

CS 4100 // artificial intelligence

instructor: [byron wallace](#)

Machine learning I

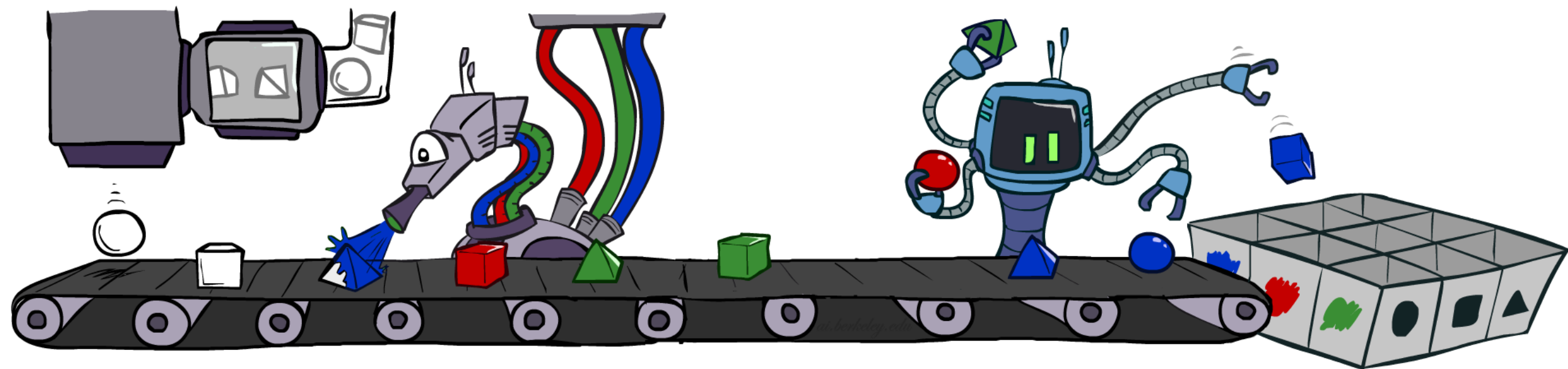


But first: finishing/recapping sampling!

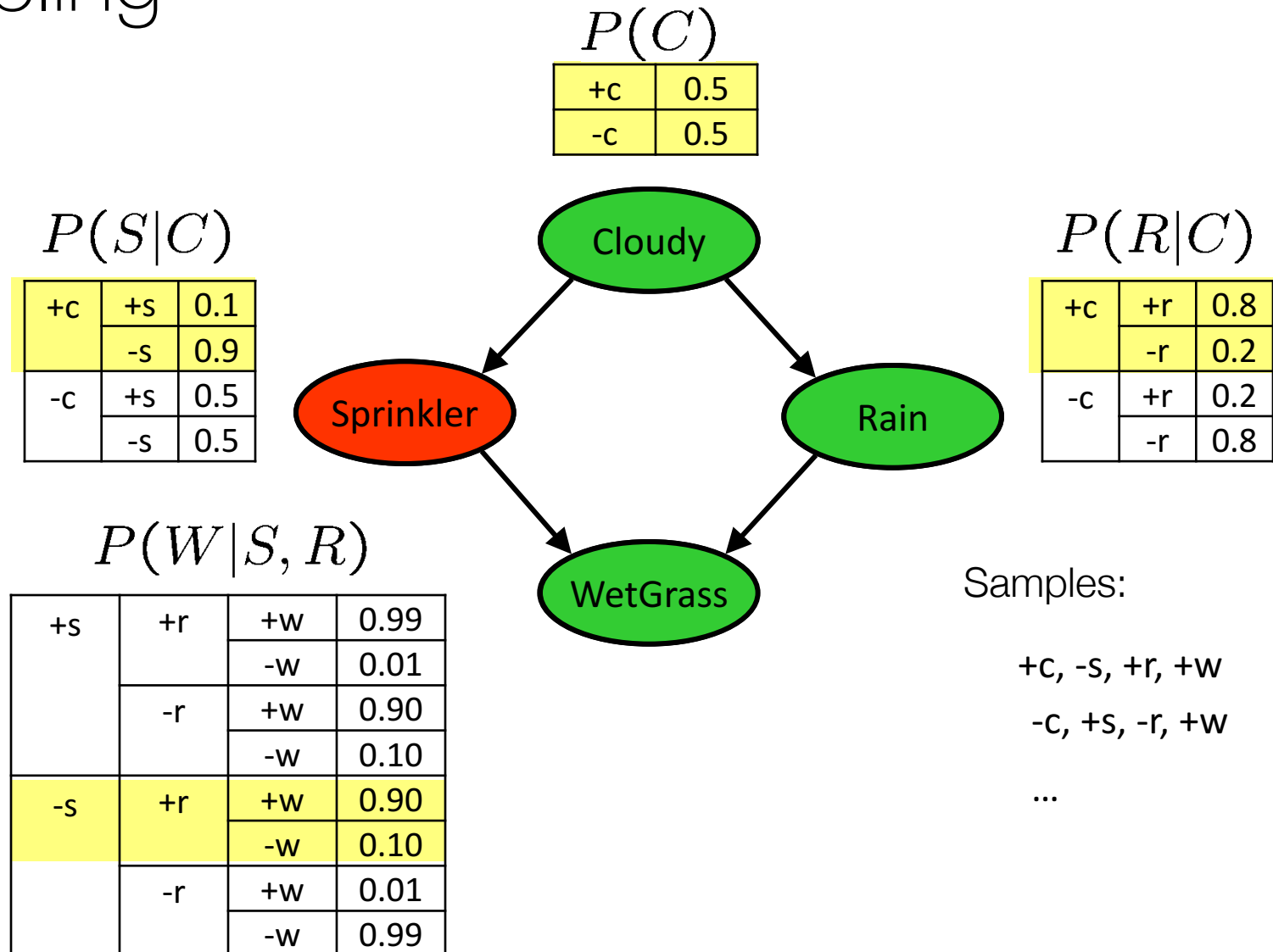
Sampling in Bayes' nets

- Prior Sampling
- Rejection Sampling
- Likelihood Weighting
- Gibbs Sampling

Prior sampling



Prior sampling

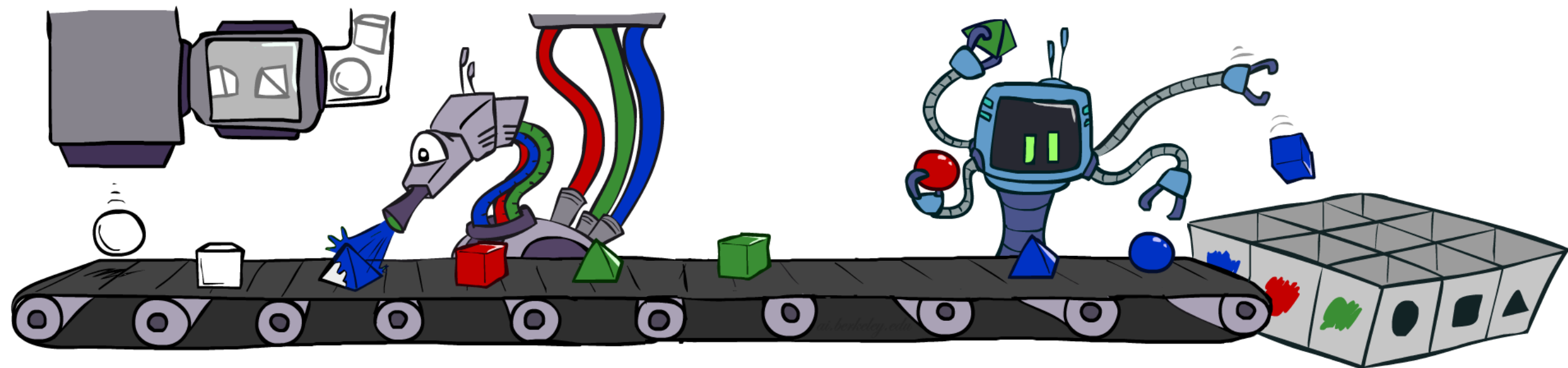


Prior sampling

For $i=1, 2, \dots, n$

Sample x_i from $P(X_i \mid \text{Parents}(X_i))$

Return (x_1, x_2, \dots, x_n)



Example

We'll get a bunch of samples from the BN:

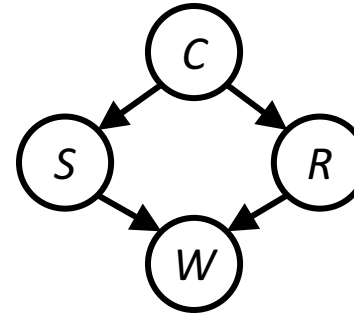
+C, -S, +r, +W

+C, +S, +r, +W

-C, +S, +r, -W

+C, -S, +r, +W

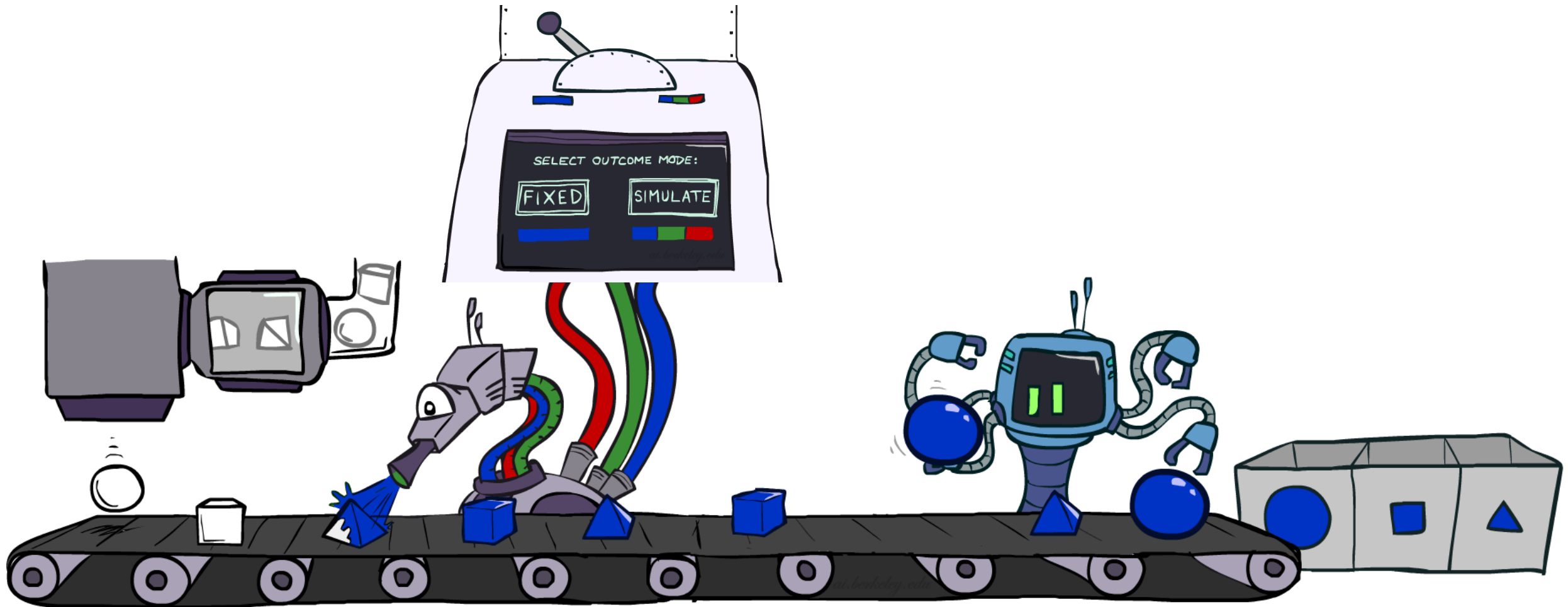
-C, -S, -r, +W



If we want to know $P(W)$

- We have counts $\langle +w:4, -w:1 \rangle$
- Normalize to get $P(W) = \langle +w:0.8, -w:0.2 \rangle$
- This will get closer to the true distribution with more samples
- Can estimate anything else, too
- What about $P(C | +w)$? $P(C | +r, +w)$? $P(C | -r, -w)$?
- Fast: can use fewer samples if less time (what's the drawback?)

