Unsupervised Clustering and Latent Dirichlet Allocation

Mark Gales

Lent 2014

Machine Learning for Language Processing: Lecture 8

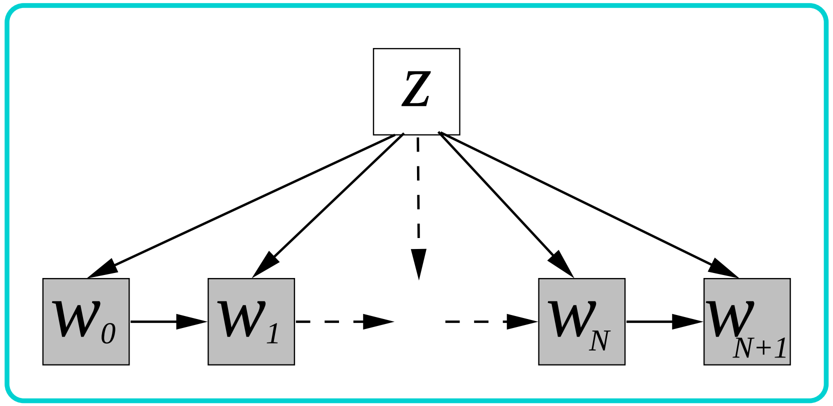
MPhil in Advanced Computer Science

Introduction

- So far described a number of models for word sequences
  - most common are based on N-grams and mixtures of N-grams
- In this lecture we will examine:
  - the application of N-grams (and extensions) to topic clustering;
  - an alternative generative model latent Dirichlet allocation
- The last slides will not be covered in the lectures - briefly mention
  - what happens as the number of clusters tends to infinity
  - infinite Gaussian mixture models
  - Dirichlet processes

Unsupervised Document Clustering

- Use a topic-dependent N-gram language model to perform clustering



  - word sequence $\boldsymbol{w} = \{w_1, \ldots, w_N\}$
  - start, w_0, and end w_{N+1} symbols added
  - z indicator variable over topics $\mathtt{s}_1, \ldots, \mathtt{s}_K$
  - plate repeated for every document

- Training data fully observed (supervised training) standard N-gram training
  - BUT interested in unsupervised clustering - indicator variable z unobserved
- Likelihood of one document with word sequence $\boldsymbol{w}$ can be written as

$$P(\boldsymbol{w}) = \sum_{k=1}^{K} P(\mathtt{s}_k) P(\boldsymbol{w}|\mathtt{s}_k) = \sum_{k=1}^{K} P(\mathtt{s}_k) \prod_{i=1}^{N+1} P(w_i|w_{i-1}, \mathtt{s}_k)$$

Unsupervised Clustering

- The likelihood has been written as marginalising over the latent variable
  - standard mixture model - use EM BUT interested in clustering documents
- Rather than using the "soft" assignment in EM, use a hard assignment

$$z_r^{[l]} = \underset{\mathrm{s}_k}{\operatorname{argmax}} \left\{ P(\mathrm{s}_k|\boldsymbol{\lambda}^{[l]}) P(\boldsymbol{w}^{(r)}|\mathrm{s}_k, \boldsymbol{\lambda}^{[l]}) \right\}$$

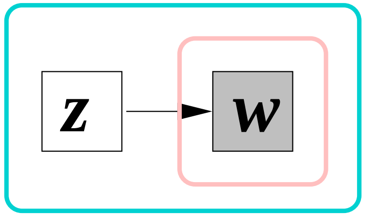
  - compare to EM where at iteration l compute $P(\mathrm{s}_k|\boldsymbol{w}, \boldsymbol{\lambda}^{[l]})$
  - allows documents to be clustered together (unique label for each document)

$$\text{For parameters of component } \mathrm{s}_k : \quad \boldsymbol{\lambda}_k^{[l+1]} = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \left\{ \prod_{r: z_r^{[l]} = \mathrm{s}_k} P(\boldsymbol{w}^{(r)}|\boldsymbol{\lambda}) \right\}$$

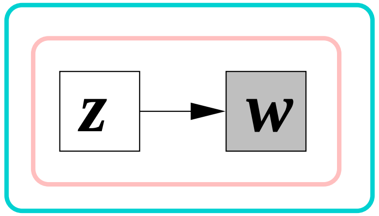
- Iterative procedure (similar to Viterbi training) - example of K-means clustering
  - can initialise model parameters by using K randomly selected examples

Language Model Components

- For simplicity only consider a unigram language model - for BNs below
  - inner plate repeated for each word (start/end symbols ignored as unigram)
  - outer plate for each document



$\sum_{k=1}^{K} P(\mathbf{s}_k) \prod_{i=1}^{N} P(w_i|\mathbf{s}_k)$



$\prod_{i=1}^{N} \left(\sum_{k=1}^{K} P(\mathbf{s}_k) P(w_i|\mathbf{s}_k) \right)$

- Interesting to contrast two forms of latent variable model
  - (left) indicator variable z over space of language models
  - (right) indicator variable z over space of language model predictions
- Possible to combine latent variable models (a hierarchical model)

