Machine Learning

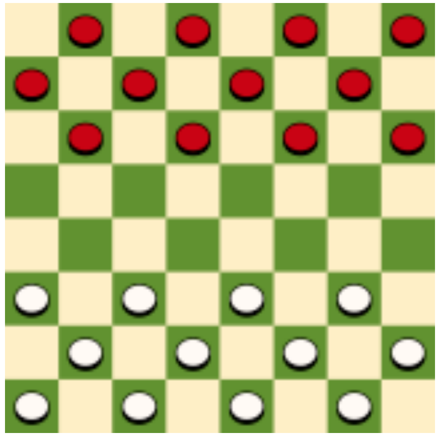
- Introduction
- Naive bayes classification
- Perceptron, linear regression
- Neural networks
- Reinforcement learning

Definition of machine learning

- Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the ability to learn without being explicitly programmed.

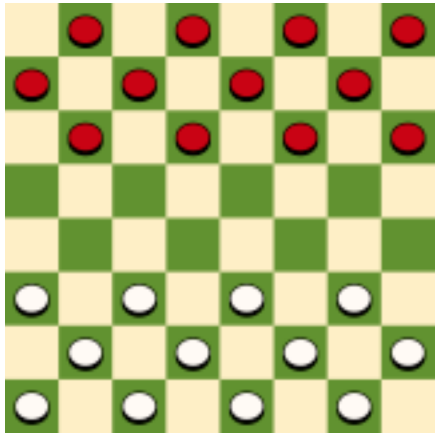
A. L. Samuel*

Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress



Definition of machine learning

- Tom Mitchell (1998): a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
- Experience (data): games played by the program (with itself)
- Performance measure: winning rate



Machine Learning

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- Machine learning : a process for improving the performance of an agent through experience
- There is no need to “learn” to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

Machine Learning

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:
- People who bought “Blink” also bought “Outliers” (www.amazon.com)
- Build a model that is a good and useful approximation to the data.

Machine Learning

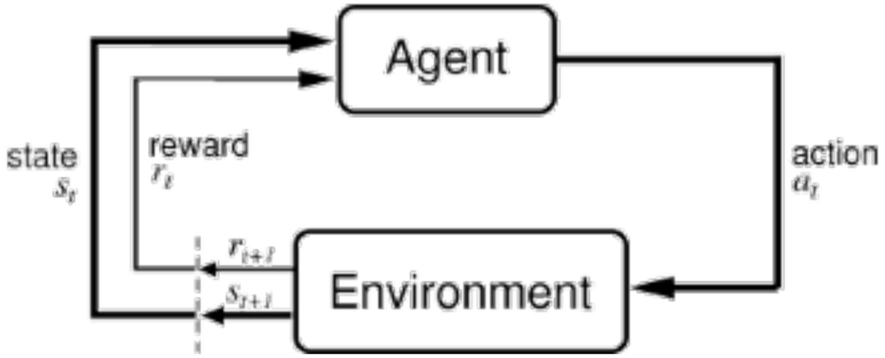
- Supervised learning : learning from labeled data we learn the correct answer for a certain examples. The algorithm learn to make prédictions.
  - Classification: learning predictor with discrete outputs
  - Regression: learning predictor with real-valued outputs
- Unsupervised learning : Finding patterns in unlabeled data. The algorithm makes sense of the data without ground truth
- Reinforcement learning : Learning through trial and error maximizing reward over time

Machine Learning

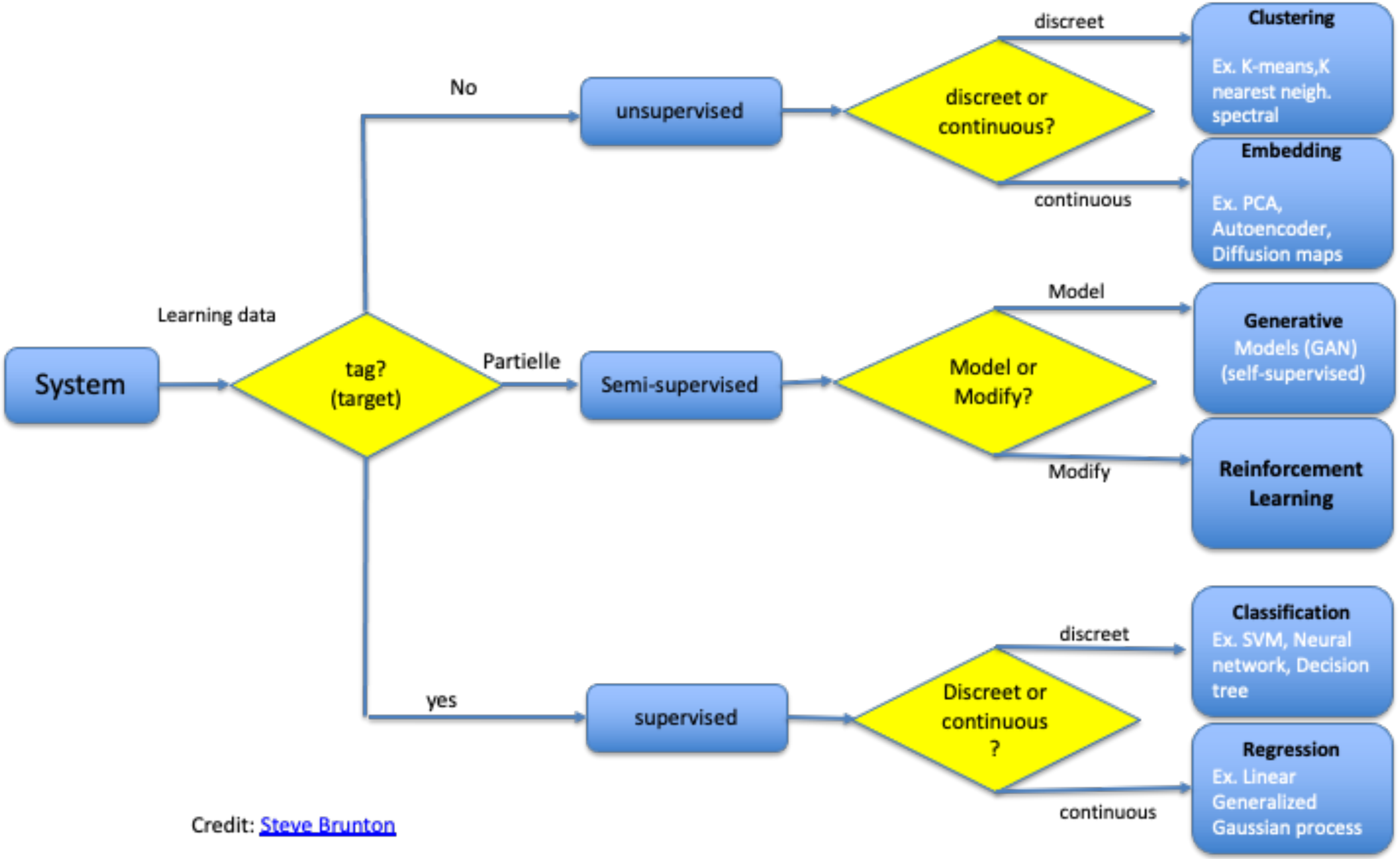
Supervised learning

Unsupervised learning

Reinforcement learning



Machine Learning



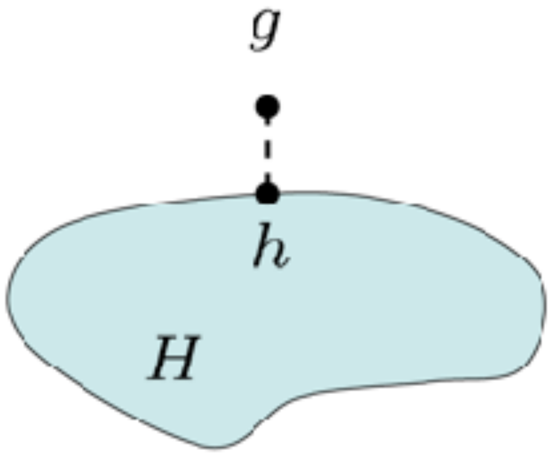
Credit: Steve Brunton

Supervised learning

- To learn an unknown **target function** f
- Input: a training set of **labeled examples** (x_j, y_j) where $y_j = f(x_j)$
  - E.g., x_j is an image, $f(x_j)$ is the label "giraffe"
  - E.g., x_j is a seismic signal, $f(x_j)$ is the label "explosion"
- Output: hypothesis *h* that is "close" to *f*, i.e., predicts well on unseen examples ("test set")
- Many possible hypothesis families for h
  - Linear models, logistic regression, neural networks, decision trees, examples (nearest-neighbor), etc
- Classification = learning *f* with discrete output value
- Regression = learning *f* with real-valued output value

