Work based on meta learning

N4A

2019 年 4 月 22 日

1 Meta learning framework

通常，应用 Meta Learning 的任务中包含多个子任务 $\{T_i|i=1,..N\}$。每个子任务都有各自的数据集，记为 $D_1,...,D_N$。一般这些数据集都由训练集 D_i^{train} 和测试集（验证集）D_i^{test} 组成。每个子任务都有一个学习目标 $L_\lambda(D_i^{train},\theta_i)$, 其中 θ_i 为子任务学习模型的参数，λ 为各子任务模型的公共参数，一般将子任务的学习模型称为 Learner。

在 Meta Learning 框架中，除了各个子任务的 Learner 之外，还有一个 Meta-Learner，Meta-Learner 的目标 $L(D,\theta,\lambda)$ 是综合考量所有子任务，使之在测试集上的损失最小。λ 是各个子任务公共的参数，也是 Meta-Learner 的参数。

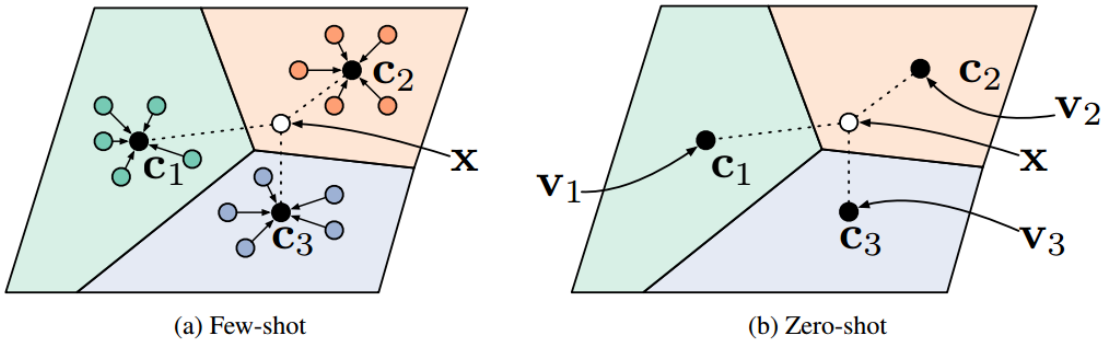
然后对于一个新来的子任务 T_{N+1}, 我们可以使用训练好的 Meta-Learner 来辅助该子任务的模型的学习。

2 Prototypical Networks for Few-Shot Learning(NIPS'17) [7]

2.1 Model Overview

如 1 (a) 所示，该模型的出发点是对于每一类的样本，它们的 embedding 在特征空间中分布应当是在该类特征 (c_1,c_2,c_3) 附近的。该类特征称为 Prototype Representation.

在 few-shot classification 的任务中，记样本集 $S=\{(x_1,y_1),...,(x_N,y_N)\}$, S_k 为 label 都为 k 的样本集，作为一个子任务的训练集，Q_k 为对应的测试集。本文中，每一个类别的数据集即为一个子任务的数据集。



(a) Few-shot (b) Zero-shot

图 1: Prototypical Network 在 few-shot 和 zero-shot 任务下的示意图

文中定义一个函数 f_ϕ 将样本映射到特征空间，其中 ϕ 为该函数的参数；定义一个距离函数 d 衡量特征空间中两个样本点之间的距离。在训练的过程中，如公式 1, 每一个类的 Prototype Representation 即为该类训练集中所有样本特征的均值。

$$c_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\phi(x_i) \tag{1}$$

然后对于一个新的样本点 x，模型通过如下公式计算 x 属于每个类别的概率。该公式即选取到对应类别的 Prototype 特征距离最小的类别作为该样本类别。

$$p_\phi(y = k|x) = \frac{exp(-d(f_\phi(x), c_k)}{\sum_{k^l} exp(-d(f_\phi(x), c_{k^l}))} \tag{2}$$

然后在子任务 $\{S_k\ Q_k\}$ 中测试集 Q_k 上的损失函数定义如下：

$$J(\phi) = -logp_\phi(y = k|x) = d(f_\phi(x), c_k) + log(\sum_{k^l} exp(-d(f_\phi(x), c_{k^l}))) \tag{3}$$

模型最终的训练目标是学习特征映射函数 f_ϕ 的参数，使得所有子任务（每个类别的数据）的损失和最小。模型算法如图 2 所示。

2.2 Zero-shot classification

如图 1 (b) 所示，该模型应用于 zero-shot 需要每个类别的相关描述信息（v_1, v_2, v_3）。该信息可以是文本描述，种类特征数据例如重量，大小，颜色等。文中直接将这些辅助信心映射的特征空间作为类别的 Prototype 特征。该过程为预训练好的。

Algorithm 1 Training episode loss computation for prototypical networks. N is the number of examples in the training set, K is the number of classes in the training set, $N_C \leq K$ is the number of classes per episode, N_S is the number of support examples per class, N_Q is the number of query examples per class. RANDOMSAMPLE(S, N) denotes a set of N elements chosen uniformly at random from set S, without replacement.

