Machine Learning 1 Naïve Bayes

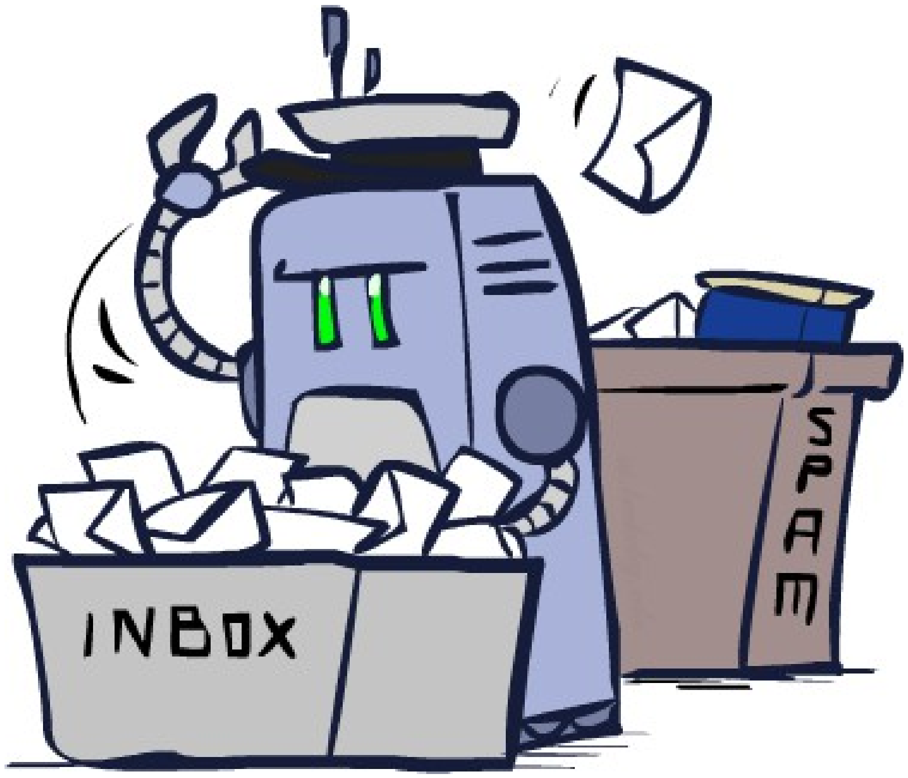
Types of Learning Problems

- **Supervised learning:** correct answers for each training example
  - **Classification:** learning predictor with *discrete* outputs
  - **Regression:** learning predictor with *real-valued* outputs
- **Unsupervised learning:** no correct answers, just find good representations / features of the data
- **Reinforcement learning:** reward function, no correct answers

Machine Learning Roadmap

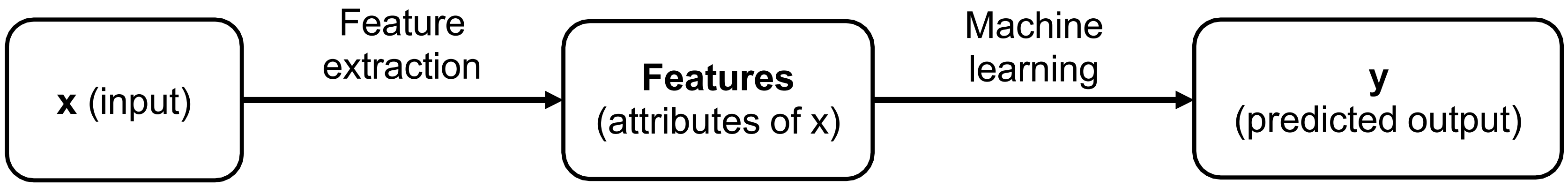
- Define the problem
  - Type of problem, domain (i.e. spam filtering, digit recognition)
- Look at several learning approaches / models
  - *Naïve Bayes* (today), *Perceptrons*, *Logistic Regression* (next week)
- How to find model parameters:
  - Maximum Likelihood
- Themes throughout
  - Working with data
  - Preventing overfitting
  - Evaluating performance

Classification



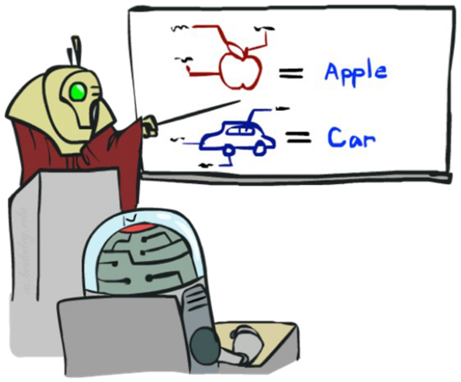
Classification and Machine Learning

- 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

- 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. …

✗

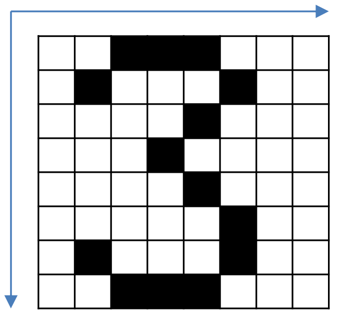
TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

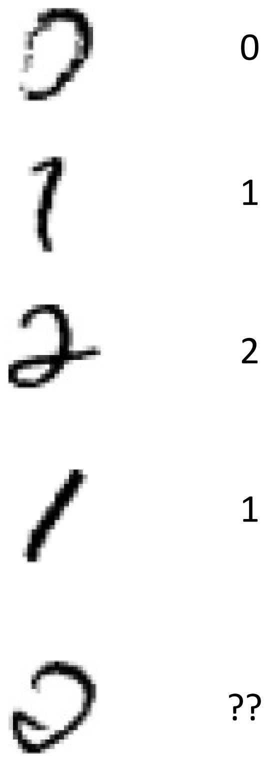
99 MILLION EMAIL ADDRESSES
 FOR ONLY $99

