DeepDPM: Deep Clustering With an Unknown Number of Clusters

摘要

DeepDPM: 聚类数未知的深度聚类。深度学习 (DL) 在无监督聚类任务中显示了巨大的前景。虽然如此，在经典 (非深度) 聚类中，非参数方法的好处是众所周知的，但大多数深度聚类方法都是参数化的：需要预定义和固定的聚类数量，用 K 表示。当 K 未知时，使用模型选择标准来选择其最优值，计算成本可能会变得很高，尤其是在深度学习中，因为必须多次重复训练过程。该论文提出一种有效的深度聚类方法，弥补了这一差距，不需要知道 K 的值，可以在学习过程中推断出 K 的值。通过使用分裂/合并框架、自适应变化的 K 的动态结构和新的损失，所提出的方法优于现有的非参数方法 (包括经典的和深度的)。虽然现有的极少数深度非参数方法缺乏可扩展性，但该论文通过首次报告该方法在 ImageNet 上的表现来进行了证明。还证明了推断 K 值的重要性，显示了当假设的 K 值与真实的 K 值相差甚远时，固定 K 值的方法的性能如何恶化，特别是在不平衡数据集上。

关键词： 非参数聚类；无监督学习；非参贝叶斯

1 引言

聚类是一种重要的无监督任务，与有监督的分类不同，类的标签是不可用的。此外，在纯无监督（更现实）的情况下，类的数量，用 K 表示，以及它们的相对大小（即类的权重）也是未知的。深度学习（DL）的出现并没有跳过聚类任务。DL 方法通常比经典（即非深度）聚类方法更好、更有效地聚类大型高维数据集。也就是说在经典聚类中，非参数方法（即找到 K 的方法）比参数方法（需要给定 K 的方法）具有优势。目前，非参数深度聚类方法很少，不幸的是，后者既不能扩展，也不够有效。[7] 提出了一种有效的深度非参数聚类方法 DeepDPM，来弥补这一差距。事实上，即使 K 已知，DeepDPM 也能与领先的参数方法相媲美。

更一般的说，推断潜在 K 的能力具有实际的好处。1）如果没有很好的 K 估计，参数方法的性能可能会收到影响。2）在训练过程中改变 K 值对优化有正向影响。3）对未知 K 的常见解决方法是使用模型选择：即，多次运行参数方法，在大范围内使用不同的 K 值，然后通过无监督标准选择“最佳”K。然而，这种方法除了错过了潜在的收益，还不能扩展，对于大型数据集来说通常是不可行的，尤其在深度学习中。

以 Dirichlet 过程混合模型（DPM）为例的非参贝叶斯（BNP）混合模型，为 K 未知时的聚类提供了一种优雅的、数据自适应的、数学的解决方案。然而，与 DPM 推断相关的高计算成本是为什么只有少数工作试图将其与深度聚类结合使用的原因。在这里，DeepDPM 建

议将 DL 和 DPM 的优点有效地结合起来。提出的方法 DeepDPM 使用簇的拆分与合并来改变 K，并使用动态架构来适应这种变化。它还为混合模型中的期望最大化 (EM) 算法使用了一种新的平摊推理。DeepDPM 可以合并到依赖于聚类的深度管道中 (例如，用于特征学习)。

2 相关工作

2.1 参数深度聚类方法

最近的此类研究可以分为两种类型：两部方法和端到端方法。在前者中，聚类作用在预先提取了特征的任务中。一个例子是 SCAN [8], 它使用无监督预训练的特征提取器。当达到 SOTA 结果时，有参数的 SCAN 依赖对 K 的估计，且当对 K 的估计过于不准确时，性能会恶化。此外，SCAN 假设统一的类权重（即平衡的数据集），这在纯粹无监督的情况下通常是不现实的。

端到端深度方法可能通过交替来同时学习特征和聚类。许多工作采用自编码器 (AE)，或者变分自编码器 (VAE)，有着额外的聚类损失。例如，DCN [9] 对预训练的 AE 的嵌入运行 K-means，并使用由重建项和基于聚类的项组成的损失对其进行重新训练，以同时更新特征、聚类中心和分配。其他作品，如 [2]，使用卷积神经网络交替学习特征和聚类。

2.2 非参数经典聚类方法

与我们工作相近的是 BNP 聚类，更准确的说是 DPM 模型 [1]。许多计算机视觉工作基于 BNP 聚类, 但这并没有成为主流选择，部分原因是缺乏高效的大规模推理工具。抽样的一个重要替代方法是变分 DPM 推理。一种流行的非参数方法的非贝叶斯例子是 DBSCAN [4]，它基于密度并将紧密排列的点分组在一起。虽然 DBSCAN 具有高效的实现，但它对难以调优的超参数非常敏感。

