

# Comparing Classifiers

Chris Piech

CS109, Stanford University

# PSet 7: Machine Learning Out!

PS6

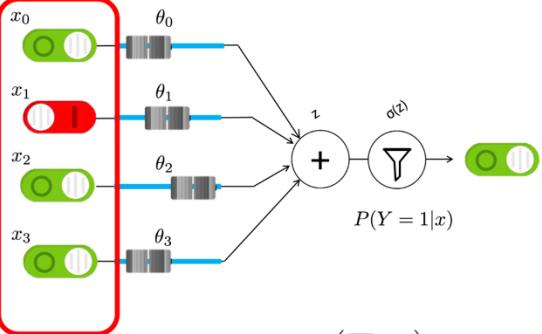
Logistic Regression: Code

Implement Logistic Regression for binary input/output data. Specifically, you should implement the gradient ascent algorithm described in class.

Train your algorithm on the data file simple-train.csv. Use learning rate  $\eta = 0.0001$  and 1,000 training steps. Test your algorithm on the data file simple-test.csv. You should be able to achieve 100% classification accuracy on the testing data.

You will need to implement your code off of the pset app. When you are done, include your logistic regression code here and report the value of the weight associated with  $x_1$  to 5 decimal places.

Warning: Answer checker is not turned on yet.



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

Simple Dataset (simple-train.csv, simple-test.csv)

Dataset: [simple.zip](#)

In this dataset, there are two input features, and the output class value is determined by the value of the first feature (i.e.,  $y = x_1$ ). The training data set and testing data set are identical, each containing four data vectors. Your

[Previous Question](#) [Next Question](#)

Answer Editor Solution

Numeric Answer: Enter your answer Check Answer

Python:

```
1 ****
2 Complete the starter code provided in logistic_regression.py in the pset6
3 directory and paste it here when you are ready. Note that you shouldn't
4 expect to run the code here. Rather, you're just sharing the implementation here
5 so we can review it, but you're expected to run this classifier outside of the
6 problem set application.
7 ****
8
```

Run

# PSet 7: Machine Learning Out!

PS6

Logistic Regression: Ancestry

Train your algorithm on the data file ancestry-train.csv. Use learning rate  $\eta = 0.0001$  and 1,000 training steps. Test your algorithm on the data file ancestry-test.csv.

In your explanation include the value of the weights of your model after training. Report the value of the weight with the largest magnitude to five decimal places. What do you think it means for a weight to have a large magnitude if all of the features are binary?

Warning: Answer checker is not turned on yet.

Genetic ancestry (ancestry-train.csv, ancestry-test.csv)

Dataset: [ancestry.zip](#)

This dataset contains DNA nucleotide readings from 467 individuals. Each input vector represents locations in the human genome and whether the

Answer Editor Solution

Numeric Answer: Enter your answer Check Answer

Explanation:

Block LaTeX ✓ Inline LaTeX Python Image

Previous Question Next Question

# PSet 7: Machine Learning Out!

Screenshot of a web-based machine learning assignment interface titled "Pset 6 - Machine Learning Ge". The URL is [https://cs109psets.netlify.app/fall24/pset6\\_test/heart](https://cs109psets.netlify.app/fall24/pset6_test/heart).

The page is titled "Logistic Regression: Heart". On the left sidebar, labeled "PS6", the fourth question is selected. The main content area contains the following text:

Train your algorithm on the data file `heart-train.csv`. Experiment with using different learning rates  $\eta$ , where you are still testing your algorithm on the data file `heart-test.csv`. Each time use 1,000 training steps. As a starting point for experimenting, try  $\eta = 0.1$  and  $\eta = 0.01$  (i.e., values for  $\eta$  that are larger and smaller by a factor of 10), and then continue experimenting from there.

Report the highest classification accuracy you could obtain on the testing data to three decimal places. In your explanation, include the value of the learning rate  $\eta$  you used to obtain your best accuracy and explain why the learning rate has such a big effect on classification accuracy.

*Warning:* Answer checker is not turned on yet.

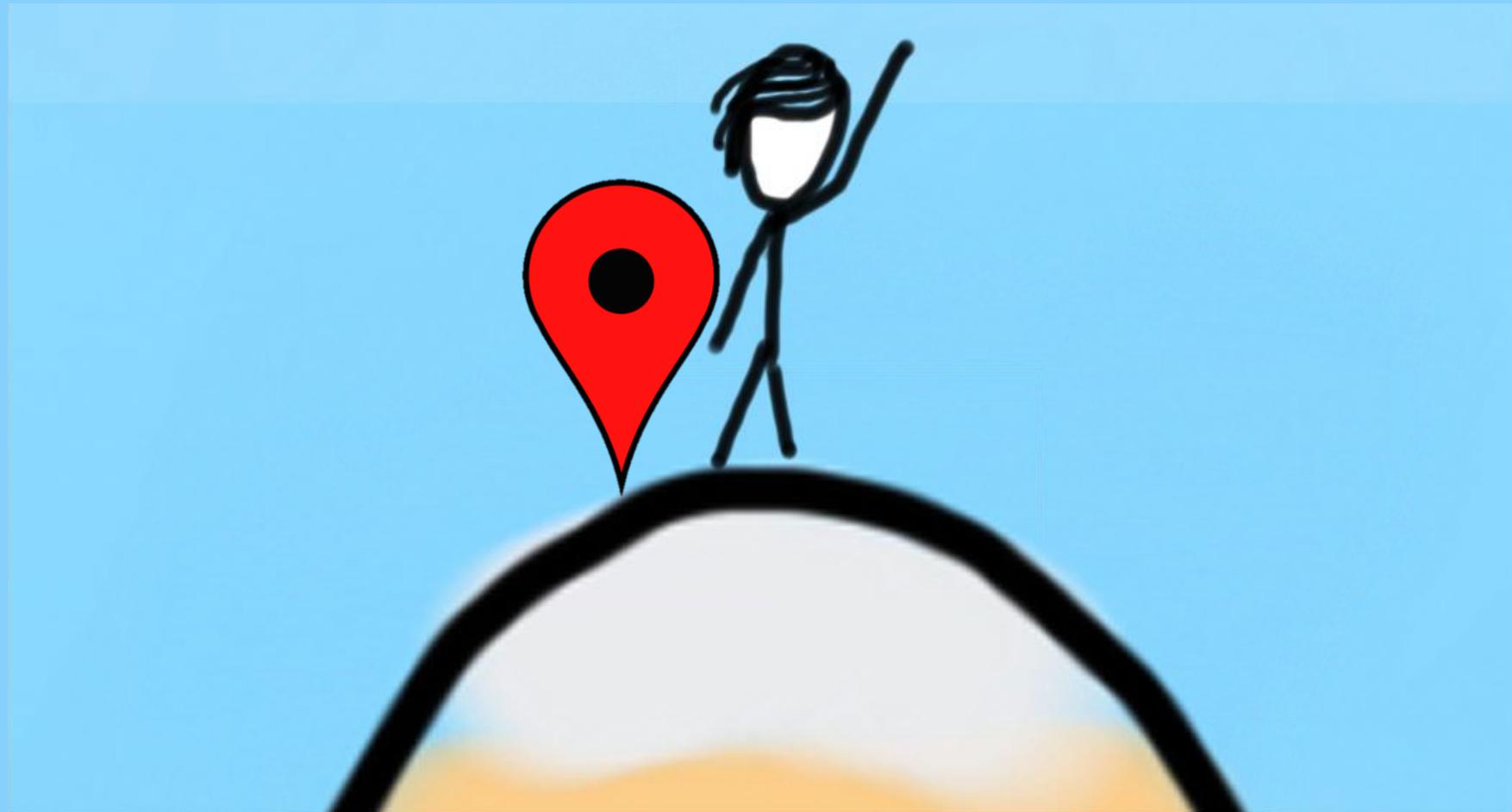
Below the text is a 3D-style illustration of a human torso showing the heart and major blood vessels in red and yellow against a blue background.

At the bottom of the page are links for "Previous Question" and "Next Question". The URL at the bottom is [https://cs109psets.netlify.app/fall24/pset6\\_test/heart](https://cs109psets.netlify.app/fall24/pset6_test/heart).

The right side of the interface features a numeric answer entry field with "Answer Editor" and "Solution" tabs, a "Check Answer" button, and an "Explanation" section with options for "Block LaTeX", "Inline LaTeX", "Python", and "Image". A "Relaunch to update" button is also present.

# Learning Goals

1. Know the derivations behind logistic regression
2. Learn about other models for classification
3. Learn proper ML etiquette for comparing datasets



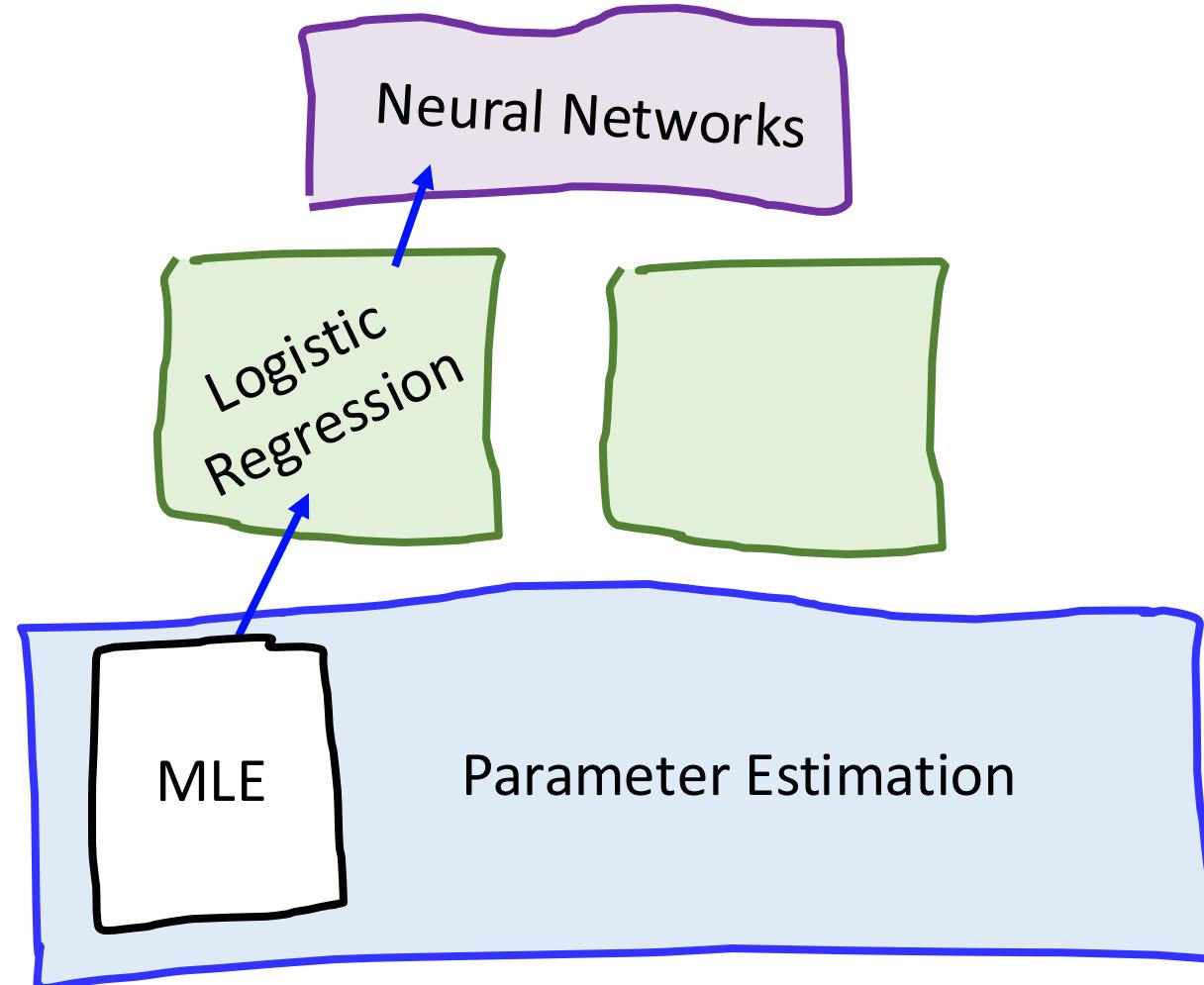
# Review

# Machine Learning in CS109

Great Idea

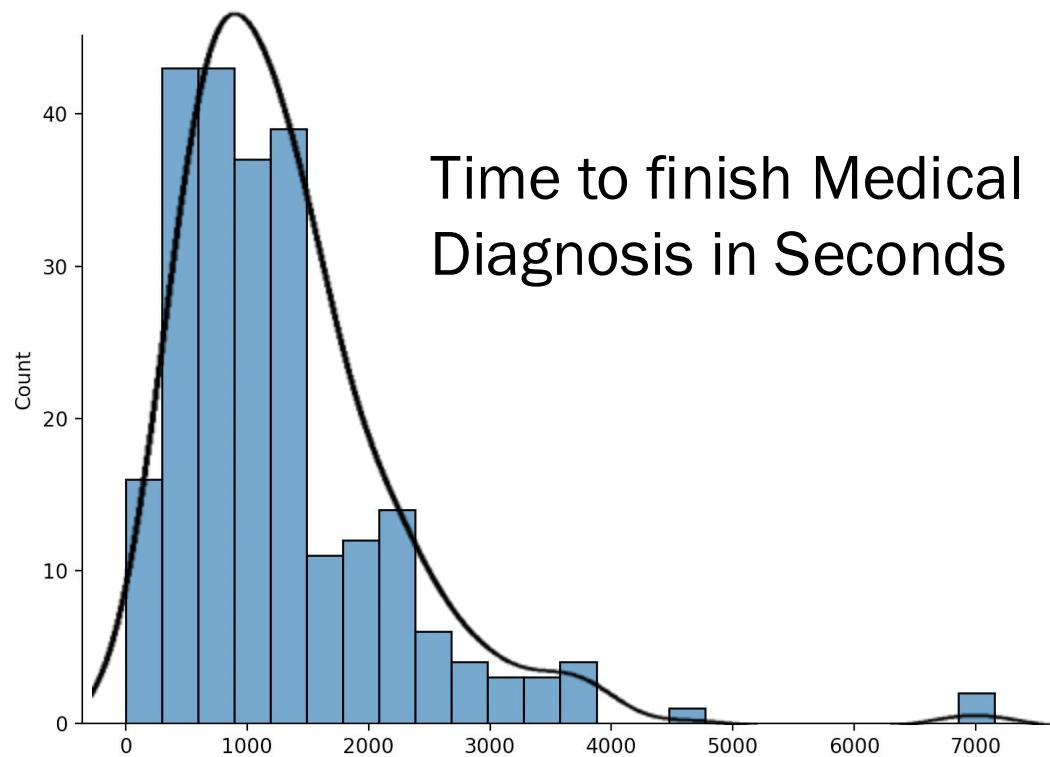
Core  
Algorithms

Theory



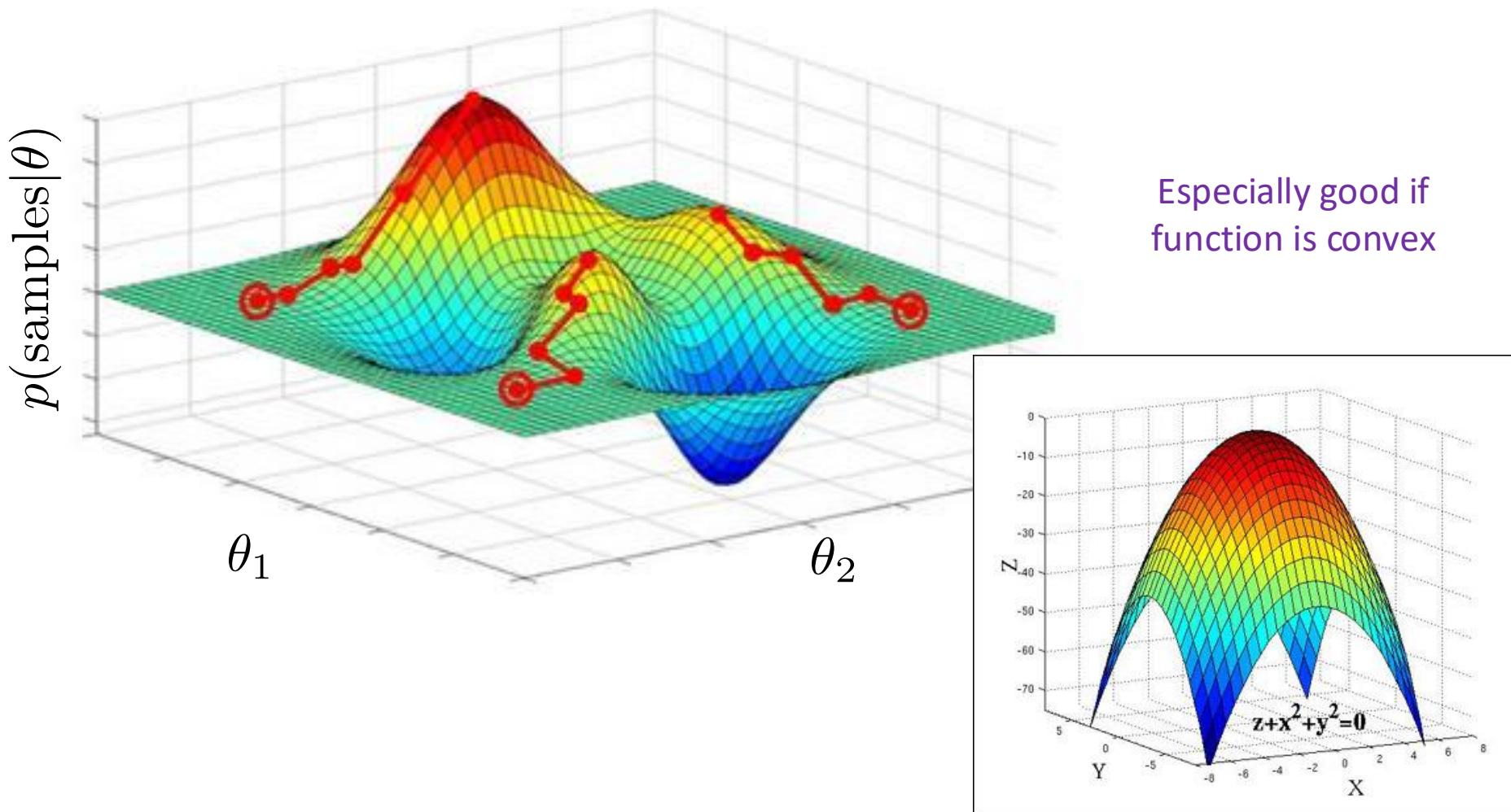
# MLE of Erlang

```
[3.002, 0.983, 2.186, 1.624, 3.997, 1.777,  
2.809, 0.42, 0.515, 1.582, 0.948, 0.458, 1.  
066, 0.8, 2.398, 0.794, 2.561, 2.61, 0.  
595, 3.897, 1.852, 1.182, 3.043, 0.905, 1.  
45, 0.405, 0.445, 2.103, 1.425, 3.12, 0.  
973, 1.056, 3.715, 2.952, 1.817, 2.686, 4.  
173, 0.358, 2.185, 2.581, 7.134, 0.206, 2.  
049, 0.896, 2.095, 4.39, 2.199, 3.434, 5.  
696, 0.819, 0.416, 1.571, 1.337, 2.79, 2.  
701, 3.061, 4.677, 0.671, 1.594, 3.586, 2.  
708, 1.417, 1.799, 1.137, 1.771, 2.12, 0.  
93, 6.835, 3.213, 2.541, 2.505, 1.257, 1.  
99, 1.5, 0.014, 3.856, 0.979, 2.413, 2.  
596, 1.653, 0.881, 4.457, 0.717, 3.305, 2.  
456, 3.462, 1.737, 0.968, 0.528, 0.18, 1.  
626, 2.224, 1.466, 1.6, 1.572, 0.12, 2.86,  
1.062, 2.139, 1.217]
```



$$f(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{\Gamma(k)}$$

# Gradient Ascent



Walk uphill and you will find a local maxima  
(if your step size is small enough)