Input: Training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where each $y_i \in \{1, \ldots, K\}$. $\mathcal{D}_k$ denotes the subset of $\mathcal{D}$ containing all elements $(\mathbf{x}_i, y_i)$ such that $y_i = k$.
Output: The loss J for a randomly generated training episode.

$V \leftarrow$ RANDOMSAMPLE$(\{1, \ldots, K\}, N_C)$ ▷ Select class indices for episode
for k in $\{1, \ldots, N_C\}$ **do**
  $S_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k}, N_S)$ ▷ Select support examples
  $Q_k \leftarrow$ RANDOMSAMPLE$(\mathcal{D}_{V_k} \setminus S_k, N_Q)$ ▷ Select query examples
  $\mathbf{c}_k \leftarrow \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$ ▷ Compute prototype from support examples
end for
$J \leftarrow 0$ ▷ Initialize loss
for k in $\{1, \ldots, N_C\}$ **do**
  for $(\mathbf{x}, y)$ in Q_k **do**
    $J \leftarrow J + \frac{1}{N_C N_Q} \left[d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k)) + \log \sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_{k'})) \right]$ ▷ Update loss
  end for
end for

图 2: Prototypical Network 算法示意图

2.3 与 Matching Network[9] 进行比较

本文与 Matching Network 的方法非常类似，核心都是要学习样本的特征映射函数 f, 然后子任务都使用了非参的分类方法，本中是计算测试样本特征和每个类别的 Prototype 特征 (训练集样本特征的均值) 的距离（l2 norm），然后选择距离最小的类别，作为测试样本的类别。

在 Matching network 中，是使用 KNN 的方法计算加权类别。对于一个样本 $\hat{x}$, 使用如下公式预测其 label $\hat{y}$:

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \tag{4}$$

其中, x_i, y_i 就是 Support Set（相当于子任务训练集）$S = \{(x_i, y_i\}_{i=1}^{k}$ 中的样本和 label。函数 a 文中称为注意力计算函数，也就是 KNN 中的相似度计算函数，其定义如下，其中函数 c 为距离函数。

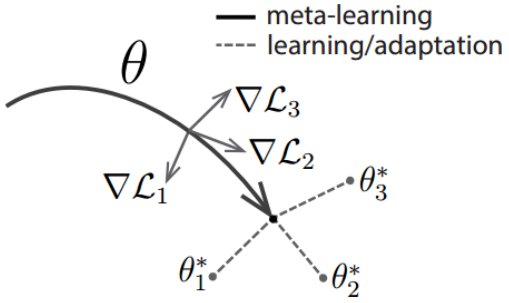
$$a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$

当训练集中每个类别只有一个样本时（one-shot），两种方法就一样了。此时只有选取的距离函数不一样，文中选取平方差距离函数，Matching network

Table 1: Few-shot classification accuracies on Omniglot.

Model	Dist.	Fine Tune	5-way Acc. 1-shot	5-way Acc. 5-shot	20-way Acc. 1-shot	20-way Acc. 5-shot
MATCHING NETWORKS [29]	Cosine	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETWORKS [29]	Cosine	Y	97.9%	98.7%	93.5%	98.7%
NEURAL STATISTICIAN [6]	-	N	98.1%	99.5%	93.2%	98.1%
PROTOTYPICAL NETWORKS (OURS)	Euclid.	N	**98.8%**	**99.7%**	**96.0%**	**98.9%**

图 3: Omniglot 数据集上实验结果



(a) Intuition

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

(b) Algorithm

图 4: 左图为模型示意图，对于一个新的任务，MAML 希望通过之前类似的任务学习到一个好的参数 θ，作为该新任务模型的初始参数。该初始参数在该新任务上能够快速收敛到适合该子任务的值，如 θ_i^* 对应于不同的新任务的情况。右图为算法框架

中选择 cosine 距离函数。文中实验也表明使用平方差距离函数效果会好一点。实验结果如图 3 所示。

3 MAML: Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks(ICML17)[2]

3.1 Model introduction

如图4(a)所示，MAML 的想法是在供训练的子任务上找到一个好的子任务参数 θ, 然后将该参数作为新任务模型的初始参数，使得新任务的模型在该参数上能够快速收敛。（这里该模型应当是假设了所有的子任务都使用了相同结构的模型）

