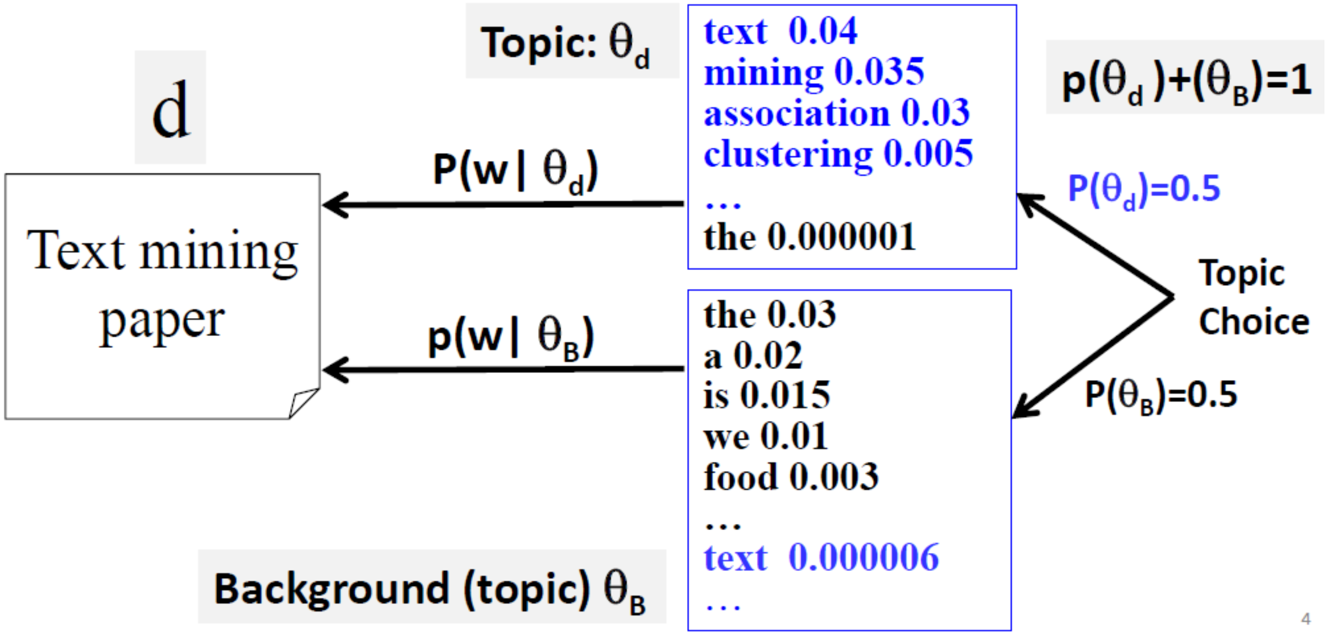
# Probabilistic Topic Models:
# Mixture of Unigram Language Models

# Factoring out Background Words: Generate d Using Two Word Distributions

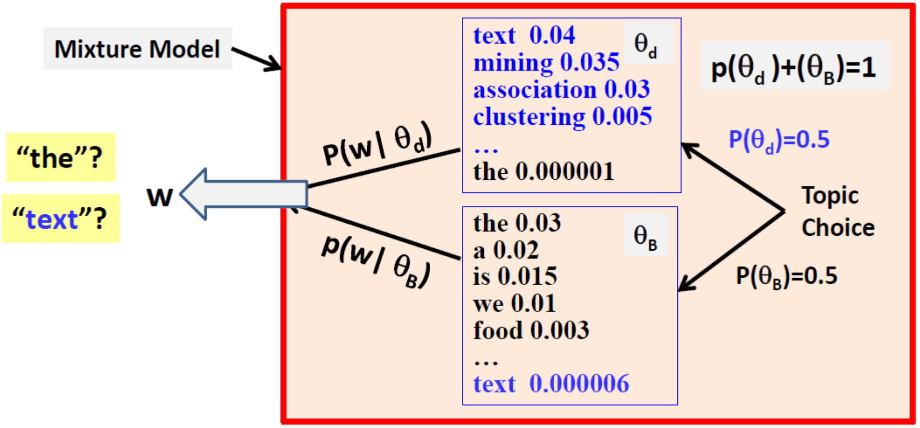





$$P(\text{"the"})=p(\theta_d)p(\text{"the"}|\theta_d) + p(\theta_B)p(\text{"the"}|\theta_B) = 0.5*0.000001+0.5*0.03$$

$$P(\text{"text"})=p(\theta_d)p(\text{"text"}|\theta_d) + p(\theta_B)p(\text{"text"}|\theta_B) = 0.5*0.04+0.5*0.000006$$

