Probabilistic Machine Learning
Lecture 23
Tuning Inference Algorithms

Philipp Hennig

07 July 2020

Eberhard Karls Universität Tübingen
Faculty of Science
Department of Computer Science
Chair for the Methods of Machine Learning

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

#	date	content	Ex	#	date	content	Ex
1	20.04.	Introduction	1	14	09.06.	Generalized Linear Models	
2	21.04.	Reasoning under Uncertainty		15	15.06.	Exponential Families	8
3	27.04.	Continuous Variables	2	16	16.06.	Graphical Models	
4	28.04.	Monte Carlo		17	22.06.	Factor Graphs	9
5	04.05.	Markov Chain Monte Carlo	3	18	23.06.	The Sum-Product Algorithm	
6	05.05.	Gaussian Distributions		19	29.06.	Example: Modelling Topics	10
7	11.05.	Parametric Regression	4	20	30.06.	Mixture Models	
8	12.05.	Learning Representations		21	06.07.	EM	11
9	18.05.	Gaussian Processes	5	22	07.07.	Variational Inference	
10	19.05.	Understanding Kernels		23	13.07.	Tuning Inference Algorithms	12
11	26.05.	Gauss-Markov Models		24	14.07.	Kernel Topic Models	
12	25.05.	An Example for GP Regression	6	25	20.07.	Outlook	
13	08.06.	GP Classification	7	26	21.07.	Revision	

Designing a probabilistic machine learning method:

1. get the **data**
   1.1 try to collect as much meta-data as possible
2. build the **model**
   2.1 identify quantities and datastructures; assign names
   2.2 design a generative process (graphical model)
   2.3 assign (conditional) distributions to factors/arrows (use exponential families!)
3. design the **algorithm**
   3.1 consider conditional independence
   3.2 try standard methods for early experiments
   3.3 run unit-tests and sanity-checks
   3.4 identify bottlenecks, find customized approximations and refinements

The Toolbox

Framework:

$$\int p(x_1, x_2)\, dx_2 = p(x_1) \qquad p(x_1, x_2) = p(x_1 \mid x_2) p(x_2) \qquad p(x \mid y) = \frac{p(y \mid x) p(x)}{p(y)}$$

Modelling:

- graphical models
- Gaussian distributions
- (deep) learnt representations
- Kernels
- Markov Chains
- Exponential Families / Conjugate Priors
- Factor Graphs & Message Passing

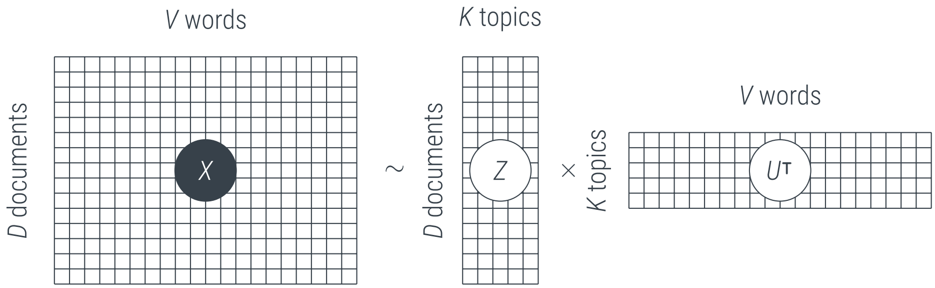
Computation:

- Monte Carlo
- Linear algebra / Gaussian inference
- maximum likelihood / MAP
- Laplace approximations
- EM / variational approximations

The Data
and an idea for a model

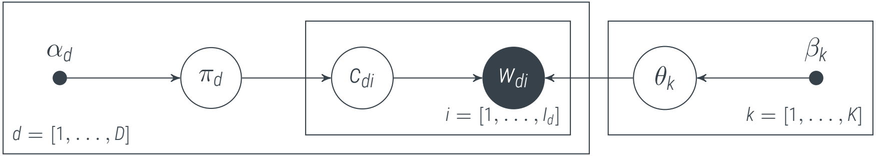
Making Assumptions

Our Data, our model



- a corpus of D documents
- each containing I_d words from a vocabulary of V words
- assumed to consist of K topics

The Model

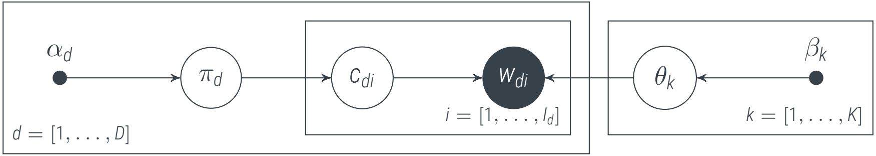


To draw I_d words $w_{di} \in [1, \dots, V]$ of document $d \in [1, \dots, D]$:

