arXiv:2603.04133v1 [stat.ML] 4 Mar 2026

利用最大加神经网络中的次梯度稀疏性

Ikhlas Enaieh [1] **Olivier Fercoq** [1]

摘要

深度神经网络是解决机器学习问题的强大工具，但其训练往往涉及稠密且代价高昂的参数更新。在本工作中，我们采用一种新颖的最大加神经架构，其中经典的加法与乘法分别被最大值与求和运算所取代。该架构在可解释性方面前景广阔，但其训练颇具挑战。一个显著特征是，这种代数结构会自然地在次梯度中诱导稀疏性，因为只有对最大值有贡献的神经元才会影响损失。然而，标准反向传播无法利用这种稀疏性，从而导致不必要的计算。在本工作中，我们聚焦于最坏样本损失的最小化，从而将这种稀疏性传递到优化损失上。为此，我们提出一种稀疏次梯度算法，显式利用这种代数稀疏性。通过使优化过程适配最大加模型的非光滑特性，我们的方法在保留理论保证的同时实现了更高效的更新。这揭示了一条将代数结构与可扩展学习相衔接的原则性路径。

1. 引言

深度神经网络（DNN）在从计算机视觉到自然语言处理的各种任务中都取得了显著成功，这得益于它们能够从高维数据中学习复杂的模型模式（LeCun 等人，2015年）。然而，这种表达能力也带来了巨大的计算成本：训练此类模型通常需要对数百万个参数进行稠密更新（Livni 等人，2014年），而不论这些参数中有多少实际上会影响模型对给定样本的输出。这种低效促使人们寻找能够利用神经网络的其他结构属性来减少冗余计算，同时不影响性能。

本文研究基于最大加代数和最小加代数构建的神经网络（Dimitriadis 与 Maragos，2021年）。与计算输入加权和的传统神经元不同，这些架构依赖于基于选择的运算。最大加神经元以最大值取代加法、以求和取代乘法，其输出形式为 $\hat{}y = \max_j\{x_j + w_j\}$，而最小加神经元则计算相应的最小值，呈现互补行为。在这两种情况下，只有达到最大值或最小值的输入才会影响神经元的输出，其余输入均不活跃。这些代数运算产生的次梯度天然稀疏，为开发更高效的学习算法奠定了良好基础。

遗憾的是，传统反向传播和自动微分框架并未针对这类稀疏、非光滑结构进行优化。无论模型的稀疏性如何，它们都会计算梯度的所有坐标（Sra 等人，2012年），从而导致冗余计算和有限的可扩展性（Daniely 等人，2023）。相比之下，最大加网络和最大减网络中的导数信息呈现为稀疏次梯度：只有位于活跃路径上的权重——即对应于取得最大值或最小值的输入的权重——会接收到非零更新，而所有其他坐标都保持为零（Tsiamis 与 Maragos，2019年）。然而，标准优化方法未能利用这一特性：它们将这些模型视为稠密模型，并将更新传播至所有参数，而不是将计算集中在真正重要的部分。为克服这些局限，我们开发了一种专为最大加/最小加神经网络的非凸、非光滑特性而设计的稀疏次梯度训练算法，从而仅沿着活跃计算路径进行更新。

我们对这些架构感兴趣，是因为它们能够在前向与反向传播中自然地产生稀疏性（Tsiamis& Maragos，2019年），因此成为稠密网络颇具吸引力的替代方案。最大 $+$）和（最小 $+$）代数已应用于以往的研究，并在各种任务中取得了有竞争力的表现。例如，最大加算子已用于滤波器选择和

[1]法国巴黎电信学院（Telecom Paris，巴黎理工学院）图像、数据与信号系（IDS）。通讯请联系：Ikhlas Enaieh <ikhlas.enaieh@telecom-paris.fr>、Olivier Fercoq<olivier.fercoq@telecom-paris.fr>。

预印本。*2026年3月5日* .

模型剪枝（Zhang et al.，2019年），而 Min-Max-Plus 架构已被证明是连续函数的通用逼近器（Luo &Fan，2021年）。这些结果表明，转向结构化稀疏架构不一定会牺牲表达能力。

2. 问题设定与符号约定

我们首先回顾模型的基本构件。

定义 2.1(形态学感知机 (Mondal 等人，2019年))。给定一个输入向量 $\mathbf{x} \in \mathbb{R}_{\max}^N$ (其中 $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$)、一个权重向量 $\mathbf{w} \in \mathbb{R}_{\max}^N$和一个偏置 $b \in \mathbb{R}_{\max}$，形态学感知机的激活计算如下

$$a(\mathbf{x}) = \max \left\{ b, \max_{1 \le i \le N} \{x_i + w_i\} \right\},$$

其中，x_i（分别是 w_i）表示$\mathbf{x}$（分别是$\mathbf{w}$）的第 i 个分量。

这个感知机凸显了（max $+$）代数的独特特征：只有达到最大值的输入才会对激活产生贡献，自然地带来稀疏前向计算。为了研究这种稀疏性在训练过程中如何传播，我们现在转向损失函数及其次梯度。

