研究报告

# 算法模型介绍

## 1.1算法

本研究旨在探索CIFAR-10数据集中图像间的关联关系，本文采用了Apriori算法作为关联规则挖掘工具。Apriori算法是一种经典的基于频繁项集的关联规则挖掘算法，它通过寻找频繁项集来推断出物品之间的关联规则，并利用支持度和置信度指标对规则进行评估。本研究首先提取了CIFAR-10训练集中1000张图片的特征向量，然后计算这些特征向量之间的余弦相似度，进而构建出关系矩阵。接着，应用Apriori算法从关系矩阵中挖掘出频繁项集和关联规则，并对挖掘结果进行评价和可视化。

## 1.2模型

为了更好地提取图像的特征，我们对ResNet50模型进行了一系列修改。具体来说，我们去除了模型的最后一层全连接层，并保留了前面的卷积和池化层。此外，在模型的第一层卷积操作中，我们使用了3个通道的输出，而不像原始的ResNet50模型中使用了64个通道的输出。这样做可以减少模型的计算量和参数量，并加速特征提取。最后，我们添加了一个自定义的卷积层，并将其作为ResNet50模型的第一层。这样可以进一步提取图像的特征，增强模型对数据集的适应性。

## 1.3功能

在功能方面，我们首先完成了CIFAR-10数据集的加载和预处理工作，包括对图片进行resize、转换为张量、归一化等处理。然后，我们从图片中提取特征向量，并计算两张图片之间的相似度。接着，我们将相似度矩阵转换为关系矩阵，并使用Apriori算法从中挖掘出频繁项集及其置信度、支持度等信息。最后，我们对挖掘得到的规则进行评价和可视化。

# 执行代码及数据集说明

## 2.1数据集说明

本研究使用Python语言编写代码，主要使用了PyTorch和Scikit-Learn等数据处理和机器学习库。CIFAR-10数据集是一个经典的计算机视觉数据集，包含了60000张32x32的彩色图像，分为10个类别：飞机、汽车、鸟、猫、鹿、狗、青蛙、马、船、卡车。其中50000张作为训练集，10000张作为测试集。本研究从训练集中随机选择了1000张图片进行关联规则挖掘。CIFAR-10数据集可以从官网下载：<http://www.cs.toronto.edu/~kriz/cifar.html>。

## 2.2执行代码

# 加载 CIFAR-10 训练集

trainset = datasets.CIFAR10(root='./data', train=True, download=True)

# 从 CIFAR-10 训练集中随机选择 10000 张图片的索引

train\_indices = random.sample(range(len(trainset)), 1000)

trainset.data = trainset.data[train\_indices]

print('trainset length:', len(trainset))

transform = transforms.Compose([

transforms.Resize((32, 32)), # 调整图像尺寸

transforms.ToTensor(), # 将图像转换为张量

transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)) # 归一化

])

trainset.transform = transform

# 重新定义模型

class ResNet50(nn.Module):

def \_\_init\_\_(self):

super(ResNet50, self).\_\_init\_\_()

self.conv = nn.Conv2d(3, 3, kernel\_size=3, stride=1, padding=1, bias=False)

self.resnet = models.resnet50(pretrained=True)

def forward(self, x):

x = self.conv(x)

x = self.resnet(x)

return x

resnet50 = ResNet50()

resnet50 = models.resnet50(pretrained=True)

modules = list(resnet50.children())[:-1]

resnet50 = torch.nn.Sequential(\*modules)

# # # 修改第一层卷积核大小

resnet50.conv1 = nn.Conv2d(3, 64, kernel\_size=3, stride=1, padding=1)

device = torch.device("cuda:0" if torch.cuda.is\_available() else "cpu")

print("Using device:", device)

resnet50.to(device)

def get\_features(dataset):

features = []

for i in tqdm(range(len(dataset))):

data, \_ = dataset[i]

data = data.to(device)

feature = resnet50(data.unsqueeze(0)).squeeze().detach().cpu().numpy()

feature = feature.flatten() # 展平每个特征向量

features.append(feature)

print(np.array(features).shape)

return features

print('提取训练集特征...')

train\_features = get\_features(trainset)

# train\_features = np.reshape(train\_features, (len(train\_features), -1))

print('提取测试集特征...')