- Draw K topic distributions θ_k over V words from $\quad p(\Theta \mid \boldsymbol{\beta}) = \prod_{k=1}^{K} \mathcal{D}(\theta_k; \beta_k)$
- Draw D document distributions over K topics from $\quad p(\Pi \mid \boldsymbol{\alpha}) = \prod_{d=1}^{D} \mathcal{D}(\pi_d; \alpha_d)$
- Draw topic assignments c_{dik} of word w_{di} from $\quad p(C \mid \Pi) = \prod_{i,d,k} \pi_{dk}^{c_{dik}}$
- Draw word w_{di} from $\quad p(w_{di} = v \mid c_{di}, \Theta) = \prod_k \theta_{kv}^{c_{dik}}$

Useful notation: $n_{dkv} = \#\{i : w_{di} = v, c_{dik} = 1\}$. Write $n_{dk:} := [n_{dk1}, \dots, n_{dkV}]$ and $n_{dk\cdot} = \sum_v n_{dkv}$, etc.

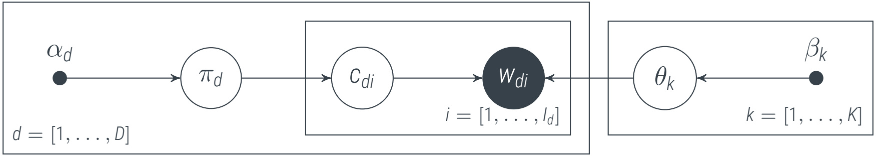
[Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003) JMLR 3, 993–1022]



$$p(C, \Pi, \Theta, W) = \underbrace{\left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d)\right)}_{p(\Pi|\boldsymbol{\alpha})} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \pi_{dk}^{c_{dik}}\right)\right)}_{p(C|\Pi)} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \theta_{kw_{di}}^{c_{dik}}\right)\right)}_{p(W|C,\Theta)} \cdot \underbrace{\left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k)\right)}_{p(\Theta|\boldsymbol{\beta})}$$

$$= \underbrace{\left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d)\right)}_{p(\Pi|\boldsymbol{\alpha})} \cdot \underbrace{\left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} (\pi_{dk}\theta_{kw_{di}})^{c_{dik}}\right)\right)}_{p(W,C|\Theta,\Pi)} \cdot \underbrace{\left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k)\right)}_{p(\Theta|\boldsymbol{\beta})}$$

$$= \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk}-1+n_{dk\cdot}}\right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv}-1+n_{\cdot kv}}\right)$$



$$p(C, \Pi, \Theta, W) = \left(\prod_{d=1}^{D} \mathcal{D}(\boldsymbol{\pi}_d; \boldsymbol{\alpha}_d)\right) \cdot \left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \pi_{dk}^{c_{dik}}\right)\right) \cdot \left(\prod_{d=1}^{D} \prod_{i=1}^{I_d} \left(\prod_{k=1}^{K} \theta_{kw_{di}}^{c_{dik}}\right)\right) \cdot \left(\prod_{k=1}^{K} \mathcal{D}(\boldsymbol{\theta}_k; \boldsymbol{\beta}_k)\right)$$

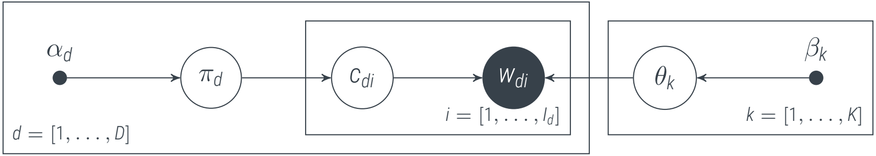
- If we had Π, Θ (which we don't), then the posterior $p(C \mid \Theta, \Pi, W)$ would be easy:

$$p(C \mid \Theta, \Pi, W) = \frac{p(W, C, \Theta, \Pi)}{\sum_C p(W, C, \Theta, \Pi)} = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \frac{\prod_{k=1}^{K} (\pi_{dk} \theta_{kw_{di}})^{c_{dik}}}{\sum_{k'} (\pi_{dk'} \theta_{k' w_{di}})}$$

- note that this conditional independence can easily be read off from the above graph!

