

神经网络与小样本学习整理

Robotmath

日期: November 25, 2020

1 神经网络

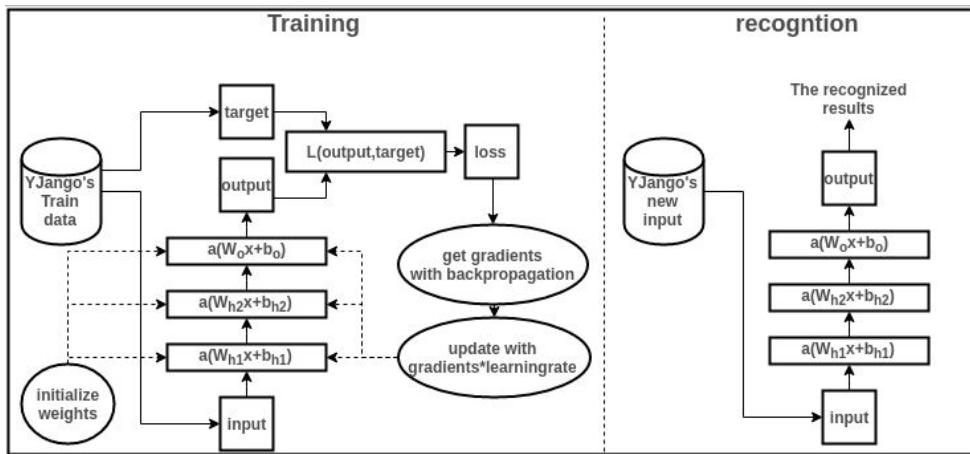


图 1: 神经网络过程

1. 神经网络每一层的行为:

$$y = a(w \cdot x + b)$$

通过如下 5 种对输入空间（输入向量的集合）的操作，完成输入空间到输出空间的变换：1. 升维/降维。2. 放大/缩小。3. 旋转。4. 平移。5. 弯曲。

2. 如何完成任务：

线性可分视角：神经网络的学习就是学习如何利用矩阵的线性变换加激活函数的非线性变换，将原始输入空间投向线性可分/稀疏的空间去分类/回归。增加节点数：增加维度，即增加线性转换能力。增加层数：增加激活函数的次数，即增加非线性转换次数。

3. 如何训练：

通过比较当前网络的预测值和我们真正想要的目标值，再根据两者的差异情况来更新每一层的权重矩阵。因此需要定义“如何比较预测值和目标值的差异”，即是损失函数或目标函数。通过使 loss 值向当前点对应梯度的反方向不断移动，来降低 loss。一次移动多少是由学习速率（learning rate）来控制的。

4. 梯度下降的问题

(a) 卡在局部极小值:

- 1. 调节步伐: 调节学习速率, 使每一次的更新“步伐”不同, 如随机梯度下降 (Stochastic Gradient Descent (SGD)): 每次只更新一个样本所计算的梯度; 小批量梯度下降 (Mini-batch gradient descent): 每次更新若干样本所计算的梯度的平均值。
- 优化起点: 合理初始化权重 (weights initialization)、预训练网络 (pre-train), 使网络获得一个较好的“起始点”。常用方法有: 高斯分布初始权重 (Gaussian distribution)、均匀分布初始权重 (Uniform distribution)、Glorot 初始权重 (根号 6)、He 初始权 (根号 2)、稀疏矩阵初始权重 (sparse matrix)。

(b) 梯度的计算: 解决方法是: 计算图: 反向传播算法。

训练过程: 1. 数据预处理 (归一化)。2. 权重初始化 (均匀分布等)。3. 训练网络: 训练过程就是用训练数据的 input 经过网络计算出 output, 再和 label 计算出 loss, 再计算出 gradients 来更新 weights 的过程。

(1) 正向传递: 算出当前网络的预测值

(2) 计算 loss

(3) 计算梯度: 从 loss 开始反向传播计算每个参数 (parameters) 对应的梯度 (gradients)

5. 反向传播算法如何工作: 反向传播的核心是对代价函数 C 关于 w (或者 b) 的偏导数 $\partial C / \partial w$ 的计算表示。该表示告诉我们在权重和偏差发生改变时, 代价函数变化的快慢。实际上它还告诉我们一些细节的关于权重和偏差的改变影响整个网络行为方面的洞察。通过矩阵与向量值函数给出一种更加全局的思考每层的激活值和前一层的关联方式: 用权重矩阵 w 作用在激活值上, 然后加上一个偏差向量 b , 最后作用 σ 函数。

$$a^l = \sigma(w^l a^{l-1} + b^l)$$

对代价函数一般要给出两个假设: 第一个假设就是代价函数可以被写成一个在每个训练样本 x 上的代价函数 C_x 的均值 $C = \frac{1}{n} \sum_x C_x$ 。需要这个假设的原因是反向传播实际上是对一个独立的训练样本计算了梯度, 有了这个假设, 可将代价函数 C_x 看做 C 。第二个假设就是代价可以写成神经网络输出的函数, 输出 y 同样是一个固定的参数。

反向传播其实是对权重和偏差变化影响代价函数过程的理解。最终极的含义其实就是计算偏导数 $\partial C / \partial w_{jk}^l$ (上下标表示 $l-1$ 层的第 k 个神经元到 l 层的第 j 个神经元的权重)。我们定义 l 层的第 j^{th} 个神经元上的误差 δ_j^l 为: $\delta_j^l \equiv \frac{\partial C}{\partial z_j^l}$ (z 是神经元的带权输入, 在激活函数前, 权重后)。

算法如下:

1. 输入 x 为输入层设置对应的激活值 a^l

2. 前向传播：对每个 $l = 2, 3, \dots, L$ 计算相应的 $z^l = w^l a^{l-1} + b^l$ 和 $a^l = \sigma(z^l)$
3. 输出层误差 δ^L ：计算向量 $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. 反向误差传播：对每个 $l = L-1, L-2, \dots, 2$ 计算 $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. 输出：代价函数的梯度由 $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ 和 $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

在实践中，通常将反向传播算法和诸如随机梯度下降这样的学习算法进行组合使用，我们会对许多训练样本计算对应的梯度。特别地，给定一个大小为 m 的 minibatch，下面的算法应用一步梯度下降学习在这个 minibatch 上：

1. 输入训练样本的集合
2. 对每个训练样本 x 设置对应的输入激活 $a^{x,1}$ ，并执行下面的步骤：。
 - 。前向传播：对每个 $l = 2, 3, \dots, L$ 计算

$$z^{x,l} = w^l a^{x,l-1} + b^l \text{ 和 } a^{x,l} = \sigma(z^{x,l})$$

。输出误差 $\delta^{x,L}$ ：计算向量

$$\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$$

。反向传播误差：对每个

$$l = L-1, L-2, \dots, 2 \text{ 计算}$$

$$\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$$

3. 梯度下降：对每个 $l = L-1, L-2, \dots, 2$ 根据 $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$ 和 $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$ 更新权重和偏差

4.SGD：每次随机选择一个样本进行计算 loss，梯度更新，进行多次。

5.mini-SGD：每次选择固定个数的样本数（一般 50-256）计算平均损失梯度更新，每次有放回抽取，直至选择了的样本数达到全部样本数。

2 端到端学习

相对于深度学习，传统机器学习的流程往往由多个独立的模块组成，比如在一个典型的自然语言处理（Natural Language Processing）问题中，包括分词、词性标注、句法分析、语义分析等多个独立步骤，每个步骤是一个独立的任务，其结果的好坏会影响到下一步骤，从而影响整个训练的结果，这是非端到端的。

End-to-end (E2E) learning refers to training a possibly complex learning system represented by a single model (specifically a Deep Neural Network) that represents the complete target system, bypassing the intermediate layers usually present in traditional pipeline designs.