# Gradient Ascent

**Initialize:**  $\theta_j = \text{random}$  for all  $0 \leq j \leq m$

**Repeat many times:**

*Calculate all gradient[j]'s based on data*

$\theta_j += \eta * \text{gradient}[j]$  for all  $0 \leq j \leq m$

# Healthy Heart Classifier

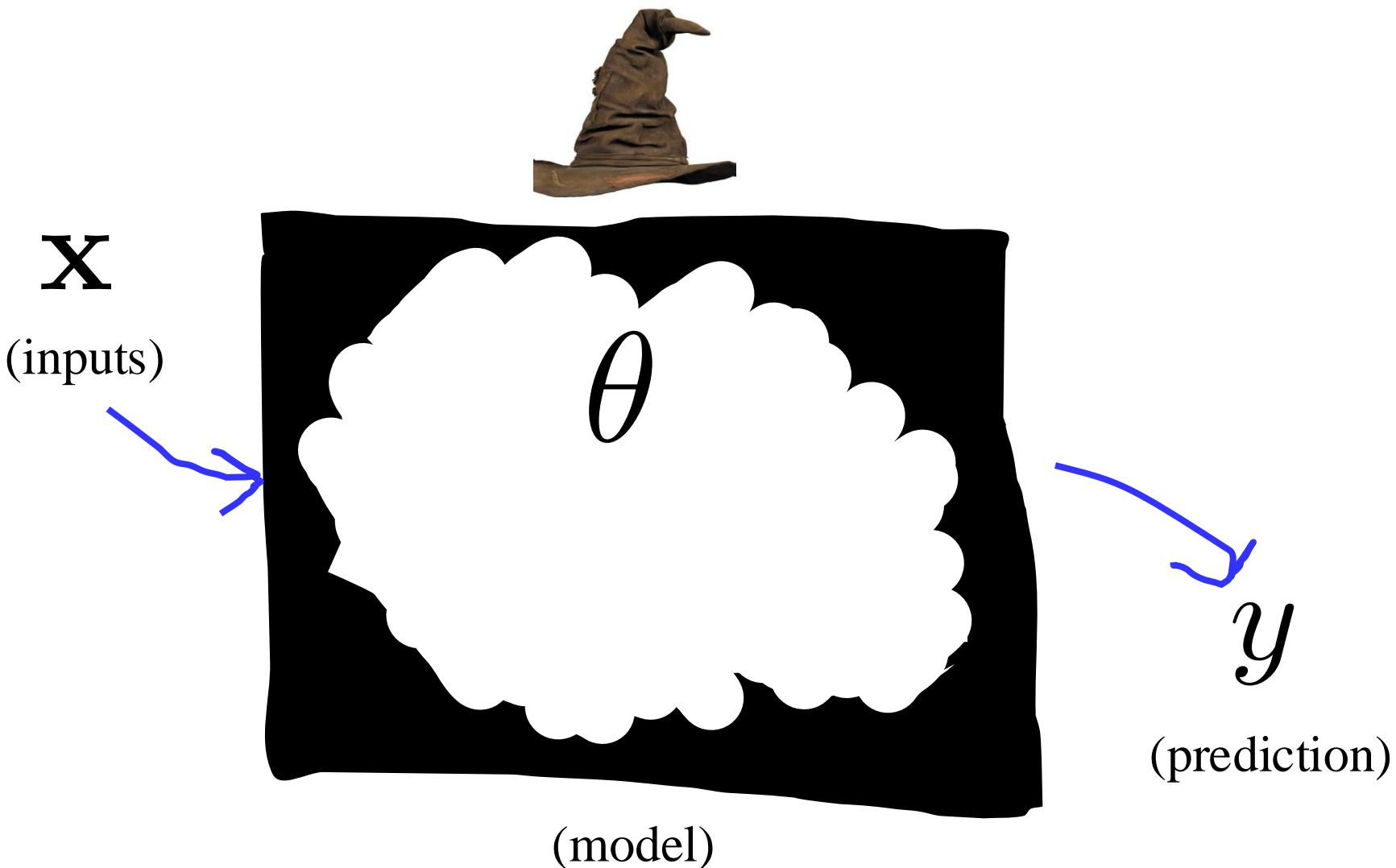
	ROI 1	ROI 2	ROI $m$	Output
Heart 1	0	1	1	0
Heart 2	1	1	1	0
	⋮	⋮	⋮	⋮
Heart $n$	0	0	0	1

# Classification is Building a Harry Potter Hat

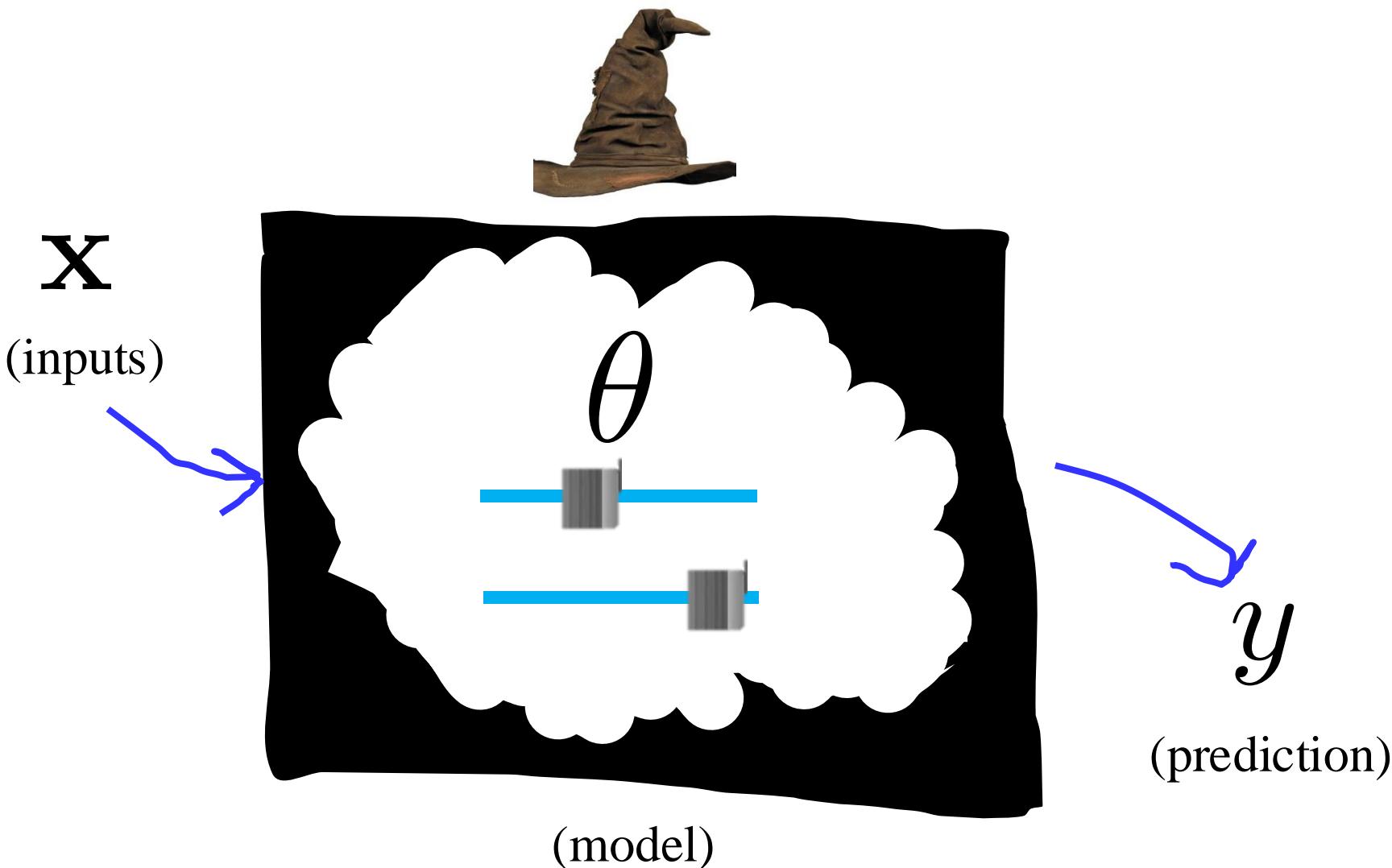


$$\mathbf{x} = [0, 1, \dots, 1]$$

# Machine Learning for Classification



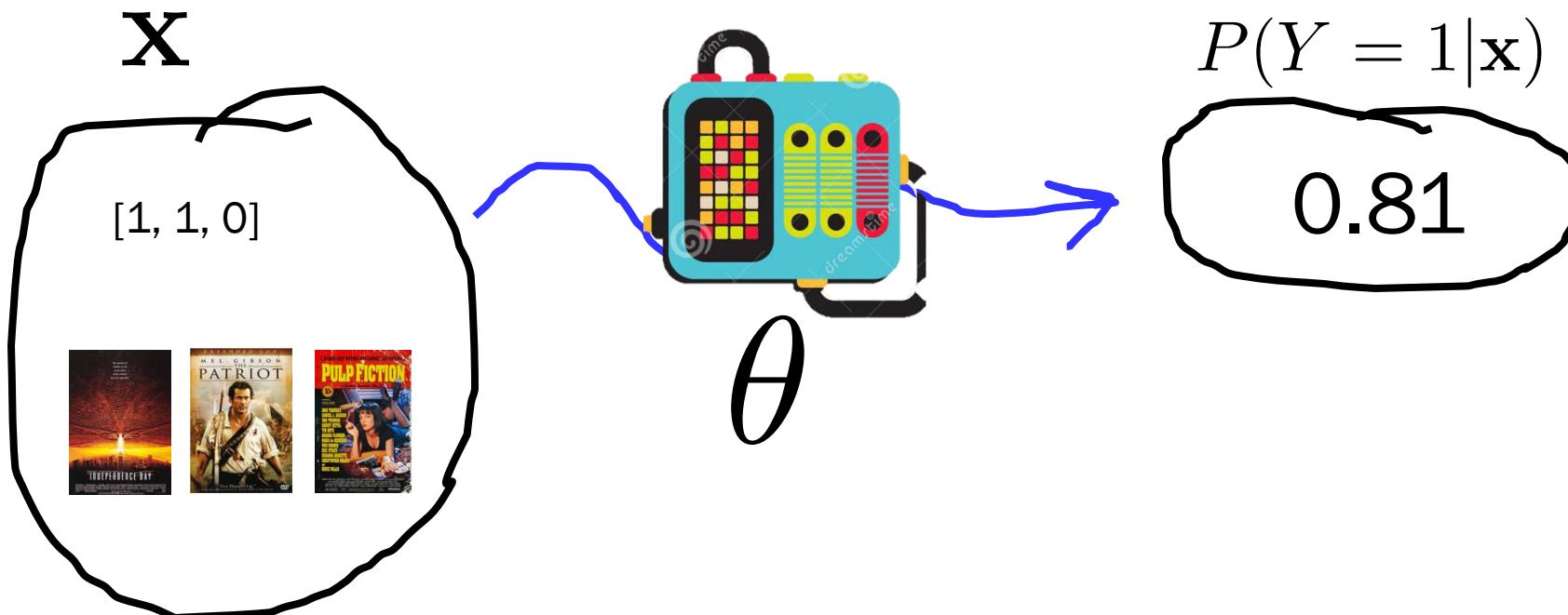
# Machine Learning for Classification



# Logistic Regression Assumption

Could we compute  $P(Y = 1|\mathbf{X} = \mathbf{x})$  via a machine?

Welcome our friend: logistic regression!

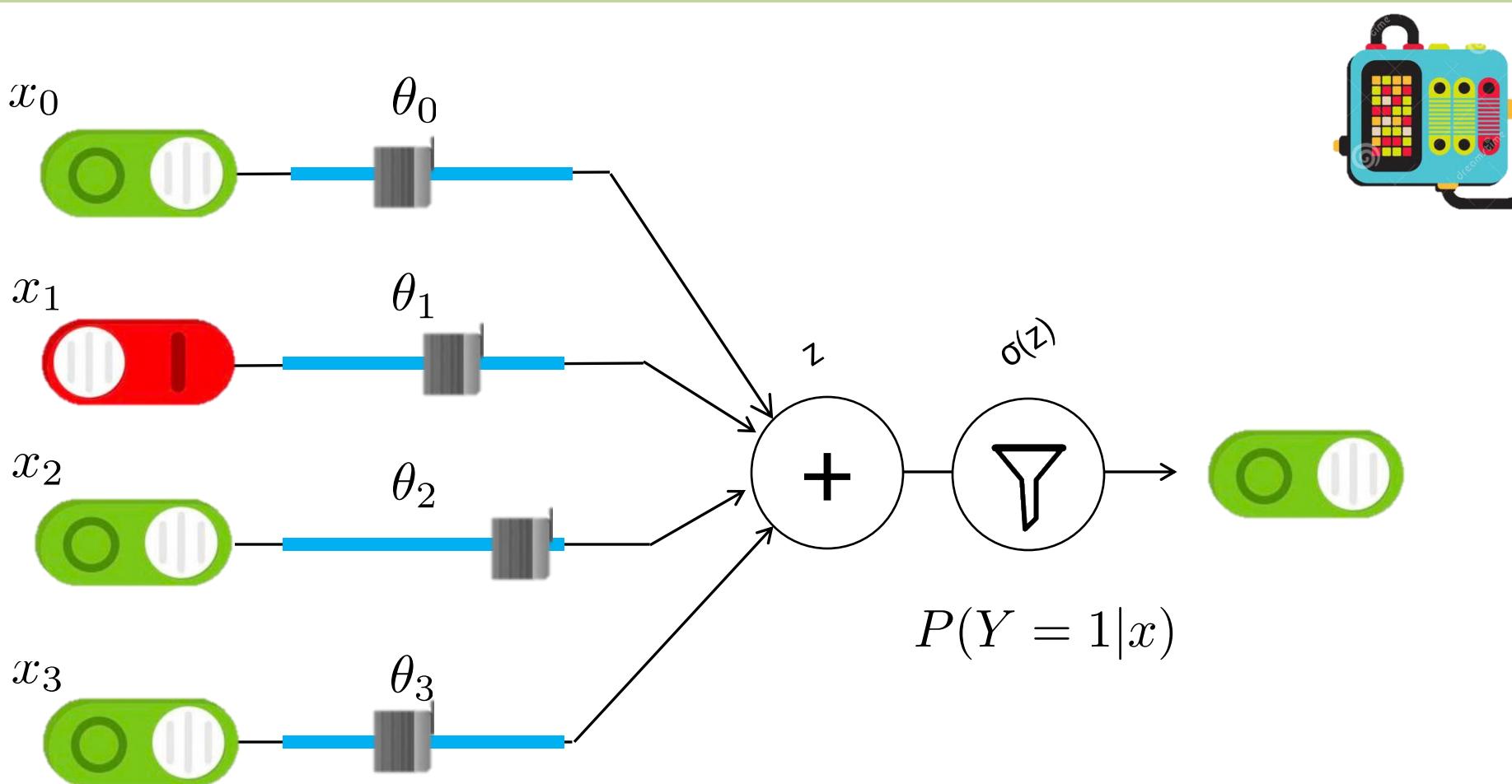


# Logistic Regression Assumption



$$P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

# Logistic Regression



$$P(Y = 1|\mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

# Math for Logistic Regression

1

Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this  
 $\hat{y}$

2

Calculate the log likelihood for all data

$$LL(\theta) = \sum_{i=0}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3

Get derivative of log likelihood with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[ y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

# Logistic Regression Training

Initialize:  $\theta_j = \text{random}$  for all  $0 \leq j \leq m$

Repeat many times:

Calculate all gradient[j]'s based on data

$\theta_j += \eta * \text{gradient}[j]$  for all  $0 \leq j \leq m$

# Logistic Regression Training

Initialize:  $\theta_j = 0$  for all  $0 \leq j \leq m$

Repeat many times:

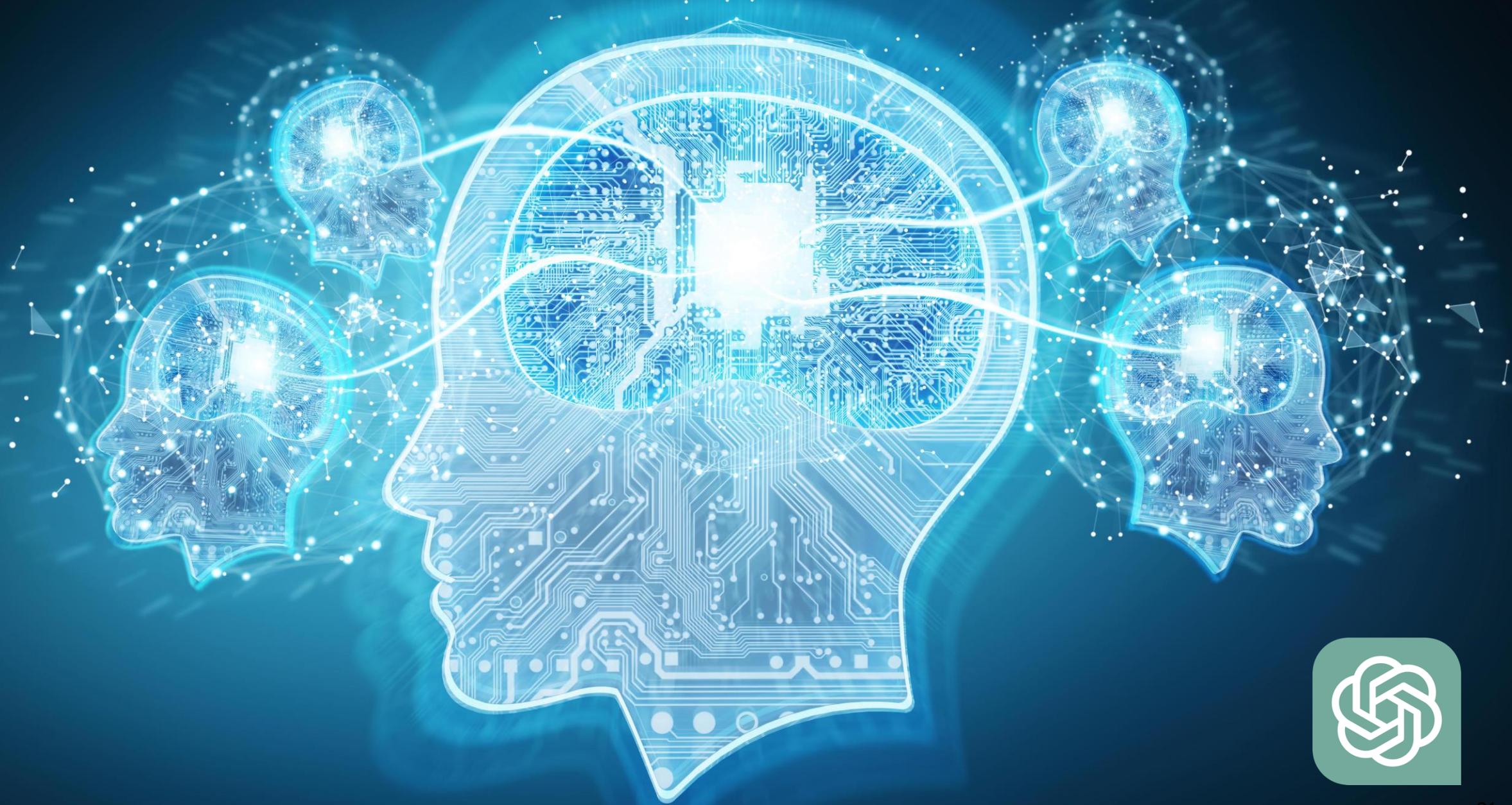
gradient[j] = 0 for all  $0 \leq j \leq m$