作者指出这样的想法来源于对任务集合 $\{T_i\}$, 一个学习模型应当能够学习出适用于所有子任务的特征表示。我们的目标就要在模型设计中显示强调这点的重要性。如图 4(b) 所示，在训练过程中算法第 6 行，每个子任务的参数都是在一个相同的参数 θ 的基础上做一个 batch 的更新（注意这里使用的数据是每个

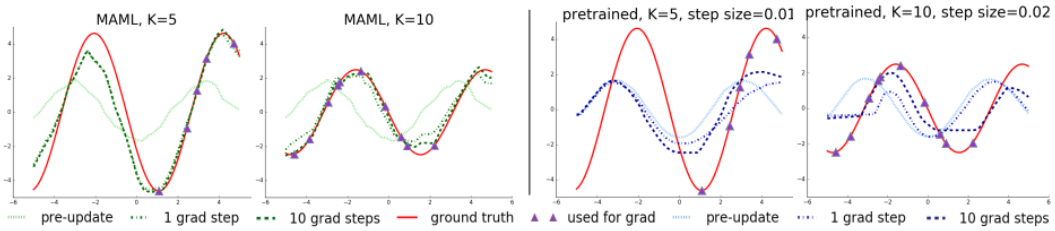


Figure 2. Few-shot adaptation for the simple regression task. Left: Note that MAML is able to estimate parts of the curve where there are no datapoints, indicating that the model has learned about the periodic structure of sine waves. Right: Fine-tuning of a model pretrained on the same distribution of tasks without MAML, with a tuned step size. Due to the often contradictory outputs on the pre-training tasks, this model is unable to recover a suitable representation and fails to extrapolate from the small number of test-time samples.

图 5: MAML: sin 函数拟合实验结果图

子任务对应训练集数据，即 D_i^{train}）。然后在第 8 行，使用使用之前的所有子任务的数据更新参数 θ。这里有两点需要注意：1. 根据代码实现，这里使用的数据是每个子任务测试集的数据，即 D_i^{test}; 2. 更新项 $\nabla_\theta \sum_{T_i \sim p(T)} L_{T_i}(f_{\theta_i'})$ 是对参数 θ 求导，而函数 $f_{\theta_i'}$ 的参数由第 6 行可以看出包含对参数 θ 的一阶导数，所以这一行需要对参数 θ 求导需要求二阶导数。

模型的想法很简单。但是很好的利用了 meta learning 的特性以及之前介绍 MatchNet[9] 时提到的机器学习模型设计的基本理念：Test and train conditions must match. 首先 MAML 模型的目标是能够学习到一个参数 θ, 然后在一个新的子任务中将其作为该子任务模型的初始参数，使得在该子任务上，只进行少量的数据（即 D_{N+1}^{train}）迭代就能快速收敛, 使得模型能够在测试集 D_{N+1}^{test} 上表现出良好的效果。根据 “Test and train conditions must match” 的理念，在训练模型时，每个子任务自然也就只能进行少量的数据迭代更新参数。如图 4(b) 中第 6 行所示，每个子任务的模型（learner）只进行一个 batch 的训练数据 (即 D_i^{train}) 迭代来更新各自的参数 θ_i，然后在第 8 行，meta learner 更新参数 θ(相当于第一部分 meta learning framework 中提到的参数 λ), 使得只经过一个 batch 的数据更新各自参数 θ_i 的模型（learner）能够在对应的测试数据集 D_i^{test} 表现出良好效果。所以第 8 行很重要一点是损失函数要对参数 θ 求导，目的是寻找好的参数 θ, 使得所有训练时的子任务只要在其基础上经过少量的 D_i^{train} 数据上更新，就能在对应的 D_i^{test} 表现出好的效果。学习到这样的参数 θ 之后，对于之后新来的子任务的目标就恰好满足训练时一直强调的参数 θ 要满足的目标。当然要实现这样的目标需要满足所有的子任务是相似的，即文中假设所有子任务采样于相同的任务分布 $p(T)$。

总结一下该方法对于子任务模型的形式基本没有要求，只要求模型有参数，并且学习目标对于该参数可以求导，所以该方法应用范围极广。图 5 是文中一个 sin 函数拟合任务的实验结果图。可以看出，在测试的时候，只使用少量的样本（K = 5 or 10），MAML 方法拟合得更好。但是该模型隐含要求所有子任务都使用相同的模型结构，这是该方法适用范围的一个限制点。

Meta learning 是一种概念性的框架，落实到具体任务中，还要设计具体的

模型。对于一个单任务模型，比如分类问题，测试标准与训练目标有显著的一致性。但是对于一个复杂方法，例如应用了 GAN 的模型，测试标准与训练目标就不是很一致了。再对于 meta learning 或者 transfer learning 基本上只是一个概念性，方向性的模型设计指导，应用到具体任务中，设计模型时，应该要考虑到 Test and train conditions must match，最好是训练目标和测试目的是等价的表达形式，实验的成功率就会高很多吧。

