Structured Bayesian Nonparametric Models with Variational Inference

ACL Tutorial

Prague, Czech Republic

June 24, 2007

Percy Liang and Dan Klein

University of California Berkeley

Probabilistic modeling of NLP

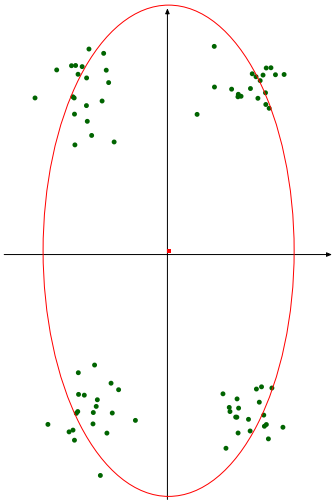
- Document clustering
- Topic modeling
- Language modeling
- Part-of-speech induction
- Parsing and grammar induction
- Word segmentation
- Word alignment
- Document summarization
- Coreference resolution
- etc.

Recent interest in Bayesian nonparametric methods

Probabilistic modeling is a core technique for many NLP tasks such as the ones listed. In recent years, there has been increased interest in applying the benefits of Bayesian inference and nonparametric models to these problems.

A motivating example

How many clusters?

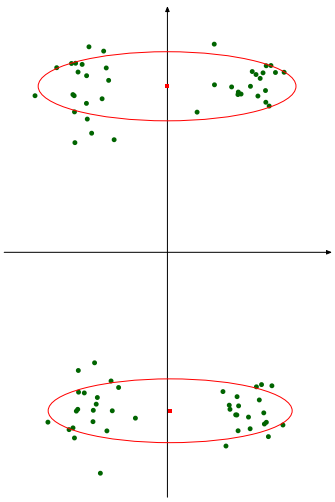


K	training log-likelihood	test log-likelihood
1	-364	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

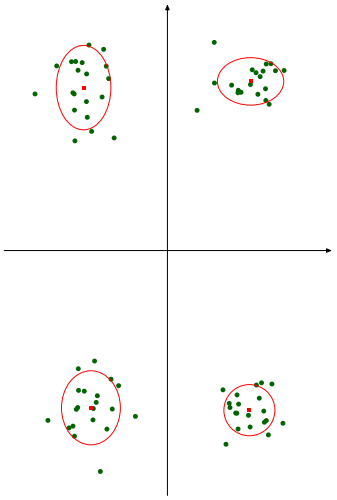


K	training log-likelihood	test log-likelihood
1	-364	
2	-204	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

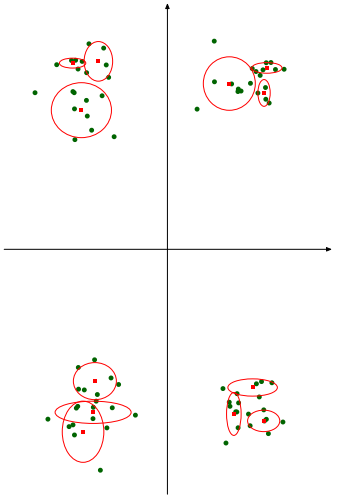


K	training log-likelihood	test log-likelihood
1	-364	
2	-204	
4	-82	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

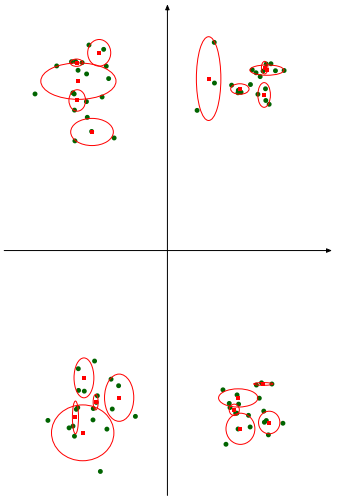


K	training log-likelihood	test log-likelihood
1	-364	
2	-204	
4	-82	
12	21	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

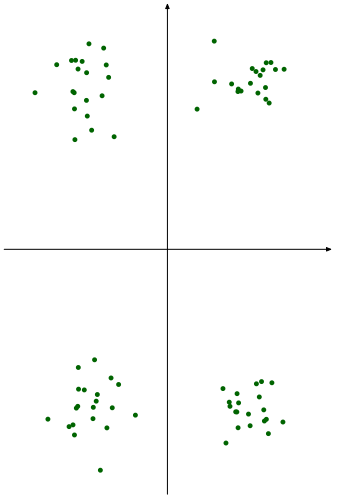


K	training log-likelihood	test log-likelihood
1	-364	
2	-204	
4	-82	
12	21	
20	86	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?



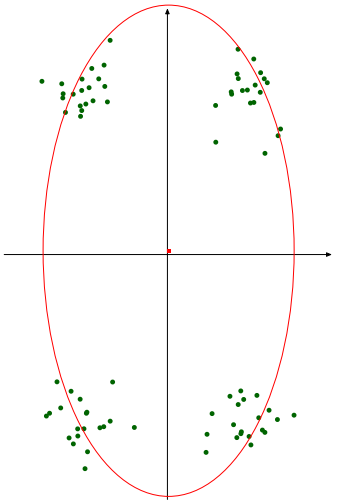
K	training log-likelihood	test log-likelihood
1	-364	
2	-204	
4	-82	
12	21	
20	86	

3

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

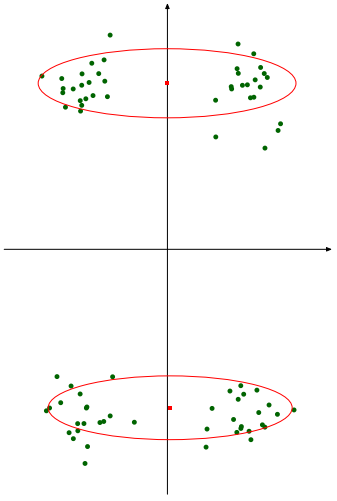


K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	
4	-82	
12	21	
20	86	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

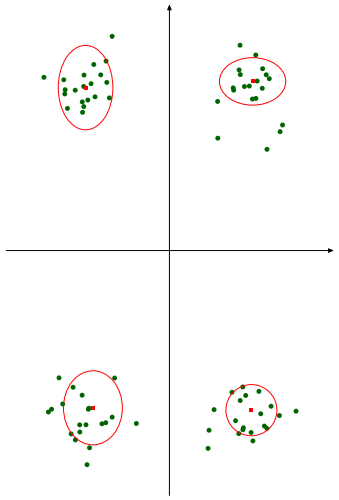


K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	-206
4	-82	
12	21	
20	86	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

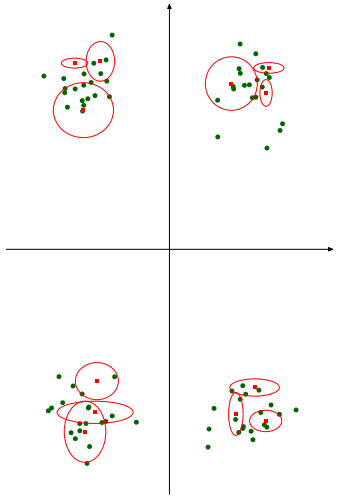


K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	-206
4	-82	-128
12	21	
20	86	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

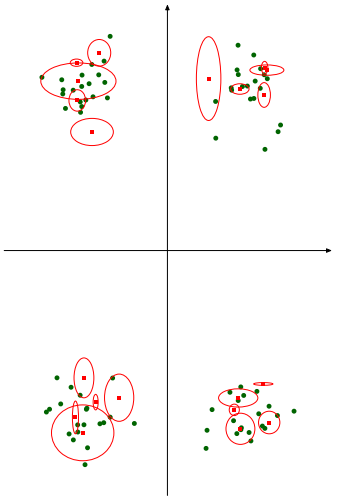


K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	-206
4	-82	-128
12	21	-147
20	86	

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?

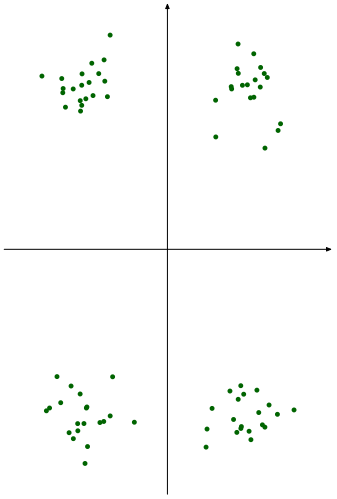


K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	-206
4	-82	-128
12	21	-147
20	86	-173

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

A motivating example

How many clusters?



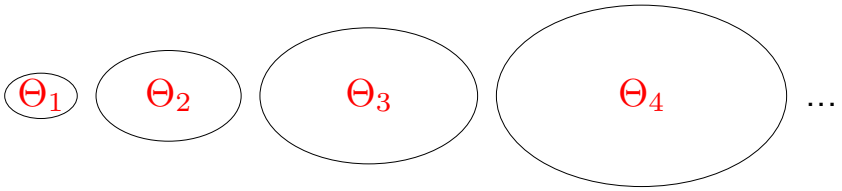
K	training log-likelihood	test log-likelihood
1	-364	-368
2	-204	-206
4	-82	-128
12	21	-147
20	86	-173

3

In this example, four mixtures of Gaussians were generated and EM was used to learn a clustering. The example shows the fundamental problem of using maximum likelihood as a criterion for selecting the complexity of a model. As the complexity (number of clusters) increases, the training likelihood strictly improves. However, the test likelihood improves initially but then degrades after a certain point.

Model selection: traditional solutions

Define an sequence of models of increasing complexity



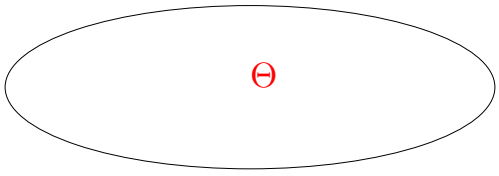
- Cross-validation: select a model based on likelihood on heldout set
- Bayesian model selection: use marginal likelihood or minimum description length

Discrete optimization: combinatorial search over models

The traditional approach to select the model of the right complexity is to search through a discrete space of models, using a complexity penalty as a criteria for guiding the search. This might be difficult if we need to integrate out model parameters (e.g., in computing marginal likelihood).

A nonparametric Bayesian approach

Define **one** model with an infinite number of clusters but penalize the use of more clusters.



This penalty is accomplished with the Dirichlet process.

Continuous optimization: model complexity governed by magnitude of parameters

Advantages:

- Works on structured models
- Allows EM-like tools

The nonparametric Bayesian approach does not choose between different models but instead defines one model, thus blurring the distinction between model and parameters. The advantage of this approach is that we will be able to use familiar tools similar to the EM algorithm for parameter estimation.

True or false?

1. ~~Being Bayesian is just about having priors.~~
   Being Bayesian is about managing uncertainty.
2. ~~Bayesian methods are slow.~~
   Bayesian methods can be just as fast as EM.
3. ~~Nonparametric means no parameters.~~
   Nonparametric means the number of effective parameters grows adaptively with the data.
4. ~~Variational inference is complicated and foreign.~~
   Variational inference is a natural extension of EM.

There are many myths about Bayesian methods being slow and difficult to use. We will show that with suitable approximations, we can get the benefits of being Bayesian without paying an enormous cost.

Tutorial outline

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

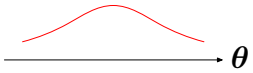
Two paradigms

Traditional (frequentist) approach

data $\mathbf{x}$ —maximum likelihood→ $\boldsymbol{\theta}^*$

Bayesian approach

data $\mathbf{x}$ —Bayesian inference→ $q(\boldsymbol{\theta})$



An example:

coin flips (data): $\mathbf{x} = (\mathrm{H})(\mathrm{T})(\mathrm{H})(\mathrm{H})$

probability of (H) (parameter): $\boldsymbol{\theta} = \frac{3}{4}?\ \frac{4}{6}?$

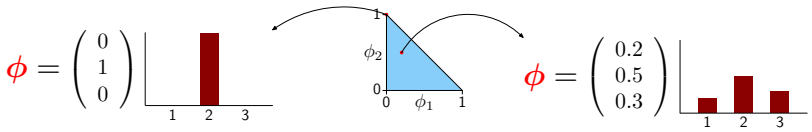
We need a notion of distributions over parameters...

The traditional frequentist approach is to estimate a single best parameter $\boldsymbol{\theta}^*$ given the data $\mathbf{x}$, using, for example, maximum likelihood. However, in reality, the data is noisy so actually we are uncertain about our parameter estimate $\boldsymbol{\theta}^*$. Therefore, we should maybe return several $\boldsymbol{\theta}$s in order to reflect our uncertainty about the best $\boldsymbol{\theta}$. This is a bottom-up motivation for the Bayesian approach.

Multinomial distribution

Most parameters in NLP are distributions $p(z)$ over discrete objects $z \in \{1, \ldots, K\}$.

