arXiv:2302.01425v3 [cs.LG] 4 Jun 2023

快速、可微且稀疏的 Top-k：凸分析透视函数

Micha el E. Sander [*1] Joan Puigcerver [2] Josip Djolonga [2] Gabriel Peyr´e [1 3] Mathieu Blondel [2]

摘要

Top-k 算子返回一个稀疏向量，其中非零值对应于输入中 k 个最大的值。遗憾的是，由于它是不连续函数，因此很难将其纳入通过反向传播进行端到端训练的神经网络中。近期研究考虑了基于正则化或扰动技术的可微松弛。然而，迄今为止，还没有一种方法能够同时实现完全可微且稀疏。在本文中，我们提出了新的可微且 稀疏 Top-k 算子。我们将 Top-k 算子视为定义在排列多面体上的线性规划问题；排列多面体是排列的凸包。随后，我们引入一个 p-范数正则化项来平滑该算子，并证明其计算可以归约为保序优化。我们的框架比现有方法通用得多，例如可以表示按幅值选择值的 Top-k 算子。在算法方面，除 Pool 邻近违反者（PAV）算法外，我们还提出了一种适用于 GPU/TPU 的新 Dykstra 算法，用于求解保序优化问题。我们成功地将这些算子用于剪枝神经网络中的权重、微调视觉 Transformer，以及充当稀疏专家混合中的路由器。

1. 引言

寻找向量中的 top-k 值及其对应索引，是现代神经网络中广泛使用的基础模块。例如，在稀疏专家混合（MoE）（Shazeer 等人，2017年；Fedus 等人，2022年）中，top-k 路由器映射将每个词元分配给一组选定的 k 专家（或将每位专家分配给一组选定的 k 词元）。在序列解码的束搜索（Wiseman& Rush，2016年）中，会保留 k 个可能的输出序列组成的束，并在每个解码步骤进行更新。

对于神经网络剪枝，top-k 算子可用于稀疏化神经网络，即移除幅值最小的权重（Han 等人，2015年；Fra nkle 和 Carbin，2018年）。最后，top-k 准确率（例如，top-3 或 top-5）常用于评估神经网络在推理时的性能。

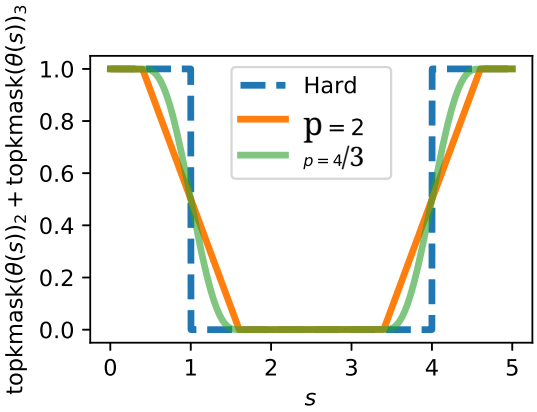


图*1*。**可微稀疏 top-k 掩码的示意图**。对于 $k = 2$，我们考虑 $\boldsymbol{\theta}(s) = (3,\ 1,\ -1 + s,\ s) \in \mathrm{R}^4$，并绘制 $\mathrm{topkmask}(\boldsymbol{\theta}(s))_2 + \mathrm{topkmask}(\boldsymbol{\theta}(s))_3$ 关于 s 的函数图像。我们将硬版本（无正则化）与我们提出的采用 p-范数正则化的算子进行比较：$p = 2$ 得到几乎处处可微的算子；$p = 4/3$则得到可微算子。两个算子都是**稀疏的**：对于 s 的某些取值，它们恰好为 0。

然而，top-k 算子是不连续分段仿射函数，其导数要么未定义，要么为常数（相关的 top-k 掩码 算子会返回与 top-k 值对应索引的二进制编码，其导数为零）。因此，很难在通过梯度反向传播训练的神经网络中使用它。近期研究探索了基于正则化或扰动技术的可微松弛方法（综述见 §2）。然而，迄今为止，还没有任何方法能够同时做到处处可微且稀疏。对于需要条件计算的神经网络而言，稀疏性至关重要。例如，在稀疏专家混合中，会使用 top-k 算子将词元“路由”给选定的专家。

[*]在 Google Research、Brain 团队实习期间完成。[1]高等师范学校 [2]Google Research、Brain 团队 [3]法国国家科学研究中心。联系方式：Michael E. Sander<michael.sander@ens.fr>、Joan Puigcerver <jpuigcerver@google.com>、Josip Djolonga< josipd@google.com>、Gabriel Peyré´e <gabriel.peyre @ens.fr>、Mathieu Blondel <mblondel@google.com>。

发表于 40^{th} 国际机器学习会议，美国夏威夷檀香山。PMLR 202，2023年。版权所有 © 2023 作者所有。

如果不具备稀疏性，所有专家都必须处理所有词元，导致计算成本高昂。神经网络中选择幅值最大的权重时也是如此：未选中的权重必须恰好为 0，才能优化计算和内存成本。

在本研究中，我们提出了处处可微且稀疏的 top-k 算子（见图1）。我们的工作基于Blondel 等人（2020b）提出的框架，该框架将排序和排名表示为置换多面体上的线性规划，并通过归约将其转化为保序优化。我们从多个方面显著拓展了这一框架。具体而言，我们作出了以下贡献：

• 在第2 节回顾相关工作，并在第3节介绍背景后，我们将在第4节介绍广义框架。我们引入了一种新的非线性函数 φ，借此可以表示新的算子（例如幅值最大的 top-k ）。在此过程中，我们还建立了所谓的k-支撑范数与置换多面体之间的新联系。

• 我们引入一个正则化项，以获得一个松弛的 top-k 算子。具体而言，采用 p-范数正则化后，我们得到了首个处处可微且稀疏的 top-k 算子（图3）。

• 在 §5中，我们推导出用于求解保序优化的相邻池违背算法（PAV），适用于使用 φ 和/或使用 p-范数正则化的情形。我们证明，我们的算子的雅可比矩阵可以用闭式形式计算。

• 作为适用于 GPU/TPU 的 PAV 替代方案，我们提出了一种 Dykstra 算法来求解保序优化；在情形 $p = 2$ 下，该算法易于向量化。

• 在 §6中，我们重点介绍了算子的三种应用。首先，我们在训练多层感知机时用它们来剪枝权重，并证明与硬 top-k相比，它们能带来更高的准确率。其次，我们定义 top-k 损失来微调视觉变换器（ViT），并获得比交叉熵损失更高的 top-k 准确率。最后，我们将算子用作视觉专家混合模型中的路由器，并证明其性能优于硬 top-k 路由器。

2. 相关工作

可微损失函数。多项研究提出了可微损失函数，用作离散且不连续的指标的替代函数。例如，已有研究为 top-k 准确率（Lapin 等人，2015年；2016年；Berrada 等人，2018年；Petersen 等人，2022年）以及各种排名指标提出了损失函数（Chapelle 和 Wu，2010年；Adams 和 Zemel，2011年；Rolínek 等人，2020年）。

可微算子。 在另一条研究路线中（也是本文的主要关注点），研究者提出了一种可微算子，可将其用作神经网络中的中间层，也可作为最终输出，并输入任意损失函数。例如，Niepert 等人（2021）提出了一个用于计算离散概率分布或优化问题梯度的框架。Amos 等人（2019）和 Qian 等人（2022）提出了一种基于二元熵正则化的平滑（但不稀疏）top-k 算子。其他研究者在不同背景下提出了有界单纯形上的相关投影（Martins & Kreutzer, 2017；Malaviya 等人，2018；Blondel, 2019）。这些工作采用临时设计的算法，而我们则将问题归约为保序优化。Cuturi 等人（2019）提出了一种基于熵正则化最优传输的排序和排名算子松弛方法，并据此构造了一个可微（但同样不稀疏）的 top-k 算子。其计算依赖于 Sinkhorn 算法（Sinkhorn, 1967；Cuturi, 2013），这也使得计算和求导可能较为缓慢。Xie 等人（2020）提出了类似的方法。Petersen 等人（2021）通过结合可微排序函数与排序网络，提出了平滑的可微排序网络。Grover 等人（2019）和 Prillo 和 Eisenschlos（2020）也提出了其他排序及算子松弛方法。

与我们最接近的工作是Blondel 等人(2020b，其中排序和排名被表述为置换多面体上的线性规划。为了使这些算子可微，研究者在公式中引入正则化，并证明由此得到的算子可通过保序优化在 $O(n\log n)$ 时间内计算。不幸的是，所提出的算子仍包含折点：它们并非处处可微。因此，如何构造一个处处可微且具有 并且稀疏松弛的算子，使其时间复杂度为 $O(n\log n)$，仍是一个悬而未决的问题。在本研究中，我们通过使用 p-范数正则化实现了这一目标。此外，通过引入一种新的非线性函数 φ，我们大幅推广了Blondel 等人(2020b）的框架，例如，使我们能够表达一种新的幅值 top-k 算子。我们提出了一种适用于 GPU/TPU 的新 Dykstra 算法，作为 PAV 的替代方案。

另一种技术并未引入正则化，而是依赖于扰动（Berthet 等人，2020年）。这项技术已被用于实现可微（但仍然不稀疏的）图像块选择的 top-k（Cordonnier 等人，2021年）。

剪除幅值较小的权重。近期许多研究都聚焦于神经网络剪枝，即移除参数以大幅缩减模型规模。近期综述见Blalock 等人(2020年)。一种简单但广受欢迎的神经网络剪枝方法是全局幅值剪枝（Collins 和 Kohli，2014年；Han 等人，2015年）：将绝对值最小的权重设为0。大多数剪枝技术都在模型完成训练后实施（Blalock 等人，2020年），但也有一些研究在训练过程中定期进行剪枝（Gale 等人，2019年）。然而，

据我们所知，按幅值剪枝并非以可微的方式进行。在这项工作中，我们通过实验证明，在训练期间使用幅值上的可微（或几乎处处可微）top-k 算子剪除权重，相较于使用“硬”算子，能够加快收敛并提高准确率。

专家混合的 Top-k 算子。稀疏专家混合模型（MoE）（Shazeer 等人，2017年）是一类深度学习模型，其特点是根据输入，仅激活模型中一小部分称为专家的模块。因此，稀疏 MoE 能够在不增加模型时间复杂度的情况下增加参数数量。稀疏 MoE 在计算机视觉（Riquelme 等人，2021年；Zhou 等人，2022年）和自然语言处理（Shazeer 等人，2017年；Lewis 等人，2021年；Fedus 等人，2021年）领域都取得了显著的实证成果。稀疏 MoE 模型的核心是其路由机制，该机制决定将哪些输入（或词元）分配给哪些专家。近期，稀疏专家混合领域的一些研究提出了新的路由模块 top-k 算子。Hazimeh 等人（2021年）提出了二进制编码公式来选择非零权重。然而，他们的公式并不能近似真正的 top-k 算子，而且仅支持在推理时实现稀疏性，不支持在训练过程中实现。Liu 等人（2022年）提出了一种支持 k稀疏性约束的最优传输公式，并将其用于稀疏专家混合。在这项工作中，我们提出用我们的平滑松弛来替代硬 top-k 路由器，后者是一个不连续函数。

3. 背景

记号。 我们将 $[n]$ 的一个排列记为 $\sigma = (\sigma_1, \dots, \sigma_n)$，其逆排列记为 σ^{-1}。将其视为向量时，记作 $\boldsymbol{\sigma}$。我们将所有 $n!$ 个排列组成的集合记为 Σ。给定一个向量 $\boldsymbol{x} \in \mathrm{R}^n$，我们将按照 σ 对 $\boldsymbol{x}$ 排列后的结果记为 $\boldsymbol{x}_\sigma := (x_{\sigma_1}, \dots, x_{\sigma_n})$。我们将 $\boldsymbol{x} \in \mathrm{R}^n$ 中第 i 大的值记为 $x_{[i]}$。不失一般性，我们始终按降序排列各值。 $f(\boldsymbol{x})$ 的共轭函数记为 $f^*(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f(\boldsymbol{x})$。