Learning

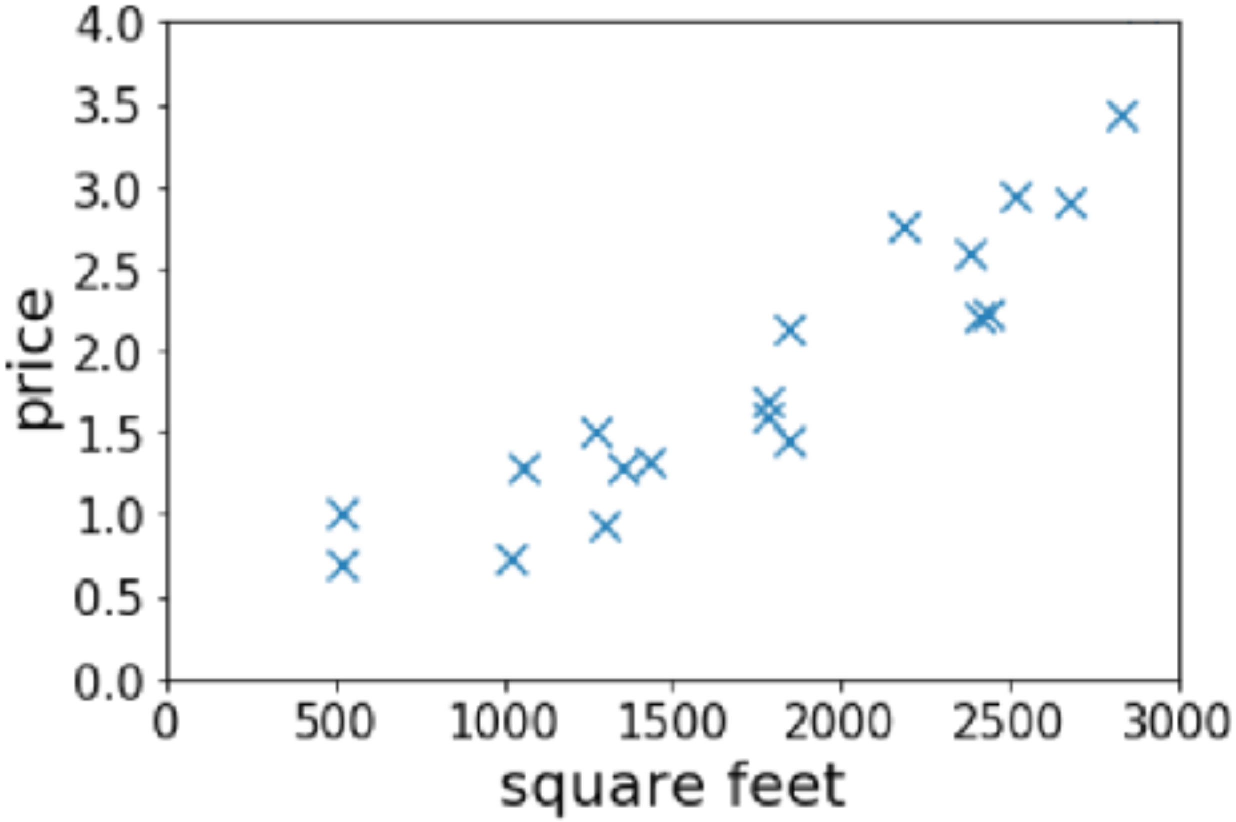
- Simplest form: learn a function from examples
  - A target function: *g*
  - Examples: input-output pairs *(x, g(x))*
    - E.g. *x* is an email and *g(x)* is spam / ham
    - E.g. *x* is a house and *g(x)* is its selling price
- Problem:
  - Given a hypothesis space H
  - Given a training set of examples x_i
  - Find a hypothesis *h(x)* such that *h ~ g*
- Includes:
  - Classification (outputs = class labels)
  - Regression (outputs = real numbers)



House pricing Prediction

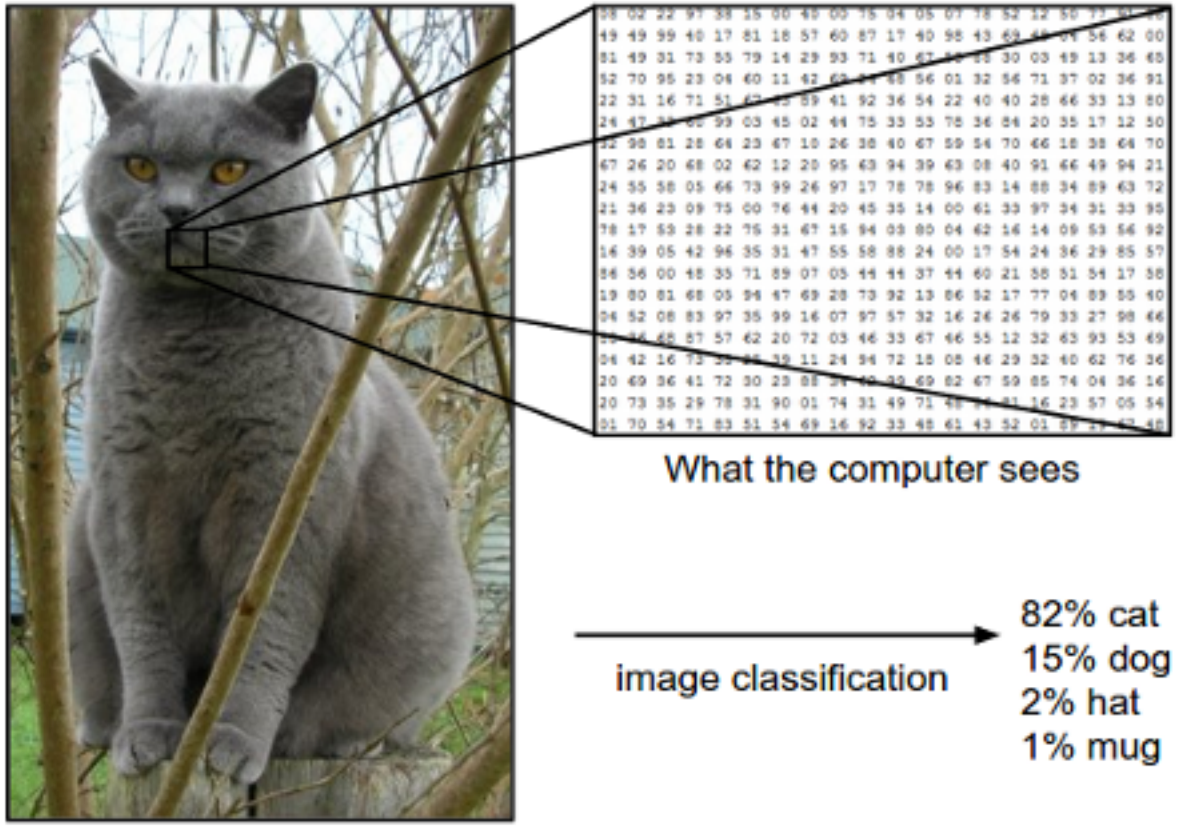
Given: a dataset that contains samples (x^i, y^i)

Task: if a residence has x square feet, predict its price?

