Announcements

- Project 4
  - Due yesterday 4/4 at 11:59pm
- Homework 7
  - Due today 4/5 at 11:59pm
- Homework 8
  - Coming out soon, due Tuesday 4/12
- Project 5 Ghostbusters
  - coming out soon

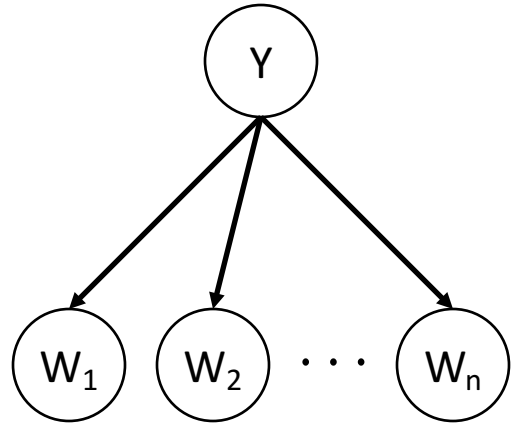
CS 188: Artificial Intelligence

Naïve Bayes

Instructors: Adam Janin & Josh Hug --- University of California, Berkeley

Challenge: Spam Detection

- We can use our idea of Bayes Nets to help us classify spam.
- Simple Bayes Net Model:
  - Define Y as label {spam, ham}.
  - Define W_i as the ith word of the email.
  - Assume $P(W_i|Y)$ is the same for every word (same CPTs).



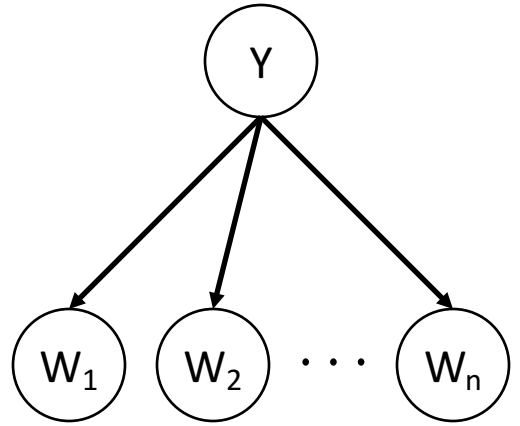
- What are some unrealistic aspects of this model?

"Yay! We are going to make tacos, with the meat optional."

w_1 = yay, w_2 = we, w_3 = are, w_4 = going, …

Challenge: Spam Detection

- We can use our idea of Bayes Nets to help us classify spam.
- Simple Bayes Net Model:
  - Define Y as label {spam, ham}.
  - Define W_i as the ith word of the email.
  - Assume $P(W_i|Y)$ is the same for every word (same CPTs).



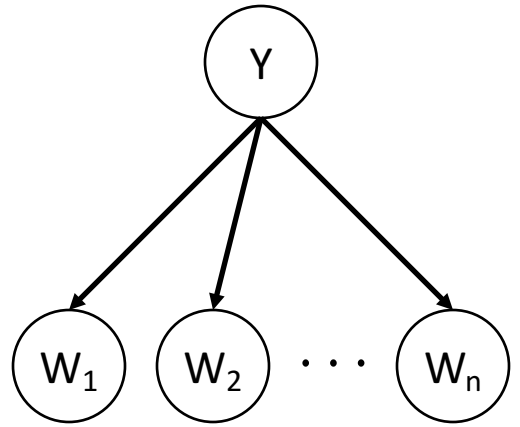
- Given an email, how could we use this model for spam detection?
  - What quantities could we calculate and how could we use them to decide?

"Yay! We are going to make tacos, with the meat optional."

w_1 = yay, w_2 = we, w_3 = are, w_4 = going, …

Challenge: Spam Detection

- We can use our idea of Bayes Nets to help us classify spam.
- Simple Bayes Net Model:
  - Define Y as label {spam, ham}.
  - Define W_i as the ith word of the email.
  - Assume $P(W_i|Y)$ is the same for every word (same CPTs).



- Given an email, how could we use this model for spam detection?
  - What quantities could we calculate?
    - $P(\text{spam}, w_1 \ldots w_n)$ and $P(\text{ham}, w_1 \ldots w_n)$
  - How would we use these quantities to decide?
    - Pick the Y with a higher joint probability $P(Y, w_1 \ldots w_n)$
    - i.e. if $P(\text{ham}, w_1 \ldots w_n) > P(\text{spam}, w_1 \ldots w_n)$, then classify as "ham"

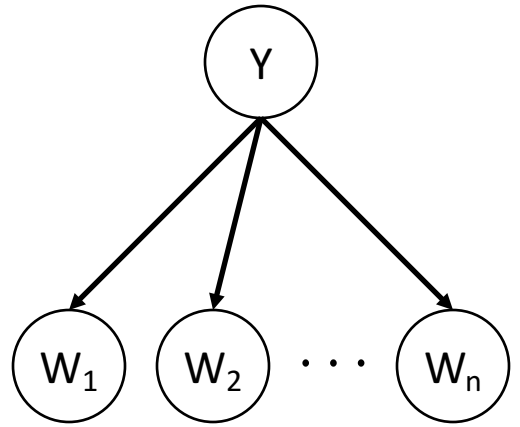
Note: Could also compute $P(\text{spam}|w_1 \ldots w_n)$ and $P(\text{ham}|w_1 \ldots w_n)$, which is trivial if you do joint PDF first.

"Yay! We are going to make tacos, with the meat optional."

w_1 = yay, w_2 = we, w_3 = are, w_4 = going, …

Challenge: Spam Detection

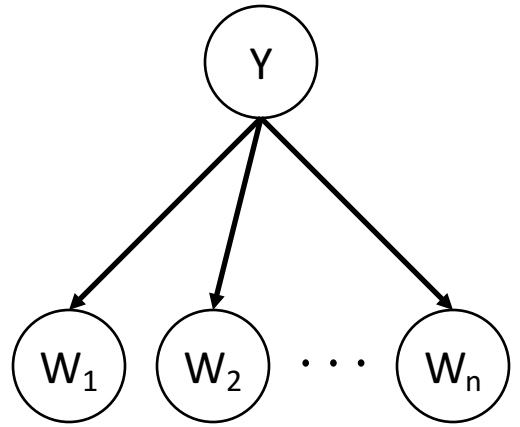
- We can use our idea of Bayes Nets to help us classify spam.
- Simple Bayes Net Model:
  - Define Y as label {spam, ham}.
  - Define W_i as the ith word of the email.
  - Assume $P(W_i|Y)$ is the same for every word (same CPTs).
- Classifying emails using a Bayes Net:
  - Calculate $P(Y, w_1 \ldots w_n)$, and return the Y with maximum probability.



- What parameters do we need for our model? How many are there? Let S_E be the number of words in English, and N be the length of the longest known email.

Challenge: Spam Detection

- We can use our idea of Bayes Nets to help us classify spam.
- Simple Bayes Net Model:
  - Define Y as label {spam, ham}.
  - Define W_i as the ith word of the email.
  - Assume $P(W_i|Y)$ is the same for every word (same CPTs).



- Classifying emails using a Bayes Net:
  - Calculate $P(Y, w_1 \ldots w_n)$, and return the Y with maximum probability.
- What parameters do we need for our model? How many are there? Let S_E be the number of words in English, and N be the length of the longest known email.
  - $P(Y)$ for each value of Y (2 parameters).
  - $P(W_i|Y)$ for each word and value of Y ($2S_E$ parameters), i.e.
    - Probability that a given word appears in spam.
    - Probability that a given word appears in ham.

Today's new topic: How do we get these?

$$P(ham) = 0.66, P(spam) = 0.33, P(\text{discount}|ham) = 0.0003, P(\text{discount}|spam) = 0.0027 \ \ldots$$

Inference Example: Spam Filtering

- Model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$
- Example parameters below:

$P(Y)$

ham	0.66
spam	0.33

$P(W|\text{spam})$

the	0.0156
to	0.0153
and	0.0115
of	0.0095
you	0.0093
a	0.0086
...	

$P(W|\text{ham})$

the	0.0210
to	0.0133
of	0.0119
2002	0.0110
with	0.0108
from	0.0107
...	

- Let's classify: "Gary, would you like to lose weight while you sleep?"

$$P(\text{spam}, \text{gary}, \text{would}, \text{you}, \dots) = P(\text{spam}) \cdot P(\text{gary}|\text{spam}) \cdot P(\text{would}|\text{spam}) \cdot \dots$$

$$P(\text{ham}, \text{gary}, \text{would}, \text{you}, \dots) = P(\text{ham}) \cdot P(\text{gary}|\text{ham}) \cdot P(\text{would}|\text{ham}) \cdot \dots$$

Inference Example: Spam Filtering

- Model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$
- Classify: "Gary, would you like to lose weight while you sleep?"

$$P(\text{spam}, \text{gary}, \text{would}, \text{you}, \dots) = P(\text{spam}) \cdot P(\text{gary}|\text{spam}) \cdot P(\text{would}|\text{spam}) \cdot \dots$$
$$P(\text{ham}, \text{gary}, \text{would}, \text{you}, \dots) = P(\text{ham}) \cdot P(\text{gary}|\text{ham}) \cdot P(\text{would}|\text{ham}) \cdot \dots$$

- Issue: Probabilities are small.
  - Numerical precision limits of real machine means products will both be zero.

- Resolution: Use logarithm of probabilities. Return Y that maximizes log.

$$\log P(\text{spam}, \text{gary}, \text{would}, \dots) = \log P(\text{spam}) + \log P(\text{gary}|\text{spam}) + \log P(\text{would}|\text{spam}) + \cdots$$
$$\log P(\text{ham}, \text{gary}, \text{would}, \dots) = \log P(\text{ham}) + \log P(\text{gary}|\text{ham}) + \log P(\text{would}|\text{ham}) + \cdots$$

Inference Example: Spam Filtering

Word	P(w|spam)	P(w|ham)	ln P(spam, words)	ln P(ham, words)
(prior)	0.33333	0.66666	-1.1	-0.4

