Maximum A Posteriori

Noah Arthurs
CS109, Stanford University

Problem Set 6

Naïve Bayes: Need today's lecture

Logistic Regression: Need Wednesday

Neural Net (EC): Need Friday

Mini-Project (EC): Whatever you'd like! Start early!

Review

Review: Maximum Likelihood Algorithm

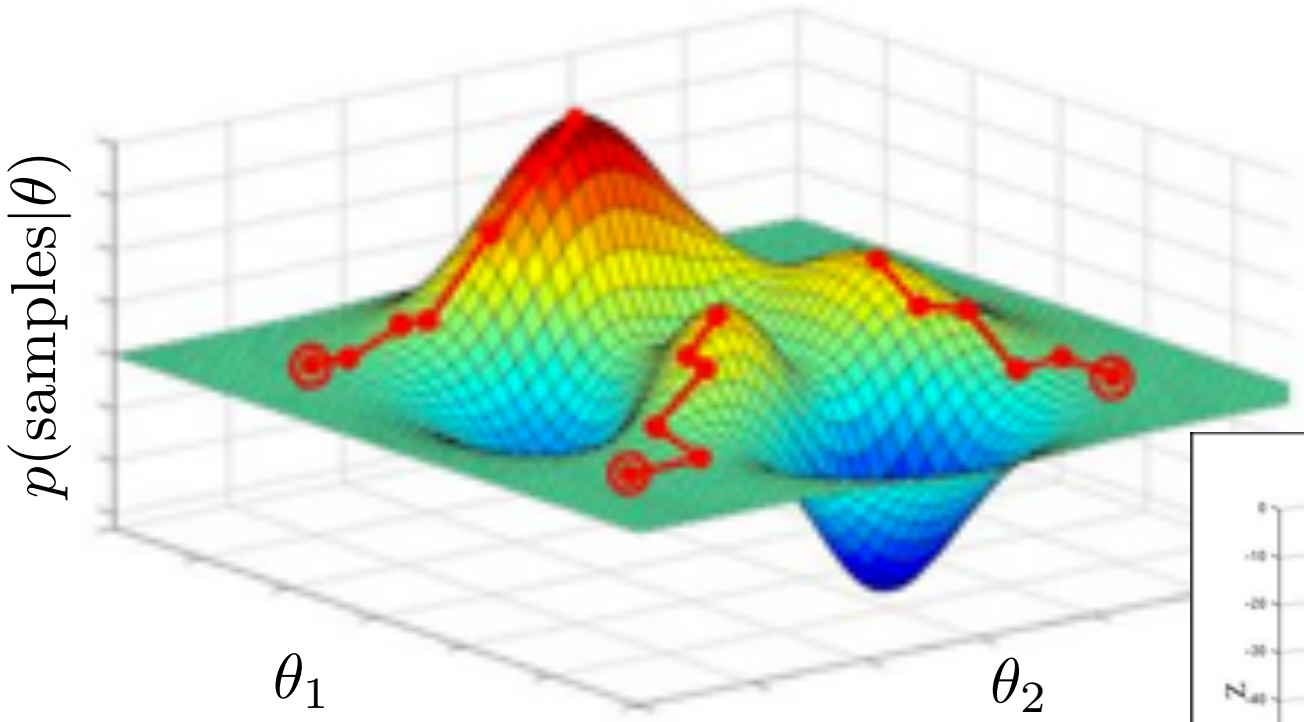
1. Decide on a model for the likelihood of your samples. This is often using a PMF or PDF.

2. Write out the log likelihood function.

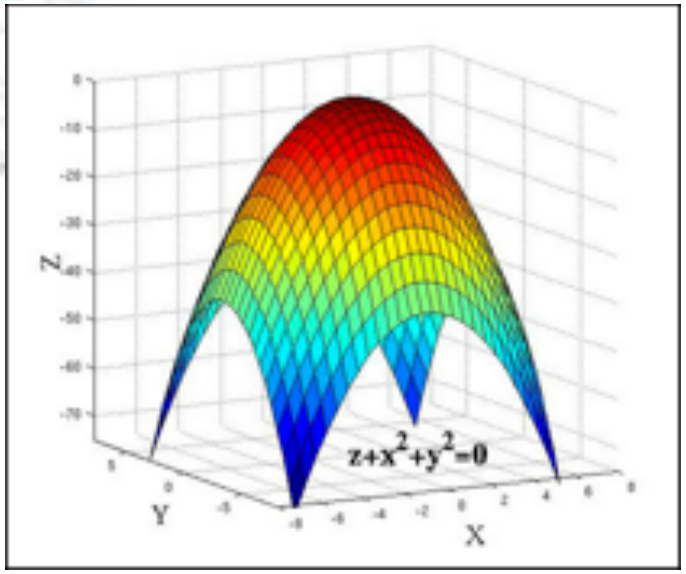
3. State that the optimal parameters are the argmax of the log likelihood function.

4. Use an optimization algorithm to calculate argmax

Gradient Ascent



Especially good if function is convex



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

Gradient Ascent

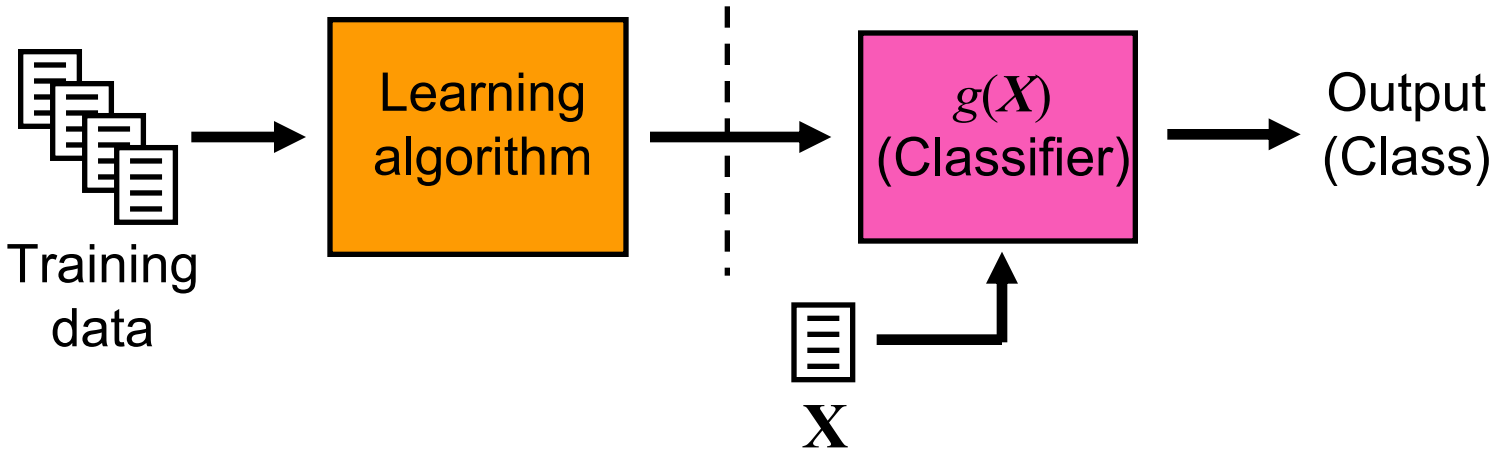
Repeat many times

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} + \eta \cdot \frac{\partial LL(\theta^{\text{old}})}{\partial \theta_j^{\text{old}}}$$

This is some **profound** life philosophy

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

The Machine Learning Process



- Training data: set of *N* pre-classified data instances
  - *N* training pairs: $(\boldsymbol{x}^{(1)}, y^{(1)})$, $(\boldsymbol{x}^{(2)}, y^{(2)})$, …, $(\boldsymbol{x}^{(n)}, y^{(n)})$
    - Use superscripts to denote *i*-th training instance
- Learning algorithm: method for determining $g(\boldsymbol{X})$
  - Given a new input observation of $\boldsymbol{x} = x_1, x_2, \ldots, x_m$
  - Use $g(\boldsymbol{x})$ to compute a corresponding output (prediction)

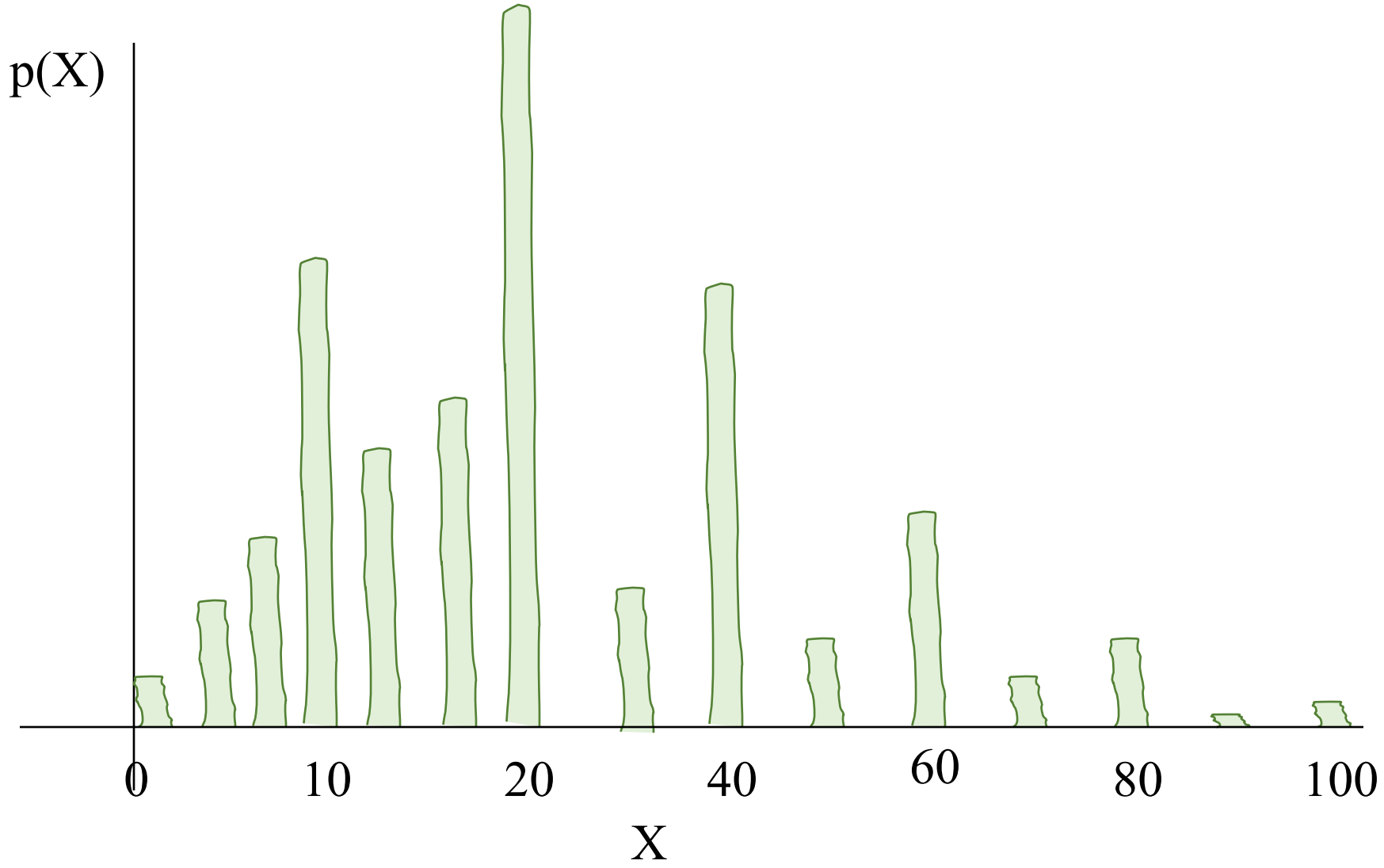
Two Envelopes

- I have two envelopes, will allow you to have one
  - One contains $X, the other contains $2X
  - Select an envelope
    - Open it!
  - Now, would you like to switch for other envelope?
  - To help you decide, compute E[$ in other envelope]
    - Let Y = $ in envelope you selected

$$E[\$ \text{ in other envelope}] = \frac{1}{2} \cdot \frac{Y}{2} + \frac{1}{2} \cdot 2Y = \frac{5}{4} Y$$

  - Before opening envelope, think either equally good
  - So, what happened by opening envelope?
    - And does it really make sense to switch?

Subjectivity of Probability



Maximum A Posteriori

data: $x^{(1)}, \ldots, x^{(n)}$ $\qquad$ $\hat{\theta}_{MAP} = \underset{\theta}{\text{argmax}}\ f(\theta | x^{(1)}, \ldots, x^{(n)})$

$$\hat{\theta}_{MAP} = \underset{\theta}{\text{argmax}}\ \frac{g(\theta) f(x^{(1)}, x^{(2)}, \ldots, x^{(n)} | \theta)}{h(x^{(1)}, x^{(2)}, \ldots x^{(n)})}$$

$$= \underset{\theta}{\text{argmax}}\ \frac{g(\theta) \prod_{i=1}^{n} f(x^{(i)} | \theta)}{h(x^{(1)}, x^{(2)}, \ldots, x^{(n)})}$$

$$= \underset{\theta}{\text{argmax}}\ g(\theta) \prod_{i=1}^{n} f(x^{(i)} | \theta)$$

$$= \underset{\theta}{\text{argmax}}\ \left(\log(g(\theta)) + \sum_{i=1}^{n} \log(f(x^{(i)} | \theta)) \right)$$

Maximum A Posteriori

Estimated parameter

Log prior

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} \left(\log(g(\theta)) + \sum_{i=1}^{n} \log(f(x^{(i)}|\theta)) \right)$$

Chose the value of theta that maximizes:

Sum of log likelihood

MLE vs MAP

Data: $x^{(1)}, \ldots, x^{(n)}$

Maximum Likelihood Estimation

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\ f(x^{(1)}, \ldots, x^{(n)}|\theta)$$

$$= \underset{\theta}{\operatorname{argmax}} \left(\sum_i \log f(x^{(i)}|\theta) \right)$$

Maximum A Posteriori

$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}}\ f(\theta|x^{(1)}, \ldots, x^{(n)})$$

$$= \underset{\theta}{\operatorname{argmax}} \left(\log(g(\theta)) + \sum_{i=1}^{n} \log(f(x^{(i)}|\theta)) \right)$$

$P(\theta \mid D)$ For Bernoulli

- Prior: θ ~ Beta(a, b); data = {n heads, m tails}
- Estimate p, *aka* θ

$$\hat{\theta}_{MAP} = \underset{\theta}{\text{argmax}}\ f(\theta|\text{data}) \qquad = \underset{\theta}{\text{argmax}}\ f(\text{data}|\theta)g(\theta)$$

This is the beta PDF

$$= \underset{\theta}{\text{argmax}}\ \log g(\theta) + \log f(\text{data}|\theta)$$

Product of thetas and (1-theta)s

$$= \underset{\theta}{\text{argmax}} \log \left[\frac{1}{\beta}\theta^{a-1}(1-\theta)^{b-1}\right] + n \log f(\text{heads}|\theta) + m \log f(\text{tails}|\theta)$$

$$= \underset{\theta}{\text{argmax}} \log \frac{1}{\beta} + (a-1)\log\theta + (b-1)\log(1-\theta) + n\log\theta + m\log(1-\theta)$$

$$= \underset{\theta}{\text{argmax}} (a-1+n)\log\theta + (b-1+m)\log(1-\theta)$$

$P(\theta \mid D)$ For Bernoulli

- Prior: θ ~ Beta(a, b); D = {n heads, m tails}
- Estimate p, *aka* θ

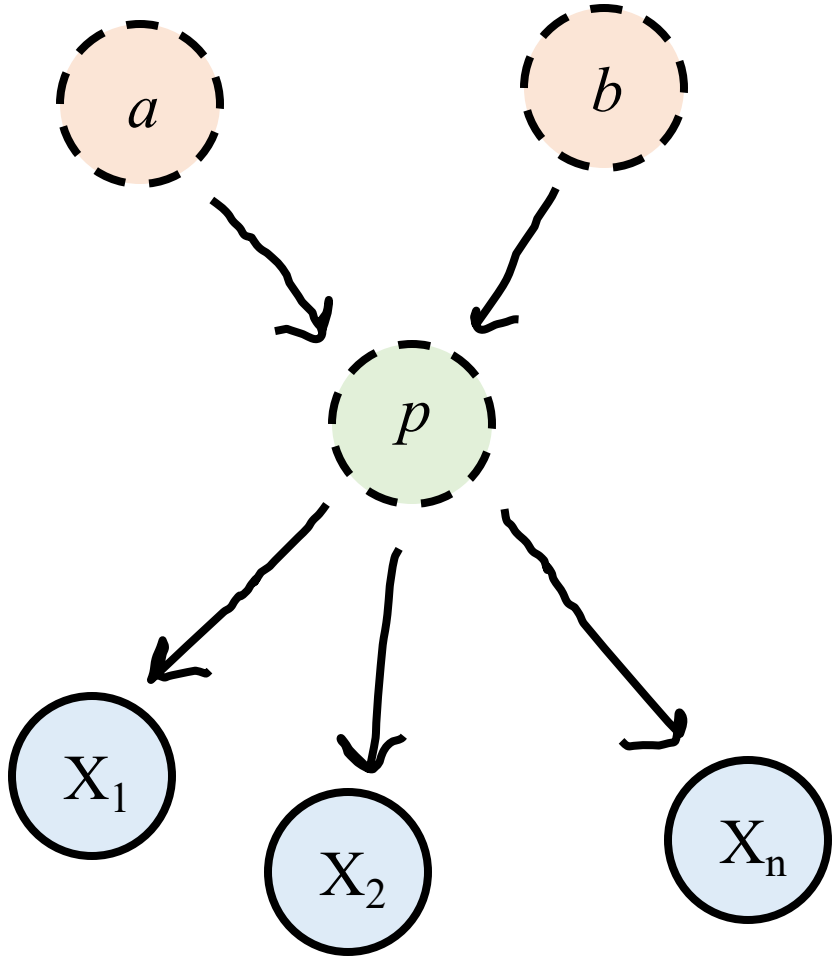
$$\hat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}}\ f(\theta|\text{data})$$

