

# CS 343H: Honors Artificial Intelligence

## Naïve Bayes



Prof. Peter Stone — The University of Texas at Austin

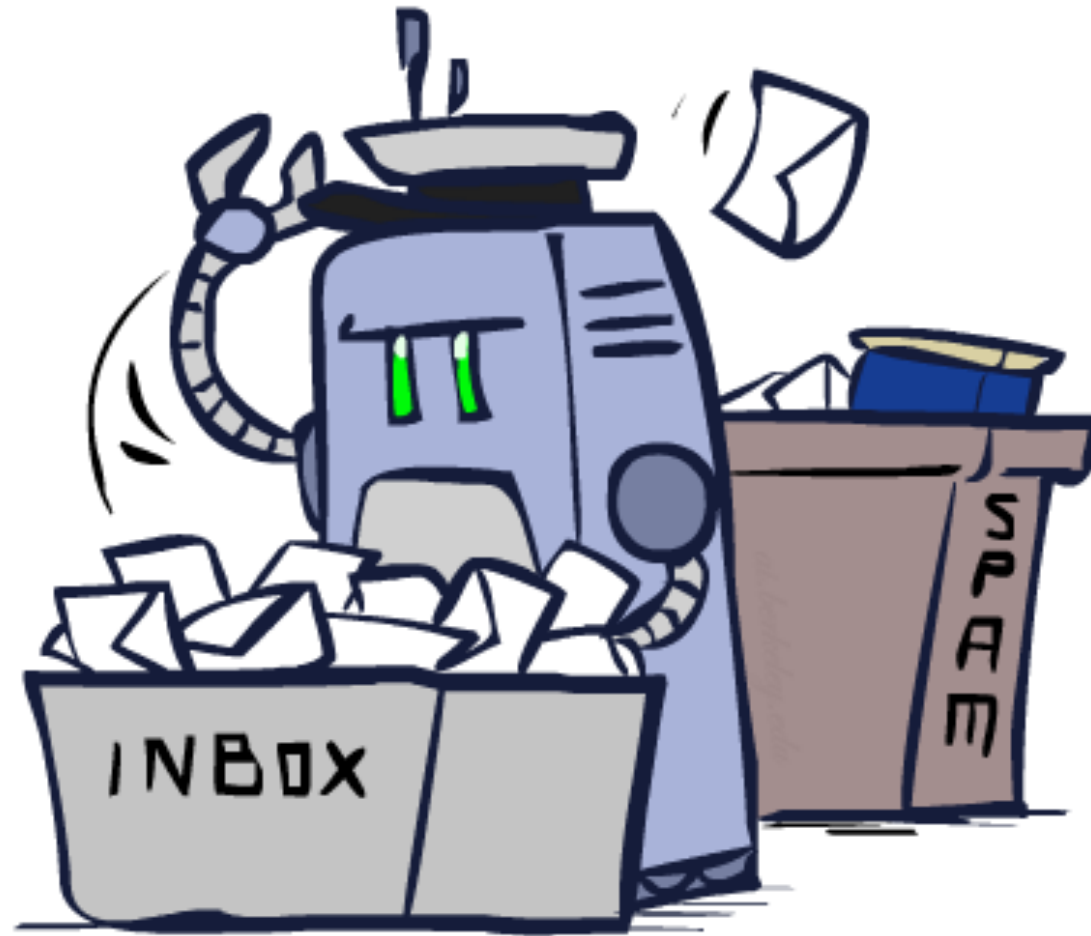
# Machine Learning

---

- 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)
- Today: model-based classification with Naive Bayes

# Classification

---



# Example: Spam Filter

- Input: an email
- Output: spam/real
- Setup:
  - Get a large collection of example emails, each labeled “spam” or “real”
  - 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 real / spam decision
  - Words: FREE!
  - Text Patterns: \$dd, CAPS
  - Non-text: SenderInContacts
  - ...



Dear Sir.

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



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

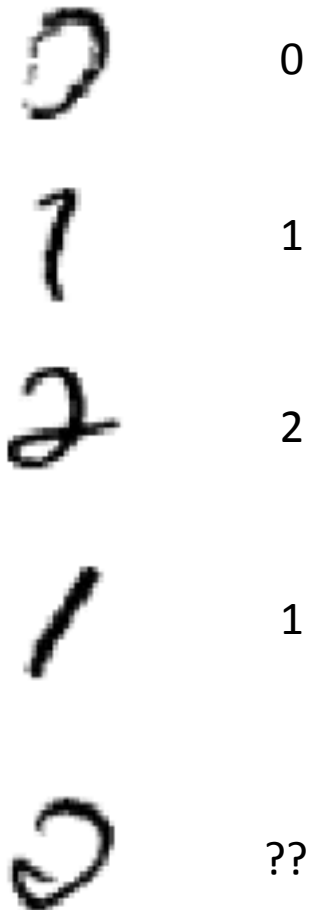
99 MILLION EMAIL ADDRESSES  
FOR ONLY \$99



Ok, I know 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.

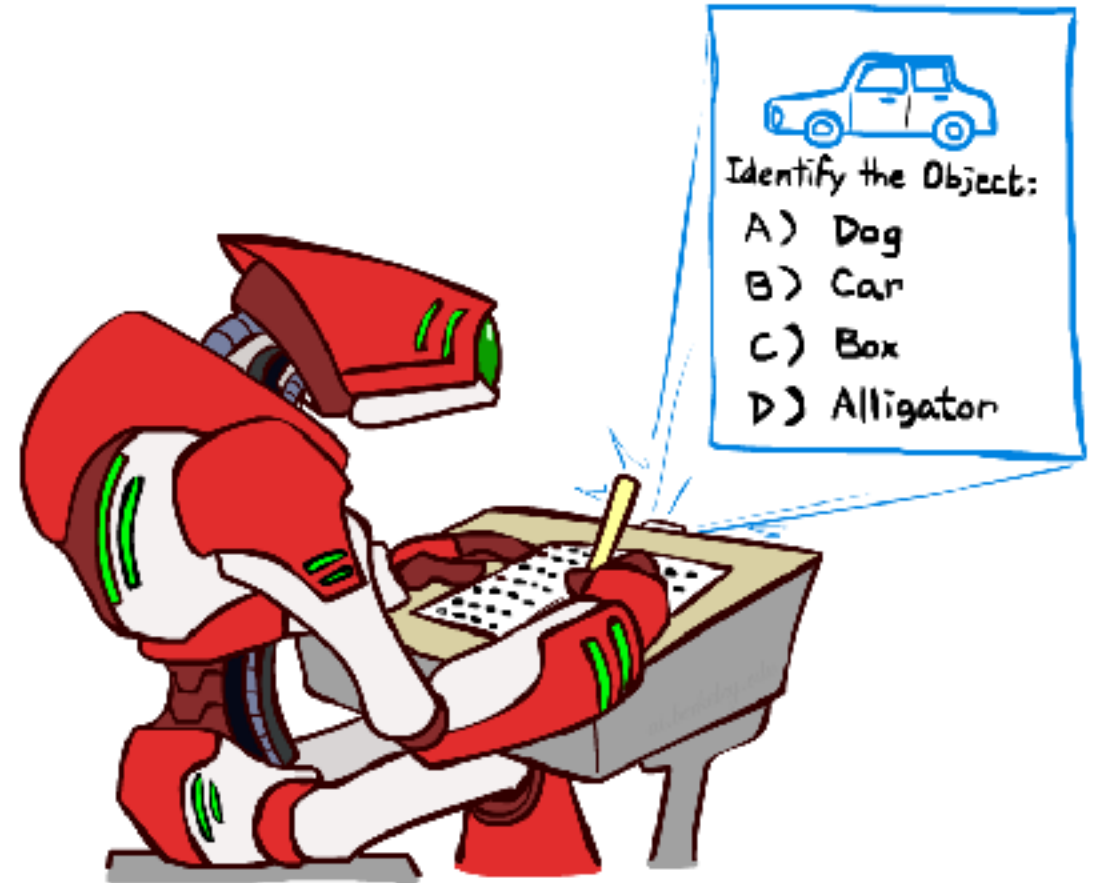
# Example: Digit Recognition

- 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
  - ...

