机器学习期末项目报告

- 组员：梅祎航 房向南
- 选题：选题二 半监督的聚类问题

具体问题

该选题提供如下数据：

训练数据集：包含5万张png图片。
测试数据集：包含1万张png图片。
关系文件：一个csv文件，每行分别是图片id、图片id、是否相关。

通过提供的数据信息，这个问题可以被归类为半监督的聚类问题。根据一篇对于半监督聚类的综述：

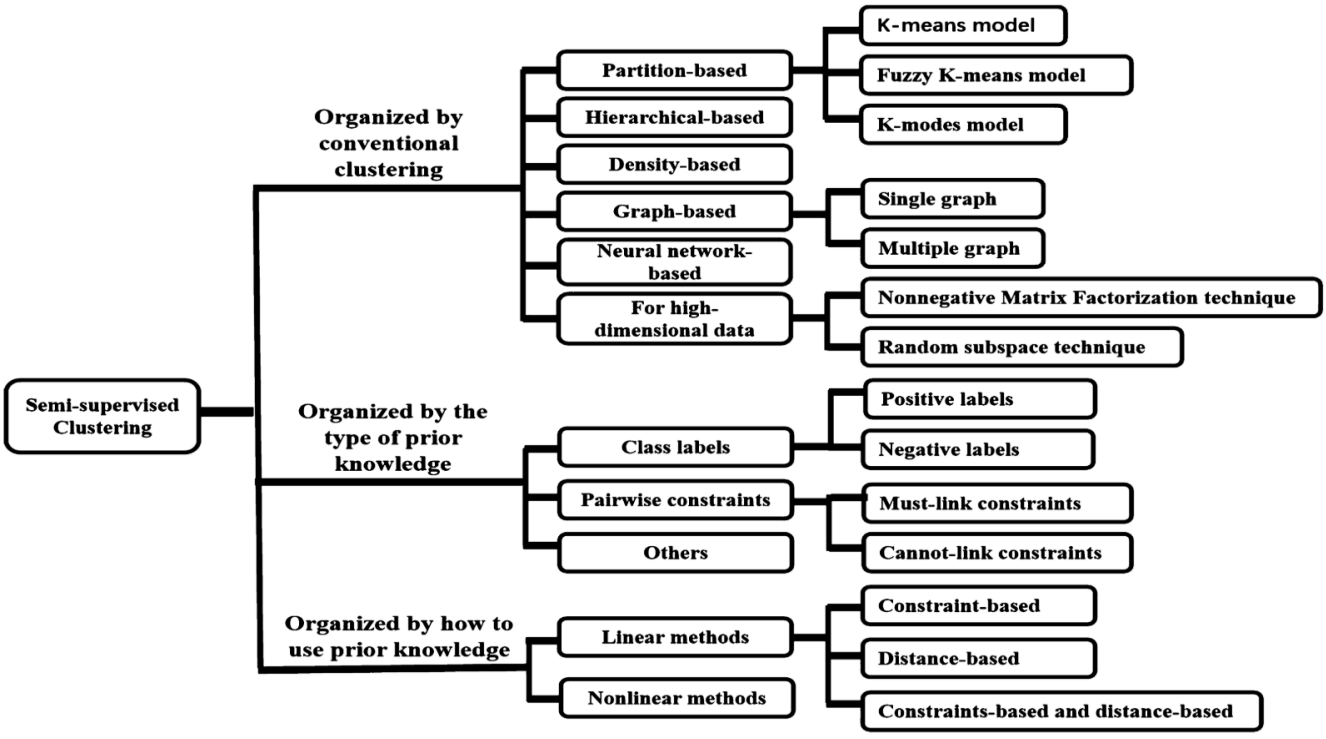

Fig. 2. Organizational structure analysis of semi-supervised clustering methods.