$$= \underset{\theta}{\operatorname{argmax}} (a-1+n)\log\theta + (b-1+m)\log(1-\theta)$$

$$= \frac{n+a-1}{n+m+a+b-2}$$

That's the mode of the updated beta

Hyper Parameters



Hyperparameter

a, b are fixed

Prior

$p \sim \text{Beta}(a, b)$

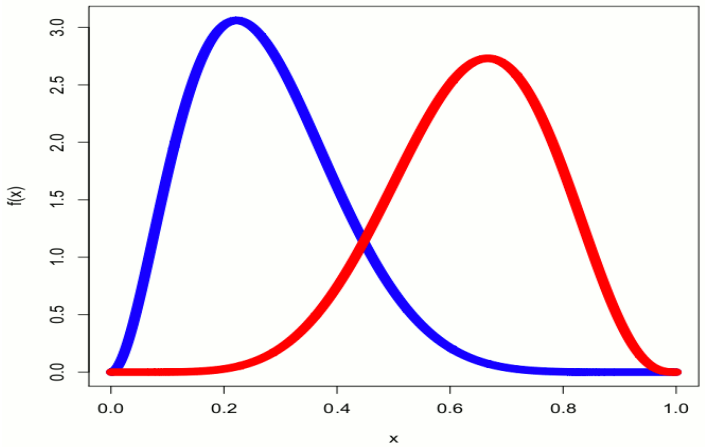
Data distribution

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

MAP will estimate the most likely value of p for this model

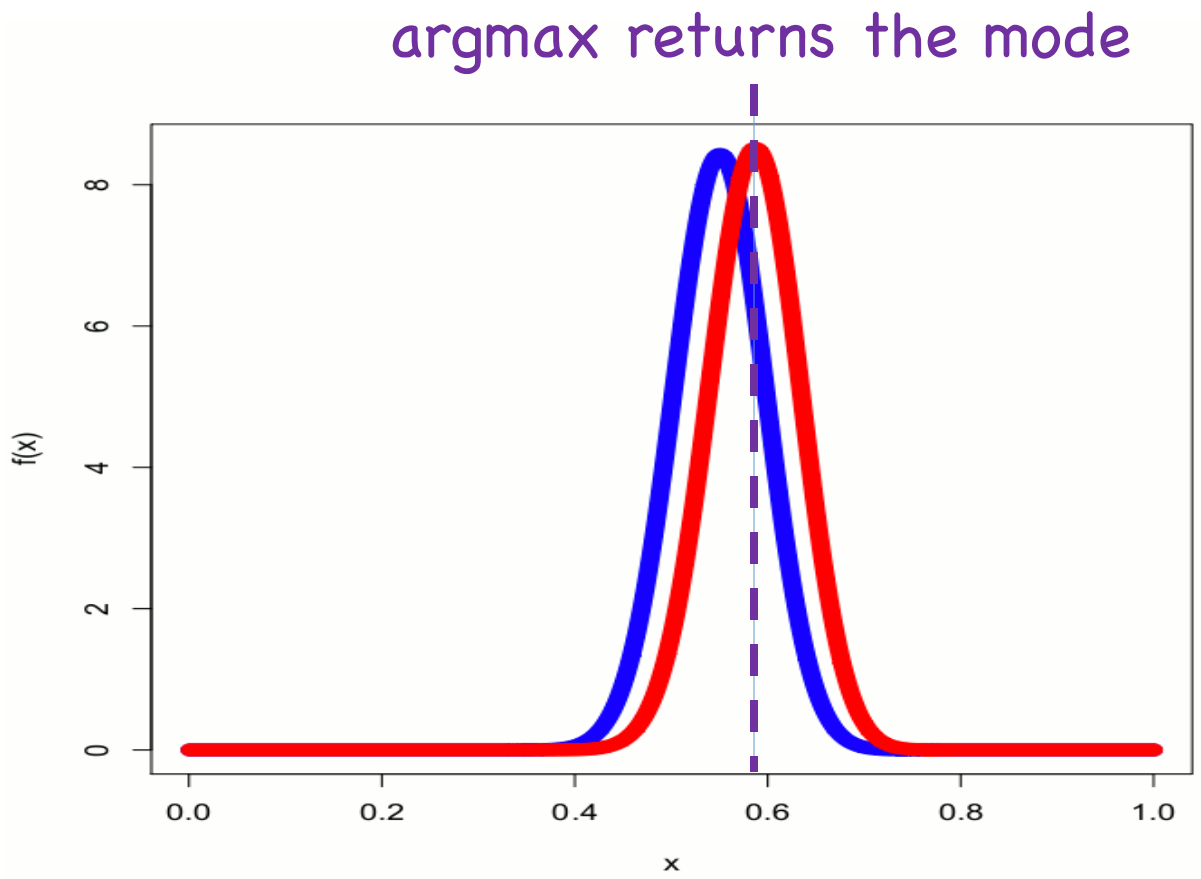
Where'd Ya Get Them P(θ)?

- θ is the probability a coin turns up heads
- Model θ with 2 different priors:
  - $P_1(\theta)$ is Beta(3,8) (blue)
  - $P_2(\theta)$ is Beta(7,4) (red)
- They look pretty different!



- Now flip 100 coins; get 58 heads and 42 tails
  - What do posteriors look like?

It's Like Having Twins



- As long as we collect enough data, posteriors will converge to the true value!

Conjugate Distributions Without Tears

- Just for review…
- Have coin with unknown probability θ of heads
  - Our prior (subjective) belief is that $\theta \sim \text{Beta}(a, b)$
  - Now flip coin $k = n + m$ times, getting n heads, m tails
  - Posterior density: $(\theta \mid n \text{ heads}, m \text{ tails}) \sim \text{Beta}(a+n, b+m)$
    - Beta is conjugate for Bernoulli, Binomial, Geometric, and Negative Binomial
  - a and b are called "hyperparameters"
    - Saw $(a + b - 2)$ imaginary trials, of those $(a - 1)$ are "successes"
  - For a coin you never flipped before, use Beta(x, x) to denote you think coin likely to be fair
    - How strongly you feel coin is fair is a function of x

End Review

Gonna Need Priors

Parameter	Distribution for Parameter
Bernoulli p	Beta
Binomial p	Beta
Poisson λ	Gamma
Exponential λ	Gamma
Multinomial p_i	Dirichlet
Normal μ	Normal
Normal σ^2	Inverse Gamma

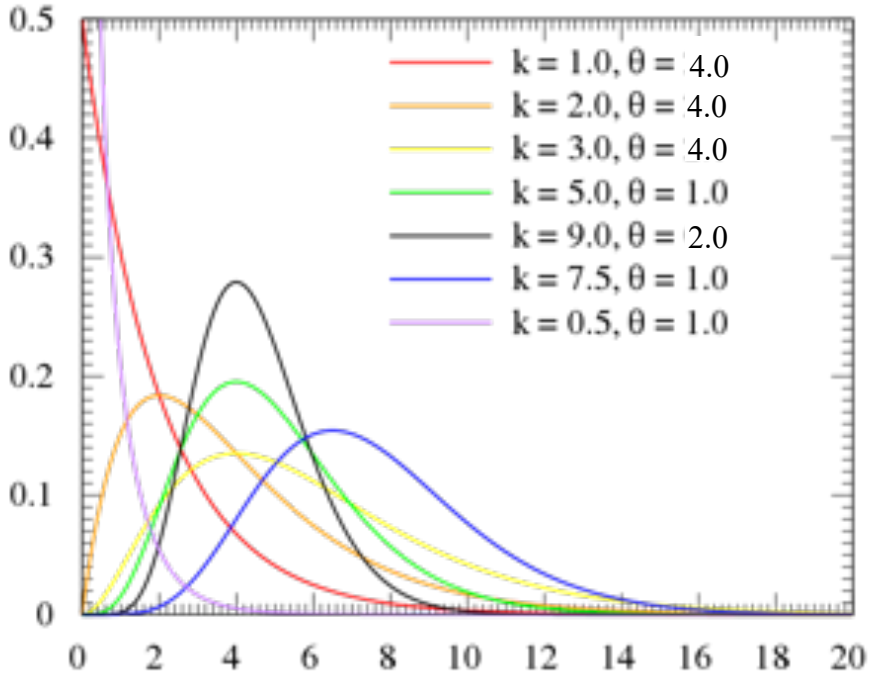
Know

Familiarity

Not necessary but good to know of

Good Times with Gamma

- Gamma(k, θ) distribution
  - Conjugate for Poisson Rate
    - Also conjugate for Exponential, but we won't delve into that
  - Intuitive understanding of hyperparameters:
    - Saw k total imaginary events during θ prior time periods



Good Times with Gamma

- Gamma(k, θ) distribution
  - Conjugate for Poisson Rate
    - Also conjugate for Exponential, but we won't delve into that
  - Intuitive understanding of hyperparameters:
    - Saw k total imaginary events during θ prior time periods
  - Updating with observations
    - After observing *n* events during next *t* time periods...
    - ... posterior distribution is Gamma(k + *n*, θ + *t*)
    - …MAP estimator for Poisson with Gamma prior is (k+n)/(θ + *t*)
    - Example: Prior for rate is Gamma(10, 5)
    - Saw 10 events in 5 time periods. Like observing at rate = 2
    - Now see 11 events in next 2 time periods → Gamma(21, 7)
    - MAP rate = 3

Reviving an Old Story Line

The Multinomial Distribution $\mathrm{Mult}(p_1, \ldots, p_k)$

$$p(x_1, \ldots, x_k) = \frac{n!}{x_1! \ldots x_k!} p_1^{x_1} \ldots p_k^{x_k}$$

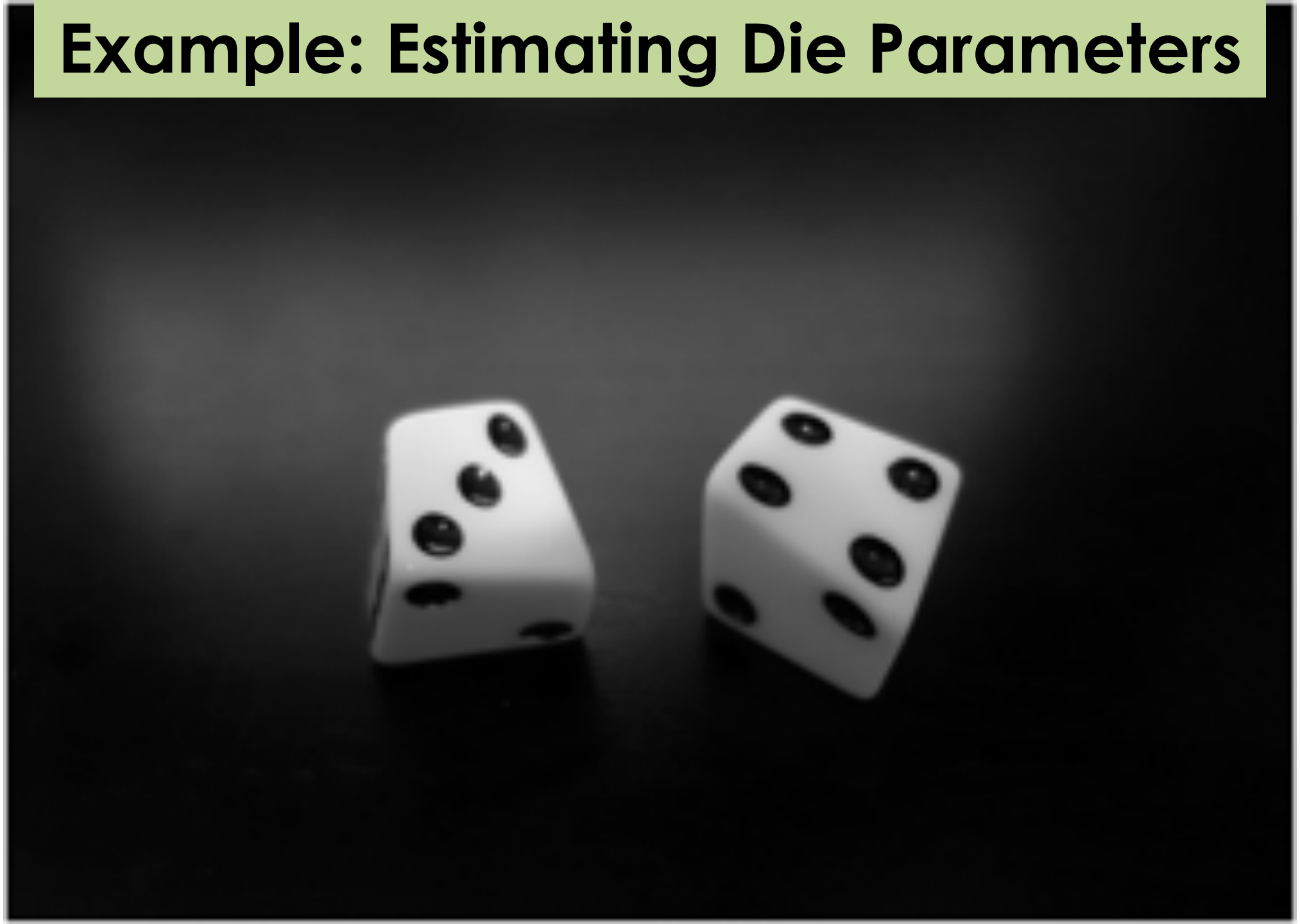
Multinomial is Multiple Times the Fun

- Dirichlet(a_1, a_2, ..., a_m) distribution
  - Conjugate for Multinomial
    - Dirichlet generalizes Beta in same way Multinomial generalizes Bernoulli

$$f(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m) = K \prod_{i=1}^{m} x_i^{a_i - 1}$$

  - Intuitive understanding of hyperparameters:
    - Saw $\sum_{i=1}^{m} a_i - m$ imaginary trials, with (a_i – 1) of outcome i
  - Updating to get the posterior distribution
    - After observing $n_1 + n_2 + \ldots + n_m$, new trials with n_i of outcome i...
    - ... posterior distribution is Dirichlet($a_1 + n_1$, $a_2 + n_2$, ..., $a_m + n_m$)

Example: Estimating Die Parameters



Your Happy Laplace

- Recall example of 6-sides die rolls:
  - X ~ Multinomial(p_1, p_2, p_3, p_4, p_5, p_6)
  - Roll n = 12 times
  - Result: 3 ones, 2 twos, 0 threes, 3 fours, 1 fives, 3 sixes
    - MLE: p_1=3/12, p_2=2/12, p_3=0/12, p_4=3/12, p_5=1/12, p_6=3/12
  - Dirichlet prior allows us to pretend we saw each outcome k times before. MAP estimate: $p_i = \frac{X_i + k}{n + mk}$
    - Laplace's "law of succession": idea above with k = 1
    - Laplace estimate: $p_i = \frac{X_i + 1}{n + m}$
    - Laplace: p_1=4/18, p_2=3/18, p_3=1/18, p_4=4/18, p_5=2/18, p_6=4/18
    - No longer have 0 probability of rolling a three!

