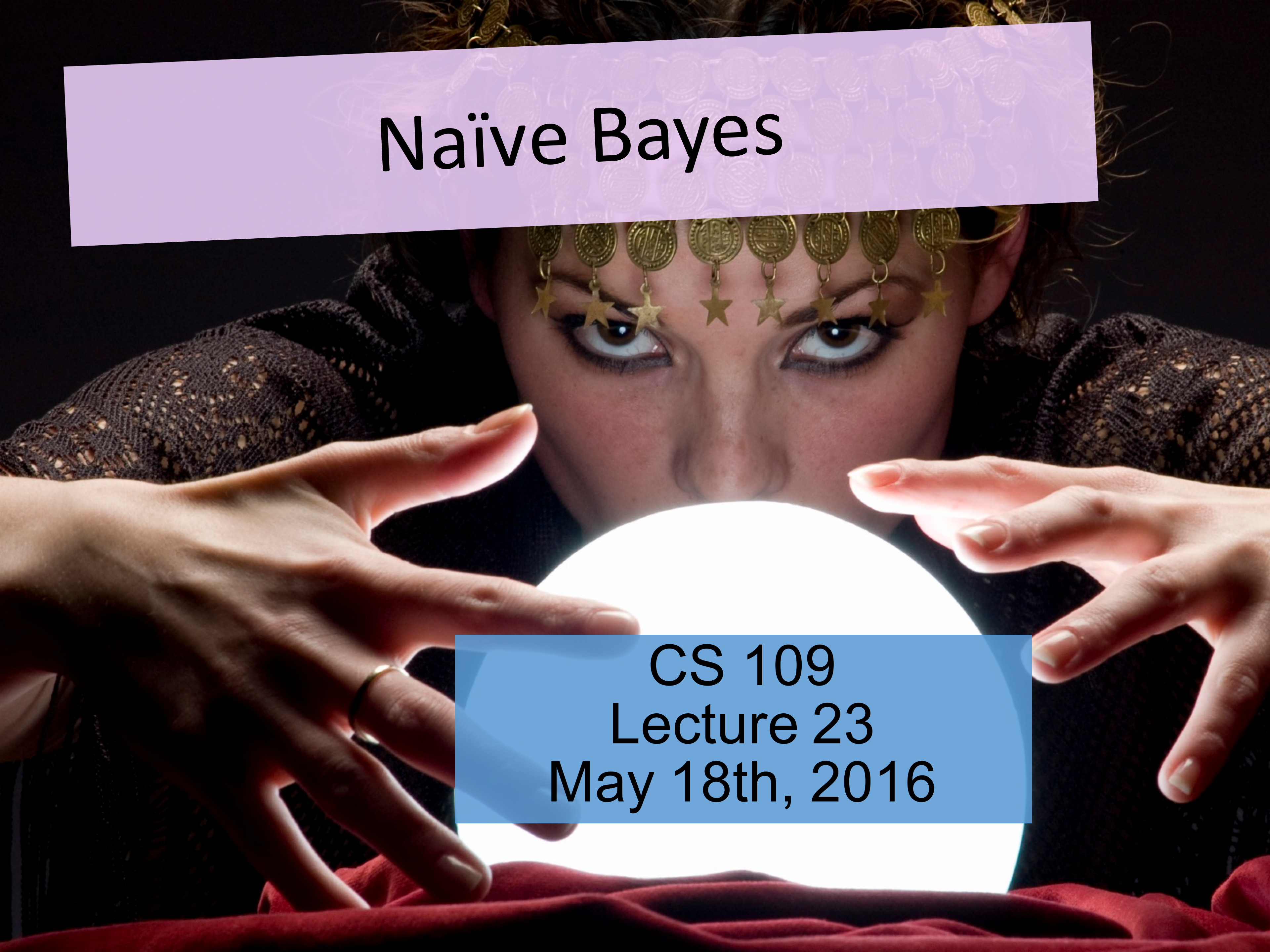
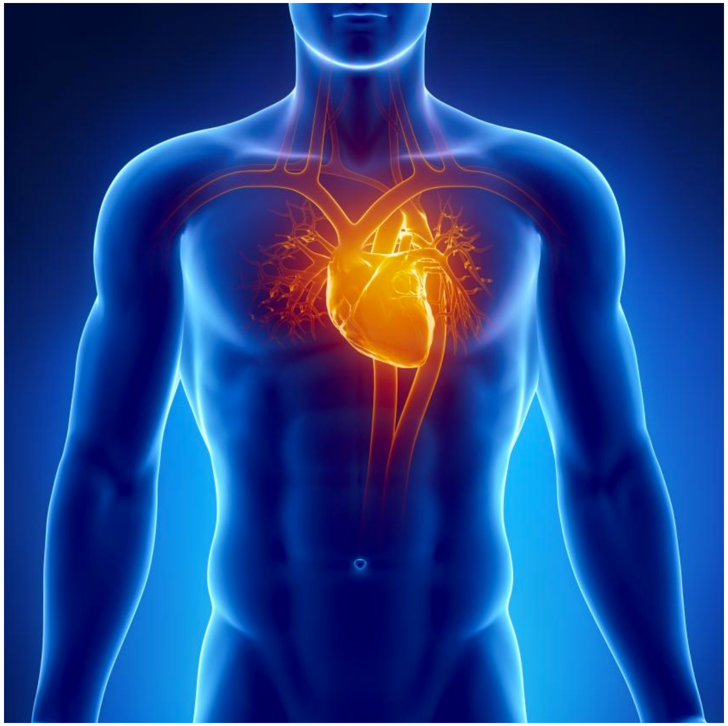