Bayesian Approaches

- Consider a generative model for class ω_j (supervised training)
  - training data: $\mathcal{D} = \{\boldsymbol{x}_1 \ldots, \boldsymbol{x}_n\}$
  - parametric form of distribution (the model), $\mathcal{M}$, is known (and fixed) with (unknown) parameters $\boldsymbol{\theta}$
- Rather than estimating the parameters of the model, $\hat{\boldsymbol{\theta}}$, use a distribution
  - from training data obtain the posterior distribution over model parameters

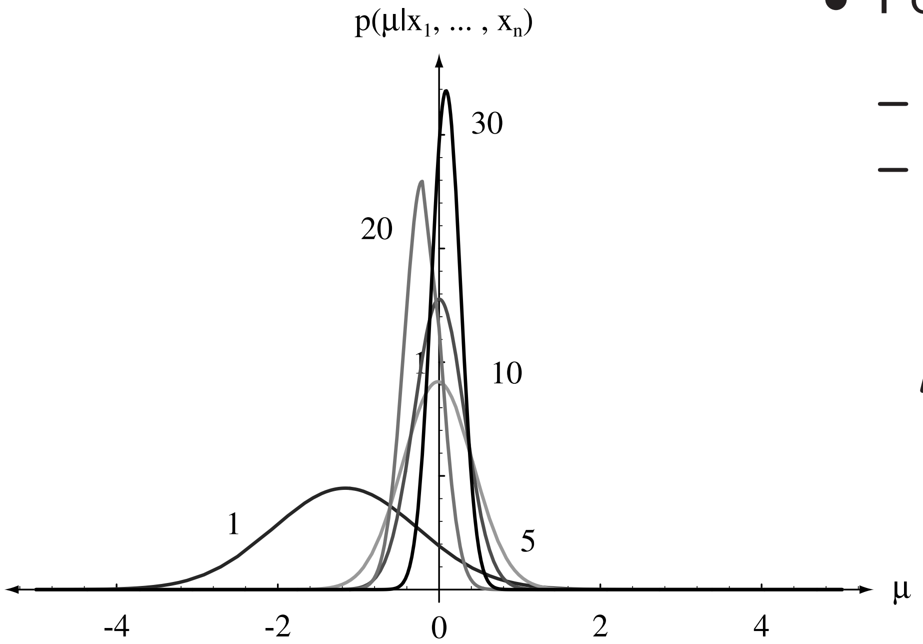
$$p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M}) = \frac{p(\mathcal{D}|\boldsymbol{\theta},\mathcal{M})p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})} \quad \text{Note MAP } \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left\{p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M})\right\}$$

  - $p(\boldsymbol{\theta}|\mathcal{M})$ is the prior distribution over the model parameters
- Likelihood of an observation $\boldsymbol{x}$ then computed as

$$p(\boldsymbol{x}|\mathcal{D},\mathcal{M}) = \int p(\boldsymbol{x}|\boldsymbol{\theta},\mathcal{M})p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M})d\boldsymbol{\theta} \text{ Note MAP } p(\boldsymbol{x}|\mathcal{D},\mathcal{M}) \approx p(\boldsymbol{x}|\hat{\boldsymbol{\theta}},\mathcal{M})$$

Distribution of the Mean Estimate

- Consider Bayesian estimation of the mean $\boldsymbol{\mu}$ of a Gaussian distribution



- Posterior $p(\boldsymbol{\mu}|\mathcal{D},\mathcal{M})$ variation (from DHS)
  - Gaussian distributed - $\boldsymbol{\mu} \sim \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$
  - prior $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\rm p})$

$$\hat{\boldsymbol{\mu}} = \left(n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_{\rm p}^{-1}\right)^{-1}\left(\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{n}\boldsymbol{x}_i\right)$$

$$\hat{\boldsymbol{\Sigma}} = \left(n\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_{\rm p}^{-1}\right)^{-1}$$

- Shape of posterior distribution changes as n increases
  - the posterior becomes more sharply peaked (reduced variance)
  - MAP estimate (the mode of the distribution) moves towards ML estimate

Latent Dirichlet Allocation

- Interested in applying Bayesian approaches to language processing
  - consider a mixture-of-unigrams language model

$$P(\boldsymbol{w}) = \prod_{i=1}^{N} \sum_{k=1}^{K} P(\mathbf{s}_k) P(w_i|\mathbf{s}_k)$$

where $P(\mathbf{s}_k)$ is estimated from training data
  - alternatively consider a Bayesian version over the topic priors

$$P(\boldsymbol{w}|\boldsymbol{\alpha}) = \int p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \left(\prod_{i=1}^{N} \sum_{k=1}^{K} P(\mathbf{s}_k|\boldsymbol{\theta}) P(w_i|\mathbf{s}_k) \right) d\boldsymbol{\theta}$$

where $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$ obtained from the training data

What form of distribution/latent variable model to use?