$\ln(0.66666 \times 0.00021)$

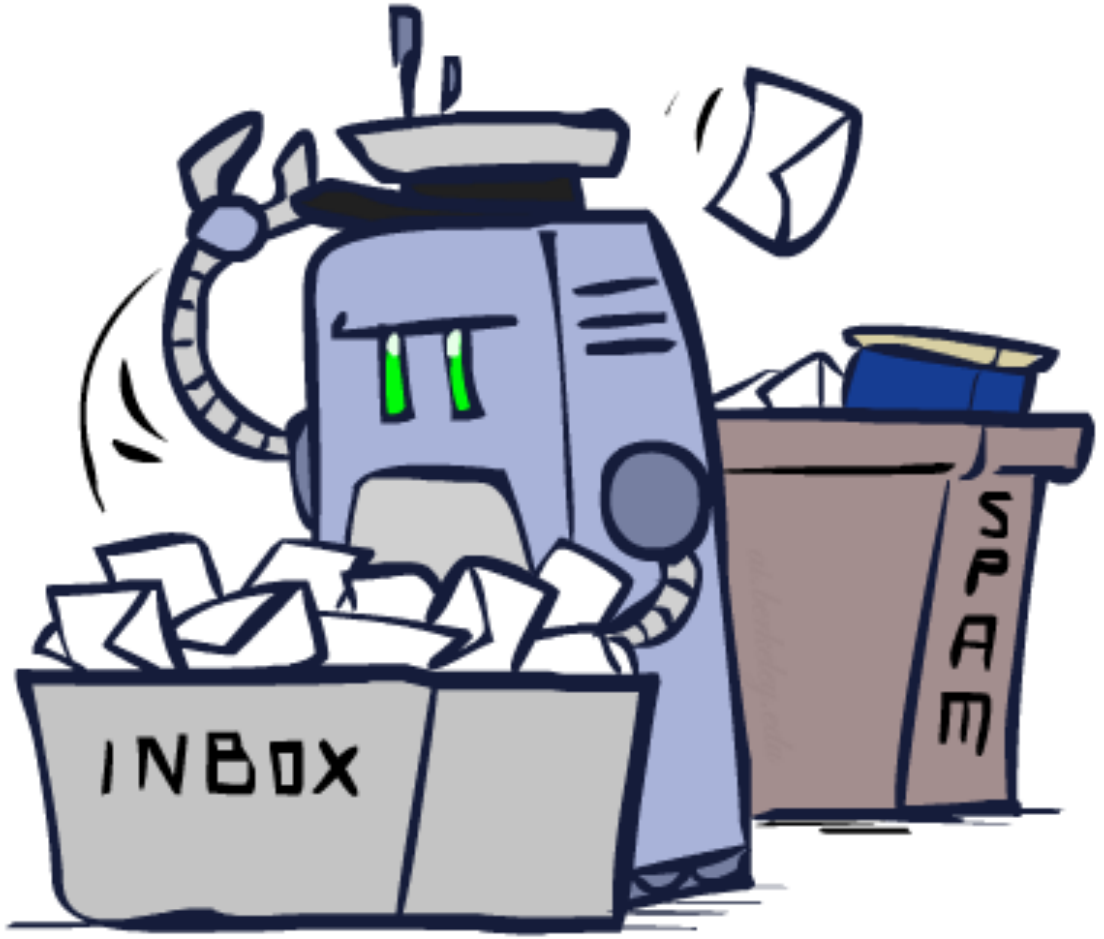
$$\frac{e^{-76.0}}{e^{-76.0} + e^{-80.5}}$$

P(spam | w) = 98.9

Machine Learning

- Up until now: how to use a model to make optimal decisions
- Machine learning: how to acquire a model from data / experience
  - Learning parameters (e.g. probabilities) (as in our warm up example)
  - Learning structure (e.g. BN graphs)
  - …and more! (some in 188, but tons more in 189)
- Today: model-based classification with Naive Bayes

Classification



Example: Spam Filter

- Input: an email
- Output: spam/ham

- To create a spam filter:
  - Get a large collection of example emails, each labeled "spam" or "ham"
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the ham / spam decision
  - Words in text: dear, sir, first, i, must, solicit…
  - Presence of text patterns: $dd, CAPS
  - Whether sender is in your contacts.
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

- To create a digit recognizer:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …


0


1


2

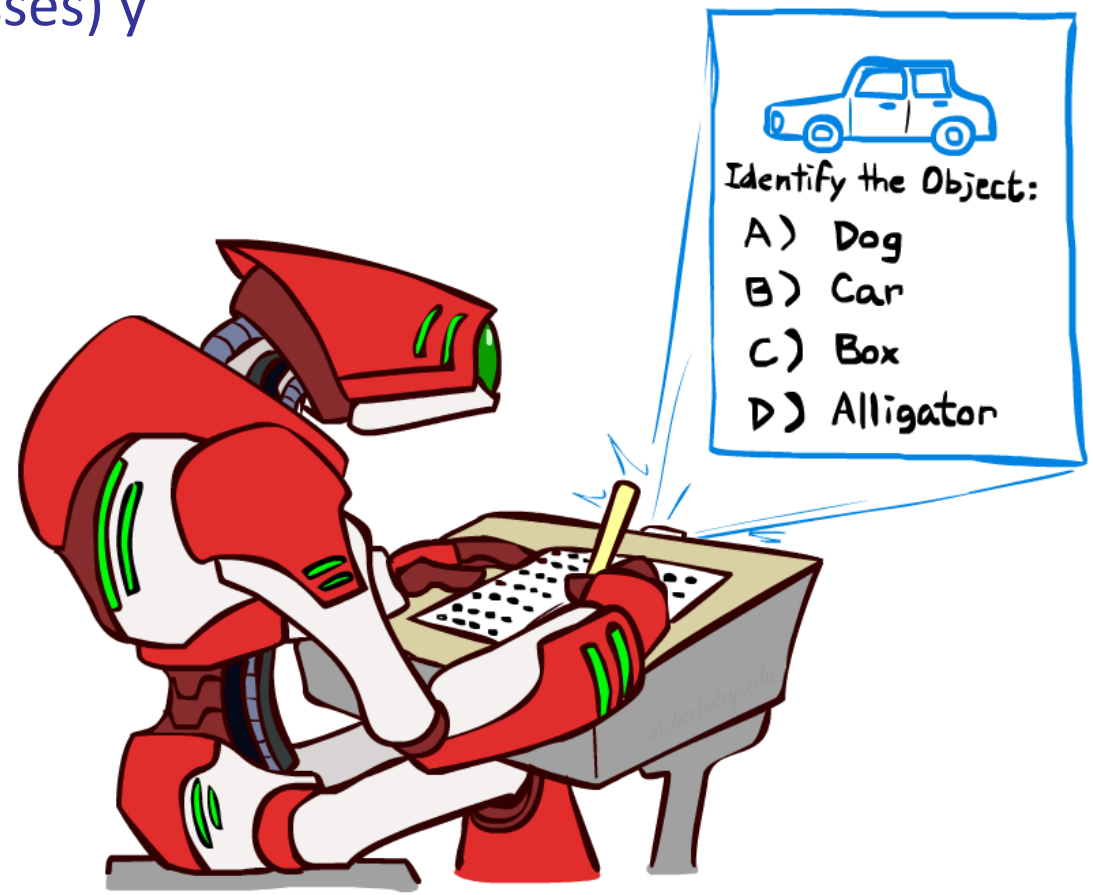

1


??

Other Classification Tasks

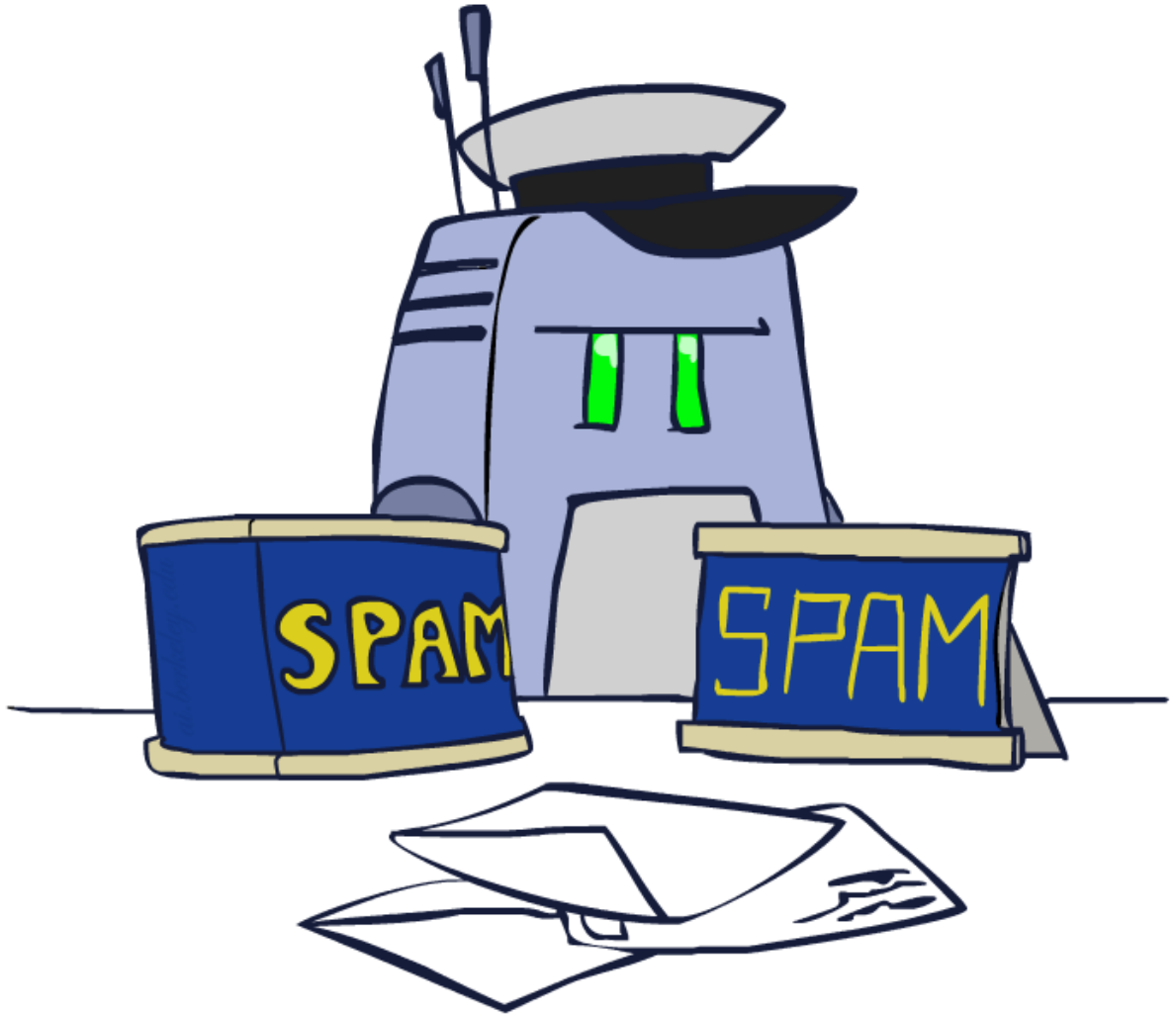
- Classification: given inputs x, predict labels (classes) y

- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - Optical Character Recognition (OCR) (input: images, classes: characters)
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Automatic essay grading (input: document, classes: grades)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Image detection (input: image, class: user name)
  - … many more



- Classification is an important commercial technology!

Model-Based Classification

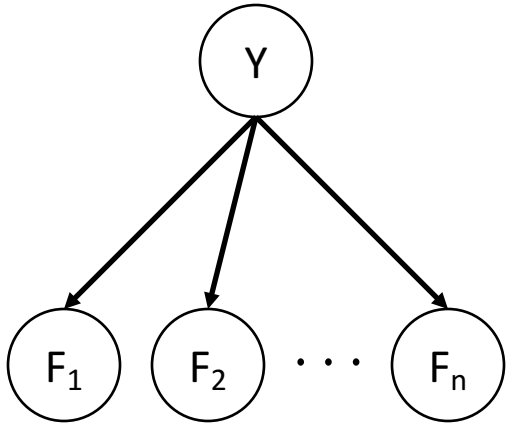


Model-Based Classification

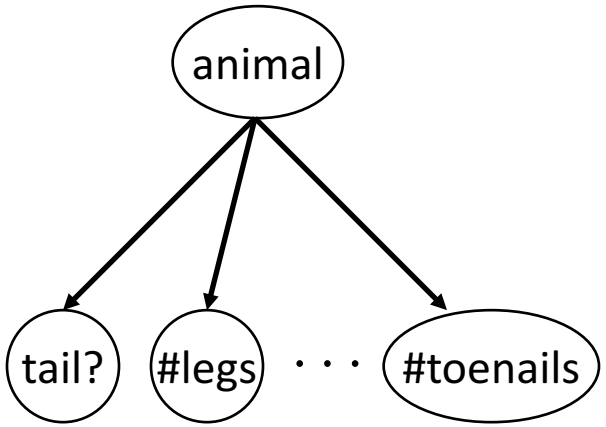
- Model-based approach
  - Build a model (e.g. Bayes' net) where both the label and features are random variables
  - Instantiate any observed features
  - Query for the distribution of the label conditioned on the features
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?
- Not a major challenge:
  - How do we classify based on BN? Already know how to do this (see warmup)

Naïve Bayes

- Naïve Bayes: **Assume** all features are independently determined only by label
  - Label also sometimes called “class”
  - Effectively dodging the challenge of “what structure should the BN have”?