3 本文方法

DeepDPM 可以看作是一种 DPM 推理算法。我们使用拆分和合并来改变 K，其中我们为每个集群维护一个子集群对。对于 K 的标称值，，我们在混合模型中使用由 EM 的新型平摊推理训练的深度网络。DeepDPM 有两个主要部分。第一个是聚类网络，而第二个由 K 个子聚类网络组成 (每个聚类 K 一个，$k \in \{1, ..., K\}$)。图1描绘了整个流程。

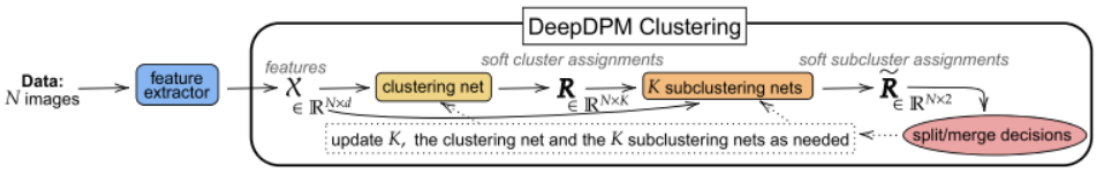

图 1. 方法示意图

3.1 基于 DPGMM 的聚类

令 $\mathcal{X} = (x_i)_{i=1}^N$ 表示在集合 $\mathbb{R}^d$ 中的 N 个数据点。聚类任务的目标是将 $\mathcal{X}$ 划分为 K 个不相交的组，其中 z_i 是 x_i 的点到簇的分配，称为簇标号。簇 k 由所有标记为 k 的点组成。经典的高斯混合模型（GMM）有一个 BNP 扩展：Dirichlet 过程 GMM（DPGMM）[4]。非正式地，DPGMM 包含了无限多个高斯函数的混合函数概念：

$$p(\boldsymbol{x}|(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)_{k=1}^{\infty}) = \sum_{k=1}^{\infty} \pi_k \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

其中 $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ 是一个高斯概率密度函数（pdf）(平均 $\boldsymbol{\mu}_k \in \mathbb{R}^d$ 和 $d \times d$ 的协方差矩阵 $\boldsymbol{\Sigma}_k$) ,$\boldsymbol{x} \in \mathbb{R}^d$，$\pi_k > 0 \forall k$，以及 $\sum_{k=1}^{\infty} \pi_k = 1$。设 $\boldsymbol{\theta_k} = (\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k)$ 为第 k 个高斯的参数。

假设分量 $\boldsymbol{\theta} = (\boldsymbol{\theta}_k)_{k=1}^{\infty}$ 和权重 $\boldsymbol{\pi} = (\boldsymbol{\pi}_k)_{k=1}^{\infty}$ 是 (独立地) 从它们自己的先验分布中得到的: 权重 $\boldsymbol{\pi}$ 是使用浓度参数 $\alpha > 0$ 的 Griffiths-Engen-Mcloskey 断棒过程 (GEM) [6] 得到的，而参数 $(\boldsymbol{\theta}_k)_{k=1}^{\infty}$ 是独立同分布的 (i.i.d) 从它们的先验 $p(\boldsymbol{\theta}_k)$ 中得到的，通常是正态逆威沙特 (NIW) 分布。虽然有无限多的分量，但仍然有有限多的聚类，因为潜在随机变量 K 的上界是 N。通过可能的重命名聚类指标，我们可以假设 $\{k : k \in \boldsymbol{z}\} = \{1, 2, ..., K\}$。

DPGMM 常用于 K 未知时的聚类。DPGMM 推理方法通常寻求 $\boldsymbol{z} = (z_i)_{i=1}^N$（这隐含着 K) 以及 $(\boldsymbol{\theta}_k, \pi_k)_{k=1}^K$。$K$ 的推断值受到以下因素的影响：$\mathcal{X}$, α, 以及 NIW 的超参数。DeepDPM 受到 Chang and Fisher 的 DPM 采样器 [3] 的启发，该采样器由拆分/合并框架和受限采样器组成。

