CS 188: Artificial Intelligence

Naïve Bayes

University of California, Berkeley

[These slides were created by Dan Klein, Pieter Abbeel, Anca Dragang, Sergey Levine, with some materials from A. Farhadi. All CS188 materials are at http://ai.berkeley.edu.]

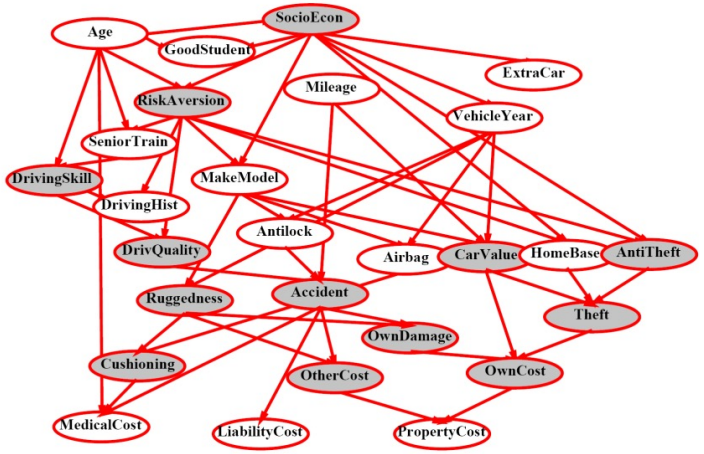
Machine Learning

- 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)
- Today:
  - model-based classification with Naive Bayes
- Next lectures:
  - Regression and perceptrons, optimization in ML, (deep) neural networks

Learning: Why?

- Learning is ***essential*** in unknown environments – when the agent designer lacks omniscience

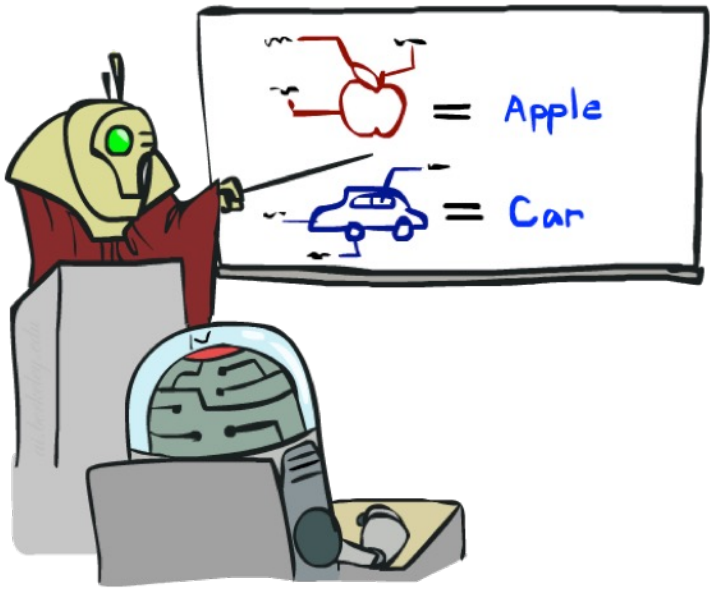
- Learning is *useful* as a system construction method, i.e., expose the system to reality rather than trying to write it down

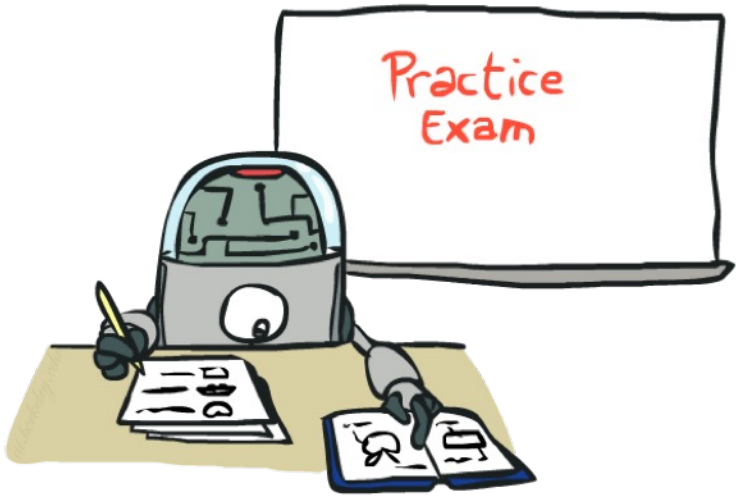


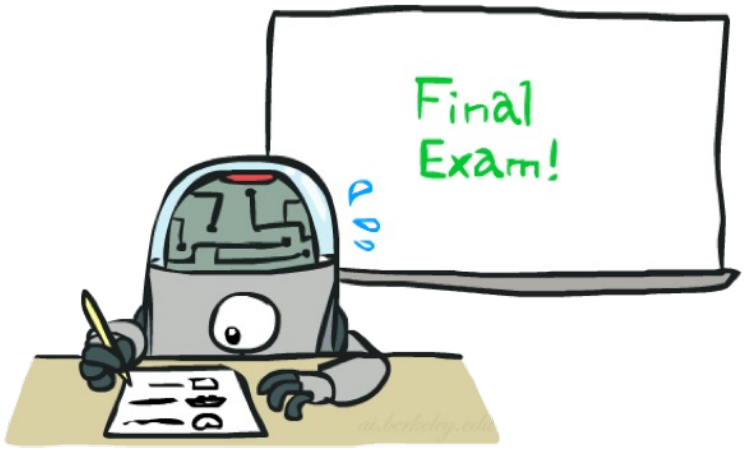
Multiple Types of Learning Problems

- **Supervised learning:** correct answers for each training instance
  - **Classification:** learning predictor with *discrete* outputs
  - **Regression:** learning predictor with *real-valued* outputs
- **Reinforcement learning:** reward sequence, no correct answers
- **Unsupervised learning:** "just make sense of the data"