(Reminder) Multinomial Distribution

- **Multinomial** distribution: $x_i \in \{0, \dots, n\}$

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{n!}{\prod_{i=1}^{d} x_i!} \prod_{i=1}^{d} \theta_i^{x_i}, \qquad n = \sum_{i=1}^{d} x_i, \quad \sum_{i=1}^{d} \theta_i = 1, \quad \theta_i \geq 0$$

- When $n = 1$ the multinomial distribution simplifies to

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{d} \theta_i^{x_i}, \quad \sum_{i=1}^{d} \theta_i = 1, \quad \theta_i \geq 0$$

  – a unigram language model with 1-of-V coding ($d = V$ the vocabulary size)
  – x_i indicates word i of the vocabulary observed, $x_i = \begin{cases} 1, & \text{word } i \text{ observed} \\ 0, & \text{otherwise} \end{cases}$
  – $\theta_i = P(w_i)$ the probability that word i is seen

(More) Probability Distributions

- **Dirichlet** (continuous) distribution with parameters $\boldsymbol{\alpha}$

$$p(\boldsymbol{x}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^{d}\alpha_i)}{\prod_{i=1}^{d}\Gamma(\alpha_i)}\prod_{i=1}^{d} x_i^{\alpha_i-1}; \quad \text{for "observations"}: \sum_{i=1}^{d} x_i = 1, \quad x_i \geq 0$$

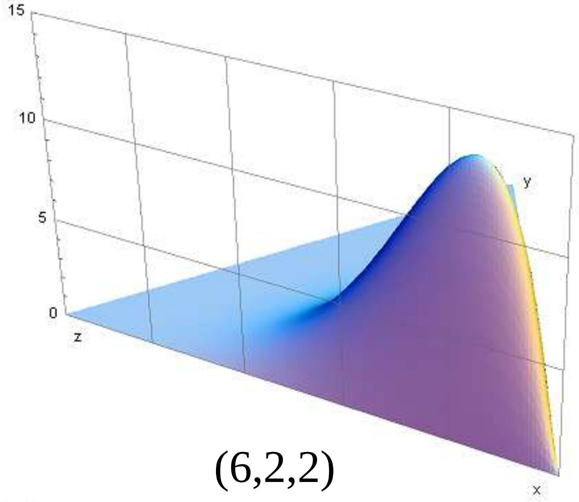
  - $\Gamma()$ is the Gamma distribution
  - Conjugate prior to the multinomial distribution
    (form of posterior $p(\boldsymbol{\theta}|\mathcal{D},\mathcal{M})$ is the same as the prior $p(\boldsymbol{\theta}|\mathcal{M})$)

- **Poisson** (discrete) distribution with parameter ξ

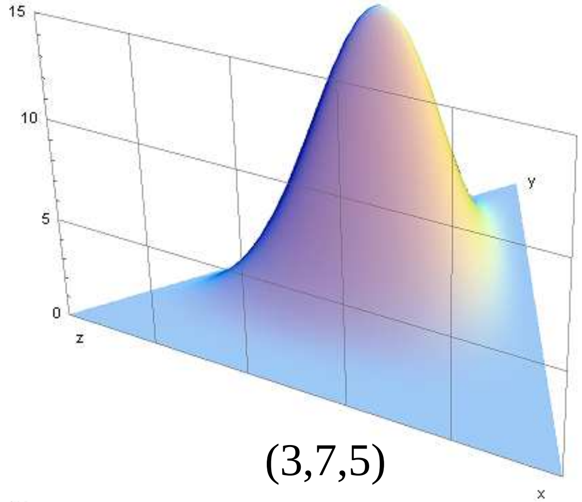
$$P(x|\xi) = \frac{\xi^x \exp(-\xi)}{x!}$$

  - probability of the number of events in a specific interval
  - here used for number of words in a document

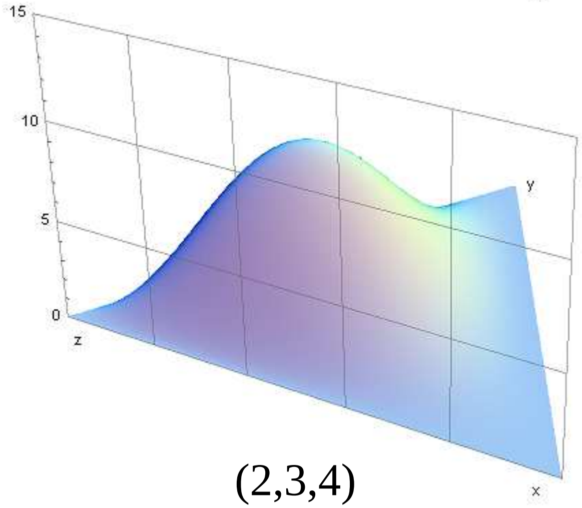
Dirichlet Distribution Example



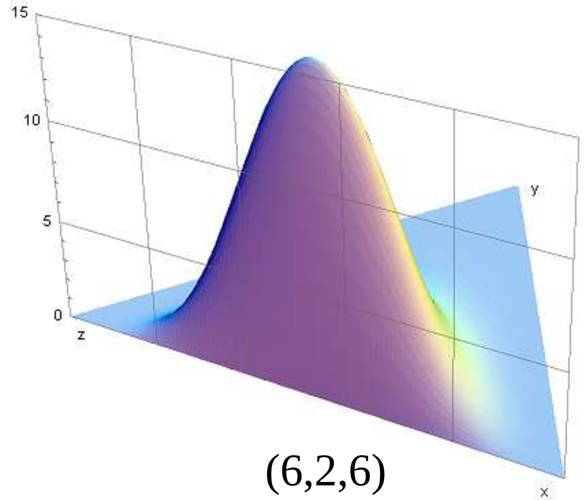
(6,2,2)



