Stanford
MEDICINE

Department of Psychiatry
and Behavioral Sciences

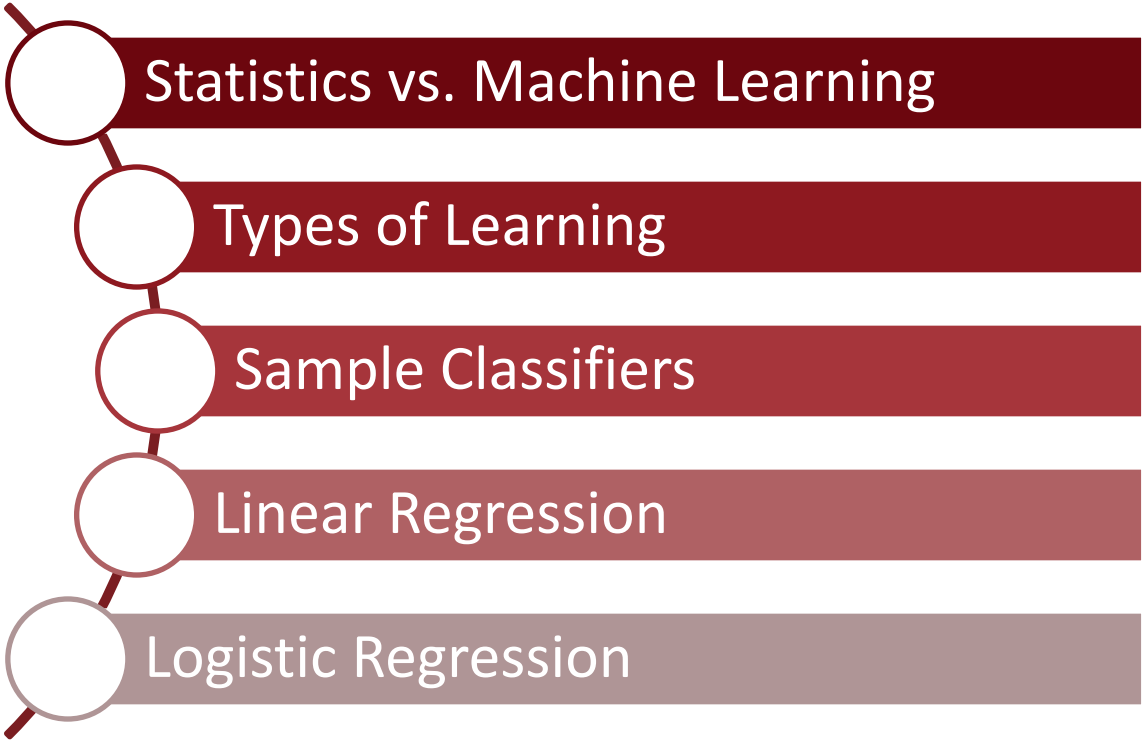
Machine Learning for Neuroimaging

Introduction to Multivariate Analysis

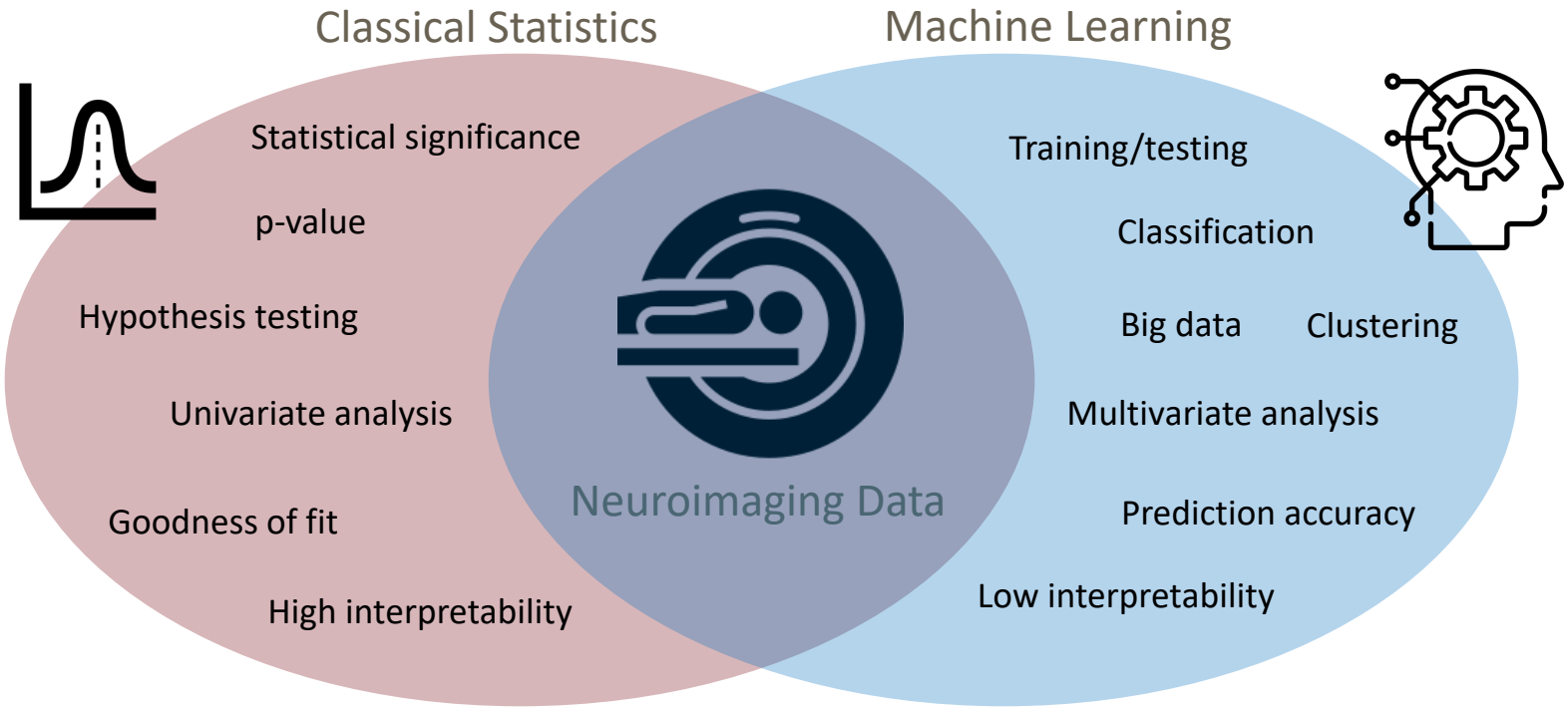
Autumn 2023
PSYC221/PSYC121/BIODS227

Session 2 – 9/28/2023

Today…


Statistics vs. Machine Learning
Types of Learning
Sample Classifiers
Linear Regression
Logistic Regression

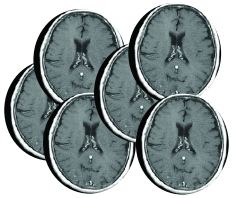
Data Science in Neuroimaging



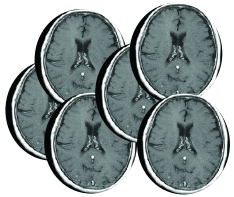
Statistical Analysis (Hypothesis Testing)

to be continued in week 3 (10/10)

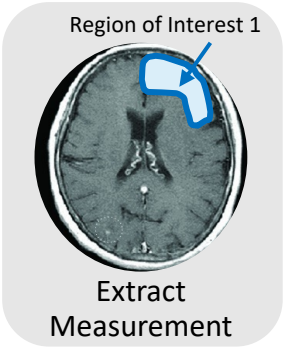
Null distribution*: probability distribution of the test statistic when the null hypothesis is true
** Shape depends on which test procedure you choose*

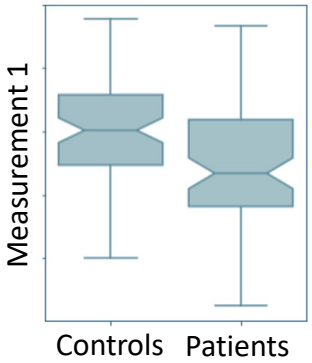


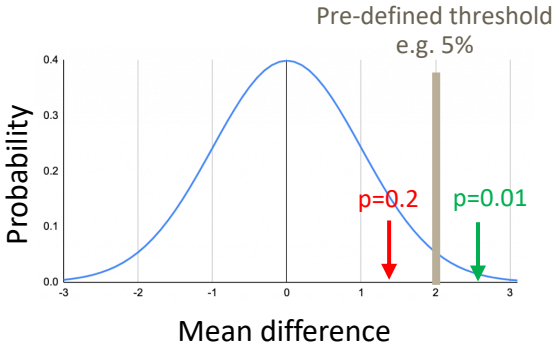
Healthy controls



Patients





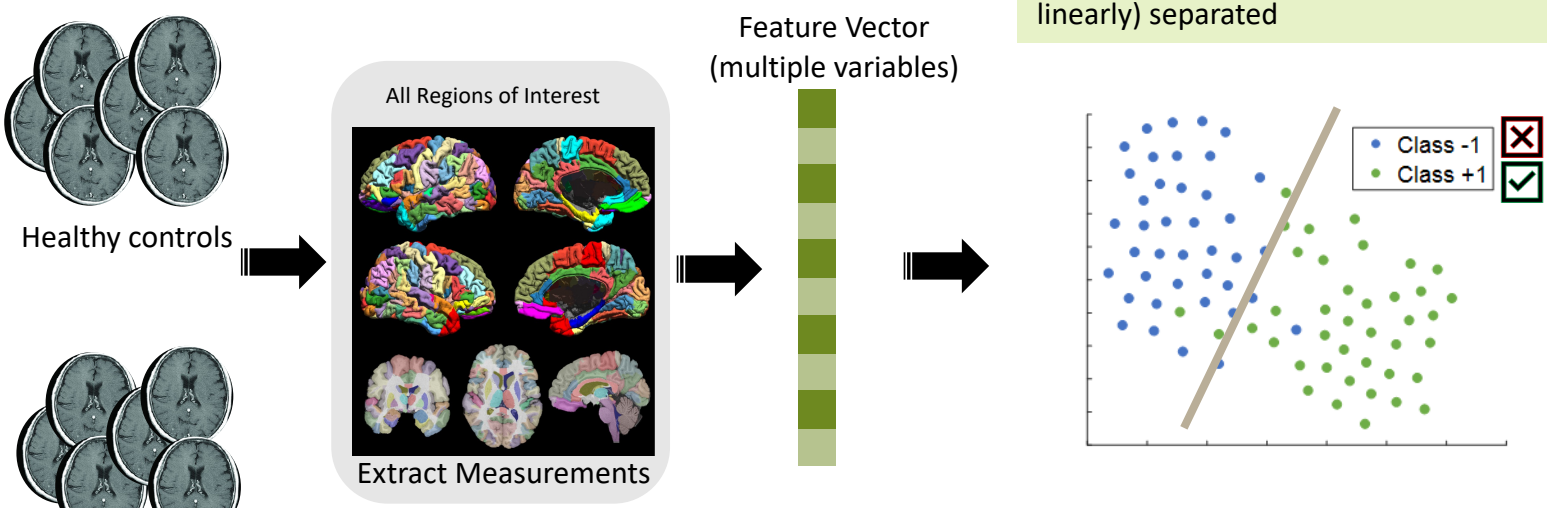


Test statistic: difference between the two means
Null hypothesis: the mean difference is 0

☑ Reject the null hypothesis (patients have lower measurements **on average**)

☒ Cannot reject the null hypothesis

Multivariate Analysis

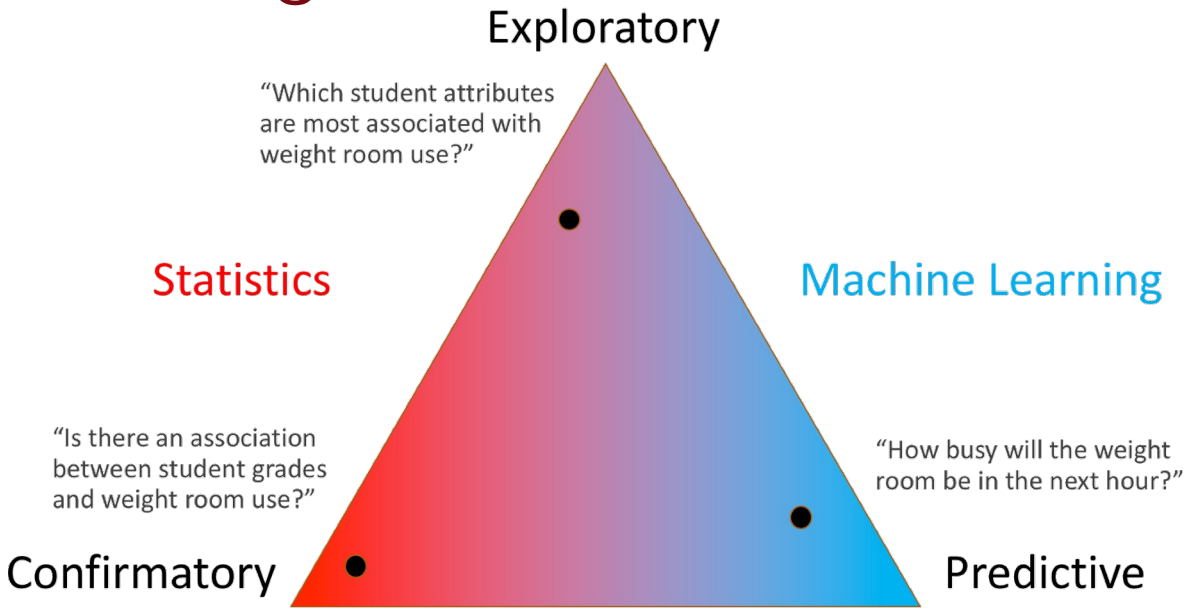


Assumption: Data could be linearly (or non-linearly) separated

Objective: to determine if a given subject should belong to the patient group or the controls

“Statistics” vs. “Machine Learning” vs. “Statistical Learning”

- Machine learning and statistics are closely related fields in terms of methods, but distinct in their principal goal
- **Statistics draws population inferences** from a sample, while **machine learning finds generalizable predictive patterns**.



Machine Learning (a tool for multi-variate analysis)

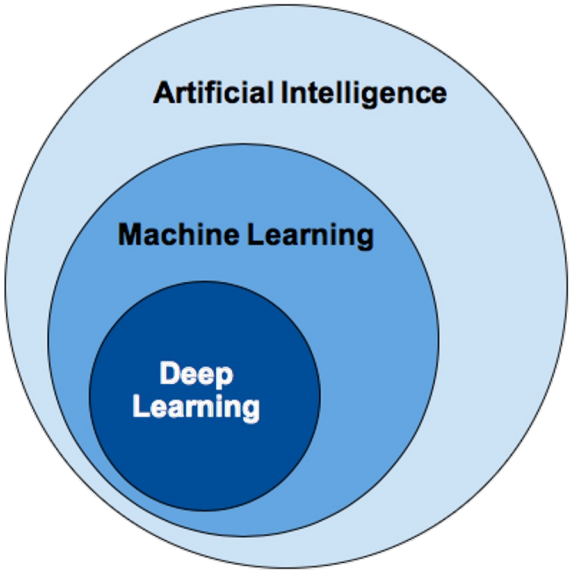
Study of algorithms that

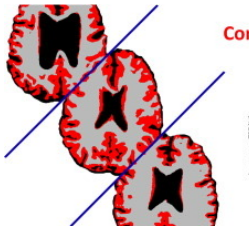
improve their performance (P)

at some task (T)

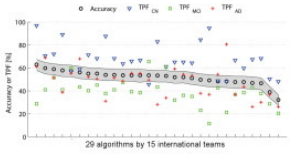
with experience (E)

- Tom Mitchell, CMU



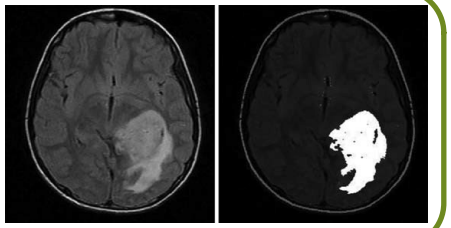


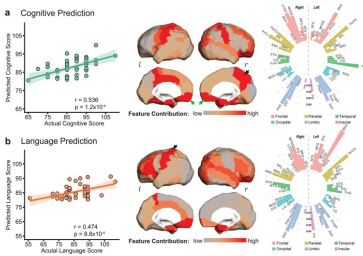
Computer-aided diagnosis of dementia



	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

Tumor Segmentation





Predicting neurodevelopmental outcomes

Brain Fingerprinting

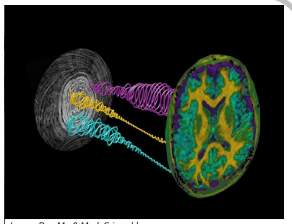
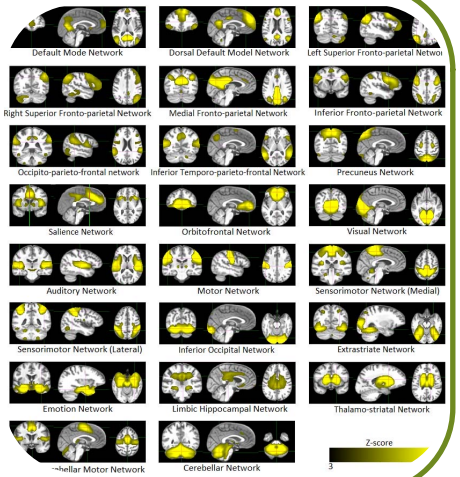
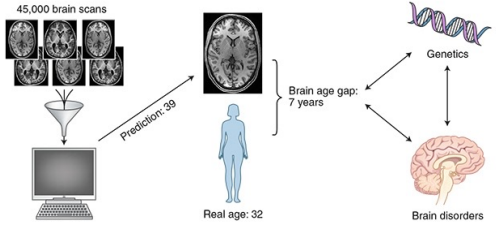


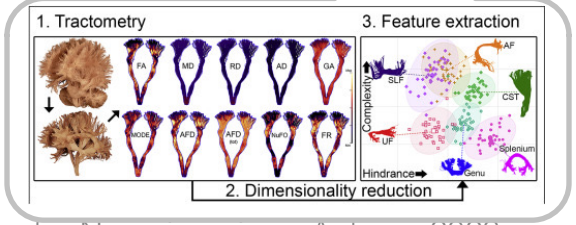
Image: Dan Ma & Mark Griswold

Clustering brain functional states



Age prediction, relations clinical test scores





Neuroimaging (from scan to evaluation)

- Where can machine learning be useful?

