Generative Models for Discrete Data

ddebarr@uw.edu

2016-04-21

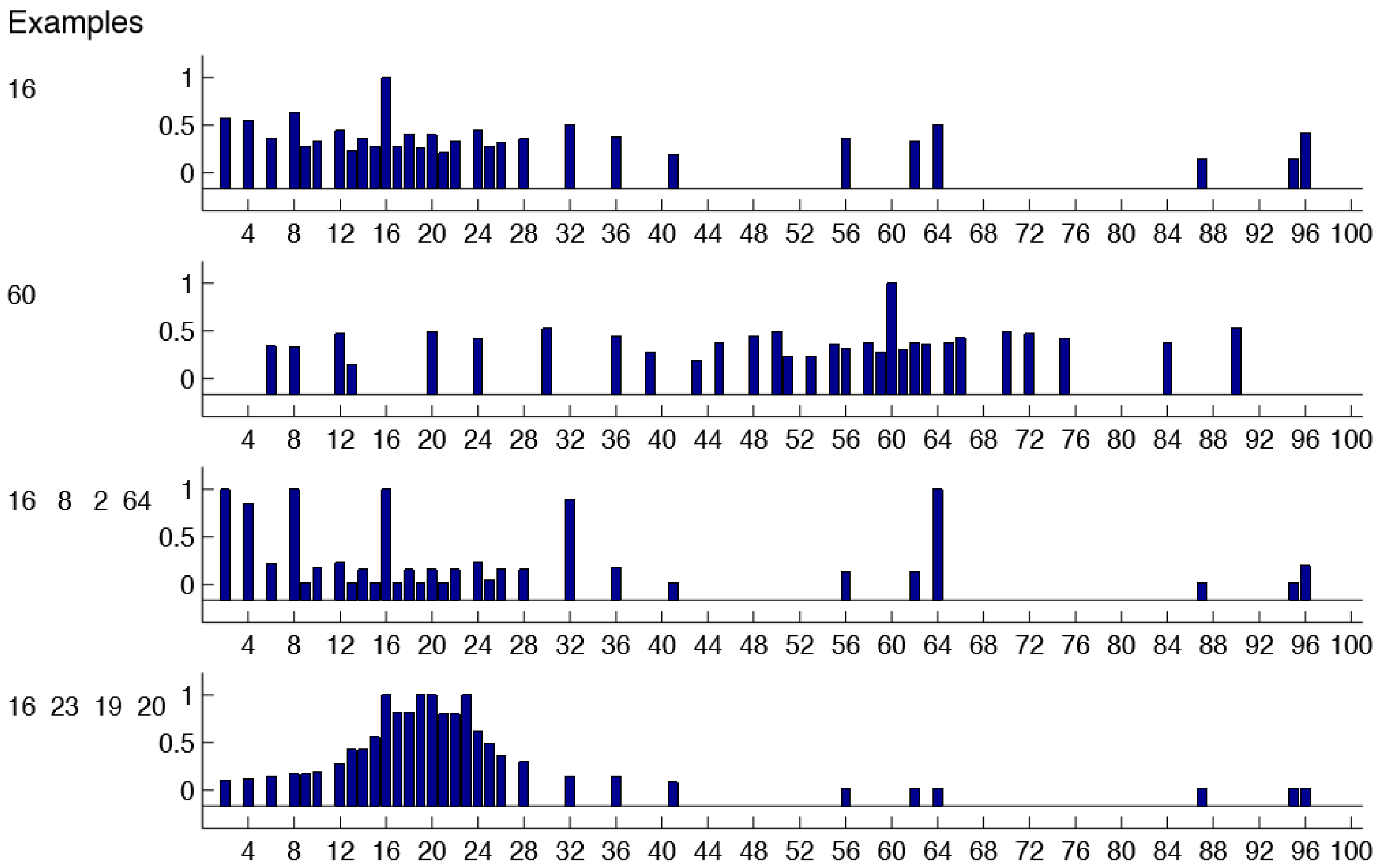
Agenda

- Bayesian Concept Learning
- Beta-Binomial Model
- Dirichlet-Multinomial Model
- Naïve Bayes Classifiers

$$p(y = c|\mathbf{x}, \boldsymbol{\theta}) \propto p(\mathbf{x}|y = c, \boldsymbol{\theta})p(y = c|\boldsymbol{\theta})$$

Numbers Game

Given positive class examples predict membership for remaining numbers [8 people]



Likelihood

$$p(\mathcal{D}|h) = \left[\frac{1}{\text{size}(h)}\right]^N = \left[\frac{1}{|h|}\right]^N$$

- Example:
  - After 1 example …

    $p(\mathcal{D}|h_{\text{two}}) = 1/6$ v $p(\mathcal{D}|h_{\text{even}}) = 1/50$
  - After 4 examples …

    $(1/6)^4 = 7.7 \times 10^{-4}$ v $(1/50)^4 = 1.6 \times 10^{-7}$
  - Occam’s razor: simpler is better [likelihood ratio is almost 5000:1]

Estimation: MAP v MLE

- Posterior = Likelihood * Prior / Evidence

- Maximum A Priori (MAP)

$$\hat{h}^{MAP} = \underset{h}{\operatorname{argmax}}\, p(\mathcal{D}|h)p(h) = \underset{h}{\operatorname{argmax}}\, [\log p(\mathcal{D}|h) + \log p(h)]$$