3.2 与 LSTM-Meta-Learner 对比

3.2.1 LSTM-Meta-Learner[6]

虽然有很多的变种，但是普通的参数更新公式基本如下：

$$\theta_t = \theta_{t-1} - \alpha_t \nabla_{\theta_{t-1}} L_t$$

LSTM 的细状态更新公式如下：

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t$$

比较之下只要令 $f_t = 1, c_{t-1} = \theta_{t-1}, i_t = \alpha_t, \hat{c}_t = -\nabla_{\theta_{t-1}} L_t$ 就可以得到普通参数更新方法的这个特例。所以利用 LSTM 自动学习参数的更新方法是一个很自然的选择。

在这个环境下，meta-learner 就是 LSTM，Learner 就是图像分类模型，其参数为 θ, 作者令细胞状态 $c_t = \theta_t$，于是就可以通过 meta-learner LSTM 学习各个子任务的参数。与前面的 meta-learning 框架比较，这里的 θ 相当于 λ, 每个子任务没有自己独立的参数。

在普通的参数更新方法下 learning rate 就是一个常数 α_t, 但是在 LSTM 中，我们可以控制输入门，使得细胞状态得到更有效的更新。如下：

$$i_t = \sigma\left(\mathbf{W}_I \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, i_{t-1}\right] + \mathbf{b}_I\right),$$

这表示当前的学习率于 θ_{t-1} $\nabla_{\theta_{t-1}} L_t, L_t, i_{t-1}$ 相关，使用这些信息，模型希望学习到更好的策略控制学习率。

对于遗忘门也是一样，总是取常数 1 肯定不是最优的，作者希望学习到一个好的遗忘策略，如下：

$$f_t = \sigma\left(\mathbf{W}_F \cdot \left[\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t, \theta_{t-1}, f_{t-1}\right] + \mathbf{b}_F\right)$$

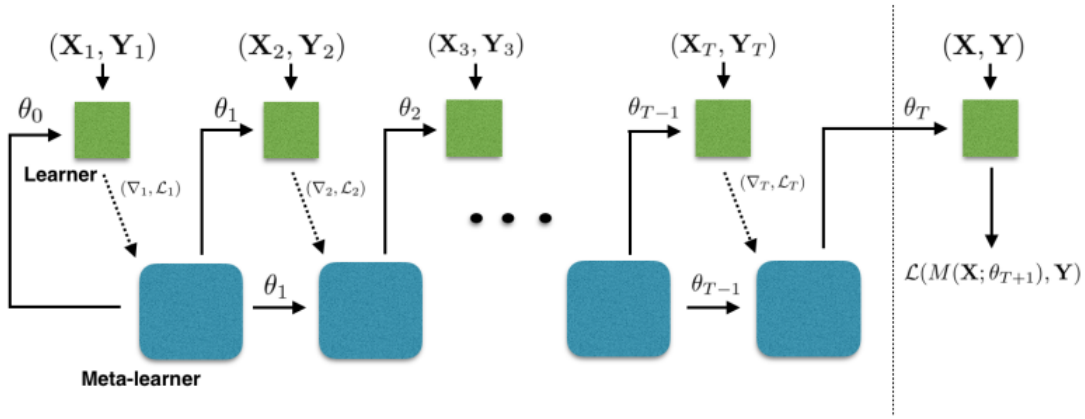


图 6: LSTM-Meta-Learner 一个子任务学习过程示意图

模型一个子任务的学习框架如图6所示，图中 (X_i, Y_i) 是子任务训练集中一个 batch 的数据，模型利用 Meta-learner 和训练数据 $\{(X_i, Y_i)\}$ 更新 Learner 的参数，然后 (X, Y) 为测试样本, 模型利用子任务中测试集中的数据设计 Meta-learner 的损失函数，目的是学习一个好的 Meta-leaner, 使之能够利用少量的子任务训练数据更新子任务模型 Learner 的参数，然后该 learner 在对应的测试数据上能够取得好的评测效果。图中虚线箭头表示参数更新时，不沿着该线反向传播，这是一个简化，否则更新 meta-learner 的参数就会变得复杂（这条线的依赖会带来 meta-learner 参数的二阶导数）。

在所有的子任务中，Meta-leaner 是不变的，Learner 是重新初始化参数的同结构网络, 数据是新的子任务的数据。在多个子任务上学习之后，对于一个新的子任务, Meta-learner 就会根据这个子任务 D_{train} 中的数据，自动更新适应这个子任务的 Learner 的参数，更新好之后，就可以在 D_{test} 上测试评估。

