Latent Feature Models

Latent feature values for N objects:

$$\mathbf{F} = [\mathbf{f}_1^T \mathbf{f}_2^T \dots \mathbf{f}_N^T]^T$$

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

where:

- $\mathbf{Z}$ is binary matrix indicating which features are possessed by an object
- $\mathbf{V}$ indicates the value of each feature for each project
- $\odot$ indicates element-wise (Hadamard) product

Can specify prior on $\mathbf{Z}$ and $\mathbf{V}$ separately:

$$P(\mathbf{F}) = P(\mathbf{Z})P(\mathbf{V})$$

Finite Feature Model

- K features
- $P(z_{i,k} \mid \pi) = \pi_k$
- $P(\mathbf{Z} \mid \pi)$ is:

$$P(\mathbf{Z} \mid \pi) = \prod_{i=1}^{N} \prod_{k=1}^{K} P(z_{i,k} \mid \pi_k)$$

Defining $m_k = \sum_{i=1}^{N} z_{i,k}$ as the number of objects having feature k:

$$P(\mathbf{Z} \mid \pi) = \prod_{k=1}^{K} \pi_k^{m_k} (1 - \pi_k)^{N - m_k}$$

Take π_k to follow Beta distribution, the conjugate prior of Binomial distribution then:

$$p(\pi_k) = \frac{\pi_k^{r-1}(1 - \pi_k)^{s-1}}{B(r, s)}$$

where $B(r, s)$ is Beta function:

$$\begin{aligned} B(r, s) &= \int_0^1 \pi_k^{r-1}(1 - \pi_k)^{s-1} \, d\pi_k \\ &= \frac{\Gamma(r)\Gamma(s)}{\Gamma(r + s)} \end{aligned}$$

Take $r = \alpha/K$, and $s = 1$, then:

$$\begin{aligned} B(\alpha/K, 1) &= \frac{\Gamma(\alpha/K)}{\Gamma(1 + \alpha/K)} \\ &= \frac{\Gamma(\alpha/K)}{(\alpha/K)\Gamma(\alpha/K)} \\ &= \frac{K}{\alpha} \end{aligned}$$

Model is:

$$\pi_k \mid \alpha \sim \text{Beta}(\alpha/K, 1)$$
$$z_{i,k} \mid \pi_k \sim \text{Bernoulli}(\pi_k)$$

Integrating over π_k, and this follows closely how it worked for finite mixture models, so we can more or less work through using the same mechanisms:

$$P(\mathbf{Z} \mid \alpha) = \prod_{k=1}^{K} \int p(\pi_k \mid \alpha) \prod_{i=1}^{N} p(z_{i,k} \mid \pi_k)\, d\pi_k$$

$$= \prod_{k=1}^{K} \int \frac{\alpha}{K} \pi_k^{\alpha/K-1} (1-\pi_k)^{N-m_k} \prod_{i=1}^{N} \pi_k^{z_{i,k}}\, d\pi_k$$

$$= \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} \int \pi_k^{\alpha/K-1} \pi_k^{m_k} (1-\pi_k)^{N-m_k}\, d\pi_k$$

$$= \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} \int \pi_k^{m_k+\alpha/K-1} (1-\pi_k)^{N-m_k}\, d\pi_k$$

$$= \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} B(m_k + \alpha, N - m_k + 1)$$

$$= \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} \frac{\Gamma(m_k + \alpha/K)\Gamma(N - m_k + 1}{\Gamma(N + \alpha/K + 1)}$$

... in agreement with the tutorial

Expectation of number non-zero entries

Griffiths and Ghahramani state that the expectation of number of non-zeros has an upper bound that is independent of K, which they then proceed to calculate. I initially wondered how it could be true that the number of non-zeros entries doesnt increase with K. Since there is no limit on the number of features that can be assigned to each object, surely adding more K features will increase the number of features? But its because the shape parameter of the Beta distribution is parameterized inversely in terms of K, ie α/K, so as $K \to \infty$, the Beta shape parameter $\to 0$.

Let's try to calculate $\mathbb{E}[\mathbf{1}^T\mathbf{Z}\mathbf{1}]$, without looking at the Griffiths and Ghahramani tutorial first. Actually, I already glanced through it, and they calculate first the expectation for one feature, $\mathbb{E}[\mathbf{1}^T\mathbf{z}_k]$. so let's try that.

$$\mathbb{E}[\mathbf{1}^T\mathbf{z}_k] = N\mathbb{E}[z_{n,k}]$$

$$= N \int p(z_{n,k} = 1 \mid \pi_k, \alpha) p(\pi_k \mid \alpha)\, d\pi_k$$

$$= N \int \pi_k \frac{\pi_k^{r-1}(1-\pi_k)^{s-1}}{B(r,s)}\, d\pi_k$$

Substitute $r = \alpha/K$, $s = 1$:

$$= N \int \pi_k \frac{\pi_k^{\alpha/K-1}(1-\pi_k)^{1-1}}{K/\alpha} \, d\pi_k$$

$$= N \int \frac{\alpha \pi_k^{\alpha/K+1-1}}{K} \, d\pi_k$$

$$= \frac{N\alpha}{K} \int \pi_k^{\alpha/K+1-1} \, d\pi_k$$

$$= \frac{N\alpha}{K} B(\alpha/K+1, 1)$$

$$= \frac{N\alpha}{K} \frac{\Gamma(\alpha/K+1)\Gamma(1)}{\Gamma(\alpha/K+2)}$$

$$= \frac{N\alpha}{K} \frac{\Gamma(\alpha/K+1)}{\Gamma(\alpha/K+2)}$$

$$= \frac{N\alpha}{K} \frac{\Gamma(\alpha/K+1)}{(\alpha/K+1)\Gamma(\alpha/K+1)}$$

$$= \frac{N\alpha}{K} \frac{1}{(\alpha/K+1)}$$

Then multiply by K to get $\mathbb{E}(\mathbf{1}^T\mathbf{Z}\mathbf{1})$:

$$= N\alpha \frac{1}{\alpha/K+1}$$

And then, glancing at the Griffiths and Ghahramani tutorial, and after correcting any obvious errors above as necessary, we get:

$$= \frac{N\alpha}{1+\alpha/K}$$

... as required.

Then, as stated, as $K \to \infty$, $\mathbb{E}(\mathbf{1}^T\mathbf{Z}\mathbf{1}) \to N\alpha$.

As stated, this is independent of K.

Lets try slotting in some concrete numbers...

```
import matplotlib.pyplot as plt
import numpy as np
import math


def plot_nonzeros(alpha, N=100):
    K_values = np.arange(0.1, 10, 0.1)
    nonzero_values = N * alpha / ( 1 + alpha / K_values)
    plt.plot(K_values, nonzero_values, label='alpha=%s' % alpha)

plt.clf()
plot_nonzeros(alpha=0.2)
plot_nonzeros(alpha=1)
plot_nonzeros(alpha=4)
plt.xlabel('K')
plt.ylabel('Expected number non-zero features')
```

```
plt.legend()
plt.title('Expected number of non-zeros for N=100')
plt.show()
```

