CS547Q Statistical Modeling with Stochastic Processes **Winter 2011**

Part 3: Applications of Dirichlet processes

Lecturer: Alexandre Bouchard-Côté *Scribe(s): Seagle Liu, Chao Xiong*

Last update: April 25, 2011

3.1 Overview

In these notes, we will see examples of how the Dirichlet process can be used as a prior in classical statistical estimation problems. We will see two examples: density estimation (and its cousin, cluster analysis), and nonparametric Bayesian GLMs (which include nonparametric regression and classification as special cases). We will introduce along the way two MCMC techniques that can be used to approximate the posterior. We will also look at the question of how to use the posterior to answer the task at hand.

3.2 First examples: applying DPs to density estimation and cluster analysis

The task in density estimation is to give an estimate, based on observed data, of an unobservable underlying probability density function. The unobservable density function is thought of as the density according to which a large population is distributed; the data are usually thought of as a random sample from that population.

In cases where the population is thought as being the union of sub-populations, the task of cluster analysis is to find the sub-population structure (usually without labeled data). Let us assume for simplicity that we wish to separate the data into two clusters.[1]

Let us take a Bayesian approach to these problems. This means that we (the modeler) need to pick a joint probability distribution over knowns and unknowns (prior+likelihood), as well as a loss function.

3.2.1 Example of joint probability distribution

When the data y_i we wish to model is continuous, e.g. when the random variables $y_i \dot{\in} \mathbb{R}^d$ have a distribution absolutely continuous with respect to the Lebesgue measure, using the Dirichlet process G directly as a prior on y_i is inappropriate, i.e. modeling the data as $y_i|G \sim G$ is not a good choice. The reason is that the predictive distribution $y'|y_1, \ldots, y_n$ will always assign positive probability to the events $(y' = a)$ for $a \in \mathbb{R}^2$, while any consistent estimator would assign probability zero to these events.[2]

[1] Note that the Dirichlet process is still a useful tool, even when the number of desired cluster is fixed. This is because each cluster that is output may need internally more than one mixture to be explained adequately under the likelihood model at hand. Consider for example doing clustering on the data used in the first problem of the second assignment.

[2] Note that in cases where y_i take values in a countable set, models of the form $y_i|G \sim G$ are used. See the next set of notes for an example.

Instead Dirichlet processes are often used as a prior over the parameters of a mixture model, as discussed in the motivation of Part 2 of this set of lecture notes. This yields a model of the form:

Definition 3.1 (Dirichlet mixture model) *Let $(\Theta, \mathcal{F}_\Theta)$ be a measurable space of parameters, and $(\mathcal{X}, \mathcal{F}_\mathcal{X})$ be a measurable space of observation. A Dirichlet mixture (DPM) model is specified by three ingredients:*

1. *A concentration parameter $\alpha_0 > 0$,*
2. *A likelihood model L (formally, a regular conditional distribution $L : \Theta \times \mathcal{F}_\mathcal{X} \to [0, 1]$),*
3. *A prior over cluster parameters G_0 (formally, a distribution $G_0 : \mathcal{F}_\Theta \to [0, 1]$).*

Given these ingredients, the DPM is defined as follows:

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha_0, G_0) \\ \underline{\theta}_i | G &\sim G \\ y_i | \underline{\theta}_i &\sim L(\underline{\theta}_i, \cdot). \end{aligned}$$

For example, in Section 2.1 of Part 2 of this set of notes, $\Theta = \mathbb{R} \times (0, \infty)$ (mean parameter times variance parameter), $\mathcal{X} = \mathbb{R}$; and L could be a normal likelihood, and G_0, a normal-scaled inverse gamma distribution (the conjugate distribution when both means and variance of L are unknown).

Note that a clustering can be obtained from a DPM by looking at the posterior frequency that pairs of customers sit together at a CRP table (i.e. a sample from the posterior of a DPM defines a clustering via the labeled partition construction of Part 2 of this set of notes). Recall that we use the notation x_i for the cluster (table) index of data point (customer) i. These variables are typically implemented by taking value in the positive integers, but note that the ordering structure is meaningless in collapsed samplers and should not be used when computing posterior statistics (and indeed, we will see in Section 3.2.3 that the sufficient statistics for computing the Bayes estimator for a classical loss take the form $\mathbf{1}[x_i = x_j]$).

A density estimator can also be obtained from a DPM by the random density $F(\tilde{y}) = \int l(\tilde{y}|\underline{\theta}) G(\mathrm{d}\underline{\theta})$, where l is a density for L.

