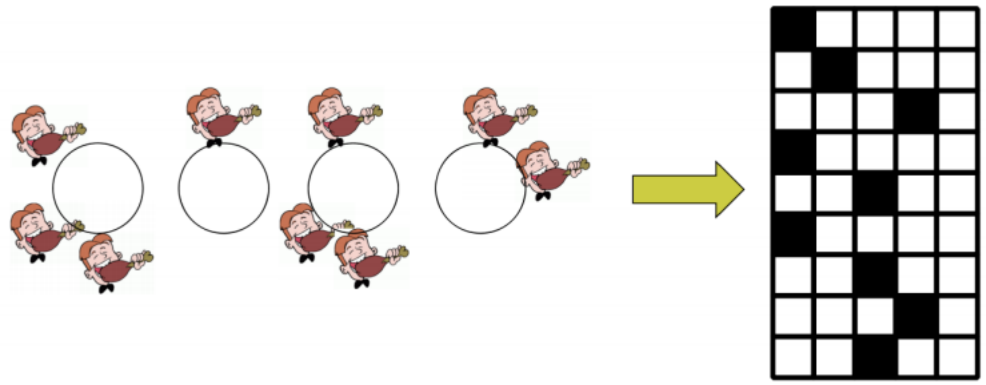
Statistical Machine Learning

Lecture 08
Dirichlet Process (DP)
Chinese Restaurant Process (CRP)
Indian Buffet Process (IBP)

Spring 2021

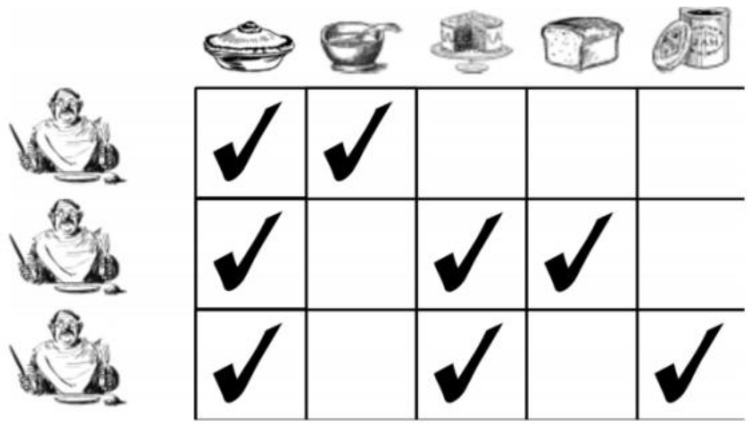
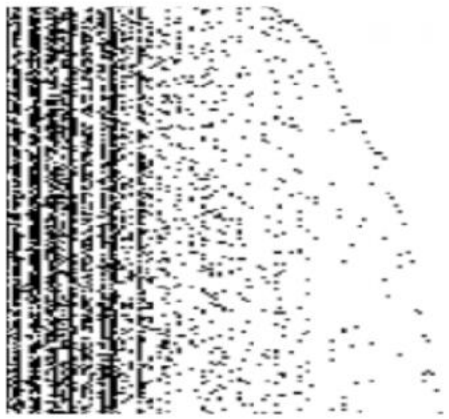
Sharif University of Technology

Recall CRP



- Rows are data points
- Columns are clusters
- Rows add up to 1
- Each Data belongs to only 1 cluster

In Summary: Indian Buffet Process

- Rows are data points
- Columns are clusters
- Rows may add up to more than 1
- Each Data may belongs to more than 1 cluster

Solution: Finite to infinite binary matrices

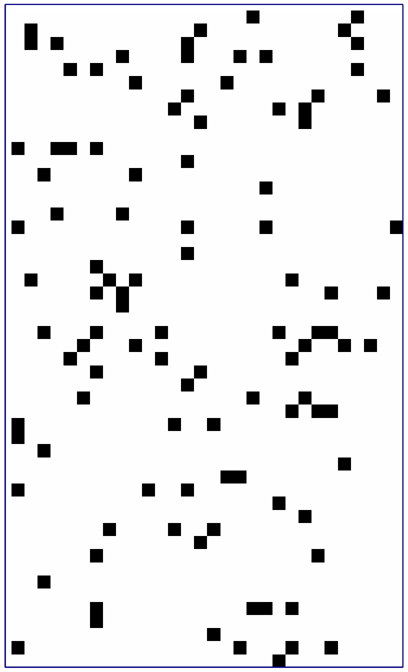
Assume:

$z_{nk} = 1$ means object n has feature k

$z_{nk} \sim \text{Bernoulli}(\theta_k)$

$\theta_k \sim \text{Beta}(\alpha/K, 1)$

- Note that $P(z_{nk} = 1 | \alpha) = E(\theta_k) = \frac{\alpha/K}{\alpha/K+1}$ and as K grows larger the matrix gets sparser.
- If $\mathbf{Z}$ is $N \times K$, the expected number of nonzero entries is $N\alpha/(1 + \alpha/K) < N\alpha$.
- Even in the $K \to \infty$ limit, the matrix is expected to have a finite number of non-zero entries.



Solution: Finite to infinite binary matrices

If we integrate out θ:

$$P(\mathbf{Z}|\alpha) = \int P(\mathbf{Z}|\theta)P(\theta|\alpha)d\theta$$
$$= \prod_k \frac{\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k + 1)}{\Gamma(\frac{\alpha}{K})} \frac{\Gamma(1 + \frac{\alpha}{K})}{\Gamma(N + 1 + \frac{\alpha}{K})}$$

The conditional feature assignments are:

$$P(z_{nk} = 1|\mathbf{z}_{-n,k}) = \int_0^1 P(z_{nk}|\theta_k)p(\theta_k|\mathbf{z}_{-n,k})\ d\theta_k$$
$$= \frac{m_{-n,k} + \frac{\alpha}{K}}{N + \frac{\alpha}{K}}$$

where $\mathbf{z}_{-n,k}$ is the set of assignments of all objects, not including n, for feature k, and $m_{-n,k}$ is the number of objects having feature k, not including n. We can take limit as $K \rightarrow \infty$.

Solution: Finite to infinite binary matrices

Problem: the probability for any particular matrix goes to zero as $K \to \infty$:

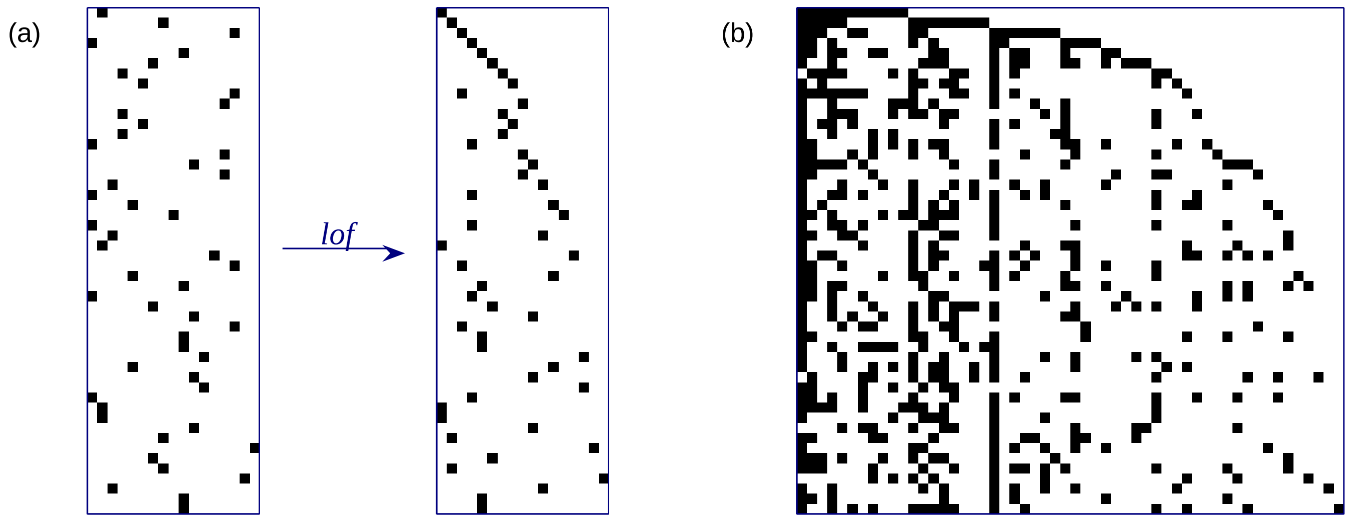
$$\lim_{K\to\infty} P(\mathbf{Z}|\alpha) = 0$$