Posteriors

Latent Dirichlet Allocation



$$p(C, \Pi, \Theta, W) = \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk}-1+n_{dk\cdot}} \right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv}-1+n_{\cdot kv}} \right)$$

- If we had C (which we don't), then the posterior $p(\Theta, \Pi \mid C, W)$ would be easy:

$$p(\Theta, \Pi \mid C, W) = \frac{p(C, W, \Pi, \Theta)}{\int p(\Theta, \Pi, C, W)\, d\Theta\, d\Pi} = \frac{\left(\prod_d \mathcal{D}(\pi_d; \alpha_d) \left(\prod_k \pi_{dk}^{n_{dk\cdot}}\right)\right) \left(\prod_k \mathcal{D}(\theta_k; \beta_k) \left(\prod_v \theta_{kv}^{n_{\cdot kv}}\right)\right)}{p(C, W)}$$

$$= \left(\prod_d \mathcal{D}(\pi_d; \alpha_{d:} + n_{d:\cdot}) \right) \left(\prod_k \mathcal{D}(\theta_k; \beta_{k:} + n_{\cdot k:}) \right)$$

- note that this conditional independence **can not** easily be read off from the above graph!

The Algorithms

A Gibbs Sampler

Simple Markov Chain Monte Carlo inference

Iterate between (recall $n_{dkv} = \#\{i : w_{di} = v, c_{ijk} = 1\}$)

$$\Theta \sim p(\Theta \mid C, W) = \prod_k \mathcal{D}(\theta_k; \beta_{k:} + n_{\cdot k:})$$

$$\Pi \sim p(\Pi \mid C, W) = \prod_d \mathcal{D}(\pi_d; \alpha_{d:} + n_{d:\cdot})$$

$$C \sim p(C \mid \Theta, \Pi, W) = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \frac{\prod_{k=1}^{K} (\pi_{dk}\theta_{kw_{di}})^{c_{dik}}}{\sum_{k'} (\pi_{dk'}\theta_{k'w_{di}})}$$

- This is *comparably* easy to implement because there are libraries for sampling from Dirichlet's, and discrete sampling is trivial. All we have to keep around are the counts n (which are sparse!) and Θ, Π (which are comparably small). Thanks to factorization, much can also be done in parallel!
- Unfortunately, this sampling scheme is relatively slow to move out of initialization, because z depends strongly on θ, π and vice versa.
- properly vectorizing the code is important for speed

Expectation Maximization

Iterative maximum likelihood (or maximum a posteriori)

To maximize $p(\Theta, \Pi \mid W)$, consider (where $\gamma_{dik} := \pi_{dk}\theta_{kw_{di}}$ and $\tilde{\gamma}_{dik} := \gamma_{dik} / \sum_{k'} \gamma_{dik'}$)

$$p(C \mid \Theta, \Pi, W) = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \frac{\prod_{k=1}^{K} (\pi_{dk}\theta_{kw_{di}})^{c_{dik}}}{\sum_{k'} (\pi_{dk'}\theta_{k'w_{di}})} = \prod_{d=1}^{D} \prod_{i=1}^{I_d} \prod_{k=1}^{K} \tilde{\gamma}_{dik}^{c_{dik}}$$

And **M**aximize the **E**xpected complete log posterior

$$\mathbb{E}_{p(C|\gamma)}[\log p(\Theta, \Pi \mid W, C)] = \sum_{d}^{D} \sum_{k}^{K} (\tilde{\gamma}_{d\cdot k} + \alpha_{dk}) \cdot \log \pi_{dk} + \sum_{k}^{K} \sum_{v}^{V} \left(\beta_{kv} + \sum_{d}^{D} \sum_{i}^{I_d} \mathbb{I}(w_{di} = v)\tilde{\gamma}_{dik} \right) \log \theta_{kv}$$

at

$$\pi_{dk} = \frac{\tilde{\gamma}_{d\cdot k} + \alpha_{dk}}{\tilde{\gamma}_{d\cdot\cdot} + \alpha_{d\cdot}} \quad \text{and} \quad \theta_{kv} = \frac{\beta_{kv} + \sum_{d,i} \mathbb{I}(w_{di} = v)\tilde{\gamma}_{dik}}{\beta_{k\cdot} + \sum_{v'} \sum_{d,i} \mathbb{I}(w_{di} = v')\tilde{\gamma}_{dik}}$$

and repeat.

