CS 188: Artificial Intelligence

Naïve Bayes

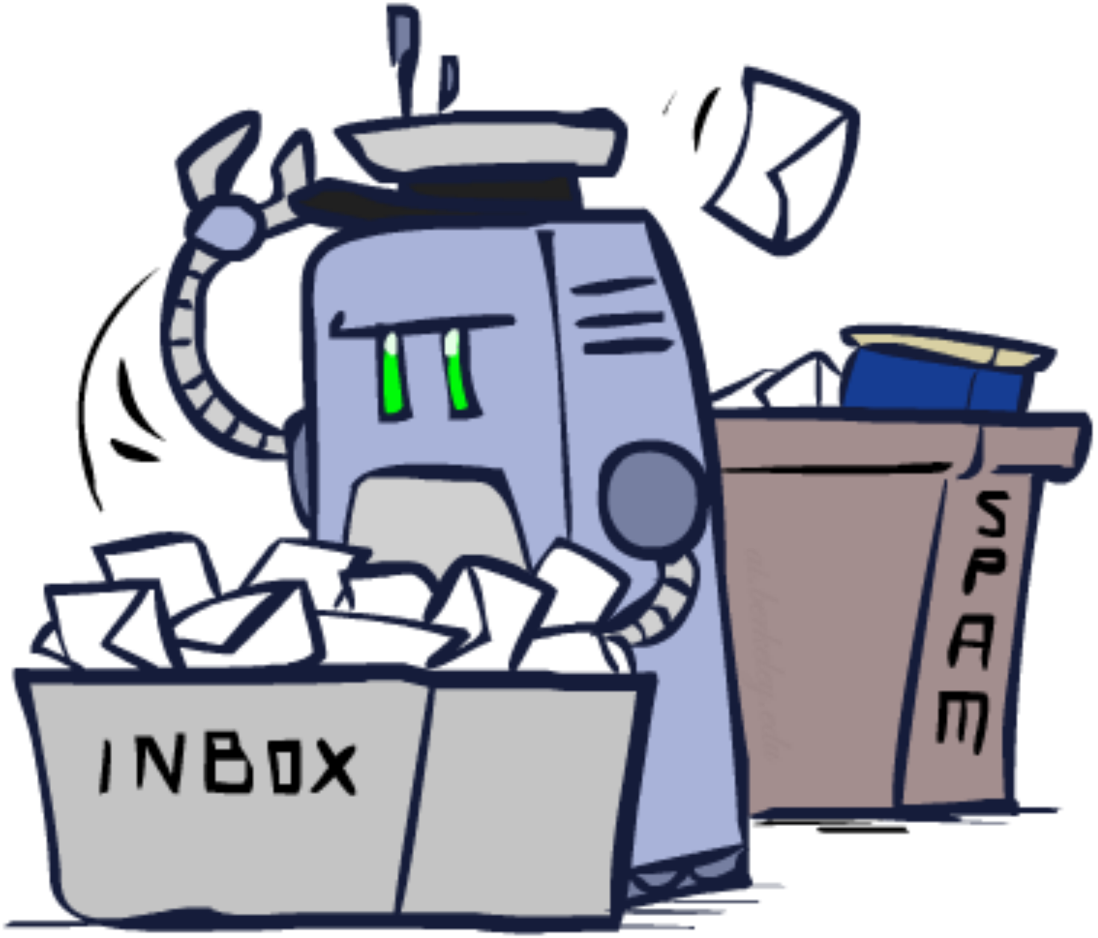
Spring 2023

[These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Machine Learning

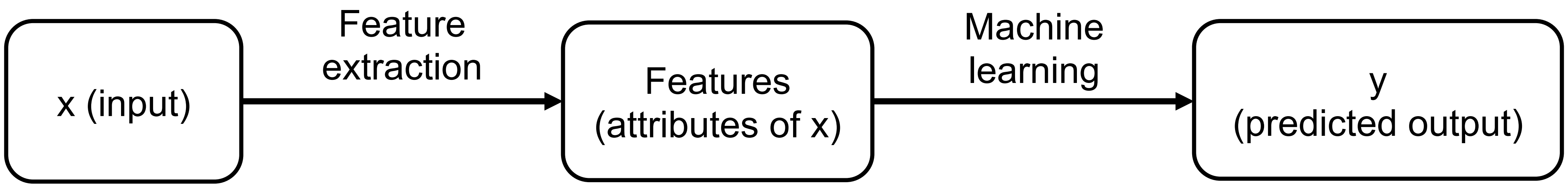
- Up until now: how use a model to make optimal decisions
- Machine learning: how to acquire a model from data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering, neural nets)
- Today: model-based classification with Naive Bayes

Classification



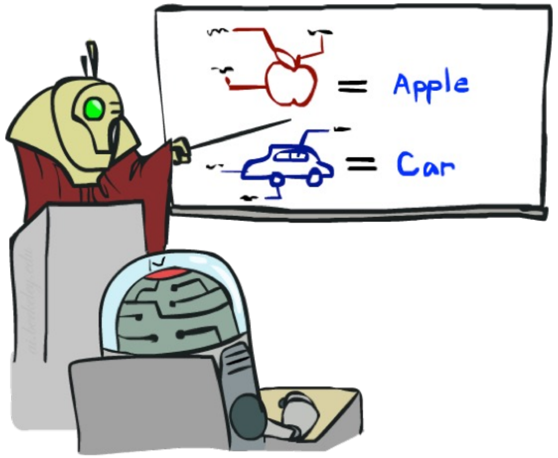
Classification and Machine Learning

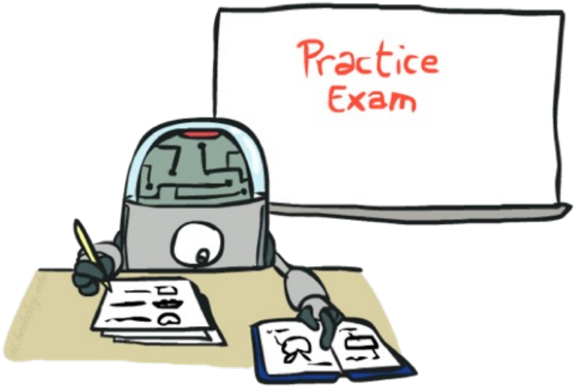
- Dataset: each data point, x, is associated with some label (aka class), y
- Goal of classification: given inputs x, write an algorithm to predict labels y
- Workflow of classification process:
  - Input is provided to you
  - Extract **features** from the input: attributes of the input that characterize each x and hopefully help with classification
  - Run some machine learning algorithm on the features: today, Naïve Bayes
  - Output a predicted label y



Training and Machine Learning

- Big idea: ML algorithms learn patterns between features and labels from *data*
  - You don't have to reason about the data yourself
  - You're given **training data**: lots of example datapoints and their actual labels





Training: Learn patterns from labeled data, and periodically test how well you're doing

Eventually, use your algorithm to predict labels for unlabeled data

Example: Spam Filter

- Input: an email
- Output: spam/ham

- Setup:
  - 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: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts, WidelyBroadcast
  - …

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

- Setup:
  - 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
  - …
  - Features are increasingly induced rather than crafted

