目录

多目标优化总结：概念、算法和应用

张大快 @ 知乎, 微信公众号：simplex101

2021 年 3 月 3 日

多目标优化在推荐系统、物流配送、路径规划等中有广泛的应用。笔者近期调研了多目标优化领域的文献，将学习过程中的感想和心得记录下来，供后续翻阅。本系列将从以下几个方面介绍：

- 多目标优化的问题定义
- 帕累托解集的定义
- 多目标优化的经典算法，如线性加权、主要目标法等
- 多目标优化的梯度下降方法
- 多任务学习与多目标优化

1　多目标优化基础

本节将介绍多目标优化的问题定义，分别从单目标、多目标，无约束和有约束方面介绍。

1.1　无约束的单目标优化问题

无约束的单目标优化问题：

$$\min_x \ f(x), x \in R^N \tag{1}$$

1.2　无约束的多目标优化问题

无约束的多目标优化问题：

$$\min_x \ F(x) = [f_1(x), f_2(x), ..., f_K(x)] \tag{2}$$

其中，K 为子目标的个数，x 的取值为 N 维实数空间 R^N，$f_k(x)$ 为连续一阶可导函数的子目标函数。

1.3　带约束的单目标优化问题

带约束的单目标优化问题：

$$\begin{aligned} \min_x \quad & f(x) \\ s.t. \quad & g_i(x) \geq 0 \quad ,i \in [1, M] \\ & h_j(x) = 0 \quad ,j \in [1, L] \end{aligned} \tag{3}$$

其中 $s.t.$ 为 subject to 的缩写，表示受限于的意思。令 D 为上述多目标优化问题的可行域，即：

$$D = \{x | g_i(x) \geq 0, i \in [1, M], h_j(x) = 0, j \in [1, L]\} \tag{4}$$

1.4 带约束的多目标优化问题

带约束的多目标问题 MOO(mulit object optimization)：

$$\begin{aligned} \min_{x} \quad & F(x) = && [f_1(x), f_2(x), ..., f_K(x)] \\ & g_i(x) \geq 0 && , i \in [1, M] \\ & h_j(x) = 0 && , j \in [1, L] \end{aligned} \tag{5}$$

令 D 为上述多目标优化问题的可行域，即

$$D = \{x | g_i(x) \geq 0, i \in [1, M], h_j(x) = 0, j \in [1, L]\} \tag{6}$$

2 多目标优化的解集

对于多目标优化问题 MOO，通常不存在解 $x^* \in D$, 使得目标 $f_i(x)\ \forall i \in [1, K]$，同时达到最小值，因此单目标优化的最优解定义在 MOO 问题中不适用。

在 MOO 问题中，其解集可以通过 **绝对最优解、有效解和弱有效解** 来描述。

在描述 MOO 的解集之前，我们先来定义多目标里面的相等、严格小于、小于、小于且不相等的含义 [1]：设 R^N 为 N 维实向量空间，$y = (y_1, y_2, ..y_N)^T, z = (z_1, z_2, ..z_N)^T$ ：

$$\begin{cases} \text{相等} & y = z \Leftrightarrow y_i = z_i, i = 1, 2, ..., N \\ \text{严格小于} & y < z \Leftrightarrow y_i < z_i, i = 1, 2, ..., N \\ \text{小于} & y \leqq z \Leftrightarrow y_i \leqslant z_i, i = 1, 2, ..., N \\ \text{小于且不相等（支配）} & y \leqslant z \Leftrightarrow y_i \leqslant z_i, i = 1, 2, ..., N,\ y \neq z \end{cases} \tag{7}$$

接下来的解集按照 (7) 的记号来定义。

2.1 Pareto 支配 (Pareto Dominance)

定义：$\forall x_1, x_2 \in R^N$，如果对于所有的 $k = 1, ..., K$，都有 $f_k(x_1) \leqslant f_k(x_2)$，则称 x_1**支配**x_2。

2.2 Pareto 解集：绝对最优解

定义：设 $x^* \in D$，如果对于任意的 $x \in D$, 都有 $f(x^*) \leqq f(x)$，即对于所有的 $k = 1, ..., K$，都有 $f_k(x^*) \leqslant f_k(x)$，则 x^* 是 MOO 问题的 **绝对最优解**。

2.3 Pareto 解集：有效解

定义：设 $x^* \in D$，如果不存在 $x \in D$, 使得 $f(x) \leqslant f(x^*)$；即下面条件不成立：

$$f_k(x) \leq f_k(x^*), and\ \exists i, f_i(x) < f_i(x^*), i \in [1, K] \tag{8}$$

则 x^* 是 MOO 问题的 **有效解**。

有效解也叫 **帕累托最优解**，其含义是如果 x^* 是帕累托最优解，则找不到这样的可行解 $x \in D$，使得 $f(x)$ 的每个目标值都不比 $f(x^*)$ 的目标值坏，并且 $f(x)$ 至少有一个目标比 $f(x^*)$ 的相应目标值好。即 x^* 是最好的，不能再进行改进（帕累托改进）。

2.4 Pareto 解集：弱有效解

定义：设 $x^* \in D$，如果不存在 $x \in D$ ，使得 $f(x) < f(x^*)$，即

$$f_k(x) < f_k(x^*), and \quad ,\forall k \in [1, K] \tag{9}$$

则 x^* 是 MOO 问题的 **弱有效解**。

其含义是如果 x^* 是弱有效解，则找不到这样的可行解 $x \in D$，使得 $f(x)$ 的每个目标值都比 $f(x^*)$ 的目标值严格 ($<$) 的好。

2.5 Pareto 最优解集 (Pareto-optimal Set)

定义:给定 MOO 问题的有效解(帕累托最优解)构成的解集,称这个解集为 **Pareto 最优解集**Pareto-optimal Set，简称 PS。

即这个集合中的解是相互非支配的，也即两两不是支配关系。

2.6 Pareto 最优前沿 (Pareto-optimal front)

定义：Pareto-optimal Set 中每个解对应的目标值向量组成的集合称之为 **Pareto 最优前沿**(Pareto-optimal front)，简称为 PF：

$$PF = \{F(x) | x \in PS\} \tag{10}$$

如下图所示 [5]