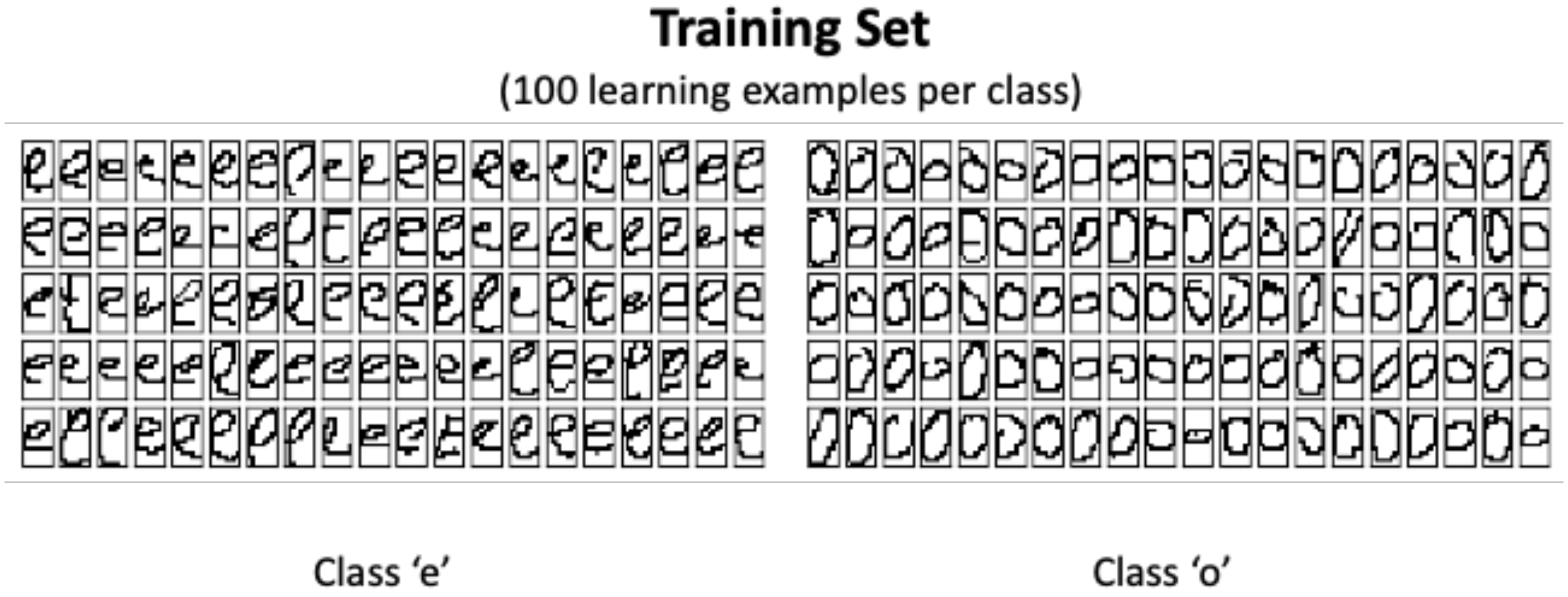


Classification and Machine Learning

- Image Classification
- x= raw pixels of the image,
- y = the main object



Classification and Machine Learning

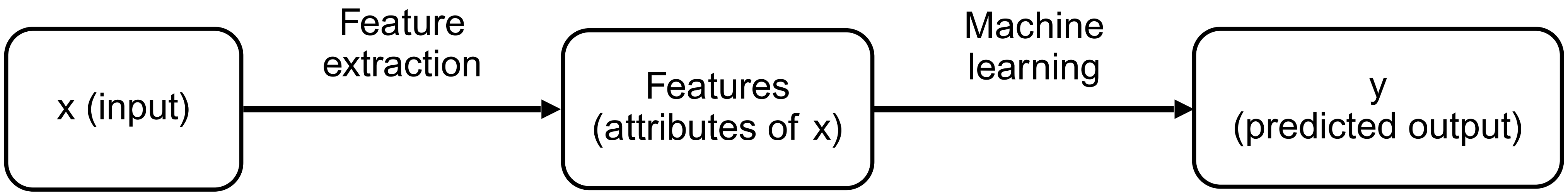


Classification and Machine Learning

- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
- Medical diagnosis: From symptoms to illnesses
- Biometrics: Recognition/authentication using physical and/or behavioral characteristics: Face, iris, signature, etc
- Outlier/novelty detection: Detecting faults

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: Naïve Bayes
  - Output a predicted label y



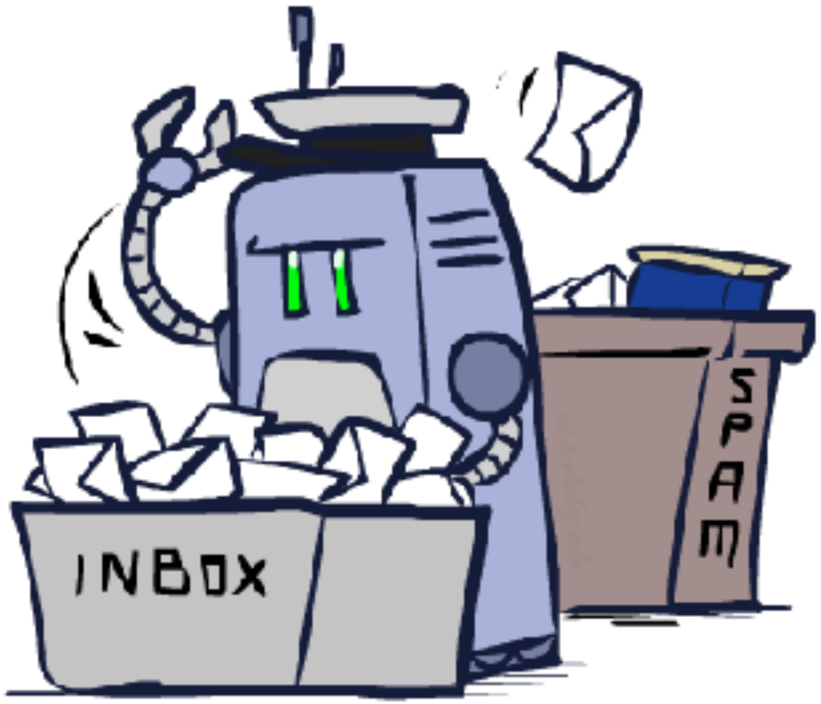
Training and Machine Learning

- 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

Naïve Bayes classification



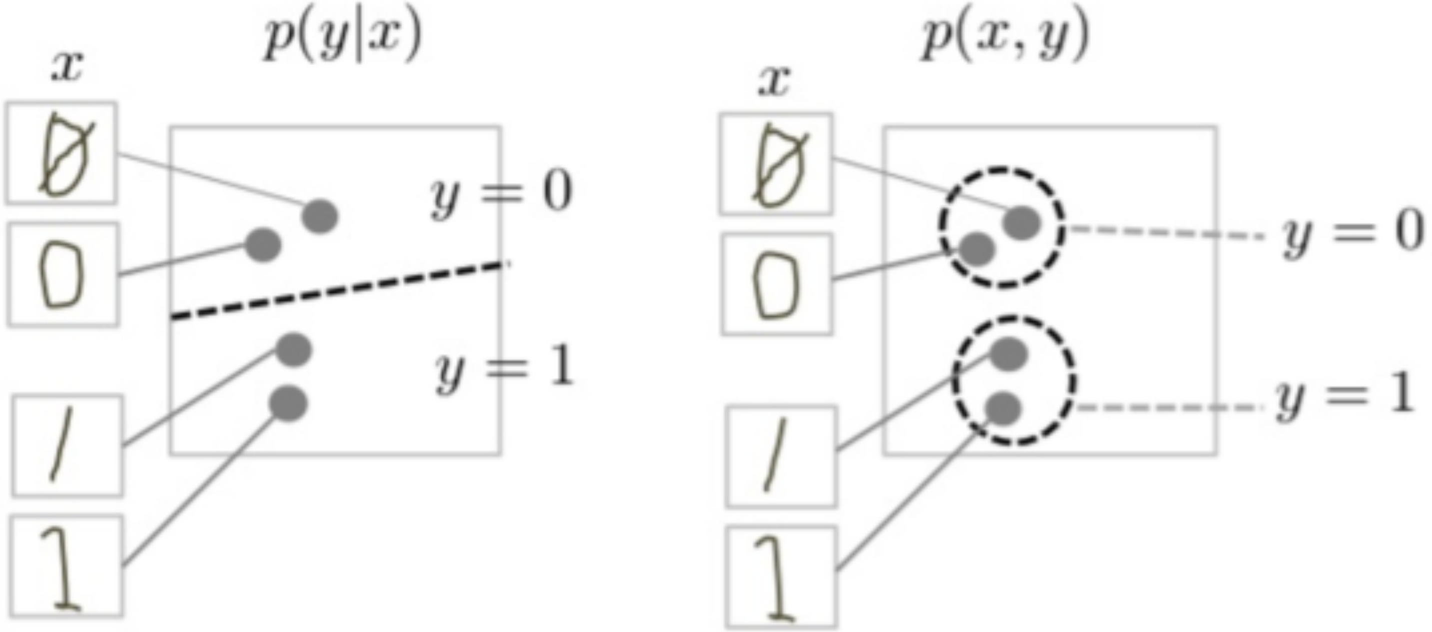
Model-based classification

- Up until now: how to use a model to make optimal decisions

- Machine learning: how to acquire a model from data / experience
  - Learning parameters (e.g. probabilities)
  - Learning structure (e.g. BN graphs)
  - Learning hidden concepts (e.g. clustering, neural nets)

