考虑参数$\theta = [\theta_1, \theta_2, \cdots, \theta_K]$和分布$P\{X = k\} = \theta_k$

狄利克雷（Dirichlet）分布：$p(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{i=k}^{K} \theta_k^{\alpha_k - 1}$，其中$\mathrm{B}(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)}, \quad \alpha = (\alpha_1, \ldots, \alpha_K)$

考虑样本$D = [x_1, x_2, \cdots, x_N]$，定义$n_k = \sum_{n=1}^{N} \mathbb{I}\{x_n = k\}$，似然函数为$p(D|\theta) = \prod_{k=1}^{K} \theta_k^{n_k}$

先验分布为狄利克雷分布$p(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_{i=k}^{K} \theta_k^{\alpha_k - 1} \sim Dir(\alpha)$，则后验分布为

$$p(\theta|D, \alpha) \propto p(D|\theta)p(\theta|\alpha) = \prod_{i=k}^{K} \theta_k^{\alpha_k + n_k - 1} \sim Dir(\alpha + n)$$

> 结论1：若先验分布为$Dir(\alpha)$，则后验分布为$Dir(\alpha + n)$
> 结论2：若先验分布为均匀分布（$Dir(0)$），则后验分布为$Dir(n)$
> 结论3：在收到一组新数据D后，n_k越多，对应θ_k更高的可能性变得更高
> 结论4：当α所有元素相同时，α越大，越倾向于生成均匀分布（指参数更均匀相当于进来得数据都是均匀的，那么样本量越大，我们就越确定是均匀的）

> 几个分布的关系：
> Beta分布是二项分布的共轭分布
> Dirichlet分布是多项函数的共轭分布
> Dirichlet分布是Beta分布的推广
> 高斯分布是自身的共轭分布

贝叶斯统计中，如果后验分布与先验分布属于同类，则先验分布与后验分布被称为**共轭分布**，而先验分布被称为似然函数的**共轭先验**（Conjugate prior）（来自[维基：共轭先验]）

> 假设先验分布是共轭先验的好处是形成先验链，当前的后验分布可以作为下一次计算的先验分布
> 均匀分布在经过一个先验链之后几乎可以模拟任何分布

> 先验分布：$p(\theta|\alpha)$
> 后验分布：$p(\theta|D, \alpha)$
> 先后是指取样的之前还是取样之后

Dirichlet分布构造Non-iid

一个用Dirichlet分布切分数据集的方式：FedMA代码。α大的时候，数据集容易被平分，α小的时候，切分差异比较大：α小可以产生更加Non-iid的数据

> 公式理解：当二维的情况下，Dirichlet分布即为Beta分布
>
> $$\mathrm{Var}(X) = \mathrm{E}(X-\mu)^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{1}$$
> $$= \frac{1}{4(2\alpha+1)} \tag{2}$$
>
> α小，方差大

```
elif partition == "hetero-dir":
    min_size = 0
    K = 10
    N = y_train.shape[0]
    net_dataidx_map = {}

    while min_size < 10:
        idx_batch = [[] for _ in range(n_nets)]
        # for each class in the dataset
        for k in range(K):
            idx_k = np.where(y_train == k)[0]
            np.random.shuffle(idx_k)
            proportions = np.random.dirichlet(np.repeat(alpha, n_nets))
            ## Balance
            proportions = np.array([p*(len(idx_j)<N/n_nets) for p,idx_j in
            proportions = proportions/proportions.sum()
            proportions = (np.cumsum(proportions)*len(idx_k)).astype(int)[:
            idx_batch = [idx_j + idx.tolist() for idx_j,idx in zip(idx_batc
            min_size = min([len(idx_j) for idx_j in idx_batch])

    for j in range(n_nets):
        np.random.shuffle(idx_batch[j])
        net_dataidx_map[j] = idx_batch[j]
```