Naïve Bayes
CS 109
Lecture 23
May 18th, 2016

New Datasets

Heart



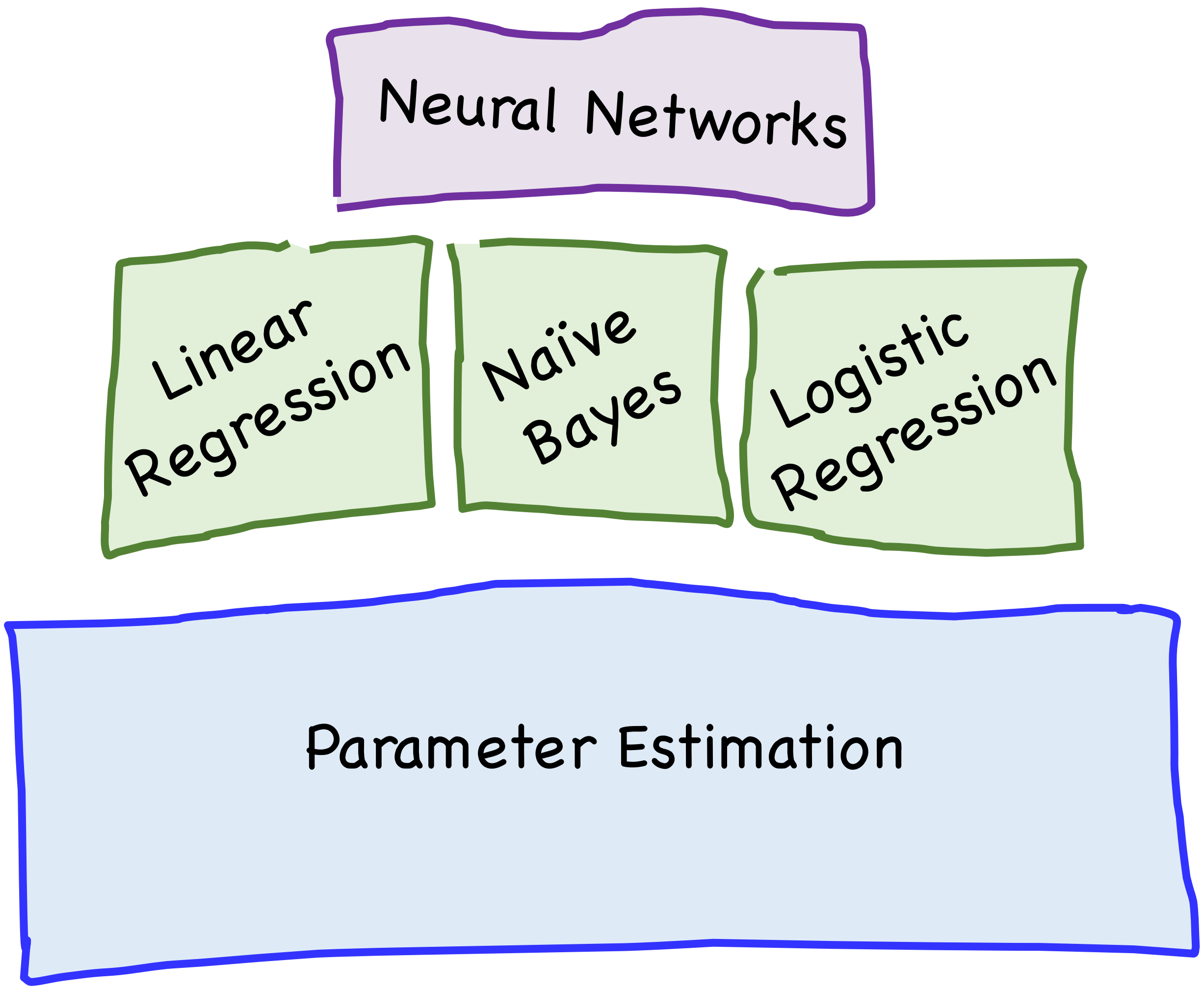
Ancestry

23andMe

Netflix

NETFLIX

Our Path


Neural Networks
Linear Regression
Naïve Bayes
Logistic Regression
Parameter Estimation

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 **X** of *m* observed variables: $<X_1, X_2, \ldots, X_m>$
    - $X_1, X_2, \ldots, X_m$ are called "input features/variables"
  - Based on observed **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"