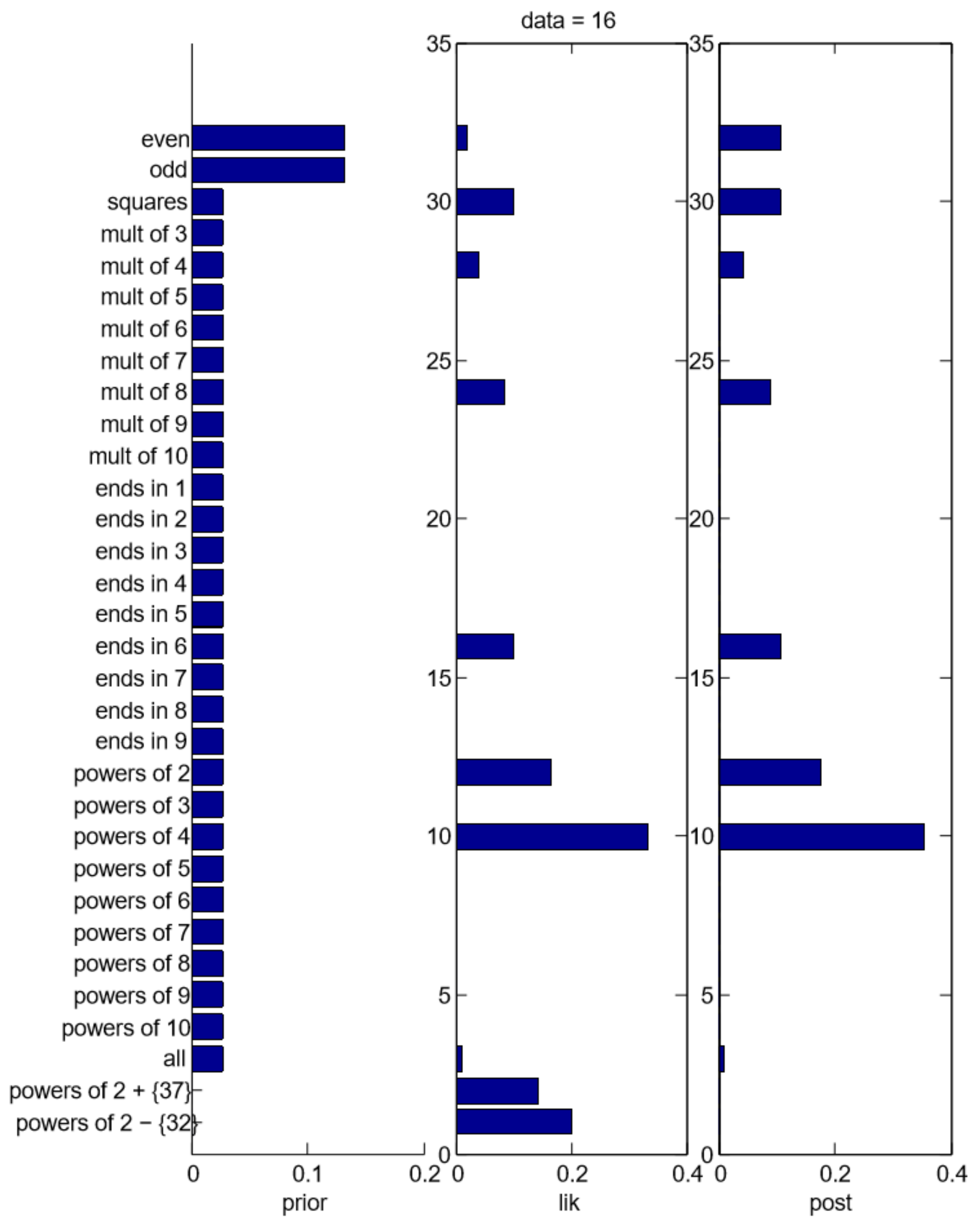
- Maximum Likelihood Estimate (MLE)

$$\hat{h}^{mle} \triangleq \underset{h}{\operatorname{argmax}}\, p(\mathcal{D}|h) = \underset{h}{\operatorname{argmax}}\, \log p(\mathcal{D}|h)$$