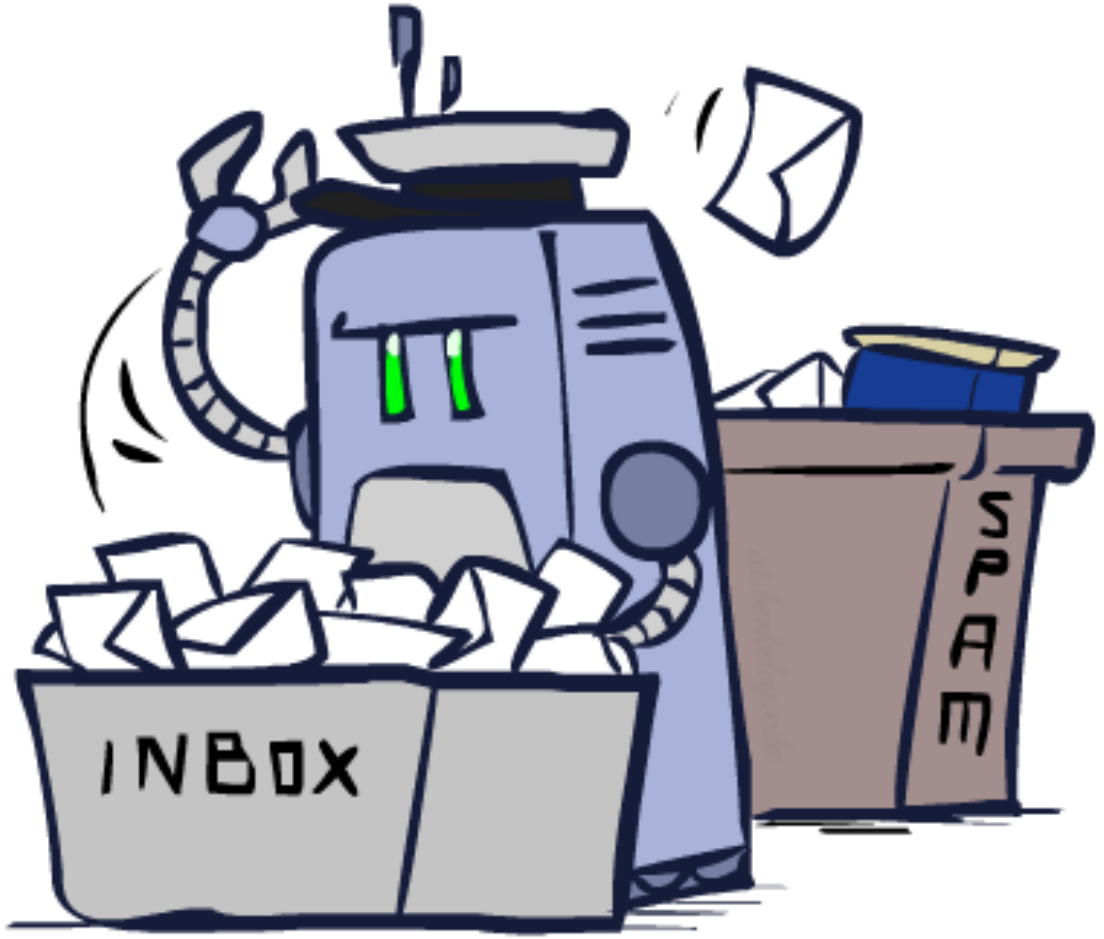
Training and Testing







Classification



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. …

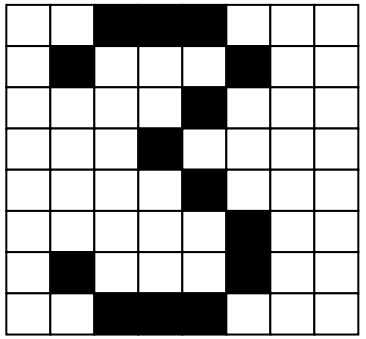
TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

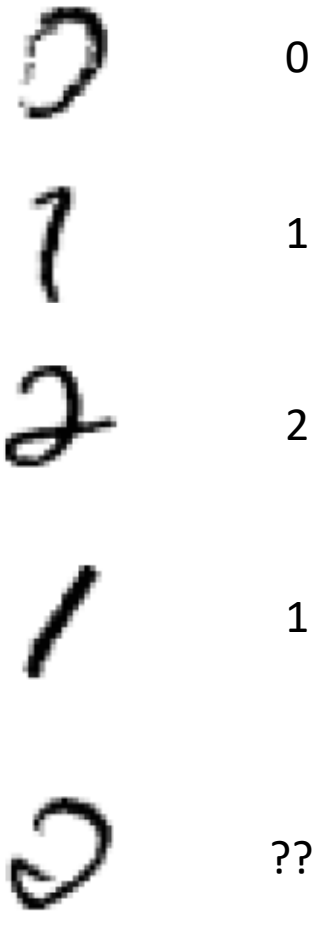
99 MILLION EMAIL ADDRESSES FOR ONLY $99

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



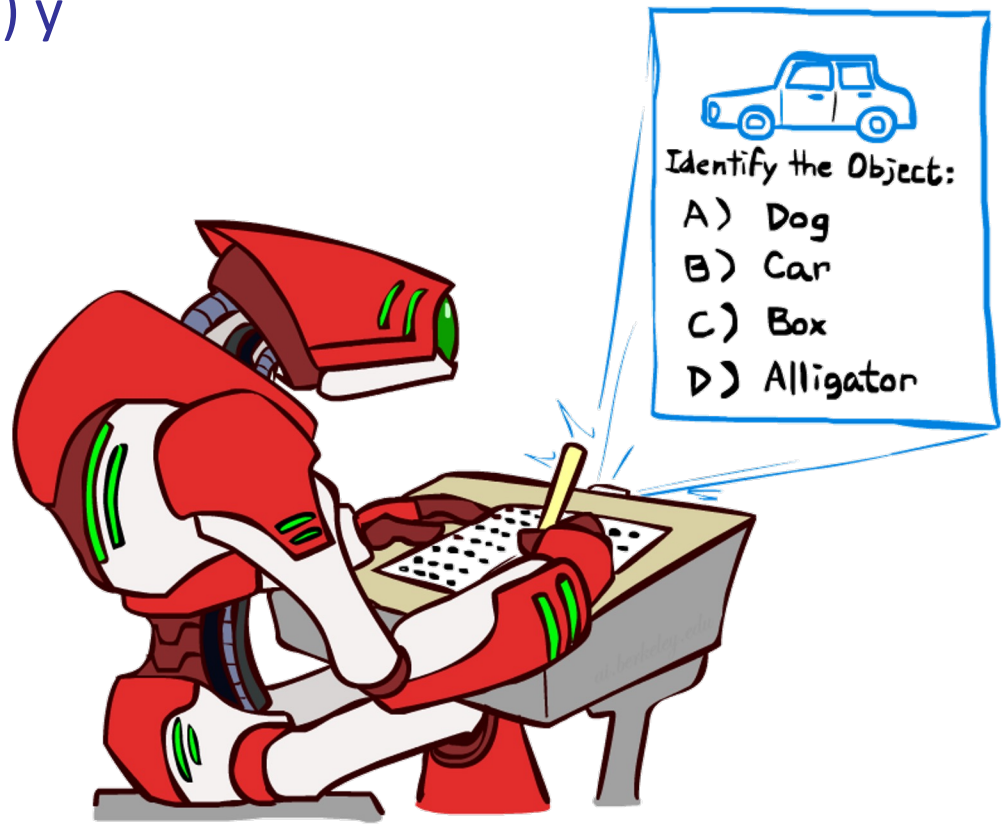


- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images
- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …

Other Classification Tasks

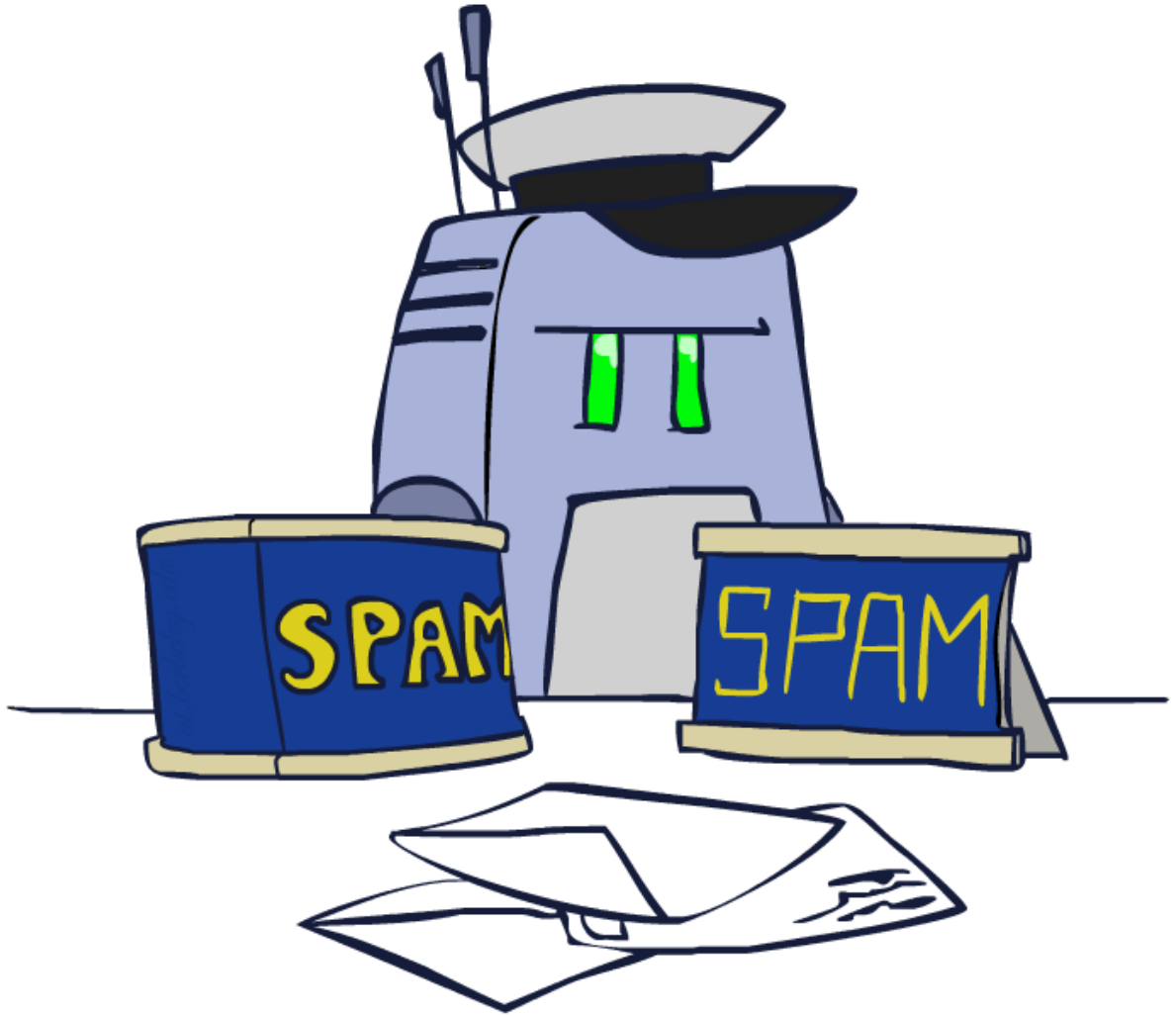
- Classification: given inputs x, predict labels (classes) y