MLE for Multinomial

MLE estimate of the probability of outcome i

number of observed outcomes of type i

$$p_i = \frac{n_i}{n}$$

number of observations

θ_i is p_i
For a multinomial

MAP for Multinomial, Laplace Prior

MAP estimate of the probability of outcome i

number of observed outcomes of type i

$$p_i = \frac{n_i + 1}{n + m}$$

number of observations

number of outcome types

θ_i is p_i
For a multinomial

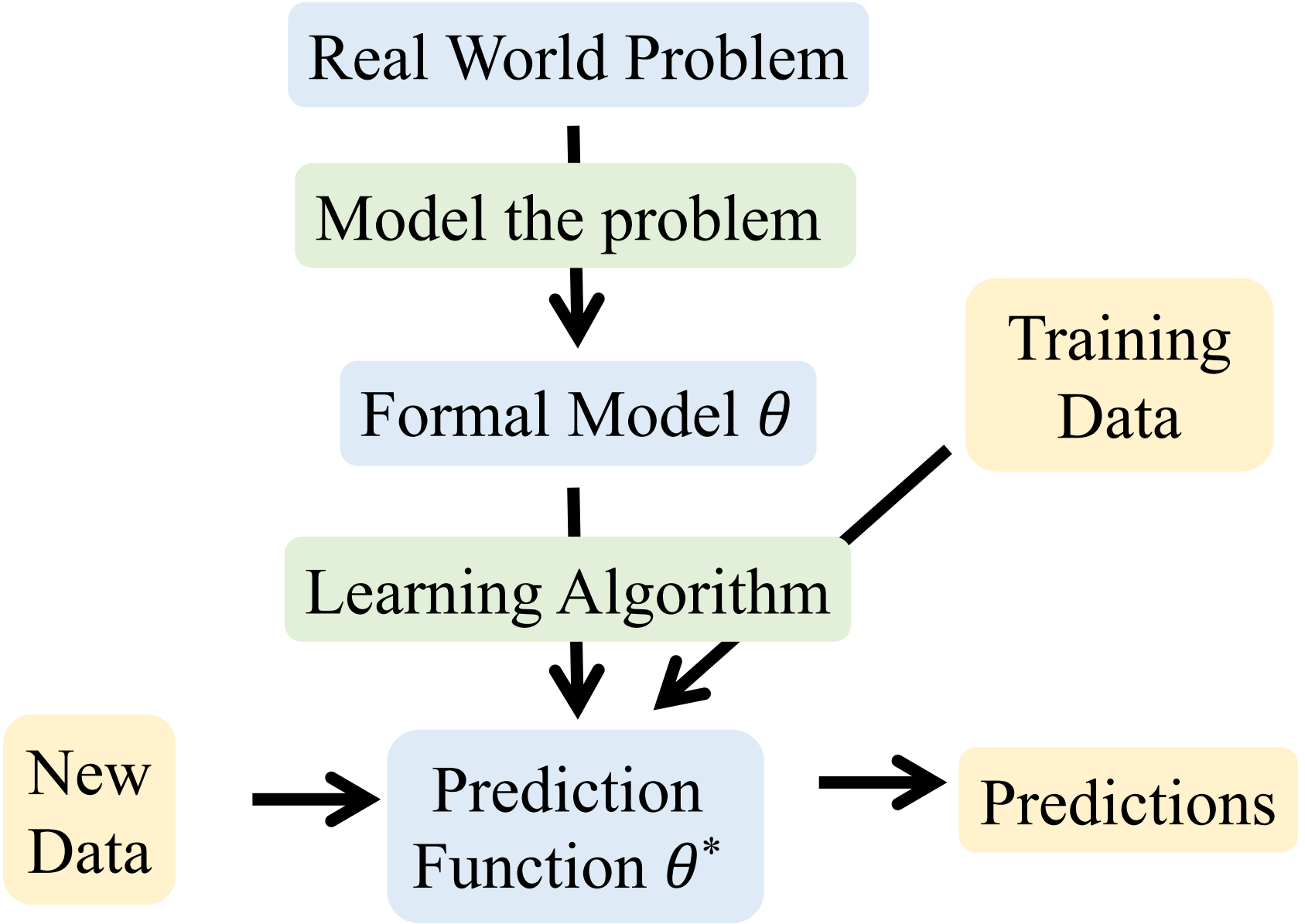
Stretch!

SCPD Code:
Purple Car

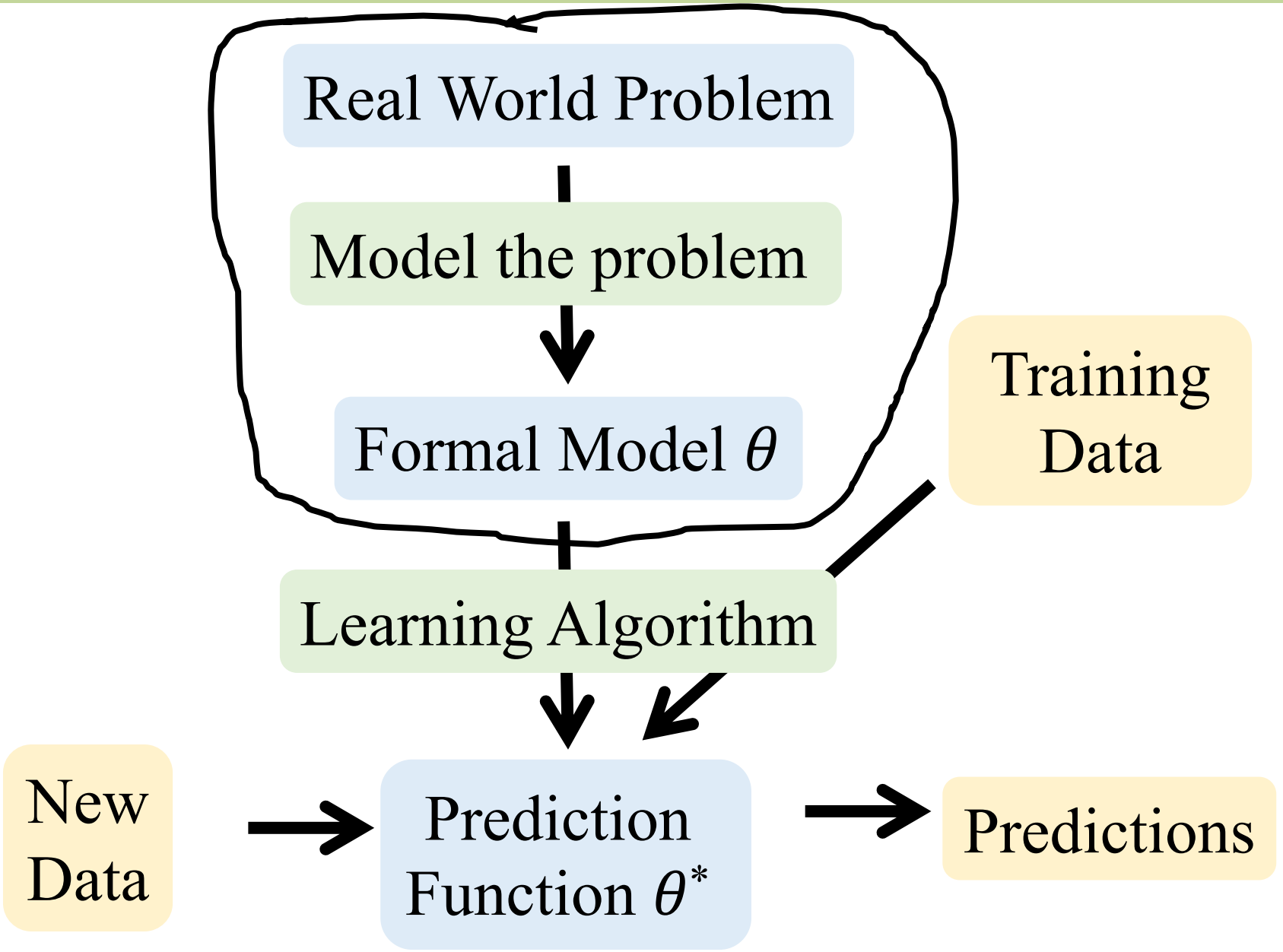
Naïve Bayes

Noah Arthurs
CS109, Stanford University

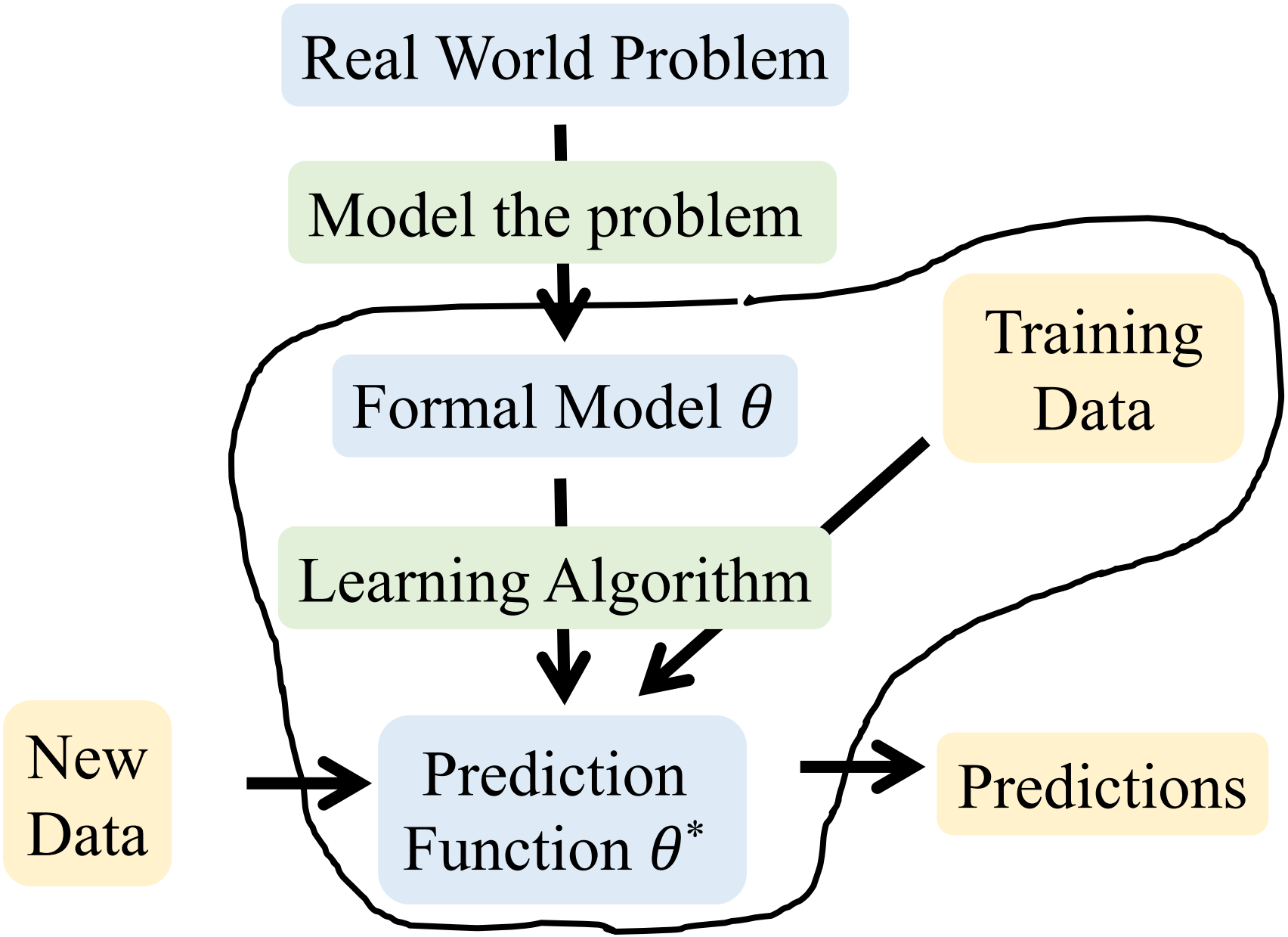
Supervised Learning



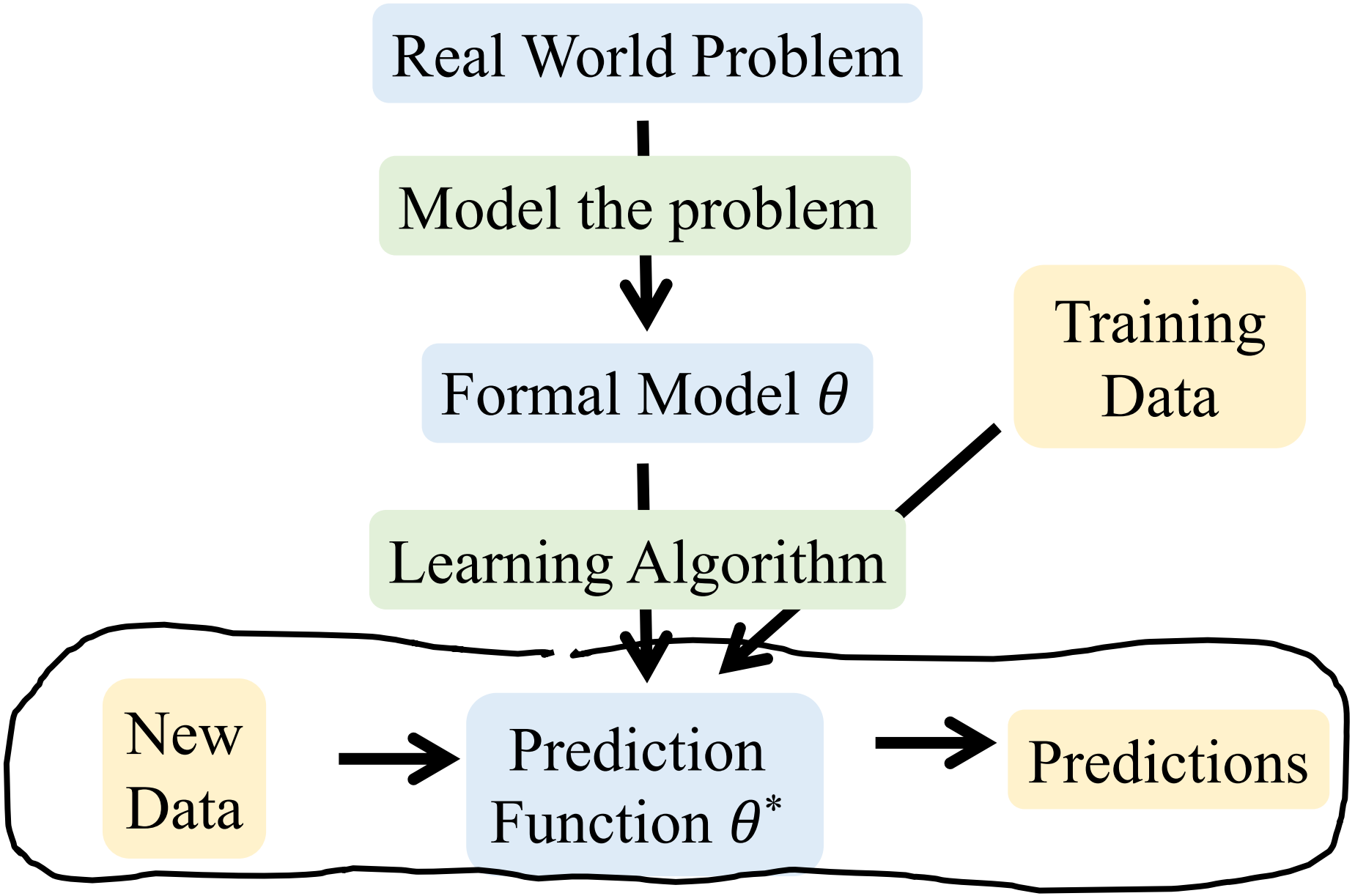
Modelling


Real World Problem
Model the problem
Formal Model θ
Training Data
Learning Algorithm
New Data
Prediction Function θ^*
Predictions

Training*


Real World Problem
Model the problem
Formal Model θ
Learning Algorithm
Training Data
New Data
Prediction Function θ^*
Predictions

Make Predictions*



Machine Learning: Formally

- Many different forms of "Machine Learning"
  - We focus on the problem of *prediction*
- Want to make a prediction based on observations
  - Vector $\mathbf{X}$ of *m* observed variables: $\mathbf{X} = [X_1 \ldots X_m]$
  - Based on observed $\mathbf{X}$, want to predict unseen variable Y
    - Y called "output feature/variable" (or the "dependent variable")
  - Seek to "learn" a function $g(\mathbf{X})$ to predict Y:
    - $\hat{Y} = g(\mathbf{X})$
    - When Y is discrete, prediction of Y is called "classification"
    - When Y is continuous, prediction of Y is called "regression"

Training Data

Training Data: assignments all random variables $\mathbf{X}$ and Y

Assume IID data:

n training datapoints

$$(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots (\mathbf{x}^{(n)}, y^{(n)})$$

$$m = |\mathbf{x}^{(i)}|$$

Each datapoint has m features and a single output

Regression

Regression: Predicting Real Numbers

	Opposing team ELO	Points in last game	…	At Home?	Output
					# Points
Game 1	84	105		1	120
Game 2	90	102		0	95
		⋮			⋮
Game n	74	120		0	115

Linear Regression

	Opposing team ELO	Points in last game		At Home?	Output
			…		# Points
Game 1	84	105		1	120
Game 2	90	102		0	95
		⋮			⋮
Game n	74	120		0	115

Linear Regression

X_1 = Opposing team ELO

X_2 = Points in last game

X_3 = Curry playing?

X_4 = Playing at home?