Possible $p(z)$s are the points $\boldsymbol{\phi}$s on the simplex. For $K = 3$:



$p(z) \quad \Rightarrow \quad p(z \mid \boldsymbol{\phi}) \quad \Rightarrow \quad \phi_z$

A random draw z from a multinomial distribution is written:

$$z \sim \text{Multinomial}(\boldsymbol{\phi}).$$

If we draw some number of independent samples from Multinomial($\boldsymbol{\phi}$), the probability of observing c_z counts of observation z is proportional to:

$$\phi_1{}^{c_1} \cdots \phi_K{}^{c_K}$$

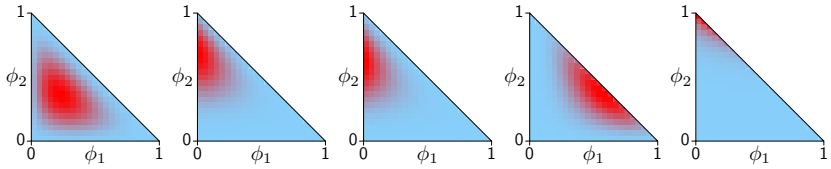
A multinomial distribution is a distribution over K possible outcomes $\{1, \ldots, K\}$. A parameter setting for a multinomial distribution is a point on the simplex: $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_K), \phi_z \geq 0, \sum_{z=1}^{K} \phi_z = 1$. We have turned multinomial distributions into first-class objects, so we can refer to $\boldsymbol{\phi}$ instead of $p(z)$. The form of the probability of drawing $(c_1, \ldots, c_K)$ will play an important role in Dirichlet-multinomial conjugacy.

Distributions over multinomial parameters

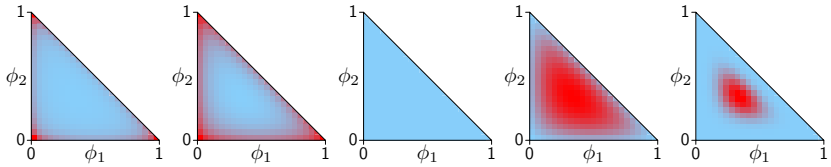
A Dirichlet distribution is a distribution over multinomial parameters $\boldsymbol{\phi}$ in the simplex.

Like a Gaussian, there's a notion of mean and variance.

Different means:



Different variances:



All distributions we encounter in practice have a mean and variance. For a Gaussian $\mathcal{N}(\mu, \sigma^2)$, they are explicitly represented in the parameters. For the Dirichlet distribution, the mapping between parameters and moments is not as simple.

Dirichlet distribution

A Dirichlet is specified by concentration parameters:

$$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K),\ \alpha_z \geq 0$$

Mean: $\left(\frac{\alpha_1}{\sum_z \alpha_z}, \ldots, \frac{\alpha_n}{\sum_z \alpha_z}\right)$

Variance: larger $\boldsymbol{\alpha}$s $\rightarrow$ smaller variance

A Dirichlet draw $\boldsymbol{\phi}$ is written $\boldsymbol{\phi} \sim \text{Dirichlet}(\boldsymbol{\alpha})$,

which means $p(\boldsymbol{\phi} \mid \boldsymbol{\alpha}) \propto {\phi_1}^{\alpha_1 - 1} \cdots {\phi_K}^{\alpha_K - 1}$

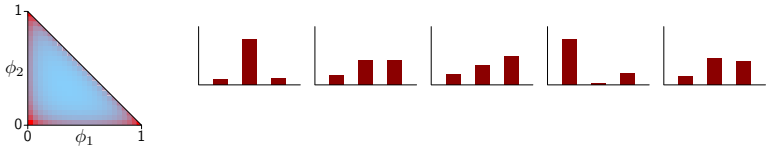
The Dirichlet distribution assigns probability mass to multinomials that are likely to yield pseudocounts $(\alpha_1 - 1, \ldots, \alpha_K - 1)$.

Mode: $\left(\frac{\alpha_1 - 1}{\sum_z (\alpha_z - 1)}, \ldots, \frac{\alpha_n - 1}{\sum_z (\alpha_z - 1)}\right)$

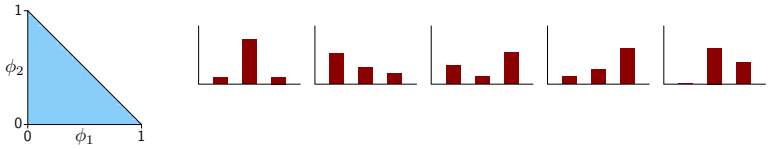
The full expression for the density of a Dirichlet is $p(\boldsymbol{\phi} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\sum_{z=1}^K \alpha_z)}{\prod_{z=1}^K \Gamma(\alpha_z)} \prod_{z=1}^K {\phi_z}^{\alpha_z}$. Note that unlike the Gaussian, the mean and mode of the Dirichlet are distinct. This leads to a small discrepancy between concentration parameters and pseudocounts: concentration parameters $\boldsymbol{\alpha}$ correspond to pseudocounts $\boldsymbol{\alpha} - 1$.

Draws from Dirichlet distributions

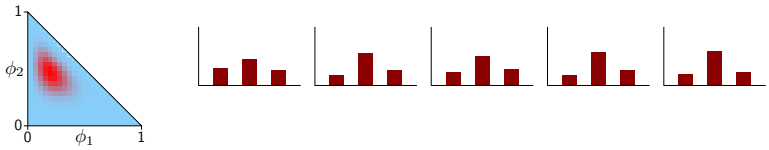
Dirichlet(.5,.5,.5)



Dirichlet(1,1,1)



Dirichlet(5,10,8)



A Dirichlet$(1, 1, 1)$ is a uniform distribution over multinomial parameters. As the concentration parameters increase, the uncertainty over parameters decreases. Going in the other direction, concentration parameters near zero encourage sparsity, placing probability mass in the corners of the simplex. This sparsity property is the key to the Dirichlet process.

The Bayesian paradigm

data $\mathbf{x}$ —Bayesian inference→ $q(\boldsymbol{\theta})$

What is $q(\boldsymbol{\theta})$?

$$q(\boldsymbol{\theta}) \stackrel{\text{def}}{=} \underbrace{p(\boldsymbol{\theta} \mid \mathbf{x})}_{\text{posterior}} = \frac{1}{p(\mathbf{x})} \underbrace{p(\boldsymbol{\theta})}_{\text{prior}} \underbrace{p(\mathbf{x} \mid \boldsymbol{\theta})}_{\text{likelihood}}$$

The Bayesian paradigm gives us a way to derive the optimal distribution over the parameters $q(\boldsymbol{\theta})$ given data. We start with a prior $p(\boldsymbol{\theta})$, multiply the likelihood of the data we observe $p(\mathbf{x} \mid \boldsymbol{\theta})$, and renormalize to get the **posterior** $p(\boldsymbol{\theta} \mid \mathbf{x})$. The computation of this posterior is generally the main focus of Bayesian inference.

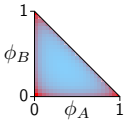
Posterior updating for Dirichlet distributions

Model:

$$\boldsymbol{\phi} \sim \text{Dirichlet}_3(\underbrace{0.5, 0.5, 0.5}_{\boldsymbol{\alpha}})$$

$$\mathbf{x} \sim \text{Multinomial}_3^7(\boldsymbol{\phi})$$

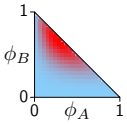
Prior: $p(\boldsymbol{\phi}) \propto \phi_A^{0.5-1} \phi_B^{0.5-1} \phi_C^{0.5-1}$



Likelihood: $p(\mathbf{x} \mid \boldsymbol{\phi}) = \phi_A^{2} \phi_B^{4} \phi_C^{1}$

(A) (B) (B) (C) (A) (B) (B)

Posterior: $p(\boldsymbol{\phi} \mid \mathbf{x}) \propto \phi_A^{2.5-1} \phi_B^{4.5-1} \phi_C^{1.5-1}$



Result: $p(\boldsymbol{\phi} \mid \mathbf{x}) = \text{Dirichlet}(2.5, 4.5, 1.5)$

The simplest example of computing the posterior is when there are no hidden variables. Assume we have 3 outcomes (A, B, C). Notation: the subscript 3 on the Dirichlet means that the dimensionality is 3 and the superscript on the multinomial means that $\mathbf{x}$ is a sequence of 7 observations. Since the Dirichlet and multinomial distributions are conjugate, the posterior can be computed in closed form and has the same form as the prior.

A two-component mixture model

$\phi_1 \sim \text{Dirichlet}(1, 1)$

$\phi_2 \sim \text{Dirichlet}(1, 1)$

$z_i \sim \text{Multinomial}(\frac{1}{2}, \frac{1}{2})$ (2)

$x_i \sim \text{Multinomial}^5(\phi_{z_i})$ (A) (A) (B) (B) (A)

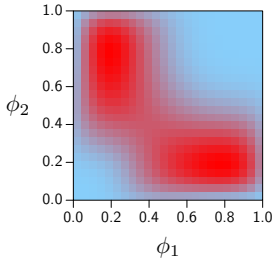
Observed data:

$x_1 =$ (A) (B) (B) (B) (B)
$x_2 =$ (A) (B) (B) (B) (B)
$x_3 =$ (B) (A) (A) (A) (A)

Unknown parameters:

$\boldsymbol{\theta} = (\phi_1, \phi_2)$

$p(\boldsymbol{\theta} \mid \mathbf{x})$



- True posterior $p(\boldsymbol{\theta} \mid \mathbf{x})$ has symmetries
- ϕ_1 explains x_1, x_2 and ϕ_2 explains x_3 in upper mode (or vice-versa in lower mode)
- The component explaining x_3 has higher uncertainty

For the Dirichlet example, we could compute the posterior analytically. This is not always the case, in particular, when there are hidden variables. In the two-component mixture model example, the posterior is shown with the hidden variables $\mathbf{z}$ marginalized out. Convention: letters denote outcomes of the data and numbers denote components.

Using the posterior for prediction

Setup:

Assume a joint probabilistic model $p(\boldsymbol{\theta})p(x, y \mid \boldsymbol{\theta})$
over input x, output y, parameters $\boldsymbol{\theta}$.
Training examples: $(x_1, y_1), \ldots, (x_n, y_n)$
Test input: x_{new}

Traditional: ${y_{\text{new}}}^* = \text{argmax}_{y_{\text{new}}}\, p(y_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta})$

Bayes-optimal prediction:

$$ {y_{\text{new}}}^* = \text{argmax}_{y_{\text{new}}}\, p(y_{\text{new}} \mid x_{\text{new}}, \{(x_i, y_i)\}) $$

Explicitly write out the integrated parameters:

$$ p(y_{\text{new}} \mid x_{\text{new}}, \{(x_i, y_i)\}) = \int p(y_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta}) \underbrace{p(\boldsymbol{\theta} \mid \{(x_i, y_i)\})}_{\text{posterior}} d\boldsymbol{\theta} $$

We can plug in an approximate posterior:

$$ p(y_{\text{new}} \mid x_{\text{new}}, \{(x_i, y_i)\}) = \int p(y_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta}) q_{\text{approx}}(\boldsymbol{\theta}) d\boldsymbol{\theta} $$

We have mentioned the desire to compute the posterior over parameters $p(\boldsymbol{\theta} \mid \mathbf{x})$. Now we motivate this computation with a prediction task. Note that this decision-theoretic framework supports arbitrary loss functions, but we have specialized to the 0-1 loss (which corresponds to maximizing probability) to ease notation.

Roadmap

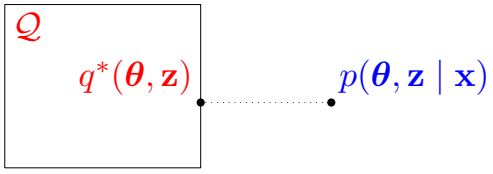
- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Variational Bayesian inference

Goal of Bayesian inference: compute posterior $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})$

Variational inference is a framework for approximating the: true posterior with the best from a set of distributions $\mathcal{Q}$:

$$q^* = \text{argmin}_{q \in \mathcal{Q}} \, \text{KL}(q(\boldsymbol{\theta}, \mathbf{z}) || p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x}))$$

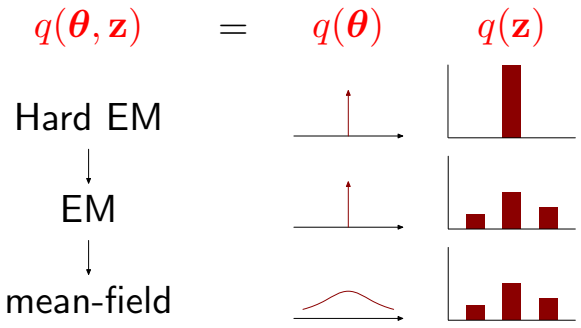


The variational inference framework gives a principled way of finding an approximate distribution which is as close (as measured by KL) to the posterior. This will allow us to tackle posterior computations for models such as mixtures.

Types of variational approximations

$$q^* = \underset{q \in \mathcal{Q}}{\text{argmin}} \, \text{KL}(q(\boldsymbol{\theta}, \mathbf{z}) || p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x}))$$

What types of $\mathcal{Q}$ can we consider?