- 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
  - Customer service email routing
  - … many more



- Classification is an important commercial technology!

Model-Based Classification

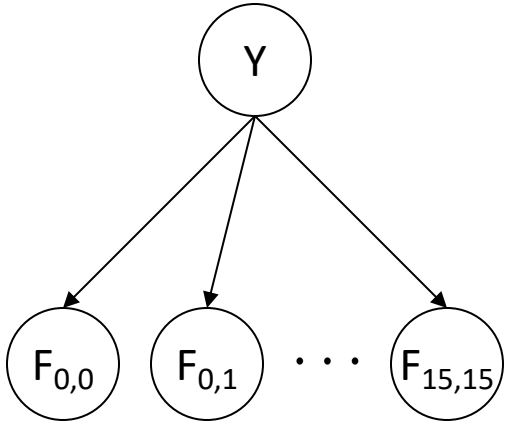


Model-Based Classification

- 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 <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

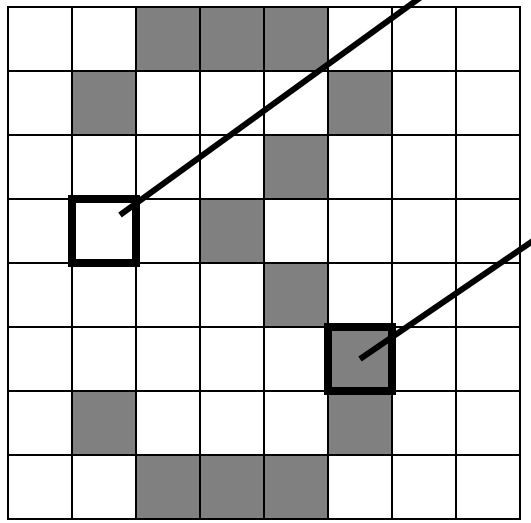
- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn?

Naïve Bayes for Digits: Parameters

$P(Y)$

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



$P(F_{3,1} = on|Y)$

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

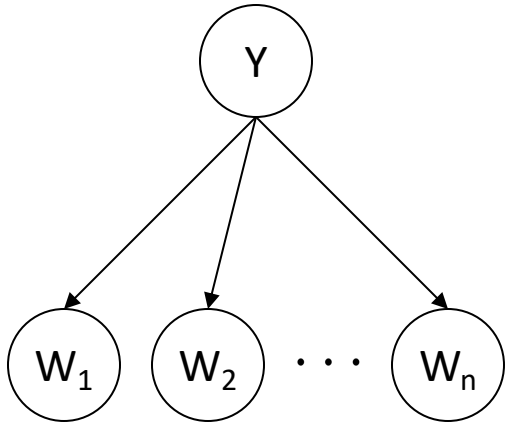
$P(F_{5,5} = on|Y)$

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 Text

- Bag-of-words Naïve Bayes:
  - Features: W_i is the word at positon 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

how many variables are there?
how many values?



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

W_i = 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

product our try please

please try our product

Naïve Bayes for Text: Parameters

- 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
...
```

General Naïve Bayes

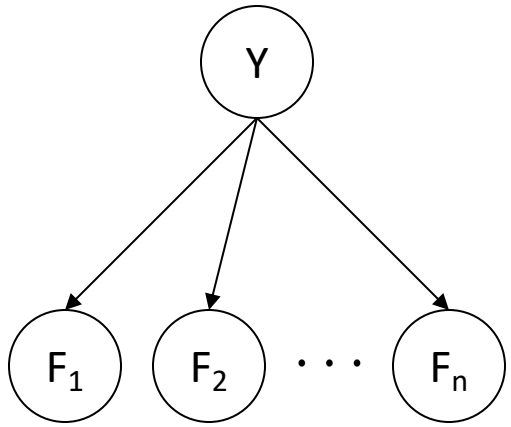
- A general Naive Bayes model:

|Y| parameters

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

|Y| x |F|^n values

n x |F| x |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

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

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

  - Step 2: sum to get probability of evidence
  - Step 3: normalize by dividing Step 1 by Step 2

Example: Spam Filtering

- Naïve Bayes spam filter
- Data:
  - Collection of emails, labeled spam or ham
  - Note: someone has to hand label all this data!
  - Split into training, held-out, test sets
- Classifiers
  - Learn on the training set
  - (Tune it on a held-out set)
  - Test it on new emails

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: Spam Filtering

- Model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$
- 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
...
```