- model-based classification with Naive Bayes

Generative/Dicriminative classifier



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
  - classifiers reject 200 bilion spam emails per day

- 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

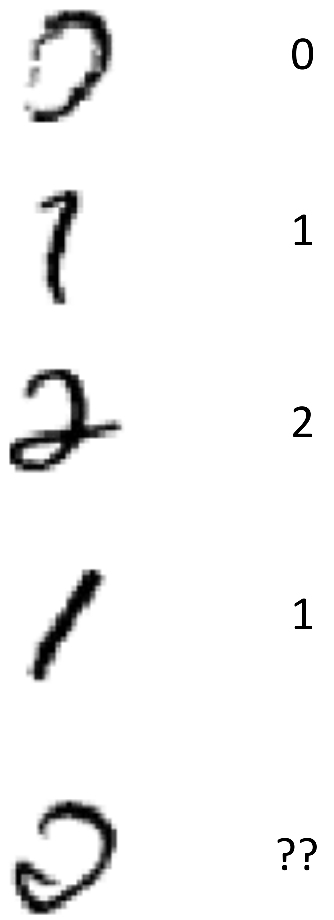
Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

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

- Setup:
  - MNIST data set of 60K collection hand-labeled images
    - 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
  - …



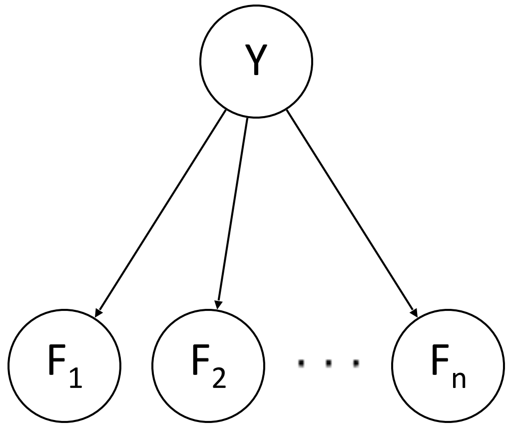
Model-Based Classification

Model-Based Classification

- Up until now: how to use a model to make optimal decisions
- Machine learning: how to acquire a model from data / experience
- 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?

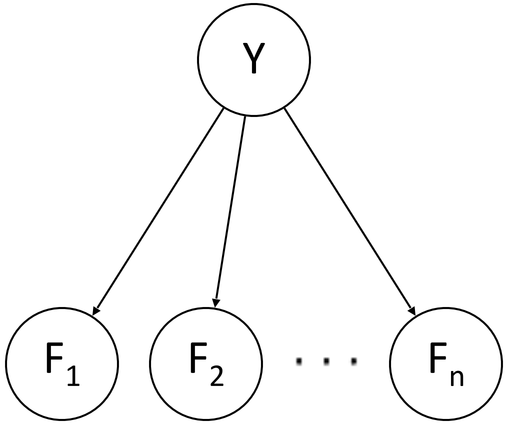
Naïve Bayes Model

- 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. Called the *prior*.
  - $P(F_i|Y)$ = One table per feature. Probability distribution over a feature, given the label.



General Naïve Bayes

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



|Y| parameters

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

|Y| x |F|^n values

n x |F| x |Y| parameters

- 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

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

+

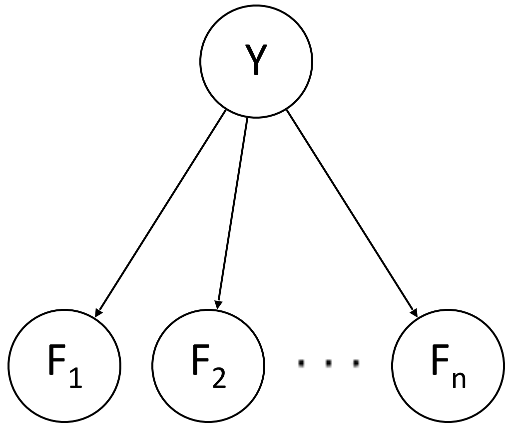
$$\Downarrow$$

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

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

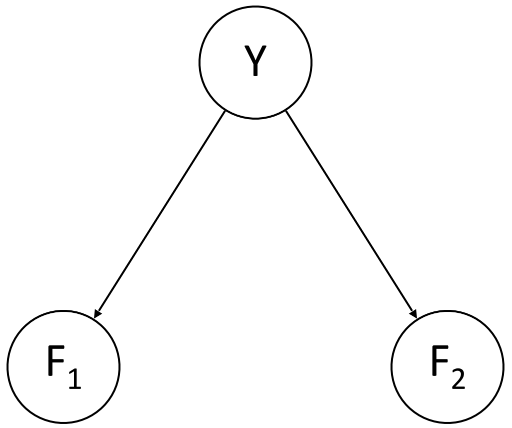
Naïve Bayes Model

- 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 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	$P(F_1\|Y)$
yes	ham	?
no	ham	?
yes	spam	?
no	spam	?

F_2: Another feature (# of occurrences of FREE)		
F_2	Y	$P(F_2\|Y)$
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.

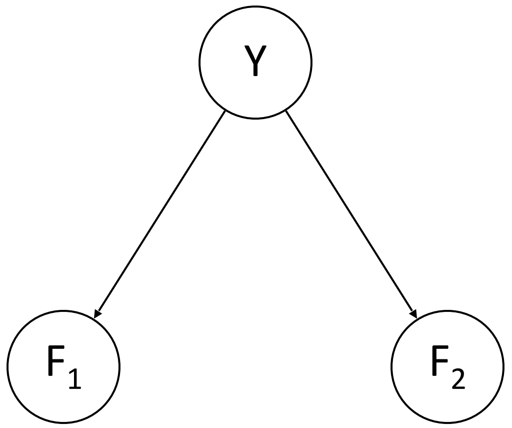
F_2: # of occurrences of FREE		
F_2	Y	$P(F_2 \mid Y)$
0	ham	0.5
1	ham	0.5
2	ham	0.0
0	spam	0.25
1	spam	0.50
2	spam	0.25

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



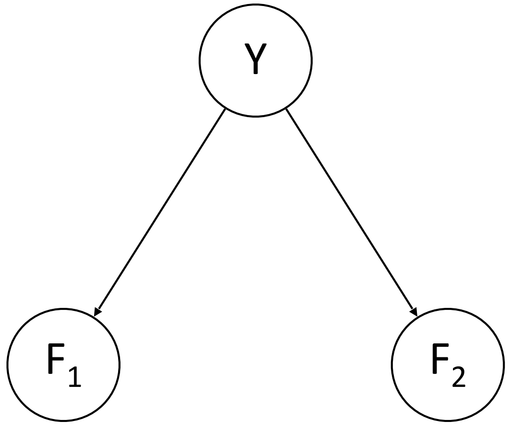
Y: The label (spam or ham)	
Y	P(Y)
ham	0.6
spam	0.4

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

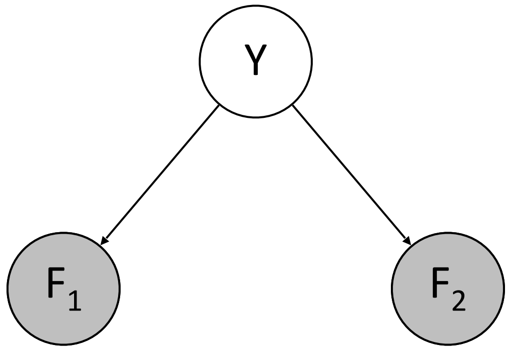
Example: Naïve Bayes for Spam Filter

- 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

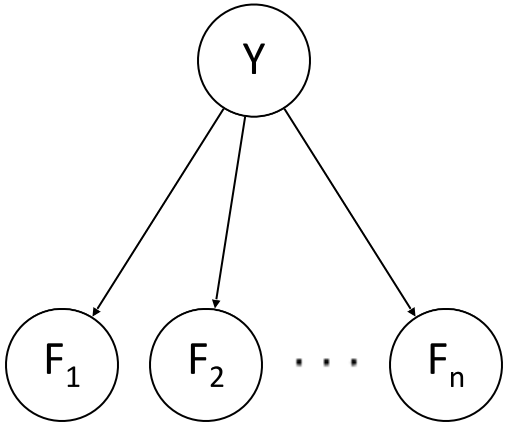
F_2: # of occurrences of FREE		
F_2	Y	$P(F_2 \mid Y)$
0	ham	0.85
1	ham	0.07
2	ham	0.08
0	spam	0.75
1	spam	0.12
2	spam	0.13

