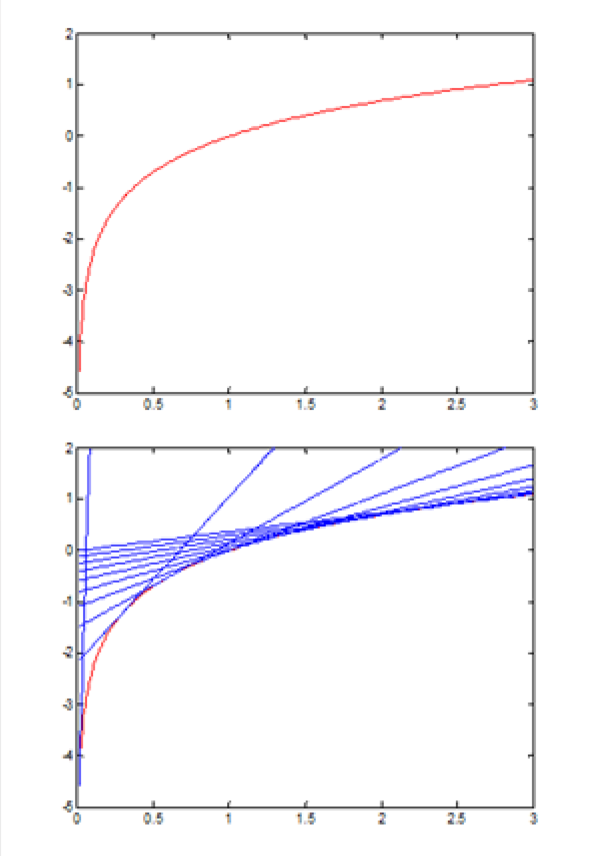
Variational Inference

Il-Chul Moon
Department of Industrial and Systems Engineering
KAIST
icmoon@kaist.ac.kr

VARIATIONAL APPROXIMATION

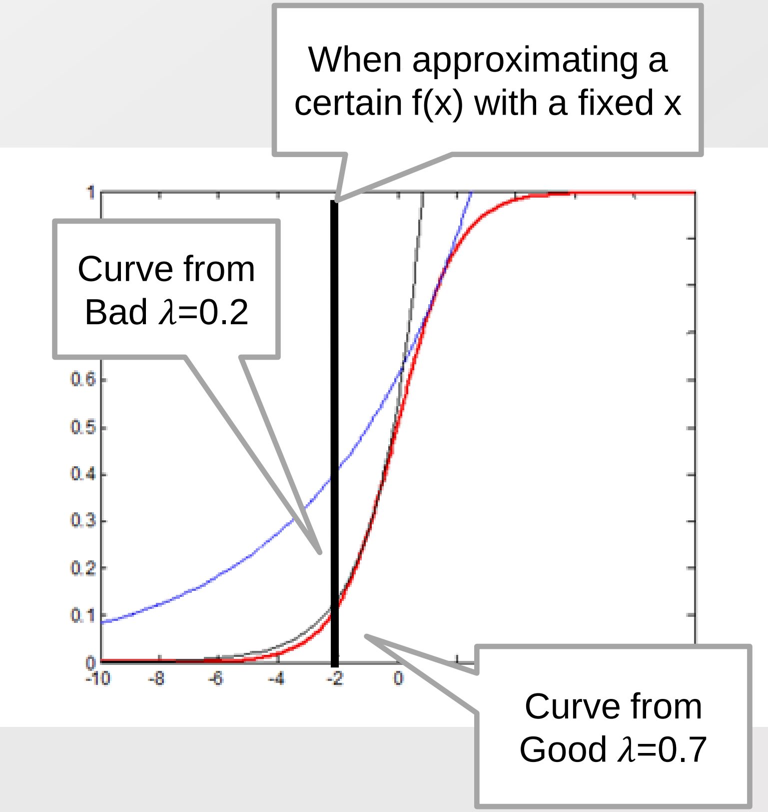
Variational Transform

- Let’s imagine drawing a log function
  - $y = \ln(x)$
- This is a typical non-linear function
  - Which is often complex and not desired
- How about transforming the function into a simpler form?
  - Preferably, a linear function…
- How about this?
  - $y = min_x\{\lambda x+b-\ln x\}$
    - $\frac{d}{dx}(\lambda x+b-\ln x)=0$
    - $\lambda = \frac{1}{x}$
    - Concave function!
- The result of the transform
  - Now, a linear function approximating the log function
  - We have a new floating parameter to optimize



Variational Transform on Logistic Function

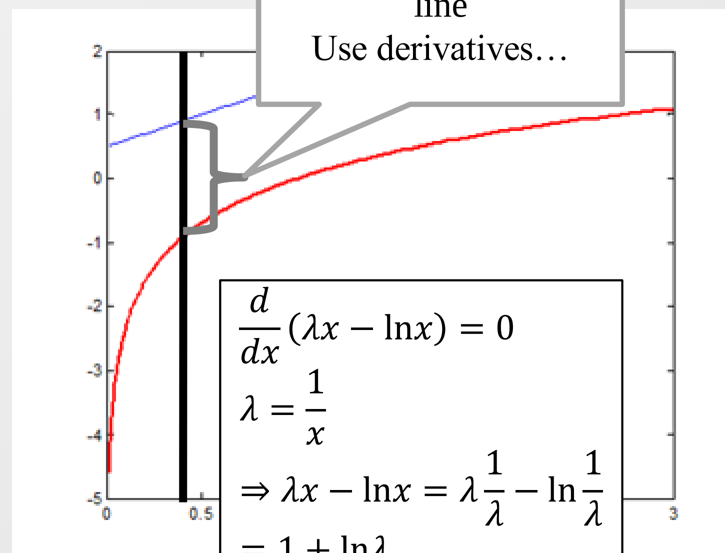
- Similar idea, more useful function than the log function
  - Logistic function
  - $f(x) = \frac{1}{1+e^{-x}}$
  - Neither concave nor convex
  - Turn it into a log-concave
  - $g(x) = -\ln(1 + e^{-x})$
- How about this?
  - $g(x) = min_{\lambda}\{\lambda x - H(\lambda)\}$
    - $H(\lambda) = -\lambda\ln\lambda - (1-\lambda)\ln(1-\lambda)$
  - $f(x) = min_{\lambda}\{e^{\lambda x - H(\lambda)}\}$
- Similarly…
  - We now have a linear function
  - Also a floating parameter to optimize by x



Convex Duality

- Systematic variational transform?
  - Utilize the convex duality
- Concave function f(x), such as log function
  - Can be represented via a conjugate or dual function as follows
  - Remember that if f(x) is not a concave function
    - You can always use the log-concave function
      - Transform using the log function
      - Re-transform using the exp function
- $f(x) = min_{\lambda}\{\lambda^T x - f^*(\lambda)\}$
  $\Leftrightarrow f^*(\lambda) = min_x\{\lambda^T x - f(x)\}$

Minimization, or optimization, through finding the tangent line
Use derivatives…

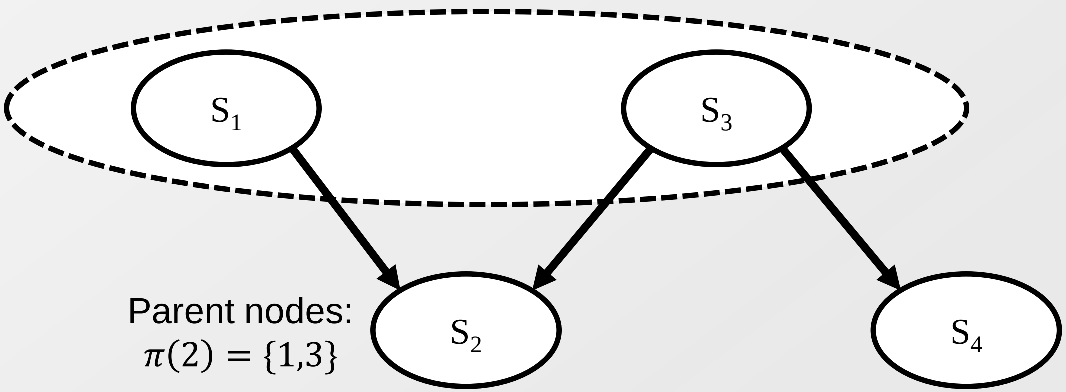


$$\frac{d}{dx}(\lambda x - \ln x) = 0$$
$$\lambda = \frac{1}{x}$$
$$\Rightarrow \lambda x - \ln x = \lambda\frac{1}{\lambda} - \ln\frac{1}{\lambda}$$
$$= 1 + \ln\lambda$$

Dual function or Conjugate function

Applying to Probability Function

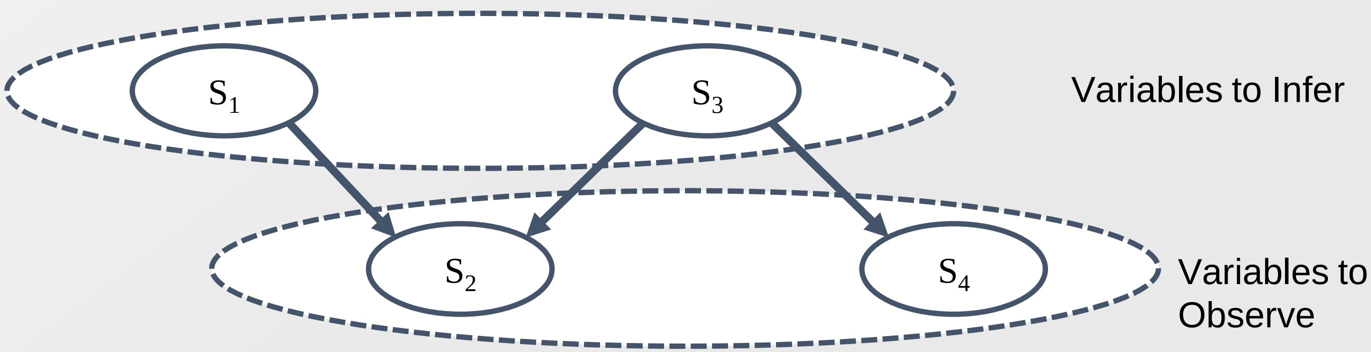
- Probability distribution function is a function, too
  - Just not a transformation to a linear function
  - Probability distribution function has its own characteristics
  - $f(x) = min_{\lambda}\{\lambda^T x - f^*(\lambda)\}$
  - $P(S) = \prod_i P(S_i|S_{\pi(i)}) = min_{\lambda} \prod_i P^U(S_i|S_{\pi(i)}, \lambda^U{}_i)$
  - $P(S) = \prod_i P(S_i|S_{\pi(i)}) \leq \prod_i P^U(S_i|S_{\pi(i)}, \lambda^U{}_i)$



$$P(S) = P(S_1)P(S_2|S_1,S_3)P(S_3)P(S_4|S_3) = \prod_i P(S_i|S_{\pi(i)})$$

Variables of E and H

- Evidence = E, Hypothesis = H
  - E is observed, fixed, and hard fact
  - H is estimated, inferred, and floating
- E and H are exclusive, and the union of E and H is the complete set of variables
  - $E \cap H = \phi, E \cup H = S$
  - $P(E) = \sum_H P(H,E) = \sum_H P(S) = \sum_H \prod_i P(S_i|S_{\pi(i)}) \le \sum_H \prod_i P^U\left(S_i\middle|S_{\pi(i)}, \lambda^U{}_i\right)$
- P(H|E) = P(H,E) / P(E)
  - This is what we need to know.
  - With the variational inference, P(E) is approximated