Posterior

Posterior probabilities for 32 different concepts after observing 1 value

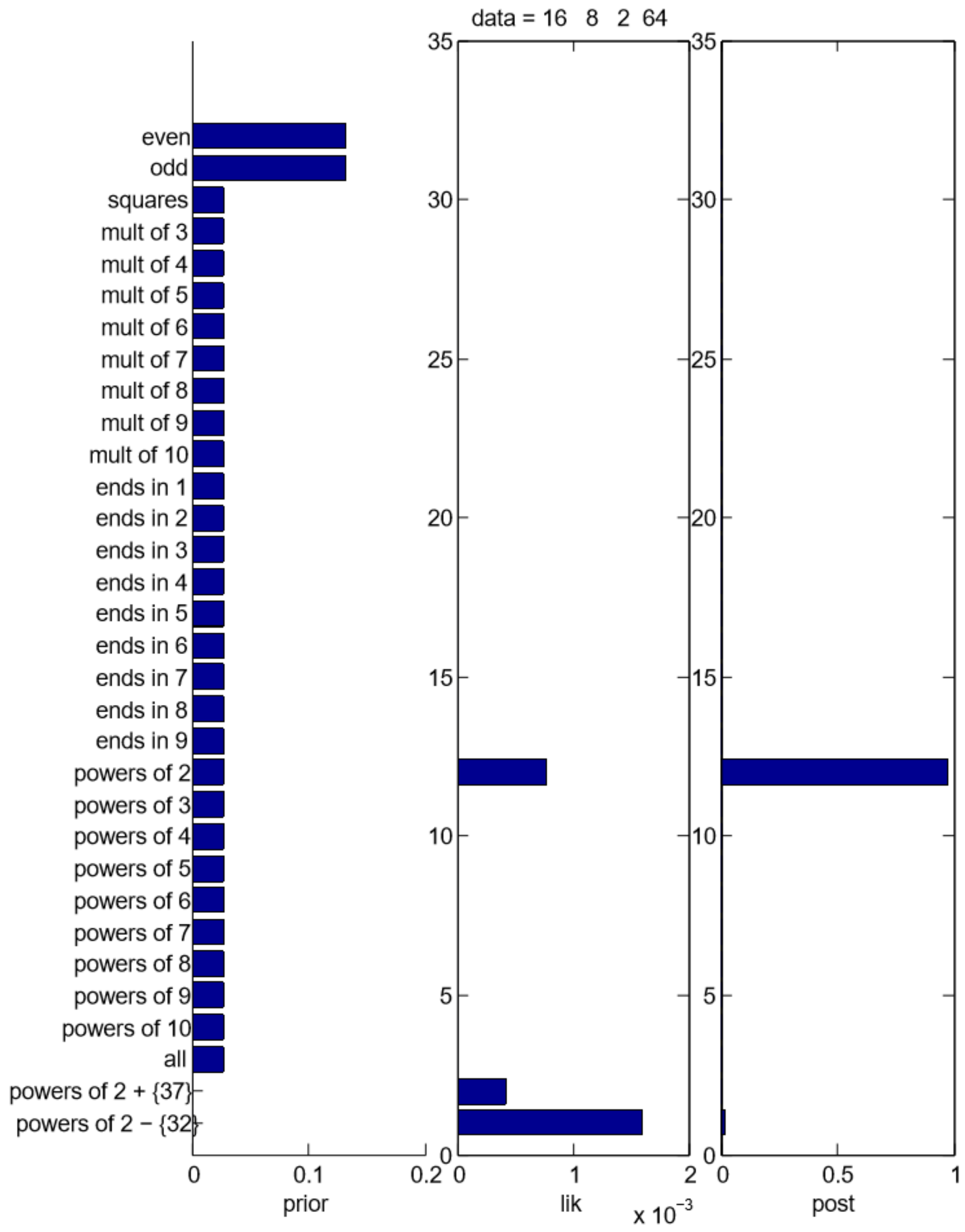
"powers of 4" it is ☺



Posterior

Posterior probabilities for 32 different concepts after observing 4 values

“powers of 2” it is ☺

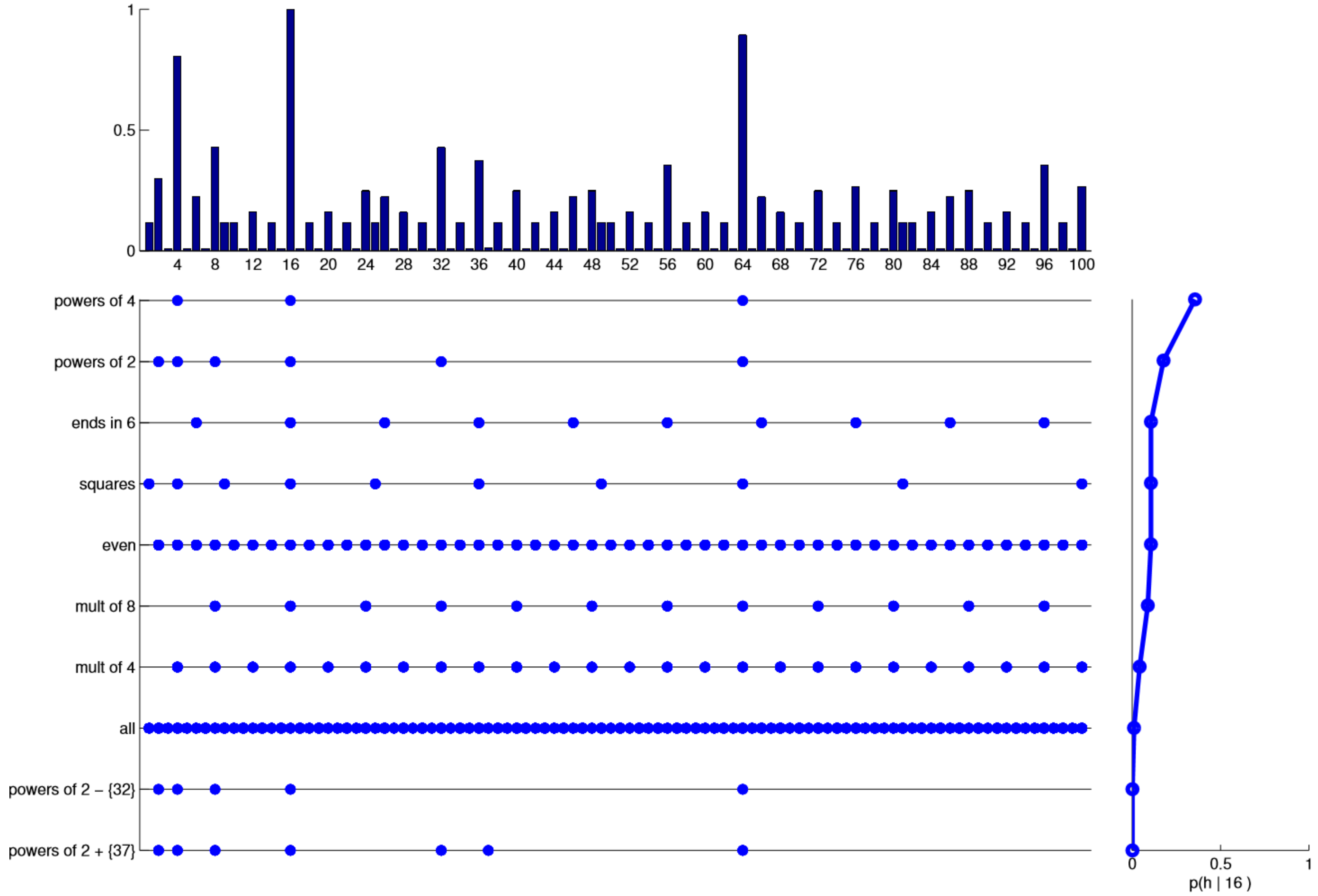


$$p(\tilde{x} \in C|\mathcal{D}) = \sum_h p(y=1|\tilde{x}, h)p(h|\mathcal{D})$$

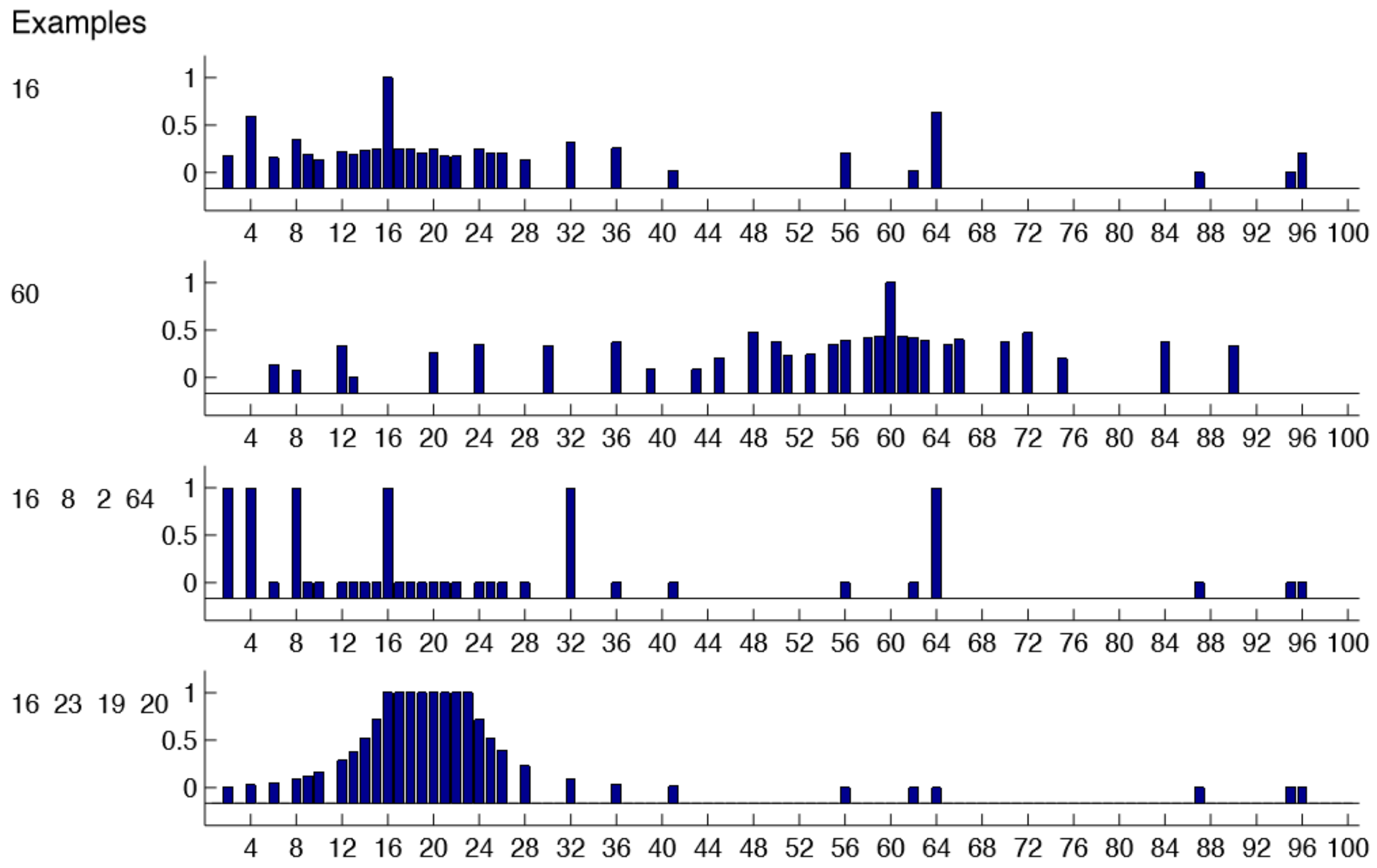
After 1 example …

Bayesian