# The Idea of a Mixture Model




6

**Formally defines the following generative model:**

$$p(w)=p(\theta_d)p(w|\theta_d) + p(\theta_B)p(w|\theta_B)$$

**Estimate of the model "discovers" two topics + topic coverage**

**What if $p(\theta_d)=1$ or $p(\theta_B)=1$?**

# Mixture of Two Unigram Language Models

- **Data**: Document d
- Mixture **Model**: **parameters** $\Lambda=(\{p(w|\theta_d)\}, \{p(w|\theta_B)\}, p(\theta_B), p(\theta_d))$
  - Two unigram LMs: $\theta_d$ **(the topic of d); $\theta_B$ (background topic)**
  - Mixing weight (topic choice): **$p(\theta_d)+p(\theta_B)=1$**
- **Likelihood** function:

$$p(d|\Lambda)=\prod_{i=1}^{|d|} p(x_i|\Lambda)=\prod_{i=1}^{|d|}[p(\theta_d)p(x_i|\theta_d)+p(\theta_B)p(x_i|\theta_B)]$$

$$=\prod_{i=1}^{M}[p(\theta_d)p(w_i|\theta_d)+p(\theta_B)p(w_i|\theta_B)]^{c(w,d)}$$

- ML **Estimate:** $\Lambda^* = \arg\max_\Lambda p(d|\Lambda)$

  **Subject to** $\sum_{i=1}^{M} p(w_i|\theta_d)=\sum_{i=1}^{M} p(w_i|\theta_B)=1 \qquad p(\theta_d)+p(\theta_B)=1$

8

# Probabilistic Topic Models: Mixture Model Estimation

## Estimation of One Topic: $P(w|\theta_d)$

Adjust $\theta_d$ to maximize $p(d|\Lambda)$
(all other parameters are known)

Would the ML estimate demote background words in $\theta_d$ ?

**d**

... text mining...
is... clustering...
we.... Text.. the

$\theta_d$:
text ?
mining ?
association ?
clustering ?
...
the ?

$\theta_B$:
the 0.03
a 0.02
is 0.015
we 0.01
food 0.003
...
text 0.000006

$p(\theta_d)+(\theta_B)=1$

$P(\theta_d)=0.5$

Topic Choice

$P(\theta_B)=0.5$



## Behavior of a Mixture Model

**d** = text the

Likelihood:

$P(\text{"text"}) = p(\theta_d)p(\text{"text"}|\theta_d) + p(\theta_B)p(\text{"text"}|\theta_B)$
$= 0.5*p(\text{"text"}|\theta_d) + 0.5*0.1$

$P(\text{"the"}) = 0.5*p(\text{"the"}|\theta_d) + 0.5*0.9$

$p(d|\Lambda) = p(\text{"text"}|\Lambda)\, p(\text{"the"}|\Lambda)$
$= [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$

$\theta_d$:
text ?
the ?

$P(\theta_d)=0.5$

$P(\theta_B)=0.5$

$\theta_B$:
the 0.9
text 0.1

How can we set $p(\text{"text"}|\theta_d)$ & $p(\text{"text"}|\theta_d)$ to maximize it?

Note that $p(\text{"text"}|\theta_d) + p(\text{"the"}|\theta_d) = 1$



## "Collaboration" and "Competition" of $\theta_d$ and $\theta_B$

$p(d|\Lambda) = p(\text{"text"}|\Lambda)\, p(\text{"the"}|\Lambda)$
$= [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1] \times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$

**d** = text the

Note that $p(\text{"text"}|\theta_d) + p(\text{"the"}|\theta_d) = 1$

$\theta_d$:
text ?
the ?