Spam Example

$P(Y)$

$P(W_1|Y)$

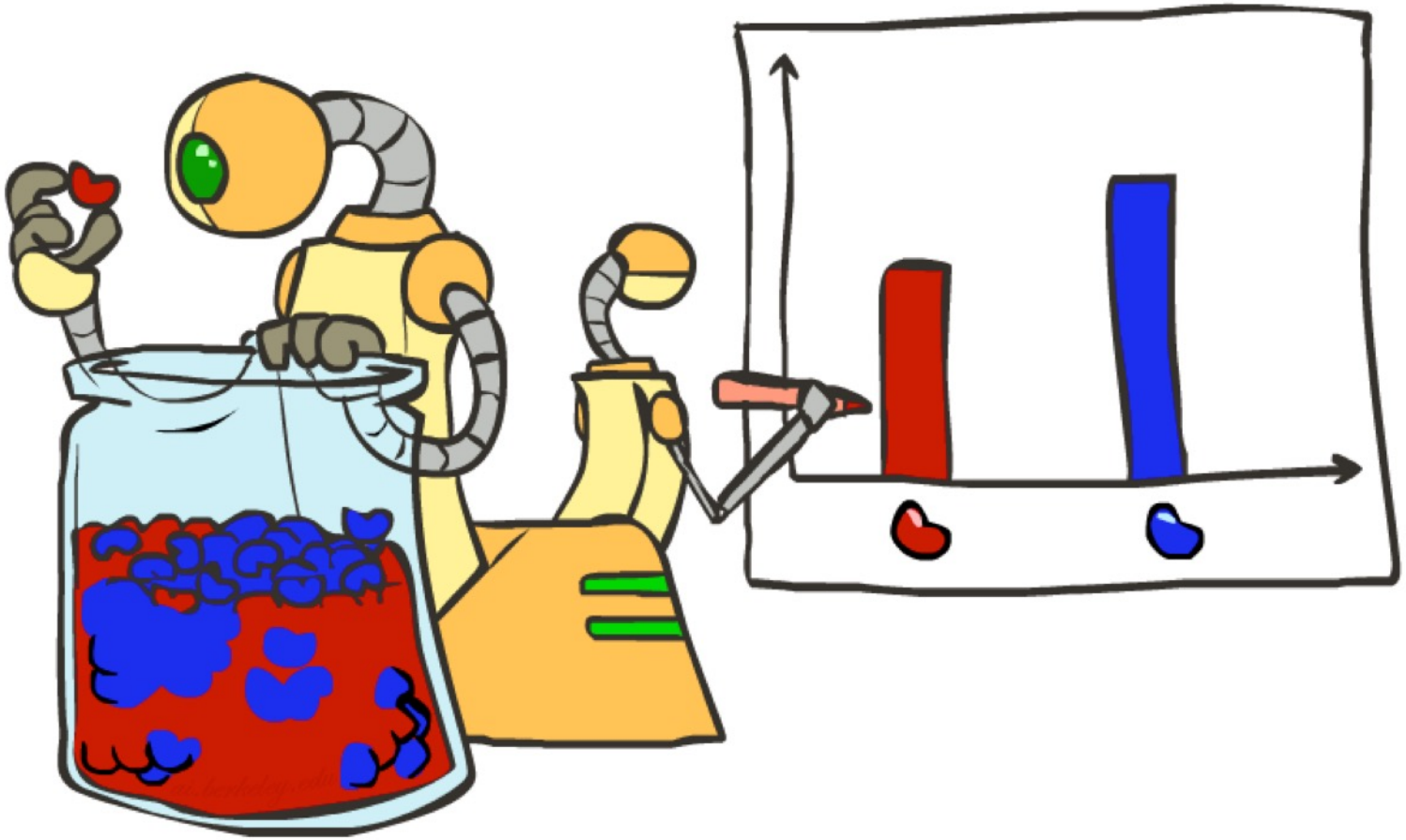
$P(W_2|Y)$

Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4

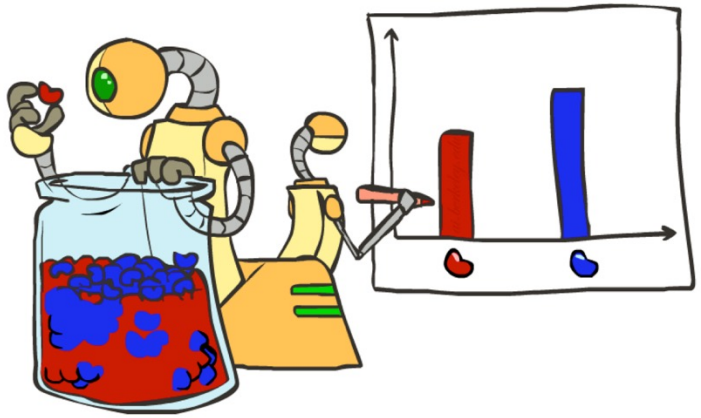
General Naïve Bayes

- 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 $P(Y|F_1 \ldots F_n)$
    - 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 with Maximum Likelihood



- 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:

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

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

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

$$L(x,\theta) = \prod_i P_\theta(x_i) = \theta \cdot \theta \cdot (1-\theta)$$

$$P_\theta(x = \text{red}) = \theta$$

$$P_\theta(x = \text{blue}) = 1 - \theta$$

General Case: n outcomes

- P(Heads) = θ, P(Tails) = 1-θ

- Flips are *i.i.d.*: $D=\{x_i | i=1...n\}$, $P(D|\theta) = \Pi_i P(x_i|\theta)$
  - Independent events
  - Identically distributed according to unknown distribution
- Sequence *D* of α_H Heads and α_T Tails

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

Parameter Estimation with Maximum Likelihood

- **Data:** Observed set D of α_H Heads and α_T Tails
- **Hypothesis space:** Binomial distributions
- **Learning:** finding θ is an optimization problem
  - What's the objective function?

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

- **MLE:** Choose θ to maximize probability of *D*

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} P(\mathcal{D} \mid \theta) \\ &= \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta) \end{aligned}$$

Parameter Estimation with Maximum Likelihood

$$\begin{aligned}\widehat{\theta} &= \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta) \\ &= \arg\max_{\theta} \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}\end{aligned}$$

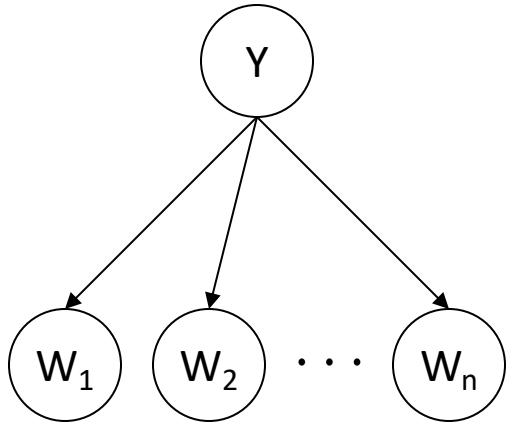
- Set derivative to zero, and solve!

$$\begin{aligned}\frac{d}{d\theta}\ln P(\mathcal{D} \mid \theta) &= \frac{d}{d\theta}\left[\ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}\right] \\ &= \frac{d}{d\theta}\left[\alpha_H \ln\theta + \alpha_T \ln(1-\theta)\right] \\ &= \alpha_H \frac{d}{d\theta}\ln\theta + \alpha_T \frac{d}{d\theta}\ln(1-\theta) \\ &= \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0\end{aligned}$$

$$\widehat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}$$

Maximum Likelihood for Naïve Bayes Spam Classifier

- **Model:**
  - Random variable $F_i = 1$ if *i*'th dictionary word is present in email
  - Random variable *Y is in {spam, ham} depending on email label*
- **Data *D*:**
  - N emails with $\mathrm{N_H}$ "hams" and $\mathrm{N_S}$ "spams"
  - $f_i^{(j)} = 1$ if *i*'th word appeared in email *j*
- **Parameters:**
  - Probability tables *P(Y)* and $P(F_i \mid Y)$
  - Collectively call them both θ
- **MLE:** Choose θ to maximize probability of *D* $\hat{\theta} = \arg\max_{\theta} P(\mathcal{D} \mid \theta)$



Maximum Likelihood for Naïve Bayes Spam Classifier*

- **Let's find single parameter P(Fi | Y = ham) (this will be our θ):**
  - Denote L(θ) = P(D | θ) for ease of notation

$$\mathscr{L}(\theta) = \prod_{j=1}^{N_h} P(F_i = f_i^{(j)} | Y = ham) = \prod_{j=1}^{N_h} \theta^{f_i^{(j)}} (1-\theta)^{1-f_i^{(j)}}$$

$$P(F_i = f_i^{(j)} | Y = ham) = \begin{cases} \theta & \text{if } f_i^{(j)} = 1 \\ (1-\theta) & \text{if } f_i^{(j)} = 0 \end{cases}$$

Maximum Likelihood for Naïve Bayes Spam Classifier*

$$\mathscr{L}(\theta) = \prod_{j=1}^{N_h} P(F_i = f_i^{(j)} | Y = ham) = \prod_{j=1}^{N_h} \theta^{f_i^{(j)}} (1-\theta)^{1-f_i^{(j)}}$$

$$\begin{aligned}
\log \mathscr{L}(\theta) &= \log \left(\prod_{j=1}^{N_h} \theta^{f_i^{(j)}} (1-\theta)^{1-f_i^{(j)}} \right) \\
&= \sum_{j=1}^{N_h} \log \left(\theta^{f_i^{(j)}} (1-\theta)^{1-f_i^{(j)}} \right) \\
&= \sum_{j=1}^{N_h} \log \left(\theta^{f_i^{(j)}} \right) + \sum_{j=1}^{N_h} \log \left((1-\theta)^{1-f_i^{(j)}} \right) \\
&= \log(\theta) \sum_{j=1}^{N_h} f_i^{(j)} + \log(1-\theta) \sum_{j=1}^{N_h} (1-f_i^{(j)})
\end{aligned}$$

