

# **Generative Clustering, Topic Modeling, & Bayesian Inference**

INFO-4604, Applied Machine Learning  
University of Colorado Boulder

**December 11-13, 2018**

Prof. Michael Paul

# Unsupervised Naïve Bayes

Last week you saw how Naïve Bayes can be used in semi-supervised or unsupervised settings

- Learn parameters with the *EM algorithm*

Unsupervised Naïve Bayes is considered a type of **topic model** when used for text data

- Learns to group documents into different categories, referred to as “topics”
- Instances are documents; features are words

Today’s focus is text, but ideas can be applied to other types of data

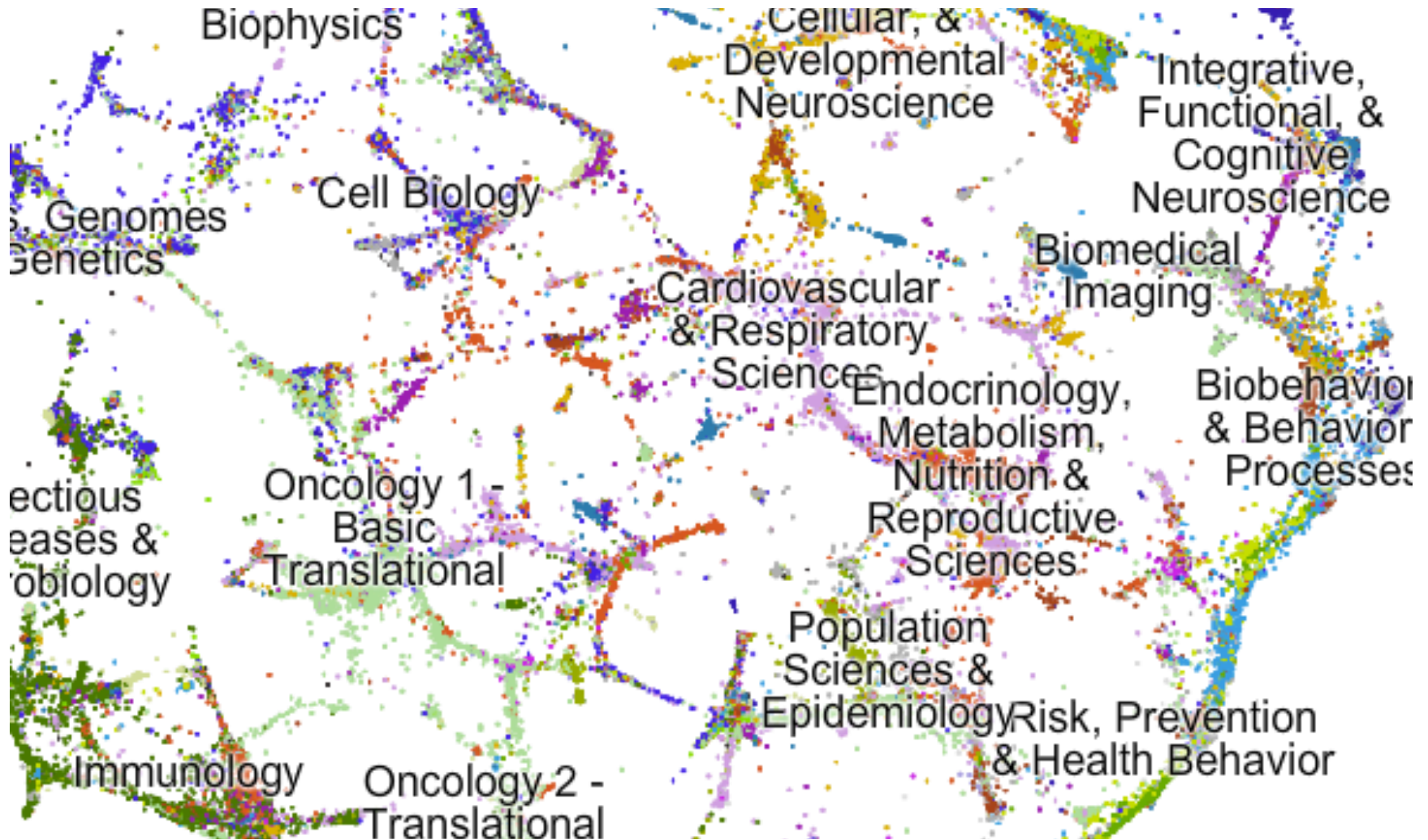
# Topic Models

Topic models are used to find common patterns in text datasets

- Method of **exploratory** analysis
- For understanding data rather than prediction (though sometimes also useful for prediction – we'll see at the end of this lecture)

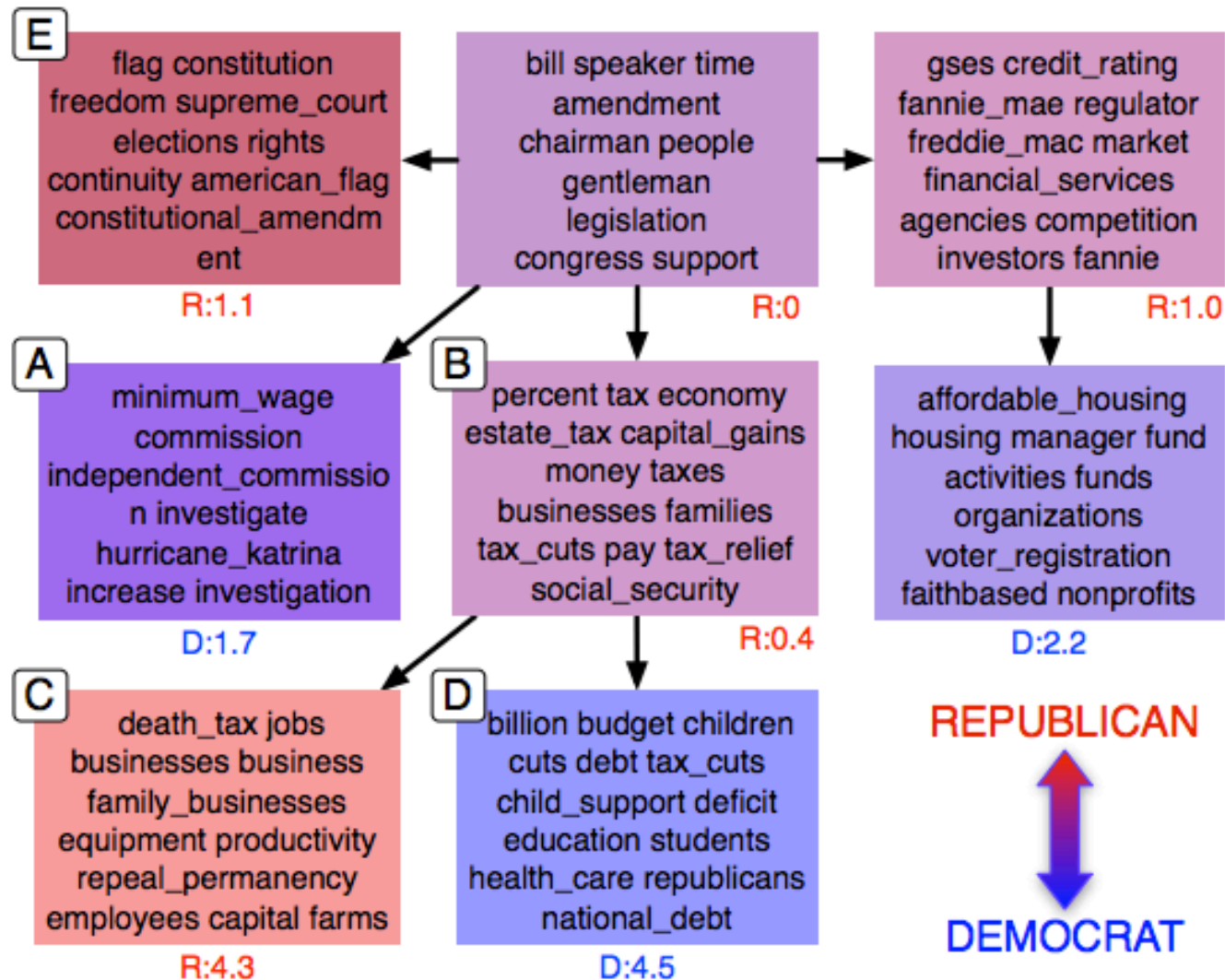
Unsupervised learning means that it can provide analysis without requiring a lot of input from a user