3 Vapnik-Chervonenkis theory

部分参考: <https://medium.com/mlreview/modern-theory-of-deep-learning-why-does-it-works-so-well-9ee1f7fb2808>

泛化理论的目的是解释和证明为什么和如何提高训练集的准确率。这两个准确率之间的差异称为“泛化误差”或“泛化间隙”。从更严格的意义上，泛化间隙可以被定义为在给定学习算法 A 的数据集 SM 上的函数 F 的不可计算的预期风险和可计算的经验风险之间的差：

$$\text{the generalization gap} \triangleq R[f_{\mathcal{A}(S_m)}] - \hat{R}_m[f_{\mathcal{A}(S_m)}]$$

KKawaguchi、LPKELING 和 YBengio 提出了更为有用的办法。与其他人不同，他们接受了这样一个事实，即通常使用训练验证范式培训深度学习模型。他们使用验证错误替代非可计算的预期风险和训练错误。在这种观点中，他们针对为什么深度学习能泛化得如此完美而提出了以下观点：“我们之所以可以泛化得这么很好是因为我们可以利用验证的错误，通过模型搜索得到一个好的模型，并证明对于任何 $\delta > 0$ ，概率至少为 $1 - \delta$ ：

$$R[f] \leq \hat{R}_{\text{val}}[f] + \frac{2C \ln\left(\frac{|F_{\text{val}}|}{\delta}\right)}{3m_{\text{val}}} + \sqrt{\frac{2\gamma^2 \ln\left(\frac{|F_{\text{val}}|}{\delta}\right)}{m_{\text{val}}}}$$

3.1 VC dimention

用来定义二分类问题假设空间的复杂度（可表示能力），通过该假设空间能打散的（存在即可）最大样本集的样本数来定义。打散意味着任一个样本可能的分类情况都能在假设空间中找到。

定义 3.1. VC dimension of a set-family

We say that a set C is shattered by H if $H \cap C$ contains all the subsets of C , i.e.:

$$|H \cap C| = 2^{|C|}.$$

The VC dimension D of H is the largest cardinality of sets shattered by H . If arbitrarily large subsets can be shattered, the VC dimension is ∞ .

定义 3.2. VC dimension of a classification model

A binary classification model f with some parameter vector θ is said to shatter a set of data points (x_1, x_2, \dots, x_n) if, for all assignments of labels to those points, there exists a θ such that the model f makes no errors when evaluating that set of data points.

The VC dimension of a model f is the maximum number of points that can be arranged so that f shatters them.

3.2 Shatter coefficient(growth function)

分散系数量测了集合族的丰富性 (richness)。通过集合族对固定基数的集合的交的基数来定义 (对于二分类问题的假设空间, 通过对 n 个样本可以存在的最多分类数来表示):

定义 3.3. The growth function measures the size of $H \cap C$ as a function of $|C|$. Formally:

$$\text{Growth}(H, m) := \max_{C: |C|=m} |H \cap C|$$

从集合族 (或假设空间) 打散集合角度来定义, 分别是可以打散的最小集合数, 及给定集合能够打散的最大程度来定义

3.3 Rademacher complexity

Rademacher complexity 被用来量测一族实值函数的丰富性 (richness)。与 VC 维不同, 这一定义不需要限制函数为二值的。

定义 3.4. Given a sample $S = (z_1, z_2, \dots, z_m) \in Z^m$, and a class F of real-valued functions defined on a domain space Z , the empirical Rademacher complexity of F given S is defined as:

$$\text{Rad}_S(F) = \frac{1}{m} \mathbb{E} \left[\sup_{f \in F} \sum_{i=1}^m \sigma_i f(z_i) \right]$$

This can also be written using the previous definition:

$$\text{Rad}_S(F) = \text{Rad}(F \circ S)$$

where $F \circ S := \{(f(z_1), \dots, f(z_m)) \mid f \in F\}$

这与 Rademacher complexity 在集合上的定义联系起来

定义 3.5. Let P be a probability distribution over Z . The Rademacher complexity of the function class F with respect to P for sample size m is:

$$\text{Rad}_{P,m}(F) := \mathbb{E}_{S \sim P^m} [\text{Rad}_S(F)]$$

向量的内积可以表示两个向量的关系程度, empirical Rademacher complexity 测量函数集 F 在样本集 S 上与 random noise 的关系程度。这样 complexity 越大的函数集就会有越多的函数, 能够更好的适应 random noise, 意味着泛化能力更强, 丰富度更高。通过函数空间与随机噪声的相关性来刻画函数空间的复杂度, 丰富性, 泛化能力

Rademacher complexity 可以被用来得到函数集在相关数据的可学习性的上界, 小复杂度的假设空间更容易被学习。

1. 这个概念的应用

- 可被用于框住可代表性（可代表性定义为真实损失与由样本得到的损失的误差对损失函数空间的上界），从而通过训练集的损失（误差）可以代表真实损失（误差）。

$$\mathbb{E}_{S \sim P^m} [\text{Rep}_P(F, S)] \leq 2 \cdot \mathbb{E}_{S \sim P^m} [\text{Rad}(F \circ S)]$$

- 可被用来框住泛化界。小复杂度的空间丰富度低，更容易利用经验误差最小在假设空间中学习到好的结果。例如对于 0-1 损失，对任意 $\delta > 0$ ，有 $1 - \delta$ 的概率成立

$$L_P(h) - L_S(h) \leq 2 \text{Rad}(F \circ S) + 4\sqrt{\frac{2 \ln(4/\delta)}{m}}$$

2. 如何限制复杂度上界。主要介绍一些复杂度的性质。

- 平移不变性
- 线性性。（集合元素均放大 c 倍，复杂度也放大 c 倍）
- 可加性
- 所有向量被某 Lipschitz 函数作用后的复杂度最多被影响 Lipschitz 常数那么多。压缩映射会严格降低复杂度。
- 凸包运算不改变复杂度
- 有限集的复杂度的界随集合基数成对数增长。（这个界可以被 VC 维表示）

3. 高斯复杂度，替换均匀分布为独立的标准正态

4 强化学习

4.1 强化学习

强化学习与监督学习，非监督学习是三种基本的机器学习范式。其与监督学习的不同之处在于不需要呈现带标签的输入/输出对，也不需要显式纠正次优动作。相反，重点是在探索（未知领域）和利用（当前知识）之间找到平衡。环境通常以马尔可夫决策过程（MDP）的形式陈述。经典动态规划方法和强化学习算法之间的主要区别在于，后者不假设 MDP 的确切数学模型，并且针对无法采用精确方法的大型 MDP。

基本的强化学习被建模为一个 Markov decision process (MDP):

- 一组环境和个体状态集， S
- 个体的一组行动即， A
- 在行动 a 下， t 时刻由状态 s 变为 $t+1$ 时刻的 s' 的概率为： $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$
- $R_a(s, s')$ 为实时回报

强化学习中的个体以离散的步长与环境交互。在 t 时刻，个体获得当前状态 s_t 与回报 r_t 。再在可选的行动集中选择 a_t ，决定个体在下一时刻的状态 s_{t+1} 与回报 r_{t+1} 。强化学习个体的目标是学习

策略

$$\pi : A \times S \rightarrow [0, 1], \pi(a, s) = \Pr(a_t = a \mid s_t = s)$$

使得累计回报最大。

如果重述问题为个体可以直接观察到当前环境状态，此时问题是完全能观的。否则，如果只能观测到部分状态或者带噪声的观测，称为部分可观的。这时需建模为部分可观察的马尔可夫决策过程，可通过限制个体的可选行动集来处理。

强化学习适合包含长期与短期奖励权衡的问题。使其有效的两个方面是：利用样本获得最大表现；利用函数逼近来处理大环境，例如

- 环境模型是已知的，但是没有解析解决方案
- 仅给出环境的仿真模型（基于仿真的优化的主题）
- 收集有关环境的信息的唯一方法是与环境进行交互

4.2 方法

- 一种方法是 ϵ -贪婪，其中 $0 < \epsilon < 1$ 是控制勘探量与开发量的参数。有概率 $1 - \epsilon$ 选择开发，其中个体选择它认为具有最佳长期效果的行动；有 ϵ 概率选择探索，然后随机均匀地选择动作。 ϵ 通常是固定参数，也可以根据计划进行调整，也可以根据启发式方法进行自适应调整。
- 状态可观测，问题也在于如何利用过去的经验来找出哪些行为会导致更高的累积奖励。

5 元学习

参考：[#a-simple-view](https://wei-tianhao.github.io/blog/2019/09/17/meta-learning.html)

5.1 N-way K-shot Q-query

假设我们有任务的分布，从分布中采样了许多任务作为训练集。元学习模型期待在这些任务训练后，能在空间所有任务具有良好表现。设 D 为一个数据集，数据集包括特征向量 x 和标签 y ，分布记为 $p(D)$ ，则参数设置为：

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{D \sim p(D)} [\mathcal{L}_{\theta}(D)]$$

其与一般学习任务的区别是：一般学习任务每个样本是一条数据；元学习模型每个样本是数据集（任务）。

数据集 D 经常被划分为两部分，一个用于学习的支持集（support set） S ，和一个用于训练和测试的预测集（prediction set） B ，即 $D = \langle S, B \rangle$ 。