Maximum Likelihood for Naïve Bayes Spam Classifier *

$$\frac{\partial}{\partial\theta}\left(\log(\theta)\sum_{j=1}^{N_h} f_i^{(j)} + \log(1-\theta)\sum_{j=1}^{N_h}(1-f_i^{(j)})\right) = 0$$

$$\frac{1}{\theta}\sum_{j=1}^{N_h} f_i^{(j)} - \frac{1}{(1-\theta)}\sum_{j=1}^{N_h}(1-f_i^{(j)}) = 0$$

$$\frac{1}{\theta}\sum_{j=1}^{N_h} f_i^{(j)} = \frac{1}{(1-\theta)}\sum_{j=1}^{N_h}(1-f_i^{(j)})$$

$$(1-\theta)\sum_{j=1}^{N_h} f_i^{(j)} = \theta\sum_{j=1}^{N_h}(1-f_i^{(j)})$$

$$\sum_{j=1}^{N_h} f_i^{(j)} - \theta\sum_{j=1}^{N_h} f_i^{(j)} = \theta\sum_{j=1}^{N_h} 1 - \theta\sum_{j=1}^{N_h} f_i^{(j)}$$

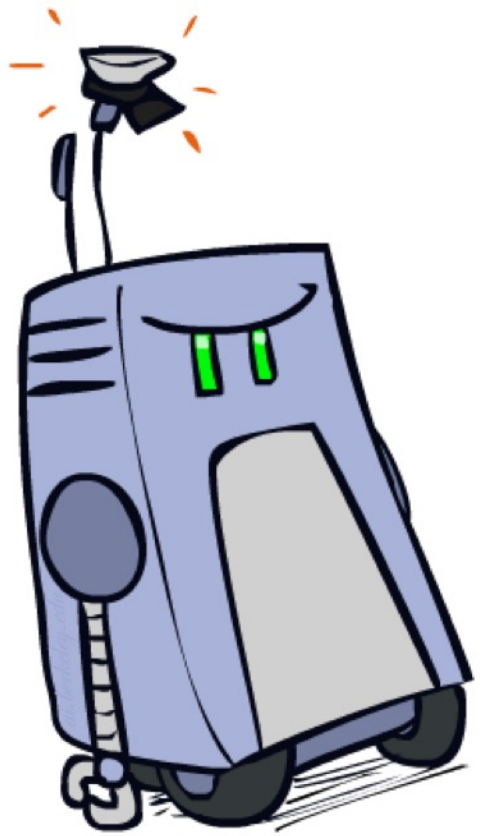
$$\sum_{j=1}^{N_h} f_i^{(j)} = \theta \cdot N_h$$

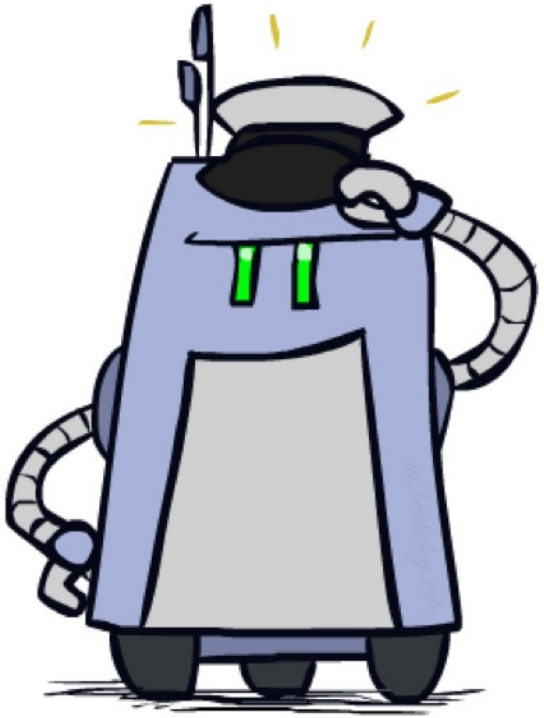
P(Fi | Y = ham): $$\theta = \frac{1}{N_h}\sum_{j=1}^{N_h} f_i^{(j)}$$

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..

Underfitting and Overfitting





Example: Overfitting

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

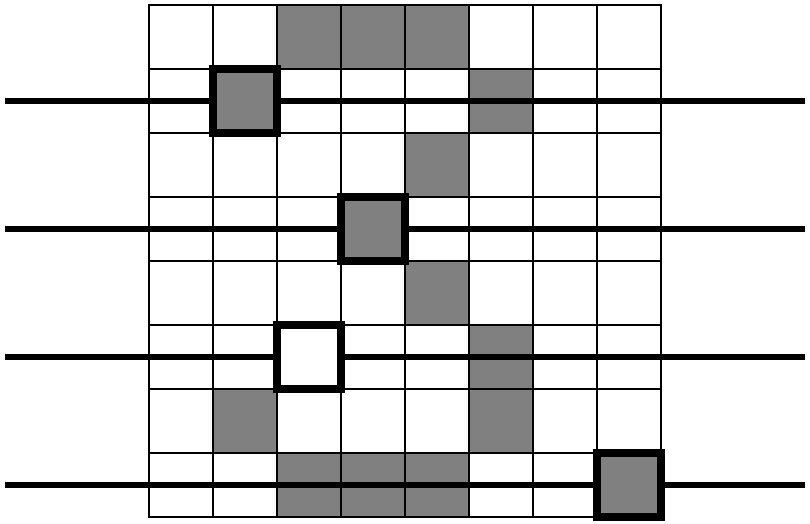
$P(C = 2) = 0.1$

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

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

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

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



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

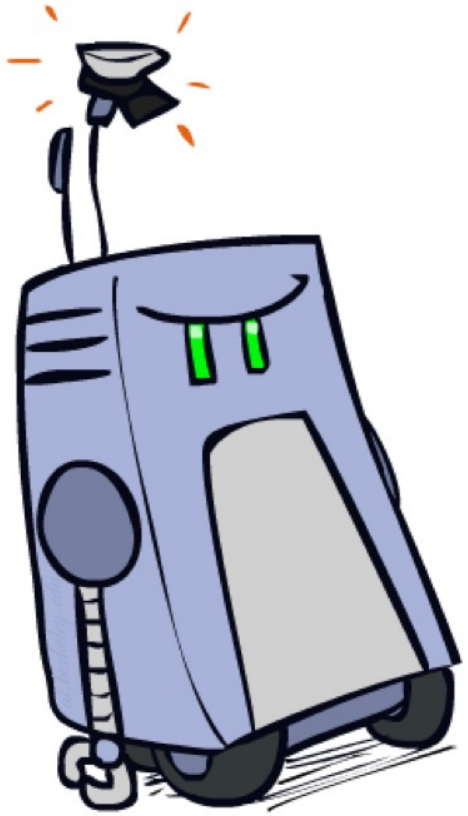
$P(C = 3) = 0.1$

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

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

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

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



2 wins!!