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



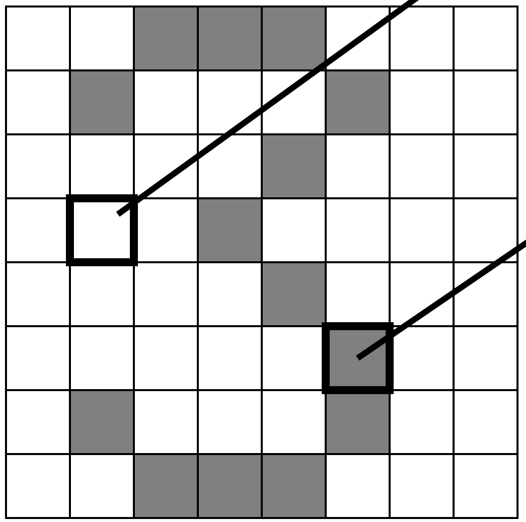
- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) = P(Y) \prod_{i,j} P(F_{i,j}|Y)$

- What do we need to learn?

Example: Conditional Probabilities

$P(Y)$

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 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? (order of the words does not count)
  - 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

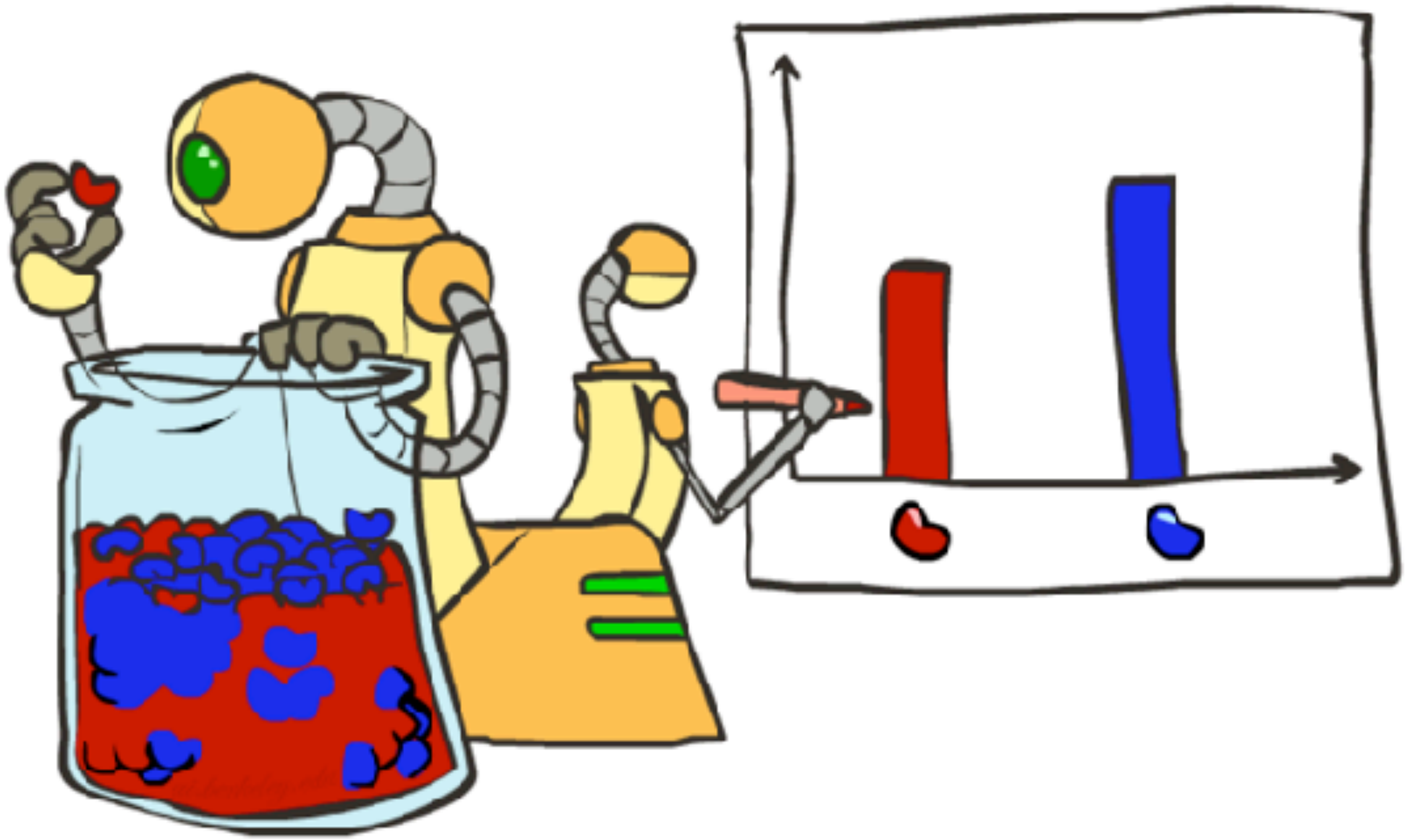
Word	P(w|spam)	P(w|ham)	Tot Spam (ln)	Tot Ham (ln)
(prior)	0.33333	0.66666	-1.1	-0.4

P(spam | w) = 98.9

General Naïve Bayes

- What do we need in order to use Naïve Bayes?
  - 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 $P(\text{observation} \mid \theta)$
  - For our problem:

$$P(\text{observation} \mid \theta)$$

$$= P(\text{randomly selected 2 red and 1 blue} \mid \theta \text{ of beans are red})$$

$$= P(\text{red} \mid \theta)\, P(\text{red} \mid \theta)\, P(\text{blue} \mid \theta)$$

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

  - We want to compute:

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

Parameter Estimation with Maximum Likelihood

- We want to compute:

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

- 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

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

Find θ that maximizes likelihood

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

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

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

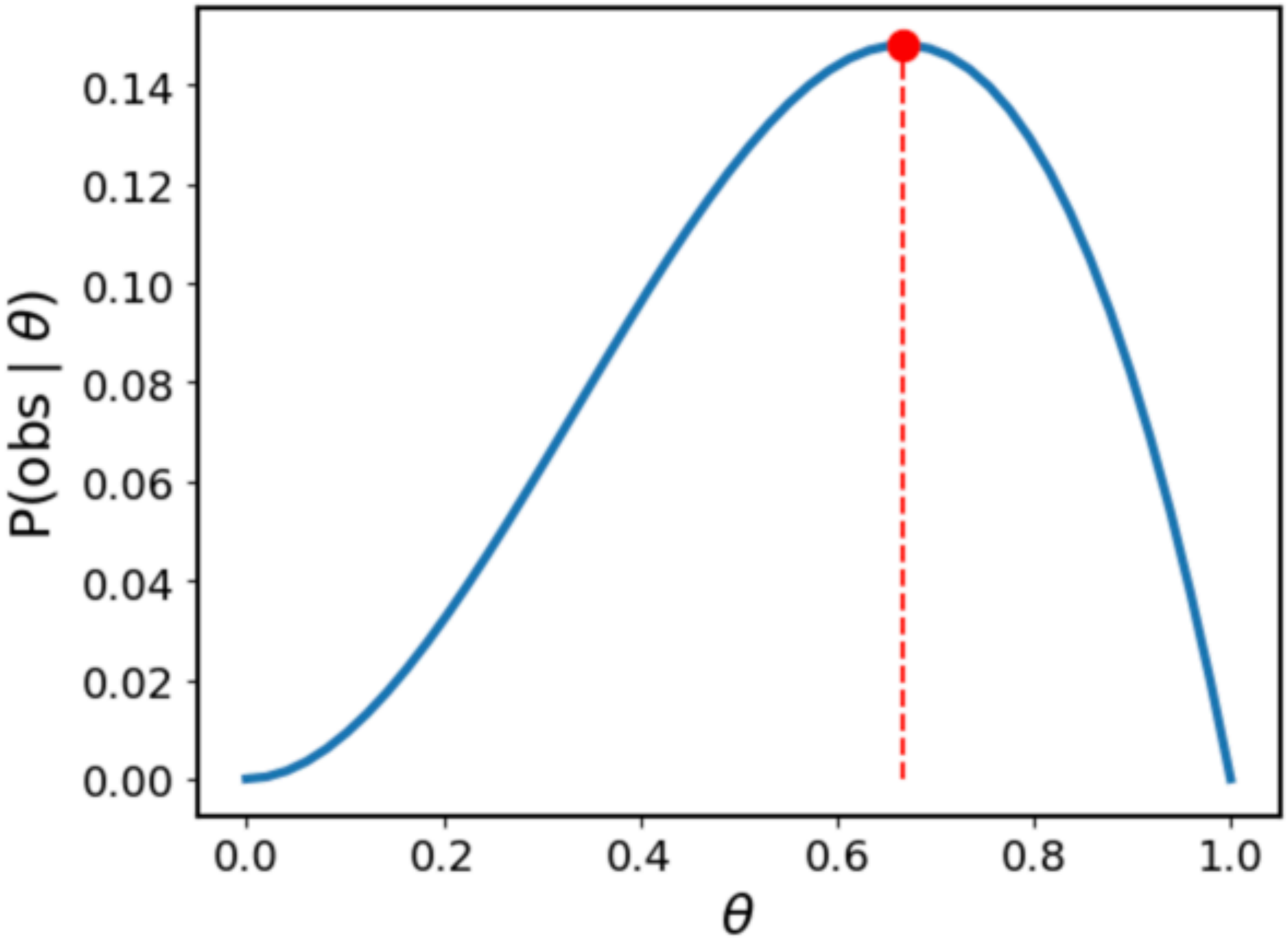
Reminder: Derivative of ln(θ) is 1/θ

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

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

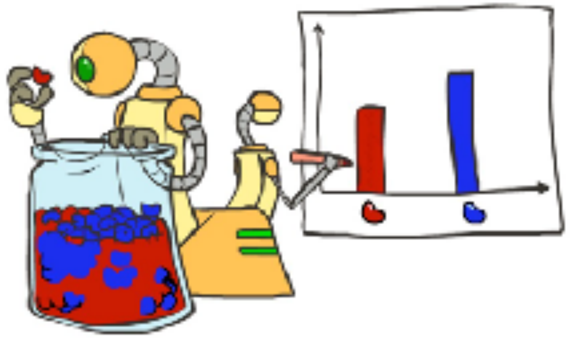
$$
\begin{aligned}
&\operatorname*{argmax}_\theta \theta^2(1-\theta) \\
=\ &\operatorname*{argmax}_\theta \ln\left(\theta^2(1-\theta)\right) \\
&\frac{d}{d\theta}\ln\left(\theta^2(1-\theta)\right) = 0 \\
&\frac{d}{d\theta}\left[\ln(\theta^2) + \ln(1-\theta)\right] = 0 \\
&\frac{d}{d\theta}\left[2\ln(\theta) + \ln(1-\theta)\right] = 0 \\
&\frac{d}{d\theta}2\ln(\theta) + \frac{d}{d\theta}\ln(1-\theta) = 0 \\
&\frac{2}{\theta} - \frac{1}{1-\theta} = 0 \\
&\theta = \frac{2}{3}
\end{aligned}
$$



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 $P(D|\theta)$:

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

Take derivative and set to 0:

$$\frac{\partial \log P(D|\theta)}{\partial \theta} = \frac{N_r}{\theta} - \frac{N_b}{1-\theta} = 0$$

$$\boxed{\rightarrow \hat{\theta} = \frac{N_r}{N_r + N_b} = \frac{\#\text{ of red balls}}{\text{total }\#\text{ of balls}}}$$

Parameter Estimation with Maximum Likelihood

- How do we estimate the conditional probability tables?
  - Maximum Likelihood, which corresponds to counting

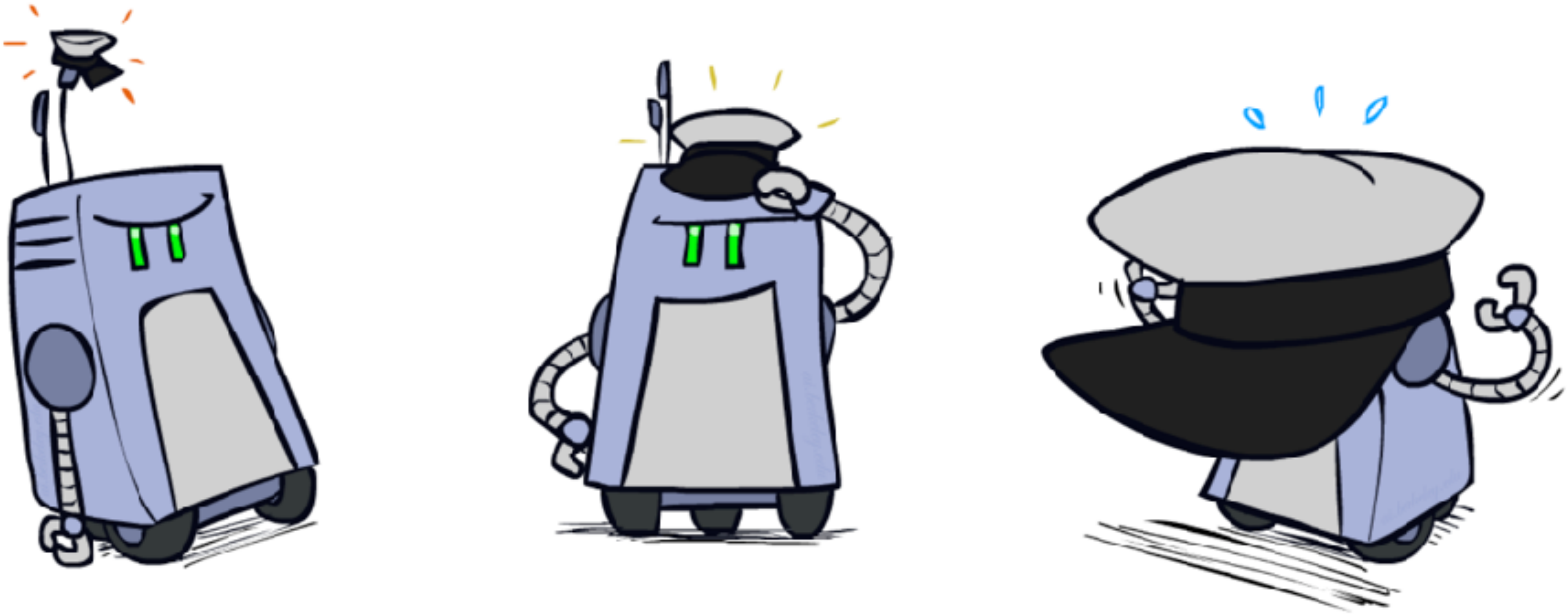
- Need to be careful though … let's see what can go wrong..

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

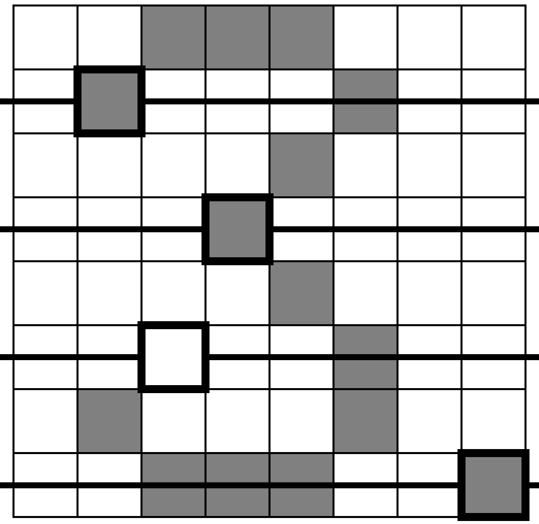
- Main worry: overfitting to the training set
  - Better with more training data (less sampling variance, training more like test)
  - Better if we limit the complexity of our hypotheses (regularization and/or small hypothesis spaces)

Generalization and Overfitting



Example: Overfitting

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



2 wins!!