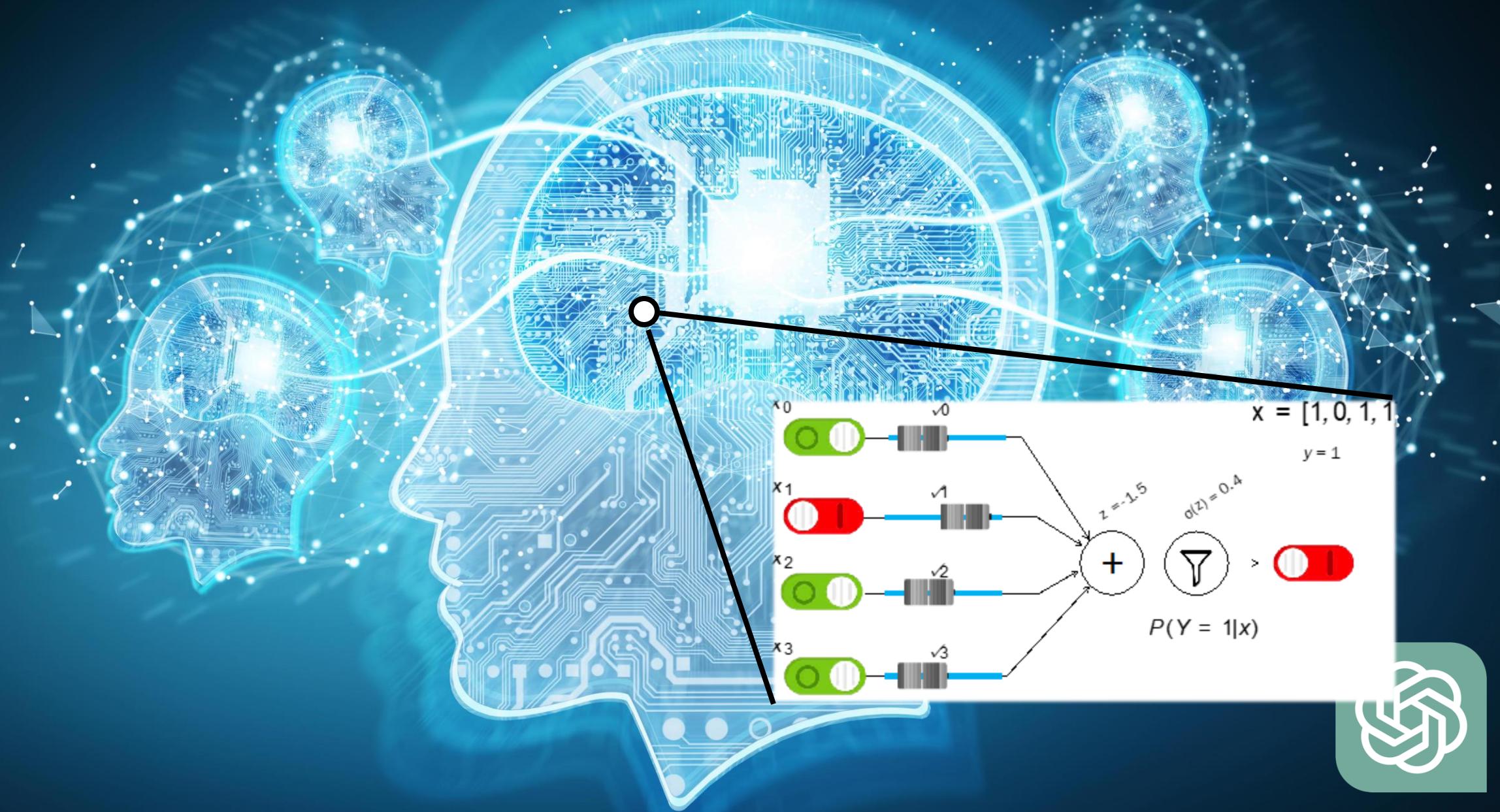
For each training example  $(x, y)$ :

For each parameter  $j$ :

$$\text{gradient}[j] += x_j \left( y - \frac{1}{1 + e^{-\theta^T x}} \right)$$

$\theta_j += \eta * \text{gradient}[j]$  for all  $0 \leq j \leq m$





End Review

# Chapter 2: How Come?

# Logistic Regression

1

Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this  
 $\hat{y}$

2

Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3

Get derivative of log probability with respect to thetas

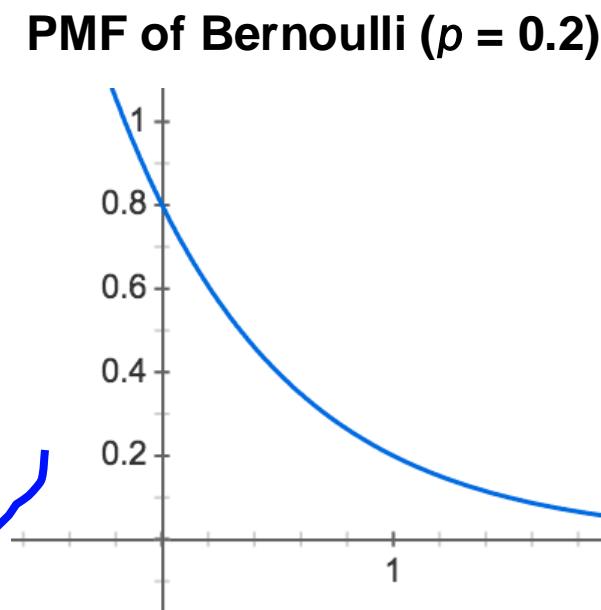
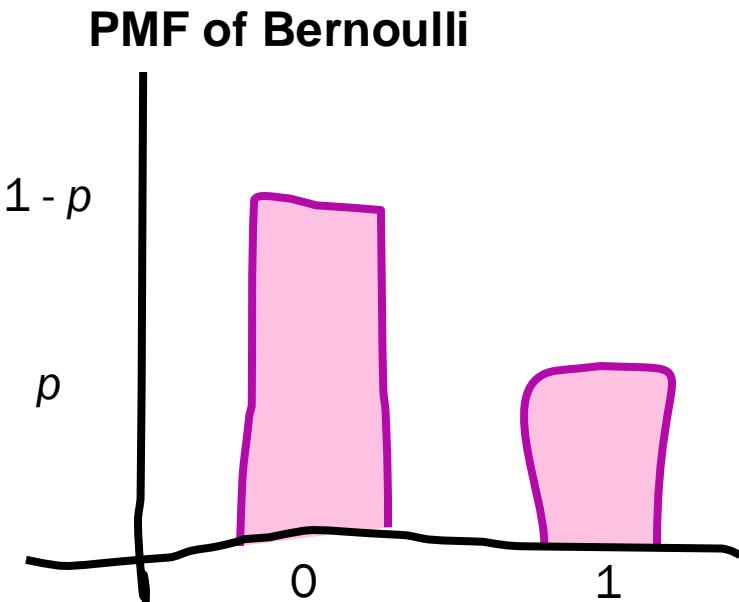
$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[ y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

How did we get that LL function?

# Recall: PMF of Bernoulli

$$Y \sim \text{Bern}(p)$$

Probability mass function:  $P(Y = y)$



$$P(Y = y) = p^y(1 - p)^{1-y}$$

$$P(Y = y) = 0.2^y(0.8)^{1-y}$$

Recall:

$$Y \sim \text{Bern}(p)$$

$$P(Y = y) = p^y(1 - p)^{1-y}$$

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

implies

$$P(Y = y|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})^y \cdot [1 - \sigma(\theta^T \mathbf{x})]^{(1-y)}$$

For IID data

$$L(\theta) = \prod_{i=1}^n P(Y = y^{(i)}|X = \mathbf{x}^{(i)})$$

$$= \prod_{i=1}^n \sigma(\theta^T \mathbf{x}^{(i)})^{y^{(i)}} \cdot [1 - \sigma(\theta^T \mathbf{x}^{(i)})]^{(1-y^{(i)})}$$

Take the log

$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log [1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

Ask Chris:  
Why not

$$P(Y = y^{(i)}, X = x^{(i)})$$

How did we get that gradient?

# Sigmoid has a Beautiful Slope

True fact about sigmoid  
functions

$$\frac{\partial}{\partial z} \sigma(z) = \sigma(z)[1 - \sigma(z)]$$

# Sigmoid has a Beautiful Slope

$$\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) ?$$

$$\frac{\partial}{\partial z} \sigma(z) = \sigma(z)[1 - \sigma(z)]$$

where  $z = \theta^T x$

$$\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) = \frac{\partial}{\partial z} \sigma(z) \cdot \frac{\partial z}{\partial \theta_j}$$

Chain rule!

$$\frac{\partial}{\partial \theta_j} \sigma(\theta^T x) = \sigma(\theta^T x)[1 - \sigma(\theta^T x)]x_j$$

Plug and chug

# Sigmoid has a Beautiful Slope

$$\hat{y} = \sigma(\theta^T x)$$

---

$$\frac{\partial \hat{y}}{\partial \theta_j} = \sigma(\theta^T x)[1 - \sigma(\theta^T x)]x_j$$

$$= \hat{y}(1 - \hat{y})x_j$$

[suspense]

ARE YOU READY???

# I think I'm Ready...

$$\frac{\partial LL(\theta)}{\partial \theta_j}$$

Where

$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$





This is Sparta!!!!



This is Sparta!!!!

↑  
Stanford

# Think About Only One Training Instance

$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$

---

We only need to calculate the gradient for one training example!

$$\frac{\partial}{\partial x} \sum_i f(x, i) = \sum_i \frac{\partial}{\partial x} f(x, i)$$

We will pretend we only have one example

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

We can sum up the gradients of each example to get the correct answer

# First, imagine only one example

$$LL(\theta) = y \log \hat{y} + (1 - y) \log[1 - \hat{y}]$$

Where  $\hat{y} = \sigma(\theta^T \mathbf{x})$

---

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \frac{\partial LL(\theta)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \theta_j}$$

CHAIN RULZ!

$$= \frac{\partial LL(\theta)}{\partial \hat{y}} \hat{y}(1 - \hat{y})x_j$$

Already did that one

$$= \left[ \frac{y}{\hat{y}} - \frac{1 - y}{1 - \hat{y}} \right] \hat{y}(1 - \hat{y})x_j$$

Derive this one

$$= (y - \hat{y})x_j$$

Simplify

# Now, all the data

$$LL(\theta) = \sum_{i=0}^n y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}]$$
$$\hat{y}^{(i)} = \sigma(\theta^T \mathbf{x}^{(i)})$$

---

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=0}^n \frac{\partial}{\partial \theta_j} \left[ y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log[1 - \hat{y}^{(i)}] \right]$$

Derivative of sum...

$$= \sum_{i=0}^n [y^{(i)} - \hat{y}^{(i)}] x_j^{(i)}$$

See last slide

$$= \sum_{i=0}^n [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)}$$

Some people don't like hats...

# Now, all the data

$$\frac{\partial LL(\theta)}{\partial \theta_j}$$

$$= \sum_{i=1}^n [y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)})] x_j^{(i)}$$

# Logistic Regression

1

Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

2

Calculate the log probability for all data

$$LL(\theta) = \sum_{i=1}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3

Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[ y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

# The Hard Way

$$LL(\theta) = y \log \sigma(\theta^T \mathbf{x}) + (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})]$$

---

$$\begin{aligned}\frac{\partial LL(\theta)}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j} y \log \sigma(\theta^T \mathbf{x}) + \frac{\partial}{\partial \theta_j} (1 - y) \log[1 - \sigma(\theta^T \mathbf{x})] \\ &= \left[ \frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1 - y}{1 - \sigma(\theta^T \mathbf{x})} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T \mathbf{x}) \\ &= \left[ \frac{y}{\sigma(\theta^T \mathbf{x})} - \frac{1 - y}{1 - \sigma(\theta^T \mathbf{x})} \right] \frac{\partial}{\partial \theta_j} \sigma(\theta^T \mathbf{x}) \\ &= \left[ \frac{y - \sigma(\theta^T \mathbf{x})}{\sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})]} \right] \sigma(\theta^T \mathbf{x})[1 - \sigma(\theta^T \mathbf{x})] x_j \\ &= [y - \sigma(\theta^T \mathbf{x})] x_j\end{aligned}$$

Phew!

# Logistic Regression

1

Make logistic regression assumption

$$P(Y = 1|X = \mathbf{x}) = \sigma(\theta^T \mathbf{x})$$

$$P(Y = 0|X = \mathbf{x}) = 1 - \sigma(\theta^T \mathbf{x})$$

Often call this  
 $\hat{y}$

2

Calculate the log probability for all data

$$LL(\theta) = \sum_{i=0}^n y^{(i)} \log \sigma(\theta^T \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log[1 - \sigma(\theta^T \mathbf{x}^{(i)})]$$

3

Get derivative of log probability with respect to thetas

$$\frac{\partial LL(\theta)}{\partial \theta_j} = \sum_{i=1}^n \left[ y^{(i)} - \sigma(\theta^T \mathbf{x}^{(i)}) \right] x_j^{(i)}$$

[pedagogical pause]

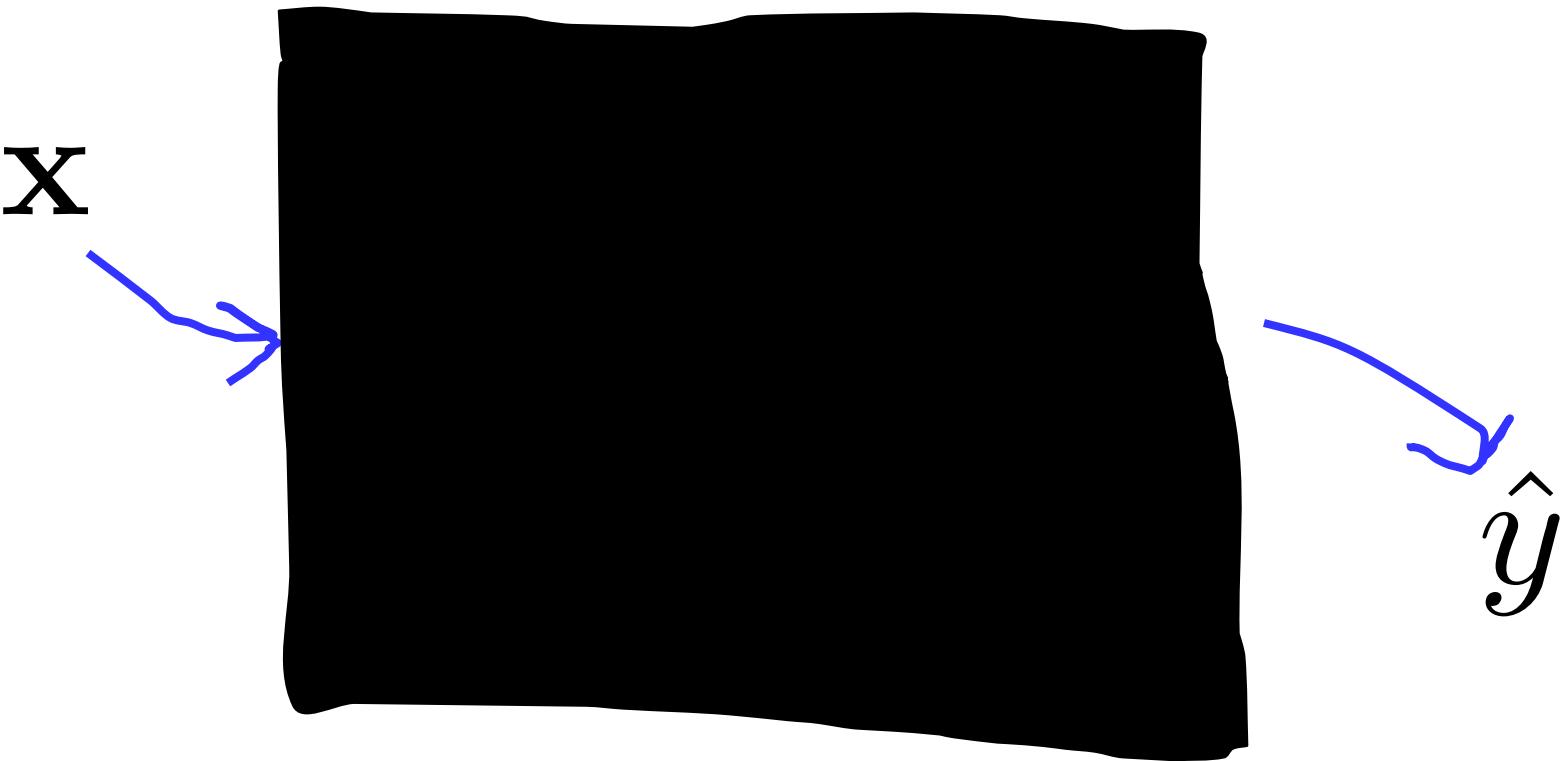
# Three Guiding Questions

---

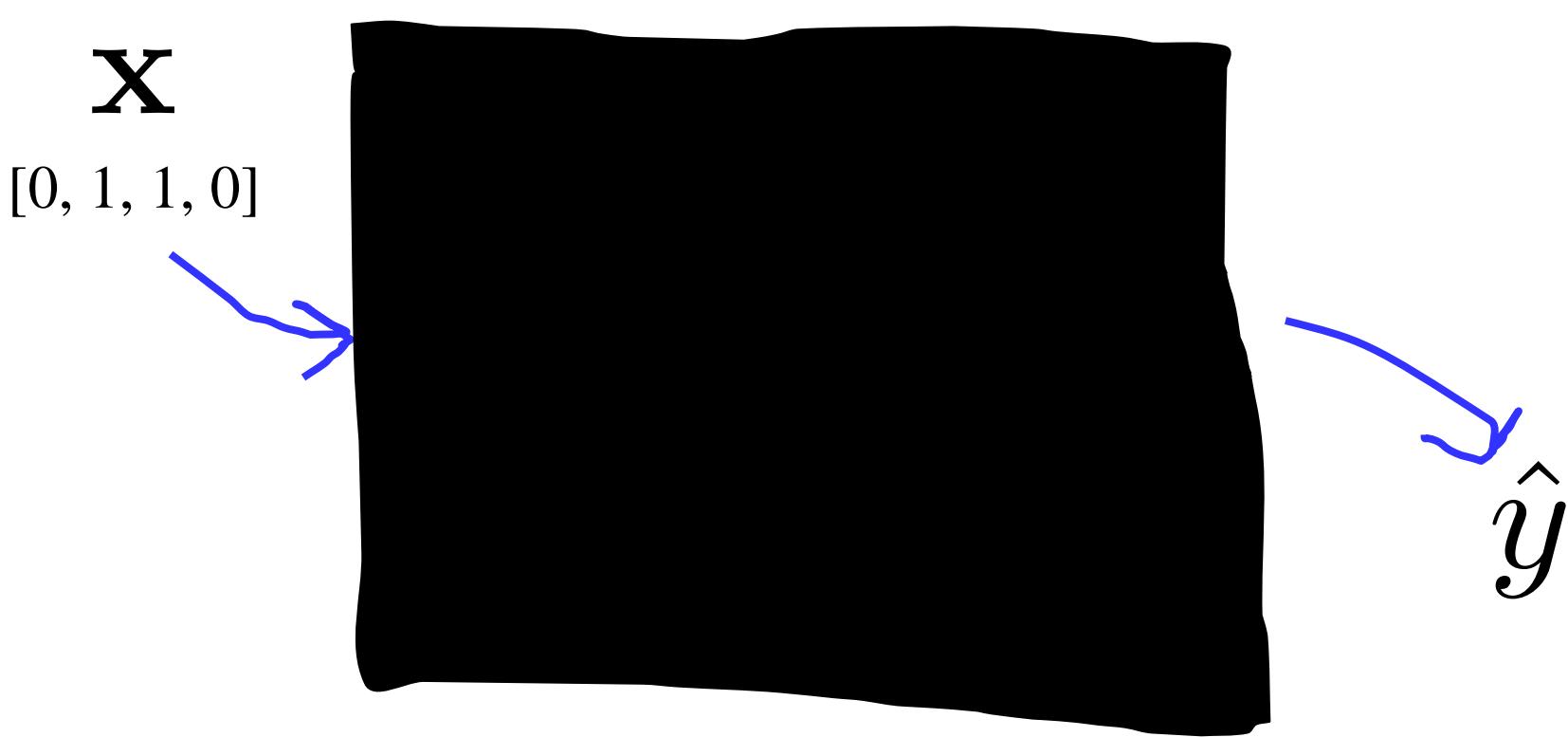
1. What are other models for classification?
2. For a dataset, how do you tell which one is best?
3. What are other validation metrics I might care about?