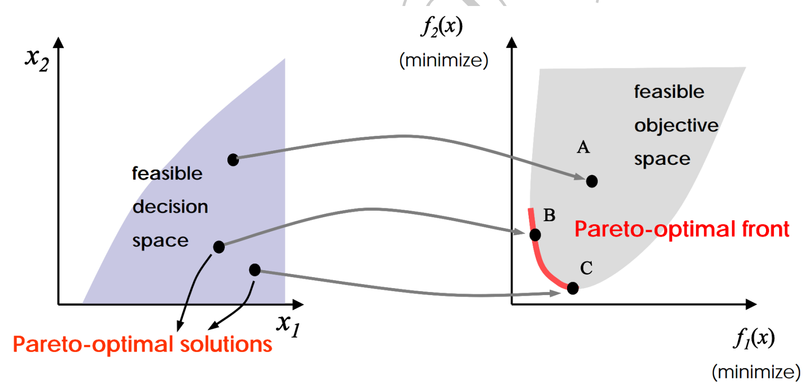

图 1: Pareto 最优前沿

2.7 多目标优化的最优性条件

约束规格定义：对优化问题的约束函数，附加某些限制条件，使得其最优解满足的最优性条件，参考 [4]。

下面给出一个严格条件下多目标优化的充分必要条件。给出的充要条件前，我们先引入了约束规格条件：

$$\begin{aligned} \min_{x \in \hat{D}} \quad F(x) &= \sum_{k=1}^{K} f_i(x) \\ \hat{D} &= x \in D | f(x) \le f(\hat{x}) \end{aligned} \tag{11}$$

定理：设 $f(x), g(x)$ 为凸函数，且在 $x \in D$ 处可微，$h(x)$ 为线性函数，且 $\hat{D} = x \in D|f(x) \leq f(\hat{x})$ 满足 KKT 约束规格，则 x^* 是 MOO 的有效解的充分必要条件是存在 $\lambda \in R^K, u \in R^M, v \in R^L$，使得

$$\begin{cases} \nabla_x L(x^*, \lambda^*, u^*, v^*) = \nabla f(x^*)\lambda^* + \nabla g(x^*)u^* + \nabla h(x^*)v^* = 0, \\ u^{*T} g(x^*) = 0, \\ \lambda^* > 0, u^* \geq 0. \end{cases} \tag{12}$$

定理的证明参考 [1]。

3 多目标优化的经典算法

3.1 线性加权法

线性加权法是多目标优化中使用比较广泛的方法，根据 $f_k(x)$ 的重要程度，设定权重进行线性加权，将多个目标表示成：

$$\begin{aligned} &\min_x \sum_{k=1}^{K} \lambda_k f_k(x) \\ \text{s.t. } &g_i(x) \geq 0, i \in [1, M] \\ &h_j(x) = 0, j \in [1, L] \end{aligned} \tag{13}$$

从而转换为单目标的优化问题。接下来我们给出在一定条件下，上述问题存在有效解的条件。

定理：对于给定的 $\lambda \in \Lambda^{++}$，则上述问题的最优解是 MOO 问题的有效解。其中：

$$\Lambda^{++} = \{\lambda | \lambda_k > 0, i = 1, 2 ... K, \sum_{k=1}^{K} \lambda_k = 1\} \tag{14}$$

详细证明参考 [1]。

线性加权法优缺点：

- 优点：实现简单，单目标优化问题有成熟的算法求解；
- 缺点：权重 λ_k 比较难确定，求出的解的优劣性没法保证。

3.2 主要目标法

除了上面介绍的线性加权法，主要目标法（也称 ϵ-约束方法），是一个应用广泛的算法：

$$\begin{aligned} &\min_x f_p(x) \\ \text{s.t. } &f_k(x) \leq \epsilon_k, k = 1, ..., K, \ k \neq p \\ &g_i(x) \geq 0, i \in [1, M] \\ &h_j(x) = 0, j \in [1, L] \end{aligned} \tag{15}$$

ϵ-约束方法从 K 个目标中选择最重要的子目标作为优化目标，其余的子目标作为约束条件。每个子目标，通过上界 ϵ_k 来约束。

3.2.1 主要目标法最优解和 MOO 的解集的关系

- 主要目标法最优解都是 MOO 的弱有效解；
- 若主要目标 $f_p(x)$ 是严格凸函数，可行域为 $\hat{D}$ 为凸集，则主要目标法的最优解是 MOO 问题的有效解。

3.2.2 界限值 ϵ_k 的选取

一般情况下，界限值可以取子目标函数的上界值：

$$\min\{f_k|f_k(x), k=1,...,K,\ k \neq p\} \leq \epsilon_k \tag{16}$$

这种取法可以使得某些 $f_k(x)$ 留在可行域 $\hat{D}$ 内，并且 $\hat{D}$ 内有较多的点靠近 $f_k(x)$ 的最优解。
主要目标法的优缺点对比：

- 优点：简单易行，保证在其他子目标取值允许的条件下，求出主要目标尽可能好的目标值；
- 缺点：ϵ_k 如果给的不合适的话，新的可行域 $\hat{D}$ 可能为空集。

3.3 逼近目标法

逼近目标法是让决策者提出一个目标值 $f^0=(f_1^0, f_2^0, ..., f_K^0)$，使得每个目标函数 $f_k(x)$ 都尽可能的逼近对应的目标值：

$$\begin{aligned} L(f(x), f^0) &= ||f(x)-f^0||_2^\lambda \\ &= \sum_{k=1}^{K} \lambda_k (f_k(x)-f_k^0)^2, \lambda \in \Lambda^{++} \end{aligned} \tag{17}$$

逼近目标法和机器学习中的损失函数类似，是一个单目标优化问题，可以通过经典的方法进行求解。这里求解的最优解和有效解及弱有效解没有直接的联系；逼近目标法反映了决策者希望的目标值。

4 梯度下降算法

前面列举的线性加权法、主要目标法和逼近目标优化法，都是采取先验的知识来将多目标优化问题简化为单目标优化问题，在一些严格的条件下，能够得到有效解（弱有效解）。能否有直接优化的方法来求解多目标优化问题呢？梯度下降就是这样一种算法。