Setting the Minimum Criteria

- $\ln P(E) = \ln \sum_H P(H,E) = \ln \sum_H Q(H|E) \frac{P(H,E)}{Q(H|E)}$
- Since, log is a concave function
- $\ln \sum_H Q(H|E) \frac{P(H,E)}{Q(H|E)}$

$$\geq \sum_H Q(H|E) \ln \left[\frac{P(H,E)}{Q(H|E)} \right]$$

$$= \sum_H Q(H|E) \ln P(H,E) - Q(H|E) \ln Q(H|E)$$

$$= \sum_H Q(H|E) \{\ln P(E|H) + \ln P(H)\} - Q(H|E) \ln Q(H|E)$$

$$= \sum_H Q(H|E) \ln P(E|H) - Q(H|E) \frac{\ln Q(H|E)}{\ln P(H)}$$

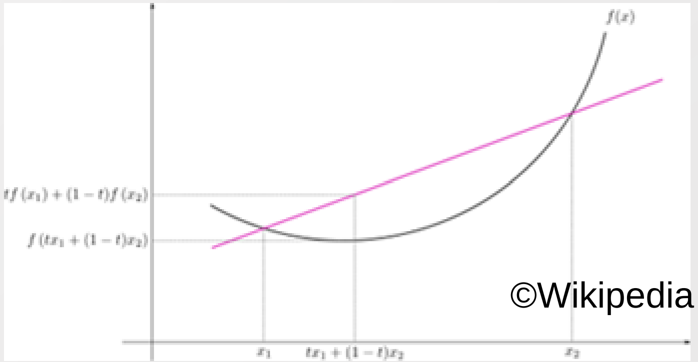
$$= E_{Q(H|E)} \ln P(E|H) - KL(Q(H|E) \parallel P(H))$$

- Using the Jensen's Inequality
- The right hand side is well known function in the statistics community
  - KL divergence

$$KL(Q \| P) = -\sum_i Q(i) \ln \left[\frac{P(i)}{Q(i)} \right]$$

<u>Minimizing KL Divergence → Finding the true ln $P(E)$</u>

Jensen's Inequality



©Wikipedia

When φ(x) is concave

$$\varphi\left(\frac{\sum a_i x_i}{\sum a_j}\right) \geq \frac{\sum a_i \varphi(x_i)}{\sum a_j}$$

When φ(x) is convex

$$\varphi\left(\frac{\sum a_i x_i}{\sum a_j}\right) \leq \frac{\sum a_i \varphi(x_i)}{\sum a_j}$$

Optimizing the Lower Bound

- $\ln P(E|\theta) \geq \sum_H Q(H|E,\lambda)\ln P(H,E|\theta) - Q(H|E,\lambda)\ln Q(H|E,\lambda)$
  $= \sum_H Q(H|E)\ln P(E|H,\theta) - Q(H|E)\ln\frac{Q(H|E)}{P(H|\theta)}$
  $= E_{Q(H|E)}\ln P(E|H) - KL(Q(H|E) \parallel P(H|\theta))$
  - The lower bound of this equation is
  - $L(\lambda,\theta) = \sum_H Q(H|E,\lambda)\ln P(H,E|\theta) - Q(H|E,\lambda)\ln Q(H|E,\lambda)$
  - How to optimize the above?
    - Selecting a good λ
      - Suppose that we setup λ to make $Q(H|E,\lambda) = P(H|E,\theta)$
      - $\sum_H P(H|E,\theta)\ln P(H,E|\theta) - P(H|E,\theta)\ln P(H|E,\theta)$
        $= \sum_H P(H|E,\theta)\ln P(H|E,\theta)P(E|\theta) - P(H|E,\theta)\ln P(H|E,\theta)$
        $= \sum_H P(H|E,\theta)\ln P(E|\theta) = \ln P(E|\theta)\sum_H P(H|E,\theta) = \ln P(E|\theta)$
      - Proven lower bound
    - Readjusting θ is needed
- This results in two sets of parameters to optimize
  - Good match for the EM approach
  - (E Step): $\lambda^{t+1} = argmax_\lambda L(\lambda^t,\theta^t)$
  - (M Step): $\theta^{t+1} = argmax_\theta L(\lambda^{t+1},\theta^t)$
- However, still updating λ^t is a conceptual idea…

Detour: Maximizing the Lower Bound in GMM

- $l(\theta) = \ln P(X|\theta) = \ln\left\{\sum_Z q(Z)\frac{P(X,Z|\theta)}{q(Z)}\right\} \geq \sum_Z q(Z)\ln\frac{P(X,Z|\theta)}{q(Z)} = Q(\theta,q)$
  - $Q(\theta,q) = E_{q(Z)}\ln P(X,Z|\theta) + H(q)$
  - $L(\theta,q) = \ln P(X|\theta) - \sum_Z\left\{q(Z)\ln\frac{q(Z)}{P(Z|X,\theta)}\right\}$
- Why do we compute $L(\theta,q)$?
  - We do not know how to optimize $Q(\theta,q)$ without further knowledge of $q(Z)$
  - The second term of $L(\theta,q)$ tells how to set $q(Z)$
    - The first term is fixed when θ is fixed **at time t**
    - The second term can be minimized to maximize $L(\theta,q)$
      - $KL(q(Z)||P(Z|X,\theta)) = 0 \rightarrow q^t(Z) = P(Z|X,\theta^t)$
  - Now, the lower bound with optimized q is
    - $Q(\theta,q^t) = E_{q^t(Z)}\ln P(X,Z|\theta^t) + H(q^t)$
- Then, optimizing θ to retrieve the tight lower bound is
  - $\theta^{t+1} = argmax_\theta Q(\theta,q^t) = argmax_\theta E_{q^t(Z)}\ln P(X,Z|\theta)$
    - $q^t(Z) \rightarrow$ Distribution parameters for latent variable is at time t
    - $\ln P(X,Z|\theta) \rightarrow$ optimized log likelihood parameters is at time $t+1$

Tells how to setup Z by setting $q^t(Z) = P(Z|X,\theta^t)$

Relax the KL divergence by updating θ^t to θ^{t+1}

Factorizing Q

- Knowing Q → Selecting a good λ and a distribution format of Q
  - We need to know more on P and Q
  - A good setup was $Q(H|E,\lambda) = P(H|E,\theta)$
  - How to find λ without knowing Q?
- P is the probability distribution function.
- Q is not known, and this is an approximation
  - We can setup Q as we want.
  - Our choice is $Q(H) = \prod_{i\le|H|} q_i(H_i|\lambda_i)$
    - Coming from the mean field theory
    - Simple. Easier to handle
    - Pretty strong assumption
- $L(\lambda,\theta) = \sum_H Q(H|E,\lambda)\ln P(H,E|\theta) - Q(H|E,\lambda)\ln Q(H|E,\lambda)$

$$= \sum_H \left\{ \prod_{i\le|H|} q_i(H_i|E,\lambda_i) \ln P(H,E|\theta) - \prod_{i\le|H|} q_i(H_i|E,\lambda_i) \ln \prod_{k\le|H|} q_k(H_i|E,\lambda_i) \right\}$$

$$= \sum_H \left\{ \prod_{i\le|H|} q_i(H_i|E,\lambda_i) \ln P(H,E|\theta) - \prod_{i\le|H|} q_i(H_i|E,\lambda_i) \sum_{k\le|H|} \ln q_k(H_k|E,\lambda_k) \right\}$$

Focusing on Single Variable in Q

- $L(\lambda,\theta)=\sum_H\{\prod_{i\le|H|}q_i(H_i|E,\lambda_i)\ln P(H,E|\theta)-\prod_{i\le|H|}q_i(H_i|E,\lambda_i)\sum_{k\le|H|}\ln q_k(H_k|E,\lambda_k)\}$
- This is a function of the vector of λ, so we need to narrow the scope down
  - This is what we intended to have the fully factorized Q with a certain distribution
- $L(\lambda_j)$

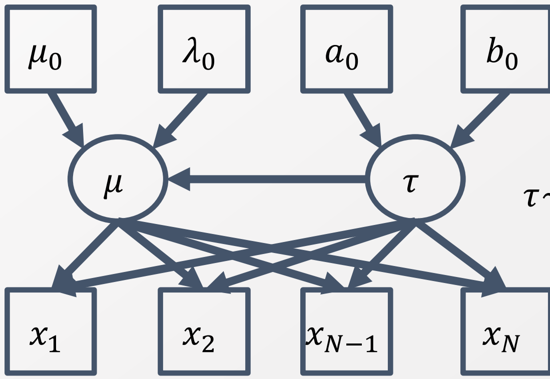
$$
\begin{aligned}
&=\sum_H\left\{\prod_{i\le|H|}q_i(H_i|E,\lambda_i)\ln P(H,E|\theta)-\prod_{i\le|H|}q_i(H_i|E,\lambda_i)\sum_{k\le|H|}\ln q_k(H_k|E,\lambda_k)\right\}\\
&=\sum_H\prod_{i\le|H|}q_i(H_i|E,\lambda_i)\left\{\ln P(H,E|\theta)-\sum_{k\le|H|}\ln q_k(H_k|E,\lambda_k)\right\}\\
&=\sum_{H_j}\sum_{H_{-j}}q_j(H_j|E,\lambda_j)\prod_{i\le|H|,i\ne j}q_i(H_i|E,\lambda_i)\left\{\ln P(H,E|\theta)-\sum_{k\le|H|}\ln q_k(H_k|E,\lambda_k)\right\}\\
&=\sum_{H_j}\sum_{H_{-j}}q_j(H_j|E,\lambda_j)\prod_{i\le|H|,i\ne j}q_i(H_i|E,\lambda_i)\ln P(H,E|\theta)\\
&\quad-\sum_{H_j}\sum_{H_{-j}}q_j(H_j|E,\lambda_j)\prod_{i\le|H|,i\ne j}q_i(H_i|E,\lambda_i)\left(\sum_{k\ne j,k\le|H|}\ln q_k(H_k|E,\lambda_k)+\ln q_j(H_j|E,\lambda_j)\right)\\
&=\sum_{H_j}q_j(H_j|E,\lambda_j)\sum_{H_{-j}}\prod_{i\le|H|,i\ne j}q_i(H_i|E,\lambda_i)\ln P(H,E|\theta)-\sum_{H_j}q_j(H_j|E,\lambda_j)\ln q_j(H_j|E,\lambda_j)+C
\end{aligned}
$$