3.2.2 MAML 和 LSTM-Meta-Learner 对比

如图7所示，从参数视角看一个普通的监督模型的训练过程。首先有一个初始参数 θ, 然后模型利用训练数据 $\{x_i, y_i\}$ 和梯度下降的方法更新模型的参数得到最终参数 θ'。但是当训练数据很少时，模型很难收敛或者迅速过拟合。为了解决这个问题，MAML 的出发点是提供一个好的初始参数 θ，使得在该参数的基础上，经过少量的样本的梯度下降更新，模型就会收敛，并且不会过拟合。与 MAML 不同，LSTM-Meta-Learner 使用普通的随机初始化的参数作为初始参数，但是其希望找到一个好的参数更新方式，使用该方法更新参数能够使得模型只经过少量样本的更新就能快速收敛并且不会过拟合。

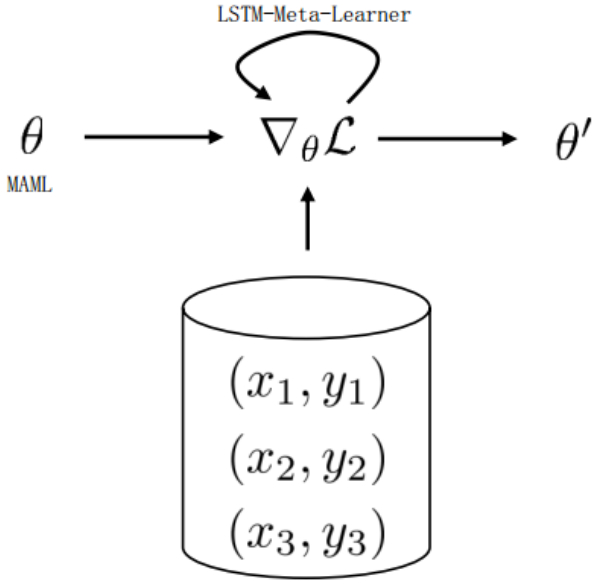


图 7: 监督模型示意图

3.3 Other works based on MAML

Automated Gradient Based Meta Learner Search[5](NIPS'18 Meta Learning workshop): 这篇文章的想法很简单，MAML 中具体任务的模型是没有限制的，这篇文章中就使用网络结构搜索的方法来找到一个合适的子任务模型结构。这样 Meta Learning 的目标就是找到一个好的模型结构和它的初始参数。本文的主要亮点是通过搜索合适的子任务模型结构使得实验结果取得了巨大的提升，如图8所示，Auto Meta 模型相比于普通的 MAML 模型在 MiniImageNet 的数据集中的 5-shot 5-way 任务上提升了 13.18%。

Algorithm	Params	5-shot 5-way
MAML + Transduction [8]	35k	$63.11 \pm 0.92\%$
Reptile + Transduction [18]	35k	$65.99 \pm 0.58\%$
Ours + Transduction ($F = 64$)	1,094k	$\mathbf{76.29 \pm 0.38}\%$
Reptile [18]	35k	$62.74 \pm 0.37\%$
Matching-Nets [25]	-	55.30%
Ravi and Laroche [19]	-	$60.20 \pm 0.71\%$
Prototypical-Nets [24]	-	$68.20 \pm 0.66\%$
Ours ($F = 64$)	1,094k	$\mathbf{70.87 \pm 0.23}\%$

Table 2: Results on Mini-ImageNet in *large* setting. For *large* setting, we use $F = 64$, feature scale rate $= 1.0$, $N = 0$, $B = 5$, inner learning rate $= 0.01$, meta iteration $= 11.2k$, and meta learning rate $= 1.0$.

图 8: Auto-Meta 在 MiniImageNet 上的实验结果

Bayesian Model-Agnostic Meta-Learning[3](NIPS'18): 这篇文章从贝叶斯概率角度来推导 MAML 的过程。

Meta-Learning and Universality: Deep Representations and Gradient Descent

Algorithm 1 CACTUs for classification

procedure CACTUs($\mathcal{U}, \mathcal{D}, P, k, T, N, K_{\text{m-tr}}, Q$)
  Run unsupervised learning algorithm $\mathcal{U}$ on $\mathcal{D}$, resulting in a model capable of producing embeddings $\{\mathbf{z}_i\}$ from observations $\{\mathbf{x}_i\}$.
  Run k-means on $\{\mathbf{z}_i\}$ P times (with random scaling) to generate a set of partitions $\{\mathcal{P}_p = \{\mathcal{C}_c\}_p\}$.
  for t from 1 to the number of desired tasks T **do**
    Sample a partition $\mathcal{P}$ from the set of partitions $\{\mathcal{P}_p\}$.
    Sample a cluster $\mathcal{C}_n$ without replacement from $\mathcal{P}$ for each of the N classes desired for each task.
    Sample an embedding $\mathbf{z}_r$ without replacement from $\mathcal{C}_n$ for each of the $R = K_{\text{m-tr}} + Q$ training and query examples desired for each class, and note the corresponding datapoint $\mathbf{x}_{n,r}$.
    Sample a permutation $(\mathbf{l}_n)$ of N one-hot labels.
    Construct $\mathcal{T}_t = \{(\mathbf{x}_{n,r}, \mathbf{l}_n)\}$.
  return $\{\mathcal{T}_t\}$