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

To find the distribution q the minimizes, subject to $q(\Pi, \Theta, C) = q(\Pi, \Theta) \cdot q(C)$

$$D_{\mathrm{KL}}(q(\Pi, \Theta, C) \| p(\Pi, \Theta, C \mid W)) = \int q \log \frac{q}{p}\, dq$$

or, equivalently, maximizes the ELBO

$$\mathcal{L}(q) = \int q(C, \Pi, \Theta) \log \left(\frac{p(C, \Pi, \Theta, W)}{q(C, \Pi, \Theta)}\, dC\, d\Pi\, d\Theta \right)$$

set $\log q(C) = \mathbb{E}_{q(\Pi,\Theta)}(\log p(C, \Pi, \Theta, W)$ and vice versa, to get…

Variational Inference

iterative minimum KL divergence

[Blei, D. M., Ng, A. Y. & Jordan, M. I. (2003) JMLR 3, 993–1022]

$$q(\boldsymbol{\pi}_d) = \mathcal{D}\left(\boldsymbol{\pi}_d; \tilde{\alpha}_{dk} := \left[\alpha_{dk} + \sum_{i=1}^{I_d} \tilde{\gamma}_{dik}\right]_{k=1,\dots,K}\right) \qquad \forall d = 1, \dots, D$$

$$q(\boldsymbol{\theta}_k) = \mathcal{D}\left(\boldsymbol{\theta}_k; \tilde{\beta}_{kv} := \left[\beta_{kv} + \sum_{d}^{D} \sum_{i=1}^{I_d} \tilde{\gamma}_{dik}\mathbb{I}(w_{di} = v)\right]_{v=1,\dots,V}\right) \qquad \forall k = 1, \dots, K$$

$$q(\mathbf{c}_{di}) = \prod_k \tilde{\gamma}_{dik}^{c_{dik}}, \qquad \forall d \quad i = 1, \dots, I_d$$

where $\tilde{\gamma}_{dik} = \gamma_{dik} / \sum_k \gamma_{dik}$ and (note that $\sum_k \tilde{\alpha}_{dk} = \text{const.}$)

$$\gamma_{dik} = \exp\left(\mathbb{E}_{q(\pi_{dk})}(\log \pi_{dk}) + \mathbb{E}_{q(\theta_{di})}(\log \theta_{kw_{di}})\right) = \exp\left(\digamma(\tilde{\alpha}_{jk}) + \digamma(\tilde{\beta}_{kw_{di}}) - \digamma\left(\sum_v \tilde{\beta}_{kv}\right)\right)$$

and repeat

An idea.

- Consider the exponential family $p_w(x \mid w) = \exp\left[\phi(x)^\intercal w - \log Z(w)\right]$
- its conjugate prior is the exponential family $F(\alpha, \nu) = \int \exp(\alpha^\intercal w - \nu^\intercal \log Z(w))\, dw$

$$p_\alpha(w \mid \alpha, \nu) = \exp\left[\begin{pmatrix} w \\ -\log Z(w) \end{pmatrix}^\intercal \begin{pmatrix} \alpha \\ \nu \end{pmatrix} - \log F(\alpha, \nu)\right]$$

because $p_\alpha(w \mid \alpha, \nu) \prod_{i=1}^{n} p_w(x_i \mid w) \propto p_\alpha\left(w \,\middle|\, \alpha + \sum_i \phi(x_i), \nu + n\right)$

- and the predictive is

$$p(x) = \int p_w(x \mid w) p_\alpha(w \mid \alpha, \nu)\, dw = \int e^{(\phi(x)+\alpha)^\intercal w - (\nu+1)\log Z(w) - \log F(\alpha,\nu)}\, dw$$
$$= \frac{F(\phi(x) + \alpha, \nu + 1)}{F(\alpha, \nu)}$$

Exponential Families, among other things (see also last lecture) provide **conjugate priors** for standard distributions (Lectures 2,15)

- Consider the exponential family $p(c \mid \pi) = \exp\left[c^\intercal(\log \pi) - \log \sum_k \pi_k\right]$
- its conjugate prior is the exponential family $B(\alpha) = \int \exp(\alpha^\intercal \log \pi - \nu \cdot 0)\, d\pi$

$$\mathcal{D}(\pi \mid \alpha) = \exp\left[\log \pi^\intercal \alpha - \log B(\alpha)\right]$$

$$\text{because } \mathcal{D}(\pi \mid \alpha) \prod_{i=1}^{n} \pi^{c_i} \propto \mathcal{D}\left(\pi \mid \alpha + \sum_i c_i\right)$$

- and the predictive is

$$p(c) = \int p(c \mid \pi)\mathcal{D}(\pi \mid \alpha)\, d\pi = \int e^{(c+\alpha)^\intercal(\log \pi) + \log B(\alpha)}\, d\pi = \frac{B(c+\alpha)}{B(\alpha)}$$

Exponential Families, among other things (see also last lecture) provide **conjugate priors** for standard distributions (Lectures 2,15)

