适用于小样本学习的原型网络

摘要

我们针对少样本分类问题提出了原型网络，其中分类器必须泛化到训练集中未见过的新类，仅给出每个新类的少量示例。原型网络学习一个度量空间，在该空间中可以通过计算到每个类的原型表示的距离来执行分类。与最近的小样本学习方法相比，它们反映了更简单的归纳偏差，这在这种有限数据的情况下是有益的，并取得了优异的结果。我们提供的分析表明，一些简单的设计决策可以比涉及复杂架构选择和元学习的最新方法产生重大改进。我们进一步将原型网络扩展到零样本学习，并在 CU-Birds 数据集上取得了最先进的结果。

关键词：小样本学习；原型网络；深度学习

1 引言

在机器学习和深度学习的分类任务中，传统的监督学习方法依赖于大量的标注数据来训练模型，以实现对新样本的准确分类。然而，在现实世界中，很多场景下的数据标注成本高昂或难以实现，尤其是在面对新的、未见过的类别时。因此，如何使模型能够在有限的训练样本下有效地进行泛化，成为了机器学习领域的一个重要研究方向。基于上述背景，论文提出了原型网络（Prototypical Networks）[2] 这一方法，旨在解决少样本学习（Few-shot Learning）问题。少样本学习要求模型能够在仅见过每个新类别少量样本的情况下，快速适应并准确分类新的类别。原型网络通过计算每个类别在嵌入空间中的原型（即类别样本的均值向量），并利用查询点与这些原型之间的距离来进行分类决策。这种方法与之前的匹配网络（Matching Networks）[3] 等方法相比，具有更简洁和高效的优点。原型网络不仅为少样本学习问题提供了一种新的解决方案，而且具有重要的理论和实践意义。在理论层面，它揭示了通过嵌入空间和原型表示可以实现高效的类别泛化。在实践层面，原型网络在多个基准数据集上取得了优异的性能，展示了其在实际应用中的潜力。此外，原型网络的思想还可以进一步扩展到零样本学习等其他相关领域，为机器学习领域的研究提供了新的视角和思路。

2 相关工作

2.1 少量样本学习（Few-shot Learning）的先前方法

在论文提出之前，少量样本学习已经受到了研究者的广泛关注。这类方法主要解决的是如何在只有极少数标注样本的情况下，让模型能够有效地学习到新的类别。例如，Matching

Networks（Vinyals et al., 2016）利用注意力机制，在学习的嵌入空间中对标记集（支持集）的嵌入进行加权，以预测未标记点（查询集）的类别。这种方法通过模拟少量样本学习的任务来设计训练集的小批量（episodes），每个小批量都通过对类别和数据点进行子采样来模仿测试环境，从而提高模型的泛化能力。另外，Ravi 和 Larochelle（2017）[1] 进一步发展了基于 episode 的训练思路，提出了一个元学习（meta-learning）方法来处理少量样本学习问题，他们训练一个 LSTM 来产生分类器的更新，以便在给定一个小批量时，它能够很好地泛化到测试集。

2.2 聚类方法与距离度量

论文还涉及到了聚类方法和距离度量的使用。特别是在处理少量样本学习任务时，模型需要有一个简单的归纳偏置（inductive bias），以避免过拟合。Prototypical Networks（本文提出的方法）就是基于这样的想法：存在一个嵌入空间，其中的点围绕每个类别的单个原型聚类。这种方法与聚类方法有着紧密的联系，特别是当使用 Bregman 散度这类距离函数时，原型网络的算法可以看作是在支持集上执行混合密度估计。

2.3 零样本学习

在零样本学习 [4] 中，模型需要在没有见过任何属于测试类别的训练样本的情况下进行分类。这通常是通过利用类别元数据（如属性向量或文本描述）来实现的。论文展示了如何修改原型网络来处理零样本学习的情况，将类别的元数据向量作为原型的替代或补充，从而在未见过的类别上进行分类。

