

CS 188: Artificial Intelligence

Machine Learning II: Naïve Bayes and Bayesian Learning



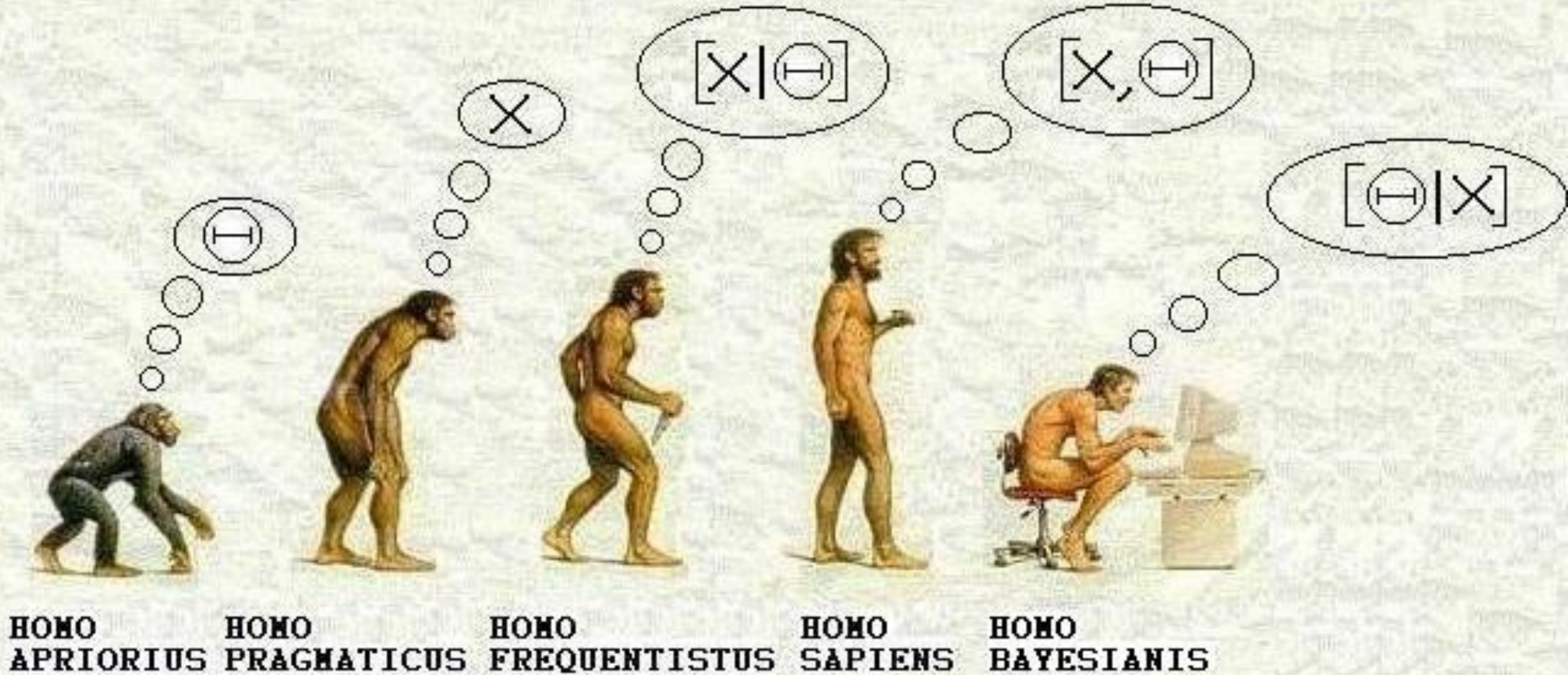
Instructor: Dan Klein and Stuart Russell

University of California, Berkeley

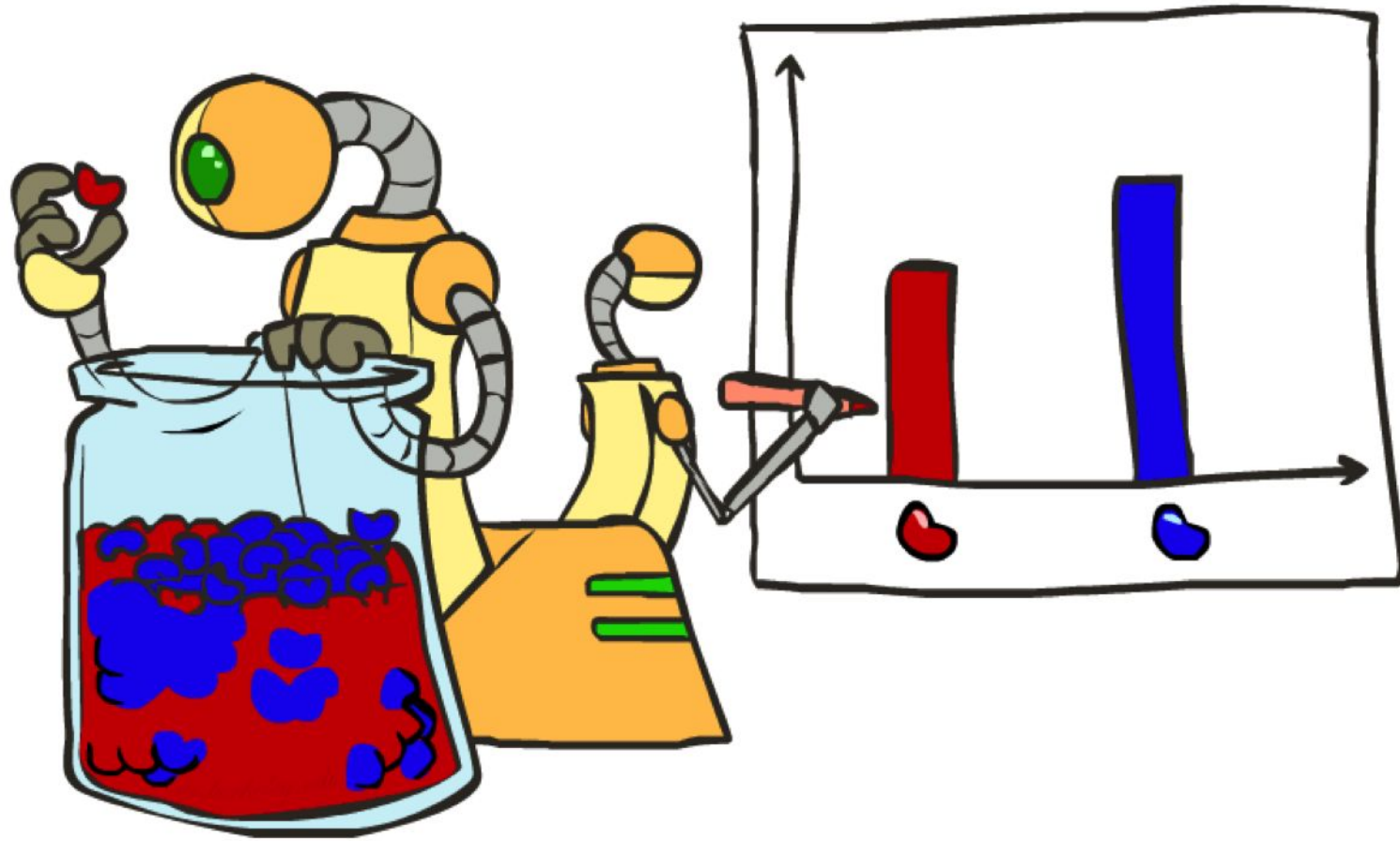
Statistical approaches to machine learning