算子回顾。 我们将 **argsort** 算子记为对 $\boldsymbol{x} \in \mathrm{R}^n$ 进行排序的排列 $\boldsymbol{\sigma}$，即

$$\operatorname{argsort}(\boldsymbol{x}) := \boldsymbol{\sigma}, \quad \text{where} \quad x_{\sigma_1} \geq \cdots \geq x_{\sigma_n}.$$

我们将 **排序** 算子记为 $\boldsymbol{x} \in \mathrm{R}^n$ 中按序排列后的值，即

$$\operatorname{sort}(\boldsymbol{x}) := \boldsymbol{x}_\sigma, \quad \text{where} \quad \boldsymbol{\sigma} = \operatorname{argsort}(\boldsymbol{x}).$$

值 $[\operatorname{sort}(\boldsymbol{x})]_i$ 也称为第 i 个顺序统计量。我们将**秩**算子表示为返回向量 $\boldsymbol{x} \in \mathrm{R}^n$ 在排序向量中的位置的函数。它在形式上等于 argsort 的逆置换：

$$\operatorname{rank}(\boldsymbol{x}) := \boldsymbol{\sigma}^{-1}, \quad \text{where} \quad \boldsymbol{\sigma} = \operatorname{argsort}(\boldsymbol{x}).$$

秩 $[\operatorname{rank}(\boldsymbol{x})]_i$ 越小， x_i 的值就越大。**top-k 掩码** 算子会返回一个位向量，用于指示每个值 x_i 是否处于前 k 个值之中：

$$[\operatorname{topkmask}(\boldsymbol{x})]_i := \begin{cases} 1, & \text{if } [\operatorname{rank}(\boldsymbol{x})]_i \leq k \\ 0, & \text{otherwise.} \end{cases}.$$

Top-k 算子会返回处于前 k 个值之中的数值本身，否则返回 0，即

$$\operatorname{topk}(\boldsymbol{x}) := \boldsymbol{x} \circ \operatorname{topkmask}(\boldsymbol{x}),$$

其中 $\circ$ 表示逐元素乘法。**按幅值取 top-k**算子的定义类似：

$$\operatorname{topkmag}(\boldsymbol{x}) := \boldsymbol{x} \circ \operatorname{topkmask}(|\boldsymbol{x}|).$$

举例来说，如果 $x_3 \geq x_1 \geq x_2$ 且 $|x_2| \geq |x_3| \geq |x_1|$，那么

- $\operatorname{argsort}(\boldsymbol{x}) = (3, 1, 2)$
- $\operatorname{sort}(\boldsymbol{x}) = (x_3, x_1, x_2)$
- $\operatorname{rank}(\boldsymbol{x}) = (2, 3, 1)$
- $\operatorname{topkmask}(\boldsymbol{x}) = (1, 0, 1)$
- $\operatorname{topk}(\boldsymbol{x}) = (x_1, 0, x_3)$,
- $\operatorname{topkmag}(\boldsymbol{x}) = (0, x_2, x_3)$,

其中在最后三种情况下，我们使用了 $k = 2$。

置换多面体。 与向量 $\boldsymbol{w} \in \mathbb{R}^n$相关联的置换多面体是组合学中一个著名对象（Bow-man，1972年；Ziegler，2012年），它是 $\boldsymbol{w}$的所有置换所构成的凸包，即

$$P(\boldsymbol{w}) := \operatorname{conv}(\{\boldsymbol{w}_\sigma : \sigma \in \Sigma\}) \subset \mathbb{R}^n.$$

我们将与 $P(\boldsymbol{w})$ 相关联的线性最大化预言机（LMO）定义为

$$\begin{aligned} f(\boldsymbol{x}, \boldsymbol{w}) &:= \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle \\ \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) &:= \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \nabla_1 f(\boldsymbol{x}, \boldsymbol{w}), \end{aligned} \tag{1}$$

其中 $\nabla_1 f(\boldsymbol{x}, \boldsymbol{w})$ 在技术上是 f相对于$\boldsymbol{x}$ 的一个次梯度。线性最大化预言机可在 $O(n \log n)$ 时间内计算。事实上，线性最大化预言机的计算可归结为排序操作，如下述已知命题所示。为完整起见，附录 A.1 中给出了证明。

命题 1. *(线性最大化预言机)*

如果 $w_1 \geq \cdots \geq w_n$ （若不是，则对 $\boldsymbol{w}$ 排序），那么

$$f(\boldsymbol{x}, \boldsymbol{w}) = \sum_{i=1}^{n} w_i x_{[i]} \quad \textit{and} \quad \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{w}_{\mathrm{rank}(\boldsymbol{x})}.$$

线性规划表述。 我们用 $\boldsymbol{\rho} := (n, n-1, \ldots\ldots, 1)$ 表示反向排列。Blondel 等人(2020b) 证明，排序算子和秩算子可以表述为置换多面体上的线性规划（LP）：

$$\mathrm{sort}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{\rho}, \boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{x})} \langle \boldsymbol{\rho}, \boldsymbol{y} \rangle$$
$$\mathrm{rank}(\boldsymbol{x}) = \boldsymbol{y}(-\boldsymbol{x}, \boldsymbol{\rho}) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\boldsymbol{\rho})} \langle -\boldsymbol{x}, \boldsymbol{y} \rangle.$$

在后一表达式中，减号源于我们采用了秩越小表示数值越大的约定（即最大值的秩为 1）。

尽管 Blondel 等人（2020b）未提及，但将 top-k 掩码算子表示为线性规划也很容易

$$\mathrm{topkmask}(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x}, \mathbf{1}_k) = \operatorname*{argmax}_{\boldsymbol{y} \in P(\mathbf{1}_k)} \langle \boldsymbol{x}, \boldsymbol{y} \rangle, \qquad (2)$$

其中 $\mathbf{1}_k := (\underbrace{1, \ldots, 1}_{k}, \underbrace{0, \ldots, 0}_{n-k})$。对于 $\boldsymbol{w}$ 的这一选择，置换多面体具有特别简洁的表达式

$$P(\mathbf{1}_k) = \{\boldsymbol{y} \in \mathbb{R}^n : \langle \boldsymbol{y}, \mathbf{1} \rangle = k, \boldsymbol{y} \in [0,1]^n\}$$

且 $P(\mathbf{1}_k/k)$ 称为有界单纯形（Warmuth &Kuzmin, 2008；Blondel 等人，2020a）。图 2对此进行了说明。为获得松弛算子，Blondel 等人（2020b）提出在（1）中引入正则化（见下一节中的“恢复先前的框架”），并采用了保序优化的归约。

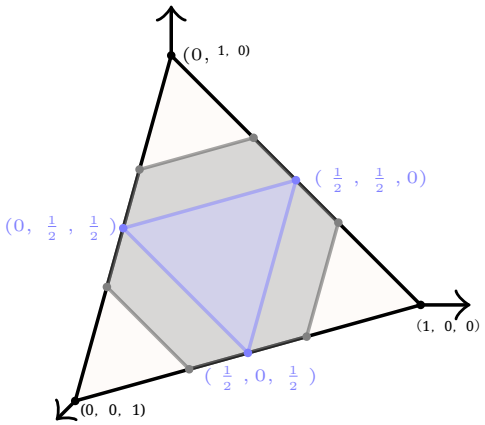


图 2。 置换多面体 $P(\boldsymbol{w})$ 是一个多面体，其顶点是 $\boldsymbol{w}$ 的排列。根据 $\boldsymbol{w}$ 的选择，它可以表示若干已知的多胞体。当 $\boldsymbol{w} = (1, 0, 0)$ 时，$P(\boldsymbol{w})$ 是概率单纯形（浅米色），对应于 top-1 设置。当 $\boldsymbol{w} = (\frac{1}{2}, \frac{1}{2}, 0)$ 时，$P(\boldsymbol{w})$ 是截断概率单纯形（蓝色），对应于 top-k 设置（此处，$k = 2$）。当 $\boldsymbol{w} = (\frac{2}{3}, \frac{1}{3}, 0)$ 时，$P(\boldsymbol{w})$ 是一个六边形，对应于部分排序设置（灰色）。

4. 提出的广义框架

在本节中，我们通过添加一个可选的非线性函数 $\varphi(\boldsymbol{x})$，对 Blondel 等人(2020b) 的框架进行推广。除了先前框架涵盖的算子之外，这还使我们能够直接表示幅值最大的 top-k 算子，这是此前无法实现的。我们还支持 p- 范数正则化，从而能够表示可微且稀疏的算子，条件是 $1 < p < 2$。

引入一个映射 φ。 考虑映射 $\varphi(\boldsymbol{x}) := (\phi(x_1) \ldots, \phi(x_n))$。给定 $\boldsymbol{x} \in \mathrm{R}^n$ 和 $\boldsymbol{w} \in \mathrm{R}^n$，我们定义

$$f_\varphi(\boldsymbol{x}, \boldsymbol{w}) := f(\varphi(\boldsymbol{x}), \boldsymbol{w})$$
$$\boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) := \nabla_1 f_\varphi(\boldsymbol{x}, \boldsymbol{w}).$$

当 $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ (恒等映射)时，我们显然可以还原现有框架，即 $f_\varphi(\boldsymbol{x}, \boldsymbol{w}) = f(\boldsymbol{x}, \boldsymbol{w})$ 和 $\boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) = \boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$。

当 $\varphi(\boldsymbol{x}) \neq \boldsymbol{x}$时，我们的框架开始与先前的框架有所不同，因为 φ 会影响微分。根据链式法则和 Danskin 定理（1966年），我们得到

$$\begin{aligned} \boldsymbol{y}_\varphi(\boldsymbol{x}, \boldsymbol{w}) &:= \nabla_1 f_\varphi(\boldsymbol{x}, \boldsymbol{w}) \\ &= \partial\varphi(\boldsymbol{x})^\top \boldsymbol{y}(\varphi(\boldsymbol{x}), \boldsymbol{w}) \\ &= (\phi'(x_1), \ldots, \phi'(x_n)) \circ \boldsymbol{y}(\varphi(\boldsymbol{x}), \boldsymbol{w}), \end{aligned} \qquad (3)$$

其中 $\partial\varphi(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$ 表示 $\varphi(\boldsymbol{x})$ 的雅可比矩阵，且$\boldsymbol{y}(\boldsymbol{x}, \boldsymbol{w})$ 由命题 1给出。

按幅值取 top-k。 正如我们强调的那样，我们提出的推广方案的一个优势是，可以表示按幅值取 top-k 算子。事实上，根据 $\phi(x) = \frac{1}{2}x^2$，由（2）和（3）可知，对所有 $\boldsymbol{x} \in \mathbb{R}^n$

$$\mathrm{topkmag}(\boldsymbol{x}) = \boldsymbol{y}_\varphi(\boldsymbol{x}, \mathbf{1}_k) = \nabla_1 f_\varphi(\boldsymbol{x}, \mathbf{1}_k). \qquad (4)$$

显然，对所有 $\boldsymbol{x} \in \mathbb{R}^n_+$，也有 $\mathrm{topkmag}(\boldsymbol{x}) = \mathrm{topk}(\boldsymbol{x})$。按幅值取 top-k 可用于剪除神经网络中幅值较小的权重，正如我们在 §6的实验中所展示的那样。