$P(\theta_d)=0.5$

If $x + y = constant$, then $xy$ reaches maximum when $x = y$.

$P(\theta_B)=0.5$

$0.5*p(\text{"text"}|\theta_d) + 0.5*0.1 = 0.5*p(\text{"the"}|\theta_d) + 0.5*0.9$

$\theta_B$:
the 0.9
text 0.1

➔ $p(\text{"text"}|\theta_d)=0.9 \gg p(\text{"the"}|\theta_d)=0.1$ !

**Behavior 1:** if $p(w1|\theta_B) > p(w2|\theta_B)$, then $p(w1|\theta_d) < p(w2|\theta_d)$



## Response to Data Frequency

**d** = text the

$p(d|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1]$
$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$

➔ $p(\text{"text"}|\theta_d)=0.9 \gg p(\text{"the"}|\theta_d)=0.1$ !

**d'** = text the
the the
the ...the

$p(d'|\Lambda) = [0.5*p(\text{"text"}|\theta_d) + 0.5*0.1]$
$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$
$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$
$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$
...
$\times [0.5*p(\text{"the"}|\theta_d) + 0.5*0.9]$

What if we increase $p(\theta_B)$?

What's the optimal solution now? $p(\text{"the"}|\theta_d) > 0.1$? or $p(\text{"the"}|\theta_d) < 0.1$?

**Behavior 2:** high frequency words get higher $p(w|\theta_d)$

# Summary

- General **behavior of a mixture model**:
  - Every component model attempts to assign high probabilities to highly frequent words in the data (to "**collaboratively maximize likelihood**")
  - Different component models tend to "bet" high probabilities on different words (to **avoid "competition"** or "waste of probability")
  - The probability of choosing each component "regulates" the collaboration/competition between the component models
- Fixing one component to a **background word distribution** (i.e., background language model):
  - Helps "get rid of background words" in other component
  - Is an example of **imposing a prior** on the model parameters (prior = one model must be exactly the same as the background LM)

# Probabilistic Topic Models: Expectation-Maximization Algorithm

## If we know which word is from which distribution…

$$p(w_i \mid \theta_d) = \frac{c(w_i, d')}{\sum_{w' \in V} c(w', d')}$$


P(w| θ_d)

p(w| θ_B)

| θ_d |
|---|
| text ? |
| mining ? |
| association ? |
| clustering ? |
| … |
| the ? |

| θ_B |
|---|
| the 0.03 |
| a 0.02 |
| is 0.015 |
| we 0.01 |
| food 0.003 |
| … |
| text 0.000006 |

$p(\theta_d)+(\theta_B)=1$

$P(\theta_d)=0.5$

Topic Choice

$P(\theta_B)=0.5$



## Given all the parameters, infer the distribution a word is from…

Is "text" more likely from $\theta_d$ or $\theta_B$ ?

From $\theta_d$ (Z=0)?

$p(\theta_d)p(\text{"text"} \mid \theta_d)$

From $\theta_B$ (Z=1)?

$p(\theta_B)p(\text{"text"} \mid \theta_B)$

P(w| θ_d)

p(w| θ_B)

| θ_d |
|---|
| text 0.04 |
| mining 0.035 |
| association 0.03 |
| clustering 0.005 |
| … |
| the 0.000001 |

| θ_B |
|---|
| the 0.03 |
| a 0.02 |
| is 0.015 |
| we 0.01 |
| food 0.003 |
| … |
| text 0.000006 |

$p(\theta_d)+p(\theta_B)=1$

$P(\theta_d)=0.5$

Topic Choice

$P(\theta_B)=0.5$

$$p(z=0 \mid w=\text{"text"}) = \frac{p(\theta_d)p(\text{"text"} \mid \theta_d)}{p(\theta_d)p(\text{"text"} \mid \theta_d) + p(\theta_B)p(\text{"text"} \mid \theta_B)}$$



## The Expectation-Maximization (EM) Algorithm

Hidden Variable: $z \in \{0, 1\}$

| | z |
|---|---|
| the | 1 |
| paper | 1 |
| presents | 1 |
| a | 1 |
| text | 0 |
| mining | 0 |
| algorithm | 0 |
| for | 1 |
| clustering | 0 |
| ... | ... |

Initialize $p(w|\theta_d)$ with random values.
Then iteratively improve it using E-step & M-step.
Stop when likelihood doesn't change.