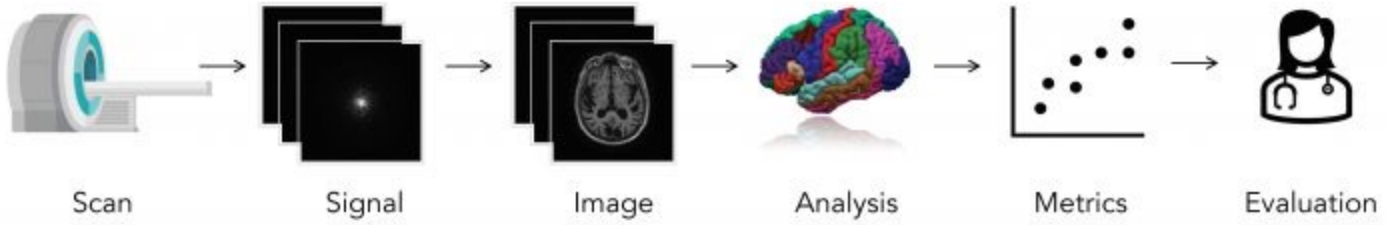


Image source: Singh et al. Neuroinformatics 2022

Neuroimaging (from scan to evaluation)

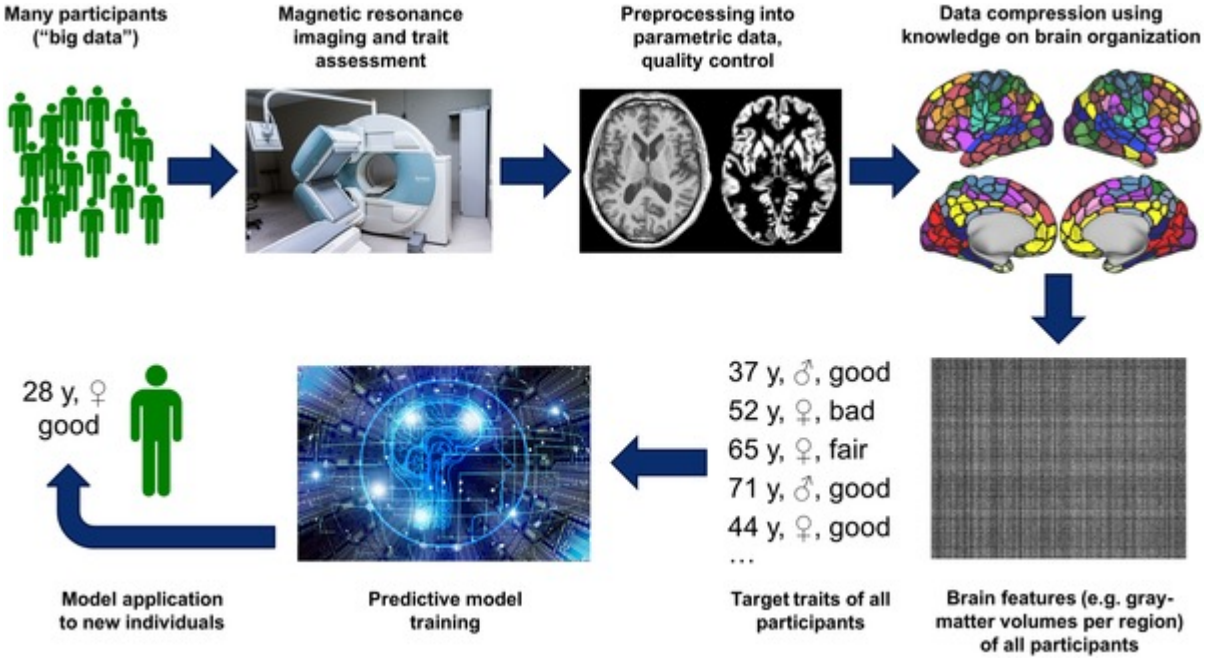
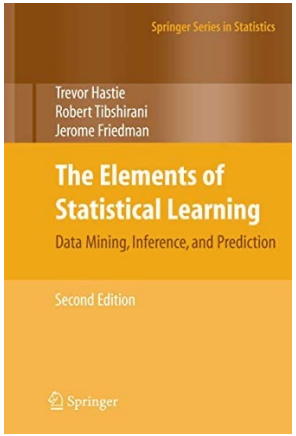
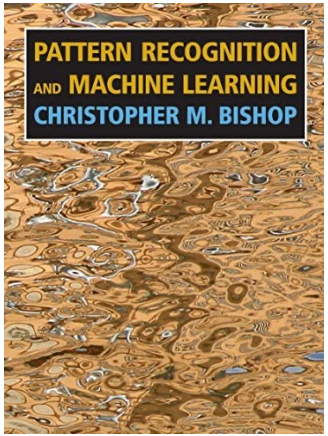


Image source: Eickhoff and Langner, Plos Biology 2019

Reading Materials

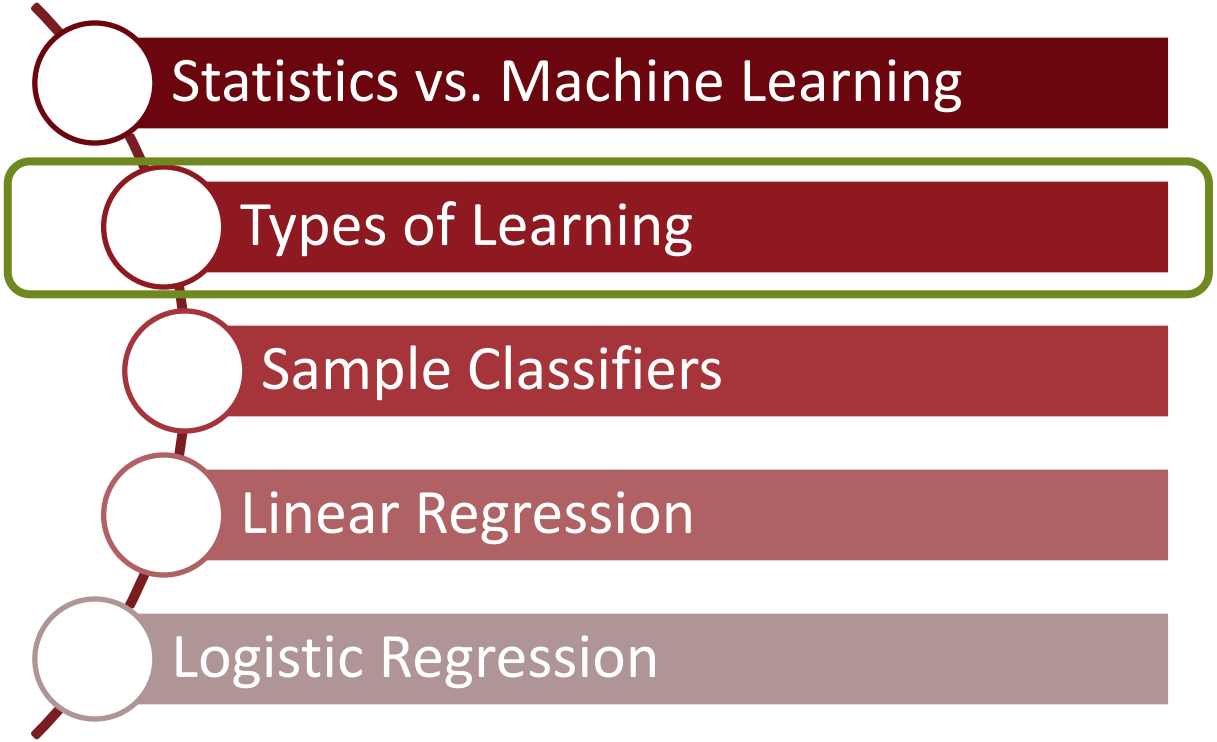
- https://web.stanford.edu/class/psyc221/#resource
- Textbook(s):





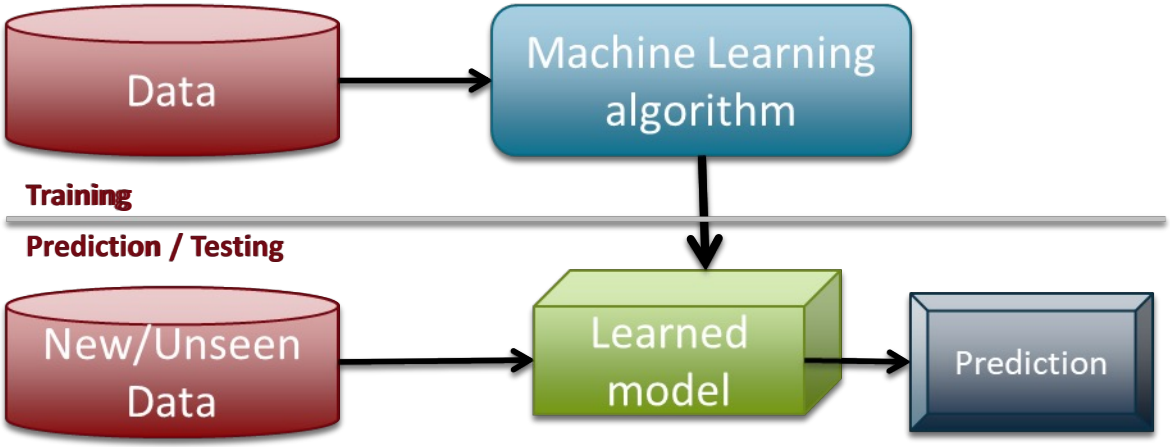
- Deep Learning Crash Course: https://chsasank.com/assets/images/crash_course/handout.pdf

Today…


Statistics vs. Machine Learning
Types of Learning
Sample Classifiers
Linear Regression
Logistic Regression

Machine Learning Basics

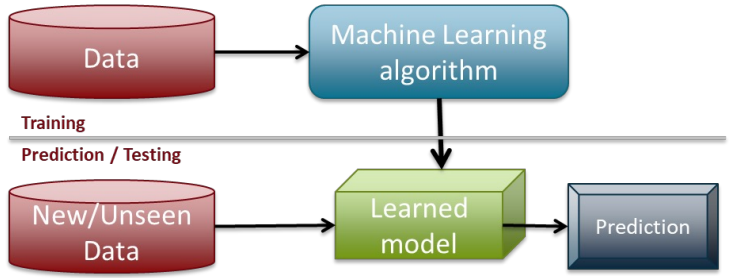
Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**



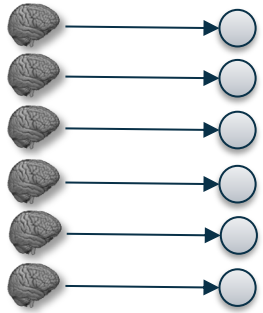
Methods that can learn from and make predictions on data

Types of Learning

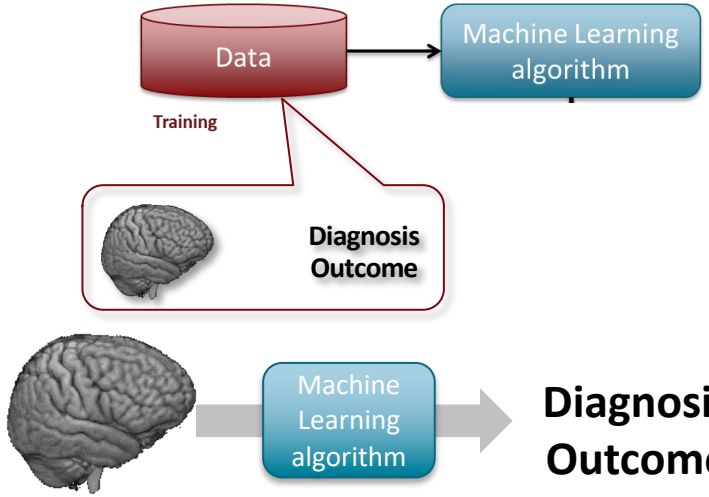
	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction



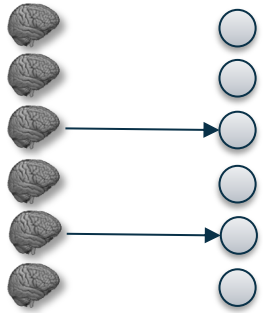
Types of Learning



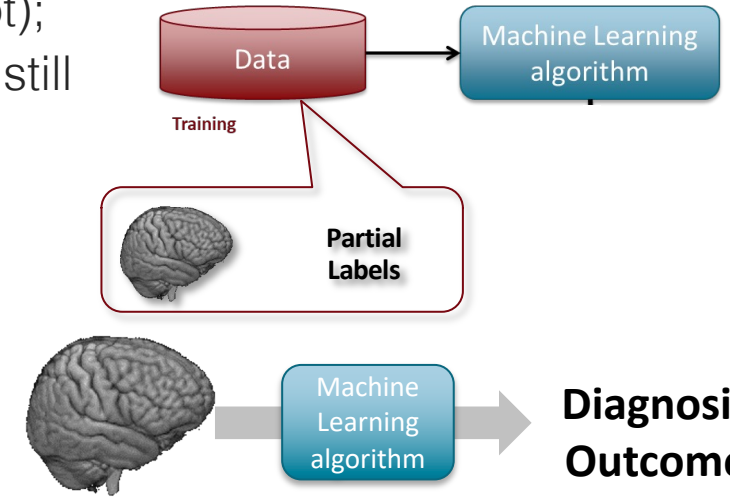
- Supervised Learning
  - data comes with the expected outputs; you tell the system what the categories or values are beforehand



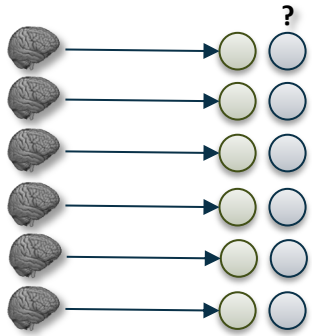
Types of Learning



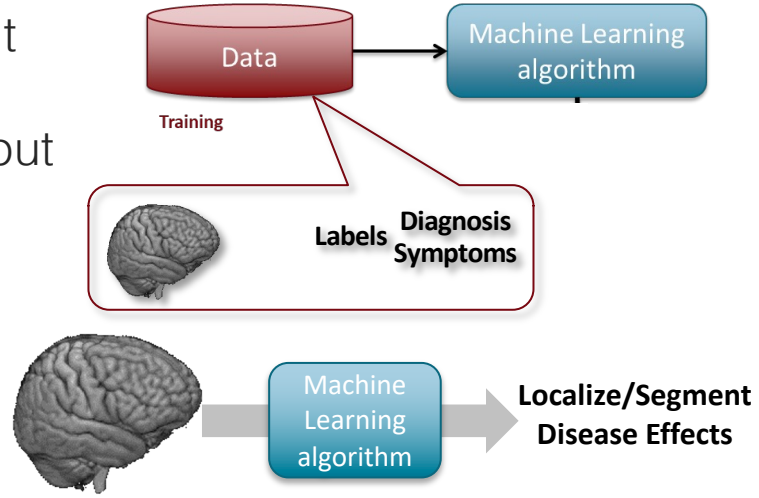
- Supervised Learning
- **Semi-Supervised Learning**
  - data partially comes with the expected outputs (some data points are labeled, some are not);
  - the system only sees partial labels, but you still expect the system to predict the same outcomes for each individual testing input



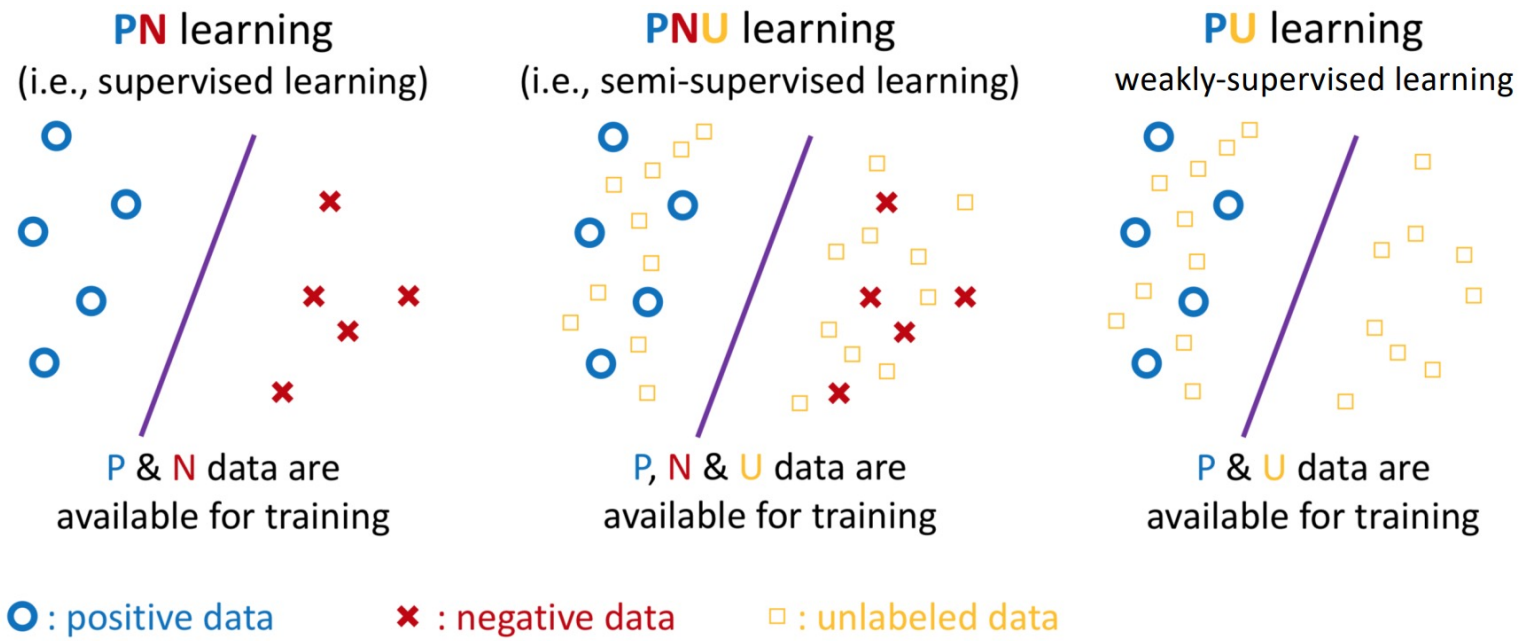
Types of Learning



- Supervised Learning
- Semi-Supervised Learning
- **Weakly-Supervised Learning**
  - data comes with the known mapping of input and some output;
  - but the system must predict some other output



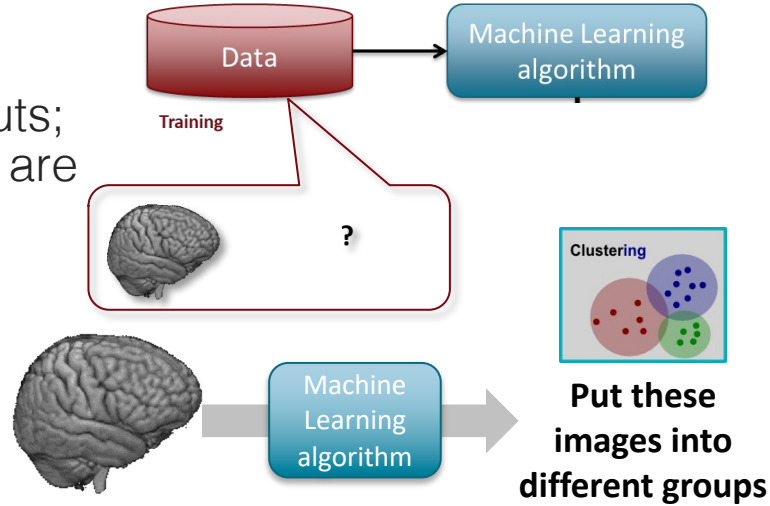
Types of Learning



https://niug1984.github.io/paper/niu_tdlw2018.pdf

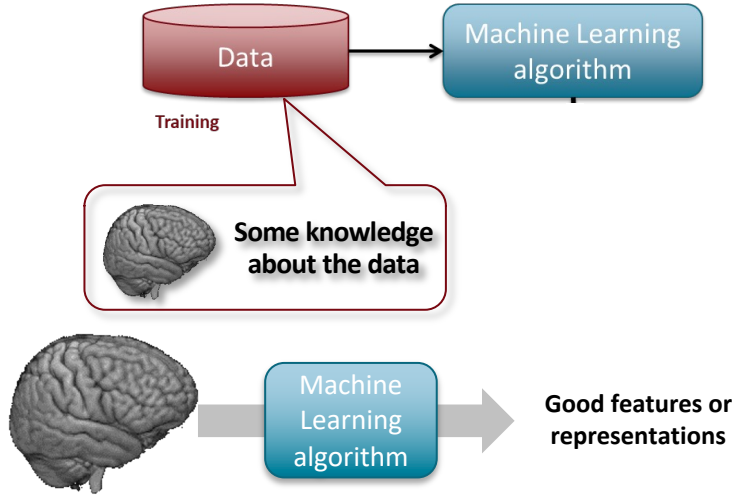
Types of Learning

- Supervised Learning
- Semi-Supervised Learning
- Weakly-Supervised Learning
- **Unsupervised Learning**
  - data does not come with the expected outputs; you let the system learn what the categories are



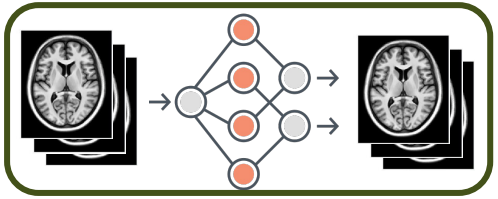
Types of Learning

- Supervised Learning
- Semi-Supervised Learning
- Weakly-Supervised Learning
- Unsupervised Learning
- **Self-Supervised Learning**
  - a representation learning method where a supervised task is created out of the unlabelled data
  - Self-supervised learning is used to reduce the data labelling cost and leverage the unlabelled data pool.