(3,7,5)



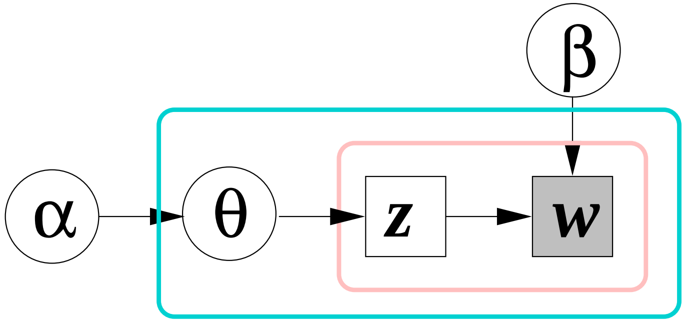
(2,3,4)



(6,2,6)

- Note: $x + y + z = 1$
- Vector: $(\alpha_1, \alpha_2, \alpha_3)$

Latent Dirichlet Allocation Bayesian Network



- Bayesian Network for Latent Dirichlet Allocation (LDA) is shown above
  - explicitly includes dependence on model parameters $\boldsymbol{\lambda} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$

$$P(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \int p(\boldsymbol{\theta}|\boldsymbol{\alpha}) \left(\prod_{i=1}^{N} \sum_{k=1}^{K} P(\mathtt{s}_k|\boldsymbol{\theta}) P(w_i|\mathtt{s}_k, \boldsymbol{\beta}) \right) d\boldsymbol{\theta}$$

  - z is an indicator variable for one of the K topics: $\{\mathtt{s}_1, \ldots, \mathtt{s}_K\}$
  - inner plate is repeated for N words, outer plate is repeated for R documents
- Bayesian approach learn posterior distribution of the component priors, $\boldsymbol{\theta}$,
  - Dirichlet distribution $p(\boldsymbol{\theta}|\boldsymbol{\alpha}) = p(\boldsymbol{\theta}|\mathcal{D}, \mathcal{M})$, and noting $P(\mathtt{s}_k|\boldsymbol{\theta}) = \theta_k$

LDA Generative Process

- LDA assumes the following generative process for the words $\boldsymbol{w}$ is a document
  1. Choose length of document - $N \sim$ Poisson(ξ)
  2. Choose parameters of multinomial - $\boldsymbol{\theta} \sim$ Dir$(\boldsymbol{\alpha})$
  3. For each of the N words w_n:
     (a) Choose topic: $z_n \sim$ Multinomial$(\boldsymbol{\theta})$
     (b) Choose word: w_n from multinomial probability conditioned on topic z_n with parameters $\boldsymbol{\beta}$
- The parameters that need to be estimated for LDA
  - $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_K\}$: K parameters
    the prior distribution over the multinomial parameters
  - $\boldsymbol{\beta} = \{\beta_{11}, \beta_{1V}, \ldots, \beta_{K1}, \ldots, \beta_{KV}\}$: KV parameters
    Note $\beta_{ki} \geq 0, \sum_{i=1}^{V} \beta_{ki} = 1 \;\; \forall k, i$ - this is the equivalent of topic-unigrams

LDA Parameter Estimation

- Given corpus of documents $\{\boldsymbol{w}^{(1)}, \ldots, \boldsymbol{w}^{(R)}\}$ need to estimate $\boldsymbol{\alpha}, \boldsymbol{\beta}$

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{r=1}^{R} \log\left(P(\boldsymbol{w}^{(r)}|\boldsymbol{\alpha}, \boldsymbol{\beta})\right)$$

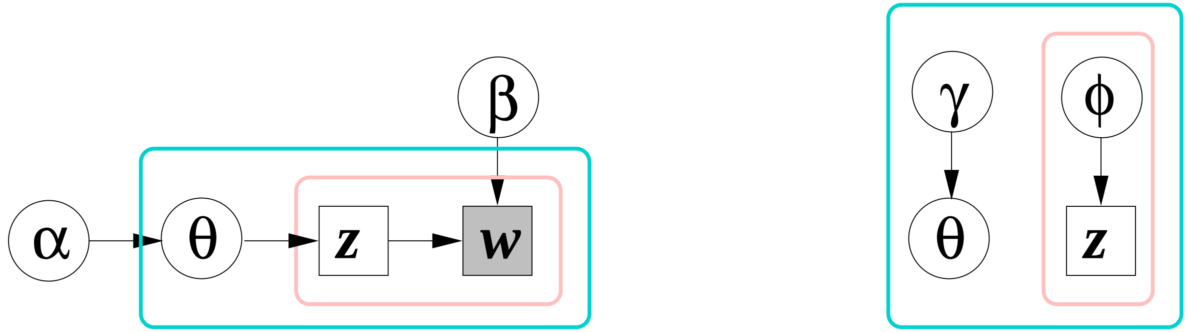
- Unfortunately likelihood calculation is intractable need to compute

$$P(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)} \int \left(\prod_{k=1}^{K}\theta_k^{\alpha_k-1}\right)\left(\prod_{i=1}^{N}\sum_{k=1}^{K}\theta_k\prod_{j=1}^{V}(\beta_{kj})^{I(w_i,j)}\right) d\boldsymbol{\theta}$$