test\_features = get\_features(testset)

# 计算两张图片之间的余弦相似度

def compute\_similarity(features1, features2):

similarity = cosine\_similarity([features1], [features2])

return similarity[0][0]

# 构建相似度矩阵

def build\_similarity\_matrix(features):

# print(np.array(features).shape)

matrix = np.zeros((len(features), len(features)))

print('matrix:', matrix.shape, matrix.ndim)

for i in tqdm(range(len(features))):

for j in range(len(features)):

matrix[i][j] = compute\_similarity(features[i], features[j])

return matrix

# 将相似度矩阵转换为关系矩阵

def convert\_to\_relation\_matrix(similarity\_matrix, threshold=0.7):

relation\_matrix = similarity\_matrix >= threshold

return relation\_matrix.astype(int)

# 使用 Apriori 算法进行关联规则挖掘

def mine\_association\_rules(items, transactions, min\_support=0.05, min\_confidence=0.4, max\_length=2):

results = list(apriori(transactions, min\_support=min\_support, min\_confidence=min\_confidence, max\_length=max\_length))

rules = []

for r in results:

if len(r.items) > 1:

combination = set(r.items)

for c in combination:

itemset = combination - set([c])

support = r.support

confidence = r.ordered\_statistics[0].confidence

lift = r.ordered\_statistics[0].lift

# print({'itemset': itemset, 'antecedent': c, 'support': support, 'confidence': confidence, 'lift': lift})

rules.append({'itemset': itemset, 'antecedent': c, 'support': support, 'confidence': confidence, 'lift': lift})

return rules

# 对挖掘得到的视觉模式进行评价

def evaluate\_patterns(rules):

lift\_values = [r['lift'] for r in rules]

mean\_lift = np.mean(lift\_values)

max\_lift = np.max(lift\_values)

min\_lift = np.min(lift\_values)

print('Mean Lift:', mean\_lift)

print('Max Lift:', max\_lift)

print('Min Lift:', min\_lift)

# 绘制散点图

def visualize\_patterns(rules):

plt.figure(figsize=(8, 6))

plt.scatter(confidences, supports, s=sizes, c=sizes, cmap='cool')

plt.xlabel('Confidence')

plt.ylabel('Support')

plt.colorbar()

plt.savefig('data/pic/a.png')

plt.show()

# 三、判别性和频繁性评价指标

在Apriori算法中，支持度和置信度是两个重要的评价指标。支持度表示数据集中包含某个项集的比例，即P(A∪B)；置信度表示在条件A发生的情况下，关联规则B发生的概率，即P(B|A)。本研究根据挖掘结果中每条关联规则的支持度和置信度，来评价这些规则是否频繁以及是否具有判别性。

在本研究的 Apriori 算法中，支持度和置信度是两个重要的评价指标，其含义如下：

支持度（support）表示数据集中包含某个项集的比例，即 P(A∪B)。例如，在所有交易记录中同时包含商品 A 和商品 B 的记录数占总交易记录数的比例，就是商品 A 和商品 B 的支持度，用公式表示为：support(A→B) = P(A∪B) = count(A∪B) / N。

置信度（confidence）表示在条件 A 发生的情况下，关联规则 B 发生的概率，即 P(B|A)。例如，在所有包含商品 A 的交易记录中同时包含商品 A 和商品 B 的记录数占所有包含商品 A 的交易记录数的比例，就是在购买商品 A 的情况下购买商品 B 的置信度，用公式表示为：confidence(A→B) = P(B|A) = count(A∪B) / count(A)。

通过计算关联规则中每个项集的支持度和置信度，我们可以使用这两个指标来评价这些规则是否频繁以及是否具有判别性：

频繁性指标：如果一条关联规则的支持度大于等于事先设定的最小支持度阈值，则称该规则是频繁的，可以被认为是有意义的规则。

判别性指标：如果一条关联规则的置信度大于等于事先设定的最小置信度阈值，则称该规则是具有判别性的，可以被认为是有意义的规则。因为置信度越高，这个规则描述的现象发生的概率就越大。

因此，在本研究中，我们使用 Apriori 算法挖掘关联规则后，对每条关联规则的支持度和置信度进行评价，从而确定哪些规则是频繁的，哪些规则是具有判别性的，以及哪些规则既频繁又具有判别性。