小样本学习类似 Fine-tuning，在小规模的支撑集上快速学习，减少在预测集上的误差。步骤如下：

- 随机采样支撑集和查询集，其标签均相同；
- 将支撑集当作模型输入，进行快速学习；（采用不同的学习算法）
- 将查询集作为输出，计算在查询集上的 loss，根据 loss 进行反向传播更新参数

$$\theta = \arg \max_{\theta} E_{L \subset \mathcal{L}} \left[E_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}} \left[\sum_{(x,y) \in B^L} P_{\theta}(x, y, S^L) \right] \right]$$

一定程度上类似于对某个只有少量数据的任务，使用在相关任务的大数据集上预训练的模型，然后进行 fine-tuning。不同的是，元学习目标是让模型优化以后能够在多个任务上表现的更好。

5.2 学习器与元学习器角度

把模型的更新划分为两个阶段：

- 在样本上直接进行，完成任务，例如分类，回归，作为学习器模型；
- 定义支撑集是一堆任务，根据支撑集学习如何更新学习器模型的参数

$$\mathbb{E}_{L \subset \mathcal{L}} \left[\mathbb{E}_{S^L \subset \mathcal{D}, B^L \subset \mathcal{D}} \left[\sum_{(x,y) \in B^L} P_{g_{\phi}(\theta, S^L)}(y | x) \right] \right]$$

元学习通常被用在：优化超参数和神经网络、探索好的网络结构、小样本图像识别和快速强化学习等

5.3 训练过程

利用 MAML 先预训练一个数学模型 M_{meta} ，再在这个基础上训练 $M_{fine-tuning}$

Algorithm 1 Model-Agnostic Meta-Learning

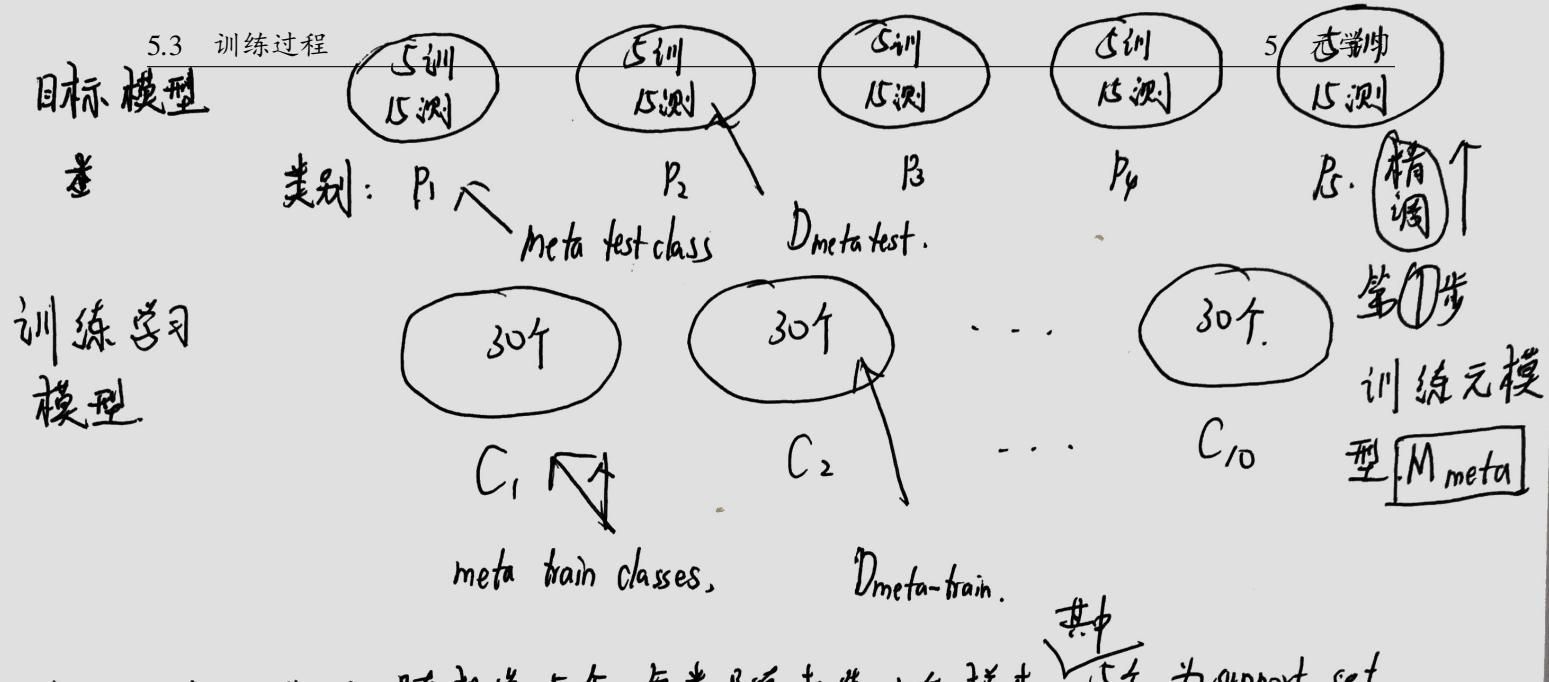
Require: $p(\mathcal{T})$: distribution over tasks

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**
-

图 2: MAML 算法

元学习 (MAML)



Meta 阶段: $C_1 - C_{10}$ 随机选 5 个. 每类随机选 20 个样本. \checkmark 其中 5 个为 support set

另外 15 个为 query test.

将整个任务当作普通神经网络模型的一条训练数据.

$M_{\text{fine-tune}}$ 阶段: 相同.

MAML, Meta 阶段

1. 随机采样若干个任务, 形成一个 batch
2. 每次任务进行一次 ~~更新~~ 更新参数: 具体为用 task 中的 support set. (NK 个样本)
计算梯度, 进行 ~~更新~~ 更新
3. 第二次梯度更新, 计算一个 batch 的 loss 总和, 采用 SGD.
参与计算的样本为 query set.

Meta fine-tune 阶段.

- 差别 1. 利用 M_{meta} 参数.
2. 只抽取一次 task 学习, 用 support set 训练, query set 测试.

6 小样本学习

参考: <https://zhuanlan.zhihu.com/p/61215293> 以及综述文章: Learning from Very Few Samples: A Survey.

6.1 介绍

人类智能一个深刻的标志是可以通过很小的示例快速建立对新颖概念的认知。例如，人类可以通过很少的图像来识别视觉对象。深度学习作为机器学习的重要里程碑已经在广泛领域（语言，语音）取得成果。其成果一般可以归因为：强大计算资源；复杂多层的神经网络及大规模数据集。但是，由于诸如隐私，安全性或数据的高标签成本等一些因素，许多现实的应用场景（例如在医学，军事和金融领域）不允许我们访问足够的带标签的训练样本。如何使学习系统能够从很少的样本中高效地学习和推广是研究人员当下所追求的。研究小样本学习的三个理论和实践意义来自三个方面：

- 将不会依赖大规模训练样本，避免特定应用下数据准备的高昂成本
- 可以缩小人类智能和人工智能的距离
- 为新出现的只有少量样本的任务实现低成本的模型设置

从泛化误差的角度理解小样本学习与传统机器学习的不同：

假设我们有有监督训练集 $\mathcal{D}_t = \{(x_i, y_i)\}_{i=1}^n$ 来自分布 $P_{X \times Y}$ ，我们期望最小的损失为：

$$\mathcal{E}_{\text{ex}} = \mathbb{E}_{(x,y) \sim P_{X \times Y}} L(f(x), y)$$

但是数据分布事先未知，使用样本替代的经验误差为：

$$\mathcal{E}_{\text{em}} = \mathbb{E}_{(x,y) \sim \mathcal{D}_t} L(f(x), y)$$

因此泛化误差是衡量学习算法好坏的重要标准。有下面的问题：

- 假设空间太大，将导致更大的泛化误差，使得过拟合频发
- 样本集包含更多有标签样本，对 f 有更多限制，会有更好的泛化能力；
- 相反，少量的样本会导致更差的泛化性能。

综上，对于少样本，传统的学习技巧将面临严重的过拟合。值得强调，FSL 与零样本学习是不相关的，零样本是借助与概念相关的辅助信息（例如利用语义信息）来支持跨概念的知识传递（0 代表对于要分类的对象一次也不学习）<https://zhuanlan.zhihu.com/p/98136045>。而小样本学习是获取少量的样本就可以很快的训练出泛化性能好的模型。

定义 6.1. (Few Shot Learning) 给定一个由数据集 D_T 描述的任务 T （其中 D_T 只包含少量的样本），以及与任务 T 无关的辅助数据集 D_A （无关指他们的数据目标不交，从这个角度也可以描述与传统机器学习的差别，即任务的目标与已有知识是否有交），FSL 目标是只通过少量样本的 D_T 以及 D_A

上得到的知识，为任务 T 建立一个从输入到目标输出的函数 f 。

小样本学习分类：

- Semi-supervised FSL (半监督)：训练集包含无标签的样本
- Unsupervised FSL (无监督)： D_A 无标签
- Cross-domain FSL (跨域)： D_A 和 D_T 来自不同的域
- Generalized FSL (广义小样本)：测试样本集中包含 D_A 中元素。即标签空间为 $\mathcal{Y}_T \cup \mathcal{Y}_A$
-

对于判别模型，有三个主要的方法：增强，度量学习，元学习。

1. 增强（是否有额外的可监督信息：如属性注释，词向量：语义信息）

- 有监督增强
- 无监督增强

2. 元学习：由学会学习怎样的目标分为

- Learn-to-Measure,—metric learning
- Learn-to-Finetune,
- Learn-to-Parameterize,
- Learn-to-Adjust
- Learn-to-Remember.