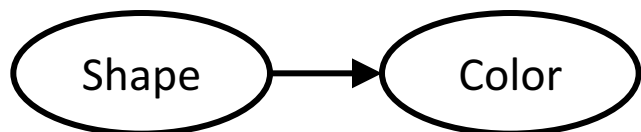
Likelihood weighting



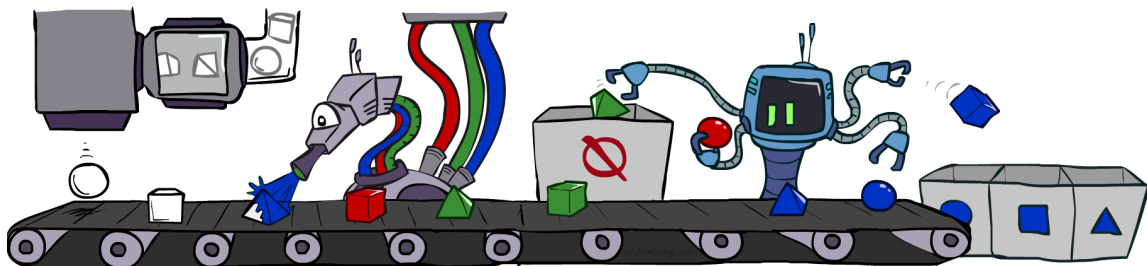
Likelihood weighting

Problem with rejection sampling:

- If evidence is unlikely, rejects lots of samples
- Evidence not exploited as you sample
- Consider $P(\text{Shape}|\text{blue})$

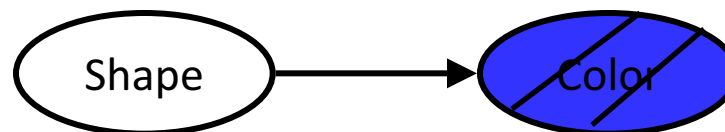


~~pyramid, green~~
~~pyramid, red~~
sphere, blue
cube, red
~~sphere, green~~

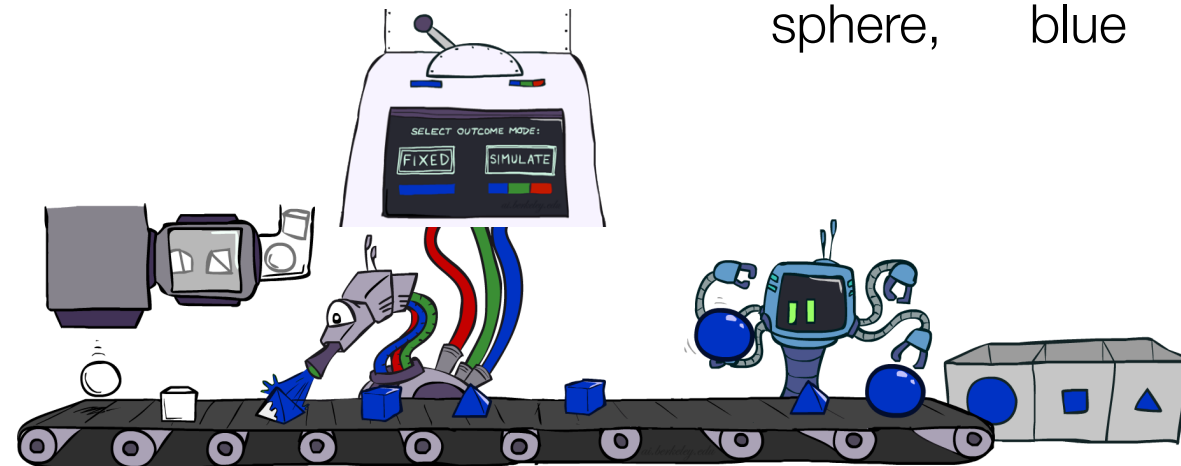


Idea: fix evidence variables and sample the rest

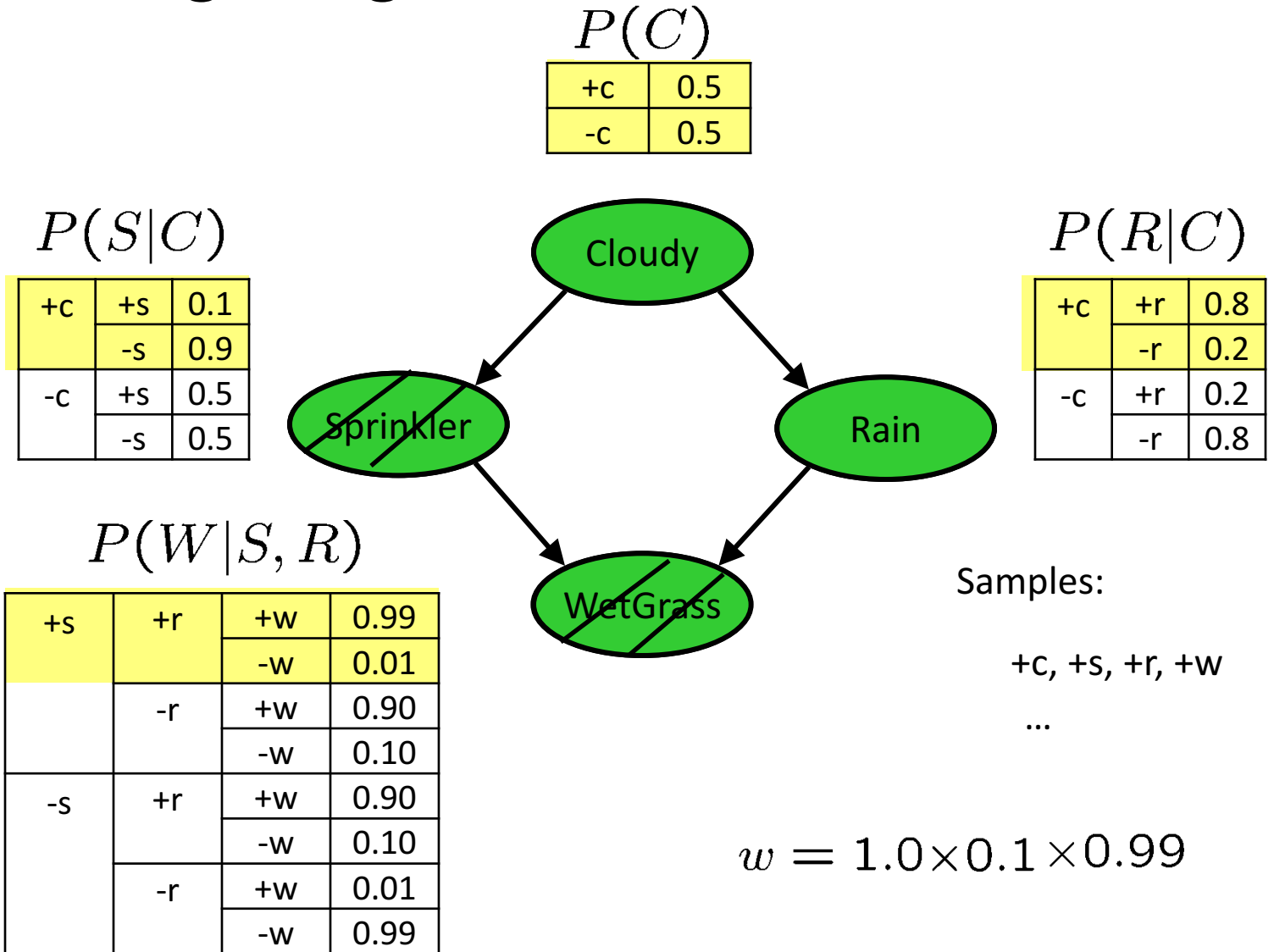
- Problem: sample distribution not consistent!
- Solution: weight by probability of evidence given parents



pyramid, blue
pyramid, blue
sphere, blue
cube, blue
sphere, blue



Likelihood weighting



Likelihood weighting

```
IN: evidence instantiation
w = 1.0
for i=1, 2, ..., n
    if  $X_i$  is an evidence variable
         $X_i = \text{observation } x_i \text{ for } X_i$ 
        Set  $w = w * P(x_i | \text{Parents}(X_i))$ 
    else
        Sample  $x_i$  from  $P(X_i | \text{Parents}(X_i))$ 
return  $(x_1, x_2, \dots, x_n), w$ 
```

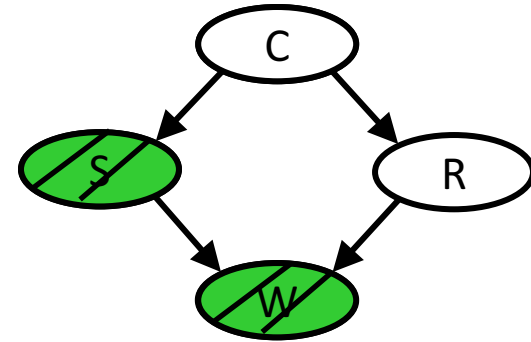
Likelihood weighting

Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(z, e) = \prod_{i=1}^l P(z_i | \text{Parents}(Z_i))$$