A (Very Short) List of Applications

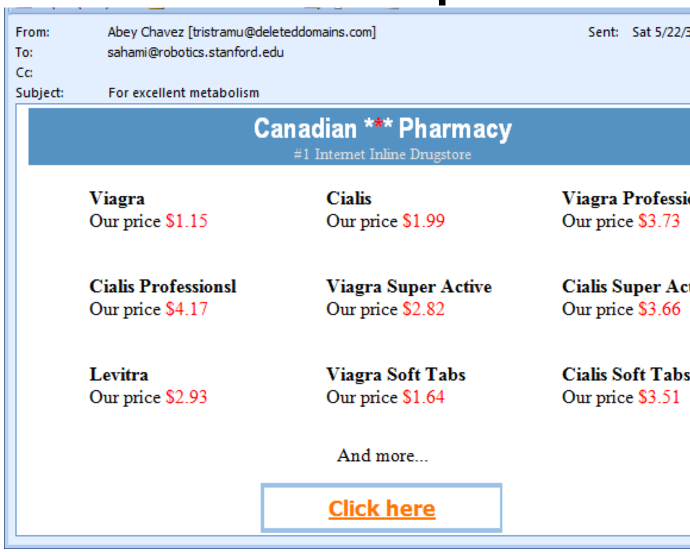
- Machine learning widely used in many contexts
  - Stock price prediction
    - Using economic indicators, predict if stock will go up/down
  - Computational biology and medical diagnosis
    - Predicting gene expression based on DNA
    - Determine likelihood for cancer using clinical/demographic data
  - Predict people likely to purchase product or click on ad
    - *“Based on past purchases, you might want to buy…”*
  - Credit card fraud and telephone fraud detection
    - Based on past purchases/phone calls is a new one fraudulent?
      - Saves companies *billions(!)* of dollars annually
  - Spam E-mail detection (gmail, hotmail, many others)

That list is ridiculously short ☺

Motivating Example

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]

Who was crazy enough to think of that?

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

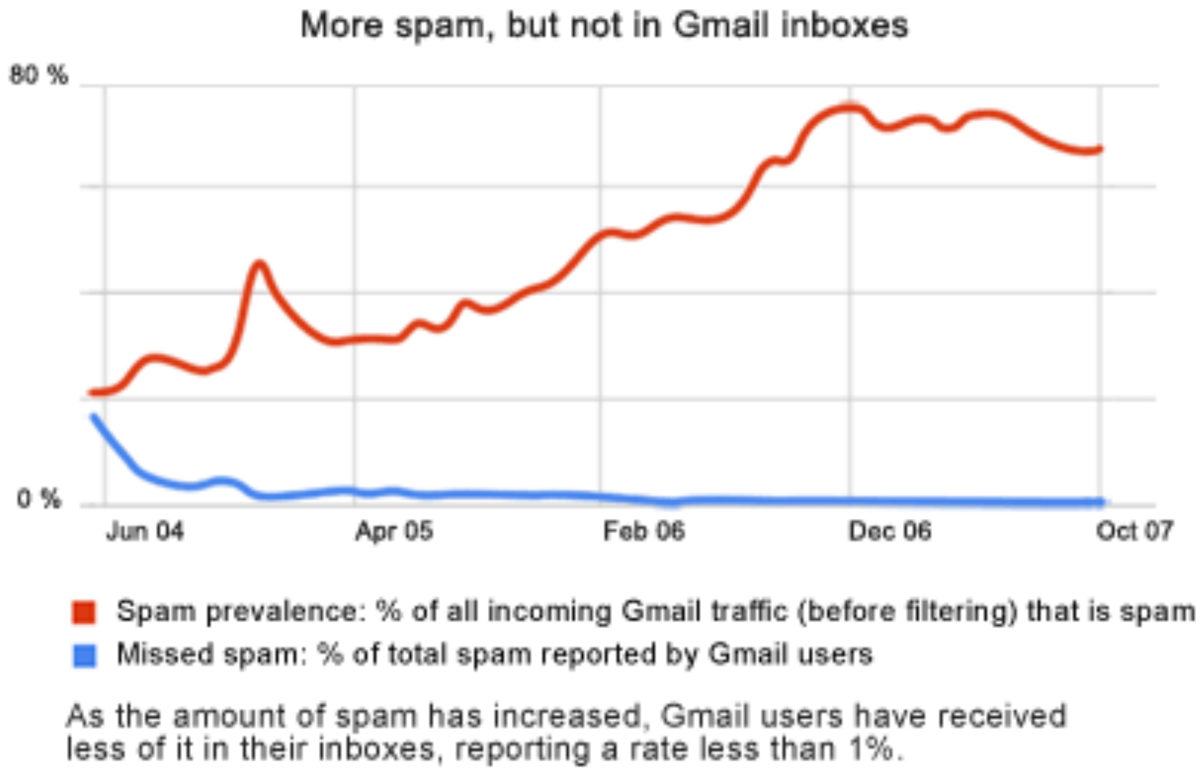
Abstract

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

contain offensive material (such as graphic pornography), there is often a higher cost to users of actually viewing this mail than simply the time to sort out the

Spam, Spam… Go Away!