- Recall the basic idea of supervised learning: find good hypotheses h in a hypothesis space H given i.i.d. data $D = (x_1, y_1), \dots, (x_N, y_N)$
- Consider *probabilistic* hypotheses $P(y | x; h)$, e.g., $Y \sim N(w^T x, \sigma^2)$
- The *likelihood* of the data is $P(D | h) = \prod_j P(y_j | x_j; h)$
 - *Maximum likelihood* hypothesis $h_{ML} = \operatorname{argmax}_h P(D | h)$
- Adding in a prior $P(h)$ to reflect prior knowledge, simplicity preference
 - *Maximum a posteriori* hypothesis $h_{MAP} = \operatorname{argmax}_h P(h | D) = \operatorname{argmax}_h P(D | h) P(h)$
 - $h_{ML} = h_{MAP}$ with a *uniform* prior $P(h)$
- *Bayesian learning*: use the posterior over H rather than picking one h :
 - Bayes predictor $P(y | x, D) = \sum_h P(y | x; h) P(D | h) P(h)$

Bayesian learning

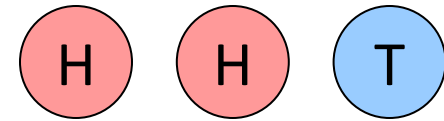


Parameter Estimation

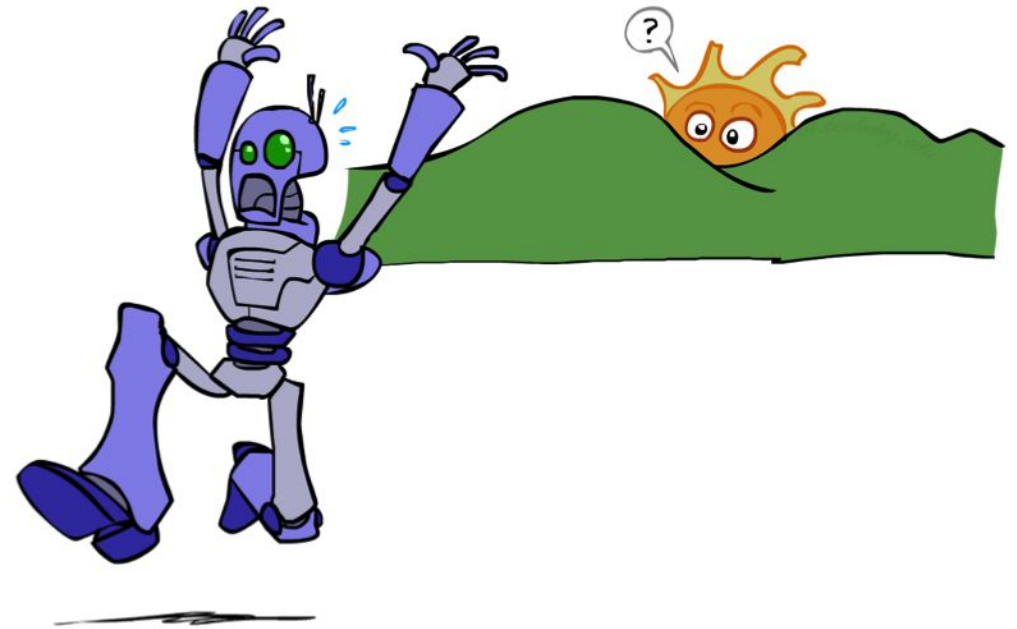
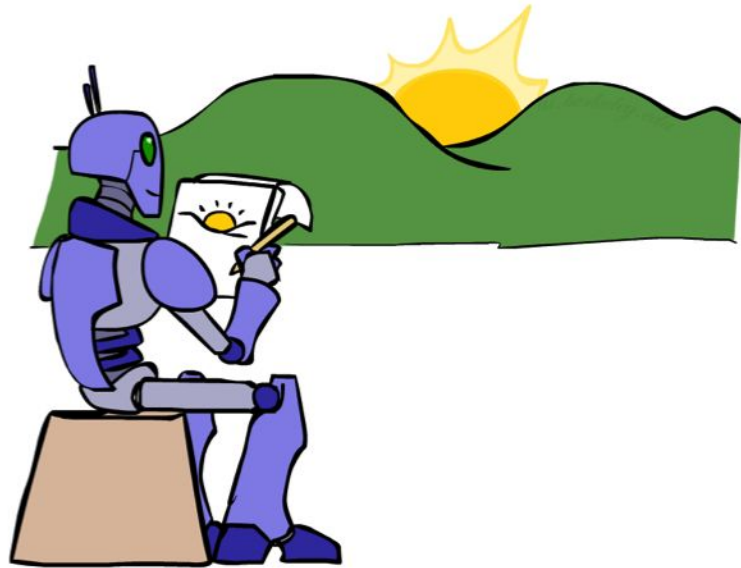


Maximum Likelihood Parameter Estimation

- Estimating the distribution of a random variable
 - E.g., here is a coin; what is the probability θ of heads?
- Evidence $\mathbf{d} = d_1, \dots, d_N$
 - E.g., three independent coin tosses $d_1=\text{heads}$, $d_2=\text{heads}$, $d_3=\text{tails}$
- Likelihood: probability of the evidence $P(d_1, \dots, d_N; \theta)$
 - E.g., $P(d_1=\text{heads}, d_2=\text{heads}, d_3=\text{tails}; \theta) = \theta^2(1-\theta)$
- Maximum likelihood: What value θ_{ML} maximizes the likelihood?
- Log likelihood: $L(\mathbf{d}; \theta) = \log P(\mathbf{d}; \theta)$
 - E.g., $L(\mathbf{d}; \theta) = 2 \log \theta + \log(1-\theta)$
- θ_{ML} also maximizes the log likelihood (why?) and it's easier to differentiate (why?)
- $\partial L / \partial \theta = 2/\theta - 1/(1-\theta) = 0$
- $\theta_{ML} = 2/3$
- For h heads and t tails, $\theta_{ML} = h/(h+t)$