Now, samples have weights

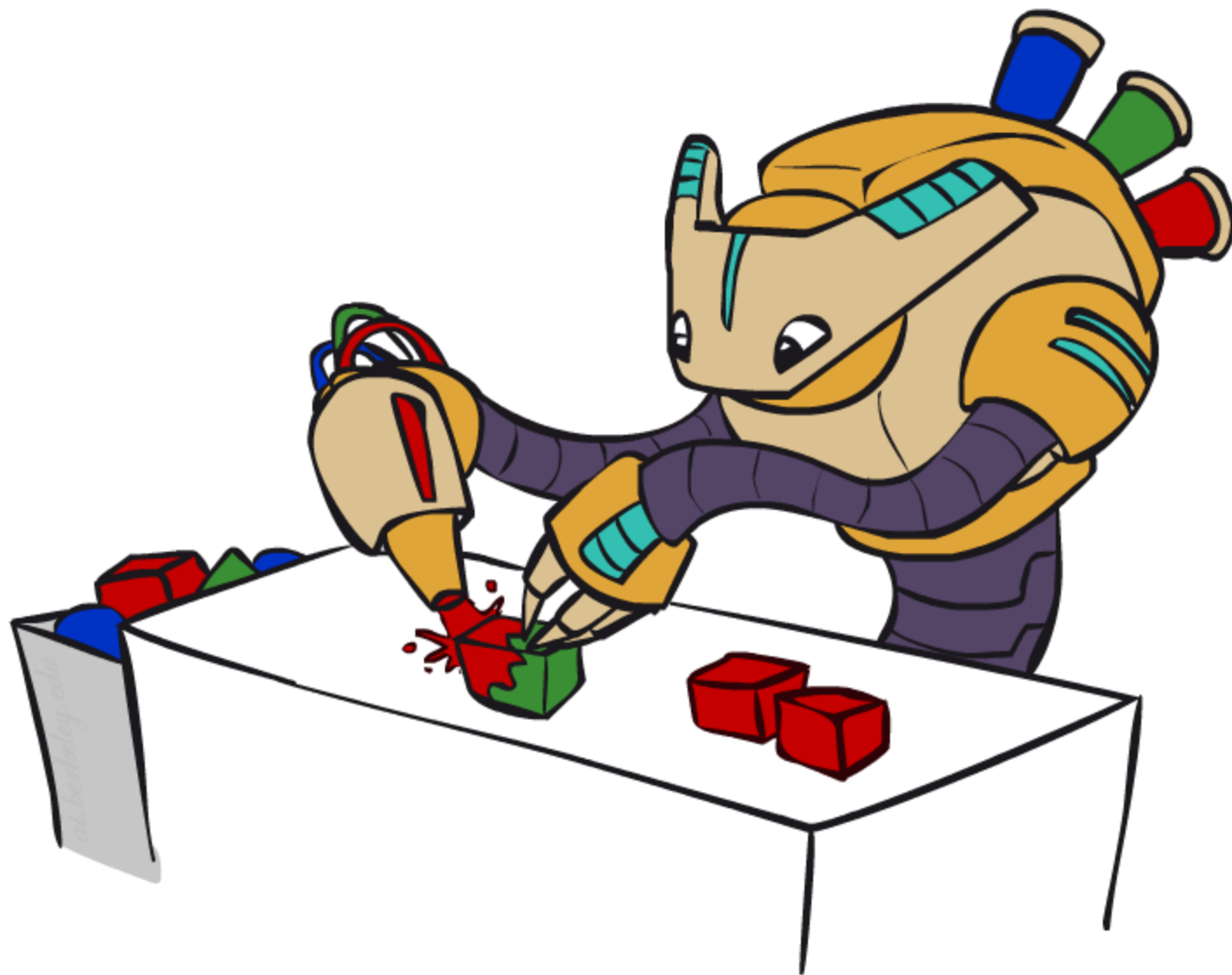
$$w(z, e) = \prod_{i=1}^m P(e_i | \text{Parents}(E_i))$$



Together, weighted sampling distribution is consistent

$$\begin{aligned} S_{WS}(z, e) \cdot w(z, e) &= \prod_{i=1}^l P(z_i | \text{Parents}(z_i)) \prod_{i=1}^m P(e_i | \text{Parents}(e_i)) \\ &= P(z, e) \end{aligned}$$

Gibbs sampling



Gibbs sampling

- *Procedure*: keep track of a full instantiation x_1, x_2, \dots, x_n . Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.

Gibbs sampling

- *Procedure*: keep track of a full instantiation x_1, x_2, \dots, x_n . Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.
- *Property*: in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution

Gibbs sampling

- *Procedure*: keep track of a full instantiation x_1, x_2, \dots, x_n . Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.
- *Property*: in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution
- *Rationale*: both upstream and downstream variables condition on evidence.

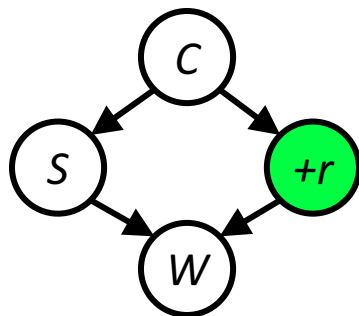
Gibbs sampling

- *Procedure*: keep track of a full instantiation x_1, x_2, \dots, x_n . Start with an arbitrary instantiation consistent with the evidence. Sample one variable at a time, conditioned on all the rest, but keep evidence fixed. Keep repeating this for a long time.
- *Property*: in the limit of repeating this infinitely many times the resulting sample is coming from the correct distribution
- *Rationale*: both upstream and downstream variables condition on evidence.
- In contrast: likelihood weighting only conditions on upstream evidence, and hence weights obtained in likelihood weighting can sometimes be very small. Sum of weights over all samples is indicative of how many “effective” samples were obtained, so want high weight.

Gibbs sampling example: $P(S \mid +r)$

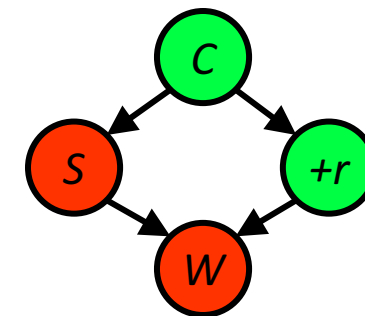
Step 1: Fix evidence

$R = +r$



Step 2: Initialize other variables

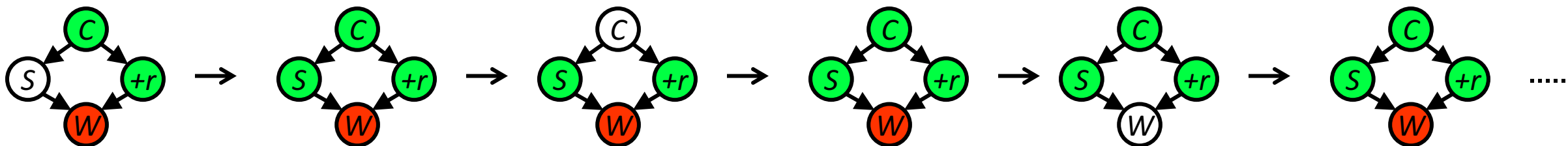
Randomly



Steps 3: Repeat

Choose a non-evidence variable X

Resample X from $P(X \mid \text{all other variables})$



Sample from $P(S \mid +c, -w, +r)$

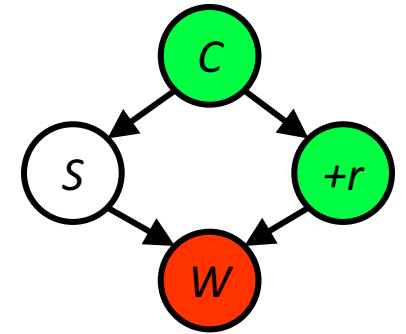
Sample from $P(C \mid +s, -w, +r)$

Sample from $P(W \mid +s, +c, +r)$

Efficient resampling of one variable

Sample from $P(S \mid +c, +r, -w)$

$$\begin{aligned} P(S \mid +c, +r, -w) &= \frac{P(S, +c, +r, -w)}{P(+c, +r, -w)} \\ &= \frac{P(S, +c, +r, -w)}{\sum_s P(s, +c, +r, -w)} \\ &= \frac{P(+c)P(S \mid +c)P(+r \mid +c)P(-w \mid S, +r)}{\sum_s P(+c)P(s \mid +c)P(+r \mid +c)P(-w \mid s, +r)} \\ &= \frac{P(+c)P(S \mid +c)P(+r \mid +c)P(-w \mid S, +r)}{P(+c)P(+r \mid +c) \sum_s P(s \mid +c)P(-w \mid s, +r)} \\ &= \frac{P(S \mid +c)P(-w \mid S, +r)}{\sum_s P(s \mid +c)P(-w \mid s, +r)} \end{aligned}$$

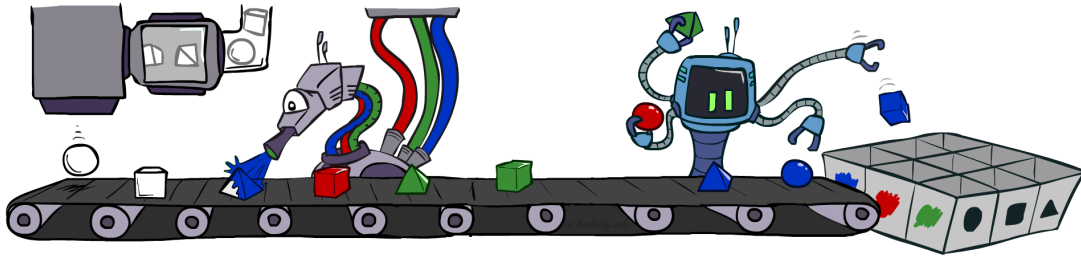


Many things cancel out – only CPTs with S remain!

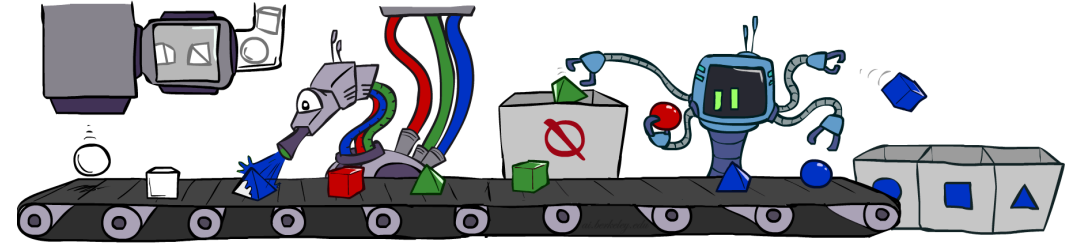
More generally: only CPTs that have resampled variable need to be considered, and joined together