✓

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

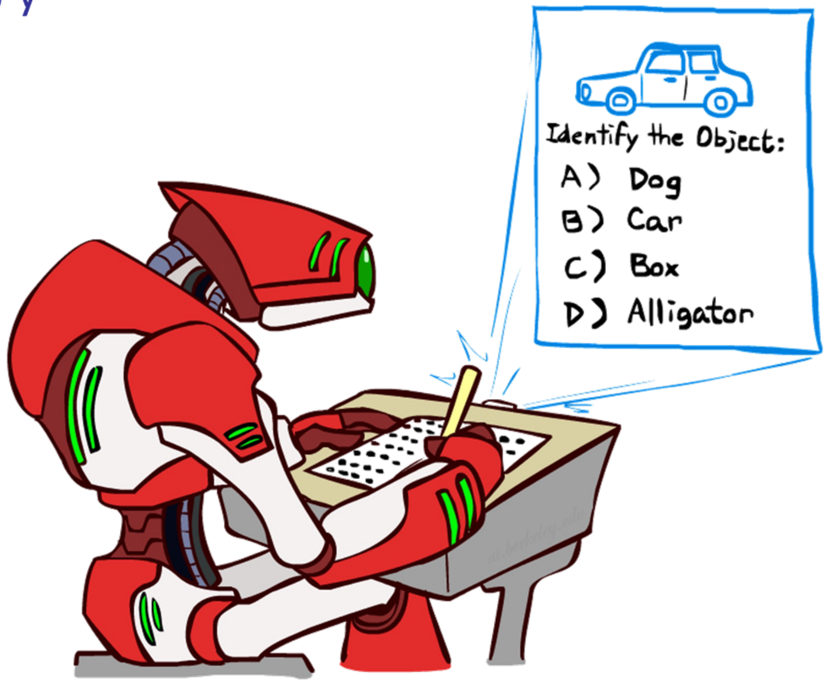
- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Note: someone has to hand label all this data!
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (1,1)=ON
  - Shape Patterns: NumComponents (Number of connected regions), AspectRatio (Height-to-width ratio), NumLoops (Number of closed loops (e.g., '8' has two loops)).
  - These features help the model distinguish between similar-looking digits like '1' and '7' or '3' and '8'.

Other Classification Tasks

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

- Examples:
  - Object recognition
    input: images; classes: object type
  - 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
    input: a customer service email; classes: billing, technical support, account management, sales, shipping, returns, product inquiry, etc.
  - … many more
- Classification is an important commercial technology!



Model-Based Classification

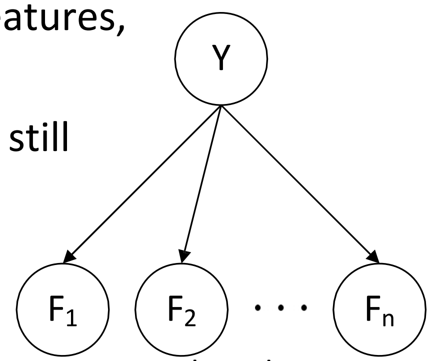
Model-Based Classification

- Model-based approach
  - Build a model (e.g. Naïve Bayes) 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 model have?
  - How should we learn its parameters?

Naïve Bayes Model

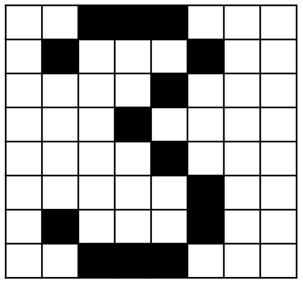
- Naïve Bayes: Assume all features are independent effects of the label.
  - This simplification is the **"naïve" assumption**: it ignores correlations between features, making computation easier
  - While this assumption is often not true in reality, it simplifies computations and still performs surprisingly well in practice.
- Random variables:
  - Y = The label
  - $F_1, F_2, \ldots, F_n$ = The n features
- Parameters: Probability tables:
  - $P(Y)$ = Probability of each label, given no information about the features.
    - Sometimes called the *prior*.
    - Example: What's the probability that any given email is spam before looking at its content?
  - $P(F_i|Y)$ = One table per feature. Probability of a feature, given the label.
    - Example: What's the chance the word "FREE" appears given that the email is spam?

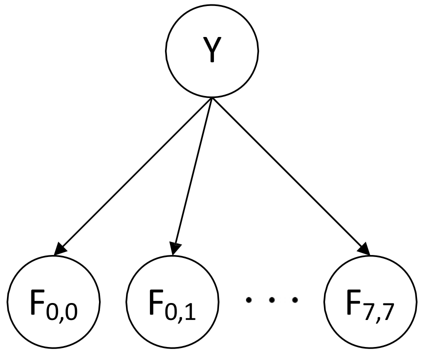


A Bayes Net. Each node represents a random variable. An arrow from A → B means that A influences B (i.e., B's probability depends on A).

Domain 1: Naïve Bayes for Digits

- Simple digit recognition version:
  - One feature (variable) $F_{i,j}$ 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.
    $F_{0,0}=0$ $F_{0,1}=0$ $F_{0,2}=1$ $F_{0,3}=1$ $F_{0,4}=1$ … $F_{7,7}=0$
  - Here: lots of features, each is binary valued
    - **n**: # of features. n = 8×8 = 64 pixels.
    - **|F|**: # of values a single feature can take. |F| = 2 (0 or 1).
    - **|Y|**: # of classes. |Y|=10.





- What are the parameters to learn of this model?
  - P(Y = y) — the prior probability of each digit class (how often each digit appears). Total # of prior probabilities to learn =|Y|=10.
  - $P(F_{i,j} \mid Y = y)$ — the likelihood: for each class and each pixel, the probability that the pixel is on (1) or off (0). Total # of conditional probabilities to learn =|Y|× n ×|F| = 10 × 64 × 2 = 1280.