  - word indicator: $I(w_i, j) = \begin{cases} 1, & w_i = \text{ word } j \text{ in the vocabulary} \\ 0, & \text{otherwise} \end{cases}$
  - $P(\mathtt{s}_k|\boldsymbol{\theta}) = \theta_k$ and $P(w_i|\mathtt{s}_k, \boldsymbol{\beta}) = \beta_{ki}$

- Not possible to use EM: require $p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})}{P(\boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})}$

Variational EM (Reference)



Latent Dirichlet Allocation — Variational Approximation

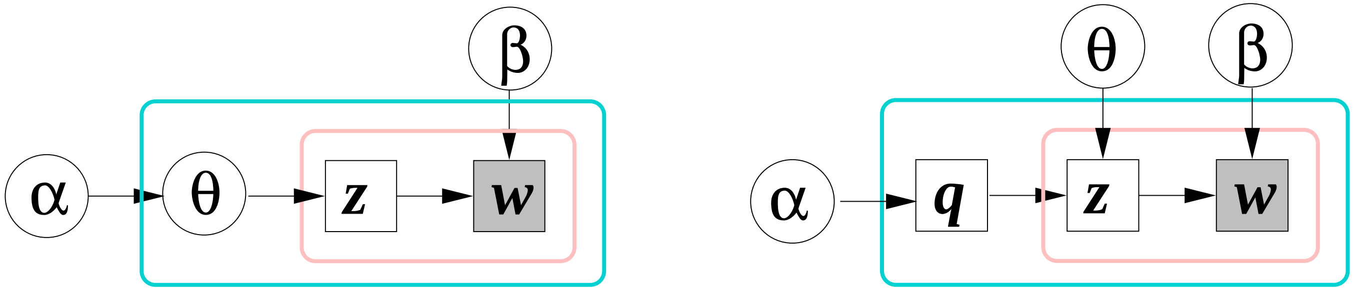
- LDA can be estimated using variational EM with the mean-field approximation
  - use a variational approximation $q(\boldsymbol{\theta}, \boldsymbol{z}|\gamma, \phi)$ - see diagram on right

$$q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{\gamma}, \boldsymbol{\phi}) = q(\boldsymbol{\theta}|\gamma) \prod_{i=1}^{N} q(z_i|\phi_i)$$

  - parameters - minimise KL-divergence: $\mathsf{KL}(q()||p()) = \int p(x) \log\left(q()/p()\right) dx$

$$\{\boldsymbol{\gamma}^{[l]}, \boldsymbol{\phi}^{[l]}\} = \operatorname*{argmin}_{\boldsymbol{\gamma}, \boldsymbol{\phi}} \left\{ \mathsf{KL}(q(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{\gamma}, \boldsymbol{\phi})||p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}^{[l]}, \boldsymbol{\beta}^{[l]}) \right\}$$

LDA and Topic Mixture of Unigrams



Latent Dirichlet Allocation   Topic Mixture of Unigrams

- Latent Dirichlet allocation - parameters $K(1+V)$ - continuous mixture

$$P(\boldsymbol{w}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \frac{\Gamma(\sum_{k=1}^{K}\alpha_k)}{\prod_{k=1}^{K}\Gamma(\alpha_k)} \int \left(\prod_{k=1}^{K}\theta_k^{\alpha_k-1}\right)\left(\prod_{i=1}^{N}\sum_{k=1}^{K}\theta_k\prod_{j=1}^{V}(\beta_{kj})^{I(w_i,j)}\right) d\boldsymbol{\theta}$$

- Topic mixture of unigrams - parameters $M+K(M+V)$ - discrete mixture

$$P(\boldsymbol{w}|\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\theta}) = \sum_{m=1}^{M}\alpha_m\left(\prod_{i=1}^{N}\sum_{k=1}^{K}\theta_{mk}\prod_{j=1}^{V}(\beta_{kj})^{I(w_i,j)}\right)$$

Properties of LDA

- LDA is a generative model of a document
  - compact model of the data
  - infinite component priors represented by K-parameter distribution $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$
  - can be combined with standard language model smoothing for $\boldsymbol{\beta}$
- Consider using LDA as a generative model for classification for
  - for each class ω_j estimate $\{\boldsymbol{\alpha}^{(j)}, \boldsymbol{\beta}^{(j)}\}$ using all documents from class ω_j
  - estimate the prior for each class $P(\omega_j)$
  - perform classification for sequence $\boldsymbol{w}$ based on

$$\hat{\omega} = \underset{\omega_j}{\operatorname{argmax}} \left\{ P(\omega_j) P(\boldsymbol{w}|\boldsymbol{\alpha}^{(j)}, \boldsymbol{\beta}^{(j)}) \right\}$$

- LDA has also been used for a range of language processing applications

How Many Topics?

