The Indian Buffet Process

Matt Dickenson mcd31
STA571/CS590.01 Due: 24 February, 2014

The following code was used to implement a generative model for the Indian Buffet Process. Example matrices generated from this code are displayed below.

Listing 1: IBP code

```
# Code up the generative model for the Indian Buffet Process.
library(ggplot2)

# generative model for ibp
ibp = function(alpha, N){
  assignments = matrix(NA, nrow=N, ncol=N*alpha)
   # extreme upper bound on expected number of columns

  dishes = rpois(1, alpha)
  zeroes = ncol(assignments) - dishes

  assignments[1,] = c(rep(1, dishes), rep(0, zeroes))

  for(i in 2:N){
    last_previously_sampled_dish =
      max(which(assignments==1, arr.ind=T)[, 'col'])

    dishes_from_previously_sampled =
      matrix(NA, nrow=1, ncol=last_previously_sampled_dish)
    for(k in 1:last_previously_sampled_dish){
      m_k = sum(assignments[1:i-1,k])
      prob = m_k / i
      dishes_from_previously_sampled[1,k] = rbinom(1,1, prob)
    }

    new_dishes = rpois(1, alpha/i)
    zeroes = ncol(assignments)-(last_previously_sampled_dish + new_dishes)
    assignments[i,] = c(dishes_from_previously_sampled,
      rep(1,new_dishes), rep(0, zeroes))
  }

  last_sampled_dish = max(which(assignments==1, arr.ind=T)[, 'col'])
  assignments = assignments[1:N, 1:last_sampled_dish]

  return(assignments)
}

# convert a matrix to its left-ordered form
lof = function(matrix){
```

```
  binary_digits = rep(NA, ncol(matrix))
  for(i in 1:ncol(matrix)){
    col = matrix[,i]
    val = binary(rev(col))
    binary_digits[i] = val
  }
  return(matrix[,rev(order(binary_digits))])
}

# turn a binary vector into a base-10 number
binary = function(vec){
  sum = 0
  for(i in 1:length(vec)){
    sum = sum + (2^(i-1) * vec[i])
  }
  return(sum)
}
```

How do the results vary with α and n? As α increases, so does the average number of dishes per customer and the total number of non-zero columns K_+ (compare Figures 1 and 2 to Figures 3 and 4). Because of the way that the Poisson parameter is reduced with each new customer, $K_+ \sim \text{Poisson}(\alpha H_N)$ (where H_N is the N^{th} harmonic number). By exchangeability, the number of dishes on each customer's plate is distributed Poisson(α).

When n increases but α is held constant, the resulting matrix is more sparse (compare Figures 1 and 3 to Figures 2 and 4). The matrix $\mathbf{Z}$ remains sparse because the effective dimensions of $\mathbf{Z}$ are $N \times K_+$, and the expected number of entries is $N\alpha$.

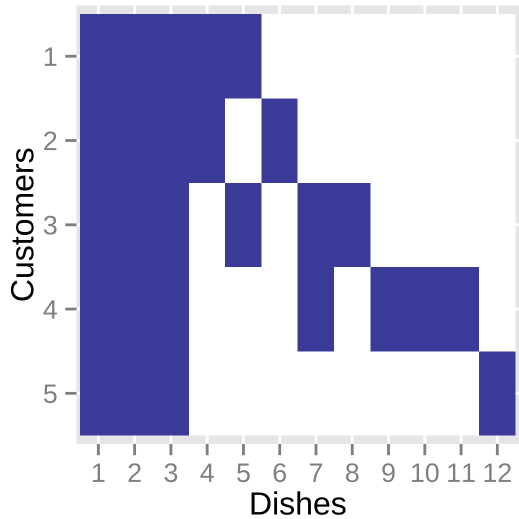

Figure 1: Example Matrix Generated from IBP with $\alpha = 5, n = 5$

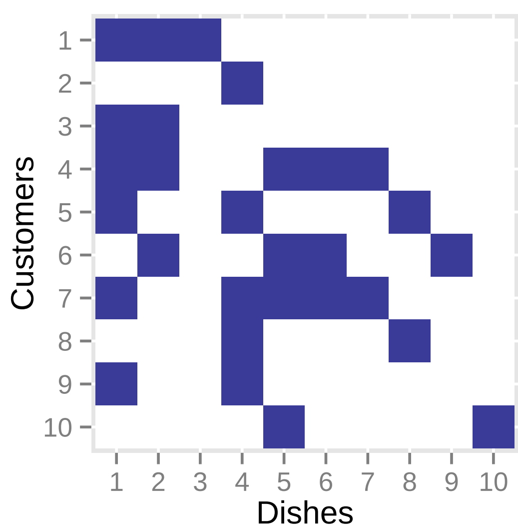

Figure 2: Example Matrix Generated from IBP with $\alpha = 5, n = 10$

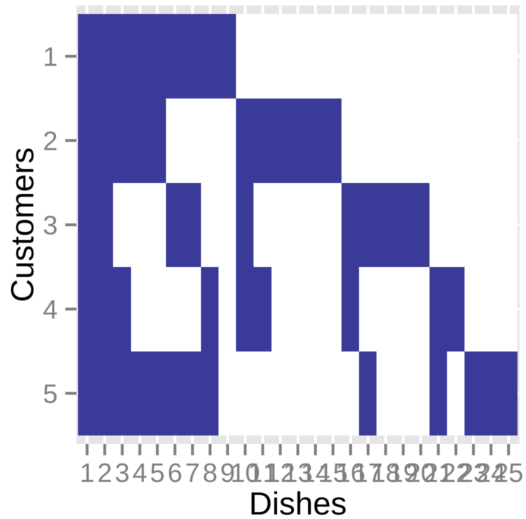

Figure 3: Example Matrix Generated from IBP with $\alpha = 10, n = 5$

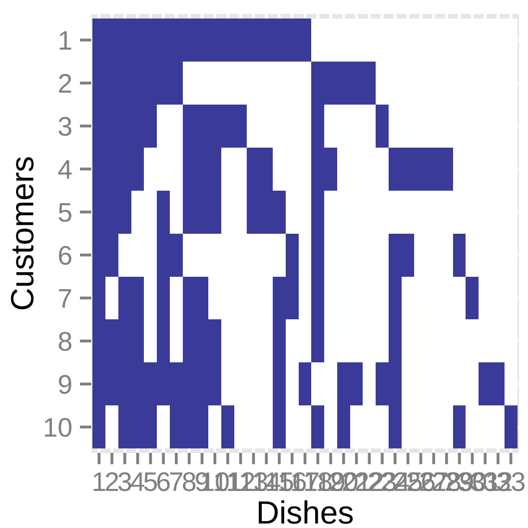

Figure 4: Example Matrix Generated from IBP with $\alpha = 10, n = 10$