3.2.2 Examples of loss functions

In the case of clustering, a popular choice is the rand loss between a true and putative labeled partitions ρ, ρ', denoted by $\mathrm{Rand}(\rho, \rho')$.[3]

Definition 3.2 *The rand loss is defined as the number of (unordered) pairs of data points indices $\{i, j\}$ such that $(i \sim_\rho j) \neq (i \sim_{\rho'} j)$, i.e.:*

$$\sum_{1 \leq i < j \leq n} \mathbf{1}[(i \sim_\rho j) \neq (i \sim_{\rho'} j)],$$

where:

$$(i \sim_\rho j) = \begin{cases} 1 & \text{if there is a } B \in \rho \text{ s.t.} \{i, j\} \subseteq B \\ 0 & \text{o.w.} \end{cases}.$$

In other words, a loss of one is incurred each time either: (1) two points are assumed to be in the same cluster when they should not, or (2) two points are assumed to be in different clusters when they should be in the same cluster.

[3]Note that we turn the standard notion of rand index into a loss by taking 1 - the rand index.

The rand loss has several problems, motivating other clustering losses such as the adjusted rand index, but we will look at the rand loss here since the derivation of the Bayes estimator is easy for that particular loss.

In the case of density estimation, if the task is to reconstruct the density itself, examples of loss functions include the Hellinger and KL losses. However, density estimation is usually an intermediate step for another task, and the loss should then be defined on this final task rather than on the intermediate density estimation task. For example, in the first part of the second assignment, the task under consideration is to construct a visualization of the predictive density using a large but finite number of points $\tilde{y}_j \in S$ on a grid S. In this case, an example of loss function is:

$$L(p, p') = ||p_j - p'_j||_2^2,$$

where p, p' are $|S|-$dimensional vectors of nonnegative real numbers (where p_j corresponds to the density sampled at $\tilde{y}_j$).

3.2.3 Combining probability models and loss functions to do inference

As reviewed earlier, the Bayesian framework is reductionist: given a loss function L and a probability model $(X, Y) \sim \mathbb{P}$, it prescribes the following estimator:

$$\operatorname*{argmin}_x \mathbb{E}[L(x, X)|Y].$$

We will revisit the examples of clustering and density estimation with the loss function defined in Section 3.2.2 to see how this abstract quantity can be computed or approximated in practice.

First, for the rand loss, we can write:

$$\begin{aligned}\operatorname*{argmin}_{\text{partition } \rho} \mathbb{E}\left[\text{Rand}(x, \rho)|y\right] &= \operatorname*{argmin}_{\text{partition } \rho} \sum_{i<j} \mathbb{E}\left[\mathbf{1}\left[\mathbf{1}[x_i = x_j]\right] \neq \rho_{ij}|y\right] \\ &= \operatorname*{argmin}_{\text{partition } \rho} \sum_{i<j} \left\{(1 - \rho_{ij})\mathbb{P}(x_i = x_j|y) + \rho_{ij}\left(1 - \mathbb{P}(x_i = x_j|y)\right)\right\}\end{aligned}$$

where $\rho_{i,j} = (i \sim_\rho j)$.

This means that computing an optimal bipartition of the data into two clusters can be done in two steps:

1. Simulating a Markov chain, and use the samples to estimate $\mu_{i,j} = \mathbb{P}(x_i = x_j|y)$ via Monte Carlo averages.
2. Minimize the linear objective function $\sum_{i<j}\left\{(1 - \rho_{ij})\mu_{i,j} + \rho_{ij}\left(1 - \mu_{i,j}\right)\right\}$ over bipartitions ρ.

Note that the second step can be efficiently computed using min-flow/max-cut algorithms (understanding how this algorithm works is outside of the scope of this lecture, but if you are curious, see [4]). Our focus will be on computing the first step, i.e. the posterior over the random cluster membership variables x_i. Note that the $\mu_{i,j}$ are easy to compute from samples since the Monte carlo average of a function f applied to MCMC samples converges to the expectation of the function under the stationary distribution (as long as f is integrable, which is the case here since the indicator function is bounded). Sampling will be the topic of the next section.

For density estimation, going over the same process for the special loss defined Section 3.2.2, we get:

$$\min_{p \in [0,\infty)^{|S|}} \mathbb{E}\left[\sum_{j=1}^{|S|}(p_j - F(\tilde{y}_j))^2 \Big| y\right] = \sum_{j=1}^{|S|} \min_{p_j \in [0,\infty)} \mathbb{E}\left[(p_j - F(\tilde{y}_j))^2 \Big| y\right],$$

so that the estimator is $\hat{p}_j = \mathbb{E}[F(\tilde{y}_j)|y]$, which we will approximate using again a MC average:

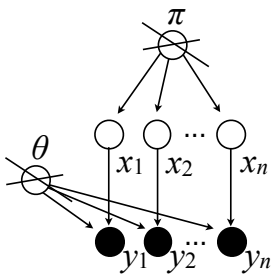
$$\begin{aligned}\hat{p}_j &= \mathbb{E}[F(\tilde{y}_j)|y] \\ &\approx \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[F(\tilde{y}_j)|x^{(t)}, y]. \end{aligned} \tag{3.1}$$

We will come back on how to compute $\mathbb{E}[F(\tilde{y}_j)|x^{(t)}, y]$ for each sample $x^{(t)}$ later on, but the important point here is that again, we need to simulate cluster membership variables x from the posterior distribution.