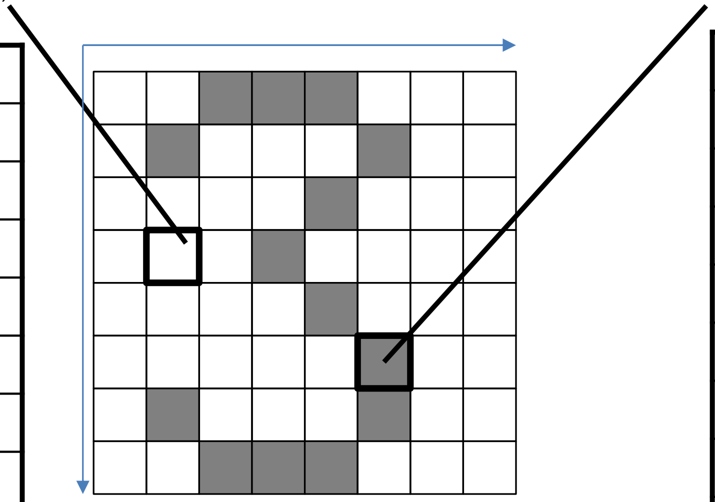
Domain 1: Naïve Bayes for Digits: Parameters

$P(Y)$

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

$P(F_{3,1} = \text{on} \mid Y)$

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

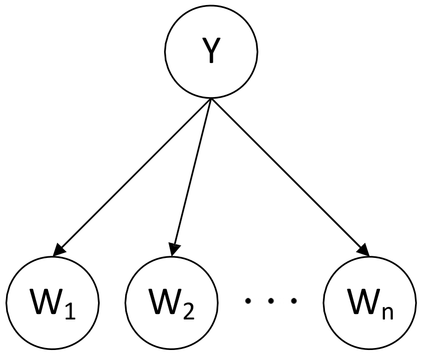


$P(F_{5,5} = \text{on} \mid Y)$

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

Domain 2: 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, so $P(W_1|Y) = P(W_2|Y) = \ldots$
    - We don't care **where** a word appears, just **whether** it appears.



- "Tied" distributions and bag-of-words
  - Usually, each feature (word position) would get its own conditional probability distribution $P(F|Y)$
  - In a bag-of-words model
    - Each position is identically distributed.
    - All positions share the same (i.e. Tied) conditional probs $P(W|Y)$.
    - Pro: simpler and more efficient. Con: the model ignores word order.
  - Called "bag-of-words" because model is insensitive to word order or reordering

free our offer try please

please try our free offer

Domain 2: Naïve Bayes for Text: Parameters

- What are the parameters?

$P(Y)$

ham :	0.67
spam:	0.33

If your training data has 300 emails, with 200 ham and 100 spam:

P(ham)=200/300=0.67

P(spam)=100/300=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
offer:	0.0035
...	

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

the :	0.0210
to :	0.0133
of :	0.0119
2002:	0.0110
with:	0.0108
from:	0.0107
and :	0.0105
offer:	0.00067
...	

- **n** = # of words.
- **|F|** = 2 (is this word in the email or not).
- **|Y|** = 2 (ham or spam).

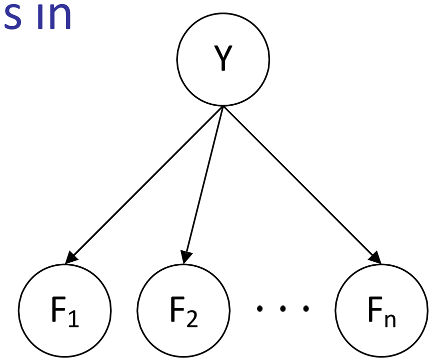
$$P(W_i|y) = \frac{count\ of\ word\ W_i\ in\ class\ y + 1}{total\ words\ in\ class\ y + V}$$

W_i: a word in the vocabulary
y: class label (spam or ham)
V: total number of unique words in the vocabulary (used for smoothing)
The **+1** and **+V** come from **Laplace smoothing**, which prevents zero probabilities for unseen words.

Naïve Bayes Model

- In general, the joint probability (posterior) of the label and all features in Naïve Bayes model is:

$$P(\mathsf{Y}, \mathsf{F}_1 \ldots \mathsf{F}_n) = \underbrace{P(\mathsf{Y})}_{|\mathsf{Y}|\text{ parameters}} \underbrace{\prod_i P(\mathsf{F}_i|\mathsf{Y})}_{n \times |\mathsf{F}| \times |\mathsf{Y}|\text{ parameters}}$$



- We only have to specify how each feature depends on the class
- Total number of parameters is *linear* in n
  - Without the Naïve assumption (then the features can be dependent on each other), we need $|Y| \times |F|^n$ values
- 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

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

$+$

$$\Downarrow$$

  - Step 2: sum to get probability of evidence

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

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

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
you :	0.0093

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

the :	0.0210
to :	0.0133

"you" is not listed: assume a small value (e.g., **Laplace smoothing** gives ~0.0001)

Suppose we have a short email with three words: **"the", "to", "you"**

Step 1 Compute the joint probabilities:

P(spam,"the","to","you") = P(spam) × P("the" | spam) × P("to" | spam) × P("you" | spam)
= 0.33 × 0.0156 × 0.0153 × 0.0093 ≈ 7.34×10^{-7}

P(ham,"the","to","you") = P(ham) × P("the" | ham) × P("to" | ham) × P("you" | ham)
= 0.66 × 0.0210 × 0.0133 × 0.0001 ≈ 1.84×10^{-7}

Step 2 Compute total probability of evidence (denominator)

P("the", "to", "you") = $7.34\times10^{-7} + 1.84\times10^{-7} = 9.18\times10^{-7}$

Step 3 Normalize to get **posterior**

P(spam | "the", "to", "you") = $\frac{7.34\times10^{-7}}{9.18\times10^{-7}} \approx 0.799$