Example: Overfitting

- *relative* probabilities (odds ratios):

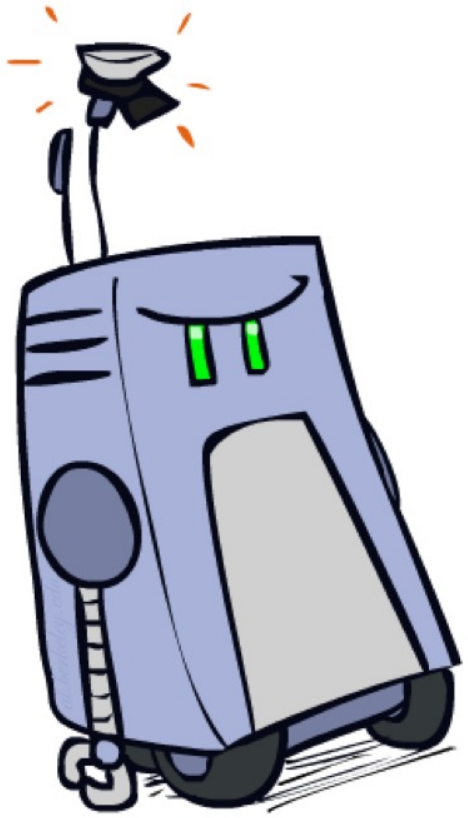
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

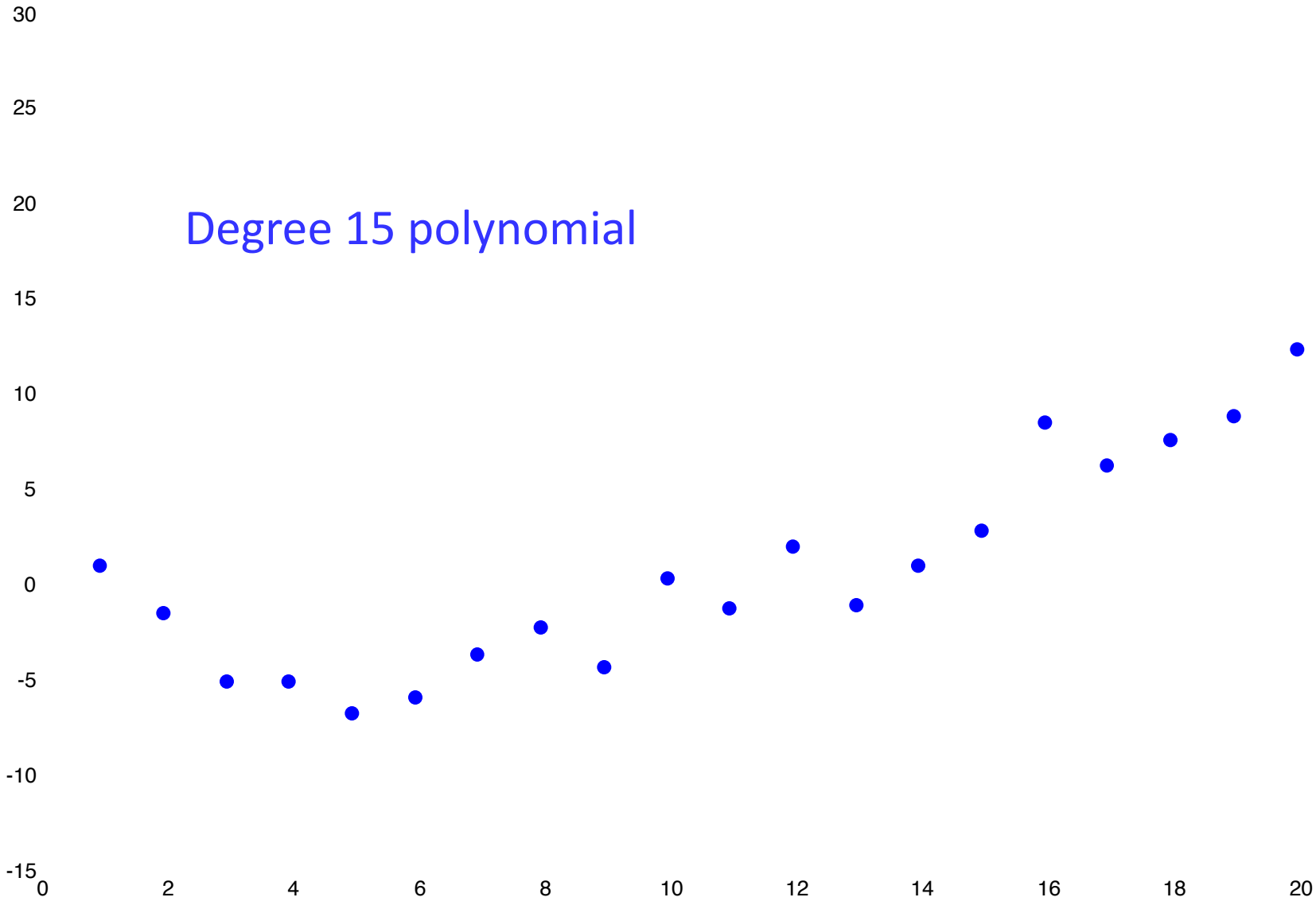
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