2.1. 损失函数与稀疏性动机

遵循（Zhang 等人，2019年），我们研究线性 Max-Plus 模型：一个带 ReLU 激活的线性层，后接一个（max $+$）层和 softmax 输出。令 $z_{i,\ d}$ 表示样本 i 在类别 d上的 softmax 前分数，并且

$$\hat{y}_{i,d} = \frac{\exp(z_{i,d})}{\sum_{d'=1}^{C} \exp(z_{i,d'})}$$

表示类别 d的预测概率，其中 $y_i \in \{1, \ldots, C\}$为真实标签。

我们比较两种设置下类别交叉熵（CCE）损失的次梯度稀疏性：

1. **单样本损失**。随机选取样本 $i \sim \mathcal{U}$（1，N）的损失为

$$\text{Loss}_i(w) = -\sum_{d=1}^{C} \mathbb{1}_{\{y_i = d\}} \log(\hat{y}_{i,d}).$$

2. **平均损失**。所有 N 个样本的 CCE 均值为

$$\text{Loss}_A(w) = \frac{1}{N} \sum_{i=1}^{N} \text{Loss}_i(w).$$

由于感知机的最大值结构，这些损失的次梯度是稀疏的，因此只有模型参数会得到部分更新。为了衡量这一效应，我们采用稀疏度指标（Nesterov，2014年）

$$\gamma(x) = \frac{\text{number of non-zero elements in } x}{\dim(x)},$$

它可以揭示向量 x中非零元素的数量。例如，我们考察了一个以独立同分布的 Glorot 均匀参数初始化的形态感知器模型（Glorot 与 Bengio，2010年），并计算了 MNIST 数据集（LeCun 等人，2002年）上次梯度的稀疏程度。我们比较了平均损失的次梯度稀疏性，以及单张图像的平均次梯度稀疏性：

$$\gamma\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial \text{Loss}_i}{\partial w}\right) = 0.82, \quad \frac{1}{N}\sum_{i=1}^{N}\gamma\left(\frac{\partial \text{Loss}_i}{\partial w}\right) = 0.048$$

这些发现表明，要有效利用 Max-Plus 结构所诱导的稀疏性，训练策略必须基于单个样本，而非平均更新。因此，我们采用一种策略，在每次迭代中选取损失最大的训练样本。该策略既利用了次梯度稀疏性（如（Nesterov，2014年）所示），也利用了鲁棒性，将学习集中于网络预测最薄弱的部分。这引出了以下优化问题：

$$\min_w \mathcal{L}(w) = \min_w \max_{1 \le i \le N} \text{Loss}_i(w)$$

平均损失反映了典型表现，而最大损失则直接决定分类最差样本的表现。以下命题表明，这种控制实际上足以确保训练集上的完美分类。

命题 2.2 (最大 SCCE 阈值下的完美分类)。如果最大稀疏分类交叉熵损失严格小于 $\log 2$，那么模型在训练集上的分类准确率达到 100%。

证明见附录 A.3。

然而，每一步都要计算大型数据集上的最大损失，需要 $\mathcal{O}(N)$ 次运算，计算开销很大。为缓解这一问题，我们引入短计算树（SCT）结构，以支持对数时间更新和最大值跟踪（Nesterov，2014年）。这使我们的最大损失表述能够扩展到实际规模的数据集。

2.2. 短计算树

定义 2.3 (短计算树（SCT）（Nesterov，2014年））。给定一个输入向量 $x \in \mathbb{R}^N$，其中有 $N = 2^k$，$k \ge 1$，短计算树（SCT）是一棵高度为 $\log_2 N$、具有 $k+1$ 层的二叉树，用于分层计算对称函数（例如，最大值）。

在底层，每个叶节点包含输入向量 x 中的一个元素。每个内部节点计算其两个子节点的二元最大值：

- 在第 0 层，树中存放着输入值：

$$v_{0,i} = x(i), \quad i = 1, \ldots, N.$$

- 树的内部节点通过递归计算：

$$v_{i+1,j} = \psi_{i+1,j}(v_{i,2j-1}, v_{i,2j}),$$
$$i = 0, \ldots, k-1, \quad j = 1, \ldots, 2^{k-i-1}.$$
In our case, $\psi_{i,j}(a, b) = \max(a, b)$.

根节点包含整体最大值。

这种分层构造通过一系列有结构的两两比较来计算最大值，并且这些比较可以在迭代过程中重复利用。虽然 SCT 的初始构造与标准线性扫描的代价相同，但当需要反复更新最大值时，其优势便显现出来。在我们的设定中，每次迭代只修改一个输入项：令 x 表示当前输入向量，$\tilde{x}$ 表示一次迭代后的更新后的向量。这一变化的影响会沿着一条唯一的路径，从对应的叶节点传递到树的根节点。因此，更新最大值时，每层只需访问一个节点。

表*1.* 最大值计算的复杂度：标准最大值计算与 SCT表示

操作	目标	标准	SCT
计算	$\max_{1\le i\le N} x_i$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
更新	$\max_{1\le i\le N} \tilde{x}_i$	$\mathcal{O}(N)$	$\boldsymbol{\mathcal{O}(\log\ N)}$

