The Chinese restaurant process

COMPSCI 3016: Computational Cognitive Science
Dan Navarro & Amy Perfors
University of Adelaide

Abstract

The Chinese restaurant process (CRP) is an extremely simple and powerful tool. Unfortunately, it's also one of the most poorly described concepts in the statistical literature. This note tries to demystify the CRP. If anything the notes doesn't make sense please contact me (these notes were written by Dan: daniel.navarro@adelaide.edu.au) and I'll try to fix them!

What is the Chinese restaurant process?

Reduced to the simplest possible description, the Chinese restaurant process (CRP) gives us a distribution over partitions. Suppose that we have a collection of observations, and we want to cluster/partition them into groups. We can do this using the CRP. The CRP gets its name from a metaphor based on Chinese restaurants in San Francisco that seem to have limitless seating capacity. In this metaphor, every possible group corresponds to a "table" in an infinitely large Chinese restaurant. Each observation corresponds to a "customer" entering the restaurant and sitting at a table. In this metaphor, the customers are assumed to prefer sitting at popular tables, but nevertheless there is always a non-zero probability that a new customer will sit at a currently unoccupied table. To see how this works, suppose that there are currently N customers sitting in the restaurant, and let z_i be an indicator variable that tells us which table the ith customer is sitting at. Thus we have a vector of "table assignments", $\mathbf{z} = (z_1, z_2, \ldots, z_N)$. For instance if the table assignments for $N = 6$ customers were given by $\mathbf{z} = (1, 1, 2, 1, 3, 4)$, then there would be three customers sitting at table 1 (i.e., customers number 1, 2 and 4), and one customer at table 2 (i.e., customer 3), table 3 (i.e., customer 5) and table 4 (i.e., customer 6). The actual table numbers don't mean anything: they're just convenient indexing variables. That is, $\mathbf{z} = (1, 1, 2, 1, 3, 4)$ and $\mathbf{z} = (2, 2, 1, 2, 4, 3)$ are effectively equivalent. Next, let n_k denote the number of people sitting at the kth table, and let K denote the total number of non-empty tables. Then the "counts" vector $\mathbf{n} = (n_1, \ldots, n_K)$ tells us how many people are at each table. To stick with our running example, if the table assignment vector is $\mathbf{z} = (1, 1, 2, 1, 3, 4)$, then the count vector would be $\mathbf{n} = (3, 1, 1, 1)$. Note that $\sum_{k=1}^{K} n_k = N$. Now that we've introduced this terminology, it is very simple to describe the CRP. If there are currently N customers sitting in the restaurant, then the probability that customer $N + 1$ sits at the kth table is

proportional to the popularity of that table:

$$P(z_{N+1} = k \,|\, \mathbf{n}, \alpha) = \frac{n_k}{N + \alpha} \tag{1}$$

where α is called the "concentration parameter" or the "dispersion parameter" of the CRP. Notice however, that there is some probability mass missing, since if we sum Equation 1 over all K tables, it only adds up to $\frac{N}{N+\alpha}$. The reason for this is that there is some probability that customer $N + 1$ decides to sit at a new table. If we decide to label this new table as table $K + 1$, then

$$P(z_{N+1} = K + 1 \,|\, \mathbf{n}, \alpha) = \frac{\alpha}{N + \alpha}. \tag{2}$$

Taken together, Equations 1 and 2 provide a characterisation of the Chinese restaurant process. The core idea behind the CRP is really that simple.