这个问题属于Organized by the type of piror knowledge中的pairwise constraints。也就是成对约束的半监督聚类问题。成对约束包含以下两种约束：

（1）必须链接约束：它表明一对数据必须分配到一个簇中；
（2）不能链接约束：它表明成对数据不能分配到一个簇中。

在该论文对于成对约束的问题描述中，更多的是将描述为生成成对约束从而利好聚类结果。我们这个问题更像这个语境下的一个下游任务。

经过初步调查研究我们决定使用先编码后聚类的思路。

模型的构建及数据的整理

在本次实验中，我们采用了一种结合Vision Transformer（ViT）和高斯混合模型（GMM）的半监督聚类方法。该方法的设计思路及数据整理过程如下：

设计思路

1. **特征提取**：
   - 我们使用了预训练的Vision Transformer（ViT）模型来提取图像特征。ViT模型能够捕捉图像中的复杂模式和高维特征，适合于处理高维度的图像数据。
   - 选择ViT模型的原因是其在各种计算机视觉任务中的出色表现，尤其是在图像分类和特征提取方面。
2. **数据预处理**：
   - 对输入图像进行统一的预处理，包括调整图像尺寸、归一化处理等，以确保输入到ViT模型中的数据格式一致。
   - 从训练集和测试集图像中提取特征，将其转换为特征向量，并保存到CSV文件中，以便后续处理。
3. **约束处理**：
   - 半监督聚类需要利用一些已知的标签信息。在本实验中，我们通过CSV文件提供了训练集样本之间的约束条件，包括must-link（同类）和cannot-link（异类）约束。
   - 将must-link约束样本合并处理，通过取其特征均值来减少类内差异。
   - 对于cannot-link约束样本，添加少量噪声以增加它们的特征差异，防止其在聚类过程中被错误地归为同一类。
4. **聚类方法选择**：
   - 选择高斯混合模型（GMM）作为聚类算法，原因在于GMM能够灵活地处理不同形状的分布，并且可以根据样本的特征分布情况自动调整模型参数。
   - 使用处理后的训练数据训练GMM模型，并对测试数据进行聚类预测。

数据整理

1. **图像预处理和特征提取**：
   - 图像通过预定义的预处理步骤（如尺寸调整、归一化）处理后，输入到ViT模型中进行特征提取。
   - 提取的特征向量保存为CSV文件，用于后续的聚类分析。
2. **训练数据和测试数据的处理**：
   - 将训练数据和测试数据分别处理，提取特征后保存，以确保两者在同一特征空间中进行聚类。
   - 通过读取训练集和测试集的图像，并按规定的顺序提取特征，确保数据的一致性和可复现性。
3. **约束条件的处理**：
   - 从CSV文件中加载约束条件，并分别处理must-link和cannot-link约束。
   - 对于must-link约束，合并同类样本并计算均值特征；对于cannot-link约束，添加噪声以增加特征差异。

通过以上设计思路和数据整理过程，我们构建了一个结合ViT和GMM的半监督聚类模型，并利用约束条件对聚类过程进行了优化，最终实现了对测试集图像的有效分类。

具体算法应用

ViT提取特征

在我们的实验中，Vision Transformer (ViT) 用于从图像中提取特征。这一步骤非常关键，因为特征的质量直接影响到后续的聚类效果。以下是ViT特征提取的详细过程和相应代码的逐步讲解。

设置设备和加载模型：

- 为了加速计算，我们使用了GPU进行训练。
- 加载一个预训练的ViT模型，这个模型已经在ImageNet数据集上进行训练，具有良好的特征提取能力。

```
import torch
from torchvision.models import vit_b_16

# 设置设备
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# 加载预训练的 ViT 模型
model = vit_b_16(pretrained=True).to(device)
model.eval()
```

图像预处理：

- 我们定义了一系列图像预处理步骤，包括调整图像尺寸、转换为张量和归一化处理。
- 需要强调的是，我们用于归一化的平均值和标准差是基于 ImageNet 数据集计算得出的，这可以将我们的数据在最大程度上契合预训练模型。

```
import torchvision.transforms as transforms

# 定义图像预处理
preprocess = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

特征提取函数：

- 定义一个函数 `extract_features` ，用于从单张图像中提取特征。
- 该函数读取图像并进行预处理，然后将图像传入ViT模型，提取特征并返回。