However, if we consider equivalence classes of matrices in left-ordered form (lof) obtained by reordering the columns: $[\mathbf{Z}] = lof(\mathbf{Z})$:

$$\lim_{K\to\infty} P([\mathbf{Z}]|\alpha) = \exp\{-\alpha H_N\}\frac{\alpha^{K_+}}{\prod_{h>0} K_h!}\prod_{k\le K_+}\frac{(N-m_k)!(m_k-1)!}{N!}$$

- K_+ is the number of features assigned (i.e. non-zero columns).
- H_N is the Nth harmonic number.
- K_h are the number of features with history h.
- This distribution is **infinitely exchangeable**, i.e. it is not affected by the ordering on objects. This is important for its use as a prior in settings where the objects have no natural ordering.

Binary matrices in left-ordered form



(a) The matrix on the left is transformed into the matrix on the right by the function *lof* (). The resulting left-ordered matrix was generated from a Chinese restaurant process (CRP) with $\alpha = 10$.

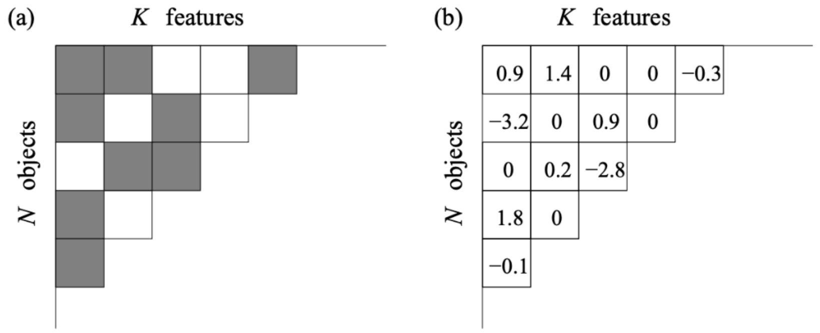
(b) A left-ordered feature matrix. This matrix was generated from the prior on infinite binary matrices with $\alpha = 10$.

From binary to non-binary latent features

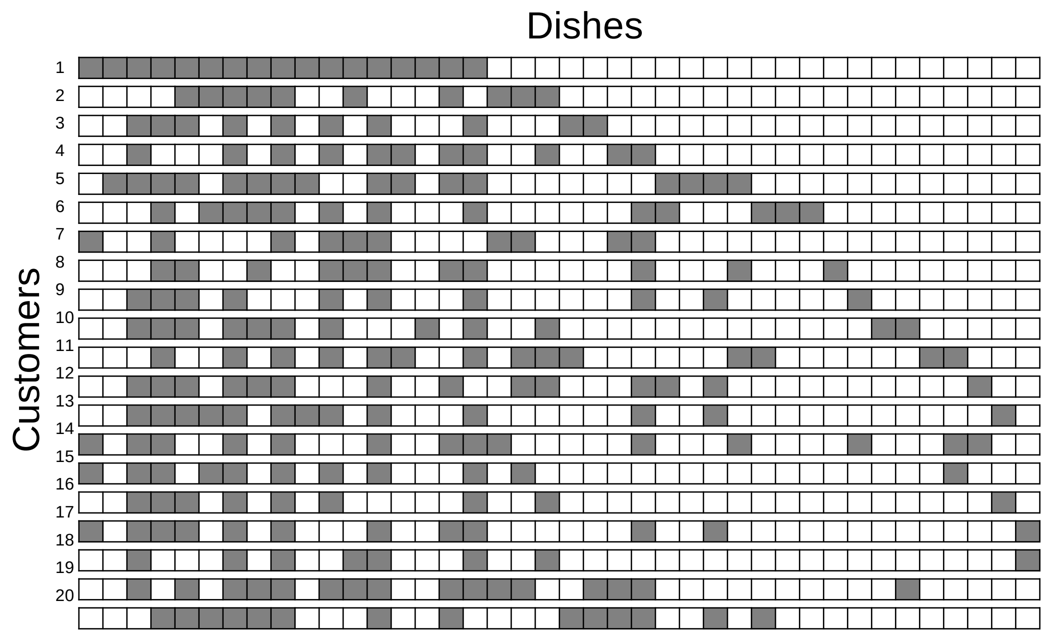
- In many models we might want non-binary latent features.
- A simple way to generate non-binary latent feature matrices from **Z**

$$\mathbf{F} = \mathbf{Z} \otimes \mathbf{V}$$

- Where $\otimes$ is the elementwise (Hadamard) product of two matrices, and V is a matrix of independent random variables (e.g. Gaussian, Poisson, ...).

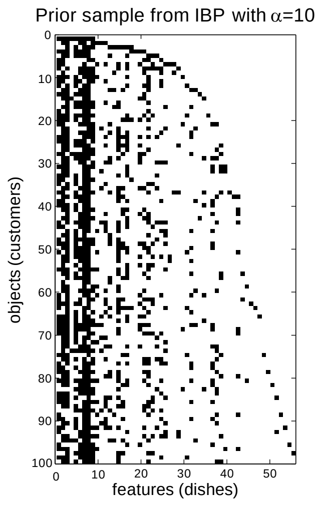


The Indian buffet process (IBP)

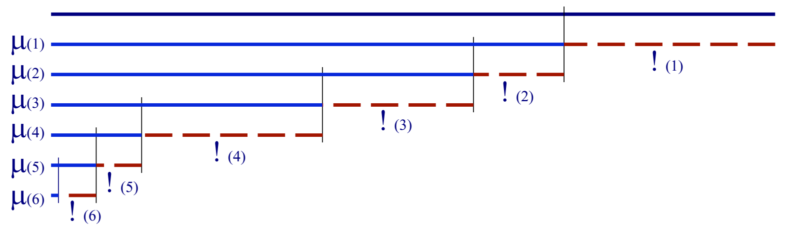


- First customer starts at the left of the buffet, and takes a serving from each dish, stopping after a Poisson(α) number of dishes as her plate becomes overburdened.
- The *n*th customer moves along the buffet, sampling dishes in proportion to their popularity, serving himself with probability m_k/n, and trying a Poisson(α/n) number of new dishes.
- The customer-dish matrix is our feature matrix, **Z**.