- So far not consider the number of topics, K, for LDA
  - how about using a Bayesian approach

$$P(\boldsymbol{w}|\boldsymbol{\alpha}^{(1)},\ldots,\boldsymbol{\alpha}^{(\infty)}) =$$
$$\sum_{K=1}^{\infty} P(K) \int p(\boldsymbol{\theta}^{(K)}|\boldsymbol{\alpha}^{(K)}) \left(\prod_{i=1}^{N} \sum_{k=1}^{K} P(\mathtt{s}_k|\boldsymbol{\theta}^{(K)}) P(w_i|\mathtt{s}_k,\boldsymbol{\beta}) \right) d\boldsymbol{\theta}^{(K)}$$

  - each of the priors of infinite mixture models has a Dirichlet distribution

- There's a infinite number of components
  - unfortunately an infinite number of parameters $\boldsymbol{\alpha}^{(1)},\ldots,\boldsymbol{\alpha}^{(\infty)},\boldsymbol{\beta}$ to train

Can we keep the infinite model, but make it tractable?

- Non-parametric Bayesian approaches: (hierarchical) Dirichlet Processes

Gaussian Mixture Models

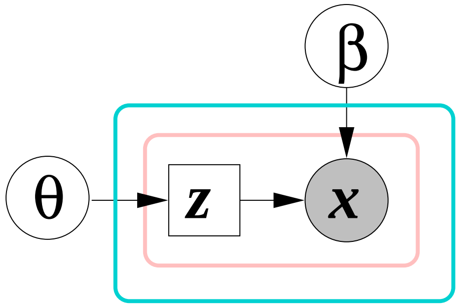
- Consider simpler (illustrative) example - the Infinite Gaussian Mixture Model
- Standard form of M-component Gaussian Mixture Model (GMM) is

$$p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\beta}) = \sum_{m=1}^{M} P(\mathsf{c}_m|\boldsymbol{\theta})p(\boldsymbol{x}|\mathsf{c}_m,\boldsymbol{\beta}) = \sum_{m=1}^{M} P(\mathsf{c}_m|\boldsymbol{\theta})\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu}_m,\boldsymbol{\Sigma}_m)$$

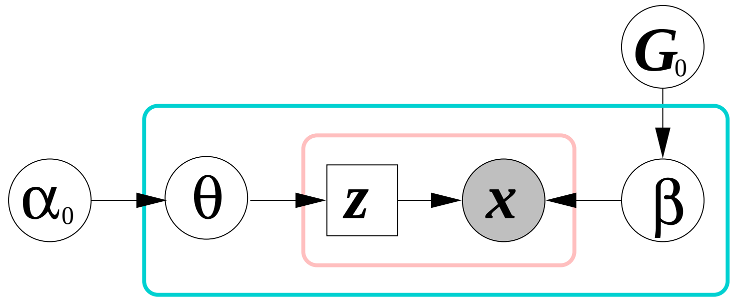
Interested in what happens as $M \rightarrow \infty$?

- Must use Bayesian approaches as the number of parameters infinite
  - what sort of prior distributions to use?
- Introduce prior distributions $\{\alpha_0, \boldsymbol{\beta}\}$
  - α_0 - prior parameter for the Dirichlet distribution
  - $\boldsymbol{\beta}$ - prior distribution for Gaussian components

Infinite Gaussian Mixture Models



Gaussian Mixture Model



Infinite Gaussian Mixture Model

- From the Bayesian network above

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0) = \int\int p(\boldsymbol{\theta}|\alpha_0)p(\boldsymbol{\beta}|G_0)\prod_{i=1}^{N}\sum_{m=1}^{M} P(\mathsf{c}_m|\boldsymbol{\theta})p(\boldsymbol{x}_i|\mathsf{c}_m, \boldsymbol{\beta})d\boldsymbol{\theta}d\boldsymbol{\beta}$$

where: $\boldsymbol{\theta}|\alpha_0 \sim \text{Dirichlet}\left(\frac{\alpha_0}{M}, \ldots, \frac{\alpha_0}{M}\right)$; $\quad \boldsymbol{\beta}_m \sim G_0$; $\quad \mathsf{c}_m|\boldsymbol{\theta} \sim \text{Multinomial}(\boldsymbol{\theta})$

- Estimate the hyper-parameters from training data, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ - maximise

$$\mathcal{L}(\alpha_0, G_0) = \log\left(p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0)\right)$$

THE END - SLIDES ARE FOR REFERENCE FROM HERE

Sample-Based Approximations

- Simple approach to approximate integrals is to use

$$\int f(\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{\theta})d\boldsymbol{x} \approx \frac{1}{N}\sum_{i=1}^{N} f(\boldsymbol{x}^{(i)}); \quad \boldsymbol{x}^{(i)} \sim p(\boldsymbol{\theta})$$

  - as $N \to \infty$ the approximation will become an equality
  - N needs to increase as dimension $\boldsymbol{x}$ increases - need to sample the space