```
from PIL import Image

def extract_features(image_path):
    """提取图像特征"""
    image = Image.open(image_path).convert('RGB')
    image = preprocess(image).unsqueeze(0).to(device)
    with torch.no_grad():
        features = model(image)
    return features.cpu().numpy().flatten()
```

数据集处理和特征提取：

- 定义一个函数 `process_dataset` ，用于处理整个数据集并提取所有图像的特征。
- 该函数遍历图像目录中的所有图像，调用 `extract_features` 函数提取特征，并将特征保存到列表中。
- 最后将提取的特征保存到CSV文件中。

```
import os
import pandas as pd

def process_dataset(image_dir, prefix):
    """处理数据集并提取特征"""
    data = []
    image_files = sorted([f for f in os.listdir(image_dir) if f.startswith(prefix) and f.en
                          key=lambda x: int(x.split('_')[1].split('.')[0]))

    for image_name in image_files:
        image_id = int(image_name.split('_')[1].split('.')[0])
        image_path = os.path.join(image_dir, image_name)
        features = extract_features(image_path)
        data.append([image_id] + features.tolist())
        if image_id % 1000 == 0:
            print('name ', image_name)

    return data
```

处理训练集和测试集：

- 分别处理训练和测试数据。

```
# 设置图片目录
train_dir = 'train'
test_dir = 'test'

# 提取训练集特征
train_data = process_dataset(train_dir, 'train')
train_df = pd.DataFrame(train_data)
train_df.to_csv('train_features1.csv', index=False, header=False)

# 提取测试集特征
test_data = process_dataset(test_dir, 'test')
test_df = pd.DataFrame(test_data)
test_df.to_csv('test_features1.csv', index=False, header=False)
```

高斯混合模型(GMM)半监督聚类

在完成了特征提取之后，我们使用高斯混合模型（GMM）进行半监督聚类。

加载特征数据和约束条件：

- 从CSV文件中加载训练集和测试集的特征数据。
- 提取每个样本的ID和特征向量。
- 从CSV文件中加载约束条件，包含两个样本ID及其关系（1表示同类，-1表示不同类）。

```
import pandas as pd
import numpy as np

# 加载特征数据
train_features = pd.read_csv('train_features1.csv', header=None)
test_features = pd.read_csv('test_features1.csv', header=None)

# 提取ID和特征
train_ids = train_features.iloc[:, 0].values
train_data = train_features.iloc[:, 1:].values
test_ids = test_features.iloc[:, 0].values
test_data = test_features.iloc[:, 1:].values

# 加载约束条件
constraints = pd.read_csv('constraints.csv')
constraints.columns = ['id1', 'id2', 'label']
```

创建约束矩阵：

- 根据约束条件创建must-link和cannot-link矩阵，用于约束聚类过程。

```
from scipy.sparse import csr_matrix

# 创建约束矩阵
n_samples = train_data.shape[0]
must_link = constraints[constraints['label'] == 1]
cannot_link = constraints[constraints['label'] == -1]

# must-link约束
must_link_matrix = csr_matrix((np.ones(must_link.shape[0]), (must_link['id1'], must_link['i

# cannot-link约束
cannot_link_matrix = csr_matrix((np.ones(cannot_link.shape[0]), (cannot_link['id1'], cannot
```

处理must-link约束：

- 使用连接组件算法合并must-link约束的样本，并将它们的特征向量取平均值，重新生成新的特征矩阵。

```
from scipy.sparse.csgraph import connected_components

# 处理must-link约束，合并相连的组件
n_components, labels = connected_components(must_link_matrix, directed=False)
print(f"Number of connected components: {n_components}")