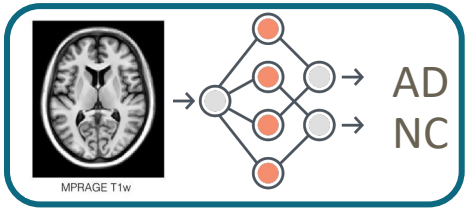


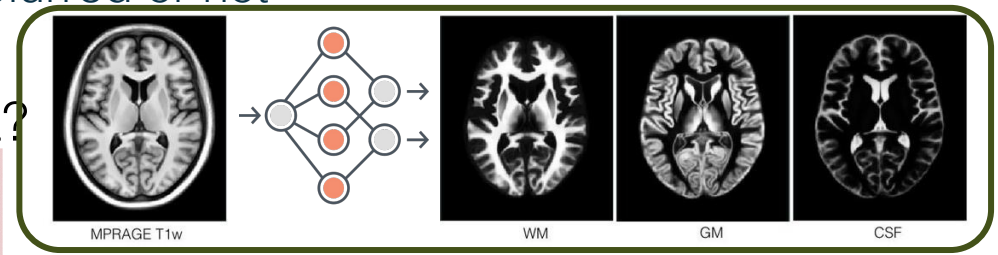
1) Supervised Learning;
2) Semi-Supervised Learning
3) Weakly-Supervised Learning
4) Unsupervised Learning
5) Self-Supervised Learning

Question

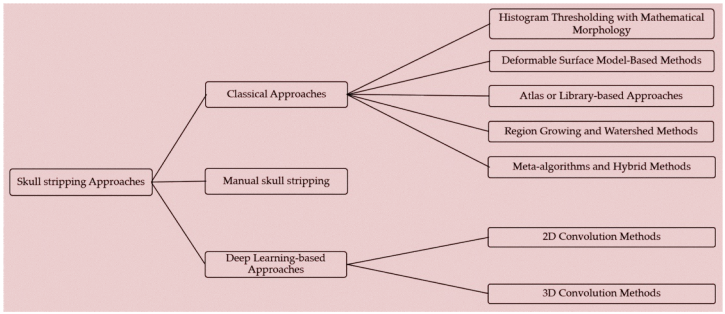


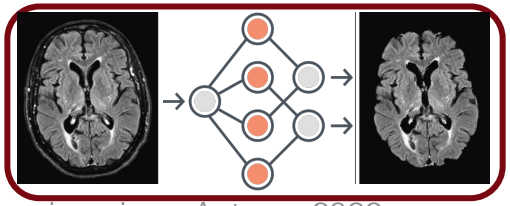
- What type of learning is each of the following tasks?
  a) Segmenting brain into GM, WM, CSF
  b) Learning to reconstruct a 3D MRI image
  c) Detecting if a person has Alzheimer's disease or not
  d) Detecting if images are motion blurred or not
  e) Skull striping

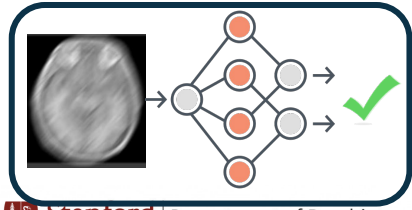




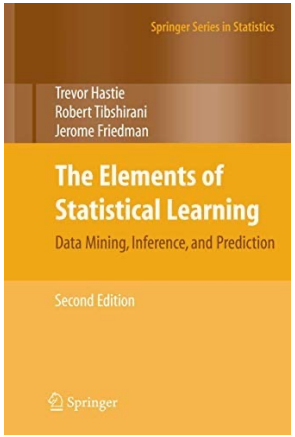
- For which task we can use SSL?



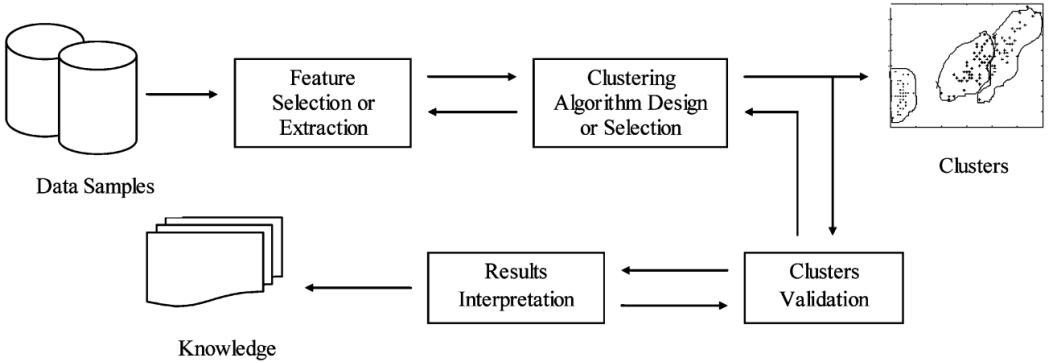




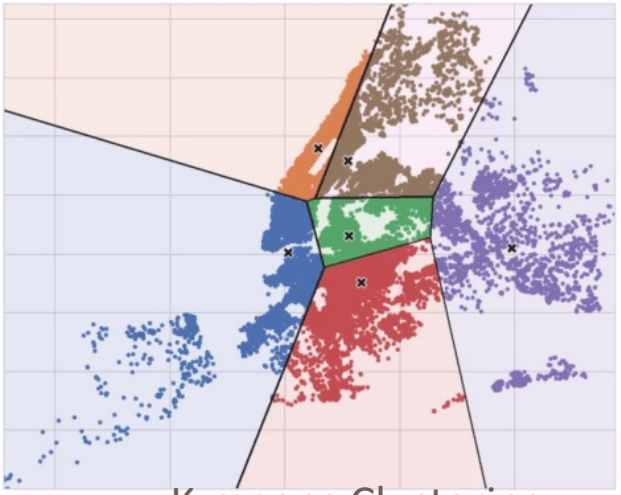
Reading assignment 1



- Overview of Supervised Learning (Elements of Statistical Learning, Chapter 2)
- Read about clustering algorithms (https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1427769)

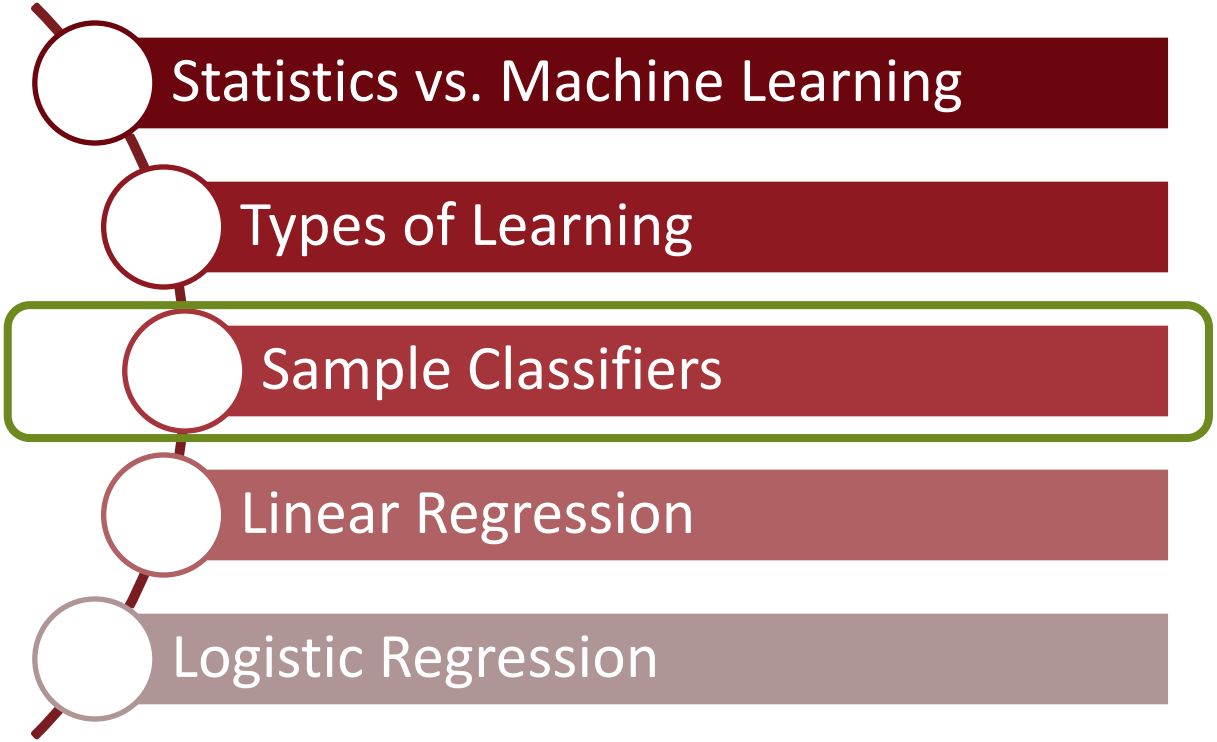


Xu, Rui, and Donald Wunsch. "Survey of clustering algorithms." *IEEE Transactions on neural networks* 16.3 (2005): 645-678.



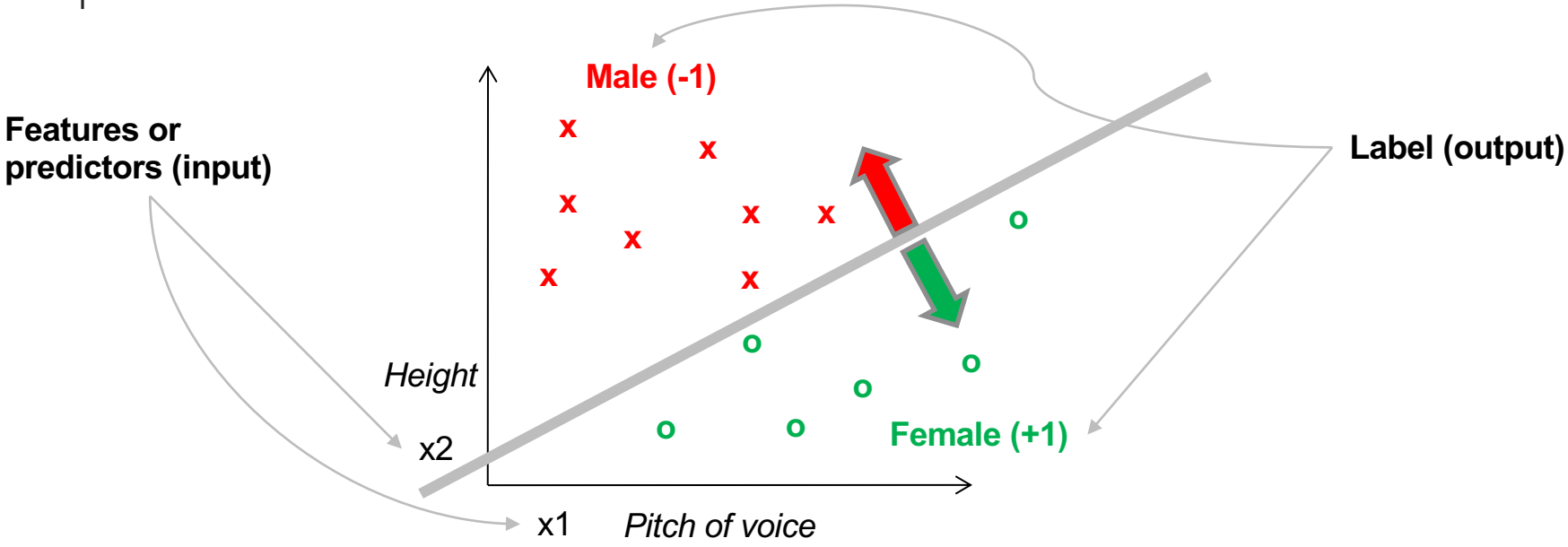
K-means Clustering

Today…

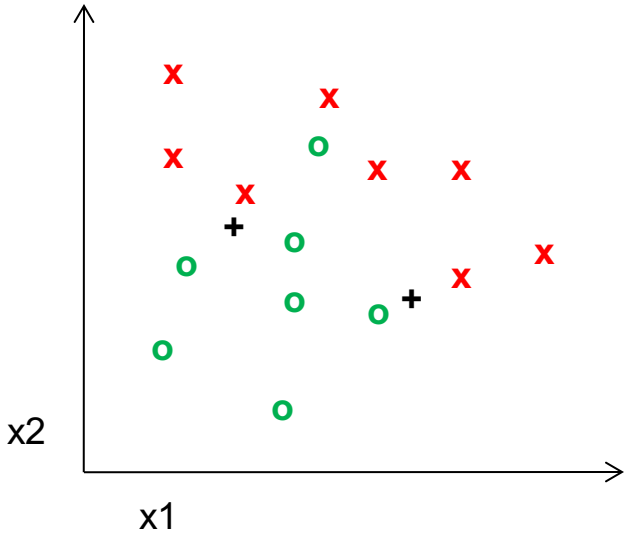

Statistics vs. Machine Learning
Types of Learning
Sample Classifiers
Linear Regression
Logistic Regression

Learning a classifier

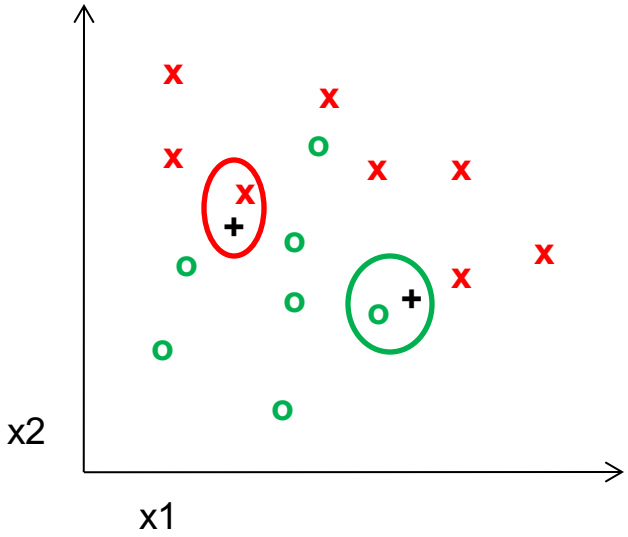
Given some set of features with corresponding labels, learn a function to predict the labels from the features



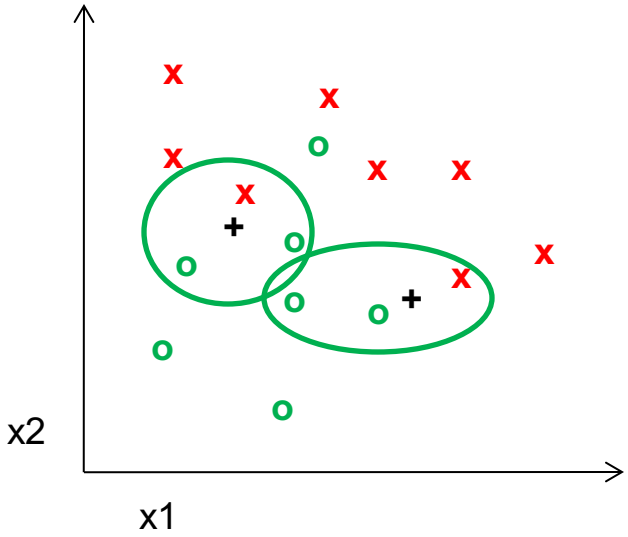
K-nearest neighbor



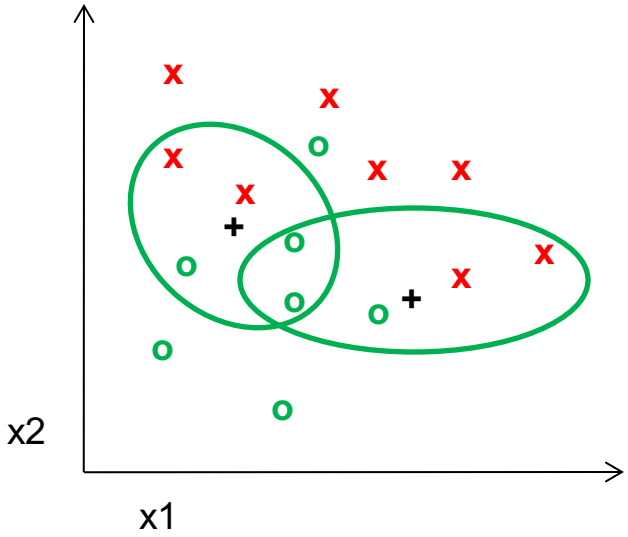
1-nearest neighbor



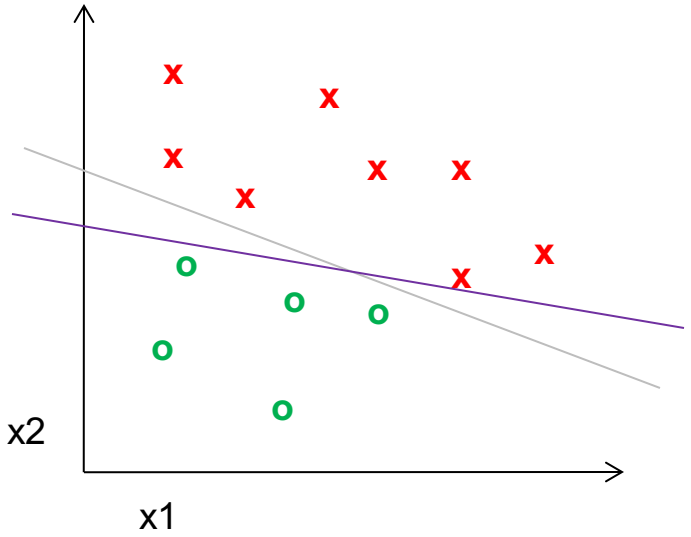
3-nearest neighbor



5-nearest neighbor



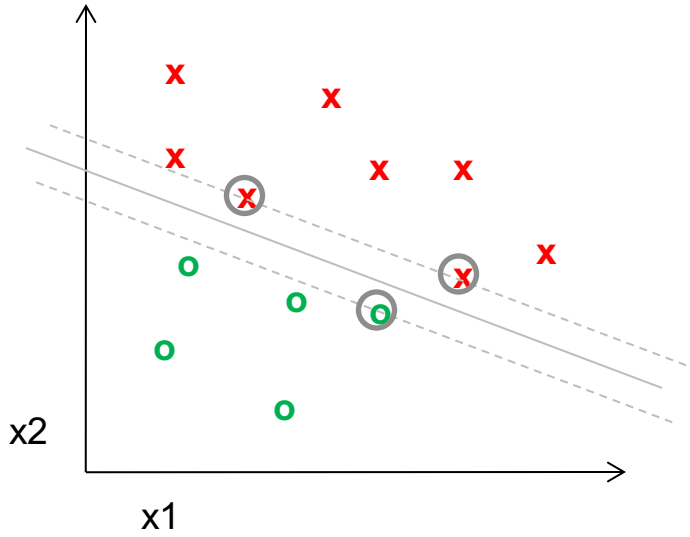
Classifiers: Support Vector Machine (SVM)



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

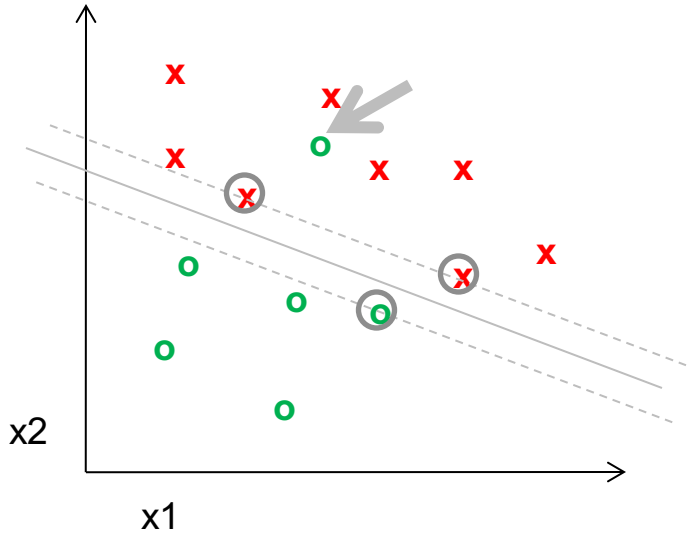
Classifiers: Linear SVM



- Find a *linear function* to separate the classes:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