- The constant battle with spam



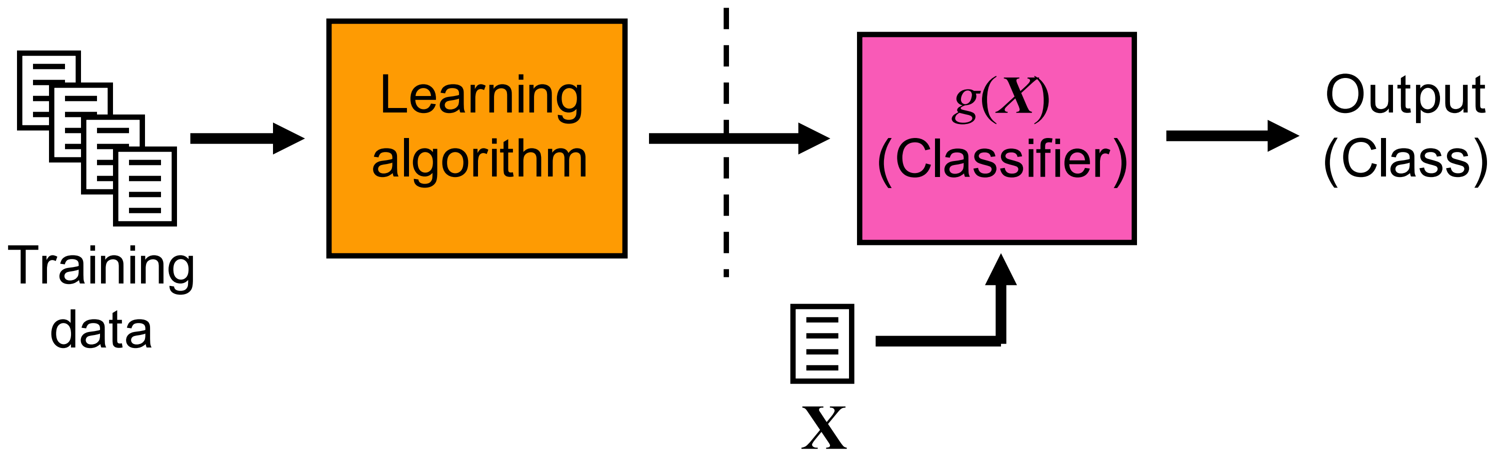
"And machine-learning algorithms developed to merge and rank large sets of Google search results allow us to combine hundreds of factors to classify spam."

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

Training a Learning Machine

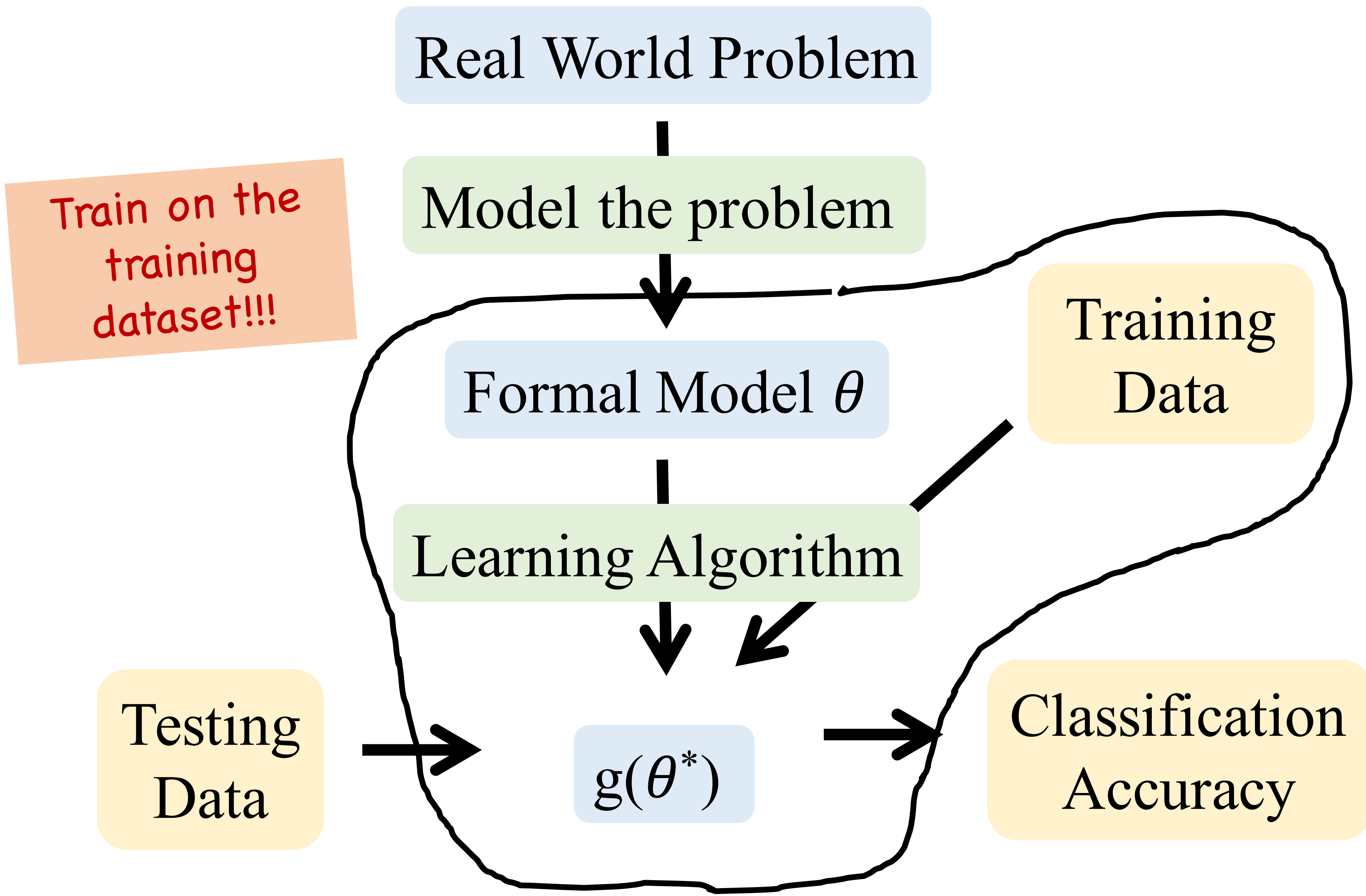
- We consider statistical learning paradigm here
  - We are given set of *N* "training" *instances*
    - Each training instance is pair: ($\langle x_1, x_2, \ldots, x_m \rangle$, y)
    - Training instances are *previously* observed data
    - Gives the output value *y* associated with each observed vector of input values $\langle x_1, x_2, \ldots, x_m \rangle$
  - Learning: use training data to specify $g(\boldsymbol{X})$
    - Generally, first select a parametric form for $g(\boldsymbol{X})$
    - Then, estimate parameters of model $g(\boldsymbol{X})$ using training data
    - For regression, usually want $g(\boldsymbol{X})$ that minimizes $E[(Y - g(\boldsymbol{X}))^2]$
      - *Mean squared error* (MSE) "loss" function. (Others exist.)
    - For classification, generally best choice of $g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid \boldsymbol{X})$