0

1

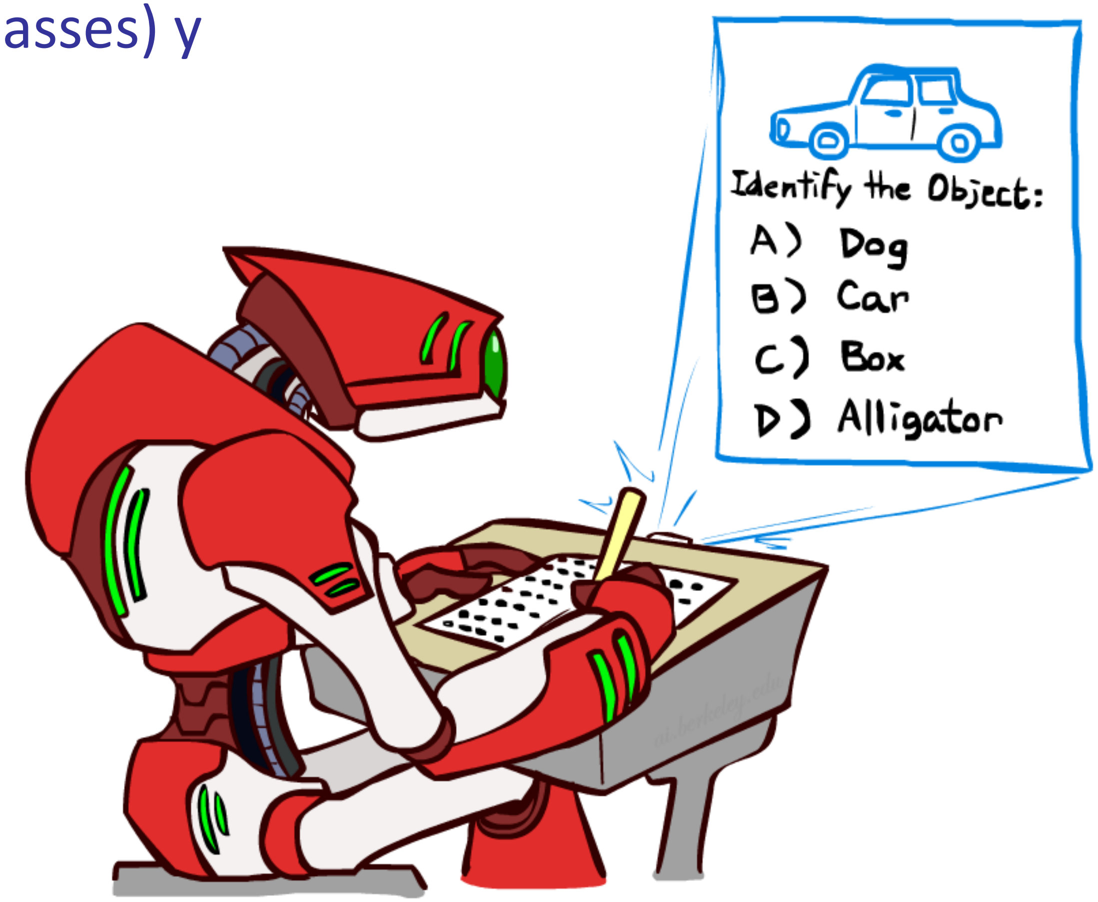
2

1

??

Other Classification Tasks

- Classification: given inputs x, predict labels (classes) y
- Examples:
  - Medical diagnosis (input: symptoms, classes: diseases)
  - Fraud detection (input: account activity, classes: fraud / no fraud)
  - Automatic essay grading (input: document, classes: grades)
  - Customer service email routing
  - Review sentiment
  - Language ID
  - … many more



- Classification is an important commercial technology!

Model-Based Classification

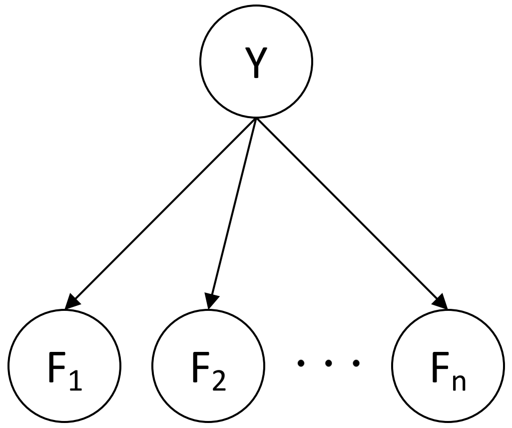
Model-Based Classification

- Model-based approach
  - Build a model (e.g. Bayes' net) where both the output label and input 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?

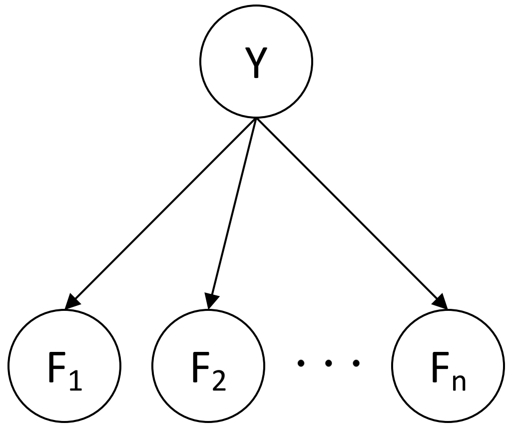
Naïve Bayes Model

- Random variables in this Bayes' net:
  - Y = The label
  - $F_1, F_2, \ldots, F_n$ = The n features
- Probability tables in this Bayes' net:
  - P(Y) = Probability of each label occurring, given no information about the features. Sometimes called the *prior*.
  - $P(F_i|Y)$ = One table per feature. Probability distribution over a feature, given the label.



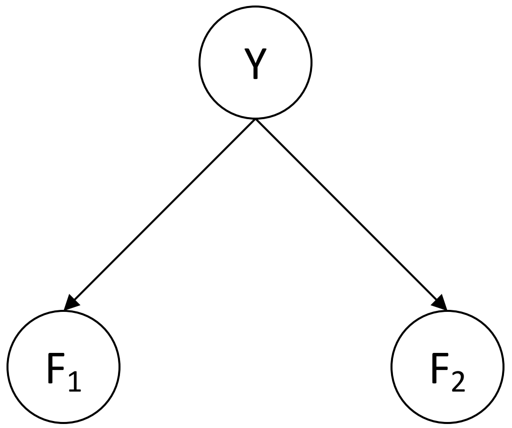
Naïve Bayes Model

- To perform training:
  - Use the training dataset to estimate the probability tables.
  - Estimate $P(Y)$ = how often does each label occur?
  - Estimate $P(F_i|Y)$ = how does the label affect the feature?
- To perform classification:
  - Instantiate all features. You know the input features, so they're your evidence.
  - Query for $P(Y|f_1, f_2, \ldots, f_n)$. Probability of label, given all the input features. Use an inference algorithm (e.g. variable elimination) to compute this.



Example: Naïve Bayes for Spam Filter

- Step 1: Select a ML algorithm. We choose to model the problem with Naïve Bayes.
- Step 2: Choose features to use.



Y: The label (spam or ham)	
Y	P(Y)
ham	?
spam	?

F_1: A feature (do I know the sender?)		
F_1	Y	$P(F_1 \mid Y)$
yes	ham	?
no	ham	?
yes	spam	?
no	spam	?

F_2: Another feature (# of occurrences of FREE)		
F_2	Y	$P(F_2 \mid Y)$
0	ham	?
1	ham	?
2	ham	?
0	spam	?
1	spam	?
2	spam	?

Example: Naïve Bayes for Spam Filter

- Step 3: Training: Use training data to fill in the probability tables.

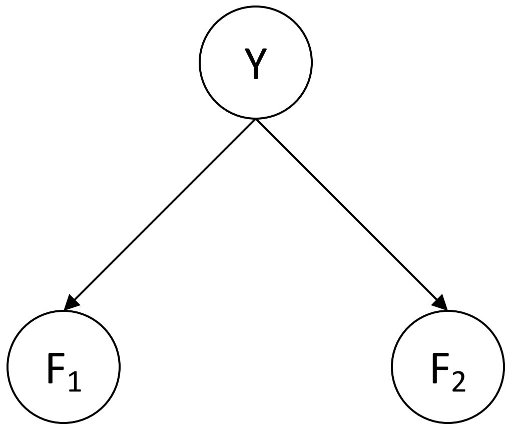
F_2: # of occurrences of FREE		
F_2	Y	$P(F_2 \mid Y)$
0	ham	0.5
1	ham	0.5
2	ham	0.0
0	spam	0.25
1	spam	0.50
2	spam	0.25

Training Data		
#	Email Text	Label
1	Attached is my portfolio.	ham
2	Are you **free** for a meeting tomorrow?	ham
3	**Free** unlimited credit cards!!!!	spam
4	Mail $10,000 check to this address	spam
5	Sign up now for 1 **free** Bitcoin	spam
6	**Free** money **free** money	spam

Row 4: $P(F_2=0 \mid Y=\text{spam}) = 0.25$ because 1 out of 4 spam emails contains “free” 0 times.
Row 5: $P(F_2=1 \mid Y=\text{spam}) = 0.50$ because 2 out of 4 spam emails contains “free” 1 time.
Row 6: $P(F_2=2 \mid Y=\text{spam}) = 0.25$ because 1 out of 4 spam emails contains “free” 2 times.