Model
Averaging
[BMA]

(weighted average)



More Complex Prior (to mimic humans)



Mixture:

$$p(h) = \pi_0 p_{\text{rules}}(h) + (1 - \pi_0) p_{\text{interval}}(h)$$

Beta-Binomial Distribution

- Likelihood

  Same form whether we observe the counts or a sequence of trials

$$p(\mathcal{D}|\theta) = \theta^{N_1}(1-\theta)^{N_0}$$

- Prior

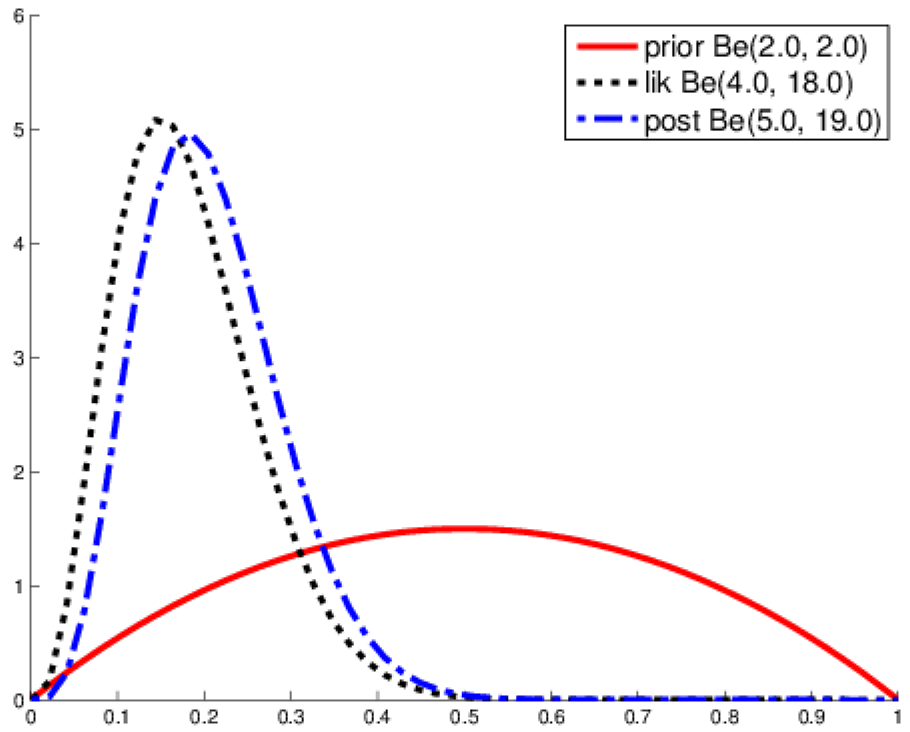
$$p(\theta) \propto \theta^{\gamma_1}(1-\theta)^{\gamma_2}$$