The Machine Learning Process



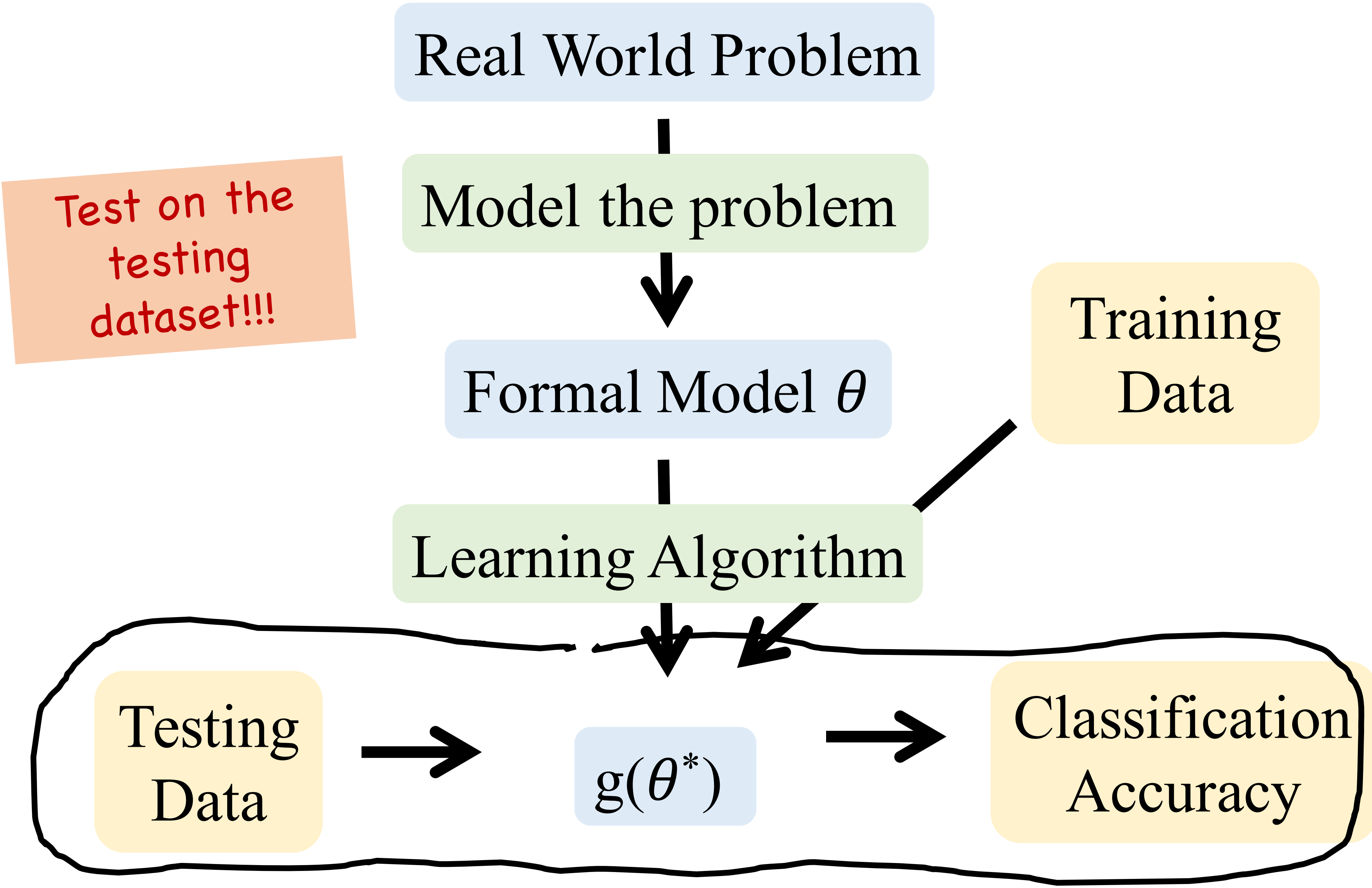
- <u>Training data</u>: set of *N* pre-classified data instances
  - *N* training pairs: $(\langle x\rangle^{(1)}, y^{(1)})$, $(\langle x\rangle^{(2)}, y^{(2)})$, …, $(\langle x\rangle^{(N)}, y^{(N)})$
    - Use superscripts to denote *i*-th training instance
- <u>Learning algorithm</u>: method for determining $g(X)$
  - Given a new input observation of $\mathbf{X} = \langle X_1, X_2, \ldots, X_m\rangle$
  - Use $g(X)$ to compute a corresponding output (prediction)
  - When prediction is discrete, we call $g(X)$ a "classifier" and call the output the predicted "class" of the input