如表 1所总结，采用 SCT 可将更新成本从 $\mathcal{O}(N)$ 降至 $\mathcal{O}(\log\ N)$，使其成为迭代式 max-plus 优化中效率显著更高的替代方案。

将最大损失表述嵌入具体网络架构时，这种效率尤为重要；我们首先从零隐藏层模型开始。

3. 模型架构

3.1. 零隐藏层模型

我们从用于分类的基线架构开始，其中网络将输入直接映射到输出类别分数，使用单个Max-Plus层，随后接softmax激活。

形式上，训练目标为

$$\min_{W} \max_{1\le n\le N} \left(-\log(\hat{y}_{n,y_n})\right),$$

其中，$y_n \in \{1, \ldots, C\}$ 是样本 n 的真实标签，而类别 d 的预测概率$\hat{}y_{n,d}$ 由下式给出

$$\hat{y}_{n,d} = \frac{\exp\big(\max_p \left(X_{n,p} + W_{d,p}\right)\big)}{\sum_{d'=1}^{C} \exp\big(\max_p \left(X_{n,p} + W_{d',p}\right)\big)}. \quad (1)$$

这里，$X \in \mathbb{R}^{N\times P}$ 是数据矩阵，其元素$X_{n,p}$表示样本 n的第 p个特征；$W \in \mathbb{R}^{C\times P}$ 是权重矩阵。

展开后可得

$$\min_{W} \max_{n} \Big(-\max_{p} \left(X_{n,p} + W_{y_n,p}\right) + \log \sum_{d=1}^{C} \exp\big(\max_{p}(X_{n,p} + W_{d,p})\big)\Big). \quad (2)$$

由于该目标函数包含多个 max(·) 项，每个类别对应一项，因此前文介绍的 SCT 优势在此尤为显著。在首次迭代进行初始 $\mathcal{O}(N)$计算后，后续迭代只需进行 $\mathcal{O}$(log N)次更新，因此在整个训练过程中，绝大多数计算的复杂度都得以降低。

为了充分发挥这一计算优势，我们现在介绍通过我们的非光滑架构进行微分所需的解析框架。

3.2. 次梯度计算

为处理模型的 max 结构所引发的非光滑算子，我们采用由（Bolte 和 Pauwels，2021年）提出的保守集值场框架。该框架提供了一种广义微分概念，在将经典微积分推广到非光滑函数的同时，保留了连贯的链式法则。允许此类场的函数称为路径可微函数，这一类别涵盖范围广泛，包括凸函数、凹函数、Clarke 正则函数以及半代数利普希茨连续函数。特别是，该框架支持最大值和 对数∑指数等算子的复合，使得反向传播计算能够严谨地推广至我们的非光滑架构。该框架构成了后续次梯度推导的分析基础。

为简化记号，我们定义临界索引。设

$$n^* \in \arg\max_n \Big(-\max_p \big(X_{n,p} + W_{y_n,p}\big) + \log \sum_{d=1}^{C} \exp\big(\max_p (X_{n,p} + W_{d,p})\big)\Big)$$

是分类最差样本之一的索引，

$$p^*(d) \in \arg\max_p \big(X_{n^*,p} + W_{d,p}\big)$$

对于给定类别 d 和分类最差样本 n^*，在特征上取最大值的项。并且 $d^* := y_{n^*}$。
使用此记号，损失相对于 $W_{i,j}$ 的次梯度变为

$$\frac{\partial \mathcal{L}}{\partial W_{i,j}}(W) = -\mathbb{I}(i = d^*)\mathbb{I}(j = p^*_{d^*}) + \hat{y}_{n^*,i}\,\mathbb{I}(j = p^*_i). \tag{3}$$

详细推导见附录 A.4。因此，稀疏次梯度矩阵至多包含 C 个非零元素，总共 $C \times P$个，每个类别对应一个元素$d \in \{1, \ldots, C\}$，位置为 (d, p^*_d)，其中包括对应真实类别的元素 d^*。

尽管这种表述简洁优雅，并突显了稀疏性的作用，但我们的实验发现，损失停滞在相对较高的数值。这一局限促使我们在下一节引入一种表达能力更强、带有隐藏层的架构。

3.3. 单隐藏层模型

现在，我们考虑一种表达能力更强的架构，该架构受线性最小–最大（LMM）网络启发（Luo & Fan, 2021年），并依次应用于每个输入样本。我们将首先介绍回归场景中的模型，这正是原始逼近定理所确立的内容；随后说明如何将其扩展到分类任务。

- **线性层：** 令 P表示输入特征的数量，$x_n = X_{n,:} = (X_{n,1}, \ldots\ldots, X_{n,P}) \in \mathbb{R}^P$表示第$n$个样本。我们应用稀疏线性变换

  $$\lambda(x) = [K_0 x_1, -K_1 x_1, \ldots, K_{2P-1} x_P, -K_{2P} x_P],$$

  其中，K 是一个参数向量。这也可以写作

  $$\lambda(x) = W^0 x \in \mathbb{R}^{2P}, \quad W^0 \in \mathbb{R}^{2P \times P},$$