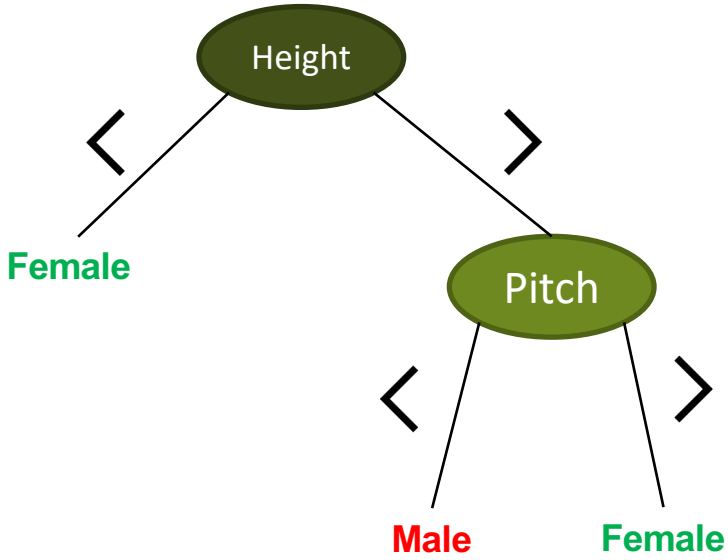
Classifiers: Linear SVM

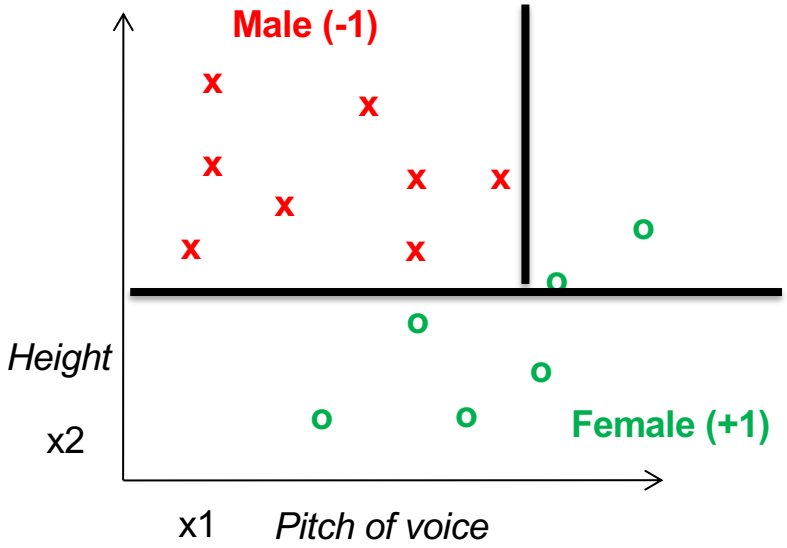


- Find a *linear function* to separate the classes:

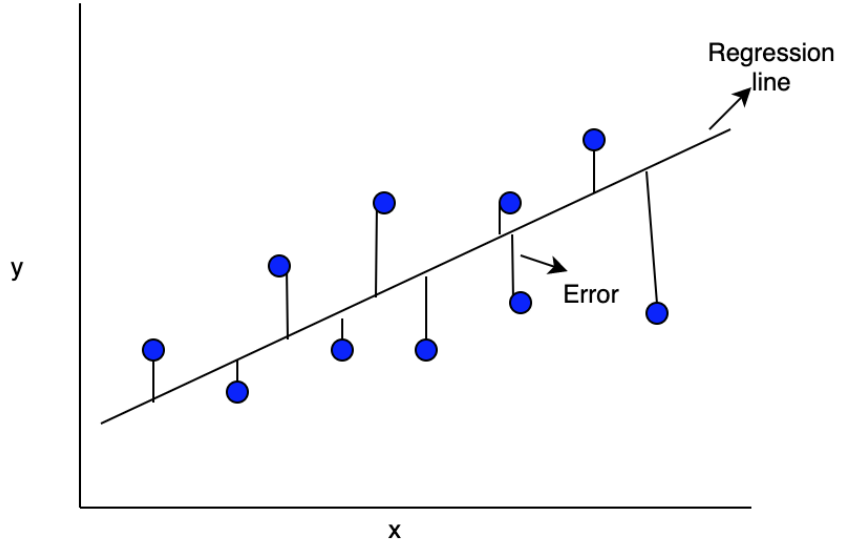
$$f(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Decision Tree

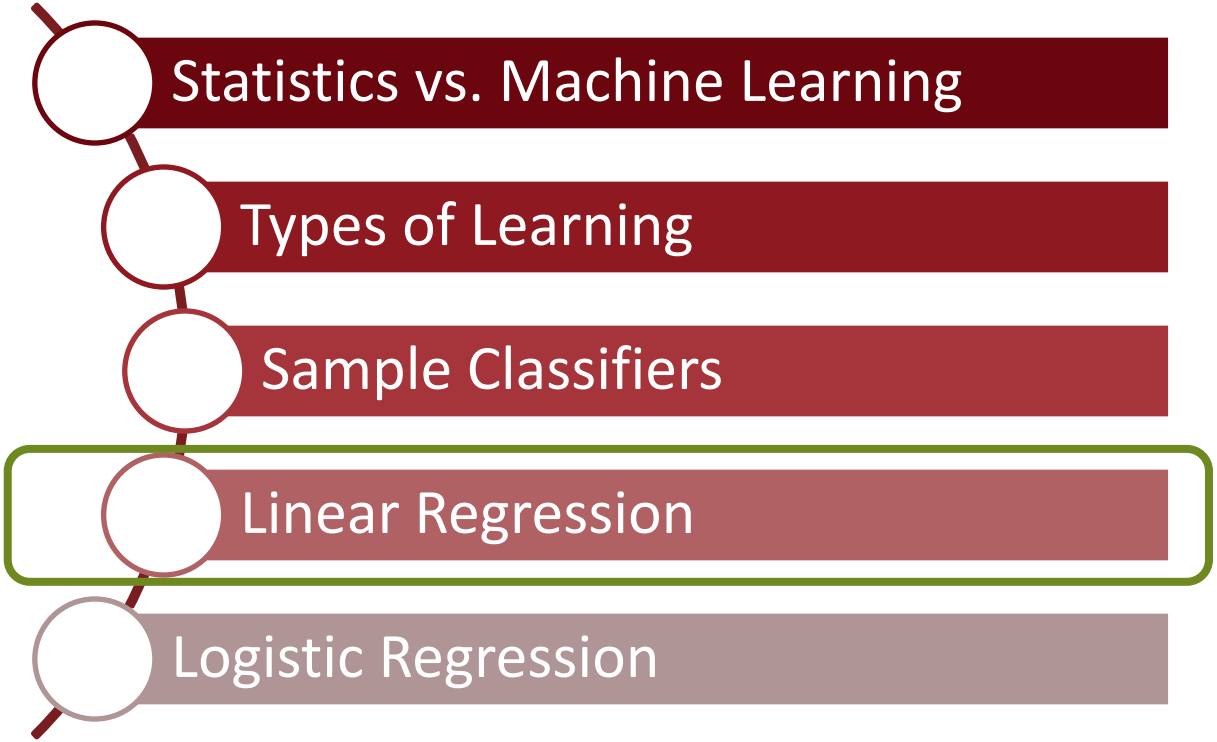




Linear Regression



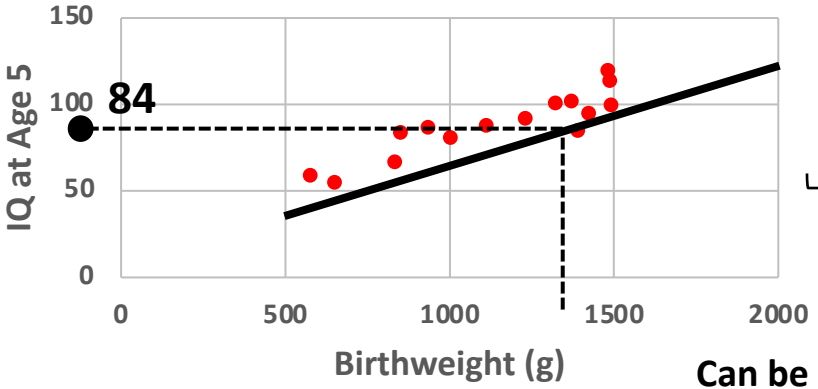
Today…


Statistics vs. Machine Learning
Types of Learning
Sample Classifiers
Linear Regression
Logistic Regression

How to Learn a Model?

How to *learn* a mathematical model, after which you can predict any *target* value (e.g., IQ) given a *feature* value (e.g., Birthweight)?

Birthweight (*x*)	IQ (*y*)
575	59
650	55
832	67
850	84
933	87
1001	81
1111	88
1230	92
1321	101
1370	102
1390	85
1422	95
1480	120
1487	114
1490	100



What will be the IQ of a baby who was born with weight 1090 gram?

Answer: 84

Inference

Can be done after the model (i.e., the line in the figure) is trained

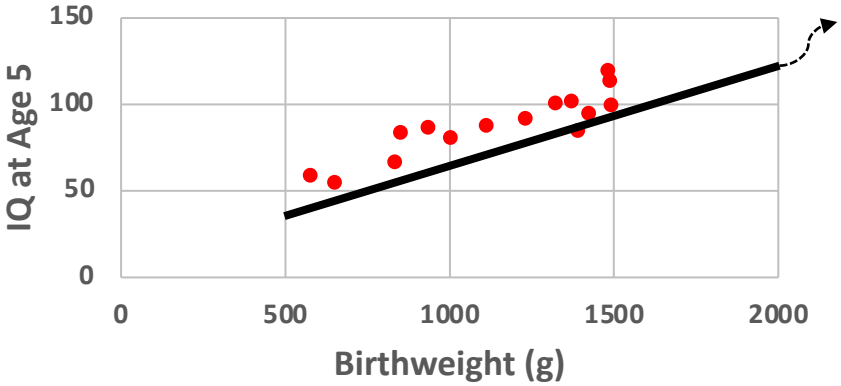
Trainig Dataset → **Can be used to train the model**

Learning a Model

- How do we represent a line in mathematics?
  - $y = mx + b$

Birthweight (*x*)	IQ (*y*)
575	59
650	55
832	67
850	84
933	87
1001	81
1111	88
1230	92
1321	101
1370	102
1390	85
1422	95
1480	120
1487	114
1490	100

Dataset

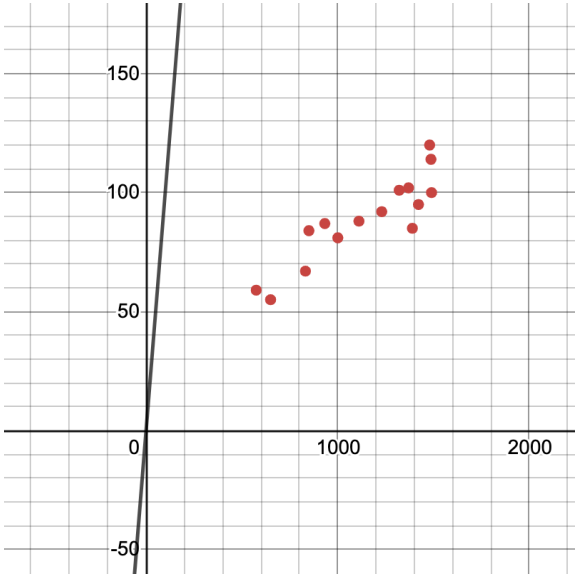


For this best fitting line:
m = 0.052
b = 29.21

But, how can we find (or *learn*) *m* and *b*?

Learning a Model

- Let us try different random values of *m* and *b*:
  - *m* = 1 and *b* = 0

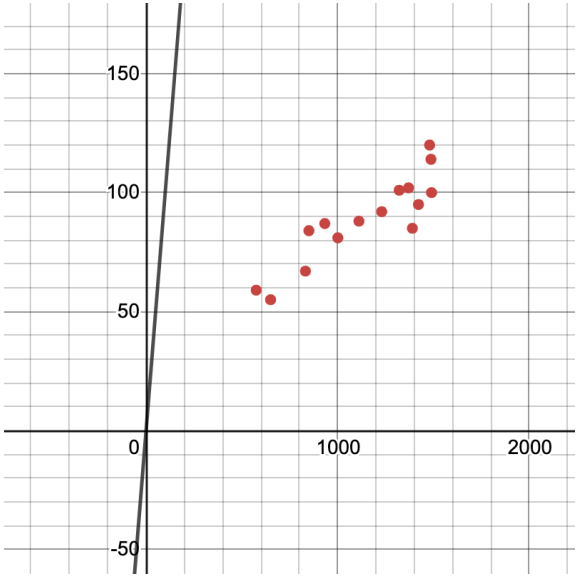


Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (y')
575	59	575
650	55	650
832	67	832
850	84	850
933	87	933
1001	81	1001
1111	88	1111
1230	92	1230
1321	101	1321
1370	102	1370
1390	85	1390
1422	95	1422
1480	120	1480
1487	114	1487
1490	100	1490

$$y' = mx + b$$
$$= 1*575 + 0$$
$$= 575$$

Learning a Model

- Let us try different random values of *m* and *b*:
  - *m* = 1 and *b* = 0



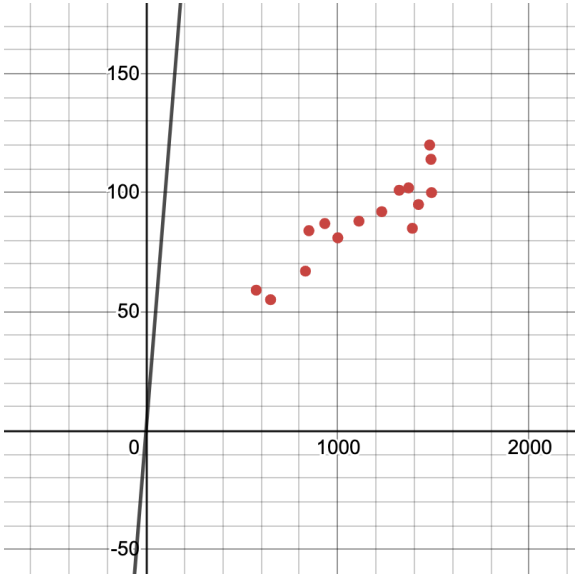
Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (y')
575	59	575
650	55	650
832	67	832
850	84	850
933	87	933
1001	81	1001
1111	88	1111
1230	92	1230
1321	101	1321
1370	102	1370
1390	85	1390
1422	95	1422
1480	120	1480
1487	114	1487
1490	100	1490

$y' = \boldsymbol{m}x + \boldsymbol{b}$
$= 1*1001 + 0$
$= 1001$

How close are these predicted IQs to the actual ones?

Learning a Model

- Let us try different random values of *m* and *b*:
  - *m* = 1 and *b* = 0



The ***Error***

Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (y')	Predicted – Actual (y' – y)
575	59	575	516
650	55	650	595
832	67	832	765
850	84	850	766
933	87	933	846
1001	81	1001	920
1111	88	1111	1023
1230	92	1230	1138
1321	101	1321	1220
1370	102	1370	1268
1390	85	1390	1305
1422	95	1422	1327
1480	120	1480	1360
1487	114	1487	1373
1490	100	1490	1390

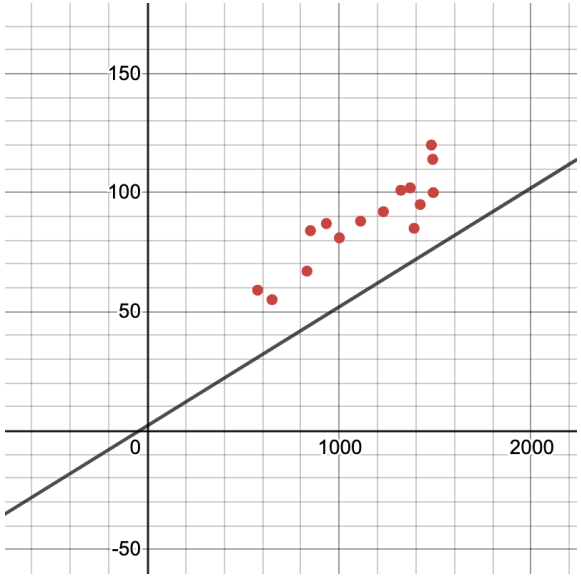
Sum = 15812

≡

∑ = 15812

Learning a Model

- Let us try different random values of *m* and *b*:
  - *m* = 0.05 and *b* = 2



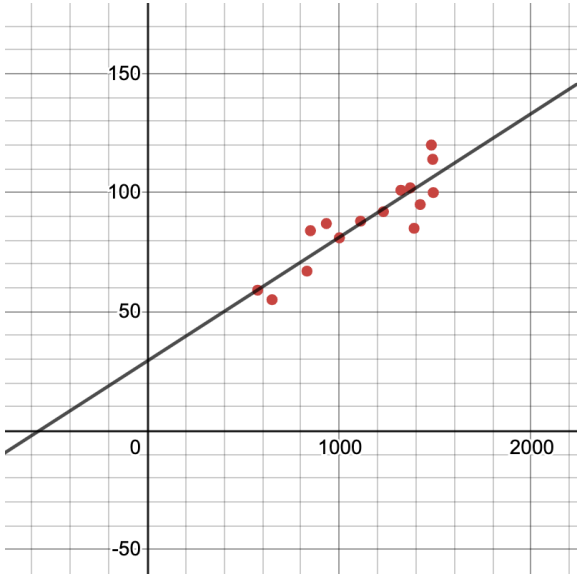
Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (y')	Predicted – Actual (y' – y)
575	59	30.75	-28.25
650	55	34.5	-20.5
832	67	43.6	-23.4
850	84	44.5	-39.5
933	87	48.65	-38.35
1001	81	52.05	-28.95
1111	88	57.55	-30.45
1230	92	63.5	-28.5
1321	101	68.05	-32.95
1370	102	70.5	-31.5
1390	85	71.5	-13.5
1422	95	73.1	-21.9
1480	120	76	-44
1487	114	76.35	-37.65
1490	100	76.5	-23.5

Error:

∑ = -442.9

Learning a Model

- Let us try different random values of *m* and *b*:
  - *m* = 0.052 and *b* = 29.21



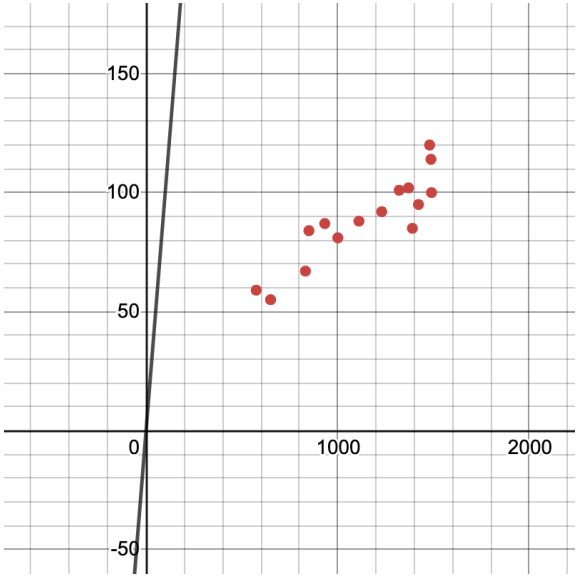
Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (*y'*)	Predicted – Actual (*y'* – *y*)
575	59	59.11	0.11
650	55	63.01	8.01
832	67	72.474	5.474
850	84	73.41	-10.59
933	87	77.726	-9.274
1001	81	81.262	0.262
1111	88	86.982	-1.018
1230	92	93.17	1.17
1321	101	97.902	-3.098
1370	102	100.45	-1.55
1390	85	101.49	16.49
1422	95	103.154	8.154
1480	120	106.17	-13.83
1487	114	106.534	-7.466
1490	100	106.69	6.69

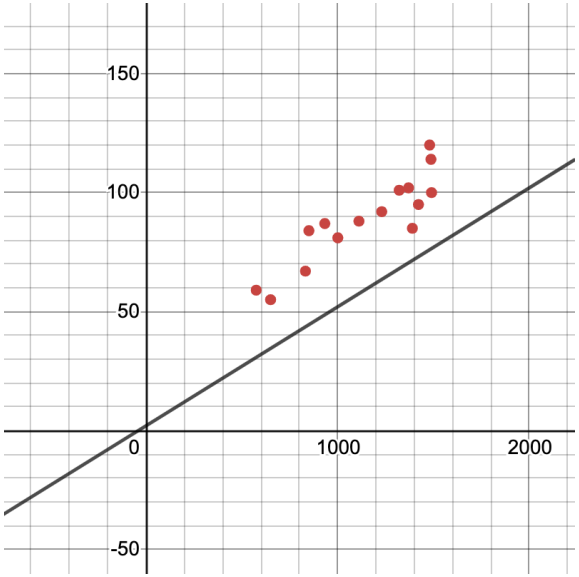
Error:

∑ = -0.466

Learning a Model

- Let us observe the three options besides each other





m = 1 and ***b*** = 0

m = 0.05 and ***b*** = 2

m = 0.052 and ***b*** = 29.21

Mean Squared Error (MSE)

- Let us compare their errors

Mean Squared Error (MSE)