The quality of the approximation depends on $\mathcal{Q}$: the bigger the $\mathcal{Q}$, the better. Two of the familiar classical algorithms (hard EM and EM) are based on a particular kind of $\mathcal{Q}$. We will show that mean-field is a natural extension, by expanding $\mathcal{Q}$ but staying within the same framework. Of course, there are even larger $\mathcal{Q}$s, but we risk losing tractability in those cases.

Heuristic derivation of mean-field

Algorithm template: iterate between the E-step and M-step.

Let us first heuristically derive the mean-field algorithm.

E-step	**M-step**
Find best $\mathbf{z}$	Find best $\boldsymbol{\theta}$
$\mathbf{z}^* = \text{argmax}_z\, p(\mathbf{z} \mid \boldsymbol{\theta}^*, \mathbf{x})$	$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}}\, p(\boldsymbol{\theta} \mid \mathbf{z}^*, \mathbf{x})$
Find best distrib. over $\mathbf{z}$ $\longrightarrow$	Take distrib. into account
$q(\mathbf{z})^* \propto p(\mathbf{z} \mid \boldsymbol{\theta}, \mathbf{x})$	~~$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \sum_{\mathbf{z}} q(\mathbf{z}) p(\boldsymbol{\theta} \mid \mathbf{z}, \mathbf{x})$~~
	$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \prod_{\mathbf{z}} p(\boldsymbol{\theta} \mid \mathbf{z}, \mathbf{x})^{q(\mathbf{z})}$
Take distrib. into account $\longleftarrow$	Find best distrib. over $\boldsymbol{\theta}$
$q(\mathbf{z})^* \propto \prod_{\boldsymbol{\theta}} p(\mathbf{z} \mid \boldsymbol{\theta}, \mathbf{x})^{q(\boldsymbol{\theta})}$	$q(\boldsymbol{\theta})^* \propto \prod_{\mathbf{z}} p(\boldsymbol{\theta} \mid \mathbf{z}, \mathbf{x})^{q(\mathbf{z})}$

Infinite product over $\boldsymbol{\theta}$ has closed form if $q(\boldsymbol{\theta})$ is a Dirichlet.

Before we formally derive algorithms based on the variational framework, let us heuristically consider what the update for mean-field might look like. Note that when we taking the approximating distribution into account, we use a "geometric" expectation rather than a simple average. This follows from using KL-divergence and makes computation tractable. The new challenge introduced by the mean-field E-step is the infinite product, but this can be handled by exploiting certain properties of q.

Formal derivation of mean-field

$$q^* = \underset{q \in \mathcal{Q}}{\text{argmin}}\, \text{KL}(q(\boldsymbol{\theta}, \mathbf{z}) || p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x}))$$

Steps:

1. Formulate as an optimization problem (variational principle)
2. Relax the optimization problem (e.g., mean-field)
3. Solve using coordinate-wise descent

Now that we know what the final algorithm will look like, let us derive the algorithms directly as consequences of the KL objective.

Kullback-Leibler divergence

KL measures how different two distributions p and q are.

Definition:

$$\mathrm{KL}(q||p) \stackrel{\text{def}}{=} \mathbb{E}_q \log \frac{q(\theta)}{p(\theta)}$$

An important property:

$$\mathrm{KL}(q||p) \geq 0 \qquad \mathrm{KL}(q||p) = 0 \text{ if and only if } q = p$$

KL is asymmetric:

Assuming $\mathrm{KL}(q||p) < \infty$,

$$p(\theta) = 0 \Rightarrow q(\theta) = 0 \quad [q\text{“} \subset \text{”}p]$$

The optimization problem is in terms of KL-divergence. Before solving the optimization problem, we review some of the properties of KL.

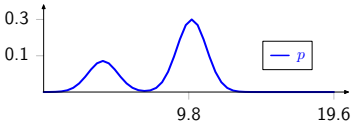
Minimizing with respect to KL

Since $\mathrm{KL} = 0$ exactly when two arguments are equal, we have:

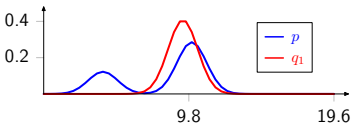
$$p = \text{argmin}_{q \in \mathcal{Q}_{\text{all}}} \mathrm{KL}(q||p) \quad \text{and} \quad p = \text{argmin}_{q \in \mathcal{Q}_{\text{all}}} \mathrm{KL}(p||q)$$

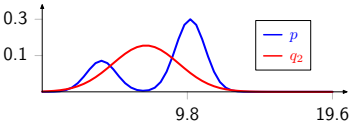
where $\mathcal{Q}_{\text{all}}$ is all possible distributions.

Asymmetry revealed when we replace $\mathcal{Q}_{\text{all}}$ with $\mathcal{Q} \subsetneq \mathcal{Q}_{\text{all}}$.

Let $p =$ 

and $\mathcal{Q} =$ all possible Gaussians

$q_1 = \underset{q \in \mathcal{Q}}{\text{argmin}}\, \mathrm{KL}(q||p)$ (mode-finding): 

$q_2 = \underset{q \in \mathcal{Q}}{\text{argmin}}\, \mathrm{KL}(p||q)$ (mean-finding): 

In fitting a model, the approximating distribution is often the second argument (with the first being the empirical distribution). These distributions are over observed data, where one wants to assign mass to more than just the observed data. For variational inference, the distributions are over parameters, and the approximating distribution appears in the first argument. This yields a tractable optimization problem. Also, it allows us to consider degenerate q with non-degenerate p (as in EM). The symmetric modes of a mixture model are redundant, so it makes sense to capture only one of them anyway.

Step 1: formulate as optimization problem

Variational principle: write the quantity we wish to compute as the solution of an optimization problem:

$$q^* \stackrel{\text{def}}{=} \operatorname*{argmin}_{q \in \mathcal{Q}_{\text{all}}} \mathrm{KL}(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta} \mid \mathbf{x})),$$

where $\mathcal{Q}_{\text{all}}$ is set of all distributions over parameters.

Solution: $q^*(\boldsymbol{\theta}) = p(\boldsymbol{\theta} \mid \mathbf{x})$, which achieves $\mathrm{KL} = 0$.

This is not very useful yet because q^* is just as hard to compute...

Normalization of p not needed:

$$\operatorname{argmin}_q \mathrm{KL}(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta} \mid \mathbf{x})) = \operatorname{argmin}_q \mathrm{KL}(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta}, \mathbf{x}))$$

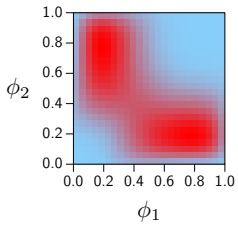
The variational principle is a general technique which originated out of physics, and can be applied to many other problems besides inference in graphical models. See [Wainwright, Jordan, 2003] for a thorough treatment of variational methods for graphical models. Note that the full posterior is $p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})$, but we integrate out $\mathbf{z}$ here so that we can visualize the posterior over parameters $p(\boldsymbol{\theta} \mid \mathbf{x}) = \sum_{\mathbf{z}} p(\boldsymbol{\theta}, \mathbf{z} \mid \mathbf{x})$ on the next slide.

Step 2: relax the optimization problem

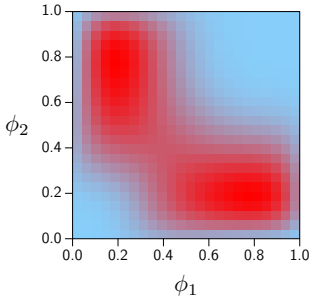
$$q^* \stackrel{\text{def}}{=} \operatorname*{argmin}_{q \in \mathcal{Q}_{\text{all}}} \mathrm{KL}(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta} \mid \mathbf{x}))$$

$\mathcal{Q}_{\text{all}}$ = all distributions

$p(\boldsymbol{\theta} \mid \mathbf{x})$



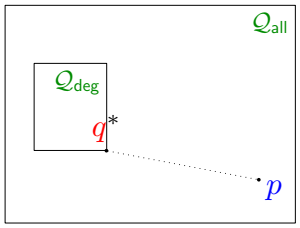
Optimal $q^*(\boldsymbol{\theta})$



The advantage of using optimization is that it leads naturally to approximations: either by changing the domain (which we do) or the objective function (which would lead to algorithms like belief propagation). Here, we show the optimal approximating distributions for various $\mathcal{Q}$s on the simple two-component mixture example.

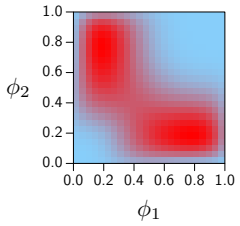
Step 2: relax the optimization problem

$$q^* \stackrel{\text{def}}{=} \underset{q \in \mathcal{Q}_{\text{deg}}}{\text{argmin}}\, \text{KL}(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta} \mid \mathbf{x}))$$

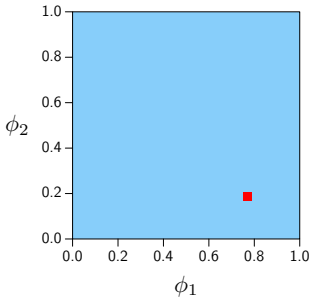


$$\mathcal{Q}_{\text{deg}} = \left\{ q : q(\boldsymbol{\theta}) = \delta_{\boldsymbol{\theta}^*}(\boldsymbol{\theta}) \right\}$$

$p(\boldsymbol{\theta} \mid \mathbf{x})$



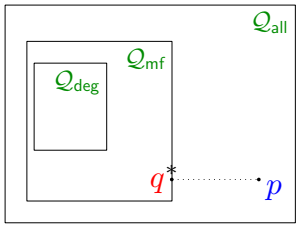
Degenerate $q^*(\boldsymbol{\theta})$



The advantage of using optimization is that it leads naturally to approximations: either by changing the domain (which we do) or the objective function (which would lead to algorithms like belief propagation). Here, we show the optimal approximating distributions for various $\mathcal{Q}$s on the simple two-component mixture example.

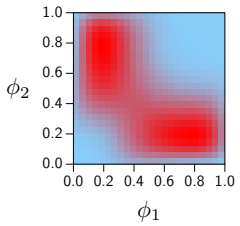
Step 2: relax the optimization problem

$$q^* \stackrel{\text{def}}{=} \underset{q \in \mathcal{Q}_{\text{mf}}}{\text{argmin}}\, \text{KL}(q(\boldsymbol{\theta}) \,||\, p(\boldsymbol{\theta} \mid \mathbf{x}))$$

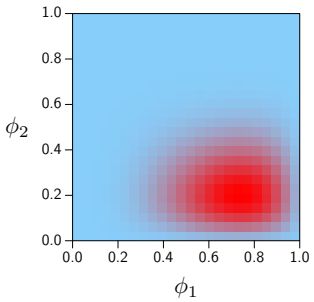


$$\mathcal{Q}_{\text{mf}} = \left\{ q : q(\boldsymbol{\theta}) = \prod_{i=1}^{n} q_i(\theta_i) \right\}$$

$p(\boldsymbol{\theta} \mid \mathbf{x})$



Mean-field $q^*(\boldsymbol{\theta})$



The advantage of using optimization is that it leads naturally to approximations: either by changing the domain (which we do) or the objective function (which would lead to algorithms like belief propagation). Here, we show the optimal approximating distributions for various $\mathcal{Q}$s on the simple two-component mixture example.

Step 3: coordinate-wise descent

Goal: minimize $\mathrm{KL}(q(\boldsymbol{\theta}) || p(\boldsymbol{\theta} \mid \mathbf{x}))$ subject to $q \in \mathcal{Q}$

Assume: $q(\boldsymbol{\theta}) = \prod_i q_i(\theta_i)$

Algorithm: for each i, optimize $q_i(\theta_i)$ holding all other coordinates $q_{-i}(\theta_{-i})$ fixed.

If q_is degenerate ($q_i(\theta_i) = \delta_{\theta_i^*}(\theta_i)$):

$$\theta_i^* = \text{argmax}_{\theta_i} \, p(\theta_i \mid \theta_{-i}^*, \mathbf{x})$$

If q_is non-degenerate: 

$$q_i(\theta_i) \propto \exp\{\mathbb{E}_{q_{-i}} \log p(\theta_i \mid \theta_{-i}, \mathbf{x})\}$$

Mean-field only specifies the optimization problem based on the variational framework. We could use any number of generic optimization algorithms to solve the mean-field objective. However, a particularly simple and intuitive method is to use coordinate-wise descent. To simplify notation, let $\boldsymbol{\theta}$ both the parameters and the hidden variables.

Mean-field recipe

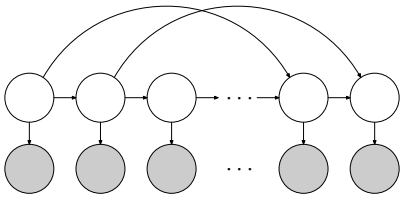
$$\begin{aligned} q_i(\theta_i) &\propto \exp\{\mathbb{E}_{q_{-i}} \log p(\theta_i \mid \theta_{-i}, \mathbf{x})\} \\ &\propto \exp\{\mathbb{E}_{q_{-i}} \log p(\boldsymbol{\theta}, \mathbf{x})\} \end{aligned}$$

Recipe for updating q_i:

1. Write down all the unobserved variables in the model (including parameters and latent variables)
2. For each variable θ_i:
   a. Write down only the factors of the joint distribution that contain θ_i
   b. Set $q_i \propto \exp\{\mathbb{E}_{q_{-i}} \log(\text{those factors})\}$

We give a generic recipe for deriving a mean-field algorithm for any probabilistic model. The remaining work is to actually compute the $\exp \mathbb{E} \log$, which is not obviously doable at all. It turns out that it works out when we have conjugate and our distributions are in the exponential family. Later we show this is true for the Dirichlet-multinomial pair.