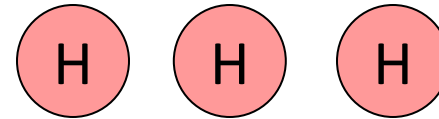


Unseen Events

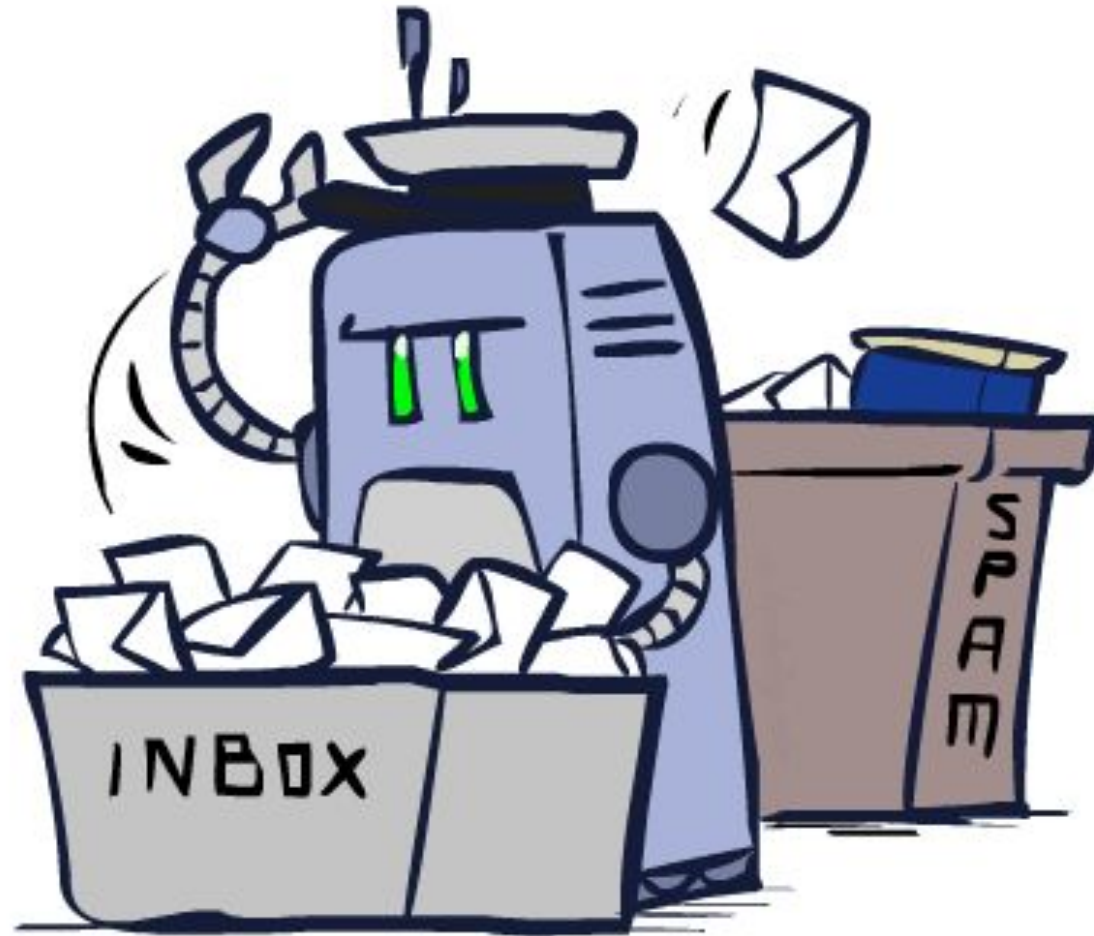


Laplace Smoothing

- Suppose we see three heads: is a $\theta_{ML} = 0$ a reasonable estimate?
- Laplace smoothing with strength α :
 - Pretend you saw every outcome α times before starting
 - $\theta_{Lap} = (h+\alpha)/[(h+\alpha) + (t+\alpha)]$
 - $= (3+\alpha)/(3+2\alpha)$
 - In general, for a K-valued variable:
 - $\theta_k = (N_k+\alpha) / \sum_k (N_k+\alpha) = (N_k+\alpha) / (N + K\alpha)$
 - For $\alpha \gg N$, θ_k tends to $1/K$ (uniform prior)
 - For $\alpha \ll N$, θ_k tends to N_k/N (ML estimate)



Probabilistic Classification: Naïve Bayes



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.

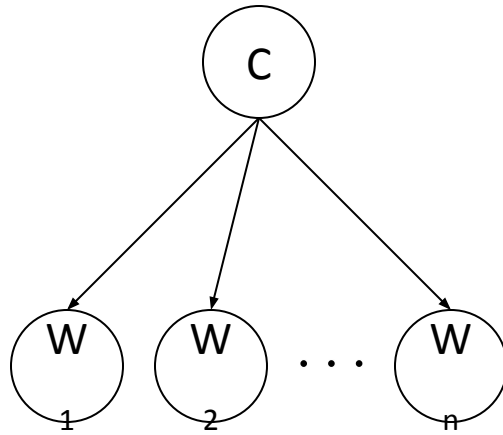
99 MILLION EMAIL ADDRESSES FOR ONLY \$99



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.

Bayes net model for ham/spam

- Class C of a document is spam or ham, with prior $P(C)$
- **Bag-of-words** model: Each word or feature W_i is generated independently from a class-specific distribution $P(W_i | C)$ over words
- This is an example of a **naïve Bayes** model



$$P(C, W_1, \dots, W_n) = P(C) \prod_i P(W_i | C)$$

Inference for Naïve Bayes

- A Naïve Bayes model is a polytree, so solvable in linear time
- To compute posterior distribution for class C given a document:
- $P(c \mid w_1, \dots, w_n) = \alpha P(c, w_1, \dots, w_n) = \alpha P(c) \prod_i P(w_i \mid c)$
- I.e., multiply $n+1$ numbers, for each value of C , then normalize