3.3 MCMC sampling on DPMs

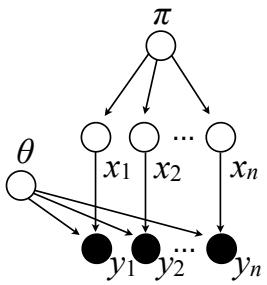
We will cover two sampling techniques, called collapsed [11] (or marginalized) and slice sampling [9]. The difference between the two lays in the variables that are explicitly represented in the state of the sampler.

In the collapsed sampler, only the cluster membership variables are stored as each Gibbs iteration:



Again, the crosses in this picture represents analytic marginalization (more on how this is done is coming soon). The shaded variables are observed.

In the slice sampler, we will represent (instantiate) more variables in the state of the sampler:



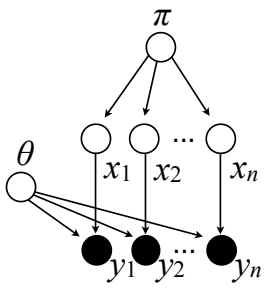
We will use auxiliary variables and lazy computation to avoid having to explicitly represent all of the infinite list of sticks and dishes.

This difference in representation has an impact on which blocks of variables can be sampled simultaneously. This in turn has an impact on both the performance of the sampler and what types of statistics can be computed from the posterior.

3.3.1 Joint distribution

We start by writing an expression for the joint distribution over a dish event $\mathrm{d}\theta$ (i.e. a measurable set of infinite list of parameters, $\mathrm{d}\theta \in \mathcal{F}_\Theta^\infty$), a stick event $\mathrm{d}\pi$ ($\mathrm{d}\pi \in \mathcal{F}_{[0,1]}^\infty$), observations $\mathrm{d}y$ ($\mathrm{d}y \in \mathcal{F}_\mathcal{X}^n$) and over an assignment $a \in \mathbb{N}^n$ of the cluster membership variables, $(x = a)$.

Finding the joint distribution is done by writing a product where each factor correspond to an edge in the joint graphical model:



This yields:

$$\mathbb{P}(\mathrm{d}\theta, \mathrm{d}\pi, x = a, \mathrm{d}y) = G_0^\infty(\mathrm{d}\theta)\,\mathrm{GEM}(\mathrm{d}\pi; \alpha_0)\mathbb{P}(x = a|\pi)\prod_{i=1}^{n} L(\mathrm{d}y_i|\theta_{a_i}).$$

In the collapsed sampler version, we will integrate over θ and π, while in the slice version, we will augment this representation to make it possible to represent the infinite lists explicitly.

3.3.2 Collapsed Gibbs sampler

The collapsed Gibbs sampler make two assumptions: that the prior over dishes, G_0 and the likelihood L are conjugate, and that the posterior of interest can be computed from samples from the cluster membership variables.

At each iteration, the collapsed sampler maintains values only for the cluster membership variables x, or more precisely, a labeled partition ρ over the datapoints, which, as will is see, is sufficient thanks to the results on the Chinese Restaurant Process representation of the part 2 of this set of notes. We will write $(\rho(x) = \rho)$ for the labeled partition induced by the cluster membership variables (overloading $\rho(\cdot)$ to denote also the function that extracts the labeled partition induced by the cluster membership variables).

Given a block $B \subset \{1, 2, \ldots, n\}$ of datapoints sharing a table, with corresponding observations $\mathrm{d}y_B = (\mathrm{d}y_i : i \in B)$, we introduce the following notation for the cluster marginal likelihood:

$$L(\mathrm{d}y_B) = \int \prod_{i \in B} L(\mathrm{d}y_i|\theta_c)G_0(\mathrm{d}\theta_c),$$

and its density $l(y_B) = \int \prod_{i \in B} l(y_i|\theta_c)G_0(\mathrm{d}\theta_c)$.

With this notation, and given that the observations $\mathrm{d}y_B$ of the customers sitting at table (or block) B share

a G_0-distributed cluster parameter θ, we get:

$$\begin{aligned}\mathbb{P}(x = a,\, \mathrm{d}y) &= \int\int \mathbb{P}(x = a|\pi) \left(\prod_{i=1}^{n} L(\,\mathrm{d}y_i|\theta_{a_i})\right) G_0^\infty(\,\mathrm{d}\theta)\,\mathrm{GEM}(\,\mathrm{d}\pi;\alpha_0) \\ &= \left(\int \mathbb{P}(x = a|\pi)\,\mathrm{GEM}(\,\mathrm{d}\pi;\alpha_0)\right)\left(\int \left(\prod_{i=1}^{n} L(\,\mathrm{d}y_i|\theta_{a_i})\right) G_0^\infty(\,\mathrm{d}\theta)\right) \\ &= \mathrm{CRP}(\rho(a);\alpha_0) \prod_{B\in\rho(a)} L(\,\mathrm{d}y_B)\end{aligned}$$

From this expression, it is apparent that $\rho(x)$ is sufficient.