See Figure 1

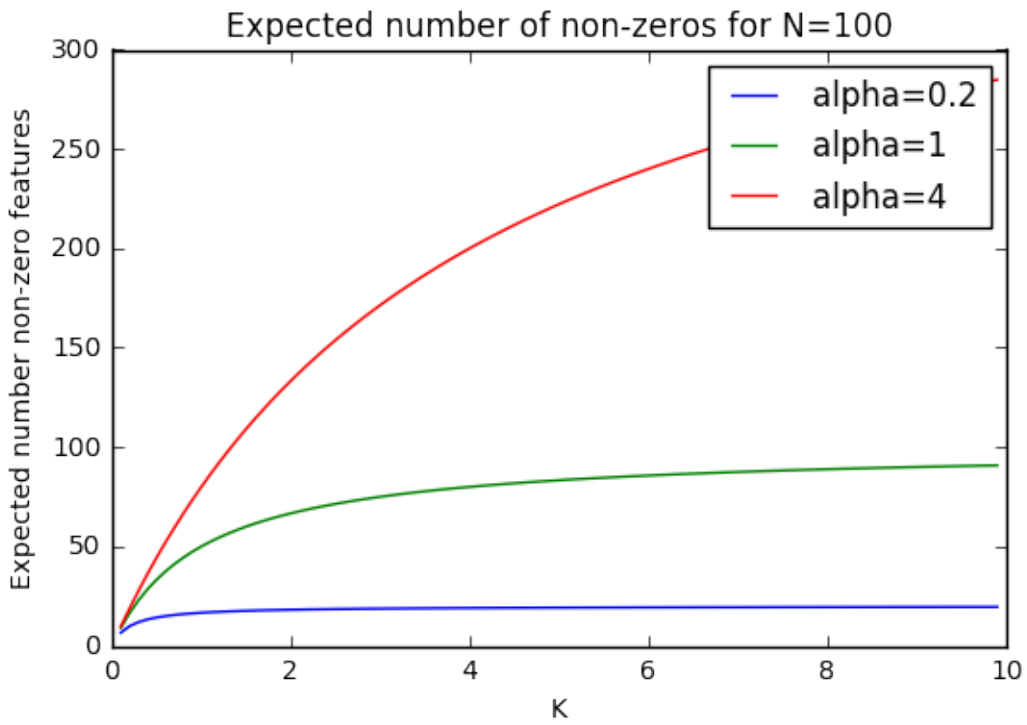

Figure 1:

For all values of α, the expected number of non-zeros increases with K, reaching an effective plateau at some point. For smaller values of α, the plateau is reached sooner, and is numerically smaller, than for larger values of α.

Equivalence classes

This section assumes one has already read the section in the Griffiths and Ghahramani tutorial on equivalence classes.

Thinking from the start, the equivalence classes will be used in the presence of some data that we will use to calculate the posterior of our prior from. So, the data will:

- have points in a certain order. we cant just swap pairs of data points (cf generation, where there is no particular order in which we need to generate new data)
- have points that are probably associated with features, which we could label, eg 'dahl', 'naan'
- however the index numbers of the features can be freely swapped around

So, for example the following two matrixes would be equivalent under these constraints:

```
Feature  1 (dahl)   2          3 (naan)
Tom      1          0          1          0  ... 0 ...
Jane     0          0          1          0  ... 0 ...

Feature  345 (naan) 100 (dahl)  222
Tom      1          1          0          0  ... 0 ...
Jane     1          0          0          0  ... 0 ...
```

...since both matrices show each person eating the same food as in the other matrix, just the feature ids have been swapped around.

If we assumed that each feature column was unique, like:

```
import matplotlib.pyplot as plt
import numpy as np


N = 5
K = 5

grid = np.zeros((N, K, 3), dtype=np.float32)
grid.fill(1.0)
for n in range(N):
    grid[n][n][:] = 0.0

plt.imshow(grid, interpolation='nearest', extent=[0, K, N, 0])
plt.xlabel('k')
plt.ylabel('n')
plt.show()
```

See Figure 2

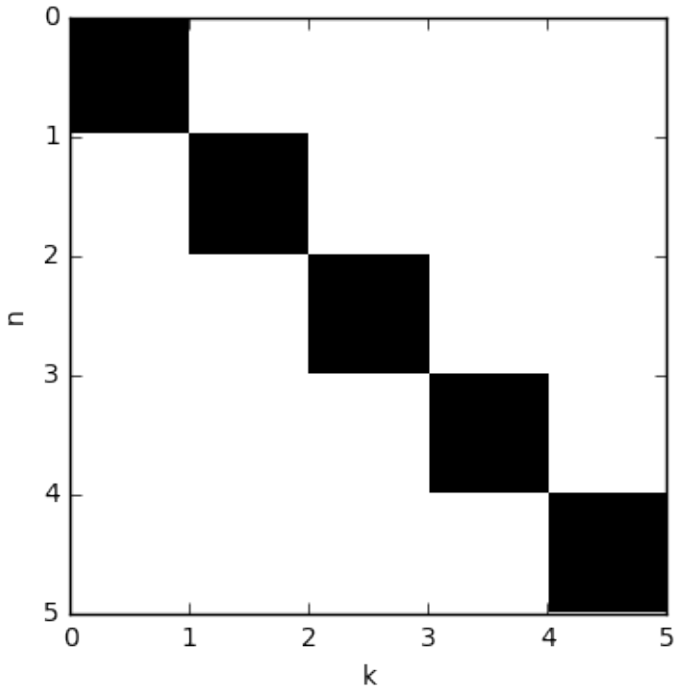

Figure 2:

Or, using just 3 features, and writing out as binary, like eg:

```
import matplotlib.pyplot as plt
import numpy as np


N = 5
K = 3

grid = np.zeros((N, K, 3), dtype=np.float32)
grid.fill(1.0)
for n in range(N):
    n_bin_str = np.binary_repr(n, width=K)
```

```
    for k, v in enumerate(n_bin_str):
        if v == '1':
            grid[n][K - 1 - k][:] = 0.0

plt.imshow(grid, interpolation='nearest', extent=[0, K, N, 0])
plt.xlabel('k')
plt.ylabel('n')
plt.show()
```

See Figure 3

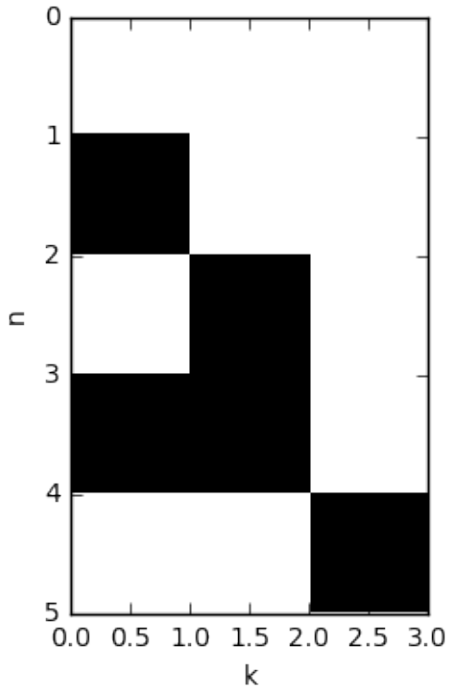

Figure 3:

... then if we know how many columns have non-zero values, ie how many features have non-zero values, then the number of ways of arranging these columns, by reordering the index numbers, would be:

$$\text{cardinality of } [\mathbf{Z}] = {}^{K}P_{\text{number non-zero features}}$$

If we define 'number features with non-zero values' to be K^+, then we can write:

$$\text{cardinality of } [\mathbf{Z}] = \frac{K!}{(K-K^+)!}$$

In a sense this is kicking the can down the line, since we don't have a formulae for K^+. But if we imagine a scenario where we are using Gibbs sampling, it seems plausible to imagine that we can easily calculate K^+, whereas we wouldnt want to actually count $\frac{K!}{K^+!}$ by hand?