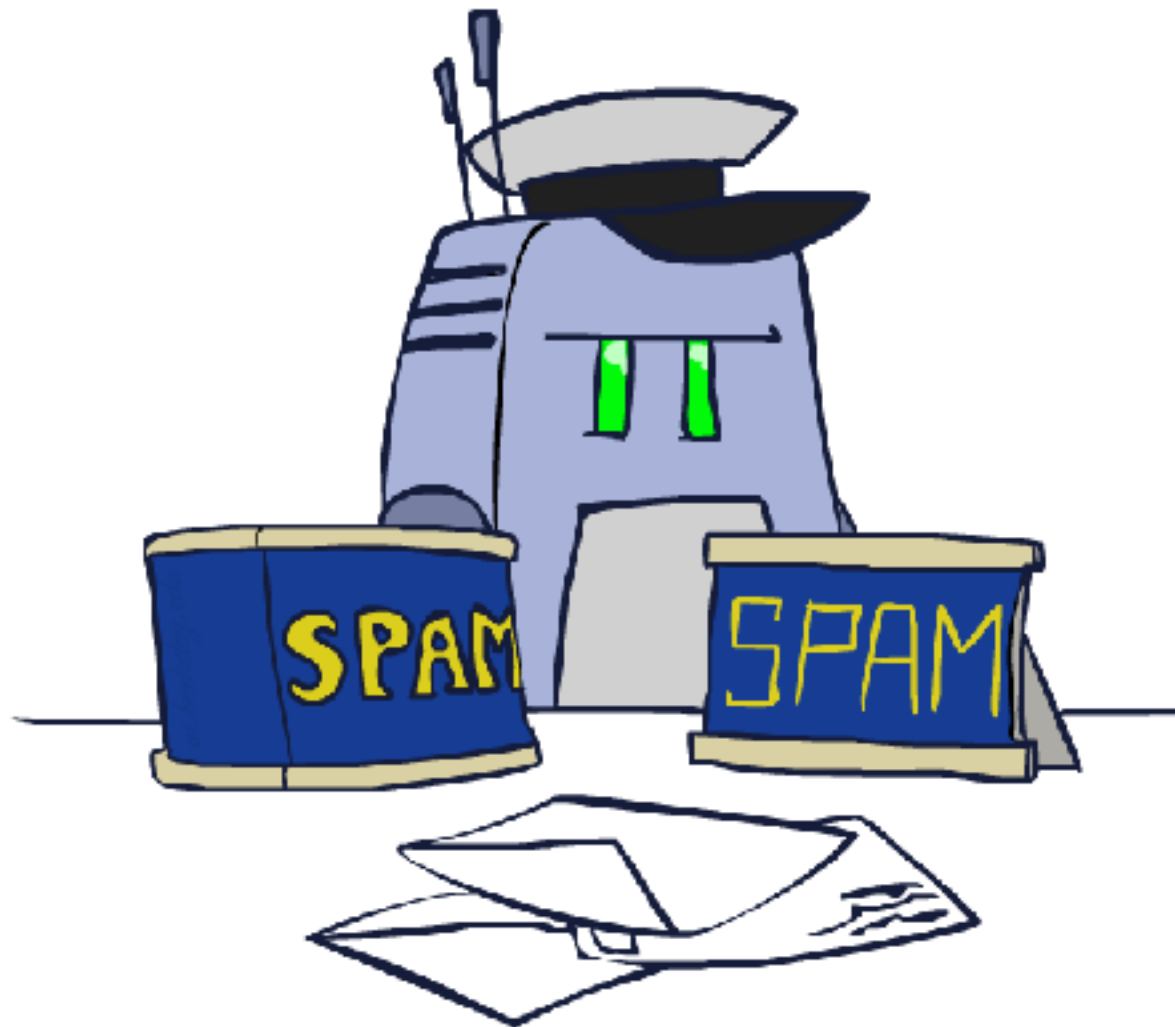


# Other Classification Tasks

- Classification: given inputs  $x$ , predict labels (classes)  $y$
- Examples:
  - Spam detection (input: document, classes: spam / ham)
  - 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)
  - Customer service email routing
  - ... many more
- Classification is an important commercial technology!



# Model-Based Classification



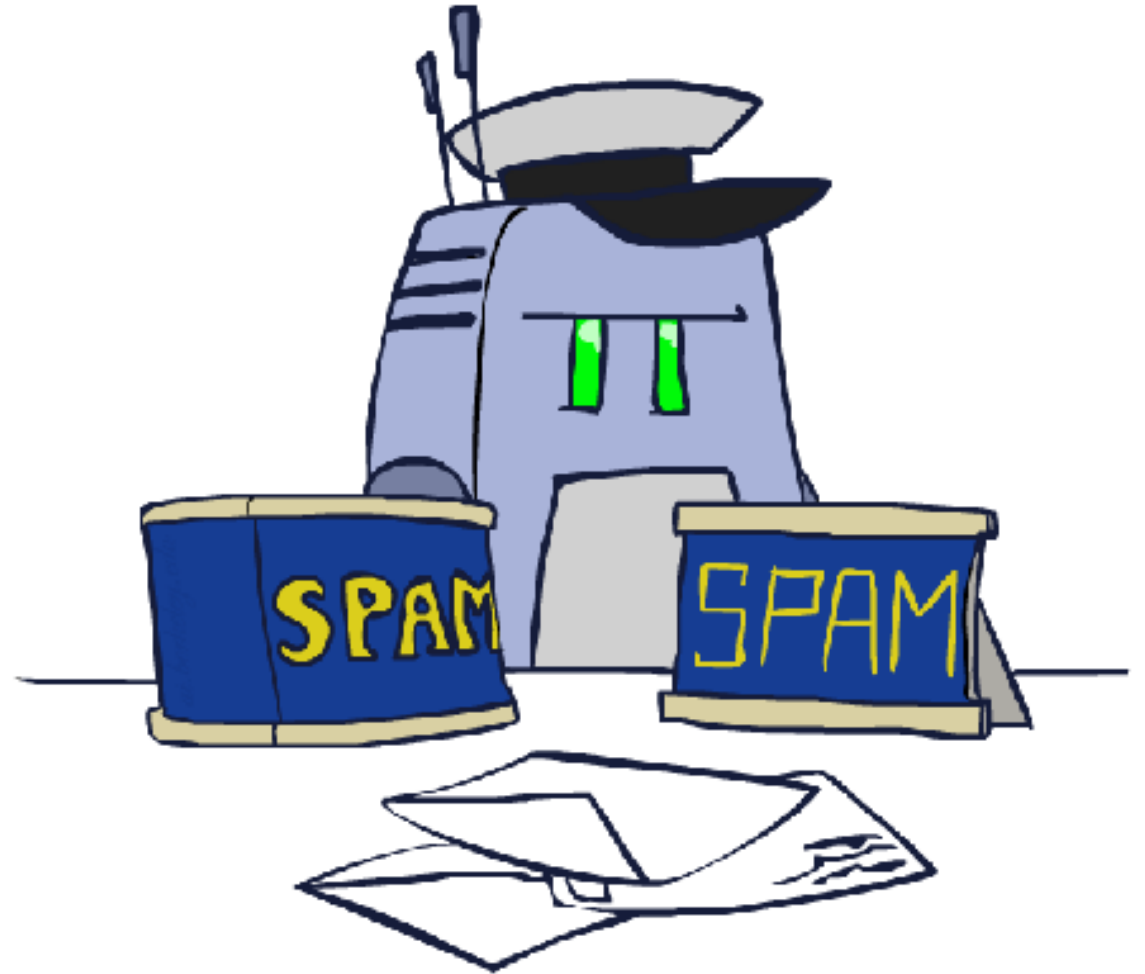
# Model-Based Classification

- 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?






# Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label

- Simple digit recognition version:

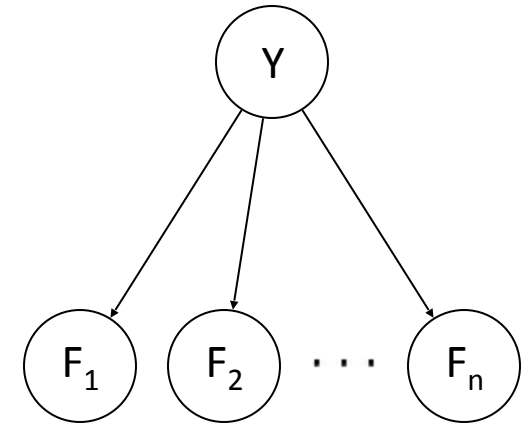
- One feature (variable)  $F_{ij}$  for each grid position  $\langle i,j \rangle$
- Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
- Each input maps to a feature vector, e.g.

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

- Here: lots of features, each is binary valued

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

- What do we need to learn?



# General Naïve Bayes

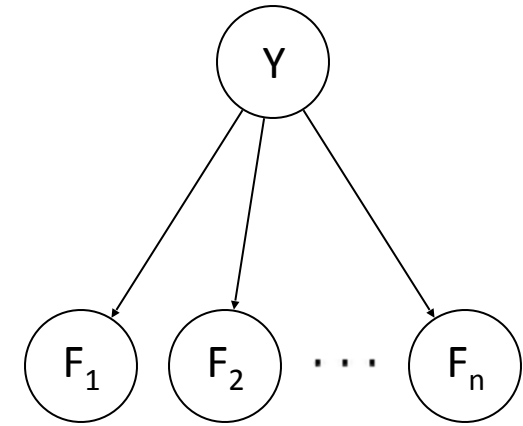
- A general Naive Bayes model:

$|Y|$  parameters

$$P(Y, F_1 \dots F_n) = P(Y) \prod_i P(F_i | Y)$$

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

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

+ ↙

- Step 2: sum to get probability of evidence

$$P(f_1 \dots f_n)$$

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

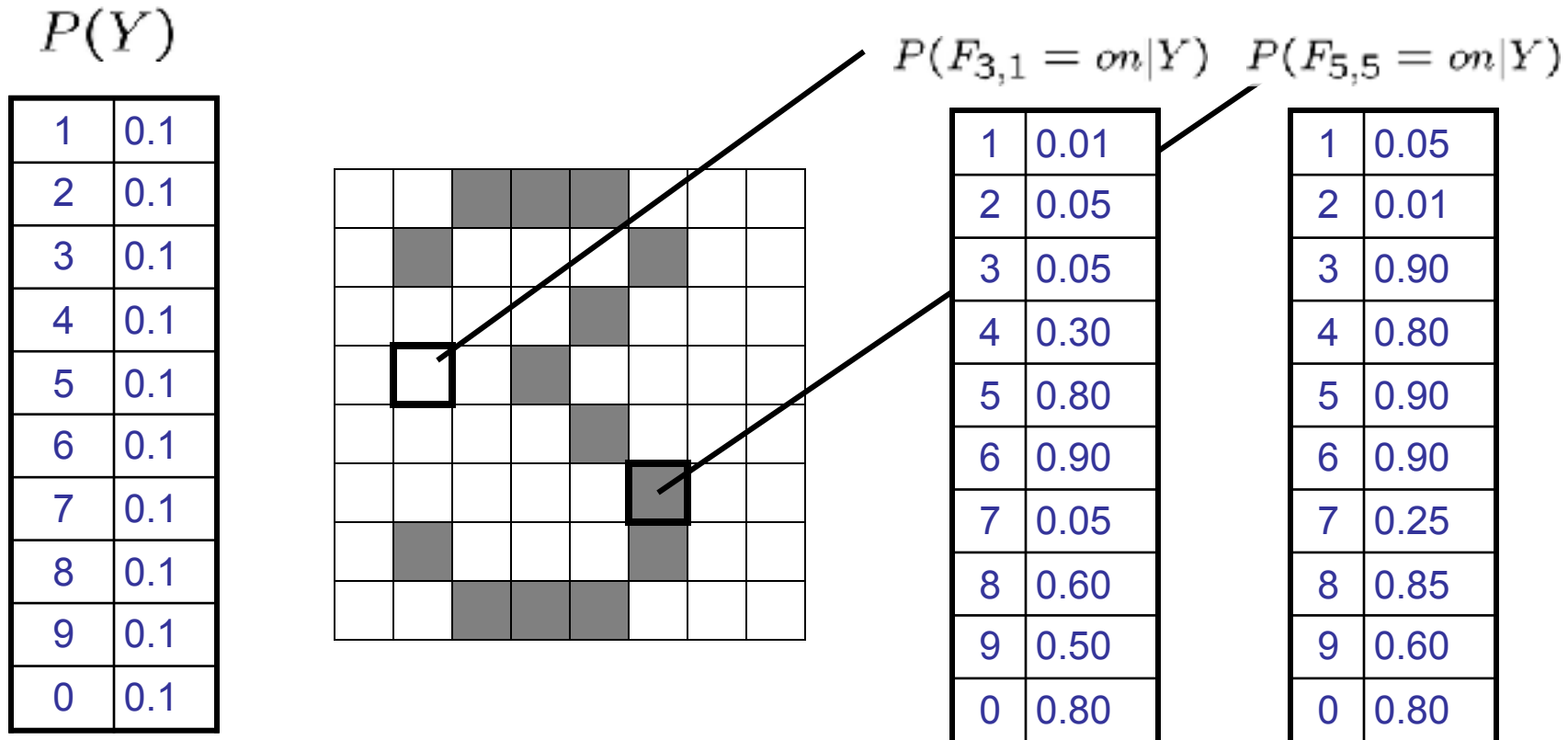
$$P(Y|f_1 \dots f_n)$$

# General Naïve Bayes

---

- What do we need in order to use Naïve Bayes?
  - Inference method (we just saw this part)
    - Start with a bunch of probabilities:  $P(Y)$  and the  $P(F_i|Y)$  tables
    - Use standard inference to compute  $P(Y|F_1...F_n)$
    - 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  $\theta$
    - Up until now, we assumed these appeared by magic, but...
    - ...they typically come from training data counts: we'll look at this soon

# Example: Conditional Probabilities




# 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  $i^{\text{th}}$  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:  $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$
- 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
...

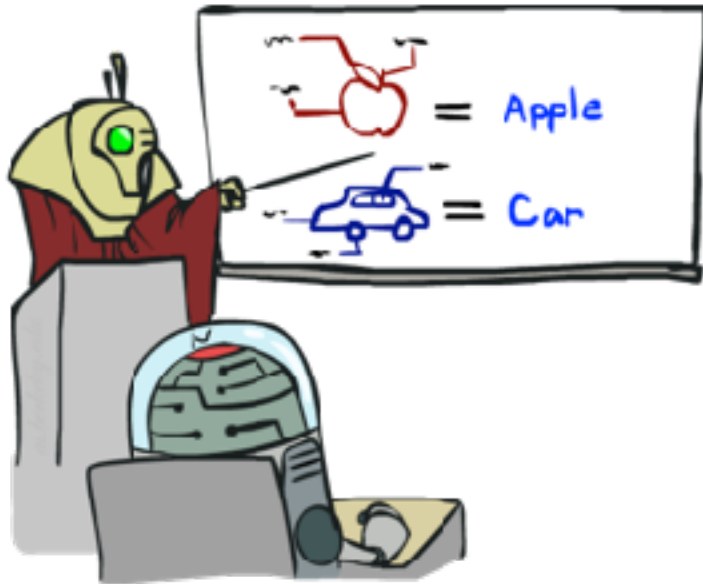
- Where do these tables come from?