4.1 最速下降方向

简单起见，我们将讨论问题限制在无约束最优化问题 (1) 上。并要求 (1) 中的 $f(x)$ 具有一阶连续偏导数。对于这类问题，我们希望能够从某一点出发，选择目标函数 $f(x)$ 下降最快的方向进行搜索，尽快达到最小值，那么下降最快的方向如何选择呢？

函数 $f(x)$ 在点 x 处沿方向 $d(d \in R^n)$ 的变化率可以用方向导数来描述

$$DF(x;d) = \nabla f(x)^T d \tag{18}$$

求解 $f(x)$ 在点 x 处下降最快的方向导数，可归结为求解如下的最优化问题：

$$\min \nabla f(x)^T d \tag{19}$$

$$\text{s.t. } ||d|| \leq 1 \tag{20}$$

其中 $||.||$ 为欧式范数。上述问题的解为：

$$d = -\frac{\nabla f(x)}{||\nabla f(x)||} \tag{21}$$

可以看出：负梯度方向为最速下降方向。最速下降法的迭代公式为：

$$x^{(t+1)} = x^{(t)} + \lambda_k d^{(k)} \tag{22}$$

其中 λ_k 可以通过一维搜索来得到。

4.2 多目标梯度下降算法

设当前为 $t+1$ 轮迭代，梯度迭代公式：

$$x^{(t+1)} = x^t + \lambda \cdot d^t \tag{23}$$

多目标优化的方向导数：

$$\nabla f_k(x)^T d, k = 1, ..., K \tag{24}$$

定义最大方向导数：

$$M_x(d^t) := max\{\nabla f_k(x)^T d^t | k = 1, ..., K\} \tag{25}$$

多目标问题的最速下降方向，可以归结为求解如下的最优化问题：

$$\min M_x(d^t) + \frac{1}{2}||d^t||^2 \tag{26}$$

$$\text{s.t. } d^t \in R \tag{27}$$

上述优化问题是闭且强凸优化问题，一定存在着最优解。我们令 $M_x(d^t) = \alpha$，则可以将一阶偏导项消去：

$$\begin{aligned} &\min \alpha + \frac{1}{2}||d^t||^2 \\ \text{s.t. } &\nabla f_k(x)^T d^t \leq \alpha, k = 1, ..., K \\ &d^t \in R \end{aligned} \tag{28}$$

上述问题为带线性不等式约束的凸二次规划问题

令 d^*, α^* 为上述优化问题的最优解，参考 [8] 和 [7]，则我们可以得到：

- 如果 x^t 是帕累托最优，则 $d^* = 0$ 且 $\alpha^* = 0$
- 如果 x^t 不是帕累托最优，则 $\alpha^* < 0$

且：

$$\begin{aligned} \alpha \leq -\frac{1}{2}||d^t|^2 < 0, \\ \nabla f_k(x)^T d^t \leq \alpha, k = 1, ..., K \end{aligned} \tag{29}$$

因此：

- 如果 $d^*(x) = 0$, 说明此时不存在下降方向，使得所有的目标都下降。
- 如果 $d^*(x) \neq 0$, 则有 $\nabla f_k(x)^T d^t < 0$，说明 d^t 是一个多目标的有效搜索方向，则按如下公式更新，即可以使目标函数下降：

$$\begin{aligned} x^{(t+1)} = x^t + \lambda \cdot d^t \\ f_k(x^{(t+1)}) \leq f_k(x^t), k = 1, ..., K \end{aligned} \tag{30}$$

5 多任务学习（MTL）

5.1 多任务学习定义

多任务学习（Multi-task learning）的目标是在同一时间学习多个任务，求得最优解。

多任务学习定义 [9]：

有 N 个输入样本点 $\{x_i, y_i^1, y_i^2, ..., y_i^T\}_{i\in N}$，其中 T 为任务数量，y_i^t 是第 t^{th} 个任务、第 i^{th} 个样本点标签 label。定义映射：

$$f^t(x;\theta^{sh},\theta^t) : X \to Y^t \tag{31}$$

其中 θ^{sh} 为多个任务共享参数，θ^t 为单个任务 t 独有的参数。

其损失函数定义：

$$L^t(.,.) : Y^t \times Y^t \to R^+ \tag{32}$$

$$\min_{\theta^{sh},\theta} \sum_{t=1}^{T} c^t \hat{L}^t(\theta^{sh},\theta^t) \tag{33}$$

其中 c^t 为每个具体任务的权重，每个具体任务 t 的损失函数定义：

$$\hat{L}^t(\theta^{sh},\theta^t) \triangleq \frac{1}{N}\sum_i L(f^t(x_i;\theta^{sh},\theta^t), y_i^t) \tag{34}$$

5.2 多任务学习转化为多目标优化

多任务学习可以转化为多目标优化问题求解 [9]。定义：

$$\min_{\theta^{sh},\theta} L(\theta^{sh},\theta^1,...,\theta^T) = \min_{\theta^{sh},\theta} (\hat{L}^1(\theta^{sh},\theta^1),...,\hat{L}^T(\theta^{sh},\theta^T)) \tag{35}$$

多目标优化的目的是获得帕累托最优解。

多任务优化的帕累托最优定义：

一个解 θ 支配另一个解 $\bar{\theta}$，如果 $\hat{L}^t(\theta^{sh},\theta^t) \leq \hat{L}^t(\theta^{sh},\theta^t)$，对于所有的任务 t 都成立, 且：

$$L(\theta^{sh},\theta^1,...,\theta^T) \neq L(\bar{\theta}^{sh},\bar{\theta}^1,...,\bar{\theta}^T) \tag{36}$$

一个解 θ^* 称作帕累托最优解，如果不存在解 θ 支配 θ^*

帕累托最优解的集合称为帕累托最优解集，其图像称为帕累托前沿（Pareto front ）

6 多任务求解: 单个帕累托解

参考论文：Sener, O. and Koltun, V. Multi-task learning as multi- objective optimization. In Advances in Neural Informa- tion Processing Systems, pp. 527–538, 2018.