Example: Naïve Bayes for Spam Filter

- Model trained on a larger dataset:



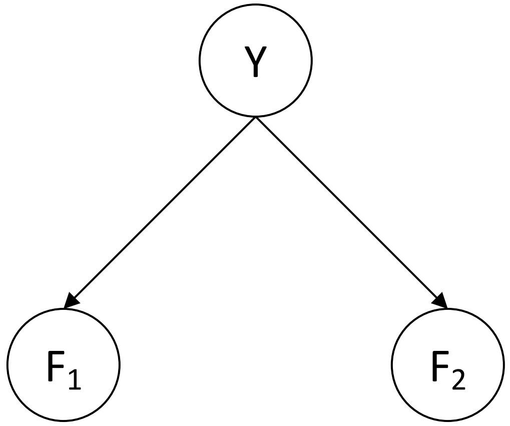
Y: The label (spam or ham)	
Y	P(Y)
ham	0.6
spam	0.4

F_1: A feature (do I know the sender?)		
F_1	Y	$P(F_1 \mid Y)$
yes	ham	0.7
no	ham	0.3
yes	spam	0.1
no	spam	0.9

F_2: Another feature (# of occurrences of FREE)		
F_2	Y	$P(F_2 \mid Y)$
0	ham	0.85
1	ham	0.07
2	ham	0.08
0	spam	0.75
1	spam	0.12
2	spam	0.13

Example: Naïve Bayes for Spam Filter

- Step 4: Classification
- Suppose you want to label this email from a known sender:
  "**Free** food in Soda 430 today"
- Step 4.1: Feature extraction:
  - F_1 = yes, known sender
  - F_2 = 1 occurrence of "free"



Example: Naïve Bayes for Spam Filter

- Step 4.2: Inference
- Instantiate features (evidence):
  - F_1 = yes
  - F_2 = 1
- Compute joint probabilities:
  - P(Y = spam, F_1 = yes, F_2 = 1) = P(Y = spam) P(F_1 = yes | spam) P(F_2 = 1 | spam)
    = 0.4 * 0.1 * 0.12 = 0.0048
  - P(Y = ham, F_1 = yes, F_2 = 1) = P(Y = ham) P(F_1 = yes | ham) P(F_2 = 1 | ham)
    = 0.6 * 0.7 * 0.07 = 0.0294
- Normalize:
  - P(Y = spam | F_1 = yes, F_2 = 1) = 0.0048 / (0.0048+0.0294) = 0.14
  - P(Y = ham | F_1 = yes, F_2 = 1) = 0.0294 / (0.0048+0.0294) = 0.86
- Classification result:
  - 14% chance the email is spam. 86% chance it's ham.
  - Or, if you don't need probabilities, note that 0.0294 > 0.0048 and guess ham.

Y: The label (spam or ham)	
Y	P(Y)
ham	0.6
spam	0.4

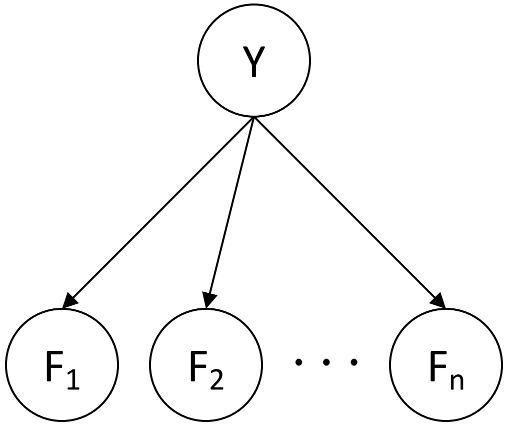
F_1: do I know the sender?		
F_1	Y	$P(F_1 \mid Y)$
yes	ham	0.7
no	ham	0.3
yes	spam	0.1
no	spam	0.9

F_2: # of occurrences of FREE		
F_2	Y	$P(F_2 \mid Y)$
0	ham	0.85
1	ham	0.07
2	ham	0.08
0	spam	0.75
1	spam	0.12
2	spam	0.13

Naïve Bayes for Digits

- 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.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$



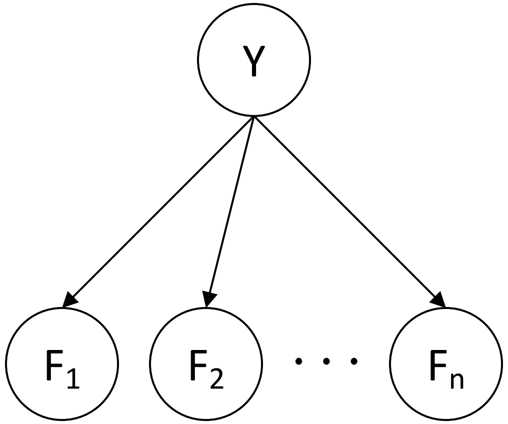
  - Here: lots of features, each is binary valued

- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn?

General Naïve Bayes

- Naïve Bayes assumes that all features are independent effects of the label
- A general Naive Bayes model:



|Y| parameters

$$P(\mathsf{Y},\mathsf{F}_1 \dots \mathsf{F}_n) = \quad P(\mathsf{Y}) \prod_i P(\mathsf{F}_i|\mathsf{Y})$$

$|Y| \times |F|^n$ values

$n \times |F| \times |Y|$ parameters

- We only have to specify how each feature depends on the class
- Total number of parameters is *linear* in n
- Model is very simplistic, but often works anyway

Inference for Naïve Bayes

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

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

$$\xrightarrow{+} \quad P(f_1 \dots f_n)$$

  - Step 2: sum to get probability of evidence

  - Step 3: normalize by dividing Step 1 by Step 2

$$\Downarrow$$

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

Naïve Bayes for Text

- Bag-of-words Naïve Bayes:
  - Features: W_i is the word at position i
  - As before: predict label conditioned on feature variables (spam vs. ham)
  - As before: assume features are conditionally independent given label
  - New: each W_i is identically distributed

- Generative model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$

Word at position i, not i^th word in the dictionary!

- "Tied" distributions and bag-of-words
  - Usually, each variable gets its own conditional probability distribution P(F|Y)
  - In a bag-of-words model
    - Each position is identically distributed
    - All positions share the same conditional probs P(W|Y)
    - Why make this assumption?
  - Called "bag-of-words" because model is insensitive to word order or reordering

Example: Spam Filtering

- Model: $P(Y, W_1 \ldots W_n) = P(Y) \prod_i P(W_i|Y)$
- What are the parameters?

$P(Y)$

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
with	0.0080
from	0.0075
...	

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

the	0.0210
to	0.0133
of	0.0119
2002	0.0110
with	0.0108
from	0.0107
and	0.0105
a	0.0100
...	

- Where do these tables come from?

Spam Example

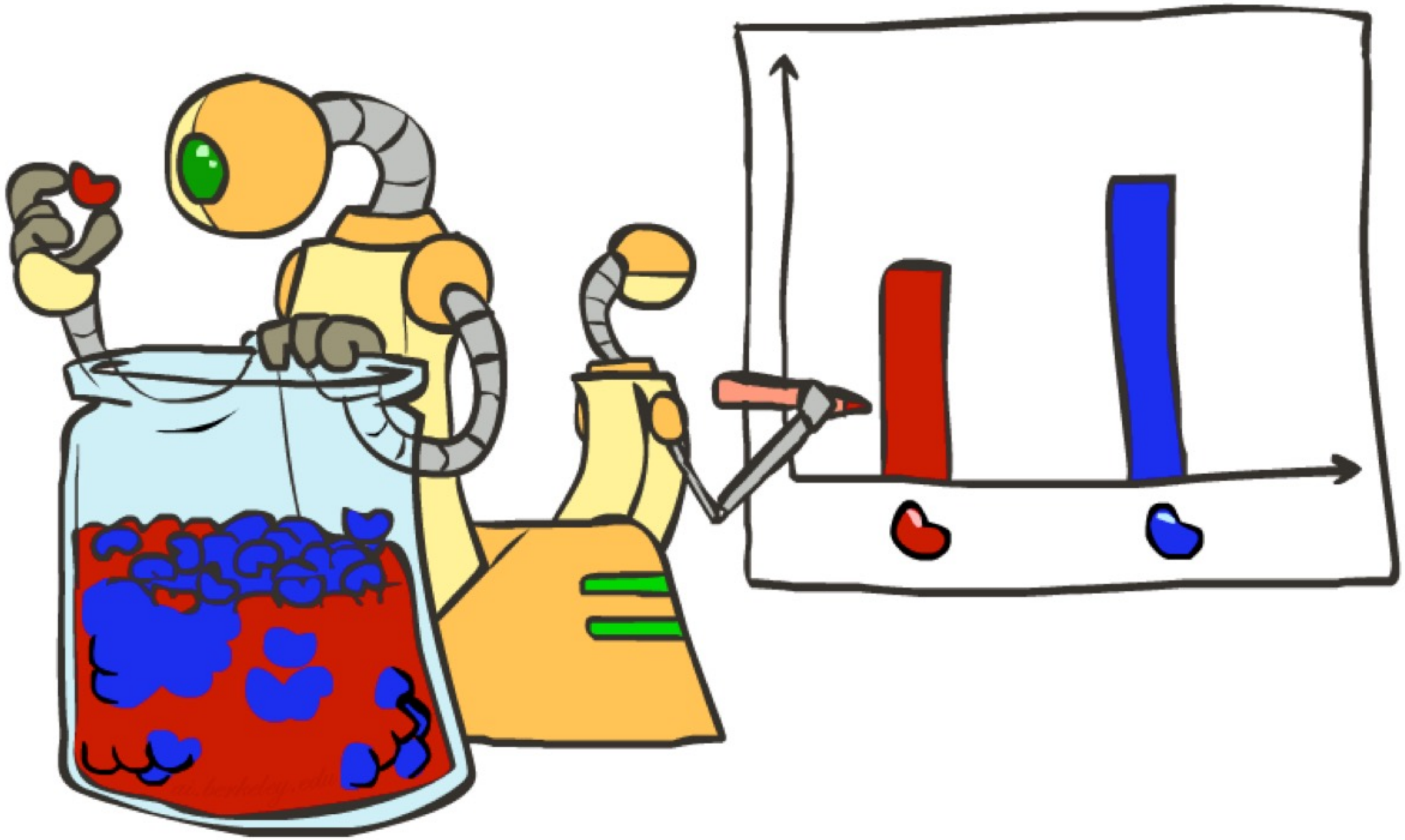
Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4
Gary	0.00002	0.00021	-11.8	-8.9
would	0.00069	0.00084	-19.1	-16.0
you	0.00881	0.00304	-23.8	-21.8
like	0.00086	0.00083	-30.9	-28.9
to	0.01517	0.01339	-35.1	-33.2
lose	0.00008	0.00002	-44.5	-44.0
weight	0.00016	0.00002	-53.3	-55.0
while	0.00027	0.00027	-61.5	-63.2
you	0.00881	0.00304	-66.2	-69.0
sleep	0.00006	0.00001	-76.0	-80.5