Properties of the Indian buffet process



$$P([\mathbf{Z}]|\alpha) = \exp\{-\alpha H_N\} \frac{\alpha^{K_+}}{\prod_{h>0} K_h!} \prod_{k \le K_+} \frac{(N-m_k)!(m_k-1)!}{N!}$$



Stick-breaking construction for the DP and IBP. The black stick at top has length 1. At each iteration the vertical black line represents the break point. The brown dotted stick on the right is the weight obtained for the DP, while the blue stick on the left is the weight obtained for the IBP.

Shown in (Griffiths and Ghahramani, 2005):

- It is infinitely exchangeable.
- The number of ones in each row is Poisson(α)
- The expected total number of ones is αN .
- The number of nonzero columns grows as $O(\alpha \log N)$.

Additional properties:

- Has a stick-breaking representation (Teh, 2007)

Markov chain Monte Carlo (MCMC)

Motivation

- Consider the Bayes formula for obtaining the posterior density:

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$$

- To compute the posterior we multiply the prior P(θ) (what we think about θ before we have seen any data) and the likelihood P(x|θ), how we think our data is distributed. This nominator is easy to solve for.
- The denominator P(x) (the evidence that the data x was generated by this model), is computed by integrating over all possible parameter values:

$$P(x) = \int_{\Theta} P(x, \theta)\, \mathrm{d}\theta$$

Motivation

- For even slightly non-trivial models you just can't compute the posterior in a closed-form way.

- Question: could we try to approximate it? For example, if we could somehow draw samples from posterior we can Monte Carlo approximate it.

- There exist a general class of algorithms that do this; called Markov chain Monte Carlo (MCMC) (constructing a Markov chain to do Monte Carlo approximation).

Aside: Monte Carlo Methods

[Basic Integration]

- We want to compute the integral,

$$I = \int h(x) f(x) dx$$

where f(x) is a probability density function.

- In other words, we want $E_f[h(x)]$.
- We can approximate this as:

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} h(X_i)$$

where $X_1, X_2, \ldots, X_N$ are sampled from f.

- By the law of large numbers,

$$\hat{I} \xrightarrow{p} I$$

Markov chain Monte Carlo (MCMC)

- Markov chain Monte Carlo (MCMC) methods consist of a class of algorithms for sampling from a probability distribution; by constructing a Markov chain that has the desired distribution as its equilibrium distribution.
- We can obtain a sample of the desired distribution by recording states from the chain; the more steps that are included, the more closely the distribution of the sample matches the actual desired distribution.
- There exist a number of algorithms for constructing the desired chains.

Metropolis–Hastings algorithm

- The Metropolis–Hastings algorithm is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution when direct sampling is difficult.

- It is generally used for sampling from multi-dimensional distributions, especially when the number of dimensions is high.

- The Metropolis–Hastings algorithm can draw samples from any probability distribution p(x) provided that we know a function f(x) proportional to the density of P(x) and the values of f(x) can be calculated.

Metropolis–Hastings algorithm

- The requirement that f(x) must only be proportional to the density, rather than exactly equal to it, makes the Metropolis–Hastings algorithm useful, because calculating the necessary normalization factor is often extremely difficult in practice.

- The Metropolis–Hastings algorithm works by generating a sequence of sample values in such a way that, as more and more sample values are produced, the distribution of values more closely approximates the desired distribution p(x).

- These sample values are produced iteratively, with the distribution of the next sample being dependent only on the current sample value (thus making the sequence of samples into a Markov chain).

Metropolis–Hastings algorithm

- Then, with some probability, the candidate is either accepted (in which case the candidate value is used in the next iteration) or rejected (in which case the candidate value is discarded, and current value is reused in the next iteration).

- More formally, the Metropolis–Hastings algorithm involves designing a Markov process (by constructing transition probabilities) that fulfills:
  - Existence of stationary distribution $\pi(x)$ (A sufficient but not necessary condition is detailed balance, which requires that each transition $x \rightarrow x'$ is reversible)
  - Uniqueness of stationary distribution $\pi(x)$ (every state must be aperiodic and positive recurrent)

  such that its stationary distribution $\pi(x)$ chosen to be $p(x)$.

Metropolis–Hastings algorithm

- The derivation of the algorithm starts with the condition of detailed balance:

$$P(x' \mid x)P(x) = P(x \mid x')P(x')$$

- The approach is to separate the transition in two sub-steps; the proposal and the acceptance-rejection.

- The proposal distribution g(x'|x), and the acceptance distribution A(x',x) is the probability to accept the proposed state x'.

- The transition probability can be written as:

$$P(x' \mid x) = g(x' \mid x)A(x', x)$$

Metropolis–Hastings algorithm

1. Initialise
    1. Pick an initial state x_0.
    2. Set $t = 0$.
2. Iterate
    1. *Generate* a random candidate state x' according to $g(x' \mid x_t)$.
    2. *Calculate* the acceptance probability $A(x', x_t) = \min\left(1, \frac{P(x')}{P(x_t)} \frac{g(x_t \mid x')}{g(x' \mid x_t)}\right)$;.
    3. *Accept or reject*:
        1. generate a uniform random number $u \in [0, 1]$;
        2. if $u \leq A(x', x_t)$, then *accept* the new state and set $x_{t+1} = x'$;
        3. if $u > A(x', x_t)$, then *reject* the new state, and copy the old state forward $x_{t+1} = x_t$.
    4. *Increment*: set $t = t + 1$.

Markov chain Monte Carlo (MCMC)

- Alternative to MH algorithm:
  - Gibbs sampling: This method requires all the conditional distributions of the target distribution to be sampled exactly. Gibbs sampling is popular partly because it does not require any tuning.
  - Pseudo-marginal Metropolis–Hastings: This method replaces the evaluation of the density of the target distribution with an unbiased estimate and is useful when the target density is not available analytically (latent variable models).
  - Slice sampling: This method depends on the principle that one can sample from a distribution by sampling uniformly from the region under the plot of its density function. It alternates uniform sampling in the vertical direction with uniform sampling from the horizontal slice defined by the current vertical position.

Important: Modelling The Data

Latent variable model: let **X** be the $N \times D$ matrix of observed data, and **Z** be the $N \times K$ matrix of binary latent features

$$P(\mathbf{X}, \mathbf{Z}|\alpha) = P(\mathbf{X}|\mathbf{Z})P(\mathbf{Z}|\alpha)$$

By combining the IBP with different likelihood functions we can get different kinds of models:

- Models for graph structures
- Models for protein complexes
- Models for overlapping clusters
- Models for choice behaviour
- Models for users in collaborative filtering
- Sparse latent factor models

Summary: Posterior Inference in IBPs

$$P(Z, \alpha | X) \propto P(X|Z) P(Z|\alpha) P(\alpha)$$

Gibbs sampling: $P(z_{nk} = 1 | \mathbf{Z}_{-(nk)}, \mathbf{X}, \alpha) \propto P(z_{nk} = 1 | \mathbf{Z}_{-(nk)}, \alpha) P(\mathbf{X}|\mathbf{Z})$

- If $m_{-n,k} > 0$, $P(z_{nk} = 1 | \mathbf{z}_{-n,k}) = \frac{m_{-n,k}}{N}$
- For infinitely many k such that $m_{-n,k} = 0$: Metropolis steps with truncation* to sample from the number of new features for each object.
- If α has a Gamma prior then the posterior is also Gamma → Gibbs sample.

Conjugate sampler: assumes that $P(\mathbf{X}|\mathbf{Z})$ can be computed.