拆分/合并框架用辅助变量增加潜在变量 $\boldsymbol{\theta} = (\boldsymbol{\theta}_k)_{k=1}^{\infty}, \boldsymbol{\pi}$, 以及 $(z_i)_{i=1}^N$。对于每个 z_i, 添加一个额外的子簇标签 $\widetilde{z}_i \in \{1, 2\}$。对每个 $\boldsymbol{\theta}_k$ 添加两个子分量 $\widetilde{\boldsymbol{\theta}}_{k,1}, \widetilde{\boldsymbol{\theta}}_{k,2}$, 非负权重 $\widetilde{\boldsymbol{\pi}}_k = (\widetilde{\pi}_{k,j})_{j \in \{1,2\}}$ (其中 $\widetilde{\pi}_{k,1} + \widetilde{\pi}_{k,2} = 1$) 这构成了一个二分量的 GMM。接着，拆分和合并可以通过 Metropolis-Hastings 框架改变 K [5]。即在推理过程中，每进行一定次数的迭代，提出将簇 k 拆分为其子簇。该拆分被接受的概率为 $\min(1, H_s)$，其中

$$H_s = \frac{\alpha \Gamma(N_{k,1}) f_{\boldsymbol{x}}(\mathcal{X}_{k,1}; \lambda) \Gamma(N_{k,2}) f_{\boldsymbol{x}}(\mathcal{X}_{k,2}; \lambda)}{\Gamma(N_k) f_{\boldsymbol{x}}(\mathcal{X}_k; \lambda)}$$

是 Hastings 比率，Γ 是伽马函数，$\mathcal{X}_k = (\boldsymbol{x}_i)_{i:z_i=k}$ 表示簇 k 中的点，$N_k = |\mathcal{X}_k|$，$\mathcal{X}_{k,j} = (\boldsymbol{x}_i)_{i:(z_i,\widetilde{z}_i)=(k,j)}$ 表示子簇 $j (j \in \{1, 2\})$ 中的点，$N_{k,j} = |\mathcal{X}_{k,j}|$，$f_{\boldsymbol{x}}(\cdot; \lambda)$ 是边际似然，其中 λ 是 NIW 超参数。在分割提议接受后，每个新生成的簇都增加两个子簇。该比值 H_s 可以理解为将数据在两个子簇下的边际似然与其在该簇下的边际似然进行比较。

3.2 给定 K 值的 DeepDPM

给定当前值 K, 数据首先传递给聚类网络 f_{cl}, 它为每个数据点 x_i 生成 K 个软分配：

$$f_{\text{cl}}(\mathcal{X}) = \boldsymbol{R} = (\boldsymbol{r}_i)_{i=1}^N \qquad \boldsymbol{r_i} = (r_{i,k})_{k=1}^N$$

其中 $r_{i,k} \in [0, 1]$ 是 $\boldsymbol{x}_i$ 对簇 k 的软分配。从 $(\boldsymbol{r}_i)_{i=1}^N$ 我们令 $z_i = \arg\max_k r_{i,k}$ 计算出硬分配 $\boldsymbol{z} = (z_i)_{i=1}^N$。接下来，每个子聚类网络 f_{sub}^k（其中 $k \in \{1, ..., K\}$）接收划分为其各自簇的数据（即 f_{sub}^k 接收 $\mathcal{X}(\boldsymbol{x}_i)_{i:z_i=k}$），并生成软子簇分配：

$$f_{\text{sub}}^k(\mathcal{X}_k) = \widetilde{\boldsymbol{R}}_k = (\widetilde{\boldsymbol{r}}_i)_{i:z_i=k} \qquad \widetilde{\boldsymbol{r}}_i = (\widetilde{r}_{i,j})_{j=1}^2$$

其中 $\widetilde{r}_{i,j} \in [0,1]$。由子聚类网络学习的子簇用于拆分提议。所有 $K+1$ 个网络都是一个简单的多层感知机，具有单个隐藏层。f_{cl} 的最后一层有 K 个神经元，每个 f_{sub}^k 的最后一层有两个神经元。

现在，我们对贝叶斯 GMM 中引入一个由 EM 驱动的新损失。在每个 epoch 中，我们的聚类网络被优化以生成与 EM-GMM 算法的 E 步所得到的分配相似的软分配。对于每个 $\boldsymbol{x}_i$ 以及每个 $k \in \{1,...,K\}$，我们计算标准的 E 步概率，$\boldsymbol{r}_i^{\text{E}} = (r_{i,k}^{\text{E}})_{k=1}^K$, 其中

$$r_{i,k}^{\text{E}} = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \qquad k \in \{1,...,K\}$$

通过上一个 epoch 的 $(\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)_{k=1}^K$ 计算。然后我们使用如下损失来鼓励 f_{cl} 生成相似的软分配：

$$\mathcal{L}_{\text{cl}} = \sum_{i=1}^N \text{KL}(\boldsymbol{r}_i || \boldsymbol{r}_i^{\text{E}})$$