Y = Warriors points

Linear Regression

Y = Warriors points

$$\hat{Y} = \theta_1 X_1 + \theta_2 X_2 + \ldots \theta_{n-1} X_{n-1} + \theta_n 1$$
$$= \theta^T \mathbf{X}$$

X_1 = Opposing team ELO — $\theta_1 = -2.3$

X_2 = Points in last game — $\theta_2 = +1.2$

X_3 = Curry playing? — $\theta_3 = +10.2$

X_4 = Playing at home? — $\theta_4 = +3.3$

$X_5 = 1$ — $\theta_5 = +95.4$

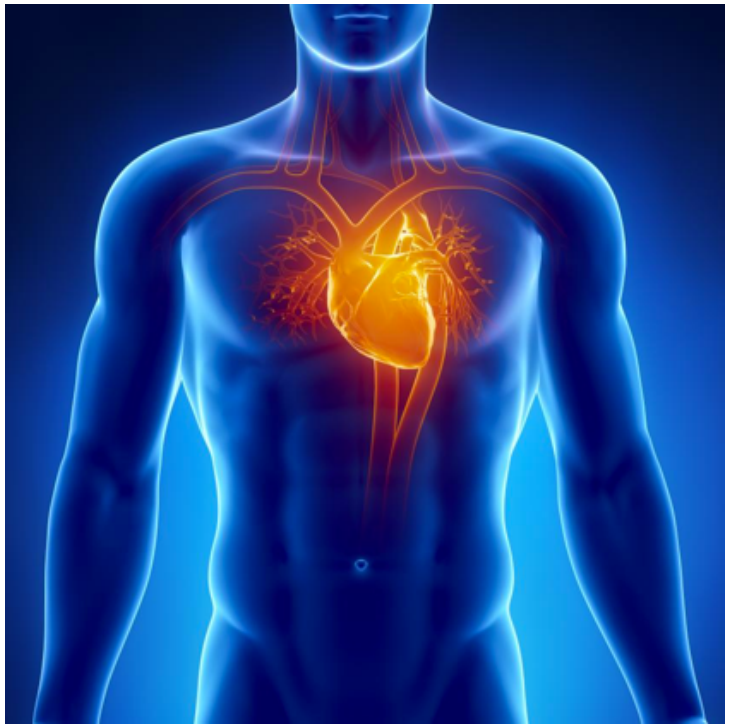
Classification

Classification is Building a Harry Potter Hat

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

Example Datasets

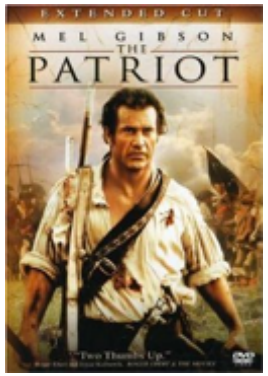
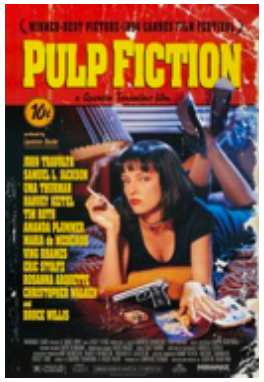
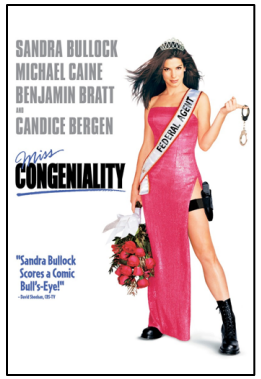
Heart



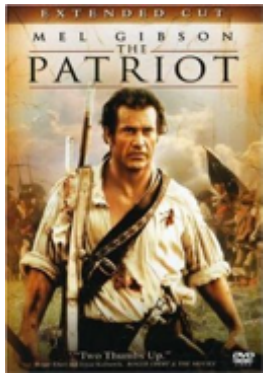
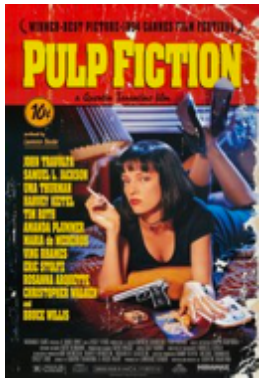
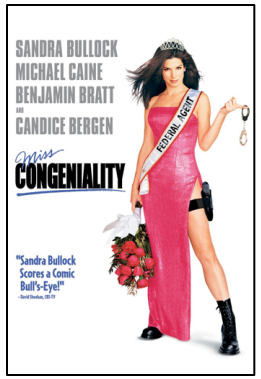
Ancestry

Netflix

Target Movie "Like" Classification

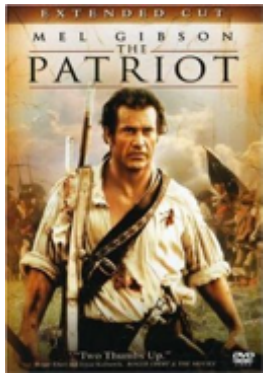
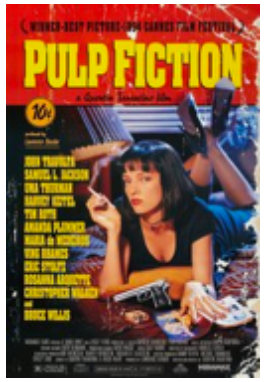
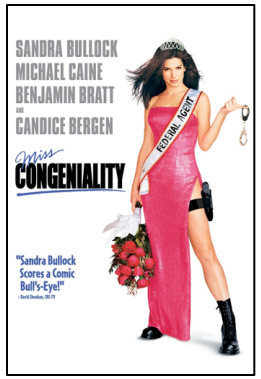
	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	0
			⋮		⋮
User n	0	0		1	1

Single Instance

	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	0
	⋮				⋮
User n	0	0		1	1

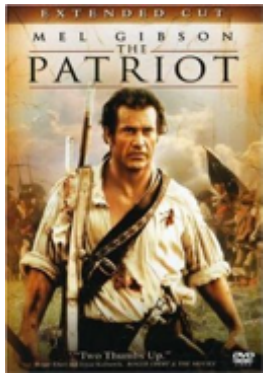
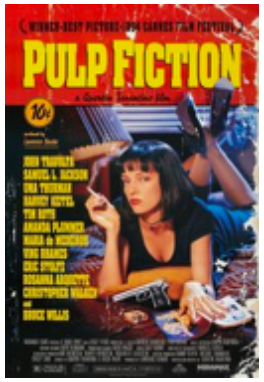
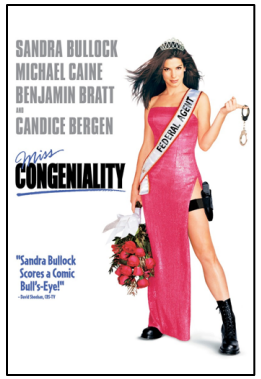
$$(\mathbf{x}^{(i)}, y^{(i)}) \text{ such that } 1 \leq i \leq n$$

Feature Vector

	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	0
	⋮				⋮
User n	0	0		1	1

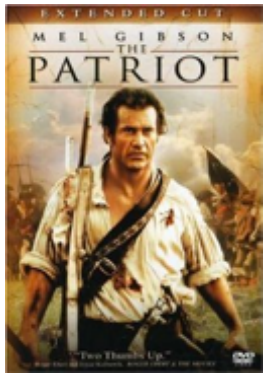
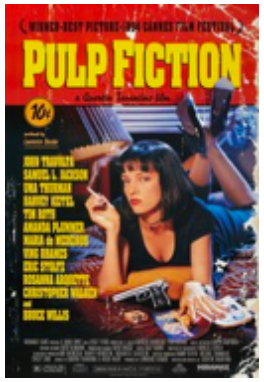
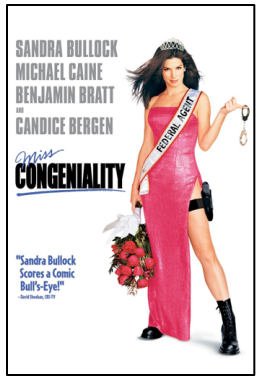
$(\mathbf{x}^{(i)}, y^{(i)})$ such that $1 \leq i \leq n$

Output Value

	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	0
⋮					⋮
User n	0	0		1	1

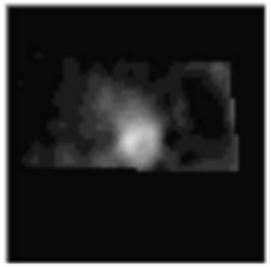
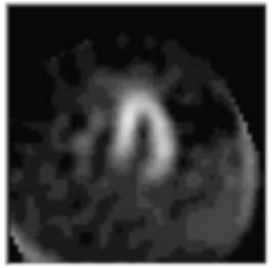
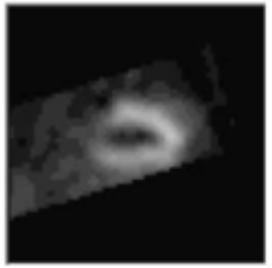
$$(\mathbf{x}^{(i)}, y^{(i)}) \text{ such that } 1 \leq i \leq n$$

Single Feature Value

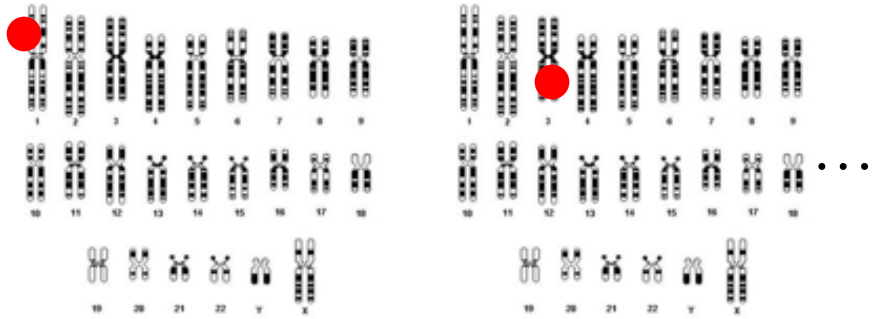
	Movie 1	Movie 2	…	Movie m	Output
					
User 1	1	0		1	1
User 2	1	1		0	0
	⋮				⋮
User n	0	0		1	1

In general: $\mathbf{x}_j^{(i)}$ In this case: $\mathbf{x}_m^{(2)}$

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

Ancestry Classifier

	SNP 1	SNP 2	⋯	SNP m	Output
					
User 1	1	0		1	0
User 2	0	0		1	1
			⋮		⋮
User n	1	1		0	1

Stretch!

Naïve Bayes

Noah Arthurs
CS109, Stanford University


NETFLIX
And Learn

Target Movie "Like" Classification

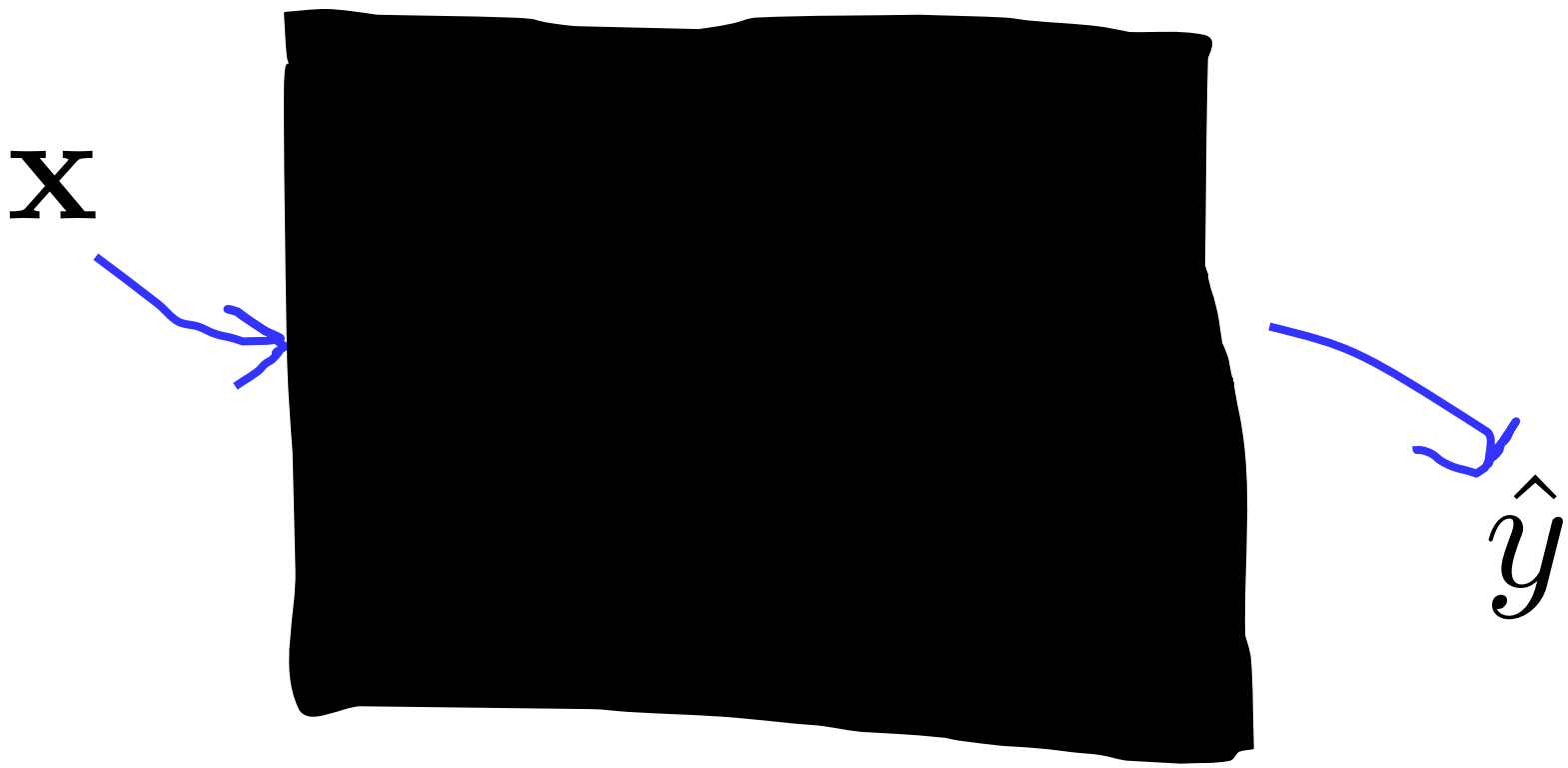
	Feature 1	Output
		
User 1	1	1
User 2	1	0
		⋮
User n	0	1

$$x_j^{(i)} \in \{0, 1\} \qquad y^{(i)} \in \{0, 1\}$$

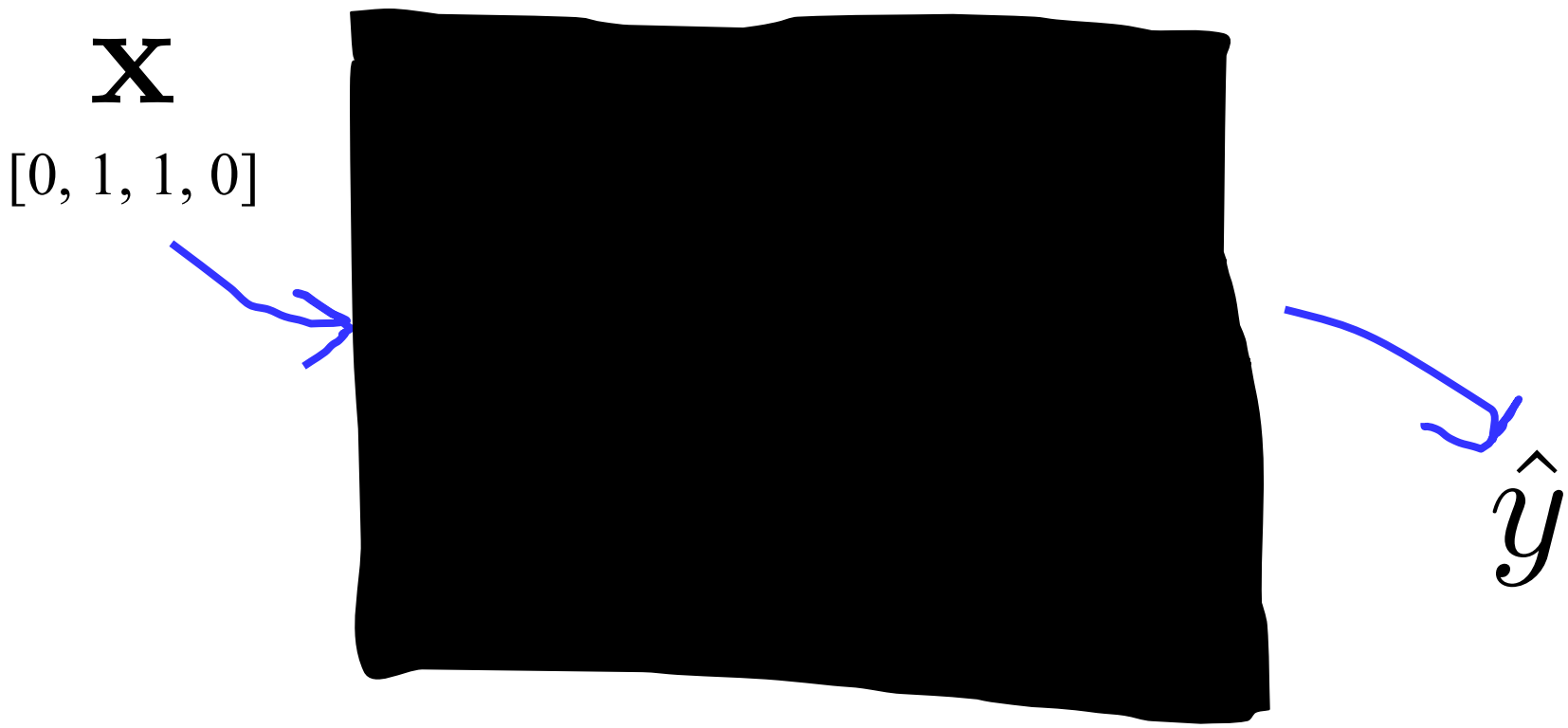
How could we predict the class label:
will the user like life is beautiful?