Training



Train on the training dataset!!!
Real World Problem
Model the problem
Formal Model θ
Learning Algorithm
Training Data
Testing Data
$g(\theta^*)$
Classification Accuracy

Testing



Test on the testing dataset!!!
Real World Problem
Model the problem
Formal Model θ
Training Data
Learning Algorithm
Testing Data
$g(\theta^*)$
Classification Accuracy

Linear Regression

A Grounding Example: Linear Regression

- Predict real value Y based on observing variable X
  - Assume model is linear: $\hat{Y} = g(\boldsymbol{X}) = aX + b$
  - Training data
    - Each vector **X** has one observed variable: <X_1> (just call it X)
    - Y is continuous output variable
    - Given *N* training pairs: (<x>$^{(1)}$,y$^{(1)}$), (<x>$^{(2)}$,y$^{(2)}$), …, (<x>$^{(N)}$, y$^{(N)}$)
      - Use superscripts to denote *i*-th training instance
  - Determine *a* and *b* by minimizing $E[(Y - g(\boldsymbol{X}))^2]$

Predicting CO_2

X_1 = Temperature

X_2 = Elevation

X_3 = CO_2 level yesterday

X_4 = GDP of region

X_5 = Acres of forest growth

Y = CO_2 levels

How Did We Get Linear Regression?

N training pairs: $(\boldsymbol{x}^{(1)}, y^{(1)})$, $(\boldsymbol{x}^{(2)}, y^{(2)})$, …, $(\boldsymbol{x}^{(N)}, y^{(N)})$

1. Linear Regression Model:

$$\begin{aligned} Y &= \theta_1 X_1 + \theta_2 X_2 + \ldots \theta_{n-1} X_{n-1} + \theta_n 1 + Z \\ &= \theta^T \mathbf{X} + Z \end{aligned}$$

$$Z \sim N(0, \sigma^2)$$

2. Find the LL function and chose thetas which maximize it

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}} \; - \sum_{i=1}^{n} (Y^{(i)} - \theta^T \mathbf{x}^{(i)})^2$$

3. Use an optimizer to calculate each theta.

Classification

A Simple Classification Example

- Predict Y based on observing variables X
  - X has discrete value from {1, 2, 3, 4}
    - X denotes temperature range today: <50, 50-60, 60-70, >70
  - Y has discrete value from {rain, sun}
    - Y denotes general weather outlook tomorrow
  - Note Bayes' Thm.: $P(Y \mid X) = \dfrac{p_{X,Y}(x,y)}{p_X(x)} = \dfrac{p_{X|Y}(x \mid y)\, p_Y(y)}{p_X(x)}$
  - For new X, predict $\hat{Y} = g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid X)$
    - Note $p_x(x)$ is not affected by choice of *y*, yielding:

$$\hat{Y} = g(\boldsymbol{X}) = \arg\max_y \hat{P}(Y \mid X) = \arg\max_y \hat{P}(X, Y) = \arg\max_y \hat{P}(X \mid Y)\hat{P}(Y)$$

Brute Force Classification

Estimating the Complete Joint

- From last slide:

$$\hat{Y} = g(\boldsymbol{X}) = \arg\max_{y} \hat{P}(Y \mid X) = \arg\max_{y} \hat{P}(X, Y) =$$

- **First idea**: Let (X,Y) be one giant multinomial! Say **X** can take on the values 1, 2, 3, 4 and Y can take on the values 1,2

Y \ X	1	2	3	4
1	$\theta_{1,1}$	$\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,4}$
2	$\theta_{2,1}$	$\theta_{2,2}$	$\theta_{2,3}$	$\theta_{2,4}$

- Estimate these and use them to make our prediction

Estimating the Complete Joint

- Given training data, compute joint PMF: $p_{X,Y}(x, y)$
  - **MLE**: count number of times each pair (x, y) appears
  - **MAP using Laplace prior**: add 1 to all the MLE counts
  - Normalize to get true distribution (sums to 1)
  - Observed 50 data points:

Y \ X	1	2	3	4
rain	5	3	2	0
sun	3	7	10	20

$$\hat{p}_{MLE} = \frac{\text{count in cell}}{\text{total \# data points}}$$

$$\hat{p}_{Laplace} = \frac{\text{count in cell} + 1}{\text{total \# data points} + \text{total \# cells}}$$

MLE estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.10	0.06	0.04	0.00	0.20
sun	0.06	0.14	0.20	0.40	0.80
$p_X(x)$	0.16	0.20	0.24	0.40	1.00

Laplace (MAP) estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.103	0.069	0.052	0.017	0.241
sun	0.069	0.138	0.190	0.362	0.759
$p_X(x)$	0.172	0.207	0.242	0.379	1.00

Classify New Observations

- Say today's temperature is 75, so X = 4
  - Recall X temperature ranges: <50, 50-60, 60-70, >70
  - Prediction for Y (weather outlook tomorrow)

$$\hat{Y} = \arg\max_{y} \hat{P}(X,Y) = \arg\max_{y} \hat{P}(X \mid Y)\hat{P}(Y)$$

MLE estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.10	0.06	0.04	0.00	0.20
sun	0.06	0.14	0.20	0.40	0.80
$p_X(x)$	0.16	0.20	0.24	0.40	1.00