P(spam | w) = 98.9

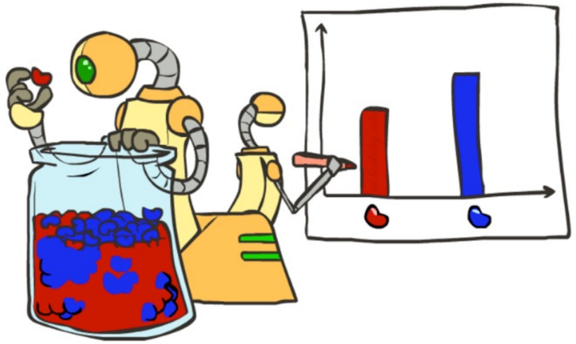
General Naïve Bayes

- What do we need in order to use Naïve Bayes?
  - Inference method
    - Start with a bunch of probabilities: P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the *parameters* of the model and denoted by θ
    - Up until now, we assumed these appeared by magic, but they typically come from training data counts

Parameter Estimation



Parameter Estimation



- Estimating the distribution of a random variable
- *Elicitation:* ask a human (why is this hard?)
- *Empirically:* use training data (learning!)
  - Example: The parameter θ is the true fraction of red beans in the jar. You don't know θ but would like to estimate it.
  - Collecting training data: You randomly pull out 3 beans:

    r r b

  - Estimating θ using counts, you guess 2/3 of beans in the jar are red.
  - Can we mathematically show that using counts is the "right" way to estimate θ?

Parameter Estimation with Maximum Likelihood

- Can we mathematically show that using counts is the "right" way to estimate θ?
- Maximum likelihood estimation: Choose the θ value that maximizes the probability of the observation
  - In other words, choose the θ value that maximizes P(observation | θ)
  - For our problem:

    P(observation | θ)

    = P(randomly selected 2 red and 1 blue | θ of beans are red)

    = P(red | θ) P(red | θ) P(blue | θ)

    = $\theta^2 (1- \theta)$

  - We want to compute:

    $$\operatorname*{argmax}_{\theta} \theta^2 (1- \theta)$$

Parameter Estimation with Maximum Likelihood

- We want to compute:

  $$\underset{\theta}{\operatorname{argmax}}\ \theta^2 (1-\theta)$$

- Set derivative to 0, and solve!
  - Common issue: The likelihood (expression we're maxing) is the product of a lot of probabilities. This can lead to complicated derivatives.
  - Solution: Maximize the log-likelihood instead. Useful fact:

    $$\underset{\theta}{\operatorname{argmax}}\ f(\theta) = \underset{\theta}{\operatorname{argmax}}\ \ln f(\theta)$$

Parameter Estimation with Maximum Likelihood

$$\underset{\theta}{\operatorname{argmax}}\, \theta^2(1-\theta)$$

Find θ that maximizes likelihood

$$= \underset{\theta}{\operatorname{argmax}} \ln\left(\theta^2(1-\theta)\right)$$

Find θ that maximizes log-likelihood (will be the same θ)

$$\frac{d}{d\theta}\ln\left(\theta^2(1-\theta)\right) = 0$$

Set derivative to 0

$$\frac{d}{d\theta}\left[\ln(\theta^2) + \ln(1-\theta)\right] = 0$$

Logarithm rule: products become sums

$$\frac{d}{d\theta}\left[2\ln(\theta) + \ln(1-\theta)\right] = 0$$

Logarithm rule: exponentiation becomes multiplication

$$\frac{d}{d\theta}2\ln(\theta) + \frac{d}{d\theta}\ln(1-\theta) = 0$$

Now we can derive each term of the original product separately

$$\frac{2}{\theta} - \frac{1}{1-\theta} = 0$$

Reminder: Derivative of ln(θ) is 1/θ

$$\theta = \frac{2}{3}$$

Use algebra to solve for θ. If we used arbitrary red and blue counts r and b instead of r=2 and b=1, we'd get θ = r / (r+b), the count estimate.

Maximum Likelihood?

- Relative frequencies are the maximum likelihood estimates

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

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

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

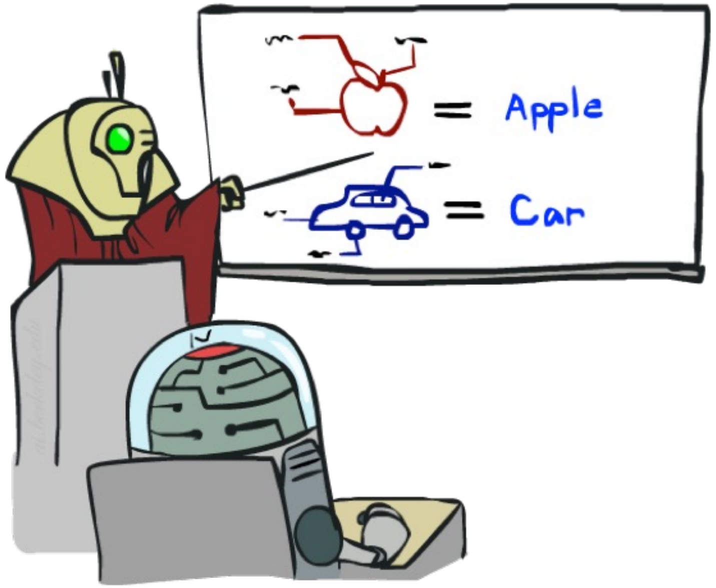
$$\theta_{MAP} = \arg\max_{\theta} P(\theta|\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X})$$
$$= \arg\max_{\theta} P(\mathbf{X}|\theta)P(\theta)$$

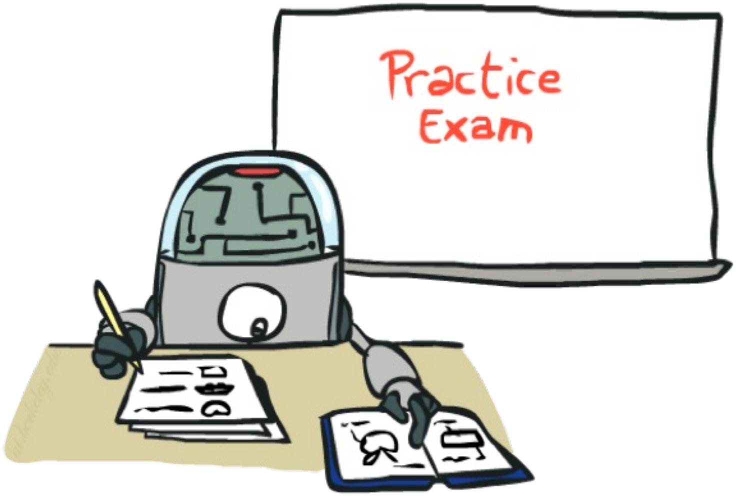
$\Rightarrow$????

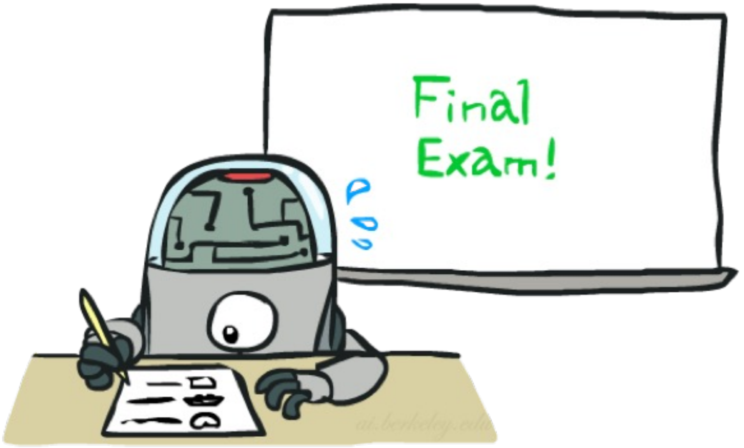
Parameter Estimation with Maximum Likelihood

- How do we estimate the conditional probability tables?
  - Maximum Likelihood, which corresponds to counting
- Need to be careful though … let’s see what can go wrong..