# Topic Models



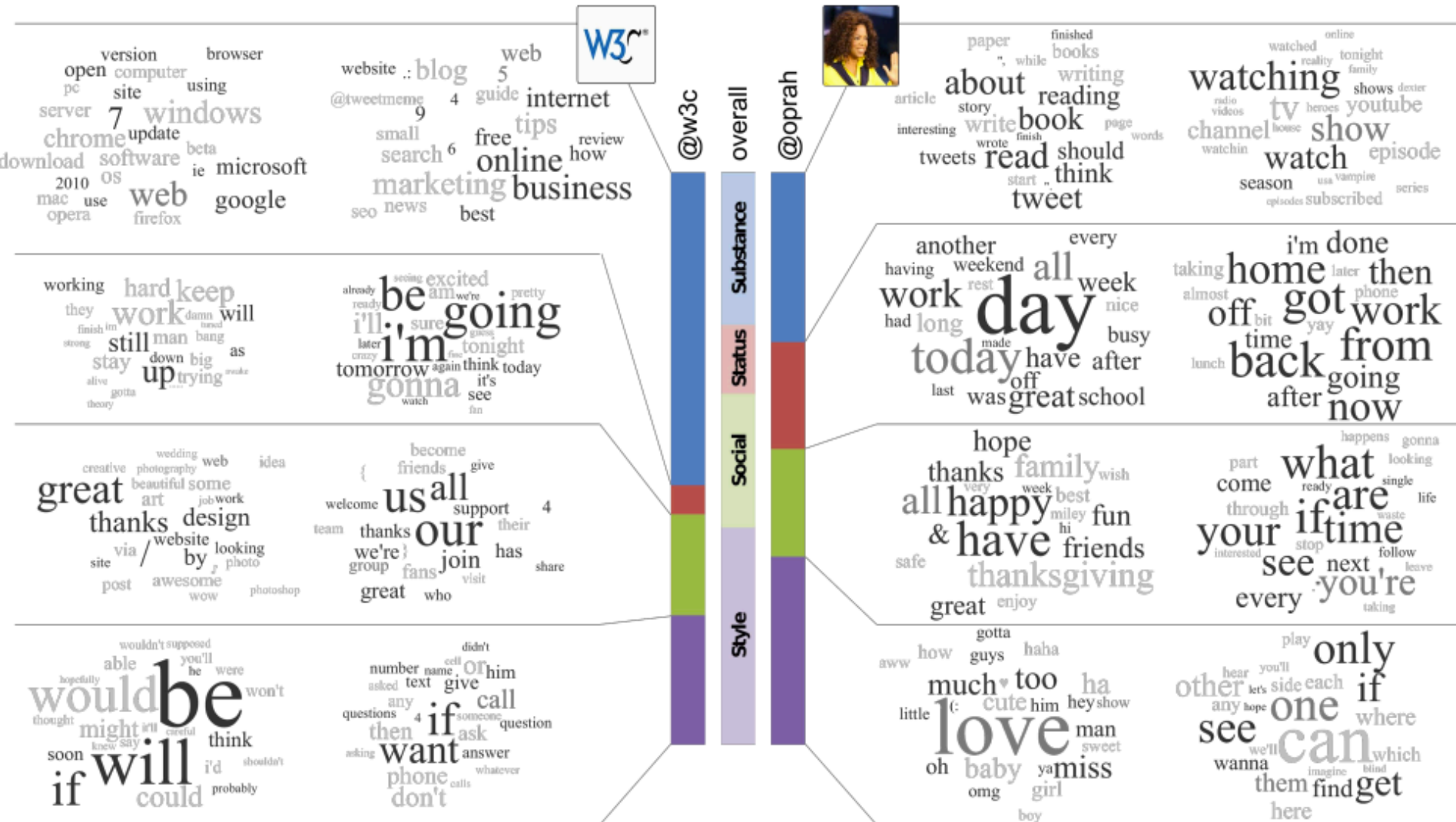
From Talley et al (2011)

# Topic Models



From Nguyen et al (2013)

# Topic Models



From Ramage et al (2010)

# Unsupervised Naïve Bayes

Naïve Bayes is not often used as a topic model

- We'll learn more common, more complex models today
- But let's start by reviewing it, and then build off the same ideas

# Generative Models

When we introduced generative models, we said that they can also be used to *generate* data



# Generative Models

How would you use Naïve Bayes to randomly generate a document?

First, randomly pick a category, ~~Y~~**Z**

- Notation convention to use  $Z$  for *latent* categories in unsupervised modeling instead of  $Y$  (since  $Y$  often implies it is a known value you are trying to predict)
- The category should be randomly sampled according to the prior distribution,  $P(Z)$

# Generative Models

How would you use Naïve Bayes to randomly generate a document?

First, randomly pick a category,  $Z$

Then, randomly pick words

- Sampled according to the distribution,  $P(W | Z)$

These steps are known as the *generative process* for this model

# Generative Models

How would you use Naïve Bayes to randomly generate a document?

This process won't result in a coherent document

- But, the words in the document are likely to be semantically/topically related to each other, since  $P(W \mid Z)$  will give high probability to words that are common in the particular category

# Generative Models

Another perspective on learning:

If you assume that the “generative process” for a model is how the data was generated, then work backwards and ask:

- What are the probabilities that most likely would have generated the data that we observe?

The generative process is almost always overly simplistic

- But it can still be a way to learn something useful

# Generative Models

With unsupervised learning, the same approach applies

- What are the probabilities that most likely would have generated the data that we observe?
- If we observe similar patterns across multiple documents, those documents are likely to have been generated from the same latent category

# Naïve Bayes

Let's first review (unsupervised) Naïve Bayes and Expectation Maximization (EM)

# Naïve Bayes

Learning probabilities in Naïve Bayes:

$$P(X_j=x \mid Y=y) =$$

$$\frac{\text{\# instances with label } y \text{ where feature } j \text{ has value } x}{\text{\# instances with label } y}$$

# Naïve Bayes

Learning probabilities in **unsupervised** Naïve Bayes:

$$P(X_j=x \mid Z=z) =$$

$$\frac{\text{\# instances with category } z \text{ where feature } j \text{ has value } x}{\text{\# instances with category } z}$$



# Naïve Bayes

Learning probabilities in unsupervised Naïve Bayes:

$$P(X_j=x \mid Z=z) =$$

$$\frac{\text{Expected \# instances with category } z \text{ where feature } j \text{ has value } x}{\text{Expected \# instances with category } z}$$

- Using Expectation Maximization (EM)

# Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

## 1. Expectation step (E-step)

Calculate  $P(Z=z \mid X_i)$  = 
$$\frac{P(X_i \mid Z=z) P(Z=z)}{\underbrace{\sum_{y'} P(X_i \mid Z=z') P(Z=z')}_{\text{These parameters come from the previous iteration of EM}}}$$
  
for every instance

# Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

## 2. Maximization step (M-step)

Update the probabilities  $P(X \mid Z)$  and  $P(Z)$ , replacing the observed counts with the **expected values** of the counts

- Equivalent to  $\sum_i P(Z=z \mid X_i)$

# Expectation Maximization (EM)

The EM algorithm iteratively alternates between two steps:

2. Maximization step (M-step)

$$P(X_j=x \mid Z=z) = \frac{\sum_i P(Z=z \mid X_i) I(X_{ij}=x)}{\underbrace{\sum_i P(Z=z \mid X_i)}}_{\text{These values come from the E-step}}$$

for each feature  $j$   
and each category  $z$

These values come  
from the E-step

# Unsupervised Naïve Bayes

1. Need to set the number of latent classes
2. Initially define the parameters *randomly*
  - Randomly initialize  $P(X | Z)$  and  $P(Z)$  for all features and classes
3. Run the EM algorithm to update  $P(X | Z)$  and  $P(Z)$  based on unlabeled data
4. After EM converges, the final estimates of  $P(X | Z)$  and  $P(Z)$  can be used for clustering

# Unsupervised Naïve Bayes

In (unsupervised) Naïve Bayes, each document belongs to one category

- This is a typical assumption for classification (though it doesn't have to be – remember multi-label classification)

# Admixture Models

In (unsupervised) Naïve Bayes, each document belongs to one category

- This is a typical assumption for classification (though it doesn't have to be – remember multi-label classification)

A better model might allow documents to contain *multiple* latent categories (aka topics)

- Called an **admixture** of topics

# Admixture Models

## Topics

gene 0.04  
dna 0.02  
genetic 0.01  
...

life 0.02  
evolve 0.01  
organism 0.01  
...

brain 0.04  
neuron 0.02  
nerve 0.01  
...

data 0.02  
number 0.02  
computer 0.01  
...

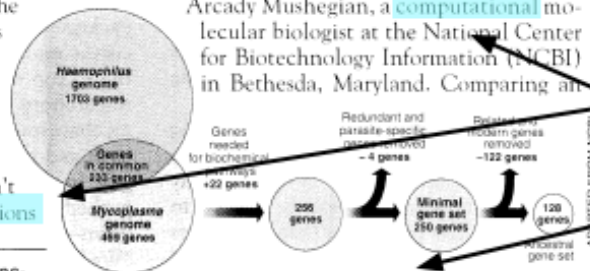
## Documents

### Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to **survive**? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for **life**. One research team, using **computer** analyses to compare known **genomes**, concluded that today's **organisms** can be sustained with just 250 genes, and that the earliest life forms required a mere 128 **genes**. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those **predictions**

"are not all that far apart," especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson at Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic numbers** game, particularly as more and more **genomes** are completely mapped and sequenced. "It may be a way of organizing any newly **sequenced genome**," explains Arcady Mushegian, a **computational** molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. **Computer analysis** yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

## Topic proportions and assignments



# Admixture Models

In an *admixture* model, each document has different *proportions* of different topics

- Unsupervised Naïve Bayes is considered a *mixture* model (the dataset contains a mixture of topics, but each instance has only one topic)

Probability of each topic in a specific document

- $P(Z \mid d)$
- Another type of parameter to learn

# Admixture Models

In this type of model, the “generative process” for a document  $d$  can be described as:

1. For each token in the document  $d$ :
  - a) Sample a topic  $z$  according to  $P(z \mid d)$
  - b) Sample a word  $w$  according to  $P(w \mid z)$

Contrast with Naïve Bayes:

1. Sample a topic  $z$  according to  $P(z)$
2. For each token in the document  $d$ :
  - a) Sample a word  $w$  according to  $P(w \mid z)$

# Admixture Models

In this type of model, the “generative process” for a document  $d$  can be described as:

1. For each token in the document  $d$ :
  - a) Sample a topic  $z$  according to  $P(z | d)$
  - b) Sample a word  $w$  according to  $P(w | z)$
- Same as in Naïve Bayes  
(each “topic” has a distribution of words)
- Parameters can be learned in a similar way
  - Called  $\beta$  (sometimes  $\phi$ ) by convention

# Admixture Models

In this type of model, the “generative process” for a document  $d$  can be described as:

1. For each token in the document  $d$ :
  - a) Sample a topic  $z$  according to  $P(z | d)$
  - b) Sample a word  $w$  according to  $P(w | z)$
- Related to but different from Naïve Bayes
  - Instead of one  $P(z)$  shared by every document, each document has its own distribution
- More parameters to learn
  - Called  $\theta$  by convention

# Admixture Models

## Topics

 $\beta_1$ 

gene	0.04
dna	0.02
genetic	0.01
...	

 $\beta_2$ 

life	0.02
evolve	0.01
organism	0.01
...	

 $\beta_3$ 

brain	0.04
neuron	0.02
nerve	0.01
...	

 $\beta_4$ 

data	0.02
number	0.02
computer	0.01
...	

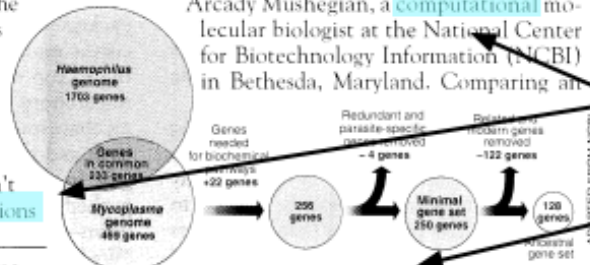
## Documents

### Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many **genes** does an **organism** need to **survive**? Last week at the genome meeting here,\* two genome researchers with radically different approaches presented complementary views of the basic genes needed for **life**. One research team, using **computer** analyses to compare known **genomes**, concluded that today's **organisms** can be sustained with just 250 genes, and that the earliest life forms required a mere 128 **genes**. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those **predictions**

"are not all that far apart," especially in comparison to the 75,000 **genes** in the human genome, notes Siv Andersson at Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a **genetic** numbers game, particularly as more and more **genomes** are completely mapped and sequenced. "It may be a way of organizing any newly **sequenced genome**," explains Arcady Mushegian, a **computational** molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

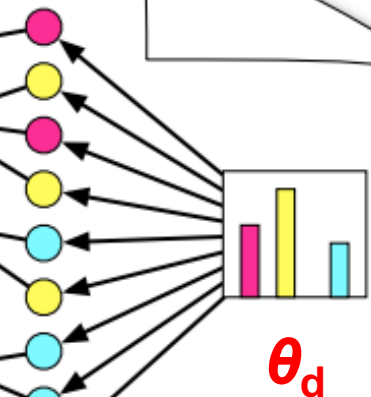


\* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

## Topic proportions and assignments



# Learning

How to learn  $\beta$  and  $\theta$ ?

Expectation Maximization (EM) once again!