图 10: 无监督 meta learning 算法图

can Approximate any Learning Algorithm[1](ICLR'18): 这篇文章之后再补充一下。这是 MAML 原作者的一篇后续文章，主要论证了 MAML 的方法可以估计任意子任务模型 $f(D_i^{train}, x)$

4 Unsupervised learning via meta-learning（arxiv'18）[4]

本文要解决的一个问题是如何利用没有 label 的数据进行模型预训练。文中为 meta-train 的过程中使用没有 label 的数据进行训练，学习好 meta-learner，然后在将来新的有 label 的子任务中，使用该 meta-learner 监督该子任务学习。

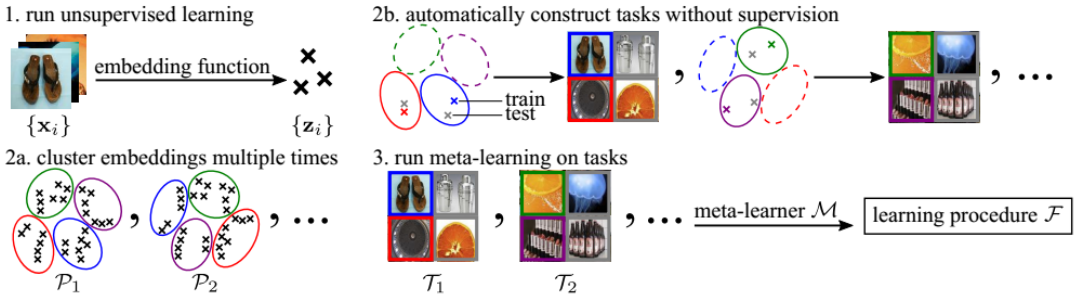


图 9: 无监督 meta learning 示意图

该方法的核心就是怎么利用没有 label 的数据集构建可以进行 meta-train 的各个子任务。该模型的示意图和算法如图 9 和 10 所示。构建 meta-train 子任务包含如下步骤。

1. 使用无监督的方法学习图片的特征表示。例如 Auto Encoder 方法。这点表现在算法第二行和示意图第一部分。

Table 1: Results of unsupervised learning on Omniglot images, averaged over 1000 downstream character recognition tasks. CACTUs experiments use $k = 500$ clusters for each of $P = 100$ partitions. Embedding cluster matching uses the same k. For complete results with confidence intervals, see Table 9 in Appendix F.

Algorithm \ (way, shot)	(5, 1)	(5, 5)	(20, 1)	(20, 5)
Training from scratch	52.50%	74.78%	24.91%	47.62%
ACAI k_{nn}-nearest neighbors	57.46%	81.16%	39.73%	66.38%
ACAI linear classifier	61.08%	81.82%	43.20%	66.33%
ACAI MLP with dropout	51.95%	77.20%	30.65%	58.62%
ACAI cluster matching	54.94%	71.09%	32.19%	45.93%
ACAI CACTUs-MAML (ours)	**68.84%**	**87.78%**	**48.09%**	**73.36%**
ACAI CACTUs-ProtoNets (ours)	**68.12%**	83.58%	**47.75%**	66.27%
BiGAN k_{nn}-nearest neighbors	49.55%	68.06%	27.37%	46.70%
BiGAN linear classifier	48.28%	68.72%	27.80%	45.82%
BiGAN MLP with dropout	40.54%	62.56%	19.92%	40.71%
BiGAN cluster matching	43.96%	58.62%	21.54%	31.06%
BiGAN CACTUs-MAML (ours)	58.18%	78.66%	35.56%	58.62%
BiGAN CACTUs-ProtoNets (ours)	54.74%	71.69%	33.40%	50.62%
Oracle-MAML (control)	94.46%	98.83%	84.60%	96.29%
Oracle-ProtoNets (control)	98.35%	99.58%	95.31%	98.81%