# Fake Algorithm: Brute Bayes Classifier

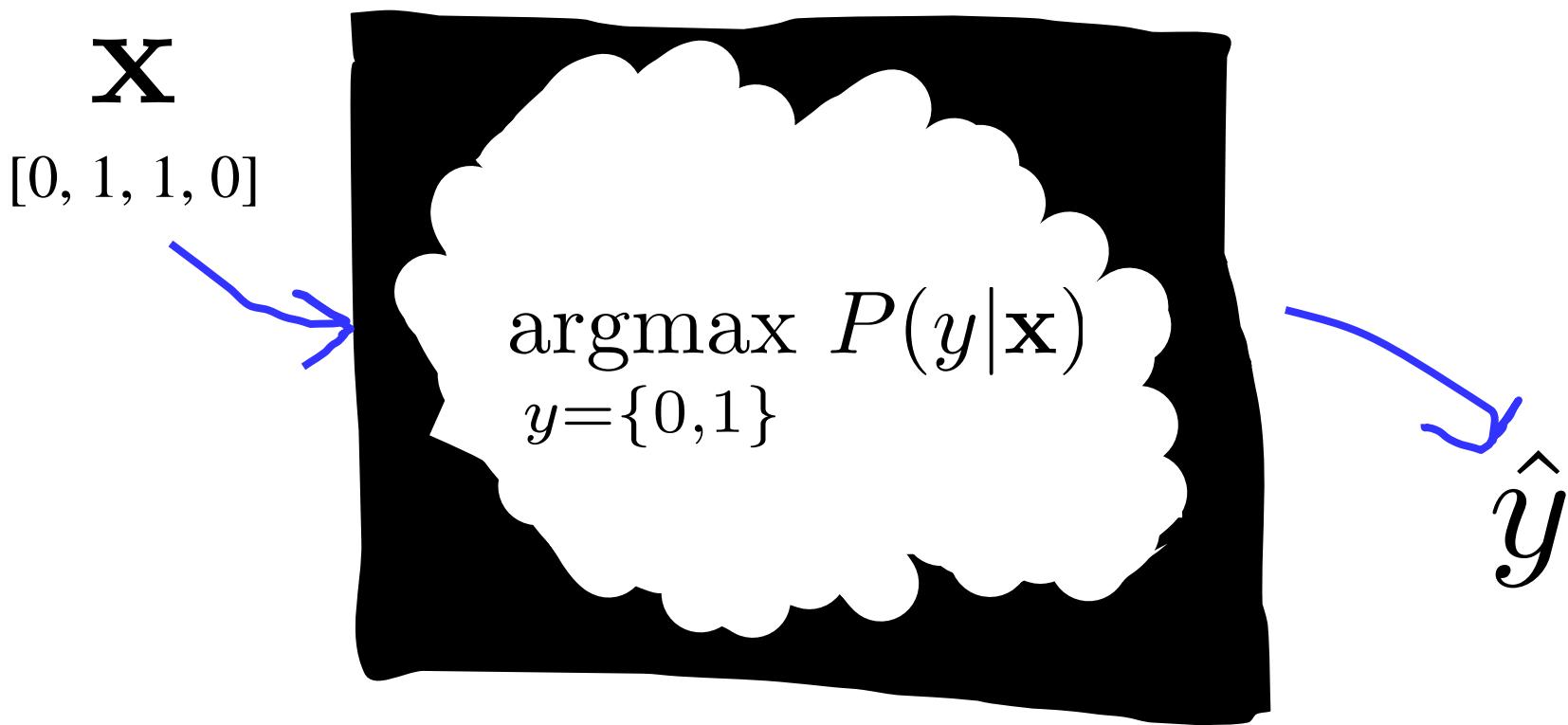
# Brute Force Bayes



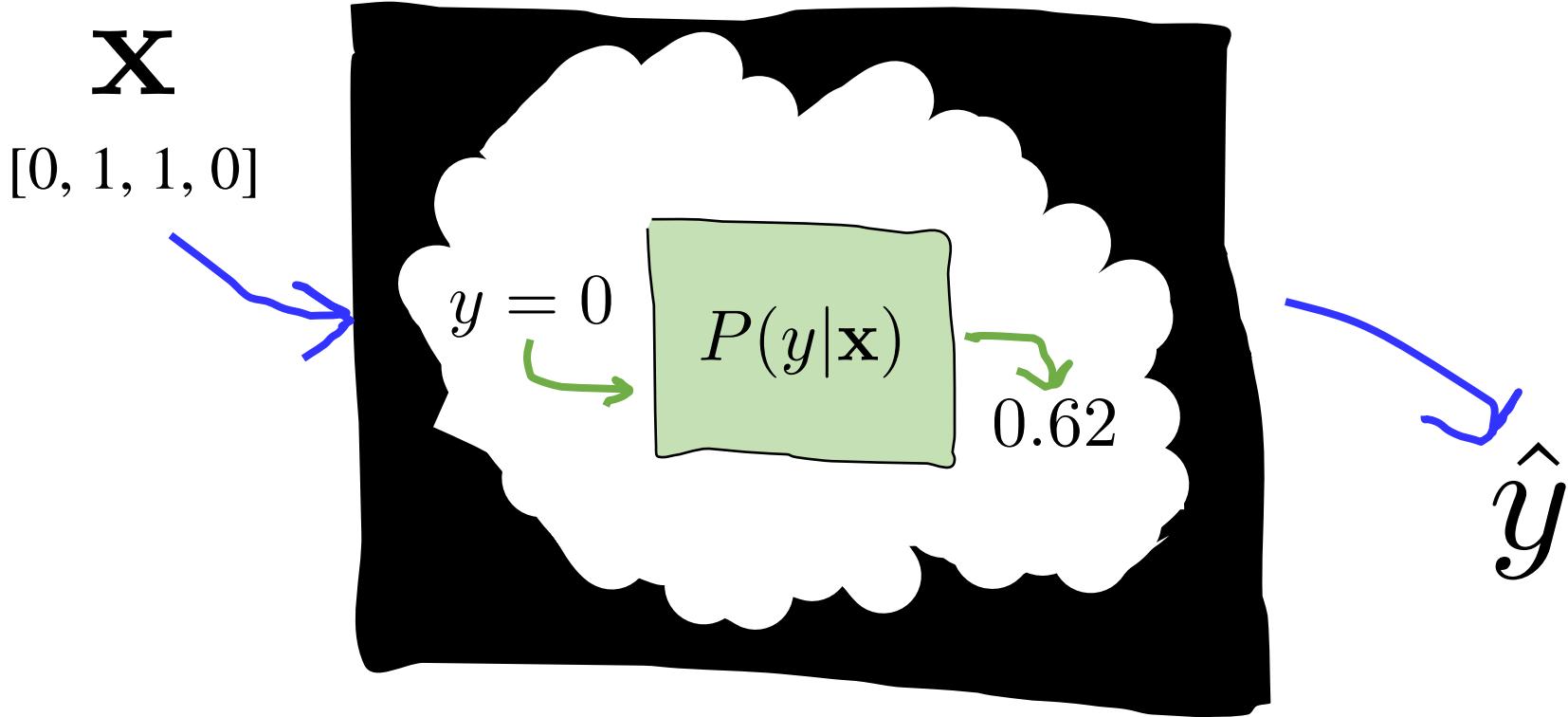
# Brute Force Bayes



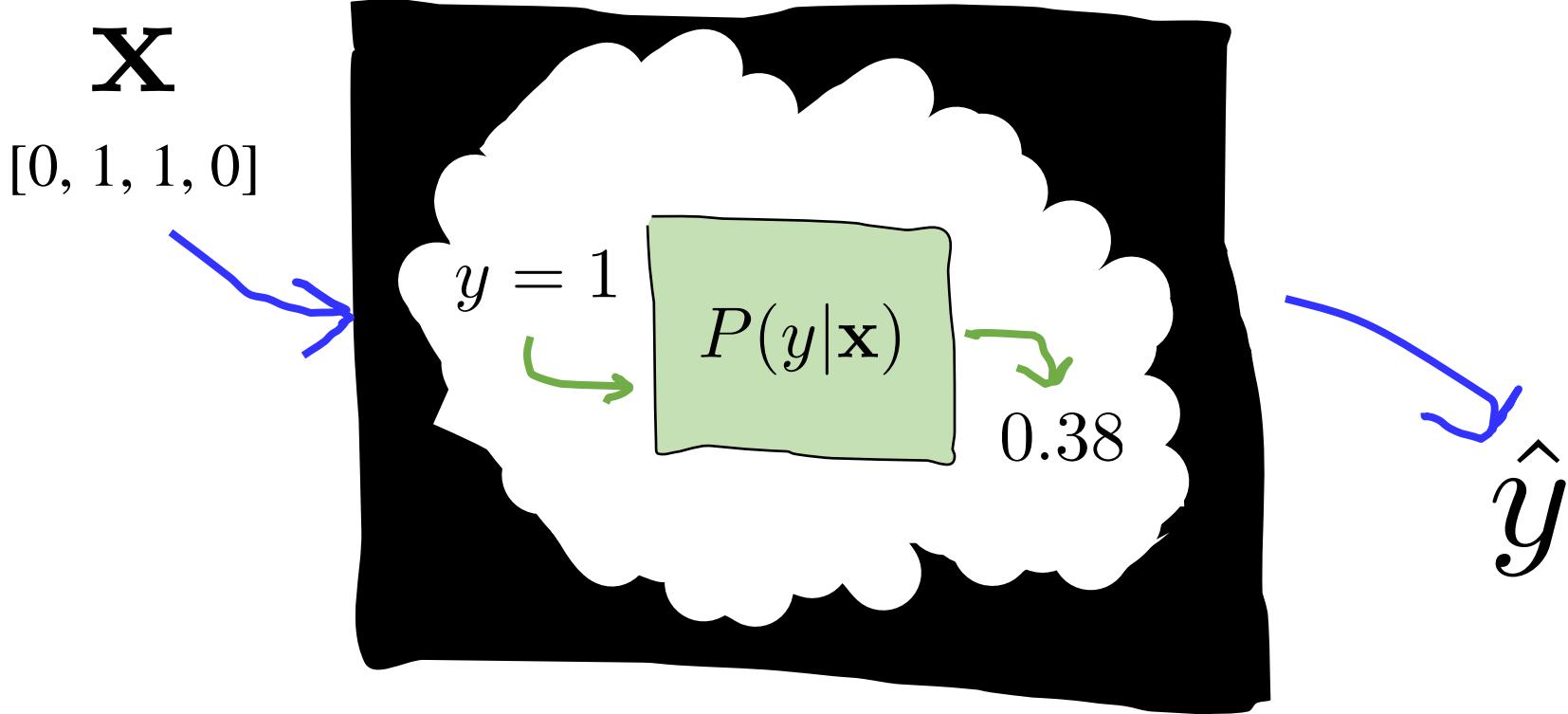
# Brute Force Bayes



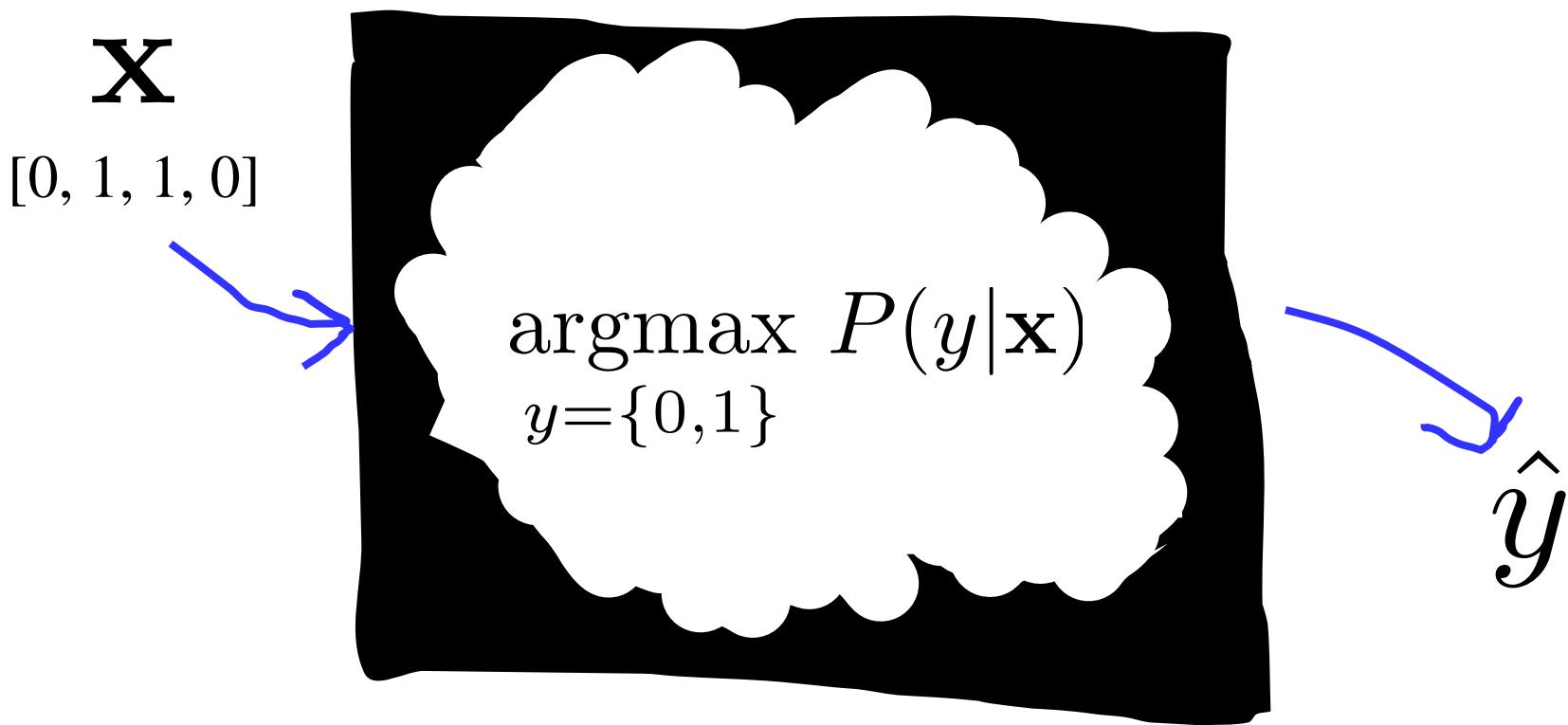
# Brute Force Bayes



# Brute Force Bayes



# Brute Force Bayes



# Brute Force Bayes

Prediction: will they like L.I.B.?

$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x})$$

If  $y = 1$ , they like L.I.B.?

Whether or not they liked Independence day

The formula  $\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x})$  represents the prediction process. It finds the class label  $\hat{y}$  that maximizes the probability  $P(y|\mathbf{x})$  for  $y \in \{0,1\}$ . The variable  $\mathbf{x}$  represents the input features, such as 'Whether or not they liked Independence day'.

Simply chose the class label that is the most likely given the data

This is for one user

# Brute Force Bayes

---

$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x})$$

Simply chose the class label that is the most likely given the data

This is for one user

# Brute Force Bayes

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\ &= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$

Simply chose the class label that is the most likely given the data

This is for one user

\* Note how similar this is to Hamilton example ☺

Stanford University

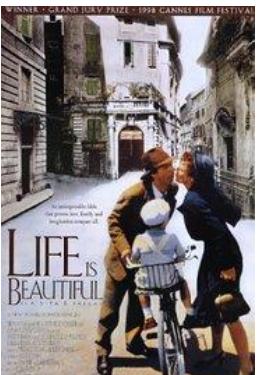
# What are the Parameters?

# Brute Force Bayes

$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)$$



Conditional probability  
table

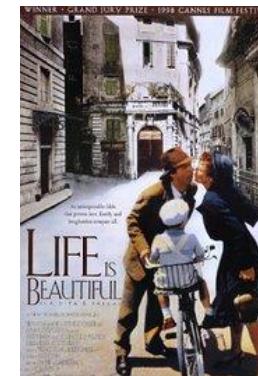


$Y = 0$

$X_1 = 0$	$\theta_0$
$X_1 = 1$	$\theta_1$

$Y = 1$

$X_1 = 0$	$\theta_2$
$X_1 = 1$	$\theta_3$



$Y = 0$

$Y = 1$

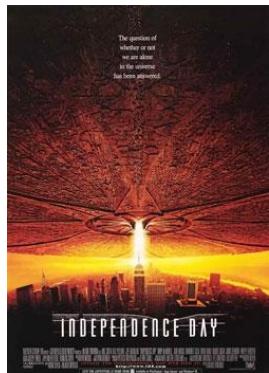
$\theta_4$
$\theta_5$

Learn these during training

Stanford University

# Brute Force Bayes

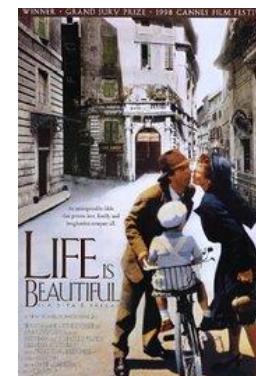
$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)$$



Conditional probability  
table



$X_1$	Y	
	0	1
0	$\theta_0$	$\theta_2$
1	$\theta_1$	$\theta_3$



$Y = 0$	$\theta_4$
$Y = 1$	$\theta_5$

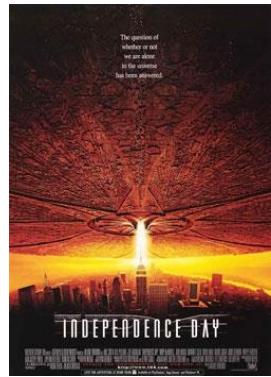
Learn these during training

Stanford University

Seems pretty good!