Word	$P(w \text{spam})$	$P(w \text{ham})$	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4

$P(\text{spam} \mid w) = 98.9$

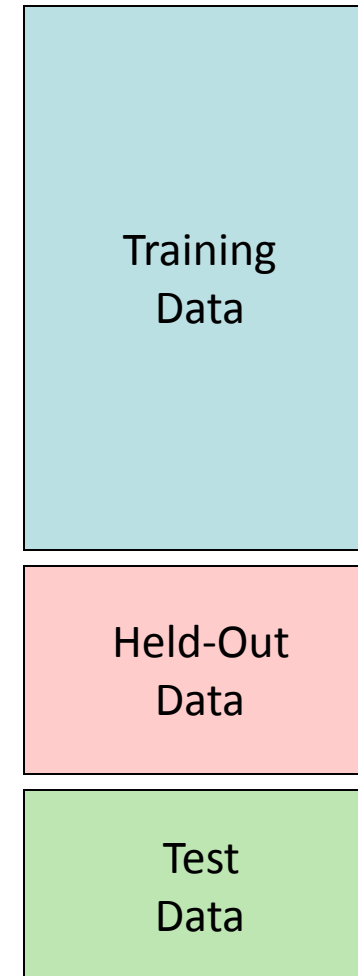


# Training and Testing

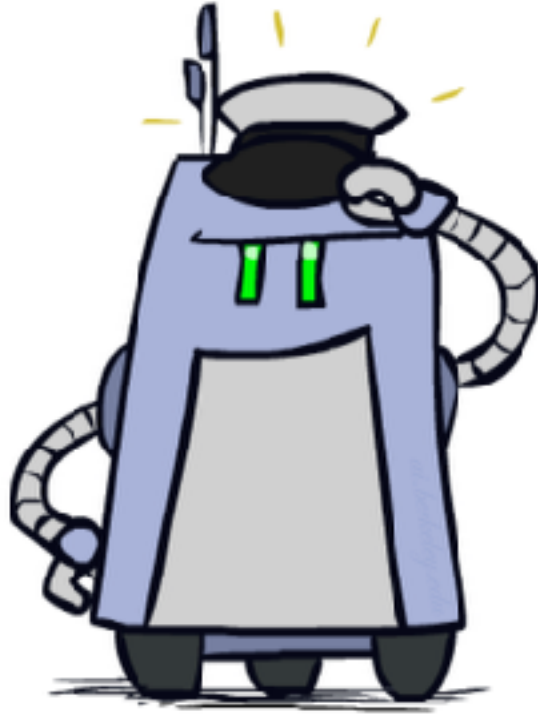
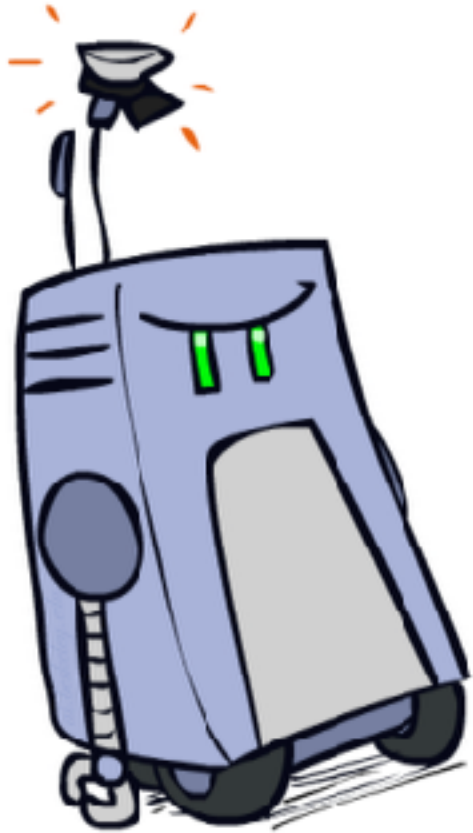


# Important Concepts

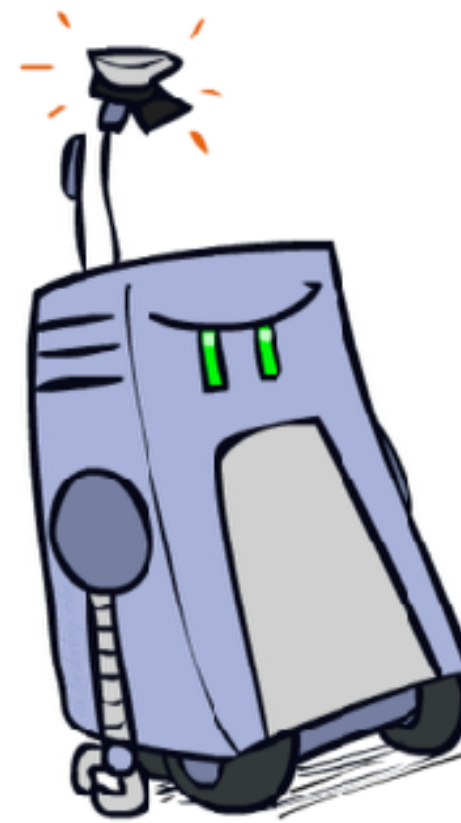
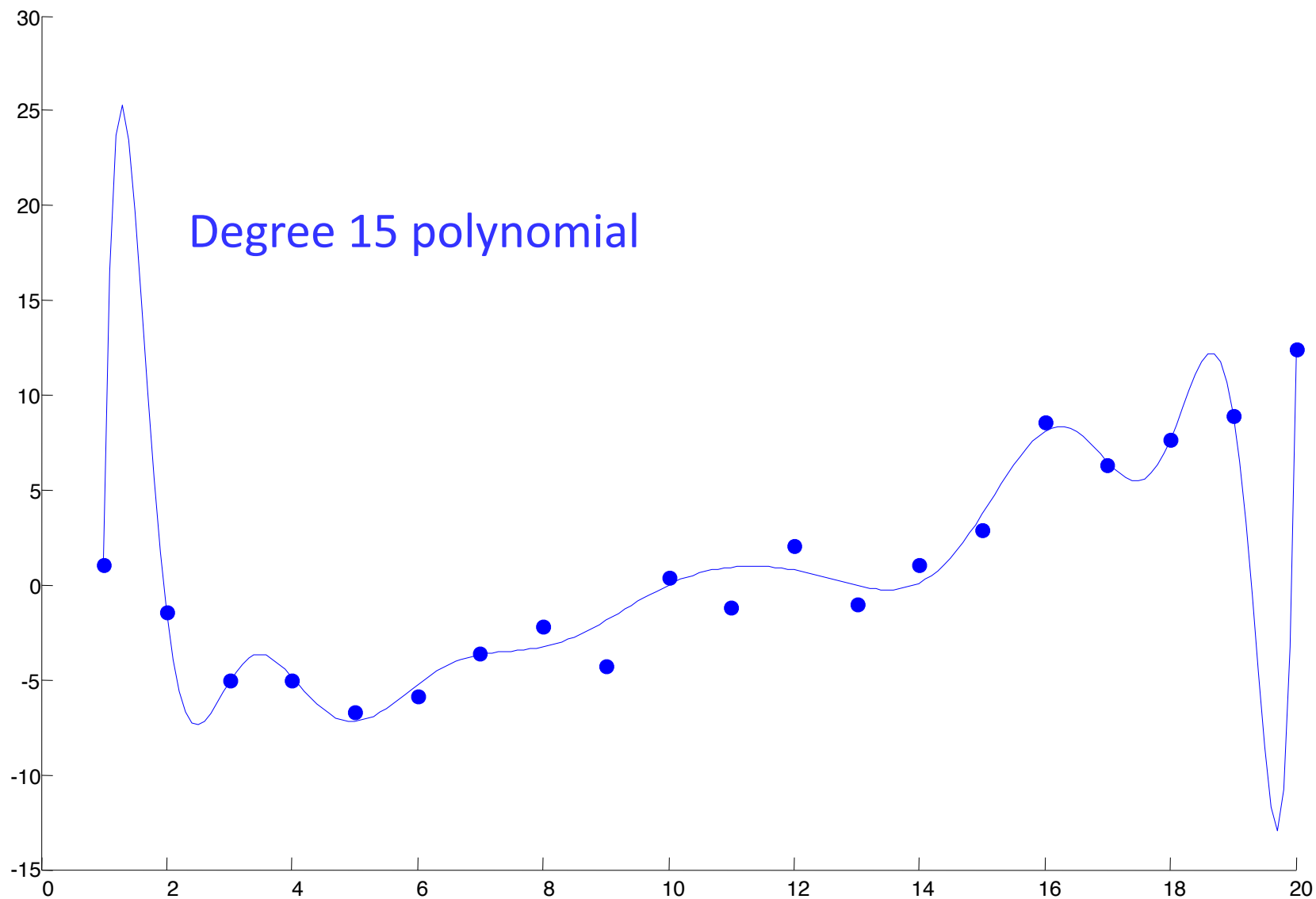
- **Data: labeled instances, e.g. emails marked spam/ham**
  - Training set
  - Held out set
  - Test set
- **Features: attribute-value pairs which characterize each  $x$**
- **Experimentation cycle**
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - Compute accuracy of 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 — tuning on held out data helps to avoid this
  - We’ll investigate overfitting and generalization formally in a few lectures