Non-conjugate sampler: $P(\mathbf{X}|\mathbf{Z}) = \int P(\mathbf{X}|\mathbf{Z}, \theta) P(\theta) d\theta$ cannot be computed, requires sampling latent θ as well (c.f. (Neal 2000) non-conjugate DPM samplers).

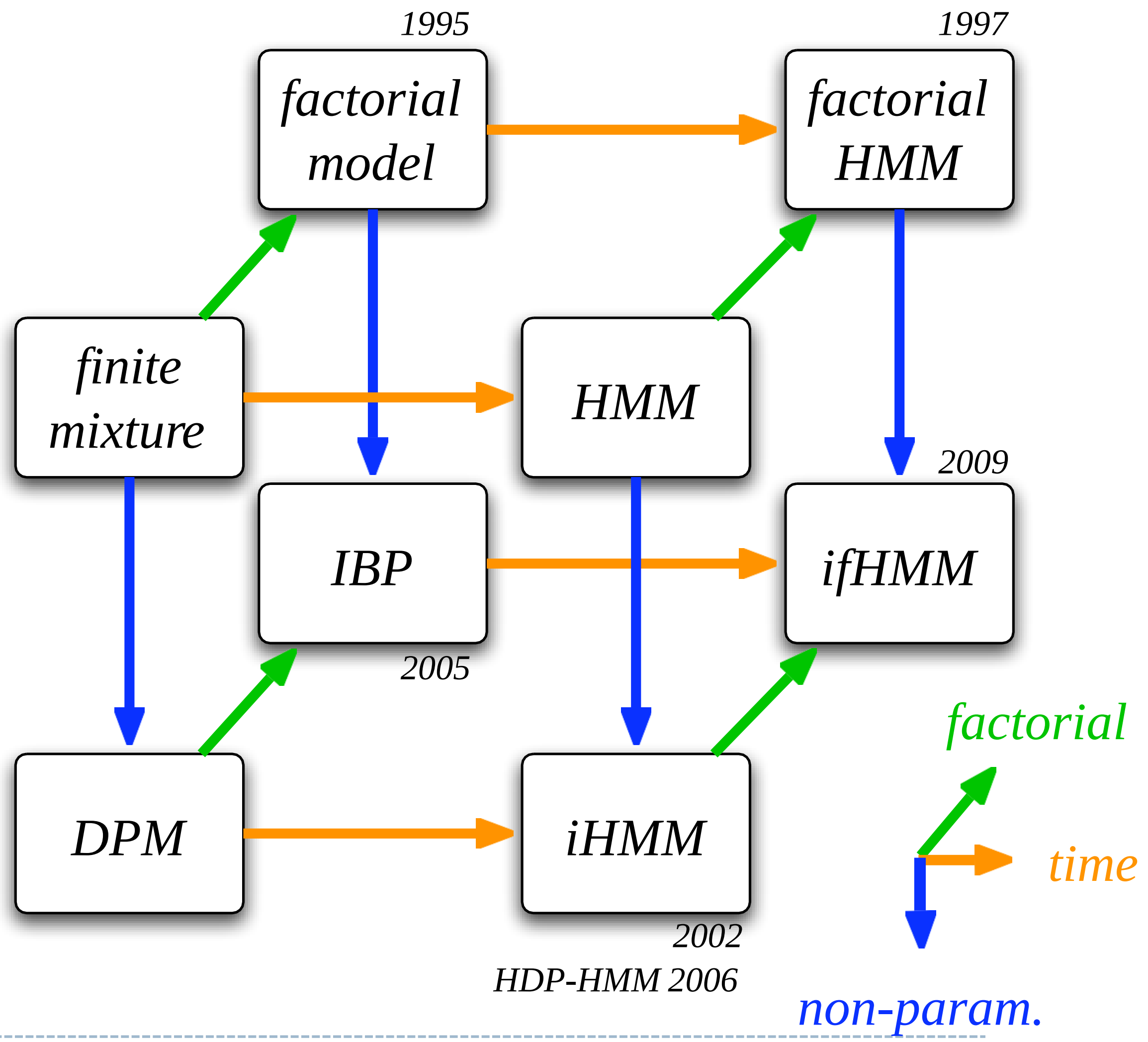
***Slice sampler:** non-conjugate case, is not approximate, and has an adaptive truncation level using a stick-breaking construction of the IBP (Teh, et al, 2007).

Particle Filter: (Wood & Griffiths, 2007).

Accelerated Gibbs Sampling: maintaining a probability distribution over some of the variables (Doshi-Velez & Ghahramani, 2009).

Variational inference: (Doshi-Velez, Miller, van Gael, & Teh, 2009).

Summary


1995
factorial model
1997
factorial HMM
finite mixture
HMM
2009
IBP
ifHMM
2005
DPM
iHMM
2002
HDP-HMM 2006
factorial
time
non-param.

Case Study:
Dirichlet Process (DP)

Case Study: Samples from a Dirichlet Process

$$X_n | X_1, \ldots, X_{n-1} = \begin{cases} X_i & \text{with probability } \frac{1}{n-1+\alpha} \\ \text{new draw from } G_0 & \text{with probability } \frac{\alpha}{n-1+\alpha} \end{cases}$$

Let there be *K* unique values for the variables:

$$X_k^* \text{ for } k \in \{1, \ldots, K\}$$

Can rewrite as:

$$X_n | X_1, \ldots, X_{n-1} = \begin{cases} X_k^* & \text{with probability } \frac{\text{num}_{n-1}(X_k^*)}{n-1+\alpha} \\ \text{new draw from } G_0 & \text{with probability } \frac{\alpha}{n-1+\alpha} \end{cases}$$

Case Study: The Dirichlet Process (DP) Simplified

- DP is a very useful and simple tool in Bayesian nonparametric statistics.

- DP is commonly used in situations where we assume there is clustering among random variables, but: 1. We do not know how many clusters there are, and 2. Which random variables belong to which cluster.

- We treat the assignment of the random variables to clusters as a random variable itself which can be estimated from the data (or integrated out).

The Dirichlet Process (DP) Simplified

- We assign elements to categories following a very simple rule when assigning the nth element:
  - We assign it to a new category with the probability $\frac{\alpha}{n-1+\alpha}$
  - We assign it to an already existing category X_k^* with probability $\frac{\mathrm{num}_{n-1}(X_k^*)}{n-1+\alpha}$

Where $\mathrm{num}_{n-1}(X_k^*)$ is the number of random variables already assigned to category X_k^* .

α is the concentration parameter.

- Repeat this rule for all elements, to group them in the categories and clusters.