Computing the class probabilities

	P(w spam)	P(w ham)	Cum LogSpam	Cum LogHam
Word	0.33333	0.66666	-1.1	-0.4

$$\alpha[e^{-76.0}, e^{-80.5}] = [0.989, 0.011]$$

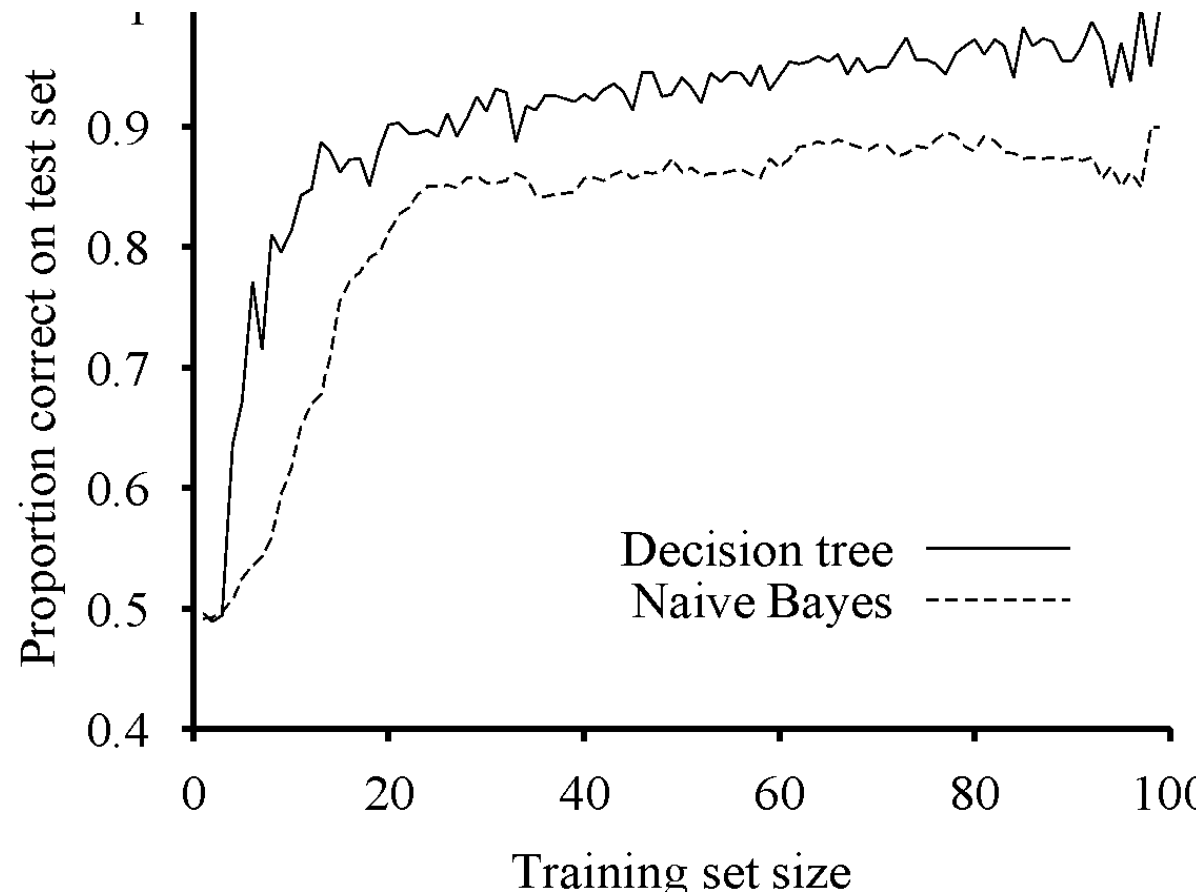
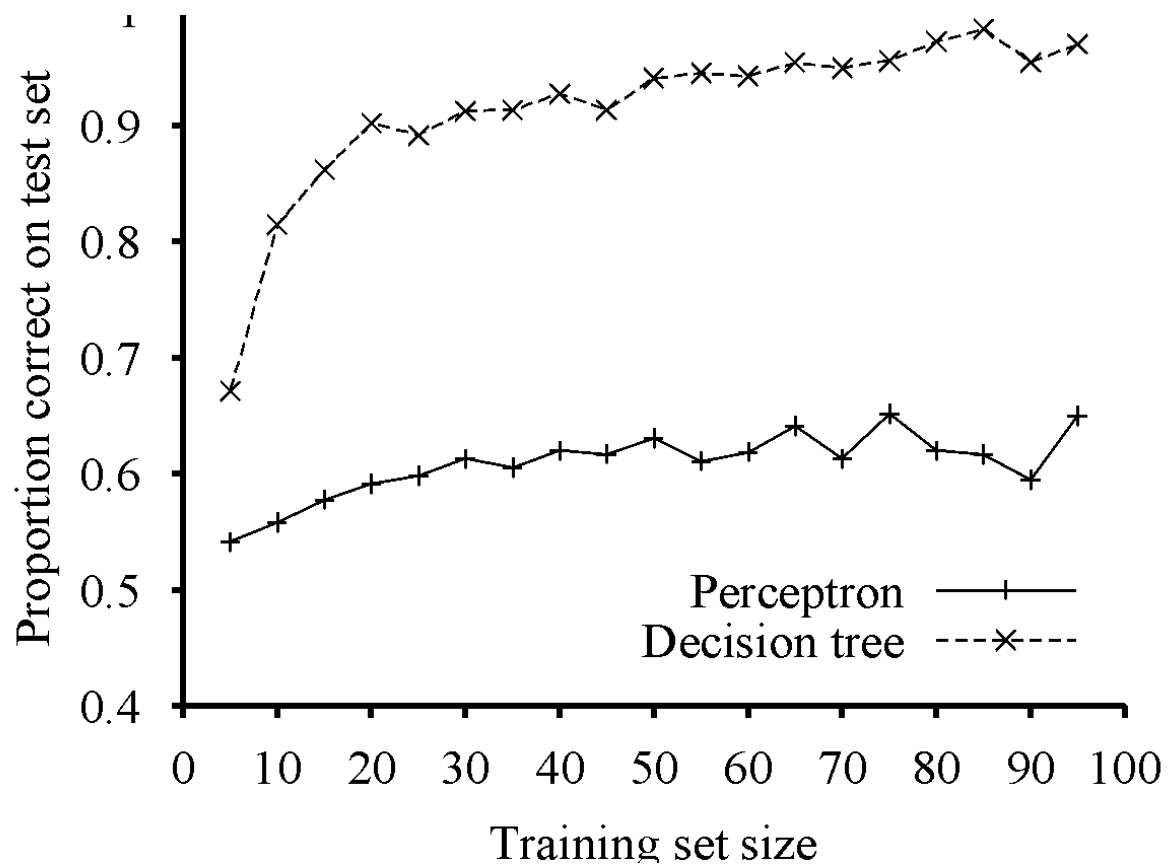
Parameter learning for Naïve Bayes