引入正则化。 现在我们解释如何使推广后的算子可微。我们在对偶空间中引入凸正则化 $R : \mathrm{R}^n \to \mathrm{R}$:

$$f^*_{\varphi,R}(\boldsymbol{y}, \boldsymbol{w}) := f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) + R(\boldsymbol{y}),$$

其中 f^*_φ 是 f_φ 关于第一个自变量的共轭函数。回到原始空间，我们得到一个新的松弛算子。证明见附录 A.2。

命题 2。 *(松弛算子)*
令 $R\colon\ \mathbb{R}^n \to \mathbb{R}$ 为一个凸正则项。则

$$f_{\varphi,R}(\boldsymbol{x},\boldsymbol{w}) \coloneqq \max_{\boldsymbol{y}\in\mathbb{R}^n} \langle \boldsymbol{y},\boldsymbol{x}\rangle - f^*_\varphi(\boldsymbol{y},\boldsymbol{w}) - R(\boldsymbol{y})$$
$$= \min_{\boldsymbol{u}\in\mathbb{R}^n} R^*(\boldsymbol{x}-\boldsymbol{u}) + f_\varphi(\boldsymbol{u},\boldsymbol{w})$$
$$\boldsymbol{y}_{\varphi,\ R}(\boldsymbol{x},\ \ \boldsymbol{w}) \coloneqq \boldsymbol{y}^\star = \nabla R^*(\boldsymbol{x}-\boldsymbol{u}^\star) = \nabla_1 f_{\varphi,\ R}(\boldsymbol{x},\ \ \boldsymbol{w})。$$

映射 φ 也会影响共轭性。我们有如下命题。

命题 3。 （f_φ 在第一个参数上的共轭函数）
若 ϕ 是凸的，且 $\boldsymbol{w}\in\mathbb{R}^n_+$，则

$$f^*_\varphi(\boldsymbol{y},\boldsymbol{w}) = \min_{\boldsymbol{z}\in P(\boldsymbol{w})} D_{\phi^*}(\boldsymbol{y},\boldsymbol{z}),$$

where $D_f(\boldsymbol{y},\boldsymbol{z}) \coloneqq \sum_{i=1}^n z_i f(y_i/z_i)$.

证明见附录 A.3。函数 $(y_i, z_i) \mapsto y_i'\phi^*(y_i/z_i)$ 称为 ϕ^* 的透视函数，并且当 $z_i > 0$ 时是联合凸的。函数 $D_f(\boldsymbol{y},\boldsymbol{z})$ 称为 $\boldsymbol{y}$ 与 $\boldsymbol{z}$ 之间的 f-散度。因此，$f^*_\varphi(\boldsymbol{y},\boldsymbol{w})$ 可以看作是在 ϕ^*-散度意义下，$\boldsymbol{y}$ 与 $P(\boldsymbol{w})$ 之间的最小“距离”。

恢复先前的框架。 若 $\phi(x) = x$，则

$$\phi^*(y_i/z_i) = \begin{cases} 0, & \text{if } y_i = z_i \\ \infty, & \text{otherwise} \end{cases}.$$

这意味着 $f^*_\varphi(\boldsymbol{y},\ \boldsymbol{w}) = f^*(\boldsymbol{y},\ \boldsymbol{w}) = \delta_{P(\boldsymbol{w})}(\boldsymbol{y})$，即 $P(\boldsymbol{w})$ 的示性函数；当 $\boldsymbol{y}\in P(\boldsymbol{w})$ 时其值为 0，否则为 ∞。在这种情况下，我们因此得到

$$f_{\varphi,R}(\boldsymbol{x},\boldsymbol{w}) = \max_{\boldsymbol{y}\in P(\boldsymbol{w})} \langle \boldsymbol{y},\boldsymbol{x}\rangle - R(\boldsymbol{y}),$$

这恰好是Blondel 等人(2020b) 的松弛形式。

可微且稀疏的 top-k 算子。 要通过我们的框架获得松弛的 top-k 算子，只需将 f_φ 替换为$f_{\varphi,\ R}$ 并将 $\boldsymbol{y}_\varphi$ 替换为$\boldsymbol{y}_{\varphi,\ R}$ 在（4）中定义

$$\text{topkmag}_R(\boldsymbol{x}) \coloneqq \boldsymbol{y}_{\varphi,R}(\boldsymbol{x},\mathbf{1}_k) = \nabla_1 f_{\varphi,R}(\boldsymbol{x},\mathbf{1}_k).$$

松弛的 top-k 掩码也可以用类似方式定义，但使用 $\phi(x) = x$ 而非 $\phi(x) = \frac{1}{2}x^2$。对于正则化R,- 我们建议使用 p-范数的 p 次幂:

$$R(\boldsymbol{y}) = \frac{1}{p}\|\boldsymbol{y}\|_p^p \coloneqq \frac{1}{p}\sum_{i=1}^n |y_i|^p.$$

先前研究中使用的选择 $p = 2$ 会产生稀疏输出，但并非处处可微。介于两者之间的任何 p 1 （不包括）和2 （不包括）可实现可微的且稀疏输出。我们提出使用 $p = 4/3$ 来获得处处可微的算子，并使用 $p = 2$ 来获得几乎处处可微的算子，这在数值计算上更方便。图3对此进行了说明。

与 k-支撑范数和 OWL 范数的联系。 当$\phi(x) = \frac{1}{2}x^2$ 和 $\boldsymbol{w} = \mathbf{1}_k$时，我们得到

$$f^*_\varphi(\boldsymbol{y},\boldsymbol{w}) = \frac{1}{2}\min_{\boldsymbol{z}\in[0,1]^n} \sum_{i=1}^n \frac{y_i^2}{z_i} \quad \text{s.t.} \quad \langle \boldsymbol{z},\mathbf{1}\rangle = k,$$

这称为平方 k-支撑范数（Argyriou 等人，2012年; McDonald 等人，2014年; Eriksson 等人，2015年）。我们的公式是平方 k-支撑范数的推广，因为它支持其他 $\boldsymbol{w}$ 和 ϕ 的选择。例如，我们用它在附录 k中定义一种新的B.1-支撑负熵概念。当 $\varphi(\boldsymbol{x}) = |\boldsymbol{x}|$时，我们得到有序加权 Lasso（OWL）范数（Zeng 和 Figueiredo，2014年），其形式为

$$f_\varphi(\boldsymbol{x},\boldsymbol{w}) = \sum_{i=1}^n w_i |x|_{[i]}.$$

取 φ后，从（3）可以很容易看出，（4）就变成了一个带符号的top-k 掩码。有趣的是，值得注意的是，k- 支撑集与 OWL 范数并非定义在同一个空间中。

双共轭解释。定义 k-稀疏向量集合如下，该集合是非凸的: $S_k \coloneqq \{\boldsymbol{x}\ \in\mathrm{R}^n\colon\ \|\boldsymbol{x}\|_0 \le k\}$，其中 $\|\boldsymbol{x}\|_0$ 表示 $\boldsymbol{x}$中的非零元素个数。我们看到，如果 $\phi(x) = \frac{1}{2}x^2$，那么 $f^*_\varphi(\boldsymbol{y},\mathbf{1}_k)$就是平方 k-支撑范数。已知它是将平方L_2 范数限制在 S_k 上所得函数的双共轭（即最紧的凸松弛）（Eriksson 等人，2015 年；Liu 等人，2022年）。现在我们证明一个更一般的结果: $f^*_\varphi(\boldsymbol{y},\mathbf{1}_k)$是将$\sum_{i=1}^n \phi^*(y_i)$限制在 S_k上所得函数的双共轭。

命题 4. （双共轭解释）
设 $\Phi_k(\boldsymbol{y}) \coloneqq \sum_{i=1}^n \phi^*(y_i) + \delta_{S_k}(\boldsymbol{y})$。假设 ϕ 是凸函数。那么 Φ_k 的双共轭函数为

$$\Phi_k^{**}(\boldsymbol{y}) = f^*_\varphi(\boldsymbol{y},\mathbf{1}_k)$$

证明见附录 A.4 。

5. 算法

本节提出用于计算我们所定义算子的高效算法。我们首先证明，我们的松弛算子的计算可归结为保序优化。

化为保序优化。 现在我们将说明如何通过化为保序优化来计算 $\boldsymbol{u}^\star$ 命题2 中的结果，由此即可恢复 $\boldsymbol{y}^\star$ $\boldsymbol{y}^\star =$

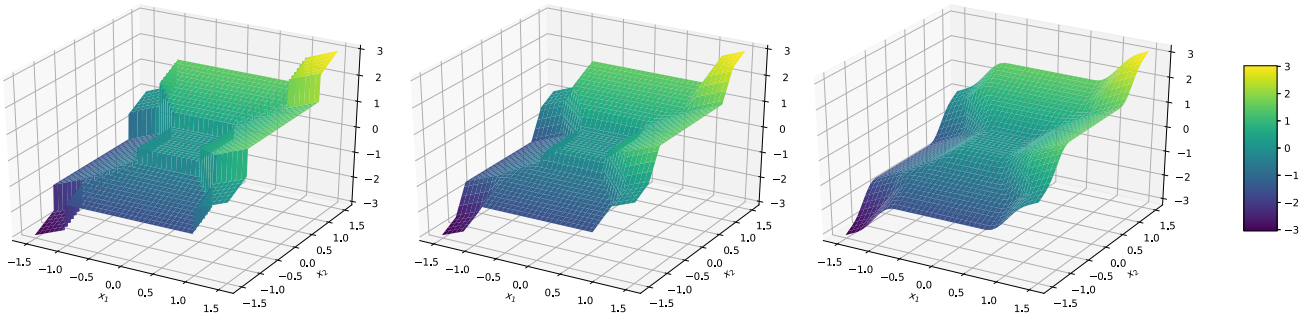


图 3. **我们提出的松弛 top-k 算子示例**。我们取 $\phi(x) = \frac{1}{2}x^2$ 和 $k = 2$。对于输入 $\boldsymbol{x} = (x_1, x_2, 1\,2, \mathbf{1})$，我们绘制了 $y_{\varphi,R}(\boldsymbol{x}, 1_k)$ $_1 + y_{\varphi,R}(\boldsymbol{x}, 1_k)_2$，分别对应 $R = 0$（左）、$R = \frac{\lambda}{2}\|\boldsymbol{x}\|_2^2$（中）和带有 $p = \frac{4}{3}$ 的 $R = \frac{\lambda}{p}\|\boldsymbol{x}\|_p^p$（右）。我们取 $\lambda = 0.3$。不加正则化会导致映射不连续；二范数正则化则带来连续性和几乎处处可微性，而$_{4\,3}$范数正则化则提供了连续可微的映射。我们强调，尽管左图中的曲线看起来是连通的，但它实际上是一个不连续函数。请注意，我们的松弛算子具有**稀疏性**，因为它们在中心处恰好为 0。

$\nabla R^*(\boldsymbol{x} - \boldsymbol{u}^\star)$。我们先回顾情形 $\varphi(\boldsymbol{x}) = \boldsymbol{x}$，这一结果已在现有研究中得到证明（Lim & Wright，2016年；Blondel 等人，2020b）。

命题 5.（归约，$\varphi(\boldsymbol{x}) = \boldsymbol{x}$ 情形）
假设 $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$。令 σ 为排列排序 $\boldsymbol{x}$, $\boldsymbol{s} := \boldsymbol{x}_\sigma$ 和

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n} R^*(\boldsymbol{s} - \boldsymbol{v}) + f(\boldsymbol{v}, \boldsymbol{w}).$$

那么 $\boldsymbol{u}^\star$ 由命题 2 可得 $\boldsymbol{u}^\star = \boldsymbol{v}^\star_{\sigma^{-1}}$。

集合 $\{\boldsymbol{v} \in \mathrm{R}^n\colon\ v_1 \geq \cdots \geq v_n\}$ 称为单调锥。接下来，我们证明当$\phi(\boldsymbol{x})$和 r^* 都是偶函数（符号不变）且在 R+上递增时，也能得到类似的结果。