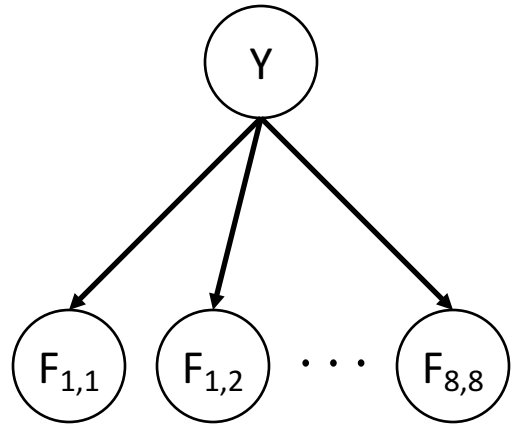


- Model is very simplistic, but often works anyway
  - Why simple? In real data, features are always going to have dependencies!
- Saw an example for emails
  - Let’s consider how we’d model digit recognition before generalizing



Naïve Bayes for Digits

- Naïve Bayes: **Assume** all features are independently determined only by label



- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{1,1} = 0 \;\; F_{1,2} = 0 \;\; F_{1,3} = 0 \;\ldots\; F_{4,4} = 1 \;\ldots\; F_{8,8} = 0 \rangle$$

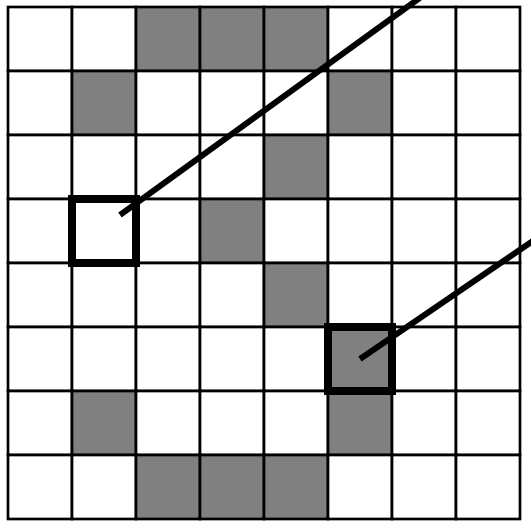
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{1,1} \;\ldots\; F_{8,8}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

Example: Parameters for Naïve Bayes for Digits

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{2,5} = \text{on}|Y)$

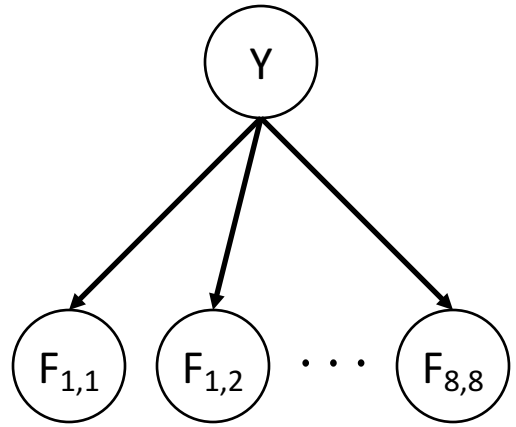
1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{6,3} = \text{on}|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independently determined only by label



- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{1,1} = 0 \;\; F_{1,2} = 0 \;\; F_{1,3} = 0 \;\; \ldots \;\; F_{4,4} = 1 \;\; \ldots \;\; F_{8,8} = 0 \rangle$$

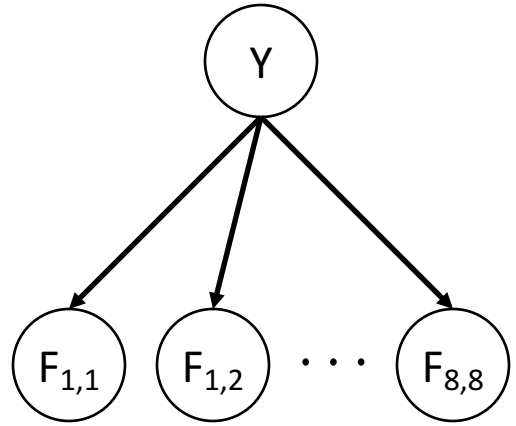
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{1,1} \;\; \ldots \;\; F_{8,8}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn? How many total parameters?

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independently determined only by label



- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{1,1} = 0 \;\; F_{1,2} = 0 \;\; F_{1,3} = 0 \;\ldots\; F_{4,4} = 1 \;\ldots\; F_{8,8} = 0 \rangle$$

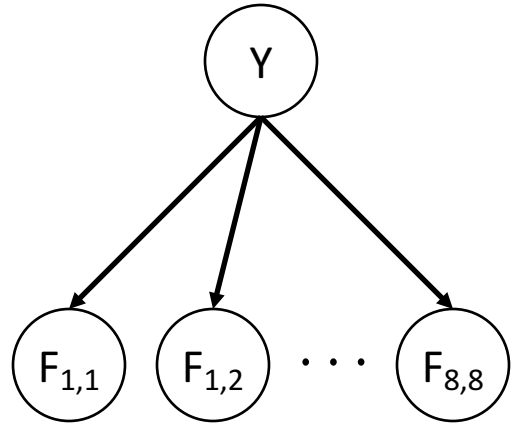
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{1,1} \;\ldots\; F_{8,8}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn? How many total parameters?
  - 65 CPTs: 10 parameters for P(Y). 64 * 2 * 10 parameters for the leaf CPTs. 1,290 total parameters.

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independently determined only by label



- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{1,1} = 0 \;\; F_{1,2} = 0 \;\; F_{1,3} = 0 \;\ldots\; F_{4,4} = 1 \;\ldots\; F_{8,8} = 0 \rangle$$

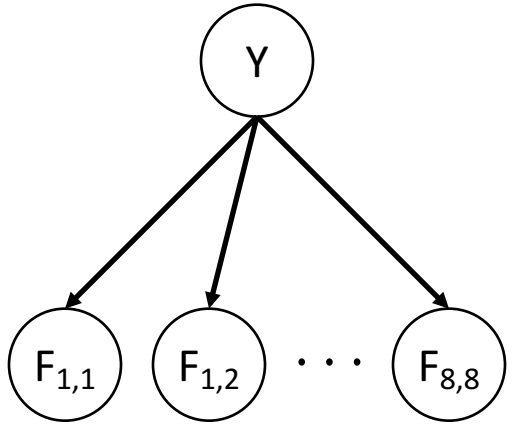
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{1,1} \;\ldots\; F_{8,8}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- With email classification, assumed all leaf CPTs were the same. If we did that for image recognition, our classifier would basically be deciding based on what?

Effect of Equal CPTs

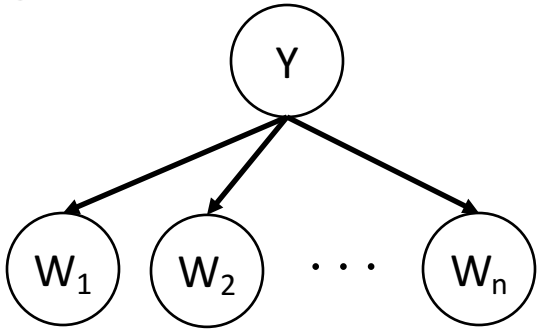
- If all leaf CPTs were the same for digits:
  - Only 30 parameters (instead of 1,290).
  - Classification would be based only on how many pixels are “on” vs. “off”.



- Our text model was a Naïve Bayes model, just like digits model:
  - Assumed featured are conditionally independent given the label (spam, ham)

- Our text model made an additional assumption:
  - Bag-of-words: each W_i is identically distributed (CPTs have same parameters)
  - Reduces number of parameters by a factor of N (where N is the longest recorded email).
  - Called “bag-of-words” because model is insensitive to word order or reordering

- Intuition for why this was ok for text: Features for email are richer!
  - Presence of specific words rather than just binary values.
  - “How many pixels are on, anywhere?” vs. “which words appear, anywhere?”



General Naïve Bayes

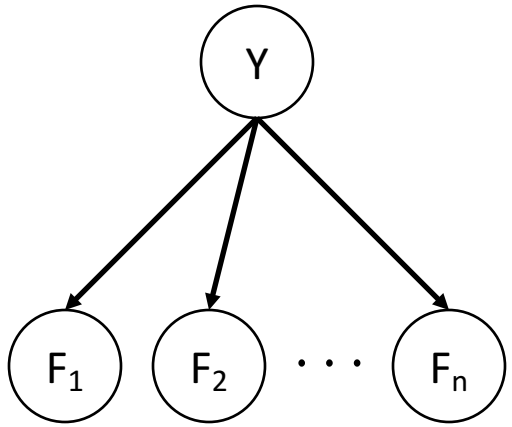
- A general Naive Bayes model:

|Y| parameters

$$P(\mathrm{Y}, \mathrm{F}_1 \ldots \mathrm{F}_n) = P(\mathrm{Y}) \prod_i P(\mathrm{F}_i|\mathrm{Y})$$

|Y| x |F|^n values

n x |F| x |Y| parameters

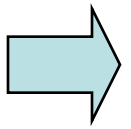


- For each feature we only have to specify how it depends on the class
- Total number of parameters is *linear* in n
  - Bag of words: Constant number of parameters in n.
- Leaf CPTs are not generally the same (but were for our email example)
- As mentioned before: model is very simplistic, but often works anyway

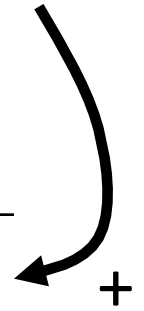
General Inference for Naïve Bayes (a la warmup)