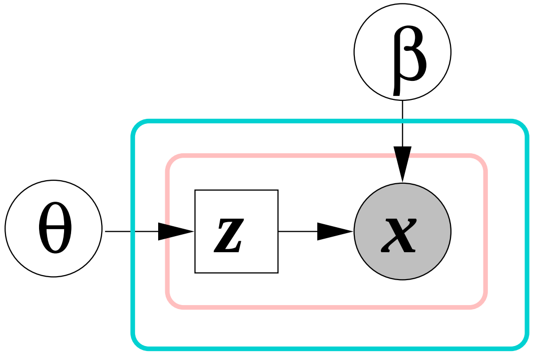
marginalising is simply sampling

- If a sample can't be directly generated from the multivariate distribution $p(\boldsymbol{\theta})$
  - Gibbs sampling from conditional distributions can be used
  - assume that we have samples $x_1^{(i)}, \ldots, x_{k-1}^{(i)}, x_{k+1}^{(i)}, x_d^{(i)}$ generate $x_k^{(i)}$
  - sample from

$$p(x_k|x_1, \ldots, x_{k-1}, x_{k+1}, x_d, \boldsymbol{\theta})$$

  - assumes that possible to sample from the conditional

Gaussian Mixture Model Sampling



$$p(\boldsymbol{x}|\boldsymbol{\theta},\boldsymbol{\beta}) = \sum_{m=1}^{M} P(\mathsf{c}_m|\boldsymbol{\theta})p(\boldsymbol{x}|\mathsf{c}_m,\boldsymbol{\beta})$$
$$= \sum_{m=1}^{M} P(\mathsf{c}_m|\boldsymbol{\theta})p(\boldsymbol{x}|\boldsymbol{\beta}_m)$$

- Sampling approach from distribution comprises
  1. Generate component indicator $z_n \sim \text{Multinomial}(\boldsymbol{\theta})$
  2. Generate observation: $\boldsymbol{x}_n \sim \mathcal{N}\left(\boldsymbol{\beta}_{z_n}\right)$

- Simple to train using EM (see lecture 5)
  - non-Bayesian - point estimates of the model parameters $\{\boldsymbol{\theta},\boldsymbol{\beta}\}$
  - number of components M fixed

IGMM Sampling Procedure

How to generate samples from infinite components?

- Gibb's Sampling process to generate $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ for N samples
  1. Generate component indicator $z_n|\boldsymbol{z}_{-n}$ ($\boldsymbol{z}_{-n} = \{z_1, \ldots, z_{n-1}\}$)

$$P(z_n = \mathsf{c}_j|\boldsymbol{z}_{-n}, \alpha_0) = \begin{cases} \frac{\sum_{i=1}^{n-1} \mathbf{1}(z_i, \mathsf{c}_j)}{n-1+\alpha_0} & \mathsf{c}_j \text{ represented} \\ \\ \frac{\alpha_0}{n-1+\alpha_0} & \mathsf{c}_j \text{ unrepresented} \end{cases}$$

  2. If component indicted by z_n is unrepresented: $\boldsymbol{\beta}_{z_n} \sim G_0$
  3. Generate observation: $\boldsymbol{x}_n \sim \mathcal{N}(\boldsymbol{\beta}_{z_n})$

- At most N of the infinite possible samples represented

IGMM Hyper-Parameter Training

- Using Gibb's sampling to training hyper-parameters of G_0
  - sampling process to generate $\{\boldsymbol{z}^{(l)}, \boldsymbol{\beta}^{(l)}\}$ for these N samples, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$

1. Generate component indicators $\boldsymbol{z}^{(l)}|\boldsymbol{z}^{(l)}_{-n}, \boldsymbol{\beta}^{(l-1)}, \boldsymbol{x}_n$ (dropped dependence)

$$P(z_n^{(l)} = \mathsf{c}_j|\alpha_0^{(l-1)}, G_0^{(l-1)}) \propto \begin{cases} \frac{\sum_{i=1}^{n-1} \mathbf{1}(z_i^{(l)}, \mathsf{c}_j)}{n-1+\alpha_0^{(l-1)}} p(\boldsymbol{x}_n|\boldsymbol{\beta}_j^{(l-1)}) & \mathsf{c}_j \text{ represented} \\ \frac{\alpha_0^{(l-1)}}{n-1+\alpha_0^{(l-1)}} \int p(\boldsymbol{x}_n|\boldsymbol{\beta})p(\boldsymbol{\beta}|G_0^{(l-1)})d\boldsymbol{\beta} & \mathsf{c}_j \text{ unrepresented} \end{cases}$$

2. Foreach `represented` component $\mathsf{c}_j, j \in \{1, \ldots, k_{\texttt{rep}}\}$

   sample component mean and variance: $\boldsymbol{\beta}_j^{(l)} = \{\boldsymbol{\mu}_j^{(l)}, \boldsymbol{\Sigma}_j^{(l)}\} \sim G_0^{(l-1)}$

3. Update hyper-parameters $\{\alpha_0^{(l)}, G_0^{(l)}\}$ using component values $\boldsymbol{\beta}_1^{(l)}, \ldots, \boldsymbol{\beta}_{k_{\texttt{rep}}}^{(l)}$
   (a) increment the counter $l = l + 1$