In general, collapsed Gibbs samplers proceed by proposing local changes $\rho'_1, \ldots, \rho'_K$ to the current labeled partition ρ, and picks one of the neighbor or outcome ρ'_k proportionally to the density of the joint:

$$p_k = \mathrm{CRP}(\rho'_k;\alpha_0) \prod_{B\in\rho'_k} l(y_B). \tag{3.2}$$

In other words, after computing $p_1, \ldots, p_K$ (perhaps up to a proportionality constant), a new labeled partition is picked with probability

$$\mathbb{P}(\rho(x) = \rho'_k|y) = \frac{p_k}{\sum_{k'} p_{k'}}.$$

In the remaining of the section, we explain how these local changes are defined, and how to compute p_k efficiently.

In the standard Gibbs sampler, a local change to a seating arrangement (labeled partition) is simply obtained by taking one of the customer i out of the restaurant, and reinserting the customer at one of the tables (including the possibility of letting the customer start a new table). In principle, this could be done by computing the Formula (3.2) for each of the $t+1$ possible ways of reinserting the customer, where t is the number of tables occupied by the customers other than i. However this would involve $O(t^2)$ cluster marginal likelihood computations (there are $t+1$ of them for each of the $t+1$ possible ρ'_k), while we now show that it can be done using only $O(t)$ cluster marginal likelihood evaluations.

A more efficient method is to compute unnormalized outcome probability via the density of the predictive distribution $\mathbb{P}(\rho(x) = \rho'_k,\, \mathrm{d}y_i|x_{\setminus i}, y_{\setminus i})$:

$$q_k = \mathrm{CRP}(\rho'_k|\rho(x_{\setminus i}))l(y_i|y_{B_k}),$$

where B_k is the indices of the customers (other than i) that customer i would share a table with if outcome k is selected, and $l(y_i|y_B)$ is the density of the predictive cluster likelihood:

$$L(\,\mathrm{d}y_i|y_B) = \begin{cases} L(\,\mathrm{d}y_i) & \text{if } B = \emptyset \\ L(y_{B\cup\{i\}})/L(y_B) & \text{o.w.}\end{cases}$$

The intuition behind this computation is to consider the customer being resampled as the last one entering the restaurant.

Note that $p_k \propto q_k$, but computing q_k takes only $O(t)$ cluster marginal likelihood evaluations.

We now turn to the problem of computing cluster marginal and predictive likelihoods, where we will use the conjugacy assumption. Let $T \doteq \mathbb{R}^d$ be the sufficient statistic for θ, and $\xi \in \Xi$ denote the hyper-parameters for the dish prior density. Recall that conjugacy means that there is a deterministic function $f : \Xi\times\mathbb{R}^d \to \Xi$ such

that the density of the cluster posterior, $g_0(\theta|y;\xi)$ is equal to the density of the prior, but with transformed (updated) hyper-parameters:

$$g_0(\theta|y;\xi) = g_0(\theta; f(\xi, T(y))).$$

Next, let us fix an arbitrary θ^* and note that by the Bayes rule we have:

$$g_0(\theta^*|y;\xi) = \frac{l(y|\theta^*)g_0(\theta^*;\xi)}{l(y)}.$$

Combining these two equations together, we get that the density of the marginal likelihood can be computed via the identity:

$$l(y) = \frac{l(y|\theta^*)g_0(\theta^*;\xi)}{g_0(\theta^*; f(\xi, T(y)))}.$$

This gives all the ingredients needed to compute an estimate for the clustering application introduced earlier. For the density estimation problem, we still have to show that samples over the cluster indicators are enough to minimized the loss function at hand in Equation (3.1):

$$\begin{aligned}\mathbb{E}\left[F(\tilde{y})\middle|x^{(t)}, y\right] &= \int l(\tilde{y}|\underline{\theta})G(\,\mathrm{d}\underline{\theta}|x^{(t)}, y) \\ &= \frac{\alpha_0 l(\tilde{y})}{\alpha_0 + n} + \frac{1}{\alpha_0 + n}\sum_{B\in\rho(x^{(t)})} l(\tilde{y}|y_B).\end{aligned}$$

One final note is that adding a prior on α_0 and resampling its value is possible and an important thing to do in practice. See [1].

3.3.3 Slice sampler

The slice sampler proceeds by introducing auxiliary variables which makes it possible to avoid storing the infinite list of sticks and stick locations. This is done by a general computational trick called lazy computation.

Before going over the trick for posterior inference, let us go over an easier instance of lazy computation, for sampling from the prior in the Polya urn example of the previous set of notes.