E-step

$$p^{(n)}(z=0 \mid w) = \frac{p(\theta_d)p^{(n)}(w \mid \theta_d)}{p(\theta_d)p^{(n)}(w \mid \theta_d) + p(\theta_B)p(w \mid \theta_B)}$$

How likely w is from $\theta_d$

M-step

$$p^{(n+1)}(w \mid \theta_d) = \frac{c(w,d)p^{(n)}(z=0 \mid w)}{\sum_{w' \in V} c(w',d)p^{(n)}(z=0 \mid w')}$$



## EM Computation in Action

E-step

$$p^{(n)}(z=0 \mid w) = \frac{p(\theta_d)p^{(n)}(w \mid \theta_d)}{p(\theta_d)p^{(n)}(w \mid \theta_d) + p(\theta_B)p(w \mid \theta_B)}$$

M-step

$$p^{(n+1)}(w \mid \theta_d) = \frac{c(w,d)p^{(n)}(z=0 \mid w)}{\sum_{w' \in V} c(w',d)p^{(n)}(z=0 \mid w')}$$

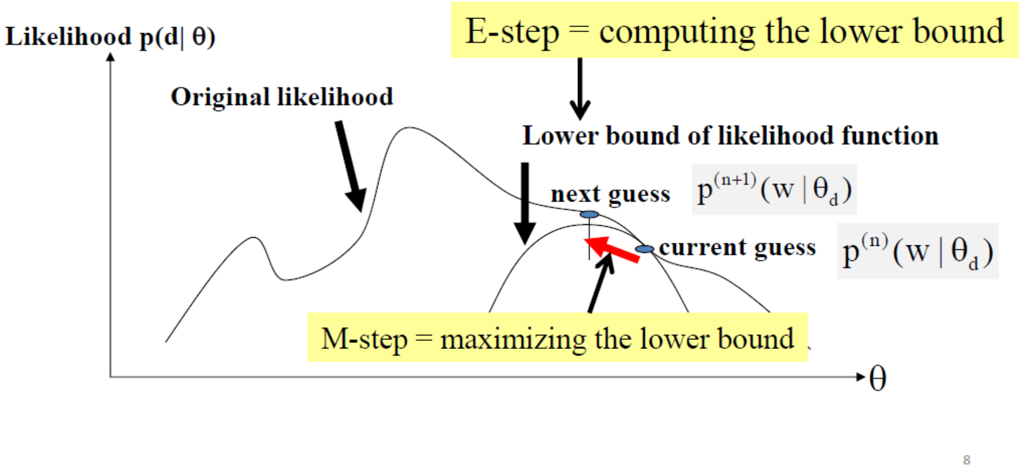
Assume $p(\theta_d)=p(\theta_B)= 0.5$ and $p(w|\theta_B)$ is known

| Word | # | $p(w\|\theta_B)$ | Iteration 1 | | Iteration 2 | | Iteration 3 | |
|---|---|---|---|---|---|---|---|---|
| | | | $P(w\|\theta)$ | $p(z=0\|w)$ | $P(w\|\theta)$ | $P(z=0\|w)$ | $P(w\|\theta)$ | $P(z=0\|w)$ |
| The | 4 | 0.5 | **0.25** | 0.33 | **0.20** | 0.29 | **0.18** | 0.26 |
| Paper | 2 | 0.3 | **0.25** | 0.45 | **0.14** | 0.32 | **0.10** | 0.25 |
| Text | 4 | 0.1 | **0.25** | 0.71 | **0.44** | 0.81 | **0.50** | 0.93 |
| Mining | 2 | 0.1 | **0.25** | 0.71 | **0.22** | 0.69 | **0.22** | 0.69 |
| Log-Likelihood | | | -16.96 | | -16.13 | | -16.02 | |

Likelihood increasing

"By products": Are they also useful?