However, in the above, we assume that each of the columns of the matrix is unique. So we might have a matrix like:

```
import matplotlib.pyplot as plt
import numpy as np


N = 5
K = 5
```

```
grid = np.zeros((N, K, 3), dtype=np.float32)

def grid_plot(grid, x, y):
    grid[y][x][:] = 0.0

grid.fill(1.0)
grid_plot(grid, 0, 0)
grid_plot(grid, 1, 2)
grid_plot(grid, 2, 2)
grid_plot(grid, 3, 3)
grid_plot(grid, 4, 2)

plt.imshow(grid, interpolation='nearest', extent=[0, K, N, 0])
plt.xlabel('k')
plt.ylabel('n')
plt.show()
```

See Figure 4

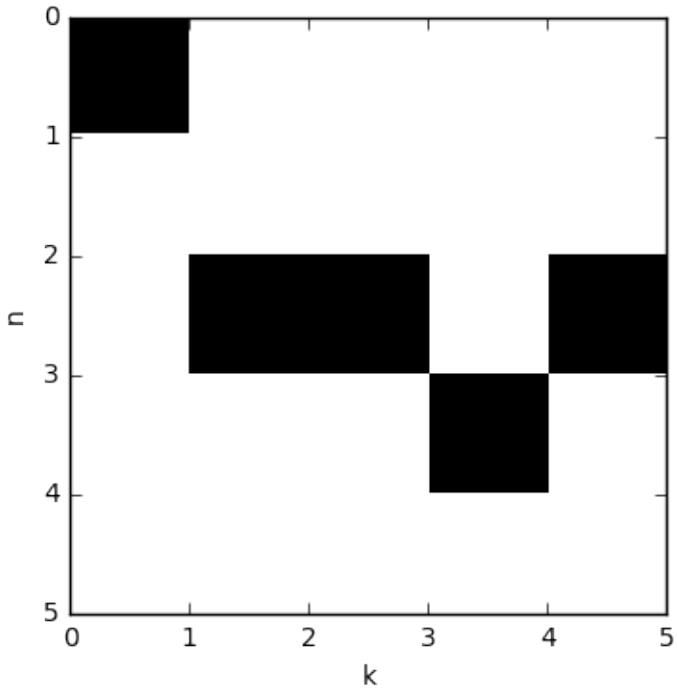

Figure 4:

In this case, the number of ways of arranging the matrix columns, as calculated by ${}^{K}P_{K^{+}}$, is an over-estimation, because permutating columns 1, 2, and 4 does not create a new matrix. So, we should compensate, in this case, by dividing by 3!, ie the number of columns that are identical to each other.

More generally, for any pattern of 1s and 0s in a feature column, we need to count the total number of columns with the exact same pattern, K_h, and divide the number of equivalent classes so far by $K_h!$.

In the tutorial, an identifier is defined and created for each pattern of 1s and 0s in a feature column, which is termed the "full history". The full history is defined to be the integer created by concatenating the 1s and 0s in a feature column into a binary number. I think the definition is somewhat arbitrary, as long as it uniquely identifies the patterns of 1s and 0s in a feature column, and can be used to concisely and repeatably label such a pattern.

Then, if we know the number of columns of each pattern, K_h, for different values of h, then the number of equivalence pairs will be:

$$\text{cardinality of } [\mathbf{Z}] = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!}$$

In this notation, we can see a key advantage of using the binary representation is that we can concisely represent all possible feature layouts, by iterating over all binary numbers, from 0, up to $2^N - 1$.

Thinking about how we'd use this, presumably, in a Gibbs sampling scenario, we could relatively easily obtain the values of K_h. We might not iterate over all the values of h as such: we might just store the counts in a hash table perhaps? For example, if we had only 1000 data samples, a tiny dataset, then 2^N is about $1e300$, which is far beyond the limits of current computers.

Thinking about this, as a hash table, we'd only need to store the actual concrete feature column identifiers we've actually seen. This is upper-bounded by the total number of non-zero values. The expectation for the total number of non-zero values was calculated in the previous section, and is $N\alpha$. So, with α of say 10, and 1000 data points, we'd need at most 10,000 keys to store the feature column identifiers in a hashmap, which is very doable. For one million data points, and $\alpha = 10$, we'd need at most 10 million keys in the hashmap, and if each key is an integer, that's about 40megabytes, ball-park, for the keys, and same again for the values, so total 80megabytes, and very doable.

Taking the infinite limit

We have:

$$P(\mathbf{Z} \mid \alpha) = \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} \frac{\Gamma(m_k + \alpha/K)\Gamma(N - m_k + 1)}{\Gamma(N + \alpha/K + 1)}$$

and:

$$\text{cardinality of } [\mathbf{Z}] = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!}$$

So:

$$P([\mathbf{Z}] \mid \alpha) = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \left(\frac{\alpha}{K}\right)^K \prod_{k=1}^{K} \frac{\Gamma(m_k + \alpha/K)\Gamma(N - m_k + 1)}{\Gamma(N + \alpha/K + 1)}$$

We want to find the limit as $K \to \infty$. Per the tutorial, we should divide the columns of $\mathbf{Z}$ into two sets: those with $m_k = 0$, and those with $m_k > 0$. We will reorder the features so that \$m_k >0 \$ iff $k \leq K^+$, and $m_k = 0$ otherwise.

So we get:

$$P([\mathbf{Z}] \mid \alpha) = \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \left(\frac{\alpha}{K}\right)^K \prod_{k=K^++1}^{K} \left(\frac{\Gamma(\alpha/K)\Gamma(N+1)}{\Gamma(N + \alpha/K + 1)}\right) \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k + \alpha/K)\Gamma(N - m_k + 1)}{\Gamma(N + \alpha/K + 1)}\right)$$

$$= \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \left(\frac{\alpha}{K}\right)^K \left(\frac{\Gamma(\alpha/K)\Gamma(N+1)}{\Gamma(N + \alpha/K + 1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k + \alpha/K)\Gamma(N - m_k + 1)}{\Gamma(N + \alpha/K + 1)}\right)$$

Looking at the limit as $K \to \infty$, any terms like $a + b/K$, where a and b are finite, will tend to a. We need to pay attention to any terms that are just in aK, or a/K which will tend to 0 or ∞, respectively. Gotcha, that got me: $\Gamma(0)$ is not 1: it's $+\infty$:

```
from scipy.special import gamma
import matplotlib.pyplot as plt
import numpy as np


def plot_gamma():
    X = np.arange(0, 5, 0.1)
    y = gamma(X)
    plt.plot(X, y)
    plt.ylim(0, 40)
    plt.show()
plot_gamma()
```

See Figure 5

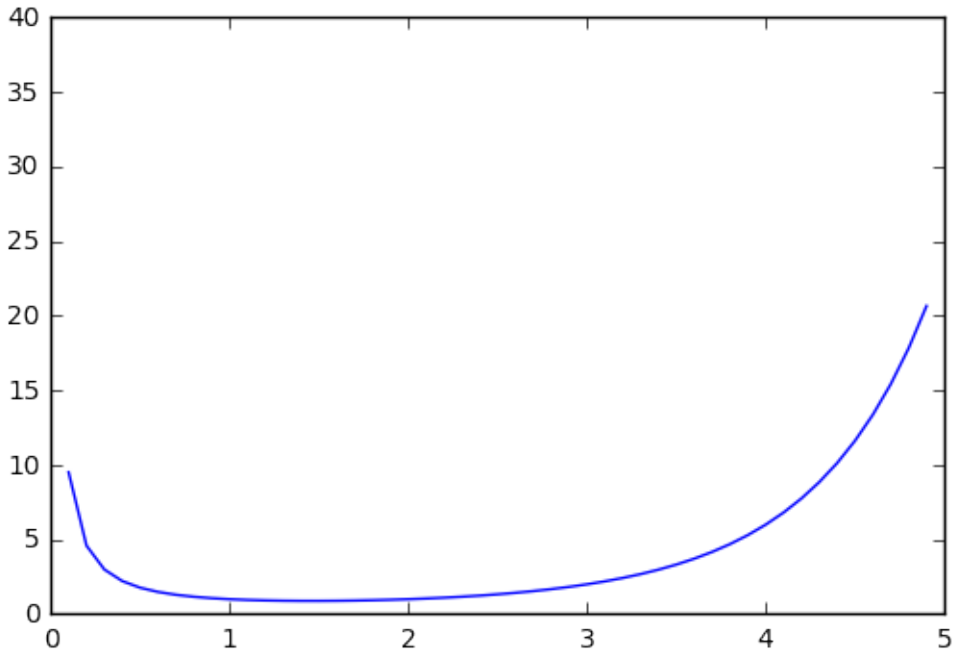

Figure 5:

... so we should expand the term in $\Gamma(\alpha/K)$, on the right hand side, from $\Gamma(\alpha/K)$ to $\frac{\Gamma(\alpha/K+1)}{\alpha/K}$. This is valid because we're reversing the normal direction, ie $\Gamma(x) = (x-1)\Gamma(x-1)$, for $x > 1$, and here our x is $\alpha/K + 1$.

$$\rightarrow_{K\rightarrow\infty} \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \left(\frac{\alpha}{K}\right)^K \left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{(\alpha/K)\Gamma(N+\alpha/K+1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

$$= \frac{K!}{\prod_{h=0}^{2^N-1} K_h!} \left(\frac{\alpha}{K}\right)^K \left(\frac{K}{\alpha}\right)^{K_0} \left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

Gotcha number two, that got me: whilst a single term in $a + b/K$ will tend to a as $K \rightarrow \infty$, the same term raised to a power of K, will not necessarily do so, and such terms must be handled as a whole.