We are going to see an alternative way of sampling a sequence of draws from the urn model. This is certainly not the simplest way to do so (using the predictive distribution is simpler, i.e. drawing balls and reinserting it plus one of the same color), but it is instructive to consider the following alternative "algorithm":

1. Sample a list of stick lengths, $\pi_1, \pi_2, \cdots \sim \mathrm{GEM}(R + B)$
2. Sample one stick 'color' for each stick, $\theta_1, \theta_2, \cdots \sim G_0 = \mathrm{Bin}(R/(R + B))$
3. To sample a ball, throw a dart on the stick and return the color of that stick segment:

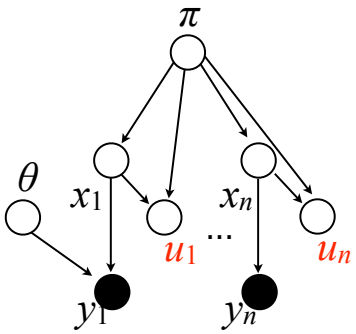
The problem with this "algorithm" is that the first step would never terminate: it requires infinite time to compute an infinite list of stick lengths.[4]

Fortunately it turns out the steps can be turned around to transform the procedure into a valid algorithm:

1. Throw the dart first, look at the position u of landing
2. Sample the minimum number of π_c's such that $\sum_{c'=1}^{c} \pi_c > u$ (finite with probability one)
3. Color the segments, return the color of the segment we landed on:

To get more samples, first throw the dart again, then check if more segments are needed (if the dart fall in an existing colored segment, return that color).

In order to apply a similar idea to posterior inference, we will have to first introduce some auxiliary variables u, more precisely one for each datapoint:



where $u_i | x_i, \pi \sim \text{Uni}(0, \pi_{x_i})$. This yields the following joint density:

$$\begin{aligned}\mathbb{P}(\,\mathrm{d}\theta,\,\mathrm{d}\pi, x = a,\,\mathrm{d}y,\,\mathrm{d}u) &= G_0^\infty(\,\mathrm{d}\theta)\,\text{GEM}(\,\mathrm{d}\pi;\alpha_0)\mathbb{P}(x = a|\pi)\prod_{i=1}^{n} L(\,\mathrm{d}y_i|\theta_{a_i})\,\text{Uni}(\,\mathrm{d}u_i;0,\pi_{x_i}).\\ &= G_0^\infty(\,\mathrm{d}\theta)\,\text{GEM}(\,\mathrm{d}\pi;\alpha_0)\prod_{i=1}^{n}\mathbf{1}[0 \le u_i \le \pi_{x_i}]\,\mathrm{d}u_i L(dy_i|\theta_{x_i}) \qquad (3.3)\end{aligned}$$

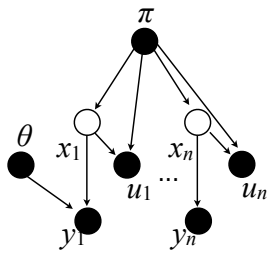
where we used $\mathbb{P}(x_i = c|\pi)\,\text{Uni}(\,\mathrm{d}u_i;0,\pi_{x_i}) = \mathbf{1}[0 \le u_i \le \pi_{x_i}]\,\mathrm{d}u_i$ by the definition of uniform distributions and $\mathbb{P}(x_i = c|\pi) = \pi_c$, also by definition.

The slice sampler proceeds by resampling the variables in three blocks: (1) all the dishes θ, (2) all the cluster membership variables, and (3), both all the stick lengths and all the auxiliary variables.

[4] In other words, naively such "algorithm" would need to be ran on a "Zeno machine," a hypothetical computer that could do a countable amount of operation in finite time.

Cluster membership variables

This move resamples the following variables:



This can be done by resampling each one independently (conditionally on the Markov blanket).

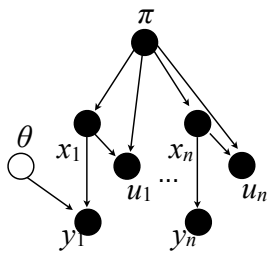
To find the conditional distribution of $x_i = c$, for each $c = 1, 2, \ldots$, we look at the factors in Equation (3.3) that depend on x_i, and obtain:

$$\mathbb{P}(x_i = c|\text{rest}) \propto \mathbf{1}[0 \leq u_i \leq \pi_c] L(\,\mathrm{d}y_i|\theta_c)$$

Thanks to lazy computation, we do not have to instantiate an infinite list of π_c in order to compute this. Instead, we find the smallest $N \in \mathbb{Z}^+$, such that $\sum_{c=1}^{N} \pi_c > 1 - u_i$. For $c > N$, $\sum_{c=1}^{N} \pi_c > 1 - u_i$, so $\mathbf{1}[0 \leq \mu_i \leq \pi_c] = 0$.

Dishes

This move resamples the following variables:



Again, this can be done by resampling each θ_c independently (conditionally on the Markov blanket).