- Goal 1: compute posterior distribution over label variable Y
  - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \dots f_n) = \begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix}$$



$$\begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}$$



$$P(f_1 \dots f_n)$$

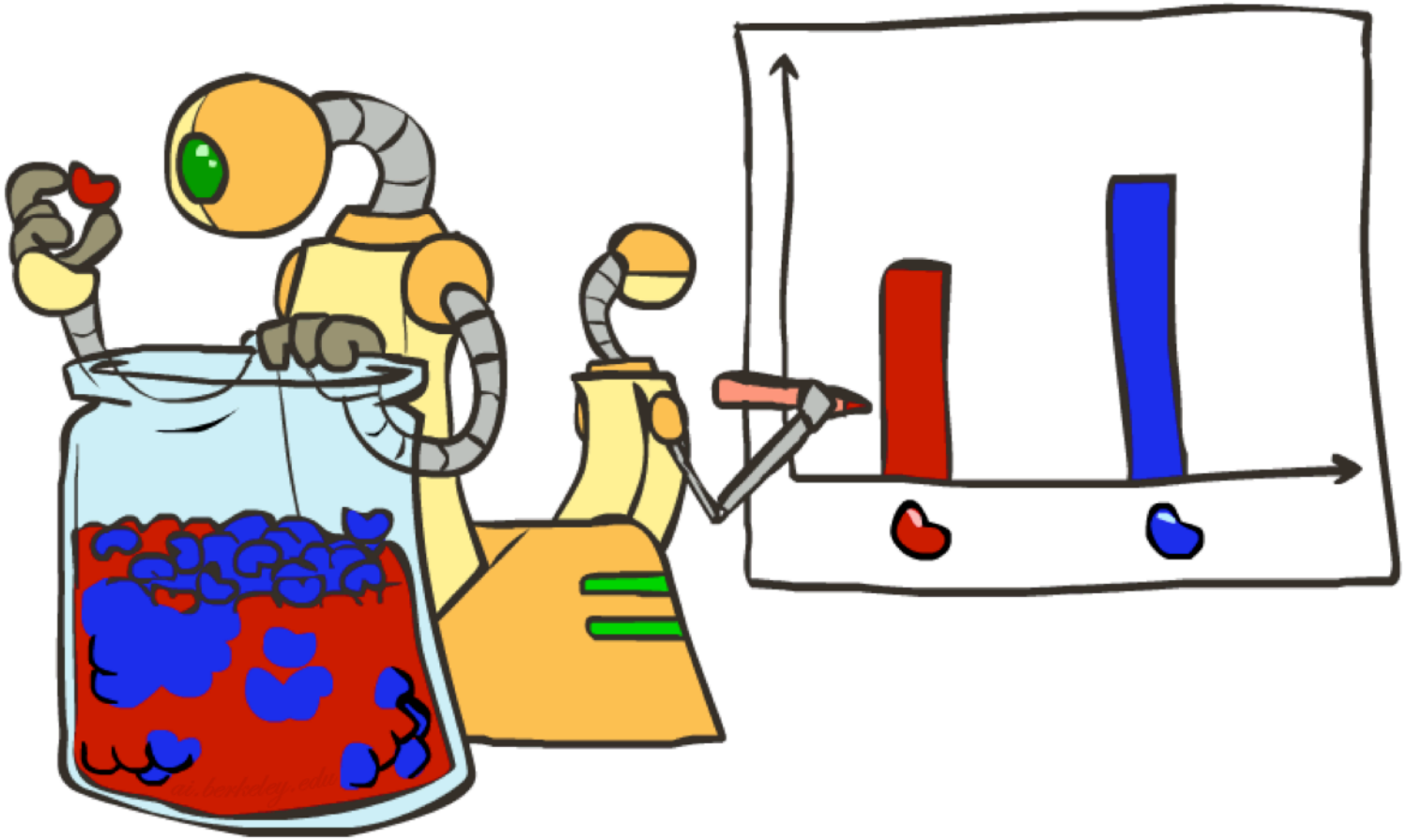
$$P(Y|f_1 \dots f_n)$$

  - Step 2: sum to get probability of evidence
  - Step 3: normalize by dividing Step 1 by Step 2
- Goal 2: determine most likely label Y
  - Simply return Y with highest probability (or log probability)

General Naïve Bayes

- What do we need in order to use Naïve Bayes?
  - Inference method (like our Gary example)
    - Start with a bunch of probabilities: P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1…F_n)$ or log $P(Y|F_1…F_n)$
    - Return Y with the largest probability (or log probability)
  - Estimates of local conditional probability tables [CPTs]
    - P(Y), the prior over labels [root CPT]
    - $P(F_i|Y)$ for each feature (evidence variable) [leaf CPTs]
    - These probabilities are collectively called the *parameters* of the model and denoted by θ
    - Up until now, we assumed these appeared by magic, but **now we'll see how to learn them from training data.**

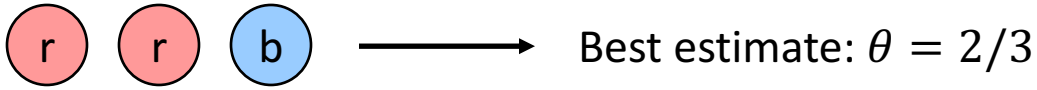
Parameter Estimation



Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation:* ask a human (why is this hard?)
- *Empirically:* use training data (learning!)
  - E.g.: for each outcome x, look at the *empirical rate* of that value
  - Suppose we have a coin where heads is marked red, and tails is marked blue. Let θ be the chance that it comes up red. Our goal: estimate probability that coin comes up heads/red.

$$\theta = P(c = red)$$



  - Above, we've picked the θ that **maximizes the likelihood of the data.**

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{ML}(\text{r}) = \theta = 2/3$$

Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation:* ask a human (why is this hard?)
- *Empirically:* use training data (learning!)
  - E.g.: for each outcome x, look at the *empirical rate* of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{\text{ML}}(\text{r}) = 2/3$$

$$\theta = P(c = red)$$

  - P_{ML} is the estimate that maximizes the *likelihood of the data*

$$L(x, \theta) = \prod_i P_\theta(x_i)$$

$$L(x, \theta) = P_\theta(r, r, b) = \theta^2 \cdot (1 - \theta) = \theta^2 - \theta^3$$

$$\frac{dP}{d\theta} = 2\theta - 3\theta^2$$

min at: $\theta = 0$
max at: $\theta = \frac{2}{3}$

$$0 = \theta(2 - 3\theta)$$

Parameter Estimation

- Estimating the distribution of a random variable
- *Elicitation:* ask a human (why is this hard?)
- *Empirically:* use training data (learning!)
  - E.g.: for each outcome x, look at the *empirical rate* of that value:

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

$$P_{\text{ML}}(\text{r}) = 2/3$$

$$\theta = P(c = red)$$

  - P_{ML} is the estimate that maximizes the *likelihood of the data*
  - There are possibilities other than P_{ML} - e.g. taking into account our intuition that coins are usually fair. More later.

Maximum Likelihood Estimation Dangers

- Example: Maximum Likelihood for Spam Classification
  - For each word w, need to estimate probability it appears in ham and spam emails:
    - Set $P(ham, w_i = w) = \frac{\text{number of times w in appears in a ham email}}{\text{total number of words in all ham emails}}$
    - Set $P(spam, w_i = w) = \frac{\text{number of times w in appears in a spam email}}{\text{total number of words in all spam emails}}$
  - Example, suppose we have 3 ham emails and 3 spam emails, as below:

1: hey horse 2: horse, are you awake 3: horse, i am a horse

ham

$$P(w_i = \text{horse}|ham) = \frac{4}{11}$$

$$P(w_i = \text{discount}|ham) = 0$$

1: buy discount pills 2: discount pills for free 3: low price pills for horse

spam

$$P(w_i = \text{horse}|spam) = \frac{1}{12}$$

$$P(w_i = \text{discount}|ham) = \frac{2}{12}$$

Maximum Likelihood Estimation Dangers

- Naively, we could just compute parameters from relative frequencies.
  - Example, suppose we have 3 ham emails and 3 spam emails, as below:

hey horse horse, are you awake horse, i am a horse

ham

$$P(w_i = \text{horse}|ham) = \frac{4}{11}$$

$$P(w_i = \text{discount}|ham) = 0$$

buy discount pills discount pills for free low price pills for horse

spam

$$P(w_i = \text{horse}|spam) = \frac{1}{12}$$

$$P(w_i = \text{discount}|ham) = \frac{2}{12}$$

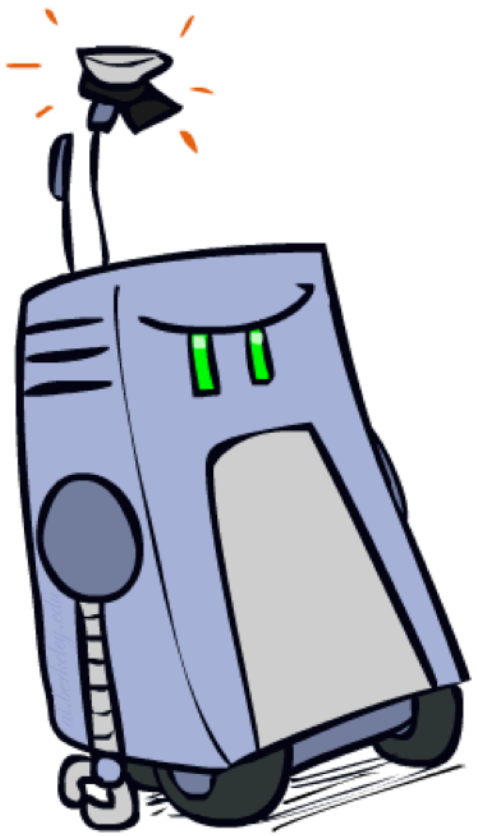
- Observation: During inference, the thing that matters is the ratio of the likelihood of the possible classes.
  - If any word has zero frequency in a corpus, that ratio is zero (or infinity), acts as a perfect shibboleth.

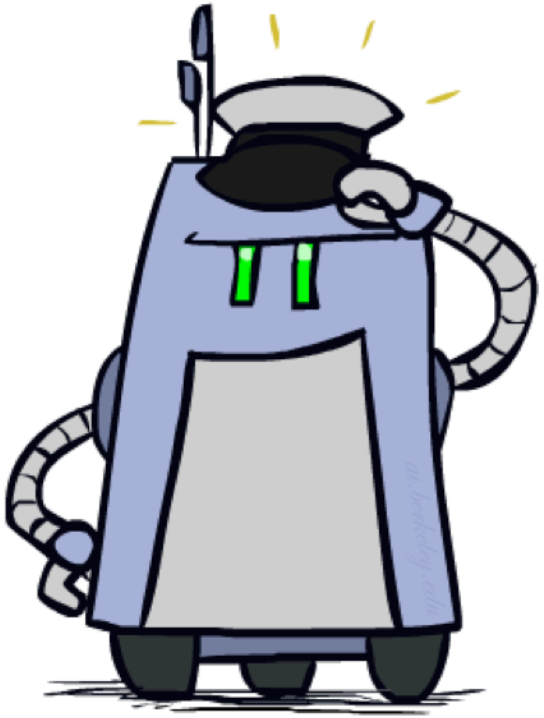
$$P(spam|w_1 \dots w_n) = P(spam)P(w_1|spam)P(w_2|spam) \dots P(w_n|spam)$$