命题 6.（归约，$\varphi(\boldsymbol{x}) = \varphi(-\boldsymbol{x})$ 情形）
假设 $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$ 且 $\varphi(\boldsymbol{x}) = (\phi(x_1), \ldots, \phi(x_n))$。假设 ϕ 和 r^* 都是偶函数（符号不变）且在 $\mathbb{R}$+上递增。令σ 为排列排序 $|\boldsymbol{x}|$, $\boldsymbol{s} := |\boldsymbol{x}|_\sigma$ 和

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n \geq 0} R^*(\boldsymbol{s} - \boldsymbol{v}) + f_\varphi(\boldsymbol{v}, \boldsymbol{w}).$$

那么，$\boldsymbol{u}^\star$（命题2）等于$\operatorname{sign}(\boldsymbol{x}) \circ \boldsymbol{v}^\star_{\sigma^{-1}}$。

证明见附录A.6 。对于Zeng 和 Figueiredo、2014年；Eriksson 等人，2015年针对 ϕ 和 R的特定情形，证明了更不一般的结果。集合 $\{v \in \mathbb{R}^n\colon\ v_1 \geq \cdots \geq v_n \geq 0\}$ 称为非负单调锥。实践中，额外的非负约束很容易处理：我们可以先求解不带此约束的保序优化问题，然后在解为非负时将其截断（N´emeth& N´emeth，2012年）。

相邻池违背算法（PAV）。在命题6的条件下，假设 $\boldsymbol{v}$ 和 $\boldsymbol{w}$ 均已排序，根据命题1 可得

$$f(\boldsymbol{v}, \boldsymbol{w}) = \sum_{i=1}^n w_i v_i, \quad f_\varphi(\boldsymbol{v}, \boldsymbol{w}) = \sum_{i=1}^n w_i \phi(v_i).$$

由此可知，命题5 和6 中的问题在坐标维度上可分离：

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{v_1 \geq \cdots \geq v_n} \sum_{i=1}^n h_i(v_i), \tag{5}$$

对于 $h_i(v_i) = r^*(s_i - v_i) + w_i\phi(v_i)$。此类问题可通过相邻池违背算法（PAV）在 $O(n)$ 时间内求解（Best 等人，2000）。该算法将集合 $[n]$ 划分为互不相交的集合 $(B_1, \cdots B_m)$，从 $m = n$ 和 $B_i = \{i\}$开始，不断合并这些集合，直至满足保序条件。为完整起见，附录B.2提供了伪代码。从本质上说，PAV 只需要一个用于求解“池化”子问题的例程

$$\gamma^\star_B = \operatorname*{argmin}_{\gamma \in \mathbb{R}} \sum_{i \in B} h_i(\gamma) \tag{6}$$

对于任意 $B \subseteq [n]$。一旦确定最优划分 $(B_1, \cdots B_m)$，我们便有

$$\boldsymbol{v}^\star = (\underbrace{\gamma^\star_{B_1}, \ldots, \gamma^\star_{B_1}}_{|B_1|}, \cdots, \underbrace{\gamma^\star_{B_m}, \ldots, \gamma^\star_{B_m}}_{|B_m|}) \in \mathbb{R}^n.$$

由于命题5和6要求预先获得排序置换 σ，我们这些算子的总时间复杂度为 $O(n$对数 $n)$。

例。假设 $\phi(x) = \frac{1}{2}x^2$ 且 $R = \frac{\lambda}{2}\|.\|_2^2$，其中 $\lambda > 0$。由于 $R^* = \frac{1}{2\lambda}\|.\|^2$，因此得到 $h_i(v_i) = \frac{1}{2}(w_i v_i^2 +$

$\frac{1}{\lambda}(s_i - v_i)^2$)。那么，子问题的解为

$$\gamma_B^\star = \frac{\sum_{i\in B} s_i}{\sum_{i\in B}(\lambda w_i + 1)}.$$

利用这个公式，当 λ 足够小时，我们可以给出硬算子与松弛算子之间误差的上界：$\|\text{topkmag}_R(\boldsymbol{x}) - \text{topkmag}(\boldsymbol{x})\|_\infty \leq \lambda\|\boldsymbol{x}\|_\infty$。详见附录 A.6，其中也讨论了情形 $p = \frac{4}{3}$。

Dykstra 交替投影算法。 PAV 算法可在 5 式的 $O(n)$ 时间内返回精确解。遗憾的是，该算法依赖逐元素的动态数组赋值，因此在 GPU 和 TPU 上可能较慢。我们提出一种替代方法来加快计算。我们的关键洞见是，通过定义

$$C_1 := \{\boldsymbol{v} \in \mathbb{R}^n : v_1 \geq v_2, v_3 \geq v_4, \dots\}$$
$$C_2 := \{\boldsymbol{v} \in \mathbb{R}^n : v_2 \geq v_3, v_4 \geq v_5, \dots\},$$

可得 $\{\boldsymbol{v} \in \mathbb{R}^n : \; v_1 \geq \cdots \geq v_n\} = C_1 \cap C_2$. 因此，我们可以将式（5）改写为

$$\boldsymbol{v}^\star = \operatorname*{argmin}_{\boldsymbol{v}\in C_1\cap C_2} \sum_{i=1}^n h_i(v_i).$$

在情形 $R = \frac{1}{2}\|\cdot\|^2$ 且 $\phi(x) = x$ 或 $\frac{1}{2}x^2$ 时，这可简化为向 $C_1 \cap C_2$ 上的投影，我们可以使用 Dykstra 著名的投影算法（Boyle 与 Dykstra，1986年；Combettes 与 Pesquet，2011年）。Jegelka 等人 (2013年) 使用该方法计算到次模多面体交集的投影，而我们投影到单个多面体面上。当 $\phi(x) = x$ 时，Dykstra 投影算法可简化为算法1中的形式。

算法 1. *(Dykstra 投影算法)*

从 $\boldsymbol{v}^0 = \boldsymbol{s}$, $\boldsymbol{p}^0 = \boldsymbol{q}^0 = \boldsymbol{0}$开始，*Dykstra* 算法迭代

$$\boldsymbol{y}^k = \operatorname*{argmin}_{\boldsymbol{y}\in C_1} \frac{1}{2}\|\boldsymbol{v}^k + \boldsymbol{p}^k - \boldsymbol{y}\|_2^2 + \langle \boldsymbol{y}, \boldsymbol{w}\rangle$$
$$\boldsymbol{p}^{k+1} = \boldsymbol{v}^k + \boldsymbol{p}^k - \boldsymbol{y}^k$$
$$\boldsymbol{v}^{k+1} = \operatorname*{argmin}_{\boldsymbol{v}\in C_2} \frac{1}{2}\|\boldsymbol{y}^k + \boldsymbol{q}^k - \boldsymbol{v}\|_2^2 + \langle \boldsymbol{v}, \boldsymbol{w}\rangle$$
$$\boldsymbol{q}^{k+1} = \boldsymbol{y}^k + \boldsymbol{q}^k - \boldsymbol{v}^{k+1}.$$

每次 argmin 计算都对应于向 C_1 或 C_2 的欧几里得投影，可以通过闭式形式计算。因此，$\boldsymbol{v}^k$ 可证明会收敛到 $\boldsymbol{v}^\star$。Dykstra 算法的每次迭代都无需学习率，时间复杂度为线性，并且可以利用掩码高效地写成矩阵向量乘积，因此特别适用于图形处理器和张量处理单元。有趣的是，我们发现，当$\boldsymbol{w} = \boldsymbol{1}_k$时，Dykstra 算法的收敛速度出奇地快，并能达到精确解。我们在图中验证了 Dykstra 算法的运行时间比 PAV 更短 4。我们使用100 次 Dykstra 迭代，并验证得到的结果与 PAV 相同。

对于一般的非欧几里得 R^* 情形，即 $p \neq 2$，我们可以在（5）的对偶问题中使用分块坐标上升。我们可以将对偶变量分为两个块，分别对应于C_1 和 C_2；参见附录A.8。事实上，已知在欧几里得情形下，原问题中的 Dykstra 算法与对偶问题中的分块坐标上升等价（Tibshirani，2017年）。因此，尽管对 $p \neq 2$进行向量化实现可能颇具挑战性，但分块坐标上升可被视为推广 Dykstra 算法的一种优雅方式。只要每个 h_i 都是严格凸的，就能保证收敛。

微分。 </style}等渗优化问题解的雅可比矩阵，对于任意 p-范数正则化，都可以用闭式表达。

命题 7。 *(微分)*

设 $\boldsymbol{v}^\star = (\gamma^\star_{B_1}, \dots, \gamma^\star_{B_1} \cdots, \gamma^\star_{B_m}, \dots, \gamma^\star_{B_m})$ 是具有 $R = \frac{1}{p}\|.\|_p^p$ 和 $p > 0$ 的保序优化问题的最优解。则 $\boldsymbol{v}^\star$ 关于 $\boldsymbol{s} = (s_1, \dots, s_n)$ 可微。此外，对于任意 $r \in \{1, \dots, m\}$ 和 $i \in B_r$,

$$\frac{\partial \gamma^\star_{B_r}}{\partial s_i} = \begin{cases} \frac{|\gamma^\star_{B_r} - s_i|^{q-2}}{\sum_{j\in B_r}|\gamma^\star_{B_r} - s_j|^{q-2}} & \text{if } \phi(x) = x \\ \frac{(q-1)|\gamma^\star_{B_r} - s_i|^{q-2}}{\sum_{j\in B_r}(q-1)|\gamma^\star_{B_r} - s_j|^{q-2} + w_j} & \text{if } \phi(x) = \frac{1}{2}x^2 \end{cases}$$

其中，q 满足 $\frac{1}{p} + \frac{1}{q} = 1$。当 $i \notin B_r$ 时，有 $\frac{\partial \gamma^\star_{B_r}}{\partial s_i} = 0$.

由此可得 $\partial v_j^\star / \partial s_i = \partial \gamma^\star_{B_{r_j}} / \partial s_i$ 其中 r_j 满足$v_j^\star = \gamma^\star_{B_{r_j}}$。证明见附录A.7 。得益于命题 7，计算解 $\boldsymbol{v}^\star$的雅可比矩阵时，我们无需求解线性系统；这与一般优化问题的隐式微分不同（Blondel 等人，2021年）。在实践中，这也意味着，为了求得包含我们算子的标量损失函数的梯度，我们无需通过 PAV 或 Dykstra 投影算法展开迭代的过程进行反向传播。尤其是，我们无需将这些算法的中间迭代结果存储在内存中。除了实现 PAV 和 Dykstra 投影算法外，我们还在 JAX 中使用命题7实现了相应的雅可比向量积例程。

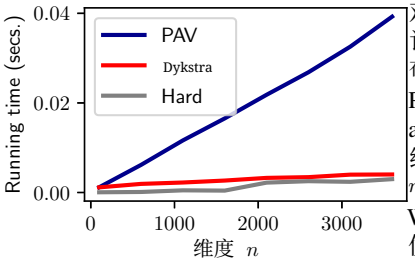


图 4。**运行时间对比**所用计算我们松弛后的在 TPU 上使用的 top-k PAV, Dykstra, as a 函数 的维度 n。对于每个 n，我们设定 $k = \lceil n/10 \rceil$。We also 比较使用硬 Top-k 计算。