Training and Testing



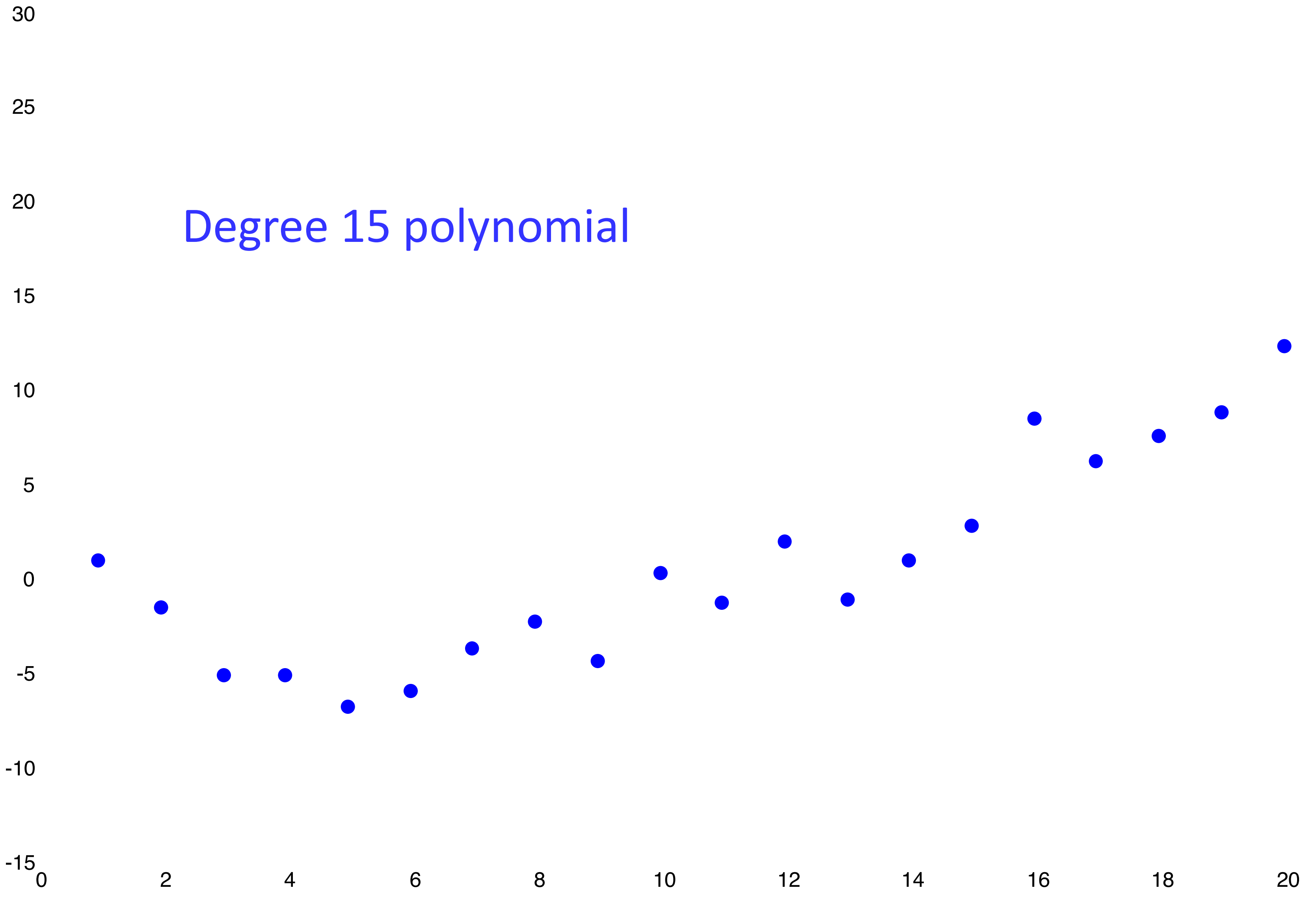


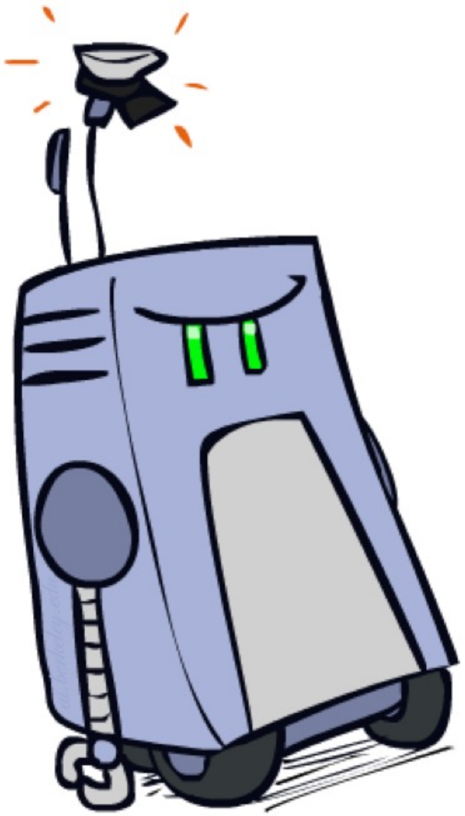


Empirical Risk Minimization

- Empirical risk minimization
  - Basic principle of machine learning
  - We want the model (classifier, etc) that does best on the true test distribution
  - Don't know the true distribution so pick the best model on our actual training set
  - Finding "the best" model on the training set is phrased as an optimization problem
- Main worry: overfitting to the training set
  - Better with more training data (less sampling variance, training more like test)
  - Better if we limit the complexity of our hypotheses (regularization and/or small hypothesis spaces)

Overfitting





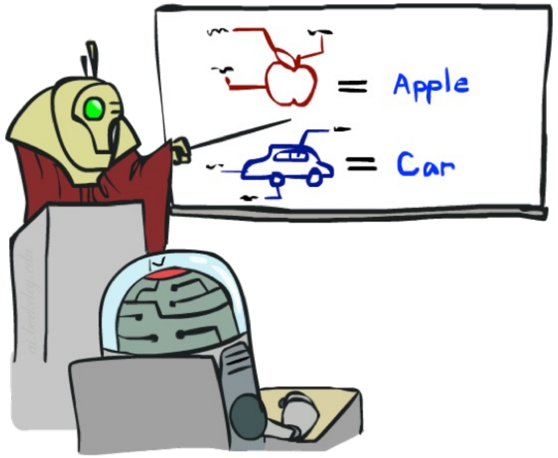
Important Concepts

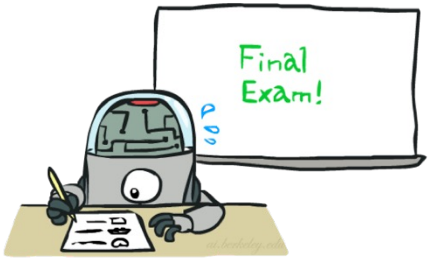
- How do we check that we're not overfitting during training?
- Split training data into 3 different sets:
  - Training set
  - Held out set (more on this later)
  - Test set
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Compute accuracy of test set
  - Very important: never "peek" at the test set!
- Evaluation (many metrics possible, e.g. accuracy)
  - 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