- Let us compare their errors

		m = 1 and ***b*** = 0		***m*** = 0.05 and ***b*** = 2		***m*** = 0.052 and ***b*** = 29.21	
Birthweight (*x*)	**Actual IQ (*y*)**	**Predicted IQ (y')**	**(Predicted – Actual)^2 (y' – y)^2**	**Predicted IQ (y')**	**(Predicted – Actual)^2 (y' – y)^2**	**Predicted IQ (y')**	**(Predicted – Actual)^2 (y' – y)^2**
575	59	59.11	266256	30.75	798.0625	59.11	0.0121
650	55	63.01	354025	34.5	420.25	63.01	64.1601
832	67	72.474	585225	43.6	547.56	72.474	29.964676
850	84	73.41	586756	44.5	1560.25	73.41	112.1481
933	87	77.726	715716	48.65	1470.7225	77.726	86.007076
1001	81	81.262	846400	52.05	838.1025	81.262	0.068644
1111	88	86.982	1046529	57.55	927.2025	86.982	1.036324
1230	92	93.17	1295044	63.5	812.25	93.17	1.3689
1321	101	97.902	1488400	68.05	1085.7025	97.902	9.597604
1370	102	100.45	1607824	70.5	992.25	100.45	2.4025
1390	85	101.49	1703025	71.5	182.25	101.49	271.9201
1422	95	103.154	1760929	73.1	479.61	103.154	66.487716
1480	120	106.17	1849600	76	1936	106.17	191.2689
1487	114	106.534	1885129	76.35	1417.5225	106.534	55.741156
1490	100	106.69	1932100	76.5	552.25	106.69	44.7561

SQUARED ERRORS: ∑ = 17922958 ∑ = 14019.9 ∑ = 936.9

Mean Squared Error (MSE)

- Let us compare their errors

			m = 1 and ***b*** = 0		***m*** = 0.05 and ***b*** = 2		***m*** = 0.052 and ***b*** = 29.21	
	Birthweight (*x*)	Actual IQ (*y*)	Predicted IQ (y′)	(Predicted – Actual)^2 (y′ – y)^2	Predicted IQ (y′)	(Predicted – Actual)^2 (y′ – y)^2	Predicted IQ (y′)	(Predicted – Actual)^2 (y′ – y)^2
15 examples	575	59	59.11	266256	30.75	798.0625	59.11	0.0121
	650	55	63.01	354025	34.5	420.25	63.01	64.1601
	832	67	72.474	585225	43.6	547.56	72.474	29.964676
	850	84	73.41	586756	44.5	1560.25	73.41	112.1481
	933	87	77.726	715716	48.65	1470.7225	77.726	86.007076
	1001	81	81.262	846400	52.05	838.1025	81.262	0.068644
	1111	88	86.982	1046529	57.55	927.2025	86.982	1.036324
	1230	92	93.17	1295044	63.5	812.25	93.17	1.3689
	1321	101	97.902	1488400	68.05	1085.7025	97.902	9.597604
	1370	102	100.45	1607824	70.5	992.25	100.45	2.4025
	1390	85	101.49	1703025	71.5	182.25	101.49	271.9201
	1422	95	103.154	1760929	73.1	479.61	103.154	66.487716
	1480	120	106.17	1849600	76	1936	106.17	191.2689
	1487	114	106.534	1885129	76.35	1417.5225	106.534	55.741156
	1490	100	106.69	1932100	76.5	552.25	106.69	44.7561

MEAN SQUARED ERRORS: ∑ = 17922958/15 ∑ = 14019.9/15 ∑ = 936.9/15

Mean Squared Error (MSE)

- Let us compare their errors

			$m = 1$ and $b = 0$		$m = 0.05$ and $b = 2$		$m = 0.052$ and $b = 29.21$	
	Birthweight (x)	Actual IQ (y)	Predicted IQ (y')	(Predicted – Actual)^2 $(y' - y)^2$	Predicted IQ (y')	(Predicted – Actual)^2 $(y' - y)^2$	Predicted IQ (y')	(Predicted – Actual)^2 $(y' - y)^2$
15 examples	575	59	59.11	266256	30.75	798.0625	59.11	0.0121
	650	55	63.01	354025	34.5	420.25	63.01	64.1601
	832	67	72.474	585225	43.6	547.56	72.474	29.964676
	850	84	73.41	586756	44.5	1560.25	73.41	112.1481
	933	87	77.726	715716	48.65	1470.7225	77.726	86.007076
	1001	81	81.262	846400	52.05	838.1025	81.262	0.068644
	1111	88	86.982	1046529	57.55	927.2025	86.982	1.036324
	1230	92	93.17	1295044	63.5	812.25	93.17	1.3689
	1321	101	97.902	1488400	68.05	1085.7025	97.902	9.597604
	1370	102	100.45	1607824	70.5	992.25	100.45	2.4025
	1390	85	101.49	1703025	71.5	182.25	101.49	271.9201
	1422	95	103.154	1760929	73.1	479.61	103.154	66.487716
	1480	120	106.17	1849600	76	1936	106.17	191.2689
	1487	114	106.534	1885129	76.35	1417.5225	106.534	55.741156
	1490	100	106.69	1932100	76.5	552.25	106.69	44.7561

MEAN SQUARED ERRORS: $\sum = 1194863.8$ $\sum = 934.6$ $\sum = 62.4$

Mean Squared Error (MSE)

- Let us compare their errors

			$m = 1$ and $b = 0$		$m = 0.05$ and $b = 2$		$m = 0.052$ and $b = 29.21$	
	Birthweight (x)	Actual IQ (y)	Predicted IQ (y')	(Predicted – Actual)^2 ($y' - y$)^2	Predicted IQ (y')	(Predicted – Actual)^2 ($y' - y$)^2	Predicted IQ (y')	(Predicted – Actual)^2 ($y' - y$)^2
15 examples	575	59	59.11	266256	30.75	798.0625	59.11	0.0121
	650	55	63.01	354025	34.5	420.25	63.01	64.1601
	832	67	72.474	585225	43.6	547.56	72.474	29.964676
	850	84	73.41	586756	44.5	1560.25	73.41	112.1481
	933	87	77.726	715716	48.65	1470.7225	77.726	86.007076
	1001	81	81.262	846400	52.05	838.1025	81.262	0.068644
	1111	88	86.982	1046529	57.55	927.2025	86.982	1.036324
	1230	92	93.17	1295044	63.5	812.25	93.17	1.3689
	1321	101	97.902	1488400	68.05	1085.7025	97.902	9.597604
	1370	102	100.45	1607824	70.5	992.25	100.45	2.4025
	1390	85	101.49	1703025	71.5	182.25	101.49	271.9201
	1422	95	103.154	1760929	73.1	479.61	103.154	66.487716
	1480	120	106.17	1849600	76	1936	106.17	191.2689
	1487	114	106.534	1885129	76.35	1417.5225	106.534	55.741156
	1490	100	106.69	1932100	76.5	552.25	106.69	44.7561

MEAN SQUARED ERRORS: $\sum = 1194863.8/2$ $\sum = 934.6/2$ $\sum = 62.4/2$

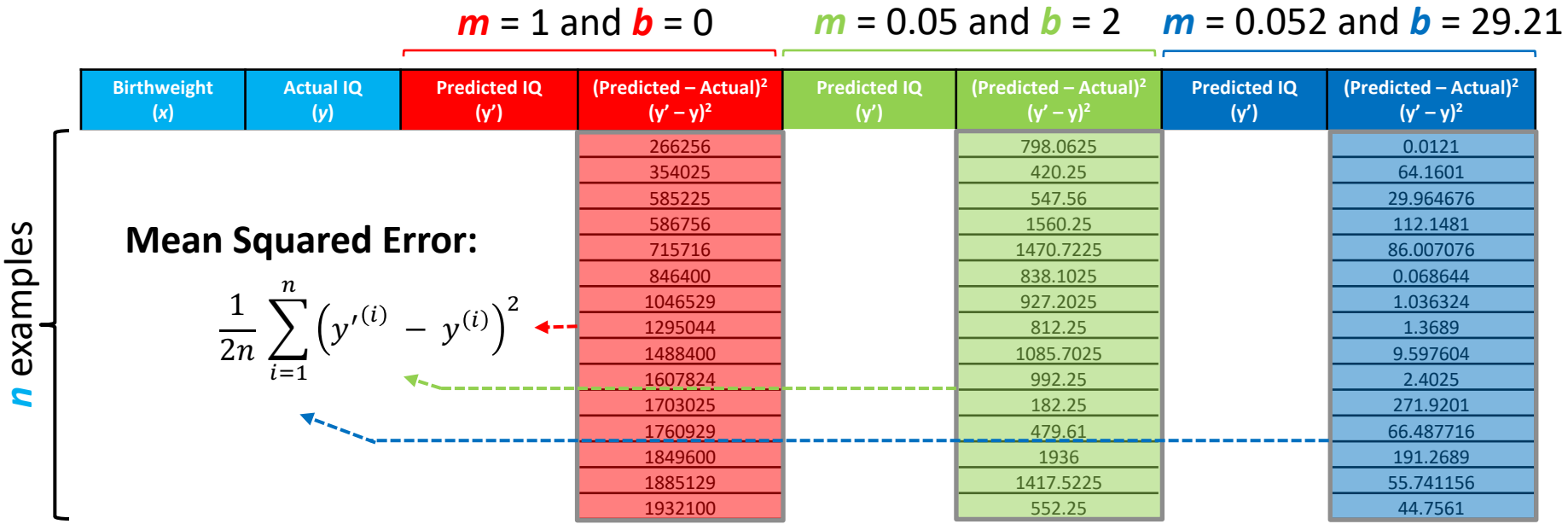
Mean Squared Error (MSE)

- Let us compare their errors

			m = 1 and ***b*** = 0		***m*** = 0.05 and ***b*** = 2		***m*** = 0.052 and ***b*** = 29.21	
	Birthweight (*x*)	**Actual IQ (*y*)**	**Predicted IQ (y′)**	**(Predicted – Actual)^2 (y′ – y)^2**	**Predicted IQ (y′)**	**(Predicted – Actual)^2 (y′ – y)^2**	**Predicted IQ (y′)**	**(Predicted – Actual)^2 (y′ – y)^2**
15 examples	575	59	59.11	266256	30.75	798.0625	59.11	0.0121
	650	55	63.01	354025	34.5	420.25	63.01	64.1601
	832	67	72.474	585225	43.6	547.56	72.474	29.964676
	850	84	73.41	586756	44.5	1560.25	73.41	112.1481
	933	87	77.726	715716	48.65	1470.7225	77.726	86.007076
	1001	81	81.262	846400	52.05	838.1025	81.262	0.068644
	1111	88	86.982	1046529	57.55	927.2025	86.982	1.036324
	1230	92	93.17	1295044	63.5	812.25	93.17	1.3689
	1321	101	97.902	1488400	68.05	1085.7025	97.902	9.597604
	1370	102	100.45	1607824	70.5	992.25	100.45	2.4025
	1390	85	101.49	1703025	71.5	182.25	101.49	271.9201
	1422	95	103.154	1760929	73.1	479.61	103.154	66.487716
	1480	120	106.17	1849600	76	1936	106.17	191.2689
	1487	114	106.534	1885129	76.35	1417.5225	106.534	55.741156
	1490	100	106.69	1932100	76.5	552.25	106.69	44.7561
	MEAN SQUARED ERRORS:			**∑ = 597431.9**		**∑ = 467.3**		**∑ = 31.2**

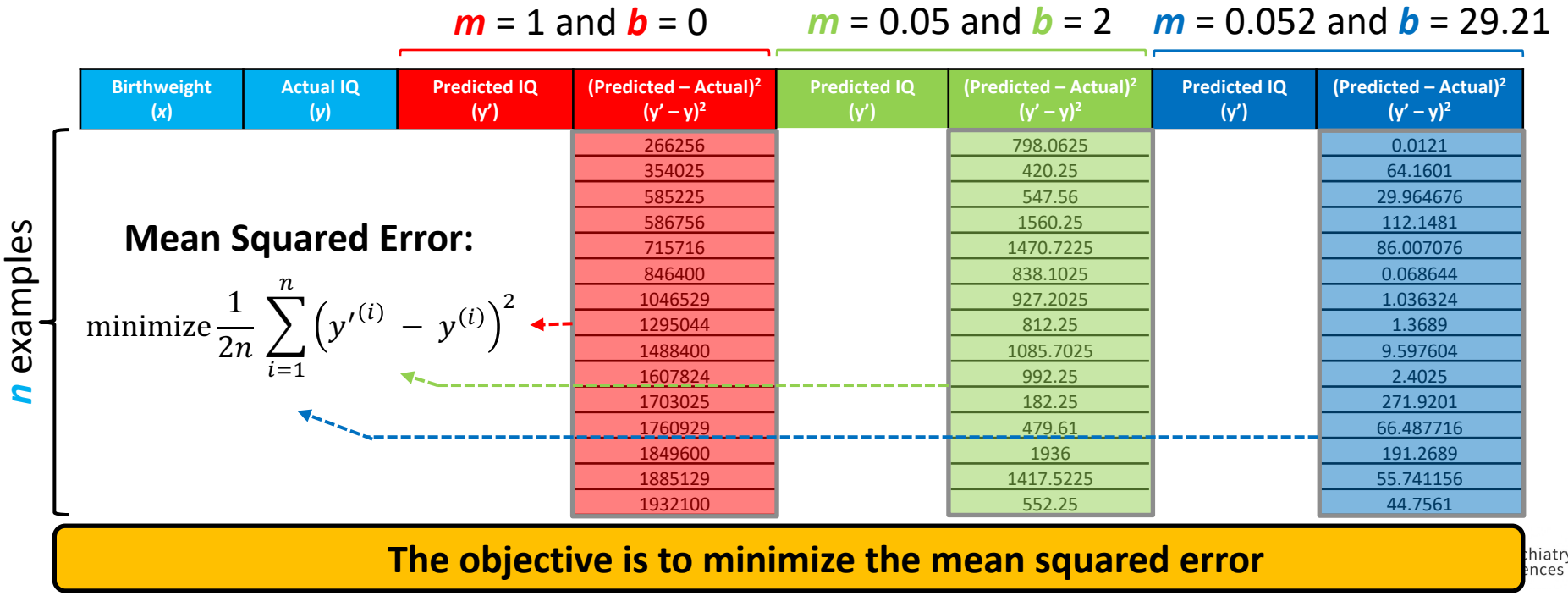
Mean Squared Error (MSE)

- Let us compare their errors



Minimizing MSE to Learn a Model

- Let us compare their errors



Learning a *Line* via Minimizing MSE

- How to learn a line of equation y' = mx + b given a *labelled* dataset?
  - By minimizing *mean squared error*. That is:

$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{m,b}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((mx + b)^{(i)} - y^{(i)} \right)^2$$

Learning a *Linear Regression Model* via Minimizing a *Cost Function*

- Or, how to learn a *linear regression model* $\boldsymbol{h_\theta(x) = \theta_0 + \theta_1 x}$ given a labelled dataset ($\boldsymbol{\theta}_0$ = b, $\boldsymbol{\theta}_1$= m, and $\boldsymbol{h_\theta(x)}$= y' when using y' = mx + b)?
  - By minimizing *mean squared error*. That is:

This problem is referred to as an ***optimization problem*** with the objective of:

$$\underset{\theta_0,\theta_1}{\text{minimizing}}\ \boldsymbol{C(\theta_0, \theta_1)}$$

$$\text{minimize}\ \frac{1}{2n}\sum_{i=1}^{n}\left(y'^{(i)} - y^{(i)}\right)^2$$

$$\equiv$$

$$\underset{\theta_0,\theta_1}{\text{minimize}}\ \frac{1}{2n}\sum_{i=1}^{n}\left((\theta_0 + \theta_1 x)^{(i)} - y^{(i)}\right)^2$$

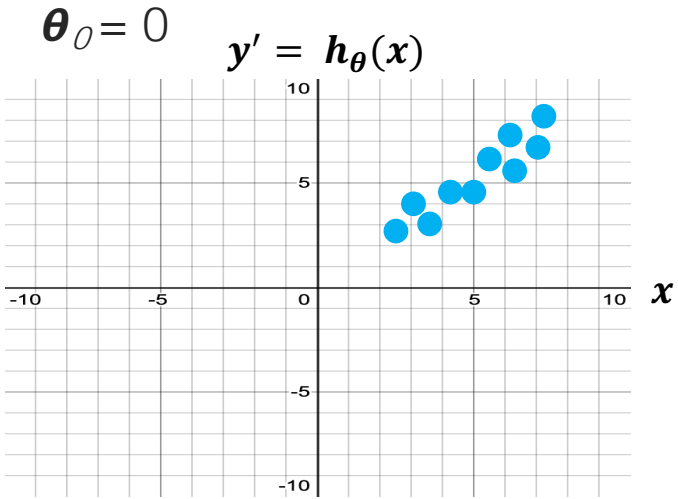
Known as ***cost function*** $\boldsymbol{C(\theta_0, \theta_1)}$

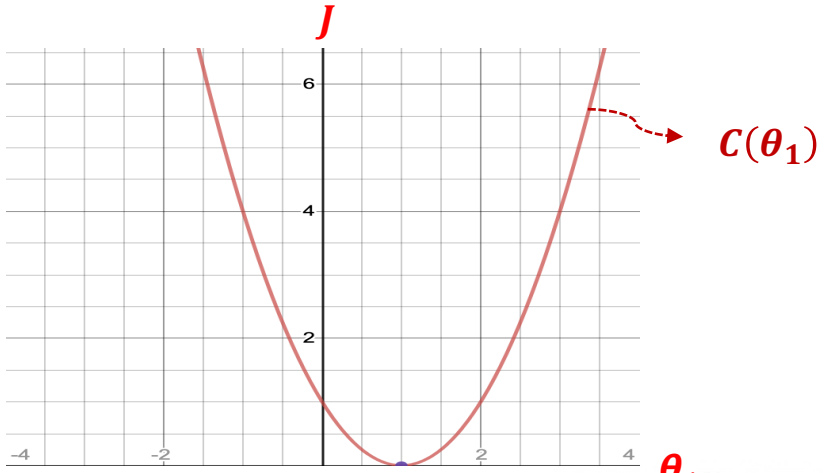
$$\equiv$$

$$\underset{\theta_0,\theta_1}{\text{minimize}}\ \boldsymbol{C(\theta_0, \theta_1)}$$

Learning a *Linear Regression Model* via Minimizing a *Cost Function*

- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,





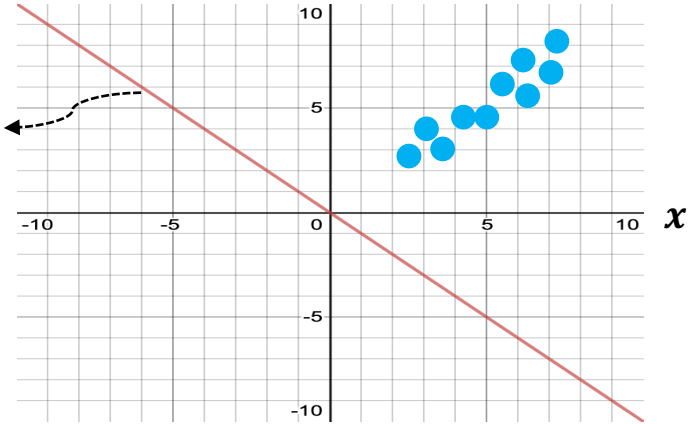
Learning a *Linear Regression Model* via Minimizing a *Cost Function*

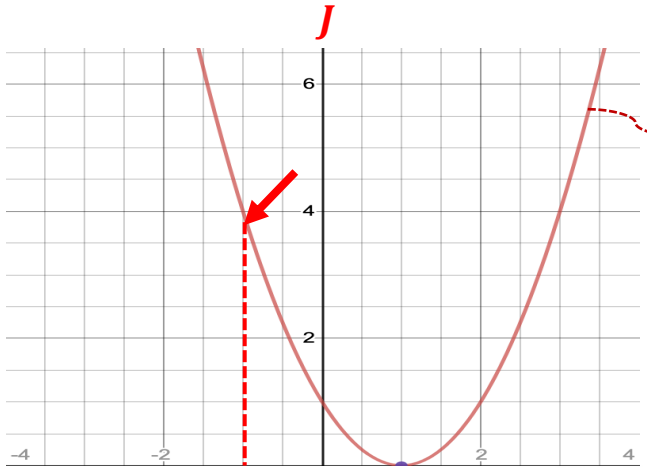
- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,