We can combine the fractions in α/K:

$$\left(\frac{\alpha}{K}\right)^K \left(\frac{K}{\alpha}\right)^{K_0} = \left(\frac{\alpha}{K}\right)^{K^+}$$

Giving:

$$\rightarrow_{K\rightarrow\infty} \alpha^{K^+} \frac{K!}{K^{K^+} \prod_{h=0}^{2^N-1} K_h!} \left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

Looking at the term:

$$\prod_{h=0}^{2^N-1} K_h!$$

$K_h \leq K^+$ for $h > 0$. However, K_0 is not finite, so we should split that out separately:

$$\rightarrow_{K\rightarrow\infty} \alpha^{K^+} \frac{K!}{K^{K^+} K_0! \prod_{h=1}^{2^N-1} K_h!} \left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

And we have:

$$\frac{K!}{K_0! K^{K^+}} \rightarrow 1$$

(from earlier). So:

$$\rightarrow_{K\rightarrow\infty} \frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1} K_h!} \left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K_0} \prod_{k=1}^{K^+} \left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

At this point, I kind of got stuck, so started to work backwards, from the result in the tutorial.

Infinite limit of middle term

I first looked at the middle term, the one raised to the power K_0. Initially, I assumed this was assymptotic to 1, from looking at the terms inside, which tend themselves to 1. However, after inspecting the answer in the tutorial, it looks like we need to handle the terms inside the brackets together with the infinite power of K_0.

The tutorial asserts that:

$$\left(\frac{N!}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^{K} \rightarrow \exp\left(-\alpha \sum_{j=1}^{N} 1/j\right)$$

I started by trying to figure out this bit. Is it using Stirling's approximation? Taylor series? Something else?

Taylor series includes things like:

$$\exp(x) = \sum_{j=1}^{\infty} \frac{x^j}{j!}$$

Stirling's approximation looks something like:

$$n! \rightarrow \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

I tried inverting the equivalence in the tutorial, to see if it looked more familiar:

$$\left(\frac{\prod_{j=1}^{N}(j+\alpha/K)}{N!}\right)^{K} \rightarrow \exp\left(\alpha\sum_{j=1}^{N}1/j\right)$$

It sort of has characteristics of both Taylor series for $\exp(x)$ and Stirling's approximation, but not clear match with either :-P

- there's a π term in the Stirling's approximation, which isnt present in the tutorial equivalence. And seems no obvious reason why the π would disappear
- the Taylor series involves a sum over infinite terms, whereas the tutorial expression has a product, over a finite number of terms

I tried taking logs, of the tutorial limit expression, to get rid of the power to K, and the $\exp(x)$ This didnt really work for me:

$$K\log\left(\prod_{j=1}^{N}(j+\alpha/K)\right) - K\log(N!) \rightarrow \alpha\sum_{j=1}^{N}\frac{1}{j}$$

So:

$$K\sum_{j=1}^{N}\log(j+\alpha/K) - K\log(N!) \rightarrow \alpha\sum_{j=1}^{N}\frac{1}{j}$$

Writing the log of the factorial also as a sum of logs:

$$K\sum_{j=1}^{N}\log(j+\alpha/K) - K\sum_{j=1}^{N}(\log(j)) \rightarrow \alpha\sum_{j=1}^{N}\frac{1}{j}$$

Looking at just the left hand side for now:

$$K\sum_{j=1}^{N}\log(j+\alpha/K) - K\sum_{j=1}^{N}(\log(j))$$

$$= K\sum_{j=1}^{N}(\log(j+\alpha/K) - \log(j))$$

$$= K\sum_{j=1}^{N}\log\left(1+\frac{\alpha}{jK}\right)$$

At this point, I gave up, and looked at the appendix :-) The identity that we want is:

$$\exp(x) = \lim_{n\rightarrow\infty}\left(1+\frac{x}{n}\right)^{n}$$

which in our case becomes, by inverting both sides:

$$\lim_{K\rightarrow\infty}\left(\frac{1}{1+x\,/\,K}\right)^{K} = \frac{1}{\exp(x)}$$

So, looking at:

$$\left(\frac{N!}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^{K}$$

This is:

$$\left(\frac{\prod_{j=1}^{N} j}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^{K}$$
$$=\left(\prod_{j=1}^{N}\frac{j}{(j+\alpha/K)}\right)^{K}$$
$$=\left(\prod_{j=1}^{N}\frac{1}{1+\frac{\alpha}{jK}}\right)^{K}$$

How to move the product to outside the power?

From basics:

$$(xyz)^a$$
$$=x^a y^a z^a$$

So,

$$\left(\prod_{i=1}^{N} x_i\right)^{a}$$
$$=\prod_{i=1}^{N} x_i^a$$

And, in our case:

$$\left(\prod_{j=1}^{N}\frac{1}{1+\frac{\alpha}{jK}}\right)^{K}$$
$$=\prod_{j=1}^{N}\left(\frac{1}{1+\frac{\alpha}{jK}}\right)^{K}$$

Using the identity from earlier:

$$\lim_{K\to\infty}\prod_{j=1}^{N}\left(\frac{1}{1+\frac{\alpha}{jK}}\right)^{K}$$
$$=\prod_{j=1}^{N}\left(\frac{1}{\exp(\alpha/j)}\right)$$
$$=\prod_{j=1}^{N}\exp\left(-\frac{\alpha}{j}\right)$$
$$=\exp\left(-\sum_{j=1}^{N}\frac{\alpha}{j}\right)$$
$$=\exp\left(-\alpha\sum_{j=1}^{N}\frac{1}{j}\right)$$

... as required.

Initial pre-limit expression

Next up, we need to get from the earlier expression for the infinite limit, as far as obtaining the $(N!/(\prod_{j=1}^{N}(j+\alpha/K))^K$ expression. The expression so far was:

$$\frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1} K_h!}\left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K_0}\prod_{k=1}^{K^+}\left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

We need to change the current middle expression to be a power of K rather than K_0 (by inspection of the tutorial, and because we'll need it for the identity in exp):

$$=\frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1} K_h!}\left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K}\left(\frac{\Gamma(N+\alpha/K+1)}{\Gamma(\alpha/K+1)\Gamma(N+1)}\right)^{K+}\prod_{k=1}^{K^+}\left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(N+\alpha/K+1)}\right)$$

Merging with the right-hand expression:

$$=\frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1} K_h!}\left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^{K}\prod_{k=1}^{K^+}\left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(\alpha/K+1)\Gamma(N+1)}\right)$$

Right-hand pre-limit expression

Looking at the right-hand expression:

$$\prod_{k=1}^{K^+}\left(\frac{\Gamma(m_k+\alpha/K)\Gamma(N-m_k+1}{\Gamma(\alpha/K+1)\Gamma(N+1)}\right)$$

Since N and m_k are integers, we have:

$$=\prod_{k=1}^{K^+}\left(\frac{\Gamma(m_k+\alpha/K)(N-m_k)!}{\Gamma(\alpha/K+1)N!}\right)$$

For the Gamma terms containing α/K, we can move the integer part, m_k, in the numerator, into a product term, leaving a $\Gamma(\alpha/K+1)$ term, which will cancel with the denominator. First expansion will be:

$$\begin{aligned}&\Gamma(m_k+\alpha/K)\\&=\Gamma((m_k-1)+1+\alpha/K)\\&=((m_k-2)+\alpha/K+1)\Gamma((m_k-2)+\alpha/K+1)\end{aligned}$$

Last expansion will be:

$$(m_k-m_k+\alpha/K+1)\ldots\Gamma(\alpha/K+1)$$

So, the full expansion will be:

$$\Gamma(m_k+\alpha/K)=\Gamma(\alpha/K+1)\prod_{j=1}^{m_k-1}(j+\alpha/K)$$

Slotting this into the right-hand expression, and cancelling the $\Gamma(1+\alpha/K)$ terms:

$$=\prod_{k=1}^{K^+}\left(\frac{(N-m_k)!\prod_{j=1}^{m_k-1}(j+\alpha/K)}{N!}\right)$$

... which matches the right-hand expression in the tutorial, prior to computing the limit.