3 本文方法

3.1 本文方法概述

设计一个嵌入函数，将图像特征映射到一个嵌入空间，在该空间中，每个类的点会围绕一个单一的原型表示聚类。原型网络通过计算每个类的支持集嵌入向量的均值来得到原型表示，然后对于查询点，通过计算其在嵌入空间中与各个类原型的距离来进行分类。训练过程采用 episode 形式，以模拟小样本学习的场景，并通过最小化真实类别的负对数概率来进行优化。分类过程如图 1所示：

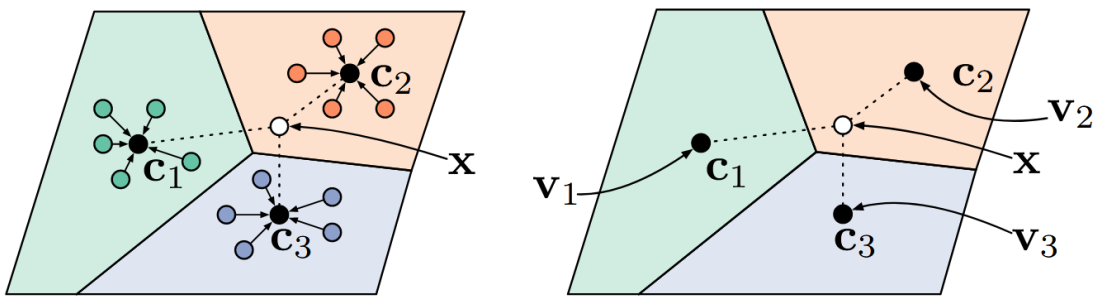

图 1. 原型网络示意图, 左侧为小样本学习，右侧为零样本学习

3.2 距离度量

在 Prototypical Networks 中，距离度量用于计算嵌入空间（embedding space）中查询点（query point）与每个类别原型（prototype）之间的距离。这个距离度量决定了分类决策，即哪个类别的原型与查询点最近。使用平方欧几里得作为距离度量，这与模型的混合密度估计特性有关。当使用 Bregman 散度（如平方欧几里得）时，原型计算（即支持集嵌入点的均值）在数学上等同于在支持集上执行硬聚类。平方欧几里得在实验中比余弦相似度表现更好。这是因为余弦相似度不是 Bregman 散度，不满足与混合密度估计等价的条件。

3.3 训练样本集的构成

在训练过程中，Prototypical Networks 使用一种称为 “episodes”（训练样本集）的方法。每个 episode 模拟了真实测试环境中的少量样本情况。在 few-shot learning 中，这意味着每个 episode 可能包含几个类别，每个类别只有少量的标记样本。构建 episode 的一种简单方式是选择 Nc 个类别和每个类别的 NS 个支持点，以匹配测试时预期的情况。然而，在训练时使用比测试时更多的类别（更高的 “way”）是有益的。这迫使模型在嵌入空间中做出更精细的决策，从而提高泛化能力。训练和测试时使用相同数量的 “shot” 通常是最佳选择。这种方法使得模型能够更好地适应在测试时遇到的少量样本的情况。

4 复现细节

4.1 与已有开源代码对比

在原型网络定义时，在官方给出的代码的基础上，加入了两层全连接层来实现原型的变换，并且设计了一个原型内聚损失和原型分离损失来计算某个类别与变换类别原型之间的距离之和，加上交叉熵损失，三者的损失之和来对原型进行更新优化。

```python
class Protonet(nn.Module):
    def __init__(self,input_dim, output_dim, encoder):
        super(Protonet, self).__init__()
        # 简单的两层全连接网络来实现原型的变换
        self.fc1 = nn.Linear(input_dim,  out_features: 128)
        self.fc2 = nn.Linear( in_features: 128, output_dim)

        self.encoder = encoder
    # 原型内聚损失
    def prototype_cohesion_loss(embeddings, transformed_prototypes, support_set):
        loss = 0.0
        for i, prototype in enumerate(transformed_prototypes):
            # 获取类别 i 的样本嵌入
            class_samples = embeddings[support_set == i]

            # 计算类别 i 中所有样本与变换原型之间的距离
            loss += torch.sum((class_samples - prototype) ** 2)

        return loss / embeddings.size(0)
    # 原型分离损失
    def prototype_separation_loss(transformed_prototypes, delta=1.0):
        num_classes = transformed_prototypes.size(0)
        loss = 0.0

        for i in range(num_classes):
            for j in range(i + 1, num_classes):
                distance = torch.norm(transformed_prototypes[i] - transformed_prototypes[j], p=2)
                # 强制要求类别 i 和 j 之间的距离大于 delta
                loss += torch.max(torch.tensor(0.0), delta - distance)

        return loss
```