6. 实验

现在，我们通过实验展示 top-k 算子的适用性。我们的 JAX（Bradbury 等人，2018年）实现可通过以下网址获取。更多实验细节见附录C 。

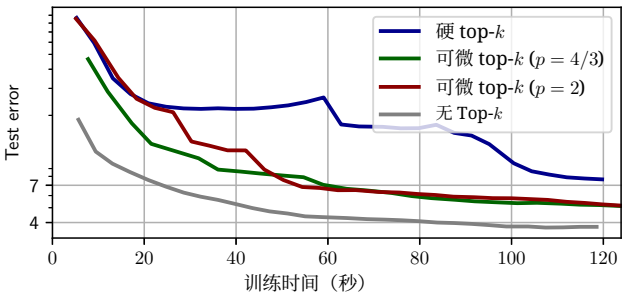


图 5. **测试误差** 在 MNIST 上训练多层感知机时，测试误差随训练时间的变化。我们比较基线（灰色）与将 90% 的权重通过幅值剪枝设为 0 的情形：使用几乎处处可微（红色）、完全可微（绿色）或硬 Top-k （蓝色）。

神经网络中的权重剪枝。 我们通过实验验证了使用平滑 top-k 进行神经网络权重剪枝的优势。我们使用一个具有 2 个隐藏层并采用 ReLU 激活函数的多层感知机（ MLP）。各层宽度依次为 784、32、32，后接一个宽度为 10 的线性分类头。更具体地说，我们的模型以图像 $\boldsymbol{a} \in \mathrm{R}^{784}$ 作为输入，并输出

$$\boldsymbol{x} = W_3\sigma(W_2\sigma(W_1\boldsymbol{a} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3 \quad \text{(logits)},$$

其中 $W_1 \in \mathrm{R}^{32\times 784}$, $\boldsymbol{b}_1 \in \mathrm{R}^{32}$, $W_2 \in \mathrm{R}^{32\times 32}$, $\boldsymbol{b}_2 \in \mathrm{R}^{32}$, $W_1 \in \mathrm{R}^{10\times 32}$, $\boldsymbol{b}_3 \in \mathrm{R}^{10}$ ，且 σ 是一个 ReLU。为了进行权重剪枝，我们将每个 W_i 参数化为$W_i = \text{topkmag}_R(W_i')$，并学习 W_i'而非直接学习W_i 。随后将输出输入交叉熵损失函数。我们比较了使用硬 top-k 算子与可微 top-k 算子时模型的性能，两者都只保留 10% 的系数。对于可微 top-k，我们使用正则化项 $R(\boldsymbol{y}) = \frac{\lambda}{p}\|\boldsymbol{y}\|^p$，其中 $p \in \{\frac{4}{3}, 2\}$ 且$\lambda = 10^{-4}$. 。我们发现，使用可微 top-k 训练的模型比使用硬 top-k 训练的模型收敛速度显著更快。我们还验证了我们的松弛 top-k 能够维持10% 的非零权重比例。MNIST 上的结果展示在图5中。我们还将其与使用 Cuturi 等人提出并由Cuturi 等人 (2019年) 的框架、并经Petersen 等人 (2022年) 改进的 top-k 算子的熵正则化近似进行了比较。为确保权重的稀疏性，我们使用“直通”技巧：前向传播时运行硬 top-k，而反向传播时使用松弛 top-k 的梯度。该方法的测试误差为5。9%，与使用我们的可微算子得到的结果相当。

平滑 Top-k 损失。 训练神经网络时，通常会最小化交叉熵损失，而网络的性能则通过 top-k 测试准确率来评估。因此，训练时使用的损失与评估时使用的指标并不匹配。Cuturi 等人(2019年) 提议用可微的 top-k 损失取代交叉熵损失。本着同样的思路，我们提议使用平滑且稀疏的 top-k 损失，而非交叉熵损失，对在 ImageNet21k 数据集上预训练的 ViT-B/16（Dosovit- skiy 等人，2020年）在 CIFAR 100（Krizhevsky 等人，2009年）上进行微调。我们使用 Fenchel-Young 损失（Blondel 等人，2020a）。其输入为向量 $\boldsymbol{a}$ 以及神经网络 θ 的参数 g_θ:

$$\boldsymbol{x} = g_\theta(\boldsymbol{a}) \quad \text{(logits)}$$
$$\ell(\boldsymbol{x}, \boldsymbol{t}) = f_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k) - \langle \boldsymbol{x}, \boldsymbol{t} \rangle,$$

其中，$f_{\varphi,R}(\boldsymbol{x}, \mathbf{1}_k)$由命题 2 给出， $\boldsymbol{t}$ 是 $\boldsymbol{a}$所属类别的独热编码。我们设置 $\phi(x) = x$ ，因为我们需要一个 top-k 掩码。对于R，我们考虑 p 范数正则化，其中 $p = 2$ 或 $p = 4/3$。我们取 $k = 3$。我们采用与 Dosovitskiy 等人（2020）所述完全相同的训练流程，并使用相应的预训练 ViT 模型 B/16，将模型训练 100 步。结果见图 6。我们发现，在不同的 k下，使用平滑 top-3 损失微调的 ViT 在 top-k 错误率方面优于使用交叉熵损失微调的 ViT。

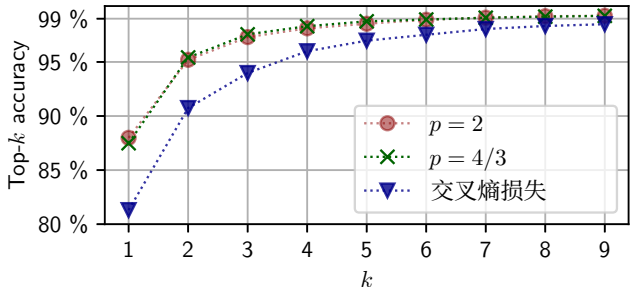


图 6。**验证集 top-k 准确率**：使用交叉熵损失或我们的平滑 top-3 损失进行训练时，在 CIFAR 100 上微调 ViT-B/16 的结果。使用 TPUv3-8 测得的运行时间如下：基线：9.5 秒/步，$p = 2$：9.7 秒/步， $p = 4/3$：10 秒/步。

稀疏混合专家模型。最后，我们通过视觉稀疏混合专家（V-MoE）（Riquelme 等人，2021）展示所提出的平滑 top-k 算子在大规模分类任务中的适用性。视觉变换器（ViT）由一系列自注意力层和 MLP 层组成。V-MoE 的思路是将 ViT 中的 MLP 替换为称为专家的稀疏门控 MLP 混合体。这样，每个图块标记只会激活部分专家。标记-专家分配的核心是路由机制，它会对门控值执行 top-k 操作。我们聚焦于采用专家选择路由框架（Zhou 等人，2022）的 MoE，其中每个专家分配到 k 个词元。我们在 JFT-300M 数据集（Sun 等人，2017）上训练 V-MoE 模型的 S/32 变体，该数据集包含超过 3.05 亿张图像，使用了 32×32 个图块。我们的模型有 32 个专家，每个专家在每个 MoE 层中从 $n = 400$ 个词元中选取 $k = 28$个进行分配。我们比较使用基线（硬 top-k）和松弛算子时的验证准确率。我们使用 $p = 2$ 和 Dykstra 投影算法，因为我们发现这是 TPU 上速度最快的方法。为与基线进行公平比较，我们采用 Zhou 等人（2022）提出的训练流程。由于 JFT-300M 数据集规模庞大（3.05 亿张图像），我们与 Liu 等人（2022）一样，仅进行了一次运行。我们发现，我们的方法提升了验证性能。结果见图 7。

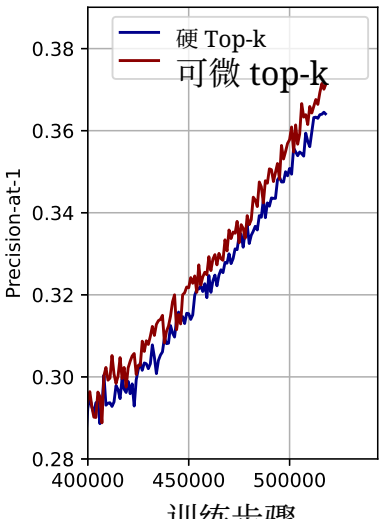


图 7。**Precision-at-1** 在 JFT-300M 数据集上，使用硬 top-k（基线，以蓝色表示）或几乎处处可微的 top-k（以红色表示）的稀疏 MoE 与 ViT-S/32 组合时的结果。我们放大了最后几个训练步骤，可以看到我们提出的方法优于基线。基线的运行时间为 10 小时，而几乎处处可微的 top-k 的运行时间为 15 小时（这里的瓶颈是梯度计算）。

7. 讨论

非线性的优势。 作为对 $\boldsymbol{x}$ 的幅值执行松弛 top-k 算子的替代方案，可以对 $|\boldsymbol{x}|$ 执行可微 top-k 掩码，然后将输出乘以 $\boldsymbol{x}$。这种替代方案同样会得到按幅值计算的可微 top-k 算子。然而，我们的算子在极限情形下的表现更有理论依据。例如，当 $\lambda \to \infty$ 时，很容易看出松弛的 top-k 掩码收敛到向量 $(k/n) \times 1_n$。因此，需要通过 n/k 重新缩放，才能在 $\lambda \to \infty$ 时得到恒等映射，这与我们的按幅值取 top-k 不同。从理论角度来看从……的角度来看，引入非线性使我们能够与 k-支撑范数建立连接。它还具有双共轭解释，我们认为这一点本身也很有意思。

对 p 的选择的敏感性。PAV 中所需的子问题只有在特定的 p取值下才有闭式形式。这就是我们在实验中重点研究 $p = 2$ 和 $p = \frac{4}{3}$ 的原因。不过，我们强调，所提出的方法适用于 p的任意取值。例如，我们提供了与图1中相同的示意图，见图8。

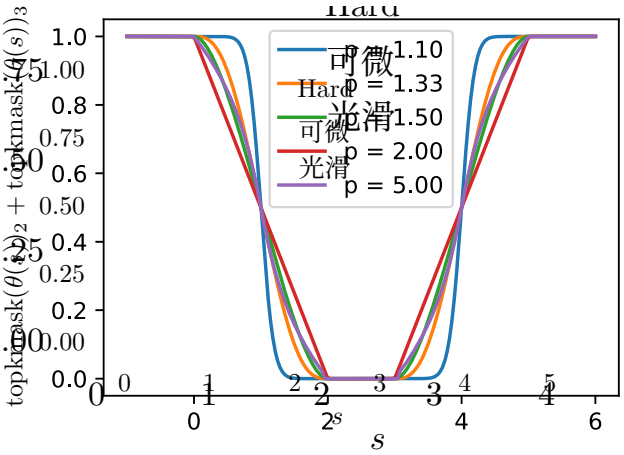


图*8*。**可微稀疏 top-k 掩码的示意图**。设置与图1相同，但 p的取值更多。

8. 结论

在这项工作中，我们提出了一个广义框架，用于获得快速、可微（或几乎处处可微）且稀疏的 top-k 和 top-k 掩码算子，包括按幅值选择数值的算子。通过将问题归约为保序优化，我们证明这些算子可以使用相邻违例合并（PAV）算法或 Dykstra 投影算法计算；后者在 TPU 硬件上速度更快。我们成功展示了这些算子在权重剪枝、top-k 损失以及作为视觉稀疏混合专家中的路由器方面的实用性。

致谢。 感谢 Vincent Roulet 和 Joelle Barral 对本文初稿提出意见。感谢 Felipe Llinares-L´opez 对实验提供的宝贵反馈，也感谢 Fabian Pedregosa 进行了富有成效的数学讨论。我们还感谢匿名审稿人提供的反馈。

参考文献

Adams, R. P. 和 Zemel, R. S. 通过 Sinkhorn 传播进行排名。*arXiv* 电子预印本，2011年。

Amos, B., Koltun, V., 和 Kolter, J. Z. 有限多标签投影层。*arXiv* 预印本 *arXiv:1906.08707*，2019年。

Argyriou, A., Foygel, R., 和 Srebro, N. 基于 k-support 范数的稀疏预测。神经信息处理系统进展，25，2012年。

Beck,A. 优化中的一阶方法。SIAM，2017年。

Berrada, L., Zisserman, A., 和 Kumar, M. P. 用于深度 Top-k分类的平滑损失函数。国际表示学习会议，2018年。

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.- P., 和 Bach, F. 使用可微扰动优化器进行学习。神经信息处理系统进展, 33:9508–9519, 2020年。

Best, M. J., Chakravarti, N., 和 Ubhaya, V. A. 在简单链式约束下最小化可分凸函数。*SIAM*优化期刊, 10(3):658–672, 2000.

Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., 和 Guttag, J. 神经网络剪枝的现状如何？机器学习与系统会议论文集, 2:129–146, 2020年。

Blondel, M. 使用投影预言机进行结构化预测。神经信息处理系统进展, 32, 2019年。

Blondel, M., Martins, A. F., 和 Niculae, V. 使用 Fenchel-Young损失进行学习。*J.Mach. Learn. Res.*, 21(35):1–69, 2020a.

Blondel, M., Teboul, O., Berthet, Q. 和 Djolonga, J. 快速可微排序与排名。载于国际机器学习会议，第 950–959 页。PMLR，2020b。

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-L´opez, F., Pedregosa, F. 和 Vert, J.-P. 高效且模块化的隐式微分。*arXiv* 预印本 *arXiv:2105.15183*，2021年。

Bowman, V. 排列多面体。*SIAM* 应用数学杂志，22(4):580–589，1972年。

Boyle, J. P. 和 Dykstra, R. L. 在希尔伯特空间中寻找凸集交集上的投影的方法。载于序约束统计推断进展，第 28–47 页。施普林格，1986年。

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. 和 Zhang, Q. JAX：Python+NumPy 程序的可组合变换，2018年。网址http://github.com/google/jax。Chapelle, O. 和 Wu, M. 平滑信息检索指标的梯度下降优化。信息检索，13(3):216–235，2010年。

Collins, M. D. 和 Kohli, P. 内存受限的深度卷积网络。*arXiv* 预印本 *arXiv:1412.1442*，2014年。

Combettes, P. L. 和 Pesquet, J.-C. 信号处理中的近端分裂方法。载于科学与工程逆问题的不动点算法，第 185–212 页。施普林格，2011年。

Cordonnier, J.-B.、Mahendran, A.、Dosovitskiy, A.、Weissenborn, D.、Uszkoreit, J. 和 Unterthiner, T. 用于图像识别的可微图像块选择。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 2351–2360 页，2021年。

Cuturi, M. Sinkhorn距离：以闪电般的速度计算最优传输。神经信息处理系统进展，第 26 卷，2013年。

Cuturi, M.、Teboul, O. 和 Vert, J.-P. 使用最优传输实现可微排名与排序。神经信息处理系统进展，第 32 卷，2019年。

Danskin, J. M. 极大极小理论及其应用。*SIAM* 应用数学杂志，14(4):641–664，1966年。

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. 等。《一张图像胜过 16×16 个词：用于大规模图像识别的 Transformer》。*arXiv* 预印本 *arXiv:2010.11929*，2020年。

Eriksson, A., Thanh Pham, T., Chin, T.-J. 和 Reid, I. k-支撑范数以及基数与秩的凸包络。发表于*IEEE* 计算机视觉与模式识别会议论文集，第 3349–3357 页，2015年。

Fedus, W., Zoph, B. 和 Shazeer, N. Switch Transformer：利用简单高效的稀疏性扩展至万亿参数模型，2021年。

Fedus, W., Dean, J., 和 Zoph, B. 深度学习中稀疏专家模型综述。*arXiv* 预印本 *arXiv:2209.01667*，2022年。

Frankle, J. 和 Carbin, M. 彩票假设：寻找稀疏、可训练的神经网络。*arXiv* 预印本 *arXiv:1803.03635*，2018年。Gale, T., Elsen, E. 和 Hooker, S. 深度神经网络中的稀疏性现状。*arXiv* 预印本 *arXiv:1902.09574*，2019年。Grover, A., Wang, E., Zweig, A. 和 Ermon, S. 通过连续松弛对排序网络进行随机优化。发表于国际学习表征会议，2019年。

Han、S.、Pool、J.、Tran、J. 和 Dally、W. 学习权重和连接，实现高效的神经网络。神经信息处理系统进展，28，2015年。

Hazimeh、H.、Zhao、Z.、Chowdhery、A.、Sathiamoorthy、M.、Chen、Y.、Mazumder、R.、Hong、L. 和 Chi、E. Dselect-k：专家混合中的可微选择及其在多任务学习中的应用。神经信息处理系统进展，34：29335–29347，2021年。

J¨arvelin、K. 和 Kek¨al¨ainen、J. 用于检索高度相关文档的信息检索评估方法。载于*ACM SIGIRForum*，第51卷，第243–250页。ACM，美国纽约州纽约市，2017年。

Jegelka、S.、Bach、F. 和 Sra、S. 面向用户友好型子模优化的反射方法。神经信息处理系统进展，26，2013年。

Krizhevsky, A.、Hinton, G. 等。通过微小图像学习多层特征。2009年。

Kyrillidis, A.、Becker, S.、Cevher, V. 和 Koch, C. 单纯形上的稀疏投影。载于 国际机器学习会议，第 235–243 页。PMLR，2013年。

Lapin, M.、Hein, M. 和 Schiele, B. Top-k 多分类支持向量机。神经信息处理系统进展，28，2015年。

Lapin, M., Hein, M. 和 Schiele, B. 针对 top-k 错误的损失函数：分析与见解。载于计算机视觉与模式识别会议论文集，2016年。Lewis, M., Bhosale, S., Dettmers, T., Goyal, N. 和 Zettlemoyer, L. 基础层：简化大型稀疏模型的训练。载于国际机器学习会议，第 6265–6274 页。PMLR，2021年。Lim, C. H. 和 Wright, S. J. 对置换多面体及相关多胞体进行高效的 Bregman 投影。载于人工智能与统计学，第 1205–1213 页。PMLR，2016年。

Liu, T., Puigcerver, J. 和 Blondel, M. 稀疏性约束最优传输。*arXiv* 预印本 *arXiv:2209.15466*，2022年。Malaviya, C., Ferreira, P. 和 Martins, A. F. 用于神经机器翻译的稀疏约束注意力。*arXiv* 预印本 *arXiv:1805.08241*，2018年。Martins, A. F. 和 Kreutzer, J. 学习易处理的内容：完全可微的神经易优先标注器。载于*2017*年自然语言处理实证方法会议论文集，第 349–362 页，2017年。McDonald, A. M., Pontil, M. 和 Stamos, D. 谱 k-支撑范数正则化。神经信息处理系统进展，第 27 卷，2014年。

N´emeth, A. 和 N´emeth, S. 如何使用相邻违例合并类型的算法投影到单调非负锥上。*arXiv* 预印本 *arXiv:1201.2343*，2012年。

Niepert, M.、Minervini, P. 和 Franceschi, L. 隐式最大似然估计：通过离散指数族分布进行反向传播。神经信息处理系统进展，34:14567–14579，2021年。

Petersen, F., Borgelt, C., Kuehne, H. 和 Deussen, O. 面向可扩展排序和秩监督的可微排序网络。发表于国际机器学习会议，第 8546–8555 页。PMLR，2021年。

Petersen, F., Kuehne, H., Borgelt, C. 和 Deussen, O. 可微 Top-k 分类学习。发表于国际机器学习会议，第 17656–17668 页。PMLR，2022年。

Prillo, S. 和 Eisenschlos, J. Softsort：argsort 算子的连续松弛。发表于国际机器学习会议，第 7793–7802 页。PMLR，2020年。

Qian, Y., Lee, J., Duddu, S. M. K., Dai, Z., Brahma, S., Naim, I., Lei, T. 和 Zhao, V. Y. 将多向量检索作为稀疏对齐。*arXiv* 预印本 *arXiv:2211.01267*，2022年。Riquelme, C., Puigcerver, J., Mustafa, B., Neumann, M., Jenatton, R., Susano Pinto, A., Keysers, D. 和 Houlsby, N. 通过稀疏专家混合扩展视觉。神经信息处理系统进展，34: 8583–8595，2021年。Rol´ınek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C. 和 Martius, G. 使用黑盒微分优化基于排名的指标。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第7620–7630页，2020年。

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G. 和 Dean, J. 超大规模神经网络：稀疏门控混合专家层。*arXiv* 预印本 *arXiv:1701.06538*，2017年。

Sinkhorn, R. 对角等价于具有指定行和列和的矩阵。《美国数学月刊》, 74(4):402–405, 1967. Sun, C., Shrivastava, A., Singh, S., and Gupta, A. 重新审视深度学习时代数据令人难以置信的有效性。载于*IEEE* 国际计算机视觉会议论文集, pp. 843–852, 2017. Tibshirani, R. J. Dykstra 算法、交替方向乘子法和坐标下降：连接、见解与拓展。神经信息处理系统进展, 30, 2017. Warmuth, M. K. and Kuzmin, D. 随机在线主成分分析算法，其遗憾界关于维度呈对数增长。《机器学习研究杂志》, 9 (Oct):2287–2320, 2008. Wiseman, S. and Rush, A. M. 将序列到序列学习视为束搜索优化。*arXiv* 预印本 *arXiv:1606.02960*, 2016. Xie, Y., Dai, H., Chen, M., Dai, B., Zhao, T., Zha, H., Wei, W., and Pfister, T. 通过最优传输实现可微 top-k。神经信息处理系统进展, 33:20520–20531, 2020. Zeng, X. and Figueiredo, M. A. 有序加权 l_1范数：原子形式、投影与算法。*arXiv* 预印本 *arXiv:1409.4271*, 2014. Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. 采用专家选择路由的混合专家模型。*arXiv* 预印本 *arXiv:2202.09368*, 2022. Ziegler, G. M. 多胞体讲义, 第 152 卷。Springer Science & Business Media, 2012.