Non-Bayesian variational inference



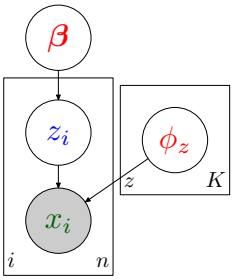
- The E-step (computing $p(\mathbf{z} \mid \mathbf{x})$) is sometimes intractable when there are long-range dependencies
- Variational EM: approximate E-step using $\text{argmin}_q \, \text{KL}(q(\mathbf{z}) || p(\mathbf{z} \mid \mathbf{x}))$
- Variational EM fits into the general variational framework: one true posterior, various approximating families

Often the term variational is associated with another setting, where we wish to do inference in a loopy graph, either as part of an E-step for a directed model with hidden variables or a gradient computation for an undirected model. These cases can be interpreted as instances of the variational framework.

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Bayesian finite mixture model



Parameters: $\boldsymbol{\theta} = (\boldsymbol{\beta}, \boldsymbol{\phi}) = (\beta_1, \ldots, \beta_K, \phi_1, \ldots, \phi_K)$
Hidden variables: $\mathbf{z} = (z_1, \ldots, z_n)$
Observed data: $\mathbf{x} = (x_1, \ldots, x_n)$

$$p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{x}) = p(\boldsymbol{\beta}) \prod_{z=1}^{K} p(\phi_z) \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\beta}) p(x_i \mid \phi_{z_i})$$

$\boldsymbol{\beta} \sim \text{Dirichlet}_K(\alpha', \ldots, \alpha')$
For each component $z \in \{1, \ldots, K\}$:
  $\phi_z \sim G_0$
For each data point $i \in \{1, \ldots, n\}$:
  $z_i \sim \text{Multinomial}(\boldsymbol{\beta})$
  $x_i \sim F(\phi_{z_i})$

Prior over component parameters:
  G_0 (e.g., $\text{Dirichlet}_V(\alpha', \ldots, \alpha')$)
Data model:
  F (e.g., Multinomial)

There are several ways to represent a probabilistic model. The graphical model allows one to visualize the dependencies between all the random variables in the model, which corresponds to a particular factoring of the joint probability distribution, where each factor is for example, $p(z_i \mid \boldsymbol{\beta})$. The procedural notation actually specifies the form of the distributions via a sequence of draws (e.g., $z_i \sim \text{Multinomial}(\boldsymbol{\beta})$). Note that we are representing the data model abstractly as (G_0, F).

Bayesian finite mixture model

Running example: document clustering

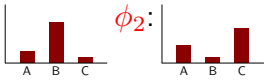
$\boldsymbol{\beta} \sim \text{Dirichlet}_K(\alpha', \ldots, \alpha')$ [draw component probabilities]

$K = 2$ clusters

$\boldsymbol{\beta}$: 

For each component (cluster) $z \in \{1, \ldots, K\}$:
  $\phi_z \sim \text{Dirichlet}_V(\alpha', \ldots, \alpha')$ [draw component parameter]

$V = 3$ word types

ϕ_1: ϕ_2: 

For each data point (document) $i \in \{1, \ldots, n\}$:
  $z_i \sim \text{Multinomial}_K(\boldsymbol{\beta})$ [choose component]
  $x_i \sim \text{Multinomial}_V^m(\phi_{z_i})$ [generate data]

$n = 5$ documents
$m = 4$ words/doc

z_1: (2) z_2: (2) z_3: (1) z_4: (2) z_5: (1)
x_1: CCAC x_2: CCAC x_3: AABB x_4: CACC x_5: ACBB

We will use the Bayesian finite mixture model as an example of applying the mean-field algorithm. Later, it will be extended to the DP mixture model.

Hard EM for the finite mixture model

Model: $p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{x}) = p(\boldsymbol{\beta}) \prod_{z=1}^{K} p(\phi_z) \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\beta}) p(x_i \mid \phi_{z_i})$

Approximation: $q(\boldsymbol{\theta}, \mathbf{z}) = \underbrace{q(\boldsymbol{\beta})}_{\delta_{\beta^*}(\boldsymbol{\beta})} \prod_{z=1}^{K} \underbrace{q(\phi_z)}_{\delta_{\phi_z^*}(\phi_z)} \prod_{i=1}^{n} \underbrace{q(z_i)}_{\delta_{z_i^*}(z_i)}$

E-step:

For each data point $i \in \{1, \ldots, n\}$:

$z_i^* = \text{argmax}_{z_i} p(z_i \mid \boldsymbol{\beta}^*, \mathbf{x}) = \text{argmax}_{z_i} p(z_i \mid \boldsymbol{\beta}^*) p(x_i \mid \phi_{z_i})$

M-step:

$\boldsymbol{\beta}^* = \text{argmax}_{\beta} p(\boldsymbol{\beta} \mid \mathbf{z}^*, \mathbf{x}) = \text{argmax}_{\beta} p(\boldsymbol{\beta}) \prod_{i=1}^{n} p(z_i^* \mid \boldsymbol{\beta})$

For each component $z \in \{1, \ldots, K\}$:

$\phi_z^* = \text{argmax}_{\phi} p(\phi) \prod_{i=1}^{n} p(x_i \mid \phi)^{1[z_i^* = z]}$

We can derive hard EM for the Bayesian finite model using the recipe given at the end of Part I. Since all qs are degenerate, optimizing the distribution is the same as optimizing a point. For example, $z_i^* = \text{argmax}_z \exp \mathbb{E}_{q_{-z_i}} \log p(z \mid \boldsymbol{\beta}) p(x_i \mid \phi_z)$. Since q is degenerate, the exp and log cancel, leaving us with the classical hard E-step.

Mean-field for the finite mixture model

Model: $p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{x}) = p(\boldsymbol{\beta}) \prod_{z=1}^{K} p(\phi_z) \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\beta}) p(x_i \mid \phi_{z_i})$

M-step: optimize $q(\phi_z)$ and $q(\boldsymbol{\beta})$

$$q(\boldsymbol{\beta}) \propto \prod_{\mathbf{z}} p(\boldsymbol{\beta} \mid \mathbf{z}, \mathbf{x})^{q(\mathbf{z})}$$
$$\propto p(\boldsymbol{\beta}) \prod_{i=1}^{n} \prod_{z_i=1}^{K} p(z_i \mid \boldsymbol{\beta})^{q(z_i)} = p(\boldsymbol{\beta}) \prod_{z_i=1}^{K} \beta_{z_i}^{\sum_{i=1}^{n} q(z_i)}$$

Define expected counts of component z: $C_z = \sum_{i=1}^{n} q_{z_i}(z)$

Recall the prior: $p(\boldsymbol{\beta}) = \text{Dirichlet}(\boldsymbol{\beta}; \alpha) \propto \prod_{z=1}^{k} \beta_z^{\alpha - 1}$

Prior: $\prod_{z=1}^{K} \beta_z^{\alpha - 1}$

"Likelihood": $\prod_{z=1}^{K} \beta_z^{C_z}$

"Posterior": $\prod_{z=1}^{K} \beta_z^{\alpha - 1 + C_z}$

Conclusion: $q(\boldsymbol{\beta}) = \text{Dirichlet}(\boldsymbol{\beta}; \alpha + C)$

Now we show the mean-field algorithm. The M-step for the mean-field algorithm involves just updating the Dirichlet distribution. Rather than normalizing, the M-step keeps the full counts obtained during the E-step. This extra degree of freedom gives mean-field the ability to deal with uncertainty. For simplicity, we only consider the update for the component probabilities $\boldsymbol{\beta}$, not the component parameters $\boldsymbol{\phi}$.

Mean-field for the finite mixture model

$$\text{Model: } p(\boldsymbol{\theta}, \mathbf{z}, \mathbf{x}) = p(\boldsymbol{\beta}) \prod_{z=1}^{K} p(\phi_z) \prod_{i=1}^{n} p(z_i \mid \boldsymbol{\beta}) p(x_i \mid \phi_{z_i})$$

E-step: optimize $q(z_i)$

$$\begin{aligned} q(z_i) &\propto \prod_{\boldsymbol{\theta}} p(z_i \mid \boldsymbol{\theta}, \mathbf{x})^{q(\boldsymbol{\theta})} \\ &\propto \underbrace{\prod_{\beta} p(z_i \mid \boldsymbol{\beta})^{q(\beta)}}_{W(z_i)} \underbrace{\prod_{\phi_{z_i}} p(x_i \mid \phi_{z_i})^{q(\phi_{z_i})}}_{W(x_i \mid z_i)} \end{aligned}$$

$W(z_i)W(x_i \mid z_i)$ is like $p(z_i)p(x_i \mid z_i)$, but multinomial weights W takes into account uncertainty in $\boldsymbol{\theta}$.

The E-step requires a bit more work, as it involves the infinite product over θ, or equivalently computing $\exp \mathbb{E} \log$. Because the approximation is fully-factorized the optimal update breaks down into independent integrals, each over a separate parameter. We call these quantities multinomial weights.

Mean-field: computing multinomial weights

$W(z) = \prod_{\beta} p(z \mid \boldsymbol{\beta})^{q(\beta)} = \exp\{\mathbb{E}_{q(\beta)} \log p(z \mid \boldsymbol{\beta})\}$

What's $q(\boldsymbol{\beta})$?

E-step: expected counts: $C_z = \sum_{i=1}^{n} q_{z_i}(z)$

M-step: $q(\beta) = \text{Dirichlet}(\beta; \underbrace{\alpha + C}_{\alpha'})$

Mean-field multinomial weight:

$$W(z) = \frac{\exp(\Psi(\alpha'_z))}{\exp(\Psi(\sum_{z'=1}^{K} \alpha'_z))}$$

Compare with EM ($q(\boldsymbol{\beta}) = \delta_{\beta^*}(\boldsymbol{\beta})$ is degenerate):

$$W(z) = \boldsymbol{\beta}^*_z = \frac{\alpha'_z - 1}{\sum_{z'=1}^{K} (\alpha'_{z'} - 1)}$$

$\Psi(\cdot)$ is the digamma function and is easy to compute.

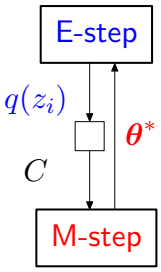
The multinomial weights are like sub-probabilities (they always sum to less than 1 by Jensen's inequality). The more uncertainty there is, the smaller the weights. The computation of the multinomial weights bridges the M-step and the E-step: they are computed based on the posterior distributions computed in the M-step and used in the E-step. The digamma function $\Psi(\cdot) = \frac{\partial \Gamma(x)}{\partial x}$ is an easy function whose code can be copied out of Numerical Recipes.

Mean-field overview

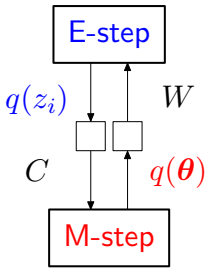
E-step: $q(z_i) \propto W(z_i) W(x_i \mid z_i)$, $W(z) = \frac{\exp \Psi(\alpha_z + C_z)}{\exp \Psi(\sum_{z'=1}^{K} \alpha_{z'} + C_{z'})}$

M-step: $q(\boldsymbol{\beta}) = \text{Dirichlet}(\boldsymbol{\beta}; \alpha + C)$, $C_z = \sum_{i=1}^{n} q_{z_i}(z)$

EM



Mean-field



This slide shows the data flow for the EM and mean-field algorithms. While the output of each step is technically a distribution over either parameters or hidden variables, the other step only depends on an aggregated value (C for the M-step, W for the E-step).

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Interpreting mean-field through $\exp(\Psi(\cdot))$

EM with $\alpha = 1$ (maximum likelihood):

$$W(z) = \frac{C_z}{\sum_{z'=1}^{K} C_{z'}}$$

Mean-field with $\alpha = 1$:

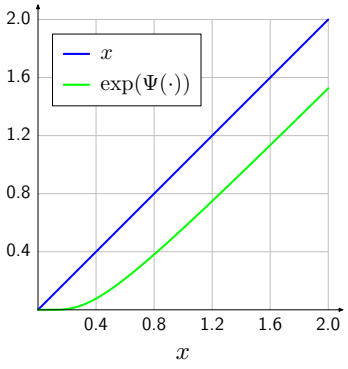
$$W(z) = \frac{\exp(\Psi(1+C_z))}{\exp(\Psi(1+\sum_{z'=1}^{K} C_{z'}))}$$

($\approxeq$ adding 0.5)

Mean-field with $\alpha = 0$:

$$W(z) = \frac{\exp(\Psi(C_z))}{\exp(\Psi(\sum_{z'=1}^{K} C_{z'}))}$$

($\approxeq$ subtracting 0.5)



$\exp(\Psi(x)) \approxeq x - 0.5$
(for $x > 1$)

We now embark on the development of the DP. Earlier, we said that the DP prior penalizes extra clusters. It turns out that this penalty is based on uncertainty in parameter estimates. The more clusters there are, the more fragmented the counts are, leading to greater uncertainty. We start by giving concrete intuition about how just the mean-field algorithm on finite mixtures can achieve this. The $\exp(\Psi(\cdot))$ plot captures the essence of the DP from the point of view of mean-field for multinomial data.