$$P(ham|w_1 \dots w_n) = P(ham)\ P(w_1|ham)\ P(w_2|ham) \dots\ P(w_n|ham)$$

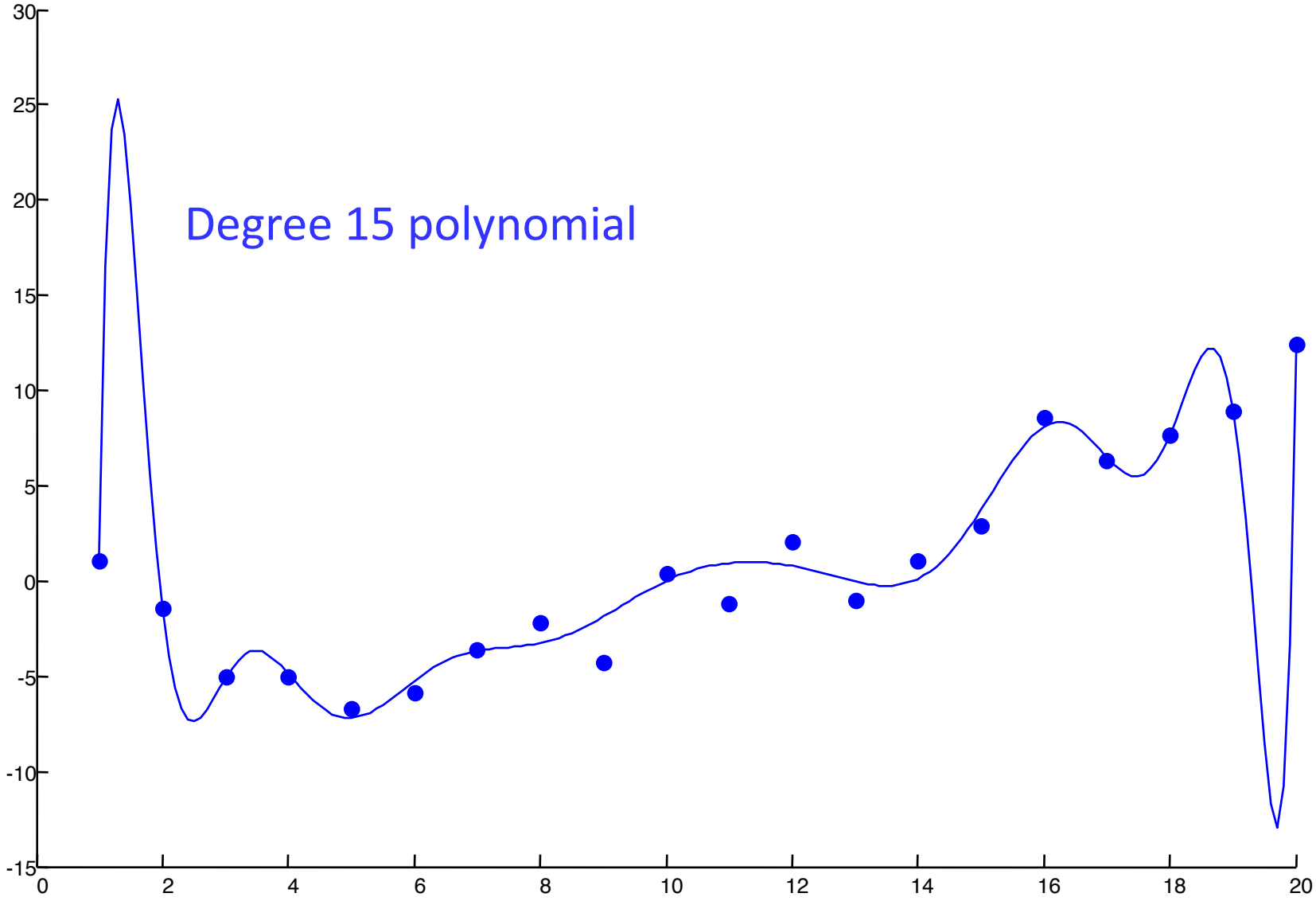
$$\frac{P(w_i = \text{discount}|spam)}{P(w_i = \text{discount}|ham)} = \infty$$

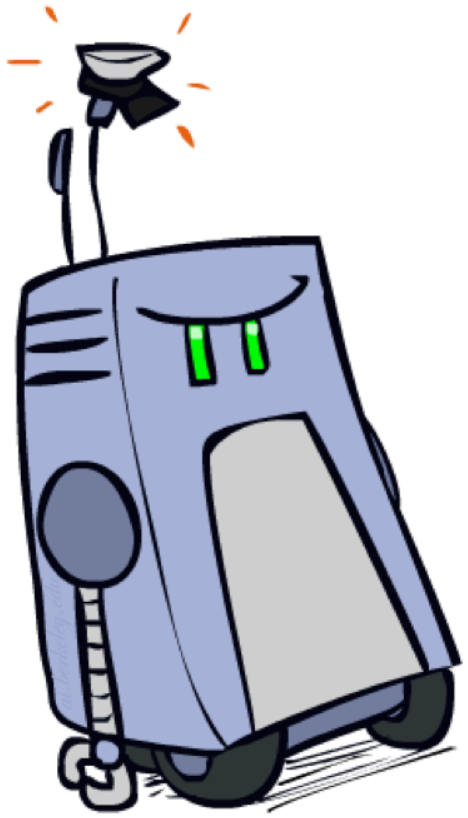
Generalization and Overfitting





Overfitting (Intuitively)





Example: Overfitting Classifier for Digits

$P(\text{features}, C = 2)$

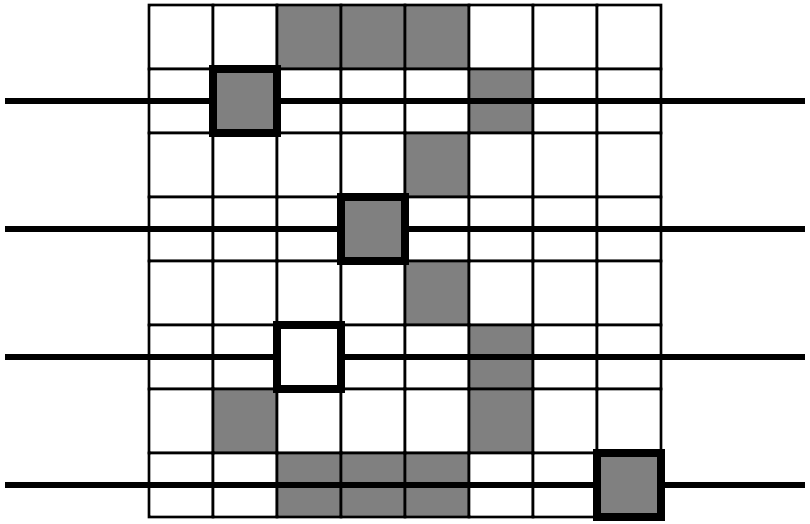
$P(C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

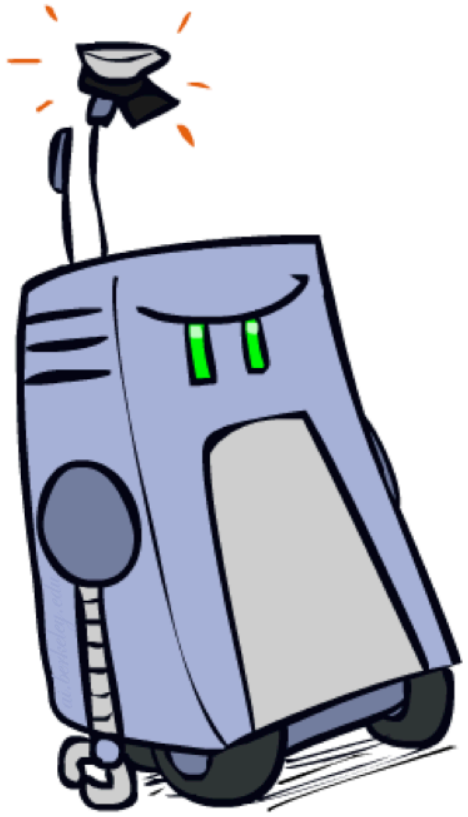
$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$



2 wins!!

Example 2: Overfitting Classifier for Email

- Posteriors determined by *relative* probabilities (odds ratios):