Bayes' net sampling summary

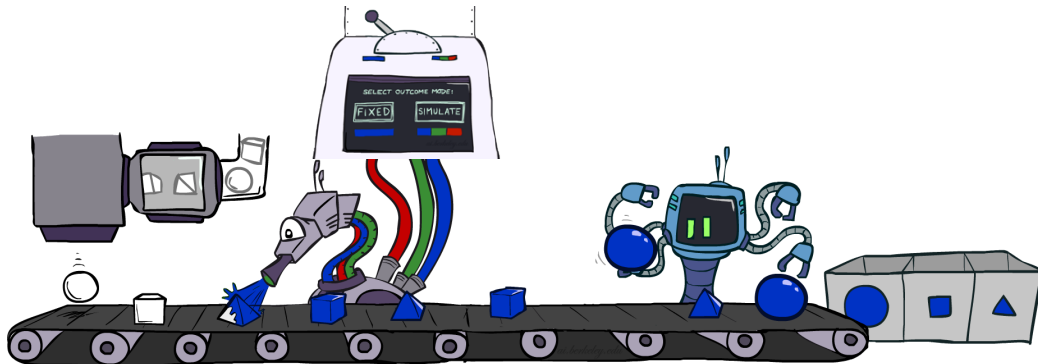
Prior Sampling P



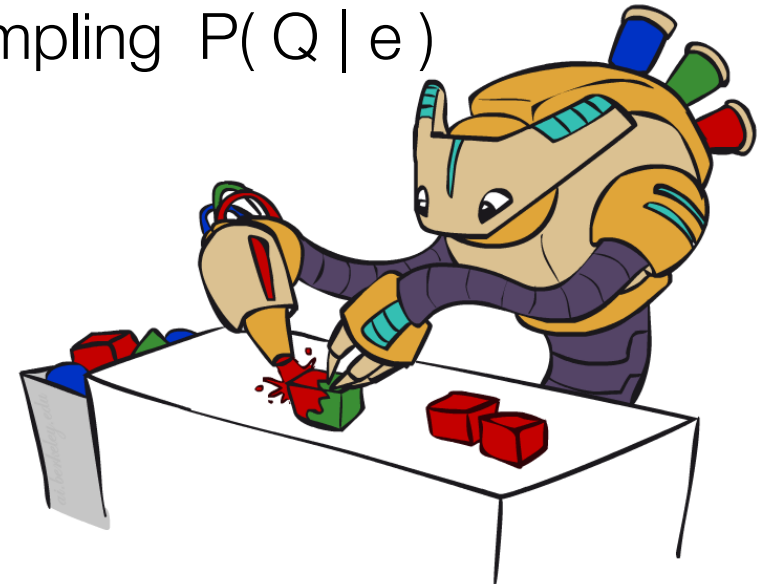
Rejection Sampling $P(Q | e)$



Likelihood Weighting $P(Q | e)$



Gibbs Sampling $P(Q | e)$



Machine learning

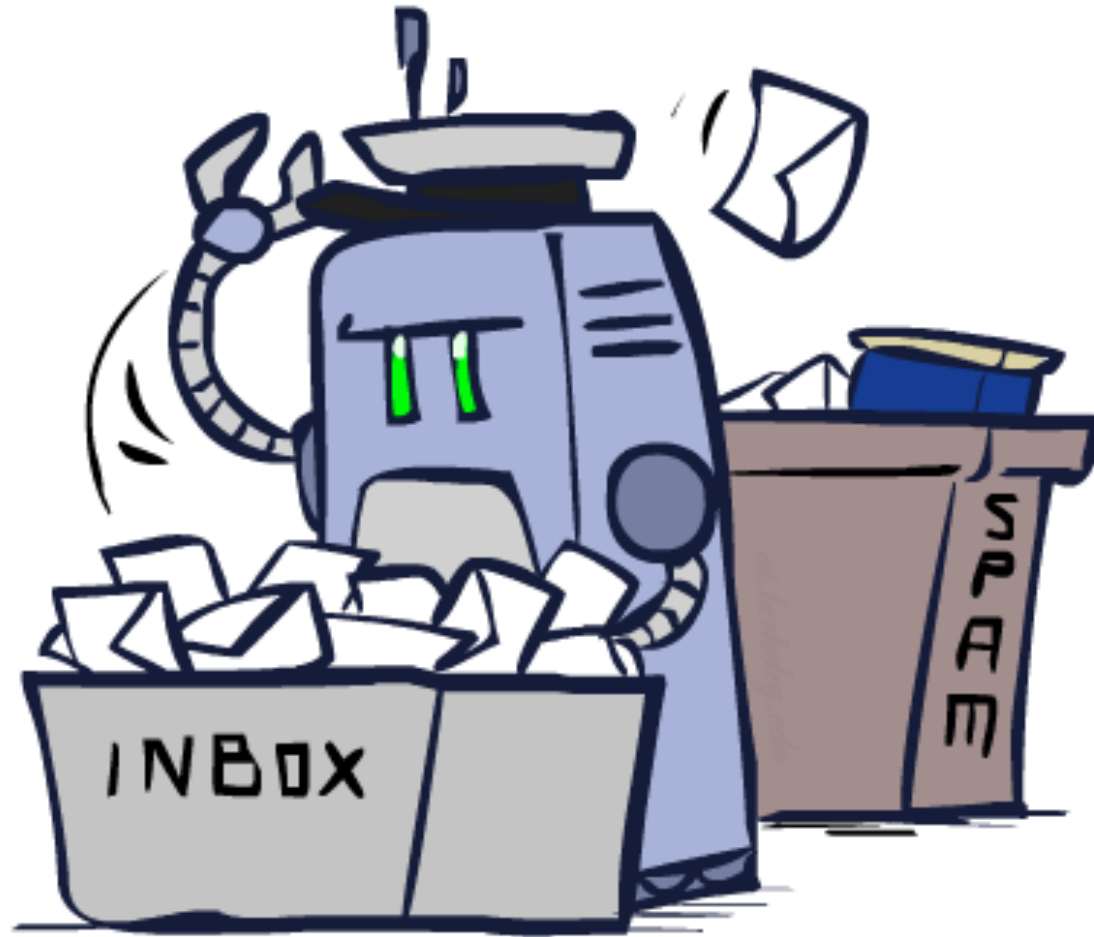
Up until now: (mostly) how use a model to make optimal decisions

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/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
- ...



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

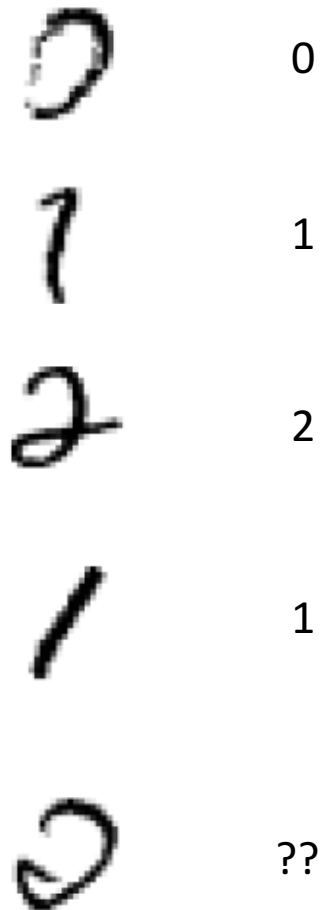
Output: a digit 0-9

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)

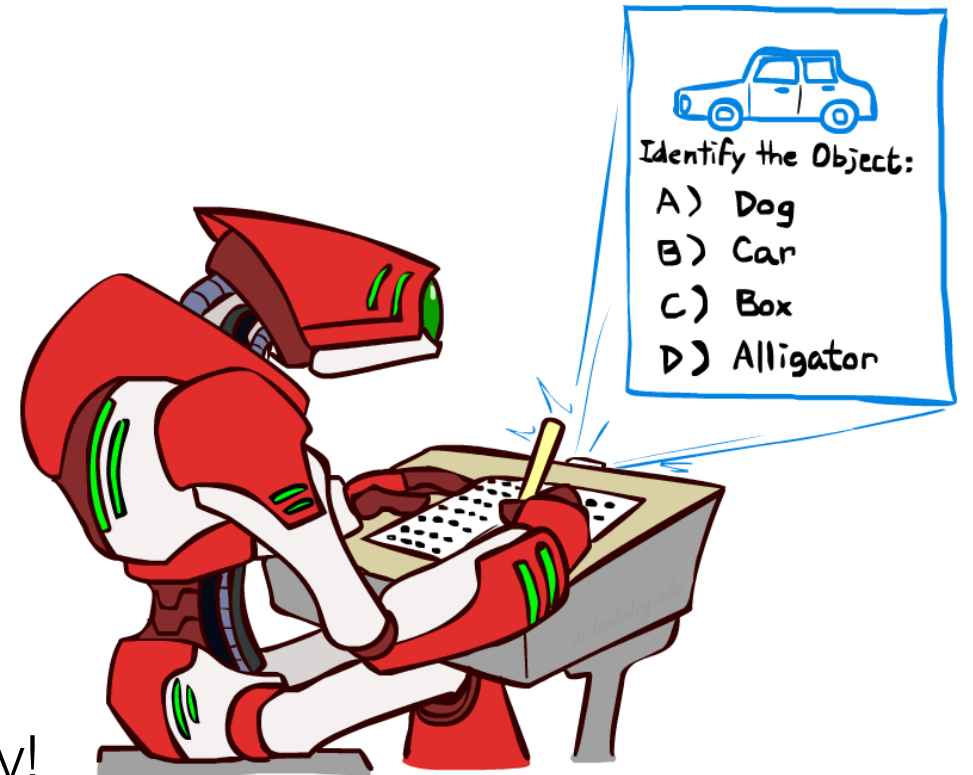
Automatic essay grading (input: document, classes: grades)

Fraud detection (input: account activity,
classes: fraud / no fraud)