The rich get richer

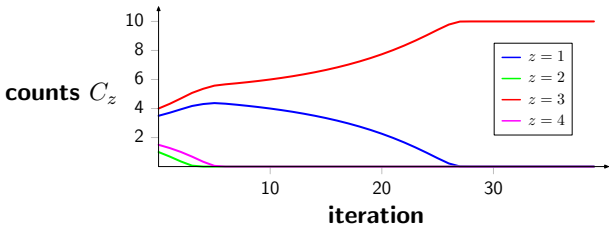
What happens when $\alpha = 0$?

E-step: $q(z_i) \propto W(z_i) W(x_i \mid z_i) \qquad C_z = \sum_{i=1}^n q(z_i)$

M-step: $W(z) = \frac{\exp(\Psi(C_z))}{\exp(\Psi(\sum_{z'=1}^K C_{z'}))} \approx \frac{C_z - 0.5}{(\sum_{z'=1}^K C_{z'}) - 0.5}$

Effect of mean-field with $\alpha = 0$ on component probabilities β?

Thought experiment: ignore $W(x_i \mid z_i)$.



When subtract 0.5, small counts are hurt more than large ones (like a regressive tax)

The algorithm achieves sparsity by choosing one component.

In general, data term fights the sparsity prior.

By ignoring the data-dependent factor $W(x_i \mid z_i)$, we can focus on the tendencies of the mean-field algorithm. As the diagram shows, $W(z_i)$ encourages very few clusters. On the other hand, $W(x_i \mid z_i)$ takes into account the data, for which we might need many clusters to explain the data. A balance is obtained by combining both factors.

The rich get even richer

E-step: $q(z_i) \propto W(z_i) W(x_i \mid z_i) \qquad C_{zj} = \sum_{i=1}^n q_{z_i}(z) x_{ij}$

M-step: $W(j \mid z) = \frac{\exp(\Psi(C_{zj}))}{\exp(\Psi(\sum_{j'=1}^V C_{zj'}))} \approx \frac{C_{zj} - 0.5}{(\sum_{j'=1}^V C_{zj'}) - 0.5}$

Thought experiment: ignore $W(z_i)$, focus on $W(x_i \mid z_i)$.

Suppose $C_{1A} = 20, C_{1B} = 20$, $C_{2A} = 0.5, C_{2B} = 0.2$

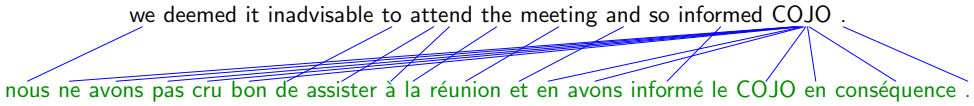
	$W(A \mid 1)$	$W(A \mid 2)$
EM:	$\frac{20}{20+20} = 0.5$	$\frac{0.5}{0.5+0.2} = \mathbf{0.71}$
Mean-field:	$\frac{e^{\Psi(20+1)}}{e^{\Psi(20+20+1)}} = \mathbf{0.494}$	$\frac{e^{\Psi(0.5+1)}}{e^{\Psi(0.5+0.2+1)}} = 0.468$

For observation A: EM prefers component 2
mean-field prefers 1

Key property: multinomial weights are not normalized, allowing global tradeoffs between components.

Now we focus on the data-dependent factor $W(x_i \mid z_i)$. The example shows that even in the absence of $W(z_i)$, there is a tendency to prefer few larger clusters over many small ones. To guard against this type of overfitting, add-ϵ smoothing is often used. However, note that add-ϵ smoothing in this case will only mitigate the difference but will never strictly prefer component 1 over 2.

Example: IBM model 1 for word alignment



$$p(\mathbf{x}, \mathbf{z}; \boldsymbol{\phi}) = \prod_{i=1}^{n} p(z_i) p(x_i \mid e_{z_i}; \boldsymbol{\phi})$$

E-step: $q_{z_i}(j) \propto p(x_i \mid e_j) \quad C_{e,x} = \sum_{i,j: e_j = e, x_i = x} q_{z_i}(j)$

M-step: $W(x \mid e) = \frac{C_{e,x}}{\sum_{x'} C_{e,x'}} \frac{\exp(\Psi(C_{e,x}))}{\exp(\Psi(\sum_{x'} C_{e,x'}))}$

Garbage collectors problem: rare source words have large probabilities for target words in the translation.

Quick experiment: EM: 20.3 AER, mean-field: 19.0 AER

This example shows a concrete case where being sensitive to uncertainty in parameters better using mean-field can improve performance with a very localized change to the original EM code. [Moore, 2004] also took note of this problem with rare words and used add-ϵ smoothing.

An approximate Dirichlet process (DP) mixture model

Take finite mixture model, define $p(\boldsymbol{\beta}) = \text{Dirichlet}(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K})$

Theorem:

As $K \to \infty$, finite mixture model $\to$ DP mixture model.

As $\frac{\alpha_0}{K} \to 0$, mean-field enters rich-gets-richer regime.

How to Bayesianify/DPify your EM algorithm

In the M-step of EM code where counts are normalized, replace $\frac{C_z}{\sum_{z'} C_{z'}}$ with $\frac{\exp \Psi(C_z)}{\exp \Psi(\sum_{z'} C_{z'})}$.

At this point, we've basically done. We have a model that acts empirically like a DP mixture model (we'll define this next) and a simple concrete algorithm for doing approximate inference.

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

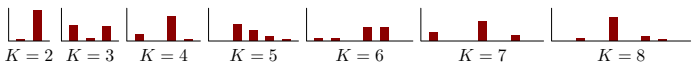
The limit of finite mixtures?

Theoretical goal:

Define a nonparametric (infinite) mixture model.

First attempt:

Look at $\boldsymbol{\beta} \sim \text{Dirichlet}(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K})$ with $\alpha_0 = 1$:



Problem: for each component z, $\mathbb{E}\beta_z = \frac{1}{K} \to 0$.

The issue is that the Dirichlet is symmetric.

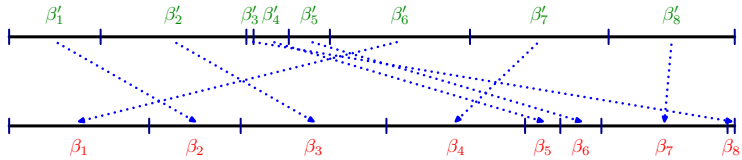
We need an asymmetric approach, where the large components are first on average.

A first attempt at defining an infinite mixture model is to take the limit of finite mixture models. This doesn't work because Dirichlet distributions are symmetric, but the limiting index set $\{1, 2, \ldots\}$ is intrinsically asymmetric.

Size-biased permutation

Solution: take a size-biased permutation of the components:

- Generate $\boldsymbol{\beta'} \sim \text{Dirichlet}(\frac{\alpha_0}{K}, \ldots, \frac{\alpha_0}{K})$:
- $S \leftarrow \emptyset$
- For $z = 1, \ldots, K$:
  - Choose $j \sim \text{Multinomial}(\boldsymbol{\beta'})$ conditioned on $j \notin S$
  - $\beta_z \leftarrow \beta'_j$
  - $S \leftarrow S \cup \{j\}$



Stick-breaking characterization of distribution of $\boldsymbol{\beta}$:

Define $v_z = \frac{\beta_z}{\sum_{z'=z}^{K} \beta_{z'}}$ as the fraction of the tail-end of the stick

Fact: $v_z \sim \text{Beta}(1 + \frac{\alpha_0}{K}, \alpha_0 - \frac{z\alpha_0}{K})$.

A way to get an asymmetric set of component probabilities is to generate from a symmetric Dirichlet and permute the probabilities. The size-biased permutation will tend to put the large ones first on average. Note: the distribution on sorted component probabilities when $K = \infty$ is the Poisson-Dirichlet distribution.

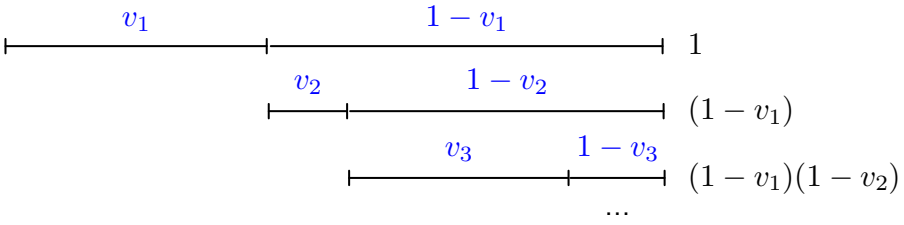
The stick-breaking (GEM) distribution

$$v_z \sim \text{Beta}(1 + \tfrac{\alpha_0}{K}, \alpha_0 - \tfrac{z\alpha_0}{K})$$

Now we can take the limit: $\downarrow K \to \infty$

$$v_z \sim \text{Beta}(1, \alpha_0)$$

1:1 relationship between stick-breaking proportions $\mathbf{v} = (v_1, v_2, \ldots)$ and stick-breaking probabilities $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots)$



$$v_z = \frac{\beta_z}{\beta_z + \beta_{z+1} + \cdots} \qquad \beta_z = (1 - v_1) \cdots (1 - v_{z-1}) v_z$$

Write $\boldsymbol{\beta} \sim \text{GEM}(\alpha_0)$ to denote the stick-breaking distribution.

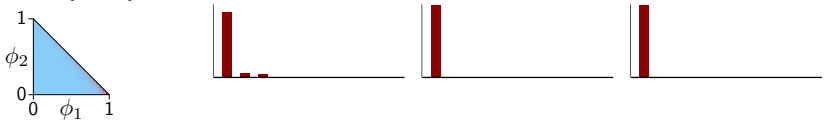
Size-biased permutation motivates the asymmetric GEM distribution, but it can be defined directly. The stick-breaking probabilities decrease exponentially in expectation, but of course sometimes a large stick can follow a small one. It can be shown that the stick-breaking probabilities sum to 1 with probability 1.

Examples of the GEM distribution

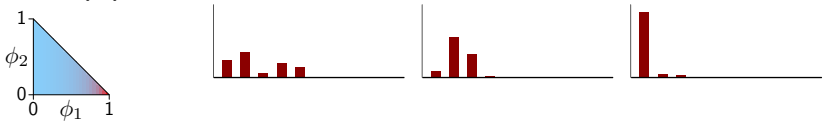
$$v_z \sim \text{Beta}(1, \alpha_0)$$

As α_0 increases, sticks decay slower $\Rightarrow$ more clusters

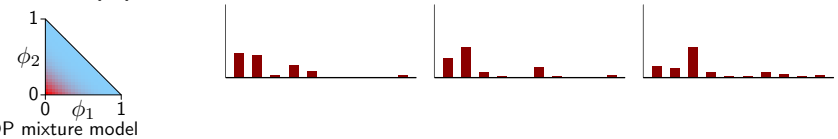
GEM(0.3)



GEM(1)



GEM(3)



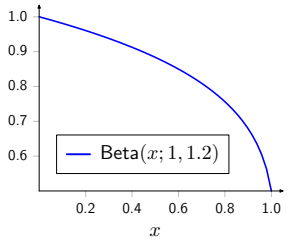
This slide shows draws from the GEM distribution for three values of α_0. No matter what value α_0 takes, the stick lengths decrease in expectation.

A cautionary tale about point estimates

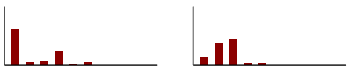
Question: what is the most likely value of $\boldsymbol{\beta} \sim \text{GEM}(1.2)$?

$p(\mathbf{v}) = \prod_{z=1}^{K} \text{Beta}(v_z; 1, 1.2)$

For each z, best $v_z = 0$, so best $\beta_z = 0$.



But in typical draws, components decay...



A contradiction? No!

- Problem: mode not representative of entire distribution
- Solution: need inference algorithms that work with entire distribution

There is another complication, which is that the densities between the two parameterizations are related through a non-identity Jacobian: $p(\boldsymbol{\beta}) = p(\mathbf{v}) \cdot \text{Jacobian}$. Therefore, $\text{argmax}_{\mathbf{v}}\, p(\mathbf{v})$ does not necessarily correspond to $\text{argmax}_{\boldsymbol{\beta}}\, p(\boldsymbol{\beta})$. The most likely point depends on which parameterization you pick. Full Bayesian inference integrates out these parameters, so parameterization is not an issue.