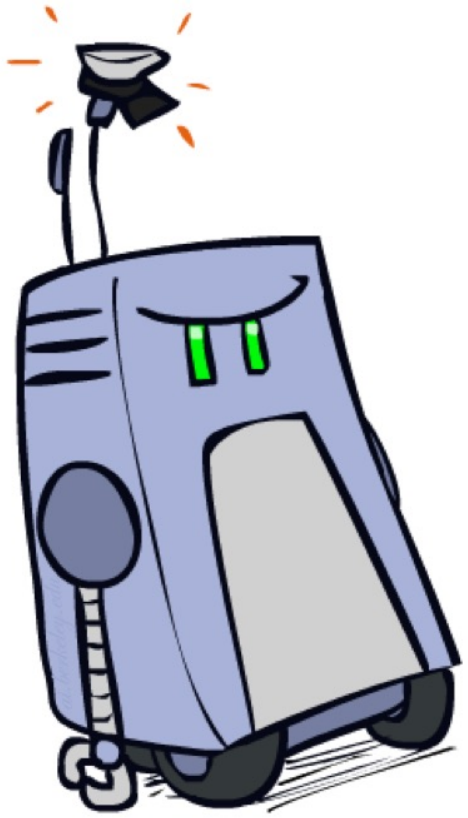
```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

What went wrong here?



Overfitting





Generalization and Overfitting

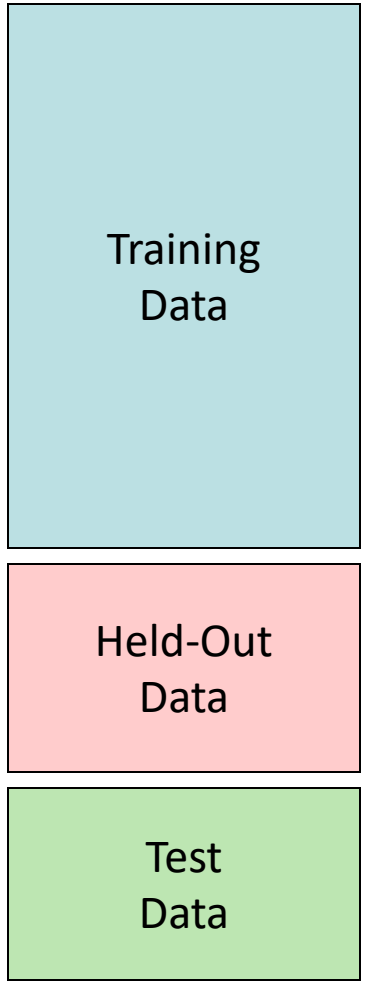
- Relative frequency parameters will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Unlikely that every occurrence of "minute" is 100% spam
  - Unlikely that every occurrence of "seriously" is 100% ham
  - 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

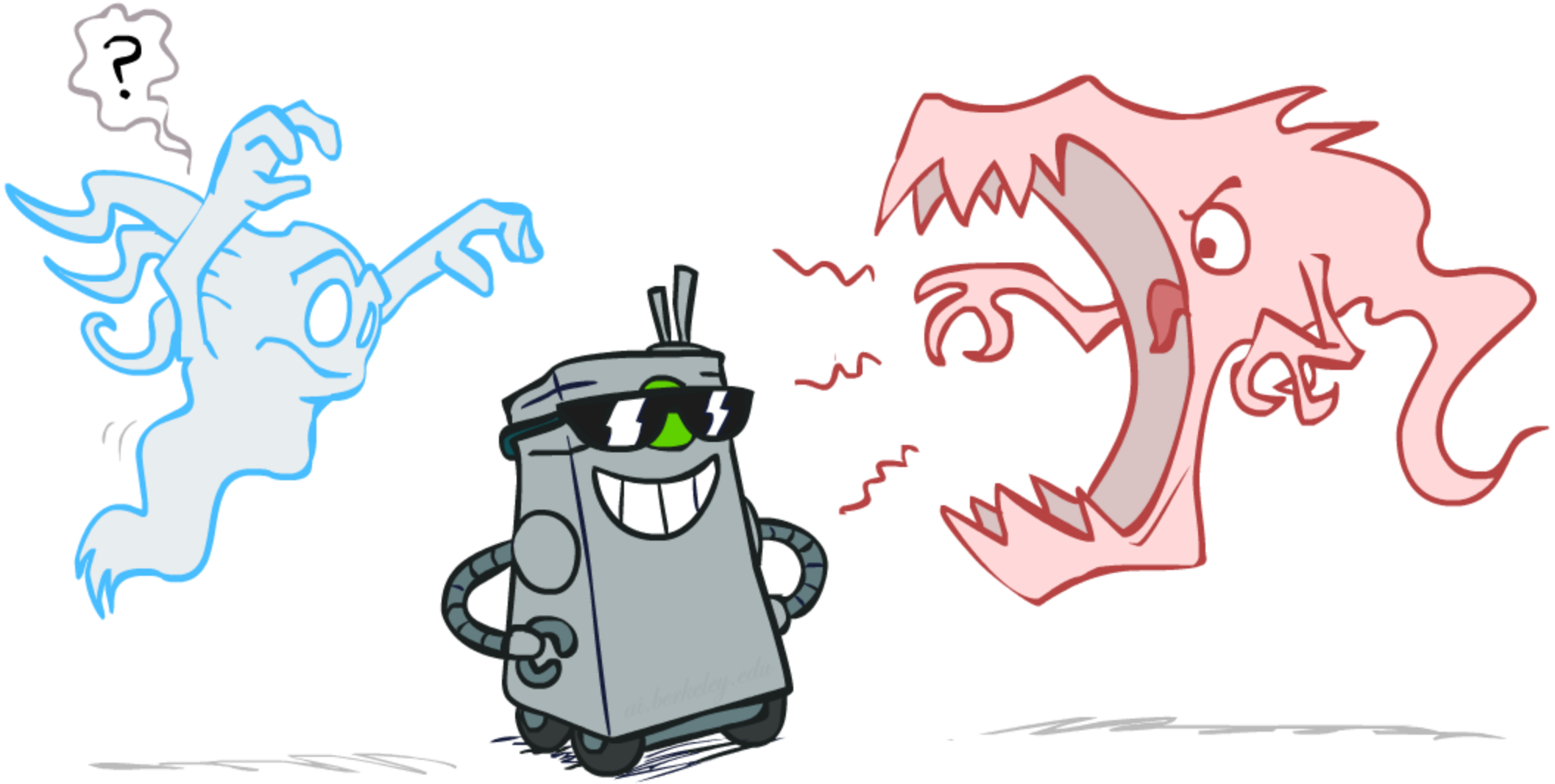
- To generalize better: we need to smooth or regularize the estimates

Important Concepts

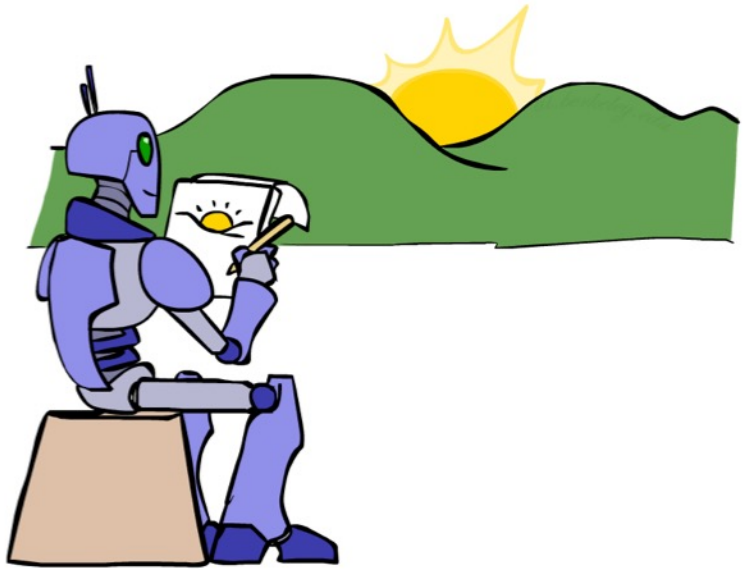
- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Features: attribute-value pairs which characterize each input
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - (Tune hyperparameters on held-out set)
  - 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
  - Underfitting: fits the training set poorly

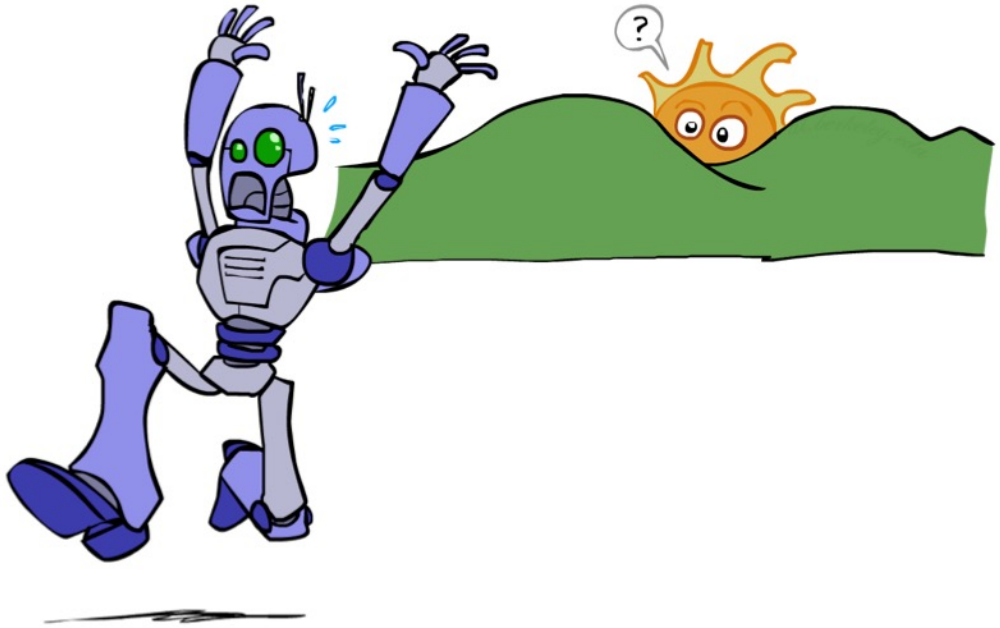


Smoothing



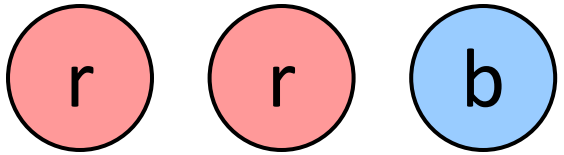
Unseen Events





Laplace Smoothing

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



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

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

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

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

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

Laplace Smoothing

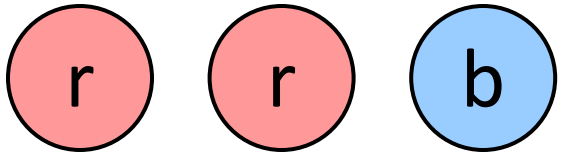
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