Middle pre-limit expression

Looking now at the middle expression, we have:

$$\left(\frac{\Gamma(\alpha/K+1)\Gamma(N+1)}{\Gamma(N+\alpha/K+1)}\right)^K$$

We want, from the tutorial:

$$\left(\frac{N!}{\prod_{j=1}^{N}(j+\frac{\alpha}{K}}\right)^K$$

Let's try the same approach as for the right-hand expression: expand the $\Gamma(N+\alpha/K+1)$, in the denominator, to give a polynomial multiplied by $\Gamma(\alpha/K+1)$, which will cancel with the Gamma in the numerator.

Looking at $\Gamma(N+\alpha/K+1)$, the first expansion will be:

$$((N-1)+\alpha/K+1)\Gamma((N-1)+\alpha/K+1)$$

The last expansion will be:

$$((N-N)+\alpha/K+1)\ldots\Gamma((N-N)+\alpha/K+1)$$

So, overall the expansion will be:

$$\Gamma(\alpha/K+1)\prod_{j=1}^{N}(j+\alpha/K)$$

Slotting this into the middle expression, and cancelling the terms in $\Gamma(\alpha/K+1)$, we get:

$$\left(\frac{\Gamma(N+1)}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^K$$

Then, since N is an integer, writing $\Gamma(N+1)$ as $N!$, we get:

$$\left(\frac{N!}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^K$$

... which matches the middle pre-limit term in the tutorial.

Taking the infinite limit

We have the pre-limit expression:

$$=\frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1}K_h!}\cdot\frac{K!}{K_0!K^{K^+}}\cdot\left(\frac{N!}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^K\cdot\prod_{k=1}^{K^+}\left(\frac{(N-m_k)!\prod_{j=1}^{m_k-1}(j+\alpha/K)}{N!}\right)$$

We've already seen that:

$$\lim_{K\to\infty}\frac{K!}{K_0!K^{K^+}}=1$$

(from the section on latent classes earlier)

We've also seen that:

$$\lim_{K\to\infty}\left(\frac{N!}{\prod_{j=1}^{N}(j+\alpha/K)}\right)^{K}=\exp\left(-\alpha\sum_{j=1}^{N}\frac{1}{j}\right)$$

So, we're left with taking the limit of:

$$\lim_{K\to\infty}\prod_{k=1}^{K^+}\left(\frac{(N-m_k)!\prod_{j=1}^{m_k-1}(j+\alpha/K)}{N!}\right)$$

My earlier intuition would be to just remove the expression in α/K, giving:

$$\prod_{k=1}^{K^+}\left(\frac{(N-m_k)!\,(m_k-1!}{N!}\right)$$

. . . which does match the tutorial, but I sort of no longer trust my intuition on this point, and looked at the appendix.

The appendix multiplies out the innermost product, the product over j, ~~using the Binomial theorem~~ by hand I think?, giving:

$$\prod_{j=1}^{m_k-1}\left(j+\frac{\alpha}{K}\right)$$

$$=(m_k-1)!+\sum_{j=1}^{m_k-1}\frac{\alpha}{K}\frac{(m_k-1)!}{j}+\left(\frac{\alpha}{K}\right)^2(\dots)\cdots+\left(\frac{\alpha}{K}\right)^{m_k-1}(\dots)$$

Since m_k is finite, and therefore also j too, then each term goes to zero, because of $\frac{1}{K}$, except for the first $(m_k-1)!$ term.

So, finally, we have the expression for the infinite limit of $P([\mathbf{Z}])$:

$$\frac{\alpha^{K^+}}{\prod_{h=1}^{2^N-1}K_h!}\cdot\exp\left(-\alpha\sum_{j=1}^{N}\frac{1}{j}\right)\cdot\prod_{k=1}^{K^+}\frac{(N-m_k)!(m_k-1)!}{N!}$$

Indian Buffet process

Per tutorial, we can sample from IBP by choosing each existing dish with probability m_k/i where i is the customer number, and m_k is the number of customers for that dish so far, then sampling Poisson(α/i) new dishes. Interestingly, the probability of choosing each existing dish is independent of α.

Let's first look at the Poisson distribution. It represents the probability of a given number of events occurring in unit interval. λ is the average number of events per unit interval. The probability function is discrete, having probability mass only for integer values.

```
import numpy as np
import matplotlib.pyplot as plt
import math

def draw_poisson(lambda_value, color_letter):
    plt.clf()
    for k in range(10 + 1):
        prob_mass = np.power(lambda_value, k) * np.exp(-lambda_value) / math.factorial(k)
        plt.plot([k, k], [0, prob_mass], color_letter)
    plt.title('Probability mass function for Poisson, lambda=%s' % lambda_value)
    plt.show()
```

```
draw_poisson(0.3, 'purple')
draw_poisson(1.0, 'g')
draw_poisson(3.0, 'b')
draw_poisson(5.0, 'yellow')
```

See Figure 6

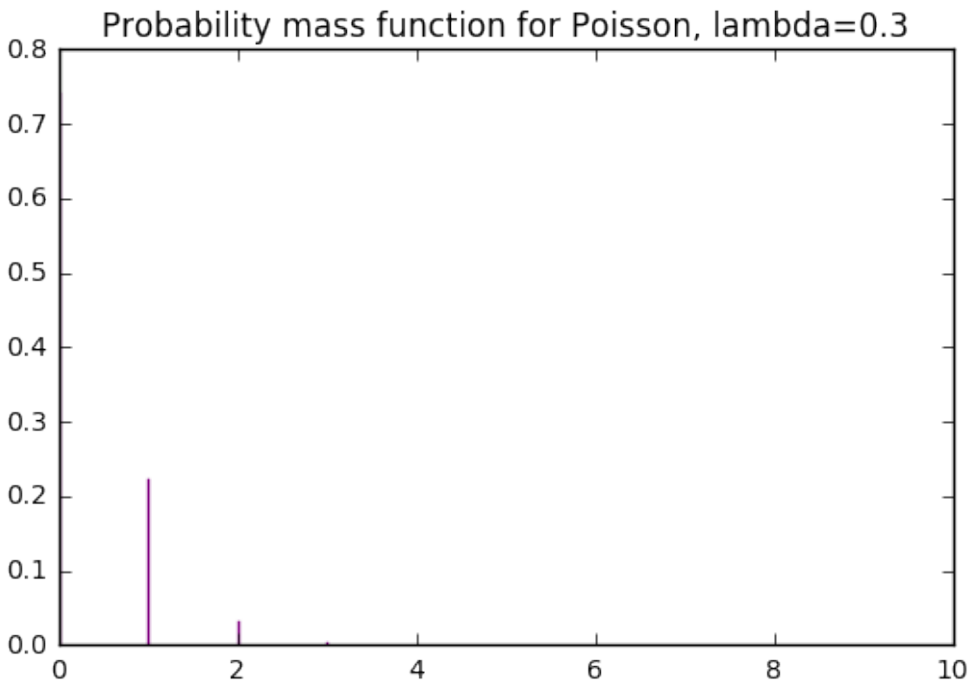

Figure 6:

See Figure 7

See Figure 8

See Figure 9

Let's try sampling from an Indian Buffet Process:

```
import numpy as np
import matplotlib.pyplot as plt
import math
import random
from collections import defaultdict


def plot_ibp(alpha=3.0, N=50):
    K_plus = 0
    count_by_k = defaultdict(int)
    chosen_by_customer = defaultdict(set)

    random.seed(123)
    np.random.seed(123)
    for n in range(N):
        # first sample existing dishes:
        for k in range(K_plus):
            prob = count_by_k[k] / (n + 1)
            if random.uniform(0, 1) <= prob:
```