# Generalization and Overfitting



# Overfitting



# Example: Overfitting

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

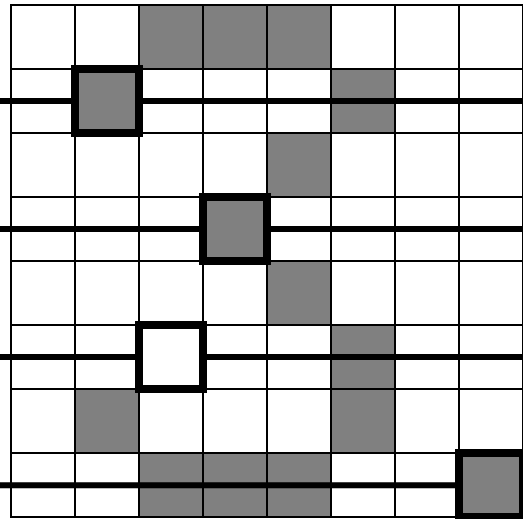
$P(C = 2) = 0.1$

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

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

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

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



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

$P(C = 3) = 0.1$

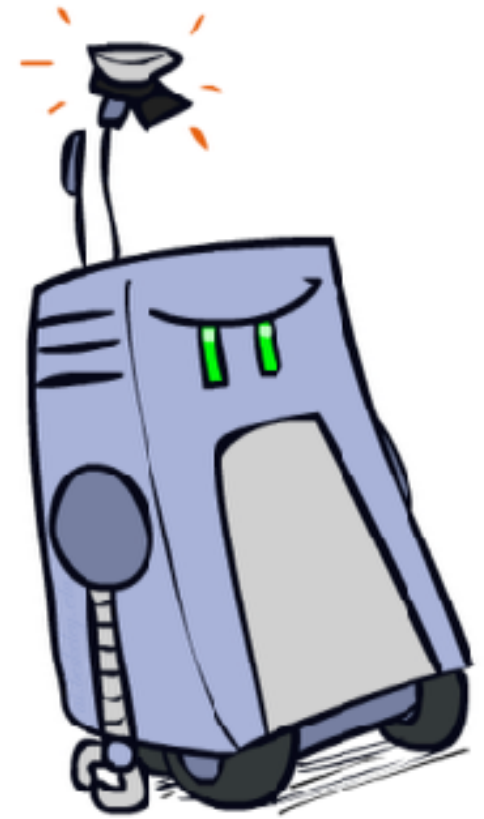
$P(\text{on}|C = 3) = 0.8$

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

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

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

*2 wins!!*



# Example: Overfitting

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

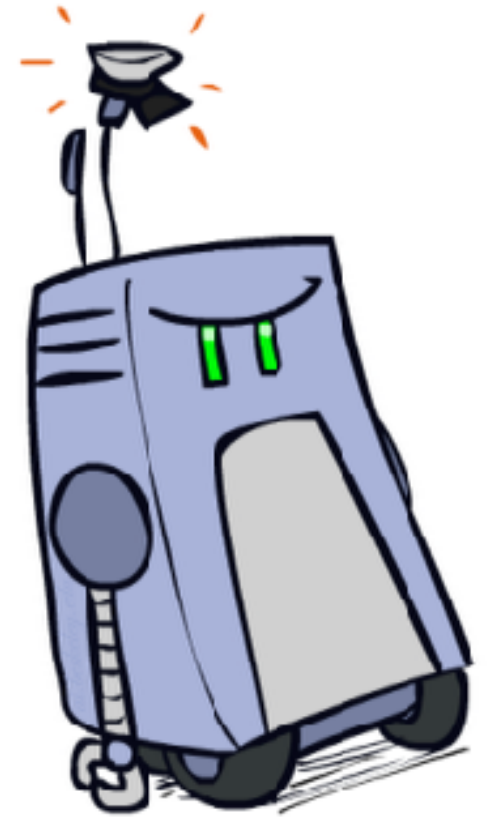
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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?*

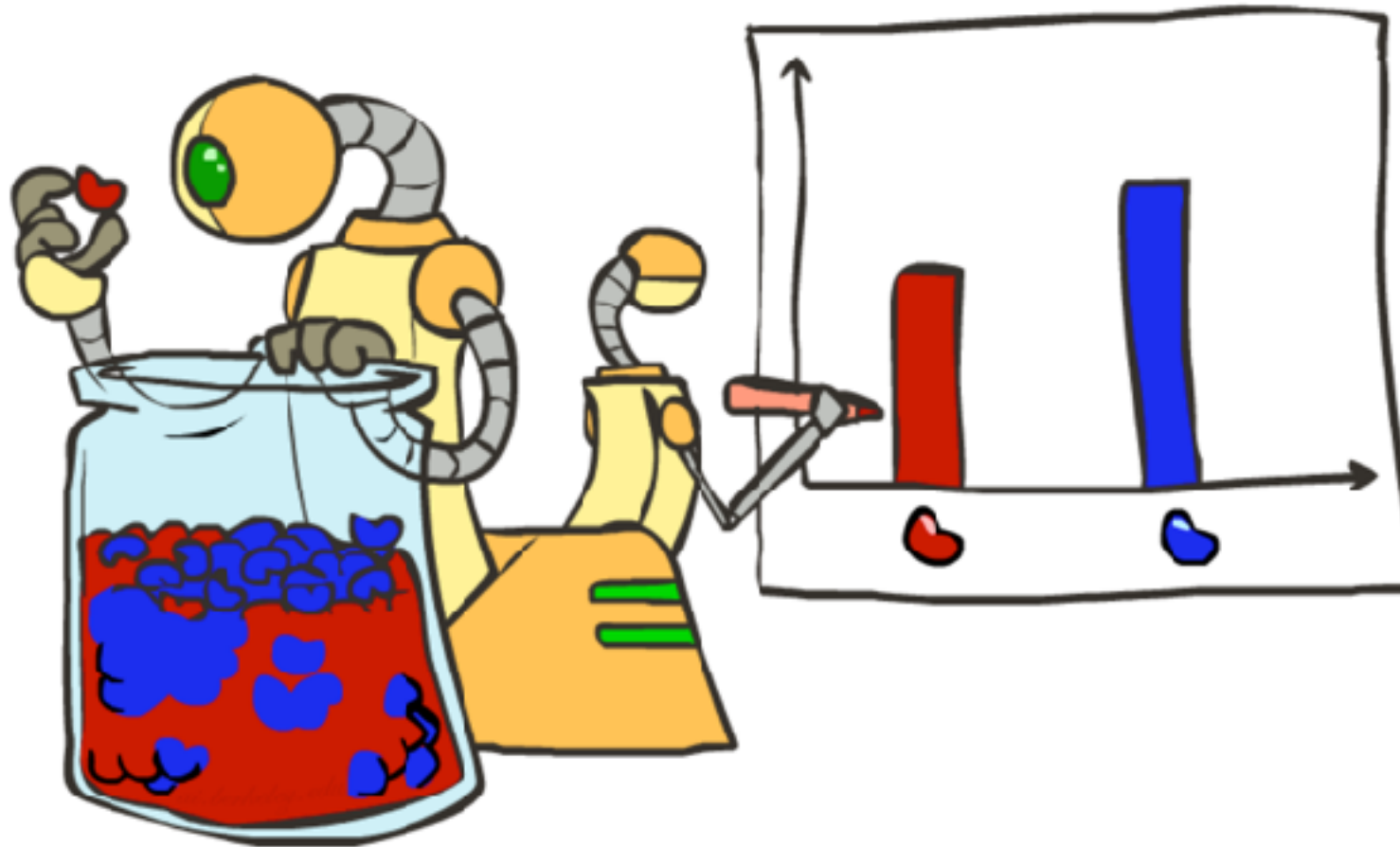


# Generalization and Overfitting

---

- 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
  - 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