# Brute Force Bayes $m = 2$

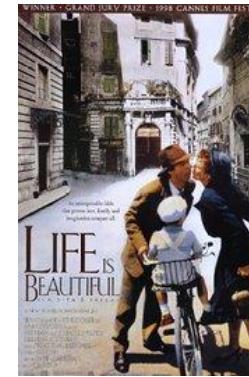
$x_1$



$x_2$



$y$



User 1

1

0

1

User 2

1

0

0

⋮

User  $n$

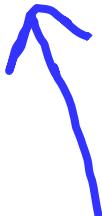
0

1

1

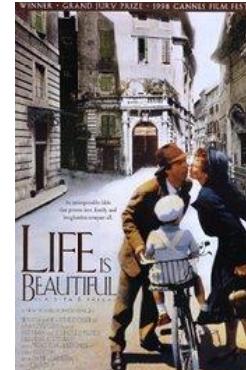
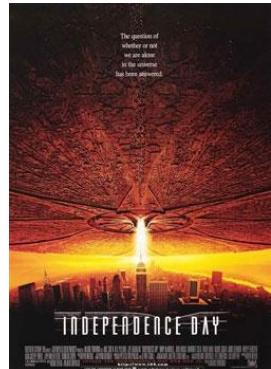
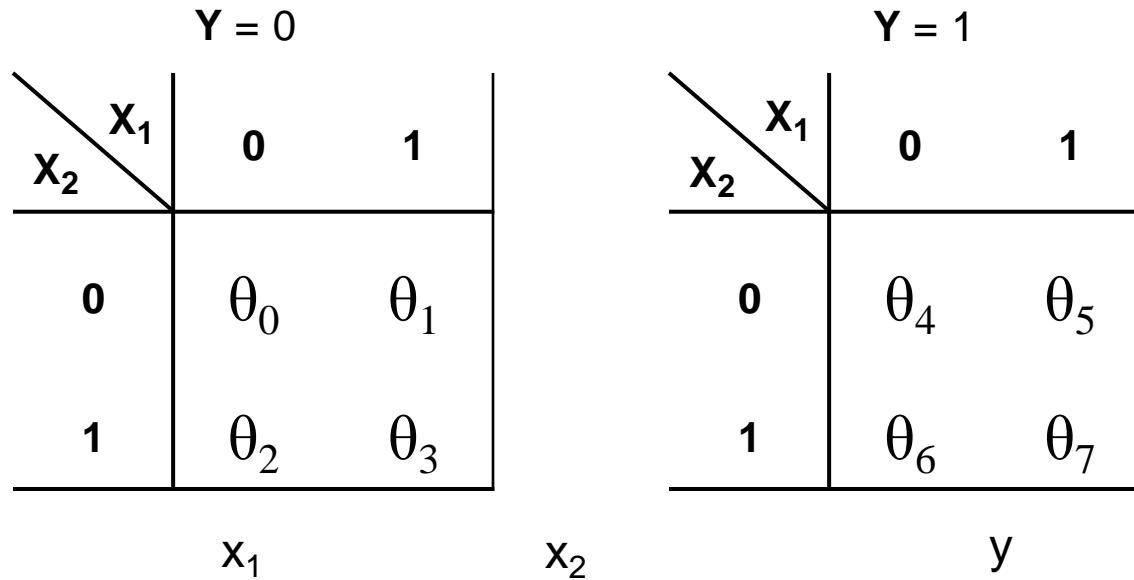
# Brute Force Bayes m = 2

Simply chose the class label that is the most likely given the data

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\ &= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$


# Brute Force Bayes

$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)$$



Fine

# Brute Force Bayes $m = 3$

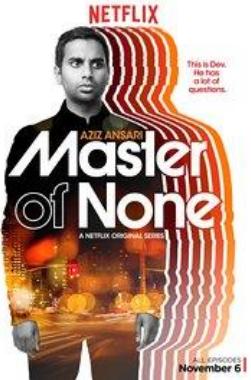
$x_1$



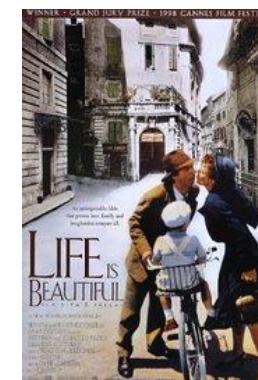
$x_2$



$x_3$



$y$



User 1

1

0

1

1

User 2

1

0

1

0

:

User  $n$

0

1

1

1

# Brute Force Bayes $m = 3$

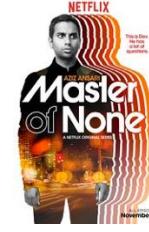
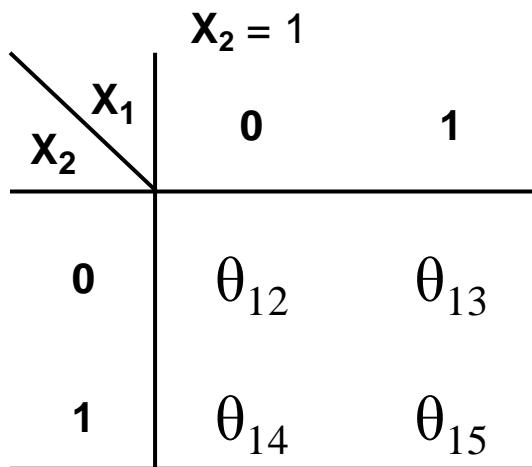
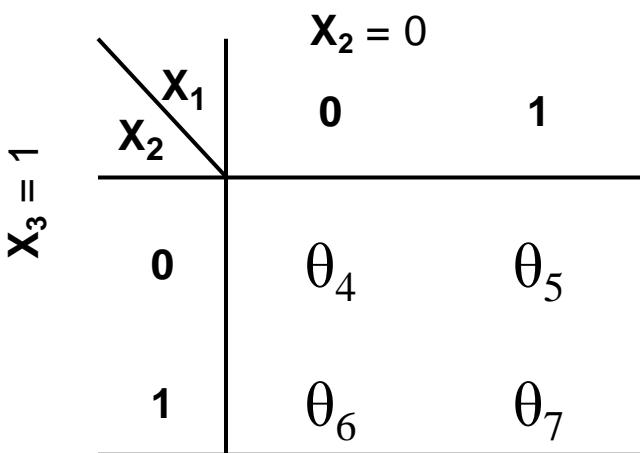
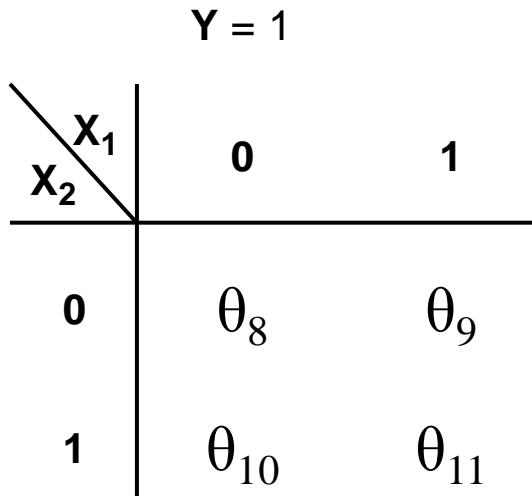
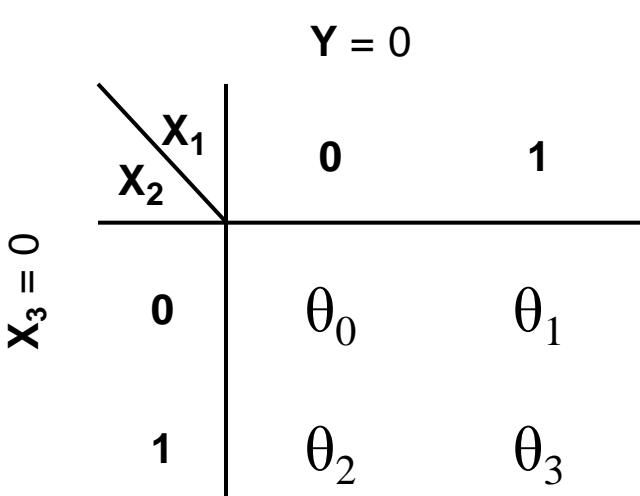
Simply chose the class label that is the most likely given the data

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\ &= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$

$$P(x_1, x_2, x_3|y)$$

# Brute Force Bayes

$$\hat{y} = \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)$$



Stanford University

And if  $m=100$ ?

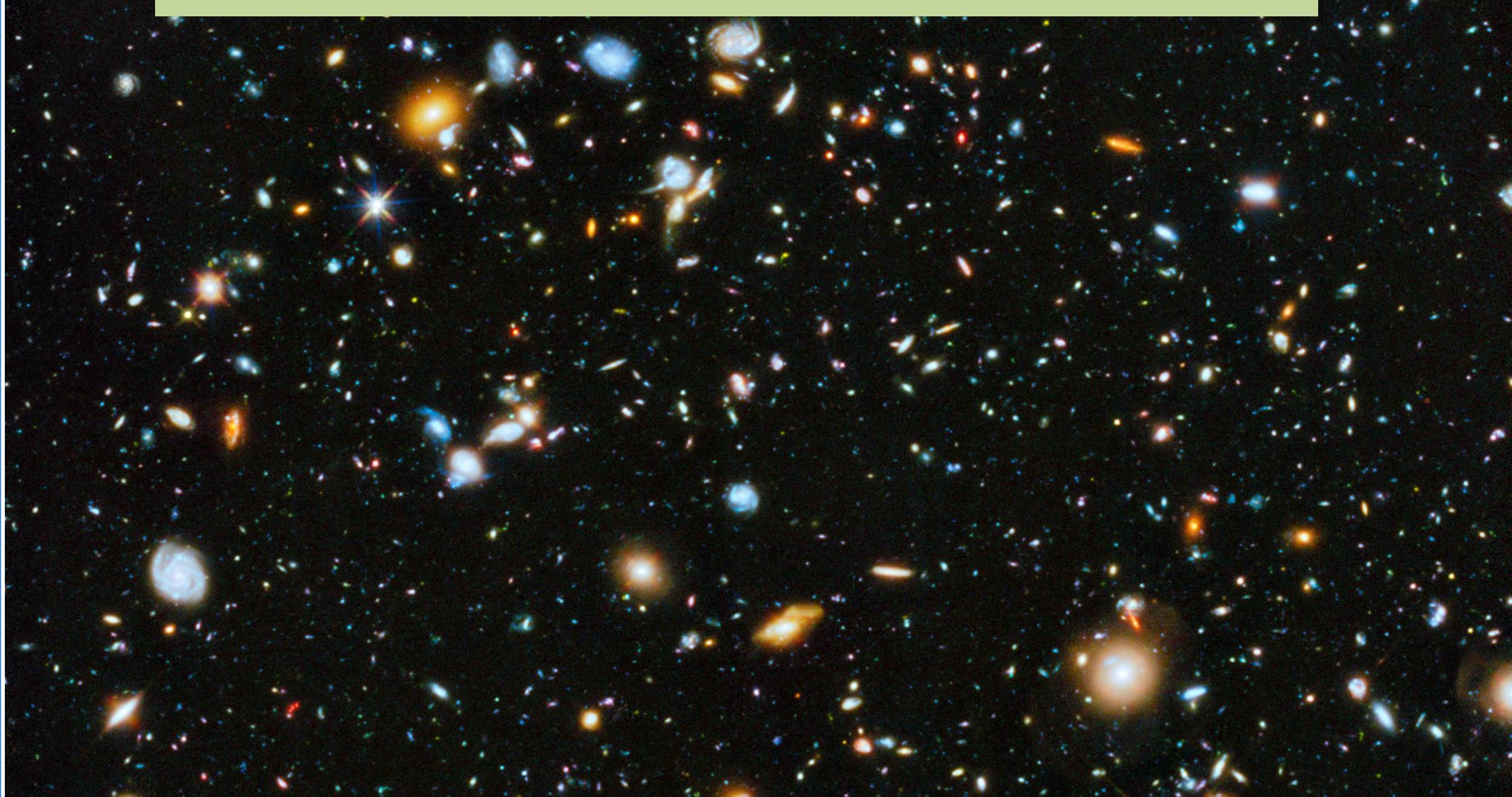
# Brute Force Bayes m = 100

Simply chose the class label that is the most likely given the data

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\ &= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\ &= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$


$$P(x_1, x_2, x_3, \dots, x_{100}|y)$$

# Oops... Number of atoms in the univserse



What is the big O for # parameters?  
 $m = \# \text{ features.}$

# Big O of Brute Force Joint

What is the big O for # parameters?  
 $m = \# \text{ features.}$

$$O(2^m)$$

Assuming each feature is binary...

Not going to cut it!

# What is the problem here?

---

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\&= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\&= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$

---

$$P(\mathbf{x}|y) = P(x_1, x_2, \dots, x_m|y)$$

# Naïve Bayes Assumption

$$\begin{aligned}\hat{y} &= \operatorname{argmax}_{y=\{0,1\}} P(y|\mathbf{x}) \\&= \operatorname{argmax}_{y=\{0,1\}} \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})} \\&= \operatorname{argmax}_{y=\{0,1\}} P(\mathbf{x}|y)P(y)\end{aligned}$$

---

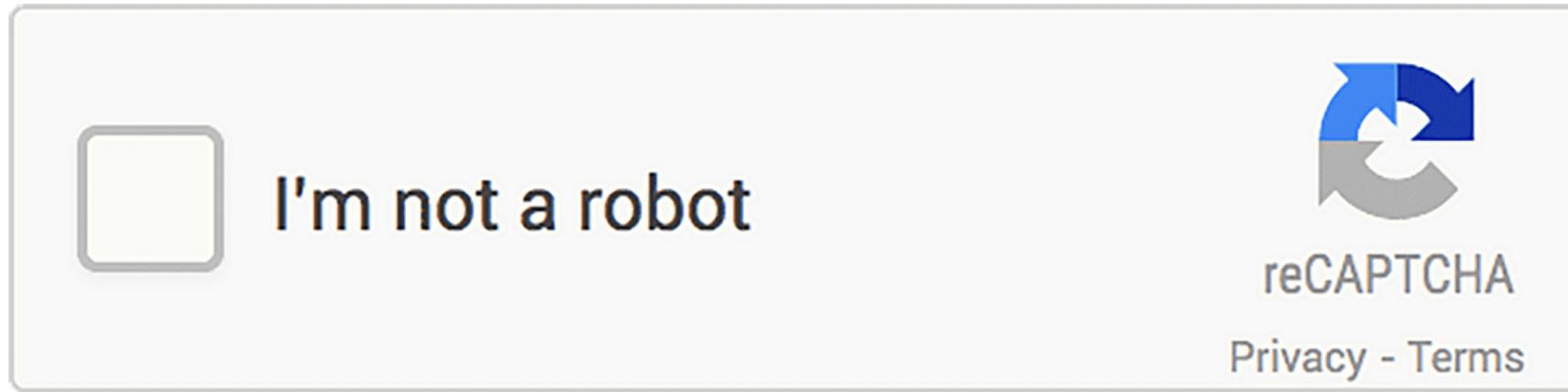
$$\begin{aligned}P(\mathbf{x}|y) &= P(x_1, x_2, \dots, x_m|y) \\&= \prod_i P(x_i|y)\end{aligned}$$

The Naïve Bayes assumption

# How Can We Compare?

# New Challenge: Captcha

---



# New Challenge: Captcha

	A	B	C	D	E	F	G
1	IP_Suspicious	X_Click	Y_Click	Time_to_Click	Browser_History_Length	Click_Deviat	Is_Human
2	0.00	49.39	44.78	-0.24	38.00	5.25	0
3	1.00	41.79	36.35	2.65	21.00	15.93	0
4	0.00	61.43	53.68	-1.38	26.00	12.01	1
5	0.00	76.02	64.43	-2.38	29.00	29.75	1
6	1.00	30.08	54.78	5.54	26.00	20.49	0
7	0.00	54.98	48.26	-0.34	31.00	5.28	1
8	0.00	56.29	53.77	1.86	20.00	7.34	0
9	0.00	65.55	49.57	-0.19	23.00	15.56	1
10	0.00	54.78	52.34	-0.37	34.00	5.33	1
11	0.00	50.19	49.23	4.68	33.00	0.80	1
12	0.00	49.49	58.41	0.03	29.00	8.43	1
13	0.00	44.71	34.81	3.39	21.00	16.08	1
14	0.00	61.97	47.61	0.13	27.00	12.21	0



This makes it a classification task

# Comparing Algorithms: Buggy Metric

---

Algorithm	Accuracy on the Data
Logistic Regression	0.7
Naïve Bayes	0.6
Nearest Neighbor	

# Comparing Algorithms: Buggy Metric

---

Algorithm	Accuracy on the Data
Logistic Regression	0.7
Naïve Bayes	0.6
Nearest Neighbor	<b>1.0</b>