- We need to estimate the following parameters:
 - $P(C) = [\theta_c, 1-\theta_c]$, the prior over classes
 - ML estimate: relative frequencies in training set
 - $P(W_i | C)$, the distribution for each word/feature given the class
 - Parameters are $\theta_{k|c} = P(W_i=k | C=c)$ for each class c and each dictionary entry k
 - E.g., $\theta_{\text{„you”}|spam} = 0.00881$ $\theta_{\text{„you”}|ham} = 0.00304$
 - Estimated by measuring frequency of occurrence in ham and spam
 - Need Laplace smoothing! Many words may not appear in training set

Naïve Bayes as a linear separator

- For simplicity, consider Boolean class variable ($C=0$ or 1) and Boolean input features x_i
- Suppose we pick the most likely class for each document
 - Classes are separated by boundary $P(C=1 \mid x_1, \dots, x_n) = P(C=0 \mid x_1, \dots, x_n)$
 - I.e., $\alpha P(C=1) \prod_i P(x_i \mid C=1) = \alpha P(C=0) \prod_i P(x_i \mid C=0)$
- Taking logs and mucking around a bit, we get an expression of the form $0 = w_0 + \sum_i w_i x_i$ where the weights depend only on the parameters θ_c and $\theta_{i|c}$
- Unlike neural nets, naïve Bayes training takes linear time, always converges, handles missing data and noise well

Performance on restaurant data

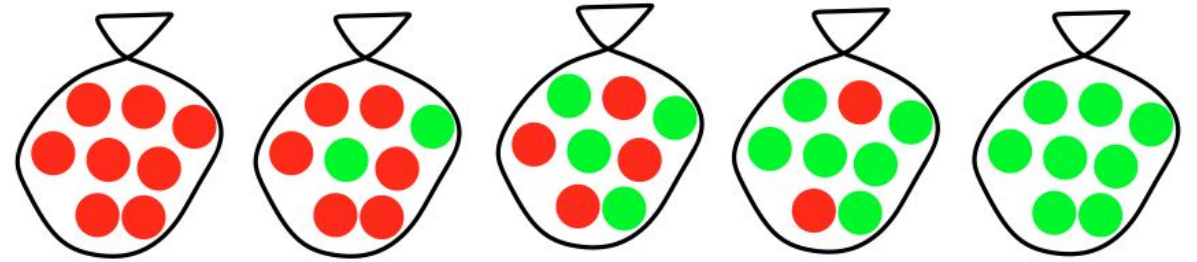


Bayesian learning

- Learning = Bayesian updating of a probability distribution over H
- Given the data so far, each hypothesis has a posterior probability:
 - $P(h_k | D) = \alpha P(D | h_k)P(h_k) = \alpha \times \text{Likelihood} \times \text{Prior}$
- Predictions use a likelihood-weighted average over the hypotheses:
 - $P(d_{N+1} | D) = \sum_k P(d_{N+1} | D, h_k)P(h_k | D) = \alpha \sum_k P(d_{N+1} | h_k) P(D | h_k)P(h_k)$
- Bayesian learning gives optimal predictions!
 - Drawback: \sum_k may be expensive/impossible for large/infinite H
 - MCMC and related methods sparked a huge revival in Bayesian learning

Example: Surprise Candy Co.

- Suppose there are five kinds of bags of candies, no labels!!
 - 10% are h_1 : 100% cherry candies
 - 20% are h_2 : 75% cherry candies + 25% lime candies
 - 40% are h_3 : 50% cherry candies + 50% lime candies
 - 20% are h_4 : 25% cherry candies + 75% lime candies
 - 10% are h_5 : 100% lime candies



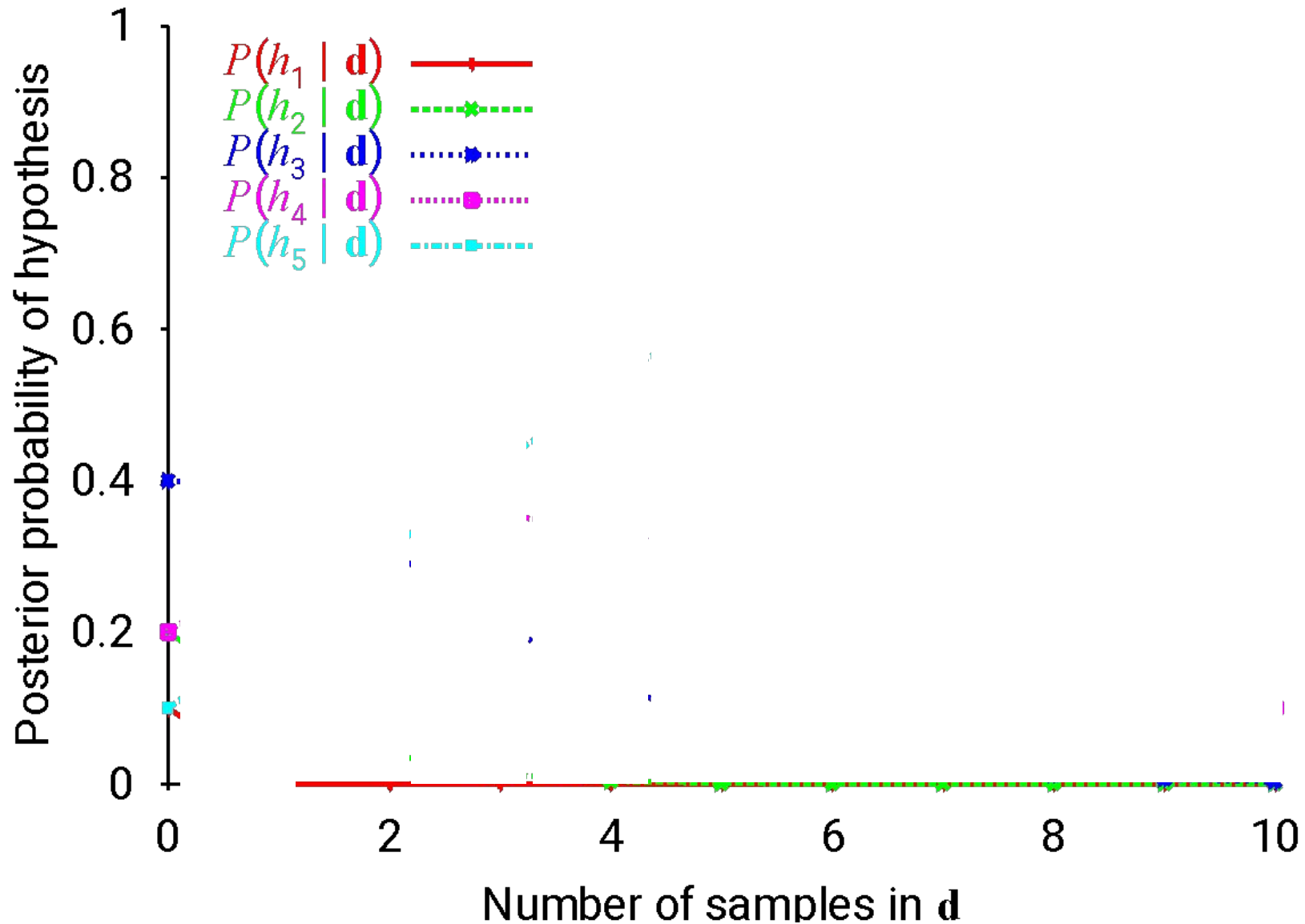
- Then we observe candies drawn from some bag:
- What kind of bag is it?
- What flavour will the next candy be?