Fake Algorithm:
Brute Bayes Classifier

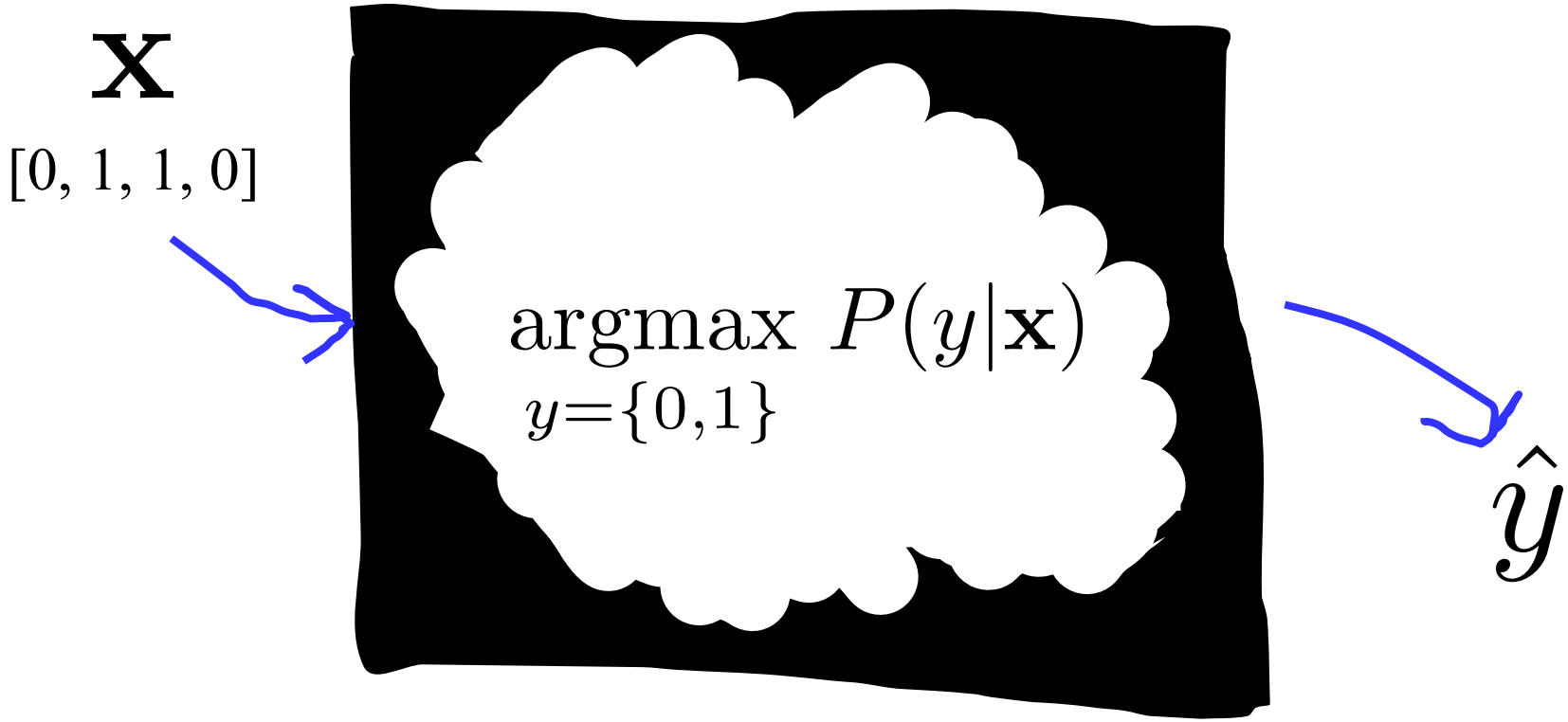
Brute Force Bayes



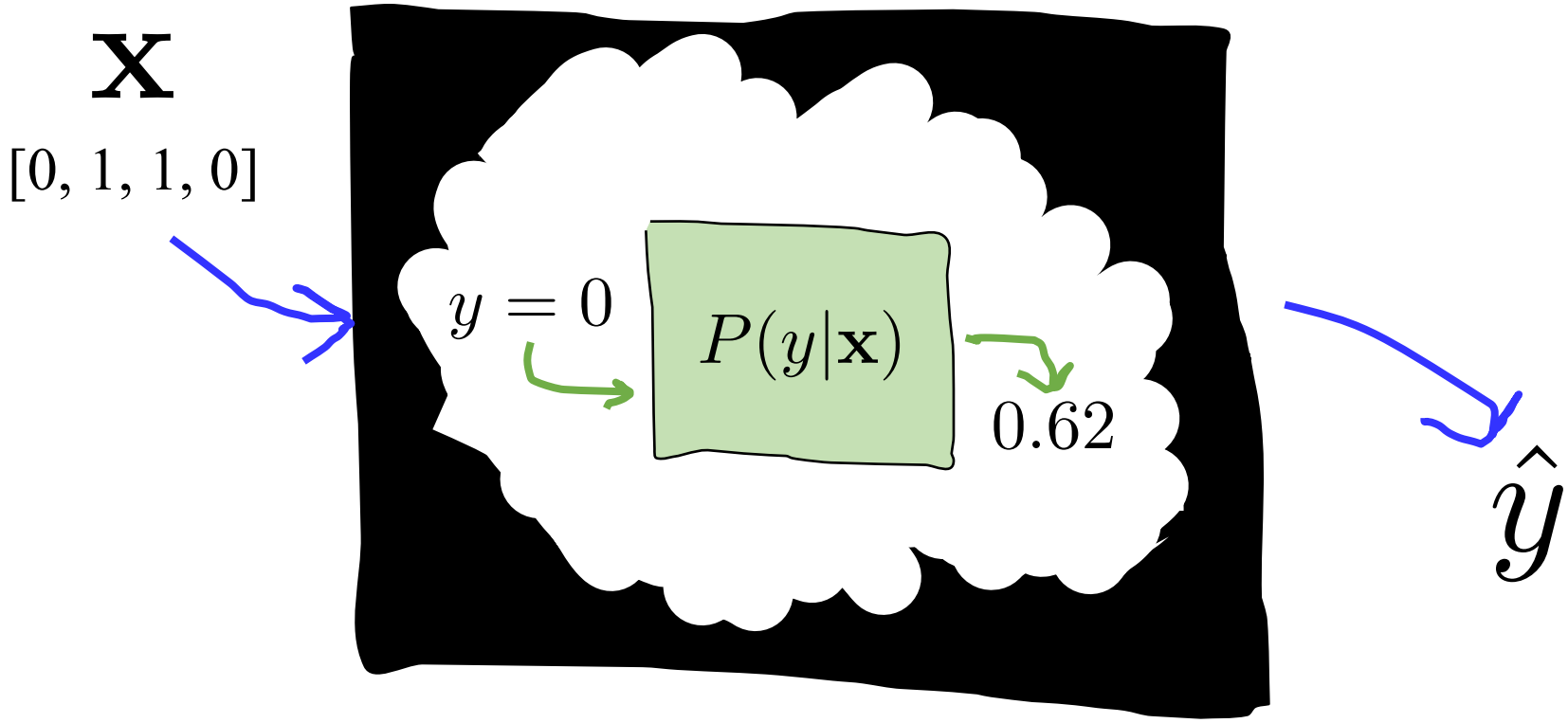
Brute Force Bayes



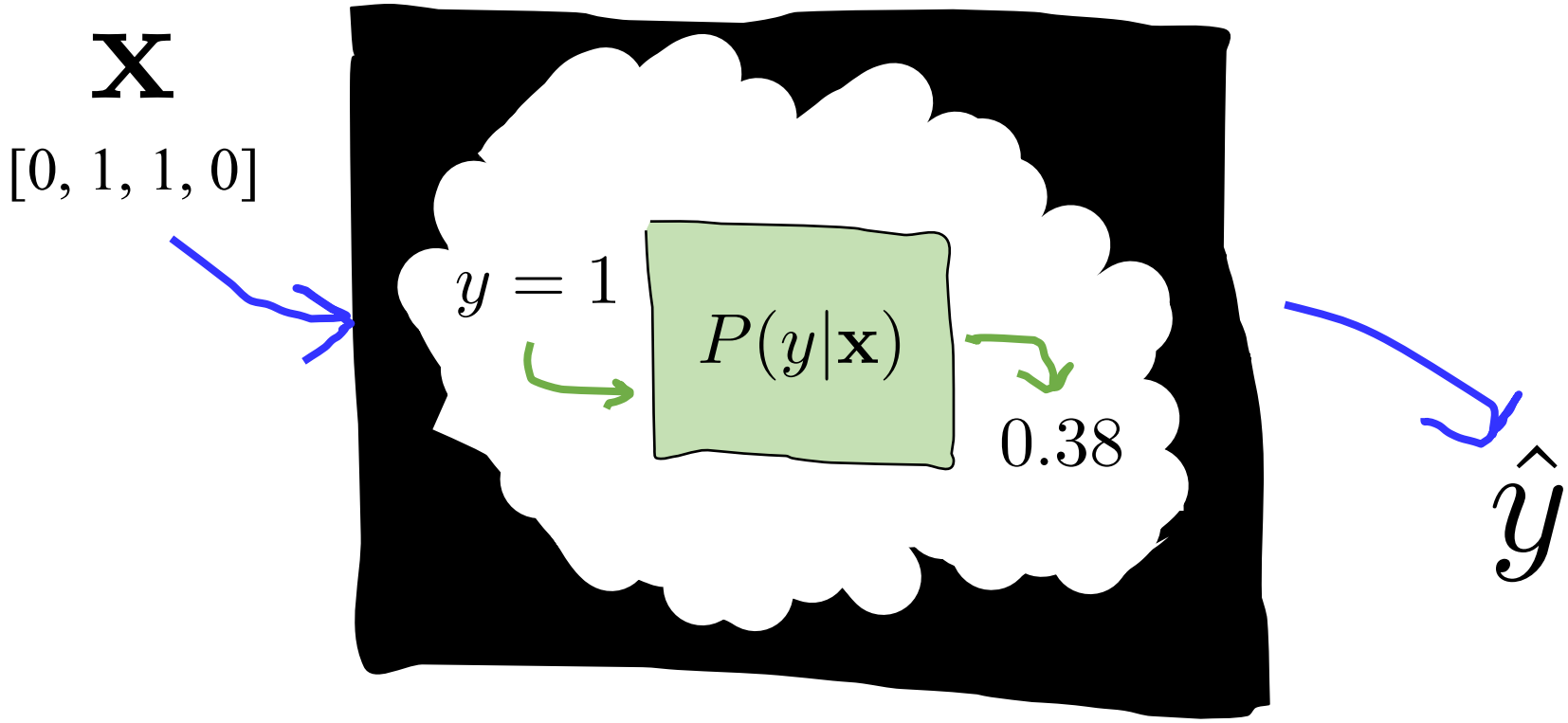
Brute Force Bayes



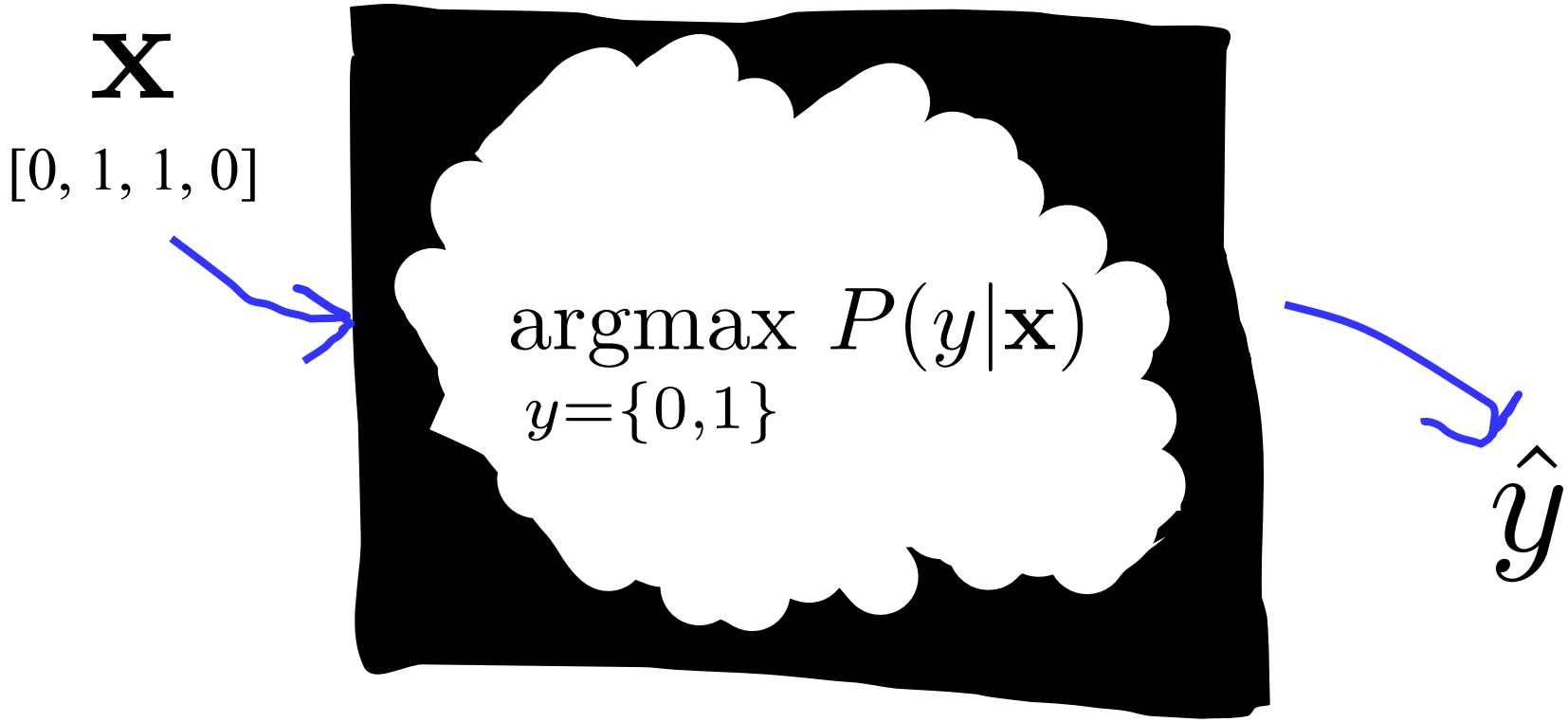
Brute Force Bayes



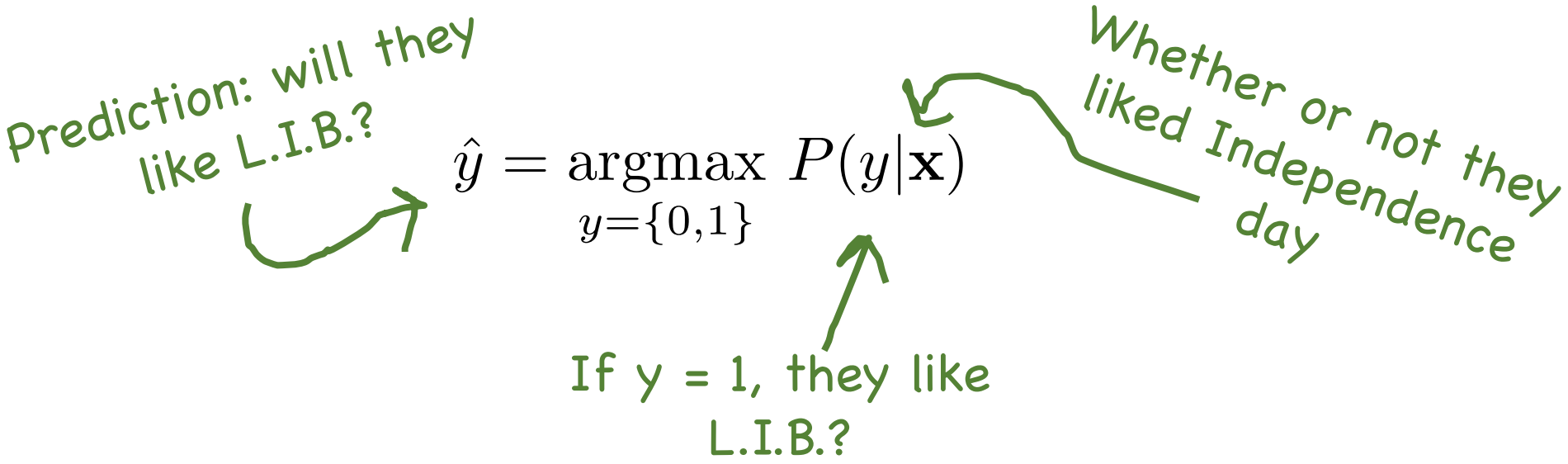
Brute Force Bayes



Brute Force Bayes



Brute Force Bayes



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

This is for one user

Brute Force Bayes

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

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

This is for one user

Brute Force Bayes

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

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

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

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 ☺

What are the Parameters?

Brute Force Bayes

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

Conditional probability table

	Y = 0	Y = 1
$X_1 = 0$	θ_0	θ_2
$X_1 = 1$	θ_1	θ_3

Y = 0	θ_4
Y = 1	θ_5

Learn these during training

Brute Force Bayes

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

Conditional probability table

X_1 \ Y	0	1
0	θ_0	θ_2
1	θ_1	θ_3

Y = 0	θ_4
Y = 1	θ_5

Learn these during training

Training

	x_1	y
		
User 1	1	1
User 2	0	0
		⋮
User n	0	1

$$P(\mathbf{x}|y)$$

X_1 \ Y	0	1
0	θ_0	θ_2
1	θ_1	θ_3

What is $P(X_1 | Y = 0)$?
What is $P(X_1 | Y = 1)$?