Freeform Optimization

- $L(\lambda_j)$
  $$= \sum_{H_j} q_j(H_j|E,\lambda_j) \sum_{H_{-j}} \prod_{i\le|H|,i\neq j} q_i(H_i|E,\lambda_i) \ln P(H,E|\theta) - \sum_{H_j} q_j(H_j|E,\lambda_j) \ln q_j(H_j|E,\lambda_j) + C$$
- What if we setup a new P function
  - $\ln\tilde{P}(H,E|\theta) \equiv \sum_{H_{-j}} \prod_{j\le|H|,j\neq i} q_j(H_j|E,\lambda_j) \ln P(H,E|\theta) = E_{q_{i\neq j}}[\ln P(H,E|\theta)] + C$
- Then,
  - $L(\lambda_i) = \sum_H q_i(H_i|E,\lambda_i) \ln\tilde{P}(H,E|\theta) - \sum_H q_i(H_i|E,\lambda_i) \ln q_i(H_i|E,\lambda_i) + C$
- How to optimize this?
  - Still, KL Divergence argument holds
    - Actually the negative KL divergence
  - Previous finding: $Q(H|E,\lambda) = P(H|E,\theta)$
  - This time?
  - $\ln q_i^*(H_i|E,\lambda_i) = \ln\tilde{P}(H,E|\theta) = E_{q_{i\neq j}}[\ln P(H,E|\theta)] + C$
- Usually, $\ln P(H,E|\theta)$ is provided by a probabilistic graphical model
  - Much more concrete in updating λ_i

EXAMPLES OF VARIATIONAL INFERENCE

Simple Example Model



$$\mu \sim \mathrm{N}(\mu_0, \frac{1}{\lambda_0\ \tau})$$
$$\tau \sim Gamma(a_0, b_0)$$
$$x_i \sim \mathrm{N}(\mu, \frac{1}{\tau})$$

$$P(\mu|\tau) = \sqrt{\frac{\lambda_0\ \tau}{2\pi}} e^{-\frac{\lambda_0\ \tau(\mu-\mu_0)^2}{2}}$$
$$\mathrm{P}(\tau) = \frac{1}{\Gamma(a_0)} b_0{}^{a_0} \tau^{a_0-1} e^{-b_0\tau}$$
$$P(x_i) = \sqrt{\frac{\tau}{2\pi}} e^{-\frac{\tau(x_i-\mu)^2}{2}}$$

- $\ln q_i{}^*(H_i|E, \lambda_i) = \ln\tilde{P}(H, E|\theta) = E_{q_{i\neq j}}[\ln P(H, E|\theta)] + C$
- We need to enumerate the joint probability
- $P(H, E|\theta) = P(X, \mu, \tau|\mu_0, \lambda_0, a_0, b_0\ )$
  $= P(X|\mu, \tau\ )P(\mu|\tau, \mu_0, \lambda_0\ )P(\tau|a_0, b_0\ )$
  $= \prod_{i\leq N} P(x_i|\mu, \tau\ )\ P(\mu|\tau, \mu_0, \lambda_0\ )P(\tau|a_0, b_0\ )$
- We need two variational parameters
  - $Q(H|E, \lambda) = Q(\mu, \tau|X, \mu^*, \tau^*) = q(\mu|X, \mu^*)q(\tau|X, \tau^*)$
  - Let's say: $q(\mu|X, \mu^*) = q_\mu{}^*(\mu)$, $q(\tau|X, \tau^*) = q_\tau{}^*(\tau)$

Calculate an Optimal Variational Parameter

$$P(\mu|\tau) = \sqrt{\frac{\lambda_0\ \tau}{2\pi}}e^{-\frac{\lambda_0\ \tau(\mu-\mu_0)^2}{2}}$$

$$\mathrm{P}(\tau) = \frac{1}{\Gamma(a_0)}{b_0}^{a_0}\tau^{a_0-1}e^{-b_0\tau}$$

$$P(x_i) = \sqrt{\frac{\tau}{2\pi}}e^{-\frac{\tau(x_i-\mu)^2}{2}}$$

- $\ln q_\mu{}^*(\mu) = E_\tau[\ln P(X,\mu,\tau|\mu_0,\lambda_0,a_0,b_0\ )] + C1$

$$= E_\tau\left[\ln\prod_{i\le N}P(x_i|\mu,\tau\ )\,P(\mu|\tau,\mu_0,\lambda_0\ )P(\tau|a_0,b_0\ )\right] + C1$$

$$= E_\tau\left[\sum_{i\le N}\left(\frac{1}{2}(ln\tau - ln2\pi) - \frac{(x_i-\mu)^2\tau}{2}\right)\right]$$

$$+ E_\tau\left[\frac{1}{2}(\ln\lambda_0+\ln\tau-\ln 2\pi) - \frac{(\mu-\mu_0)^2\lambda_0\tau}{2}\right] + C2$$

Absorb terms that are not related to μ as a constant

$$= E_\tau\left[\sum_{i\le N} - \frac{(x_i-\mu)^2\tau}{2}\right] + E_\tau\left[-\frac{(\mu-\mu_0)^2\lambda_0\tau}{2}\right] + C3$$

$$= -\frac{E_\tau[\tau]}{2}\left\{\sum_{i\le N}(x_i-\mu)^2 + (\mu-\mu_0)^2\lambda_0\right\} + C3$$

Quadratic function with respect to μ

$$= -\frac{1}{2}\left\{(\lambda_0+N)E_\tau[\tau]\left(\mu - \frac{\lambda_0\mu_0 + \sum_{i\le N}x_i}{\lambda_0+N}\right)^2\right\} + C4$$

Calculate an Optimal Variational Parameter

$$P(\mu|\tau) = \sqrt{\frac{\lambda_0 \tau}{2\pi}} e^{-\frac{\lambda_0 \tau(\mu-\mu_0)^2}{2}}$$

$$P(\tau) = \frac{1}{\Gamma(a_0)} {b_0}^{a_0} \tau^{a_0-1} e^{-b_0\tau}$$

$$P(x_i) = \sqrt{\frac{\tau}{2\pi}} e^{-\frac{\tau(x_i-\mu)^2}{2}}$$

- $\ln q_\mu{}^*(\mu) = -\frac{1}{2}\left\{(\lambda_0 + N)E_\tau[\tau]\left(\mu - \frac{\lambda_0\mu_0 + \sum_{i\le N} x_i}{\lambda_0 + N}\right)^2\right\} + C4$
- We have not decided the distribution shape of the $q_\mu{}^*(\mu)$
  - We only decided that $Q(H|E,\lambda) = Q(\mu,\tau|X,\mu^*,\tau^*) = q(\mu|X,\mu^*)q(\tau|X,\tau^*)$
  - Factorization assumption
- What if we assume that $q_\mu{}^*(\mu)$ is also a normal distribution?
  - Quite similar to the PDF of the normal distribution $P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)}{2\sigma^2}}$
  - Need to match up the parameters
- The result of match up is
  - $q_\mu{}^*(\mu) \sim N\left(\mu \middle| \frac{\lambda_0\mu_0 + \sum_{i\le N} x_i}{\lambda_0 + N}, \frac{1}{(\lambda_0+N)E_\tau[\tau]}\right)$
  - What we know already is μ_0, λ_0, $\sum_{i\le N} x_i$, and N
  - What we don't know is $E_\tau[\tau]$
    - **Which asks us to investigate $\ln q_\tau{}^*(\tau)$**

Calculate an Optimal Variational Parameter

$$P(\mu|\tau) = \sqrt{\frac{\lambda_0 \tau}{2\pi}} e^{-\frac{\lambda_0 \tau(\mu-\mu_0)^2}{2}}$$

$$\mathrm{P}(\tau) = \frac{1}{\Gamma(a_0)} {b_0}^{a_0} \tau^{a_0-1} e^{-b_0\tau}$$

$$P(x_i) = \sqrt{\frac{\tau}{2\pi}} e^{-\frac{\tau(x_i-\mu)^2}{2}}$$

- $\ln q_\tau^*(\tau) = E_\mu[\ln P(X,\mu,\tau|\mu_0,\lambda_0,a_0,b_0\,)] + C1$

$$= E_\mu\left[\sum_{i\le N}\left(\frac{1}{2}(ln\tau - ln2\pi) - \frac{(x_i-\mu)^2\tau}{2}\right)\right] + E_\mu\left[\frac{1}{2}(\ln\lambda_0+\ln\tau-\ln 2\pi) - \frac{(\mu-\mu_0)^2\lambda_0\tau}{2}\right]$$
$$+E_\mu[-ln\Gamma(a_0) + a_0\ln b_0 + (a_0-1)ln\tau - b_0\tau] + C1$$
$$= E_\mu\left[\sum_{i\le N}\left(-\frac{(x_i-\mu)^2\tau}{2}\right)\right] + E_\mu\left[-\frac{(\mu-\mu_0)^2\lambda_0\tau}{2}\right] + \frac{N}{2}ln\tau + \frac{1}{2}ln\tau + (a_0-1)ln\tau - b_0\tau + C2$$
$$= -\frac{\tau}{2}E_\mu[\sum_{i\le N}(x_i-\mu)^2 + (\mu-\mu_0)^2\lambda_0] + \frac{N}{2}ln\tau + \frac{1}{2}ln\tau + (a_0-1)ln\tau - b_0\tau + C2$$
$$= -\tau\left(b_0 + \frac{1}{2}E_\mu[\sum_{i\le N}(x_i-\mu)^2 + (\mu-\mu_0)^2\lambda_0]\right) + \left(a_0 + \frac{N+1}{2} - 1\right)\ln\tau + C2$$

- Again, this function is very familiar(?!), and we have not set the actual distribution of $q_\tau^*(\tau)$
  - Gamma distribution: $P(X) = \frac{x^{k-1}e^{-\frac{x}{\theta}}}{\theta^k\Gamma(k)}, when\, X{\sim}Gamma(k,\theta)$
- Matching parameters
  - $q_\tau^*(\tau){\sim}\mathrm{Gamma}(\tau|a_0 + \frac{N+1}{2}, b_0 + \frac{1}{2}E_\mu[\sum_{i\le N}(x_i-\mu)^2 + (\mu-\mu_0)^2\lambda_0])$
  - What we already know is a_0, b_0, N, x_i, μ_0, and λ_0
  - What we don't know is μ and its expectation terms