3.

6.2 判别模型方法

1. 增强：增加训练数据集，增强数据多样性。如 rotating, flipping, cropping, translation, and adding noise (旋转，翻转，裁剪，平移和添加噪声)。额外的信息例如词向量，属性等可以被考虑。
2. 度量学习：度量学习的一般目标是学习一种相似性度量 $\mathcal{S}(\cdot, \cdot)$ ，使得相似样本有较高的取值，非相似样本获得较低的得分。在辅助数据集 D_A 上创建相似性度量，并推广到包含新类别的任务 T 上。其中相似性度量可以是简单的度量、复杂的网络和其他可行的模块或算法，用来估计样本或特征之间的成对相似性。
3. 元学习：Learn to learn

元学习方法在两个阶段处理 FSL 问题：元训练和元测试。在元训练中，模型会作用于在辅助数据集 D_A (也称为"episode") 上构建的许多独立的监督任务 $T \sim p(T)$ ，以学习如何适应未来相关任务，其中 $p(T)$ 是任务的分布，训练中此外所有任务都来自 T 并遵循相同任务范式。例如，所有任务都是 C-way K-shot 问题。每个元训练任务 T 都包含一个特定于任务的数据集 $D_T = \{D_{\text{trn}}, D_{\text{tst}}\}$ 。在元测试中，模型在新任务 $T \sim p(T)$ 上测试，该任务的标签空间与元训练

期间看到的标签不相交。在大多数情况下， D_{trn} 被称为支持集，而 D_{tst} 被称为查询集。元学习的目的是找到使所有任务中的损失最小的模型参数：

$$\min_{\theta} \mathbb{E}_{T \sim P(T)} L(D_T; \theta)$$

Learn-to-Measure

L2M 方法本质上继承了度量学习的主要思想，但在实现级别上基于度量学习的 FSL 方法有所不同：L2M 方法采用元学习策略来学习相似性度量。一些 L2M 的方法有：匹配网络，原型网络，关系网络。

L2M 的基本过程为：对一个任务 T ，对于任务 T ，让 x_i 是 D_{trn} 中的支持样本， x_j 是 D_{tst} 中的查询样本，并且让 $f(\cdot; \theta_f)$ 和 $g(\cdot; \theta_g)$ 分别是将支持和查询样本映射到特征空间的嵌入模型。此外，所有 L2M 方法都包含度量模块 $S(f; g; \theta_S)$ ，用于测量支持样本和查询样本之间的相似性，该度量可以是无参数距离度量（例如，欧几里得距离，余弦距离）或可学习的网络。该度量模块输出的相似度用于形成查询样本的最终预测概率。

- 原型网络：

可获得的数据集包含一些支持和查询样本，这些样本属于少数几个非任务类。嵌入模型 f 和 g 是相同的卷积神经网络，相似度 S 是通过欧氏距离实现的。KNN 用于最后的分类任务。同类支持样本的特征表示中心作为该类的原型。

$$p_c = \frac{1}{K} \sum_{(x_i, y_i) \in D_{trn}} \mathbb{1}(y_i == c) f(x_i; \theta_f)$$

新样本属于各类的概率（得分）为：

$$P(y_j = c | x_j) = \frac{\exp(-d(g(x_j; \theta_g), p_c))}{\sum_{c'=1}^C \exp(-d(g(x_j; \theta_g), p_{c'}))}$$

- 匹配网络（参考<https://github.com/karpathy/paper-notes/blob/master/matchingnetworks.md>）

匹配网络预测的测试集样本属于各类的概率，是通过这个样本和每个支撑集样本每一类中样本的归一化后的余弦相似度的和做为其属于该类的概率，

$$p(\hat{y}_j | x_j, D_{trn}) = \sum_{(x_i, y_i) \in D_{trn}} a(x_j, x_i) \cdot \mathbf{y}_i$$

其中 \mathbf{y}_i 是 C 维的 one-hot 标签（task 为 C -way），并且

$$a(x_j, x_i) = \frac{\exp(c(g(x_j; \theta_g), f(x_i; \theta_f)))}{\sum_{(x,y) \in D_{trn}} \exp(c(g(x_j; \theta_g), f(x; \theta_f)))}$$

匹配网络与原型网络不同点在于：

- 匹配网络嵌入模型 f 和 g 是不同的。具体来说， f 是 CNN 和 LSTM 的结合，旨在为少数支持样本实现完全上下文嵌入（FCE）， g 是和 f 相关的模型，可根据内容生成查询样本的特征注意机制；

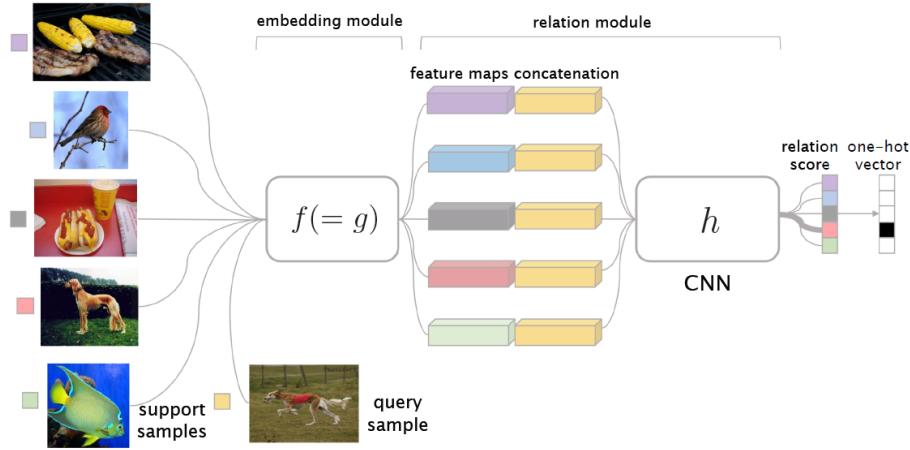


图 3: 关系网络过程

- 匹配网络相似性度量选择的是余弦相似度。
- 关系网络 (Relation Net)

关系网采用了可以学习的 CNN (用 $h(\cdot; \theta_h)$ 表示) 来量测成对相似性。网络将支持样本 x_i 和查询样本 x_j 的特征图 (与前面不同, 得到的是特征图) 的级联作为输入, 并输出他们的关系得分, 如图8。

7 Margin

7.1 Large Margin Softmax Loss

假设第 i 个输入的特征为 x_i , 相应的标签为 y_i , 原始的 Softmax loss 可以被写为:

$$L = \frac{1}{N} \sum_i L_i = \frac{1}{N} \sum_i -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

其中 f_j 表示第 j 个样本各类得分, 具体写为:

$$L_i = -\log \left(\frac{e^{\|\mathbf{W}_{y_i}\| \|x_i\| \cos(\theta_{y_i})}}{\sum_j e^{\|\mathbf{W}_j\| \|x_i\| \cos(\theta_j)}} \right)$$

考虑二分类。对于属于 1 类的样本 x , 其被分类正确等价于:

$$\|\mathbf{W}_1\| \|x\| \cos(\theta_1) > \|\mathbf{W}_2\| \|x\| \cos(\theta_2)$$

但划分边界处的样本容易受到扰动而分类错误, 因此训练时进一步要求成立:

$$\|\mathbf{W}_1\| \|x\| \cos(m\theta_1) > \|\mathbf{W}_2\| \|x\| \cos(\theta_2) \left(0 \leq \theta_1 \leq \frac{\pi}{m} \right)$$