  其中 W^0 是具有预定义稀疏模式的稀疏矩阵。

- **最小加层：** 令 $h = \{1, \ldots, H_1\}$ 表示该层中的隐藏神经元。则：

  $$g_h(x) = \min_{i \in \{1,\ldots,2P\}} \big(\lambda_i(x) + W^1_{i,h}\big),$$

  其中，$W^1 \in \mathbb{R}^{2P \times H_1}$是（min，+）权重矩阵。

- **Max-Plus感知机：** LMM 的输出为

  $$\hat{f}_{W^0,W^1,W^2}(x) = \max_{h \in \{1,\ldots,H_1\}} \big(g_h(x) + W^2_h\big).$$

  我们工作的一个关键理论基础是（Luo &Fan, 2021年）中的以下结果：

  定理 3.1（基于LMM网络的Lipschitz函数万能逼近）。设 f 是定义在紧致域 $\mathcal{X} \subset \mathbb{R}^P$ 上的任意 *Lipschitz*连续函数。那么，存在一个稀疏线性映射 W^0 以及权重矩阵序列 $(W^{1,m})_m$和 $(W^{2,m})_m$，使得相应的 *LMM* 网络序列$\hat{}$（$f_{W^0, W^{1,m}, W^{2,m}}$）$_{m\geq 1}$一致收敛于 f，当 $m \to \infty$ 时。也就是说，

  $$\lim_{m\to\infty} \sup_{x \in X} |\hat{f}_{W^0,W^{1,m},W^{2,m}}(x) - f(x)| = 0.$$

  该定理为我们的方法奠定了基础。具体而言，附录A.1 中的详细证明表明，变换 λ 可以选为稀疏形式，这一性质直接启发了我们的参数初始化策略。具体来说，W^0 仅取决于目标函数 f，而矩阵$W^{1,m}$和 $W^{2,m}$则随近似参数 m增大而扩展。对于每个 m，我们在 $\mathcal{X}$中选取若干点（x^1，……，x^m），每个隐藏神经元负责在 f 附近对 x^m进行插值。我们的初始化方法将在第 3.4节中正式介绍，其做法是选取少量样本，并按照针对这少量样本的插值结果来初始化神经网络参数。这为我们的模型设计提供了一个有原则且有效的起点。

  虽然定理 3.1为实值函数建立了近似结果，我们在此框架基础上构建用于向量值输出的 LMM 模型 $f: \mathbb{R}^P \to \mathbb{R}^C$。我们将单个 Max-Plus 感知机替换为一个 Max-Plus层和一个 softmax激活。

- **Max-Plus层与 Softmax：** 对于每个类别 $d \in \{1, \ldots, C\}$，类别得分为

  $$z_d(x) = \max_{h \in \{1,\ldots,H_1\}} \big(g_h(x) + W^2_{h,d}\big),$$

其中 $W^2 \in \mathbb{R}^{H_1 \times C}$，预测概率为

$$\hat{y}_d(x) \;=\; \frac{\exp\big(z_d(x)\big)}{\sum_{d'=1}^{C} \exp\big(z_{d'}(x)\big)}.$$

相应的近似保证以及 $(W^{1,\,m})_m$ 和 $(W^{2,\,m})_m$ 的显式构造见附录 A.2。

3.4. 初始化

令 $(x_n,\;\; y_n) \in \mathbb{R}^P \times \{1,\dots,C\}$ 表示一个带标签样本。我们将 LMM 网络初始化为由三层组成：一个权重矩阵为$W^0 \in \mathbb{R}^{2P\times P}$的稀疏线性变换 λ，接着是权重为 $W^1 \in \mathbb{R}^{2P\times H}$的最小加层，最后是由 $W^2 \in \mathbb{R}^{H\times C}$参数化的 Max-Plus层，其中 H 为隐藏神经元的数量。变换 λ 对每个特征的正向和负向进行编码，定义如下：

$$W^0_{2i-1,i} = k, \quad W^0_{2i,i} = -k, \quad \text{for } i = 1,\dots,P,$$

其中 W^0 的所有其他元素均为零。标量 k作为初始缩放参数，在训练过程中通过次梯度更新进行调整，实际上是一个可学习量。相应的变换后输入为 $\lambda(x_n) = W^0 x_n \in \mathbb{R}^{2P}$。

隐藏层权重 W^1 通过选取一部分 H 随机选择的训练样本 $\{x_{n_h}\}_{h=1}^{H}$进行初始化，从而确保插值：

$$W^1_{2i-1,h} = -k \cdot x_{n_h,i}, \quad W^1_{2i,h} = k \cdot x_{n_h,i}.$$

输出层权重 W^2 通过提升真实类别并惩罚其他类别来实现类别区分：

$$W^2_{h,d} = \begin{cases} k, & \text{if } d = y_{n_h}, \\ -k, & \text{otherwise.} \end{cases}$$