MLE Estimate

	x_1	y
		
User 1	1	1
User 2	0	0
		⋮
User n	0	1

$P(\mathbf{x}|y)$

X_1 \ Y	0	1
0	0.0	0.4
1	1.0	0.6

MLE: Just count

MAP Estimate

	x_1	y
		
User 1	1	1
User 2	0	0
		⋮
User n	0	1

$P(\mathbf{x}|y)$

X_1 \ Y	0	1
0	0.01	0.42
1	0.99	0.58

MAP: Just count and add imaginary trials

Testing

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

X_1 \ Y	0	1
0	0.01	0.42
1	0.99	0.58

Y = 0	0.21
Y = 1	0.79

Test user: Likes independence day

$$P(x_1 = 1|y = 0)P(y = 0)$$

vs

$$P(x_1 = 1|y = 1)P(y = 1)$$

Testing

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

X_1 \ Y	0	1
0	0.01	0.42
1	0.99	0.58

Y = 0	0.21
Y = 1	0.79

Test user: Likes independence day

$$P(x_1 = 1|y = 0)P(y = 0) \qquad 0.208$$

vs

$$P(x_1 = 1|y = 1)P(y = 1)$$

Testing

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

X_1 \ Y	0	1
0	0.01	0.42
1	0.99	0.58

Y = 0	0.21
Y = 1	0.79

Test user: Likes independence day

$$P(x_1 = 1|y = 0)P(y = 0) \qquad 0.208$$

vs

$$P(x_1 = 1|y = 1)P(y = 1) \qquad 0.458$$

That was 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

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

$$= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

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

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

Brute Force Bayes

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

Y = 0

X_2 \ X_1	0	1
0	θ_0	θ_1
1	θ_2	θ_3

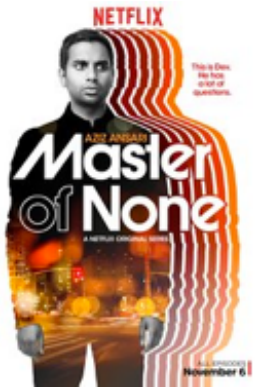
Y = 1

X_2 \ X_1	0	1
0	θ_4	θ_5
1	θ_6	θ_7

x_1 x_2 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

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

$$= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

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

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

Brute Force Bayes

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

Y = 0

$X_3 = 0$

X_2 \ X_1	0	1
0	θ_0	θ_1
1	θ_2	θ_3

Y = 1

$X_3 = 0$

X_2 \ X_1	0	1
0	θ_8	θ_9
1	θ_{10}	θ_{11}

X_2 = 0

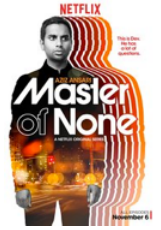
$X_3 = 1$

X_2 \ X_1	0	1
0	θ_4	θ_5
1	θ_6	θ_7

X_2 = 1

$X_3 = 1$

X_2 \ X_1	0	1
0	θ_{12}	θ_{13}
1	θ_{14}	θ_{15}



And if m=100?

Brute Force Bayes m = 100

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

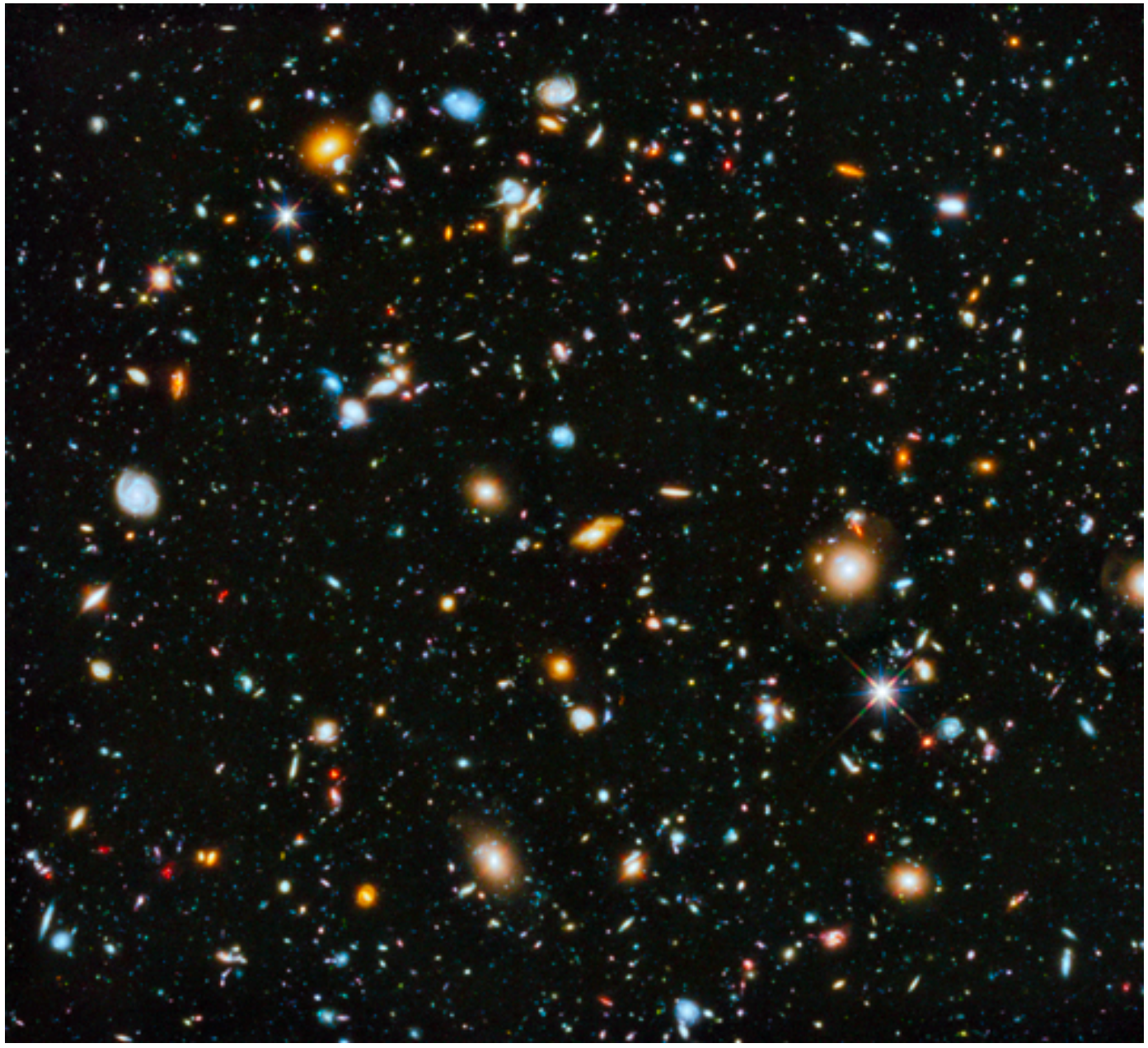
$$\hat{y} = \underset{y=\{0,1\}}{\operatorname{argmax}} \; P(y|\mathbf{x})$$

$$= \underset{y=\{0,1\}}{\operatorname{argmax}} \; \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

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

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

Oops… Number of atoms in the univserse



What is the big O for # parameters?
m = # features.

Big O of Brute Force Joint

What is the big O for # parameters?
m = # features.

$$O(2^m)$$

Assuming each feature is binary…

Not going to cut it!

What is the problem here?

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

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

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

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

Naïve Bayes Assumption

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

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

The Naïve Bayes assumption

Naïve Bayes Assumption:

$$P(\mathbf{x}|y) = \prod_i P(x_i|y)$$

Naïve Bayes Classifier

Naïve Bayes

Our prediction for y

Is a function of x

That choses the best value of y given x

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

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

Bayes rule!

$$= \underset{y}{\operatorname{argmax}} \left(\prod_{i=1}^{n} \hat{P}(x_i|y) \right) \hat{P}(y)$$

Naïve Bayes Assumption

$$= \underset{y}{\operatorname{argmax}} \log \hat{P}(y) + \sum_{i=1}^{m} \log \hat{P}(x_i|y)$$

This log version is useful for numerical stability

Naïve Bayes Example

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate params: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MLE estimates	
0	3	10	0.23	0.77
1	4	13	0.24	0.76

Y \ X_2	0	1	MLE estimates	
0	5	8	0.38	0.62
1	7	10	0.41	0.59

Y	#	MLE est.
0	13	0.43
1	17	0.57

  - Say someone likes **Star Wars ($X_1 = 1$),** but not **Harry Potter ($X_2 = 0$)**
  - Will they like "Lord of the Rings"? Need to predict Y:

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(\mathbf{x}|y)\hat{P}(y) = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(x_1|y)\hat{P}(x_2|y)\hat{P}(y)$$

Naïve Bayes Example

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate params: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MLE estimates	
0	3	10	0.23	0.77
1	4	13	0.24	0.76

Y \ X_2	0	1	MLE estimates	
0	5	8	0.38	0.62
1	7	10	0.41	0.59

Y	#	MLE est.
0	13	0.43
1	17	0.57

  - Say someone likes **Star Wars ($X_1 = 1$),** but not **Harry Potter ($X_2 = 0$)**
  - Will they like "Lord of the Rings"? Need to predict Y:

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(X_1 = x_1|Y = y)\hat{P}(X_2 = x_2|Y = y)\hat{P}(Y = y)$$

Naïve Bayes Example

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate params: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MLE estimates	
0	3	10	0.23	0.77
1	4	13	0.24	0.76

Y \ X_2	0	1	MLE estimates	
0	5	8	0.38	0.62
1	7	10	0.41	0.59

Y	#	MLE est.
0	13	0.43
1	17	0.57

  - Say someone likes **Star Wars ($X_1 = 1$),** but not **Harry Potter ($X_2 = 0$)**
  - Will they like "Lord of the Rings"? Need to predict Y:

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(X_1 = 1|Y = y)\hat{P}(X_2 = 0|Y = y)\hat{P}(Y = y)$$

Naïve Bayes Example

Y \ X_1	0	1	MLE estimates	
0	3	10	0.23	0.77
1	4	13	0.24	0.76

Y \ X_2	0	1	MLE estimates	
0	5	8	0.38	0.62
1	7	10	0.41	0.59

Y	#	MLE est.
0	13	0.43
1	17	0.57

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(X_1 = 1|Y = y)\hat{P}(X_2 = 0|Y = y)\hat{P}(Y = y)$$

- Let Y = 0

$$\hat{P}(X_1 = 1|Y = 0)\hat{P}(X_2 = 0|Y = 0)\hat{P}(Y = 0)$$
$$= (0.77)(0.38)(0.43) = 0.126$$

- Let Y = 1

$$\hat{P}(X_1 = 1|Y = 1)\hat{P}(X_2 = 0|Y = 1)\hat{P}(Y = 1)$$
$$= (0.76)(0.41)(0.57) = 0.178$$

Since term is greatest when Y = 1, we predict $\hat{Y} = 1$

$$P(Y = 1) = K \cdot 0.178 \qquad P(Y = 0) = K \cdot 0.126 \qquad K = \frac{1}{0.126 + 0.178}$$

MAP Naïve Bayes

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate PMFs: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MAP estimates
0	3	10	
1	4	13	

Y \ X_2	0	1	MAP estimates
0	5	8	
1	7	10	

Y	#	MAP est.
0	13	
1	17	

What prior?

MAP Naïve Bayes

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate PMFs: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MAP estimates	
0	3	10	0.27	0.73
1	4	13		

Y \ X_2	0	1	MAP estimates
0	5	8	
1	7	10	

Y	#	MAP est.
0	13	
1	17	

Laplace! $p_i = \frac{n_i + 1}{n + m}$ $p_i = \frac{n_i + 1}{n + 2}$

MAP Naïve Bayes

- Predict Y based on observing variables X_1 and X_2
  - X_1 and X_2 are both indicator variables
    - X_1 denotes "likes Star Wars", X_2 denotes "likes Harry Potter"
  - Y is indicator variable: "likes Lord of the Rings"
    - Use training data to estimate PMFs: $\hat{P}(x_i|y)$ $\hat{P}(y)$

Y \ X_1	0	1	MAP estimates	
0	3	10	0.27	0.73
1	4	13	0.26	0.74

Y \ X_2	0	1	MAP estimates	
0	5	8	0.4	0.6
1	7	10	0.42	0.58

Y	#	MAP est.
0	13	0.44
1	17	0.56

Laplace! $p_i = \frac{n_i + 1}{n + m}$ $p_i = \frac{n_i + 1}{n + 2}$

Training Naïve Bayes is just estimating conditional probabilities.

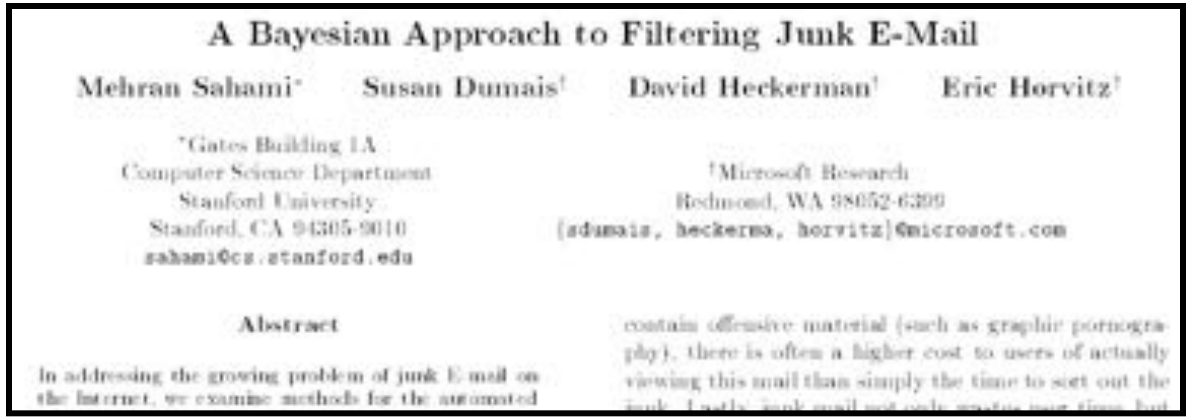
Thus training is just counting.

What is Bayes Doing in my Mail Server

- This is spam:

Let's get Bayesian on your spam:

```
Content analysis details:   (49.5 hits, 7.0 required)
 0.9 RCVD_IN_PBL            RBL: Received via a relay in Spamhaus PBL
                            [93.40.189.29 listed in zen.spamhaus.org]
 1.5 URIBL_WS_SURBL         Contains an URL listed in the WS SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_JP_SURBL         Contains an URL listed in the JP SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_OB_SURBL         Contains an URL listed in the OB SURBL blocklist
                            [URIs: recragas.cn]
 5.0 URIBL_SC_SURBL         Contains an URL listed in the SC SURBL blocklist
                            [URIs: recragas.cn]
 2.0 URIBL_BLACK            Contains an URL listed in the URIBL blacklist
                            [URIs: recragas.cn]
 8.0 BAYES_99               BODY: Bayesian spam probability is 99 to 100%
                            [score: 1.0000]
```


A Bayesian Approach to Filtering Junk E-Mail

Mehran Sahami* Susan Dumais† David Heckerman† Eric Horvitz†

*Gates Building 1A
Computer Science Department
Stanford University
Stanford, CA 94305-9010
sahami@cs.stanford.edu

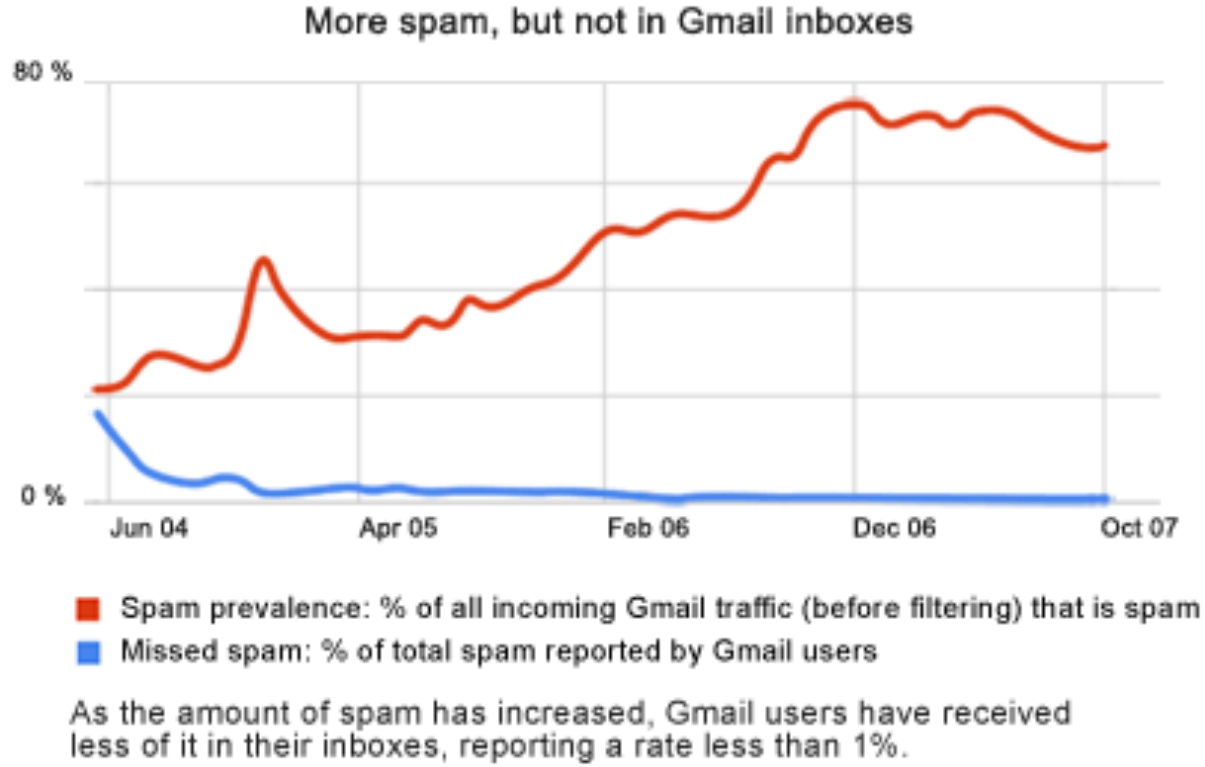
†Microsoft Research
Redmond, WA 98052-6399
{sdumais, heckerma, horvitz}@microsoft.com

Abstract

In addressing the growing problem of junk E-mail on the Internet, we examine methods for the automated

Spam, Spam… Go Away!

- The constant battle with spam



"And machine-learning algorithms developed to merge and rank large sets of Google search results allow us to combine hundreds of factors to classify spam."