Training Data

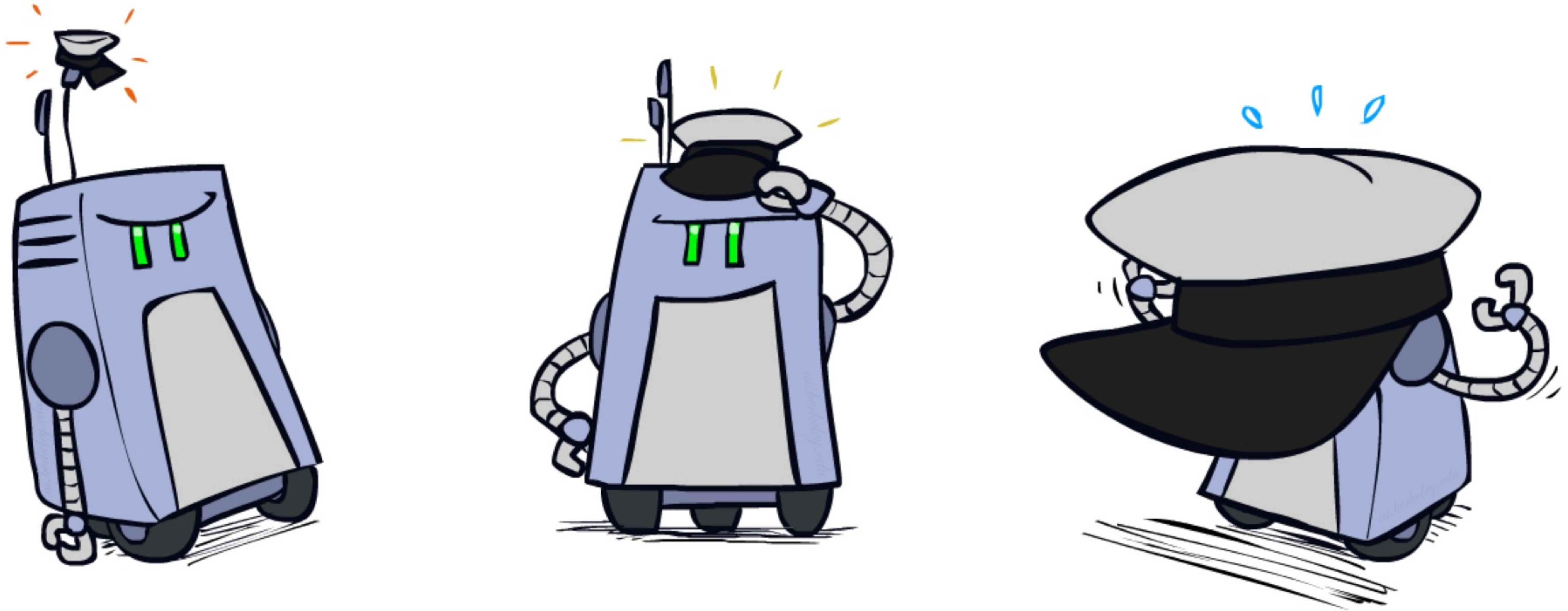
Held-Out Data

Test Data





Generalization and Overfitting



Example: Overfitting

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

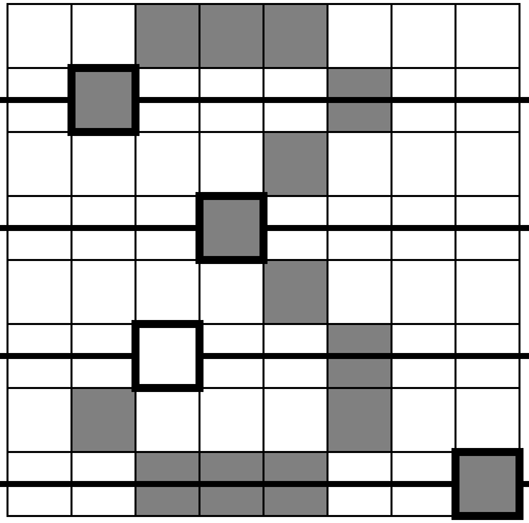
$P(C = 2) = 0.1$

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

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

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

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



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

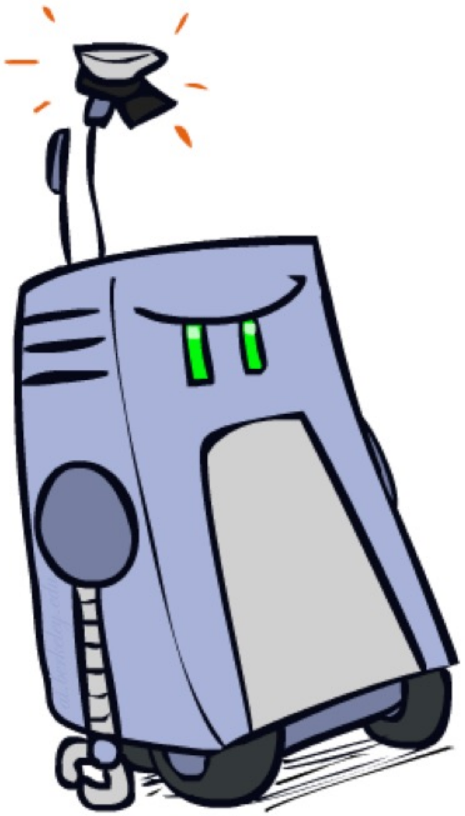
$P(C = 3) = 0.1$

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

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

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

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



2 wins!!

Example: Overfitting

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

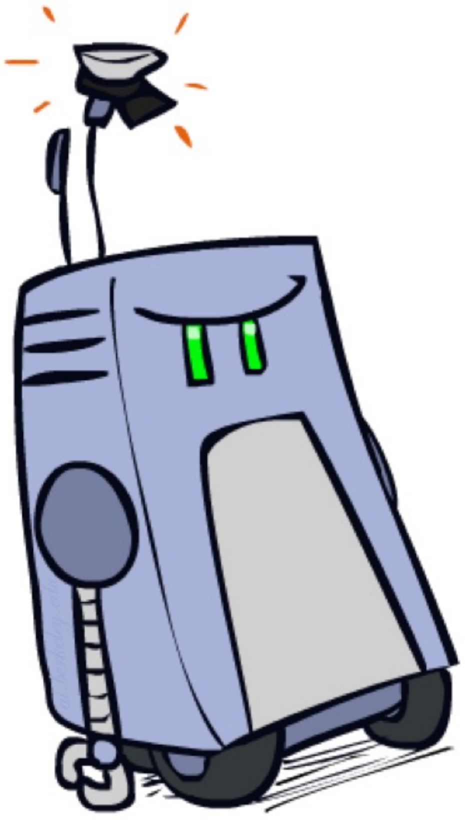
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

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

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

What went wrong here?



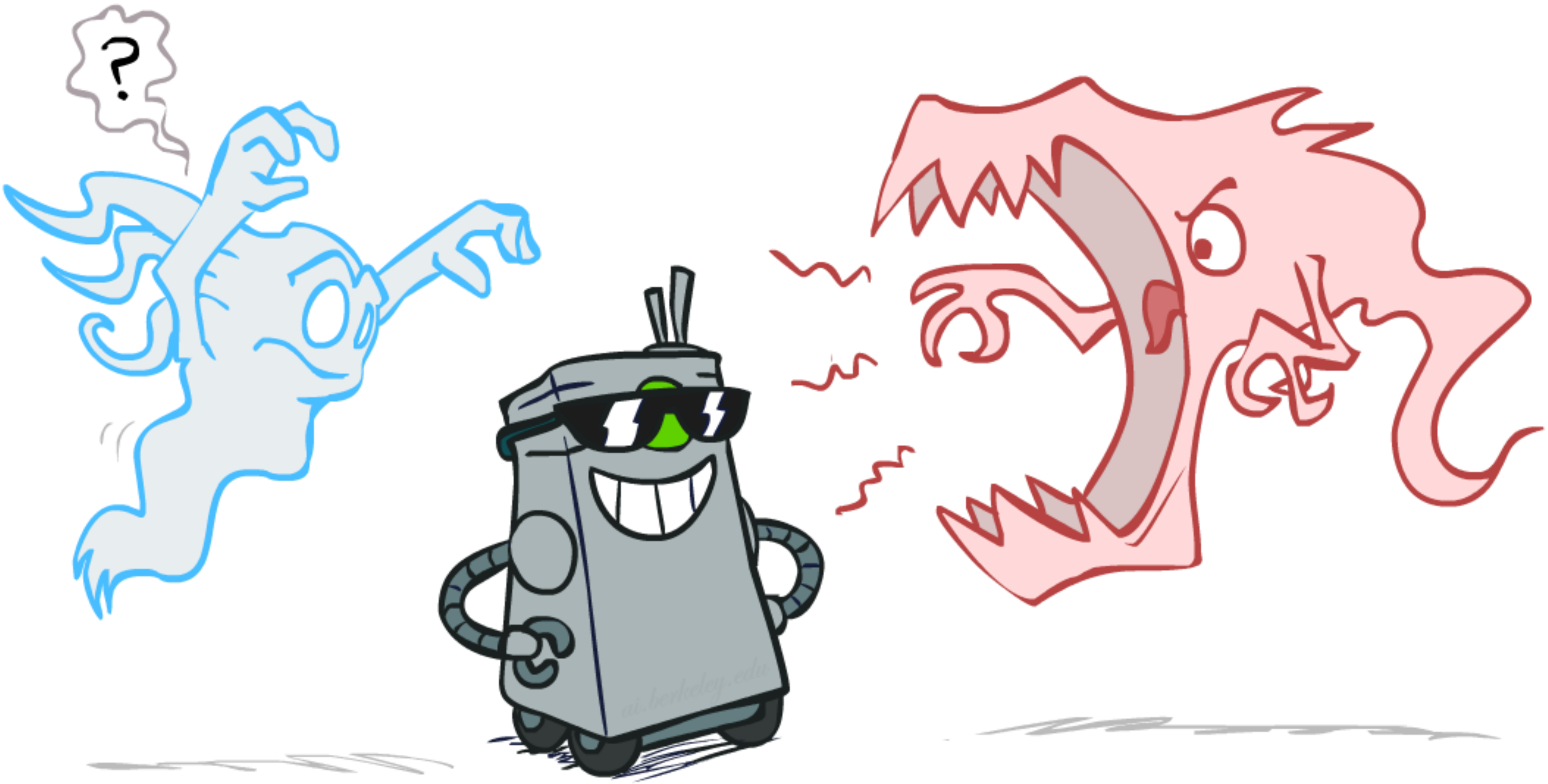
Generalization and Overfitting

- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) 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