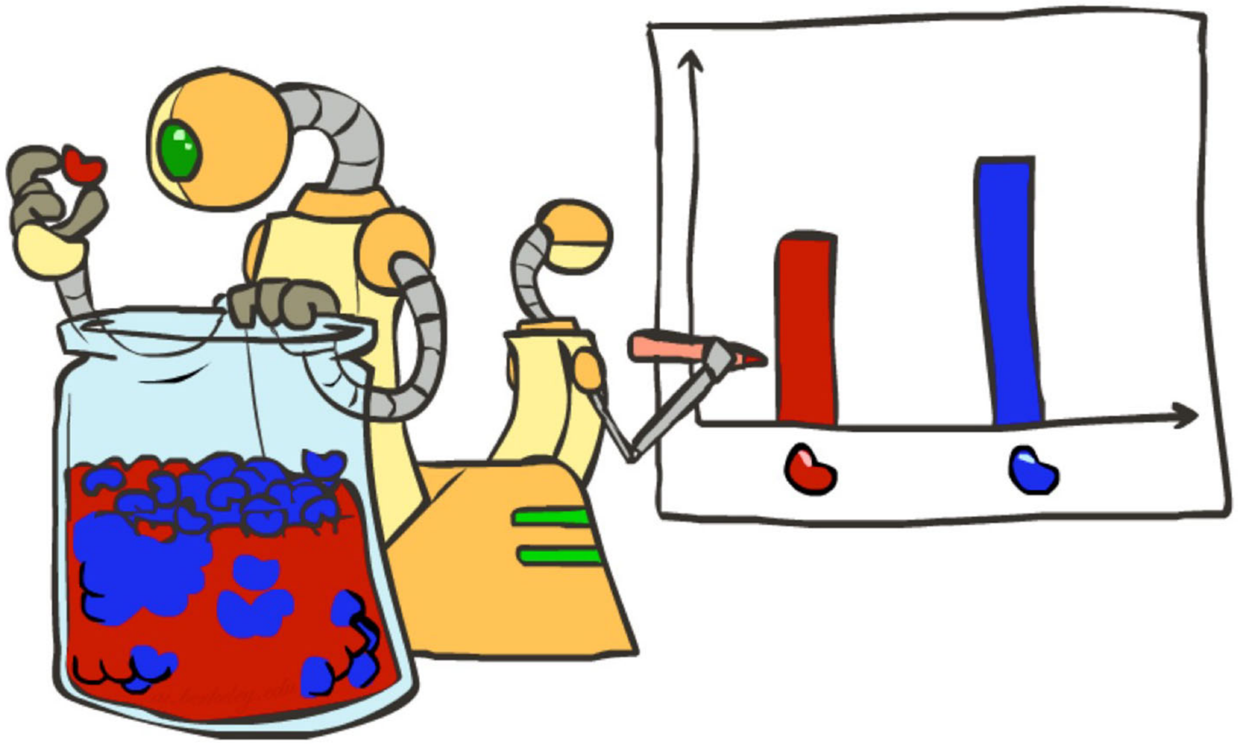
P(ham | "the", "to", "you") = $\frac{1.84\times10^{-7}}{9.18\times10^{-7}} \approx 0.201$

Final Classification → **spam**, since it has a higher posterior probability (≈ 80%).

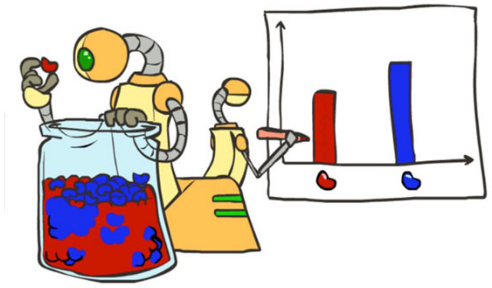
General Naïve Bayes

- 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
    - 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

- θ is the true fraction of red beans in the jar (i.e. P(red | θ) = θ)
- 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)

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

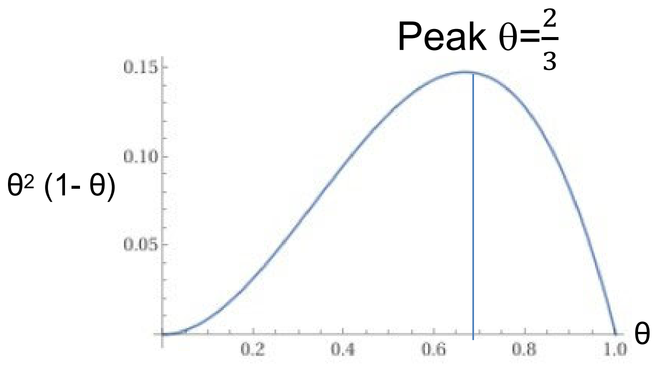
  - We want to compute:

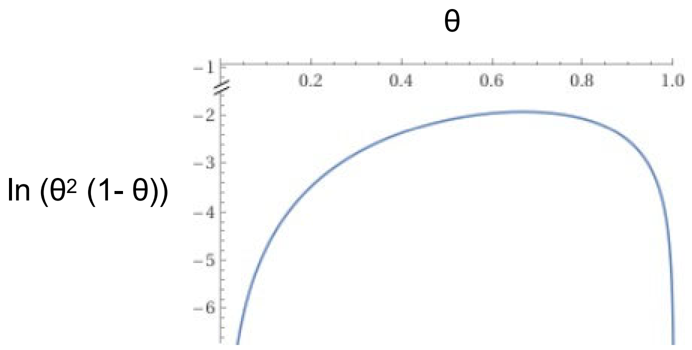
    $$\underset{\theta}{\operatorname{argmax}}\ \theta^2 (1 - \theta)$$

Parameter Estimation with Maximum Likelihood

- We want to compute:

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





- 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 (this will turn products to sums), which are easier to differentiate.
  - Useful fact:

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

Parameter Estimation with Maximum Likelihood

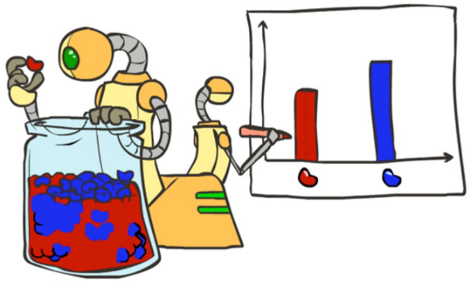
$\underset{\theta}{\operatorname{argmax}}\, \theta^2(1-\theta)$	Find θ that maximizes likelihood
$= \underset{\theta}{\operatorname{argmax}}\, \ln\left(\theta^2(1-\theta)\right)$	Find θ that maximizes log-likelihood (will be the same θ)
$\frac{d}{d\theta} \ln\left(\theta^2(1-\theta)\right) = 0$	Set derivative 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.