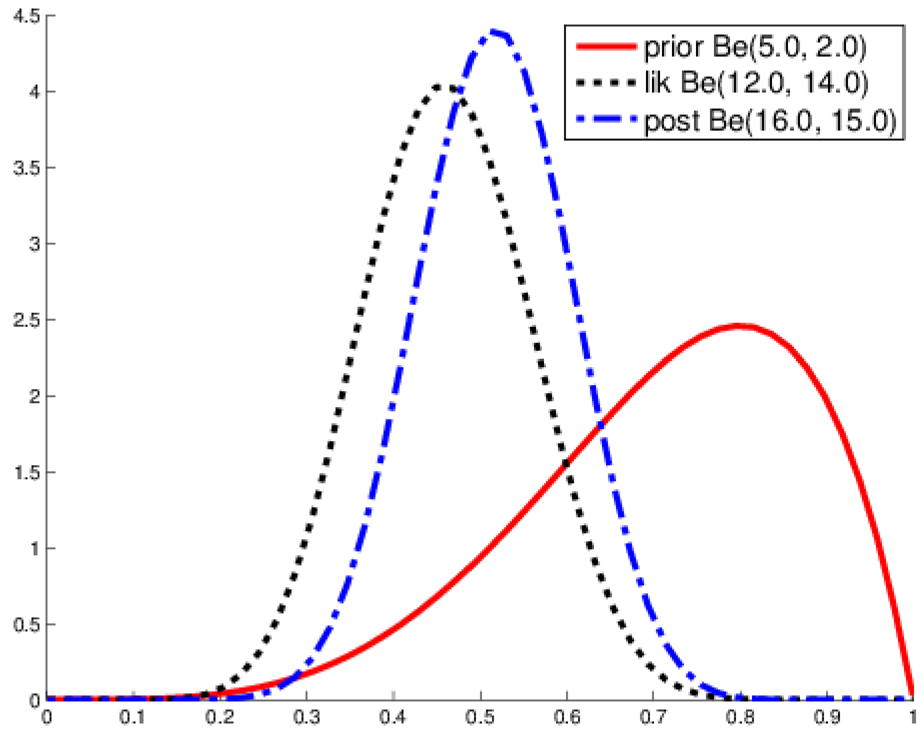
- Posterior

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) = \theta^{N_1}(1-\theta)^{N_0}\theta^{\gamma_1}(1-\theta)^{\gamma_2} = \theta^{N_1+\gamma_1}(1-\theta)^{N_0+\gamma_2}$$

  Conjugate Prior: prior and posterior have the same form

Effect of the Prior





Hyperparameters also known as (aka) pseudo counts: the equivalent sample size of the prior

Overfitting and the Black Swan Problem

- We've never seen a black swan, so it's not possible

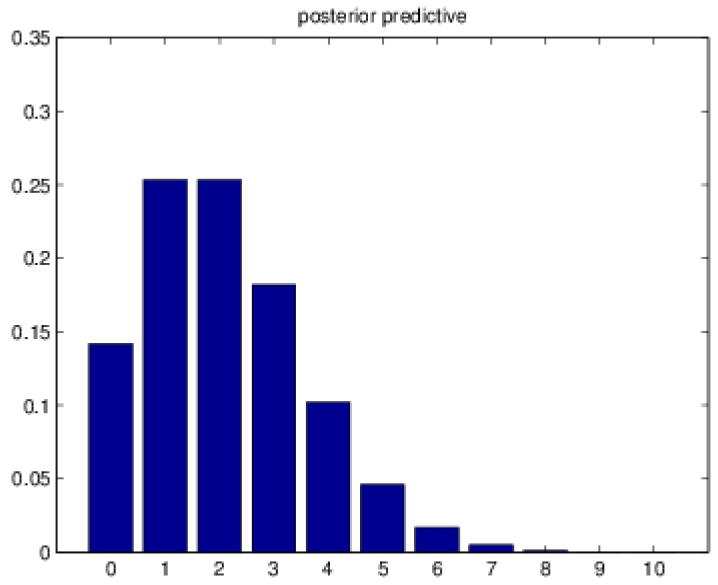
$$\text{MLE is } \hat{\theta} = 0/3 = 0$$

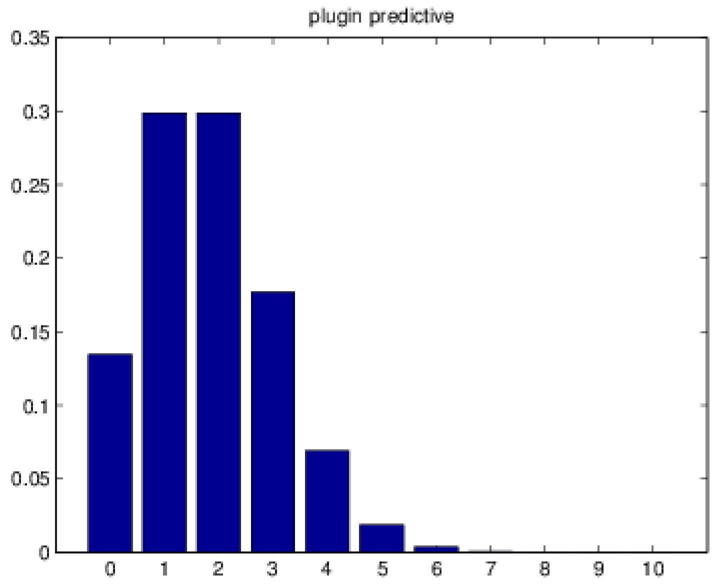
- Laplace's rule of succession [aka add-one smoothing]

$$p(\tilde{x} = 1|\mathcal{D}) = \frac{N_1 + 1}{N_1 + N_0 + 2}$$

Use the Force [Prior]

After observing 3 heads out of 20 coin flips …





Posterior predictive distribution based on prior of Beta(2, 2): heavier tail

Simpler plug-in approximation: exhibiting a bit more certainty

Dirichlet-Multinomial Distribution

- Likelihood

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{K} \theta_k^{N_k}$$

- Prior

$$\mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} \theta_k^{\alpha_k - 1} \mathbb{I}(\mathbf{x} \in S_K)$$

- Posterior

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &\propto p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &\propto \prod_{k=1}^{K} \theta_k^{N_k} \theta_k^{\alpha_k - 1} = \prod_{k=1}^{K} \theta_k^{\alpha_k + N_k - 1} \\ &= \mathrm{Dir}(\boldsymbol{\theta}|\alpha_1 + N_1, \ldots, \alpha_K + N_K) \end{aligned}$$

Bag of Words Representation

```
Mary had a little lamb, little lamb, little lamb,
Mary had a little lamb, its fleece as white as snow
```

```
mary lamb little big fleece white black snow rain unk
1    2    3      4   5      6     7     8    9    10
```

```
1 10 3 2  3 2 3 2
1 10 3 2 10 5 6 8
```

Token	1	2	3	4	5	6	7	8	9	10
Word	mary	lamb	little	big	fleece	white	black	snow	rain	unk
Count	2	4	4	0	1	1	0	1	0	3

Warning: consistency issue: count=3 v count=4 for “unk”

$$p(\tilde{X} = j|D) = E[\theta_j|D] = \frac{\alpha_j + N_j}{\sum_{j'} \alpha_{j'} + N_{j'}} = \frac{1 + N_j}{10 + 17}$$

$$p(\tilde{X} = j|D) = (3/27, 5/27, 5/27, 1/27, 2/27, 2/27, 1/27, 2/27, 1/27, 5/27)$$

Naïve Bayes Classifier (NBC)

- Features assumed to be conditionally independent given the class label

$$p(\mathbf{x}|y = c, \boldsymbol{\theta}) = \prod_{j=1}^{D} p(x_j|y = c, \boldsymbol{\theta}_{jc})$$