IGMM Classification

- So how can we perform classification - need the class-likelihood (prior simple)
  - consider observation $\boldsymbol{x}$ given training data for class ω_j: $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$

$$p(\boldsymbol{x}|\mathcal{D}, \alpha_0, G_0) = \frac{p(\boldsymbol{x}, \mathcal{D}|\alpha_0, G_0)}{p(\mathcal{D}|\alpha_0, G_0)} = \frac{p(\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0)}{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0)}$$

  - clearly a non-parametric model - explicit dependence on training observations

- Use a sample-based approximations for numerator/denominator thus

$$p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0) \approx \frac{1}{L}\sum_{l=1}^{L}\prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{z}^{(l)}, \boldsymbol{\beta}^{(l)})$$

  - follow hyper-parameter training without update to hyper-parameters
  - similar for $p(\boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N|\alpha_0, G_0)$

Dirichlet Processes

- Dirichlet Processes are a generalisation of the Dirichlet distribution
  - both can be viewed as distributions over distributions
  - BUT Dirichlet processes act over infinite components

- Model has the form

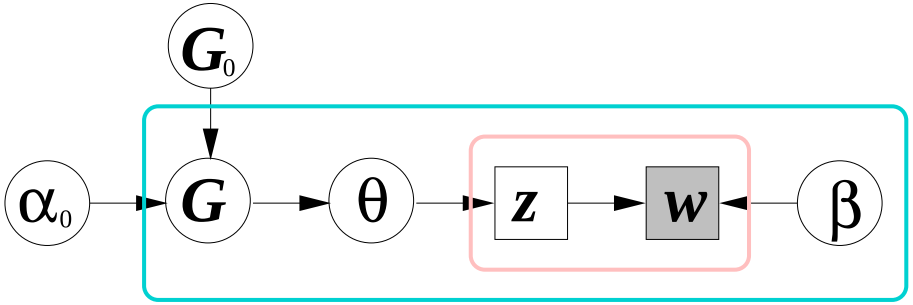
$$G \sim \mathrm{DP}(\alpha_0, G_0);$$

  - G_0 is the base measure (distribution)
  - α_0 is the concentration parameter

- If the measure is parametrised with $\boldsymbol{\theta}$
  - each draw of G from G_0 yields $\boldsymbol{\theta}_k \sim G_0$
  - $\delta_{\boldsymbol{\theta}_k}$ indicates a δ function at the parameters for draw k , $\boldsymbol{\theta}_k$
  - Reminder:

$$\int f(\boldsymbol{x}|\boldsymbol{\theta})\delta_{\boldsymbol{\theta}_k} d\boldsymbol{\theta} = f(\boldsymbol{x}|\boldsymbol{\theta}_k)$$

Example Dirichlet Process



- The likelihood of the word sequence $\boldsymbol{w} = \{w_1, \ldots, w_N\}$ can be expressed as

$$P(\boldsymbol{w}|\alpha_0, G_0) = \int P(G|\alpha_0, G_0) \int P(\boldsymbol{\beta}) \int p(\boldsymbol{\theta}|G) P(\boldsymbol{w}|\boldsymbol{\theta}, G, \boldsymbol{\beta}) d\boldsymbol{\theta} d\boldsymbol{\beta} dG$$

  – G is distributed according to the Dirichlet Process DP(α_0, G_0)
  – if K is the number of components associated with the G

$$P(\boldsymbol{w}|\boldsymbol{\theta}, G, \boldsymbol{\beta}) = \prod_{i=1}^{N} \sum_{k=1}^{K} P(\mathtt{s}_k|\boldsymbol{\theta}) P(w_i|\mathtt{s}_k, \boldsymbol{\beta})$$

- BUT can't share cluster parameters ($\boldsymbol{\beta}$) across different draws

  – no relationship between clusters ... hierarchical Dirichlet priors

Dirichlet Processes Generative Process

- Can't directly sample from Dirichlet process - use Gibb's sampling
  - behaviour of $\boldsymbol{\theta}_n$ given previous $n-1$ draw $\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{n-1}$

$$\boldsymbol{\theta}_n | \boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{n-1}, \alpha_0, G_0 \sim \frac{\alpha_0}{n-1+\alpha_0} G_0 + \sum_{i=1}^{n-1} \frac{1}{n-1+\alpha_0} \delta_{\boldsymbol{\theta}_i}$$

  - this is the equivalent of the generative process where

$$\boldsymbol{\theta}_n = \begin{cases} \boldsymbol{\theta}_i & \text{with probability } \frac{1}{n-1+\alpha_0} \text{ for } 1 \leq i \leq (n-1) \\ \boldsymbol{\theta} \sim G_0() & \text{with probability } \frac{\alpha_0}{n-1+\alpha_0} \end{cases}$$

- A draw from a Dirichlet process (stick-breaking representation)

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\boldsymbol{\theta}_k}; \quad \boldsymbol{\theta}_k \sim G_0; \quad \psi_k \sim \text{Beta}(1, \alpha_0); \quad \pi_k = \psi_k \prod_{i=1}^{k-1} (1 - \psi_k)$$

  - Google Chinese Restaurant Process for a simple example