DP mixture model

~~Finite~~ DP mixture model

$\boldsymbol{\beta} \sim$ ~~Dirichlet$_K(\alpha, \ldots, \alpha)$~~ GEM$(\alpha)$
For each component $z \in$ ~~$\{1, \ldots, K\}$~~ $\{1, 2, \ldots\}$:
  $\phi_z \sim G_0$
For each data point $i \in \{1, \ldots, n\}$:
  $z_i \sim$ Multinomial$(\boldsymbol{\beta})$
  $x_i \sim F(\phi_{z_i})$

Mean-field inference [Blei, Jordan, 2005]:

Approximation in terms of stick-breaking proportions $\mathbf{v}$:

$$q(\boldsymbol{\beta}) = \prod_{z=1}^{\infty} q(v_z)$$

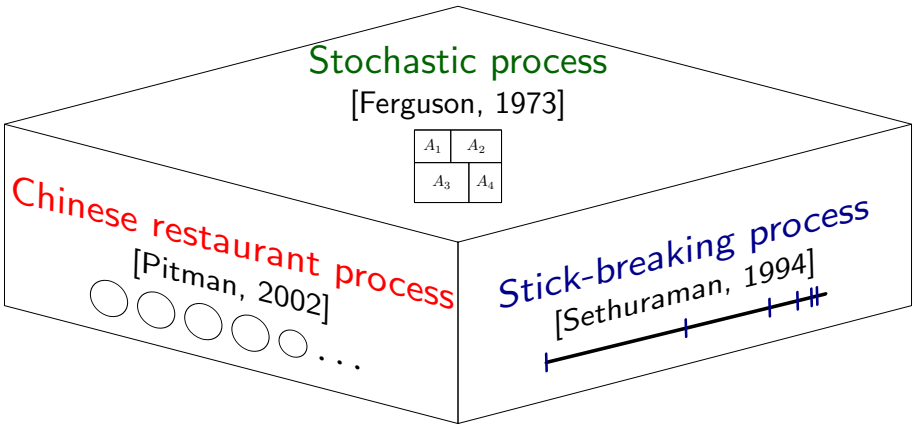
How to deal with an infinite number of parameters?
Choose a truncation level K and force $q(v_K) = 1$.
Now $q(v_z)$ and $q(\phi_z)$ for $z > K$ are irrelevant.

Finally, we formally define the DP mixture model. In practice, there is not a noticeable difference between using a finite Dirichlet prior with small concentration parameters and using a truncated stick-breaking prior. After all, both converge to the DP. A more empirically important knob is the choice of inference algorithm. It is important to note that using a stick-breaking truncation of K is not the same as just using a K-component mixture model with concentration parameters that do not scale with $1/K$. In the former, as K increases, the approximation to the same DP becomes strictly better, whereas in the latter, the models become more complex.

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Three views of the Dirichlet process



We have focused on the definition of the Dirichlet process via the stick-breaking definition. Later, we will continue to use it to define structured models. But for completeness, we present some other definitions of the DP, which are useful for theoretical understanding and developing new algorithms.

Definition of the Dirichlet process

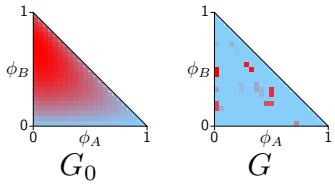
DP mixture model

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each component $z \in \{1, 2, \dots\}$:
  $\phi_z \sim G_0$
$\Big\}\ G \sim \text{DP}(\alpha, G_0)$

For each data point $i \in \{1, \dots, n\}$:
  $z_i \sim \text{Multinomial}(\boldsymbol{\beta})$ $\}\psi_i \sim G$
  $x_i \sim F(\phi_{z_i})$ $\}x_i \sim F(\psi_i)$

$G = \sum_{z=1}^{\infty} \beta_z \delta_{\phi_z}$ fully specifies the parameters.

Write $G \sim \text{DP}(\alpha, G_0)$ to mean G has a Dirichlet process prior.



We present one definition of the Dirichlet process based on the stick-breaking construction, building from the DP mixture model. The Dirichlet process is a distribution on distributions G, where G is constructed by combining stick-breaking probabilities and i.i.d. draws from G_0. This allows us to encapsulate both the component probabilities and parameters into one object G. It turns out this prior over G can be defined in other ways as we shall see.

Stochastic process definition

$G \sim \text{DP}(\alpha, G_0)$

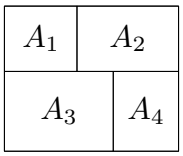
$\Updownarrow$

Finite partition property:

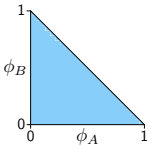
$(G(A_1), \ldots, G(A_m)) \sim \text{Dirichlet}(\alpha G_0(A_1), \ldots, \alpha G_0(A_m))$

for all partitions $(A_1, \ldots, A_m)$ of Ω,

where Ω is the set of all possible parameters.



Example: $\Omega =$



(the set of multinomial parameters)

When Ω is finite, Dirichlet process = Dirichlet distribution.

Significance: theoretical properties, compact notation, defining HDPs

The stochastic process definition is an alternative way to define G. Unlike the stick-breaking construction, the stochastic process definition is more declarative, expressing the distribution of G in terms of the distribution of parts of G. This is typically the way a stochastic process is defined. Note that when Ω is a finite set, the Dirichlet process is equivalent to an ordinary Dirichlet distribution. Viewed in this light, the Dirichlet process is just a generalization of the Dirichlet distribution.

Chinese restaurant process

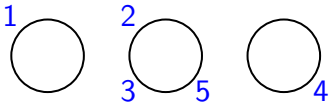
What is the distribution of $\psi_i \mid \psi_1, \ldots, \psi_{i-1}$ marginalizing out G?

Metaphor: customer = data point, dish = parameter,
table = cluster

A Chinese restaurant has an infinite number of tables.

Customer i:

- joins the table of a previous customer j and shares the dish, or
- starts a new table and randomly picks a new dish from G_0



$\ldots$ probability: $\frac{\alpha_0}{0+\alpha_0}$ $\frac{\alpha_0}{1+\alpha_0}$ $\frac{1}{2+\alpha_0}$ $\frac{\alpha_0}{3+\alpha_0}$ $\frac{2}{4+\alpha_0}$

In symbols:

$$\psi_i \mid \psi_1, \ldots, \psi_{i-1} \sim \frac{1}{\alpha+i-1}\left(\sum_{j=1}^{i-1} \delta_{\psi_j} + \alpha G_0\right)$$

Rich-gets-richer property: tables with more customers get more.

Significance: leads to efficient sampling algorithms.

The Chinese restaurant process (CRP) is yet a third way to view the Dirichlet process. This time, we are not interested in G itself, but rather draws from G with G integrated out. Formally, the CRP is a distribution over partitions (clusterings) of the data points. The Pólya urn refers to distribution over dishes. An important property about the CRP is that despite its sequential definition, the dishes (and tables) are actually exchangeable, meaning $p(\psi_1, \ldots, \psi_n) = p(\psi_{\pi(1)}, \ldots, \psi_{\pi(n)})$ for any permutation π. This can be directly seen as a consequence of de Finetti's theorem.

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Building DP-based structured models

DP

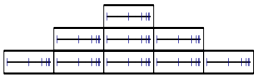
single mixture model

HDP

several mixture models
sharing same components

HDP-HMM

each state has a
mixture model over next states



HDP-PCFG

each state has a
mixture model over pairs of states

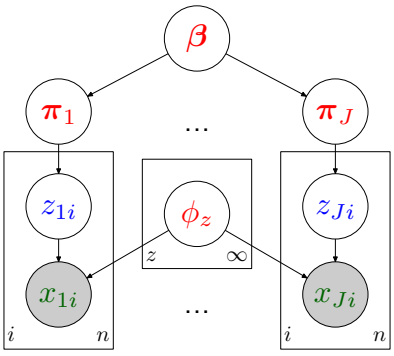
We will use the DP mixture model as a building block in creating more complex models. First, we develop the HDP mixture model, which allows many mixture models to share the same inventory of components. Based on this, we then create structured models such as HDP-HMMs and HDP-PCFGs.

[Teh, et al., 2006]

Hierarchical DP mixture models

Suppose we have a collection of J mixture models.

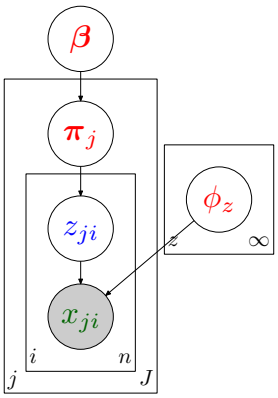
Goal: share common inventory of mixture components



- Component parameters $\{\phi_z\}$ are shared globally
- Each mixture model has individual component probabilities π_j tied via β

The top-level probabilities β determine the overall popularity of components, to a first-order approximation, which components are active. Each π_j is based on β, but is tailored for group j. An application is topic modeling, where each group is a document, each component is a topic (distribution over words), and each data point is a word. In this light, the HDP is an nonparametric extension of LDA. Note: the HDP mixture model can also be defined using the stochastic process definition or the Chinese restaurant franchise, the hierarchical analogue of the CRP.

Hierarchical DP mixture models



$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each component $z \in \{1, 2, \ldots\}$:
  $\phi_z \sim G_0$
For each group $j \in \{1, \ldots, J\}$:
  $\boldsymbol{\pi}_j \sim \text{DP}(\alpha', \boldsymbol{\beta})$
  For each data point $i \in \{1, \ldots, n\}$:
    $z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j)$
    $x_{ji} \sim F(\phi_{z_{ji}})$

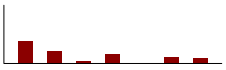
$\boldsymbol{\beta}$ determines the rough component probabilities

$\boldsymbol{\pi}_j \sim \text{DP}(\alpha', \boldsymbol{\beta})$ makes $\boldsymbol{\pi}_j$ close to $\boldsymbol{\beta}$

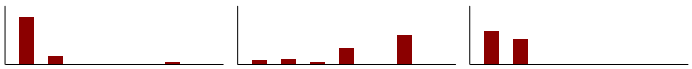
We tie the π_js together using a Dirichlet process, which is best understood in terms of the stochastic process definition, where the distributions involved are over the positive integers. Note: there is another definition of the HDP where for each group j, we draw a separate set of stick-breaking probabilities from GEM(α'), which then can be used to induce the corresponding distribution on $\boldsymbol{\pi}_j$. The disadvantage of this approach is that it introduces extra variables and indirect pointers which complicates variational inference.

Sharing component probabilities

Draw top-level component probabilities $\boldsymbol{\beta} \sim \text{GEM}(\alpha)$:

$\alpha = 2$ 

Draw group-level component probabilities $\boldsymbol{\pi} \sim \text{DP}(\alpha', \boldsymbol{\beta})$:

$\alpha' = 1$ 

$\alpha' = 5$ 

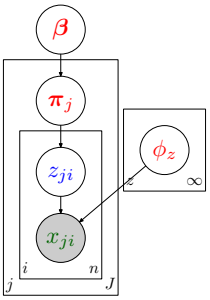
$\alpha' = 20$ 

$\alpha' = 100$ 

As α' increases, the more $\boldsymbol{\pi}$ resembles $\boldsymbol{\beta}$.

There are two concentration parameters, α and α'. The parameter α controls how many components there are overall and α' controls how similar the prior distributions over components across groups.

Mean-field inference for the HDP



$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each component $z \in \{1, 2, \dots\}$:
  $\phi_z \sim G_0$
For each group $j \in \{1, \dots, J\}$:
  $\boldsymbol{\pi}_j \sim \text{DP}(\alpha', \boldsymbol{\beta})$
  For each data point $i \in \{1, \dots, n\}$:
    $z_{ji} \sim \text{Multinomial}(\boldsymbol{\pi}_j)$
    $x_{ji} \sim F(\phi_{z_{ji}})$

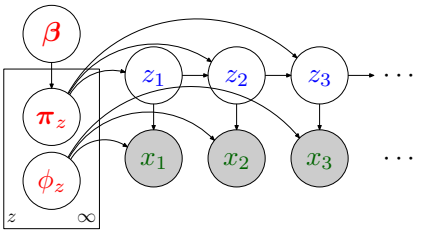
Mean-field approximation: $q(\boldsymbol{\beta}) \prod_{z=1}^{\infty} q(\phi_z) \prod_{j=1}^{J} q(\boldsymbol{\pi}_j) \prod_{i=1}^{n} q(z_i)$

- As in variational DP, truncate $\boldsymbol{\beta}$ at level K
- $\boldsymbol{\pi}_j \sim \text{DP}(\alpha', \boldsymbol{\beta})$ reduces to finite K-dimensional Dirichlet $\boldsymbol{\pi}_j \sim \text{Dirichlet}(\alpha', \boldsymbol{\beta})$, so we can use finite variational techniques for updating $q(\boldsymbol{\pi})$.
- Let $q(\boldsymbol{\beta}) = \delta_{\beta^*}(\boldsymbol{\beta})$ for tractability, optimize with gradient descent

The only added challenge in doing mean-field inference in the HDP is how to optimize the top-level components. Because the GEM prior is not conjugate with the DP draws from $\boldsymbol{\beta}$, it's convenient to let $q(\boldsymbol{\beta})$ be degenerate and optimize it using standard gradient methods.