The probability of a particular set of assignments $\mathbf{z}$ (with corresponding count vector $\mathbf{n}$) for a CRP with concentration parameter α is as follows:

$$P(\mathbf{z}|\alpha, N) = \frac{\Gamma(\alpha) \prod_{k=1}^{K} \Gamma(n_k)}{\Gamma(N + \alpha)} \alpha^K \tag{3}$$

Using the properties of the gamma function (see the note on beta-binomial models if you've forgotten about gamma functions), you can usually simplify this to something easier to compute. However, there is very rarely any need to do this, since Equation 3 is almost never used in practice. Almost every algorithm in which you might be interested will use the conditional probabilities in Equations 1 and 2. For instance, suppose you wanted to sample a set of assignments from the CRP. That is, generate

$$\mathbf{z}|N, \alpha \sim \text{CRP}(\alpha) \tag{4}$$

The simplest way to do so is to loop through the observations, making assignments using the conditional probability distributions. MATLAB code for doing this is given in Figure 1. This code samples partitions from the distribution described in Equation 3, but uses the conditional probabilities in Equations 1 and 2 to do so.

Why use the CRP?

The main attraction to the CRP is that it lets you define a "clumpy" distribution over partitions, without needing to specify in advance how many groups there are. This is rather useful. To illustrate the idea, here's a simple example:

> *Suppose we asked one hundred people which number was the most unlucky. Of those people, fifty said '13', forty said '4', and ten said '87'. These individual differences are unlikely to be caused by chance. Instead, what we appear to have are three distinct groups of people. How should we model these data?*

Let's consider a few possible approaches:

```
function [assignments,counts] = crprand(alpha,N)

% initialise everything
assignments = zeros(N,1); % assignments for each object
counts = zeros(N,1); % table counts (N is the max possible K)
assignments(1) = 1; % assign object 1 to table 1
counts(1) = 1; % adjust counts
counts(2) = alpha; % "fake" counts for table K+1
K = 1; % number of unique clusters

% sequentially assign other objects via CRP
for i = 2:N

    % generate random number, and convert to a "quasi-count"
    u = rand; % generate uniform random number
    u = u * (i - 1 + alpha); % multiply by the CRP normalising contant

    % find the corresponding table
    z = 1; % indexing variable for the table
    while u > counts(z)
        u = u - counts(z); % subtract off that probability mass
        z = z + 1; % move to the next table
    end

    % record the outcome and adjust
    assignments(i) = z; % make the assignment
    if t == K+1 % if it's a new table
        counts(z) = 1; % assign real count
        counts(z+1) = alpha; % move the "fake" counts to next table
        K = K+1; % update the number of clusters
    else % if it's an old table
        counts(z) = counts(z) + 1; % increment count
    end
end

% truncate the counts matrix for neatness. also, this takes
% care of the "fake" count mass in count(K+1)
counts = counts(1:K);
```

Figure 1. MATLAB code that generates a random partition of N objects, from a CRP with concentration parameter α.

1. **All partitions equally likely**. What happens if we remove the clumpiness property, and specify a uniform distribution over partitions? The number of ways of dividing N objects into groups is given by the Nth Bell number, so this distribution is not too hard to build. The main problem is that this approach makes the prediction that participant number 101 is just as likely to say 91 as 13. Clumpiness is critical – it tells you that new observations are more likely to be similar to old ones!

2. **Fixed number of groups**. What happens if we specify a fixed number of groups, and use a K-means clustering algorithm or some similar method? Besides the obvious problem that it you need to know the value of K, approach 2 has the exact opposite problem to approach 1. Specifically, it makes no allowance for the unlikely-but-possible situation in which participant 101 really does say 91.

3. **Prior over the number of groups**. Okay, why not go for the best of both worlds? Specify some sensible distribution over the value of K, and then say that all partitions of N items into exactly K groups are equally likely. That way, it's still *possible* for participant 101 to choose 91 as the unlucky number, but we can choose a prior over K so that it's more likely that he or she will say 13, 4 or 87. However, this does have a problem: it implies that participant 101 is just as likely to say 13 than 87, even though 87 was much more common.

4. **Model the probabilities associated with each group**. One sneaky idea might be to try something like this: specify a prior over K, and assign some probability θ_k that any given observation falls within the kth group. Then we could specify a prior over the θ_k values, and integrate them out (very much like what Dan did with the particle filtering version of the AI survey problem). This should mean that new observations are more likely to be assigned to groups that already have a lot of objects assigned to them; but still allows the value of K to grow as new observations arrive. It sounds like a lot of work, but it would solve all of our problems. Actually, this is a *very* good idea. Better yet, someone has already gone out and done all the hard work for you, and constructed a distribution that does something very similar (though not exactly equivalent): it's called the Chinese restaurant process!