# EM As Hill-Climbing ➔ Converge to Local Maximum




## Summary

- **Expectation-Maximization** (EM) algorithm
  - **General** algorithm for computing **ML estimate of mixture models**
  - Hill-climbing, so can only converge to a local maximum (depending on initial points)

- E-step: "augment" data by **predicting values of useful hidden variables**
- M-step: exploit the "augmented data" to **improve estimate of parameters** ("improve" is guaranteed in terms of likelihood)
- "Data augmentation" is probabilistic ➔ Split counts of events probabilistically

# Probabilistic Latent Semantic Analysis (PLSA)

# Document as a Sample of Mixed Topics

Topic $\theta_1$: government 0.3, response 0.2, ...

Topic $\theta_2$: city 0.2, new 0.1, orleans 0.05, ...

. . .

Topic $\theta_k$: donate 0.1, relief 0.05, help 0.02, ...

Background $\theta_B$: the 0.04, a 0.03, ...

Blog article about "Hurricane Katrina"

[ Criticism of government response to the hurricane primarily consisted of criticism of its response to the approach of the storm and its aftermath, specifically in the delayed response ] to the [ flooding of New Orleans. … 80% of the 1.3 million residents of the greater New Orleans metropolitan area evacuated ] …[ Over seventy countries pledged monetary donations or other assistance]. …

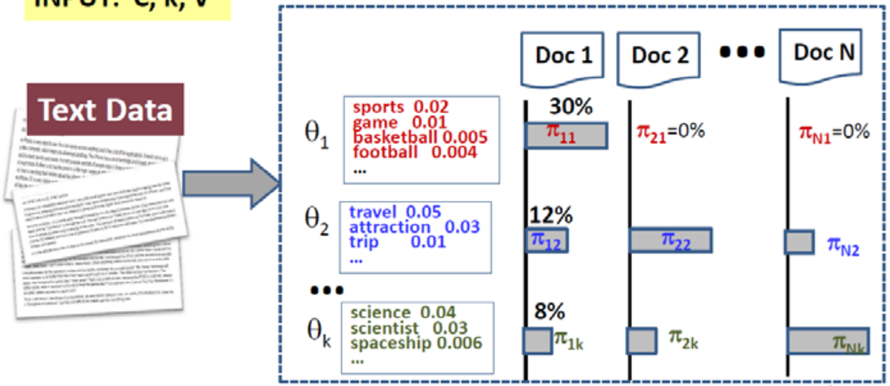
Many applications are possible if we can "decode" the topics in text…



# Mining Multiple Topics from Text

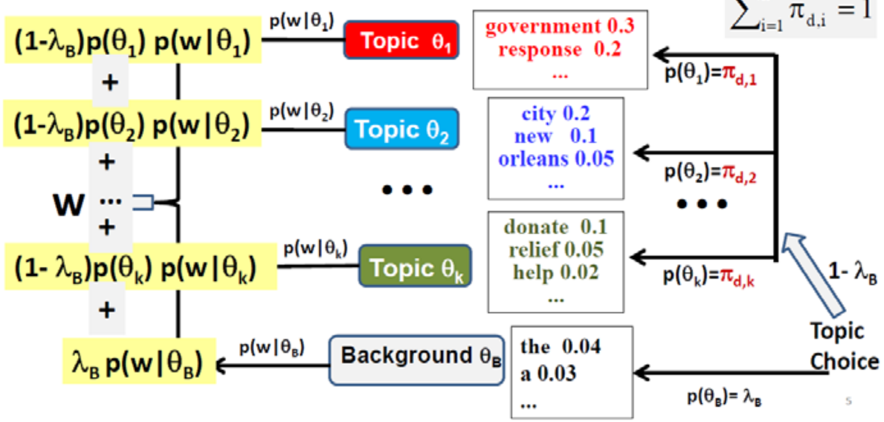
INPUT: C, k, V

OUTPUT: { $\theta_1$, ..., $\theta_k$ }, { $\pi_{i1}$, ..., $\pi_{ik}$ }






# Generating Text with Multiple Topics: p(w)=?




$$\sum_{i=1}^{k} \pi_{d,i} = 1$$



# Probabilistic Latent Semantic Analysis (PLSA)

Percentage of background words (known)