Generative model

- Multivariate Bernoulli Naïve Bayes

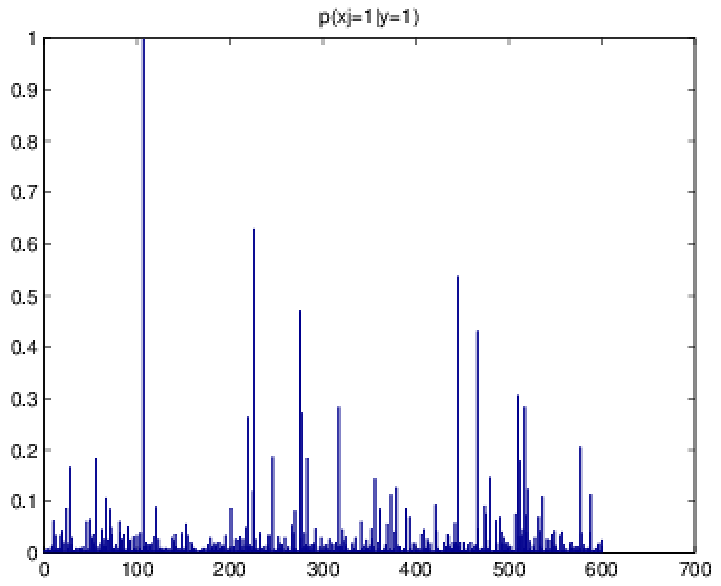
$$\prod_{j=1}^{D} \text{Ber}(x_j|\mu_{jc})$$

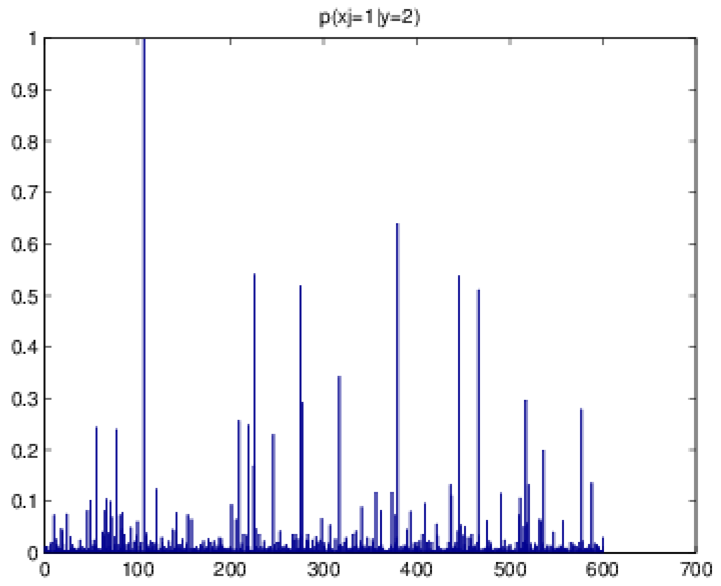
Naïve Bayes Parameter Estimation

Algorithm 3.1: Fitting a naive Bayes classifier to binary features

```
$N_c = 0$, $N_{jc} = 0$;
for $i = 1 : N$ do
    $c = y_i$ // Class label of $i$'th example;
    $N_c := N_c + 1$ ;
    for $j = 1 : D$ do
        if $x_{ij} = 1$ then
            $N_{jc} := N_{jc} + 1$
$\hat{\pi}_c = \frac{N_c}{N}$, $\hat{\theta}_{jc} = \frac{N_{jc}}{N_c}$
```

Generative Model Examples





Same term spikes for both classes: not so useful for classification purposes

Log-Sum-Exp Trick

- Computing the product of a bunch of conditionally independent probabilities can result in numerical underflow [everything looks like zero]; so we compute the sum of log values instead
- If we're just predicting the class, we only need to compute the numerator; but if we need to estimate the probability, we'll need to compute the denominator as well

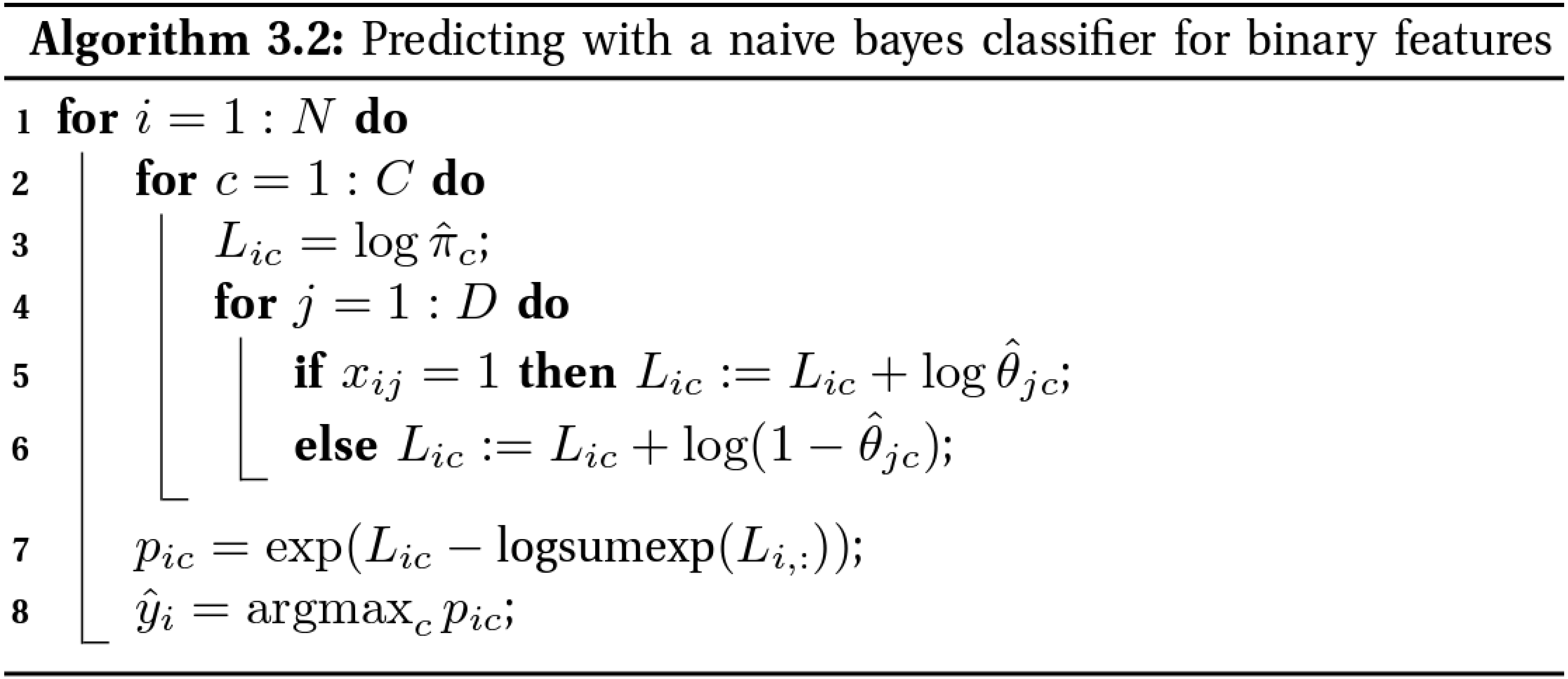
$$p(y=c|\mathbf{x}) = \frac{p(y=c)p(\mathbf{x}|y=c)}{\sum_{c'} p(y=c'|\boldsymbol{\theta})p(\mathbf{x}|y=c')}$$