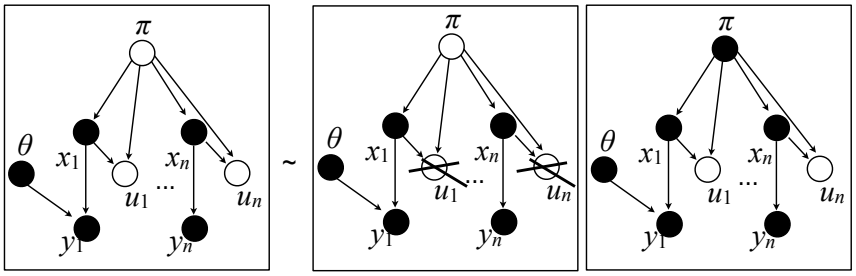
The posterior distribution of $\theta_c|rest$ is

$$\mathbb{P}(\,\mathrm{d}\theta_c|\text{rest}) \propto G_0(\,\mathrm{d}\theta_c) \prod_{i:x_i=c} L(\,\mathrm{d}y_i|\theta_c),$$

which can be resampled by Gibbs sampling in some conjugate models, and can be resampled using a Metropolis-Hastings step more generally.

Auxiliary variables and stick lengths

This resamples both all the auxiliary variables and all the stick lengths in one step. Using the chain rule, this step is subdivided in two substeps:



In other words:

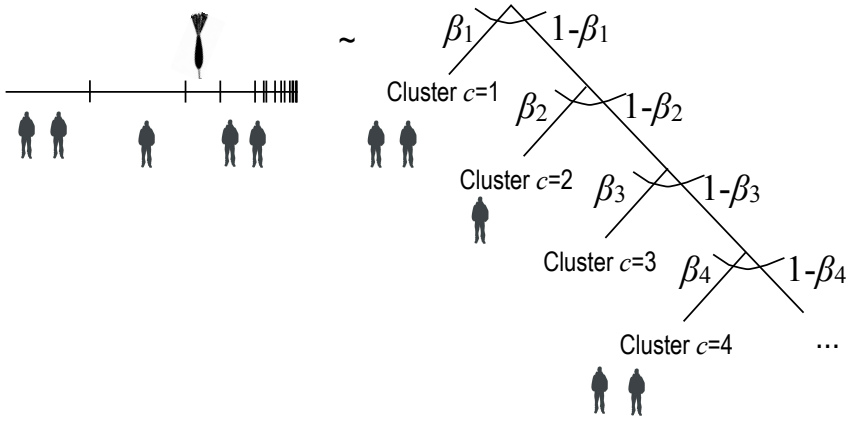
$$\mathbb{P}\left(\mathrm{d}\pi,\, \mathrm{d}u \middle| \text{rest}\right) = \mathbb{P}\left(\mathrm{d}\pi \middle|\ \text{rest except for } u\right) \mathbb{P}\left(\mathrm{d}u | \text{rest}\right)$$

The second factor can be sampled from readily:

$$\mathbb{P}(\mathrm{d}u_i | rest) = \mathrm{Uni}(\mathrm{d}u_i; 0, \pi_{x_i}).$$

To sample from the first factor, we look at a sequential scheme for sampling sticks that is an alternative to dart throwing. This sequential scheme makes it easier to see how to resample π while integrating out the auxiliary variables.

This alternative consists in visiting the sticks in order, and flipping a coin each time to determine if we are going to pick the current stick, or a stick of larger index:



Here the persons represent datapoints, and the left-hand-side represents a decision tree. Since the $\beta_c \sim \mathrm{Beta}(1, \alpha_0)$, and that each decision in the decision tree is multinomial, we get by multinomial-dirichlet conjugacy:

$$\mathbb{P}(\mathrm{d}\pi_c |\ \text{rest except for } u) = \mathrm{Beta}(\mathrm{d}\pi_i; a_c, b_c)$$

where:

$$a_c = 1 + \sum_{i=1}^{n} \mathbf{1}[x_i = c]$$
$$b_c = \alpha_0 + \sum_{i=1}^{m} \mathbf{1}[x_i > c]$$

3.3.4 Comparison

We conclude this section by summarizing the advantages and disadvantages of each methods:

	Collapsed sampler	Slice sampler
Pros	+ Easy to implement + Rao-Blackwellized	+ Flexible conditions on loss + Easy to parallelize
Cons	- Restrictions on the loss - Restr. on the likelihood - No easy parallelization	- Harder to implement - Aux. variables: less efficient - Memory needs

The restriction on the loss is that the expected loss needs to be computable from only samples from the cluster indicators. The restriction on the likelihood is the conjugacy assumption discussed in the section on collapsed sampler. Note that Rao-Blackwellization does not necessarily means that the collapsed sampler will be more efficiently, since each sampler resamples different blocks of variables with a different computation cost per sample.

The memory needs of the slice sampler can get large the case where the value of the auxiliary variables is low. Note that using a non-uniform distribution on the auxiliary variables could potentially be used to alleviate this problem.

Note also that for some other prior distributions (for example general stick-breaking distributions, which are covered in the next set of notes), only the slice sampler may be applicable. In other extensions of the DP, both slice and collapsed samplers are available.

3.3.5 Other posterior inference methods

Other techniques have been proposed to approximate the posterior of DPMs.

One approach is to use the collapsed sampler with different proposal distributions, to prevent the sampler from getting stuck in local optima (note for example that splitting a cluster into two takes a long time if we move one customer at the time). See for example [8] and [10].