# Parameter Estimation

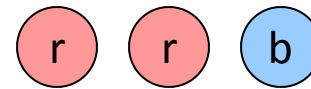




# Parameter Estimation

- 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:

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



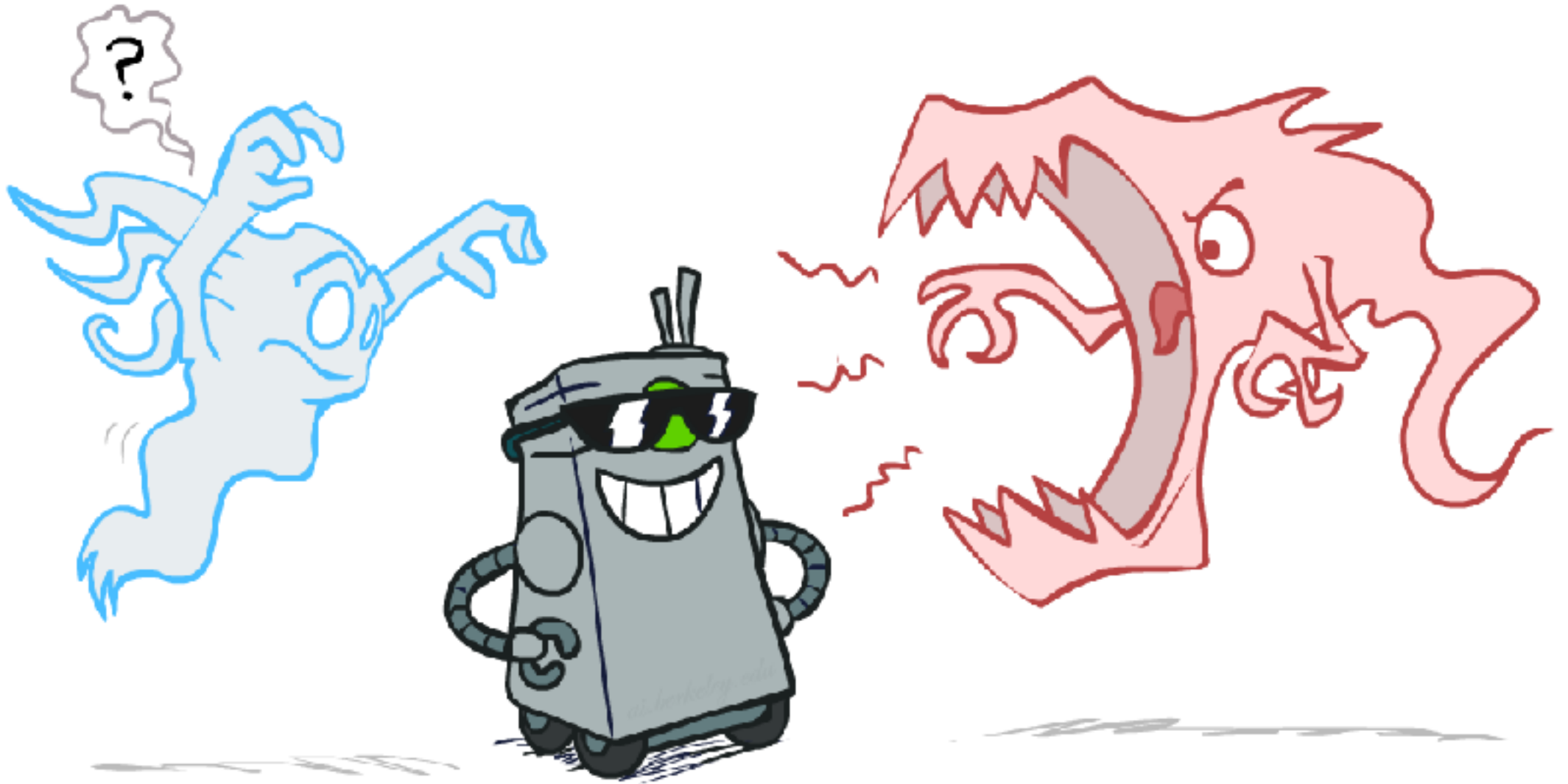
$$P_{\text{ML}}(\textcolor{red}{r}) = 2/3$$

- This is the estimate that maximizes the *likelihood of the data*

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



# Smoothing



# Maximum Likelihood?

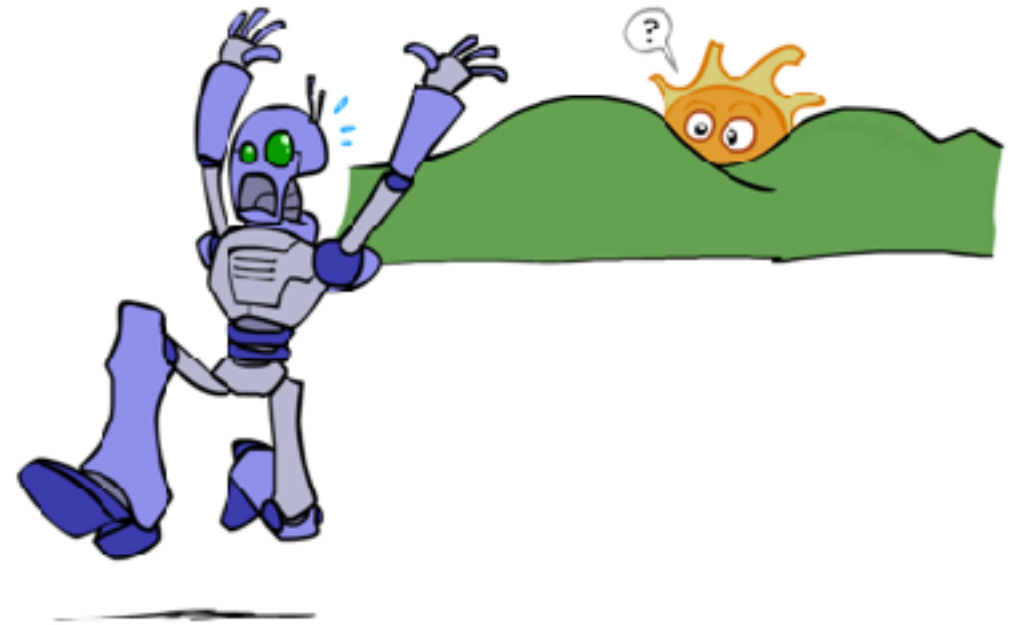
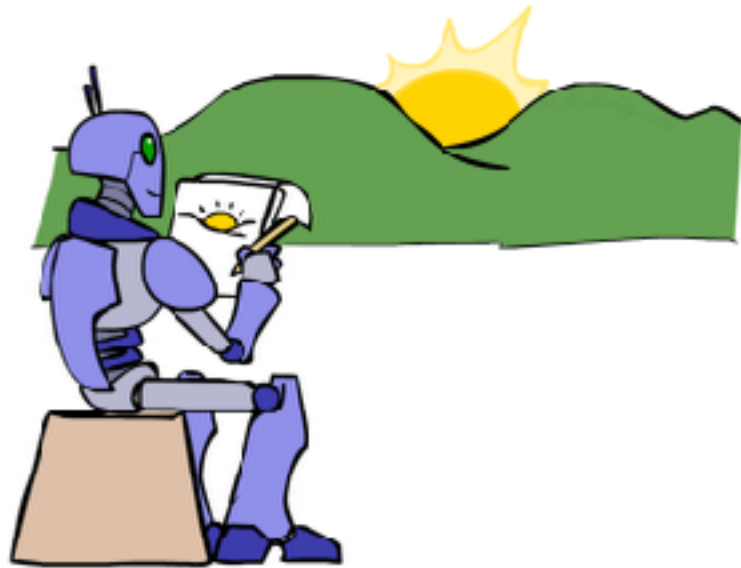
- Relative frequencies are the maximum likelihood estimates