其中 KL 是 KL 散度。

对 $(f_{\text{sub}}^k)_{k=1}^K$ 可以使用与 $\mathcal{L}_{\text{cl}}$ 类似的损失。这里原文提出了一种各向同性损失：

$$\mathcal{L}_{\text{sub}} = \sum_{k=1}^K \sum_{i=1}^{N_k} \sum_{j=1}^2 \widetilde{r}_{i,j} \left\| \boldsymbol{x}_i - \widetilde{\mu}_{k,j} \right\|_{l_2}^2$$

其中 $N_k = |\mathcal{X}_k|$, $\widetilde{\mu}_{k,j}$ 是簇 k 的子簇 j 的均值。上面描述的迭代过程需要初始化，采用 K-means 对簇进行初始化。

3.3 通过拆分和合并来改变 K

在训练过程中，使用分割和合并来改变 K。每隔几个 epoch，我们就提出分裂或合并的建议。由于 k 在变化，网络结构，即聚类网络的最后一层和子聚类网络的数量也必须改变。

在每个分裂步骤中，我们建议将每个簇分为两个子簇。一个拆分方案以概率 $\min(1, H_s)$ 随机接受。为了适应 K 的增加，如果对簇 K 接受分割建议，则将聚类网络最后一层的第 K 个单元及其连接到前一隐藏层的权重复制，并使用子聚类网络学习的参数初始化两个新聚类的参数：

$$\boldsymbol{\mu}_{k_1} \leftarrow \widetilde{\boldsymbol{\mu}}_{k,1}, \quad \boldsymbol{\Sigma}_{k_1} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,1}, \quad \pi_{k_1} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,1}$$
$$\boldsymbol{\mu}_{k_2} \leftarrow \widetilde{\boldsymbol{\mu}}_{k,2}, \quad \boldsymbol{\Sigma}_{k_2} \leftarrow \widetilde{\boldsymbol{\Sigma}}_{k,2}, \quad \pi_{k_2} \leftarrow \pi_k \times \widetilde{\boldsymbol{\pi}}_{k,2}$$

然后，我们还向每个新聚类添加一个新的子聚类网络。

在考虑合并时，我们必须确保我们永远不会同时接受如，合并集群 k_1 和 k_2 以及合并集群 k_2 和 k_3，从而错误地将三个集群合并在一起。因此，与并行进行的拆分提议不同，并非所有可能的合并都可以同时考虑。为了避免顺序考虑所有可能的合并，我们 (顺序地) 考虑每个集群仅与其 3 个最近邻居的合并。合并建议的接受/拒绝使用 Hastings 比率，$H_m = 1/H_s$。如果一个提议被接受，两个集群将被合并，并初始化一个新的子集群网络。从技术上讲，从 f_{cl} 中删除合并后的聚类的最后一层的单元，以及将其连接到前一个隐藏层的网络的权重，并使用加权 MAP 估计初始化新聚类的参数和权重。

4 复现细节

4.1 与已有开源代码对比

本次论文复现基于 DeepDPM 原文所提供的开源代码。新添了可视化聚类结果功能:

```
def save_cluster_examples(args,predict,x_for_vis,
                          labels,num_img,grid_size):

    def save_image(
    tensor,
    fp,
    ) -> None:

        grid = make_grid(tensor)
        ndarr = (grid.clamp_(0, 255).permute(1, 2, 0)
                 .to("cpu", torch.uint8).numpy())
        im = Image.fromarray(ndarr)
        im.save(fp)

    if not os.path.exists(f"./{args.dataset}_imgs/"):
        os.mkdir(f"./{args.dataset}_imgs/")
    count=0
    for k in np.unique(predict):
        count+=1
        x_k = x_for_vis[predict == k][:num_img]
        y_gt = labels[predict == k][:num_img]
        if not os.path.exists(f"./{args.dataset}_imgs/{count}"):
            os.mkdir(f"./{args.dataset}_imgs/{count}")

        for i in range(min(num_img, x_k.shape[0])):
            save_image(x_k[i], f"{args.dataset}_imgs/"
                               f"{count}/clusternet_clus"
                               f"{count}_label{y_gt[i]}_{i}.jpeg")
        # save as a grid
        num_imgs = min(grid_size, x_k.shape[0])
        if num_imgs > 0:
            grid = make_grid(x_k[:num_imgs], nrow=num_imgs)
            save_image(grid, f"{args.dataset}_imgs/"
                             f"{count}/clusternet_clus{count}.jpeg")
```

4.2 实验环境搭建

本次实验在个人电脑上运行。计算机采用 RTX4060 显卡，显存为 8G。cpu 使用英特尔 i5-12400f 处理器，物理内存 32G。运行操作系统为 win10，使用的编程语言为 python。采用 pytorch 库构建神经网络框架。