这种初始化将每个隐藏神经元与一个特定的训练样本关联起来，同时输出层会惩罚该样本对应的错误类别。

缩放参数 k 在塑造网络输出的几何形态和置信度方面起着核心作用。在我们的初始化中， k 控制近似中所用分段线性函数的斜率（如定理3.1的证明所示）。

3.5. 次梯度推导

定义训练目标：

$$\mathcal{L}(W) \;=\; \max_{1\le n\le N} \mathrm{Loss}_n(W) \;=\; \max_{1\le n\le N} \big(-\log \hat{y}_{y_n}(x_n)\big).$$

令 $n^\star \in \arg\max_n$ 损失$_n(W)$。

对于每个类别 d，选择

$$h_d^\star \in \arg\max_h \Big(g_h(x_{n^*}) + W^2_{h,d}\Big),$$

$$i_d^\star \in \arg\min_i \Big(\lambda_i(x_{n^*}) + W^1_{i,h_d^\star}\Big)$$

$\mathcal{L}$在 $W = (W^0, W^1, W^2)$处的一个次梯度可按如下方式求得：

(a) 关于 W^2 的次梯度：计算与（3）非常相似

$$\frac{\partial \mathcal{L}}{\partial W^2_{h,d}} \;=\; \mathbb{I}_{\{h=h_d^*\}}(\hat{y}_d(x_{n^*}) - \mathbb{I}_{\{d=y_{n^*}\}}) \,.$$

(b) 关于 W^1 的次梯度：

$$\frac{\partial \mathcal{L}}{\partial W^1_{i,h}} \;=\; \begin{cases} \dfrac{\partial \mathcal{L}}{\partial W^2_{h,d}}, & \text{if } h = h_d^\star \text{ and } i = i_d^\star, \\ 0, & \text{otherwise.} \end{cases}$$

(c) 关于 W^0 的次梯度：

$$\frac{\partial \mathcal{L}}{\partial W^0_{i,p}} \;=\; \begin{cases} \dfrac{\partial \mathcal{L}}{\partial W^2_{h^\star,d}} x_{n^\star,p_d}, & \text{if } i = i_d^\star \\ 0 & \text{otherwise.} \end{cases}$$

次梯度推导的详细证明见附录 A.5。

有趣的是，即使在用不同类型的神经元丰富该架构之后，次梯度仍然极其稀疏：每层最多包含 C 个非零元素，每个类别一个。

定理 3.2（次梯度的稀疏性）。 对于使用最大 *SCCE* 损失训练的任意 *LMM* 网络，每层关于参数矩阵的次梯度最多包含 C 个非零元素，其中 C 为类别数。

4. 优化算法

为了训练线性极小极大模型，我们将考虑两种算法：针对平均损失的随机梯度下降，以及针对最大损失的稀疏次梯度下降。

4.1. 针对平均损失的随机梯度下降

这是最自然的选择，并作为我们提出的另一种算法的基线。在每次迭代中，我们考虑单个样本，对该样本执行前向传播——这是每次迭代中计算量最大的部分——然后计算该样本损失的次梯度。请注意，得益于次梯度的稀疏性，这一反向传播的计算代价非常低。

4.2. 最大损失上的稀疏次梯度下降

在 Nesterov 关于非光滑优化次梯度方法的开创性工作基础上，我们通过整合三个关键要素将该框架扩展到我们的设定中：(i) 短计算树（SCT）结构，使最大值操作能够在对数时间内更新；(ii) Polyak 的自适应步长规则，可稳定非光滑优化中的收敛；(iii) 参数在稀疏初始化时的自然稀疏性，我们利用它来降低更新的代价。

形式化地，我们考虑以下问题

$$\min_{W=W^0,W^1,W^2} \mathcal{L}(W) = \min_{W=W^0,W^1,W^2} \max_{1\le n\le N} \mathrm{Loss}_n(W)$$

稀疏次梯度算法定义如下

$$\begin{aligned} &\text{Initialize } W_0 = (W_0^0, W_0^1, W_0^2) \\ &W_{k+1} = W_k - \alpha_k \mathcal{L}'(W_k), \quad k \ge 0, \end{aligned}$$

其中， $\mathcal{L}'(W_k)$ 表示属于某个保守场的稀疏次梯度，并且

$$\alpha_k = \frac{\mathcal{L}(W_k) - \mathcal{L}^*}{\|\mathcal{L}'(W_k)\|^2}.$$

这里， α_k 是 Polyak 步长，它会在每次迭代时根据当前次优性 $\mathcal{L}(x_k) - \mathcal{L}^*$ 和次梯度的平方范数自适应选取（Loizou等人，2021年），其中 $\mathcal{L}^* = \inf_W \mathcal{L}(W)$ 表示最优函数值。一般来说，我们不知道 $\mathcal{L}^*$的值，但在我们的设定中，目标是 $\mathcal{L}^* = 0$，对应于稀疏分类交叉熵损失的最小值。我们还在最后几次迭代中考虑使用一个较小的常数步长 $\alpha_k = \alpha$，以应对我们对α 的估计可能过于乐观的情况。 $\mathcal{L}^*$