[Beal, et al., 2002; Teh, et al., 2006]

HDP hidden Markov models



$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each state $z \in \{1, 2, \dots\}$:
  $\phi_z \sim G_0$
  $\boldsymbol{\pi}_z \sim \text{DP}(\alpha', \boldsymbol{\beta})$
For each time step $i \in \{1, \dots, n\}$:
  $z_{i+1} \sim \text{Multinomial}(\boldsymbol{\pi}_{z_i})$
  $x_i \sim F(\phi_{z_i})$

Each state z is a component and has the following:

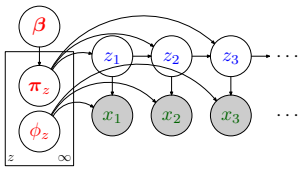
- $\boldsymbol{\pi}_z$: transition parameters
- ϕ_z: emission parameters

Key:

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$ specifies which states will be used (global)
$\boldsymbol{\pi}_z \sim \text{DP}(\alpha', \boldsymbol{\beta})$ specifies distribution over next states (per state)

We can think of an HMM as a mixture model, where each component corresponds to a state of the HMM. Given a state, we need to emit an observation and advance to the next state. The component parameters must specify the distributions associated with these stochastic choices. The transition parameters must specify a distribution over states, which is naturally represented with a draw from a DP. The HDP framework allows us to tie all the transition DPs.

Mean-field inference for the HDP-HMM



$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each state $z \in \{1, 2, \dots\}$:
  $\phi_z \sim G_0$
  $\boldsymbol{\pi}_z \sim \text{DP}(\alpha', \boldsymbol{\beta})$
For each time step $i \in \{1, \dots, n\}$:
  $z_{i+1} \sim \text{Multinomial}(\boldsymbol{\pi}_{z_i})$
  $x_i \sim F(\phi_{z_i})$

Structured mean-field approximation: $q(\boldsymbol{\beta})\left(\prod_{z=1}^{\infty} q(\phi_z)q(\boldsymbol{\pi}_z)\right)q(\mathbf{z})$

EM:

E-step: run forward-backward using $p(z' \mid z, \boldsymbol{\pi}) = \pi_{zz'}$

M-step: normalize transition counts: $\pi_{zz'} \propto C(z \to z')$

Mean-field:

E-step: run forward-backward using multinomial weights $W(z, z')$

M-step: compute multinomial weights given transition counts:

$$W(z, z') = \frac{\exp \Psi(\alpha' \beta_{z'} + C(z \to z'))}{\exp \Psi(\alpha' + C(z \to *))}$$

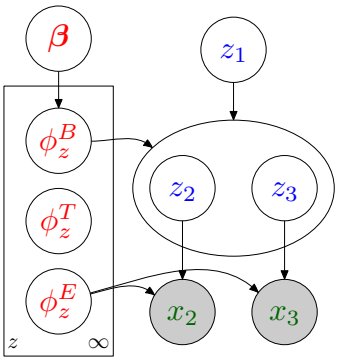
Top-level: optimize $q(\boldsymbol{\beta}) = \delta_{\boldsymbol{\beta}^*}(\boldsymbol{\beta})$ using gradient descent

Mean-field inference is similar to EM for classical HMMs. It is straightforward to check that the forward-backward dynamic program for the E-step still remains tractable when using a non-degenerate $q(\boldsymbol{\phi})$. Here, K is the truncation applied to $\boldsymbol{\beta}$.

[Liang, et al., 2007]

HDP probabilistic context-free grammars

$\boldsymbol{\beta} \sim \text{GEM}(\alpha)$
For each symbol $z \in \{1, 2, \dots\}$:
  $\phi_z^T \sim \text{Dirichlet}$
  $\phi_z^E \sim G_0$
  $\phi_z^B \sim \text{DP}(\alpha', \boldsymbol{\beta}\boldsymbol{\beta}^T)$
For each node i in the parse tree:
  $t_i \sim \text{Multinomial}(\phi_{z_i}^T)$ [rule type]
  If $t_i = \textsc{Binary-Production}$:
    $(z_{L(i)}, z_{R(i)}) \sim \text{Multinomial}(\phi_{z_i}^B)$ [children symbols]
  If $t_i = \textsc{Emission}$:
    $x_i \sim \text{Multinomial}(\phi_{z_i}^E)$ [terminal symbol]



	HDP-HMM	HDP-PCFG
At each i...	transition and emit	produce or emit
DP is over...	1 next state	2 children symbols

Variational inference: modified inside-outside

The nonparametric Bayesian machinery carries over from HMMs to PCFGs again by replacing normalization with an application of $\exp(\Psi(\cdot))$. The addition of DPs centered on the cross-product of the top-level distribution does complicate the optimization of $\boldsymbol{\beta}$.

Prediction revisited

Train on $\mathbf{x}$; now get test x_{new}, want to find best z_{new}.

$$p(z_{\text{new}} \mid x_{\text{new}}) = \mathbb{E}_{p(\theta|\mathbf{x})} p(z_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta})$$
$$\approxeq \mathbb{E}_{q(\theta)} p(z_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta})$$

Contrast with training (has $\log$):

$$q(z_{\text{new}}) \propto \exp \mathbb{E}_{q(\theta)} \log p(z_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta})$$

To find $\text{argmax}_{z_{\text{new}}} \mathbb{E}_{q(\theta)} p(z_{\text{new}} \mid x_{\text{new}}, \boldsymbol{\theta})$, consider all z_{new}: intractable when z_{new} is a parse tree because cannot use dynamic programming (unlike training)

Approximations:

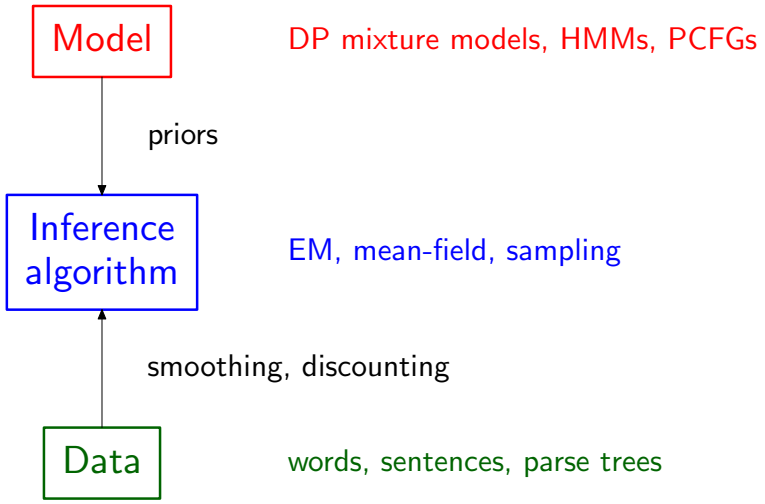
- Use mode of $q(\boldsymbol{\theta})$
- Maximize expected log-likelihood as in training
- Reranking: get an n-best list using proxy; choose best one

We have spent most of our time approximating posteriors. Now let's see how we can use them for structured prediction. When we have point estimates of parameters, the standard prediction rule is the same as running a hard E-step during training. However, with non-degenerate parameters, the $\log$ during training decouples parts of the structure, allowing dynamic programming. At test time, dynamic programming can no longer be used in the absence of the $\log$.

Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

Top-down and bottom-up



The top-down (Bayesian) approach: define a model (such as a DP mixture model) and then approximate the posterior using an inference algorithm. The bottom-up approach: design an algorithm directly based on properties of the data. Sometimes the two approaches coincide: for example, smoothing and discounting can have interpretations as Bayesian priors. Many NLP methods are reasonable and effective but lack a top-level interpretation, which could provide additional insight.

Why doesn't my variational DP work?

Modeling issues

- Setting concentration parameters is still an art.
- Is the DP even the right prior to use?

Inference issues

- Mean-field is an approximation of the true posterior.
- The coordinate-wise descent algorithm for optimizing the mean-field objective is susceptible to local minima problems, just as is EM.

We have focused on using Dirichlet processes with variational inference for clustering, sequence modeling, etc. However, there are two places something could go wrong: having bad modeling assumptions or having a local optima problem with inference.

Other inference methods

We focused on variational inference using the stick-breaking construction.

Algorithm:

- Variational (mean-field, expectation propagation, etc.): deterministic, fast, converge to local optima
- Sampling (Gibbs, split-merge, etc.): converges to true posterior (but don't know when), could be stuck in local optima for a long time

Representation:

- Stick-breaking construction: simple, concrete
- Chinese restaurant process: works well for simple mixture models

If inference is a problem, one could consider other inference methods. There are two major axes along which we could classify an inference algorithm: variational/sampling or CRP/stick-breaking, leading to four different algorithms. Each has its advantages, but the problem of posterior inference is fundamentally a challenging one.

The Bayesian elephant

Even just the elephant posterior is intractable.

Mean-field guy: "feels like smoothing/discounting"
Sampling guy: "feels like stochastic hill-climbing"

A common property: rich gets richer

Viewing a model through one particular inference algorithm leads to a particular way of thinking about the problem. It is important to remember that it is not the complete story. Of course, we are talking about one model, so there are common properties (e.g., rich gets richer) which are shared by the different inference algorithms.

Other nonparametric priors

- Pitman-Yor process: component probabilities less sparse compared to the DP; yields power-law distributions useful for language modeling. [Pitman, Yor, 1997; Teh, 2006]
- Beta process / Indian buffet process: allows data points to belong to more than one cluster; infinite number of latent **features**, each data point generated using a finite subset. [Griffiths, Ghahramani, 2007; Thibaux, Jordan, 2007]
- etc.

Many of these priors also have stochastic process, Chinese restaurant, stick-breaking counterparts.

If inference is not the problem, perhaps the model is not suitable for the task at hand. For example, Pitman-Yor priors give power law distributions more suitable for modeling the distribution of words than Dirichlet priors.

Other types of problems

Clustering (unsupervised)

	non-Bayesian	Bayesian
parametric	k-means, EM	Bayesian mixture models
nonparametric	agglomerative clustering	Dirichlet processes

Classification (supervised)

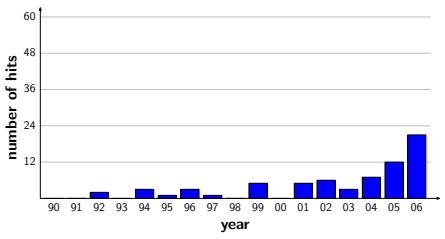
	non-Bayesian	Bayesian
parametric	logistic regression, SVMs	Bayesian logistic regression
nonparametric	nearest neighbors, kernel methods	Gaussian processes

We've focused primarily on models built from mixtures, which are useful for inducing hidden structure in data. We have extended classical mixture models along two axes: being more Bayesian and being more nonparametric. Moving along these two axes is also applicable in other methods, e.g. classification, regression, etc.

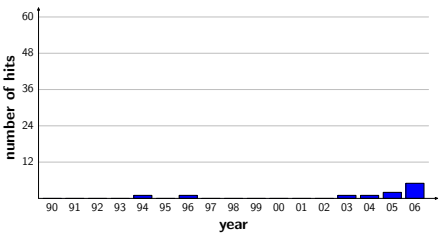
Roadmap

- Part I
  - Distributions and Bayesian principles
  - Variational Bayesian inference
  - Mean-field for mixture models
- Part II
  - Emulating DP-like qualities with finite mixtures
  - DP mixture model
  - Other views of the DP
- Part III
  - Structured models
  - Survey of related methods
  - Survey of applications

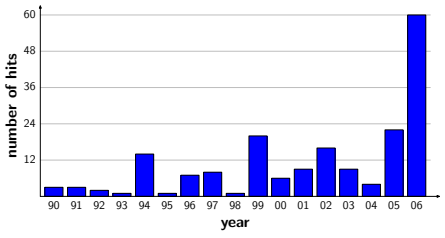
Bayesian trends in NLP



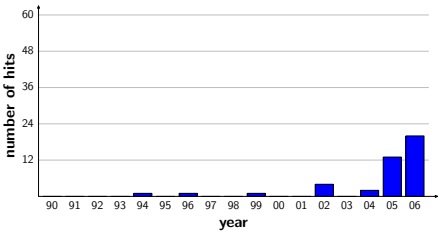
"Bayesian"



"Dirichlet"



"prior"



"posterior"

There has been a significant increase in the number of papers at ACL about Bayesian modeling, as measured by the number of Google hits on ACL papers.

Applications

- Topic modeling
  - finite Bayesian model; variational [Blei, et al., 2003]
  - HDP-based model; sampling [Teh, et al., 2006]
- Language modeling
  - Pitman-Yor $\Rightarrow$ power-law; sampling [Goldwater, et al., 2005]
  - Kneser-Ney $\Leftrightarrow$ Pitman-Yor; sampling [Teh, 2006]
- POS induction using a finite Bayesian HMM
  - Collapsed sampling [Goldwater, Griffiths, 2007]
  - Variational [Johnson, 2007]
- Parsing using nonparametric grammars
  - Collapsed sampling [Johnson, et al., 2006]
  - Collapsed sampling [Finkel, et al., 2007]
  - Variational stick-breaking representation [Liang, et al., 2007]
- Coreference resolution
  - Supervised clustering; collapsed sampling [Daume, Marcu, 2005]
  - HDP-based model; sampling [Haghighi, Klein, 2007]

This is a list of some recent work that applies Bayesian and/or nonparametric methods to NLP problems.