The point behind this discussion is that the CRP is a way of specifying a prior over partitions that satisfies two critical requirements: (1) the number of groups K can grow as the number of observations N increases, and (2) new observations are more likely to belong to those groups that were "popular" among the old observations. To oversimplify a bit: *the CRP assumes that new observations are probably going to be similar to old ones, but they might be different.* This qualitative principle is very easy to state, and the CRP lets you implement it in a very simple fashion, but a lot of other "obvious" models don't.

What's all this "infinite models" stuff?

[NOTE: You're <u>not</u> expected to know the material in this section, and it's not examinable. We've only included it because it's useful background material]

If you do go on to read any of the literature on the CRP, you'll very quickly find that people talk about the CRP as if it were an "infinite model", and they seem to use the following terms as if they mean the same thing:

- Chinese restaurant process (CRP)
- Dirichlet process (DP)
- Pólya urn scheme
- Stick-breaking process

Beware! These are *not* the same thing, but they are very similar. As a consequence, a lot of academic papers are actually very unclear about which of these four things they mean, and often use the wrong terms. In this section, we'll try to clear up what each of these terms means, and explain what the "infinite models" terminology is all about. However, we won't go into a lot of technical details, since it gets messy very quickly, and is beyond the scope of this subject.

The place to start is with the "infinite models" terminology. To begin with, notice that Equation 4 refers to "CRP(α)" and not to "CRP(α, N)". This might strike you as strange: since N is an input into the function in Figure 1, why is it not a parameter of the CRP. The reason for this odd notation is that we've actually been a little imprecise about what the CRP really is. Without going into technical details, the CRP is actually a *process*, not a single distribution. That is, the CRP with parameter α actually defines a *sequence* of distributions, for all $N = 1, 2, \ldots, \infty$. Viewed from that perspective, N isn't a parameter of the CRP at all, because the CRP encompasses all possible values of N. This might seem like pointless nitpicking, since in practice we need to specify the value of N in order to be able to use the CRP, but it is closely related to how the CRP is actually derived. Again, we won't go into details, but the earlier description under the "modelling the probabilities..." paragraph is a little bit misleading. The CRP doesn't actually specify a prior distribution over K. Rather, the CRP is constructed by taking a very particular limit as $N \to \infty$, $K \to \infty$. That is, it commits to the idea that there really is an infinitely large partition "out there", but since you only ever see a finite number of observations, you only ever observe a tiny, finite part of that partition. So when we choose $N = 10$ to specify one of the distributions encompassed by the CRP(α) process, what we're really doing is asking something like "if there's really a latent infinite partition out there, how much of it would we expect to observe if we only have 10 data points?" That's why people refer to the CRP as if it defines a distribution over "infinite partitions". There is genuinely a sense in which it does exactly that: it's just that you only ever see a finite part of that partition.

The next thing to do is explain the relationship between the CRP and these other terms ("Dirichlet process", etc). Again, we won't include any of the maths and we won't go into much detail. However, here's the quick summary:

• **Chinese restaurant process**. As we've described, the CRP defines a distribution over a *partition*. If we let $\mathbf{z}_\infty$ denote the infinite vector of assignments that you would obtain if you followed the CRP until $N \to \infty$, then we would say that $\mathbf{z}_\infty | \alpha \sim \text{CRP}(\alpha)$. Since we've already described the CRP in detail, there's not much else to say.