6.1 问题转化

单个帕累托解，主要参考了论文 [9]。这里使用了多重梯度下降算法，基本原理参考本文第 4 章多目标梯度下降算法一节。由多目标优化的 KKT 条件，我们可以得到：

- 存在 $\alpha^1, ...\alpha^T \geq 0$ 使得：

$$\sum_{t=1}^{T} \alpha^t = 1,$$

$$\sum_{t=1}^{T} \alpha^t \nabla_{\theta^{sh}} \hat{L}^t(\theta^{sh},\theta^t) = 0 \tag{37}$$

- 对应所有的任务 t：

$$\nabla_{\theta^t}\hat{L}^t(\theta^{sh},\theta^t)=0 \tag{38}$$

满足式 (37) 、(38) 的解称为帕累托平稳点（Pareto stationary point）。帕累托最优点都是帕累托平稳点，反之不一定成立。考虑如下的优化问题：

$$\begin{gathered}\min_{\alpha^1,\ldots\alpha^T}||\sum_{t=1}^{T}\alpha^t\nabla_{\theta^{sh}}\hat{L}^t(\theta^{sh},\theta^t)|| \\ \sum_{t=1}^{T}\alpha^t=1,\alpha^t\geq 0\ ,\forall t\end{gathered} \tag{39}$$

上述优化问题的解存在两种情况：

- 最优值 $=0$ ，则对应的解满足 KKT 条件
- 最优值 $\neq 0$，则对应的解给出了下降方向，使得多任务目标函数提升（函数值下降）

上述优化问题等价于在输入点集凸包中找到最小模点。

6.2 考虑两个任务的情形

考虑两个任务的情形，则 (37) 式可以表示为：

$$\begin{gathered}\min_{\alpha^1,\ldots\alpha^T}||\gamma\theta+(1-\gamma)\bar{\theta}|| \\ \gamma+(1-\gamma)=1,\gamma\geq 0\end{gathered} \tag{40}$$

其中 $\theta,\bar{\theta}$ 定义为：

$$\begin{gathered}\theta\triangleq\nabla_{\theta^{sh}}\hat{L}^1(\theta^{sh},\theta^1) \\ \bar{\theta}\triangleq\nabla_{\theta^{sh}}\hat{L}^2(\theta^{sh},\theta^2)\end{gathered} \tag{41}$$

其解的情况枚举如下：

- 当 $\theta^T\bar{\theta}\geq\theta^T\theta,\gamma=1$
- 当 $\theta^T\bar{\theta}\geq\bar{\theta}^T\bar{\theta},\gamma=0$
- 其他情况

$$\gamma=\frac{(\bar{\theta}-\theta)^T\bar{\theta}}{||\bar{\theta}-\theta)||_2^2} \tag{42}$$

几何解释如下图所示：

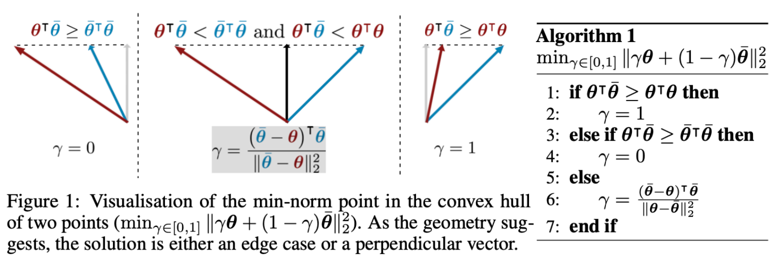

图 2: 两个任务的情形的几何解释

基于 Frank-wolfe 算法，可以得到求解 MTL 任务算法：

Algorithm 2 Update Equations for MTL

for $t = 1$ **to** T **do**
  $\boldsymbol{\theta}^t = \boldsymbol{\theta}^t - \eta \nabla_{\boldsymbol{\theta}^t} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on task-specific parameters
end for
$\alpha^1, \ldots, \alpha^T$ = FRANKWOLFESOLVER($\boldsymbol{\theta}$) ▷ Solve (3) to find a common descent direction
$\boldsymbol{\theta}^{sh} = \boldsymbol{\theta}^{sh} - \eta \sum_{t=1}^{T} \alpha^t \nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^t(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^t)$ ▷ Gradient descent on shared parameters

procedure FRANKWOLFESOLVER($\boldsymbol{\theta}$)
  Initialize $\boldsymbol{\alpha} = (\alpha^1, \ldots, \alpha^T) = (\frac{1}{T}, \ldots, \frac{1}{T})$
  Precompute $\mathbf{M}$ st. $\mathbf{M}_{i,j} = \left(\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^i(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^i)\right)^\intercal \left(\nabla_{\boldsymbol{\theta}^{sh}} \hat{\mathcal{L}}^j(\boldsymbol{\theta}^{sh}, \boldsymbol{\theta}^j)\right)$
  repeat
    $\hat{t} = \arg\min_r \sum_t \alpha^t \mathbf{M}_{rt}$
    $\hat{\gamma} = \arg\min_\gamma \left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)^\intercal \mathbf{M}\left((1-\gamma)\boldsymbol{\alpha} + \gamma \boldsymbol{e}_{\hat{t}}\right)$ ▷ Using Algorithm 1
    $\boldsymbol{\alpha} = (1-\hat{\gamma})\boldsymbol{\alpha} + \hat{\gamma}\boldsymbol{e}_{\hat{t}}$
  until $\hat{\gamma} \sim 0$ **or** Number of Iterations Limit
  return $\alpha^1, \ldots, \alpha^T$
end procedure

图 3: Frank-wolfe 算法求解 MTL

7 多任务求解：多个帕累托解

参考：Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. Pareto multi-task learning. In Advances in Neural Information Processing Systems, pp. 12037–12047, 2019.