Parameter Estimation with Maximum Likelihood (General Case)

- **Model:**

X	red	blue
$P(X\|\theta)$	θ	$1-\theta$

- **Data:** draw N balls, N_r come up red and N_b come up blue
  - Dataset $D = x_1, \ldots, x_N$ of N ball draws



$$P(D|\theta) = \prod_i P(x_i|\theta) = \theta^{N_r} \cdot (1-\theta)^{N_b}$$

- **Maximum Likelihood Estimation**: find θ that maximizes $P(D|\theta)$:

$$\hat{\theta} = \underset{\theta}{argmax}\, P(D|\theta) = \underset{\theta}{argmax}\, \boxed{\log P(D|\theta)} \leftarrow N_r \log(\theta) + N_b \log(1-\theta)$$

Take derivative and set to 0:

$$\frac{d}{d\theta}\log P(D|\theta) = \frac{d}{d\theta}[N_r \log\theta + N_b \log(1-\theta)] = N_r \frac{d}{d\theta}\log\theta + N_b \frac{d}{d\theta}\log(1-\theta)$$

$$= \frac{N_r}{\theta} + \frac{N_b}{1-\theta}\cdot(-1) = N_r(1-\theta) - N_b\theta = N_r - \theta(N_r + N_b) = 0 \rightarrow \hat{\theta} = \frac{N_r}{N_r + N_b} = \frac{\#\ of\ red\ balls}{total\ \#\ of\ balls}$$

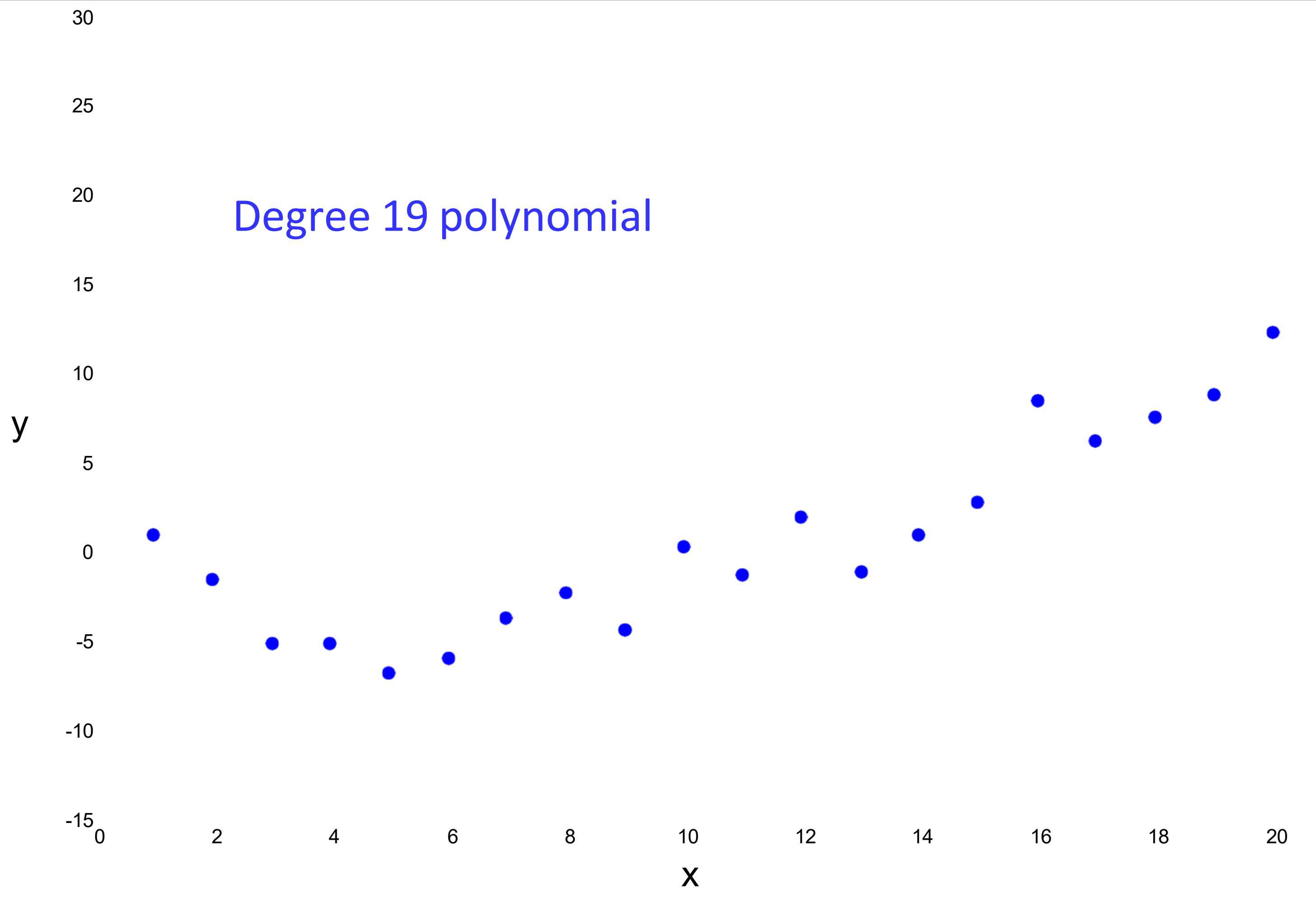
Parameter Estimation with Maximum Likelihood

- Collectively name all model parameters (i.e. probability tables) as θ
- **Maximum Likelihood Estimation**: find θ that maximizes $\mathrm{P}(\mathrm{Data}|\theta)$
  - In practice, maximize log P instead because computation is easier
  - To solve, take derivative and set to 0
- For Naïve Bayes maximum likelihood estimates of prob. tables are:

$$P(y) = \frac{\text{\# of occurences of class } y}{\text{total \# of observations}} \qquad P(f \mid y) = \frac{\text{\# of occurences of feature } f \text{ and class } y}{\text{total \# of occurences of class } y}$$

- Need to be careful though … let’s see what can go wrong..