Background LM (known)

Coverage of topic $\theta_j$ in doc d

Prob. of word w in topic $\theta_j$

$$p_d(w) = \lambda_B p(w \mid \theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j}\, p(w \mid \theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w,d) \log[\lambda_B p(w \mid \theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j}\, p(w \mid \theta_j)]$$

$$\log p(C \mid \Lambda) = \sum_{d \in C}\sum_{w \in V} c(w,d) \log[\lambda_B p(w \mid \theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j}\, p(w \mid \theta_j)]$$

**Unknown Parameters: $\Lambda=(\{\pi_{d,j}\}, \{\theta_j\})$, j=1, …, k**

How many unknown parameters are there in total?

# ML Parameter Estimation

$$p_d(w) = \lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)$$

$$\log p(d) = \sum_{w \in V} c(w,d) \log[\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)]$$

$$\log p(C|\Lambda) = \sum_{d \in C}\sum_{w \in V} c(w,d) \log[\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j} p(w|\theta_j)]$$

**Constrained Optimization:** $\Lambda^* = \arg\max_{\Lambda} p(C|\Lambda)$

$\forall j \in [1,k], \sum_{i=1}^{M} p(w_i|\theta_j) = 1$ $\quad\quad$ $\forall d \in C, \sum_{j=1}^{k} \pi_{d,j} = 1$



# EM Algorithm for PLSA: E-Step

**Hidden Variable (=topic indicator):** $z_{d,w} \in \{B, 1, 2, ..., k\}$

Probability that **w in doc d** is generated from **topic** $\theta_j$

**Use of Bayes Rule**

$$p(z_{d,w} = j) = \frac{\pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)}{\sum_{j'=1}^{k} \pi_{d,j'}^{(n)} p^{(n)}(w|\theta_{j'})}$$

$$p(z_{d,w} = B) = \frac{\lambda_B p(w|\theta_B)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)}$$

Probability that **w in doc d** is generated from **background** $\theta_B$



# EM Algorithm for PLSA: M-Step

**Hidden Variable (=topic indicator):** $z_{d,w} \in \{B, 1, 2, ..., k\}$

**ML Estimate based on "allocated" word counts to topic** $\theta_j$

Re-estimated **probability** of **doc d** covering **topic** $\theta_j$

$$\pi_{d,j}^{(n+1)} = \frac{\sum_{w \in V} \boxed{c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)}}{\sum_{j'}\sum_{w \in V} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j')}$$

$$p^{(n+1)}(w|\theta_j) = \frac{\sum_{d \in C} \boxed{c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)}}{\sum_{w' \in V}\sum_{d \in C} c(w',d)(1-p(z_{d,w'}=B))p(z_{d,w'}=j)}$$

Re-estimated **probability** of **word w** for **topic** $\theta_j$



# Computation of the EM Algorithm

- Initialize all unknown parameters randomly
- Repeat until likelihood converges
  - E-step $p(z_{d,w} = j) \propto \pi_{d,j}^{(n)} p^{(n)}(w|\theta_j)$ $\quad$ $\sum_{j=1}^{k} p(z_{d,w} = j) = 1$

    $p(z_{d,w} = B) \propto \lambda_B p(w|\theta_B)$ $\quad$ What's the normalizer for this one?
  - M-step

$\pi_{d,j}^{(n+1)} \propto \sum_{w \in V} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)$ $\quad$ $\forall d \in C, \sum_{j=1}^{k} \pi_{d,j} = 1$

$p^{(n+1)}(w|\theta_j) \propto \sum_{d \in C} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)$ $\quad$ $\forall j \in [1,k], \sum_{w \in V} p(w|\theta_j) = 1$

**In general, accumulate counts, and then normalize**

# Summary

- PLSA = **mixture model** with k unigram LMs (k topics)
- Adding a **pre-determined background LM** helps discover discriminative topics
- ML estimate **"discovers" topical knowledge** from text data
  - k word distributions (**k topics**)
  - **proportion** of each topic in each document
- The output can enable many applications!
  - Clustering of terms and docs (treat each **topic as a cluster**)
  - Further associate topics with **different contexts** (e.g., time periods, locations, authors, sources, etc.)