$$\begin{aligned}\theta_{ML} &= \arg \max_{\theta} P(\mathbf{X}|\theta) \\ &= \arg \max_{\theta} \prod_i P_{\theta}(X_i)\end{aligned} \quad \Rightarrow \quad P_{ML}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

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

$$\begin{aligned}\theta_{MAP} &= \arg \max_{\theta} P(\theta|\mathbf{X}) \\ &= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)/P(\mathbf{X}) \quad \Rightarrow \quad \text{????} \\ &= \arg \max_{\theta} P(\mathbf{X}|\theta)P(\theta)\end{aligned}$$

# Unseen Events



# Laplace Smoothing

- Laplace's estimate:

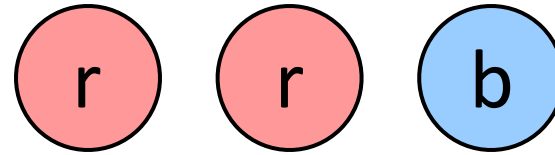
- Pretend you saw every outcome once more than you actually did

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

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

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

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



- Can derive this estimate with *Dirichlet priors*

# Laplace Smoothing

- Laplace's estimate (extended):

- Pretend you saw every outcome  $k$  extra times

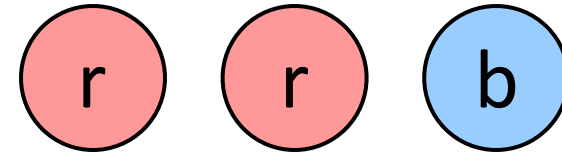
$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

- 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  $\alpha$  is 0? 1?

# Spam Filtering: Smoothing

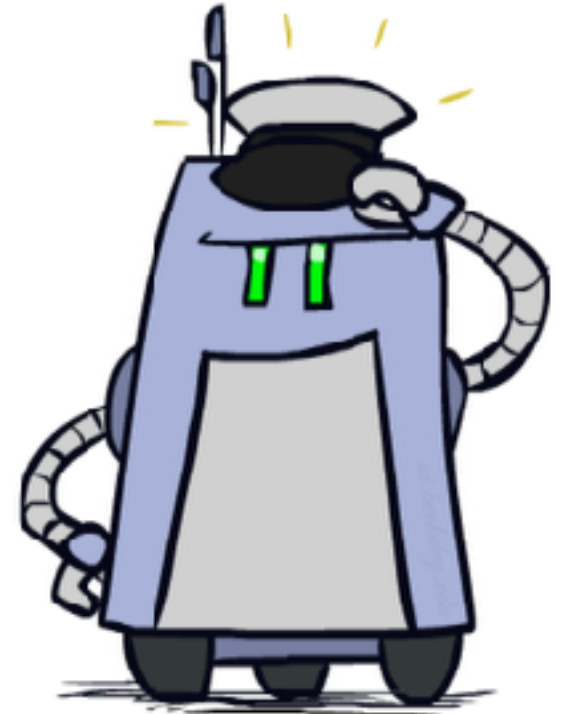
- For real classification problems, smoothing is critical
- New odds ratios:

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

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

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

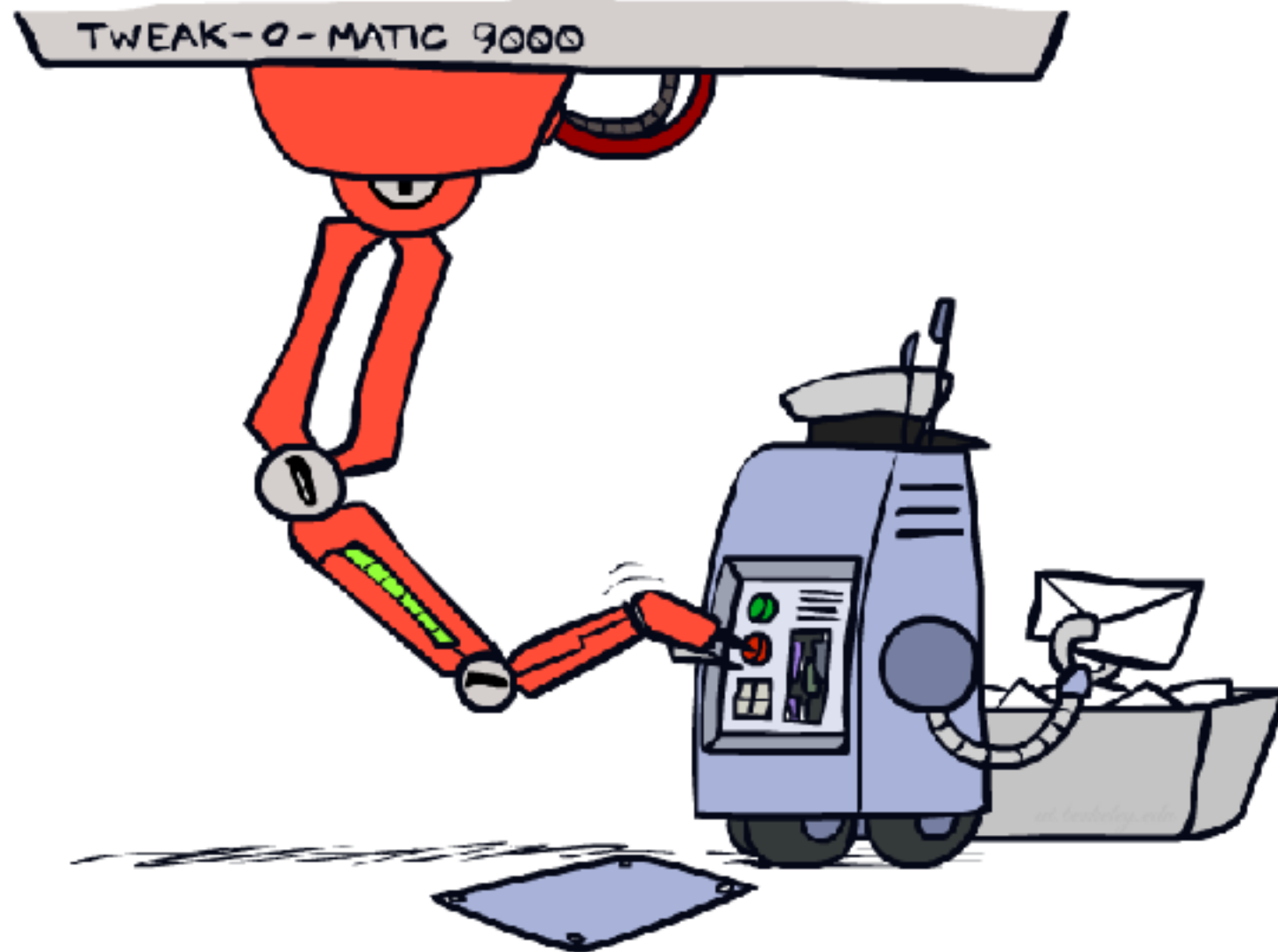
verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
<FONT>	:	26.9
money	:	26.5
...		