Recall $\Gamma(x+1) = x \cdot \Gamma(x) \ \forall x \in \mathbb{R}_+$

$$p(C, \Pi, \Theta, W) = \left(\prod_{d=1}^{D} \frac{\Gamma(\sum_k \alpha_{dk})}{\prod_k \Gamma(\alpha_{dk})} \prod_{k=1}^{K} \pi_{dk}^{\alpha_{dk}-1+n_{dk\cdot}} \right) \cdot \left(\prod_{k=1}^{K} \frac{\Gamma(\sum_v \beta_{kv})}{\prod_v \Gamma(\beta_{kv})} \prod_{v=1}^{V} \theta_{kv}^{\beta_{kv}-1+n_{\cdot kv}} \right)$$

$$= \left(\prod_{d=1}^{D} \frac{B(\alpha_d + n_{d:\cdot})}{B(\alpha_d)} \mathcal{D}(\pi_d; \alpha_d + n_{d:\cdot}) \right) \cdot \left(\prod_{k=1}^{K} \frac{B(\beta_k + n_{\cdot k:})}{B(\beta_k)} \mathcal{D}(\theta_k; \beta_k + n_{\cdot k:}) \right)$$

$$p(C, W) = \left(\prod_{d=1}^{D} \frac{B(\alpha_d + n_{d:\cdot})}{B(\alpha_d)} \right) \cdot \left(\prod_{k=1}^{K} \frac{B(\beta_k + n_{\cdot k:})}{B(\beta_k)} \right)$$

$$= \left(\prod_{d} \frac{\Gamma(\sum_{k'} \alpha_{dk'})}{\Gamma(\sum_{k'} \alpha_{dk'} + n_{dk'\cdot})} \prod_k \frac{\Gamma(\alpha_{dk} + n_{dk\cdot})}{\Gamma(\alpha_{dk})} \right) \left(\prod_{k} \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv} + n_{\cdot kv})}{\Gamma(\beta_{kv})} \right)$$

$$p(c_{dik} = 1 \mid C^{\setminus di}, W) = \frac{(\alpha_{dk} + n_{dk\cdot}^{\setminus di})(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di})(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di})^{-1}}{\sum_{k'} (\alpha_{dk'} + n_{dk'\cdot}^{\setminus di}) \cdot \sum_{w'} (\beta_{kw'} + n_{\cdot kw'}^{\setminus di}) \cdot \sum_{v'} (\beta_{kv'} + n_{\cdot kv'}^{\setminus di})^{-1}}$$

A Collapsed Gibbs Sampler for LDA

It pays off to look closely at the math!

T. L. Griffiths & M. Steyvers, *Finding scientific topics*, PNAS **101**/1 (4/2004), 5228–5235

$$p(C, W) = \left(\prod_d \frac{\Gamma(\sum_k \alpha_{dk})}{\Gamma(\sum_k \alpha_{dk} + n_{dk\cdot})} \prod_k \frac{\Gamma(\alpha_{dk} + n_{dk\cdot})}{\Gamma(\alpha_{dk})} \right) \left(\prod_k \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv} + n_{\cdot kv})}{\Gamma(\beta_{kv})} \right)$$

A **collapsed** sampling method can converge much faster by eliminating the latent variables that mediate between individual data.

procedure LDA(W, α, β)
- $\gamma_{dkv} \leftarrow 0 \;\forall d, k, v$ // initialize counts
- **while** true **do**
  - **for** $d = 1, \dots, D; i = 1, \dots, I_d$ **do** // can be parallelized
    - $c_{di} \propto (\alpha_{dk} + n_{dk\cdot}^{\backslash di})(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\backslash di})(\sum_v \beta_{kv} + n_{\cdot kv}^{\backslash di})^{-1}$ // sample assignment
    - $n \leftarrow \textsc{UpdateCounts}(c_{di})$ // update counts (check whether first pass or repeat)
  - **end for**
- **end while**

end procedure

Collapsed Sampling is quite efficient

The Mean Field argument [figure: T. L. Griffiths & M. Steyvers, *Finding scientific topics*, PNAS **101**/1 (4/2004), 5228–5235]

Thomas Griffiths

image: Princeton U

Mark Steyvers

image: UC Irvine

The collapsed sampler operates on the **mean field**

$$p(C \mid W) = \int p(C \mid \Theta, \Pi, W) p(\Theta, \Pi \mid W) \, d\Theta \, d\Pi$$

The *expected* value of the variables Θ, Π that mediate between the "particles" (words). This works well because each word's topic is approximately independent of all individual other words' topics (but together they create the whole thing).