上一节介绍的方法，只能求得一个帕累托解；在实际情况中，可能需要多个帕累托解才能做出更好的决策。好比你毕业时只拿到了阿里的 offer，这时你几乎没有多余的选择；但你如果拿到了除阿里之外，腾讯、头条、小米、百度的 offer，这时你的选择就多起来了，总 package 一样的情况，你会选择自己更看重的公司。

言归正传, 文章 [10] 介绍了一种求解多个帕累托解的方法。

7.1 主要思想

主要思想是将多任务学习分解多个带约束的多目标子问题，通过对子问题进行并行求解。

原始多任务学习定义：

$$\min_{\theta} L(\theta) = (L_1(\theta), L_2(\theta), ..., L_i(\theta), ..., L_m(\theta)) \tag{43}$$

$L_i(\theta)$ 是第 i 个任务的损失函数。

用一组分布良好的 Preference Vectors(简称 PV) ，将多任务学习的目标空间分解为 K 个子区域，如下图所示：

$$PV = \{u_1, u_k, ..., u_K\}, where\ u_k \in R_+^m \tag{44}$$

重新定义多任务学习：

$$\begin{aligned} &\min_{\theta} L(\theta) = (L_1(\theta), L_2(\theta), L_m(\theta)) \\ &\quad \text{s.t.}\ \ L(\theta) \in \Omega_k, k = 1, ..., K \end{aligned} \tag{45}$$

其中 Ω_k 是目标空间的子区域：

$$\Omega_k = \{v \in R_+^m | u_j^T v \le u_k^T v, \forall j = 1, ..., K\} \tag{46}$$

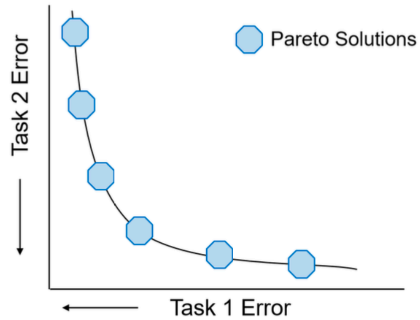

Figure 1: **Pareto MTL** can find a set of widely distributed Pareto solutions with different trade-offs for a given MTL. Then the practitioners can easily select their preferred solution(s).

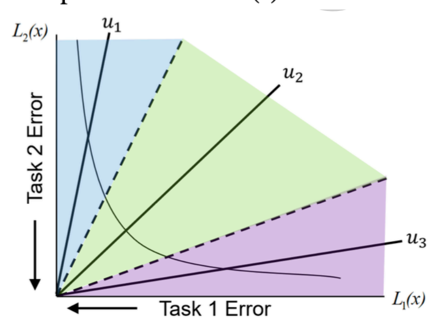

Figure 3: **Pareto MTL** decomposes a given MTL problem into several subproblems with a set of preference vectors. Each MTL subproblem aims at finding one Pareto solution in its restricted preference region.

图 4: 目标空间分解为子区域

上述 (46) 式说明，对应 Ω_k 中的元素 v:

$$v \in \Omega_k \text{当且仅当} v \text{和} u_k \textbf{有一个微小的夹角}$$
$$u_k^T v = ||u_k|| \cdot ||v|| cos\alpha \tag{47}$$

$u_k^T v$ 为最大的内积。

将问题 (45) 改写为:

$$\begin{aligned} &\min_\theta L(\theta) = (L_1(\theta), L_2(\theta), L_m(\theta)) \\ &\text{s.t. } G_j(\theta_t) = (u_j - u_k)^T L(\theta_t) \leq 0, j = 1, ..., K \end{aligned} \tag{48}$$

PV 向量将目标空间分解为不同的子区域，求解问题 (48) 得到的解集将会分布在不同的子区域。

7.2 子问题的梯度下降方法

7.2.1 寻找初始解 θ_r

求解问题 (45)，需要找到一个满足约束的基本可行解。对于随机产生的可行解 θ_r ，一种最直接的方法找到初始可行解 θ_0

$$\begin{aligned} &\min_{\theta_0} ||\theta_0 - \theta_r||^2 \\ &\text{s.t.} L(\theta_0) \in \Omega_k \end{aligned} \tag{49}$$

上述问题 (49) 投影方法求解的效率不高，特别是对于大规模的深度神经网络。这里将上述问题改写为无约束优化问题，使用序列梯度方法找到初始解 θ_0。

定义活跃限制集合（activated constraints):

$$I(\theta_r) = \{j | G_j(\theta_r) \geq 0, j = 1, ..., K\} \tag{50}$$

我们找到能令 $I(\theta_r)$ 中所有的活跃限制函数值下降的方向:

$$\begin{aligned} (d_r, \alpha_r) &= \text{argmin}_{\ d \in R^n, \alpha in R} \alpha + \frac{1}{2}||d||^2 \\ &\text{s.t.} \nabla G_j(\theta_r)^T d \leq \alpha, j \in I(\theta_r) \end{aligned} \tag{51}$$

求解上述问题后，可以得到对应的更新公式:

$$\theta_{r_{t+1}} = \theta_{r_t} + \eta_r d_{r_t} \tag{52}$$

上述问题 (51) 能将活跃集内的约束目标值减少，使得越来越多的约束目标小于 0 。$I(\theta_r)$ 最后变成空集，则 θ_r 是可行解。

7.2.2 求解子问题

定义 **受限帕累托最优**: θ^* 是多任务 $L(\theta)$ 在子区域 Ω_k 的最优解，如果 $\theta^* \in \Omega_k$ 且不存在 $\hat{\theta} \in \Omega_k$ 使得 $\hat{\theta} < \theta^*$。

考虑如下的多目标优化问题:

$$\begin{aligned} (d_t, \alpha_t) \quad &= \quad \text{argmin}_{\ d \in R^n, \alpha \in R} \alpha + \frac{1}{2}||d||^2 \\ \text{s.t.} \quad & \nabla L_i(\theta_t)^T d \leq \alpha, i = 1, ..., m \\ & \nabla G_j(\theta_t)^T d \leq \alpha, j \in I_\epsilon(\theta_t) \end{aligned} \tag{53}$$

其中 $I_\epsilon(\theta_t)$ 定义:

$$I_\epsilon(\theta_t) = \{j \in I | G_j(\theta) \geq -\epsilon\} \tag{54}$$

参考本文第 4 章这一节，我们有:

令 (d^k, α^k) 是多任务学习问题 (41) 的解，则

- 如果 θ_t 是严格受限于 Ω_k, 则 $d_t = 0 \in R^n$ 且 $\alpha_t = 0$
- 如果 θ_t 不是严格受限于 Ω_k，则：

$$\begin{aligned} \alpha_t &\leq -\frac{1}{2}||d_t||^2 < 0, \\ \nabla L_i(\theta_t)^T d &\leq \alpha, i = 1, ..., m \\ \nabla G_j(\theta_t)^T d &\leq \alpha, j \in I_\epsilon(\theta_t) \end{aligned} \tag{55}$$