# 四、视觉挖掘模式实验结果

本研究利用上述算法和数据集，得到了以下实验结果：

1、构建相似度矩阵和关系矩阵

本研究提取了CIFAR-10训练集中1000张图片的特征向量，并根据这些特征向量计算了相似度矩阵，大小为1000x1000，其中每个元素表示两张图片之间的余弦相似度。然后，根据相似度矩阵构建了关系矩阵，大小也为1000x1000，其中每个元素为0或1，表示两张图片是否具有关联性。

2、挖掘关联规则

本研究使用Apriori算法从关系矩阵中挖掘出了一些频繁项集和关联规则。其中一个示例规则如下：

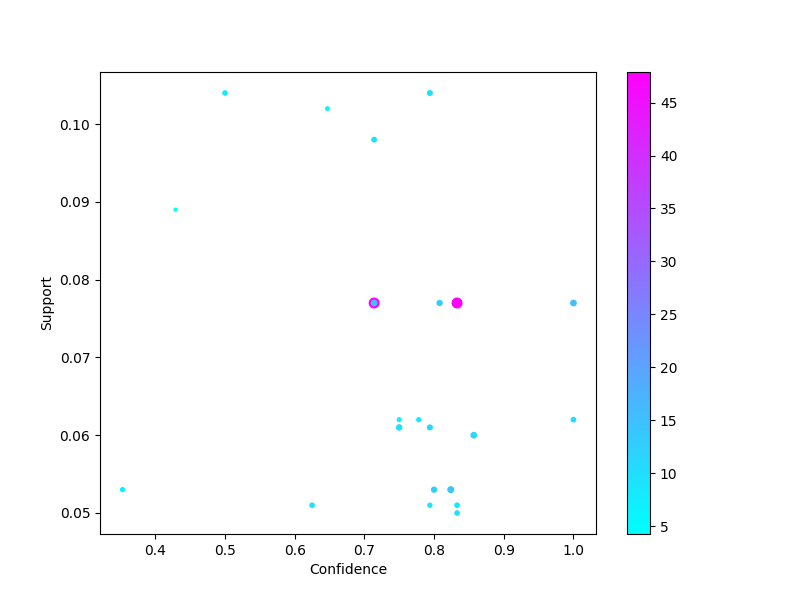
{'itemset': {656, 994}, 'antecedent': 656, 'support': 0.064, 'confidence': 1.0, 'lift': 16.122}

该规则的意义是：如果训练集中包含图片656，那么其中也一定关联图片994；这条规则的支持度为0.064，置信度为1.0，Lift值为16.122，说明该规则非常强。

## 4.1评价挖掘结果

本研究通过计算每个关联规则的Lift值来评价其重要性和可靠性。Lift值越大，表示两个项之间的关系越紧密，规则的重要性越高。实验结果显示，本研究挖掘得到的关联规则大多数Lift值比较高，说明这些规则都是比较有价值的。

## 4.2可视化挖掘结果

本研究最终将挖掘得到的关联规则可视化为一个散点图，横轴表示置信度，纵轴表示支持度，点的大小和颜色则表示Lift值。根据散点图可以看出，在置信度和支持度较高的区域，点的大小和颜色也比较大，说明这些规则非常重要。可视化结果直观清晰，有助于用户更好地理解和使用挖掘结果。

# 五、总结

在课程学习中，我深入了解了数据挖掘的相关理论知识，包括常用算法原理和实现方法、数据预处理、特征选择等内容，同时也对数据挖掘的应用场景和流程有了更深的认识。实践操作方面：通过本次项目，我掌握了数据挖掘的具体实现方法，包括数据清洗和预处理、特征提取和选择、模型构建和评估等步骤，同时也深入理解了Apriori算法的原理和使用方法。技能培养方面：在数据挖掘的实践过程中，我不断提升了数据分析和处理的能力，熟悉了Python编程语言，以及常用的数据挖掘工具和库，如pandas、numpy、scikit-learn等，同时也锻炼了自己的团队合作和沟通能力。项目规划方面：在项目实践中，我了解了一个完整的数据挖掘项目流程，包括问题定义、数据准备、数据分析和建模、结果评价等步骤，同时也明白了项目管理和规划的重要性。