- Deriving our variational bound, we previously imposed the factorization

$$q(\Pi, \Theta, C) = q(\Pi, \Theta) \cdot q(C), \quad \text{but can we get away with less? Like,}$$
$$q(\Pi, \Theta, C) = q(\Theta, \Pi \mid C) \cdot q(C)$$

- Note $p(C, \Theta, \Pi \mid W) = p(\Theta, \Pi \mid C, W)p(C \mid W)$. So when we minimize

$$\begin{aligned}
D_{\mathrm{KL}}(q(\Pi, \Theta, C) \| p(\Pi, \Theta, C \mid W)) &= \int q(\Pi, \Theta \mid C)q(C) \log\left(\frac{q(\Pi, \Theta | C)q(C)}{p(\Pi, \Theta \mid C, W)p(C \mid W)}\right) dC \, d\Pi \, d\Theta \\
&= \int q(\Pi, \Theta \mid C)q(C) \left[\log\left(\frac{q(\Pi, \Theta \mid C)}{p(\Pi, \Theta \mid C, W)}\right) + \log\left(\frac{q(C)}{p(C \mid W)}\right)\right] dC \, d\Pi \, d\Theta \\
&= D_{\mathrm{KL}}(q(\Pi, \Theta \mid C) \| p(\Pi, \Theta \mid C, W)) + D_{\mathrm{KL}}(q(C) \| p(C \mid W))
\end{aligned}$$

we will just get $q(\Theta, \Pi) = p(\Theta, \Pi \mid C, W)$ and the bound will be *tight* in Π, Θ.

A Collapsed Variational Bound

Why don't we use the mean field in our variational bound?

Yee Whye Teh, David Newman & Max Welling, NeurIPS 2007

$$p(C,W) = \left(\prod_d \frac{\Gamma(\sum_k \alpha_{dk})}{\Gamma(\sum_k \alpha_{dk} + n_{dk\cdot})} \prod_k \frac{\Gamma(\alpha_{dk} + n_{dk\cdot})}{\Gamma(\alpha_{dk})}\right) \left(\prod_k \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv} + n_{\cdot kv})}{\Gamma(\beta_{kv})}\right)$$

- The remaining **collapsed variational bound** (ELBO) becomes

$$\mathcal{L}(q) = \int q(C) \log p(C,W)\, dC + \mathbb{H}(q(C))$$

- because we make strictly less assumptions about q than before, we will get a strictly better approximation to the true posterior!
- The bound is maximized for c_{di} if

$$\log q(c_{di}) = \mathbb{E}_{q(C^{\setminus di})}(\log p(C,W)) + \text{const.}$$

Constructing the Algorithm

Why didn't we do this earlier?

- Note that $c_{di} \in \{0;1\}^K$ and $\sum_k c_{dik} = 1$. So $q(c_{di}) = \prod_k \gamma_{dik}$ with $\sum_k \gamma_{dik} = 1$
- Also: $\Gamma(\alpha + n) = \prod_{\ell=0}^{n-1}(\alpha + \ell)$, thus $\log \Gamma(\alpha + n) = \sum_{\ell=0}^{n-1} \log(\alpha + \ell)$

$$p(C, W) = \left(\prod_d \frac{\Gamma(\sum_k \alpha_{dk})}{\Gamma(\sum_k \alpha_{dk} + n_{dk\cdot})} \prod_k \frac{\Gamma(\alpha_{dk} + n_{dk\cdot})}{\Gamma(\alpha_{dk})}\right) \left(\prod_k \frac{\Gamma(\sum_v \beta_{kv})}{\Gamma(\sum_v \beta_{kv} + n_{\cdot kv})} \prod_v \frac{\Gamma(\beta_{kv} + n_{\cdot kv})}{\Gamma(\beta_{kv})}\right)$$

$$\begin{aligned}
\log q(c_{di}) &= \mathbb{E}_{q(C^{\setminus di})}(\log p(C, W)) + \text{const.} \\
\log \gamma_{dik} &= \log q(c_{dik} = 1) \\
&= \mathbb{E}_{q(C^{\setminus di})}\left[\log \Gamma(\alpha_{dk} + n_{dk\cdot}) + \log \Gamma(\beta_{kw_{di}} + n_{\cdot kw_{di}}) - \log \Gamma\left(\sum_v \beta_{kv} + n_{\cdot kv}\right)\right] + \text{const.} \\
&= \mathbb{E}_{q(C^{\setminus di})}\left[\log(\alpha_{dk} + n_{dk\cdot}^{\setminus di}) + \log(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di}) - \log\left(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di}\right)\right] + \text{const.}
\end{aligned}$$

(note all terms in $p(C, W)$ that don't involve c_{dik} can be moved into the constant, as can all sums over k. We can also *add* terms to const., such as $\sum_{\ell=0}^{n^{\setminus di}-1} \log(\alpha + \ell)$, effectively cancelling terms in $\log \Gamma$)

A Complication

Ah, that's why!

