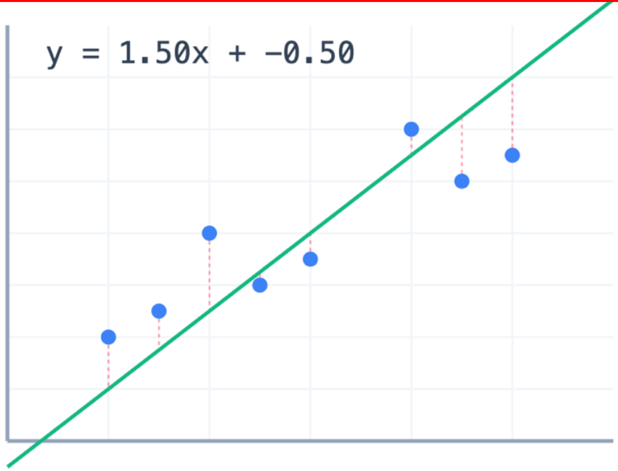
- Perceptrons (linear separator) can achieve 88% accuracy on MNIST.
- Linear separation tends to become “easier” in higher dimensional spaces

Recap



We must always test for (and balance) **overfitting and underfitting**

Loss Functions tell us about the performance of the model (which we will also optimize for)



A perceptron/neuron works just like a linear regression, but has a different **activation function**