# Learning

## E-step

$$\begin{aligned} &P(\text{topic}=j \mid \text{word}=v, \theta_d, \beta_j) \\ &= \frac{P(\text{word}=v, \text{topic}=j \mid \theta_d, \beta_j)}{\sum_k P(\text{word}=v, \text{topic}=k \mid \theta_d, \beta_k)} \end{aligned}$$

# Learning

## M-step

$$\begin{aligned} &\text{new } \theta_{dj} \\ &= \frac{\# \text{ tokens in } d \text{ with topic label } j}{\# \text{ tokens in } d} \end{aligned}$$

*if the topic labels were observed!*

- just counting



# Learning

## M-step

new  $\theta_{dj}$

$$= \frac{\sum_{i \in d} P(\text{topic } i=j \mid \text{word } i, \theta_d, \beta_j)}{\sum_k \sum_{i \in d} P(\text{topic } i=k \mid \text{word } i, \theta_d, \beta_k)}$$

← just the number of tokens in the document

sum over each token  $i$  in document  $d$

- numerator: the expected number of tokens with topic  $j$  in document  $d$
- denominator: the number of tokens in document  $d$

# Learning

## M-step

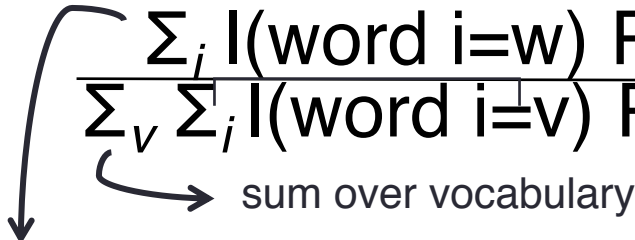
$$\begin{aligned} &\text{new } \beta_{jw} \\ &= \frac{\# \text{ tokens with topic label } j \text{ and word } w}{\# \text{ tokens with topic label } j} \end{aligned}$$

*if the topic labels were observed!*

- just counting

# Learning

## M-step

$$\text{new } \beta_{jw} = \frac{\sum_i I(\text{word } i=w) P(\text{topic } i=j \mid \text{word } i=w, \theta_d, \beta_j)}{\sum_v \sum_i I(\text{word } i=v) P(\text{topic } i=j \mid \text{word } i=v, \theta_d, \beta_j)}$$


sum over each token  $i$  in the entire corpus

- numerator: the expected number of times word  $w$  belongs to topic  $j$
- denominator: the expected number of all tokens belonging to topic  $j$

# Smoothing

From last week's Naïve Bayes lecture:

Adding “pseudocounts” to the observed counts when estimating  $P(X | Y)$  is called **smoothing**

Smoothing makes the estimated probabilities less extreme

- It is one way to perform regularization in Naïve Bayes (reduce overfitting)

# Smoothing

Smoothing is also commonly done in unsupervised learning like topic modeling

- Today we'll see a mathematical justification for smoothing

# Smoothing: Generative Perspective

In general models, we can also treat the parameters themselves as random variables

- $P(\theta)$ ?
- $P(\beta)$ ?

Called the **prior** probability of the parameters

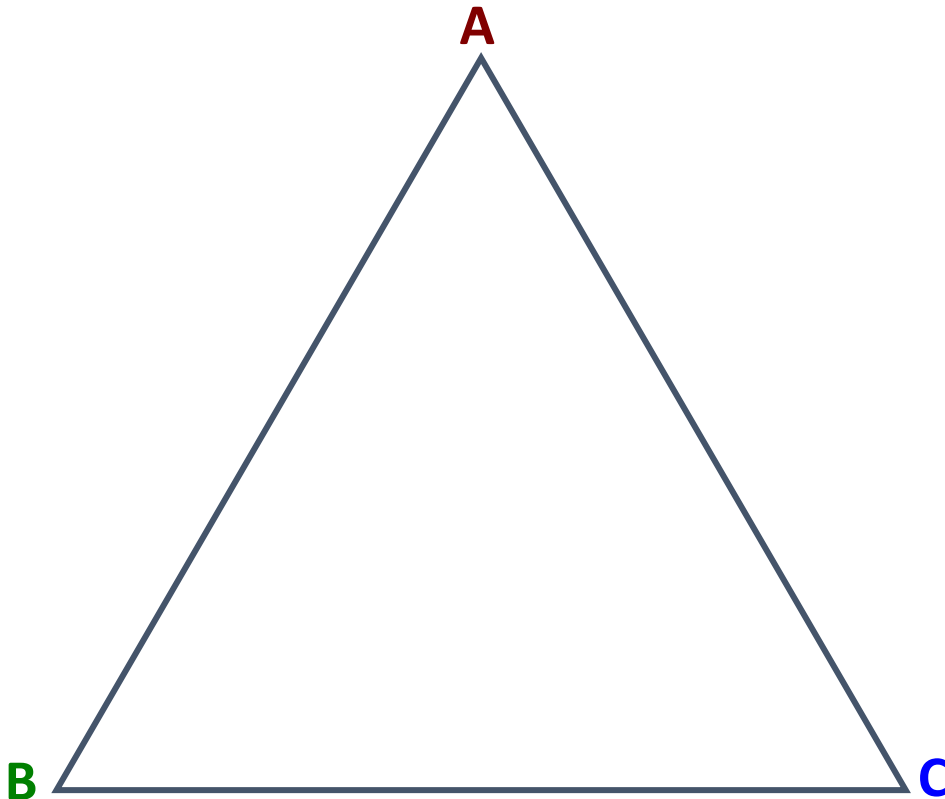
- Same concept as the prior  $P(Y)$  in Naïve Bayes

We'll see that pseudocount smoothing is the result when the parameters have a prior distribution called the **Dirichlet** distribution