• **Stick-breaking process**. To explain the distinction between the stick-breaking process and the CRP, let's suppose that $\mathbf{z}_\infty | \alpha \sim \text{CRP}(\alpha)$, and (if you'll pardon the slight abuse of notation) let $\theta_k = n_k / N$ denote the *proportion* of this infinite collection of observations that belong to the kth group. Since $K \to \infty$, there are actually an infinite number of θ_k values, so we can talk about the infinite vector $\boldsymbol{\theta}_\infty$. The stick breaking process works much like the CRP, in that it is a process that encompasses all $K = 1, 2, \ldots, \infty$, and as $K \to \infty$ we can say that $\boldsymbol{\theta}_\infty \sim \text{Stick}(\alpha)$. We won't go into details, but the basic reason for calling this a "stick breaking" process is by way of the following metaphor: imagine starting with a stick of length 1. You generate θ_1 by snapping the stick into two pieces. The length of one of the two pieces becomes the θ_1 value. To generate θ_2, you then snap the other piece in two; one of those two pieces becomes the value of θ_2. As you repeat this process, the length of the stick that you have left approaches zero. A little more formally, if the length of the stick at iteration k is ℓ_k, then you can generate θ_k by sampling $x \sim \text{Beta}(1, \alpha)$, setting $\theta_k = x\ell_k$, and setting $\ell_{k+1} = (1 - x)\ell_k$.

• **Pólya urn scheme**. In the CRP, we explicitly stated that the "table numbers" were meaningless (and in the CRP they are). However, suppose that, instead of being seated at a table, every customer in the restaurant is assigned a "colour", and this colour *is*

meaningful. So the ith customer is assigned colour c_i, and we assume that every customer who *would* have been seated at the same table (had we been following a CRP) is assigned the same colour. This is the Pólya urn scheme. Like the CRP, it is defined in terms of a conditional distribution $c_i|c_1, \ldots, c_{i-1}$. However, since the "colours" are supposed to correspond to meaningful parameters rather than arbitrary indexing variables, they are generated from some distribution, usually denoted G_0. When the Pólya urn scheme is defined on its own terms, the description usually goes like this: in a Pólya urn scheme, we imagine an urn full of α coloured balls, such that the proportion of balls with colour c is equal to $G_0(c)$ where G_0 is some probability distribution that is called the "base distribution". We sample the first colour c_1 by drawing a ball at random from the urn and recording its colour. We then return that ball to the urn and drop in *another* ball of the same colour, "updating" the urn.

• **Dirichlet process**. The Dirichlet process (DP) is the most abstract of the lot, but is the most general. It encompasses all of the above. Historically, the DP was invented first, and the other things were all invented to provide tractable methods for using the DP. The gist of the idea is this. Suppose we have a collection of N data points, $\mathbf{c}_N = (c_1, \ldots, c_N)$, all of which have been independently generated from some generic (but unknown) distribution G. The Dirichlet process is a way of defining a prior over generic distributions, G. Our prior over G says that the most likely possibility is that the distribution is G_0, but we're not very confident about this prediction. Let's let α denote our confidence: small α means we're not very confident so our prior is dispersed over lots of possible distributions G, and large α means that the possible distributions are very concentrated around G_0. So we have:

$$c_i|G \sim G \tag{5}$$

$$G|G_0, \alpha \sim \mathrm{DP}(G_0, \alpha) \tag{6}$$

The key thing here is that G is generated in a manner that is consistent with the CRP, the stick breaking process and the Pólya urn scheme. For instance, one way to sample $\mathbf{c}_N$ would be to just generate the values using the Pólya urn. An alternative but equivalent method would be to first sample $\mathbf{z}_N$ from the CRP. Then, every customer at the kth table is given the same colour, and that colour is independently sampled from G_0. Finally, you could (inefficiently) construct the entire distribution G by sampling $\boldsymbol{\theta}_\infty$ using the stick breaking process, and associating each of the θ_k values with a colour (sampled from G_0). The probability that the ith customer is assigned the kth such colour is θ_k. The basic point here is that the DP describes a "distribution over distributions", and it does so in a way that is consistent with all of the simpler things described above. Note, though, that the DP can only generate *discrete* distributions. It can't be used as a prior over continuous distributions, and you get some very strange results if you try to use it that way.