Example: Overfitting

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

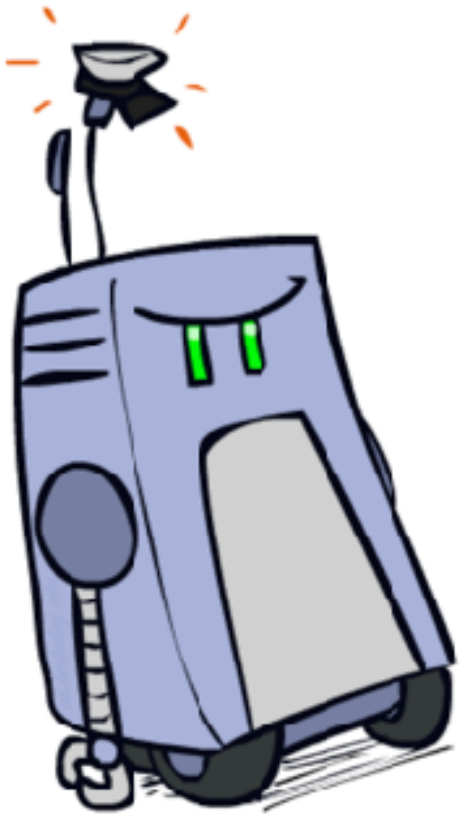
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
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?



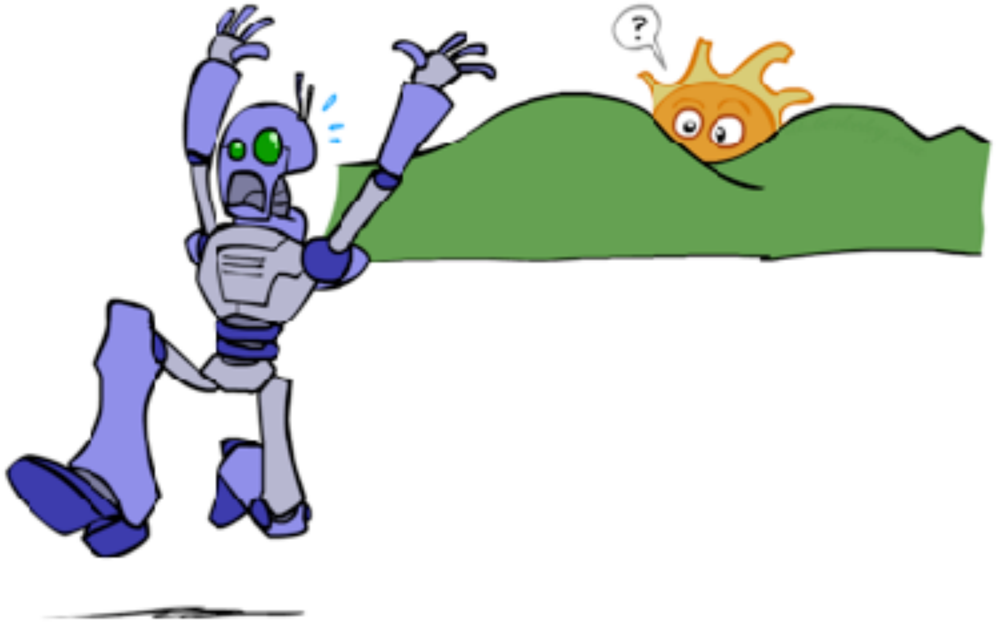
Generalization and Overfitting

- 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

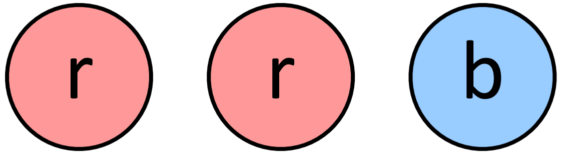
- To generalize better: we need to smooth or regularize the estimates

Unseen Events



Laplace Smoothing

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



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

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

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

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

Laplace Smoothing

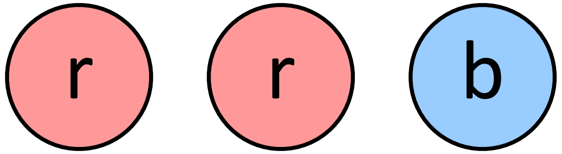
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

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

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

- Laplace for conditionals:
  - Smooth each condition independently:

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



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

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

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

Real Naïve Bayes: Smoothing

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

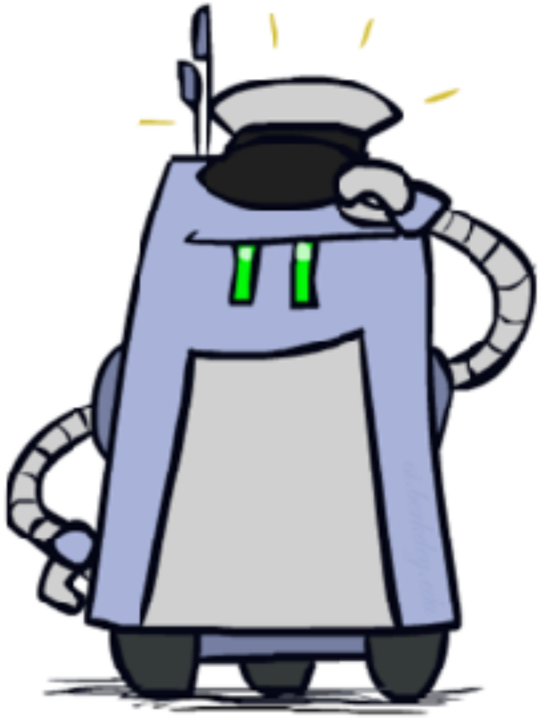
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

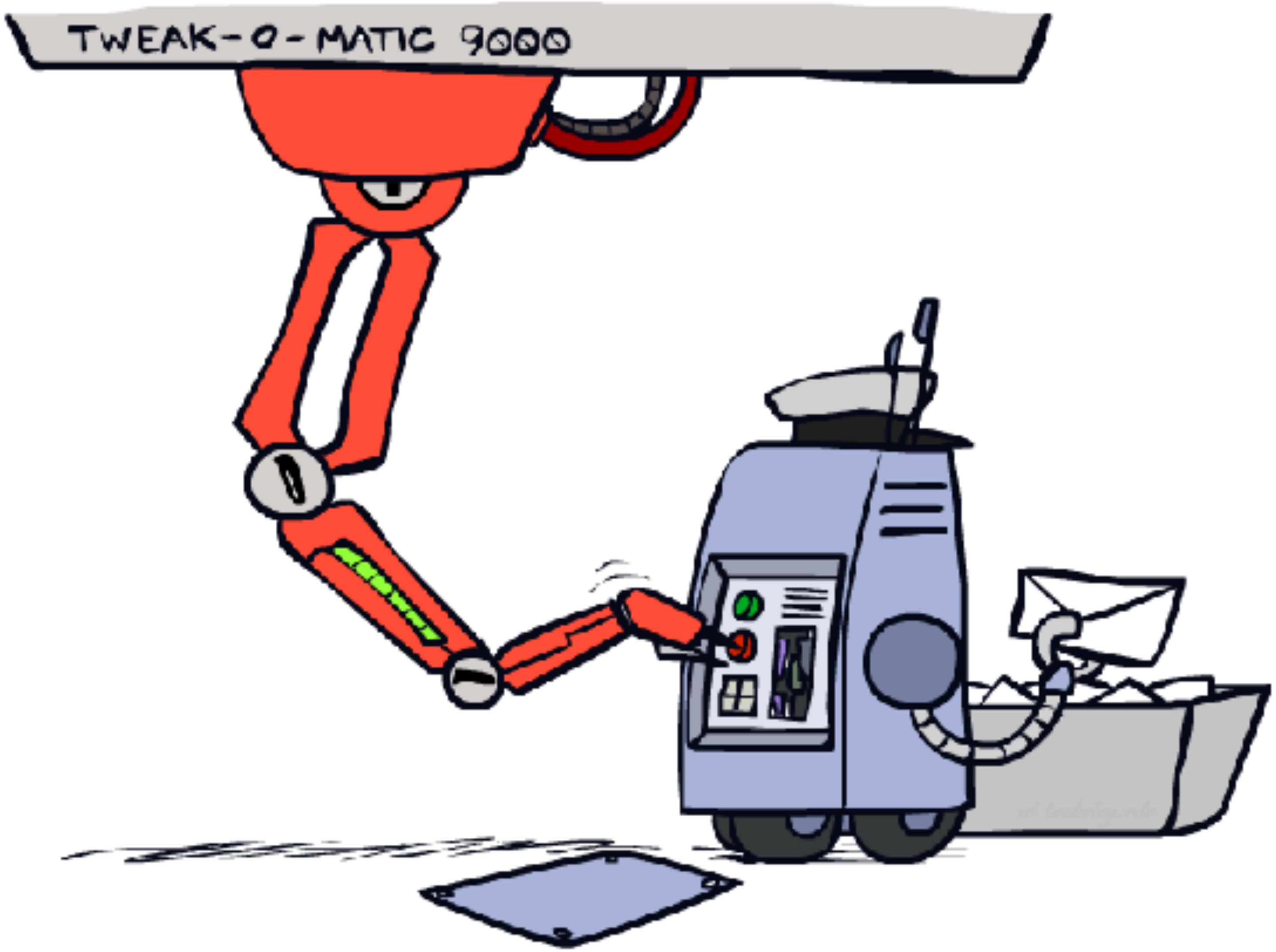
$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

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

Do these make more sense?