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

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

- Laplace for conditionals:
  - Smooth each condition independently:

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



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

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

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

Laplace Smoothing Can Be More Formally Derived

- Relative frequencies are the maximum likelihood estimates

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

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

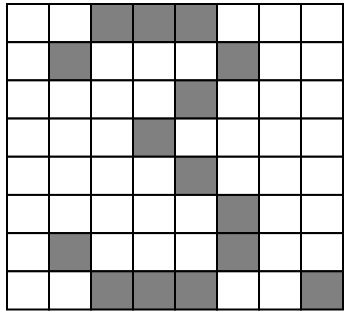
- Another option is to consider the most likely parameter value given the data

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

⇒ "right" choice of P(theta)
-> Laplace estimates

Estimation: Linear Interpolation*

- In practice, Laplace can perform 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

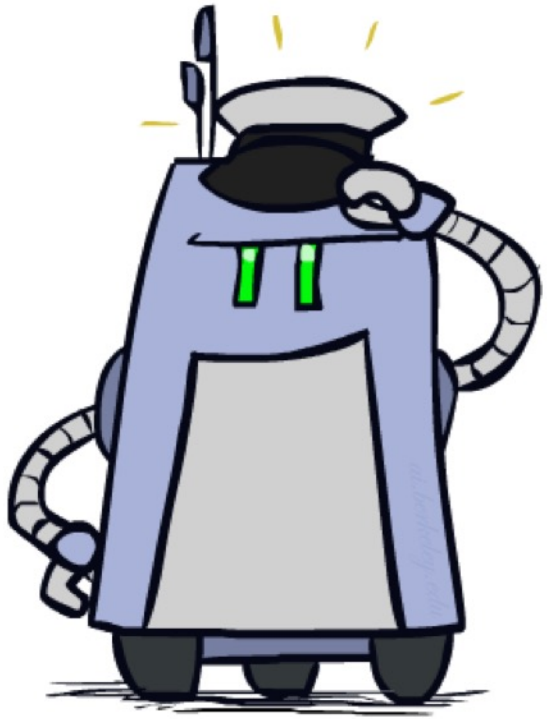
- For real classification problems, smoothing is critical
- New odds ratios:

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

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

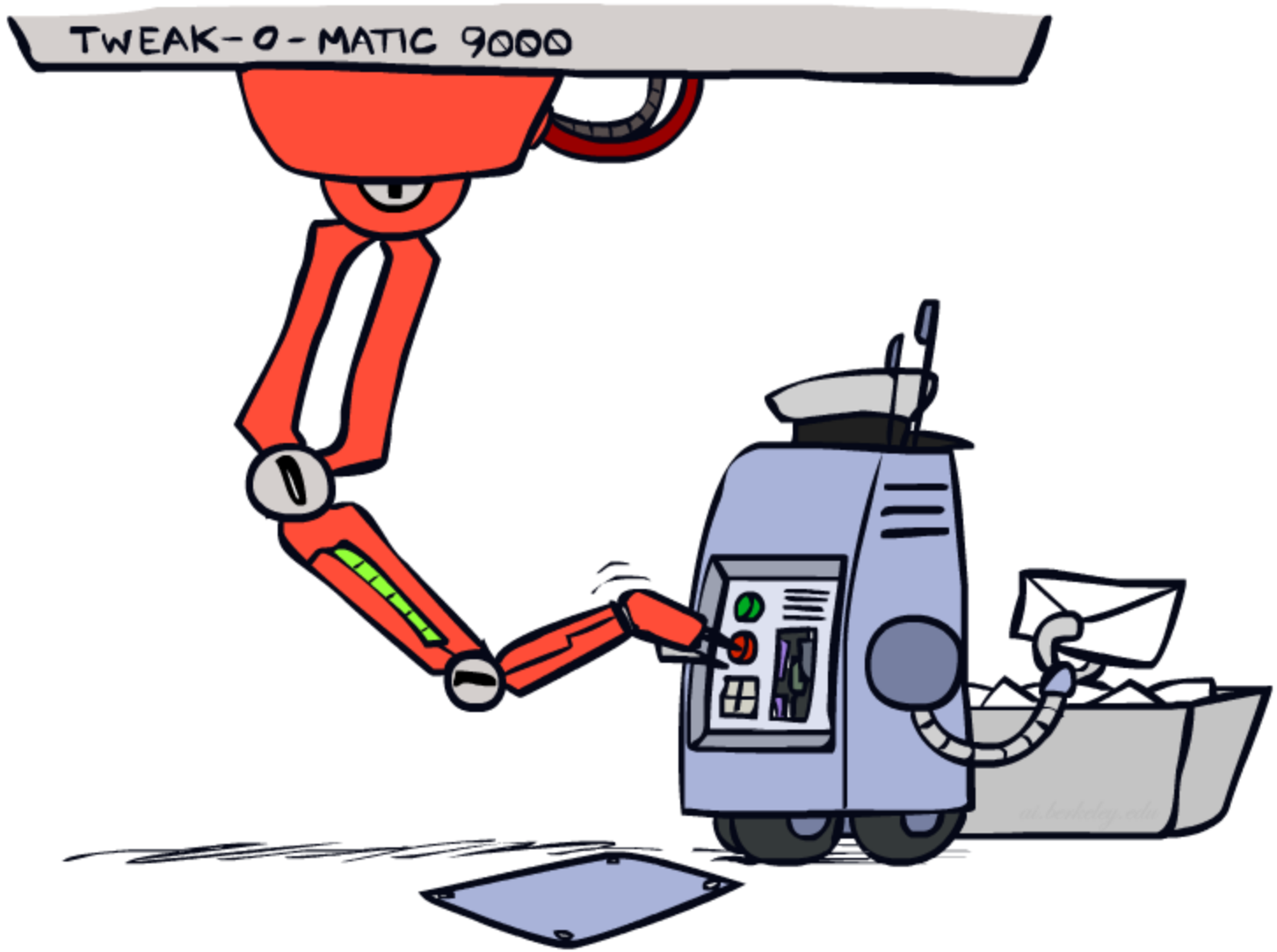
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

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



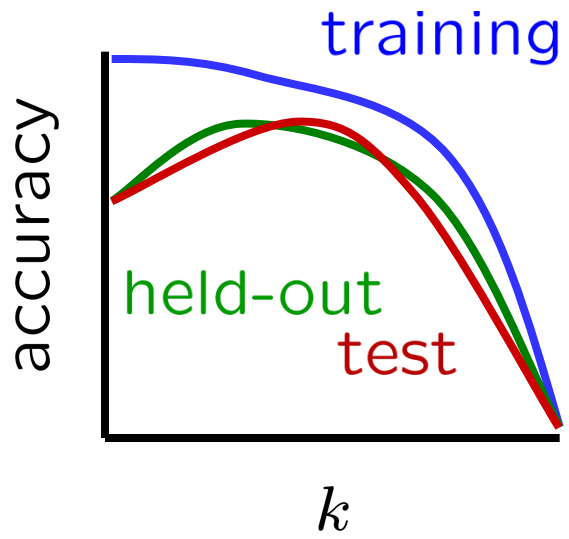
Do these make more sense?

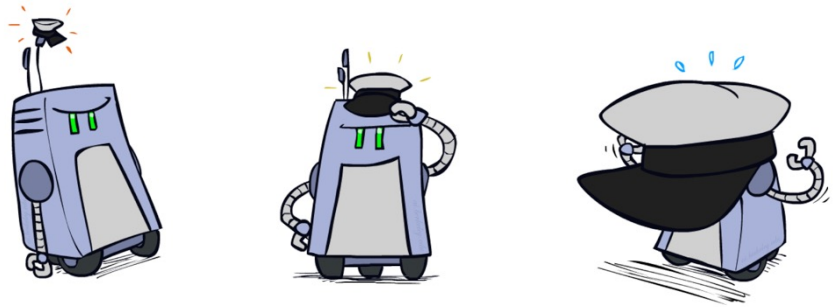
Tuning



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





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