# Geometry of Probability

A distribution over  $K$  elements is a point on a  $K-1$  **simplex**

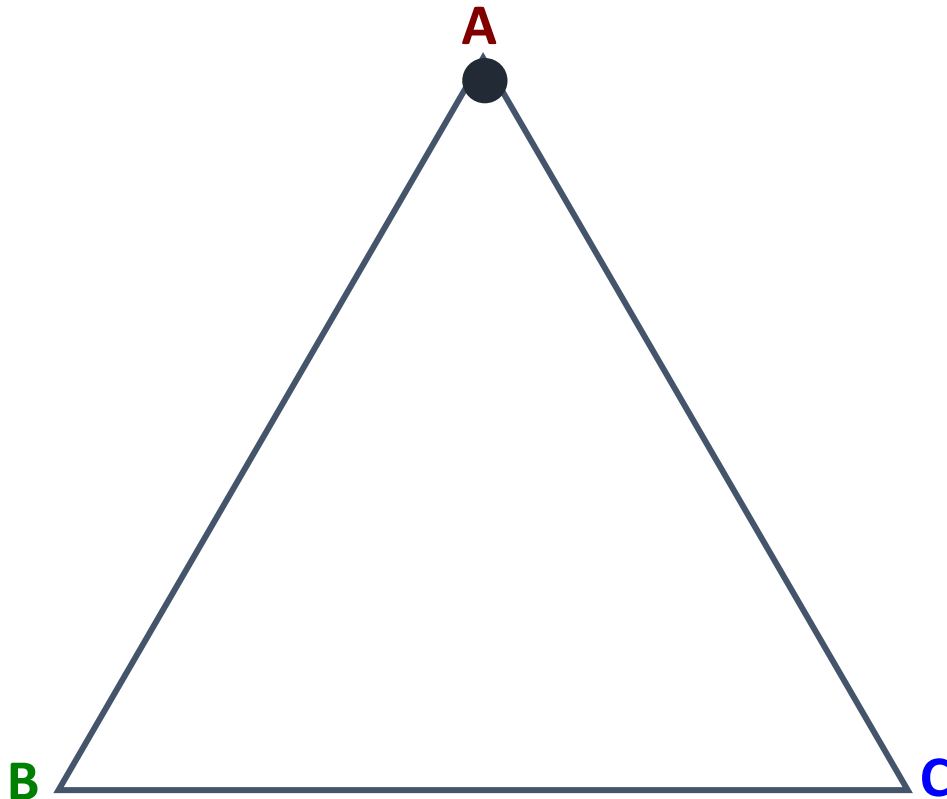
- a 2-simplex is called a triangle



# Geometry of Probability

A distribution over  $K$  elements is a point on a  $K-1$  **simplex**

- a 2-simplex is called a triangle



$$P(\textcolor{red}{A}) = 1$$

$$P(\textcolor{green}{B}) = 0$$

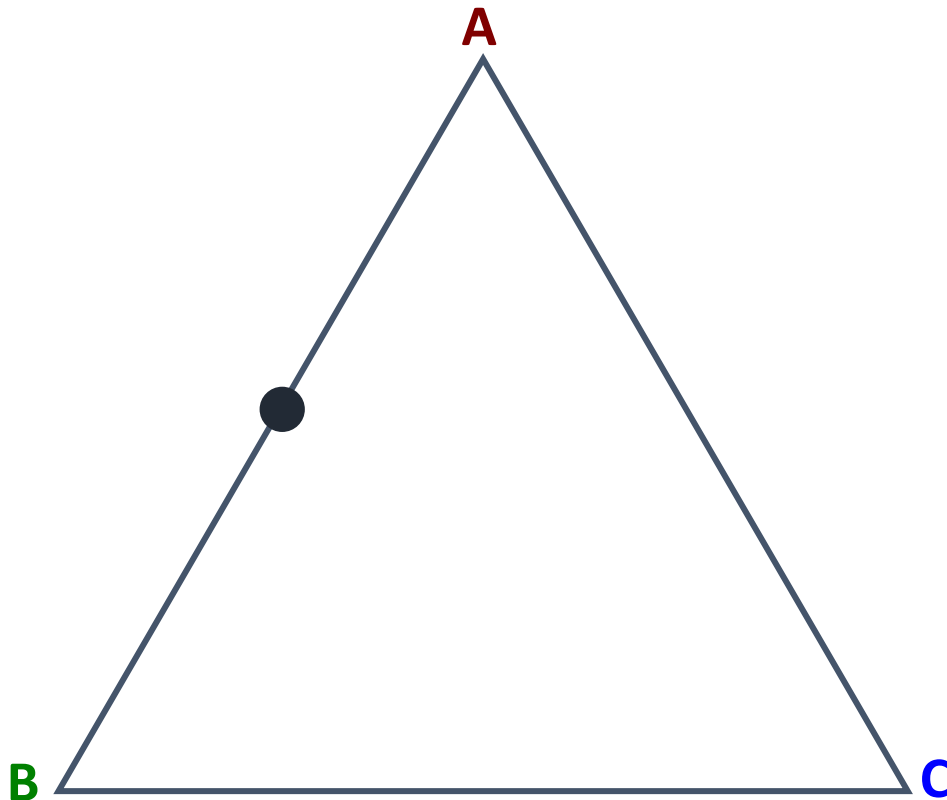
$$P(\textcolor{blue}{C}) = 0$$



# Geometry of Probability

A distribution over  $K$  elements is a point on a  $K-1$  **simplex**

- a 2-simplex is called a triangle



$$P(\textcolor{red}{A}) = 1/2$$

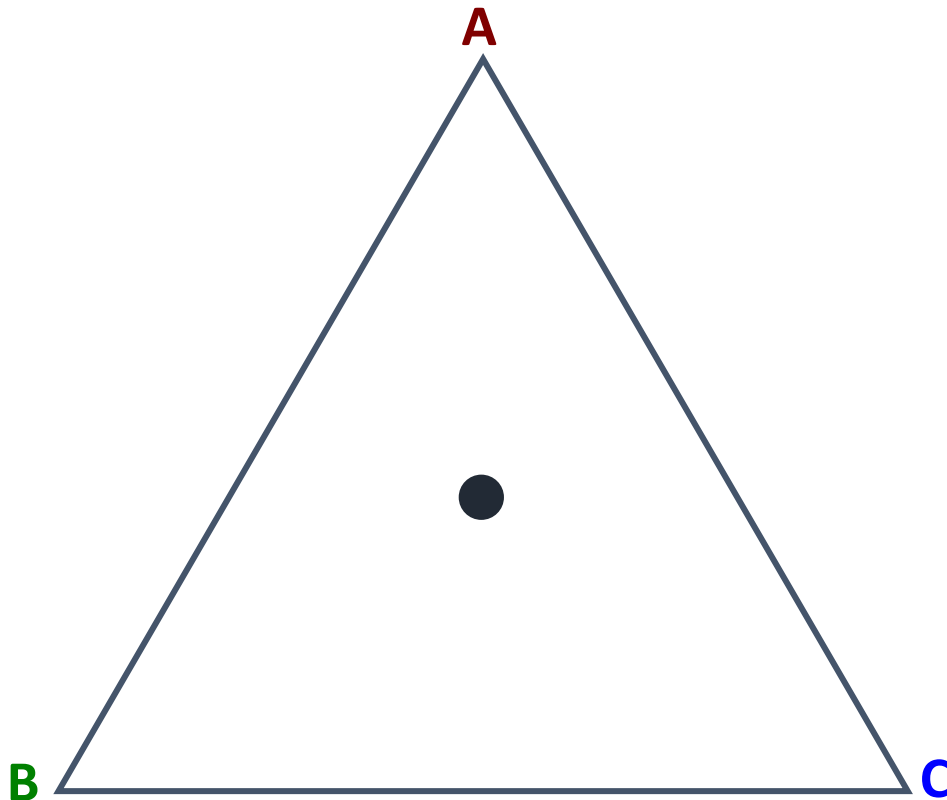
$$P(\textcolor{green}{B}) = 1/2$$

$$P(\textcolor{blue}{C}) = 0$$

# Geometry of Probability

A distribution over  $K$  elements is a point on a  $K-1$  **simplex**

- a 2-simplex is called a triangle



$$P(\textcolor{red}{A}) = 1/3$$

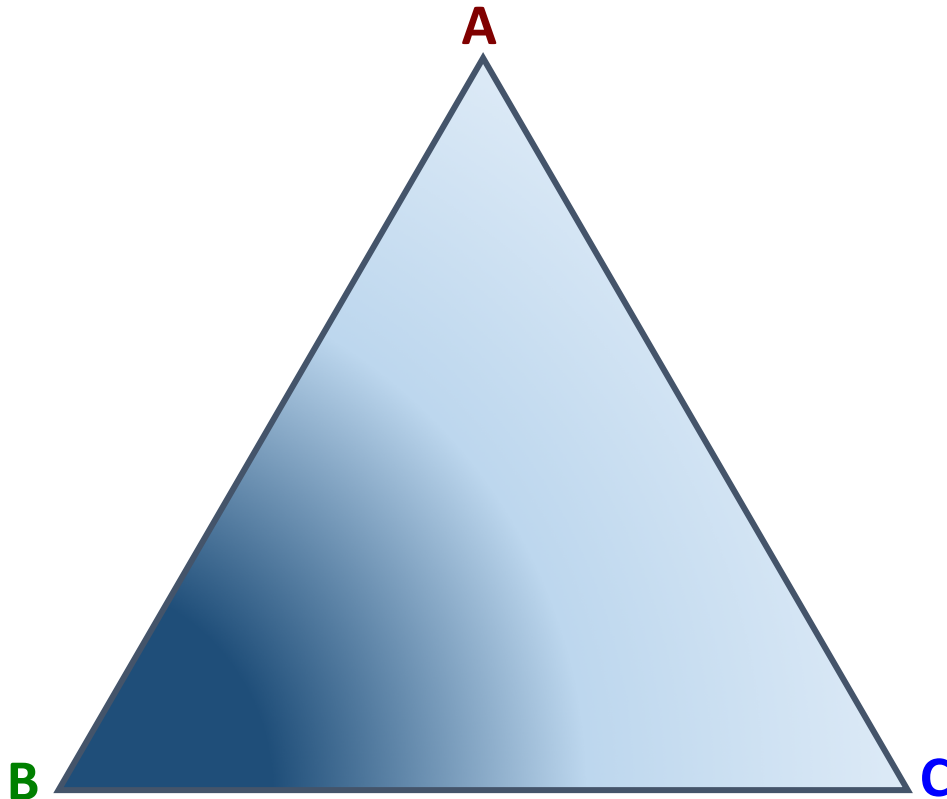
$$P(\textcolor{green}{B}) = 1/3$$

$$P(\textcolor{blue}{C}) = 1/3$$

# Dirichlet Distribution

Continuous distribution (probability density) over points in the simplex

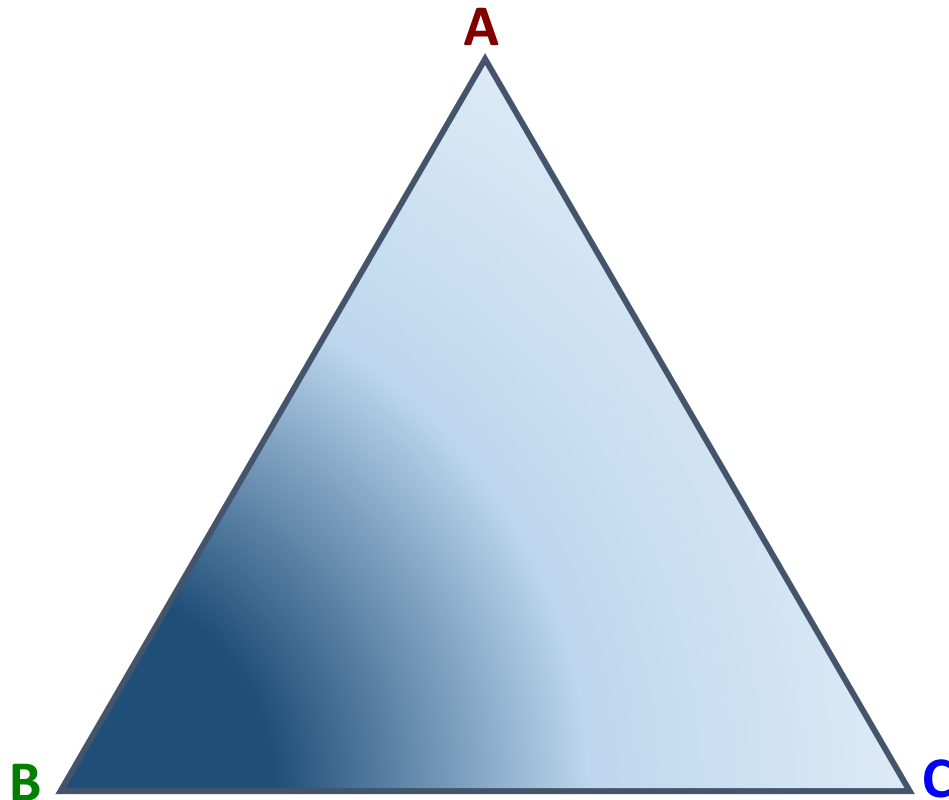
- “distribution of distributions”



# Dirichlet Distribution

Continuous distribution (probability density) over points in the simplex

- “distribution of distributions”



Denoted  $\text{Dirichlet}(\boldsymbol{a})$