图 2. 原型变换与损失计算

4.2 创新点

在标准的原型网络中，原型是通过计算每个类别支持集样本的均值来得到的，这种方式假设了每个类别的分布是均匀的。然而，在实际应用中，类别之间的分布可能并不均匀，某些类别的样本可能存在较大的分散度而另一些类别的样本则可能比较集中。此时，静态的原型（即不进行调整的原型）可能无法很好地反映类别的真实分布。原型变换损失的目的就是通过动态调整原型，使得原型能更好地适应数据分布，从而提高分类精度。核心思想：通过一个变换网络（可以是一个小的神经网络或一个自注意力机制）对原型进行动态调整。该变换网络的输入是原型（即每个类别的均值嵌入向量），输出是一个变换后的原型。变换的目标是使得同一类别的样本更加聚集于原型附近，不同类别的原型之间的距离更远，从而提高分类的准确性。

5 实验结果分析

对实验所得结果进行分析，模型在训练过程中表现出了显著的学习能力，训练损失稳步下降，准确率持续上升，最终在训练集上达到了接近完美的准确率，这表明模型能够很好地捕捉到训练数据中的特征和规律。然而，在验证集上，虽然准确率也达到了一个很高的水平，但损失的轻微上升和波动揭示了模型可能对训练数据存在一定程度的过拟合现象。这意味着

模型在面对新的、未见过的数据时，其泛化能力可能受到一定的限制。为了进一步提升模型的泛化性能，采取多种策略来优化模型。首先，增加正则化措施，例如引入 L1 或 L2 正则化，这有助于限制模型参数的复杂度，减少过拟合的风险。此外，使用 dropout 技术可以在训练过程中随机丢弃一些神经元，迫使模型不依赖于任何单一的特征，从而提高其在不同数据上的适应能力。通过旋转、缩放、翻转等操作，生成更多的训练样本，使模型在更加多样化的数据环境中进行学习，增强其对各种情况的处理能力。调整学习率策略，采用学习率衰减，在训练初期快速下降学习率以加速收敛，在训练后期则减缓学习率的下降速度，使模型能够更细致地调整参数，在保持稳定的同时进一步提升性能。在官方原代码的基础上进行改动，重新训练模型，证明了该创新点是有一定的效果的。