$$\begin{aligned}\log p(y=c|\mathbf{x}) &= b_c - \log\left[\sum_{c'=1}^{C} e^{b_{c'}}\right] \\ b_c &\triangleq \log p(\mathbf{x}|y=c) + \log p(y=c)\end{aligned}$$

$$\log \sum_c e^{b_c} = \log\left[(\sum_c e^{b_c - B})e^B\right] = \left[\log(\sum_c e^{b_c - B})\right] + B$$

where $B = \max_c b_{c'}$

Naïve Bayes Prediction

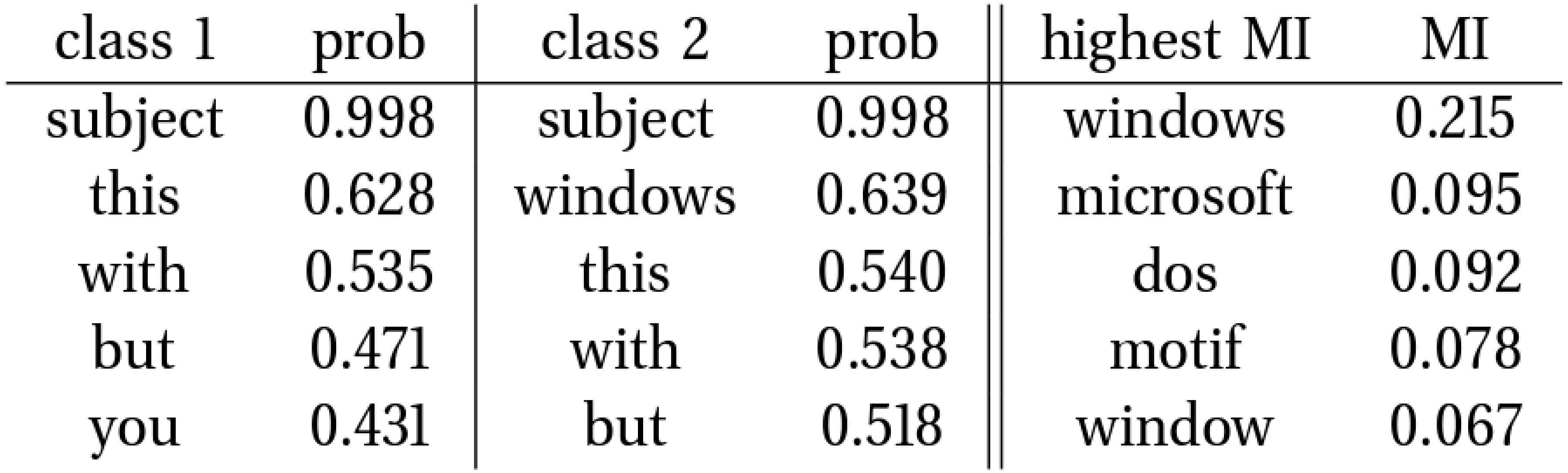


Algorithm 3.2: Predicting with a naive bayes classifier for binary features

```
for $i = 1 : N$ do
    for $c = 1 : C$ do
        $L_{ic} = \log \hat{\pi}_c$;
        for $j = 1 : D$ do
            if $x_{ij} = 1$ then $L_{ic} := L_{ic} + \log \hat{\theta}_{jc}$;
            else $L_{ic} := L_{ic} + \log(1 - \hat{\theta}_{jc})$;
    $p_{ic} = \exp(L_{ic} - \text{logsumexp}(L_{i,:}))$;
    $\hat{y}_i = \text{argmax}_c\, p_{ic}$;
```

Feature Selection: Filter Method

class 1 = x windows   versus   class 2 = ms windows



class 1	prob	class 2	prob	highest MI	MI
subject	0.998	subject	0.998	windows	0.215
this	0.628	windows	0.639	microsoft	0.095
with	0.535	this	0.540	dos	0.092
but	0.471	with	0.538	motif	0.078
you	0.431	but	0.518	window	0.067

(450 + 1) / ((450 + 1) + (0 + 1)) = 451 / 452 = 0.998

Mutual Information Example

		class		
		x windows	ms windows	
microsoft	present	6	107	113
	absent	444	343	787
		450	450	

$$I(X,Y) = \sum_{x_j} \sum_{y} p(x_j, y) \log \frac{p(x_j, y)}{p(x_j)p(y)}$$

```
(  7 / 904) * log2( (  7 / 904) / ((114 / 902) * (451 / 902)) ) + ...
(108 / 904) * log2( (108 / 904) / ((114 / 902) * (451 / 902)) ) + ...
(445 / 904) * log2( (445 / 904) / ((788 / 902) * (451 / 902)) ) + ...
(344 / 904) * log2( (344 / 904) / ((788 / 902) * (451 / 902)) )
= 0.095
```

Note: ((6 + 1) / ((6 + 1) + (444 + 1))) * ((450 + 1) / ((450 + 1) + (450 + 1))) = (7 / 904)