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

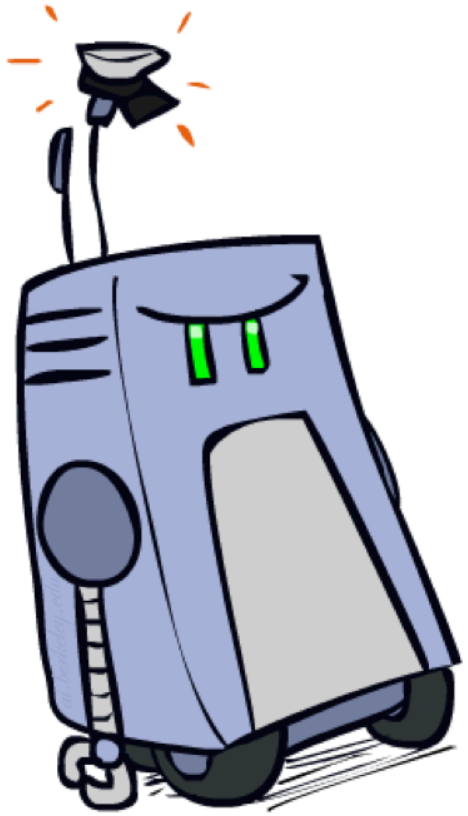
```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

If we see “nicely”, then we decide it is definitely ham!

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

If we see “screens”, then we decide it is definitely spam!



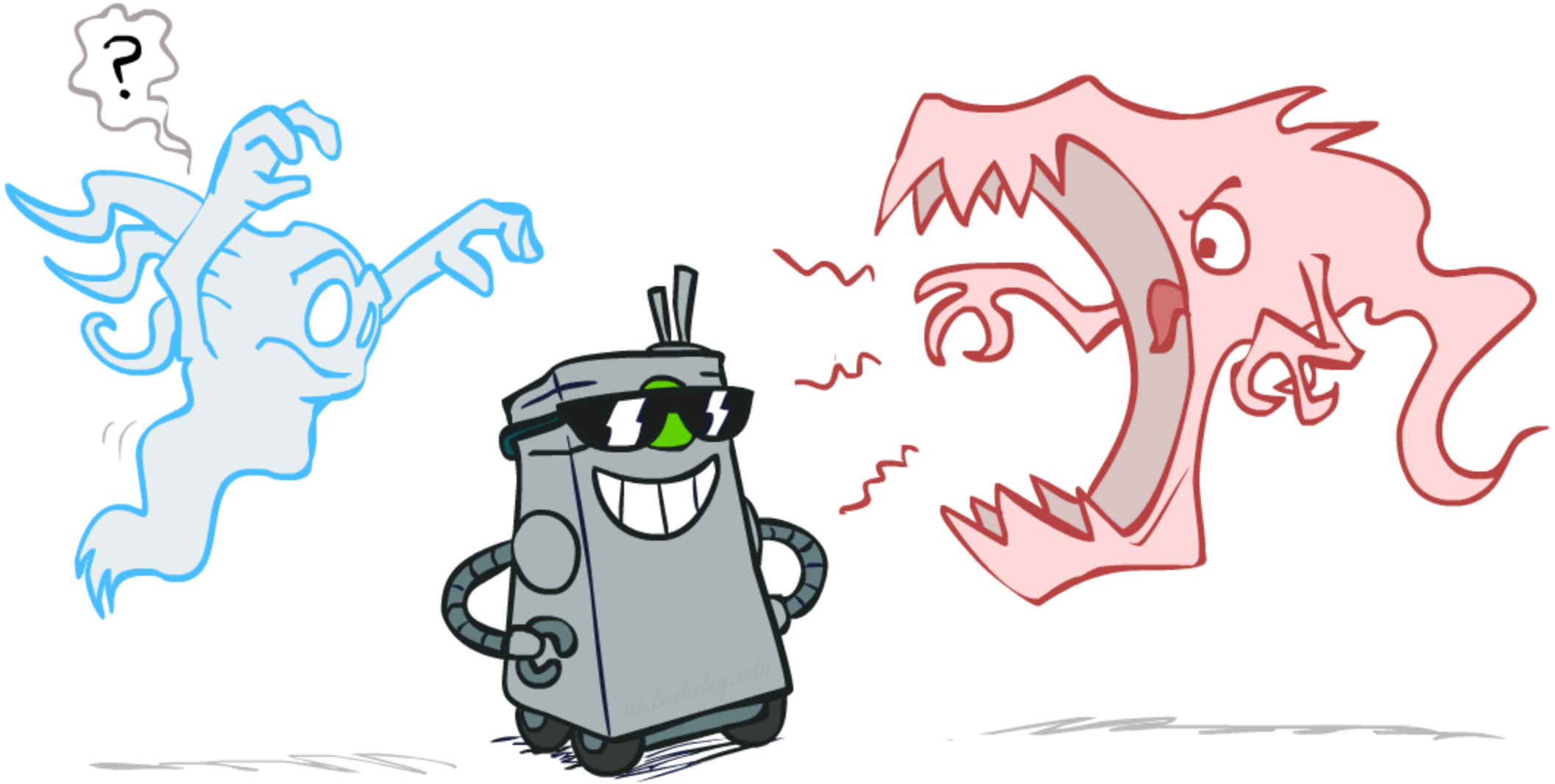
Generalization and Overfitting

- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (8,8) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - What about all the words that don't occur in the training set at all?
  - In general, we can't go around giving unseen events zero probability

- As an extreme case, imagine using the entire email as the only feature
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough

- To generalize better: we need to smooth or regularize the estimates

Smoothing



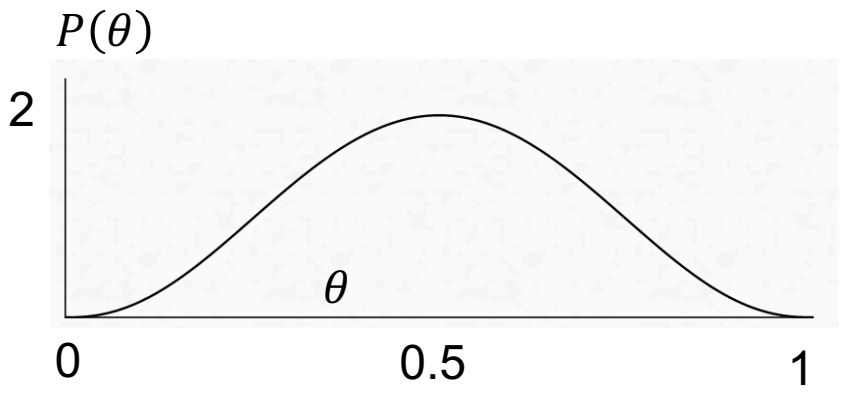
Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates

$$\theta_{ML} = \arg\max_{\theta} P(\mathbf{X}|\theta)$$
$$= \arg\max_{\theta} \prod_i P_\theta(X_i)$$

$$\Rightarrow \quad P_{\mathsf{ML}}(x) = \frac{\mathsf{count}(x)}{\mathsf{total\ samples}}$$

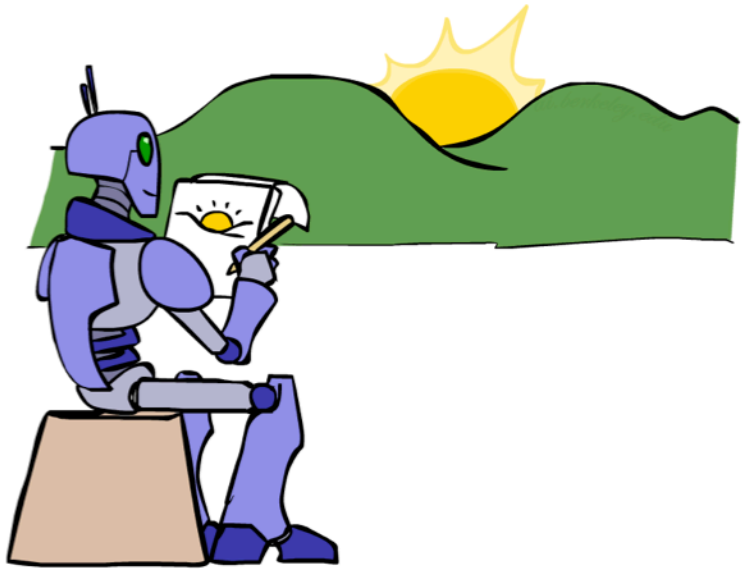
- Another option is to consider the most likely parameter value given the data

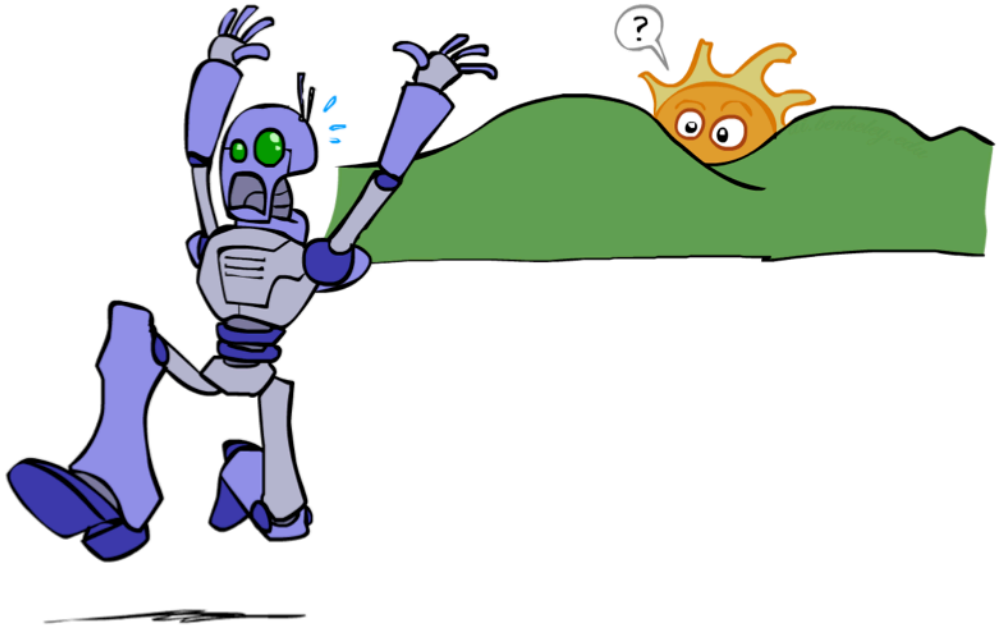


$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

$\Rightarrow$????

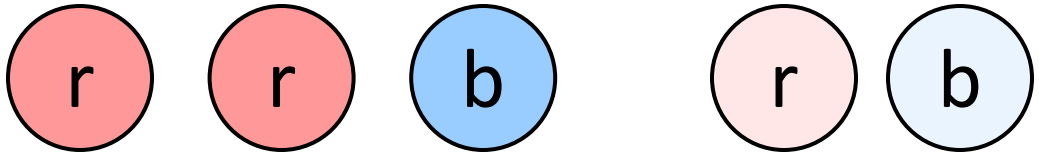
Unseen Events





Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did



$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

$$P_{ML}(X) =$$

$$P_{LAP}(X) =$$

  - Can derive this estimate with *Dirichlet priors* (see cs281a)

Laplace Smoothing

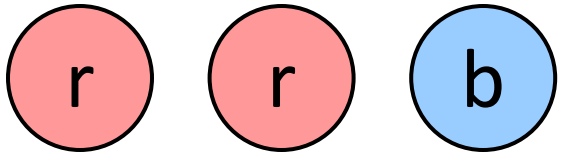
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - What's Laplace with k = 0?
  - k is the strength of the prior

- Laplace for conditionals:
  - Smooth each condition independently:

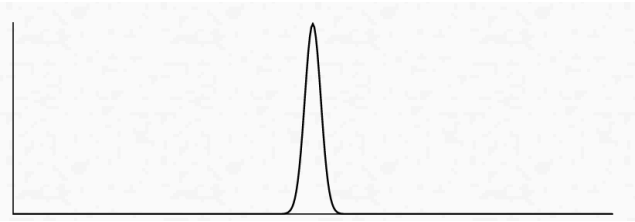
$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$



$$P_{LAP,0}(X) =$$

$$P_{LAP,1}(X) =$$

$$P_{LAP,100}(X) =$$



Estimation: Linear Interpolation*

- In practice, Laplace often performs poorly for P(X|Y):
  - When |X| is very large
  - When |Y| is very large
- Another option: linear interpolation
  - Also get the empirical P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from the empirical P(X)

$$P_{LIN}(x|y) = \alpha \hat{P}(x|y) + (1.0 - \alpha)\hat{P}(x)$$

  - What if α is 0? 1?
- For even better ways to estimate parameters, as well as details of the math, see cs281a, cs288

Real NB: Smoothing

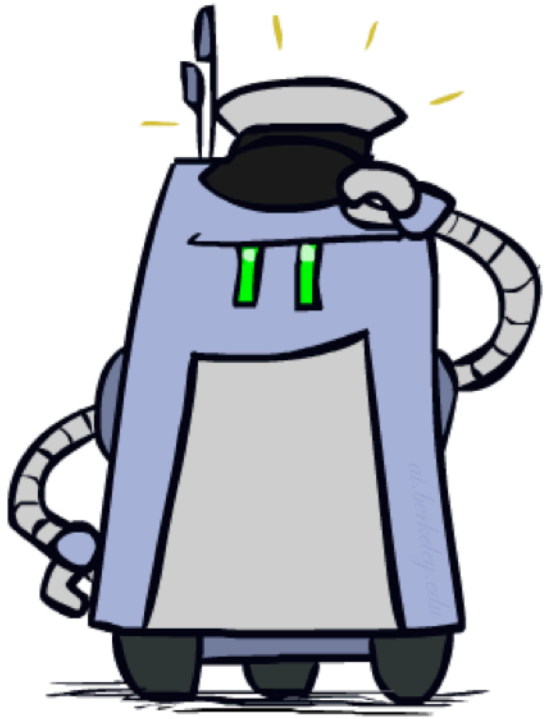
- For real classification problems, smoothing is critical
- New odds ratios:

$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
helvetica : 11.4
seems     : 10.8
group     : 10.2
ago       :  8.4
areas     :  8.3
...
```

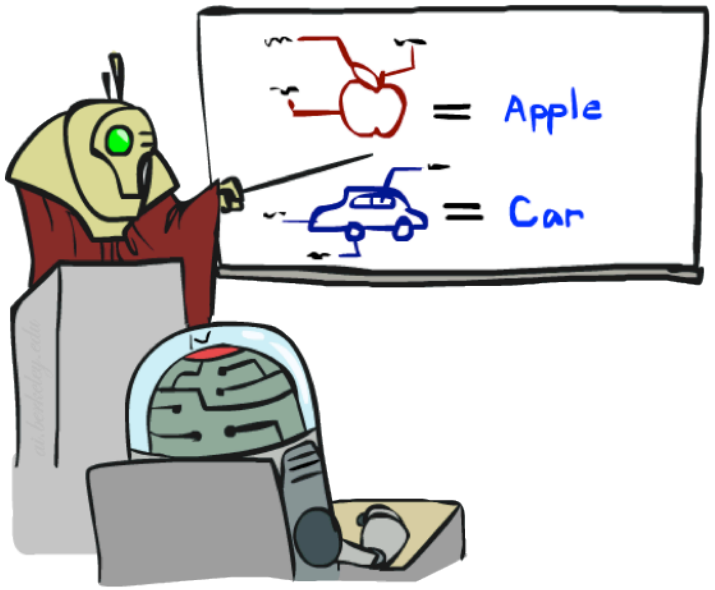
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

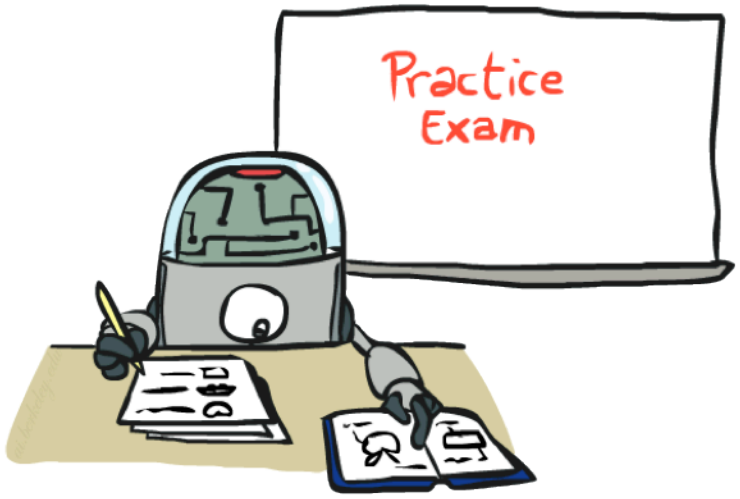
```
verdana : 28.8
Credit  : 28.4
ORDER   : 27.2
<FONT>  : 26.9
money   : 26.5
...
```

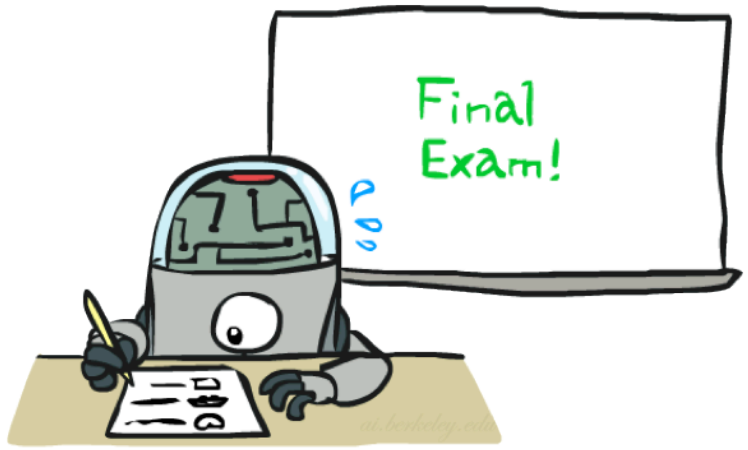


Do these make more sense?