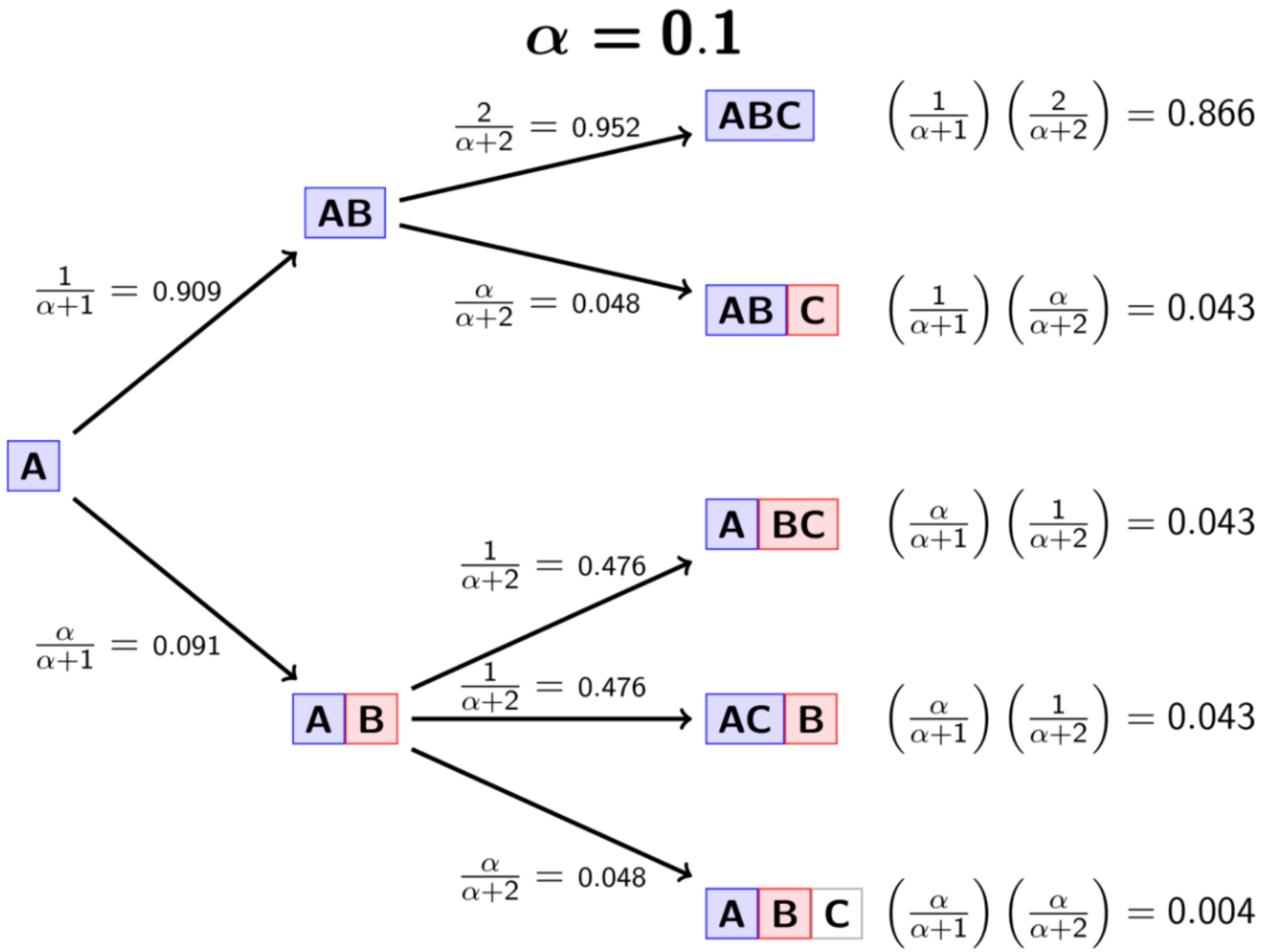
Case Study: The Dirichlet Process (DP) Simplified

- Assume we want to assign three elements, A, B, and C, to an unknown number of categories according to the Dirichlet process.
- Here, there could be 1, 2, or 3 categories, and 5 possible assignments(partitions): ABC, A|BC, B|AC, C|AB, and A|B|C.
- We start with element A, for which our only option is to assign it to its own category, let say the blue category. This happens with a probability of 1.
- For element B. We have two options:
  - We can assign B to the same blue category as A with probability 1/(α+1) (the numerator 1 is simply the number of things in the "blue" category)
  - we can assign it to a new red category with probability α/(α+1)

Case Study: The Dirichlet Process (DP) Simplified

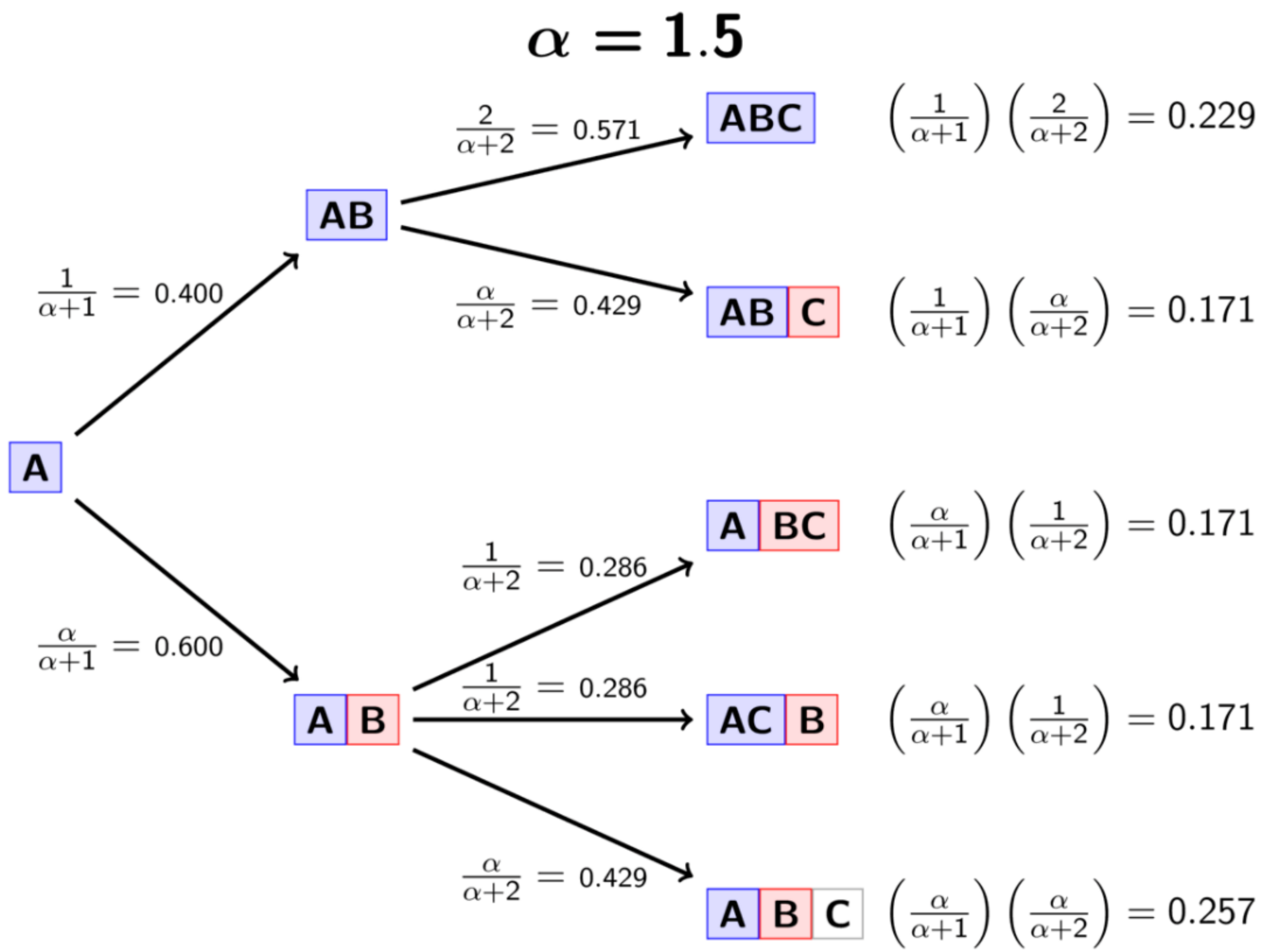
- Assume B gets assigned to the blue category, along with A.
- Finally, we need to assign C:
  - We can assign it to either a new category with probability $\alpha/(\alpha+2)$
  - We can assign C to the same blue category as A and B with probability $2/(\alpha+2)$ (the numerator is now 2, because there are two elements in blue)
- If we had more elements, we would continue this logic until they were all assigned to categories.
- We can draw all possible assignments as a probability tree, which allows us to see how rule of the Dirichlet process determines the probability of all possible assignments.