5 实验结果分析

本次复现我在 STL10、IMAGENET50,FASHION,USPS,MNIST 等数据集上测试算法。采用 acc、ari、nmi 三个指标评估聚类结果的好坏，这三个指标都是越接近 1 代表聚类效果越好。

从表 1 中可以看到 DeepDPM 在简单的数据集如 USPS、MNIST 能达到较好的效果，其中 MNIST 的效果好于原论文的结果。而在较为复杂的数据中，由于算力资源有限，并没有得到较好的效果。从图2中可以看到 DeepDPM 对 ImageNet 数据集聚类的效果，DeepDPM 成功的将鸟、车、马划分为不同的簇。

表 1 DeepDPM 在不同数据集上效果

数据集	acc	ari	nmi
STL10	0.7842	0.664718	0.776863
IMAGENET50	0.55601	0.470949	0.698714
FASHION	0.56851	0.477132	0.6599
USPS	0.80964	0.807141	0.864411
MNIST	0.97873	0.953716	0.941651

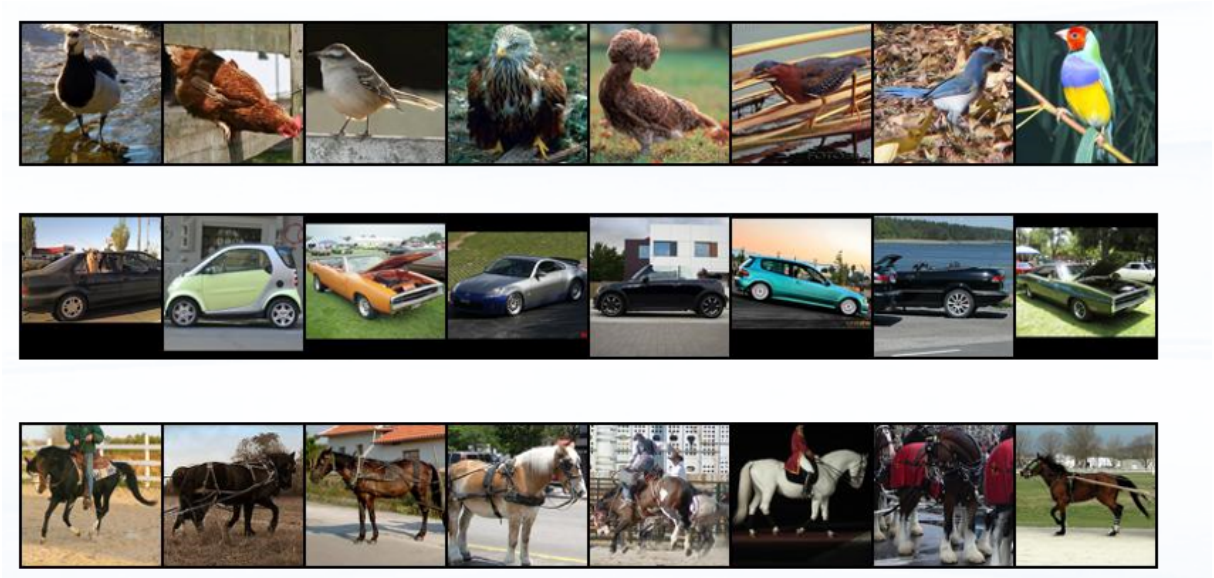

图 2. ImageNet 聚类结果示意

6 总结与展望

本次论文复现中我学习了 Ronen 等人提出的 DeepDPM 深度非参数聚类方法。该方法以 DPM 为理论依据，采用深度学习方法训练数据的划分，并采用拆分与合并的操作动态改变簇的数量，从而实现了非参数深度聚类的目标。该方法能很好的结合特征提取的过程，解决了先前深度聚类难以与其他深度学习目标相结合的痛点。

DeepDPM 的局限性在于，当输入的特征很差时，他将难以恢复。此外，如果簇数是已知的，且数据集是平衡的，那么非参数方法可能是更好的选择。

参考文献

[1] Charles E Antoniak. Mixtures of dirichlet processes with applications to bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174, 1974.

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[3] Jason Chang and John W Fisher III. Parallel sampling of dp mixture models using subcluster splits. *Advances in Neural Information Processing Systems*, 26, 2013.

[4] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[5] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[6] Jim Pitman. *Combinatorial stochastic processes: Ecole d'eté de probabilités de saint-flour xxxii-2002*. Springer, 2006.

[7] Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, pages 9861–9870, 2022.

[8] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pages 268–285. Springer, 2020.

[9] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.