Source: **`http://www.google.com/mail/help/fightspam/spamexplained.html`**

Email Classification

- Want to predict if an email is spam or not
  - Start with the input data
    - Consider a lexicon of *m* words (Note: in English $m \approx 100{,}000$)
    - Define *m* indicator variables $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
    - Each variable X_i denotes if word *i* appeared in a document or not
    - Note: *m* is huge, so make "Naive Bayes" assumption
  - Define output classes Y to be: {spam, non-spam}
  - Given training set of *N* previous emails
    - For each email message, we have a training instance:
      $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$ noting for each word, if it appeared in email
    - Each email message is also marked as spam or not (value of Y)

Training the Classifier

- Given *N* training pairs:

  $(\mathbf{x}^{(1)}, y^{(1)})$, $(\mathbf{x}^{(2)}, y^{(2)})$, …, $(\mathbf{x}^{(n)}, y^{(n)})$

- Learning
  - Estimate probabilities $\mathrm{P}(y)$ and $\mathrm{P}(x_i \mid y)$ for all i
    - Many words are likely to not appear at all in given set of email
  - Laplace estimate: $\hat{p}(X_i = 1 \mid Y = spam)_{Laplace} = \frac{(\#\text{ spam emails with word } i) + 1}{\text{total } \#\text{ spam emails} + 2}$
- Classification
  - For a new email, generate **X** = <X_1, X_2, …, X_m>
  - Classify as spam or not using: $\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(\mathbf{x}|y)\hat{P}(y)$
  - Employ Naive Bayes assumption: $P(\mathbf{x}|y) = \prod_i P(x_i|y)$

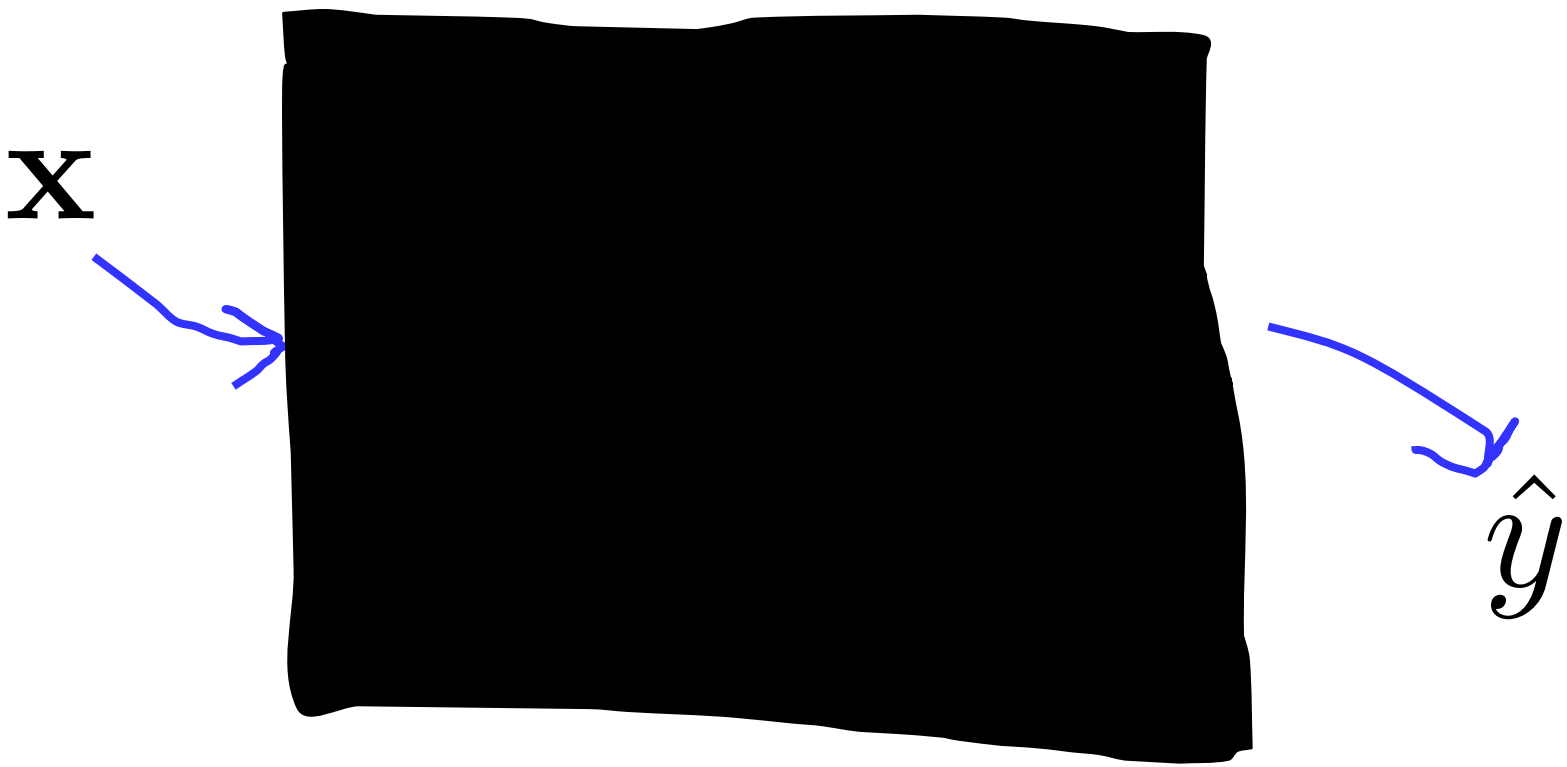
How Does This Do?

- After training, can test with another set of data
  - "Testing" set also has known values for Y, so we can see how often we were right/wrong in predictions for Y
  - Spam data
    - Email data set: 1789 emails (1578 spam, 211 non-spam)
    - First, 1538 email messages (by time) used for training
    - Next 251 messages used to test learned classifier
  - Criteria:
    - Precision = # *correctly* predicted class Y/ # predicted class Y
    - Recall = # *correctly* predicted class Y / # real class Y messages

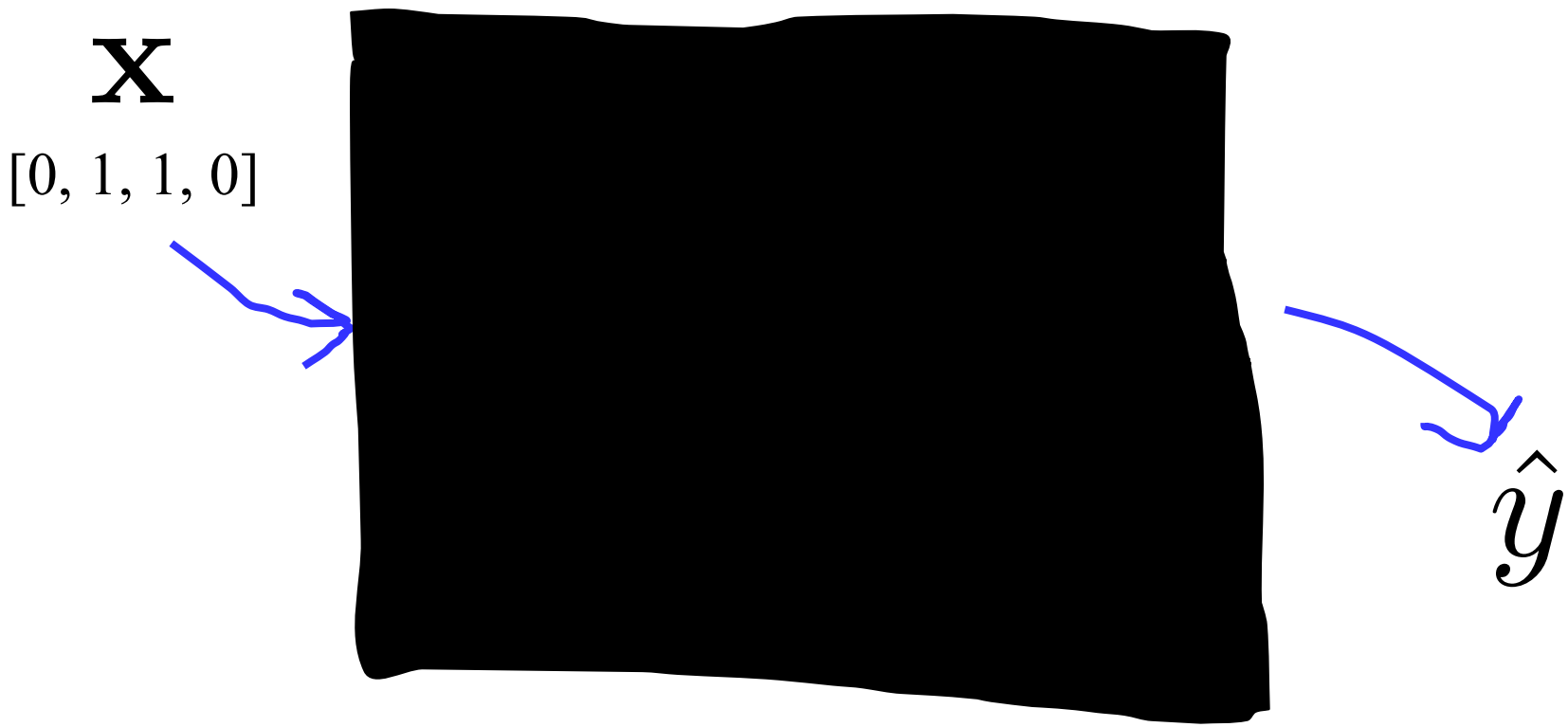
	Spam		Non-spam	
	Precision	**Recall**	**Precision**	**Recall**
Words only	**97.1%**	**94.3%**	**87.7%**	**93.4%**
Words + add'l features	**100%**	**98.3%**	**96.2%**	**100%**

Naïve Bayes Classification

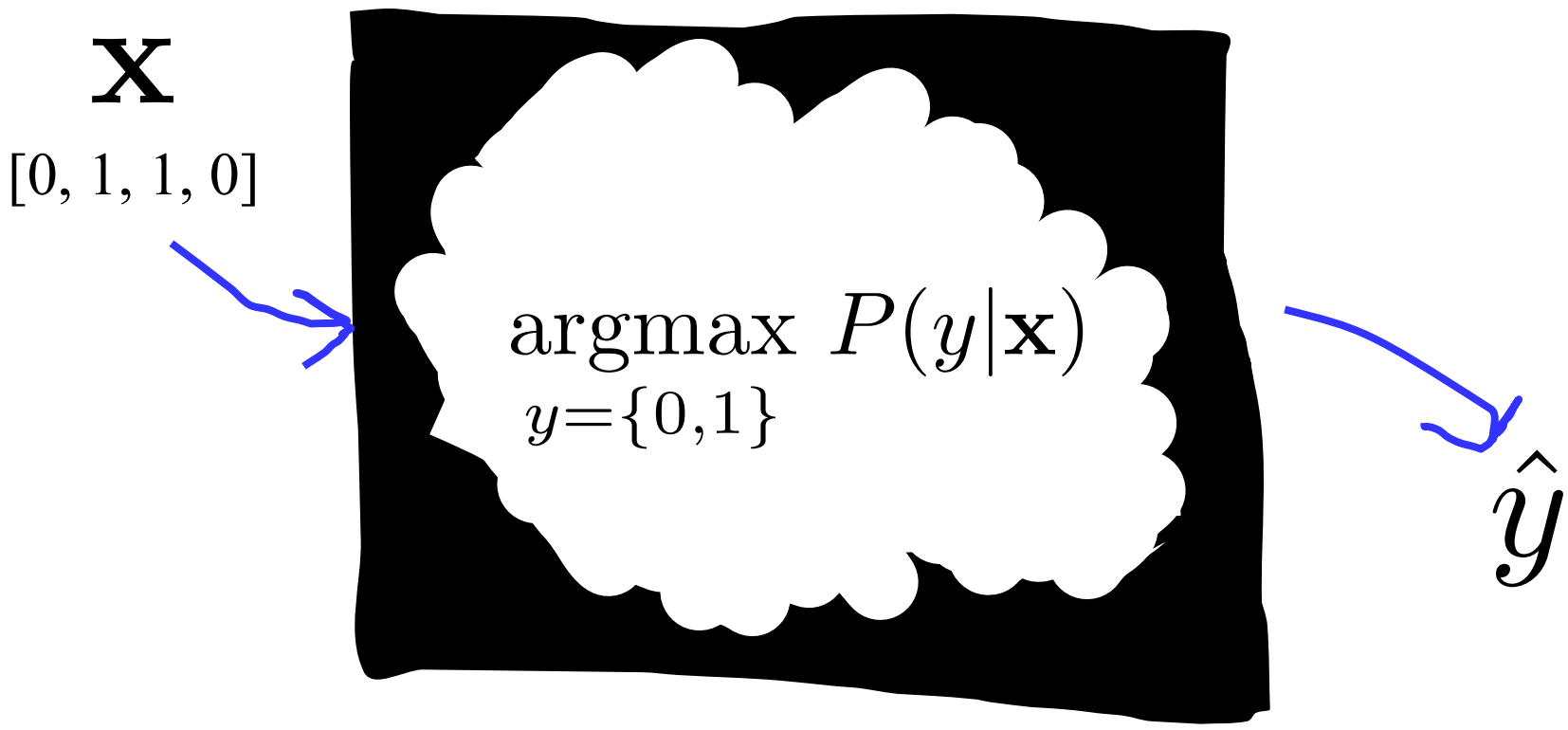
$\mathbf{g}_\theta(\mathbf{x})$?



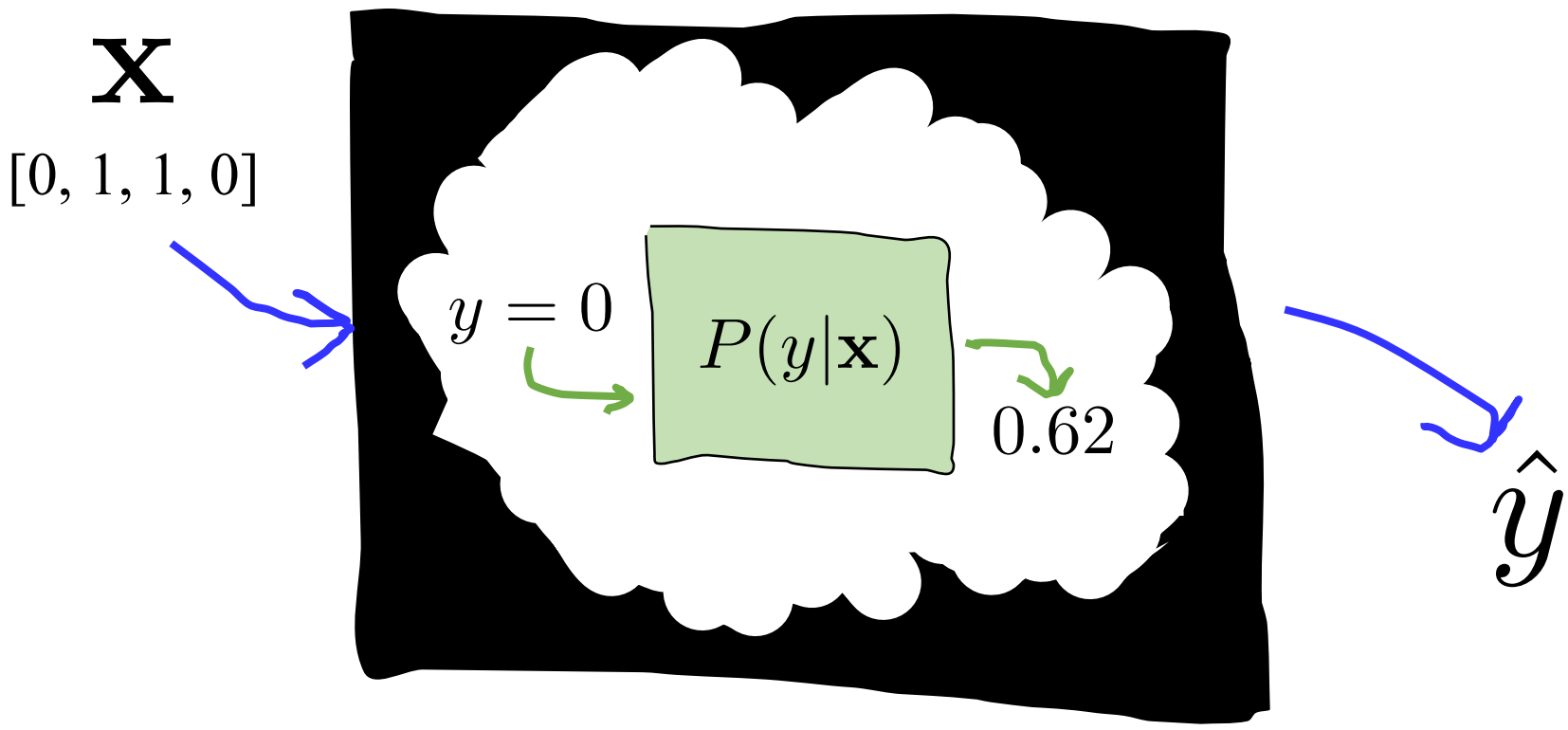
$g_\theta(\mathbf{x})$?