初始化 W_0时，我们会执行一次完整的前向传播，并存储所有短计算树。这需要占用内存，但随后便可像（Nesterov，2014年）那样执行稀疏更新。我们可以利用数据集中的稀疏性来更新第一层（即 W^0 矩阵），并利用前向传播的稀疏性，仅更新其他层中发生变化的短计算树。

5. 实验

5.1. 多层感知机与 LMM 在鸢尾花数据集上的比较

我们在Iris数据集上评估所提出的线性极小极大（LMM）模型，并将其与标准多层感知机（MLP）进行比较。数据随机划分为70%的训练样本和30%的测试样本。为确保公平比较，两个模型采用相同的深度和相近的宽度。该架构由三层组成，分别包含（P， $2P$， H， C）个神经元，其中 $P = 4$ 是输入维度，$H = 20$ 是隐藏层宽度， $C = 3$ 是类别数。所有模型均训练50，000次迭代。

两种方法的训练目标函数有所不同。多层感知机通过使用学习率为 0.01 的 Adam优化器最小化平均交叉熵损失进行训练（表 $\frac{1}{N}$ 中的多层感知机（6））。LMM 模型则采用以下任一方式进行训练：使用随机梯度下降最小化平均损失（LMM_($\frac{1}{N}$)），或使用所提出的稀疏次梯度算法最小化最大损失（LMM（max$_n$））。

由于目标函数并不相同，我们同时报告平均损失和最大损失，它们是根据最终损失向量计算得出的。

表2。**训练**在 Iris数据集上的性能。

模型	多层感知机($\frac{1}{N}$)	LMM ($\frac{1}{N}$)	LMM (max$_n$)
最大损失	1.839	1.025	0.426
平均损失	0.025	0.16	0.245
准确率	99%	98.1%	100%

最终的 Max-SCCE 损失约为 0.426，这表明在最大损失准则下实现了完美分类（2.2），同时保持了中等置信度水平。相比之下，多层感知机以高度自信的预测为代价获得了较低的平均损失，因此其最大损失值显著更大，如表7所示。

表3. **在**Iris数据集上的测试性能。

模型	多层感知机($\frac{1}{N}$)	LMM ($\frac{1}{N}$)	LMM（最大值$_n$）
最大损失	**8.744**	1.253	1.03
平均损失	0.047	0.227	0.311
准确率	88%	91.11%	93.33%

总体而言，结果揭示了这两种模型之间明显的定性差异。多层感知机虽然取得了有竞争力的平均损失值，但代价是严重的过度自信，这体现在其较大的最大损失上。相比之下，LMM 保持了受控的置信度，因此最大损失值较低。这种受控表达能力在实践中是一项理想的特性，尤其是在重视鲁棒性和可解释性的场景中。

最后，将隐藏神经元数量增加到与训练样本数量相同（例如，Iris 数据集中的 $H = 150$）即可实现精确插值，最大损失收敛至零。这一结果凸显了 LMM 模型在稀疏训练下的表达能力，且无需依赖密集梯度更新。

5.2. 权重初始化对鸢尾花数据集的影响

鉴于形态网络对权重初始化的敏感性（Dimitrova 等人，2025年），我们分析不同初始化策略如何影响 LMM 模型的训练行为和最终性能。我们针对权重矩阵 $(W^0,\ W^1,\ W^2)$考虑三种初始化策略：(i) 受第 3.4 节中 LMM 模型理论构造启发的结构化初始化；(ii) 从 $\mathcal{N}(0,\ k^2)$中抽取的高斯随机初始化；(iii) 从 $\mathcal{U}(-k,\ k)$中抽取的均匀随机初始化。除初始化方案外，所有实验条件均保持一致。为考虑随机初始化带来的变异，每种配置都使用不同的随机种子独立运行 10 次。下图展示了各次运行所得最终 Max-SCCE 值的分布。

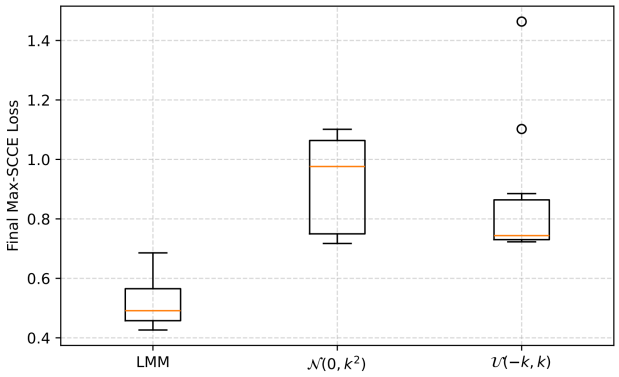

图 *1.* 在鸢尾花数据集上采用三种不同权重初始化策略，进行 10 次独立运行后得到的最终 Max-SCCE 值。

结果显示，结构化初始化与随机初始化方法之间存在明显差距。基于 LMM 的初始化始终能收敛到更低的最终 Max-SCCE 值，并且多次运行间的波动显著更小。值得注意的是，即使是结构化初始化中表现最差的一次运行，其最终损失也低于采用高斯随机初始化或均匀随机初始化时的最佳结果。虽然鸢尾花数据集是一个小规模基准，但这些发现清楚地表明，理论驱动的初始化在引导稀疏次梯度优化进入参数空间中的有利区域方面发挥着决定性作用。