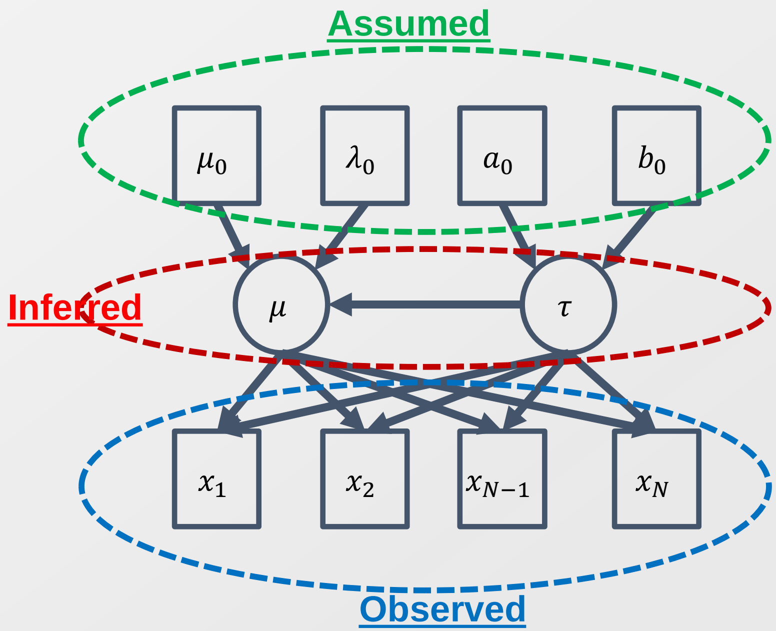
KAIST

Coordinated Update

- What we know and what we should calculate
  - $q_\mu^*(\mu) \sim N\left(\mu \middle| \frac{\lambda_0\mu_0+\sum_{i\le N} x_i}{\lambda_0+N}, \frac{1}{(\lambda_0+N)E_\tau[\tau]}\right) = N(\mu|\mu^*, {\lambda^*}^{-1})$
    - What we know already is μ_0, λ_0, $\sum_{i\le N} x_i$, and N
    - What we don't know is $E_\tau[\tau]$
  - $q_\tau^*(\tau) \sim \text{Gamma}(\tau|a_0 + \frac{N+1}{2}, b_0 + \frac{1}{2}E_\mu[\sum_{i\le N}(x_i-\mu)^2 + (\mu-\mu_0)^2\lambda_0])$ $=\text{Gamma}(\tau|a^*, b^*)$
    - What we already know is a_0, b_0, N, x_i, μ_0, and λ_0
    - What we don't know is μ and its expectation terms
- Since we know the distributions and the parameters, we know mean!
  - $E_\tau[\tau] = \frac{a_0+\frac{N+1}{2}}{b_0+\frac{1}{2}E_\mu[\sum_{i\le N}(x_i-\mu)^2+(\mu-\mu_0)^2\lambda_0]} = \frac{a^*}{b^*}$: Needs $E_\mu[\mu]$ and $E_\mu[\mu^2]$
  - $E_\mu[\mu] = \frac{\lambda_0\mu_0+\sum_{i\le N} x_i}{\lambda_0+N} = \mu^*$: Don't need anything
  - $E_\mu[\mu^2] = \frac{1}{(\lambda_0+N)E_\tau[\tau]} + (\frac{\lambda_0\mu_0+\sum_{i\le N} x_i}{\lambda_0+N})^2 = {\lambda^*}^{-1} + (\mu^*)^2$: Needs $E_\tau[\tau]$
- Since the two terms are interlocked, we need a coordinated optimization

Algorithm for the Parameter Update

- Structure of inference algorithm



- Inference(X, $a_0, b_0, \mu_0, \lambda_0$)
  - $a^* = a_0 + \frac{N+1}{2}$
  - $\mu^* = \frac{\lambda_0\mu_0 + \sum_{i\leq N} x_i}{\lambda_0 + N}$
  - $\lambda^* = (arbitary\ number)$
  - Iteration until converge
    - $b^* = b_0 + \frac{1}{2}E_\mu[\sum_{i\leq N}(x_i - \mu)^2 + (\mu - \mu_0)^2\lambda_0]$
      - With $E_\mu[\mu] = \mu^*$, $E_\mu[\mu^2] = {\lambda^*}^{-1} + (\mu^*)^2$
    - $\lambda^* = (\lambda_0 + N)E_\tau[\tau]$
      - With $E_\tau[\tau] = \frac{a^*}{b^*}$
  - Return Approximated $\mu \sim N(\mu|\mu^*, {\lambda^*}^{-1})$, $\tau \sim Gamma(\tau|a^*, b^*)$

Generalized Step-By-Step Instruction

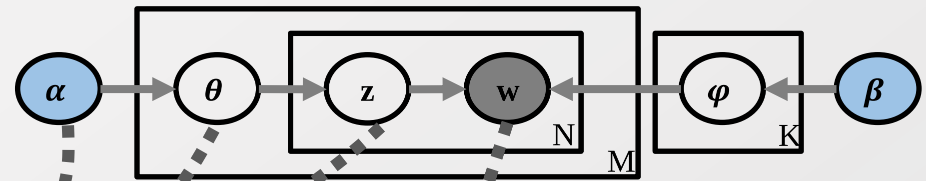
1. Create a Probabilistic Graphical Model
2. Find the data (E), the parameters (θ), and a latent variables (H)
3. Partition H into small subsets: Full factorization → Mean field approximation
4. For each partition H_i: Find a best approximating distribution $q_j^*(H_j|E)$ by performing
   $$\ln q_j^*(H_j|E) = E_{i\neq j}[lnP(Z,X)] + C$$
5. Find the sets of probability distribution parameters by each $q_j^*(H_j|E)$
6. If there are interacting distribution parameters across distributions of partitions, then use an iterative approach like Expectation-Maximization Approach
7. You now have an optimized set of parameters for each latent variation partitions

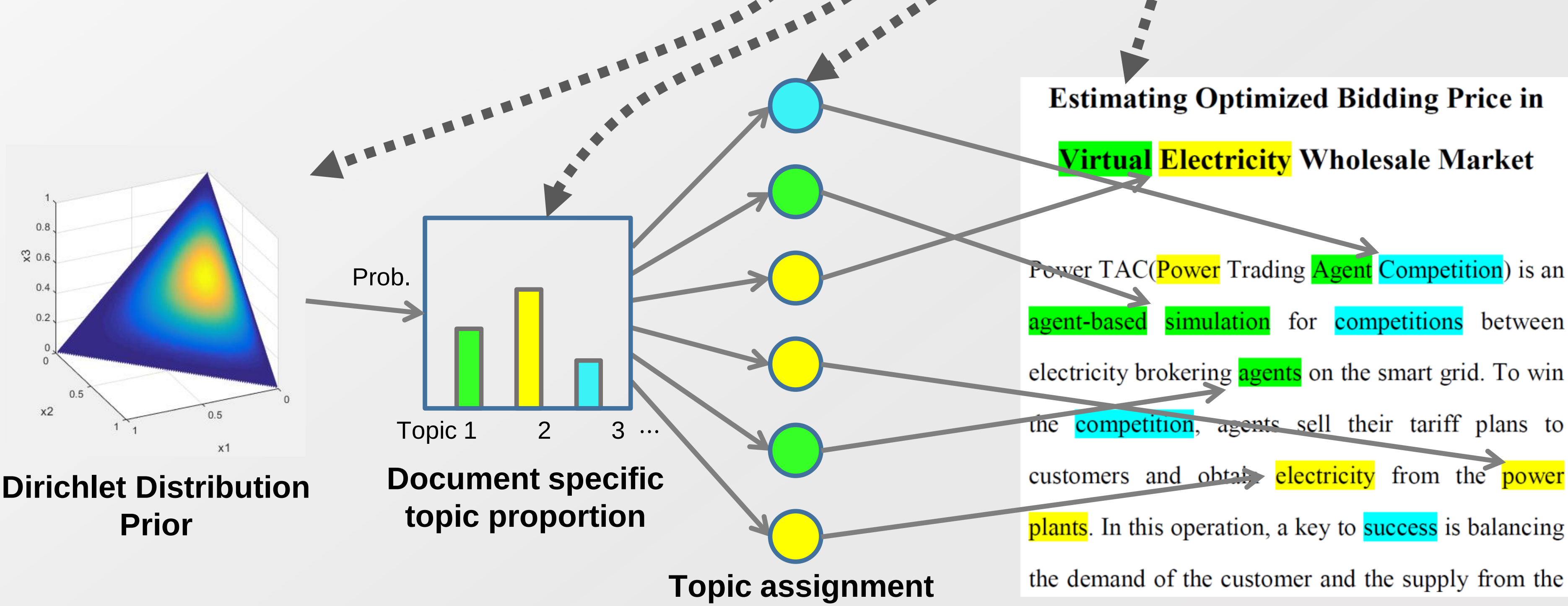
VARIATIONAL INFERENCE OF LATENT DIRICHLET ALLOCATION

Detour: Latent Dirichlet Allocation

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3.Jan (2003): 993-1022.

- Latent Dirichlet Allocation
  - Soft clustering in text data
  - Has the structure of text corpus
  - Is a Bayesian model with priors
- For each word w, sample topic assignment z





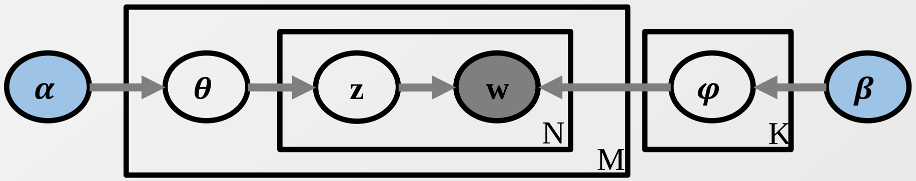
Dirichlet Distribution Prior

Document specific topic proportion

Topic assignment

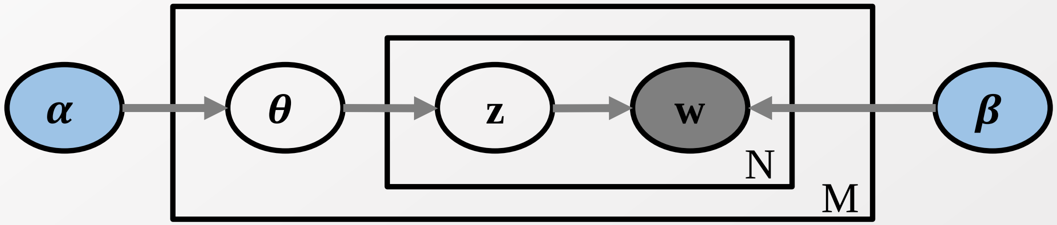
Detour: Finding Topic Assignment Per Word

- Let's treat this as a Bayesian network
  - **Generative Process**
    - $\theta_i \sim Dir(\alpha), i \in \{1, \dots, M\}$
    - $\varphi_k \sim Dir(\beta), k \in \{1, \dots, K\}$
    - $z_{i,l} \sim Mult(\theta_i), i \in \{1, \dots, M\}, l \in \{1, \dots, N\}$
    - $w_{i,l} \sim Mult\left(\varphi_{z_{i,l}}\right), i \in \{1, \dots, M\}, l \in \{1, \dots, N\}$



  - A word **w** is generated from the distribution of $\boldsymbol{\varphi_z}$ word-topic distribution
  - **z** topic is generated from the distribution of $\boldsymbol{\theta}$ document-topic distribution
  - $\boldsymbol{\theta}$ document topic distribution is generated from the distribution of $\boldsymbol{\alpha}$
  - $\boldsymbol{\varphi}$ word-topic distribution is generated from the distribution of $\boldsymbol{\beta}$
- If we have Z distribution, we can find the most likely θ and φ
  - θ: Topic distribution in a document
  - φ: Word distribution in a topic
  - Finding the most likely allocation of Z is the key of inference on θ and φ

Evidence Lower Bound of LDA



- $\ln P(E|\theta) \geq \sum_H Q(H|E,\lambda)\ln P(H,E|\theta) - Q(H|E,\lambda)\ln Q(H|E,\lambda)$