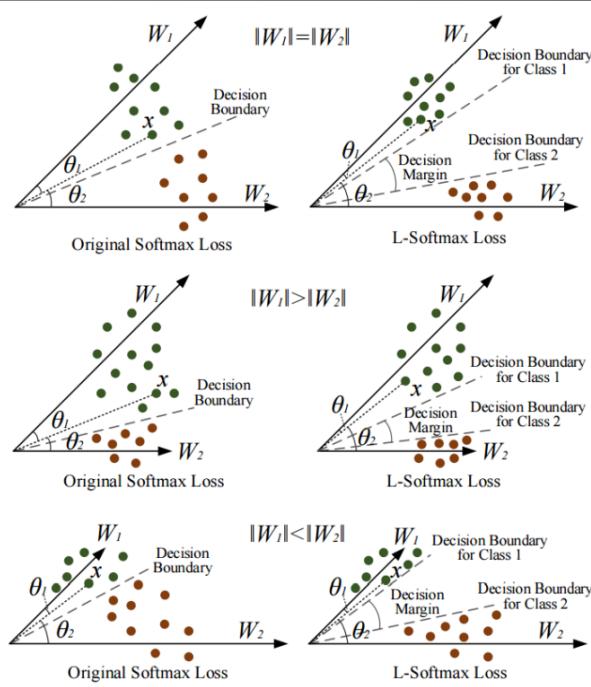


图 4: Large Soft margin

借用余弦函数的单调性 ($(0, \pi)$ 上), 有分类正确的结果, 即

$$\|\mathbf{W}_1\| \|\mathbf{x}\| \cos(\theta_1) \geq \|\mathbf{W}_1\| \|\mathbf{x}\| \cos(m\theta_1) > \|\mathbf{W}_2\| \|\mathbf{x}\| \cos(\theta_2)$$

综上, 设置如下的损失函数:

$$L_i = -\log \left(\frac{e^{\|\mathbf{W}_{y_i}\| \|\mathbf{x}_i\| \psi(\theta_{y_i})}}{e^{\|\mathbf{W}_{y_i}\| \|\mathbf{x}_i\| \psi(\theta_{y_i})} + \sum_{j \neq y_i} e^{\|\mathbf{W}_j\| \|\mathbf{x}_i\| \cos(\theta_j)}} \right)$$

其中 $\psi(\theta) = (-1)^k \cos(m\theta) - 2k$, $\theta \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m} \right]$ 。这样设置的目的是让 $\psi(\theta)$ 是一个减函数, 感觉上是减少损失函数能利于 $\theta < \pi/m$, 达到创造边界的目的, 使类内更紧凑, 类之间更散。最后

$$\begin{aligned} \cos(m\theta_{y_i}) &= C_m^0 \cos^m(\theta_{y_i}) - C_m^2 \cos^{m-2}(\theta_{y_i}) (1 - \cos^2(\theta_{y_i})) \\ &\quad + C_m^4 \cos^{m-4}(\theta_{y_i}) (1 - \cos^2(\theta_{y_i}))^2 + \dots \\ &\quad (-1)^n C_m^{2n} \cos^{m-2n}(\theta_{y_i}) (1 - \cos^2(\theta_{y_i}))^n + \dots \end{aligned}$$

其中 $\cos(\theta_j)$ 用 $\frac{\mathbf{W}_j^T \mathbf{x}_i}{\|\mathbf{W}_j\| \|\mathbf{x}_i\|}$ 代替。

7.2 Cosface:large margin cosine loss

和 Large Margin Softmax Loss 的做法类似, 我们通过 L_2 正则使得 $\|\mathbf{W}_j\| = 1$, 由于测试阶段我们一般采用余弦相似度距离, 这将与样本的范数无关, 由此启发我们在训练阶段的样本范数对得分函

数没有影响，故假设 $\|x\| = s$ ，因此损失函数变为：

$$L_{ns} = \frac{1}{N} \sum_i -\log \frac{e^{s \cos(\theta_{y_i, i})}}{\sum_j e^{s \cos(\theta_{j, i})}}$$

这样，需要优化的从内积变为了角度。上述的损失有个问题：只注重正确的分类，因此考虑添加余弦边界，一个直接的想法是直接给 $\cos\theta$ 减去一个常数 m ，得到

$$L_{lmc} = \frac{1}{N} \sum_i -\log \frac{e^{s(\cos(\theta_{y_i, i})-m)}}{e^{s(\cos(\theta_{y_i, i})-m)} + \sum_{j \neq y_i} e^{s \cos(\theta_{j, i})}}$$

同时满足

$$W = \frac{W^*}{\|W^*\|}$$

$$x = \frac{x^*}{\|x^*\|}$$

$$\cos(\theta_j, i) = W_j^T x_i$$

7.3 Normface

将样本特征归一化，目的是消除样本特征之间的数量差异对结果的影响。此外可以将权重与特征全部正则化到范数为某个 l 。

8 相似性距离

机器学习的大多数研究，无论是理论研究还是经验研究，都假设使用从固定分布中获得的数据对模型进行了训练和测试。已经对该单一域设置进行了充分研究，并且统一收敛理论保证了在这种假设下模型的经验训练误差接近其真实误差。但是，在许多实际情况下，我们希望在一个或多个源域中训练模型，然后将其应用于其他目标域。例如，我们可能有一个垃圾邮件过滤器，该过滤器是从一组当前用户（源域）收到的大型电子邮件集中训练而来的，并希望使其适应新用户（目标域）。

领域适应就是解决这样的问题：有大量的从一个或多个源分布中提取的标记训练数据，而很少或没有从目标分布中提取的标记训练数据。为完成该问题，我们需要根据分类器的源域错误和两个域之间的差异度量来对分类器的目标域错误进行界定。在无分布的情况下，我们无法使用 L_1 之类的度量，当面对复杂度有限的分布，利用由分类器引起的散度去度量这些差异可以达到需要的效果。

8.1 $\mathcal{H}\Delta\mathcal{H}$ 距离

一个自然的描述分布差异的距离可以是如下的 L_1 差异：

$$d_1(\mathcal{D}, \mathcal{D}') = 2 \sup_{B \in \mathcal{B}} |\Pr_{\mathcal{D}}[B] - \Pr_{\mathcal{D}'}[B]|$$

这里涉及到了对所有可测集的上确界，但我们只关心假设空间分类的错误率，因此限制取上确界的范围，只需让其可以表征假设空间分类的错误率。

定义 8.1. (\mathcal{H} 距离) 给定域 X 及其上的分布 \mathcal{D} 和 \mathcal{D}' , \mathcal{H} 是一个假设空间, I 是示性函数, 则这两个分布的 \mathcal{H} 距离定义为:

$$d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2 \sup_{h \in \mathcal{H}} |\Pr_{\mathcal{D}}[I(h \neq y)] - \Pr_{\mathcal{D}'}[I(h \neq y)]|$$

这个距离在有限 VC 维的假设空间上可以被有限样本估计（指用样本估计的距离与真实距离差可以很小），并且这个距离小于上述的 L_1 差异。对于二分类，特别地定义 $\mathcal{H}\Delta\mathcal{H}$ 空间与距离。

定义 8.2. ($\mathcal{H}\Delta\mathcal{H}$ 假设空间)

$$g \in \mathcal{H}\Delta\mathcal{H} \iff g(\mathbf{x}) = h(\mathbf{x}) \oplus h'(\mathbf{x}) \quad \text{for some } h, h' \in \mathcal{H}$$

可以使用 $\mathcal{H}\Delta\mathcal{H}$ 距离来给出目标域上误差的界，即

$$\epsilon_T(h) \leq \epsilon_S(h) + \frac{1}{2} \hat{d}_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{U}_S, \mathcal{U}_T) + 4 \sqrt{\frac{2d \log(2m') + \log\left(\frac{2}{\delta}\right)}{m'}} + \lambda$$

进一步，为了可以讨论更一般的任务，可以用损失函数来代替其中的误差:

定义 8.3. ((Discrepancy Distance)) H 是 X 到 Y 的假设空间, L 是 $Y \times Y$ 到 R_+ 的损失函数, 则分布 Q_1 和 Q_2 的距离为

$$\text{disc}_L(Q_1, Q_2) = \max_{h, h' \in H} |\mathcal{L}_{Q_1}(h', h) - \mathcal{L}_{Q_2}(h', h)|$$

但这样定义需要的损失函数 L 需满足有界, 对称和三角不等式才能有较好的结果。许多被广泛使用的 loss, 如 Margin loss 不满足这些条件。此外, 对于多分类的 $\mathcal{H}\Delta\mathcal{H}$ 距离被定义为:

$$d_{\mathcal{H}\Delta\mathcal{H}} = \sup_{h, h' \in \mathcal{H}} |\mathbb{E}_Q \mathbf{1}[h' \neq h] - \mathbb{E}_P \mathbf{1}[h' \neq h]|$$