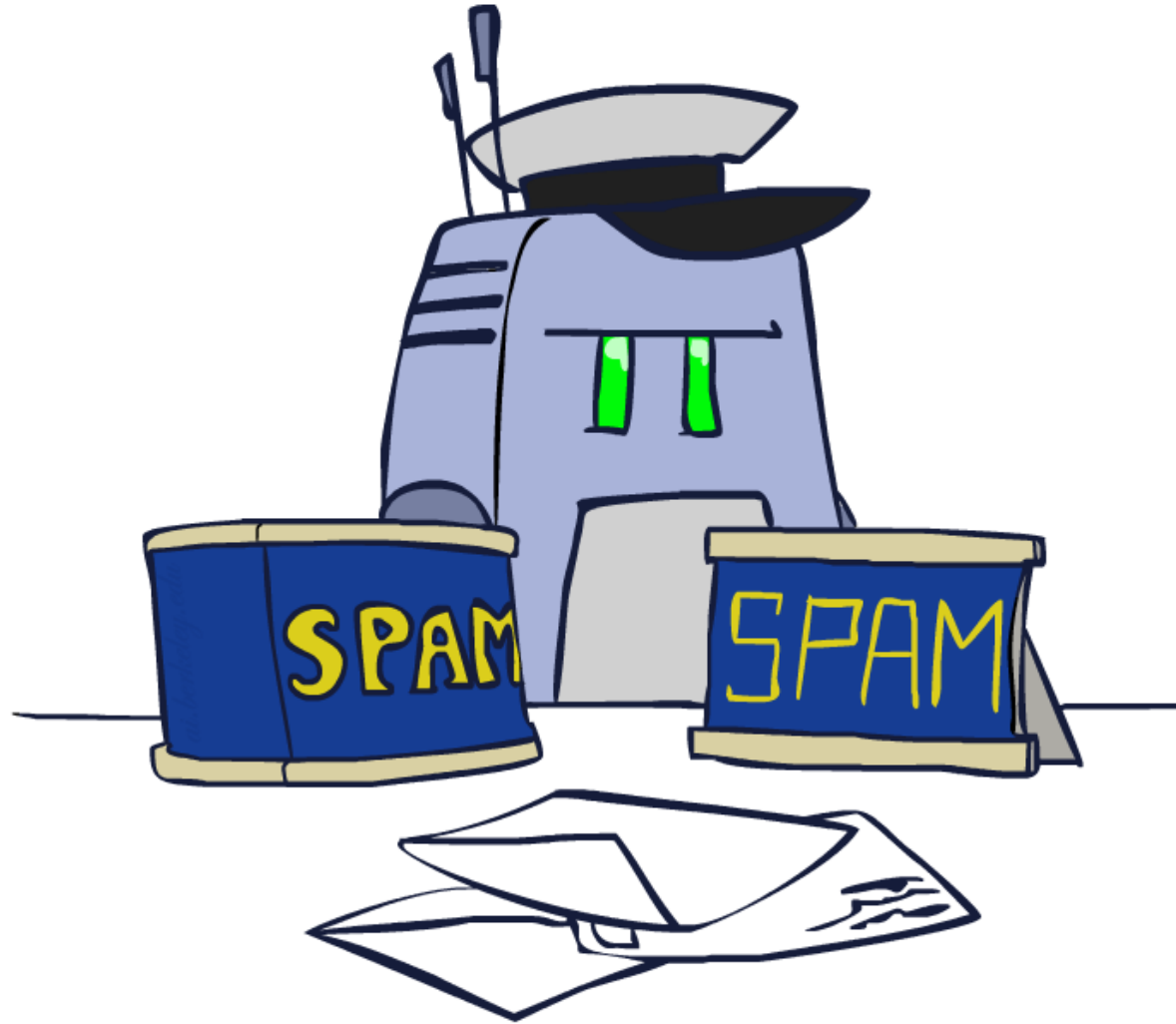
Customer service email routing

... *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

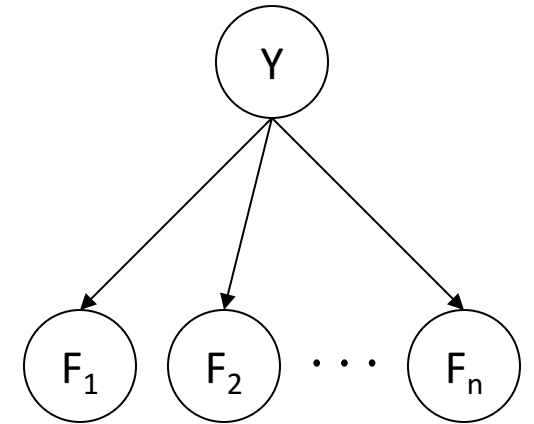
Naïve Bayes: Assume all features are independent effects of the label

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

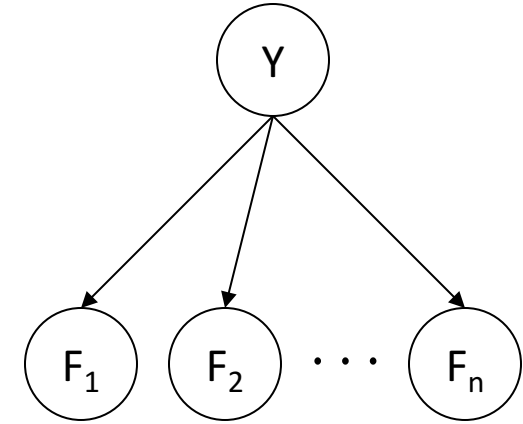
A general Naive Bayes model:

$|Y|$ 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

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

$$P(f_1 \dots f_n)$$

+ ↶

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

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

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 $P(Y|F_1 \dots F_n)$
- Nothing new here

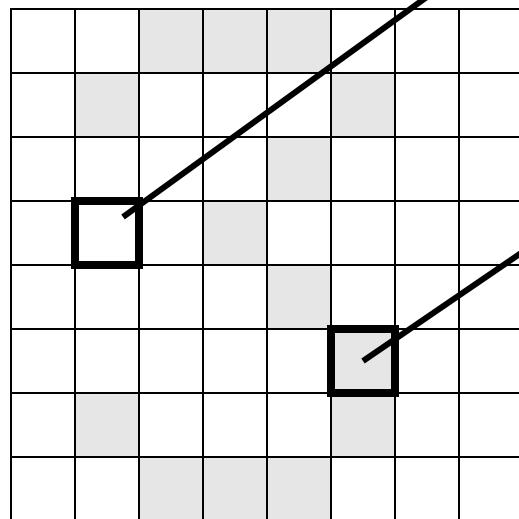
Estimates of local conditional probability tables

- $P(Y)$, 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 to be estimated (*learned*) from *training data*

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?
- 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
...	

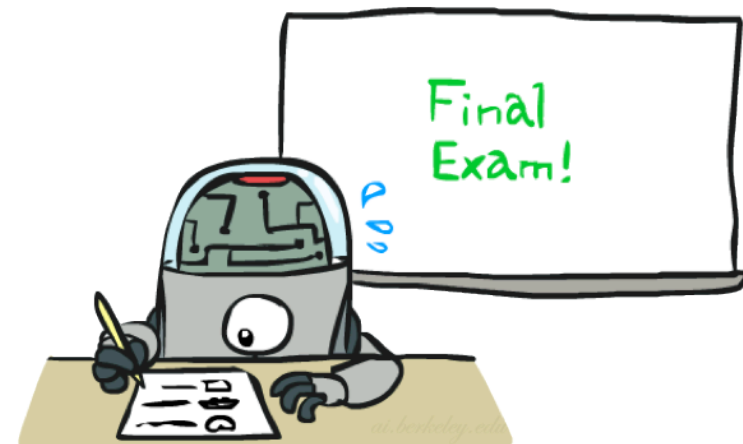
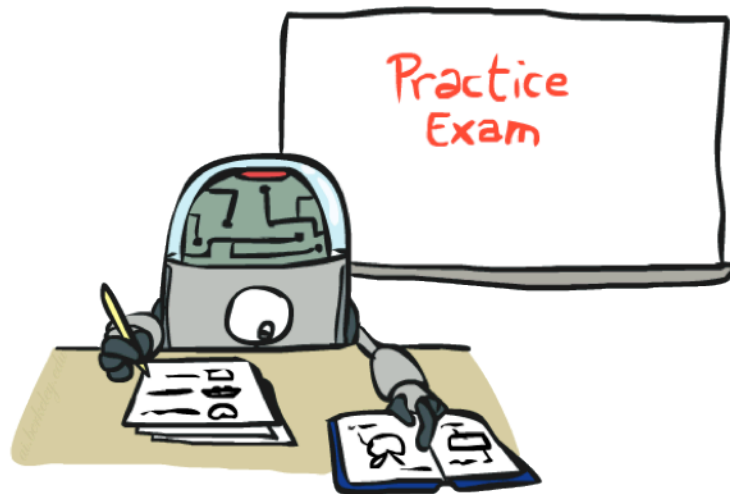
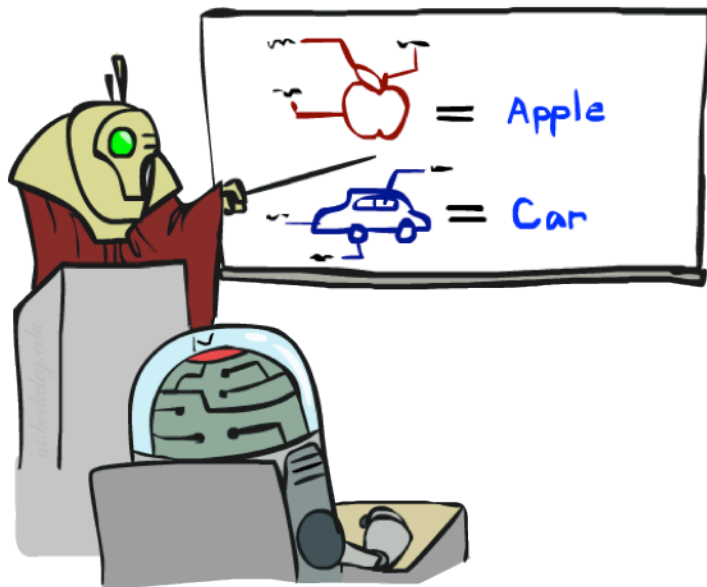
- Where do these tables come from?

Spam example

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

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

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

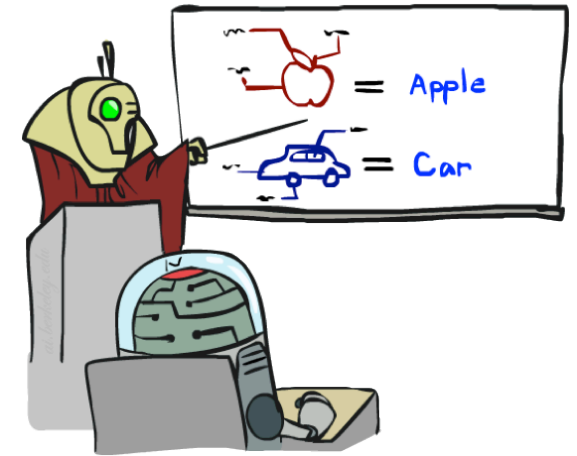
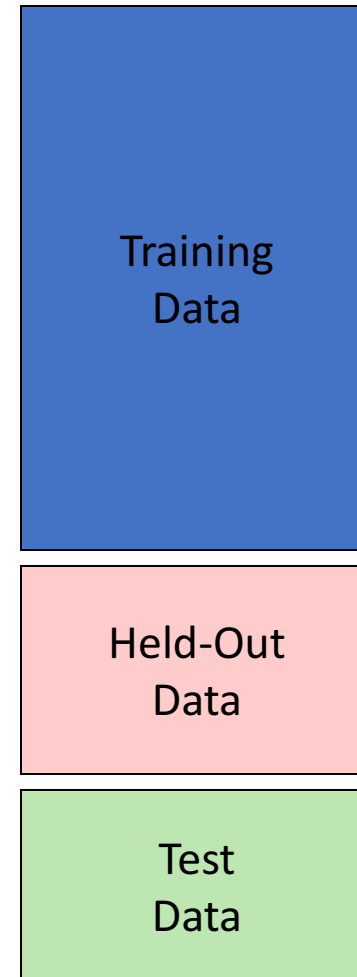
- Learn parameters (e.g. model probabilities) on training set
- (Tune hyperparameters on held-out set)
- Compute accuracy of test set
- Very important: never “peek” at the test set!