  $= \sum_H Q(H|E)\ln P(E|H,\theta) - Q(H|E)\ln \frac{Q(H|E)}{P(H|\theta)}$

- $\ln P(w|\alpha,\beta) \geq \int \sum_z q(\theta,z|\gamma,\phi)\log \frac{P(\theta,z,w|\alpha,\beta)}{q(\theta,z|\gamma,\phi)}\,d\theta$

$$= \int \sum_z q(\theta,z|\gamma,\phi)\log P(\theta,z,w|\alpha,\beta)\,d\theta - \int \sum_z q(\theta,z|\gamma,\phi)\log q(\theta,z)\,d\theta$$

$$= \int \sum_z q(\theta,z|\gamma,\phi)\log P(\theta|\alpha)\,d\theta + \int \sum_z q(\theta,z|\gamma,\phi)\log P(z|\theta)\,d\theta$$

$$+ \int \sum_z q(\theta,z|\gamma,\phi)\log P(w|z,\beta)\,d\theta - \int \sum_z q(\theta,z|\gamma,\phi)\log q(\theta,z)\,d\theta$$

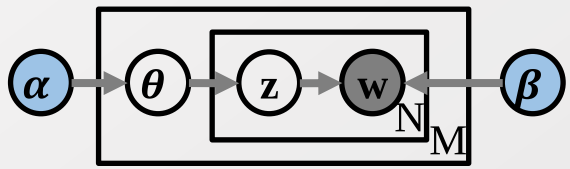
$$= E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$$

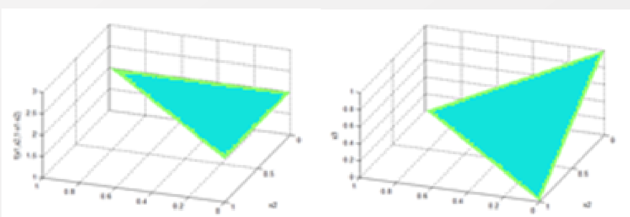
$$= L(\gamma,\phi|\alpha,\beta)$$

$$H(p) = -\sum_i p(x_i)\log p(x_i)$$

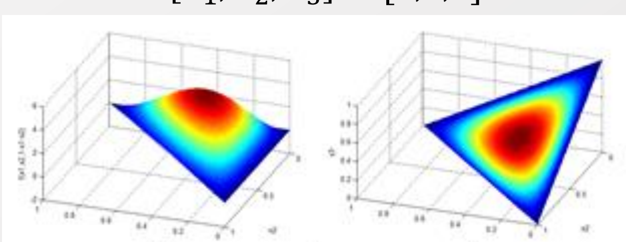
Detour: Dirichlet Distribution

- **Generative Process**
  - $\theta_i \sim Dir(\alpha), i \in \{1, \dots, M\}$, $\varphi_k \sim Dir(\beta), k \in \{1, \dots, K\}$
  - $z_{i,l} \sim Mult(\theta_i), i \in \{1, \dots, M\}, l \in \{1, \dots, N\}$, $w_{i,l} \sim Mult\left(\varphi_{z_{i,l}}\right), i \in \{1, \dots, M\}, l \in \{1, \dots, N\}$
- Dirichlet Distribution
  - $P(x_1, \dots, x_K | \alpha_1, \dots, \alpha_K) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} {x_i}^{\alpha_i - 1}$
    - $x_1, \dots, x_{K-1} > 0$
    - $x_1 + \cdots + x_{K-1} < 1$
    - $x_K = 1 - x_1 - \cdots - x_{K-1}$
    - $\alpha_i > 0$

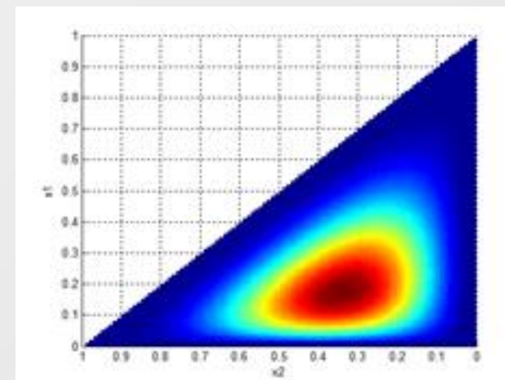


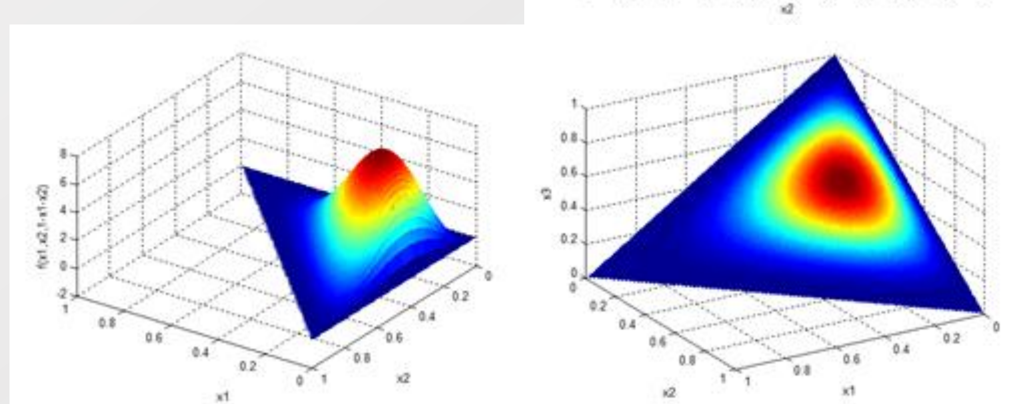


$[\alpha_1, \alpha_2, \alpha_3] = [1,1,1]$



$[\alpha_1, \alpha_2, \alpha_3] = [2,2,2]$





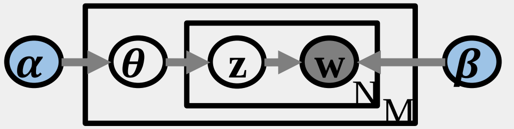
$[\alpha_1, \alpha_2, \alpha_3] = [2,3,4]$

Detour: Exponential Family

- Exponential Family
  - $P(x|\theta) = h(x)\exp(\eta(\theta) \cdot T(x) - A(\theta))$
    - Sufficient statistics : $T(x)$, Natural parameter : $\eta(\theta)$
    - Underlying measure : $h(x)$, Log normalizer : $A(\theta)$
  - Normal Distribution : $P(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
    - Sufficient statistics : $(x, x^2)^T$, Natural parameter : $\left(\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right)^T$
    - Underlying measure :$\frac{1}{\sqrt{2\pi}}$, Log normalizer : $\frac{\mu^2}{2\sigma^2} + \log|\sigma|$
  - Dirichlet Distribution : $P(x_1, \dots, x_K|\alpha_1, \dots, \alpha_K) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} {x_i}^{\alpha_i - 1}$
    - Sufficient statistics : $(\log x_1, \dots, \log x_K)^T$, Natural parameter :$(\alpha_1 - 1, \dots, \alpha_K - 1)^T$
    - Underlying measure :1, Log normalizer : $-\log\Gamma\left(\sum_{i=1}^K \alpha_i\right) + \log\prod_{i=1}^K \Gamma(\alpha_i)$
- Derivative of log normalizer → Moments of sufficient statistics
  - $\frac{d}{d\eta}a(\eta) = \frac{d}{d\eta}\log\int h(x)\exp\{\eta^T T(x)\}dx = \frac{\int T(x)h(x)\exp\{\eta^T T(x)\}dx}{\int h(x)\exp\{\eta^T T(x)\}dx}$

$$= \frac{\int T(x)h(x)\exp\{\eta^T T(x)\}dx}{\exp(a(\eta))} = \int T(x)h(x)\exp\{\eta^T T(x) - a(\eta)\}\,dx = E_P[T(x)]$$

Derivation of $E_q(\log P(\theta|\alpha))$

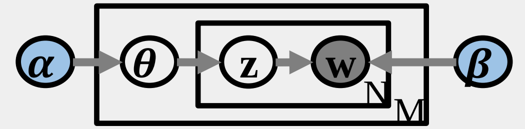


$P(x_1, \ldots, x_K | \alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} x_i^{\alpha_i - 1}$

- Further derivation of the first term in the evidence lower bound of LDA
  - $L(\gamma, \phi | \alpha, \beta) = E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z, \beta)) + H(q)$
- $E_q(\log P(\theta|\alpha)) = E_q(\sum_{d=1}^{M} \sum_{i=1}^{k} (\alpha_i - 1) \log \theta_{d,i} + \log \Gamma(\sum_{i=1}^{K} \alpha_i) - \sum_{i=1}^{K} \log \Gamma(\alpha_i))$
  $= \sum_{d=1}^{M} \sum_{i=1}^{k} (\alpha_i - 1) E_q(\log \theta_{d,i}) + \log \Gamma(\sum_{i=1}^{K} \alpha_i) - \sum_{i=1}^{K} \log \Gamma(\alpha_i)$
- $E_{q(\theta, z|\gamma, \phi)}(\log \theta_{d,i}) = E_{q(\theta|\gamma) q(z|\phi)}(\log \theta_{d,i}) = E_{q(\theta|\gamma)}(\log \theta_{d,i})$
  - $q(\theta|\gamma)$ can be assumed to follow the Dirichlet distribution
  - Derivative of log normalizer → Moments of sufficient statistics
    - Sufficient statistics : $(\log \theta_{d,1}, \ldots, \log \theta_{d,K})^T$
    - Log normalizer : $-\log \Gamma(\sum_{i=1}^{K} \gamma_{d,i}) + \log \prod_{i=1}^{K} \Gamma(\gamma_{d,i})$
  - $E_{q(\theta|\gamma)}(\log \theta_{d,i}) = \frac{d}{d\gamma_{d,i}} (-\log \Gamma(\sum_{i=1}^{K} \gamma_{d,i}) + \log \prod_{i=1}^{K} \Gamma(\gamma_{d,i})) = -\psi(\sum_{i=1}^{K} \gamma_{d,i}) + \psi(\gamma_{d,i})$
    - $\psi(\gamma_{d,i}) = \frac{d}{d\gamma_i} \log \Gamma(\gamma_{d,i}) = \frac{\Gamma'(\gamma_{d,i})}{\Gamma(\gamma_{d,i})}$
      - Digamma function, calculation is based upon mathematical libraries (ex. scipy)
- $E_q(\log P(\theta|\alpha))$

$$= \sum_{d=1}^{M} \sum_{i=1}^{k} (\alpha_i - 1) \left(-\psi \left(\sum_{i=1}^{K} \gamma_{d,i} \right) + \psi(\gamma_{d,i}) \right) + \log \Gamma \left(\sum_{i=1}^{K} \alpha_i \right) - \sum_{i=1}^{K} \log \Gamma(\alpha_i)$$

Derivation of $E_q(\log P(z|\theta))$ and $E_q(\log P(w|z,\beta))$



$P(x_1,\dots,x_K|\alpha_1,\dots,\alpha_K) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} x_i^{\alpha_i - 1}$

- Further derivation of the second and the third terms in the evidence lower bound of LDA
  - $L(\gamma,\phi|\alpha,\beta) = E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$
- $E_q(\log P(z|\theta)) = \sum_{d=1}^M \sum_{n=1}^{N_d} E_q(\log P(z_{d,n}|\theta_d)) = \sum_{d=1}^M \sum_{n=1}^{N_d} \sum_{i=1}^K E_q(\log P(z_{d,n,i}|\theta_{d,i}))$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} E_q(\log \theta_{d,i}{}^{z_{d,n,i}}) = \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} E_{q(\theta|\gamma)q(z|\phi)}(z_{d,n,i}\log\theta_{d,i})$$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} E_{q(z|\phi)}(z_{d,n,i}) E_{q(\theta|\gamma)}(\log\theta_{d,i})$$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} \phi_{n,i}(E_{q(\theta|\gamma)}(\log\theta_{d,i})) = \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} \phi_{d,n,i}\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})\right)$$