5.3. 在 MNIST 上训练 LMM

为评估可扩展性，我们将采用所提出的稀疏次梯度算法的 LMM 模型扩展到 MNIST 数据集。

实验设置。所有 MNIST 实验均在使用 30 个处理器的 CPU 集群上进行。所有变体均在相同配置下评估，使用 $n = 60,\ 000$ 个训练样本、 $H = 500$ 个隐藏神经元，更新的跳过率为 100 次迭代 W^0，以及 $k = 4$。图2展示了 LMM 模型在 200,000 次迭代中的 max-SCCE 损失收敛情况。

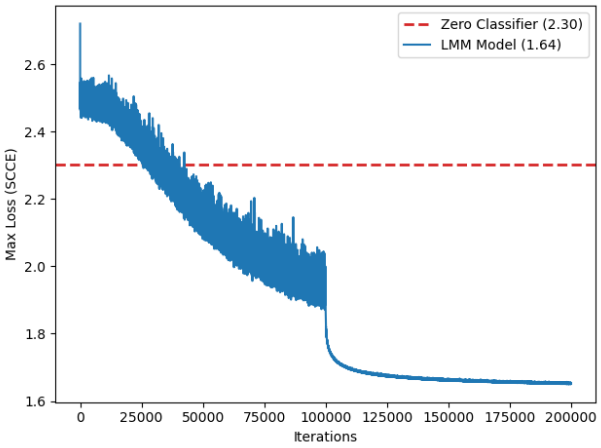

图 *2.* LMM 模型在 MNIST 上的收敛情况

我们在训练过程中采用两阶段步长策略。在前 100,000 次迭代中，我们使用 Polyak 步长。众所周知，当距离最优解较远时，该步长能够采取更大的自适应步长，并避免在非光滑优化景观中陷入平坦的局部极小值。

在第二阶段（从第 100,000 次迭代开始），我们切换为固定步长 $\alpha = \frac{\varepsilon}{2\sqrt{2C}}$，其中 ε 是用户定义的学习率，C 是输出类别数。这个固定学习率使收敛曲线更加平滑，并进一步将 Max-SCCE 损失降至约 1.64。该数值显著优于零分类器基线损失 $\log(C) \approx 2.30$ （红色虚线），后者对应于对各类别进行均匀预测。

定量训练和测试结果（包括最大损失、平均损失和分类准确率）见附录 A.7。这些结果凸显了优化行为的明显差异：最小化最大损失可显著降低最坏情况损失，同时提高训练集和测试集上的分类准确率。这证实，对于大规模数据集上的 LMM，最大损失最小化是比平均损失最小化更有效的训练目标。

为了更好地理解模型预测的性质，我们分析了表4中 MNIST 测试集上的混淆矩阵。所得宏平均 F1 分数达到0.89，表明分类性能均衡，尽管

t模型对最小化最坏情况误差的关注 rs.

表 *4*。 LMM 模型在 MNIST 测试集上的混淆矩阵（88。6%准确率）。

	0	1	2	3	4	5	6	7	8	9
0	922	0	8	5	3	26	5	2	5	4
1	0	1089	12	2	8	2	3	4	14	1
2	9	1	923	30	11	44	21	25	4	1
3	7	3	31	849	0	64	0	9	30	17
4	4	2	3	4	820	1	5	8	14	121
5	8	0	2	42	3	784	14	8	20	11
6	10	5	3	0	28	31	868	1	9	3
7	2	4	21	16	6	2	0	930	4	43
8	5	7	13	32	21	23	8	9	822	34
9	8	7	6	11	25	15	2	50	28	857

综合来看，这些结果突显了 LMM 架构的两大优势：(i) 能够在保持高准确率的同时，高效优化非光滑最大损失目标；(ii) 倾向于生成适度且分布均匀的置信度分数，而不是过度自信的预测（A.6）。在鲁棒性和不确定性感知与实现高准确率同等重要的应用中，这一特性尤其理想。

5.4. 稀疏更新与密集更新：每次迭代的计算成本

所提出训练流程的一个核心动机，是利用次梯度更新的稀疏性来降低优化 LMM 模型的计算成本。为量化这一设计选择的实际影响，我们比较了三种更新策略下每次训练迭代的平均耗时：(i) 稀疏更新；(ii) 跳过输入层参数 W^0 更新的稀疏更新；以及 (iii) 稠密更新。

为关注运行时间而非最终收敛情况，每种方法均运行 300 次迭代，并报告每次迭代的平均耗时（单位为秒/次迭代）。对于稀疏方法和周期性跳过 W^0 的稀疏方法变体，可以进行更长时间的运行，且每次迭代的耗时保持稳定。相比之下，将稠密更新策略延长至数百次迭代以上会产生高昂的计算成本，因此，为公平比较每次迭代的复杂度，我们将稠密基线的迭代预算限制为相同的 300 次。