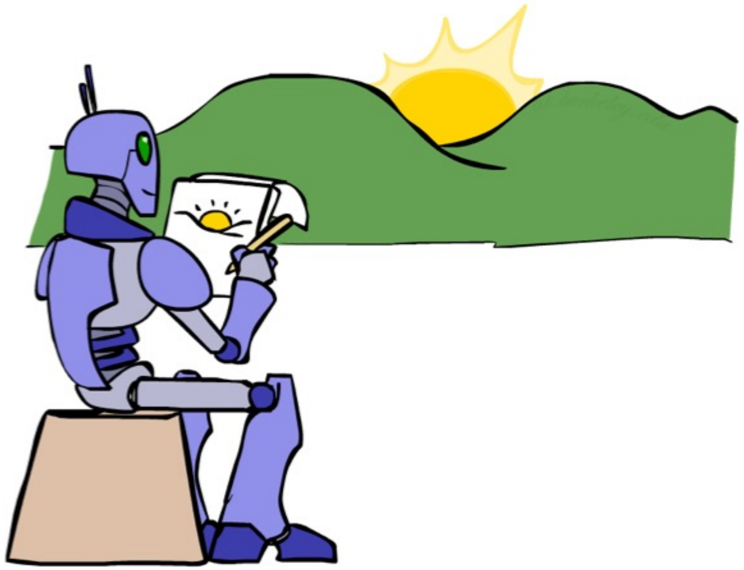
- As an extreme case, imagine using the entire email as the only feature (e.g. document ID)
  - Would get the training data perfect (if deterministic labeling)
  - Wouldn't *generalize* at all
  - Just making the bag-of-words assumption gives us some generalization, but isn't enough

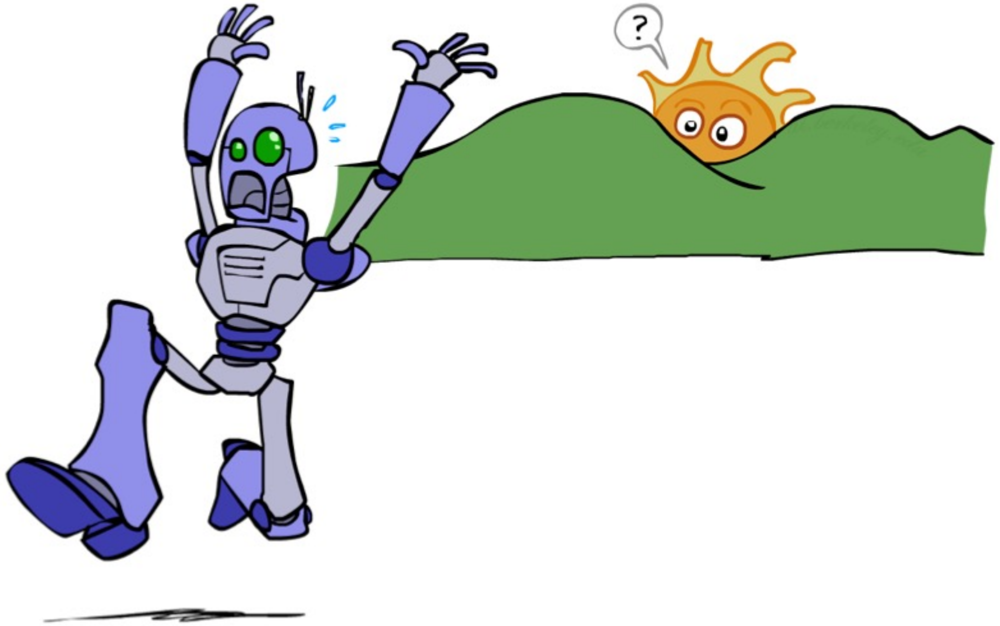
- To generalize better: we need to smooth or regularize the estimates

Smoothing



Unseen Events





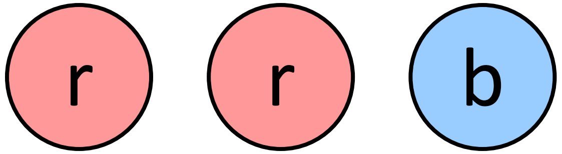
Laplace Smoothing

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

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

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

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



$$P_{ML}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

$$P_{LAP}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

Laplace Smoothing

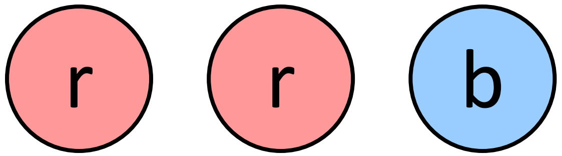
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

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

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

- Laplace for conditionals:
  - Smooth each condition independently:

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



$$P_{LAP,0}(X) = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$$

$$P_{LAP,1}(X) = \left\langle \frac{3}{5}, \frac{2}{5} \right\rangle$$

$$P_{LAP,100}(X) = \left\langle \frac{102}{203}, \frac{101}{203} \right\rangle$$

Real Naïve Bayes: Smoothing

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

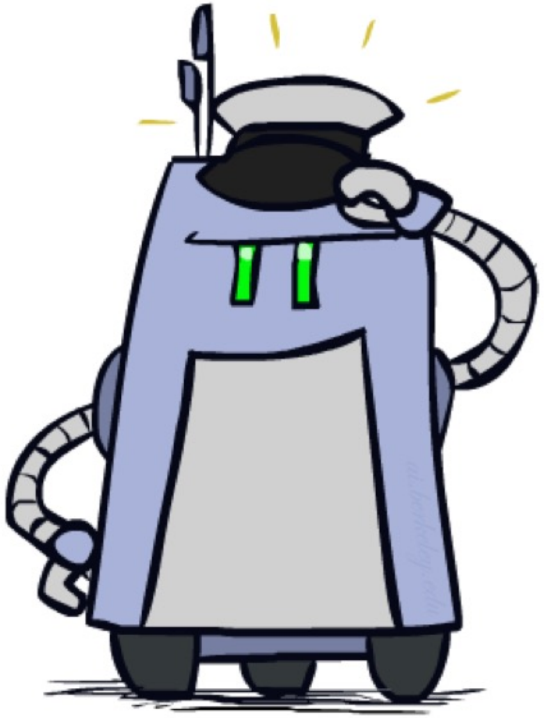
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

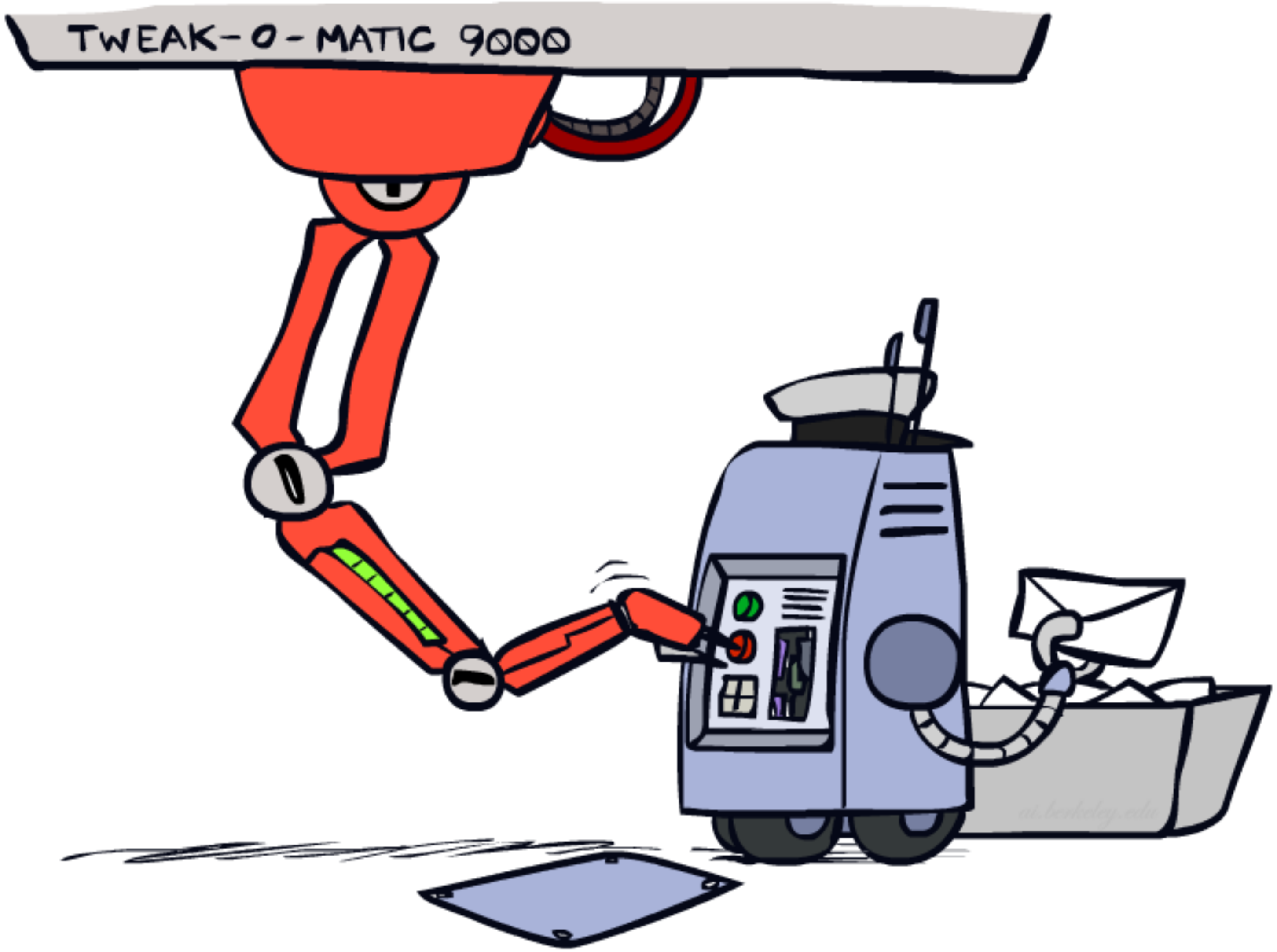
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

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

Do these make more sense?