# Latent Dirichlet Allocation (LDA)

# Extensions of PLSA

- PLSA with prior knowledge ➔ User-controlled PLSA
- PLSA as a generative model ➔ Latent Dirichlet Allocation (LDA)

# PLSA with Prior Knowledge

- **Users** may have expectations about **which topics to analyze**:
  – We expect to see **"retrieval models" as a topic** in IR
  – We want aspects such as "battery" and "memory" for opinions about a laptop
- **Users may know** what **topics** are (are NOT) **covered** in a doc
  – Tags = topics ➔ A <u>doc</u> can only be <u>generated</u> using <u>topics corresponding to the tags</u> assigned to the document
- We can **incorporate such knowledge** as **priors** of PLSA model

# Maximum a Posteriori (**MAP**) Estimate

$$\Lambda^* = \arg\max_{\Lambda} p(\Lambda)p(Data \mid \Lambda)$$

- We may use $p(\Lambda)$ to **encode all kinds of preferences and constraints**, e.g.,
  - $p(\Lambda)>0$ if and only if one topic is **precisely "background"**: $p(w|\theta_B)$
  - $p(\Lambda)>0$ if and only if for **a particular doc d**, $\pi_{d,3}=0$ and $\pi_{d,1}=1/2$
  - $p(\Lambda)$ favors a $\Lambda$ with topics that assign **high probabilities to some particular words**
- The **MAP** estimate (with **conjugate prior**) can be computed using a **similar EM algorithm** to the ML estimate **with smoothing to reflect prior preferences**

conjugate prior - сопряженное априорное распределение

# EM Algorithm with Conjugate Prior on $p(w| \theta_i)$

$$p(z_{d,w}=j) = \frac{\pi^{(n)}_{d,j} p^{(n)}(w|\theta_j)}{\sum_{j'=1}^{k} \pi^{(n)}_{d,j'} p^{(n)}(w|\theta_{j'})}$$

**Prior: $p(w|\theta'_j)$**

battery 0.5
life 0.5

$$p(z_{d,w}=B) = \frac{\lambda_B p(w|\theta_B)}{\lambda_B p(w|\theta_B) + (1-\lambda_B)\sum_{j=1}^{k} \pi^{(n)}_{d,j} p^{(n)}(w|\theta_j)}$$

$$\pi^{(n+1)}_{d,j} = \frac{\sum_{w\in V} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j)}{\sum_{j'}\sum_{w\in V} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j')}$$

Pseudo counts of w from prior θ'

$$p^{(n+1)}(w|\theta_j) = \frac{\sum_{d\in C} c(w,d)(1-p(z_{d,w}=B))p(z_{d,w}=j) + \mu p(w|\theta'_j)}{\sum_{w'\in V}\sum_{d\in C} c(w',d)(1-p(z_{d,w'}=B))p(z_{d,w'}=j) + \mu}$$

Sum of all pseudo counts

**What if $\mu=0$? What if $\mu=+\infty$?**

**We may also set any parameter to a constant (including 0) as needed**

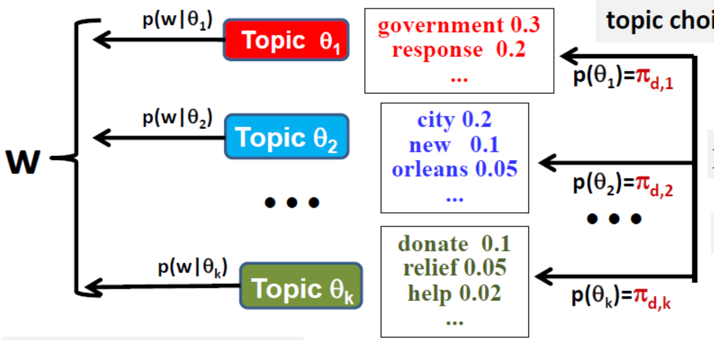
## Deficiency of PLSA

- **Not a generative model**
  - Can't compute probability of a new document
  - Heuristic workaround is possible, though
- Many parameters ➔ **high complexity** of models
  - Many local maxima
  - Prone to overfitting
- Not necessarily a problem for text mining (only interested in fitting the "training" documents)