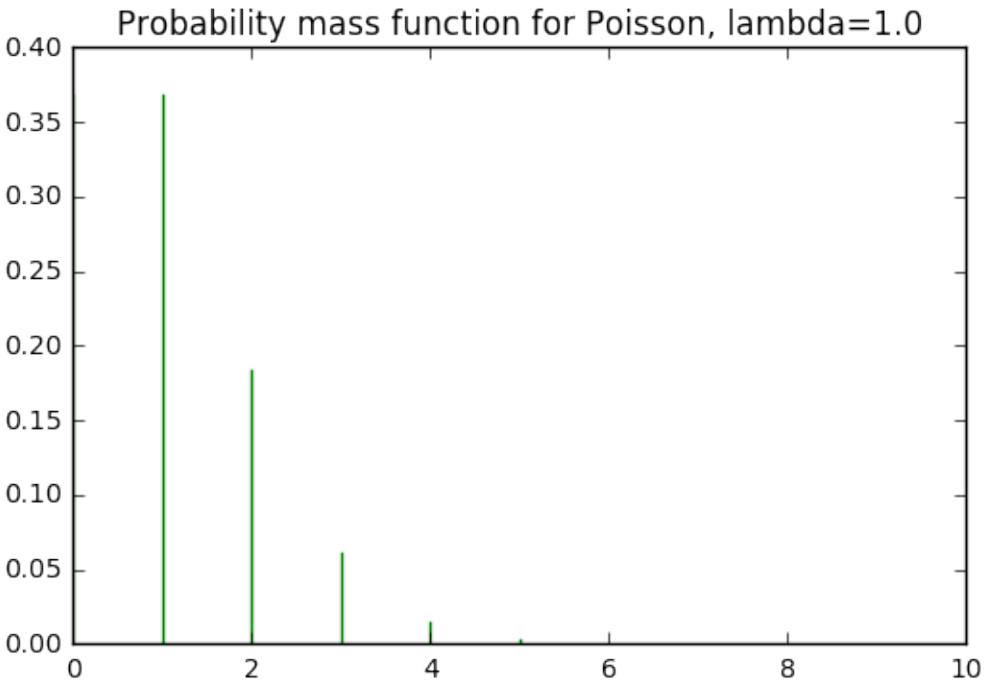

Figure 7:

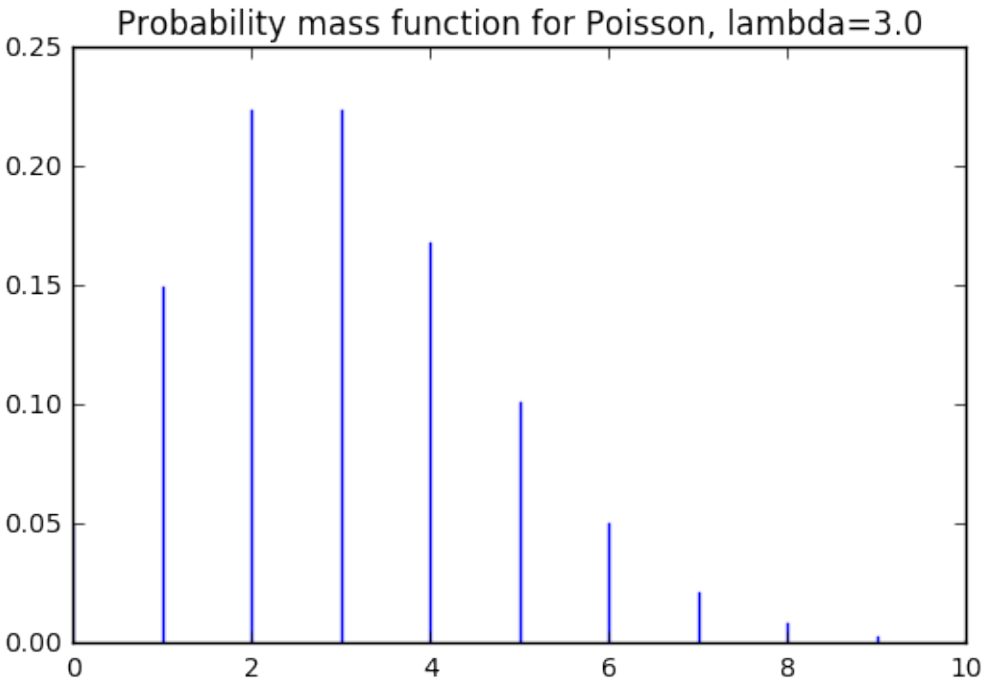

Figure 8:

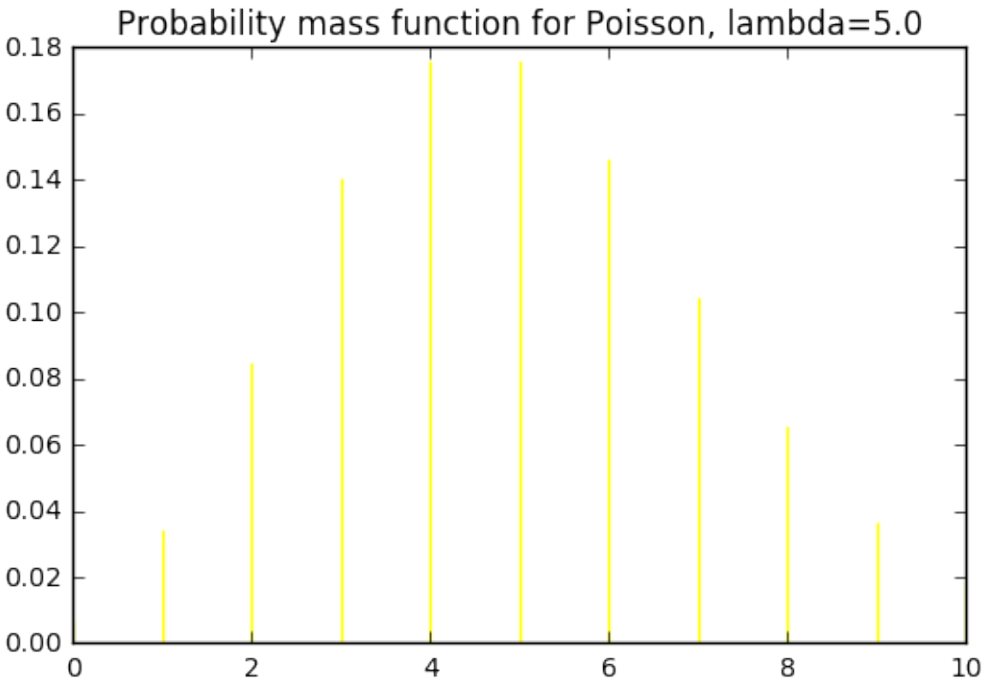

Figure 9:

```
                chosen_by_customer[n].add(k)
                count_by_k[k] += 1
        # now try some new ones
        num_new = np.random.poisson(lam=alpha / (n + 1))
        for i in range(num_new):
            k = K_plus
            chosen_by_customer[n].add(k)
            count_by_k[k] += 1
            K_plus += 1

    print('K_plus', K_plus)
    grid = np.zeros((N, K_plus, 3), dtype=np.float32)
    grid.fill(1.0)
    for n, chosen in chosen_by_customer.items():
        for k in chosen:
            grid[n][k][0:2] = 0.0

    plt.clf()
    plt.imshow(grid, interpolation='nearest', extent=[0, K_plus, N, 0])
    plt.xlabel('k')
    plt.ylabel('n')
    plt.title('Indian Buffet Process, alpha=%s' % alpha)
    plt.show()

plot_ibp(alpha=3.0)
plot_ibp(alpha=1.0)
plot_ibp(alpha=0.3)
plot_ibp(alpha=10.0)
```

```
K_plus 11
```