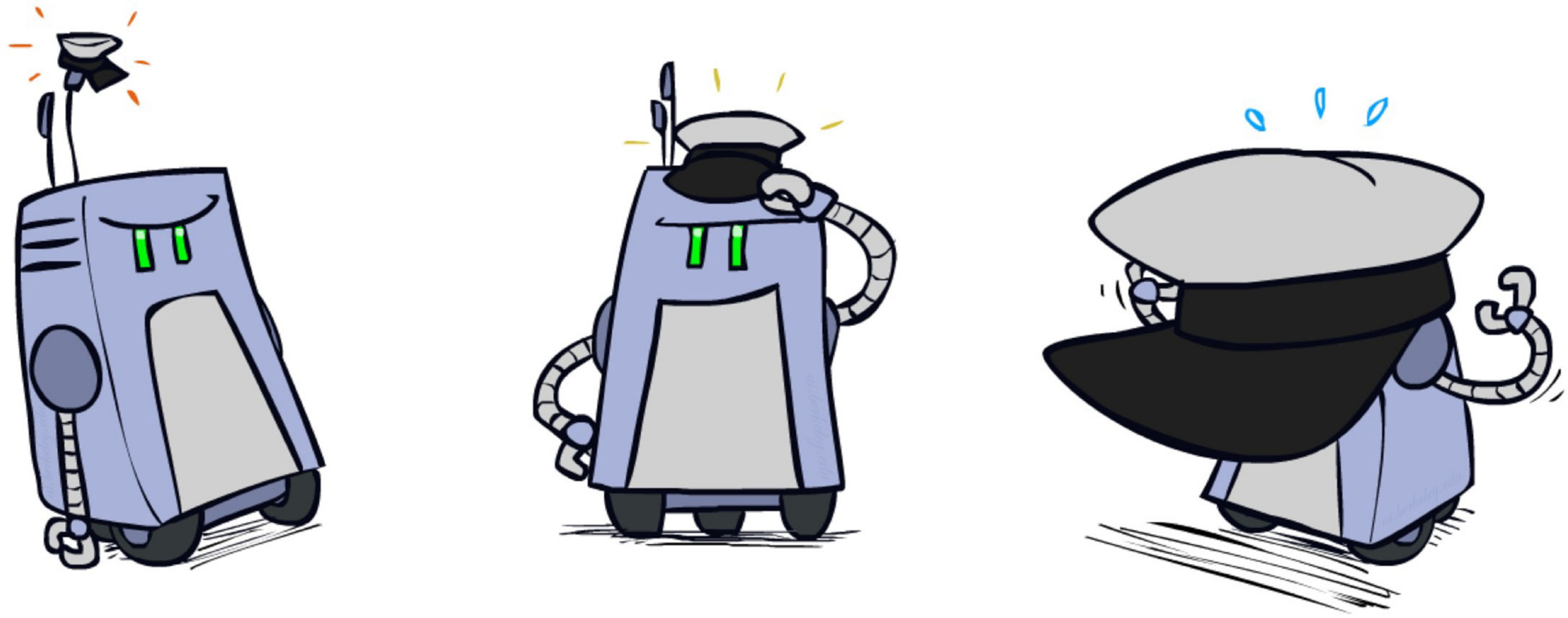
What is the best way to fit this data?



Empirical Risk (training error) Minimization

- How should we evaluate the quality of our model?
- 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)
  - Better if we limit the complexity of our hypotheses (regularization and/or small hypothesis spaces)
- Another worry: our training distribution doesn’t match true distribution

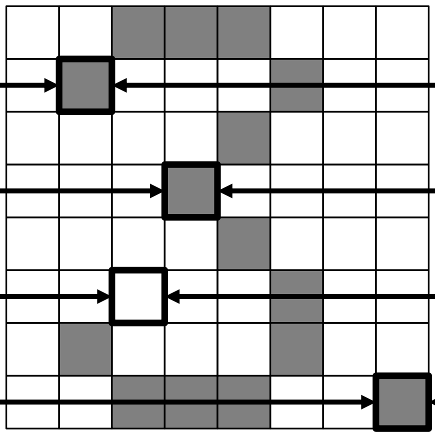
Underfitting and Overfitting



Example: Overfitting

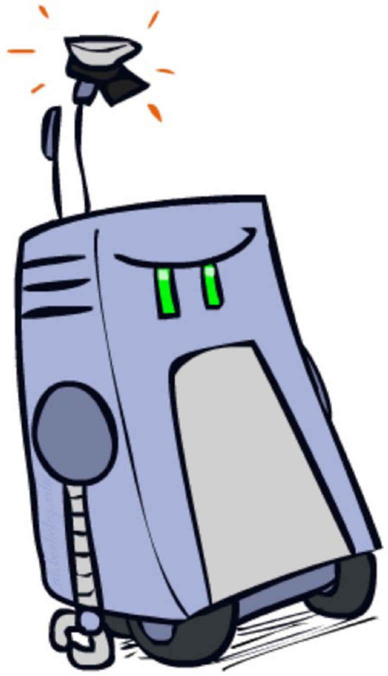
$P(\text{features}, C = 2)$
$P(C = 2) = 0.1$

$P(\text{features}, C = 3)$
$P(C = 3) = 0.1$



$P(\text{on}\|C = 2) = 0.8$	$P(\text{on}\|C = 3) = 0.8$
$P(\text{on}\|C = 2) = 0.1$	$P(\text{on}\|C = 3) = 0.9$
$P(\text{off}\|C = 2) = 0.1$	$P(\text{off}\|C = 3) = 0.7$
$P(\text{on}\|C = 2) = 0.01$	$P(\text{on}\|C = 3) = 0.0$

2 wins!!