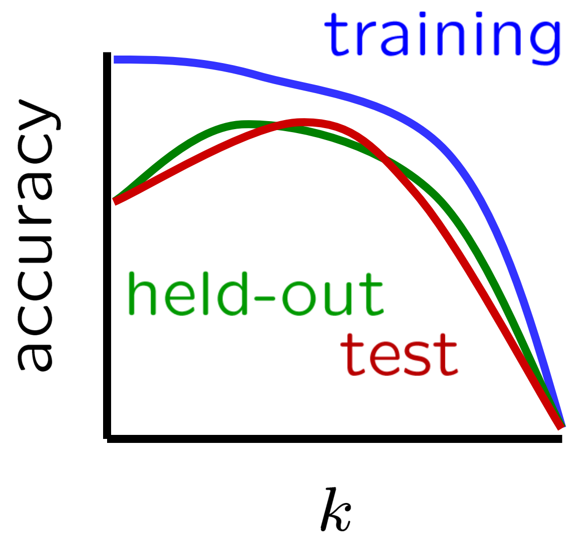
Tuning

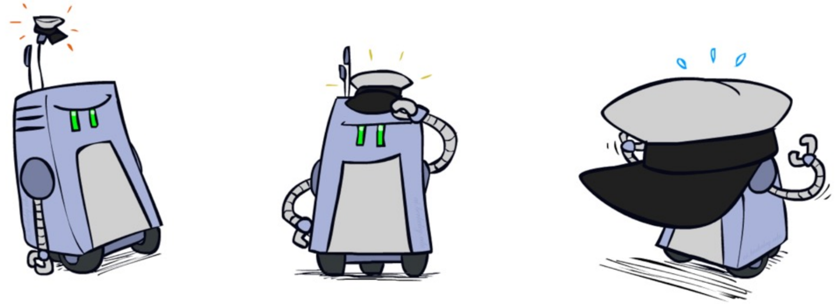


Tuning on Held-Out Data

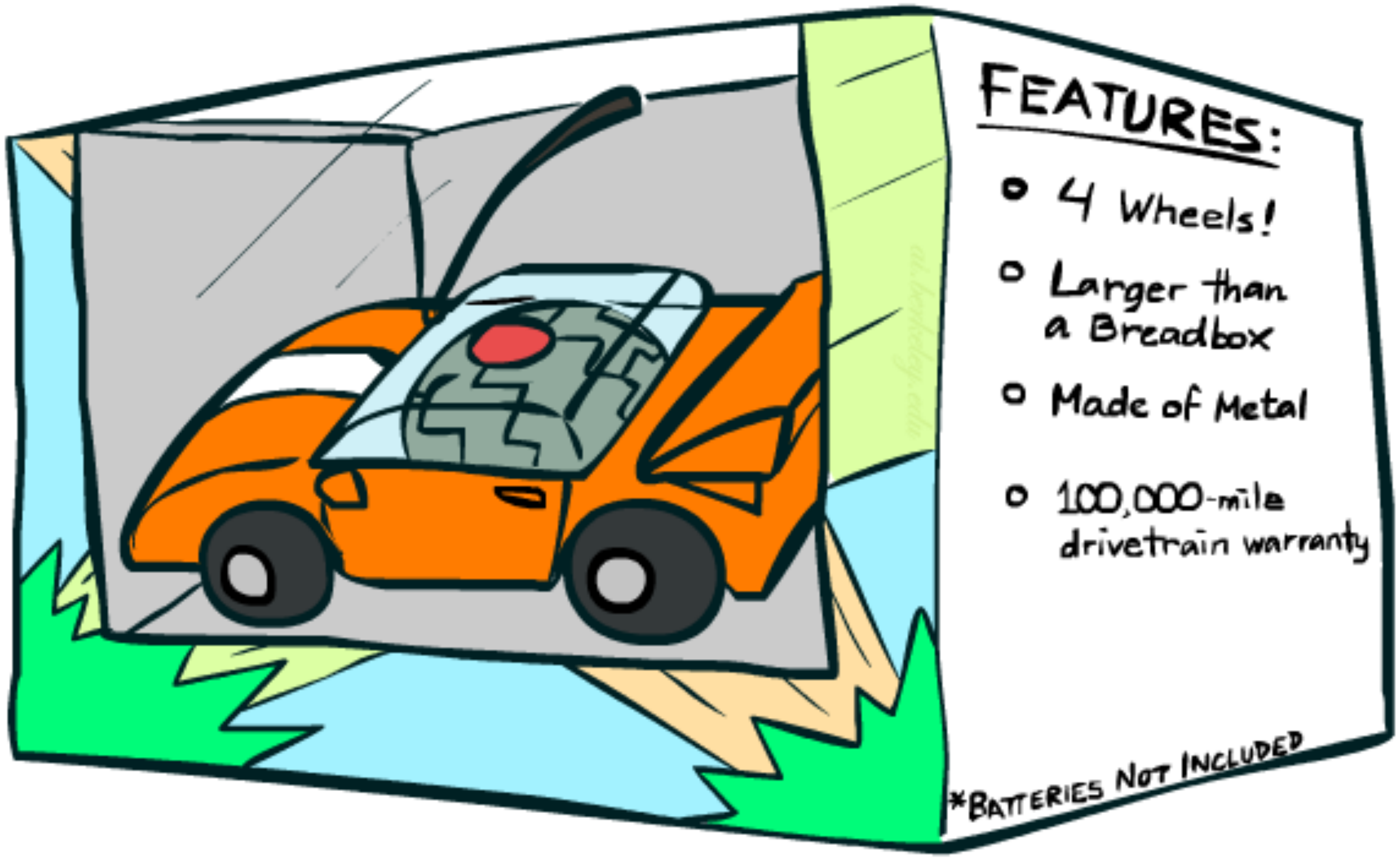
- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α

- 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





Features


FEATURES:
4 Wheels!
Larger than a Breadbox
Made of Metal
100,000-mile drivetrain warranty
*BATTERIES NOT INCLUDED

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 . . .
```

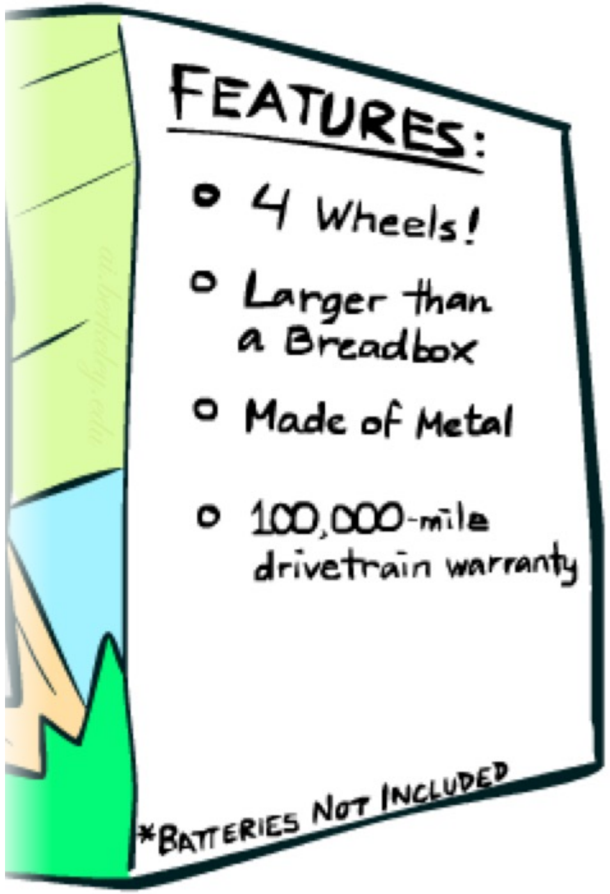
```
. . . To receive your $30 Amazon.com promotional certificate,
click through to

  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, and, later, how to induce new features



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. calling everything “ham” gets 66%, so a classifier that gets 70% isn’t very good…

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

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

Next Time: Discriminative Learning