Case Study: The Dirichlet Process (DP) Simplified



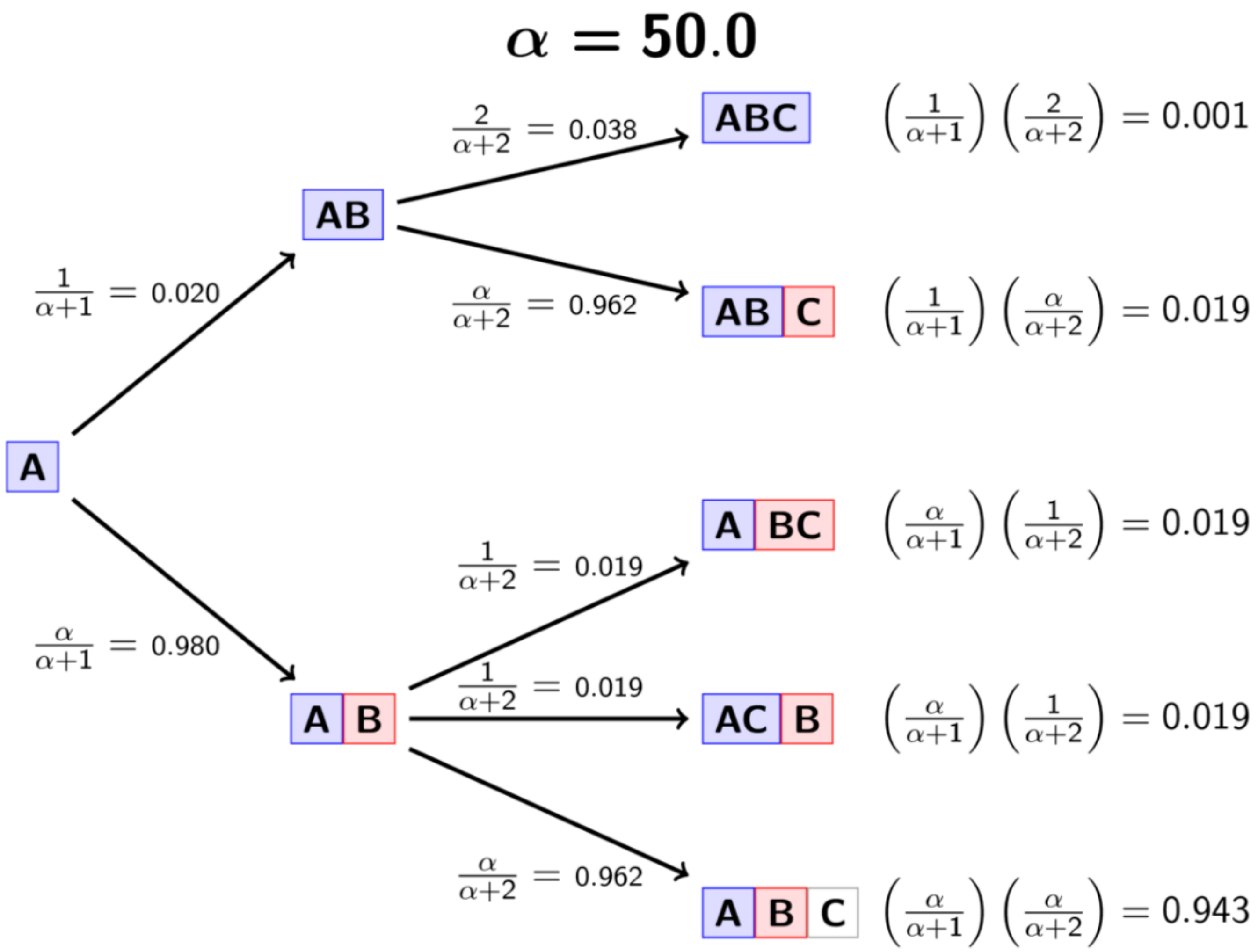
$\alpha = 0.1$
A
$\frac{1}{\alpha+1} = 0.909$
$\frac{\alpha}{\alpha+1} = 0.091$
AB
A | B
$\frac{2}{\alpha+2} = 0.952$
$\frac{\alpha}{\alpha+2} = 0.048$
$\frac{1}{\alpha+2} = 0.476$
$\frac{1}{\alpha+2} = 0.476$
$\frac{\alpha}{\alpha+2} = 0.048$
ABC
AB | C
A | BC
AC | B
A | B | C
$\left(\frac{1}{\alpha+1}\right)\left(\frac{2}{\alpha+2}\right) = 0.866$
$\left(\frac{1}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.043$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.043$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.043$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.004$

Case Study: The Dirichlet Process (DP) Simplified



$\alpha = 1.5$
A
$\frac{1}{\alpha+1} = 0.400$
$\frac{\alpha}{\alpha+1} = 0.600$
AB
A | B
$\frac{2}{\alpha+2} = 0.571$
$\frac{\alpha}{\alpha+2} = 0.429$
$\frac{1}{\alpha+2} = 0.286$
$\frac{1}{\alpha+2} = 0.286$
$\frac{\alpha}{\alpha+2} = 0.429$
ABC
AB | C
A | BC
AC | B
A | B | C
$\left(\frac{1}{\alpha+1}\right)\left(\frac{2}{\alpha+2}\right) = 0.229$
$\left(\frac{1}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.171$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.171$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.171$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.257$

Case Study: The Dirichlet Process (DP) Simplified

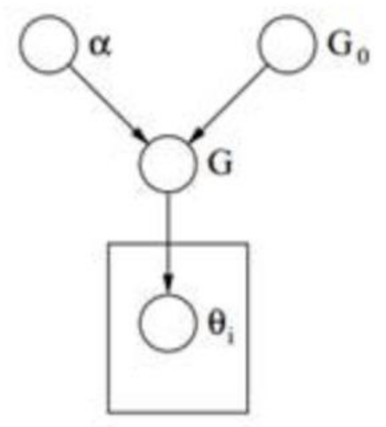

$\alpha = 50.0$
A
AB
A B
ABC
AB C
A BC
AC B
A B C
$\frac{1}{\alpha+1} = 0.020$
$\frac{\alpha}{\alpha+1} = 0.980$
$\frac{2}{\alpha+2} = 0.038$
$\frac{\alpha}{\alpha+2} = 0.962$
$\frac{1}{\alpha+2} = 0.019$
$\frac{1}{\alpha+2} = 0.019$
$\frac{\alpha}{\alpha+2} = 0.962$
$\left(\frac{1}{\alpha+1}\right)\left(\frac{2}{\alpha+2}\right) = 0.001$
$\left(\frac{1}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.019$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.019$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{1}{\alpha+2}\right) = 0.019$
$\left(\frac{\alpha}{\alpha+1}\right)\left(\frac{\alpha}{\alpha+2}\right) = 0.943$

Case Study: The Dirichlet Process (DP) Simplified

- Notice that as the concentration parameter increases:
  - the probability of more clustered assignments (i.e., fewer categories) decreases.
  - while the probability of less clustered assignments (i.e., more categories) increases.
- Remember that the concentration parameter is in the numerator for the probability of assigning an element to a new category, you can conclude that as α increases, you'll tend to get more categories.
- By studying this example, you might notice that the elements, despite not being independent, are exchangeable (i.e., the probability of the assignment does not depend on the identity of the elements).