A. 证明

A.1. 线性最大化预言机 - 命题证明1

假设 $\boldsymbol{w}$ 按降序排列，则对于任意 $\boldsymbol{x} \in \mathrm{R}^n$

$$\begin{aligned}\sum_{i=1}^{n} w_i x_{[i]} &= \max_{\sigma \in \Sigma} \langle \boldsymbol{x}_\sigma, \boldsymbol{w} \rangle \\ &= \max_{\pi \in \Sigma} \langle \boldsymbol{x}, \boldsymbol{w}_\pi \rangle \\ &= \max_{\boldsymbol{y} \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle.\end{aligned}$$

第一行中，我们利用内积在找到按降序对 σ 排序的排列时取得最大值。第二行中，我们利用了 $\langle \boldsymbol{x}_\sigma, \boldsymbol{w} \rangle = \langle \boldsymbol{x}, \boldsymbol{w}_\pi \rangle$，其中 π 是 σ 的逆置换。第三行中，我们利用了线性规划基本定理，该定理保证解在多面体的某个顶点处取得。总之，若 σ 是按降序对 $\boldsymbol{x}$ 排序的排列，则 $\boldsymbol{y}^\star = \boldsymbol{w}_{\sigma^{-1}}$。

A.2. 松弛算子 - 命题证明2

回顾一下，$f^*_{\varphi,\, R}(\boldsymbol{y},\ \boldsymbol{w}) \coloneqq f^*_\varphi(\boldsymbol{y},\ \boldsymbol{w}) + R(\boldsymbol{y})$。于是我们有

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \max_{\boldsymbol{y} \in \mathbb{R}^n} \langle \boldsymbol{y}, \boldsymbol{x} \rangle - f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) - R(\boldsymbol{y}).$$

众所周知，若 h_1 和 h_2 是两个凸函数，则 $(h_1 + h_2)^*$ 等于 h_1^* 与 h_2^* (Beck，2017年，定理 4.17) 中的下确界卷积：

$$(h_1 + h_2)^*(\boldsymbol{x}) = (h_1^* \square h_2^*)(\boldsymbol{x}) \coloneqq \min_{\boldsymbol{u} \in \mathbb{R}^n} h_1^*(\boldsymbol{u}) + h_2^*(\boldsymbol{x} - \boldsymbol{u}).$$

由 $h_1 = f^*_\varphi$ 和 $h_2 = R$，因此可得

$$f_{\varphi,R}(\boldsymbol{x}, \boldsymbol{w}) = \min_{\boldsymbol{u} \in \mathbb{R}^n} R^*(\boldsymbol{x} - \boldsymbol{u}) + f_\varphi(\boldsymbol{u}, \boldsymbol{w}).$$

最后，$\boldsymbol{y}^\star$ 的表达式可由应用 Danskin 定理得到。

A.3. 共轭函数 - 命题3的证明

我们有

$$\begin{aligned}f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - f_\varphi(\boldsymbol{x}, \boldsymbol{w}) \\ &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \max_{\boldsymbol{y}' \in P(\boldsymbol{w})} \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle \\ &= \max_{\boldsymbol{x} \in \mathbb{R}^n} \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle.\end{aligned}$$

若 $\boldsymbol{w} \in \mathrm{R}^n_+$，则对所有 $\boldsymbol{y}' \in P(\boldsymbol{w})$，都有 $\boldsymbol{y}' \in \mathrm{R}^n_+$。于是，函数 $(\boldsymbol{x},\ \boldsymbol{y}') \mapsto \langle \boldsymbol{x},\ \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x})\ \boldsymbol{y}' \rangle$ 是凹凸函数，因此我们可以交换最小值和最大值，得到

$$\begin{aligned}f^*_\varphi(\boldsymbol{y}, \boldsymbol{w}) &= \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \max_{\boldsymbol{x} \in \mathbb{R}^n} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \langle \varphi(\boldsymbol{x}), \boldsymbol{y}' \rangle \\ &= \min_{\boldsymbol{y}' \in P(\boldsymbol{w})} \sum_{i=1}^{n} y'_i \phi_i^*(y_i / y'_i).\end{aligned}$$

A.4. 双共轭解释 - 命题的证明 4

有

$$\Phi_k^*(\boldsymbol{x}) = \max_{\boldsymbol{y} \in S_k} \langle \boldsymbol{x}, \boldsymbol{y} \rangle - \sum_{i=1}^{n} \phi^*(y_i).$$

As in (Kyrillidis 等人，2013年)，令 Σ_k 为以下集合：由 $[n]$ 中基数小于 k 的子集构成。Th en

$$\Phi_k^*(\boldsymbol{x}) = \max_{I\subset\Sigma_k} \max_{\boldsymbol{y}\in\mathbb{R}^n} \sum_{i\in I} x_i y_i - \phi^*(y_i).$$

由此可得

$$\Phi_k^*(\boldsymbol{x}) = \max_{I\subset\Sigma_k} \sum_{i\in I} \phi(x_i) = \sum_{i=1}^{k} \varphi(\boldsymbol{x})_{[i]}.$$

取共轭函数即可得到所需结果。

A.5. 化简为保序优化

我们关注 φ 符号不变的情形，即 $\varphi(\boldsymbol{x}) = \varphi(-\boldsymbol{x})$，因为情形 $\varphi(\boldsymbol{x}) = \boldsymbol{x}$ 已在（Lim &Wright，2016年；Blondel 等人，2020b）中解决。

我们首先证明 $\boldsymbol{u}^\star$ 保持 $\boldsymbol{x}$ 的符号。具体而言，我们证明对于任意 $\boldsymbol{u}\in\mathrm{R}^n$， $\boldsymbol{u}' \coloneqq \mathrm{sign}(\boldsymbol{x})\circ|\boldsymbol{u}|$ 的目标值都小于 $\boldsymbol{u}$。回忆

$$f_{\varphi,R}(\boldsymbol{x},\boldsymbol{w}) = \min_{\boldsymbol{u}\in\mathbb{R}^n} R^*(\boldsymbol{x}-\boldsymbol{u}) + f_\varphi(\boldsymbol{u},\boldsymbol{w}) = \min_{\boldsymbol{u}\in\mathbb{R}^n} R^*(\boldsymbol{x}-\boldsymbol{u}) + f(\varphi(\boldsymbol{u}),\boldsymbol{w}).$$

显然，我们有 $f(\varphi(\boldsymbol{u}')\ \boldsymbol{w}) = f(\varphi(\boldsymbol{u})\ \boldsymbol{w})$。此外，若 $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$，则 $R^*(\boldsymbol{x}-\boldsymbol{u}) = \sum_{i=1}^n r^*(x_i-u_i)$。如果 r^* 在 R+上符号不变且单调递增，那么我们有

$$\begin{aligned} r^*(x_i-u_i') &= r^*(\mathrm{sign}(x_i)(|x_i|-|u_i|)) \\ &= r^*(|x_i|-|u_i|) \\ &\le r^*(x_i-u_i), \end{aligned}$$

这里用到了反三角不等式 $||x_i-|u_i|| \le |x_i-u_i|$。由此可得， $\boldsymbol{u}^\star$ 与 $\boldsymbol{x}$同号。因此，从现在起，我们可以假设 $\boldsymbol{x}\in\mathrm{R}^n_+$，这意味着 $\boldsymbol{u}\in\mathrm{R}^n_+$。

由于 $\boldsymbol{u}\in\mathrm{R}^n_+$，且 ϕ 在 R+上单调递增，因此有 $u_{\sigma_1}\ge\cdots\ge u_{\sigma_n} \Rightarrow \phi(u_{\sigma_1})\ge\cdots\ge\phi(u_{\sigma_n})$。我们知道，对于所有 $\boldsymbol{u}\in\mathrm{R}^n$和 $\boldsymbol{w}\in\mathrm{R}^n$，都有 $f(\varphi(u)\ \boldsymbol{w}) = \langle\varphi(\boldsymbol{u})_\sigma,\ \boldsymbol{w}\rangle = \langle\varphi(\boldsymbol{u}_\sigma)\ \boldsymbol{w}\rangle$，其中 σ 是将 $\boldsymbol{u}$ 按降序排列的排列。从现在起，我们固定 σ 为将 $\boldsymbol{u}^\star$排序的排列。我们将在下文证明，这与将 $\boldsymbol{x}$排序的排列相同。于是有

$$f_{\varphi,R}(\boldsymbol{x},\boldsymbol{w}) = \min_{\boldsymbol{u}\in\mathbb{R}^n} R^*(\boldsymbol{x}-\boldsymbol{u}) + \langle\varphi(\boldsymbol{u}_\sigma),\boldsymbol{w}\rangle.$$

通过变量替换 $\boldsymbol{v} = \boldsymbol{u}_\sigma \Leftrightarrow \boldsymbol{v}_{\sigma^{-1}} = \boldsymbol{u}$，我们得到

$$f_{\varphi,R}(\boldsymbol{x},\boldsymbol{w}) = \max_{v_1\ge\cdots\ge v_n} R^*(\boldsymbol{x}_\sigma-\boldsymbol{v}) + \langle\varphi(\boldsymbol{v}),\boldsymbol{w}\rangle$$

其中我们用到了：若 $R(\boldsymbol{y}) = \sum_{i=1}^n r(y_i)$，则

$$R^*(\boldsymbol{x}-\boldsymbol{u}) = R^*(\boldsymbol{x}-\boldsymbol{v}_{\sigma^{-1}}) = R^*(\boldsymbol{x}_\sigma-\boldsymbol{v}).$$

令 $\boldsymbol{s}\coloneqq\boldsymbol{x}_\sigma$。还需证明 $s_1\ge\cdots\ge s_n$，也就是说， $\boldsymbol{s}$和 $\boldsymbol{v}^\star$均按降序排列。假设 $s_j > s_i$对于某个 $i<j$。令 $\boldsymbol{s}'$为 $\boldsymbol{s}$的副本，其中 s_i和 s_j互换。由于 R^*是凸函数，根据（Blondel 等人，2020b，引理4），

$$R^*(\boldsymbol{s}-\boldsymbol{v}^\star) - R^*(\boldsymbol{s}'-\boldsymbol{v}^\star) = r^*(s_i-v_i^\star) + r^*(s_j-v_j^\star) - r^*(s_j-v_i^\star) - r^*(s_i-v_j^\star) \ge 0,$$

这与 $\boldsymbol{v}^\star$ 及其对应的 σ 均为最优解的假设相矛盾。

A.6. 子问题推导

情形 $\phi(x) = \frac{1}{2}x^2$**和** $R(\boldsymbol{x}) = \frac{\lambda}{2}\|\boldsymbol{x}\|^2$。有 $h_i(\gamma) = \frac{1}{2}(w_i\gamma^2 + \frac{1}{\lambda}(s_i-\gamma)^2)$，使得 $\frac{dh_i}{d\gamma} = (w_i + \frac{1}{\lambda})\gamma - \frac{1}{\lambda}s_i$.因此，

$$\frac{d\sum_{i\in B}h_i(\gamma)}{d\gamma} = \gamma\sum_{i\in B}(w_i+\frac{1}{\lambda}) - \frac{1}{\lambda}\sum_{i\in B}s_i.$$

由于此外　　由于 $\sum_{i\in B} h_i$ 是凸函数，我们可知其最小值由令导数为零得到，因此

$$\gamma_B^\star = \frac{\sum_{i\in B} s_i}{\sum_{i\in B}(\lambda w_i + 1)}.$$

备注。该结果表明，当 λ 足够小时，相较于硬算子，我们可以控制所提出的算子所引入的近似误差。为简单起见，我们来考察不存在并列的情形：$\forall i \neq j, x_i \neq x_j$。这意味着 $s_1 > s_2 > \cdots > s_n$。在这种情况下，当 λ 足够小时，我们得到

$$\gamma^\star_{\{1\}} > \gamma^\star_{\{2\}} > \cdots > \gamma^\star_{\{n\}}$$

从而最优的 p　　划分由 PAV 算法通过取 $B_i = \{i\}$给出。因此，$v_i^\star = \frac{s_i}{\lambda w_i + 1}$. 接下来有——　　s

$$\boldsymbol{u}^\star = \operatorname{sign}(\boldsymbol{x}) \circ \boldsymbol{v}^\star_{\sigma^{-1}} = \frac{\boldsymbol{x}}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1}.$$

将其代入 $\boldsymbol{y}^\star$ 命题2 可得

$$\boldsymbol{y}^\star = \frac{\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1}.$$

由于硬算子为　　由 $\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}$ 给出，因此无穷范数下的近似误差 $e_\lambda(\boldsymbol{x}, \boldsymbol{w})$ 仅有如下界 b　　y

$$e_\lambda(\boldsymbol{x}, \boldsymbol{w}) := \|\boldsymbol{w}_{\sigma^{-1}} \circ \boldsymbol{x}(\frac{1}{\lambda \boldsymbol{w}_{\sigma^{-1}} + 1} - 1)\|_\infty \leq \|\boldsymbol{w}\|_\infty \|\boldsymbol{x}\|_\infty \|\frac{\lambda \boldsymbol{w}}{\lambda \boldsymbol{w} + 1}\|_\infty.$$

在 topkmag 情形下，$w = \mathbf{1}_k$，因此 $e_\lambda(\boldsymbol{x}, \boldsymbol{w}) \leq \lambda \|\boldsymbol{x}\|_\infty$。