Evaluation

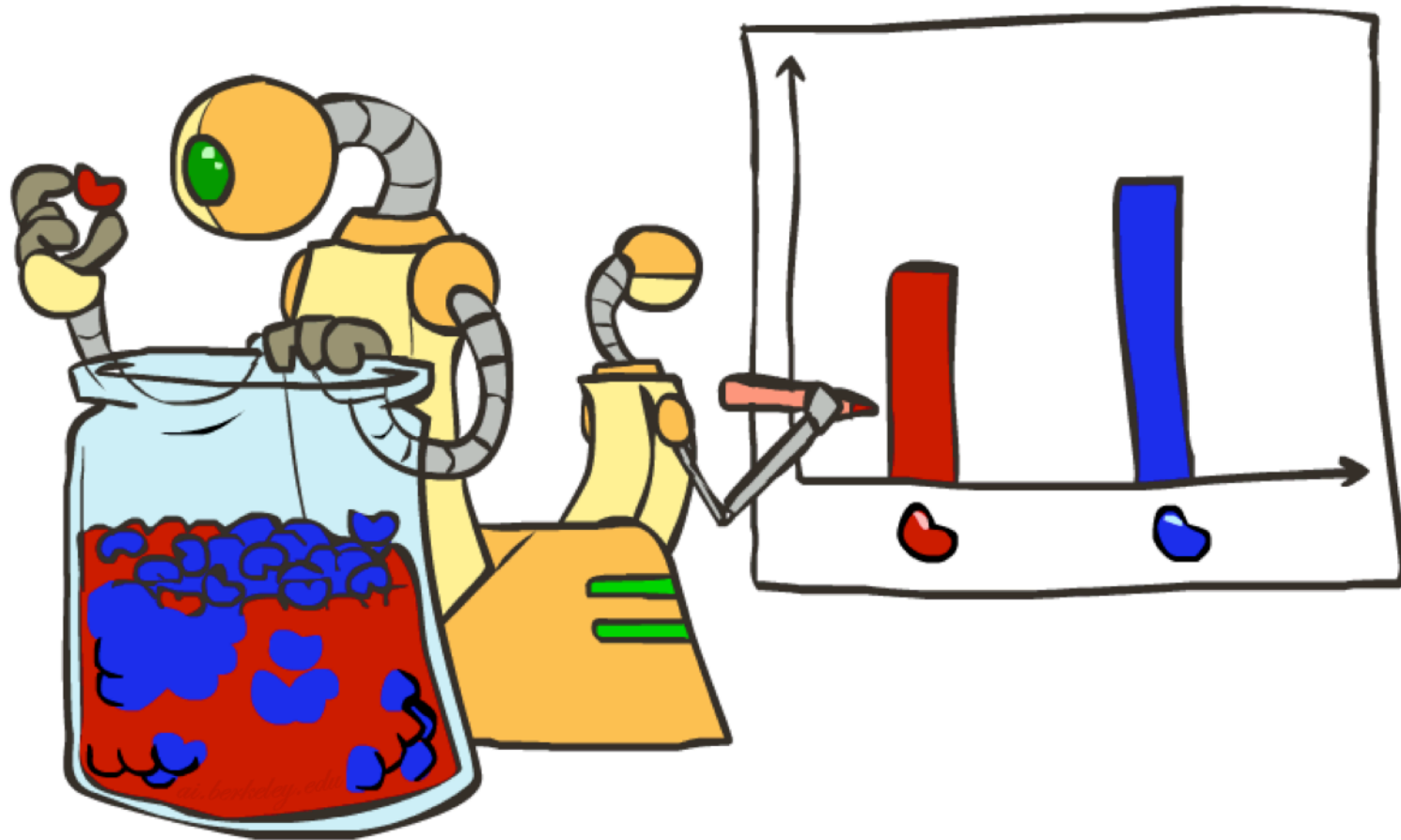
- Accuracy: fraction of instances predicted correctly

Overfitting and generalization

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



Parameter estimation



Parameter estimation

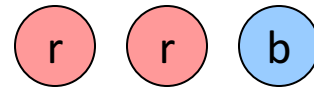
Estimating the distribution of a random variable

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:

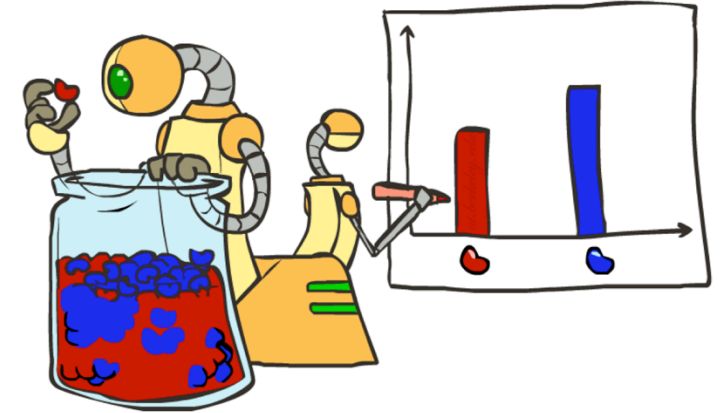
$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$



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

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

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



Maximum likelihood

Relative frequencies are the maximum likelihood estimates

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

Your turn: exercise on Naïve Bayes!

Maximum likelihood

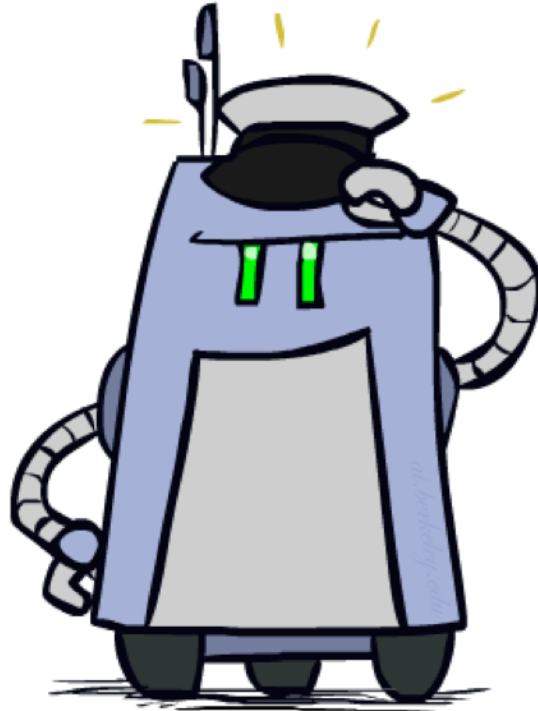
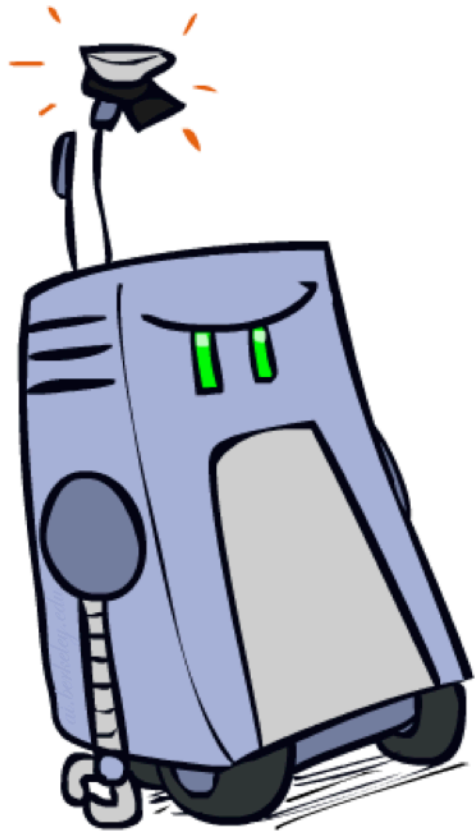
Relative frequencies are the maximum likelihood estimates

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

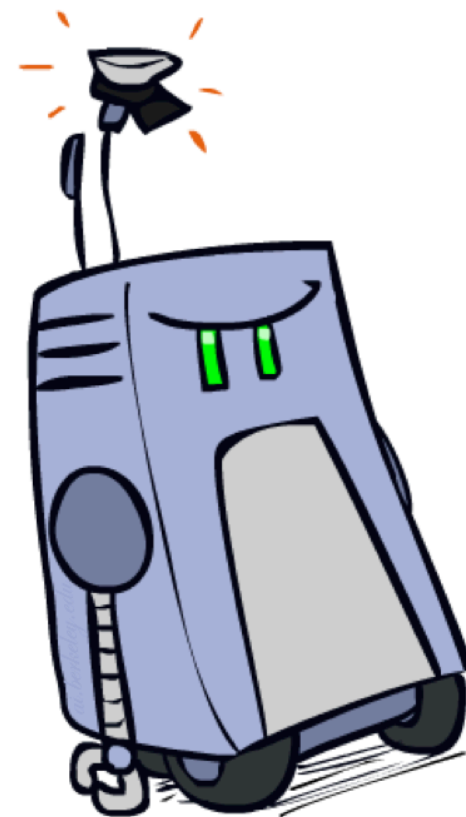
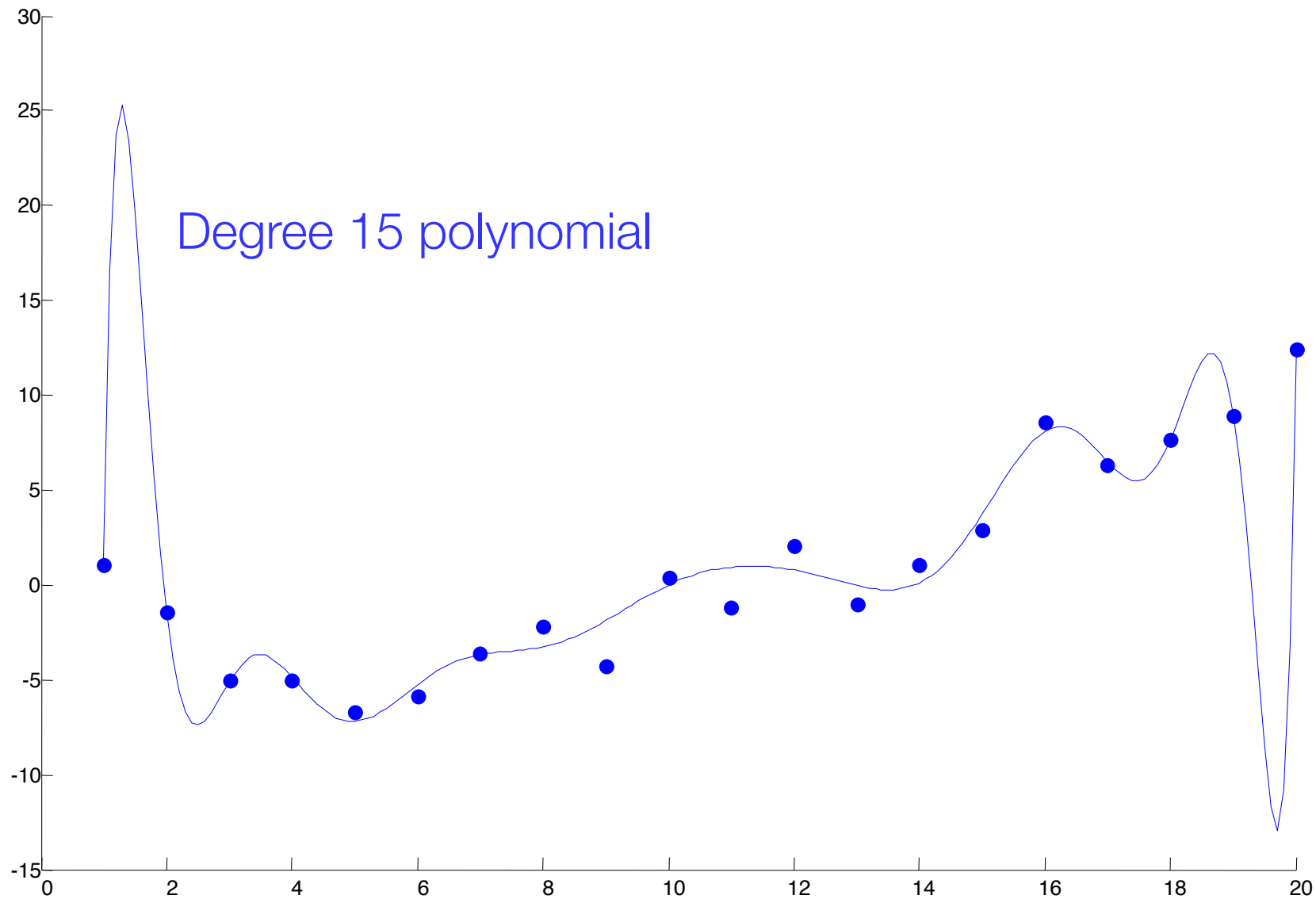
Another option is to consider the most likely parameter value given the data