这个距离的缺点是第一只考虑了分类器的分类结果, 没有考虑得分函数的取值 (即没有考虑边界损失); 第二需要遍历空间 $\mathcal{H}\Delta\mathcal{H}$ 。

8.2 MDD

因此, 需要考虑的有两点: 第一, 如何加入边界损失的信息; 第二, 如何将在 $\mathcal{H}\Delta\mathcal{H}$ 上的遍历转换为 \mathcal{H} 的遍历。

定义 0 – 1 损失与经验版本为:

$$\begin{aligned} \text{disp}_D(h', h) &\triangleq \mathbf{E}_D \mathbf{1}[h' \neq h] \\ \text{disp}_{\widehat{D}}(h', h) &\triangleq \mathbb{E}_{\widehat{D}} \mathbf{1}[h' \neq h] = \frac{1}{n} \sum_{i=1}^n \mathbf{1}[h'(x_i) \neq h(x_i)] \end{aligned}$$

由此定义 Disparity Discrepancy:

定义 8.4. (Disparity Discrepancy) 给定假设空间 \mathcal{H} , 对一个固定的假设 h , 定义

$$\begin{aligned} d_{h,\mathcal{H}}(P,Q) &\triangleq \sup_{h' \in \mathcal{H}} (\text{disp}_Q(h', h) - \text{disp}_P(h', h)) \\ &= \sup_{h' \in \mathcal{H}} (\mathbb{E}_Q \mathbf{1}[h' \neq h] - \mathbb{E}_P \mathbf{1}[h' \neq h]) \end{aligned}$$

以及经验版本:

$$d_{h,\mathcal{H}}(\widehat{P}, \widehat{Q}) \triangleq \sup_{h' \in \mathcal{H}} (\text{disp}_{\widehat{Q}}(h', h) - \text{disp}_{\widehat{P}}(h', h))$$

带边界损失的情形: 加边界是为了考虑得分函数次最大的取值。边界定义为:

$$\rho_f(x, y) \triangleq \frac{1}{2} \left(f(x, y) - \max_{y' \neq y} f(x, y') \right)$$

相关的边界损失为:

$$\begin{aligned} \text{err}_D^{(\rho)}(f) &\triangleq \mathbb{E}_{x \sim D} \Phi_\rho \circ \rho_f(x, y) \\ \text{err}_{\widehat{D}}^{(\rho)}(f) &\triangleq \mathbb{E}_{x \sim \widehat{D}} \Phi_\rho \circ \rho_f(x, y) = \frac{1}{n} \sum_{i=1}^n \Phi_\rho(\rho_f(x_i, y_i)), \end{aligned}$$

其中

$$\Phi_\rho(x) \triangleq \begin{cases} 0 & \rho \leq x \\ 1 - x/\rho & 0 \leq x \leq \rho \\ 1 & x \leq 0 \end{cases}$$

一个直接的想法是: 如果选用更加光滑的描述边界差异的函数 Φ 是否有较好的结果, 如果有, 如何加

由此, 加入边界后的 margin disparity 为:

$$\begin{aligned} \text{disp}_D^{(\rho)}(f', f) &\triangleq \mathbb{E}_D \Phi_\rho \circ \rho_{f'}(\cdot, h_f) \\ \text{disp}_{\widehat{D}}^{(\rho)}(f', f) &\triangleq \mathbb{E}_{\widehat{D}} \Phi_\rho \circ \rho_{f'}(\cdot, h_f) \\ &= \frac{1}{n} \sum_{i=1}^n \Phi_\rho \circ \rho_{f'}(x_i, h_f(x_i)) \end{aligned}$$

定义 8.5. (Margin Disparity Discrepancy) 给定假设空间 \mathcal{H} , 对一个固定的假设 h , 定义

$$\begin{aligned} d_{f,\mathcal{F}}^{(\rho)}(P,Q) &\triangleq \sup_{f' \in \mathcal{F}} (\text{disp}_Q^{(\rho)}(f', f) - \text{disp}_P^{(\rho)}(f', f)) \\ d_{f,\mathcal{F}}^{(\rho)}(\widehat{P}, \widehat{Q}) &\triangleq \sup_{f' \in \mathcal{F}} (\text{disp}_{\widehat{Q}}^{(\rho)}(f', f) - \text{disp}_{\widehat{P}}^{(\rho)}(f', f)) \end{aligned}$$

容易得到一个上界为:

$$\text{err}_Q(h_f) \leq \text{err}_P^{(\rho)}(f) + d_{f,\mathcal{F}}^{(\rho)}(P, Q) + \lambda, \quad \lambda = \min_{f^* \in \mathcal{H}} \left\{ \text{err}_P^{(\rho)}(f^*) + \text{err}_Q^{(\rho)}(f^*) \right\}$$

这个界可以被解释为:

- 第一项表示在源域上训练的结果
- 第二项表示领域转换带来的差异

8.2 MDD

- 第三项是表征了适应性

下面的任务是寻找 MDD 的界。定义

$$\Pi_1 \mathcal{F} = \{x \mapsto f(x, y) \mid y \in \mathcal{Y}, f \in \mathcal{F}\}$$

$$\Pi_{\mathcal{H}} \mathcal{F} \triangleq \{x \mapsto f(x, h(x)) \mid h \in \mathcal{H}, f \in \mathcal{F}\}$$

这两个集合都是将得分函数从向量值函数变为实值函数。

引理 8.1. 对于任意 $\delta > 0$, 以 $1 - 2\delta$ 概率成立:

$$\begin{aligned} & \left| d_{f,F}^{(\rho)}(\widehat{P}, \widehat{Q}) - d_{f,F}^{(\rho)}(P, Q) \right| \\ & \leq \frac{2k}{\rho} \mathfrak{R}_{n,P}(\Pi_{\mathcal{H}} \mathcal{F}) + \frac{2k}{\rho} \mathfrak{R}_{m,Q}(\Pi_{\mathcal{H}} \mathcal{F}) + \sqrt{\frac{\log \frac{2}{\delta}}{2n}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \end{aligned}$$

进一步可以得到目标域真实误差的上界:

定理 8.2. 对于任意 $\delta > 0$, 以 $1 - 3\delta$ 概率成立:

$$\begin{aligned} \text{err}_Q(h_f) & \leq \text{err}_{\widehat{P}}^{(\rho)}(f) + d_{f,\mathcal{F}}^{(\rho)}(\widehat{P}, \widehat{Q}) + \lambda \\ & + \frac{2k^2}{\rho} \mathfrak{R}_{n,P}(\Pi_1 \mathcal{F}) + \frac{2k}{\rho} \mathfrak{R}_{n,P}(\Pi_{\mathcal{H}} \mathcal{F}) + 2\sqrt{\frac{\log \frac{2}{\delta}}{2n}} \\ & + \frac{2k}{\rho} \mathfrak{R}_{m,Q}(\Pi_{\mathcal{H}} \mathcal{F}) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}} \end{aligned}$$

对于二分类问题 RC 复杂度是如何随着样本数的增加而变化的问题, 特别在权重集合有界的假设空间上, 利用 VC 维得到进一步的上界。

对于一般的情形, 考虑覆盖数: 可以覆盖假设空间的给定半径的球的最小数目。

下面考虑对 MDD 如何设计算法:

由于 MDD 中存在的上确界, 使 MDD 最小的问题是一个 minimax 问题, 可以写为:

$$\min_{f, \psi} \max_{f'} \text{err}_{\psi(\widehat{P})}^{(\rho)}(f) + \left(\text{disp}_{\psi(\widehat{Q})}^{(\rho)}(f', f) - \text{disp}_{\psi(\widehat{P})}^{(\rho)}(f', f) \right)$$

但由于 margin loss 容易造成梯度消失, 改用下面的损失代替:

$$\begin{aligned} \mathcal{E}(\widehat{P}) & = -\mathbb{E}_{(x^s, y^s) \sim \widehat{P}} \log [\sigma_{y^s}(f(\psi(x^s)))] \\ \mathcal{D}(\widehat{P}, \widehat{Q}) & = \mathbb{E}_{x^t \sim \widehat{Q}} \log [1 - \sigma_{h_f(\psi(x^t))}(f'(\psi(x^t)))] \\ & + \mathbb{E}_{x^s \sim \widehat{P}} \log [\sigma_{h_f(\psi(x^s))}(f'(\psi(x^s)))] \end{aligned}$$