Training and Testing

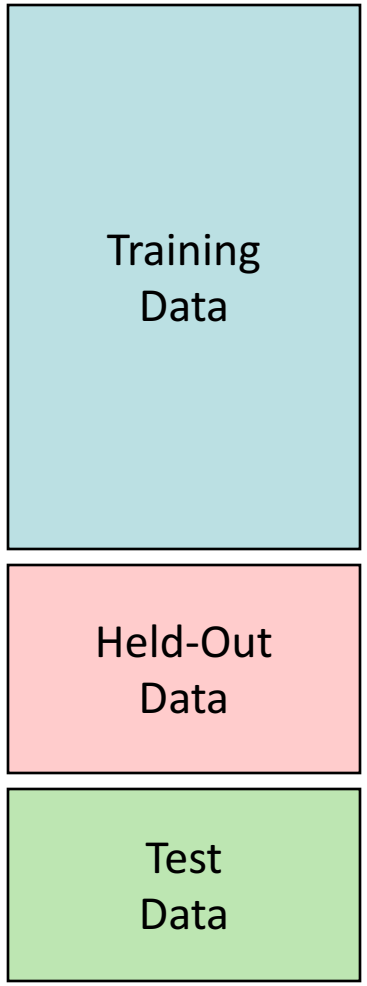




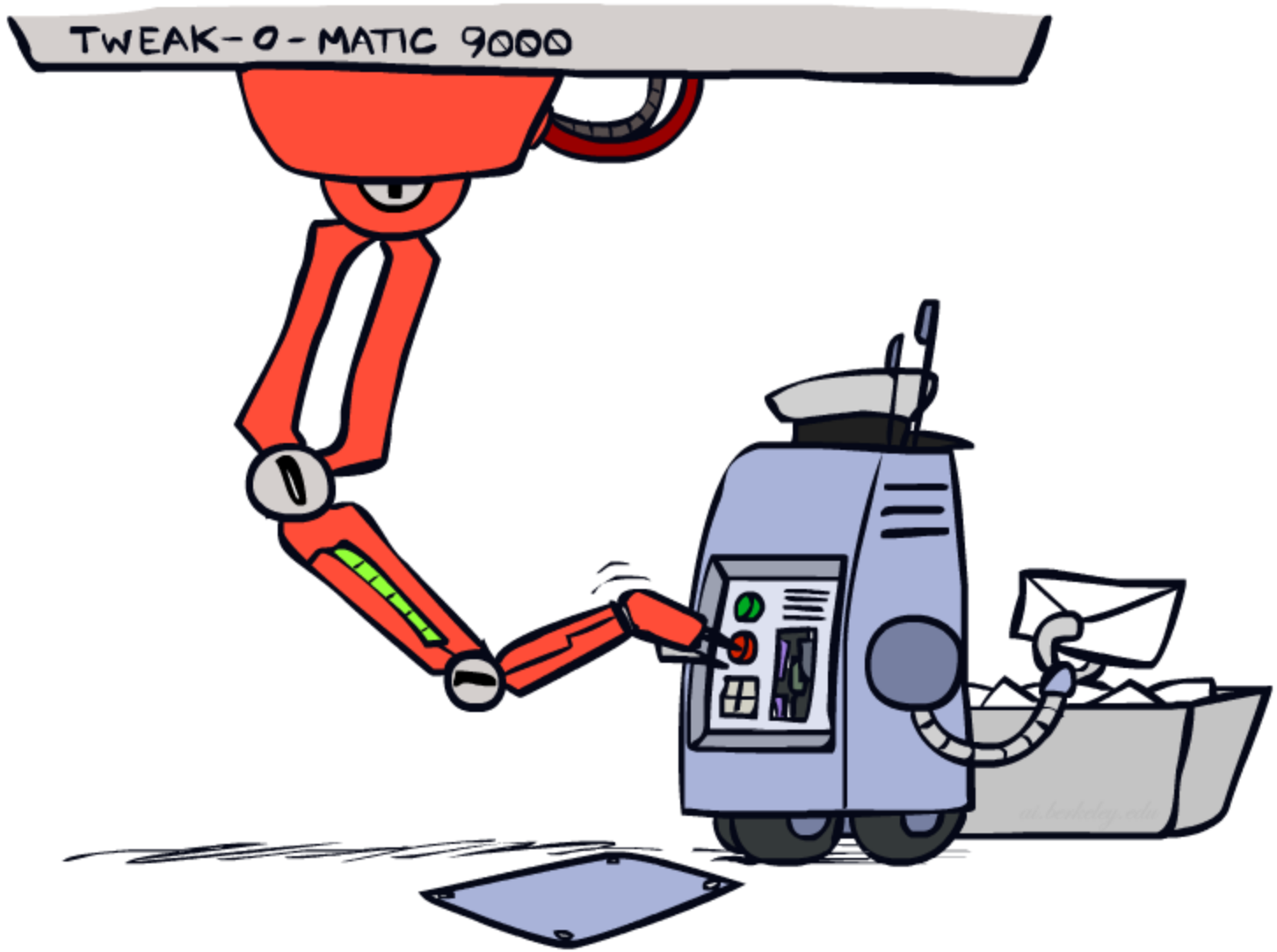


Important Concepts

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- **Features**: attribute-value pairs that characterize each instance
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - **Tune** hyperparameters on held-out set
  - Compute accuracy on test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well
  - We'll investigate overfitting and generalization formally in a few lectures

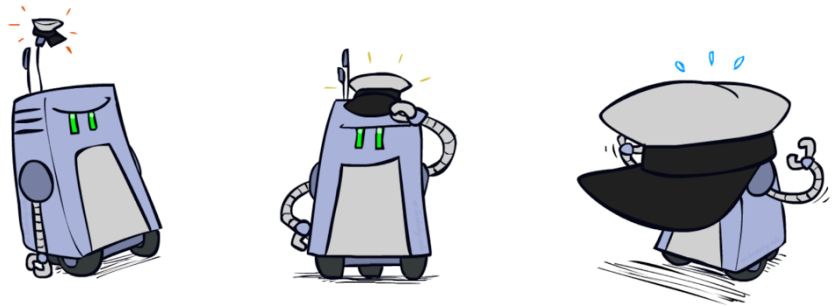


Tuning

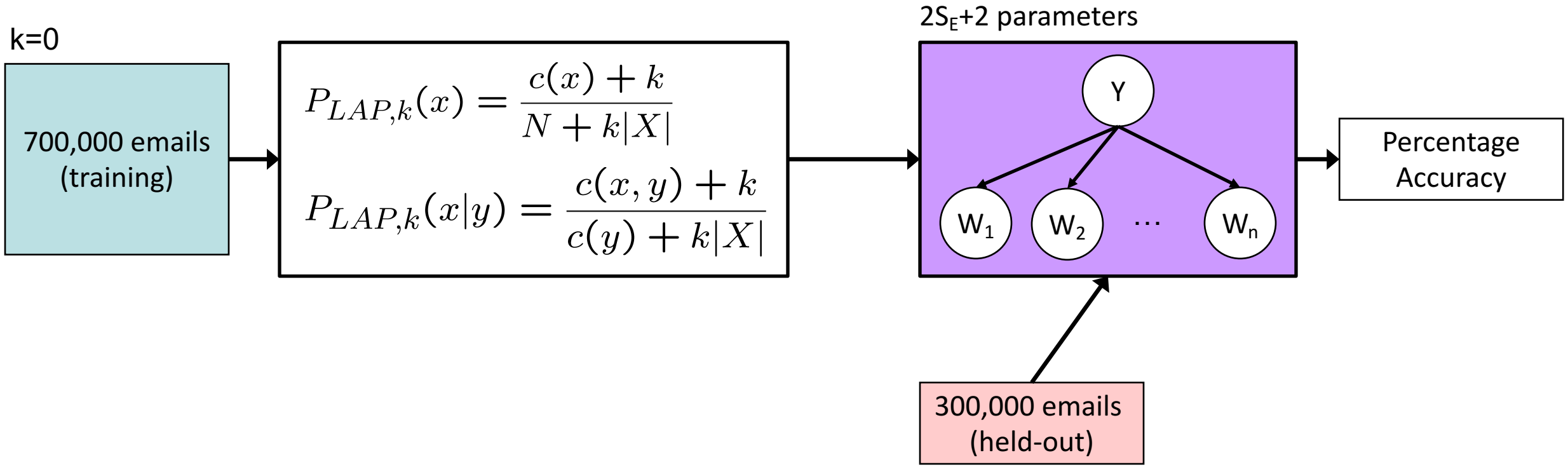


Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data



Tuning on Held-Out Data (Example)



- Repeat process above for k = 1, 2, …
  - Each time, generating $2S_E+2$ parameters.
- Return set of parameters with best accuracy on held-out set.

Tuning on Held-Out Data (Example)

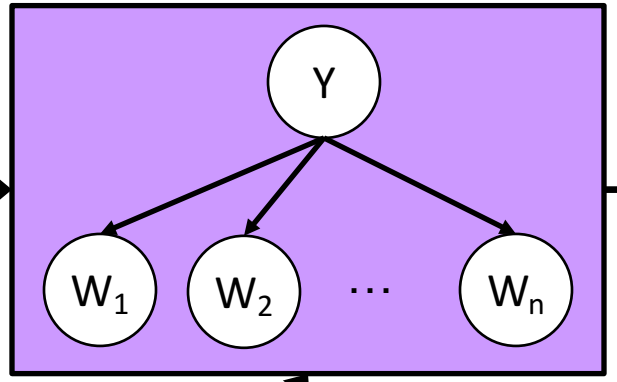
k=0

700,000 emails (training)

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$

$2S_E+2$ parameters



Percentage Accuracy

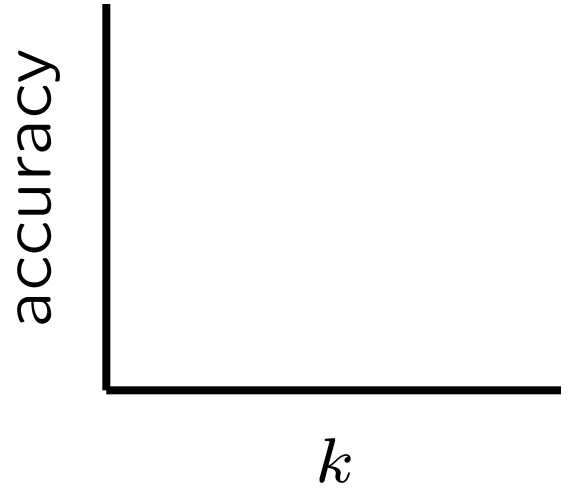
300,000 emails (held-out)

Test Data

- Why don't we test the percentage accuracy on the training data? The test data?
- How many total CPT parameters will be generated if we try out k=0, 1, 2, …, 99?