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

Generalization and overfitting



Overfitting



Example: overfitting

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

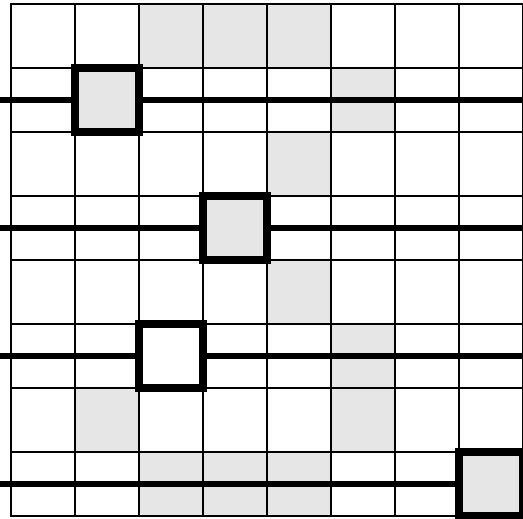
$$P(C = 2) = 0.1$$

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

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

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

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



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

$$P(C = 3) = 0.1$$

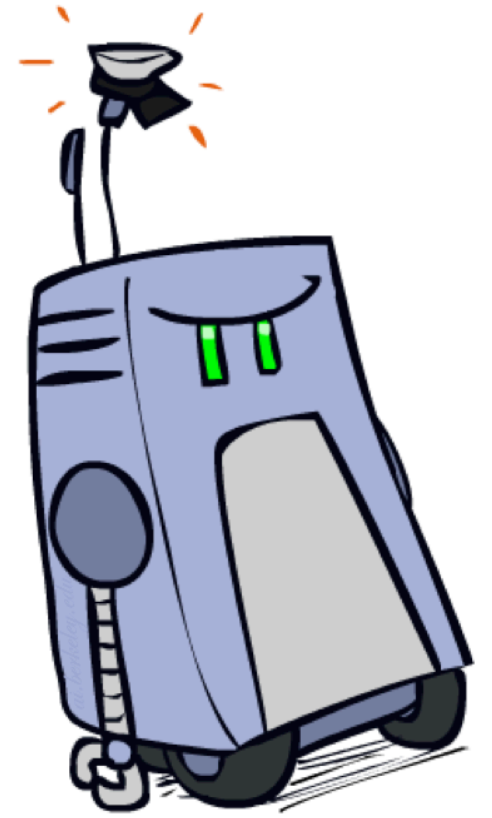
$$P(\text{on} | C = 3) = 0.8$$

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

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

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

2 wins!!



Example: overfitting

- Posteriors determined by relative probabilities (odds ratios):

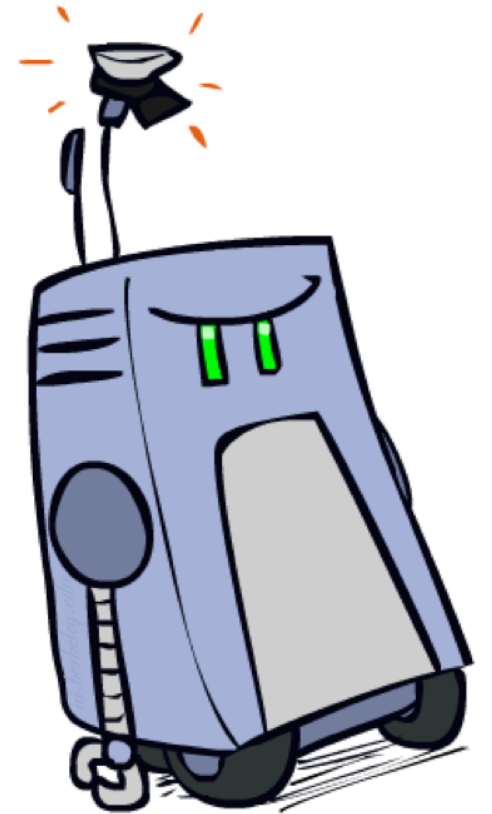
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

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

- 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

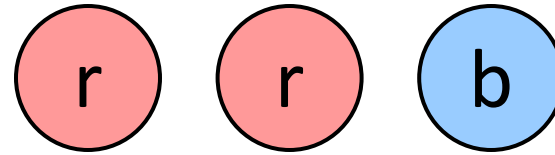
Laplace smoothing

Laplace's estimate:

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

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

- Can derive this estimate with Dirichlet priors



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

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

Laplace smoothing

Laplace's estimate (extended):

- Pretend you saw every outcome k extra times

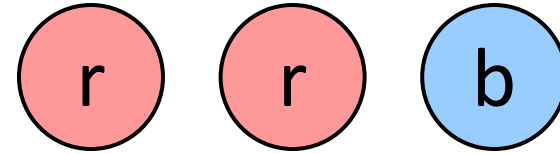
$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

- 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 NB: smoothing

For real classification problems, smoothing is critical

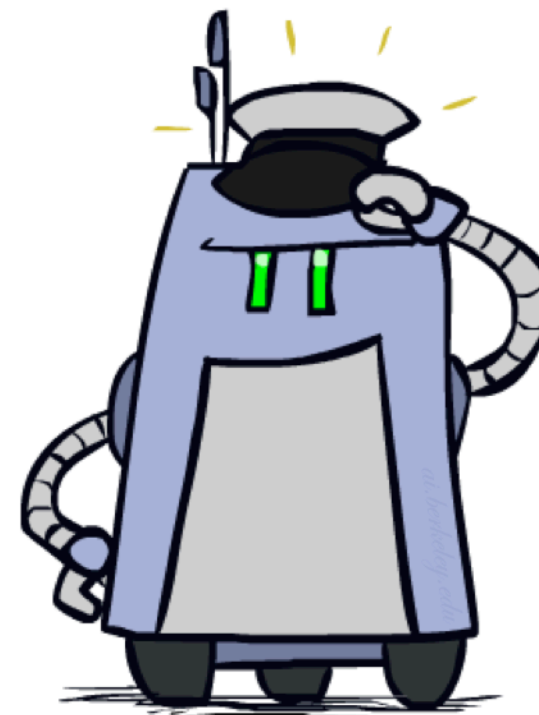
New odds ratios:

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

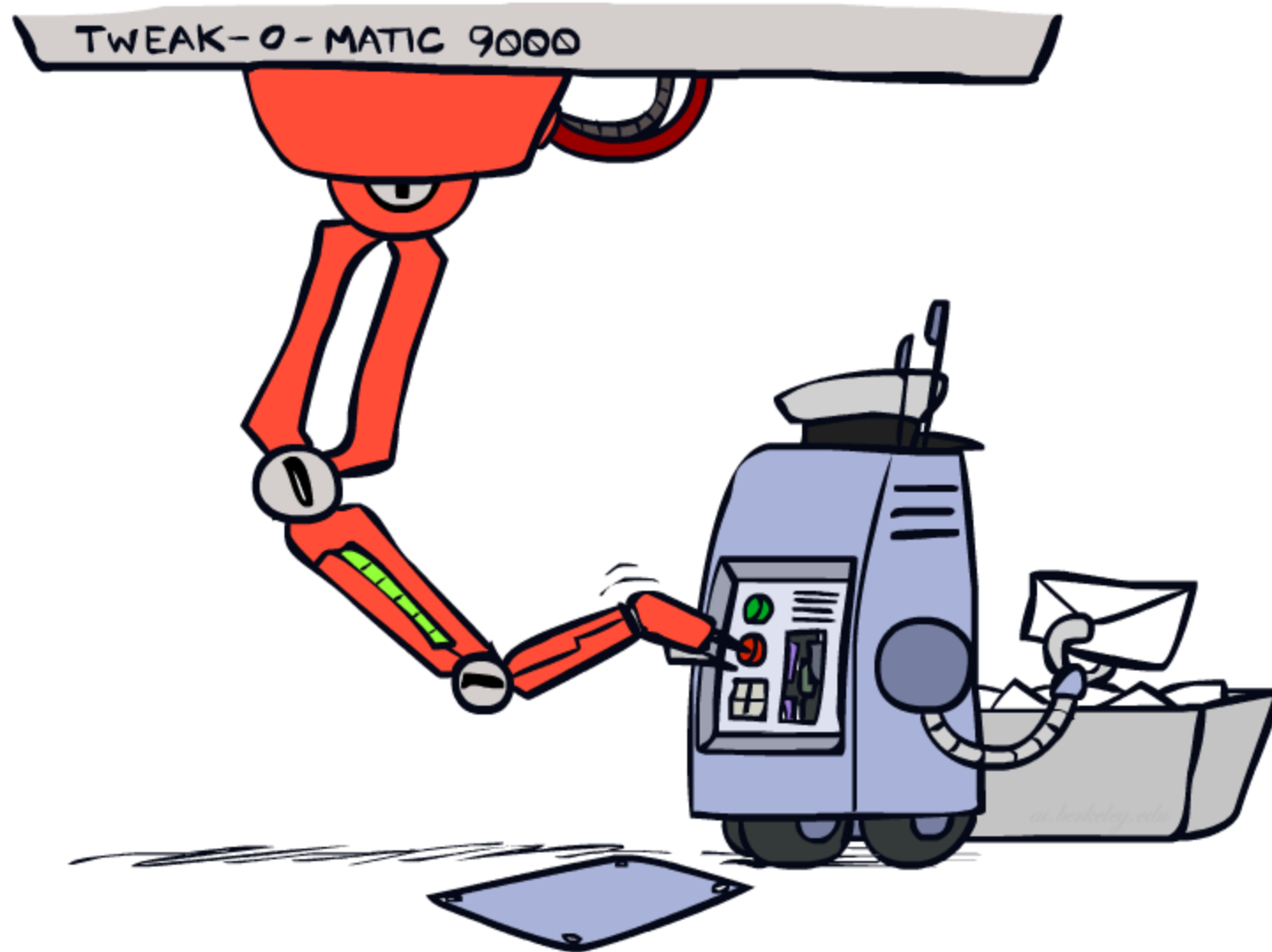
helvetica	:	11.4
seems	:	10.8
group	:	10.2
ago	:	8.4
areas	:	8.3
...		

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

verdana	:	28.8
Credit	:	28.4
ORDER	:	27.2
	:	26.9
money	:	26.5
...		