  - $\because E_{q(\theta|\gamma)}(\log\theta_{d,i}) = \frac{d}{d\gamma_i}(-\log\Gamma(\sum_{i=1}^K \gamma_{d,i}) + \log\prod_{i=1}^K \Gamma(\gamma_{d,i})) = -\psi(\sum_{i=1}^K \gamma_{d,i}) + \psi(\gamma_{d,i})$
- $E_q(\log P(w|z,\beta)) = \sum_{d=1}^M \sum_{n=1}^{N_d} E_q(\log P(w_{d,n}|z_{d,n},\beta)) = \sum_{d=1}^M \sum_{n=1}^{N_d} \sum_{i=1}^K E_q(\log \beta_{i,w_{d,n}}{}^{z_{d,n,i}})$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} E_{q(\theta|\gamma)q(z|\phi)}(z_{d,n,i}\log\beta_{i,w_{d,n}})$$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} E_{q(z|\phi)}(z_{d,n,i})\log\beta_{i,w_{d,n}} = \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K} \phi_{d,n,i}\log\beta_{i,w_{d,n}}$$

Derivation of $H(q)$

- Further derivation of the fourth term in the evidence lower bound of LDA
  - $L(\gamma,\phi|\alpha,\beta) = E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$
- $H(q) = -\int \sum_z q(\theta,z)\log q(\theta,z)\,d\theta = -\int q(\theta)\log q(\theta)\,d\theta - \sum_z q(z)\log q(z)$
  - $\int q(\theta)\log q(\theta)\,d\theta = E_{q(\theta|\gamma)}(\log q(\theta|\gamma))$

$$= \sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) + \log\Gamma\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right) - \sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})$$

    - $E_{q(\theta|\gamma)}(\log P(\theta|\alpha))$

$$= \sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) + \log\Gamma\left(\sum\nolimits_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K}\log\Gamma(\alpha_i)$$

  - $\sum_z q(z)\log q(z) = E_{q(z|\phi)}(\log q(z|\phi))$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}$$

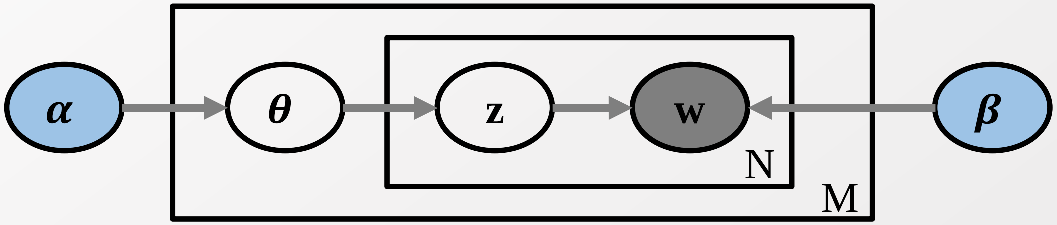
    - $E_{q(\theta|\gamma)q(z|\phi)}(\log P(z|\theta))$

$$= \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)$$

- $H(q)$

$$= -\sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) - \log\Gamma\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right) + \sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})$$
$$-\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}$$

Evidence Lower Bound of LDA after Derivation



- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$

$$= E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$$

$$= \sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i - 1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})\right) + \log\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K}\log\Gamma(\alpha_i)$$

$$+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})\right)$$

$$+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}$$

$$- \sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i} - 1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})\right) - \log\Gamma\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})$$

$$- \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}$$

Learning Variational Parameters, ϕ

$$
\begin{aligned}
&L(\gamma,\phi|\alpha,\beta)\\
&=\sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\log\Gamma\left(\sum\nolimits_{i=1}^{K}\alpha_i\right)-\sum_{i=1}^{K}\log\Gamma(\alpha_i)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}\\
&-\sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)-\log\Gamma\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})\\
&-\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}
\end{aligned}
$$

- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$
- $\frac{d}{d\phi_{d,n,i}}L(\gamma,\phi|\alpha,\beta)$

$$
\begin{aligned}
&=\frac{d}{d\phi_{d,n,i}}\left[\left\{\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}\right.\right.\\
&\left.\left.-\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}\right\}+\lambda_{d,n}\left(\sum_{i=1}^{K}\phi_{d,n,i}-1\right)\right]\\
&=\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\log\beta_{i,w_{d,n}}-\log\phi_{d,n,i}-1+\lambda_{d,n}=0
\end{aligned}
$$

$\rightarrow \log\phi_{d,n,i}=\log\beta_{i,w_{d,n}}+\lambda_{d,n}+\psi(\gamma_{d,i})-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)-1$

$\rightarrow \exp(\log\phi_{d,n,i})=\exp\left(\log\beta_{i,w_{d,n}}+\lambda_{d,n}+\psi(\gamma_{d,i})-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)-1\right)$

$\rightarrow \phi_{d,n,i}=\beta_{i,w_{d,n}}\exp\left(\lambda_{d,n}+\psi(\gamma_{d,i})-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)-1\right)$

$\rightarrow \phi_{d,n,i}\propto\beta_{i,w_{d,n}}\exp\left(\psi(\gamma_{d,i})\right)$

Learning Variational Parameters, γ

$$
\begin{aligned}
&L(\gamma,\phi|\alpha,\beta)\\
&=\sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\log\Gamma\left(\sum_{i=1}^{K}\alpha_i\right)-\sum_{i=1}^{K}\log\Gamma(\alpha_i)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}\\
&-\sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)-\log\Gamma\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})\\
&-\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}
\end{aligned}
$$

- $\ln P(w|\alpha,\beta)\geq L(\gamma,\phi|\alpha,\beta)$
- $\frac{d}{d\gamma_{d,i}}L(\gamma,\phi|\alpha,\beta)$

$$
\begin{aligned}
&=\frac{d}{d\gamma_{d,i}}\Bigg[\sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)\\
&-\sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)-\log\Gamma\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})\Bigg]\\
&=(\alpha_i-1)\left(-\psi'\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi'(\gamma_{d,i})\right)+\sum_{n=1}^{N_d}\phi_{d,n,i}\left(-\psi'\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi'(\gamma_{d,i})\right)\\
&-\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)-(\gamma_{d,i}-1)\left(-\psi'\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi'(\gamma_{d,i})\right)\\
&-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})=\left(-\psi'\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi'(\gamma_{d,i})\right)\left(\alpha_i-1+\sum_{n=1}^{N_d}\phi_{d,n,i}-(\gamma_{d,i}-1)\right)=0
\end{aligned}
$$

→ $\alpha_i-1+\sum_{n=1}^{N_d}\phi_{d,n,i}-(\gamma_{d,i}-1)=0$

→ $\gamma_{d,i}=\alpha_i+\sum_{n=1}^{N_d}\phi_{d,n,i}$

Learning Model Parameters, β

$$
\begin{aligned}
&L(\gamma,\phi|\alpha,\beta) \\
&= \sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i - 1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) + \log\Gamma\left(\sum\nolimits_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K}\log\Gamma(\alpha_i) \\
&+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) \\
&+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}} \\
&- \sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i} - 1)\left(-\psi\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right) - \log\Gamma\left(\sum\nolimits_{i=1}^{K}\gamma_{d,i}\right) + \sum_{i=1}^{K}\log\Gamma(\gamma_{d,i}) \\
&- \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}
\end{aligned}
$$

- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$
- $\frac{d}{d\beta_{i,w_{d,n}}} L(\gamma,\phi|\alpha,\beta)$

$$
= \frac{d}{d\beta_{i,w_{d,n}}}\left[\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}} + \sum_{i=1}^{K}\rho_i\left(\sum_{v=1}^{V}\beta_{i,v} - 1\right)\right]
$$

- $\beta_{i,w_{d,n}} \rightarrow \beta_{i,v}$: v is a unique word index of $w_{d,n}$
- $\frac{d}{d\beta_{i,v}} L(\gamma,\phi|\alpha,\beta)$

$$
= \frac{d}{d\beta_{i,v}}\left[\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{v=1}^{V}\sum_{i=1}^{K}\phi_{d,n,i}1(v = w_{d,n})\log\beta_{i,v} + \sum_{i=1}^{K}\rho_i\left(\sum_{v=1}^{V}\beta_{i,v} - 1\right)\right]
$$

$$
= \frac{\sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{d,n,i}1(v = w_{d,n})}{\beta_{i,v}} + \rho_i = 0
$$

$\rightarrow \beta_{i,v} \propto \sum_{d=1}^{M}\sum_{n=1}^{N_d}\phi_{d,n,i}1(v = w_{d,n})$

Learning Model Parameters, α

$$
\begin{aligned}
&L(\gamma,\phi|\alpha,\beta)\\
&=\sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\log\Gamma\left(\sum_{i=1}^{K}\alpha_i\right)-\sum_{i=1}^{K}\log\Gamma(\alpha_i)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)\\
&+\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}\\
&-\sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i}-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)-\log\Gamma\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\sum_{i=1}^{K}\log\Gamma(\gamma_{d,i})\\
&-\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}
\end{aligned}
$$

- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$
- $\frac{d}{d\alpha_i}L(\gamma,\phi|\alpha,\beta)$

$$
\begin{aligned}
&=\frac{d}{d\alpha_i}\left[\sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i-1)\left(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right)+\log\Gamma\left(\sum_{i=1}^{K}\alpha_i\right)-\sum_{i=1}^{K}\log\Gamma(\alpha_i)\right]\\
&=\sum_{d=1}^{M}\left[-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})+\psi\left(\sum_{i=1}^{K}\alpha_i\right)-\psi(\alpha_i)\right]
\end{aligned}
$$

  - Unable to create a closed form solution → Approximated Optimization : $max_\alpha L(\gamma,\phi|\alpha,\beta)$
  - We will use the Newton-Rhapson method

- $\frac{d^2}{d\alpha_i d\alpha_j}L(\gamma,\phi|\alpha,\beta)=\frac{d}{d\alpha_j}\left[\sum_{d=1}^{M}\left[-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi(\gamma_{d,i})\right]+M\psi\left(\sum_{i=1}^{K}\alpha_i\right)-M\psi(\alpha_i)\right]$

$$
=M\psi'\left(\sum_{i=1}^{K}\alpha_i\right)-\psi'(\alpha_i)M1(i=j)
$$

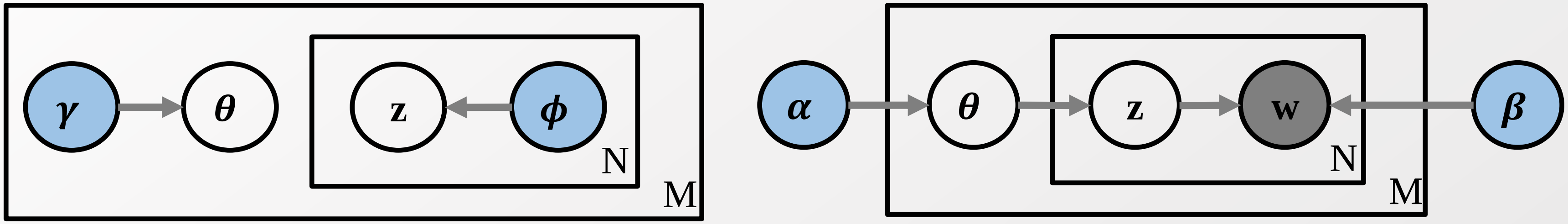
  - Hessian Matrix : a square matrix of second-order partial derivatives of a scalar-valued function. The matrix describes the curvature of the function from different perspectives.