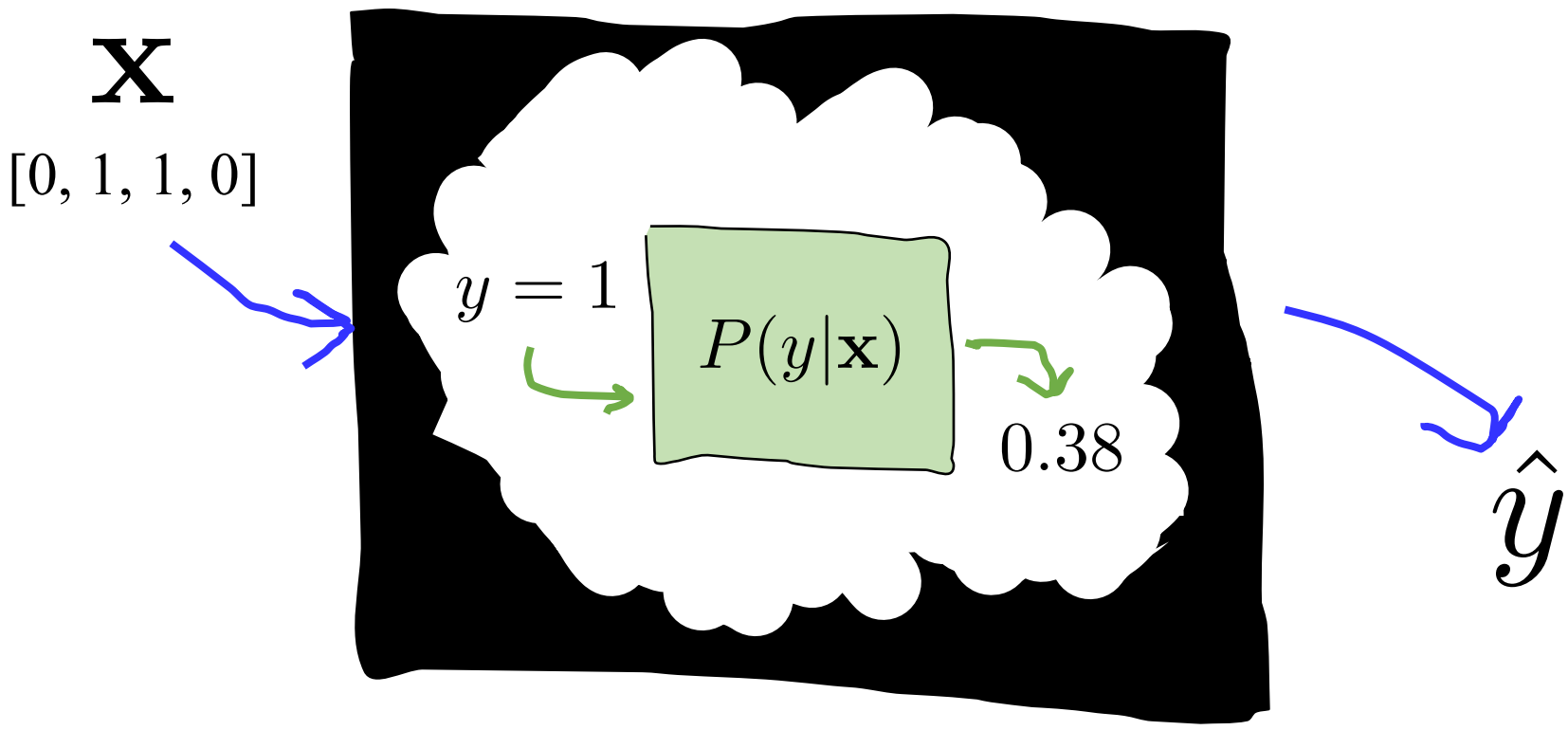
$g_\theta(\mathbf{x})$?



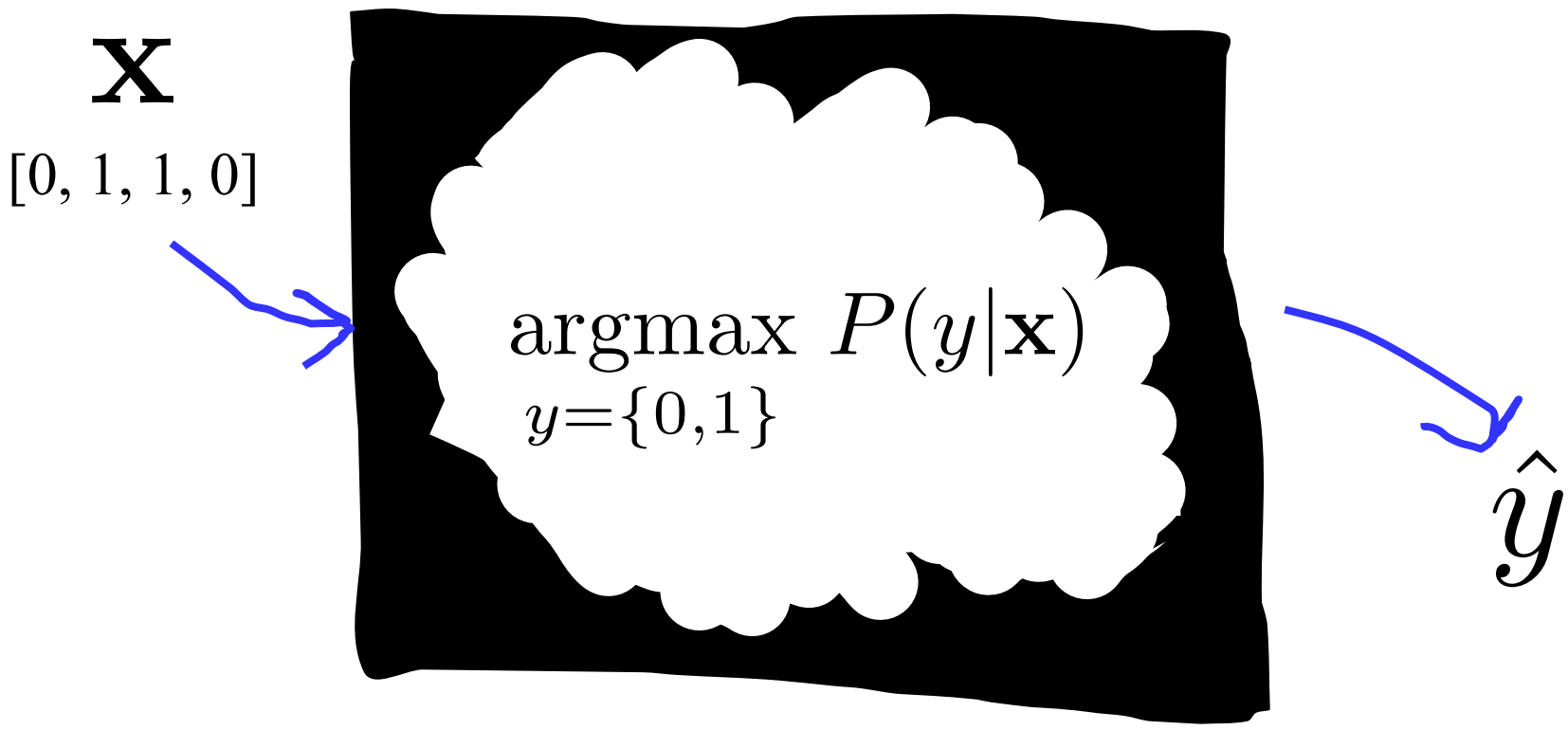
$g_\theta(\mathbf{x})$?



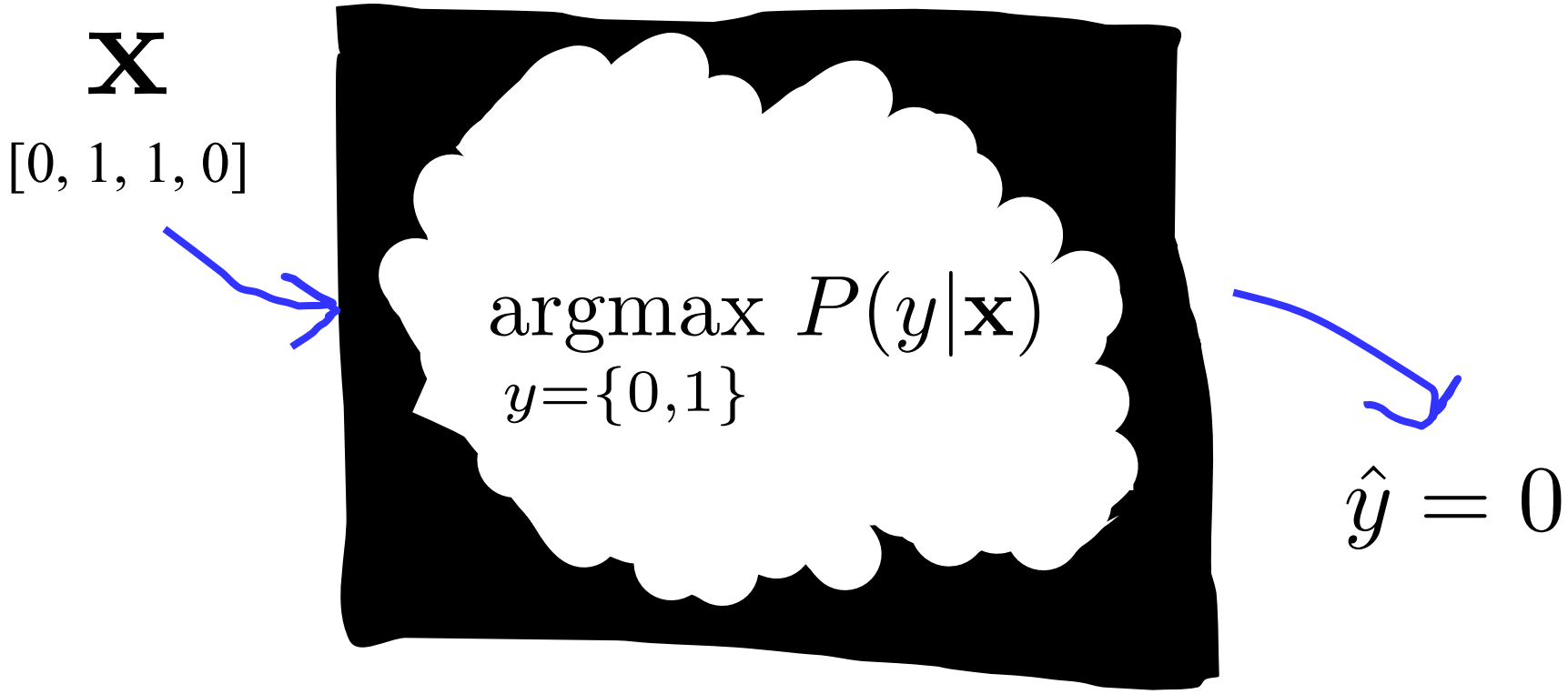
$g_\theta(\mathbf{x})$?



$g_\theta(\mathbf{x})$?



$g_\theta(\mathbf{x})$?



Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$\hat{y} = \mathrm{g}(\mathbf{x})$$
$$= \underset{y=\{0,1\}}{\arg\max}\ P(Y = y|\mathbf{X} = \mathbf{x})$$

Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$
\begin{aligned}
\hat{y} &= \mathrm{g}(\mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} P(Y = y | \mathbf{X} = \mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} \frac{P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y)}{P(\mathbf{X} = \mathbf{x})} \\
&= \arg\max_{y=\{0,1\}} P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y) \\
&= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y) \\
&= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)
\end{aligned}
$$

Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$
\begin{aligned}
\hat{y} &= \mathrm{g}(\mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} P(Y = y | \mathbf{X} = \mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} \frac{P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y)}{P(\mathbf{X} = \mathbf{x})} \\
&= \arg\max_{y=\{0,1\}} P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y) \\
&= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y) \\
&= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)
\end{aligned}
$$

Woot, Bayes!

Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$
\begin{aligned}
\hat{y} &= \mathrm{g}(\mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} P(Y = y | \mathbf{X} = \mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} \frac{P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y)}{P(\mathbf{X} = \mathbf{x})} \\
&= \arg\max_{y=\{0,1\}} P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y) \\
&= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y) \\
&= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)
\end{aligned}
$$

argmax is unaffected by P(X)

Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$
\begin{aligned}
\hat{y} &= \mathrm{g}(\mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} P(Y = y | \mathbf{X} = \mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} \frac{P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y)}{P(\mathbf{X} = \mathbf{x})} \\
&= \arg\max_{y=\{0,1\}} P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y) \\
&= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y) \\
&= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)
\end{aligned}
$$

Naïve Bayes Assumption

Naïve Bayes

Simply chose the class label that is the most likely given the data. Make Naïve Bayes assumption

$$
\begin{aligned}
\hat{y} &= \mathrm{g}(\mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} P(Y = y | \mathbf{X} = \mathbf{x}) \\
&= \arg\max_{y=\{0,1\}} \frac{P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y)}{P(\mathbf{X} = \mathbf{x})} \\
&= \arg\max_{y=\{0,1\}} P(Y = y) P(\mathbf{X} = \mathbf{x} | Y = y) \\
&= \arg\max_{y=\{0,1\}} P(Y = y) \prod_i P(X_i = x_i | Y = y) \\
&= \arg\max_{y=\{0,1\}} \log P(Y = y) + \sum_i \log P(X_i = x_i | Y = y)
\end{aligned}
$$

Argmax of log

Computing Probabilities from Data

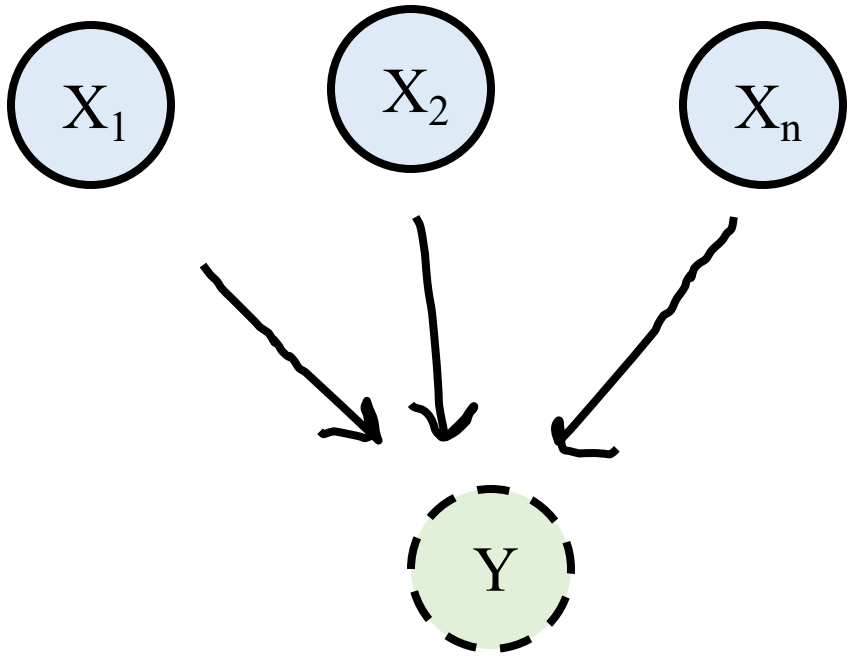
- Various probabilities you will need to compute for Naive Bayesian Classifier (using MLE here):

$$\hat{p}(X_i = 1|Y = 0) = \frac{(\#\text{ training examples where } X_i = 1 \text{ and } Y = 0)}{(\#\text{ training examples where } Y = 0)}$$

$$\hat{p}(Y = 1) = \frac{(\#\text{ training examples where } Y = 1)}{(\#\text{ training examples})}$$

Deeper Understanding

Brute Force Bayes Bayes Net

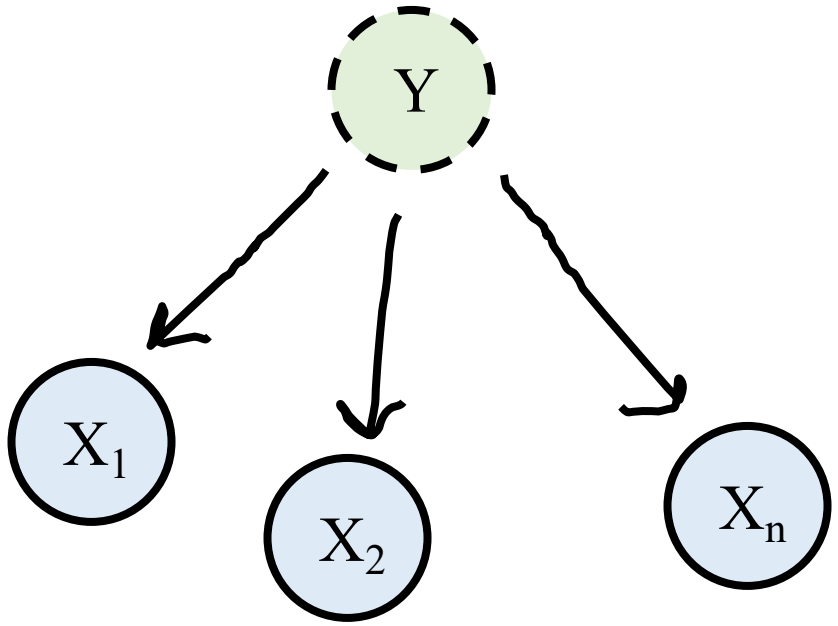


Parameters:

$$P(Y = y|\text{Parents of } Y \text{ take on specified values})$$

$$P(X_i = x_i)$$

Naïve Bayes Bayes Net



Assumption:

$$P(\mathbf{x}, y) = P(y) \prod_i P(x_i|y)$$

Parameters:

$$P(X_i = x_i|\text{Parents of } X_i \text{ take on specified values})$$

$$P(Y = y)$$