# 创建新的特征矩阵
print("Creating new train data with must-link constraints...")
new_train_data = train_data.copy()
for component in range(n_components):
    component_indices = np.where(labels == component)[0]
    if len(component_indices) > 1:
        mean_vector = np.mean(train_data[component_indices], axis=0)
        new_train_data[component_indices] = mean_vector
print("Must-link constraints processing complete.")
```

处理cannot-link约束：

- 对于距离过近的cannot-link约束样本，添加少量噪声以增加它们之间的距离，防止其在聚类过程中被错误地归为同一类。

```
from sklearn.metrics import pairwise_distances
import statistics

# 处理cannot-link约束（简单方法，通过添加噪声）
print("Processing cannot-link constraints...")
scale = 0.01 * np.std(new_train_data, axis=0)
distances = pairwise_distances(new_train_data)
discount = []
for i, row in cannot_link.iterrows():
    id1, id2 = int(row['id1']), int(row['id2'])
    discount.append(distances[id1, id2])
    if distances[id1, id2] < 27:  # 距离太近，添加噪声
        new_train_data[id1] += np.random.normal(0, 0.01, size=new_train_data[id1].shape)
        new_train_data[id2] += np.random.normal(0, 0.01, size=new_train_data[id2].shape)
print("Cannot-link constraints processing complete.")
```

训练高斯混合模型（GMM）：

- 使用处理后的训练数据训练GMM模型，并获取聚类标签。

```
from sklearn.mixture import GaussianMixture

# 定义聚类算法
n_clusters = 4
print("Training Gaussian Mixture Model (GMM)...")
gmm = GaussianMixture(n_components=n_clusters, random_state=42)

# 训练GMM模型
gmm.fit(new_train_data)
print("GMM training complete.")

# 获取训练数据的聚类标签
train_labels = gmm.predict(new_train_data)
print("Train labels predicted.")
```

预测测试数据的聚类标签：

- 使用训练好的GMM模型对测试数据进行聚类预测，并生成提交文件。

```
# 预测测试数据的聚类标签
test_labels = gmm.predict(test_data)
print("Test labels predicted.")

# 创建提交文件
submission = pd.DataFrame({'ID': test_ids, 'Class': test_labels})

# 确保按ID排序
submission = submission.sort_values(by='ID').reset_index(drop=True)
print("Submission file created and sorted by ID.")