迭代公式：

$$\theta_{t+1} = \theta_t + \eta d_t \tag{56}$$

通过求解上述问题，能够获得一个有效的搜索方向。

7.2.3 大规模求解方法

上述方法能够获得一个有效的搜索方向，对于大规模的问题，求解起来会比较困难。这里将问题 (53) 重写，将其表示为对偶形式：

- KKT 条件

$$d_t = -\sum_{i=1}^{m} \lambda_i \nabla L_i(\theta_t) + \sum_{j \in I_\epsilon(\theta)} \beta_i \nabla G_j(\theta_t)) \tag{57}$$

$$\sum_{i=1}^{m} \lambda_i + \sum_{j \in I_\epsilon(\theta)} \beta_j = 1 \tag{58}$$

- 对偶问题

$$\max_{\lambda_i, \beta_j} -\frac{1}{2}||\sum_{i=1}^{m} \lambda_i \nabla L_i(\theta_t) + \sum_{j \in I_\epsilon(\theta)} \beta_i \nabla G_j(\theta_t)||^2 \tag{59}$$

$$\text{s.t.} \sum_{i=1}^{m} \lambda_i + \sum_{j \in I_\epsilon(\theta)} \beta_j = 1, \lambda_i \geq 0, \beta_j \geq, \forall i = 1, ..., m, \forall j \in I_\epsilon(\theta) \tag{60}$$

将多任务学习转化为其对偶问题后，求解空间不再是参数空间，而是变成了任务个数和受限条件数，使得求解问题极大的减少了。

Pareto MTL 算法如下图：

8 多任务求解：连续帕累托解

参考文献：Ma, Pingchuan, Tao Du, and Wojciech Matusik. "Efficient Continuous Pareto Exploration in Multi-Task Learning." arXiv preprint arXiv:2006.16434 (2020).

8.1 主要思想

前面我们介绍了单个帕累托解和多个帕累托的求解方法，接下来我们介绍一种能够输出局部连续帕累托解，进一步构建帕累托前沿 (Pareto Front) 的方法 [11]。分如下的两步：

- 离散帕累托求解（本文 8.4 节）：给定初始点 x_0，在求出一个帕累托平稳点 x_0^* 后，从过点 x_0^* 的平滑曲线 $x(t)$ 切线出发，进行 K 次搜索计算搜索方向 v_i，扩展出平稳点 $\{x_i\}$；
- 连续帕累托解（前沿）构建（本文 8.5 节）：由初始点 x_0^* 及扩展出的平稳点集 $\{x_i\}$，进而构建出连续帕累托前沿。

Algorithm 1 Pareto MTL Algorithm

Input: A set of evenly distributed vectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_K\}$
Update Rule:
(can be solved in parallel)
for $k = 1$ to K **do**
  randomly generate parameters $\theta_r^{(k)}$
  find the initial parameters $\theta_0^{(k)}$ from $\theta_r^{(k)}$ using gradient-based method
  for $t = 1$ to T **do**
    obtain $\lambda_{ti}^{(k)} \geq 0, \beta_{ti}^{(k)} \geq 0, \forall i = 1, ..., m, \forall j \in I_\epsilon(\theta)$ by solving subproblem (14)
    calculate the direction $d_t^{(k)} = -(\sum_{i=1}^m \lambda_{ti}^{(k)} \nabla \mathcal{L}_i(\theta_t^{(k)}) + \sum_{j \in I_{\epsilon(\boldsymbol{x})}} \beta_{ti}^{(k)} \nabla \mathcal{G}_j(\theta_t^{(k)})$
    update the parameters $\theta_{t+1}^{(k)} = \theta_t^{(k)} + \eta d_t^{(k)}$
  end for
end for
Output: The set of solutions for all subproblems with different trade-offs $\{\theta_T^{(k)} | k = 1, \cdot, K\}$

图 5: Pareto MTL 算法

为表述方便，这里引用论文中关于多任务学习的定义：
设 $f(x)$ 光滑，

$$
\begin{aligned}
f(x) &: \mathcal{R}^n \to \mathcal{R}^m \\
f_i(x) &: \mathcal{R}^n \to \mathcal{R}, i = 1, ..., m
\end{aligned} \tag{61}
$$

8.2 预备知识：Krylov 子空间

这一节内容参考潘建瑜老师《线性方程组迭代方法》课程，第四讲《Krylov 子空间方法》[12]。

大规模稀疏线性方程组 $AX = b$ 求解的首选方法是 Krylov 子空间方法，其基本思想是在一个维数较小的子空间 $\mathcal{K} \subset R_n$ 中寻找近似解.

Krylov 子空间定义：设 $A \in R^{n\times n}$,$r \in R^n$, 我们称

$$\mathcal{K}_m(A, r) \triangleq span\{r, Ar, ..., A^{m-1}r\} \subseteq R_n \tag{62}$$

是由 A 和 r 生成的 Krylov 子空间，通常简记为 $\mathcal{K}_m$。Krylov 子空间有如下的 3 个性质：

- Krylov 子空间嵌套性：$\mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq ... \subseteq \mathcal{K}_m$；
- $\mathcal{K}_m$ 的维数不超过 m；
- $\mathcal{K}_m(A, r) = \{x = p(A)r$: 为次数小于 m 的多项式 $\}$。

简单来说，通过求解 Krylov 子空间的解来近似原始线性方程组的解。

8.3 基本概念

定义 1：帕累托平稳点 (Pareto Stationary)：设 $f_i(x)$ 连续可微，点 x 称为帕累托平稳点，如果存在 $\alpha \in R^m$ $\alpha_i \geq 0$ 使得下式成立：

$$
\begin{aligned}
\sum_{i=1}^m \alpha_i \nabla f_i(x) &= 0 \\
\sum_{i=1}^m \alpha_i &= 1
\end{aligned} \tag{63}
$$