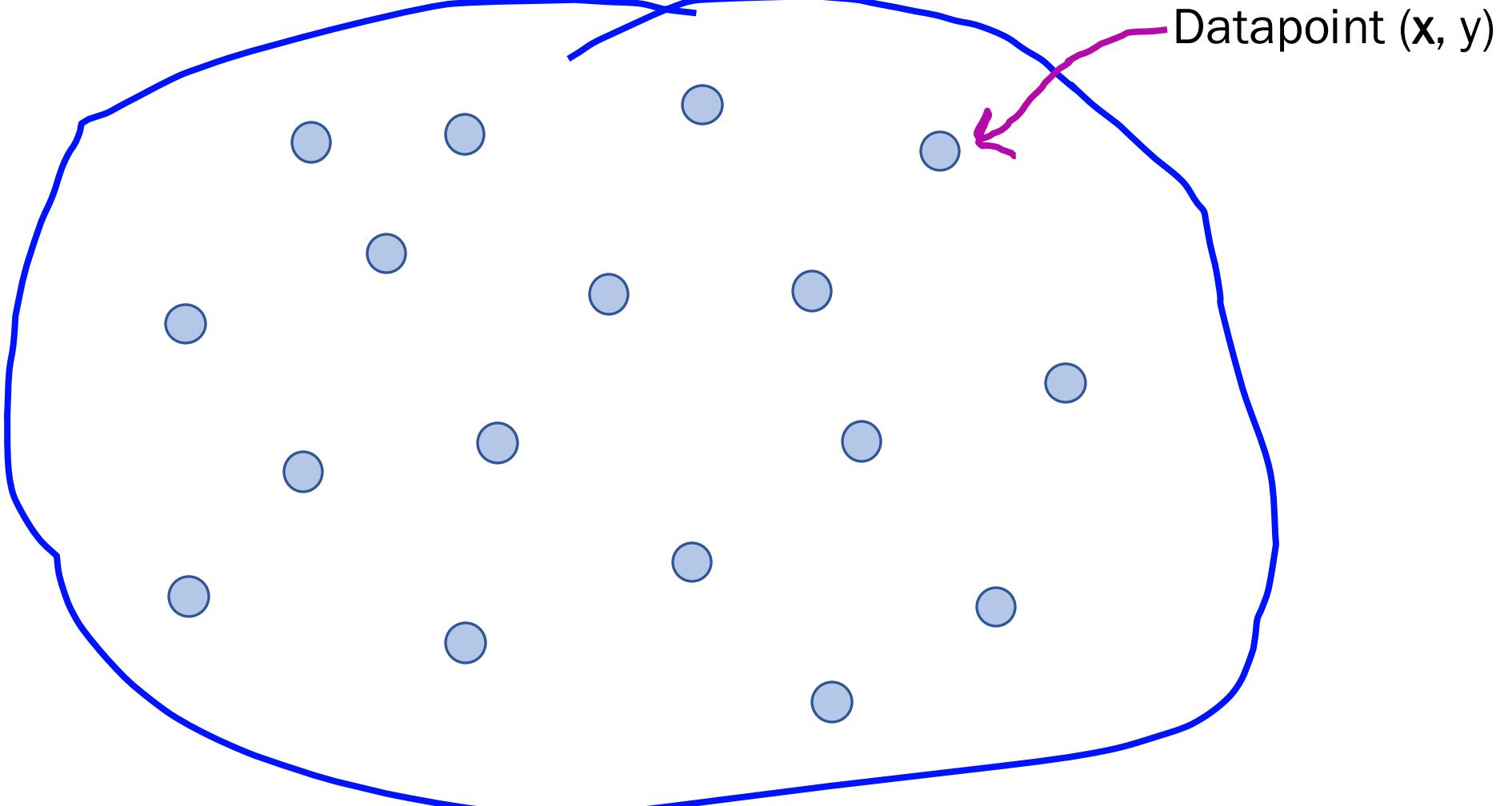
# Comparing Algorithms: Better Metric

---

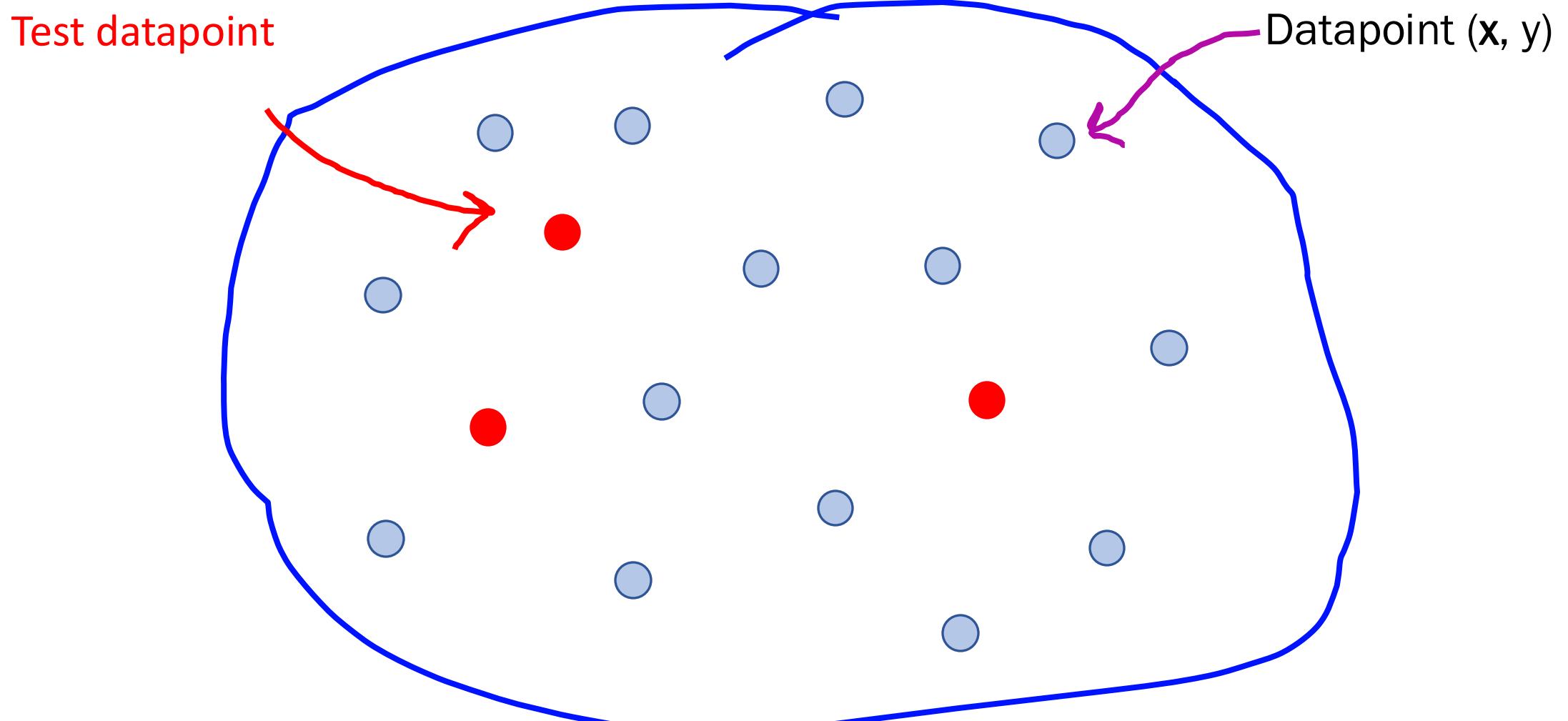
Algorithm	Accuracy on Train Data	Accuracy on Test Data
Logistic Regression	0.7	0.65
Naïve Bayes	0.6	0.60
Nearest Neighbor	1.0	0.56

# Train / Test Split

---



# Train / Test Split

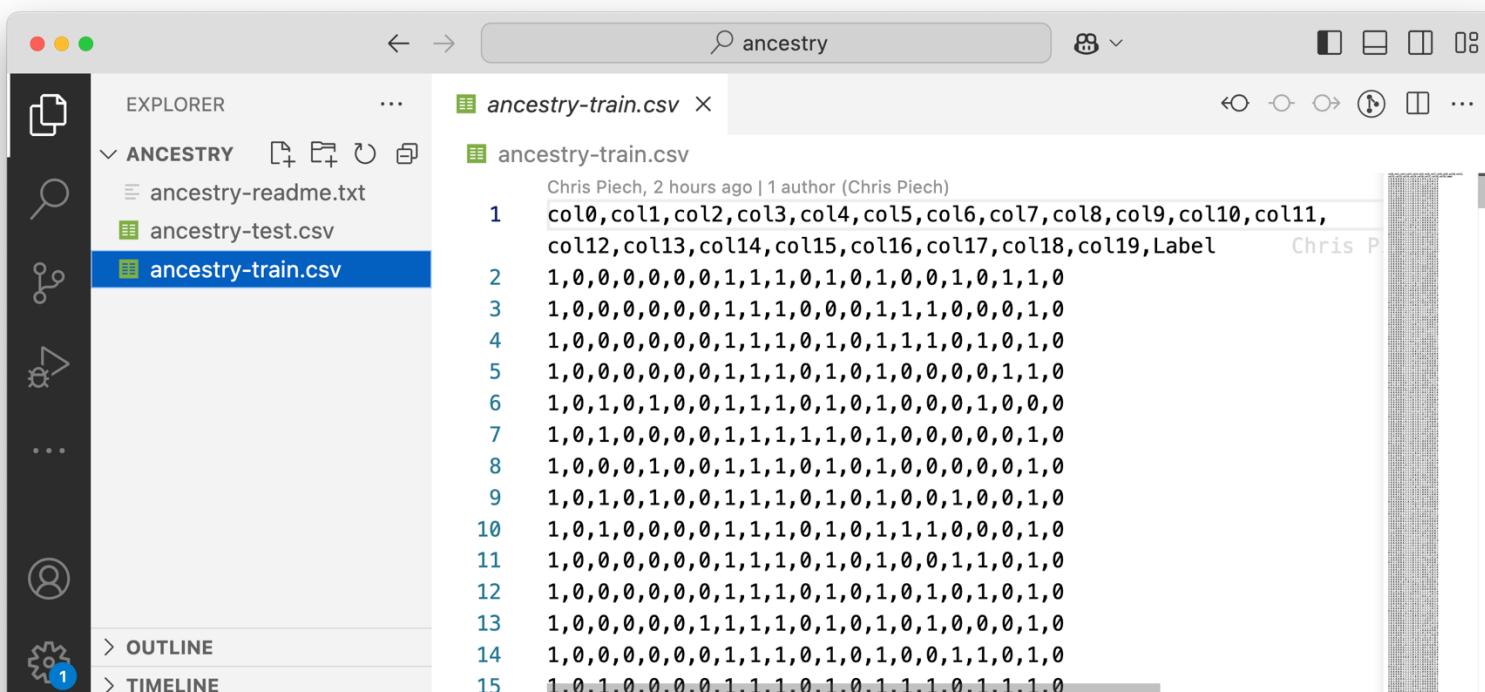


# Train / Test Split

```
# Train/test split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)  
  
# Train dataset  
print(X_train, y_train)  
  
# Test dataset  
print(X_test, y_test)
```

In your pset we will give you these pre-split

train.csv

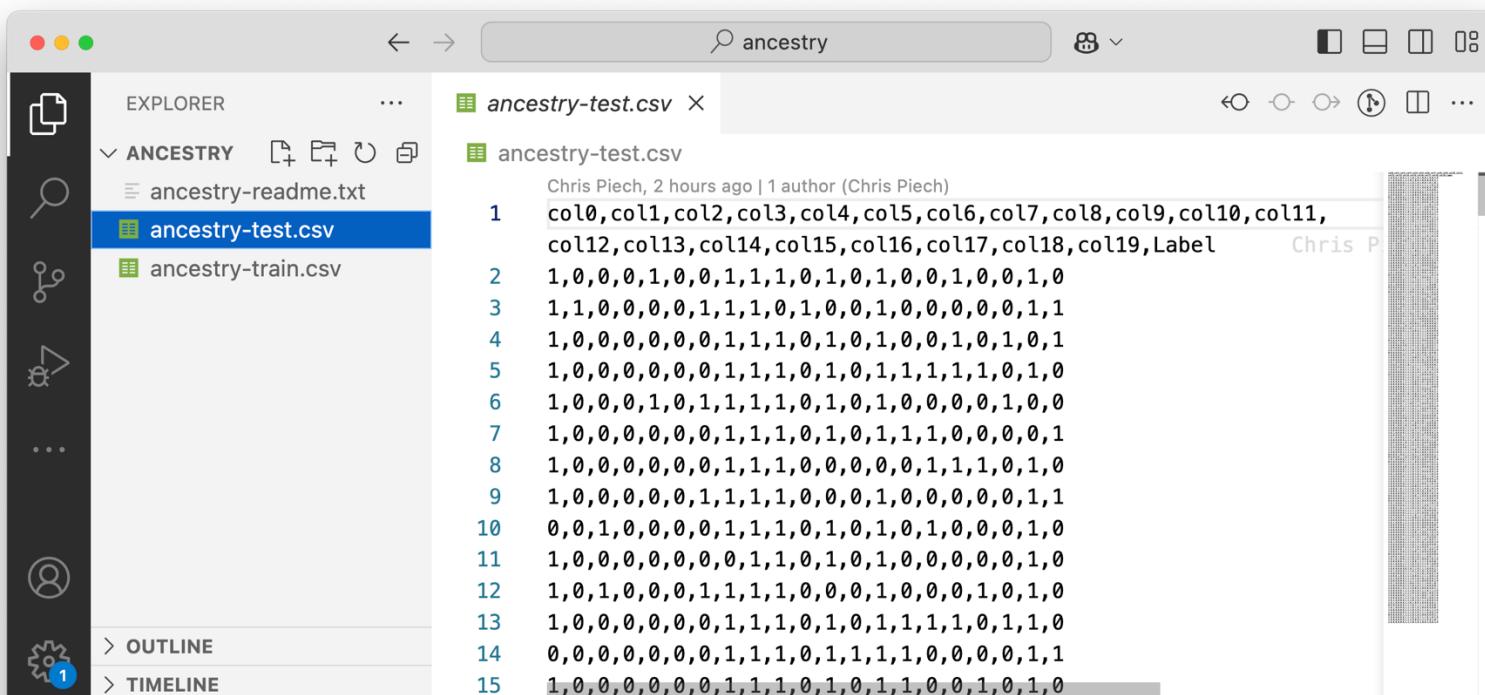


# Train / Test Split

```
# Train/test split  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)  
  
# Train dataset  
print(X_train, y_train)  
  
# Test dataset  
print(X_test, y_test)
```

In your pset we will give you these pre-split

test.csv



To the Code

# Results

---

Model	Train Accuracy	Test Accuracy
Baseline	0.6138	0.6300
Logistic Regression	0.7300	0.7200
Naive Bayes	0.7275	0.7200
Decision Tree	0.7975	0.6150
Random Forest	0.7950	0.7100
Gradient Boosting	0.7738	0.7250
AdaBoost	0.7588	0.7100

# Results

---

Model	Train Accuracy	Test Accuracy
<hr/>		
<b>Baseline</b>	<b>0.6138</b>	<b>0.6300</b>
Logistic Regression	0.7300	0.7200
Naive Bayes	0.7275	0.7200
Decision Tree	0.7975	0.6150
Random Forest	0.7950	0.7100
Gradient Boosting	0.7738	0.7250
AdaBoost	0.7588	0.7100

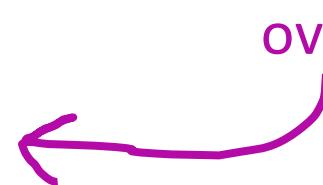


Always predict 1

# Results

---

Model	Train Accuracy	Test Accuracy
Baseline	0.6138	0.6300
Logistic Regression	0.7300	0.7200
Naive Bayes	0.7275	0.7200
Decision Tree	<b>0.7975</b>	<b>0.6150</b>
Random Forest	<b>0.7950</b>	<b>0.7100</b>
Gradient Boosting	<b>0.7738</b>	<b>0.7250</b>
AdaBoost	<b>0.7588</b>	<b>0.7100</b>



overfitting

# Results

Model	Train Accuracy	Test Accuracy
Baseline	0.6138	0.6300
Logistic Regression	0.7300	0.7200
Naive Bayes	0.7275	0.7200
Decision Tree	0.7975	0.6150
Random Forest	0.7950	0.7100
<b>Gradient Boosting</b>	<b>0.7738</b>	<b>0.7250</b>
AdaBoost	0.7588	0.7100

The Kaggle Champion



The **Gradient Boosting** is by far the most successful single model.

Especially the package XGB is used in pretty much every winning (and probably top 50%) solution. XGB has essentially become the first model you try and the best performing single model by the end.



Kaggle

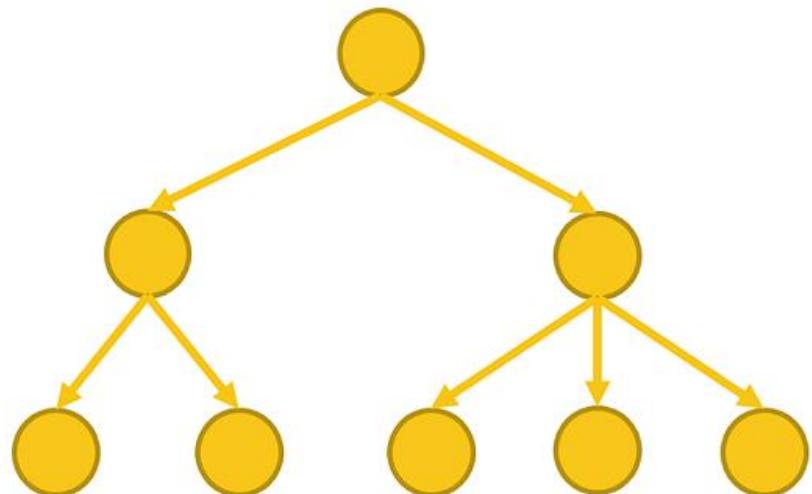
<https://www.kaggle.com> › general ::

[What machine learning approaches have won most ... - Kaggle](#)

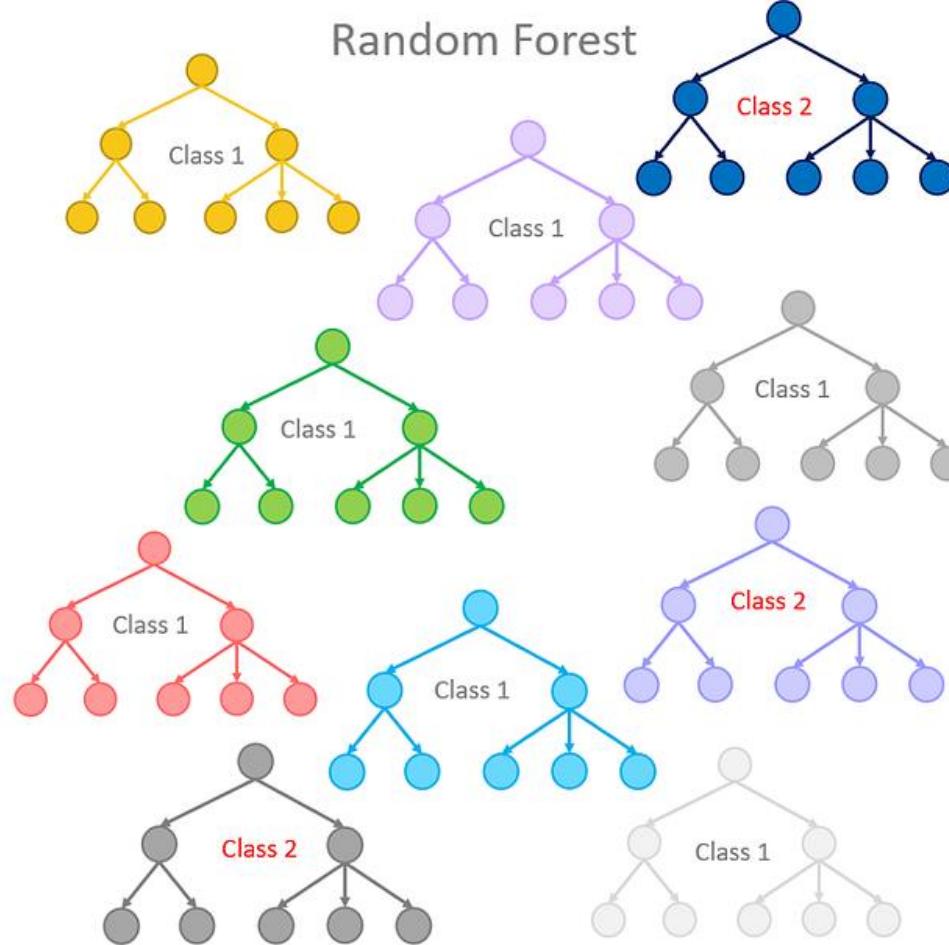
XGB is for  
extreme  
gradient  
boosting

# Decision Trees vs Random Forests

Single Decision Tree



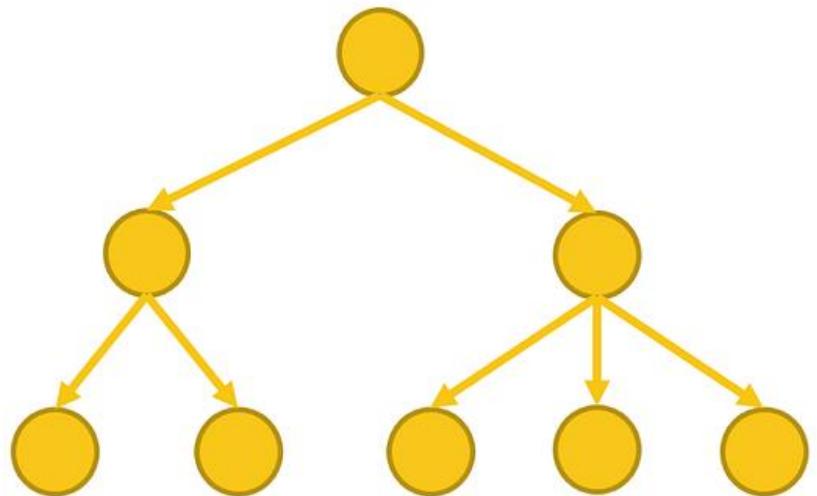
Random Forest



# Gradient Boosting

---

Successive small decision trees



Are predicting the “residual”

$$\frac{\partial LL(\theta)}{\partial \hat{y}^{(i)}} = y^{(i)} - \hat{y}^{(i)}$$

# Three Guiding Questions

---

1. What are other models for classification?



2. For a dataset, how do you tell which one is best?



3. What are other validation metrics I might care about?

How **well calibrated** are my probabilities?

# Measuring if Probabilities are Calibrated

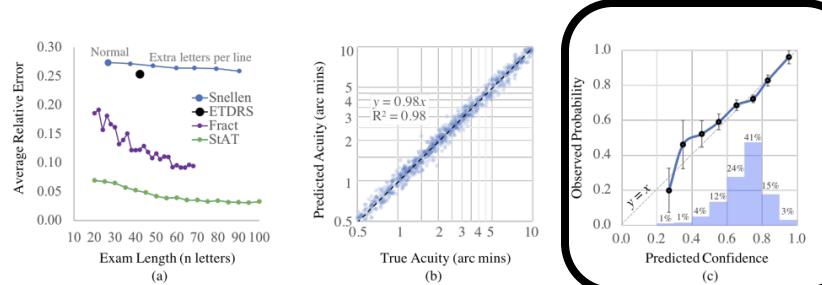


Figure 4: (a) The tradeoff between length of exam and error for the different algorithms. (b) A visualization of the predictions made by StACT. (c) Calibration test: StACT confidences correspond to how often it is correct.

## 4.2 Baseline Acuity Tests

We use the following baselines and prior algorithms to compare against the StACT algorithm.

**Const Policy.** This policy always predicts the most common visual acuity in our data i.e. the mode of the visual acuity prior. This serves as a true null model because it doesn't take patient responses into account at all.

**Snellen and ETDRS.** The Revised 2000 Series ETDRS charts and the Traditional Snellen Eye Chart were programmed so that we could simulate their response to different virtual patients. Both exams continue until the user incorrectly answers questions for more than half of the letters on a line. ETDRS has a function for predicted acuity score that takes into account both the last line passed, and how many letters were read on the last line not-passed. Both charts use 19 unique optotypes.