为了能够解决 minimax 问题, 实现端到端的训练, 引入 GRL 层这与 DANN 的对抗的思想相同。

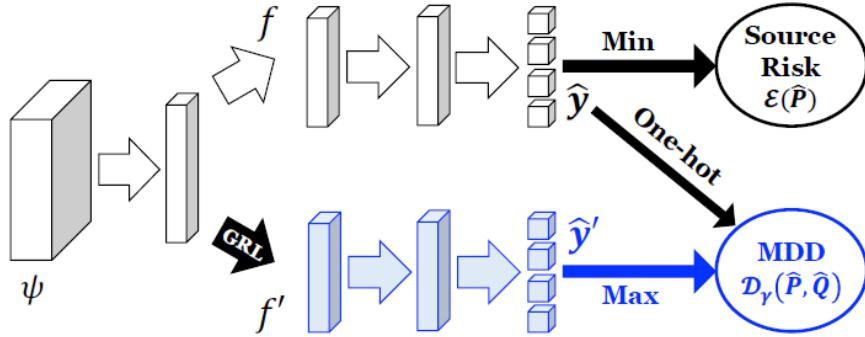


图 5: 替换后的 MDD 训练过程图

GRL 层满足在前向传播时不改变，反向传播时梯度自动取反，表示如下

$$\begin{aligned} R_\lambda(x) &= x \\ \frac{dR_\lambda}{dx} &= -\lambda I \end{aligned}$$

有两个想法，第一个 margin loss 采用的是线性函数，尝试换成平滑的非线性函数；第二个，这类领域适应假设源域与目标域有相同的标签空间，如果标签空间不同，能否得到相应的泛化上界。如果限制分类器的形式为 KNN，定义新的 MDD 为给定一个距离 d 后，对距离空间取极大得到的差异的差异，能否有改进的泛化上界

8.3 EMD 与 Wasserstein-距离

Wasserstein 距离或 Kantorovich-Rubinstein 度量是在给定度量空间 M 上的概率分布之间定义的距离函数。直观上，可以将每个分布看作堆积在地面上的一堆单位数量的土壤，这个距离的定义是将其中一个土堆变成另一个土堆的最小成本，其中成本被定义为需要移动的土量乘以必须移动的平均距离。

对于两个离散分布 P, Q , $P = \{(p_1, w_{p1}), (p_2, w_{p2}), \dots, (p_m, w_{pm})\}$, $Q = \{(q_1, w_{q1}), (q_2, w_{q2}), \dots, (q_n, w_{qn})\}$, 令 $D = [d_{i,j}]$ 为 p_i 到 q_j 的距离，最优化总运输成本：

$$\min \sum_{i=1}^m \sum_{j=1}^n f_{i,j} d_{i,j}$$

$$f_{i,j} \geq 0, 1 \leq i \leq m, 1 \leq j \leq n$$

$$\sum_{j=1}^n f_{i,j} \leq w_{pi}, 1 \leq i \leq m$$

$$\sum_{i=1}^m f_{i,j} \leq w_{qj}, 1 \leq j \leq n$$

$$\sum_{i=1}^m \sum_{j=1}^n f_{i,j} = \min \left\{ \sum_{i=1}^m w_{pi}, \sum_{j=1}^n w_{qj} \right\}$$

那么 EMD (Earth mover's distance) 距离被定义为

$$\text{EMD}(P, Q) = \frac{\sum_{i=1}^m \sum_{j=1}^n f_{i,j} d_{i,j}}{\sum_{i=1}^m \sum_{j=1}^n f_{i,j}}$$

上式中 f_{ij} 可以看作 (x, y) 处联合概率密度值, w_{pi} 是 x 处边缘概率密度, w_{qj} 是 y 出边缘概率密度值, d_{ij} 是 x, y 两点处距离, 用来表征成本函数。自然的推广到连续情形, 对于空间 X 上的分布 $\mu(x)$, 希望能用转化分布的方式将其运输到 $\nu(x)$ 的位置。当想要被移动到的分布与需要移动的分布有相同的质量的时候这个问题有意义, 假设这两个分布都是概率分布, 总质量为 1, 并且给定一个成本函数: $c(x, y) \mapsto [0, \infty)$ 。运输计划由 $\gamma(x, y)$ 来描述, 表示从 x 到 y 要移动的质量。可以想象为移动一堆 μ 组成的土堆到 ν 这样的孔中去, 为了使之有意义, 需满足:

$$\begin{aligned}\int \gamma(x, y) dy &= \mu(x) \\ \int \gamma(x, y) dx &= \nu(y)\end{aligned}$$

两式分别表示:

- 从点 x 移动出去的部分需要与这个点移动前存在的部分相等
- 移动进去 y 点的部分需要与这个点移动前缺失的部分相等

最优运输为这样的运输计划成本最低的计划。最优计划的成本为:

$$C = \inf_{\gamma \in \Gamma(\mu, \nu)} \int c(x, y) d\gamma(x, y)$$

完整定义为:

定义 8.6. (p^{th} Wasserstein distance) 设 (M, d) 是度量空间, 其上的每个概率测度都是 Radon 测度 (内开, 外闭, 局部值有限), 给 $p \geq 1$, 令 $P_p(M)$ 是 M 上 p 阶矩有限的概率测度全体。则在空间 $P_p(M)$ 上的测度 μ 和 ν 之间的 p^{th} Wasserstein 距离定义为

$$W_p(\mu, \nu) := \left(\inf_{\gamma \in \Gamma(\mu, \nu)} \int_{M \times M} d(x, y)^p d\gamma(x, y) \right)^{1/p}$$

其中, $\Gamma(\mu, \nu)$ 表示 $M \times M$ 上任意一个边缘为 μ, ν 的联合测度。

这个距离是一个自然的比较两个变量分布差异的度量。Wasserstein GAN” 中使用 Wasserstein-1 度量作为改进原始的对抗网络 (GAN) 框架的方法, 以减轻消失梯度和模式崩溃问题。特别是两个分布重叠部分很少的时候也可以做。(相比较 KL 散度和 JS 散度)

8.4 MMD

除了添加边界的办法, 还可以对输入空间做特征扩维, 把数据集映射到高维中去, 形成了特征空间, 并且默认原本在低维中线性不可分的数据集在足够高的维度中存在线性可分的超平面。

MD 均值差异:

$$MD(P, Q) = \left\| \frac{1}{|P|} \sum_{x \in P} \phi(x) - \frac{1}{|Q|} \sum_{y \in Q} \phi(y) \right\|$$

MMD 均值差异:

$$MMD(P, Q) = \sup_{f \in \mathcal{F}} \left\| \frac{1}{|P|} \sum_{x \in P} f(\phi(x)) - \frac{1}{|Q|} \sum_{y \in Q} f(\phi(y)) \right\|$$

优化目标:

$$\min_{\phi} L = \min_{\phi} J + \lambda MMD(P, Q) = \min_{\phi} J_{\phi} + \lambda \sup_{f \in \mathcal{F}} \left\| \frac{1}{|P|} \sum_{x \in P} f(\phi(x)) - \frac{1}{|Q|} \sum_{y \in Q} f(\phi(y)) \right\|$$

再生核希尔伯特上的 MMD

???????

9 参数迁移

10 DANN

迁移学习是利用已有的先验知识让算法来学习新的知识，即找到先验知识与新知识之间的相似性。领域适应是当前迁移学习领域中解决问题的主要思路。在目标域与源域的数据分布不同但任务相同的迁移学习是领域适应的任务，即减小源域和目标域的数据分布差异。DANN 借鉴了 GAN 的思想，图像分类器与域分类器在训练过程中相互对抗实现最终实现了图像分类损失与域分类损失之间的相互平衡。

在 DANN 训练，网络的输入为带图像分类标签的源域数据集与不带图像分类标签的目标域数据集，以及源域与目标域数据集的域分类标签。即我们知道源域数据集的图像分类标签，无目标域数据集的图像分类标签。具体步骤如下：

- 第一步，图像 x （包含带标签源域数据与不带标签目标域数据）通过特征提取网络 $G_f(x; \theta_f)$ 转换为特征向量；
- 第二步，由源域数据转换得到的特征向量通过源于分类预测器 $G_y(x; \theta_y)$ 得到预测标签，进而可以得到损失 L_y ；由全部数据，标签为属于源域或目标域构成的样本对，通过域判别器 $G_d(x; \theta_d)$ ，得到属于源域或目标域的预测，由此可以计算损失 L_d 。