$$\gamma_{dik} \propto \exp\left(\mathbb{E}_{q(C^{\setminus di})}\left[\log(\alpha_{dk} + n_{dk\cdot}^{\setminus di}) + \log(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di}) - \log\left(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di}\right)\right]\right)$$

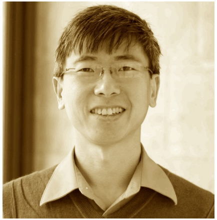
- Under $q(C) = \prod_{di} c_{di}$, the counts $n_{dk\cdot}$ are sums of independent Bernoulli variables (i.e. they have a **multinomial** distribution). Computing their expected logarithm is tricky ($\mathcal{O}(n_{d\cdot\cdot}^2)$):

$$\mathbb{H}(q(n_{dk\cdot})) = \mathbb{E}[\log n_{dk\cdot}] = -\log(I_d!) - I_d \sum_k^K \gamma_{dk\cdot} \log(\gamma_{dk\cdot}) + \sum_{k=1}^K \sum_{n_{dk\cdot}=1}^{I_d} \binom{I_d}{n_{dk\cdot}} \gamma_{dk\cdot}^{n_{dk\cdot}} (1 - \gamma_{dk\cdot})^{I_d - n_{dk\cdot}} \log(n_{dk\cdot}!)$$

- That's likely why the original paper (and `scikit-learn`) don't do this.

If arithmetic doesn't work, try creativity!

Yee Whye Teh, David Newman & Max Welling, NeurIPS 2007



Yee Whye Teh

image: Oxford U

Max Welling

image: U v Amsterdam

$$\gamma_{dik} \propto \exp\left(\mathbb{E}_{q(c^{\setminus di})}\left[\log(\alpha_{dk} + n_{dk\cdot}^{\setminus di}) + \log(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di}) - \log\left(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di}\right)\right]\right)$$

Statistics for the rescue

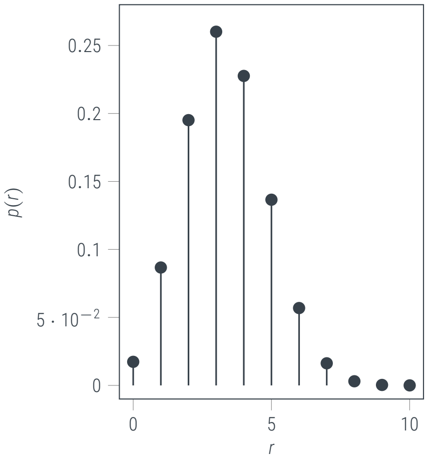
recall Lecture 3

Yee Whye Teh, David Newman & Max Welling, NeurIPS 2007

The probability measure of $R = \sum_i^N x_i$ with discrete x_i of probablity f is

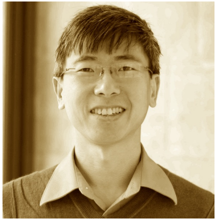
$$
\begin{aligned}
P(R = r \mid f, N) &= \frac{N!}{(N-r)! \cdot r!} \cdot f^r \cdot (1-f)^{N-r} \\
&= \binom{N}{r} \cdot f^r \cdot (1-f)^{N-r} \\
&\approx \mathcal{N}(r; Nr, Nr(1-r))
\end{aligned}
$$

$f = 1/3, N = 10$



If arithmetic doesn't work, try creativity!

Yee Whye Teh, David Newman & Max Welling, NeurIPS 2007



Yee Whye Teh

image: Oxford U

Max Welling

image: U v Amsterdam

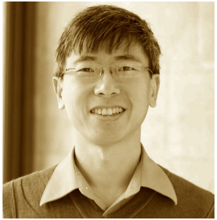
but the CLT applies! So a Gaussian approximation should be good:

$$p(n_{dk\cdot}^{\setminus di}) \approx \mathcal{N}(n_{dk\cdot}^{\setminus di}; \mathbb{E}_q[n_{dk\cdot}^{\setminus di}], \operatorname{var}_q[n_{dk\cdot}^{\setminus di}]) \quad \text{with} \quad \mathbb{E}_q[n_{dk\cdot}^{\setminus di}] = \sum_{j \neq i} \gamma_{dkj}, \quad \operatorname{var}_q[n_{dk\cdot}^{\setminus di}] = \sum_{j \neq i} \gamma_{dkj}(1 - \gamma_{dkj})$$

If arithmetic doesn't work, try creativity!