$\theta_0 = 0$

$y' = h_\theta(x)$

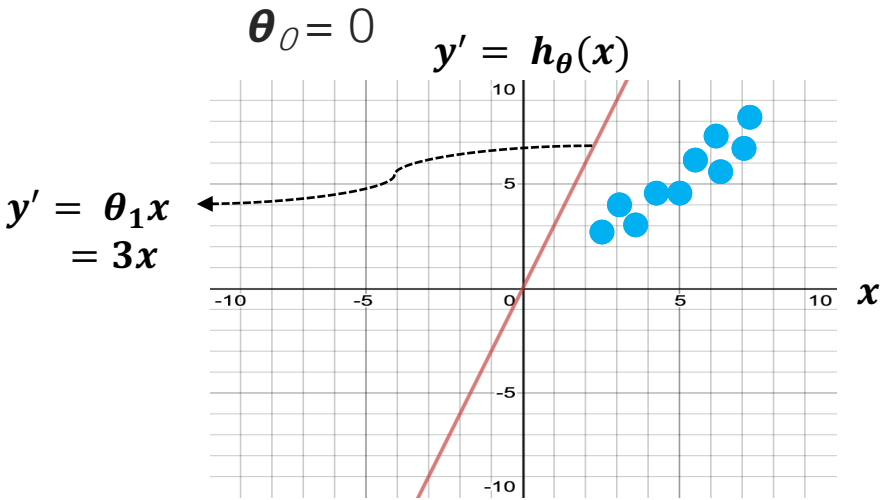
$y' = \theta_1 x$
$= -1x$

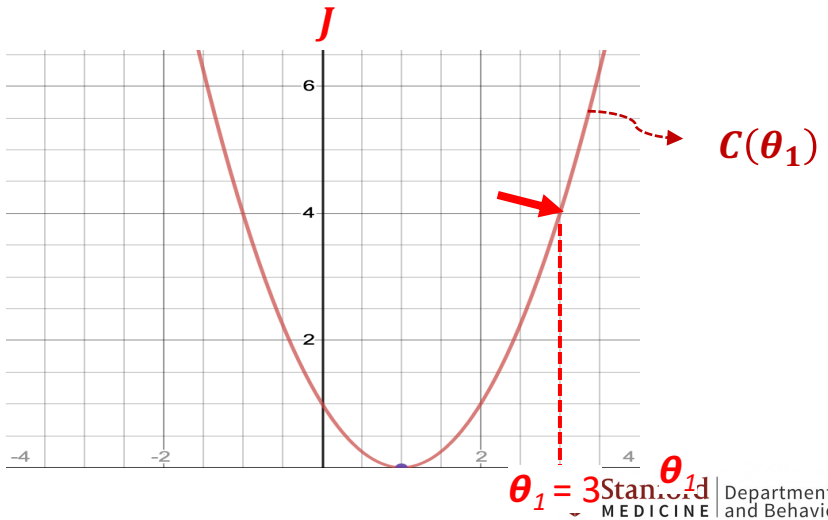




Learning a *Linear Regression Model* via Minimizing a *Cost Function*

- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,





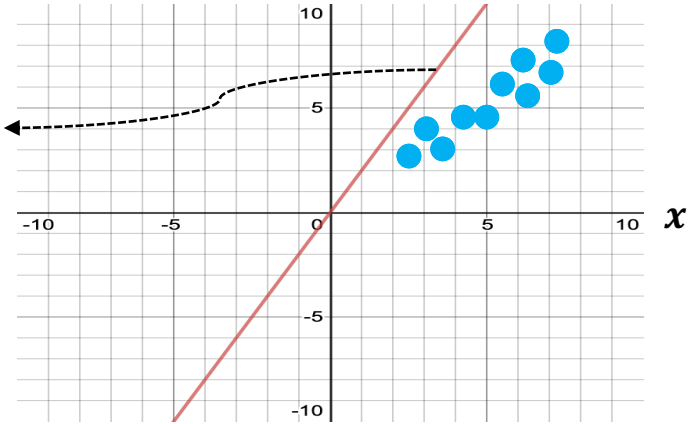
Learning a *Linear Regression Model* via Minimizing a *Cost Function*

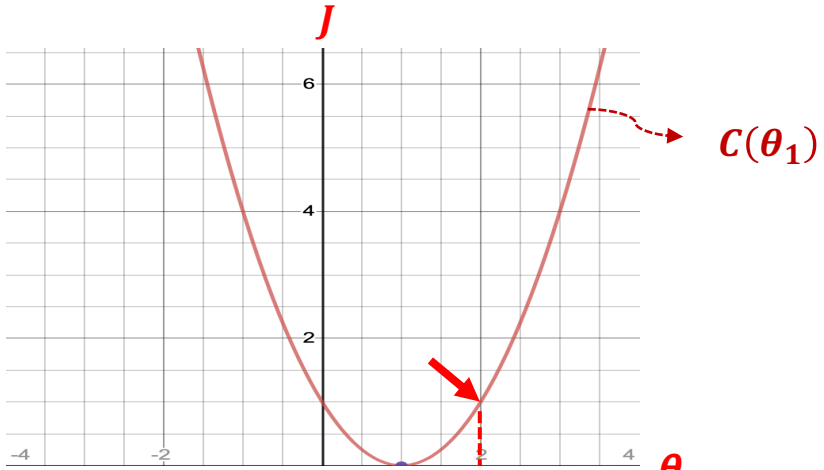
- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,

$\theta_0 = 0$

$y' = h_\theta(x)$

$y' = \theta_1 x$
$= 2x$





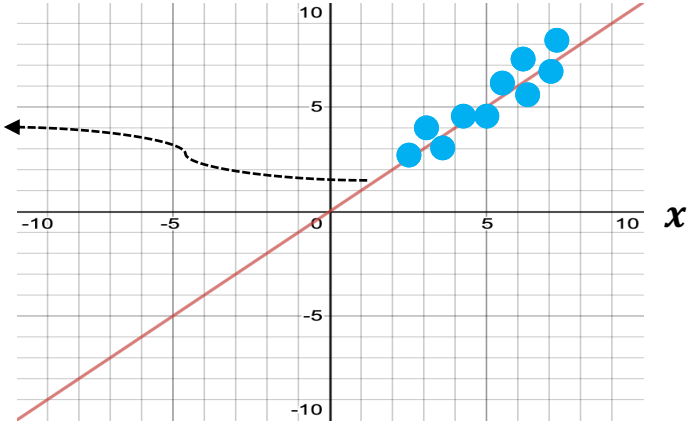
Learning a *Linear Regression Model* via Minimizing a *Cost Function*

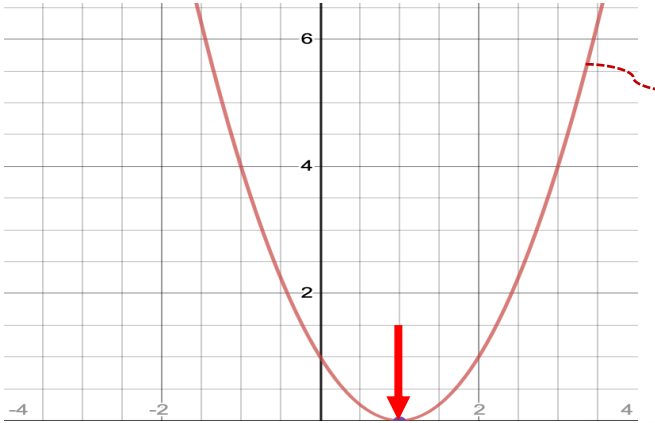
- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,

$\theta_0 = 0$

$y' = h_\theta(x)$

$y' = \theta_1 x$
$= 1x$





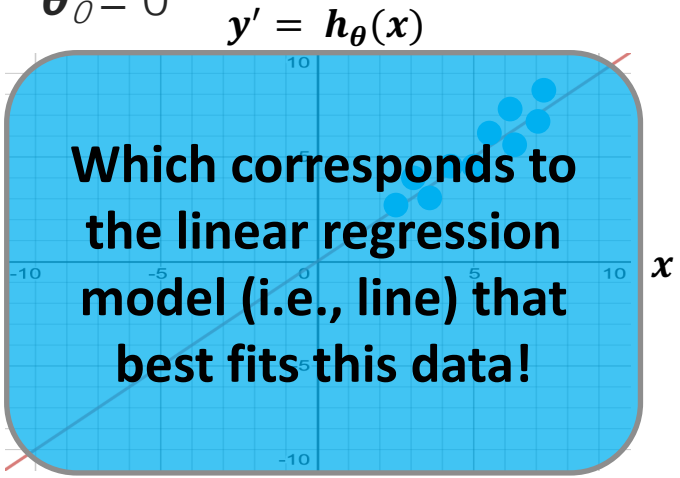
$C(\theta_1)$

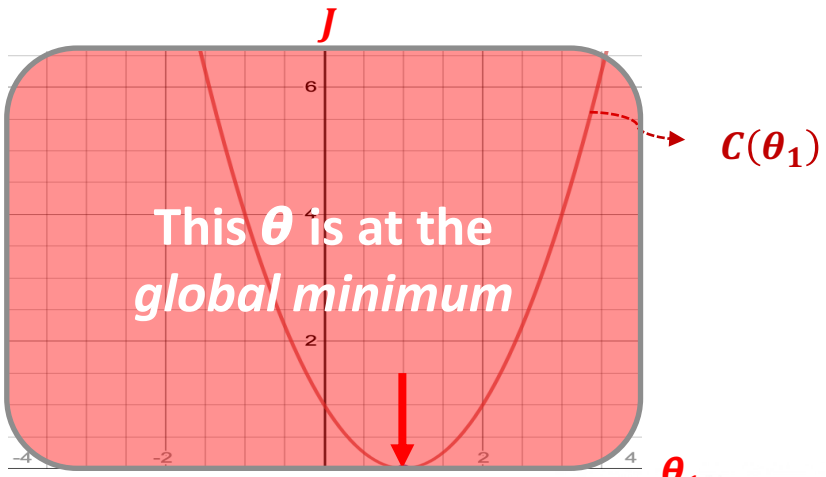
$\theta_1 = 1$

Learning a *Linear Regression Model* via Minimizing a *Cost Function*

- But, how minimizing a cost function (e.g., a mean squared error) can lead to fitting a given training dataset?
  - Let us assume our optimization objective is to $\underset{\theta_0,\theta_1}{\text{minimize}}\, C(\theta_1)$, thus,

$\theta_0 = 0$





Gradient Descent For Linear Regression

- How to $\underset{\theta_0, \theta_1}{\text{minimize}}\, \boldsymbol{C(\theta_0, \theta_1)}$ and locate the (global) minimum, which corresponds to $h_\theta(x)$ that best fits the given data?

- More generally, how to $\underset{\theta_0, \ldots, \theta_{n-1}}{\text{minimize}}\, \boldsymbol{C(\theta_0, \ldots, \theta_{n-1})}$ and locate the (global) minimum, which corresponds to $h_\theta(x)$ that best fits the given data?
  - By using the *gradient descent algorithm*
  - *Note*: The gradient descent algorithm can be utilized also to learn many other mathematical models and not only a linear regression model

Gradient Descent For Linear Regression

- Outline:
  - Have some cost function $\boldsymbol{C(\theta_0, \ldots, \theta_{n-1})}$
  - Start off with some guesses for $\theta_0, \ldots, \theta_{n-1}$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Keep changing $\theta_0, \ldots, \theta_{n-1}$ to reduce $\boldsymbol{C(\theta_0, \ldots, \theta_{n-1})}$ until we hopefully end up at a minimum location
    - When you are at a certain position on the surface of $\boldsymbol{J}$, look around, then take a little step in the direction of *the steepest descent*, then repeat

Gradient Descent For Linear Regression

- Outline:
  - Have some cost function $\boldsymbol{C(\theta_0, \dots, \theta_{n-1})}$
  - Start off with some guesses for $\theta_0, \dots, \theta_{n-1}$
    - It does not really matter what values you start off with, but a common choice is to set them all initially to zero
  - Repeat until convergence{

$$\theta_j = \theta_j - \alpha \frac{\partial \boldsymbol{C(\theta_0, \dots, \theta_{n-1})}}{\partial \theta_j}$$

Partial Derivative

Learing Rate

}

Gradient Descent For Linear Regression

- Outline (considering only two varilables θ_0 and θ_1):
  - Have some cost function $\boldsymbol{J(\theta_0, \theta_1)}$
  - Start off with some guesses for θ_0, θ_1
    - It does not really matter what values you start off with, but a common choice is to set them both initially to zero
  - Repeat until convergence{

$$temp_0 = \theta_0 - \alpha \frac{\partial\, \boldsymbol{J(\theta_0, \theta_1)}}{\partial\, \theta_0}$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x)^{(i)} - y^{(i)}\right)$$

$$temp_1 = \theta_1 - \alpha \frac{\partial\, \boldsymbol{J(\theta_0, \theta_1)}}{\partial\, \theta_1}$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(h_\theta(x)^{(i)} - y^{(i)}\right) .x^{(i)}$$

$$\theta_0 = temp_0$$
$$\theta_1 = temp_1$$

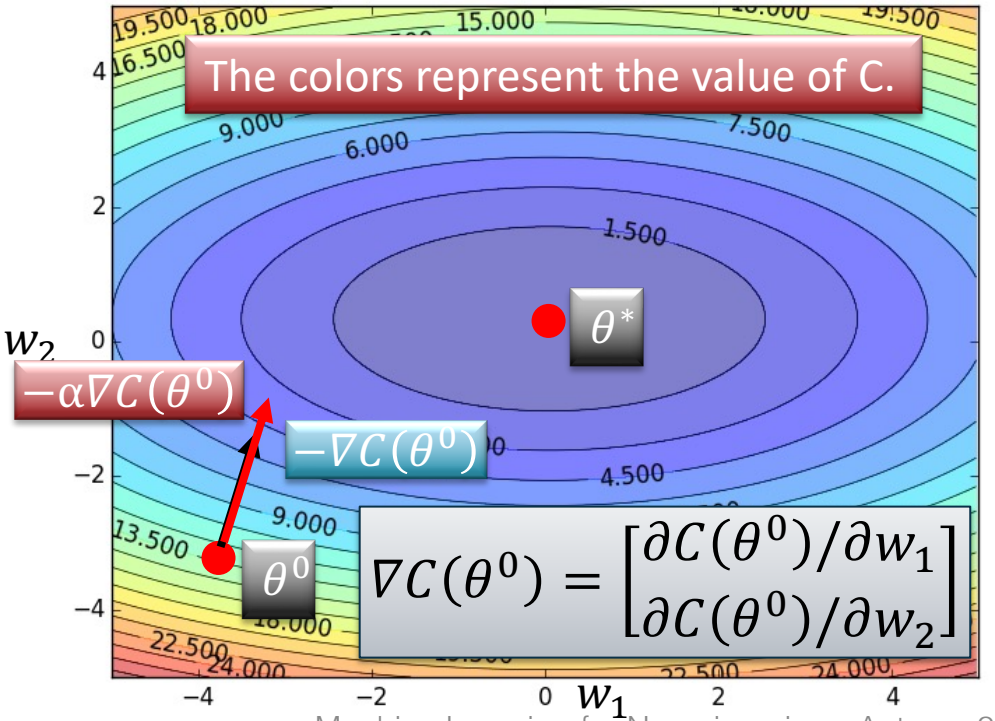
}

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



Randomly pick a starting point θ^0

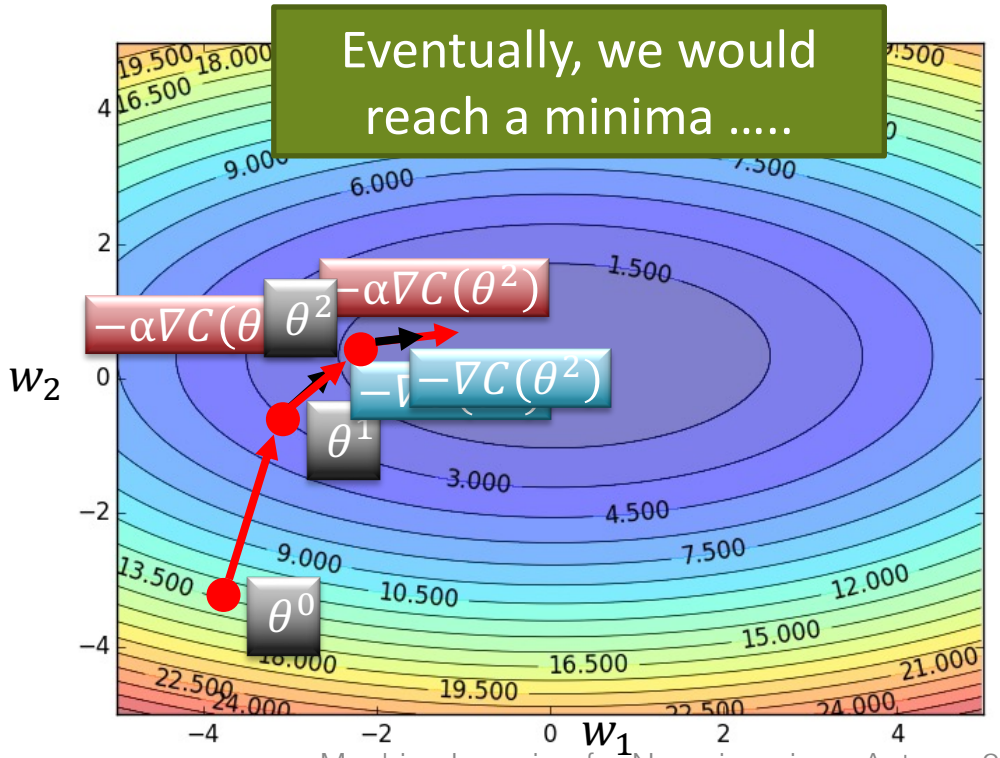
Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta\nabla C(\theta^0)$

Gradient Descent



Randomly pick a starting point θ^0

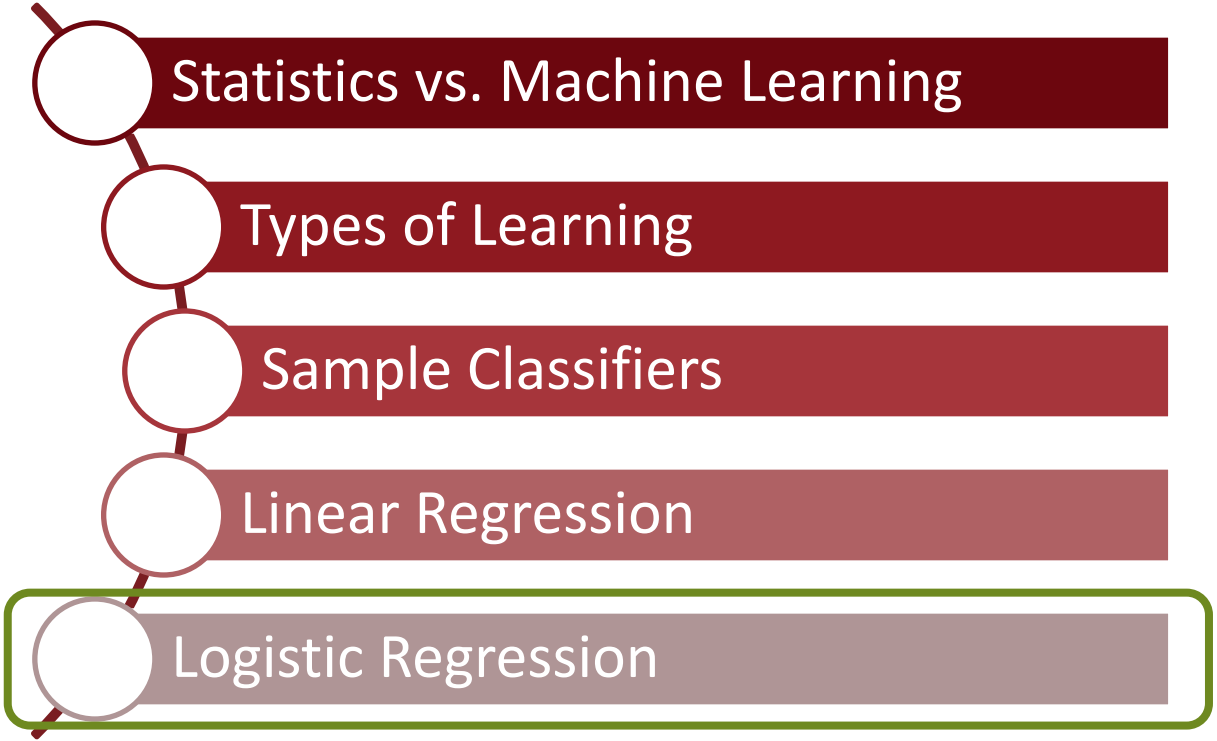
Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