Laplace (MAP) estimate

Y \ X	1	2	3	4	$p_Y(y)$
rain	0.103	0.069	0.052	0.017	0.241
sun	0.069	0.138	0.190	0.362	0.759
$p_X(x)$	0.172	0.207	0.242	0.379	1.00

  - What if we asked what is probability of rain tomorrow?
    - MLE: absolutely, positively no chance of rain!
    - Laplace estimate: small chance → "never say never"

Classification with Multiple Observations

- Say, we have *m* input values **X** = <X_1, X_2, …, X_m>
  - Note that variables $X_1, X_2, \ldots, X_m$ can be dependent!
  - In *theory*, could predict Y as before, using

$$\hat{Y} = \arg\max_y \hat{P}(\boldsymbol{X}, Y) = \arg\max_y \hat{P}(\boldsymbol{X} \mid Y)\hat{P}(Y)$$

    - Why won't this necessarily work in practice?
  - Need to estimate $P(X_1, X_2, \ldots, X_m \mid Y)$
    - Fine if *m* is small, but what if m = 10 or 100 or 10,000?
    - Need ridiculous amount of data for good probability estimates!
    - Likely to have many 0's in table (bad times)
  - Need to consider a simpler model

NETFLIX

And Learn

Netflix and Learn

- Say, we have m input values **X** = $<X_1, X_2, \ldots, X_m>$ and a single Y. Each X_i represents if a user liked movie i*.*
- Let's think about the joint distribution for different values of m

Netflix and Learn: m = 1

- Say, we have *m* input values $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$ and a single Y. Each X_i represents if a user liked movie i.

Y \ X_1	0	1
0	$\theta_{0,0}$	$\theta_{0,1}$
1	$\theta_{1,0}$	$\theta_{1,1}$

Netflix and Learn: m = 2

- Say, we have m input values $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$ and a single Y. Each X_i represents if a user liked movie i.

X_2 = 0

Y \ X_1	0	1
0	$\theta_{0,0,0}$	$\theta_{0,1,0}$
1	$\theta_{1,0,0}$	$\theta_{1,1,0}$

X_2 = 1

Y \ X_1	0	1
0	$\theta_{0,0,1}$	$\theta_{0,1,1}$
1	$\theta_{1,0,1}$	$\theta_{1,1,1}$

Netflix and Learn: m = 3

- Say, we have *m* input values $\mathbf{X} = <X_1, X_2, \ldots, X_m>$ and a single Y. Each X_i represents if a user liked movie i.

$X_3 = 0$

$X_2 = 0$

Y \ X_1	0	1
0	$\theta_{0,0,0,0}$	$\theta_{0,1,0,0}$
1	$\theta_{1,0,0,0}$	$\theta_{1,1,0,0}$

$X_2 = 1$

Y \ X_1	0	1
0	$\theta_{0,0,1,0}$	$\theta_{0,1,1,0}$
1	$\theta_{1,0,1,0}$	$\theta_{1,1,1,0}$

$X_3 = 1$

$X_2 = 0$

Y \ X_1	0	1
0	$\theta_{0,0,0,1}$	$\theta_{0,1,0,1}$
1	$\theta_{1,0,0,1}$	$\theta_{1,1,0,1}$

$X_2 = 1$

Y \ X_1	0	1
0	$\theta_{0,0,1,1}$	$\theta_{0,1,1,1}$
1	$\theta_{1,0,1,1}$	$\theta_{1,1,1,1}$

And if $m=100$?

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

Big O of Brute Force Joint

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

$$\mathcal{O}(2^n)$$

Assuming each feature
is binary...

Not going to cut it!

Naïve Bayes Classifier

- Say, we have *m* input values **X** = <X_1, X_2, …, X_m>
  - Assume variables X_1, X_2, …, X_m are <u>**conditionally independent**</u> given Y
    - Really don't believe X_1, X_2, …, X_m are conditionally independent
    - Just an approximation we make to be able to make predictions
    - This is called the "Naive Bayes" assumption, hence the name
  - Predict Y using $\hat{Y} = \arg\max_y P(\mathbf{X}, Y) = \arg\max_y P(\mathbf{X} \mid Y) P(Y)$
    - But, we now have:

$$P(\mathbf{X} \mid Y) = P(X_1, X_2, \ldots X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$$ by conditional independence

  - Note: computation of PMF table is <u>linear</u> in *m* : O(*m*)
    - Don't need much data to get good probability estimates

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 PMFs: $\hat{p}_{X_i,Y}(x_i, y), \ \hat{p}_Y(y)$

Y \ X_1	0	1	MLE estimates	
0	3	10	0.10	0.33
1	4	13	0.13	0.43

Y \ X_2	0	1	MLE estimates	
0	5	8	0.17	0.27
1	7	10	0.23	0.33

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:

$$\hat{Y} = \arg\max_y \hat{P}(\mathbf{X} \mid Y)\hat{P}(Y) = \arg\max_y \hat{P}(X_1 \mid Y)\hat{P}(X_2 \mid Y)\hat{P}(Y)$$

One SciFi/Fantasy to Rule them All

Y \ X_1	0	1	MLE estimates	
0	3	10	0.10	0.33
1	4	13	0.13	0.43

Y \ X_2	0	1	MLE estimates	
0	5	8	0.17	0.27
1	7	10	0.23	0.33

Y	#	MLE est.
0	13	0.43
1	17	0.57

- Prediction for Y is value of Y maximizing P(**X**, Y):

$$\hat{Y} = \arg\max_y \hat{P}(\mathbf{X} \mid Y)\hat{P}(Y) = \arg\max_y \hat{P}(X_1 \mid Y)\hat{P}(X_2 \mid Y)\hat{P}(Y)$$