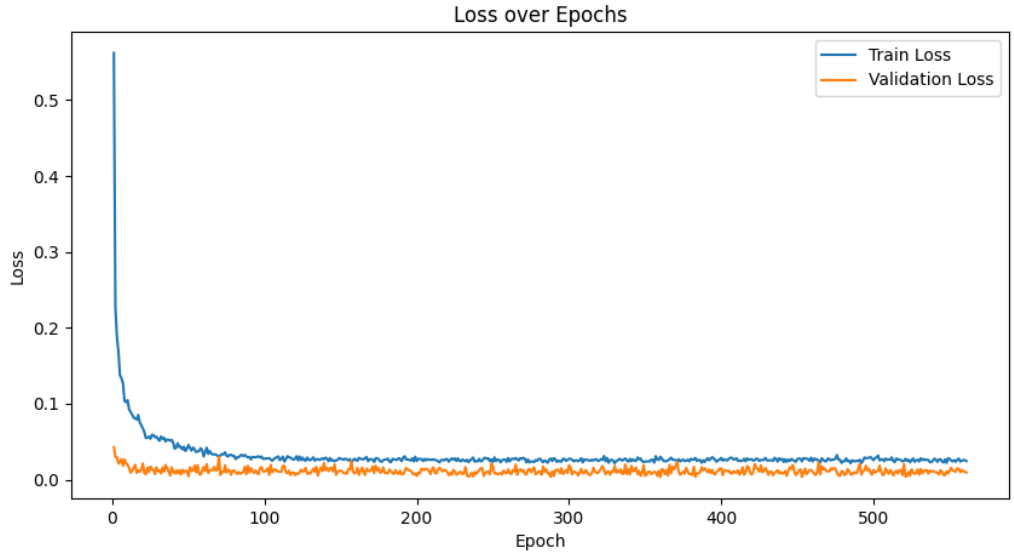

图 3. 实验结果示意

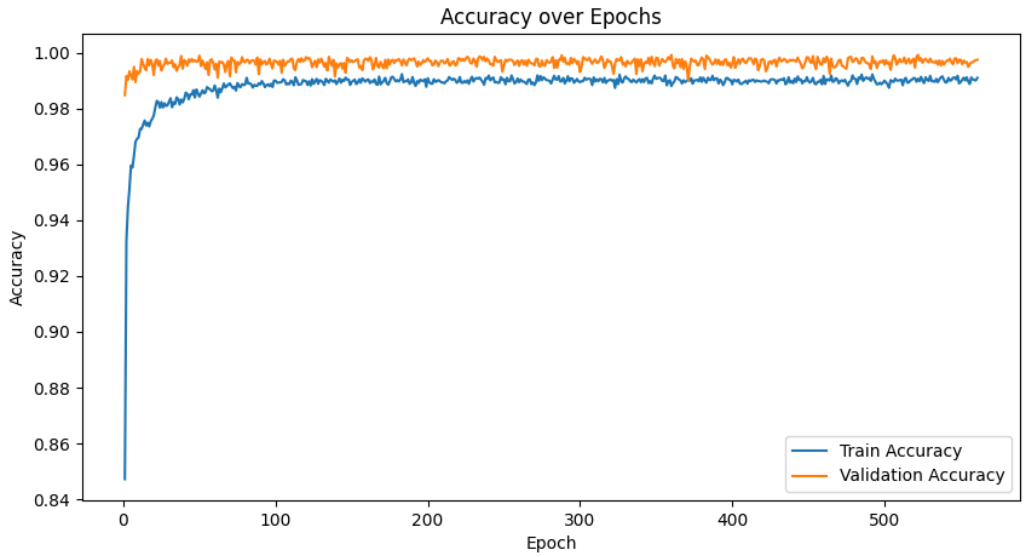

图 4. 实验结果示意

6 总结与展望

本研究报告对原型网络进行了复现，并在其基础上做了一点改进，对原型进行了两层全连接层的简单变换，将损失计算改为内聚损失 + 分离损失 + 交叉熵损失来更新模型。实现过程的不足主要体现在以下几个方面：首先，虽然对原型网络进行了改进，加入了两层全连接层并引入了内聚损失、分离损失和交叉熵损失，但实验结果表明，模型的复杂度增加却未能显著提升性能，可能是由于损失项之间未能达到理想的平衡。其次，超参数的调优不够充分，未能通过系统的实验探索找到最佳配置。此外，训练数据集的局限性也影响了模型的泛化能力，而损失函数的设计和优化仍存在一定问题，未能充分发挥各个损失项的优势。再者，增加的网络层和损失函数导致了训练时间的显著延长，计算开销也有所增加。最后，模型的可解释性下降，调试和性能分析变得更加复杂，过拟合的风险也随之增大。未来的研究可以集中在几个关键方向：首先，优化损失函数设计，通过动态调整损失项权重以提高模型的训练效果。其次，扩大数据集规模并增加数据的多样性，以提升模型的泛化能力。网络结构方面，可以尝试更复杂的深度学习模型，增强特征提取能力。自监督学习和迁移学习的结合也能有效提升小样本学习的表现。此外，针对模型的可解释性问题，未来可以探索可视化和分析技术，以增加模型的透明度。最后，优化训练效率，采用高效的优化器和分布式训练策略，同时探索模型压缩与量化技术，以减少计算开销。

参考文献

[1] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

[2] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[3] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[4] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning —a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2251–2265, 2018.