Recommendation Systems

Approaches

- Collaborative Filtering
  - Using a ratings matrix to impute missing ratings and make recommendations
  - Ratings may be either explicit (e.g. 5 stars) or implicit (e.g. purchase)
  - Popular method: Alternating Least Squares (ALS)
  - Cold start problem: new users and new items won't have ratings
- Content Filtering
  - Using features of users (e.g. age, gender, location) and items (e.g. release date, genre, leading actress for a movie) to make recommendations
  - Supervised learning methods apply
  - Example: ICDM 2013 Competition to Personalize Expedia Hotel Search Results: gradient boosting for the Normalized Discounted Cumulative Gain (NDCG) loss function
  - If ratings are available, a collaborative filtering prediction can be used as a feature [stacking: treating the output of one model as an input for another]

Collaborative Filtering: Nearest Neighbor

- User-based model:

  Assumes similar users will have similar ratings for the same item

- Item-based model:

  Assumes similar items will have similar ratings from the same user

Nearest Neighbor Rating Example

- users are indexed by ‘u’ and ‘v’
- ‘s’ measures similarity between users
- ‘r’ represents a rating for item ‘i’

$$s_{u,v} = \frac{\vec{r}_u \cdot \vec{r}_v}{\|\vec{r}_u\| \|\vec{r}_v\|} = \frac{\sum_{i \in \{RatedByBoth\}} r_{u,i} * r_{v,i}}{\sqrt{\sum_{i \in \{RatedByBoth\}} r_{u,i}^{\ 2}} \sqrt{\sum_{i \in \{RatedByBoth\}} r_{v,i}^{\ 2}}}$$

$$r_{u,i} = \sum_{v \in \{NearestNeighborSet[u]\}} \left[\frac{s_{u,v}}{\sum_{v \in \{NearestNeighborSet[u]\}} s_{u,v}} r_{v,i} \right]$$

We can replace $r_{u,i}$ with $r_{u,i} - \overline{r}_u$ (for each user) to adjust for differences in mean rating between users

Alternating Least Squares Matrix Factorization

UserLatentFactorMatrix * ItemLatentFactorMatrix = RatingMatrix

[m users, f factors] * [f factors, n items]

$$\begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \\ u_{3,1} & u_{3,2} \\ u_{4,1} & u_{4,2} \\ u_{5,1} & u_{5,2} \\ u_{6,1} & u_{6,2} \end{pmatrix} \times \begin{pmatrix} i_{1,1} & i_{1,2} & i_{1,3} & i_{1,4} \\ i_{2,1} & i_{2,2} & i_{2,3} & i_{2,4} \end{pmatrix} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\ r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \\ r_{5,1} & r_{5,2} & r_{5,3} & r_{5,4} \\ r_{6,1} & r_{6,2} & r_{6,3} & r_{6,4} \end{pmatrix}$$

Alternating Least Squares Algorithm

1. Initialize the items matrix with random entries
2. Repeat until convergence (“no” change) or max iterations reached:
   a) Fix the items matrix, and solve for regression coefficients for each row of the users matrix [a model for each user]
   b) Fix the users matrix, and solve for regression coefficients for each column of the items matrix [a model for each item]

The regression coefficients form latent (unobserved) factors for users and items

References

“Matrix Factorization Techniques for Recommender Systems”

[from part of the winning team for the $1 million NetFlix prize]

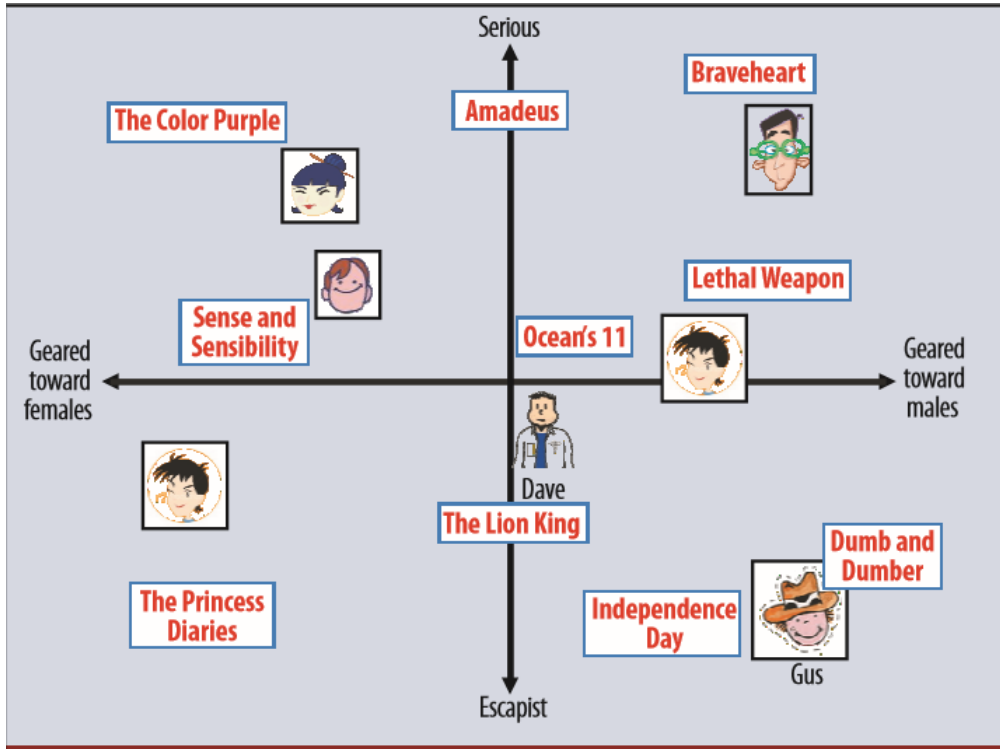


Figure 2. A simplified illustration of the latent factor approach, which characterizes both users and movies using two axes—male versus female and serious versus escapist.

Section 27.6.2 of our text book addresses “Probabilistic Matrix Factorization for Collaborative Filtering”

Apache Spark

- Apache Spark is a data processing platform, based on the concept of the Resilient Distributed Dataset (RDD)
  - Unlike Hadoop, the RDD persists data to memory across iterations
- Python API
  - http://spark.apache.org/docs/latest/api/python/index.html
  - We'll be using the ALS class of the recommendation module within the pyspark.mllib package: pyspark.mllib.recommendation.ALS

Strong versus Weak Scalability

- Strong scalability
  - Given 10 times as many computers, we should ideally be able to generate a model using the same data in 1/10th as much time [compared to using 1 computer: time shrinks]
  - Example: random forests
- Weak scalability
  - Given 10 times as many computers, we should ideally be able to generate a model using 10 times as much data in the same amount of time [compared to using 1 computer: data grows (big data)]
  - Example: gradient descent