Posterior probability of hypotheses

- $P(h_k | D) = \alpha P(D | h_k) P(h_k)$
 - $P(h_1 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_1) P(h_1) = \alpha \cdot 0.0^5 \cdot 0.1 = 0$
 - $P(h_2 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_2) P(h_2) = \alpha \cdot 0.25^5 \cdot 0.2 = 0.000195 \alpha$
 - $P(h_3 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_3) P(h_3) = \alpha \cdot 0.5^5 \cdot 0.4 = 0.0125 \alpha$
 - $P(h_4 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_4) P(h_4) = \alpha \cdot 0.75^5 \cdot 0.2 = 0.0475 \alpha$
 - $P(h_5 | 5 \text{ limes}) = \alpha P(5 \text{ limes} | h_5) P(h_5) = \alpha \cdot 1.0^5 \cdot 0.1 = 0.1 \alpha$
- $\alpha = 1 / (0 + 0.000195 + 0.0125 + 0.0475 + 0.1) = 6.2424$
- $P(h_1 | 5 \text{ limes}) = 0$
- $P(h_2 | 5 \text{ limes}) = 0.00122$
- $P(h_3 | 5 \text{ limes}) = 0.07803$
- $P(h_4 | 5 \text{ limes}) = 0.29650$
- $P(h_5 | 5 \text{ limes}) = 0.62424$

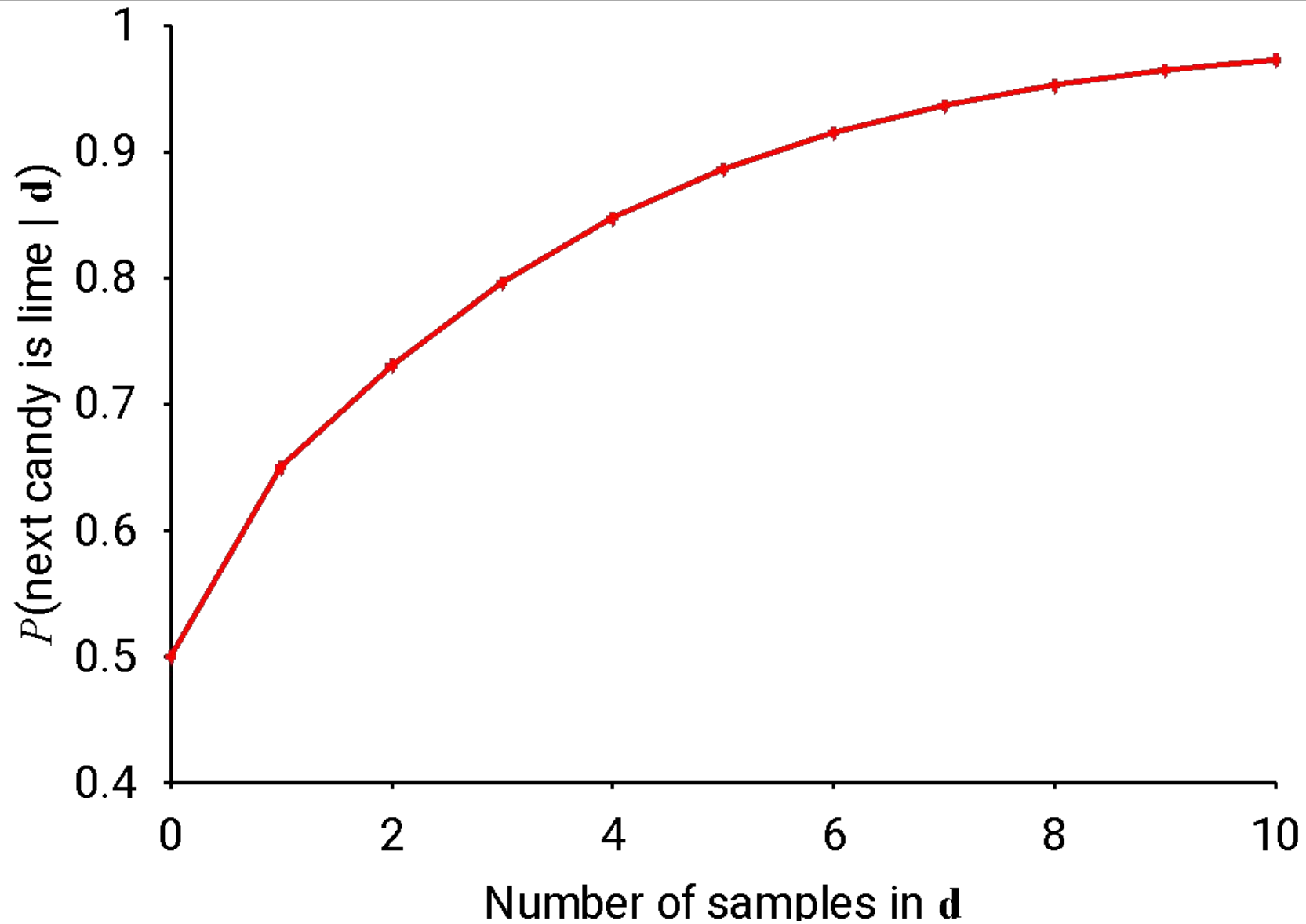
Posterior probability of hypotheses



Prediction probability

- $P(d_{N+1} | D) = \sum_k P(d_{N+1} | h_k)P(h_k | D)$
- $P(\text{lime on 6} | 5 \text{ limes})$
 - $= P(\text{lime on 6} | h_1)P(h_1 | 5 \text{ limes})$
 - $+ P(\text{lime on 6} | h_2)P(h_2 | 5 \text{ limes})$
 - $+ P(\text{lime on 6} | h_3)P(h_3 | 5 \text{ limes})$
 - $+ P(\text{lime on 6} | h_4)P(h_4 | 5 \text{ limes})$
 - $+ P(\text{lime on 6} | h_5)P(h_5 | 5 \text{ limes})$
 - $= 0 \times 0 + 0.25 \times 0.00122 + 0.5 \times 0.07830 + 0.75 \times 0.29650 + 1.0 \times 0.62424$
 - $= 0.88607$