**FrACT.** We use an implementation of the FrACT algorithm (Bach and others 1996), with the help of code graciously shared by the original author. We also included the ability to learn the “*s*” parameter as suggested by the 2006 paper (Bach 2006), and verified that it improved performance.

## 5 Results and Evaluation

The results of the experiments can be seen in Table 1.

**Accuracy and error.** As can be seen from Table 1, the StACT test has substantially less error than all the other baselines. After 20 optotype queries, our algorithm is capable of predicting acuity with an average relative error of 0.069. This prediction is a 74% reduction in error from our implementation of the ubiquitous Snellen test (average error = 0.276), as well as a 67% reduction in error from the FrACT test (average error = 0.212). One possible reason for the improvement over FrACT is that the simulations used in our evaluations are based off the Floored-Exponential model that StACT uses. However, even when we evaluate StACT on simulations drawn from the FrACT logistic assumption we still achieve a 41% reduction. The improved accuracy of the StACT algorithm suggests our Bayesian approach

	$\mu$ Acuity Error	$\mu$ Test length
Const	0.536	0
Snellen <sup>†</sup>	0.264	27
ETDRS <sup>†</sup>	0.254	42
FrACT	0.212	20
StACT	<b>0.069</b>	20
StACT-noSlip	0.150	20
StACT-greedyMAP	0.132	20
StACT-logistic	0.125	20
StACT-noPrior	0.090	20
StACT-goodPrior	<b>0.047</b>	20
StACT-star	<b>0.038</b>	63

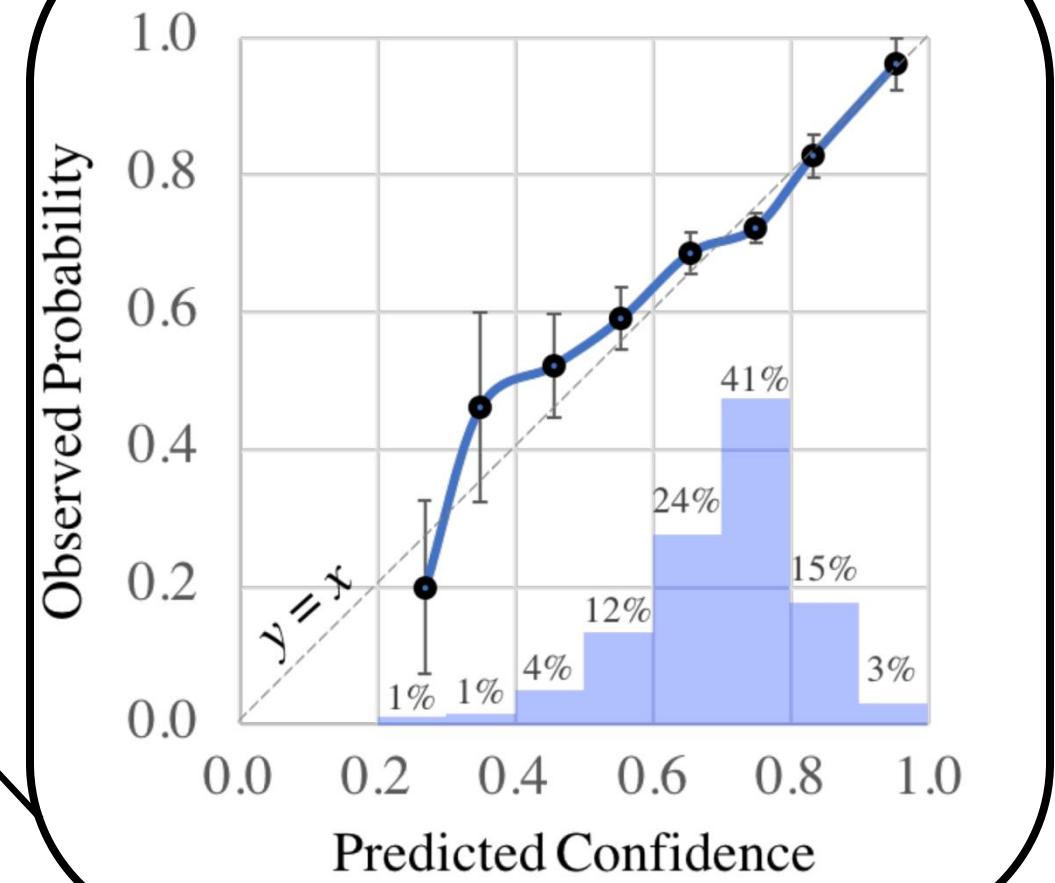
Table 1: Average relative error for each algorithm. Except for Snellen each test was allowed 20 letters. Results are average relative error after 1000 tests. <sup>†</sup> Snellen and ETDRS used 19 unique optotypes.

to measuring acuity is a fruitful proposal both because of our introduction of the floored exponential as well as our Thompson-sampling inspired algorithm to choose a next letter size.

Figure 4 (b) visualizes what StACT's small relative error means in terms of predictions. Each point in the plot is a single patient. The x-axis is the true acuity of the patient and the y-axis is the predicted accuracy. We can qualitatively observe that the predictions are often accurate, there are no truly erroneous predictions, and that the exam is similarly accurate for patients of all visual acuities.

Moreover, as seen in Figure 4 (a), StACT's significant improvement in error rate holds even when the length of the exam is increased. It is also evident that increasing exam length reduces our error rate: if we increase the exam length to 200 letters, the average error of StACT falls to 0.020. While this is highly accurate, its far too long an exam, even for patients who need to know their acuity to high precision.

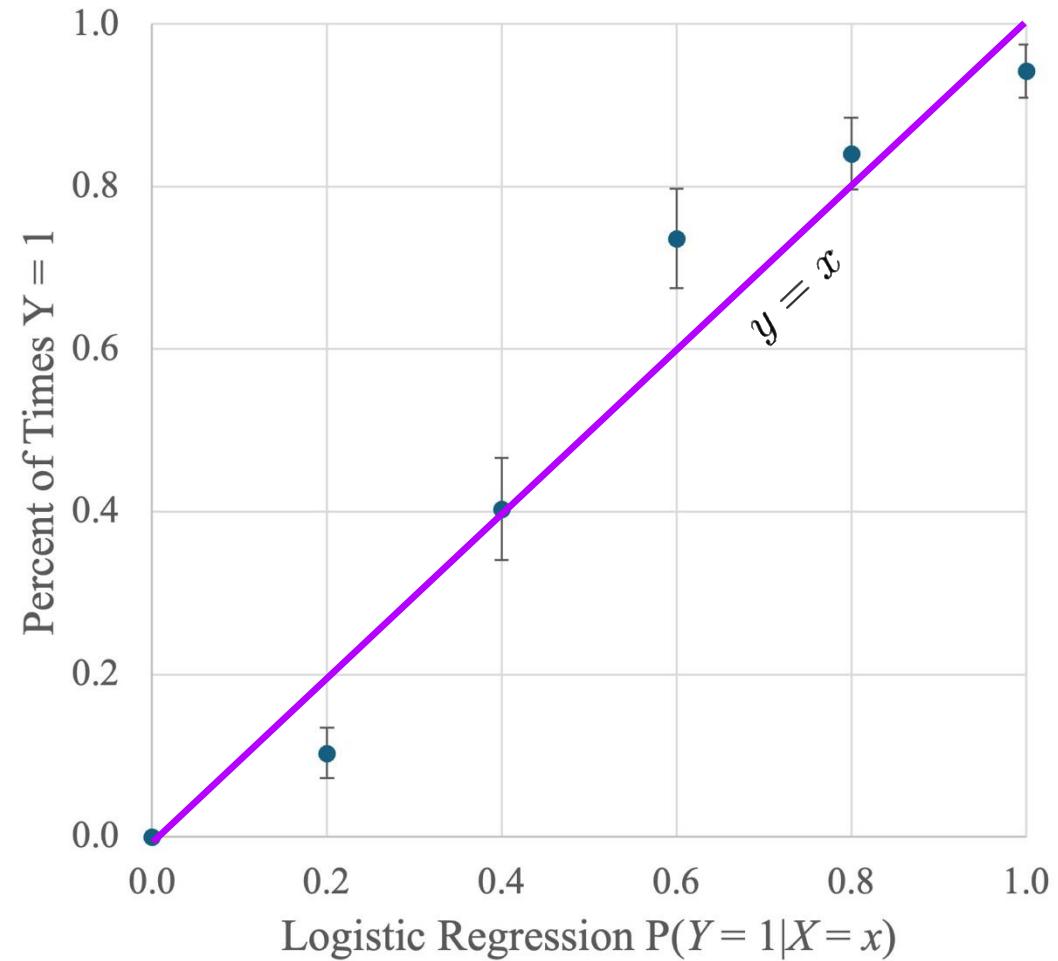
**StACT Star Exam.** Our primary experiments had a fixed



# Logistic Regression Calibration Curve

$$x_1 \quad x_2 \quad x_{19} \quad y \quad \checkmark \quad P(Y = 1 | \mathbf{X} = \mathbf{x}) = \sigma\left(\sum_i \theta_i x_i\right)$$

	A	B	T	U	V
1	col1	col2	col19	Label	LogRegPr
2	1	0	1	0	0.237
3	1	1	1	1	0.928
4	1	0	0	1	0.541
5	1	0	1	0	0.003
6	1	0	0	0	0.914
7	1	0	0	1	0.432
8	1	0	1	0	0.001
9	1	0	1	1	0.530
10	0	0	1	0	0.090
11	1	0	1	0	0.439
12	1	0	1	0	0.032
13	1	0	1	0	0.114
14	0	0	1	1	0.848
15	1	0	1	0	0.025
16	1	0	1	0	0.180
17	0	0	1	1	0.672
18	1	0	1	1	0.531
19	1	0	1	0	0.012
20	1	0	1	1	0.560
21	1	0	1	1	0.502



Is the Model Fair?

# Two Philosophic Values of Fairness

---

## Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

## Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

# Two Philosophic Values of Fairness

---

## Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

## Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.

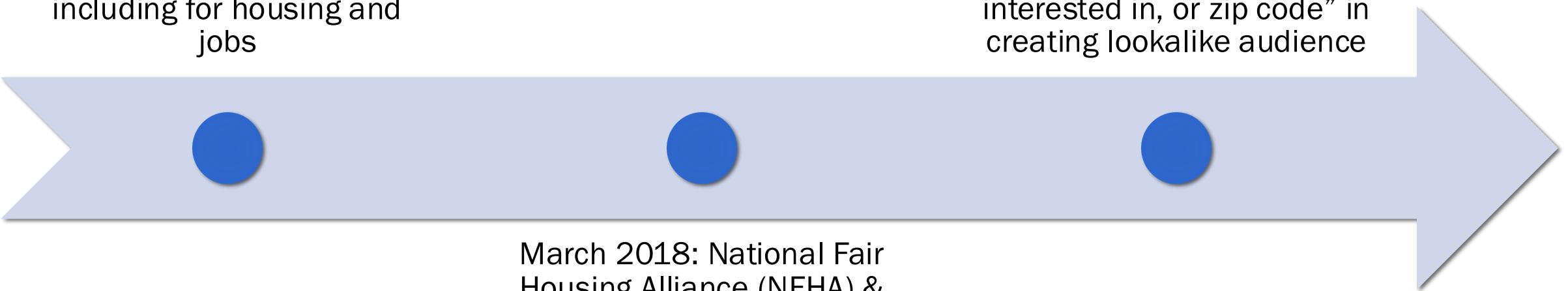
Fairness through unawareness



# Case Study: Facebook Ads & Job/Housing Recommendations

Facebook creates “Lookalike” feature for advertisers: upload a “source list” and find users with “common qualities” to target ads, including for housing and jobs

March 2019: As part of settlement, Facebook agrees not to use “age, gender, relationship status, religious views, school, political views, interested in, or zip code” in creating lookalike audience



March 2018: National Fair Housing Alliance (NFHA) & other civil rights groups sue Facebook over violations of the Fair Housing Act

# Facebook Input Lookalikes

The screenshot shows the 'Create a Lookalike Audience' interface in the Facebook Ads Manager. It consists of three main steps:

- 1 Select Your Lookalike Source**: A dropdown menu labeled 'Select an existing audience or data source' with a 'Create New Source' button.
- 2 Select Audience Location**: A dropdown menu showing 'Countries > North America' and 'United States' selected. A search bar below it says 'Search for regions or countries'.
- 3 Select Audience Size**: A slider labeled 'Number of lookalike audiences' set to 1. The slider scale ranges from 0% to 8%, with '2.3M' displayed above the 1% mark. A note at the bottom states: 'Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% lookalike consists of the people in your lookalike source. Increasing the percentage creates a bigger, broader audience.'

The screenshot shows the 'Create a Special Ad Audience' interface in the Facebook Ads Manager. It follows a similar three-step process:

- 1 Select Your Source**: A dropdown menu labeled 'Select an existing audience or data source'.
- 2 Select Audience Location**: A dropdown menu showing 'Countries > North America' and 'United States' selected. A search bar below it says 'Search for regions or countries'.
- 3 Select Audience Size**: A slider labeled 'Number of Special Ad Audiences' set to 1. The slider scale ranges from 0% to 8%, with '2.3M' displayed above the 1% mark. A note at the bottom states: 'Audience size ranges from 1% to 10% of the combined population of your selected locations. A 1% Special Ad Audience consists of the most similar online behavior to your source. Increasing the percentage creates a bigger, broader audience.'

# New “Special Ad” Audiences Still Biased

Gender: Equally Biased

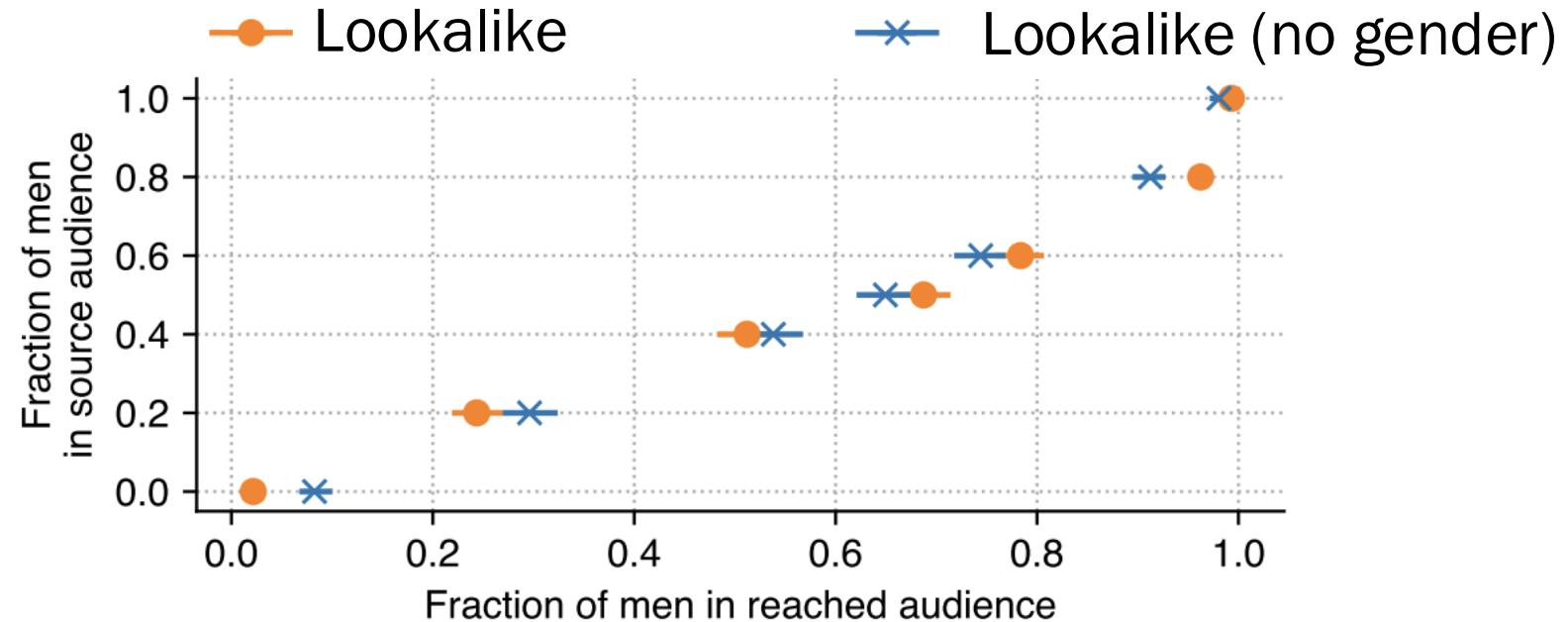
Age: Almost as Biased

Race: more difficult to measure given the tools provided but still somewhat biased

Political Views: Less Biased

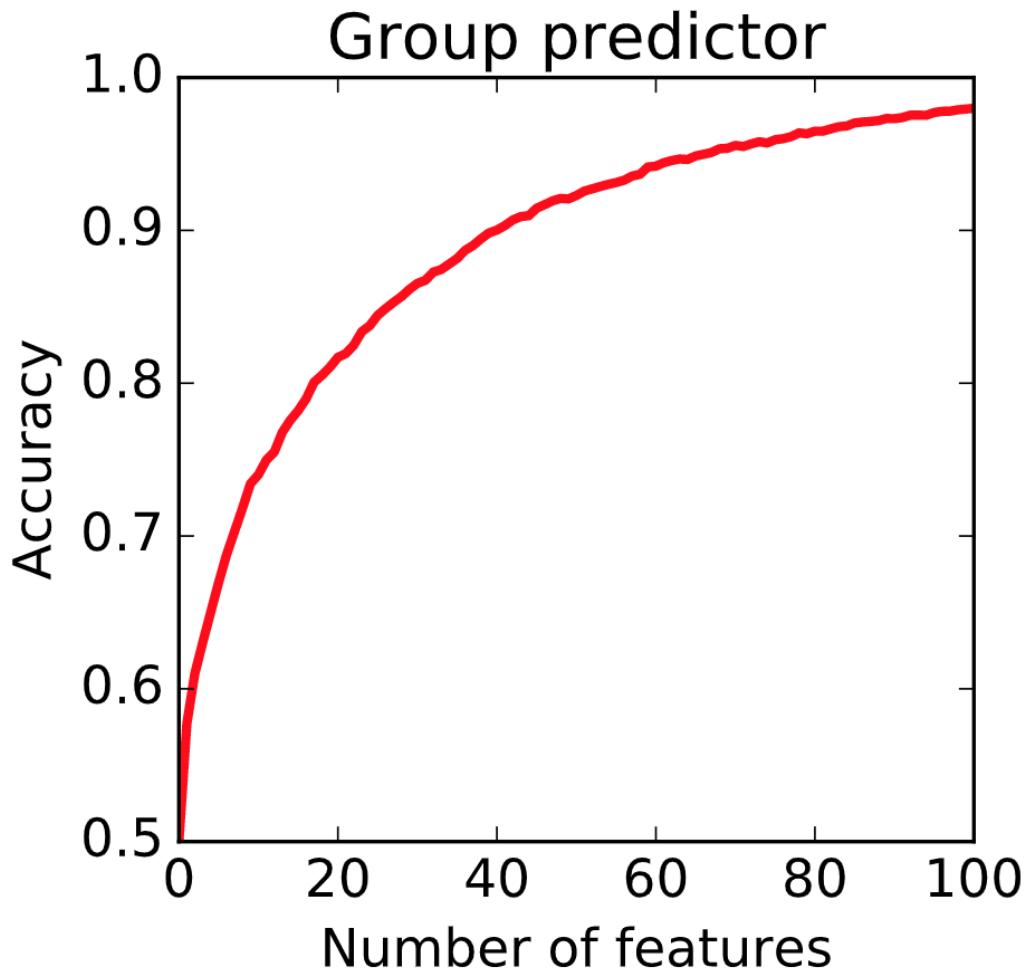
Sapiezynski et. al 2019,

<https://sapiezynski.com/papers/sapiezynski2019algorithms.pdf>



**Figure 2: Gender breakdown of ad delivery to Lookalike and Special Ad audiences created from the same source audience with varying fraction of male users, using the same ad creative. We can observe that both Lookalike and Special Ad audiences reflect the gender distribution of the source audience, despite the lack of gender being provided as an input to Special Ad Audiences.**

Yo, Piotr, you got your axis backwards 😊



## Many Features = Accurate Group Prediction

Sensitive attributes are often “redundantly encoded” in the dataset

Many of the features or datapoints are correlated with the sensitive attribute

# Two Philosophic Values of Fairness

---

## Procedural Fairness:

Focuses on the decision-making or classification process, ensures that the algorithm does not rely on unfair features.