$\boldsymbol{a}$  is a vector that gives the mean/variance of the distribution

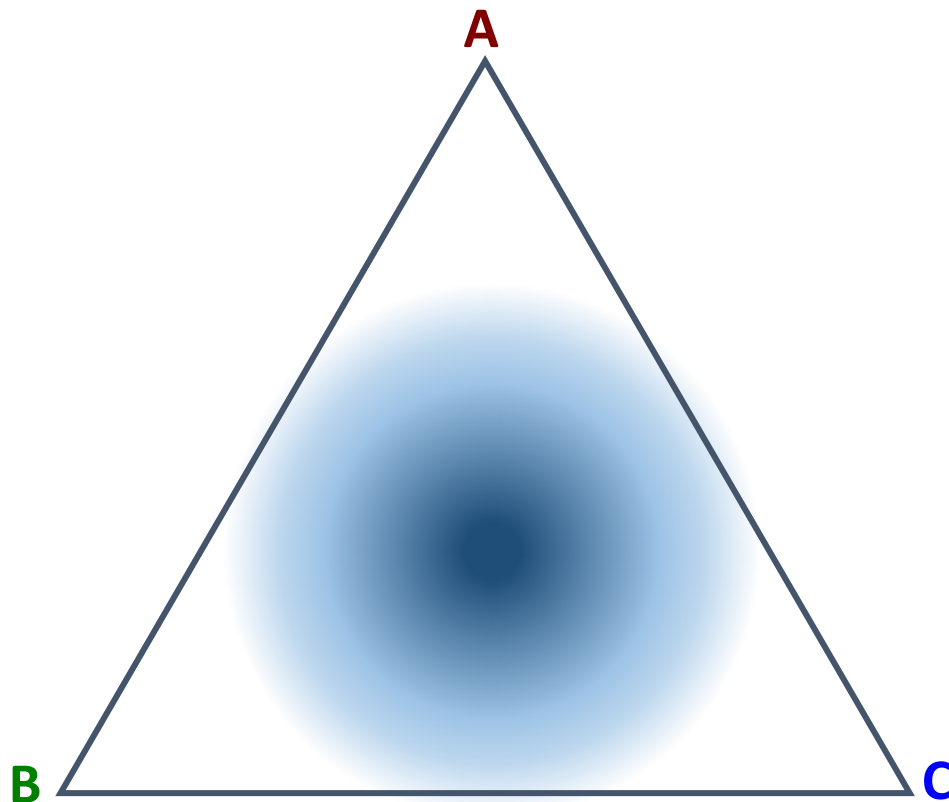
In this example,  $a_B$  is larger than the others, so points closer to **B** are more likely

- Distributions that give **B** high probability are more likely than distributions that don't

# Dirichlet Distribution

Continuous distribution (probability density) over points in the simplex

- “distribution of distributions”



Denoted  $\text{Dirichlet}(\boldsymbol{a})$

$\boldsymbol{a}$  is a vector that gives the mean/variance of the distribution

In this example,  $a_A = a_B = a_C$ , so distributions close to uniform are more likely

Larger values of  $\boldsymbol{a}$  give higher density around mean (lower variance)

# Latent Dirichlet Allocation (LDA)

LDA is the topic model previous slides, but with Dirichlet priors on the parameters  $\theta$  and  $\beta$

- $P(\theta \mid \alpha) = \text{Dirichlet}(\alpha)$
- $P(\beta \mid \eta) = \text{Dirichlet}(\eta)$

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) \\ = \prod_{i=1}^K p(\beta_i) \prod_{d=1}^D p(\theta_d) \left( \prod_{n=1}^N p(z_{d,n} \mid \theta_d) p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right)$$

- Most widely used topic model
  - Lots of different implementations / learning algorithms

# MAP Learning

How to learn  $\beta$  and  $\theta$  with Dirichlet priors?