Prediction probability



The “poverty of the stimulus” argument

- Children learn the grammar of language quite quickly
- A grammar is a Boolean hypothesis about what sentences are OK
- Children receive only positive examples of sentences
- If all possible grammars are allowed, then the simplest grammar consistent with the data is $S \rightarrow \text{word}^*$, i.e., all possible strings
- This is not what children learn, therefore they must have strong innate grammatical structures
- These structures must be common to all languages
- Language learning is just filling in the details
- **What's wrong with this argument???**

A simple experiment: The numbers game

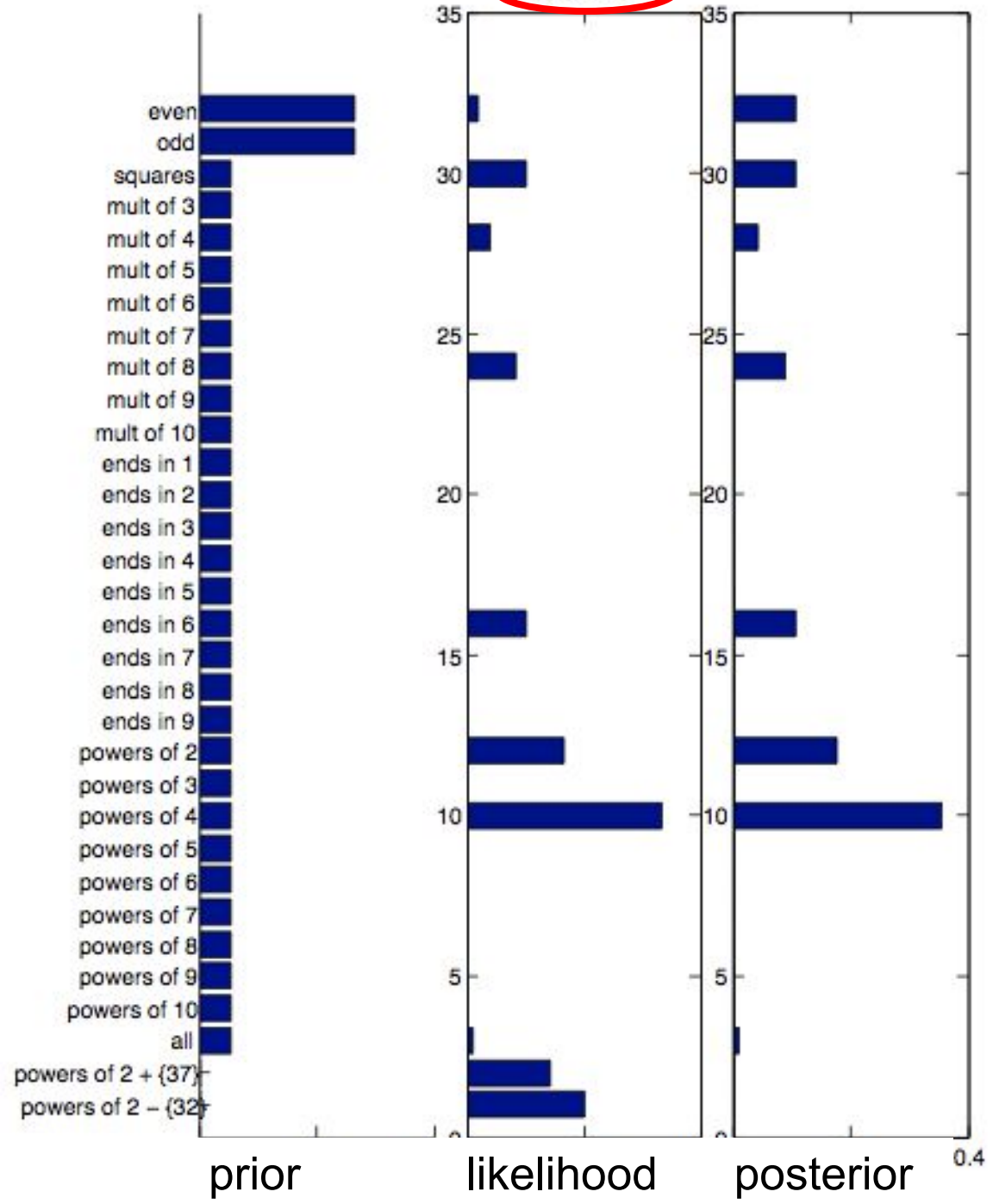
(Example from Tenenbaum via Murphy, Ch.3)

- Given examples of some unknown class, a predefined subset of $\{1, \dots, 100\}$, output a hypothesis as to what the class is
- E.g., $\{16, 8, 2, 64\}$
- This is a Boolean classification problem; simplest consistent solution is “everything.”
- What do you think the class is?
- Why?