## Distributive Fairness:

Focuses on the decision-making or classification *outcome*, ensures that the distribution of good and bad outcomes is equitable.



Fairness through unawareness  
(facebook example shows this is hard)

# Let's Try Fairness Through Awareness!

Awareness of what?

# Fairness Through Awareness Terms

---

$D$ : protected demographic

$G$ : guess of your model (aka  $\hat{y}$ )

$T$ : the true value (aka  $y$ )

$D = 0$

$D = 1$

	$G = 0$	$G = 1$
$T = 0$	0.21	0.32
$T = 1$	0.07	0.28

	$G = 0$	$G = 1$
$T = 0$	0.01	0.01
$T = 1$	0.02	0.08

# Distributive Fairness #1: Parity

---

## Fairness definition #1: Parity

An algorithm satisfies “parity” if the probability that the algorithm makes a positive prediction ( $G = 1$ ) is the same regardless of begin conditioned on demographic variable.

$D$ : protected demographic

$G$ : guess of your model (aka  $y$  hat)

$T$ : the true value (aka  $y$ )

$$P(G=1|D=1) = P(G = 1 \mid D = 0)$$

# Distributive Fairness #2: Calibration

---

## Fairness definition #2: Calibration

An algorithm satisfies “calibration” if the probability that the algorithm is correct ( $G = T$ ) is the same regardless of demographics.

$D$ : protected demographic

$G$ : guess of your model (aka  $\hat{y}$ )

$T$ : the true value (aka  $y$ )

$$P(G = T|D = 0) = P(G = T|D = 1)$$

# Calibration (Relaxed)

## Fairness definition #2: Calibration

An algorithm satisfies “calibration” if the probability that the algorithm is correct ( $G = T$ ) is the same regardless of demographics.

$D$ : protected demographic

$G$ : guess of your model (aka  $y$  hat)

$T$ : the true value (aka  $y$ )

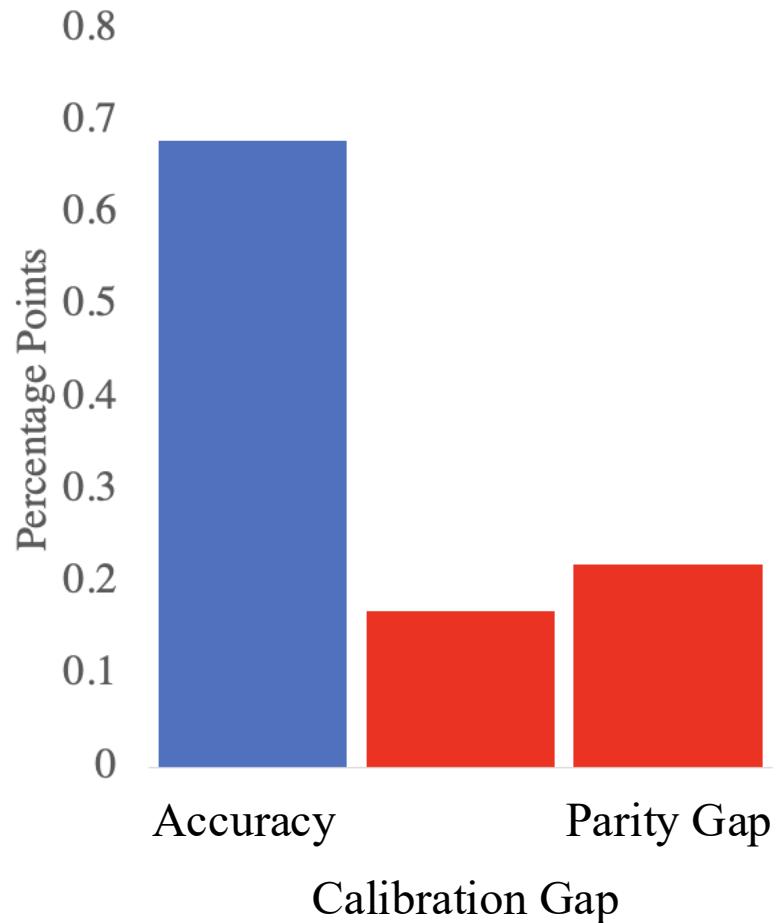
$$\frac{P(G = T|D = 1)}{P(G = T|D = 0)} \geq 1 - \epsilon \quad \text{Where epsilon} = 0.2$$

US legal standard: “disparate impact,” also known as the 80% rule.

# COMPAS: Biased Against Black Inmates

---

## Before: Compas is Biased



# Train bias out

# Advanced Idea: Adversarial Learning

---

## Achieving Fairness through Adversarial Learning: an Application to Recidivism Prediction

---

**Christina Wadsworth**

Stanford University

Stanford, CA

[cwads@cs.stanford.edu](mailto:cwads@cs.stanford.edu)

**Francesca Vera**

Stanford University

Stanford, CA

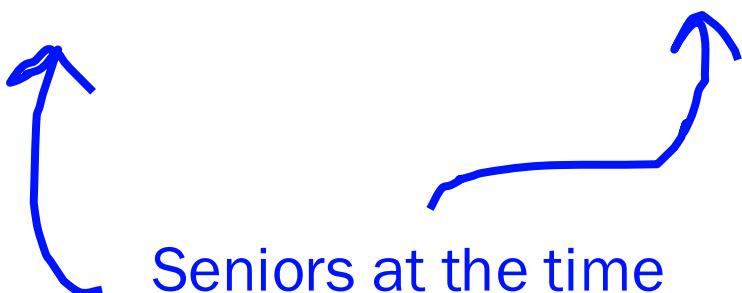
[fvera@cs.stanford.edu](mailto:fvera@cs.stanford.edu)

**Chris Piech**

Stanford University

Stanford, CA

[piech@cs.stanford.edu](mailto:piech@cs.stanford.edu)



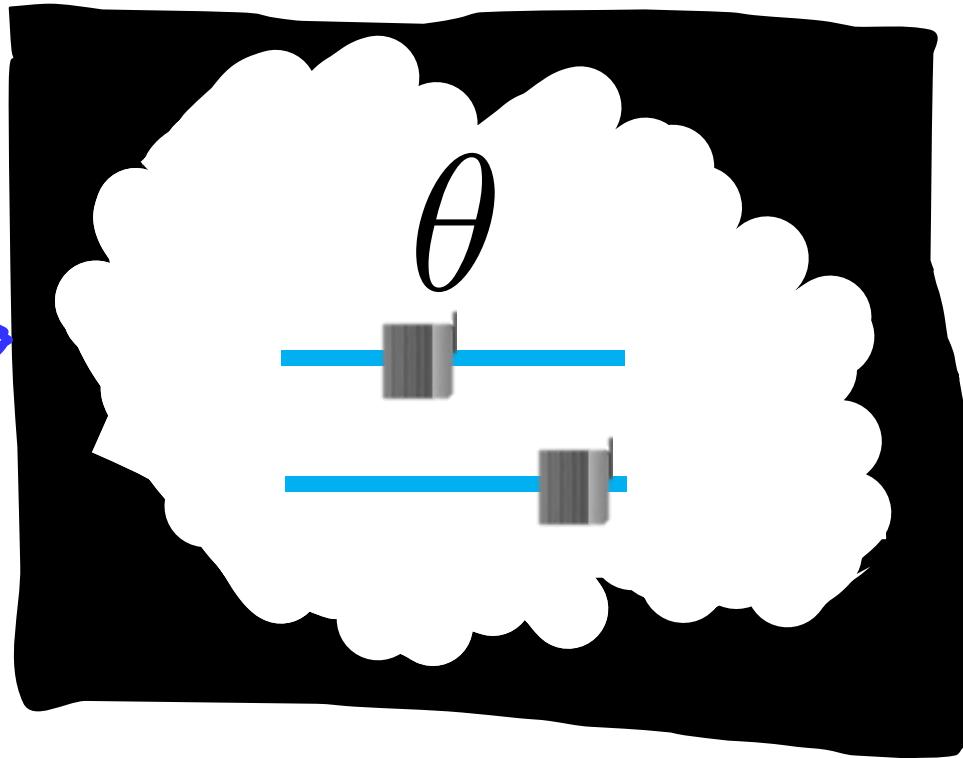
220+ citations

Seniors at the time  
they wrote it

# COMPAS: Predicting “Recidivism”

**X**

Data about an inmate:  
Their zip code,  
past crimes, etc



$$\hat{y} = 0$$

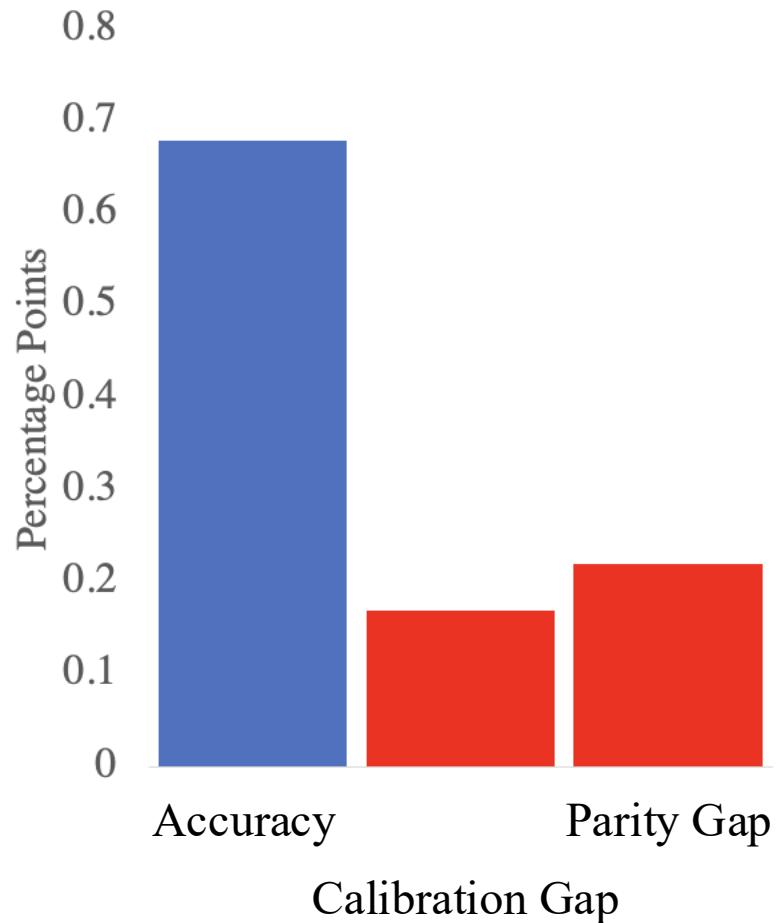
Will they commit a crime again

Was in use in California and Florida

# COMPAS: Biased Against Black Inmates

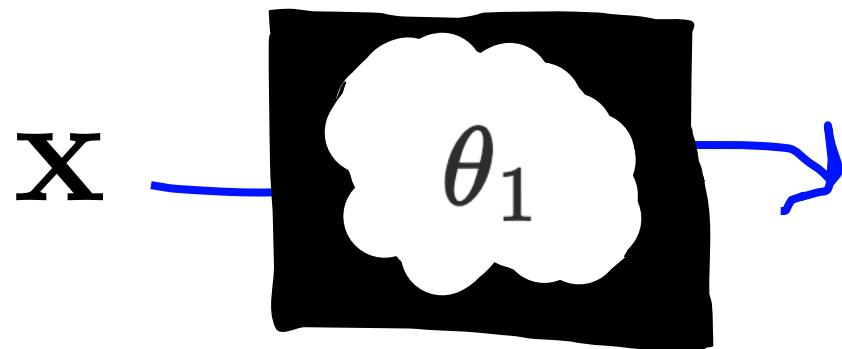
---

## Before: Compas is Biased

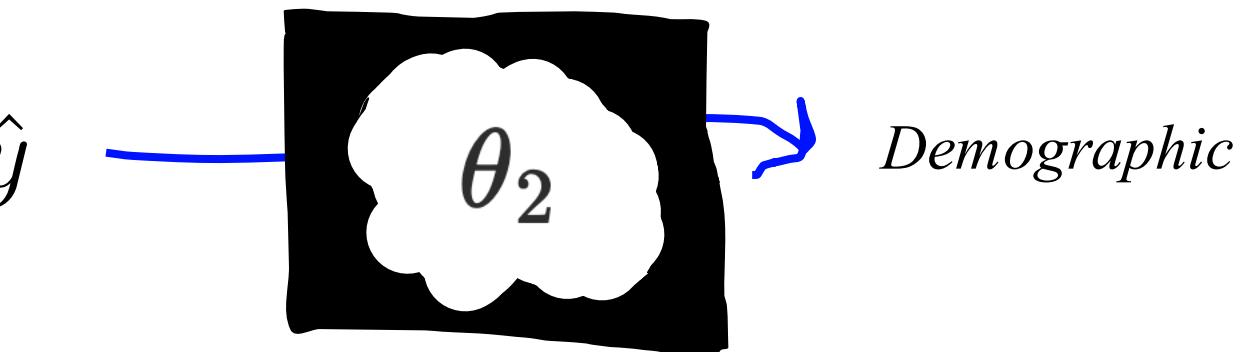


# Can We Train Out Bias?

Model 1: Prediction



Model 1: Extract Demographic



*Model 1 should  
be accurate*

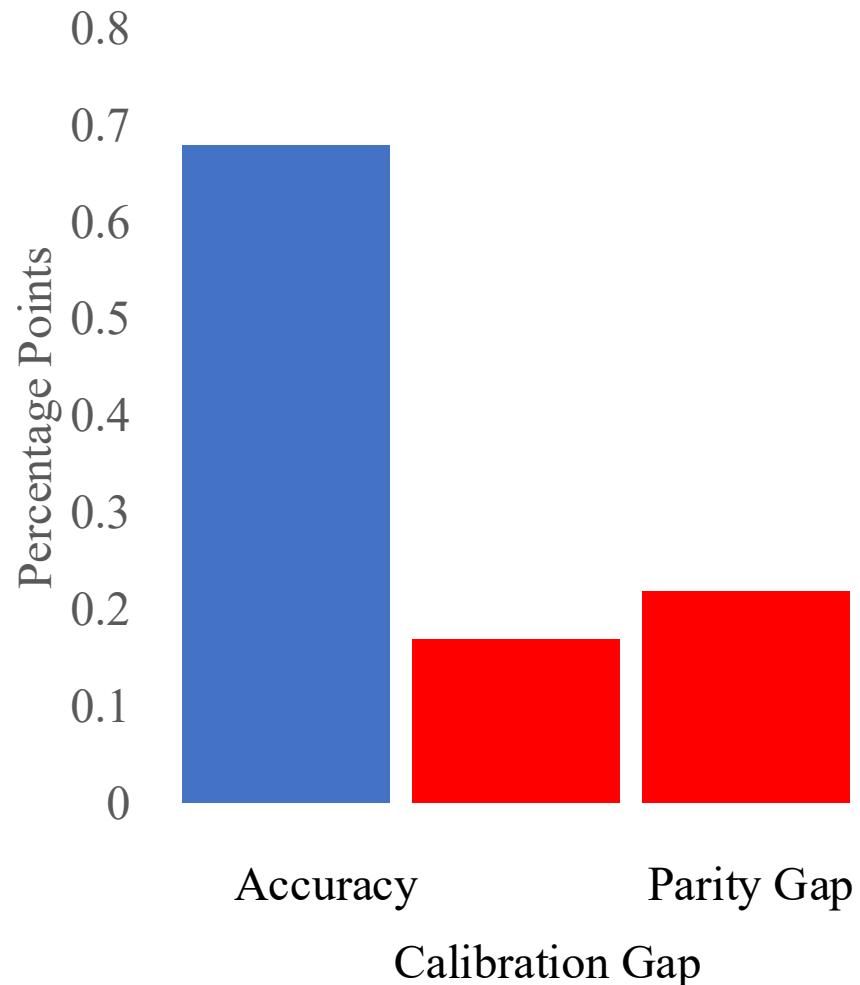
*Model 2 should  
be inaccurate*

$$\theta_1, \theta_2 = \operatorname{argmax}_{\theta_1, \theta_2} L_1(\theta_1) - L_2(\theta_2)$$

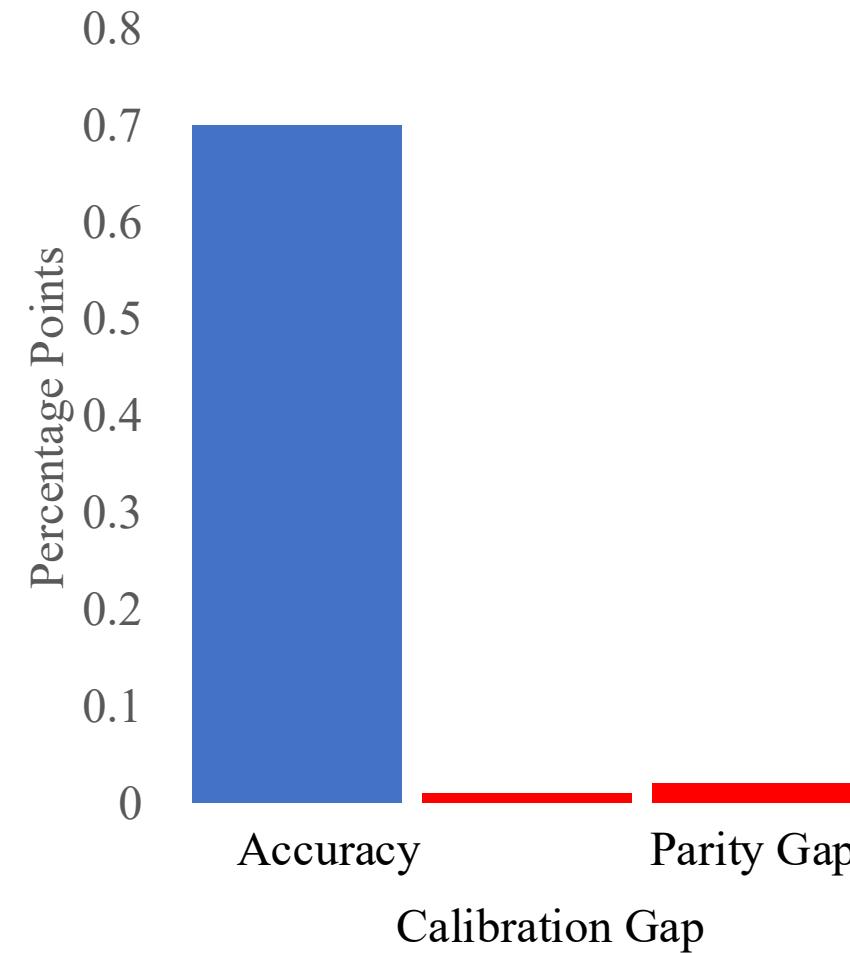
\*note in the paper these were neural nets

# Can We Train Out Bias?

**Before: Compas is Biased**



**After: Gaps are reduced**



## Their Conclusion

---

DON'T USE BLACK  
BOX ALGORITHMS TO  
MAKE RECIDIVISM  
PREDICTIONS

# Machine Learning in CS109

Great Idea

Core  
Algorithms

Theory