$$
\begin{bmatrix}
\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 x_n}\\
\frac{\partial^2 f}{\partial x_2 x_1} & \frac{\partial^2 f}{\partial x_2^2} & & \frac{\partial^2 f}{\partial x_2 x_n}\\
\vdots & & \ddots & \vdots\\
\frac{\partial^2 f}{\partial x_n x_1} & \frac{\partial^2 f}{\partial x_n x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2}
\end{bmatrix}
$$

Detour: Newton-Rhapson Method

- $f(x)$: differentiable function in the interested range
- We want to find r in the range of x such that $f(r)=0$
- Deriving the Newton-Rhapson iteration
  - $f'(x_0)=\lim_{h\to 0}\frac{f(x_0+h)-f(x_0)}{h}$
    - Assume $r=x_0+h$
  - $f(r)=f(x_0+h)=f(x_0)+hf'(x_0)\approx 0$, if h is very small
  - $h\approx-\frac{f(x_0)}{f'(x_0)}\to r=x_0-\frac{f(x_0)}{f'(x_0)}$
  - $x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}$
- We want to find r in the range of x such that $f(r)=\max_x f(x)$
  - The gradient, $f'(x)$, must be zero
    - Apply the Newton-Rhapson to the gradient function
  - $x_{n+1}=x_n-\frac{f'(x_n)}{f''(x_n)}\to \mathrm{x}_{n+1}=\mathrm{x}_n-\mathrm{H}^{-1}(\mathrm{x}_n)f'(\mathrm{x}_n)$
- $max_\alpha L(\gamma,\phi|\alpha,\beta)$
  - $\frac{d}{d\alpha_i}L(\gamma,\phi|\alpha,\beta)=\sum_{d=1}^{M}\left[-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right)+\psi\left(\gamma_{d,i}\right)+\psi\left(\sum_{i=1}^{K}\alpha_i\right)-\psi(\alpha_i)\right]$
  - $\frac{d}{d\alpha_i\alpha_j}L(\gamma,\phi|\alpha,\beta)=M\psi'\left(\sum_{i=1}^{K}\alpha_i\right)-\psi'(\alpha_i)M1(i=j)$

Parameter Optimization of Evidence Lower Bound



- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$
  $= E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$
- Learning parameters of the evidence lower bound
  - Variational parameter learning
    - $\phi_{d,n,i} \propto \beta_{i,w_{d,n}} \exp\left(\psi(\gamma_{d,i})\right)$
    - $\gamma_{d,i} = \alpha_i + \sum_{n=1}^{N_d} \phi_{d,n,i}$
  - Model parameter learning
    - $\beta_{i,v} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{d,n,i} 1(v = w_{d,n})$
    - $\alpha_{n+1} = \alpha_n - \mathrm{H}^{-1}(\alpha_n) f'(\alpha_n)$
      - $f' = \frac{d}{d\alpha_i} L(\gamma,\phi|\alpha,\beta) = \sum_{d=1}^{M}\left[-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i}) + \psi\left(\sum_{i=1}^{K}\alpha_i\right) - \psi(\alpha_i)\right]$
      - $\mathrm{H} = \frac{d}{d\alpha_i \alpha_j} L(\gamma,\phi|\alpha,\beta) = M\psi'\left(\sum_{i=1}^{K}\alpha_i\right) - \psi'(\alpha_i) M 1(i=j)$
      - Newton-Rhapson method on α

Implementation of LDA-Variational Inference (1)

- Variational parameter learning
  - $\phi_{d,n,i} \propto \beta_{i,w_{d,n}} \exp\left(\psi(\gamma_{d,i})\right)$
  - $\gamma_{d,i} = \alpha_i + \sum_{n=1}^{N_d} \phi_{d,n,i}$
- Model parameter learning
  - $\beta_{i,v} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{d,n,i} 1(v = w_{d,n})$
  - $\alpha_{n+1} = \alpha_n - \mathrm{H}^{-1}(\alpha_n) f'(\alpha_n)$

```python
def performLDA(self):
    for iteration in range(self.numIterations):
        # E-Step : Learning phi and gamma
        # initialize the variational parameter
        self.phi = []
        self.gamma = zeros(shape=(self.intNumDoc, self.intNumTopic), dtype=float)
        for d in range(self.intNumDoc):
            self.phi.append(zeros(shape=(self.numWordPerDoc[d], self.intNumTopic), dtype=float))
            for n in range(self.numWordPerDoc[d]):
                for k in range(self.intNumTopic):
                    self.phi[d][n][k] = 1.0 / float(self.intNumTopic)
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):
                self.gamma[d][k] = self.alpha[k] + float(self.intUniqueWord) / float(self.intNumTopic)

        for d in range(self.intNumDoc):
            for iterationInternal in range(self.numInternalIterations):
                expDigammaGammaDK = zeros(shape=(self.intNumTopic),dtype=float)
                for k in range(self.intNumTopic):
                    expDigammaGammaDK[k] = exp(digamma(self.gamma[d][k]))
                # Learning phi
                for n in range(self.numWordPerDoc[d]):
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.beta[k][self.corpusList[d][n]] * expDigammaGammaDK[k]
                    normalizeConstantPhi = sum(self.phi[d][n])
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.phi[d][n][k] / normalizeConstantPhi
                # Learning gamma
                for k in range(self.intNumTopic):
                    self.gamma[d][k] = self.alpha[k]
                    for n in range(self.numWordPerDoc[d]):
                        self.gamma[d][k] = self.gamma[d][k] + self.phi[d][n][k]

        # M-Step : Learning alpha and beta
        # Learning Beta
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):
                for n in range(self.numWordPerDoc[d]):
                    self.beta[k][self.corpusList[d][n]] = self.beta[k][self.corpusList[d][n]] + self.phi[d][n][k]
            normalizeConstantBeta = sum(self.beta[k])
            for v in range(self.intUniqueWord):
                self.beta[k][v] = self.beta[k][v] / normalizeConstantBeta

        # Learning Alpha
        # Newton-Rhapson optimization
        for itr in range(self.numNewtonIteration):
            # Building Hessian Matrix and Derivative Vector
            H = zeros(shape=(self.intNumTopic,self.intNumTopic), dtype=float)
            g = zeros(shape=(self.intNumTopic), dtype=float)
            for k1 in range(self.intNumTopic):
                g[k1] = float(self.intNumDoc)*(digamma(sum(self.alpha))-digamma(self.alpha[k1]))
                for d in range(self.intNumDoc):
                    g[k1] = g[k1] + ( digamma(self.gamma[d][k1]) - digamma(sum(self.gamma[d])) )
                for k2 in range(self.intNumTopic):
                    H[k1][k2] = 0
                    if k1 == k2:
                        H[k1][k2] = H[k1][k2] - float(self.intNumDoc) * polygamma(1,self.alpha[k1])
                    H[k1][k2] = H[k1][k2] + float(self.intNumDoc) * polygamma(1,sum(self.alpha))

            deltaAlpha = np.dot(np.linalg.inv(H),g)
            self.alpha = self.alpha - deltaAlpha
```

Implementation of LDA-Variational Inference (2)

APPLIED ARTIFICIAL INTELLIGENCE LAB

- Variational parameter learning
  - $\phi_{d,n,i} \propto \beta_{i,w_{d,n}} \exp\left(\psi(\gamma_{d,i})\right)$
  - $\gamma_{d,i} = \alpha_i + \sum_{n=1}^{N_d} \phi_{d,n,i}$

```
def performLDA(self):
    for iteration in range(self.numIterations):
        # E-Step : Learning phi and gamma
        # initialize the variational parameter
        self.phi = []
        self.gamma = zeros(shape=(self.intNumDoc, self.intNumTopic), dtype=float)
        for d in range(self.intNumDoc):
            self.phi.append(zeros(shape=(self.numWordPerDoc[d], self.intNumTopic), dtype=float))
            for n in range(self.numWordPerDoc[d]):
                for k in range(self.intNumTopic):
                    self.phi[d][n][k] = 1.0 / float(self.intNumTopic)
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):

        for d in range(self.intNumDoc):
            for iterationInternal in range(self.numInternalIterations):
                expDigammaGammaDK = zeros(shape=(self.intNumTopic),dtype=float)
                for k in range(self.intNumTopic):
                    expDigammaGammaDK[k] = exp(digamma(self.gamma[d][k]))
                # Learning phi
                for n in range(self.numWordPerDoc[d]):
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.beta[k][self.corpusList[d][n]] * expDigammaGammaDK[k]
                    normalizeConstantPhi = sum(self.phi[d][n])
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.phi[d][n][k] / normalizeConstantPhi
                # Learning gamma
                for k in range(self.intNumTopic):
                    self.gamma[d][k] = self.alpha[k]
                    for n in range(self.numWordPerDoc[d]):
                        self.gamma[d][k] = self.gamma[d][k] + self.phi[d][n][k]

        # Learning Beta
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):
                for n in range(self.numWordPerDoc[d]):
                    self.beta[k][self.corpusList[d][n]] = self.beta[k][self.corpusList[d][n]] + self.phi[d][n][k]
            normalizeConstantBeta = sum(self.beta[k])
            for v in range(self.intUniqueWord):
                self.beta[k][v] = self.beta[k][v] / normalizeConstantBeta

        # Learning Alpha
        # Newton-Rhapson optimization
        for itr in range(self.numNewtonIteration):
            # Building Hessian Matrix and Derivative Vector
            H = zeros(shape=(self.intNumTopic,self.intNumTopic), dtype=float)
            g = zeros(shape=(self.intNumTopic), dtype=float)
            for k1 in range(self.intNumTopic):
                g[k1] = float(self.intNumDoc)*(digamma(sum(self.alpha))-digamma(self.alpha[k1]))
                for d in range(self.intNumDoc):
                    g[k1] = g[k1] + ( digamma(self.gamma[d][k1]) - digamma(sum(self.gamma[d])) )
                for k2 in range(self.intNumTopic):
                    H[k1][k2] = 0
                    if k1 == k2:
                        H[k1][k2] = H[k1][k2] - float(self.intNumDoc) * polygamma(1,self.alpha[k1])
                    H[k1][k2] = H[k1][k2] + float(self.intNumDoc) * polygamma(1,sum(self.alpha))

            deltaAlpha = np.dot(np.linalg.inv(H),g)
            self.alpha = self.alpha - deltaAlpha
```

```
for d in range(self.intNumDoc):
    for iterationInternal in range(self.numInternalIterations):
        expDigammaGammaDK = zeros(shape=(self.intNumTopic),dtype=float)
        for k in range(self.intNumTopic):
            expDigammaGammaDK[k] = exp(digamma(self.gamma[d][k]))
        # Learning phi
        for n in range(self.numWordPerDoc[d]):
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.beta[k][self.corpusList[d][n]] * expDigammaGammaDK[k]
            normalizeConstantPhi = sum(self.phi[d][n])
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.phi[d][n][k] / normalizeConstantPhi
        # Learning gamma
        for k in range(self.intNumTopic):
            self.gamma[d][k] = self.alpha[k]
            for n in range(self.numWordPerDoc[d]):
                self.gamma[d][k] = self.gamma[d][k] + self.phi[d][n][k]
```