情形 $\phi(x) = x$ **和** $R(\boldsymbol{x}) = \frac{\lambda}{p}\|\boldsymbol{x}\|^p$ **其中** $p = \frac{4}{3}$。有 $h_i(\gamma) = w_i\gamma + \frac{1}{4\lambda^3}(s_i - \gamma)^4$，因此 $\frac{dh_i}{d\gamma} = w_i + \frac{1}{\lambda^3}(\gamma - s_i)^3$. 所以，

$$\frac{d\sum_{i\in B} h_i(\gamma)}{d\gamma} = \frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \sum_{i\in B} w_i.$$

此外，由于 $\sum_{i\in B} h_i$ 是凸函数，我们可以通过令导数为零得到其最小值，因而只需解三次多项式方程

$$\frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \sum_{i\in B} w_i = 0.$$

在实践中，我们使用 *numpy* 库中的求根器来解这个方程。请注意，取 $p = \frac{4}{3}$ 会得到比 $p = \frac{3}{2}$ 更简单的子问题，因此我们选择 p。

情形 $\phi(x) = \frac{1}{2}x^2$ **和** $R(\boldsymbol{x}) = \frac{\lambda}{p}\|\boldsymbol{x}\|^p$ **其中** $p = \frac{4}{3}$。推导与前一种情形非常相似。事实上，有$h_i(\gamma) = \frac{1}{2}w_i\gamma^2 + \frac{1}{4\lambda^3}(s_i - \gamma)^4$因此，$\frac{dh_i}{d\gamma} = w_i\gamma + \frac{1}{\lambda^3}(\gamma - s_i)^3$. 所以，　　—

$$\frac{d\sum_{i\in B} h_i(\gamma)}{d\gamma} = \frac{1}{\lambda^3}\sum_{i\in B}(\gamma - s_i)^3 + \gamma\sum_{i\in B} w_i.$$

A.7. 微分 - 命题7

情形 $\phi(x) = x$。有 $h_i(\gamma) = \frac{1}{q}|s_i - \gamma|^q + w_i v_i$. 对于 PAV 中任意一个最优的 B，均有最优性条件

$$\sum_{i\in B}(\operatorname{sign}(\gamma_B^\star - s_i)|\gamma_B^\star - s_i|^{q-1} + w_i) = 0.$$

利用隐函数定理可知，$\gamma_B^\star$ 是可微的，对任意 s_i 求关于 $i \in B$ 的导数可得

$$\frac{\partial \gamma_B^\star}{\partial s_i} = \frac{|\gamma_B^\star - s_i|^{q-2}}{\sum_{j\in B}|\gamma_B^\star - s_j|^{q-2}}.$$

情形 $\phi(x) = -\frac{1}{2}x^2$。 类似的计算可得

$$\frac{\partial \gamma_B^\star}{\partial s_i} = \frac{(q-1)|\gamma_B^\star - s_i|^{q-2}}{\sum_{j\in B}(q-1)|\gamma_B^\star - s_j|^{q-2} + w_j}.$$

A.8. 保序优化的对偶

$$\begin{aligned}
\min_{v_1\geq\cdots\geq v_n} \sum_{i=1}^n h_i(v_i) &= \min_{\boldsymbol{v}\in\mathbb{R}^n} \max_{\boldsymbol{\alpha}\in\mathbb{R}_+^{n-1}} \sum_{i=1}^n h_i(v_i) - \alpha_i(v_i - v_{i+1}) \\
&= \min_{\boldsymbol{v}\in\mathbb{R}^n} \max_{\boldsymbol{\alpha}\in\mathbb{R}_+^{n-1}} \sum_{i=1}^n h_i(v_i) - v_i(\alpha_i - \alpha_{i-1}) \\
&= \max_{\boldsymbol{\alpha}\in\mathbb{R}_+^{n-1}} -\left[\sum_{i=1}^n h_i^*(\alpha_i - \alpha_{i-1})\right]
\end{aligned}$$

其中， $\alpha_0 \coloneqq 0$ 和 $\alpha_n \coloneqq 0$ 为常数（即不参与优化）。通过$v_i^\star = (h_i^*)'(\alpha_i^\star - \alpha_{i-1}^\star)$，可由 $\boldsymbol{\alpha}^\star$ 恢复出最优解 $\boldsymbol{v}^\star$ 。由于 $\boldsymbol{\alpha}$ 仅受非负约束，我们可以通过坐标上升求解对偶问题。对于 $i \in \{1,\dots,n-1\}$，与 α_i相关的子问题为

$$\max_{\alpha_i\in\mathbb{R}_+} -h_i^*(\alpha_i - \alpha_{i-1}) - h_{i+1}^*(\alpha_{i+1} - \alpha_i).$$

该子问题是一个带非负约束的简单单变量问题。定义 α_i' 为 $(h_i^*)'(\alpha_i' - \alpha_{i-1}) - (h_{i+1}^*)'(\alpha_{i+1} - \alpha_i') = 0$的解。则解为 $\alpha_i^\star = [\alpha_i']_+$。
实际上，我们可以交替进行并行更新 $\alpha_1, \alpha_3, \dots$， 以及并行更新 $\alpha_2, \alpha_4, \dots$。奇数坐标对应的对偶变量属于集合 $C_1 = \{\boldsymbol{v} \in \mathrm{R}^n\colon\ v_1 \geq v_2, v_3 \geq v_4, \dots\}$，偶数坐标对应的对偶变量属于集合 $C_2 = \{\boldsymbol{v} \in \mathrm{R}^n\colon\ v_2 \geq v_3, v_4 \geq v_5, \dots\}$。假设每个 h_i^* 都可微，则坐标上升会收敛到最优对偶解；这等价于每个 h_i 都是严格凸的。

注意，这个子问题可以在原始空间中改写为

$$\begin{aligned}
&\max_{\alpha_i\in\mathbb{R}_+} -h_i^*(\alpha_i - \alpha_{i-1}) - h_{i+1}^*(\alpha_{i+1} - \alpha_i) \\
&= \max_{\alpha_i\in\mathbb{R}_+} -\left[\max_{v_i}(\alpha_i - \alpha_{i-1})v_i - h_i(v_i)\right] - \left[\max_{v_{i+1}}(\alpha_{i+1} - \alpha_i)v_{i+1} - h_{i+1}(v_{i+1})\right] \\
&= \min_{v_i, v_{i+1}} \alpha_{i-1}v_i + h_i(v_i) - \alpha_{i+1}v_{i+1} + h_{i+1}(v_{i+1}) + \max_{\alpha_i\in\mathbb{R}_+} \alpha_i(v_{i+1} - v_i) \\
&= \min_{v_i\geq v_{i+1}} h_i(v_i) + h_{i+1}(v_{i+1}) + \alpha_{i-1}v_i - \alpha_{i+1}v_{i+1}.
\end{aligned}$$

事实上，在欧几里得情形下，已知 Dykstra 算法在原始空间中的形式与对偶空间中的分块坐标上升等价（Tibshirani，2017年）。因此，分块坐标上升可视为将 Dykstra 算法推广到非欧几里得情形的一种优雅方式。

B. 补充材料

B.1. k-支撑负熵

当 $\phi(x) = e^{x-1}$ 和 $\boldsymbol{w} = \mathbf{1}_k$时，我们得到

$$f_\varphi^*(\boldsymbol{y}, \boldsymbol{w}) = \min_{\boldsymbol{z}\in[0,1]^n} \sum_{i=1}^n y_i \log(\frac{y_i}{z_i}) \quad \text{s.t.} \quad \langle \boldsymbol{z}, \mathbf{1}\rangle = k.$$

我们称之为 k-支撑负熵。

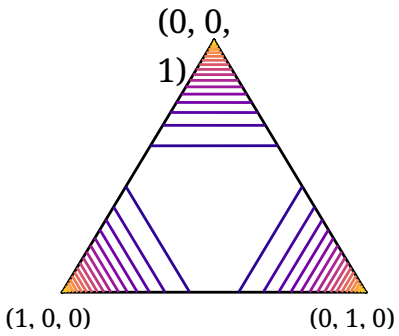


图 *9.* **单纯形上 k-支撑熵的等高线**，其中 $n = 3$ 和 $k = 2$。颜色越浅，表示数值越低。

B.2. PAV 算法

我们给出 PAV 的伪代码，改编自Lim& Wright (2016年)。回顾一下，我们定义

$$\gamma_B^\star = \operatorname*{argmin}_{\gamma \in \mathbb{R}} \sum_{i \in B} h_i(\gamma).$$

算法 1相邻违例合并（PAV）

输入：凸函数 $\{h_i : \mathrm{R} \to \mathrm{R}\}_{i \in [n]}$初始化分区
$P \leftarrow \{\{i\} | i \in [n]\}$初始化 $v_i \leftarrow \gamma^\star_{\{i\}}$ 对所有 $i \in [n]$ 当存在
i 使得 $v_i < v_{i+1}$ 时执行 查找 B_{r_i} 和 $B_{r_{i+1}}$，使其属于
P 且满足 $i \in B_{r_i}$ 和 $i+1 \in B_{r_{i+1}}$从 P 中移除 B_{r_i} 和
$B_{r_{i+1}}$ 将 $B_{r_i} \cup B_{r_{i+1}}$ 添加至 P通过求解 (6) 计算
$\gamma^\star_{B_{r_i} \cup B_{r_{i+1}}}$ 为所有 $r \in B_{r_i} \cup B_{r_{i+1}}$ 赋值 $v_r \leftarrow \gamma^\star_{B_{r_i} \cup B_{r_{i+1}}}$
结束循环 返回 v

C. 实验细节

C.1. 神经网络中的权重剪枝

在 MNIST 数据集上进行实验时，我们使用随机梯度下降训练多层感知机，批量大小为 128，学习率固定为 10^{-2}。模型训练了30轮。在硬件方面，我们使用一块 GPU。

C.2. 平滑 Top-k 损失

在 CIFAR-100 数据集上的实验中，我们使用随机梯度下降训练 ViT-B/16，动量设为 0.9，批量大小为 512。

对于交叉熵损失，我们遵循 Dosovitskiy 等人 (2020年的训练流程：预热阶段持续至学习率达到 3×10^{-3}。模型使用余弦学习率调度器训练 100 个步骤。对于这一训练步骤数，该学习率设置取得了最佳性能。

对于我们的前三名损失：预热阶段持续至学习率达到 5×10^{-3}。模型使用余弦学习率调度器训练 100 个步骤。

在硬件方面，我们使用 8 个张量处理单元。

C.3. 稀疏混合专家模型

我们在 JFT-300M 数据集（Riquelme 等人，2021年）上训练 V-MoE S/32 模型（Sun 等人，2017年）。JFT 是一个多标签数据集，因此准确率并非合适的指标，因为每张图像可能有多个标签。因此，我们使用常用的 *Top-1* 精确率指标（J¨arvelin& Kek¨al¨ainen，2017年）来衡量模型质量。训练流程与 Riquelme 等人（2021年）中所述的流程类似，但我们替换了路由算法。具体而言，我们以 Zhou 等人（2022年）所述的专家选择路由算法作为基线，并将其中使用的不可微 top-k 操作替换为我们的可微方法（我们执行 10 次 Dykstra 算法迭代）。我们使用与Riquelme 等人 (2021年)中所述完全相同的超参数，但专家选择路由无需任何辅助损失。具体而言，我们使用 4 096 的批量大小训练 7 个训练轮数。我们使用 Adam 优化器 ($\beta_1 = 0.9$, $\beta_2 = 0.999$)，峰值学习率为 10^{-3}，先预热 10 000 步，然后进行线性衰减。我们采用温和的数据增强（随机裁剪和水平翻转），并对所有参数使用 10^{-1} 权重衰减进行正则化。我们在 TPUv2-128 设备上训练了这两个模型。