*Do these make more sense?*

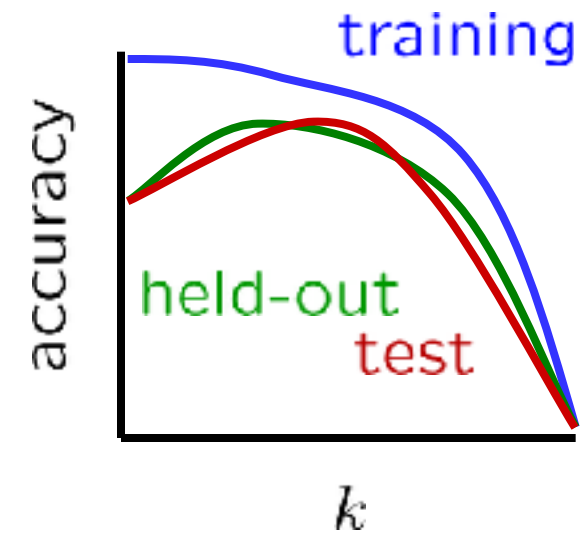


# Tuning

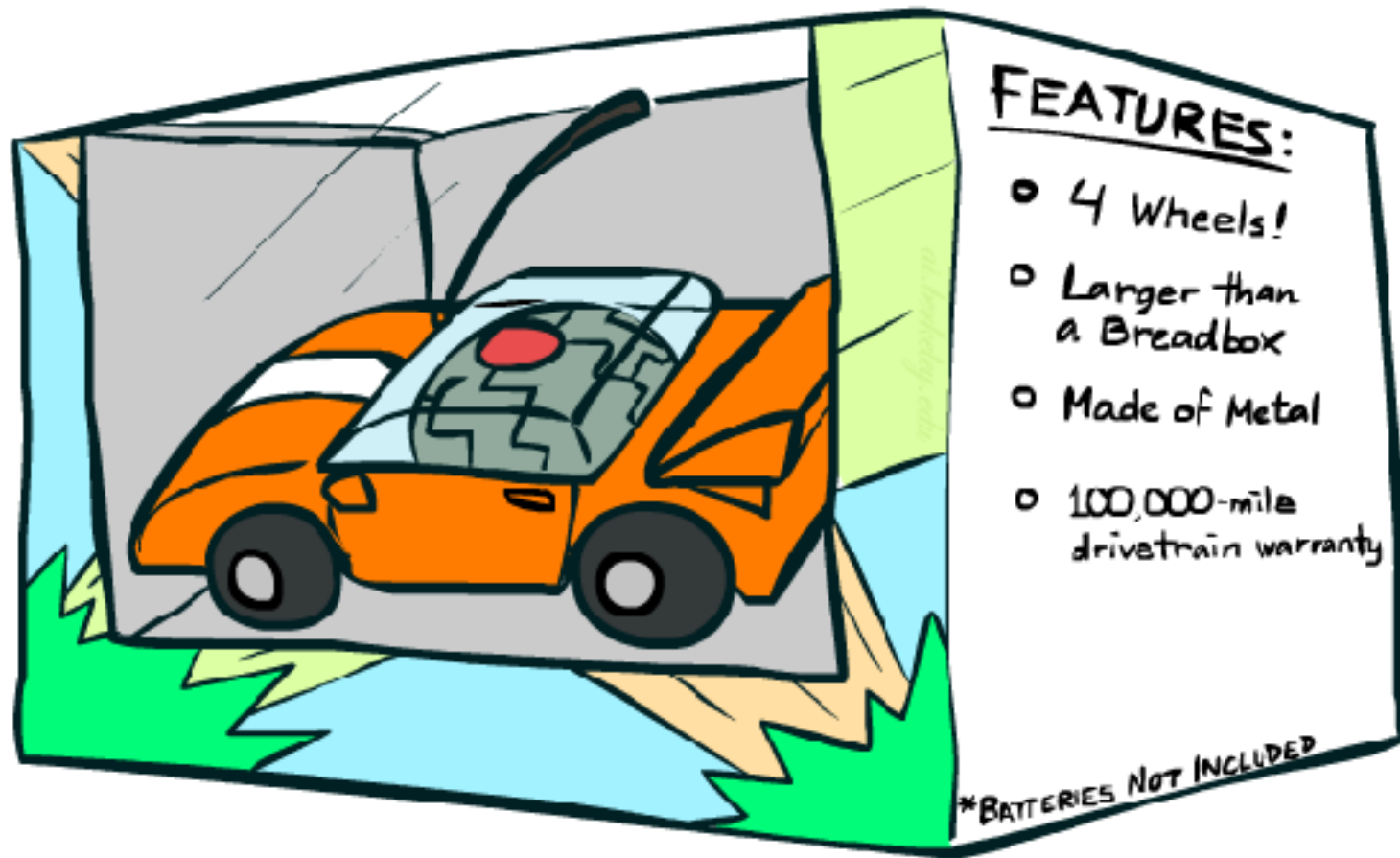


# Tuning on Held-Out Data

- 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,  $k$ ,  $\alpha$
- 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



# 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 . . .

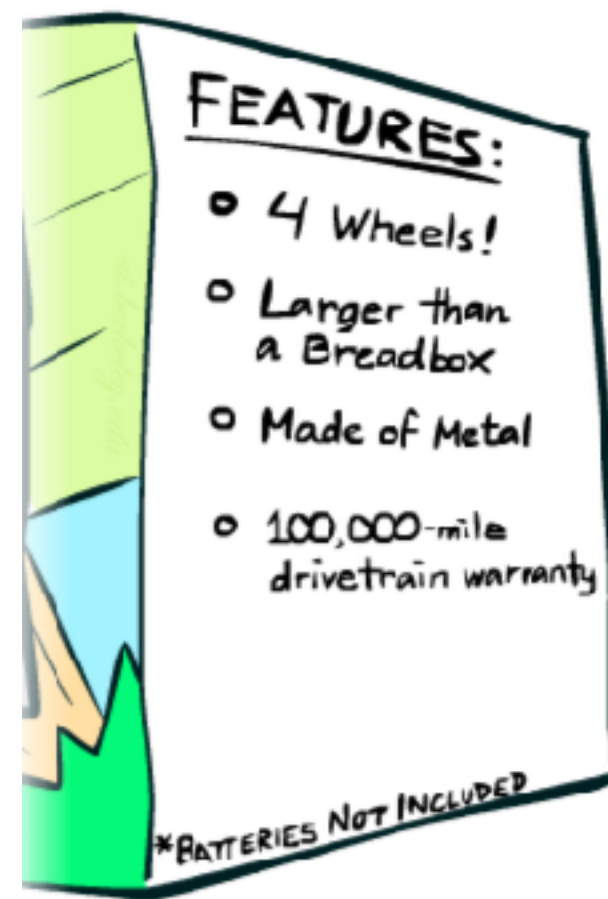
. . . To receive your \$30 Amazon.com promotional certificate, click through to

<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 model
- Next class we'll talk about classifiers which let you easily add arbitrary features more easily



# 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

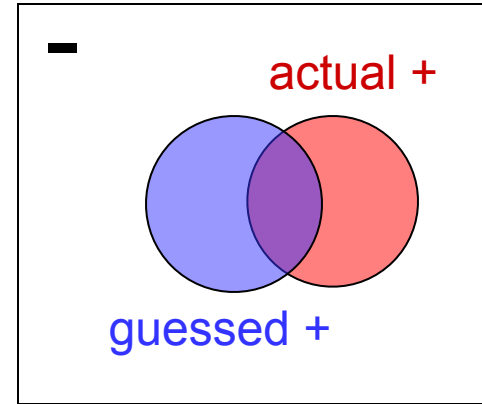
# Summary

---

- 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

# 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:  $2 \text{ correct} / 5 \text{ guessed} = 0.4$
  - Recall:  $2 \text{ correct} / 3 \text{ true} = 0.67$
- 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  $r$ :

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

