# 9-25 问题

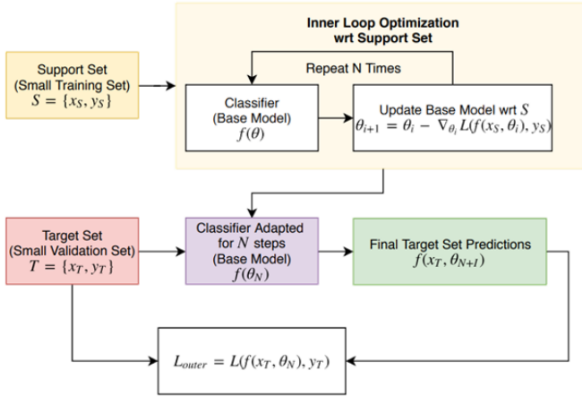
## MAML

➢ MAML

Meta-Train

**Algorithm 1** Model-Agnostic Meta-Learning

**Require:** $p(\mathcal{T})$: distribution over tasks
**Require:** $\alpha$, $\beta$: step size hyperparameters
randomly initialize $\theta$
**while** not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  **for all** $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to $K$ examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ — Support Set
  **end for**
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ — Query Set
**end while**




我的理解是，learning to learn 试图提高模型在不同任务中的泛化能力，相当于将任务当成了样本，让网络的loss在“任务测试集”上达到最低，这样在面对**新的任务**（*此处是否要求新任务中的class是没有出现过的？还是说是在$p(T)$中新生成的task？我觉得应该是后者吧*）时，此网络可以在很少的样本下快速finetune，适应任务。

训练的方法是用两重的循环。每个task中包含support set 和 query set 对应 任务训练集和任务测试集（*验证集是否存在？是包含在support set当中吗？每个task中的样本数应该不少吧，内层的训练应该就是普通的训练？*），通过内层的训练对网络参数进行第一次的更新。

所有的task都跑完内循环以后，来到外循环。把更新过的网络在所有task 的Query Set上面test，计算所有task的loss的和，并基于此进行梯度下降对**原始网络参数**进行正式的更新。（*为什么不直接对$\theta'$进行更新呢？是因为会$\theta'$过度偏向于这一次循环中的那些任务吗？*）

还有一个问题是，内循环和外循环都是对本次抽取的的所有tasks分别进行support set和query set的遍历，感觉模糊了task内部的结构特征，即task中有多少类有多少样本都不重要了，反正是一起算的。那为什么不直接抽取等量的sample直接进行训练，再用等量的sample去test去梯度下降呢？感觉从算法上说这是等价的？

## ProtoNet

**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**



我的理解是，MAML是用元学习去进行小样本学习， 而ProtoNet是用embedding learning去进行小样本学习。ProtoNet具有把一个sample快速embed到一个“原型空间”的能力，其实是提取出了里面可以用于分类的关键特征。

这里的训练方法也有support set 和query set，但感觉与MAML中的完全不同。此处的support set 和query set 基于class来区分，而不是task，support set 只是用于计算出原型$c_k$的位置，不用于训练。训练在$Q_k$中进行，观察损失函数感觉训练的最终目的是让属于此类的样本尽可能接近原型而不属于此类的样本尽可能远离原型。

最后通过梯度下降法去更新$f_\phi$中的权重。

问题是$f_\phi$的初始结构是如何确定的？参数太少不足以描述特征，参数太多小样本学习难以实现，怎样知道$f_\phi$能否很好地提取出特征？从数学上理解，我感觉这种“万物皆有原型”的假设是不大靠谱的。

以及最终应用ProtoNet的时候是把一个未参与过训练的但与其他类相似的类丢进去让它输出一个原型用于分类吗？