The **posterior** distribution of parameters for LDA:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} \mid w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

- Want to maximize this

# MAP Learning

So far we have used EM to find parameters that maximize the likelihood of the data

EM can also find the **maximum a posteriori (MAP)** solution

- the parameters that maximize the posterior probability


$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})} \leftarrow \text{constant}$$

- Similar objective as before, but with additional terms for the probability of  $\theta$  and  $\beta$



# MAP Learning

- E-step is the same
- M-step is modified

new  $\theta_{d1}$   pseudocounts

$$= \frac{\overline{a_1 - 1} + \sum_{i \in d} P(\text{topic } i=1 \mid \text{word } i, \theta_d, \beta_1)}{\sum_k (\overline{a_k - 1} + \sum_{i \in d} P(\text{topic } i=k \mid \text{word } i, \theta_d, \beta_k))}$$

# MAP Learning

Where do the pseudocounts come from?

The probability of observing the  $k$ th topic  $n$  times given the parameter  $\theta_k$  is proportional to:

$$\theta_k^n$$

The probability density of the parameter  $\theta_k$  given the Dirichlet parameter  $\alpha_k$  is proportional to:

$$\theta_k^{\alpha_k-1}$$

The product of these probabilities is proportional to:

$$\theta_k^{n+\alpha_k-1}$$

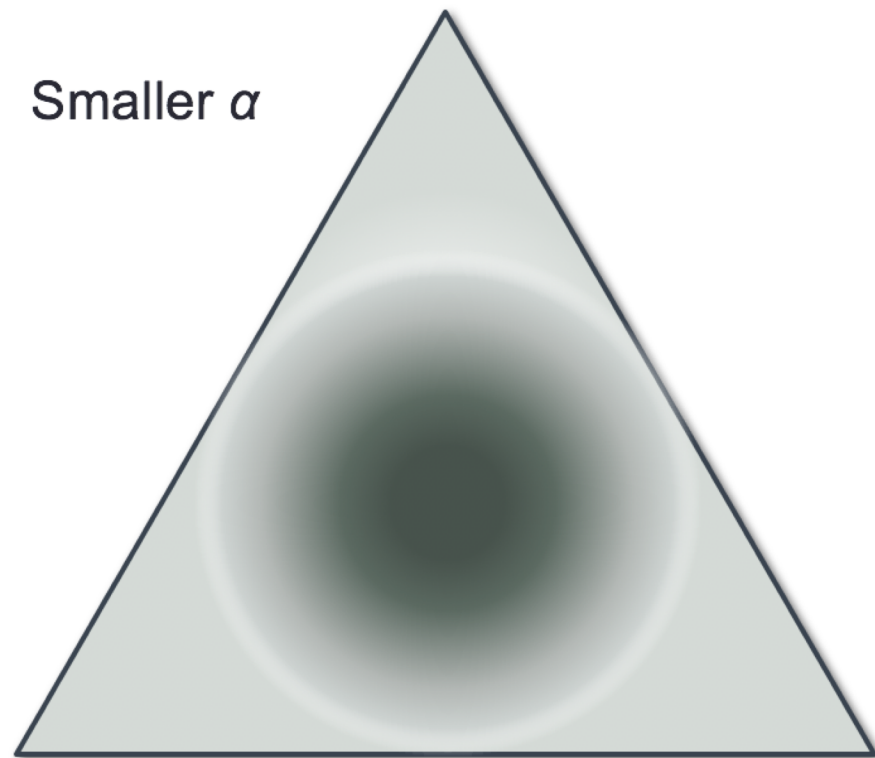
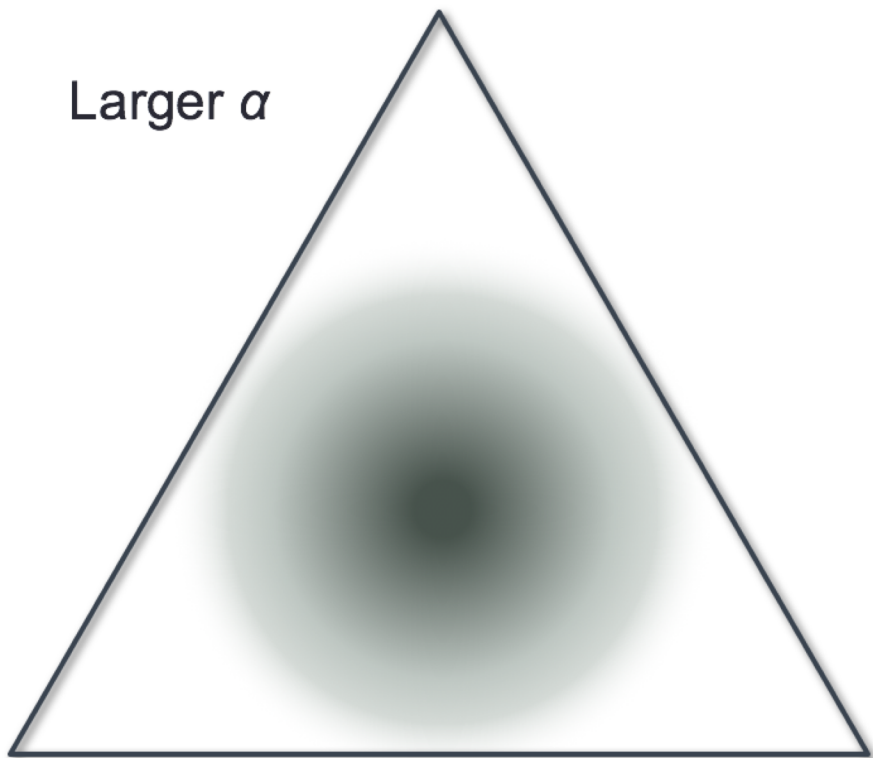
# Smoothing: Generative Perspective

Larger pseudocounts will bias the MAP estimate more heavily

Larger Dirichlet parameters concentrate the density around the mean

Larger  $\alpha$

Smaller  $\alpha$



# Smoothing: Generative Perspective

Dirichlet prior MAP estimation yields “ $\alpha - 1$ ” smoothing

- So what happens if  $\alpha < 1$ ?

Highest density around edges of simplex

- Prior favors small number of topics per document

# Posterior Inference

What if we don't just want the parameters that maximize the posterior?

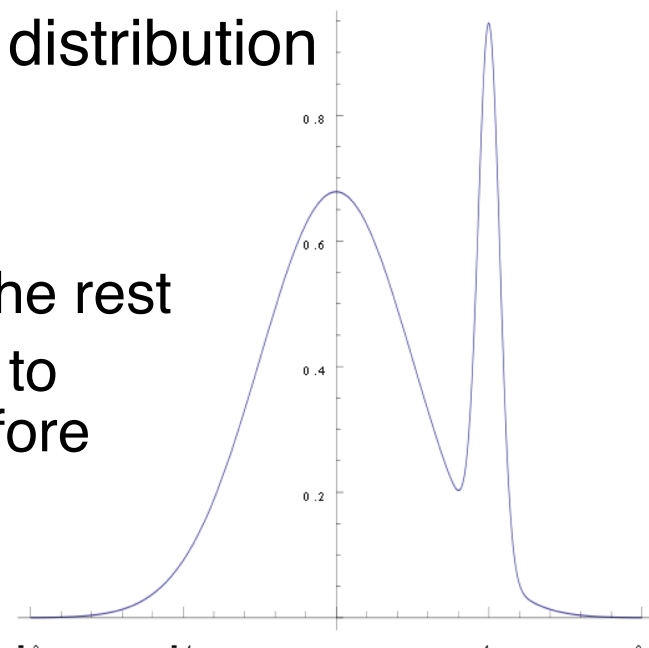
$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

What if we care about the entire posterior distribution?

- or at least the mean of the posterior distribution

Why?

- maybe the maximum doesn't look like the rest
- other points of the posterior more likely to generalize to data you haven't seen before



# Posterior Inference

What if we don't just want the parameters that maximize the posterior?