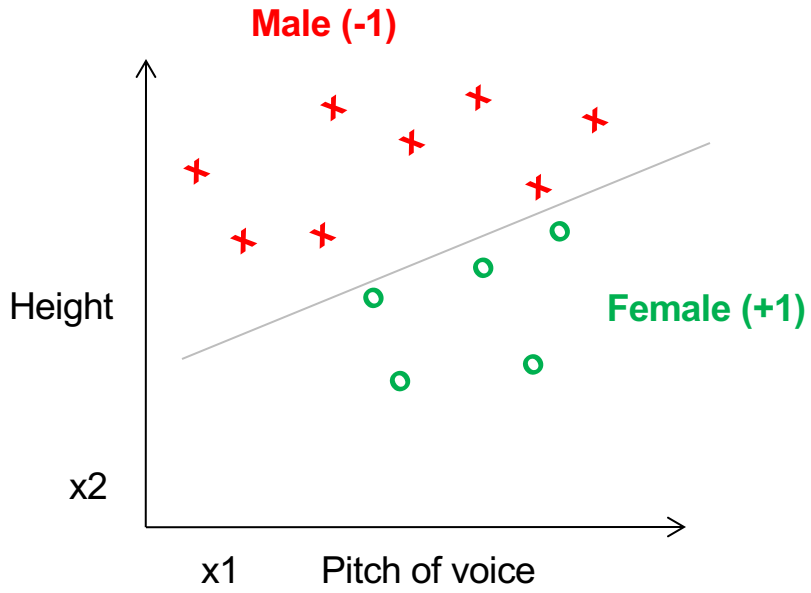
$\Rightarrow -\eta\nabla C(\theta^0)$

Today…


Statistics vs. Machine Learning
Types of Learning
Sample Classifiers
Linear Regression
Logistic Regression

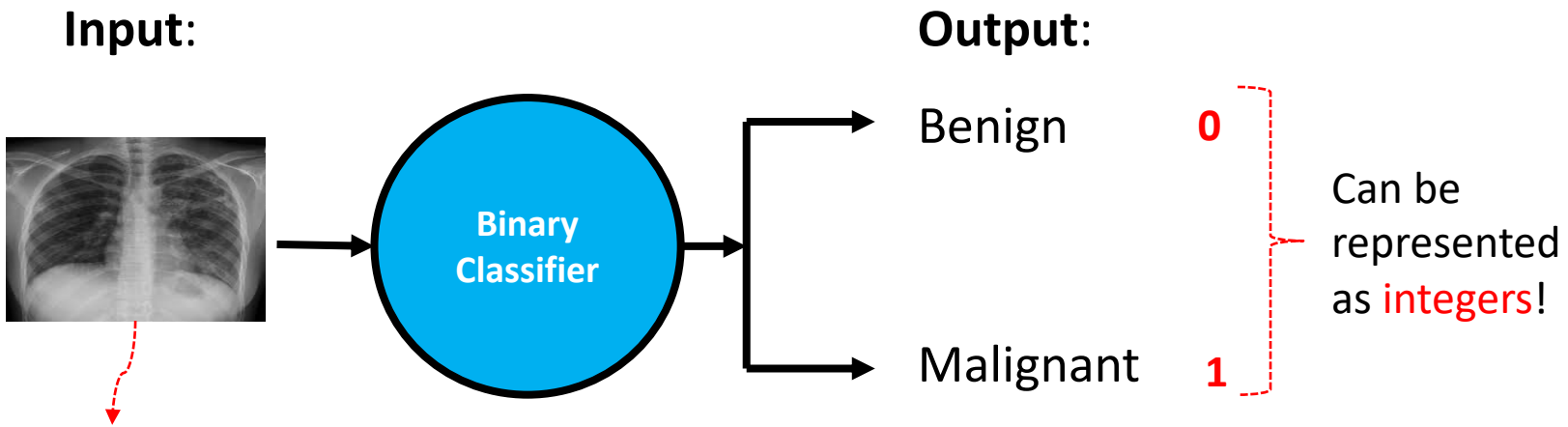
Logistic Regression

Maximize likelihood of label given data, assuming a log-linear model



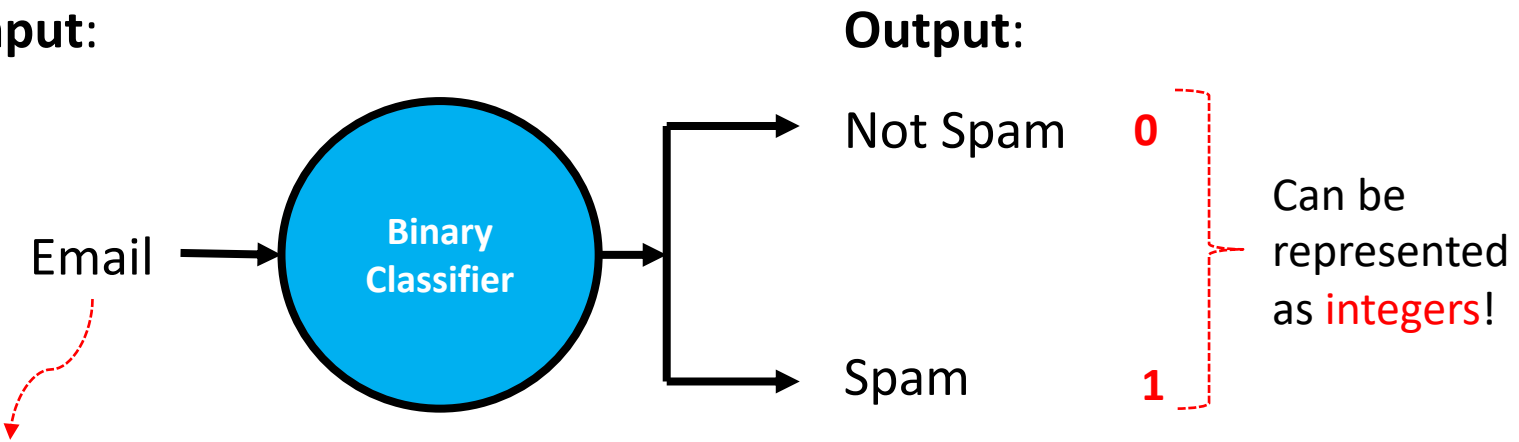
Example 1: Malignant or Benign

- Consider the example of recognizing whether a tumor in an input image is malignant or benign



Example 2: Spam or Not Spam

- As another example, consider the problem of detecting whether an email is a spam or not a spam



Can be represented as a vector $\mathbf{x} = [x_1, x_2, \ldots, x_d]$, with each component x_i corresponding to the presence ($x_i = 1$) or absence ($x_i = 0$) of a particular *word* (or *feature*) in the email

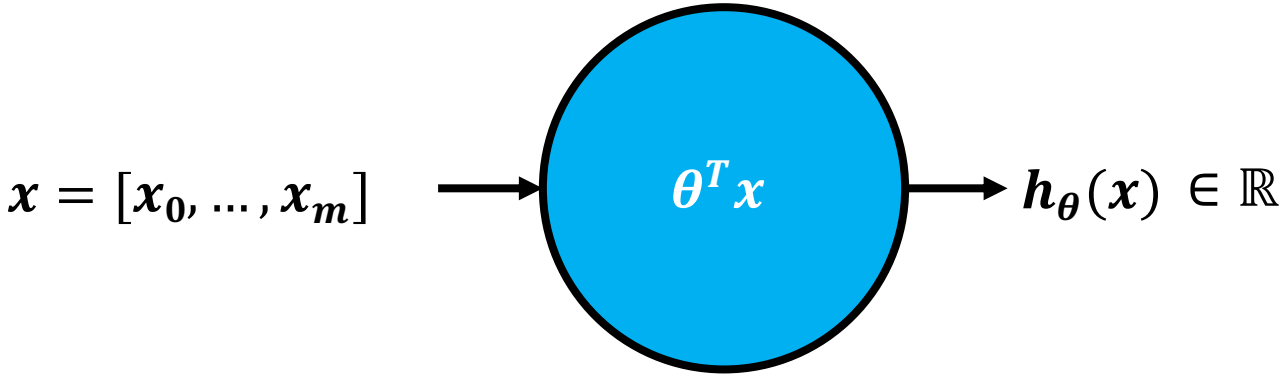
Regression vs. Classification

- What are the possible output values of the *linear regression model* $\boldsymbol{h_\theta(x)} = \boldsymbol{\theta^T x}$?

$\boldsymbol{\theta}$ is the ***parameter vector***, that is, $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_m}]$ (assuming $m + 1$ parameters) and $\boldsymbol{x}$ is the ***feature vector***, that is, $\boldsymbol{x} = [\boldsymbol{x_0}, \boldsymbol{x_1}, \ldots, \boldsymbol{x_m}]$ (assuming $m + 1$ features and $\boldsymbol{x_0} = 1$ to account for the intercept term, namely, $\boldsymbol{\theta_0}$)

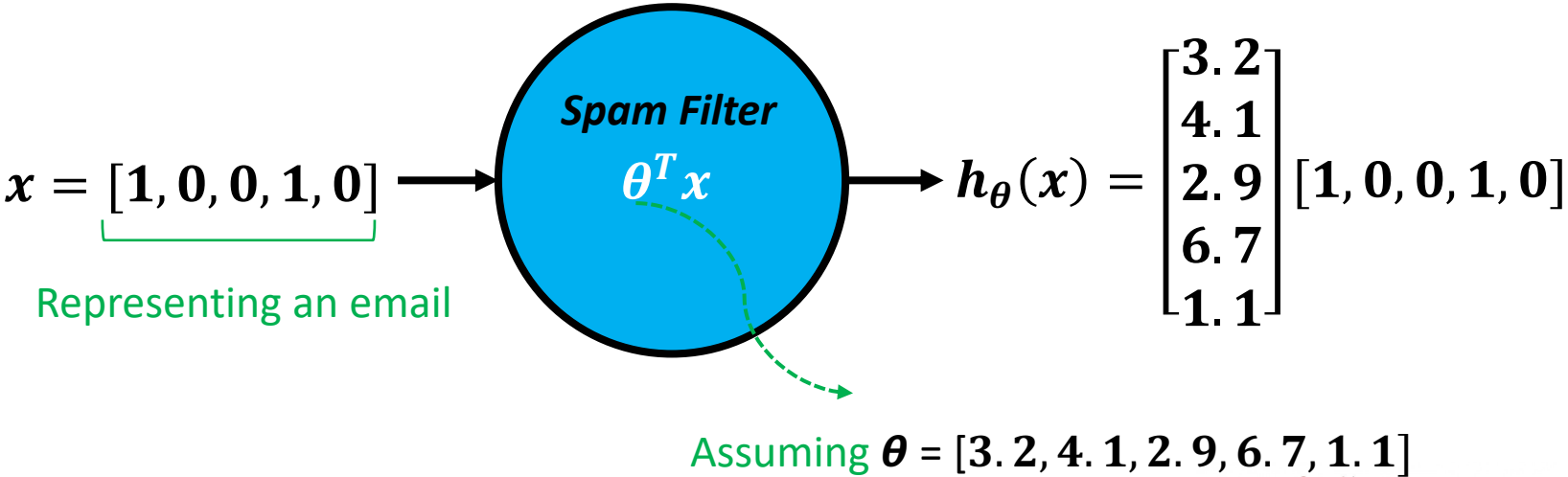
Regression vs. Classification

- What are the possible output values of the *linear regression model* $\boldsymbol{h_\theta(x) = \theta^T x}$?
  - Real-valued outputs



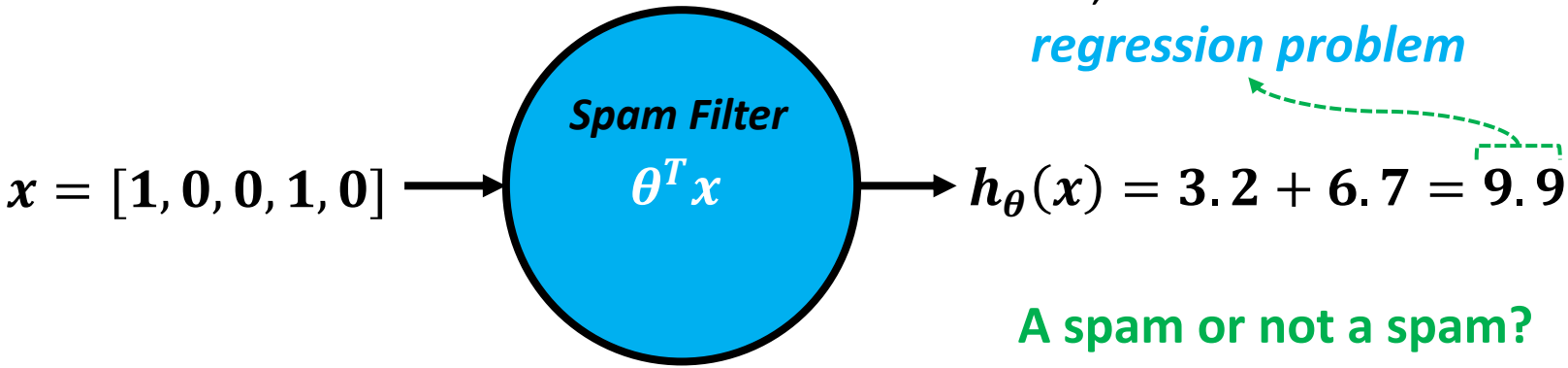
Regression vs. Classification

- What are the possible output values of the *linear regression model* $\boldsymbol{h_\theta(x) = \theta^T x}$?
  - Real-valued outputs



Regression vs. Classification

- What are the possible output values of the *linear regression model* $\boldsymbol{h_\theta(x) = \theta^T x}$?
  - Real-valued outputs

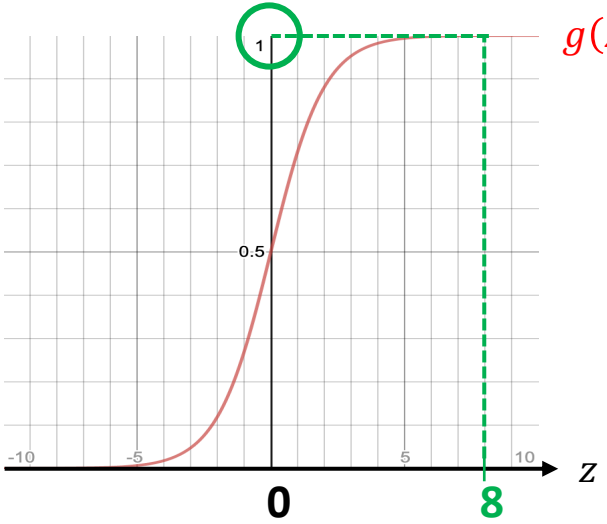


Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an *activation function* (e.g., *sigmoid or logistic function*)

$$g(z) = \frac{1}{1 + e^{-z}}$$

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$



Assume a labeled example ($\boldsymbol{x}$, y):

If y = **1**, we want $g(z) \approx 1$ (i.e., we want a correct prediction)

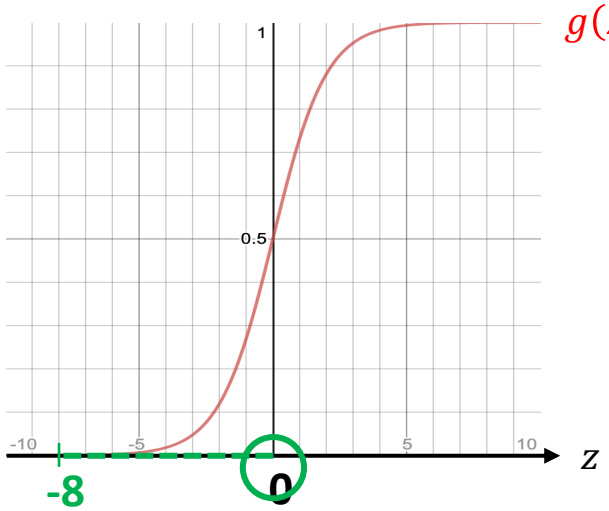
For this to happen, $\boldsymbol{z \gg 0}$

Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an *activation function* (e.g., *sigmoid or logistic function*)

$$g(z) = \frac{1}{1 + e^{-z}}$$

$z \in \mathbb{R}, but$
$g(z) \in [0,1]$



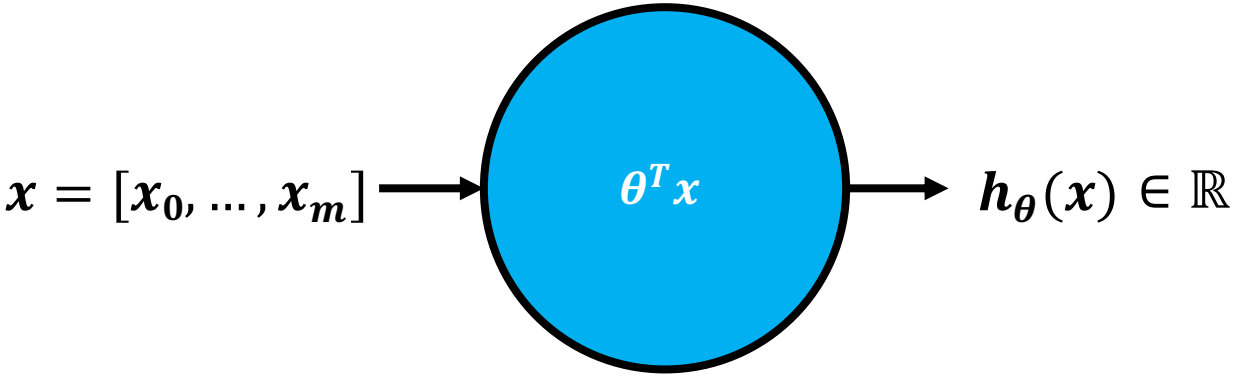
Assume a labeled example ($\boldsymbol{x}$, y):

If y = **0**, we want $g(z) \approx 0$ (i.e., we want a correct prediction)

For this to happen, $\boldsymbol{z \ll 0}$

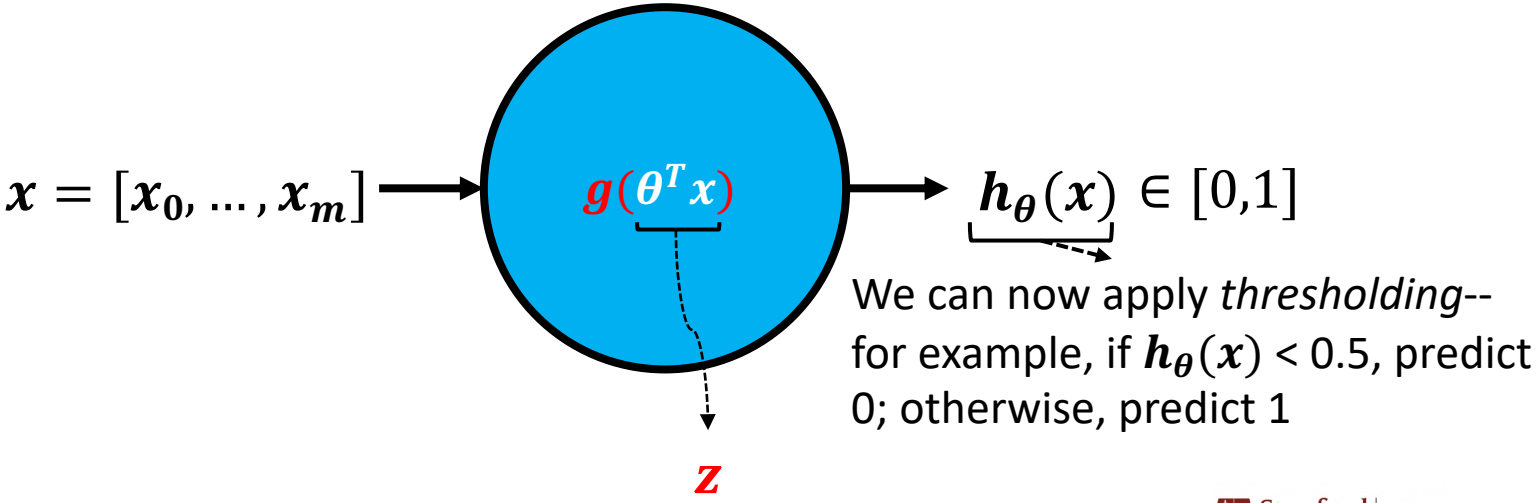
Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an *activation function* (e.g., *sigmoid or logistic function*)



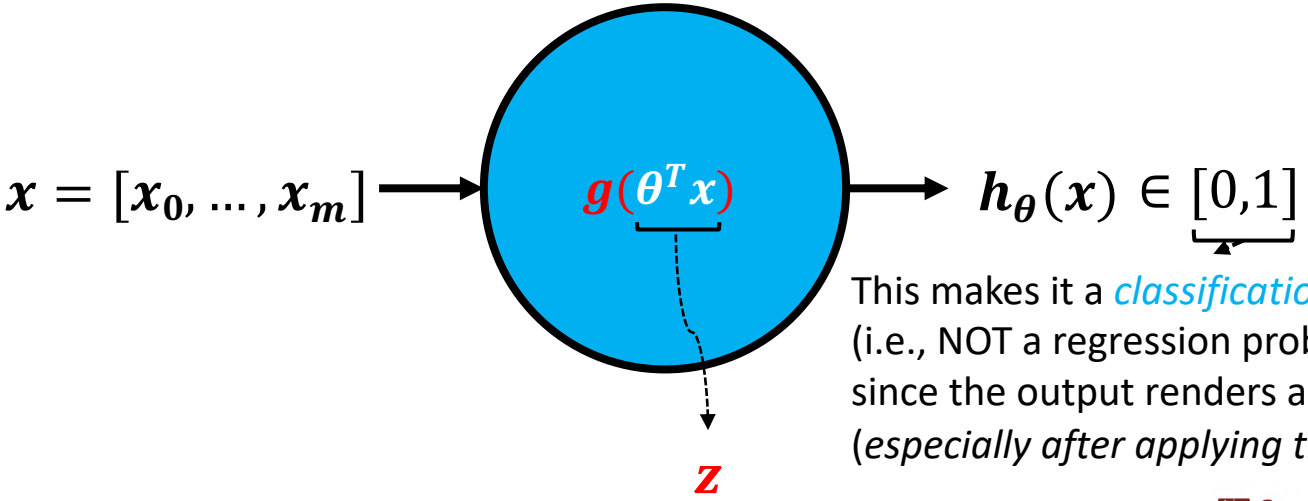
Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an *activation function* (e.g., *sigmoid or logistic function*)



Regression vs. Classification

- How can we make the possible outputs of $\boldsymbol{h_\theta(x) = \theta^T x}$ discrete-valued (as opposed to real-valued)?
  - By using an *activation function* (e.g., *sigmoid or logistic function*)



This makes it a *classification problem* (i.e., NOT a regression problem anymore) since the output renders a discrete one (*especially after applying thresholding*)

So, What is Logistic Regression?