引理 2： 帕累托点都是帕累托平稳点。

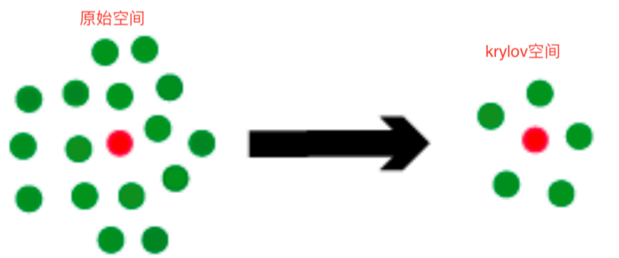

图 6: 原始空间张成 kryov 子空间

引理 3：设 $f(x)$ 是光滑且 x^* 是帕累托点，$x(t)$ 是过点 x^* 的曲线：

$$\begin{aligned} x(t): t \in (-\epsilon, \epsilon) \rightarrow R^n \\ x(0) = x^* \end{aligned} \tag{64}$$

则存在 $\beta \in R^m$ 使得：

$$\begin{aligned} H(x^*)x^{'}(t) = \nabla f(x^*)^T \beta \\ H(x^*) = \sum_{i=1}^{m} \alpha_i \nabla^2 f_i(x^*), x^{'}(t) 为切线 \end{aligned} \tag{65}$$

上式表明, 算子 $H(x^*)$ 将点 x^* 处的切向量 $v = x^{'}(t)$ 变换为由 $\nabla f_i(x^*)$ 扩张成的子空间 (Krylov 子空间) 的向量。图

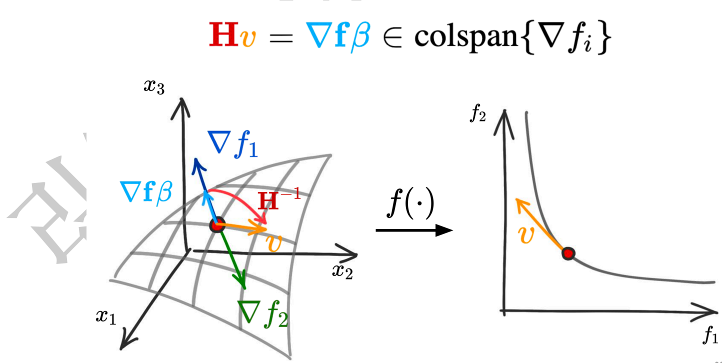

图 7: 扩张子空间

8.4 离散帕累托求解

给定初始点 $x_0 \in R^n$,$f_i(x)$ 光滑，可以从如下三步来获取连续帕累托解：

- 求解帕累托平稳点: 从初始点 x_0 出发，通过梯度下降的方法求解帕累托平稳点 x_0^*

- 扩展帕累托平稳点：$f(x)$ 在点 x_0^* 处光滑，如果帕累托前沿存在，则在点 x_0^* 处的某个邻域内存在着帕累托平稳点。由此出发，可以求得一系列的帕累托平稳点 x_i^*；
- 将上述的平稳点所在的局部帕累托前沿进行连接合并，扩充成更大的连续帕累托前沿。

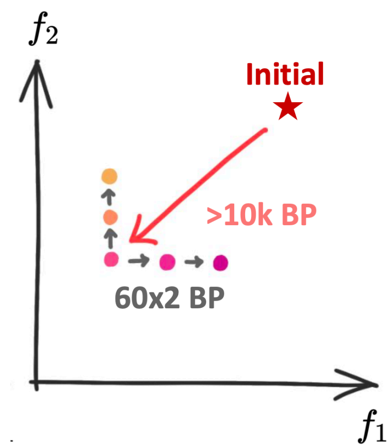

图 8: 帕累托前沿

我们先来看下如何获取一系列的帕累托平稳点。

8.4.1 梯度求解方法

这里可以通过前面介绍的梯度求解算法，参考本文第 4 部分。

8.4.2 一阶方法扩张

通过梯度求解方法求解出帕累托平稳点 x_0^* 后，可以基于该点扩展出局部帕累托集（目标函数光滑）$\{x_i\}$。这一过程可以分解为两步：

- 计算 α：计算 (63) 式中的 α;
- 求解搜索方向 v: 估计梯度迭代的搜索方向 v_i。

有了搜索方向，可以通过如下更新公式求解：

$$x_i = x_0^* + sv_i \tag{66}$$

我们先来看第一步。

1). 计算 α

计算 α 可以归结为求解如下的约束问题：

$$\begin{aligned} &\min_{\alpha} || \sum_{i=1}^{m} \alpha_i \nabla f_i(x_0^*) ||^2 \\ &\text{s.t.} \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1 \end{aligned} \tag{67}$$

上述问题规模为 m, 量级较小，可以很方便的求解出来。

2). 求解搜索方向 v

求解得到 α 后，由引理 3，我们可以给出待求解的线性方程组：

$$H(x_0^*)v = \nabla f(x_0^*)^T \beta \tag{68}$$

其中 v 为待求解的变量。上述问题求解有两个难点：

- x_0^* 不一定是帕累托平稳点
- 问题 (68) 的复杂度是 $O(n^3)$ ，当 n 非常大时，求解起来非常困难。

为此引入校正向量 c (correction vector), 将 (67) 改写为：

$$\min_{\alpha,c} ||c||_2$$

$$\text{s.t. } \alpha \geq 0, \sum_{i=1}^{m} \alpha_i = 1$$

$$\sum_{i=1}^{m} \alpha_i(\nabla f_i(x_0^*) - c) = 0 \tag{69}$$

(69) 式引入了校正向量 (correction vector) , 用 $\nabla f_i(x_0^*) - c$ 近似 $\nabla f_i(x_0^*)$，x_0^* 将会是帕累托平稳点。

引理 4：设 α^* 是问题 (67) 解，则问题 (69) 的解是：

$$(\alpha, c) = (\alpha^*, \nabla f(x_0^*)^T \alpha^*) \tag{70}$$

在计算出 α^*、帕累托平稳点 x_0^*、校正向量 c，可以计算出 $\nabla f(x_0^*)$。现在考虑如下的稀疏线性方程组：

$$H(x_0^*)v = (\nabla f(x_0^*)^T - c1^T)\beta \tag{71}$$

β 为随机生成的向量，v 为待求解的变量。式 (70) 可以通过 krylov 子空间，MINERS 方法进行求解。详细 MINERS 算法可以参考潘建瑜老师《线性方程组迭代方法》课程，第四讲《Krylov 子空间方法》[12]