## Latent Dirichlet Allocation (LDA)

- Makes PLSA a **generative model** by imposing a **Dirichlet prior** on the model parameters ➔
  - **LDA = Bayesian version of PLSA**
  - Parameters are regularized
- Can achieve the same goal as PLSA for text mining purposes
  - Topic coverage and topic word distributions can be **inferred using Bayesian inference**

# PLSA → LDA




Both word distributions and topic choices are free in PLSA

$\vec{\pi}_d = (\pi_{d,1},\ldots,\pi_{d,k})$

$p(\vec{\pi}_d) = \mathrm{Dirichlet}(\vec{\alpha})$

$\vec{\alpha} = (\alpha_1,\ldots,\alpha_k), \alpha_i > 0$

$\vec{\theta}_i = (p(w_1|\theta_i),\ldots,p(w_M|\theta_i))$

$p(\vec{\theta}_i) = \mathrm{Dirichlet}(\vec{\beta})$

$\vec{\beta} = (\beta_1,\ldots,\beta_M), \beta_i > 0$

LDA imposes a prior on both



# Likelihood Functions for PLSA vs. LDA

PLSA

$$p_d(w|\{\theta_j\},\{\pi_{d,j}\}) = \underline{\sum_{j=1}^{k} \pi_{d,j}\, p(w|\theta_j)}$$

Core assumption in all topic models

$$\log p(d|\{\theta_j\},\{\pi_{d,j}\}) = \sum_{w\in V} c(w,d) \log\Big[\sum_{j=1}^{k} \pi_{d,j}\, p(w|\theta_j)\Big]$$

$$\log p(C|\{\theta_j\},\{\pi_{d,j}\}) = \sum_{d\in C} \log p(d|\{\theta_j\},\{\pi_{d,j}\})$$

PLSA component

LDA

$$p_d(w|\{\theta_j\},\{\pi_{d,j}\}) = \underline{\sum_{j=1}^{k} \pi_{d,j}\, p(w|\theta_j)}$$

$$\log p(d|\vec{\alpha},\{\theta_j\}) = \int \boxed{\sum_{w\in V} c(w,d) \log\Big[\sum_{j=1}^{k} \pi_{d,j}\, p(w|\theta_j)\Big]}\, \underline{p(\vec{\pi}_d|\vec{\alpha})}\, d\vec{\pi}_d$$

$$\log p(C|\vec{\alpha},\vec{\beta}) = \int \sum_{d\in C} \log p(d|\vec{\alpha},\{\theta_j\}) \underline{\prod_{j=1}^{k} p(\theta_j|\vec{\beta})}\, d\theta_1 \ldots d\theta_k$$

Added by LDA

## Parameter Estimation and Inferences in LDA

- Parameters can be estimated using **ML estimator**

$$(\hat{\vec{\alpha}}, \hat{\vec{\beta}}) = \arg\max_{\vec{\alpha}, \vec{\beta}} \log p(C \mid \vec{\alpha}, \vec{\beta})$$

**How many parameters in LDA vs. PLSA?**

- However, **$\{\theta_j\}$ and $\{\pi_{d,j}\}$** must now be **computed** using posterior inference
  - Computationally **intractable** (*complex*)
  - Must resort to **approximate inference**
  - Many **different inference methods** are available

## Summary of Probabilistic Topic Models

- **Probabilistic topic models** provide a general principled way of **mining and analyzing topics** in text with many applications

- Basic task setup:
  - Input: Text data
  - Output: **k topics + proportions** of these topics covered **in each document**

- **PLSA** is the **basic topic model**, often adequate **for most applications**

- **LDA improves** over PLSA by imposing priors
  - Theoretically more appealing
  - Practically, LDA and PLSA **perform similarly** for many tasks