Conclusions

- Bayesian methods are about keeping uncertainty in parameters
- Variational inference: hard EM $\rightarrow$ EM $\rightarrow$ mean-field
  - Represent a distribution over parameters
- The nonparametric Dirichlet process prior penalizes use of extra clusters
  - Inference algorithms have rich-gets-richer property
- Structured models can be built from nonparametric Bayesian parts

This ends our tutorial on Structured Bayesian Nonparametric Models with Variational Inference.

Resources

- References
- Derivations
- Glossary and notation

References (DP theory)

1. T. S. Ferguson. A Bayesian analysis of some nonparametric problems. Annals of Statistics, 1973. Defines the Dirichlet process using stochastic processes.
2. D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. Annals of Statistics, 1973. Connection between DP and the Pólya urn.
3. J. Pitman. Combinatorial stochastic processes. UC Berkeley, 2002. A comprehensive set of tutorial notes.
4. C. E. Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. Annals of Statistics, 1974.
5. M. Beal and Z. Ghahramani and C. Rasmussen. The infinite hidden Markov model. NIPS, 2002. Introduces the (infinite) HDP-HMMs.
6. Y. W. Teh and M. I. Jordan and M. Beal and D. Blei. Hierarchical Dirichlet processes. JASA, 2006. Sets up the HDP, a mechanism of tying different DPs together (gives another treatment of HDP-HMMs).

References (variational inference)

1. D. Blei and A. Ng and M. I. Jordan. Latent Dirichlet allocation. JMLR, 2003. Uses variational inference for LDA.
2. D. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. Bayesian Analysis, 2005. First mean-field algorithm for DP mixtures (stick-breaking representation).
3. K. Kurihara and M. Welling and N. Vlassis. Accelerated variational Dirichlet mixture models. NIPS, 2007. Use KD-trees to speed up inference for DP mixtures (stick-breaking representation).
4. Y. W. Teh and D. Newman and M. Welling. A collapsed variational Bayesian inference algorithm for Latent Dirichlet Allocation. NIPS, 2007. Variational inference in the collapsed CRP representation for LDA.
5. K. Kurihara and M. Welling and Y. W. Teh. Collapsed variational Dirichlet process mixture models. IJCAI, 2007. Compares several variational inference algorithms.

References (Bayesian applications)

1. S. Goldwater and T. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. ACL, 2007.
2. M. Johnson. Why doesn't EM find good HMM POS-taggers?. EMNLP/CoNLL, 2007.
3. K. Kurihara and T. Sato. An application of the variational Bayesian approach to probabilistic context-free grammars. International Joint Conference on Natural Language Processing Workshop Beyond Shallow Analyses, 2004.
4. K. Kurihara and T. Sato. Variational Bayesian grammar induction for natural language. International Colloquium on Grammatical Inference, 2006.
5. M. Johnson and T. Griffiths and S. Goldwater. Adaptor grammars: a framework for specifying compositional nonparametric Bayesian models. NIPS, 2006.
6. H. Daume and D. Marcu. Bayesian query-focused summarization. COLING/ACL, 2006.

References (nonparametric applications)

1. M. Johnson and T. Griffiths and S. Goldwater. Adaptor grammars: a framework for specifying compositional nonparametric Bayesian models. NIPS, 2006.
2. J. R. Finkel and T. Grenager and C. Manning. The infinite tree. ACL, 2007.
3. P. Liang and S. Petrov and M. I. Jordan and D. Klein. The infinite PCFG using hierarchical Dirichlet processes. EMNLP/CoNLL, 2007.
4. S. Goldwater and T. Griffiths and M. Johnson. Interpolating between types and tokens by estimating power-law generators. NIPS, 2005.
5. Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. COLING/ACL, 2006.
6. H. Daume and D. Marcu. A Bayesian model for supervised clustering with the Dirichlet process prior. JMLR, 2005.
7. S. Goldwater and T. Griffiths and M. Johnson. Contextual dependencies in unsupervised word segmentation. COLING/ACL, 2006.
8. A. Haghighi and D. Klein. Unsupervised coreference resolution in a nonparametric Bayesian model. ACL, 2007.

Resources

- References
- Derivations
- Glossary and notation

KL-divergence and normalization

$$\mathrm{KL}(q||p) \stackrel{\text{def}}{=} \mathbb{E}_q \log \frac{q(\theta)}{p(\theta)}$$

Usually, we only know p up to normalization:

Example: $p = p(\boldsymbol{\theta} \mid \mathbf{x}), \tilde{p} = p(\boldsymbol{\theta}, \mathbf{x}), Z = p(\mathbf{x})$

$$p = \frac{\tilde{p}}{Z} \begin{array}{l} \text{(unnormalized distribution: tractable)} \\ \text{(normalization constant: intractable)} \end{array}$$

No problem, due to the following identity:

$$\mathrm{KL}(q||p) = \mathrm{KL}(q||\tilde{p}) + \log Z$$ 

- $\operatorname{argmin}_q \mathrm{KL}(q||p) = \operatorname{argmin}_q \mathrm{KL}(q||\tilde{p})$
- Since $\mathrm{KL} \geq 0$, we get a lower bound on $\log Z$ as a bonus:

$$\underbrace{\log Z}_{\text{intractable}} \geq \underbrace{-\mathrm{KL}(q||\tilde{p})}_{\text{tractable}}$$

Note that the optimization problem we wish to solve is to minimize $\mathrm{KL}(q(\boldsymbol{\theta}, \mathbf{z})||p(\boldsymbol{\theta}, \mathbf{z} \mid x)$, but unfortunately this quantity is in terms of the posterior we are trying to compute! Fortunately, to optimize this objective, it suffices to know the posterior up to a normalization constant, as we demonstrate here.

KL-divergence and normalization

$$\begin{aligned}
&\mathrm{KL}(q||p) && (1)\\
&= \mathbb{E}_q[\log \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}] && (2)\\
&= \mathbb{E}_q[\log q(\boldsymbol{\theta}) - \log p(\boldsymbol{\theta})] && (3)\\
&= \mathbb{E}_q[\log q(\boldsymbol{\theta})] - \mathbb{E}_q[\log p(\boldsymbol{\theta})] && (4)\\
&= -H(q) - \mathbb{E}_q[\log p(\boldsymbol{\theta})] && (5)\\
&= -H(q) - \mathbb{E}_q[\log \frac{\tilde{p}(\boldsymbol{\theta})}{Z}] && (6)\\
&= \underbrace{-H(q) - \mathbb{E}_q[\log \tilde{p}(\boldsymbol{\theta})]}_{\text{free energy}} + \log Z && (7)\\
&= \mathrm{KL}(q||\tilde{p}) + \log Z && (8)
\end{aligned}$$

We relate KL-divergence with a normalized second argument to the case where the second argument is not normalized.

Entropy

Entropy measures the amount of uncertainty in a distribution.

Definition: $H(q) \stackrel{\text{def}}{=} -\mathbb{E}_q \log q(\boldsymbol{\theta}) = -\int q(\boldsymbol{\theta}) \log q(\boldsymbol{\theta}) d\boldsymbol{\theta}$

Property: $H(q) \geq 0$

Suppose q is fully-factorized (as in mean-field): $q(\boldsymbol{\theta}) = \prod_{i=1}^n q_i(\theta_i)$

Then the entropy decomposes:

$$
\begin{aligned}
& H(q) && (1)\\
&= -\mathbb{E}_q[\log \textstyle\prod_{i=1}^n q_i(\theta_i)] && (2)\\
&= \textstyle\sum_{i=1}^n -\mathbb{E}_q[\log q_i(\theta_i)] && (3)\\
&= \textstyle\sum_{i=1}^n -\mathbb{E}_{q_i}[\log q_i(\theta_i)] && (4)\\
&= \textstyle\sum_{i=1}^n H(q_i) && (5)
\end{aligned}
$$

Entropy is a measure of uncertainty. An important property about the entropy of distributions that factorize is that the entropy also decomposes.

Formal derivation of mean-field updates

Goal: optimize each q_i holding q_{-i} fixed.

Strategy: manipulate $\mathrm{KL}(q||p)$, throwing away terms that do not depend on q_i.

$$
\begin{aligned}
& \mathrm{KL}(q||p) && (1)\\
&= -H(q) - \mathbb{E}_q[\log p(\boldsymbol{\theta})] && (2)\\
&= \Big(\sum_j -H(q_j)\Big) - \mathbb{E}_q[\log p(\boldsymbol{\theta})] && (3)\\
&= -H(q_i) - \mathbb{E}_q[\log p(\boldsymbol{\theta})] + C && (4)\\
&= -H(q_i) - \mathbb{E}_{q_i}[\mathbb{E}_{q_{-i}} \log p(\boldsymbol{\theta})] + C && (5)\\
&= -H(q_i) - \mathbb{E}_{q_i}[\log(\exp \mathbb{E}_{q_{-i}} \log p(\boldsymbol{\theta}))] + C && (6)\\
&= \mathrm{KL}(q_i||\exp \mathbb{E}_{q_{-i}} \log p(\boldsymbol{\theta})) + C && (7)
\end{aligned}
$$

Recall that KL is minimized when the two arguments are equal.

Conclusion: $\operatorname{argmin}_{q_i} \mathrm{KL}(q||p) \propto \exp\{\mathbb{E}_{q_{-i}} \log p(\boldsymbol{\theta})\}$.

We derive the formal generic update for optimizing one coordinate in the mean-field approximating distribution, holding all else fixed.

Derivation of $\exp \Psi$

Goal: show $W(z) \stackrel{\text{def}}{=} \exp\{\mathbb{E}_{\text{Dirichlet}(\phi;\alpha)} \log \phi_z\} = \frac{\exp\{\Psi(\alpha_z)\}}{\exp\{\Psi(\sum_{z'} \alpha_{z'}\}}$

Write the Dirichlet distribution in exponential family form:

$$p(\phi \mid \alpha) = \exp\Big\{ \sum_z \alpha_z \log \phi_z - \Big[\sum_z \log \Gamma(\alpha_z) - \log \Gamma\Big(\sum_z \alpha_z\Big)\Big]\Big\}$$

Sufficient statistics: $\log \phi_z$

Log-partition function: $\sum_z \log \Gamma(\alpha_z) - \log \Gamma\Big(\sum_z \alpha_z\Big)$

Fact: mean of sufficient statistics = derivative of log-partition function

Definition: the digamma function is $\Psi(x) = \frac{\partial \log \Gamma(x)}{\partial x}$

$$\mathbb{E} \log \phi_z = \frac{\partial A(\alpha)}{\partial \alpha_z} = \Psi(\alpha_z) - \sum_{z'} \Psi(\alpha_{z'})$$

Exponentiating both sides:

$$\exp\{\mathbb{E} \log \phi_z\} = \frac{\exp\{\Psi(\alpha_z)\}}{\exp\{\Psi(\sum_{z'} \alpha_{z'}\}}$$

Resources

- References
- Derivations
- Glossary and notation

Glossary

Bayesian inference: A methodology whereby a prior over parameters is combined with the likelihood of observed data to produce a posterior using the laws of probability.

Chinese restaurant process: The distribution of the clustering induced by draws from the Dirichlet process (marginalizing out component probabilities and parameters).

Conjugacy: Two distributions are conjugate (e.g., Dirichlet and multinomial).

Dirichlet distribution: A distribution over (parameters of) finite probability distributions.

Dirichlet process: A distribution over arbitrary distributions (generalizes the Dirichlet distribution).

Glossary

Likelihood: The probability of observing the data.

Marginalization: Integrating or summing out unobserved quantities in a model.

Marginal likelihood: The probability of the observed data, marginalizing out unknown parameters. This quantity is lower bounded in variational inference.

Markov Chain Monte Carlo: A randomized approximate inference algorithm with nice asymptotic properties.

Mean-field: A fully-factorized approximation for variational inference.

Glossary

Nonparametric: Refers to models where the number of effective parameters can grow with the amount of data. The Dirichlet process is an example of a nonparametric prior.

Posterior: The distribution over unknown quantities in a model (parameters) conditioned on the observed data.

Prior: A distribution over unknown quantities in the model (parameters) before observing data.

Stick-breaking representation: A constructive definition of the Dirichlet process prior.

Variational Bayes: Variational inference for computing the posterior in Bayesian models.

Notation

$\boldsymbol{\theta}$	All parameters $(\boldsymbol{\beta}, \boldsymbol{\phi})$
$\mathbf{z}$	All hidden variables
β_z	Probability of component z
v_z	Stick-breaking proportion of component z
ϕ_z	Parameters of component z
z_i	Component that point i is assigned to
x_i	Data point i
α_0	Concentration parameter of the Dirichlet process prior
G_0	Base distribution parameter of the Dirichlet process prior
G	A draw from the Dirichlet process
ψ_i	Component parameters used to generate point i