Tuning



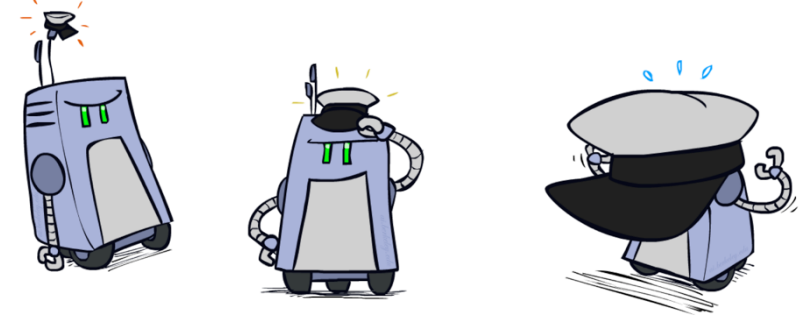
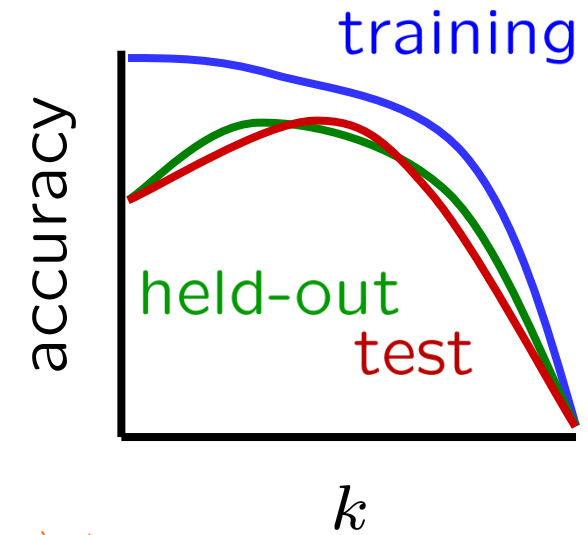
Tuning on held-out data

Now we've got two kinds of unknowns

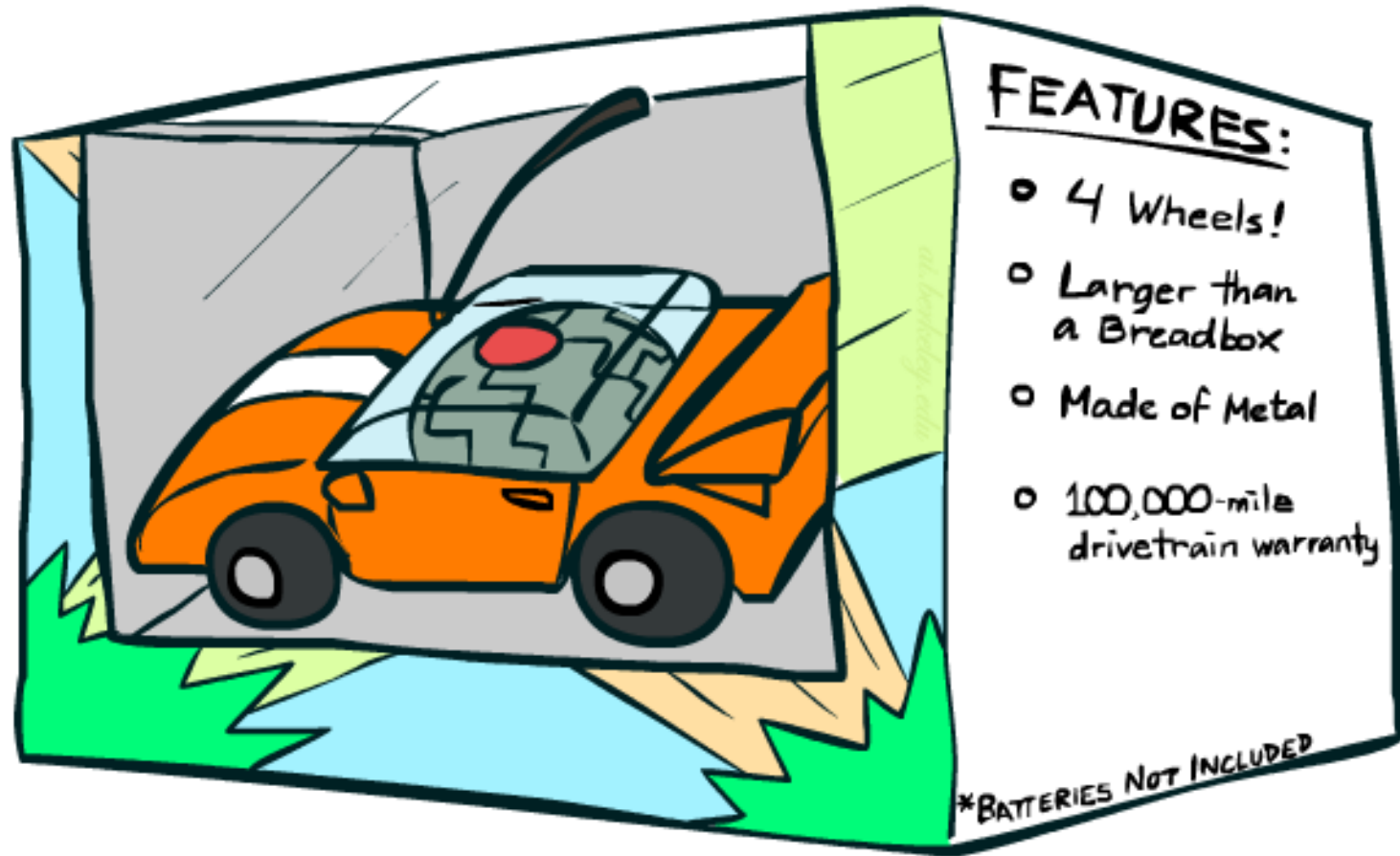
- Parameters: the probabilities $P(X|Y)$, $P(Y)$
- Hyperparameters: e.g. the amount / type of smoothing to do, k , α

What should we learn where?

- Learn parameters from training data
- Tune hyperparameters on different data
 - Why?
- For each value of the hyperparameters, train and test on the held-out data
- Choose the best value and do a final test on the test data



Features



Errors, and what to do

Dear GlobalSCAPE Customer,

GlobalSCAPE has partnered with ScanSoft to offer you the latest version of OmniPage Pro, for just \$99.99* - the regular list price is \$499! The most common question we've received about this offer is - Is this genuine? We would like to assure you that this offer is authorized by ScanSoft, is genuine and valid. You can get the . . .

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

<http://www.amazon.com/apparel>

and see the prominent link for the \$30 offer. All details are there. We hope you enjoyed receiving this message. However, if you'd rather not receive future e-mails announcing new store launches, please click . . .

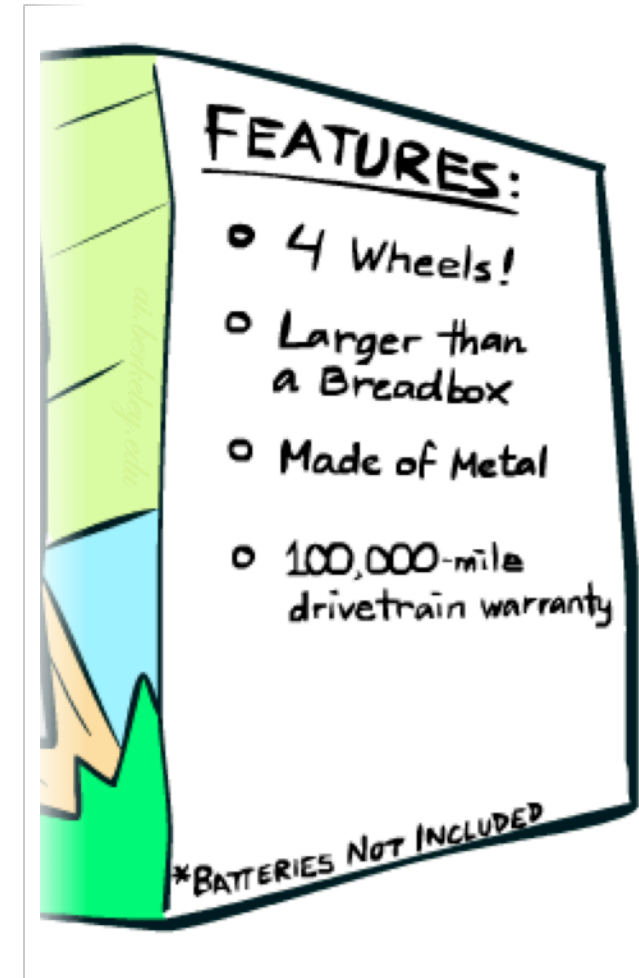
What to do about errors?

Need more features– words aren't enough!

- Have you emailed the sender before?
- Have 1K other people just gotten the same email?
- Is the sending information consistent?
- Is the email in ALL CAPS?
- Do inline URLs point where they say they point?
- Does the email address you by (your) name?

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

Next class we'll talk about classifiers that 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

Confidences from a classifier

The **confidence** of a probabilistic classifier:

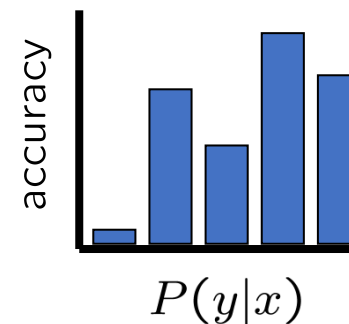
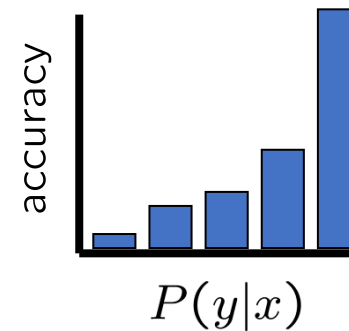
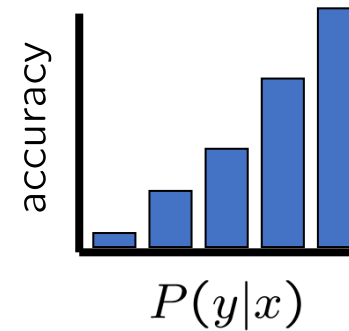
- Posterior over the top label

$$\text{confidence}(x) = \max_y P(y|x)$$

- Represents how sure the classifier is of the classification
- Any probabilistic model will have confidences
- No guarantee confidence is correct

Calibration

- Weak calibration: higher confidences mean higher accuracy
- Strong calibration: confidence predicts accuracy rate
- What's the value of calibration?



Summary

- Bayes rule lets us do diagnostic queries with causal probabilities
- The naïve Bayes assumption takes all features to be independent given the class label
- We can build classifiers out of a naïve Bayes model using training data
- Smoothing estimates is important in real systems
- Classifier confidences are useful, when you can get them

Coming full circle back to sampling...

We *could* take a Bayesian view of Naïve Bayes and estimate parameters using Gibbs sampling rather than MLE

- If interested, see: <http://www.cs.umd.edu/~hardisty/papers/gsfu.pdf>

OK, that's all for today!

- Next week: more machine learning!
- Reminders:
 - **Homework 4** due in 1 week.
 - You should be thinking seriously about projects! (Maybe today's lecture inspired you???)