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Yee Whye Teh, David Newman & Max Welling, NeurIPS 2007



Yee Whye Teh

image: Oxford U

Max Welling

image: U v Amsterdam

$$\mathbb{E}_q[\log(\alpha_{dk} + n_{dk\cdot}^{\backslash di})] \approx \log(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\backslash di}]) - \frac{\operatorname{var}_q[n_{dk\cdot}^{\backslash di}]}{2(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\backslash di}])^2}$$

$$\gamma_{dik} \propto \exp\left(\mathbb{E}_{q(C^{\setminus di})}\left[\log(\alpha_{dk} + n_{dk\cdot}^{\setminus di}) + \log(\beta_{kw_{di}} + n_{\cdot kw_{di}}^{\setminus di}) - \log\left(\sum_v \beta_{kv} + n_{\cdot kv}^{\setminus di}\right)\right]\right)$$

$$\mathbb{E}_q[\log(\alpha_{dk} + n_{dk\cdot}^{\setminus di})] \approx \log(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\setminus di}]) - \frac{\mathrm{var}_q[n_{dk\cdot}^{\setminus di}]}{2(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\setminus di}])^2}$$

$$\gamma_{dik} \propto (\alpha_{dk} + \mathbb{E}[n_{dk\cdot}^{\setminus di}])(\beta_{kw_{di}} + \mathbb{E}[n_{\cdot kw_{di}}^{\setminus di}])\left(\sum_v \beta_{kv} + \mathbb{E}[n_{\cdot kv}^{\setminus di}]\right)^{-1}$$

$$\cdot \exp\left(-\frac{\mathrm{var}_q[n_{dk\cdot}^{\setminus di}]}{2(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\setminus di}])^2} - \frac{\mathrm{var}_q[n_{\cdot kw_{di}}^{\setminus di}]}{2(\beta_{kw_{di}} + \mathbb{E}_q[n_{\cdot kw_{di}}^{\setminus di}])^2} + \frac{\mathrm{var}_q[n_{\cdot k\cdot}^{\setminus di}]}{2(\sum_v \beta_{kv} + \mathbb{E}_q[n_{\cdot kv}^{\setminus di}])^2}\right)$$

Complexity

The algorithm

$$\gamma_{dik} \propto (\alpha_{dk} + \mathbb{E}[n_{dk\cdot}^{\setminus di}])(\beta_{kw_{di}} + \mathbb{E}[n_{\cdot kw_{di}}^{\setminus di}]) \left(\sum_v \beta_{kv} + \mathbb{E}[n_{\cdot kv}^{\setminus di}]\right)^{-1}$$

$$\cdot \exp\left(-\frac{\mathrm{var}_q[n_{dk\cdot}^{\setminus di}]}{2(\alpha_{dk} + \mathbb{E}_q[n_{dk\cdot}^{\setminus di}])^2} - \frac{\mathrm{var}_q[n_{\cdot kw_{di}}^{\setminus di}]}{2(\beta_{kw_{di}} + \mathbb{E}_q[n_{\cdot kw_{di}}^{\setminus di}])^2} + \frac{\mathrm{var}_q[n_{\cdot k\cdot}^{\setminus di}]}{2(\sum_v \beta_{kv} + \mathbb{E}_q[n_{\cdot kv}^{\setminus di}])^2}\right)$$

Note that γ_{dik} doesn't depend on $i \in 1, \dots, I_d$, it's the same for all w_{di} in d with $w_{di} = v$!

- memory requirement: $\mathcal{O}(DKV)$, since we have to store γ_{dik} for each value of $i \in 1, \dots, V$ and
  - $\mathbb{E}[n_{dk\cdot}], \mathrm{var}[n_{dk\cdot}] \in \mathbb{R}^{D \times K}$
  - $\mathbb{E}[n_{\cdot kv}], \mathrm{var}[n_{\cdot kv}] \in \mathbb{R}^{K \times V}$
  - $\mathbb{E}[n_{\cdot k\cdot}], \mathrm{var}[n_{\cdot k\cdot}] \in \mathbb{R}^{K}$
- computational complexity: $\mathcal{O}(DKV)$ We can loop over V rather than I_d (good for long documents!) Often, a document will be sparse in V, so iteration cost can be much lower.

Designing a probabilistic machine learning method:

1. get the **data**
   1.1 try to collect as much meta-data as possible
2. build the **model**
   2.1 identify quantities and datastructures; assign names
   2.2 design a generative process (graphical model)
   2.3 assign (conditional) distributions to factors/arrows (use exponential families!)
3. design the **algorithm**
   3.1 consider conditional independence
   3.2 try standard methods for early experiments
   3.3 run unit-tests and sanity-checks
   3.4 identify bottlenecks, find customized approximations and refinements

Simple, generic algorithms or toolboxes can give an idea of a model's efficacy.
To make an efficient product, you might have to build your own approximation, and tune it.