图 11: Omniglot 图片分类数据集上的实现结果

2. 对图片的特征进行多次聚类, 得到多个聚类结果。例如 K-means 算法。这点表现在算法第三行和示意图 2a 部分。

3. 构建子任务数据集。该部分是该方法的核心部分。如算法 5-9 行所示，首先从第二步中的多个聚类结果中随机采样一个聚类结果 P（5），然后从 P 中选择 N 个聚类族 $\{C_n\}$（6），对于每一个聚类族 C_n，随机选取其中的 R 个数据点（包含 K_{m-tr} 个子任务训练数据和 Q 个子任务测试数据），然后选取特征对应的图片作为数据集 (7)。到这里虽然分割了一个子任务的数据集，但是依然没有 label，文中的方法是随机为每一个聚类族分配一个 label(既然特征表示相近，那么它们对应的图片应该有某种意义上的相似，就将其标注为同一个 label)。这样就够建好了一个子任务 $T_t = \{(x_{n,r}, l_n)\}$ (8-9)。循环执行 5-9 即可构建多个子任务。这点也表现在示意图 2b 部分。

之后就可以用这些数据集训练 meta-learner，对于之后有真实 label 的图片分类任务，则可以利用该预训练的 meta-learner 去监督该子任务的学习。

这篇文章基于 meta learning 提出了一种无监督学习的方法，想法非常新颖，部分实验结果如图 11 所示。相比于，直接在新的子任务上训练，该方法确实可以使得训练准确度得到提高。其中 Oracle-* 方法为在 meta-train 的过程中，每一个数据使用真实的 label 来训练。

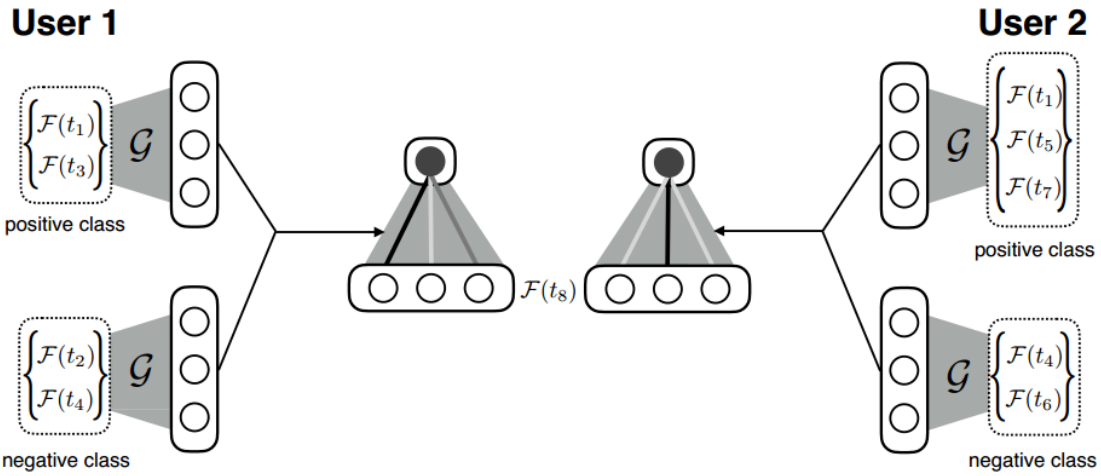


图 12: Linear Classifier with Weight Adaptation

5 A Meta-Learning Perspective on Cold-Start Recommendations for Items(NIPS'17)[8]

这篇文章尝试解决物品冷启动的问题。例如在新闻推荐的应用中，新闻时时刻刻不断更新，要立马推送给用户，这时还没有用户反馈的针对该新闻的喜好信息。文中，将根据一个用户的历史交互记录，向其推荐一个新的物品视为 meta learning 的子任务，然后使用 meta learning 的方法使得各个子任务的训练不是独立的，而是相互促进的。

模型的第一部分是一个物品特征提取函数 F，文中使用深度神经网络来实现该函数。而用户的特征则有其交互过的物品的特征聚合得到。文中将推荐任务视为一个二分类问题，即喜欢或不喜欢。用户 j 的交互过的物品集合记为 T_j，交互记录记为 $V_j = \{(t_m, e_{mj})\}$，其中 $t_m \in T_j, e_{mj} \in \{0, 1\}$。文中将交互记录中的两类物品分开聚合，作为两种一个用户的两种特征。

$$R_j^c = g(F(t_m) : t_m \in T_j \& e_{mj} = c) \tag{5}$$

用户特征提取如公式 5 所示，其中，$c \in 0, 1$ 表示特征类型，g 为特征聚合函数，文中采用了求均值的方法。然后，文中提出了两种方法来结合用户的特征和物品的特征来计算推荐概率。

第一种方法称为 Linear Classifier with Weight Adaptation，如图 13 所示。作者首先将提取的正样本特征和负样本特征线性结合在一起，然后和待推荐的物品特征计算内积得到推荐概率。图中表示为将用户特征作为网络权重，物品