See Figure 10

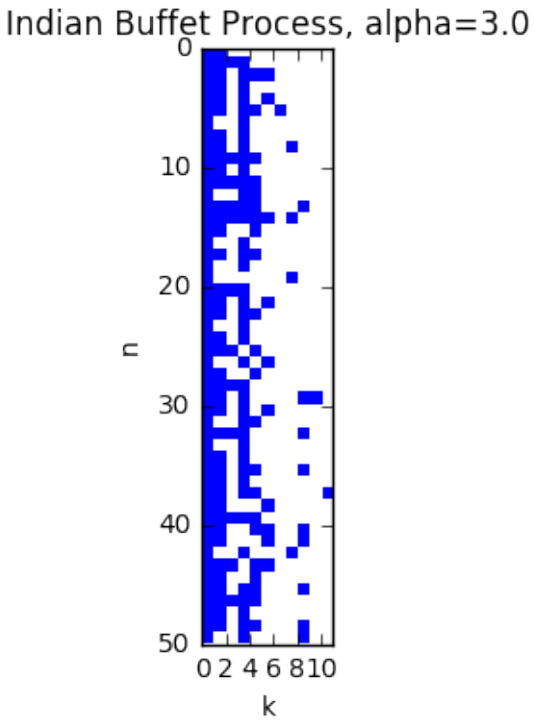

Figure 10:

`K_plus 3`

See Figure 11

`K_plus 1`

See Figure 12

`K_plus 44`

See Figure 13

Analysis of values of K^+, for some values of α and N

Looking at the number of features, K^+, for different values of N and α, empirically, for the ranges sampled, we can see that:

- K_+ tends to be less than N, at least for $\alpha <= 10$, and $N >= 40$
- it looks like the ratio of K^+/N decreases with increasing N, which makes sense, given that the Poisson parameter decreases for each new customer.

Thinking about an actual use-case scenario, for example distribution of purchases amongst users of Amazon, I wonder what value of α would match approximately the number of purchases per customer, and the number of customers? Lets say there are 50 million customers, and 100,000 products. Then we have:

- $K_+ = 1e5$
- $N = 5e7$

To save on maths, let's try empirically :-P

Nuance: to make the program run in reasonable time, lets say there are 1 million customers, and 100,000 produdts. And we'll plot K^+ vs α.

```
import numpy as np
import matplotlib.pyplot as plt
import math
```

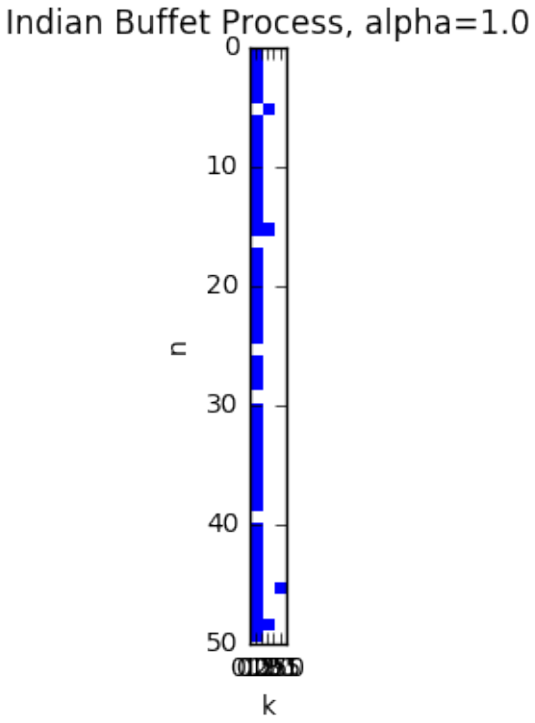

Figure 11:

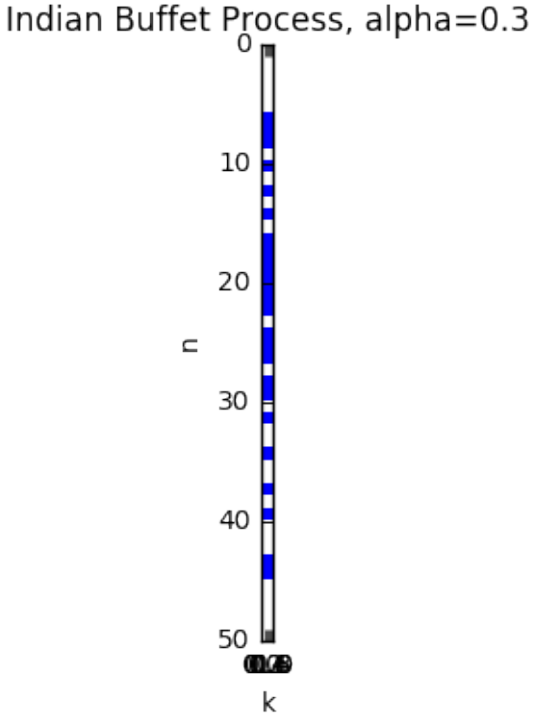

Figure 12:

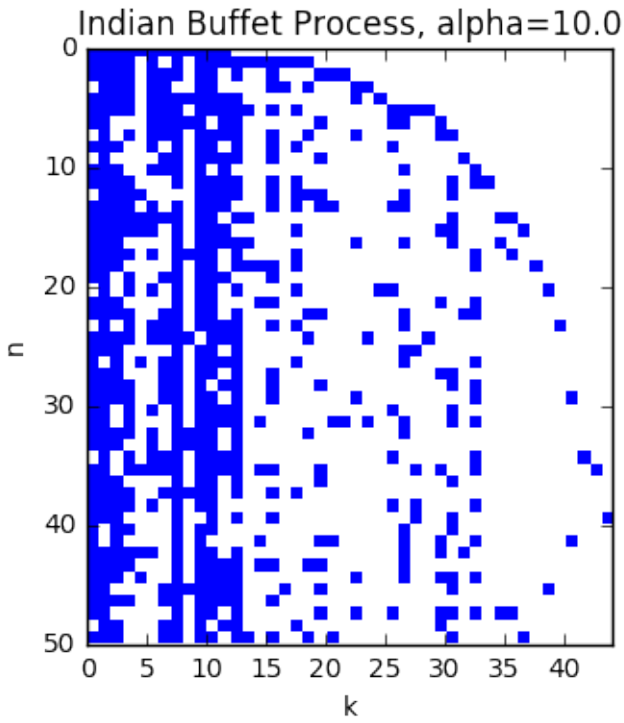

Figure 13:

```
import random
from collections import defaultdict


def calc_kplus(alpha=3.0, N=50):
    K_plus = 0
    random.seed(123)
    np.random.seed(123)
    # we simply sample from Poisson each time, and add
    # up the resuls
    for n in range(N):
        num_new = np.random.poisson(lam=alpha / (n + 1))
        K_plus += num_new

    print('N', N, 'alpha', alpha, 'K_plus', K_plus)
    return K_plus

X = range(1000, 10000 + 1000, 1000)
y = np.zeros((len(X), ), dtype=np.int32)
for i, alpha in enumerate(X):
    kplus = calc_kplus(N=int(1e6), alpha=alpha)
    y[i] = kplus
plt.clf()
plt.plot(X, y)
plt.show()
```

```
N 1000000 alpha 1000 K_plus 14296
N 1000000 alpha 2000 K_plus 28906
N 1000000 alpha 3000 K_plus 43305
N 1000000 alpha 4000 K_plus 57735
N 1000000 alpha 5000 K_plus 72091
```

```
N 1000000 alpha 6000 K_plus 86674
N 1000000 alpha 7000 K_plus 100901
N 1000000 alpha 8000 K_plus 115377
N 1000000 alpha 9000 K_plus 129694
N 1000000 alpha 10000 K_plus 144163
```