我们来看下寻找离散帕累托解集合的求解算法，如下图所示：

随机初始化网络，输出 N 个帕累托平稳网络。

- 输入：随机初始化网络
- $ParetoExpand(x^*)$ 生成点 x^* 的 K 个搜索方向 v_i；由 K 个搜索方向扩展出 K 个子网络；
- 更新子网络节点：$x_i = x^* + sv_i$；
- $ParetoOptimize(x_i)$ 输出帕累托平稳点 x_i^*。
- 输出：N 个帕累托平稳网络

8.5 连续帕累托解（前沿）构建

通过前面的 Algorithm 1 求解出来 N 个帕累托平稳网络 (父节点及 K 个子网络)；接下来介绍如何由离散的帕累托点合并成更大的连续帕累托前沿 (Front)。

给定 x_i^* 及其对应的 K 个子节点 $\{x_{i_1}^*, ...x_{i_K}^*\}$, 定义连续变量 $r_{i\to i_j} \in [0,1]$ 以及搜索方向：

$$v_{i\to i_j} = x_{i_j}^* - x_i^*, j = 1, 2, ..., K \tag{72}$$

点 x_i^* 处的局部帕累托集可以通过下式进行构建：

$$S(x_i^*) = \{x_i^* + \sum_{i=1}^{K} r_{i\to i_j} v_{i\to i_j} | r_{i\to i_j} \geq 0, \sum_{i=1}^{K} r_{i\to i_j} \leq 1\} \tag{73}$$

$S(x_i^*)$ 是由点 x_i^* 及对应的 K 个子节点 $\{x_{i_1}^*, ...x_{i_K}^*\}$ 构成的凸包；切平面中切向量的线性组合仍然在切平面。

对于 N 个局部帕累托集：

$$\{S(x_1^*), ..., S(x_N^*)\} \tag{74}$$

可以将两两接壤处合并成一个更大的局部帕累托集合，全部合并完后，就可以生成多个的连续帕累托前沿 (Front)。

Algorithm 1 Efficient Pareto Set Exploration

Input: a random initial neural network $\boldsymbol{x}_0 \in \mathbb{R}^n$
Output: N Pareto stationary networks
$\boldsymbol{x}_0^* \leftarrow$`ParetoOptimize`$(\boldsymbol{x}_0)$
Initialize a queue $q \leftarrow [\boldsymbol{x}_0^*]$
Initialize an empty list to store the output: $output \leftarrow \emptyset$
repeat
  Pop a neural network $\boldsymbol{x}^*$ from q
  for $i = 1$ **to** K **do**
    $\boldsymbol{v}_i \leftarrow$`ParetoExpand`$(\boldsymbol{x}^*)$
    $\boldsymbol{v}_i /= \|\boldsymbol{v}_i\|_2$
    $\boldsymbol{x}_i \leftarrow \boldsymbol{x}^* + s\boldsymbol{v}_i$
    $\boldsymbol{x}_i^* \leftarrow$`ParetoOptimize`$(\boldsymbol{x}_i)$
    if No points in $output$ dominates $\boldsymbol{x}_i^*$ **then**
      Append $\boldsymbol{x}_i^*$ to q
      Append $(\boldsymbol{x}_i^*, \boldsymbol{f}(\boldsymbol{x}_i^*), \nabla \boldsymbol{f}(\boldsymbol{x}_i^*), \boldsymbol{x}^*)$ to $output$
    end if
  end for
until The size of $output$ reaches N

图 9: 帕累托平稳点扩展算法

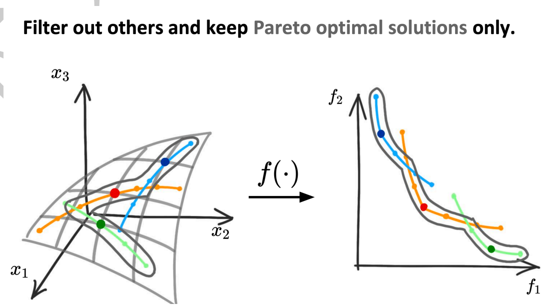

图 10: 连续帕累托前沿

参考文献

[1] 贺莉，刘庆怀著。《多目标优化理论与连续化方法》。2015-06。科学出版社

[2] 陈宝林著。《最优化理论与算法》（第 2 版）。2005-10。清华大学出版社

[3] KKT 条件，Karush–Kuhn–Tucker conditions,https://en.wikipedia.org/wiki/Karush

[4] 约束规格，constraint qualifications ,https://en.wikipedia.org/wiki/Karush

[5] S.D. Sudhoff. Lecture 9:Multi-Objective Optimization,https://engineering.purdue.edu/ sudhoff/ee630/Lecture09.pdf

[6] Fliege, J., Svaiter, B. Steepest descent methods for multicriteria optimization. Mathematical Methods of OR 51, 479–494 (2000). https://doi.org/10.1007/s001860000043

[7] Désidéri, Jean-Antoine. "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization." Comptes Rendus Mathematique 350.5-6 (2012): 313-318.

[8] Gebken, Bennet, Sebastian Peitz, and Michael Dellnitz. A descent method for equality and inequality constrained multiobjective optimization problems. Numerical and Evolutionary Optimization. Springer, Cham, 2017.

[9] Sener, O. and Koltun, V. Multi-task learning as multi- objective optimization. In Advances in Neural Informa- tion Processing Systems, pp. 527–538, 2018.

[10] Lin, X., Zhen, H.-L., Li, Z., Zhang, Q.-F., and Kwong, S. Pareto multi-task learning. In Advances in Neural Information Processing Systems, pp. 12037–12047, 2019.

[11] Ma, Pingchuan, Tao Du, and Wojciech Matusik. "Efficient Continuous Pareto Exploration in Multi-Task Learning." arXiv preprint arXiv:2006.16434 (2020).

[12] 潘建瑜《线性方程组迭代方法》课程，第四讲《Krylov 子空间方法》http://math.ecnu.edu.cn/ jypan/Teaching/MatrixIter/lect04_Krylov_ssm.pdf