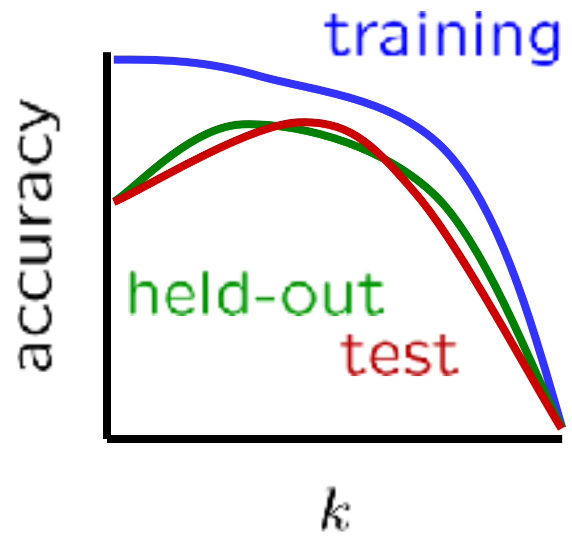


Tuning



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
  - 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



Training and Testing

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
  - Test set

- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Compute accuracy of test set
- Evaluation (many metrics possible, e.g. accuracy)
  - Accuracy: fraction of instances predicted correctly

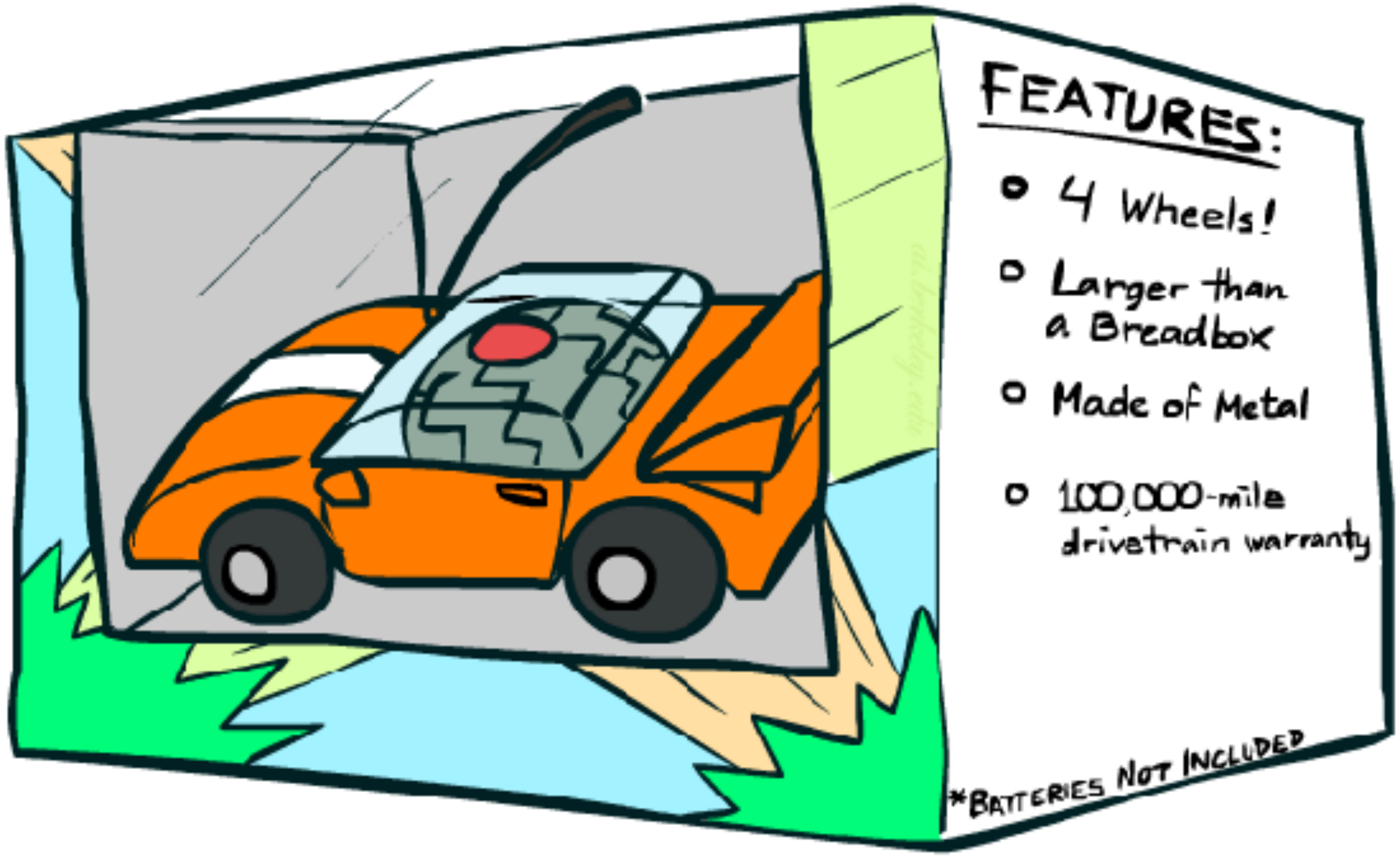
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well

Training Data

Held-Out Data

Test Data

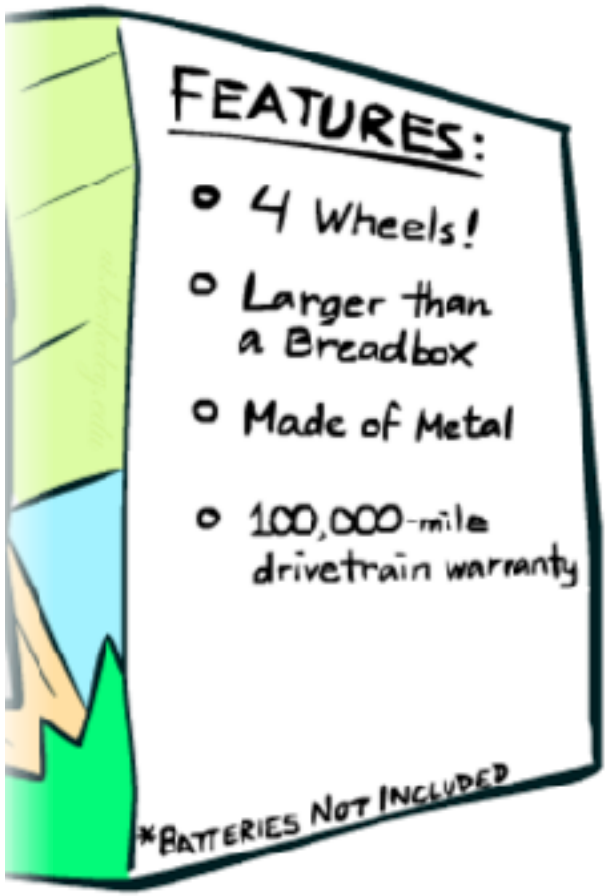
Features


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

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?

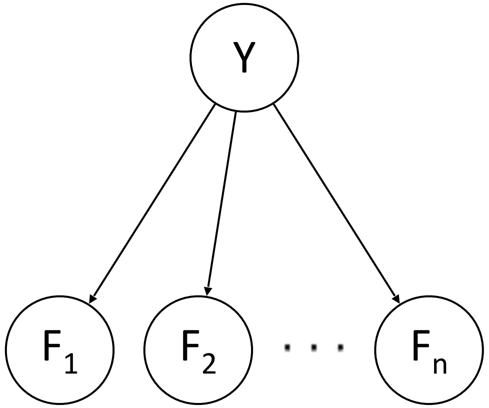
- Can add these information sources as new variables in the NB model



Summary

- The naïve Bayes assumption takes all features to be independent given the class label

  Parameters θ: probability tables $P(Y), P(F_1|Y), \ldots, P(F_n|Y)$



- We can build classifiers out of a naïve Bayes model using training data

$$P(y) = \frac{\text{\# of occurences of class } y}{\text{total \# of observations}}$$

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

- Smoothing estimates is important in real systems

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