Case Study: The Dirichlet Process (DP) Simplified

- In this example, we ignored the base distribution.
- To incorporate the base distribution, all we need to do is think of the categories above as a particular random value drawn from a distribution.
- That is, all the random variables in a category share the same value, and the values are distributed according to our chosen base distribution (mixture model).



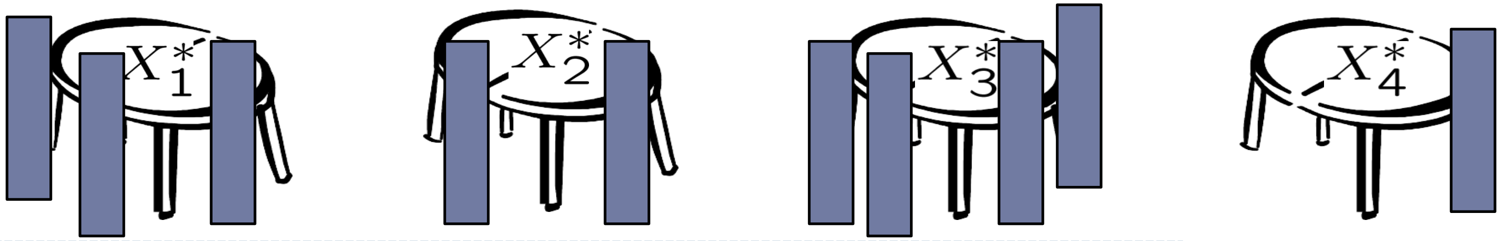
Case Study:
Chinese Restaurant Process (CRP)

Case Study: Chinese Restaurant Process (CRP)

- Recall the following from DP:

$$X_n | X_1, \ldots, X_{n-1} = \begin{cases} X_k^* & \text{with probability } \frac{\text{num}_{n-1}(X_k^*)}{n-1+\alpha} \\ \text{new draw from } G_0 & \text{with probability } \frac{\alpha}{n-1+\alpha} \end{cases}$$

- Consider a restaurant with infinitely many tables, where the X_n's represent the customers of the restaurant.
- From the above conditional probability distribution, we can see that a customer is more likely to sit at a table if there are already many people sitting there.
- However, with probability proportional to α, the customer will sit at a new table.



Case Study: CRP

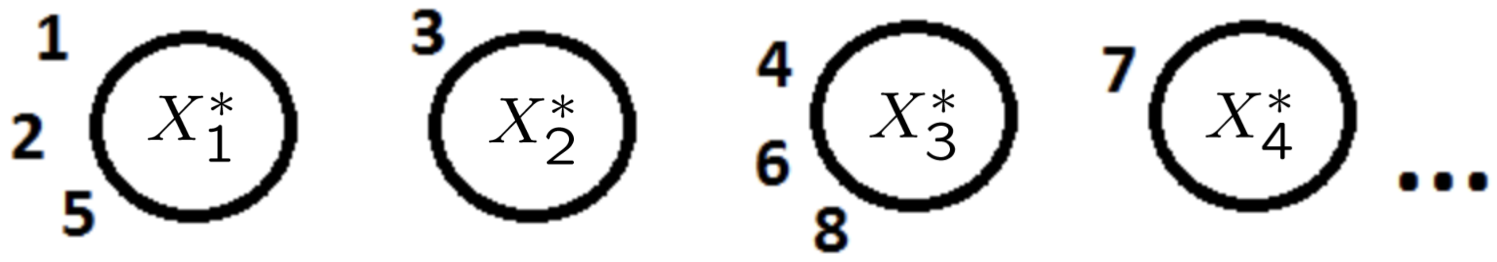
- The Chinese Restaurant Process is a metaphorical way for how a Dirichlet process generates data.

- Recall that, DP models randomness of a probability mass function (PMF) with unlimited number of samples.

- It is called the Chinese Restaurant Process (CRP) because the algorithm’s creators, Jim Pitman and Lester Dubins, named it after seeing massive Chinese restaurants in San Francisco’s Chinatown, where the probability of more clustered assignments (i.e., fewer categories) decreases.

Case Study: CRP

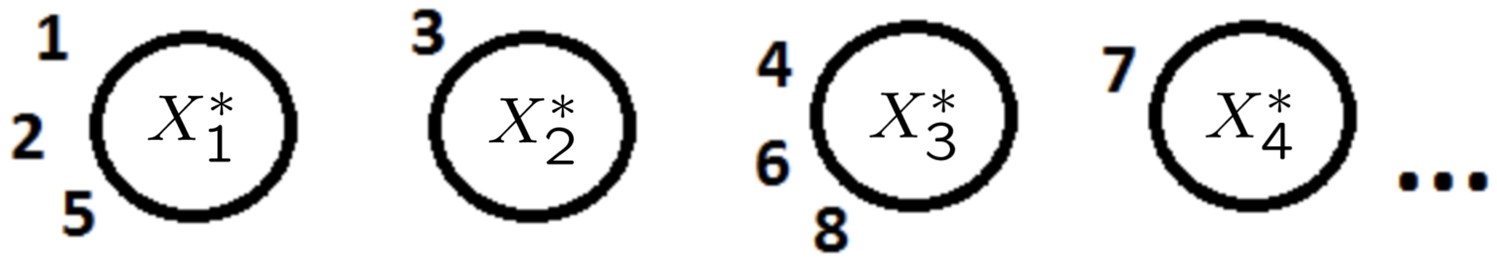
- Given the following CRP with infinite number of tables and 8 customers:



- In the above example, customers 1, 3, 4, and 7 are sat at empty tables; customers 2, 5, 6, and 8 are sat at existing tables.
- The number of tables in the restaurant is infinite.

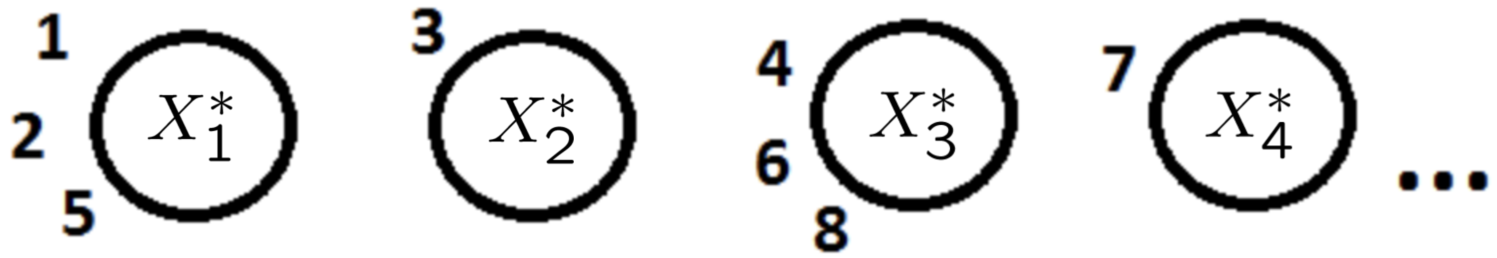
Case Study: CRP

- Given the following CRP with infinite number of table and 8 customers:



- When customer 1 enters, he can sit anywhere he likes. Customer 2 can sit in any empty seat, with the following probabilities:
- Table 1: 1 / (1 + α)
- New Table (i.e. any empty table): α / (1 + α)

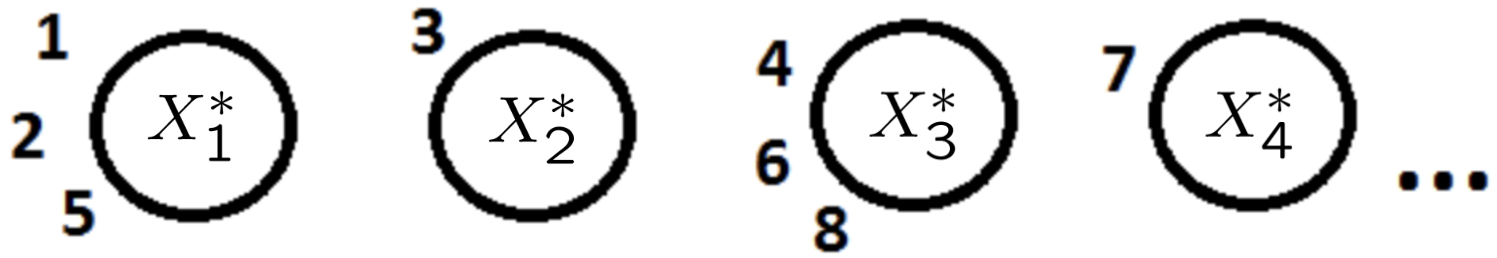
Case Study: CRP



- The probabilities for where customer 9 will sit are as follows:

Table 1: 3 / (8 + α)
Table 2: 1 / (8 + α)
Table 3: 3 / (8 + α)
Table 4: 1 / (8 + α)
New Table: α / (8 + α)

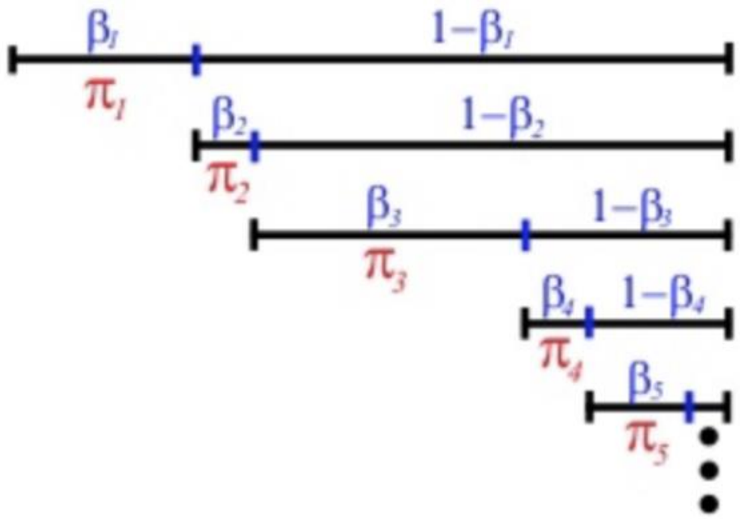
Case Study: CRP



- The numerator is the number of people already sat at a particular table
- The denominator is the number of customers in the restaurant (n – 1) plus α (a positive scalar hyperparameter)
- The probability of the nth customer sitting at an existing table is $\text{Num}_{n-k}(X^*_k)/(\alpha+n-1)$.
- The probability of the nth customer sitting at a new table is $\alpha/(\alpha+n-1)$.
- As more people sit at a particular table, those tables increase in popularity, so new customers are less likely to sit at empty tables.

Case Study: Stick-Breaking Process

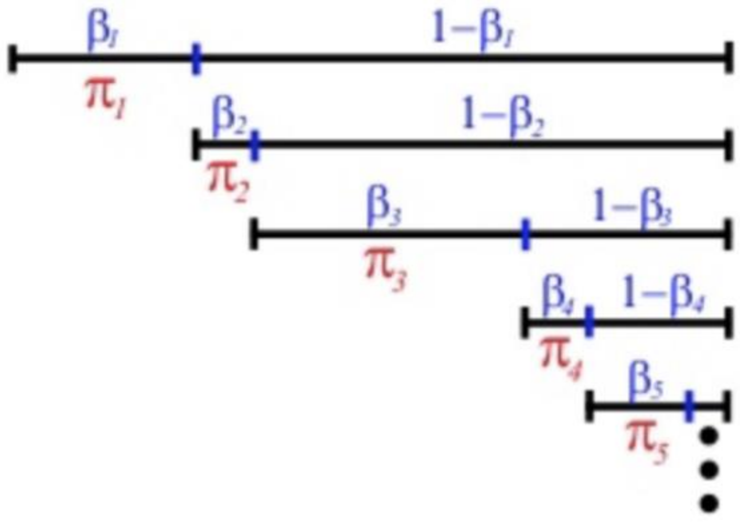
- The Stick-breaking construction is an alternative way to represent a Dirichlet process which was introduced by Sethuraman.
- Start with a stick of length 1, and break it at β_1, then the length of the broken part of the stick is π_1.
- Recursively break the rest of the stick to obtain β_1, β_2, …. and π_1, π_2. …



Stick-Breaking Process

- Where β_k has *Beta* Distribution $\beta(1,\alpha)$ & $\pi_k \propto \beta_k \prod_{l=1}^{k-1} 1 - \beta_l$
- Also we draw θ_k^* from a base distribution *H*.
- Then G has the DP distribution:

$$\sum_{k=1}^{\infty} \pi_k \theta_k^* \propto DP(\alpha, H)$$



The Blackwell-MacQueen urn scheme

- The Blackwell-MacQueen urn scheme can be used to represent a Dirichlet Process (introduced by Blackwell and MacQueen).
- It is based on the Polya urn scheme which can be seen as the opposite model of sampling without replacement.
- In the Polya urn scheme we assume that we have a non-transparent urn that contains colored balls and we draw balls randomly.
- When we draw a ball, we observe its color, we put it back in the urn and we add an additional ball of the same color.
- A similar scheme is used by Blackwell and MacQueen to construct a Dirichlet Process (Modified Pilya urn).

The Blackwell-MacQueen urn scheme

- Yet another way to visualize the Dirichlet process and CRP is as a modified Pólya urn scheme sometimes called the Blackwell-MacQueen sampling scheme.
- Imagine that we start with an urn filled with α black balls. Then:
  - Each time we need an observation, we draw a ball from the urn.
  - If the ball is black, we generate a new (non-black) color uniformly, label a new ball this color, drop the new ball into the urn along with the ball we drew, and return the color we generated.
  - Otherwise, label a new ball with the color of the ball we drew, drop the new ball into the urn along with the ball we drew, and return the color we observed.

The Blackwell-MacQueen urn scheme

- The resulting distribution over colors is the same as the distribution over tables in the Chinese Restaurant Process.
- Furthermore, when we draw a black ball, if rather than generating a new color, we instead pick a random value from a base distribution *H* and use that value to label the new ball, the resulting distribution over labels will be the same as the distribution over values in a Dirichlet process.