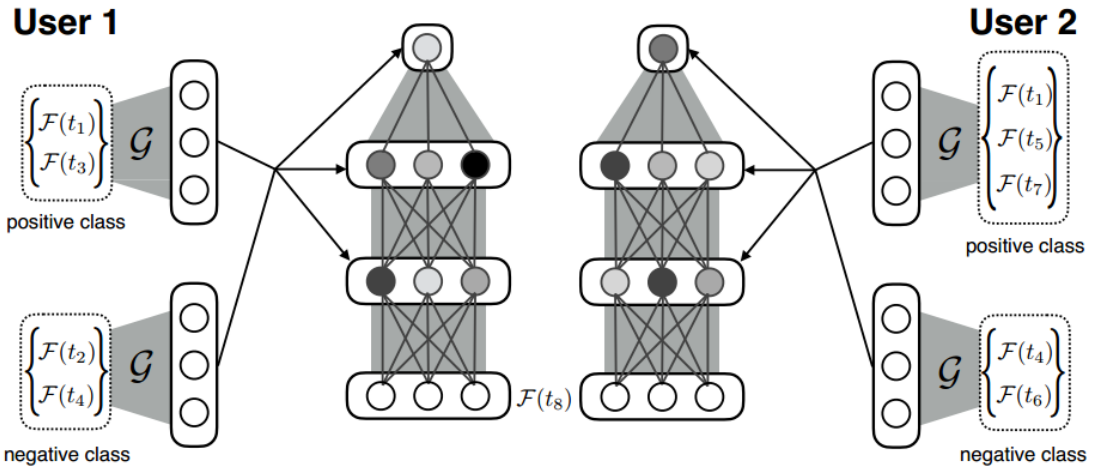


图 13: Non-linear Classifier with Bias Adaptation

特征作为网络层。公式如下所示。

$$Pr(e_{ij} = 1|t_i, u_j) = \sigma(b + F(t_i)(w_0 R_j^0 + w_1 R_j^1)) \tag{6}$$

第二种方法称为 Non-linear Classifier with Bias Adaptation，如图 13 所示。作者将 R_j^c 经过计算输出作为神经网络每一层的 Bias。如下公式，从左到右每一项以此为，最后一层的 bias，最后一层的权重参数，网络 H 个隐藏层的 bias，网络 H 个隐藏层的权重参数。其中 v, w, V, W 为待学习的参数。

$$[v^0 R_j^0 + v^1 R_j^1, w, \{V_l^0 R^0 + V_l^1 R_j^1\}_{l=1}^H, \{W_l\}_{l=1}^H] \tag{7}$$

文中设计了两个 baseline 对比实验。数据集为 Tweet 数据集，Tweet 为 Twiter 的一个推荐动态的功能。

1. Prod-base. 该方法仅使用当前物品的特征，和用户的辅助信息（用户活跃程度，用户和当前物品（twiter 文章）的作者的关系）做出推荐。
2. Prod-base with MF. 使用 MF 训练好用户特征，补充 Prod-base 方法
3. Prod-best. 使用当前物品特征和历史交互信息。（和本文使用信息量相同，采用不同特征提取方法）

实验结果如图 14 所示，其中 AUROC 表示 AUC 值。

本文应用 meta learning 点在于每个子任务（用户）共享相同的特征提取函数 F。然后要为每个用户学习各自的特征组合参数。例如方法一中的 $\{w_0, w_1\}$，方法二中的 $\{v, w, V, W\}$。这篇文章对 meta learning 的应用方法可解释性不强，实验也比较简单。主要亮点在于可以应用于物品冷启动问题以及可以作为 online learning 的方法。

Model	AUROC	AUROC (w/CTR)
MF (shallow)	+0.22%	+1.32%
MF (deep)	+0.55%	+1.87%
PROD-BEST	+2.54%	**+2.54%**
LWA	+1.76%	+2.43%
LWA*	+1.98%	+2.31%

Table 1: Performance with LWA

Model	AUROC	AUROC (w/CTR)
MF (shallow)	+0.22%	+1.32%
MF (shallow)	+0.55%	+1.87%
PROD-BEST	+2.54%	+2.54%
NLBA	+2.65%	**+2.76%**

Table 2: Performance with NBLA

图 14: LWA 和 BLBA 实验结果图

参考文献

[1] Sergey Levine Chelsea Finn. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *ICLR*, 2018.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[3] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[4] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018.

[5] Sungwan Kim Moonsu Cha Jung Kwon Lee Youngduck Choi Yongseok Choi Dong-Yeon Cho Jiwon Kim Jaehong Kim, Sangyeul Lee. Auto-meta: Automated gradient based meta learner search. *NIPS workshop on meta learning*, 2018.

[6] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[7] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

[8] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. In *Advances in Neural Information Processing Systems*, pages 6904–6914, 2017.

[9] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.