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data



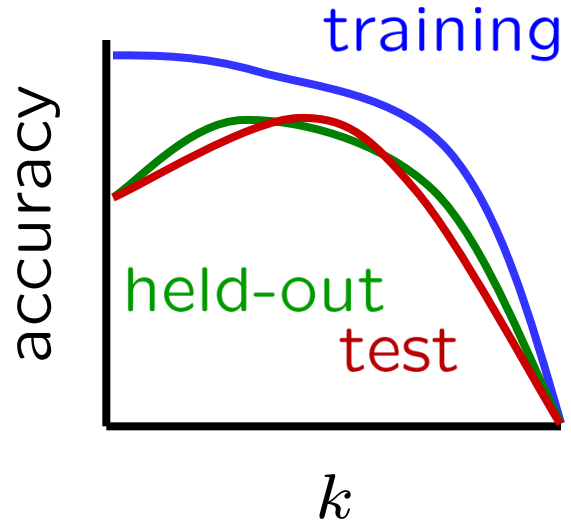
Draw three curves representing the accuracy of our classified on:

- Training data
- Hold-out data

Draw as a function of k. Assume the prior is "reasonable".

Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameters, train and test on the held-out data
  - Choose the best value and do a final test on the test data

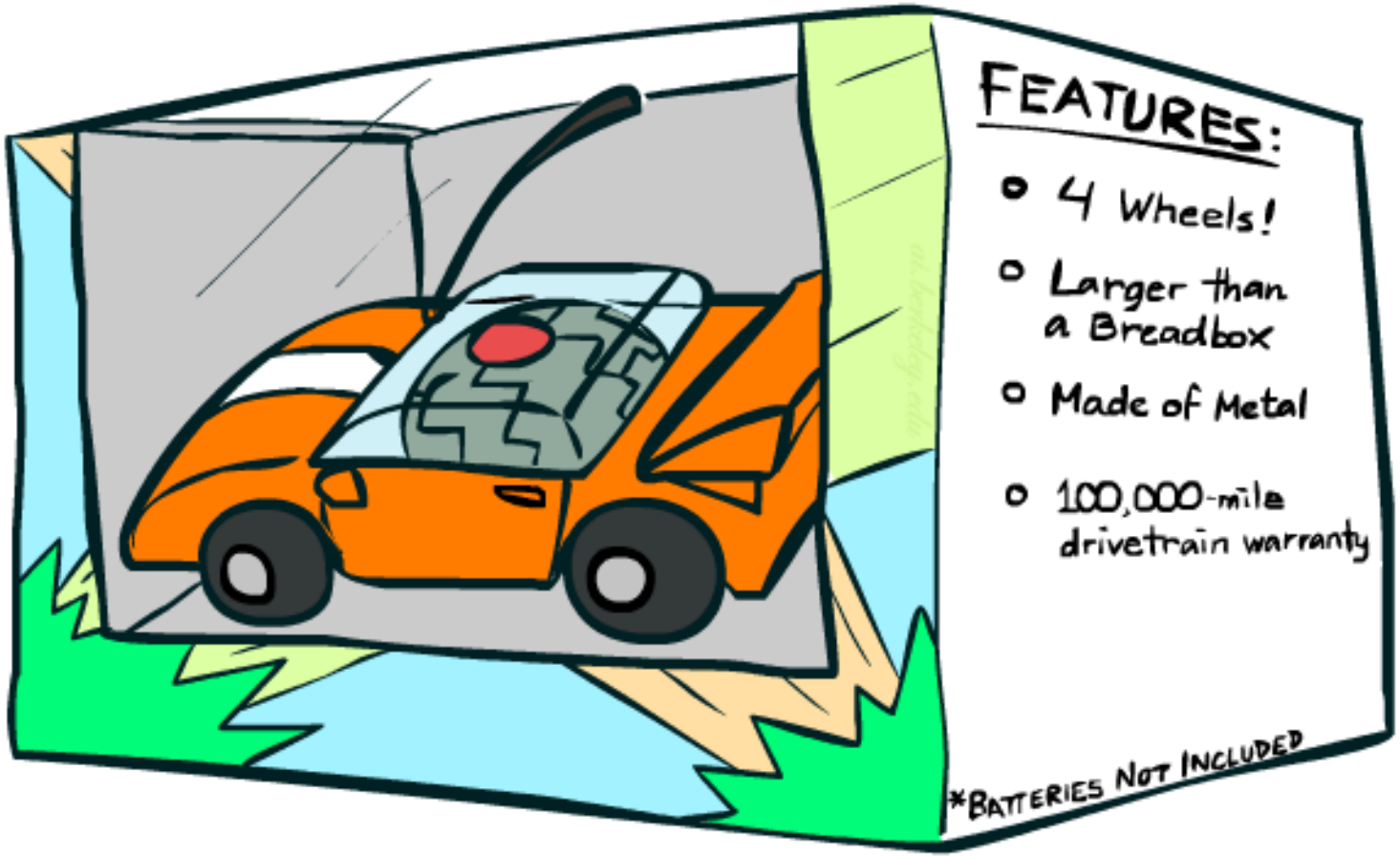


Draw three curves representing the accuracy of our classified on:

- Training data
- Hold-out data

Draw as a function of k. Assume the prior is "reasonable".

Features (time permitting)



Errors, and What to Do

- Examples of errors

```
Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the
latest version of OmniPage Pro, for just $99.99* - the regular
list price is $499! The most common question we've received
about this offer is - Is this genuine? We would like to assure
you that this offer is authorized by ScanSoft, is genuine and
valid. You can get the . . .
```

```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  http://www.amazon.com/apparel

and see the prominent link for the $30 offer. All details are
there. We hope you enjoyed receiving this message. However, if
you'd rather not receive future e-mails announcing new store
launches, please click . . .
```

What to Do About Errors?

- Need more features– words aren't enough!
  - Have you emailed the sender before?
  - Have 1K other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Can add these information sources as new variables in the NB model
- Next class we'll talk about classifiers which let you easily add arbitrary features more easily

Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them

Baselines

- First step: get a baseline
  - Baselines are very simple “straw man” procedures
  - Help determine how hard the task is
  - Help know what a “good” accuracy is

- Weak baseline: most frequent label classifier
  - Gives all test instances whatever label was most common in the training set
  - E.g. for spam filtering, might label everything as ham
  - Accuracy might be very high if the problem is skewed
  - E.g. if calling everything “ham” gets 66%, then a classifier that gets 70% isn’t very good…

- For real research, usually use previous work as a (strong) baseline

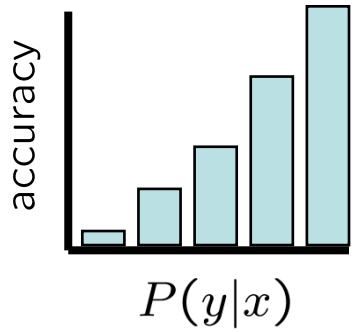
Confidences from a Classifier

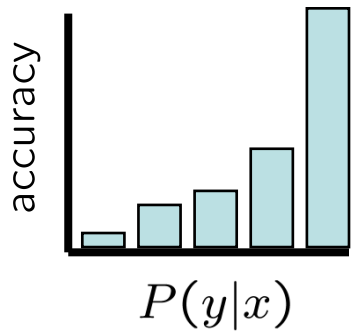
- The confidence of a probabilistic classifier:
  - Posterior over the top label

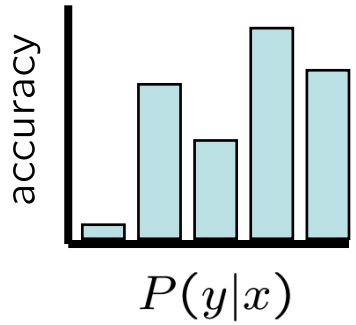
$$\text{confidence}(x) = \max_y P(y|x)$$

  - Represents how sure the classifier is of the classification
  - Any probabilistic model will have confidences
  - No guarantee confidence is correct

- Calibration
  - Weak calibration: higher confidences mean higher accuracy
  - Strong calibration: confidence predicts accuracy rate
  - What's the value of calibration?



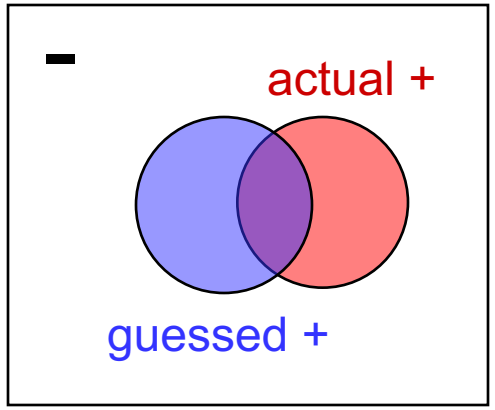




Next Time: Perceptron!

Precision vs. Recall

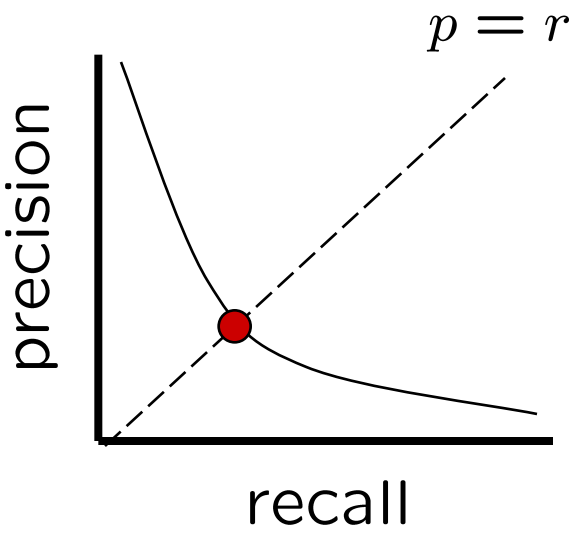
- Let's say we want to classify web pages as homepages or not
  - In a test set of 1K pages, there are 3 homepages
  - Our classifier says they are all non-homepages
  - 99.7 accuracy!
  - Need new measures for rare positive events



- Precision: fraction of guessed positives which were actually positive
- Recall: fraction of actual positives which were guessed as positive
- Say we guess 5 homepages, of which 2 were actually homepages
  - Precision: 2 correct / 5 guessed = 0.4
  - Recall: 2 correct / 3 true = 0.67
- Which is more important in customer support email automation?
- Which is more important in airport face recognition?

Precision vs. Recall

- Precision/recall tradeoff
  - Often, you can trade off precision and recall
  - Only works well with weakly calibrated classifiers



- To summarize the tradeoff:
  - Break-even point: precision value when p = r
  - F-measure: harmonic mean of p and r:

$$F_1 = \frac{2}{1/p + 1/r}$$