See Figure 14

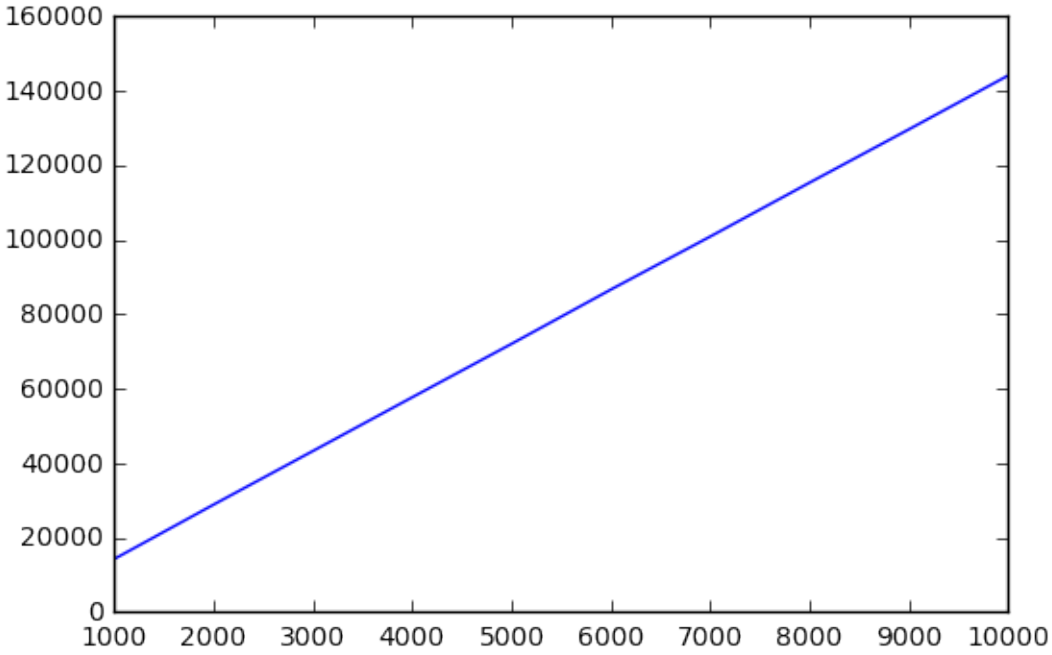

Figure 14:

It looks like for N of 1 million, we need α of about 8,000, in order to get K^+ of around one hundred thousand. Interestingly, for these numbers, K^+ is approximately linear with α. Which makes sense, since the average number of dishes sampled by each new customer is proportional to α.

Actually, the value of K^+ should be the sum of a harmonic series, if we assume that the Poisson function simply returns α/i each time. So, K^+ will be:

$$K^+ \approx \sum_{i=1}^{N} \frac{\alpha}{i}$$

This is approximately $\alpha \log(N)$

Let's examine how $\alpha \log(N)$ compares with αH_N. Actually, this formula shows that K^+ is directly proportional to α, assuming the result of the Poisson gives its expected value each time, which it will on average do. So, let's just plot for $\alpha = 1$. Let's plot for N up to 10 million, which corresponds approximately to about the order of magnitude of for example Amazon, I guess.

```
import matplotlib.pyplot as plt
import numpy as np
import math


def compare_logN_HN():
    X = np.arange(2e5, int(1e7), 2e5)
    y_log = np.log(X)
    plt.clf()
```

```
    plt.plot(X, y_log, label='log(n)')

    y_hn = np.zeros((len(y_log),), dtype=np.float32)
    HN = 0
    X_values = set(X)
    for n in range(1, int(1e7) + 1):
        HN += 1 / n
        if n in X_values:
            y_hn[n // int(2e5) - 1] = HN
    plt.plot(X, y_hn, label='Harmonic Series')

    plt.xlabel('N')
    plt.ylabel('y')

    plt.legend()
    plt.show()

compare_logN_HN()
```

See Figure 15

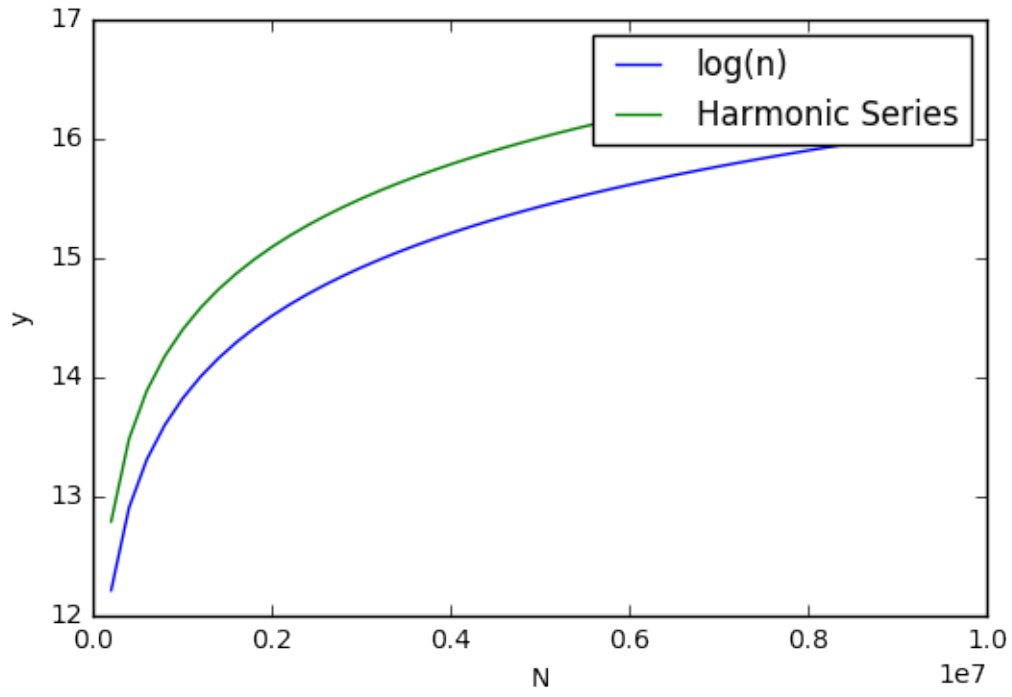

Figure 15:

We can see that $log(n)$ follows the curve of the harmonic series, H_N quite closely, but is slightly translated from it. Close enough for ball-park estimation of K^+ given N and α though. So, if we tried to estimate α directly, for the earlier scenario:

For $N = 1e6$, $K^+ = 1e5$, we have:

$$K^+ \approx \alpha \log(N)$$

So,

$$\alpha \approx \frac{K^+}{log(N)}$$

$$\approx 7000$$

Seems pretty close to the K^+ estimated by drawing Poisson samples one by one earlier:-)

If we choose $N = 5e7$, and $K^+ = 1e5$, then we get $\alpha \approx 6000$

(Since N is greater, so we need a slightly lower α to get the same K^+ value)

More generally, since many datasets will have a size that lies in the range $1e5 <= N <= 1e9$, $\log(N)$ will tend to lie in the range $10 \leq \log(N) \leq 20$, and so we can approximate K^+ to within an order of magnitude as:

$$K^+ \approx 10\alpha$$

(Independent of N, for a wide range of realistic values of N). This holds approximately for example in the Amazon guesstimate scenario above.

Probability of matrices sampled from IBP

The tutorial states that the probability of a matrix sampled from the IBP, by the process of drawing sequentially, as above, is:

$$P(\mathbf{Z})_{\text{IBP}} = \frac{1}{\prod_{h=1}^{N} K_1^{(i))}!} P(\mathbf{Z})$$

where:

- $P(\mathbf{Z})_{\text{IBP}}$ is the probability of drawing a matrix from this process, and
- $P(\mathbf{Z})$ is the probability of drawing a matrix from the Dirichlet prior, from earlier, without taking into account cardinality/partition-equivalance.

Initially, I was interpreting $P(\mathbf{Z})_{IBP}$ to mean the equivalence class cardinality for the draws from the IBP, which confused me, because I couldnt see how $\prod_{h=1}^{N} K_1^{(i)}!$ gave this. However, later I realized that in fact $P(\mathbf{Z})_{IBP}$ is the probability of drawing this matrix, taking into account that not all matrices are possible through this drawing scheme.