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

**This is harder**



- Computing the denominator involves summing over all possible configurations of the latent variables/parameters

# Posterior Inference

Various methods existing for approximating the posterior (also called **Bayesian** inference)

- Random sampling
  - Monte Carlo methods
- Variational inference
  - Optimization using EM-like procedure
  - MAP estimation is a simple case of this

# Dimensionality Reduction

Recall:

Methods like PCA can transform a high-dimensional feature space (e.g., each word is a feature) into a low-dimensional space

- Each feature vector is rewritten as a new vector



# Dimensionality Reduction

Topic models can also be used as a form of dimensionality reduction

- Each document's feature vector is  $\theta_d$ , aka  $P(Z | d)$ 
  - With 100 topics, this is a 100-dimensional vector
  - Semantically similar words will map to a similar part of the feature space, since they tend to be grouped into the same topics

This is similar to the ideas behind “embedding” methods like *word2vec*

# Priors as Regularization

We saw that Dirichlet priors are equivalent to pseudocount smoothing, which is used as regularization in Naïve Bayes

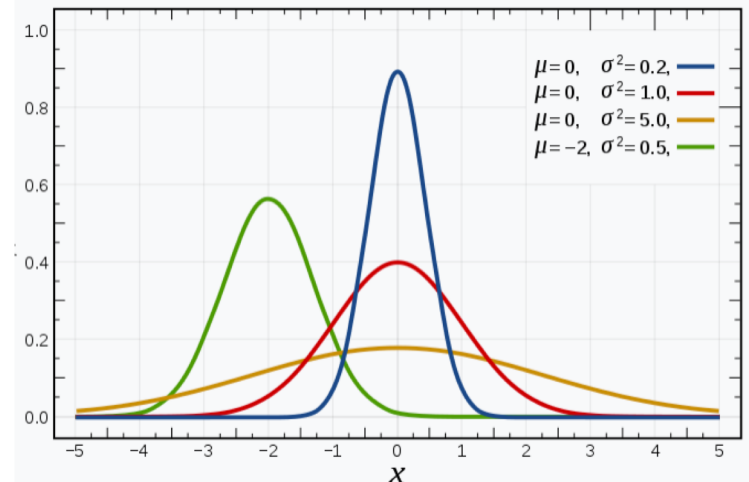
Other types of priors are equivalent to other types of regularization you've seen!

# Priors as Regularization

Recall: For real-valued weights (e.g., SVM or logistic regression), the most common type of regularization is to minimize the L2 norm of the weights

Minimizing the L2 norm ends up being mathematically equivalent to having a prior distribution on the weights where the prior is the Gaussian (normal) distribution!

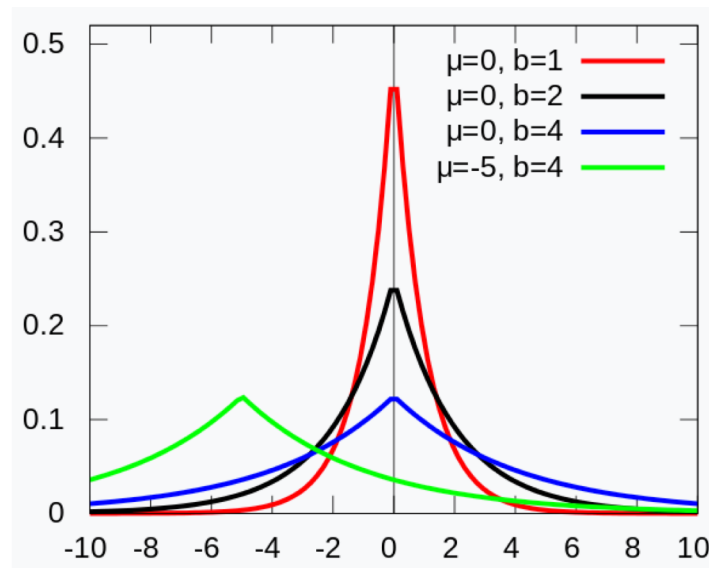
- The mean of the Gaussian is 0
- The variance of the Gaussian acts as the regularization strength ('C' or 'alpha')



# Priors as Regularization

L1 regularization, which favors weights that are exactly 0, is equivalent to the Laplace (double exponential) distribution as the prior

- Like with Gaussian, the mean is 0 and variance adjusts the regularization strength



# Priors as Inductive Bias

Recall that an *inductive bias* intentionally biases what a classifier learns toward certain characteristics that you think will be useful

- Regularization toward small weights is a common type of inductive bias in machine learning
- There are other useful inductive biases that can be encoded as priors
  - Any prior on the parameters is an inductive bias

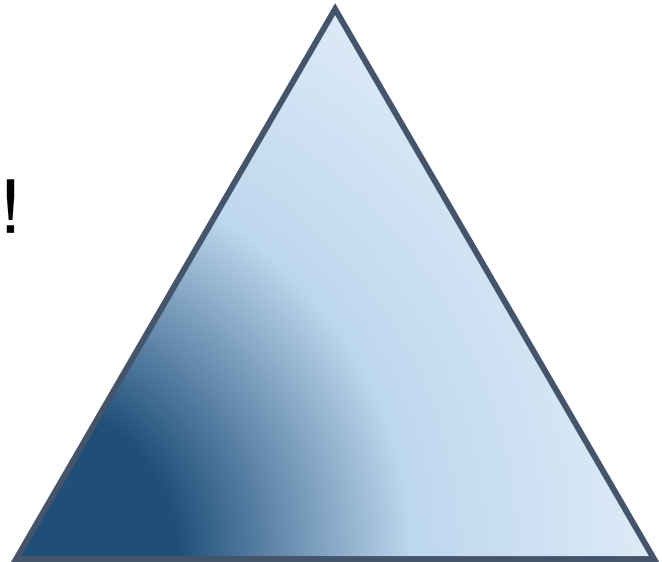
# Priors as Inductive Bias

In topic models:

Dirichlet priors bias the learned distributions toward the uniform distribution

- Yields less extreme probabilities, reducing overfitting

But Dirichlet priors don't  
have to bias toward uniform!  
Other biases can be useful.



# Priors as Inductive Bias

In topic models:

Symmetric $\alpha$	0.080	a <b>field emission</b> an <b>electron</b> the
	0.080	a the <b>carbon</b> and <b>gas</b> to an
	0.080	the of a to and about at
	0.080	of a <b>surface</b> the with in <b>contact</b>
	0.080	the a and to is of <b>liquid</b>
Asymmetric $\alpha$	0.895	the a of to and is in
	0.187	<b>carbon nanotubes nanotube catalyst</b>
	0.043	sub is c or and n sup
	0.061	<b>fullerene compound fullerenes</b>
	0.044	<b>material particles coating inorganic</b>

# Priors as Inductive Bias

For real-valued parameters, a Gaussian prior with mean of 0 is equivalent to L2 regularization

Can also use a Gaussian prior with a mean set to some value other than 0!

- If you believe certain features should have a positive or negative weight, you could set the mean of the prior to a positive or negative value to bias it in that direction



# Priors as Inductive Bias

Example: domain adaptation

What to do when your training data includes different *domains* (distributions of data)?

- e.g., sentiment classification on reviews of movies and reviews of mattresses
- Challenge in machine learning: might learn patterns that work in one domain but not another

# Priors as Inductive Bias

One idea: learn each domain separately

- But this is limited because you have less training data for each domain
- How to learn domain-specific parameters while still using all of the training data?

One approach (Finkel and Manning 2009):

- Learn “overall” feature weights for all domains
- Learn domain-specific feature weights
  - The prior for the domain-specific weights is a Gaussian distribution where the mean is the “overall” weight