# 保存为CSV文件
submission.to_csv('submission.csv', index=False)
print("Submission file saved as 'submission.csv'.")
```

算法的横向比较

这部分是顺着我们最初定下的思路，对我们曾经尝试过的算法和模型的横向比较。

卷积编码器+自组织映射：

我们在随堂汇报中对这部分做了较为详细的介绍，包括算法思路和优化过程。该模型的准确率低的原因有三：

1.自己定义的编码器结构过于简单，不能有效的提取图片特征。
2.设计的模型参数过多，在有限的时间内很难有较好的优化结果。
3.因为约束数据的全局性，该模型不能很好的分批次训练。

其他预训练模型+机器学习分类：

我们还尝试了用其他预训练模型的特征提取，并接着进行降维或者聚类。该方法不如用VIt提取特征的原因有三：

1.transformer架构的优秀特征提取能力
2.其他模型没有像用VIt那样的较高准确率提升，导致在尝试过后就被放弃，没有针对模型进行优化。
3.使用了不当的降维方式，丢失了图片特征。

深度聚类网络：

通过结合GAN或者VAE等深度学习的模型，深度聚类网络可以很好的进行无监督的图片聚类工作。但是对于该问题而言，深度聚类网络有两个问题：

1.难以实现：模型如GAN的代码较为复杂，难以在短时间内实现并优化。
2.不好结合约束数据：即使采用现成的代码，也不好在作者已经优化好的各项超参数之外添加一个成对

总结：通过我们最后采用的算法与其他算法的横向比较，可以清晰地看出**ViT+GMM**这个模型有如下优点：

1.代码量小，训练简单。
2.可以在GMM部分很好的结合约束条件进行训练。
3.可以充分利用图片特征进行训练。

所以这个模型在准确率上取得了最好的结果。

应用前景与可能的发展方向

ViT模型的实际应用

根据我们的调查，ViT模型自提出以来，已经在很多领域有了实际投入生产的应用。

1. Google 搜索和谷歌相册
   Google 利用 ViT 在其搜索和照片管理应用中进行图像分类和对象识别。这些应用通过 ViT 模型自动对用户上传的照片进行分类和标记，从而提高了搜索和整理照片的效率和准确性。

2. NVIDIA Clara 医疗平台
   NVIDIA 的 Clara 医疗平台利用 ViT 模型来进行医学图像分析，包括放射影像的分类和分割。ViT 在该平台上帮助医生更快速地识别疾病，并提供更精确的诊断信息。
3. Facebook (Meta)
   Meta在其计算机视觉应用中使用 ViT 模型进行内容理解和自动化内容管理。例如，ViT 可以用于图像审核、内容推荐等方面，提升平台的自动化处理能力和用户体验。
4. Adobe Photoshop
   Adobe 在其 Photoshop 中应用了 ViT 模型，用于图像编辑和增强功能。ViT 能够帮助识别和处理图像中的复杂结构，使用户能够更轻松地进行图像修复、风格迁移等操作。

还有一些自动驾驶、图像识别、金融、安防领域的应用。

本模型的应用前景

我们认为我们使用的模型可以应用于以下方面：

1. 推荐系统
   结合信息系统领域的知识，这个半监督的聚类模型可以在购物网站等的推荐系统方面起作用，其中约束数据可以从用户搜索完一个关键词后的点击动作中获取，或者可以通过衡量商品文本相似性来生成关系数据。
2. 异常检测
   在工业生产中，可以将之前的生产结果当成先验数据，训练模型对机器生产的零部件进行异常检测。
3. 生物信息学
   可以结合分子生物学领域的知识，在蛋白质、细胞的同类型图像识别上做出应用。

可能的发展方向

目前我们使用的模型的不足之处还有很多，比如准确率还是不高，模型的可解释性不强。我认为还能有以下的发展：

1. 可以研究ViT特征提取的特点，解决为何在cannot-link数据中添加噪声的界限会对结果造成较大的影响，探究其对每个指标（acc,nmi,ari）的影响。
2. 研究如何不通过预训练模型，即不引入额外信息的情况下，通过从头开始训练达到较好的结果。
3. 研究如何将模型与生产生活实际相结合，改进模型之后利用自己收集到的数据集进行实验。

参考文献

1 Cai, Jianghui, et al. *A review on semi-supervised clustering.* Information Sciences 632 (2023): 164-200.

2 Guérin, Joris, et al. *CNN features are also great at unsupervised classification.* arXiv preprint arXiv:1707.01700 (2017).

3 Dosovitskiy, Alexey, et al. *An image is worth 16x16 words: Transformers for image recognition at scale.* arXiv preprint arXiv:2010.11929 (2020).

难点与收获

难点：

1. 条件受限:在训练时间和训练资源有限的情况下，难以做到复杂模型的实现与训练。
2. 理想与现实:同时，我们也无法短时间内掌握使用深度学习框架如pytorch自如的实现一些想法，或者改造开源代码。
3. 模型选择与沉没成本：在某个方向上走了太久之后，想要轻易的更换模型已经是一件难事。
4. 问题本身：对于这个机器学习项目本身，难点主要在如何更好地提取到图片的特征，如何更好地利用约束数据进行半监督学习，如何在保证特征的情况下进行数据降维或者数据简化，从而保证结果有一定的准确率而不会有太高的错误率。

收获：

1. 了解到了很多半监督聚类的手段，通过阅读文献也了解到了这个机器学习问题的研究历程，应用前景与前沿的方法。
2. 巩固了从零到一编写机器学习模型的能力与灵活调用接口、化用别人的模型的能力。
3. 通过机器学习这个课程的期末项目，实际应用了上学习到的一些方法，比如数据降维可视化方法t-SNE。
4. 增进了对机器学习的了解与热爱。