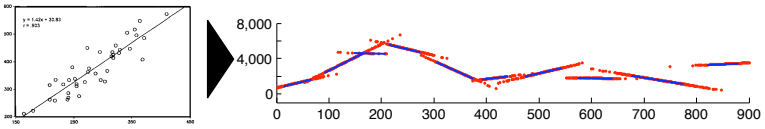
Another approach is to use the stick breaking representation while ignoring the small sticks [7, 6] (i.e. setting $\beta_N = 1$ for some large enough N). This technique is called *truncation*. Using the Levy construction that we will cover in the next set of notes, the error of this type of approximation can be bounded. A diagnostic is also available to set the value of the truncation automatically.

Finally, one can use variational techniques instead of MCMC, see for example [2], which is based on the representation of [7].

3.4 DP for GLMs

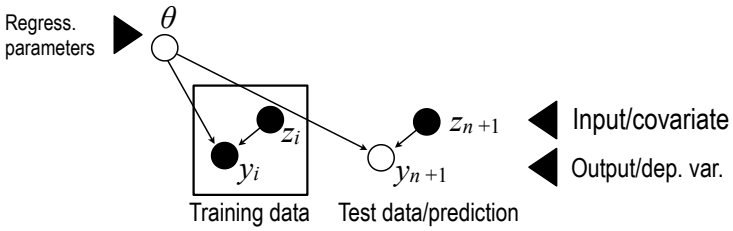
We now outline an application DPMs to regression and clustering. This application was described in [5].

The goal is to transform a globally (generalized) linear model into a locally (generalized) linear model, within a Bayesian framework. For example instead of getting a fit as shown on the left, we would like a collection of locally linear fits:



We start by reviewing standard Bayesian regression (for more detailed introduction see [3] or [12]), then introduce the DPM approach.

A basic Bayesian linear regression model has the following form:



where z_i is a D-dimensional vector of input/covariates, θ is a D-dimensional parameter vector and y_i is a 1-dimensional response say. Let Z denote the n by D data matrix, and Y, the n by 1 training responses. We put the following distributions on these variables:

$$\theta^{(d)} \sim N\left(0, \frac{1}{\tau_2}\right), \; d = 1, \ldots, D.$$
$$y_i|\theta, z_i \sim N\left(\langle \theta, z_i \rangle, \frac{1}{\tau}\right).$$

where τ_1 is a noise precision parameter, and τ_2 is an isotropic parameter regularization.[5]

By conjugacy, we get the following posterior on the parameters:

$$\theta|y_{1:n}, z_{1:n} \sim N(M_n, S_n)$$

where

$$S_n = (\tau^2 + \tau Z^{\mathrm{T}} Z)^{-1}$$
$$M_n = \tau S_n Z^{\mathrm{T}} Y.$$

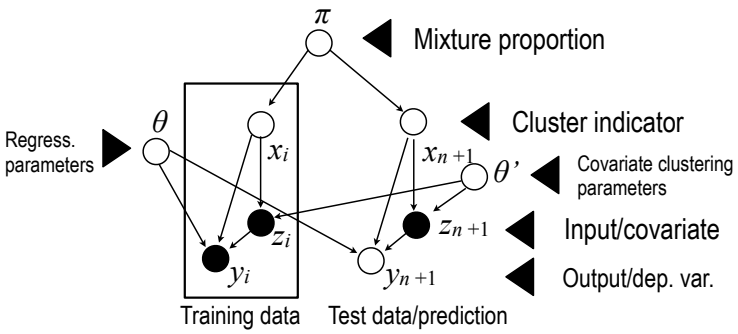
[5]Note that this model does not depend on the prior on the covariates. This has motivated G-priors [12] with parameters that depend on X, which allow putting a prior over the precision parameters considered fixed in the model above. On the other hand, in the DPM extension that will follow shortly, we will need to put a distribution on the input variable, since new datapoints will be assigned to a cluster using this distribution, which will then allow using the most appropriate set of regression parameters with higher probability.

Given a new covariate z_{n+1}, the predictive distribution over y_{n+1} is then:

$$y_{n+1}|z_{n+1}, y_{1:n}, z_{1:n} \sim N({M_n}^{\mathrm{T}} z_{n+1}, \sigma_n^2(z_{n+1}))$$

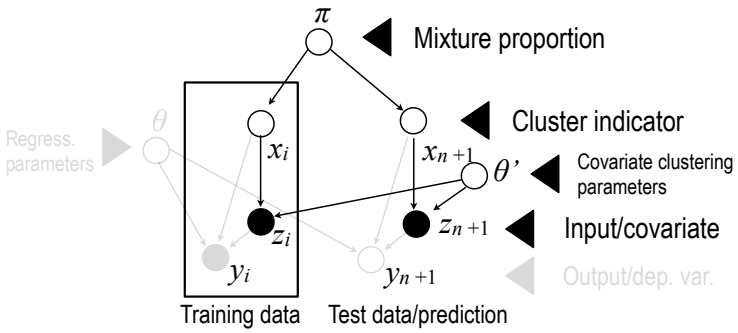
where $\sigma_n^2(z) = \frac{1}{\tau} + z^{\mathrm{T}} S_n z$.