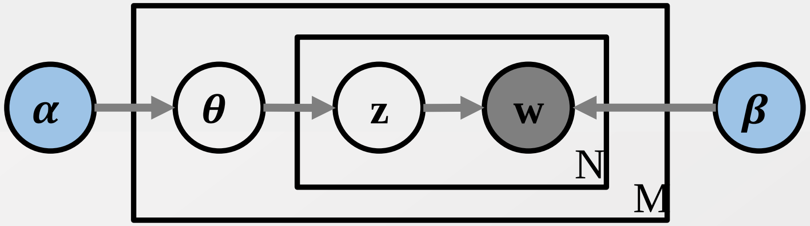
Implementation of LDA-Variational Inference (3)

- Model parameter learning
  - $\beta_{i,v} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{d,n,i} 1(v = w_{d,n})$
  - $\alpha_{n+1} = \alpha_n - \mathrm{H}^{-1}(\alpha_n) f'(\alpha_n)$

```python
def performLDA(self):
    for iteration in range(self.numIterations):
        # E-Step : Learning phi and gamma
        # initialize the variational parameter
        self.phi = []
        self.gamma = zeros(shape=(self.intNumDoc, self.intNumTopic), dtype=float)
        for d in range(self.intNumDoc):
            self.phi.append(zeros(shape=(self.numWordPerDoc[d], self.intNumTopic), dtype=float))
            for n in range(self.numWordPerDoc[d]):
                for k in range(self.intNumTopic):
                    self.phi[d][n][k] = 1.0 / float(self.intNumTopic)
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):
                self.gamma[d][k] = self.alpha[k] + float(self.intUniqueWord) / float(self.intNumTopic)

        for d in range(self.intNumDoc):
            for iterationInternal in range(self.numInternalIterations):
                expDigammaGammaDK = zeros(shape=(self.intNumTopic),dtype=float)
                for k in range(self.intNumTopic):
                    expDigammaGammaDK[k] = exp(digamma(self.gamma[d][k]))
                # Learning phi
                for n in range(self.numWordPerDoc[d]):
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.beta[k][self.corpusList[d][n]] * expDigammaGammaDK[k]
                    normalizeConstantPhi = sum(self.phi[d][n])
                    for k in range(self.intNumTopic):
                        self.phi[d][n][k] = self.phi[d][n][k] / normalizeConstantPhi
                # Learning gamma
                for k in range(self.intNumTopic):
                    self.gamma[d][k] = self.alpha[k]
                    for n in range(self.numWordPerDoc[d]):
```

```python
        # M-Step : Learning alpha and beta
        # Learning Beta
        for k in range(self.intNumTopic):
            for d in range(self.intNumDoc):
                for n in range(self.numWordPerDoc[d]):
                    self.beta[k][self.corpusList[d][n]] = self.beta[k][self.corpusList[d][n]] + self.phi[d][n][k]
            normalizeConstantBeta = sum(self.beta[k])
            for v in range(self.intUniqueWord):
                self.beta[k][v] = self.beta[k][v] / normalizeConstantBeta

        # Learning Alpha
        # Newton-Rhapson optimization
        for itr in range(self.numNewtonIteration):
            # Building Hessian Matrix and Derivative Vector
            H = zeros(shape=(self.intNumTopic,self.intNumTopic), dtype=float)
            g = zeros(shape=(self.intNumTopic), dtype=float)
            for k1 in range(self.intNumTopic):
                g[k1] = float(self.intNumDoc)*(digamma(sum(self.alpha))-digamma(self.alpha[k1]))
                for d in range(self.intNumDoc):
                    g[k1] = g[k1] + ( digamma(self.gamma[d][k1]) - digamma(sum(self.gamma[d])) )
                for k2 in range(self.intNumTopic):
                    H[k1][k2] = 0
                    if k1 == k2:
                        H[k1][k2] = H[k1][k2] - float(self.intNumDoc) * polygamma(1,self.alpha[k1])
                    H[k1][k2] = H[k1][k2] + float(self.intNumDoc) * polygamma(1,sum(self.alpha))

            deltaAlpha = np.dot(np.linalg.inv(H),g)
            self.alpha = self.alpha - deltaAlpha
```

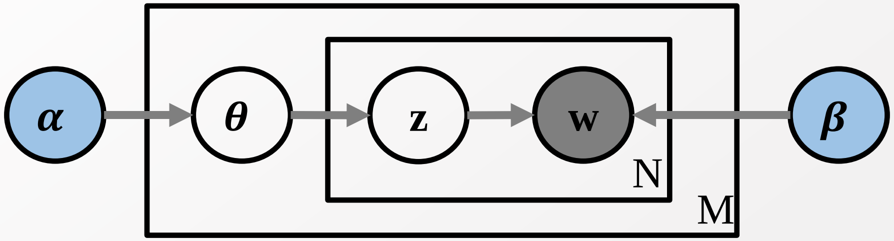
Evaluation of LDA

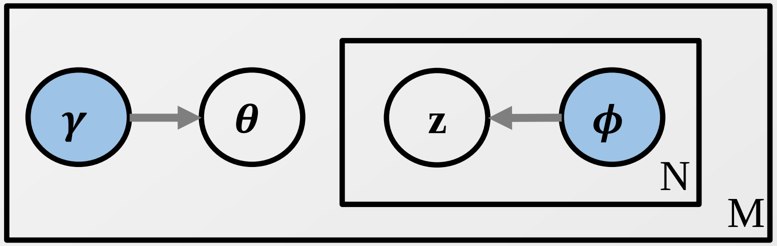


- Supervised learning evaluation
  - Training dataset for learning the parameters of a model
  - Testing dataset for evaluating a model with the trained parameter
- If a model becomes complicated to require hyper-parameters OR, if a model is deployed to the real-world
  - Training dataset for learning the parameters of a model
  - Validation dataset for tuning the hyper-parameters of a model
  - → Testing dataset for evaluating a model with the trained parameter
- Log likelihood of LDA
  - Training document sets
  - Testing document sets → Held-out log likelihood

KAIST

Stochastic Variational Inference





- Scalability problem of the vanilla version of variational inference
- When we have a large corpus, we have a problem in variational E-Step
  - Local variational parameters need to be updated per documents
  - Learn variational parameters only to the mini-batch documents
- Variational M-Step → Learning the global parameter per corpus
  - Learning should be normalized to consider the sampling rate
- Some might consider this as an online learning

# of Docs.

```
for d in range(self.intNumDoc):
    for iterationInternal in range(self.numInt
        expDigammaGammaDK = zeros(shape=(self.
        for k in range(self.intNumTopic):
            expDigammaGammaDK[k] = exp(digamma
        # Learning phi
        for n in range(self.numWordPerDoc[d]):
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.beta[
            normalizeConstantPhi = sum(self.ph
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.phi[d
        # Learning gamma
        for k in range(self.intNumTopic):
            self.gamma[d][k] = self.alpha[k]
            for n in range(self.numWordPerDoc[
                self.gamma[d][k] = self.gamma[
```

Hoffman, Matthew D., et al. "Stochastic variational inference." Journal of Machine Learning Research 14.1 (2013): 1303-1347.

Stochastic Variational Inference as Implementation

- Initialize $\lambda^{(0)}$ randomly
- Set the step-size schedule ρ_t appropriately
- Repeat
  - Sample a data point x_i uniformly from the dataset
  - Compute the local variational parameter of x_i

$$\phi = E_{\lambda^{(t-1)}}[\eta_g(x_i^{(N)}, z_i^{(N)})]$$

  - Compute the intermediate global parameters as through x_i is replicated N times

$$\hat{\lambda} = E_{\phi}[\eta_g(x_i^{(N)}, z_i^{(N)})]$$

  - Update the current estimate the global variational parameters

$$\lambda^{(t)} = (1 - \rho_t)\lambda^{(t-1)} + \rho_t\hat{\lambda}$$

- Until forever

```
for d in range(self.intNumDoc):
    for iterationInternal in range(self.numInt
        expDigammaGammaDK = zeros(shape=(self.
        for k in range(self.intNumTopic):
            expDigammaGammaDK[k] = exp(digamma
        # Learning phi
        for n in range(self.numWordPerDoc[d]):
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.beta[
            normalizeConstantPhi = sum(self.ph
            for k in range(self.intNumTopic):
                self.phi[d][n][k] = self.phi[d
        # Learning gamma
        for k in range(self.intNumTopic):
            self.gamma[d][k] = self.alpha[k]
            for n in range(self.numWordPerDoc[
                self.gamma[d][k] = self.gamma[
```

Sampled Instead of the whole corpus

KAIST

Black-Box Variational Inference

- $\ln P(w|\alpha,\beta) \geq L(\gamma,\phi|\alpha,\beta)$
  $= E_q(\log P(\theta|\alpha)) + E_q(\log P(z|\theta)) + E_q(\log P(w|z,\beta)) + H(q)$

$$= \sum_{d=1}^{M}\sum_{i=1}^{k}(\alpha_i - 1)(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})) + \log\Gamma\left(\sum_{i=1}^{K}\alpha_i\right) - \sum_{i=1}^{K}\log\Gamma(\alpha_i)$$
$$+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i}))$$
$$+ \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\beta_{i,w_{d,n}}$$
$$- \sum_{d=1}^{M}\sum_{i=1}^{k}(\gamma_{d,i} - 1)(-\psi\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \psi(\gamma_{d,i})) - \log\Gamma\left(\sum_{i=1}^{K}\gamma_{d,i}\right) + \sum_{i=1}^{K}\log\Gamma(\gamma_{d,i}) - \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{i=1}^{K}\phi_{d,n,i}\log\phi_{d,n,i}$$

Long derivation
+ Conjugate Prior Selection….

- One hurdle is the expectation on "…" with "q" distribution
  - $L(\gamma,\phi|\alpha,\beta) = E_{q(z|\lambda)}(\log P(x,z) - \log q(z))$
  - Previously, considering the expectation exactly from the mathematical derivation of PDF
  - Suggestion, sampling some points of Z and calculate the expectation through empirical simulations
    - Simply "weighted average" → weight = $q(z_{sampled}|\lambda)$

Ranganath, Rajesh, Sean Gerrish, and David M. Blei. "Black Box Variational Inference." AISTATS. 2014.

Black-Box Variational Inference as Implementation

- Initialize $\lambda^{(0)}$ randomly
- Set the step-size schedule ρ_t appropriately
- Repeat
  - for s = 1 to S
    - $z[s] \sim q$
  - Update the current estimate the variational parameters

$$\lambda^{(t)} = \lambda^{(t-1)} + \rho_t \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q(z[s] || \lambda) \ (\log p(x, z[s]) - \log q(z[s] || \lambda))$$

- Until λ does not change much

- Could have a high variance from the sampling based expectation
  - Further controls are needed

Further Readings

- Bishop Chapter 10
- Murphy Chapter 21