- Logistic regression is a machine learning algorithm that can be used to *classify* input data into discrete output (e.g., *input emails into spam or non-spam and tumour images into benign or malignant*)
  - Note: The word "regression" in the name does not mean that the algorithm is a regression algorithm (rather it is a classification algorithm)

- Major questions about logistic regression:
  - What is the *hypothesis function* (or *model*)?
  - What is the *cost function*?
  - How can we learn the parameters of the model?

The Logistic Regression Model

- What will be the output of the model $\boldsymbol{h_\theta(x) = \theta^T x}$, where $\boldsymbol{\theta} = [\boldsymbol{\theta_0}, \ldots, \boldsymbol{\theta_m}]$ and $\boldsymbol{x} = [\boldsymbol{x_0}, \ldots, \boldsymbol{x_m}]$?
  - Real-valued

- How can we make the output of $\boldsymbol{h_\theta(x)}$ discrete?
  - By using the logistic function as follows:

$$\boldsymbol{h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}}$$

This is the logistic regression model or hypothesis function

  - And then applying thresholding *after* learning the model to predict the output:

$$\begin{cases} \boldsymbol{if\ h_\theta(x) < 0.5\ predict\ 0} \\ \boldsymbol{if\ h_\theta(x) \geq 0.5\ predict\ 1} \end{cases}$$

Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Perhaps, by minimizing *Mean Squared Error (MSE)*. That is:

$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

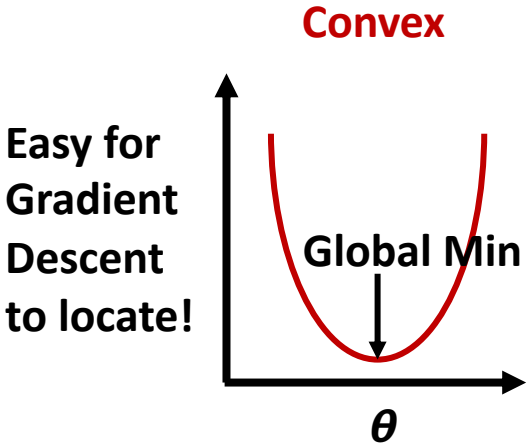
Cost function $\boldsymbol{J(\theta)}$

$$\equiv$$

$$\underset{\theta}{\text{minimize}}\ \boldsymbol{J(\theta)}$$

Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Perhaps, by minimizing *Mean Squared Error (MSE)*. That is:



$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

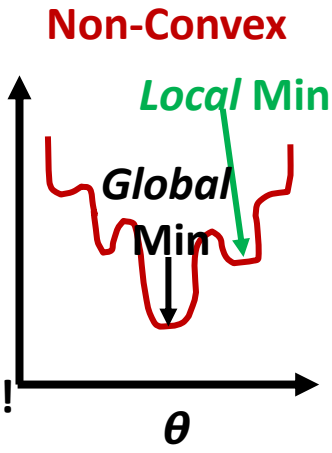
$$\underset{\theta}{\text{minimize}}\, J(\theta)$$

Unfortunately, if we plot this cost function, it will turn out to be "**non-convex**"

Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Perhaps, by minimizing *Mean Squared Error (MSE)*. That is:

Gradient Descent might get stuck at a *local* min and fail to locate the *global* min!



$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}}\, J(\theta)$$

Unfortunately, if we plot this cost function, it will turn out to be "**non-convex**"

Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Perhaps, by minimizing *Mean Squared Error (MSE)*. That is:

We need a cost function that is *convex*, hence, being more suitable for Gradient Descent

$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

$$\underset{\theta}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((g(\theta^T x))^{(i)} - y^{(i)} \right)^2$$

$$\equiv$$

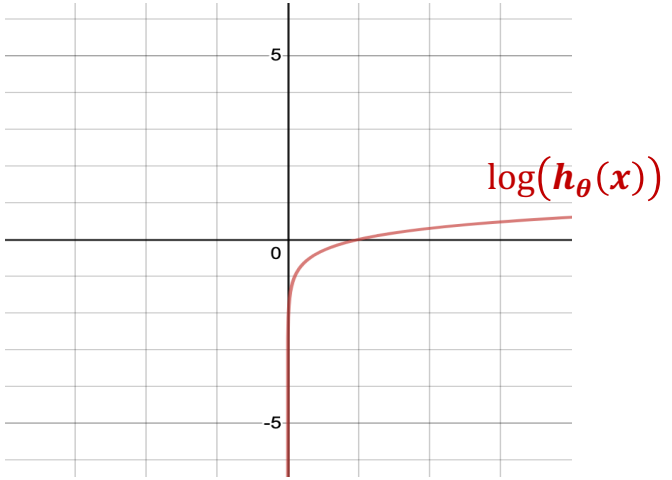
$$\underset{\theta}{\text{minimize}}\, \boldsymbol{J(\theta)}$$

Unfortunately, if we plot this cost function, it will turn out to be "**non-convex**"

Towards Identifying the Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

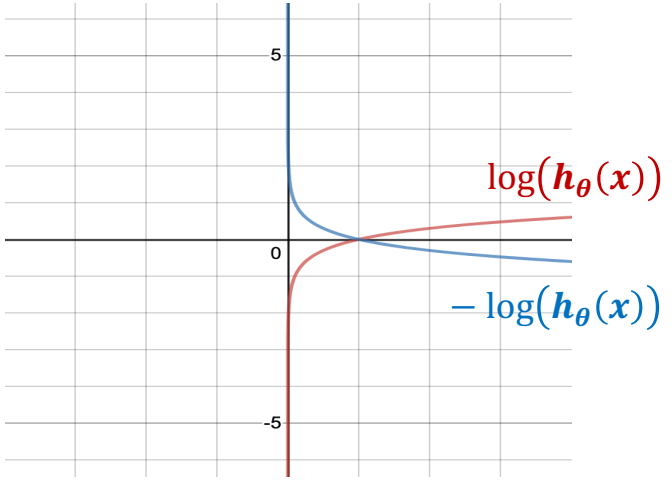
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

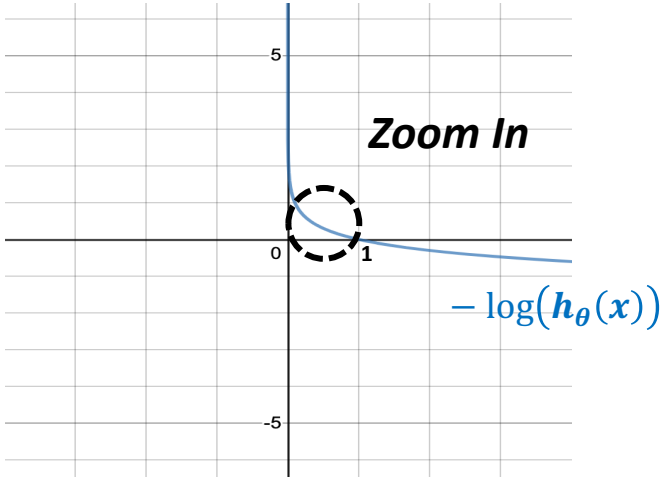
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

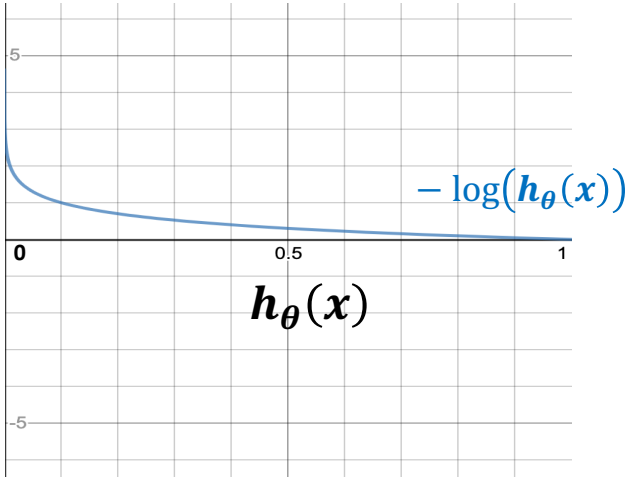
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$

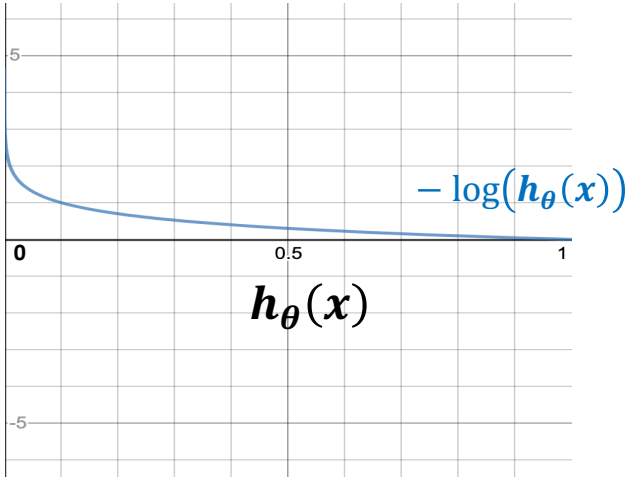


$$\boldsymbol{If \; y = 1} \begin{cases} \text{As } \boldsymbol{h_\theta(x)} \rightarrow 0, \; -\log\big(\boldsymbol{h_\theta(x)}\big) \rightarrow \infty \\ \text{As } \boldsymbol{h_\theta(x)} \rightarrow 1, \; -\log\big(\boldsymbol{h_\theta(x)}\big) \rightarrow 0 \text{ (i.e., cost } \rightarrow 0) \end{cases}$$

The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \dots, \theta_m]}$ and $\boldsymbol{x = [x_0, \dots, x_m]}$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$

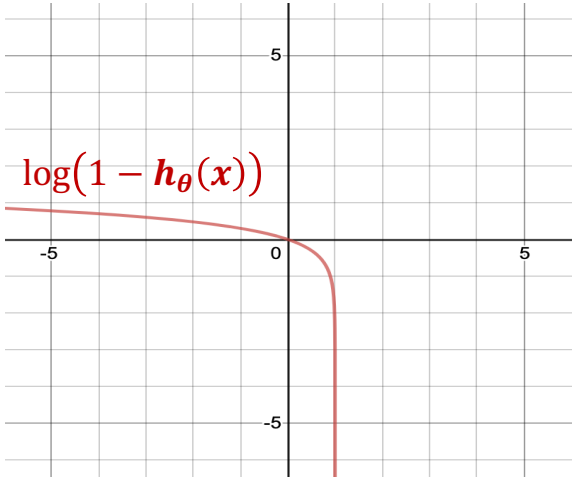


This captures the intuition that if $\boldsymbol{h_\theta(x) = 0}$ (i.e., what we will predict is **0**), but $\boldsymbol{y = 1}$ (hence, we are mispredicting!), we shall penalize the learning algorithm by a very large cost!

The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \dots, \theta_m]}$ and $\boldsymbol{x = [x_0, \dots, x_m]}$?
  - Let us try a different cost function. That is:

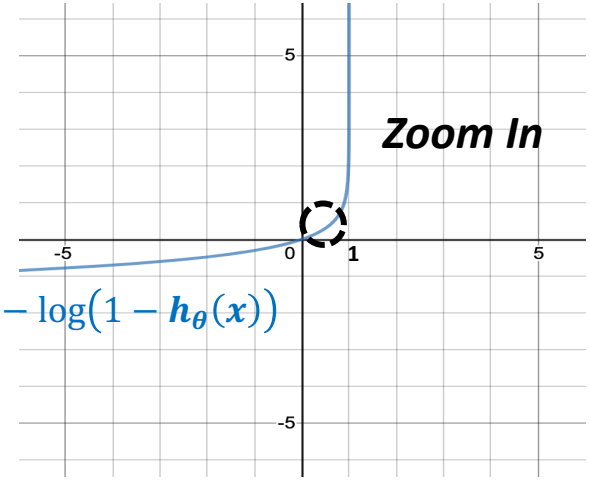
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

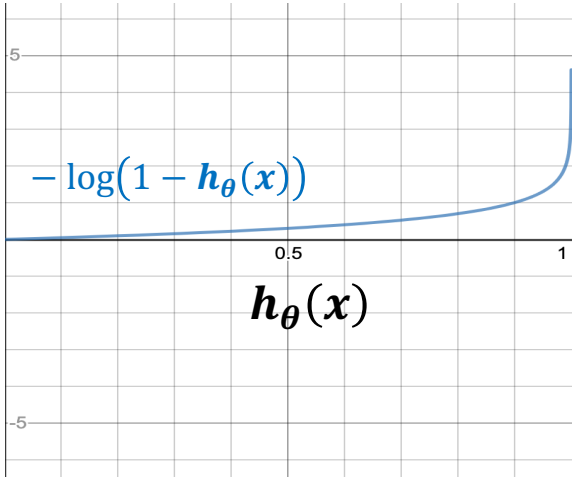
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$

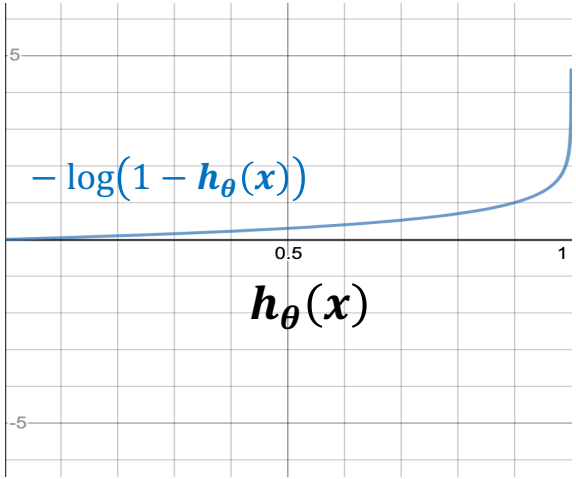


$$\boldsymbol{If \; y = 0} \begin{cases} \text{As } \boldsymbol{h_\theta(x)} \rightarrow 0, \; -\log\big(\boldsymbol{h_\theta(x)}\big) \rightarrow 0 \\ \text{As } \boldsymbol{h_\theta(x)} \rightarrow 1, \; -\log\big(\boldsymbol{h_\theta(x)}\big) \rightarrow \infty \text{ (i.e., cost } \rightarrow \infty) \end{cases}$$

The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \ldots, \theta_m]}$ and $\boldsymbol{x = [x_0, \ldots, x_m]}$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$



This captures the intuition that if $\boldsymbol{h_\theta(x) = 1}$ (i.e., what we will predict is **1**), but $\boldsymbol{y = 0}$ (i.e., we are mispredicting!), we shall penalize the learning algorithm by a very large cost!

The Logistic Regression Cost Function

- How to learn a *logistic regression model* $\boldsymbol{h_\theta(x) = g(\theta^T x)}$, where $\boldsymbol{\theta = [\theta_0, \dots, \theta_m]}$ and $\boldsymbol{x = [x_0, \dots, x_m]}$?
  - Let us try a different cost function. That is:

$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = \begin{cases} -\log\big(\boldsymbol{h_\theta(x)}\big) \; \boldsymbol{if \; y = 1} \\ -\log\big(\boldsymbol{1 - \; h_\theta(x)}\big) \; \boldsymbol{if \; y = 0} \end{cases}$$

Equivalent To

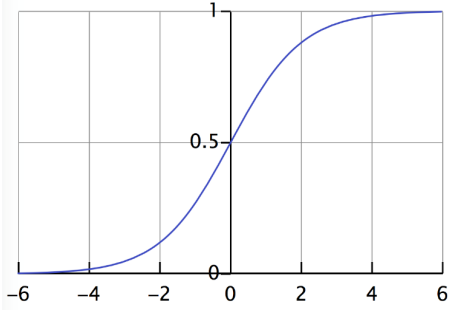
$$\text{Cost}(\boldsymbol{h_\theta(x)}, \text{y}) = -y \log\big(\boldsymbol{h_\theta(x)}\big) - (1 - y) \log\big(\boldsymbol{1 - \; h_\theta(x)}\big)$$

Logistic Regression

- For a binary classification problem, where $Y \in \{0,1\}$:
  - An $Odds\ Ratio\ (OR)$ is defined as $OR = \frac{P(Y=1|\boldsymbol{X})}{P(Y=0|\boldsymbol{X})} \in [0, \infty)$.
    - If $OR > 1$, $P(Y = 1|\boldsymbol{X})$ is more likely to occur.
    - If $OR < 1$, $P(Y = 0|\boldsymbol{X})$ is more likely to occur.
  - A $logit$ is defined as $logit = \ln(OR) \in (-\infty, \infty)$
- *Logistic Regression* is defined as:

$$logit = \ln\left(\frac{P(Y = 1|\boldsymbol{X})}{P(Y = 0|\boldsymbol{X})}\right) = \boldsymbol{\beta'X} \Rightarrow P(Y = 1|\boldsymbol{X}) = \frac{1}{1 + \exp(\boldsymbol{\theta X})} = g(\boldsymbol{\theta X})$$

- *Logistic function* $g(\boldsymbol{\theta X})$:



Fitting Logistic Regression Models

Objective of model fitting:

- Given the data, $(\boldsymbol{X}, \boldsymbol{Y}) = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\}$,
- Find $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\beta}_K)$ to **maximize the conditional log-likelihood** of the data, $\ell(\boldsymbol{\theta}) = \ln(P(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}))$.
- Formally, we want to optimize:

$$argmax_{\boldsymbol{\theta}}\{C(\boldsymbol{\theta}) = \sum_{n=1}^{N} y_n \ln\big(g(\boldsymbol{\theta}\boldsymbol{x}_n)\big) + (1 - y_n) \ln\big(1 - g(\boldsymbol{\theta}\boldsymbol{x}_n)\big)\}$$

Reading assignment 3

- Random Forest (https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/ensembles/RandomForests.pdf)
- Linear Methods for Regression
  - (Elements of Statistical Learning, Chapter 3)
- Linear Methods for Classification
  - (Elements of Statistical Learning, Chapter 4)
- Regularization
  - (Elements of Statistical Learning, Chapter 5)
- Nonlinear SVM -Kernel Methods
  - (Elements of Statistical Learning, Chapter 6)