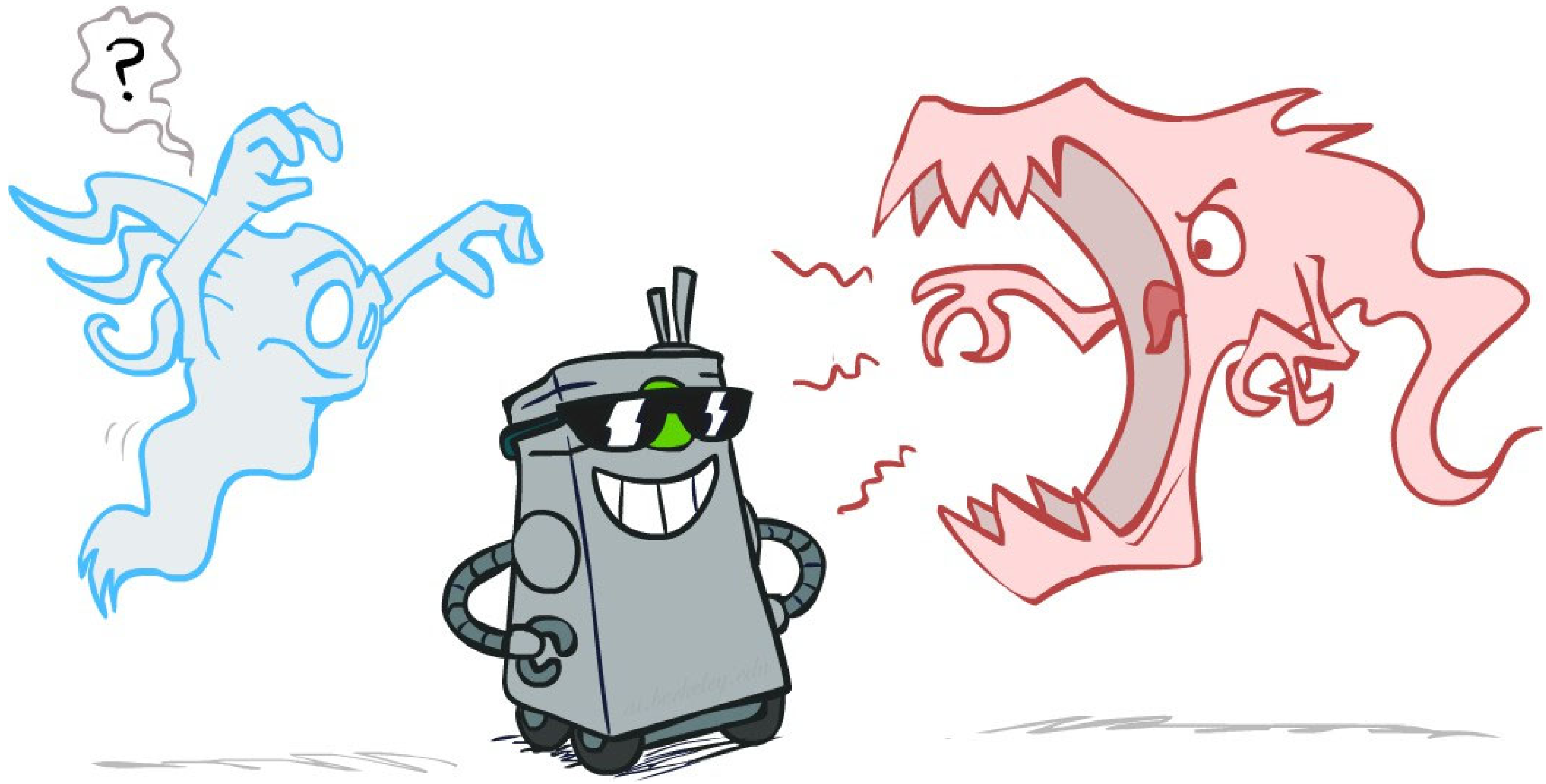
Generalization and Overfitting

- 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/regularize the estimates

Smoothing



Laplace Smoothing

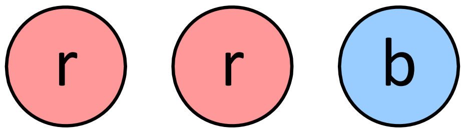
- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

Add 1 to each count as if we had seen each outcome once more than we did

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

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

- c(x): count of x
- N: total observations
- |X|: number of possible outcomes (e.g., red, blue → |X| = 2)



P(red) = 2/3≈0.67
P(blue) = 1/3 ≈0.33
P_{LAP}(red) = (2+1)/(3+2)=3/5=0.6
P_{LAP}(blue) = (1+1)/(3+2)=2/5=0.4

Generalized Laplace Smoothing

- Laplace with strength k:
  - Pretend you saw every outcome k extra times

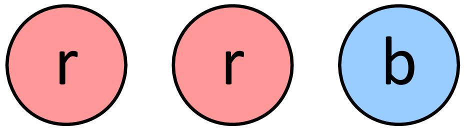
$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - k is the strength of the prior: how strongly your prior belief (before seeing the data) influences the resulting probability estimate.
    - k = 0: You are relying **entirely** on the observed data, no smoothing → back to MLE
    - k = 1: basic Laplace
    - k > 1: stronger smoothing — useful when you have less data or expect more unseen outcomes

Generalized Laplace Smoothing

- Laplace for conditionals:
  - Smooth each condition independently:

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



Suppose we are trying to estimate the probability of drawing a red or blue bean given some condition y. For example, y is the day we drew the beans.

We’ve observed:

- c(red, y) = 2
- c(blue, y) = 1
- c(y) = 3 (total draws with condition y)

What’s P(green | y)? 0! zero probability for unseen outcomes = bad for generalization!