  - Compute P(**X**, Y=0): $\hat{P}(X_1 = 1 \mid Y = 0)\hat{P}(X_2 = 0 \mid Y = 0)\hat{P}(Y = 0)$

$$= \frac{\hat{P}(X_1 = 1, Y = 0)}{\hat{P}(Y = 0)} \frac{\hat{P}(X_2 = 0, Y = 0)}{\hat{P}(Y = 0)} \hat{P}(Y = 0) \approx \frac{0.33}{0.43} \frac{0.17}{0.43} 0.43 \approx 0.13$$

  - Compute P(**X**, Y=1): $\hat{P}(X_1 = 1 \mid Y = 1)\hat{P}(X_2 = 0 \mid Y = 1)\hat{P}(Y = 1)$

$$= \frac{\hat{P}(X_1 = 1, Y = 1)}{\hat{P}(Y = 1)} \frac{\hat{P}(X_2 = 0, Y = 1)}{\hat{P}(Y = 1)} \hat{P}(Y = 1) \approx \frac{0.43}{0.57} \frac{0.23}{0.57} 0.57 \approx 0.17$$

  - Since P(**X**, Y=1) > P(**X**, Y=0), we predict Ŷ = 1

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 $\mathbf{X} = <X_1, X_2, \ldots, X_m>$
    - 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:
      $\mathbf{X} = <X_1, X_2, \ldots, X_m>$ 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:

  $(\langle x \rangle^{(1)}, y^{(1)}), (\langle x \rangle^{(2)}, y^{(2)}), \ldots, (\langle x \rangle^{(N)}, y^{(N)})$

- Learning
  - Estimate probabilities P(Y) and each $P(X_i \mid Y)$ for all *i*
    - Many words are likely to not appear at all in given set of email
  - Laplace estimate: $\hat{p}(X_i = 1 \mid Y = spam)_{Laplace} = \dfrac{(\#\text{spam emails with word } i) + 1}{\text{total } \#\text{ spam emails} + 2}$
- Classification
  - For a new email, generate $\mathbf{X} = \langle X_1, X_2, \ldots, X_m \rangle$
  - Classify as spam or not using: $\hat{Y} = \arg\max_y \hat{P}(\mathbf{X} \mid Y)\hat{P}(Y)$
  - Employ Naive Bayes assumption: $\hat{P}(\mathbf{X} \mid Y) = \prod_{i=1}^{m} \hat{P}(X_i \mid Y)$

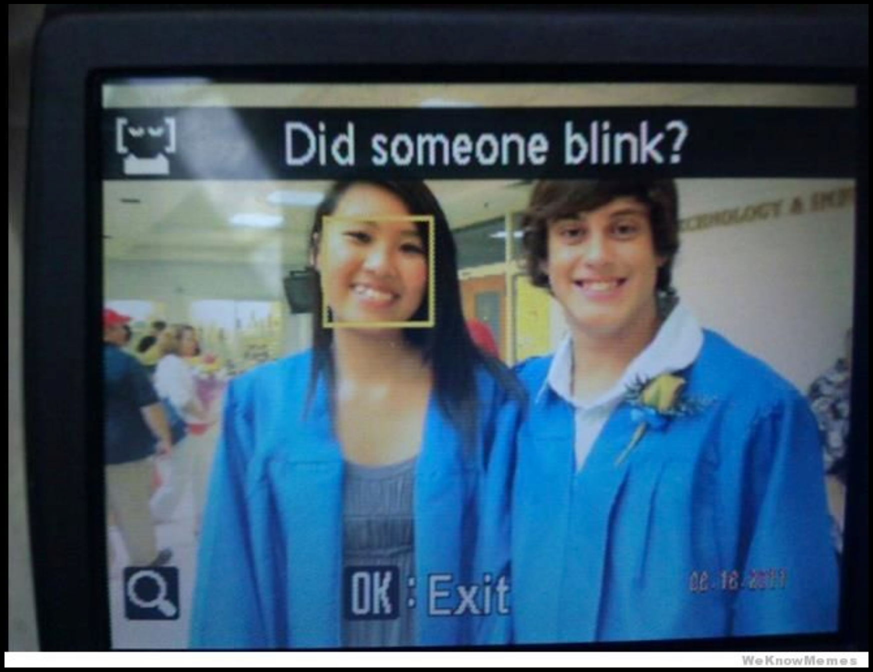
How Does This Do?

- 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%**

On biased datasets

Ethics and Datasets?



Sometimes machine learning feels universally unbiased.

We can even prove our estimators are “unbiased” ☺

Google/Nikon/HP had biased datasets

Ancestry dataset prediction

East Asian
or
Ad Mixed American (Native, European and African Americans)

Is the ancestry dataset biased?

Yes!

It is much easier
to write a binary classifier
when learning ML
for the first time

Learn Two Things From This

1. What classification with DNA Single Nucleotide Polymorphisms looks like.

2. That genetic ancestry paints a more realistic picture of how we are mixed in many nuanced ways.

3. The importance of choosing the right data to learn from. Your results will be as biased as your dataset.

Know it so you can beat it!

Ethics in Machine Learning
is a whole new field