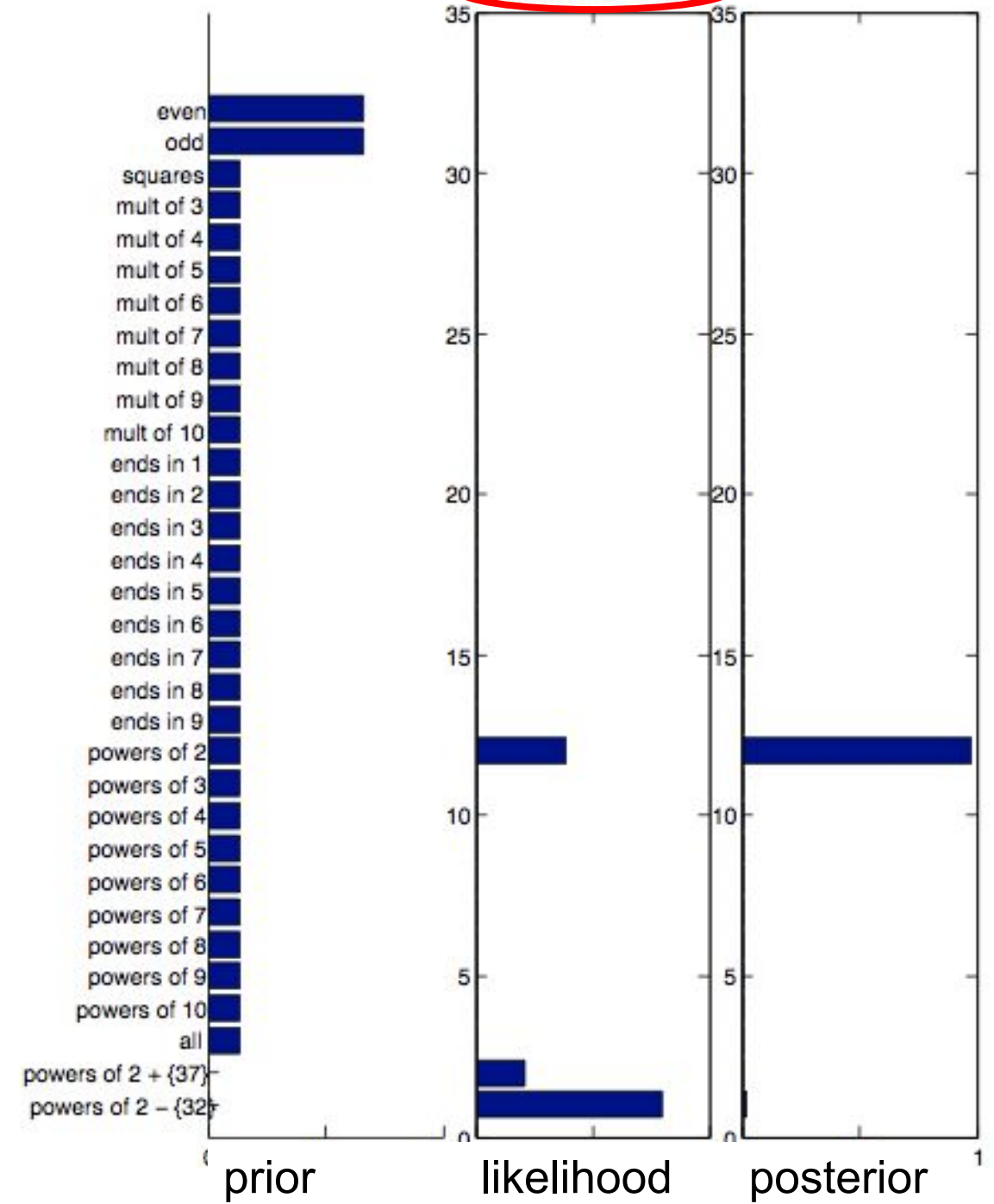
A Bayesian analysis

- Assuming numbers are sampled uniformly from the class, we can compute the data likelihoods:
 - $P(\{16, 8, 2, 64\} \mid \text{powers of } 2) = (1/7)^4 \approx 4.2 \times 10^{-4}$
 - $P(\{16, 8, 2, 64\} \mid \text{everything}) = (1/100)^4 = 10^{-8}$
- This difference far outweighs any reasonable simplicity-based prior
- Another way to put it: if “everything” was the right hypothesis, we’d see all possible strings of words, but we don’t

data = 16



data = 16 8 2 64

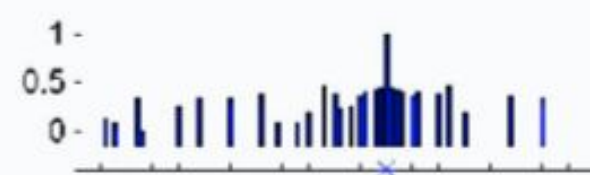
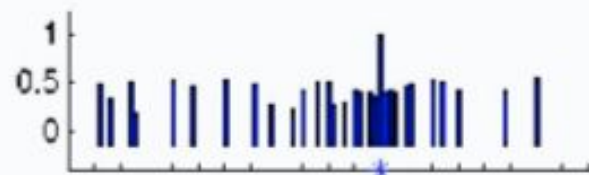


+ Examples

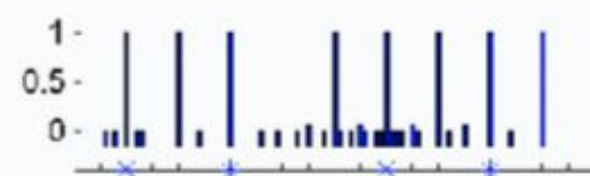
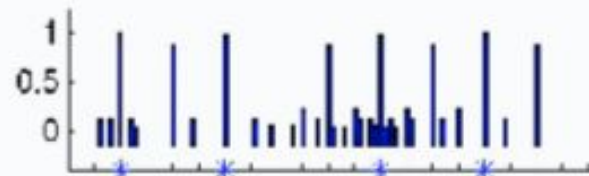
Human generalization

Bayesian Model

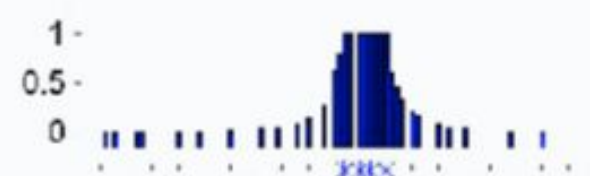
60



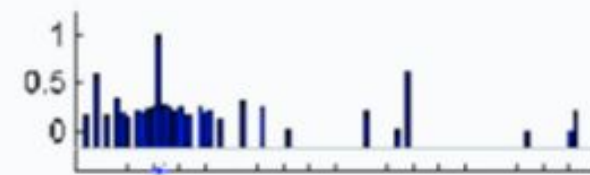
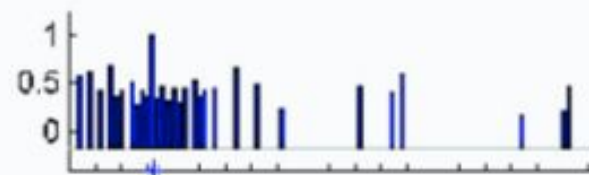
60 80 10 30



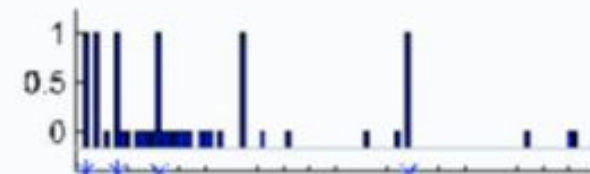
60 52 57 55



16



16 8 2 64



16 23 19 20



Summary

- Statistical learning replaces an arbitrary loss function (e.g., squared error) with a general approach based on probability
 - Maximum likelihood: choose h to maximize probability of data
 - Maximum a posteriori: choose h that is most likely given data
 - Bayesian: update probability over H given data
- For parameters of a simple discrete distribution, ML = empirical frequency
 - Smoothing avoids some problems with unseen events
- Naïve Bayes classifiers are simple Bayes nets that work well for many tasks
- Bayesian learning is the most general approach, but may be intractable