训练过程中的任务主要为：

- 第一个则是实现源域数据集准确分类，实现现图像分类误差 L_y 的最小化
- 第二个任务则就是要混淆源域数据集和目标域数据集，实现域分类误差 L_d 的最大化，混淆目标域数据集与源域数据集。

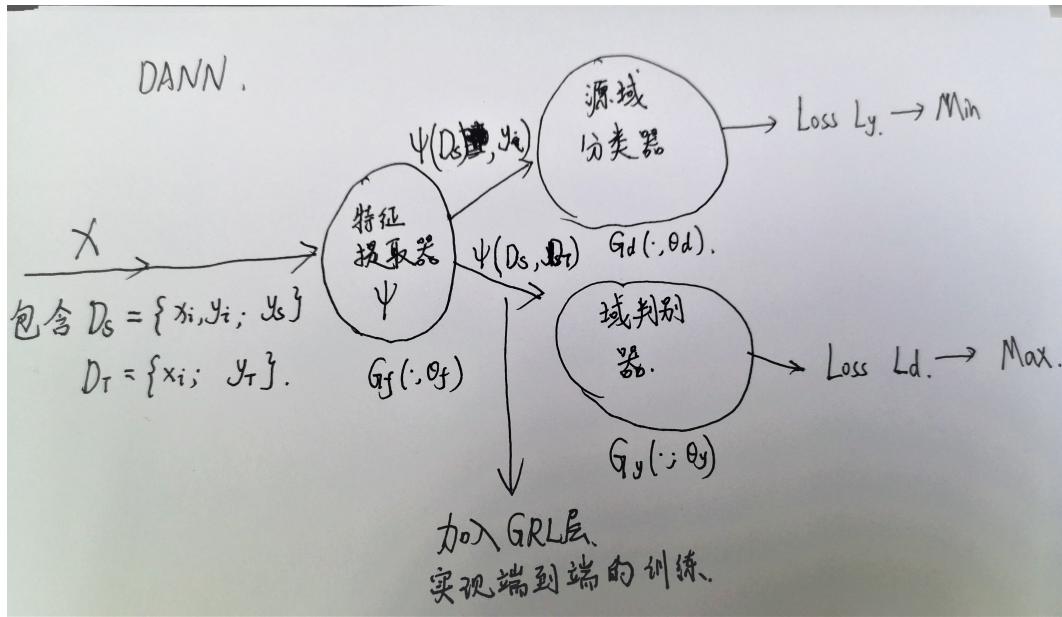


图 6: DANN 手写流程图

总的损失函数定义为:

$$\begin{aligned}
 E(\theta_f, \theta_y, \theta_d) &= \sum_{i=1, \dots, N} L_y(G_y(G_f(x_i; \theta_f); \theta_y), y_i) \\
 &\quad - \lambda \sum_{i=1, \dots, N} L_d(G_d(G_f(x_i; \theta_f); \theta_d), y_i) \\
 &= \sum_{i=1, \dots, N} L_y^i(\theta_f, \theta_y) - \lambda \sum_{i=1, \dots, N} L_d^i(\theta_f, \theta_d)
 \end{aligned}$$

相关的需要训练的参数为:

$$\begin{aligned}
 (\hat{\theta}_f, \hat{\theta}_y) &= \arg \min_{\theta_f, \theta_y} E(\theta_f, \theta_y, \hat{\theta}_d) \\
 \hat{\theta}_d &= \arg \max_{\theta_d} E(\hat{\theta}_f, \hat{\theta}_y, \theta_d)
 \end{aligned}$$

为了可以不使用 GAN 中固定其中一个模型更新另一个模型参数的方式训练，引入了 GRL (梯度反转层)。

11 MLMT

这个提供了一种训练阶段样本选择方法与训练策略，其将相同类别的样本与不同类别的样本使用不同的策略来训练:

- 具有不相干类集的两个元任务，类似源域与目标域的差别，采用 MDD 领域适应方法训练，使元学习模型对域差距更加稳健；
- 具有相同类集的两个元任务可以被用来学习对采样较差的样本也具有鲁棒性的模型，采用了

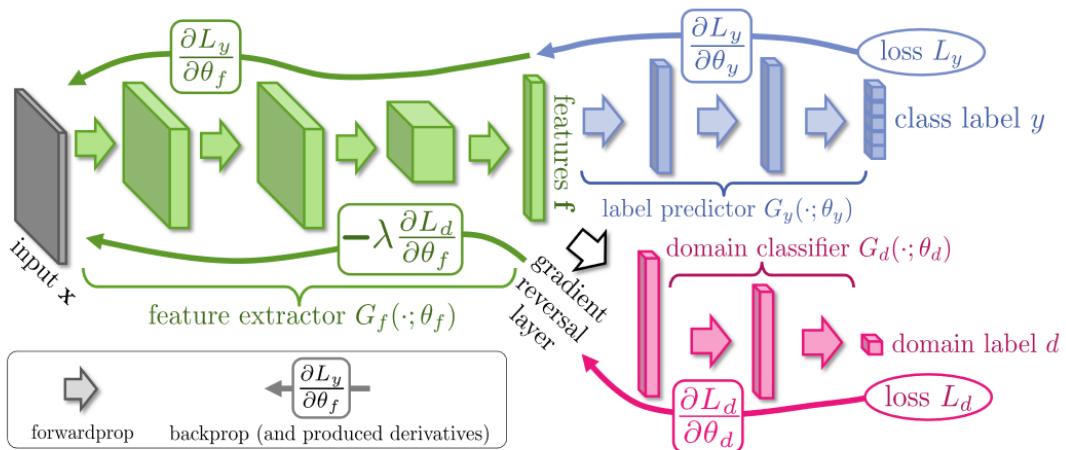


图 7: DANN 流程图

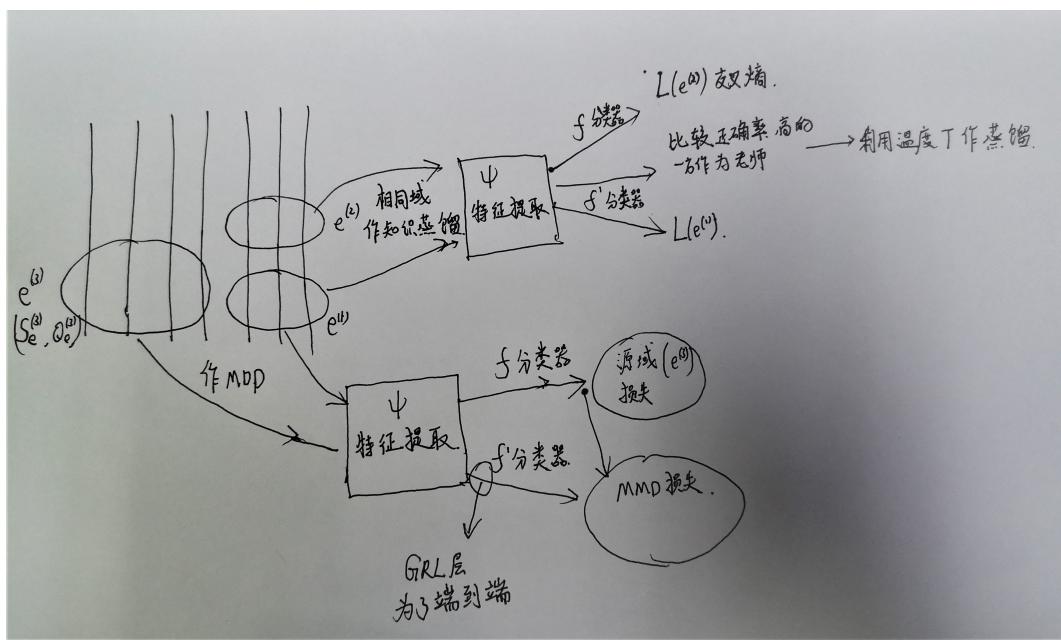


图 8: MLMT 流程图

一种知识蒸馏的方法

11.1 知识蒸馏

知识蒸馏是一种模型压缩方法，从已经训练好的大模型（老师）快速得到效果好的小模型的训练。具体损失为小模型的预测与真实值的交叉熵损失与小模型的预测与大模型预测结果的交叉熵损失的加权和。但是直接使用大模型的输出，由于其对正确答案的高的置信度，大模型训练得到的结果将很难传向小模型，因此对大模型的输出做个改变，以平滑预测的输出：

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

12 ResNet

ResNet 在 PyTorch 的官方代码中共有 5 种不同深度的结构，深度分别为 18、34、50、101、152（各种网络的深度指的是“需要通过训练更新参数”的层数，如卷积层，全连接层等）