We now turn to the non parametric version of this model, which has the following graphical model:



where $\theta_c = (\theta_c^{(1)}, \ldots, \theta_c^{(D)})$ are vectors of regression parameters, and $\theta'_c = (\theta_c'^{(1)}, \ldots, \theta_c'^{(D')})$ are vectors of clustering parameters.

This model seems complicated at first glance, but note that a standard DPM model on the covariates appears as a submodel:



The rest of the model is the same as the standard (parametric) Bayesian regression model, but with the set of parameter determined by the cluster indicator.

The intuitive idea is that given a new datapoint, the prior over the z's enable us to get a posterior over which cluster it belongs to. For each cluster, we have a standard Bayesian linear regression model.

Formally, we define the distributions as follows:

$$\begin{aligned}
\pi &\sim \text{GEM}(\alpha_0) \\
x_i|\pi &\sim \text{Mult}(\pi) \\
\theta_c^{'(d)} &\sim N\left(0, \frac{1}{\tau_3}\right), d = 1, \ldots, D; c = 1, 2, \ldots \\
\theta_c^{(e)} &\sim N\left(0, \frac{1}{\tau_2}\right) \; e = 1, \ldots, D' \\
z_i|\theta, x_i &\sim N\left(\theta'_{x_i}, \frac{1}{\tau_4}\right), \; i = 1, \ldots, n+1 \\
y_i|z_i, x_i, \theta &\sim N\left(\langle \theta_{x_i}, z_i \rangle, \frac{1}{\tau}\right),
\end{aligned}$$

where τ_3 acts as a regularization on the clustering parameter, and τ_4 as a noise precision parameter on the input variables.

Note that simulating the posterior of the cluster variables can be done using collapsed sampling by conjugacy. Given such samples $x_{1:n_1}^{(s)}$, the regression estimator under L^2 loss on y takes the form:

$$\begin{aligned}
\mathbb{E}(y_{n+1}|D) &= E[E[y_{n+1}|D, x_{1:(n+1)}]] \\
&\approx \frac{1}{S}\sum_{s=1}^{S} \mathbb{E}[y_{n+1}|D, x_{1:(n+1)}^{(s)}] \\
&= \frac{1}{S}\sum_{s=1}^{S} \left(M_n(x_{1:(n+1)}^{(s)})\right)^{\mathrm{T}} z_{n+1},
\end{aligned}$$

where D denotes the training data (inputs and outputs) as well as the new input x_{n+1}. The posterior mean takes a form similar to the parametric case, but defined on the subset of datapoints in the same cluster as the new data point in the current sample:

$$\begin{aligned}
S_n(x_{1:(n+1)}) &= (\tau^2 + \tau Z(x_{1:(n+1)})^{\mathrm{T}} Z(x_{1:(n+1)}))^{-1} \\
M_n(x_{1:(n+1)}) &= \tau S_n(x_{1:(n+1)}) Z(x_{1:(n+1)})^{\mathrm{T}} Y.
\end{aligned}$$

where

$$Z(x_{1:n}) = \begin{bmatrix} - \; z_{i_1} - \\ \vdots \\ - \; z_{i_k} - \end{bmatrix}$$

and $(i_1, \ldots, i_k)$ are the indices of the data matrix rows in the same cluster as x_{n+1}.

See [5] for a generalization of this idea to other GLMs, including an application to classification.

References

[1] Author. Hyperparameter estimation in Dirichlet process mixture models. Technical report, Duke University, 1995.

[2] D. Blei and M. I. Jordan. Variational methods for the Dirichlet process. *ICML*, 2004.

[3] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis*. CHAPMAN&HALL/CRC, 2004.

[4] Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2009.

[5] Lauren A. Hannah, David M. Blei, and Warren B. Powell. Dirichlet process mixtures of generalized linear models. *Journal of Machine Learning Research*, 2010.

[6] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *JASA*, 2001.

[7] H. Ishwaran and M. Zarepour. Markov chain Monte Carlo in approximate Dirichlet and beta two-parameter process hierarchical models. *Biometrika*, 2000.

[8] S. Jain and R. Neal. A split-merge markov chain Monte Carlo procedure for the Dirichlet process mixture model. Technical report, University of Toronto, 2000.

[9] Maria Kalli, Jim E. Griffin, and Stephen G. Walker. Slice sampling mixture models. *Statistics and Computing*, 2011.

[10] P. Liang, M. I. Jordan, and B. Taskar. A permutation-augmented sampler for Dirichlet process mixture models. In *International Conference on Machine Learning (ICML)*, 2007.

[11] R. Neal. Markov chain sampling methods for dirichlet process mixture models. Technical report, U of T, 2000.

[12] C. Robert. *The Bayesian Choice*. Springer, 2007.