Laplace Smoothing with k = 1 per condition:

• c(red, y) = 2	P(red | y) = (2+1)/(3+3)=3/6=0.5
• c(blue, y) = 1	P(blue | y) = (1+1)/(3+3)=2/6≈0.33
• c(green, y) = 0	P(green | y) = (0+1)/(3+3)=1/6=0.167
• c(y) = 3	
• |X|=3	

Naïve Bayes: No Smoothing (Overfitting)

- *Relative* probabilities (odds ratios):

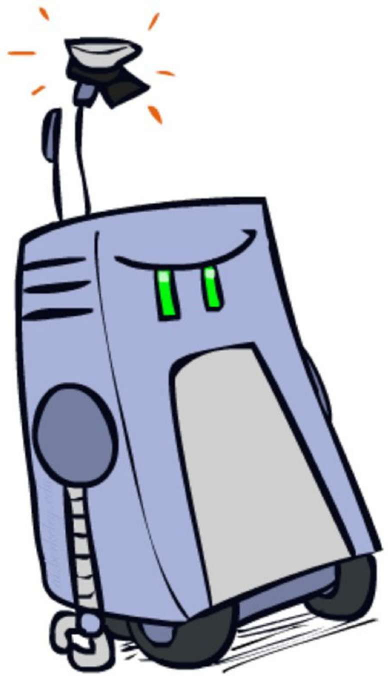
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
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?



Naïve Bayes: With Smoothing

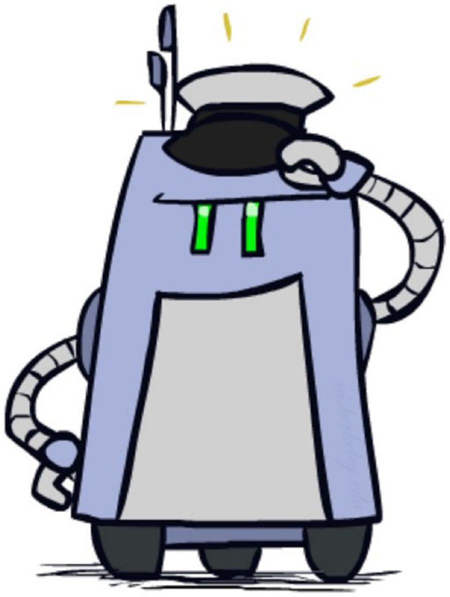
- For real classification problems, smoothing is critical
- New odds ratios:

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

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

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

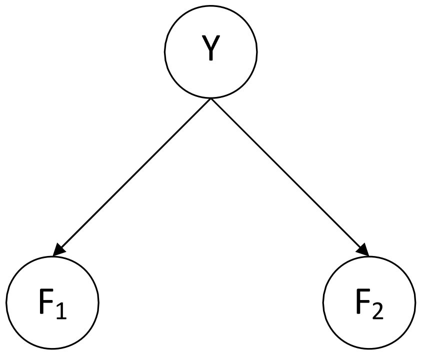
verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
<FONT>	:	26.9
money	:	26.5
...		



Do these make more sense?

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.

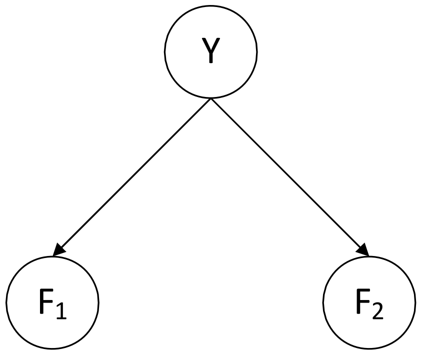
F_2: # of occurrences of FREE		
F_2	**Y**	**$P(F_2 \| Y)$**
0	**ham**	0.5
1	**ham**	0.5
2	**ham**	0.0
0	**spam**	0.25
1	**spam**	0.50
2	**spam**	0.25

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:



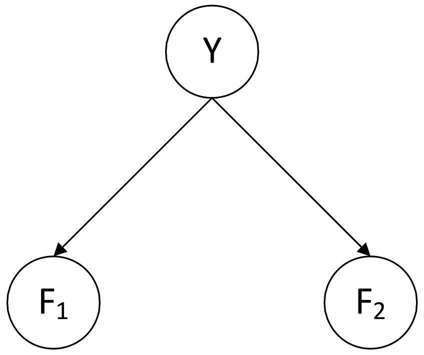
Y: The label (spam or ham)	
Y	P(Y)
ham	0.6
spam	0.4

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

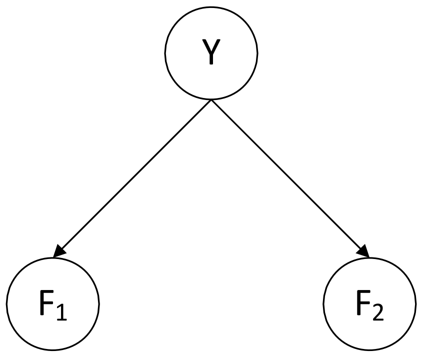
Example: Naïve Bayes for Spam Filter

- 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:
  - $P(Y = \text{spam}, F_1 = \text{yes}, F_2 = 1) = P(Y = \text{spam})\ P(F_1 = \text{yes} \mid \text{spam})\ P(F_2 = 1 \mid \text{spam})$
    $= 0.4 * 0.1 * 0.12 = 0.0048$
  - $P(Y = \text{ham}, F_1 = \text{yes}, F_2 = 1) = P(Y = \text{ham})\ P(F_1 = \text{yes} \mid \text{ham})\ P(F_2 = 1 \mid \text{ham})$
    $= 0.6 * 0.7 * 0.07 = 0.0294$
- Normalize:
  - $P(Y = \text{spam} \mid F_1 = \text{yes}, F_2 = 1) = 0.0048 / (0.0048+0.0294) = 0.14$
  - $P(Y = \text{ham} \mid F_1 = \text{yes}, F_2 = 1) = 0.0294 / (0.0048+0.0294) = 0.86$
- 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

What we did today

- Saw our first machine learning algorithm: Naïve Bayes
  - Model is a Bayes Net where features are independent given class label
  - Classification is just inference in Bayes Nets
  - Learning is just counting feature occurrences in training data
- Saw *Maximum Likelihood* as a principled way to estimate parameters
  - Maximize probability of the data given model parameters
  - For Naïve Bayes, we solved maximization problem analytically
- Saw that fitting training data too well can cause overfitting and we can smooth it