*Ta*表 *5.* M 上每次迭代的平均计算时间 NIST。

更新模式	**时间（秒）**	**秒/次迭代**	**内存（GB）**
稀疏 + 跳过 W^0	36	0.12	513
稀疏	1045	3.48	513
稠密	5688	18.96	10

表5 中的结果表明，利用稀疏性能够显著提升计算效率。跳过部分 W^0 更新可将每次迭代的平均耗时从 3.48 秒降至 0.12 秒，相较于标准稀疏更新，速度约提升 29× 倍。即使不跳过 W^0，稀疏更新相较于稠密更新仍有显著优势，每次迭代的成本降低约 5.5×倍。

值得注意的是，我们发现，定期跳过对 W^0 的更新并不会导致预测性能下降。这表明，稀疏变体中的大量计算开销来自维护和更新与输入层相关的量，而有选择地冻结这些参数可以显著提高效率，同时不影响准确率。

5.5. 局限性与挑战

尽管我们已证明 LMM 模型可训练且具有良好的特性，但仍存在若干局限。首先，其训练时间显著长于标准多层感知机。例如，在 Iris数据集上，训练 LMM 模型需要 151 秒，而在前文所述的实验条件下，训练多层感知机仅需 33 秒。这种差距在意料之中：PyTorch 和 TensorFlow 等广泛使用的深度学习框架已历经多年优化，而我们的实现还只是一个初始原型。因此，提高计算效率并实现 GPU 加速是未来工作的重要方向。

其次，由于需要显式表示短计算树（SCT），LMM 模型的内存需求更高。我们计划研究随机替代方案，以减轻这一影响。主要挑战在于找到偏置较低且更新成本较低的最大损失估计值。

5.6. 结论

我们的实验表明，LMM 模型可以利用稀疏次梯度算法进行有效训练，在理论可靠性的同时展现出强劲的实证性能。在 MNIST 上，该模型在优化最坏情况损失的同时，实现了 92% 的类别分类准确率。这证实了该训练流程从优化角度和机器学习角度来看都行之有效：前者表现为目标函数持续下降，后者则体现为具有竞争力的预测准确率。与标准多层感知机相比，LMM网络表现出较低的过度自信，这一点可以从预测置信度的分布中得到佐证。可解释性、鲁棒性与审慎预测的结合，凸显了其在安全关键应用中的潜力，尤其是在医疗领域，因为过度自信所导致的错误代价是不可接受的。

参考文献

Bolte, J. 和 Pauwels, E. 保守集值场、自动微分、随机梯度方法与深度学习。*MathematicalProgramming*，188(1)：19–51，2021年。

Daniely, A.、Srebro, N.与Vardi, G. 学习神经网络的计算复杂度：光滑性与退化性。神经信息处理系统进展，36:76272–76297，2023年。

Dimitriadis, N.与Maragos, P. 形态学神经网络的进展：训练、剪枝与施加形状约束。*ICASSP 2021 - 2021年IEEE声学、语音与信号处理国际会议 (ICASSP)*，第3825–3829页，2021年。

Dimitrova, M.、Blusseau, S.与Velasco-Forero, S. 学习图像变换的形态学表示：初始化与层的可微性的影响。2025年。Glorot, X.与Bengio, Y. 理解训练深度前馈神经网络的困难。载于第十三届人工智能与统计国际会议论文集，第249–256页。JMLR研讨会与会议论文集，2010年。

LeCun, Y., Bottou, L., Bengio, Y. 和 Haffner, P. 将基于梯度的学习应用于文档识别。*IEEE* 会议录，86(11):2278–2324，2002年。

LeCun, Y., Bengio, Y. 和 Hinton, G. 深度学习。自然，521(7553):436–444，2015年。

Livni, R., Shalev-Shwartz, S. 和 Shamir, O. 训练神经网络的计算效率。载于 Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. 和 Weinberger, K.（编），神经信息处理系统进展，第 27 卷。Curran Associates, Inc.，2014年。Loizou, N., Vaswani, S., Laradji, I.H. 和 Lacoste-Julien, S. 用于随机梯度下降的随机Polyak步长：一种用于快速收敛的自适应学习率。载于人工智能与统计国际会议，第 1306–1314 页。PMLR，2021年。

Luo, Y. 和 Fan, S. 最小-最大-加法神经网络。预印本 *arXiv:2102.06358*，2021年。

Mondal, R., Mukherjee, S. S., Santra, S. 和 Chanda, B. 形态学网络：形态学神经元还能走多远？载于英国机器视觉会议，2019年。

Nesterov, Y. 大规模优化问题的次梯度方法。《数学规划》，146(1-2)：275–297，2014年。Sra, S., Nowozin, S., and Wright, S. J.（编）。《机器学习优化》。麻省理工学院出版社，马萨诸塞州剑桥，2012年。Tsiamis, A. and Maragos, P. 最大加代数与系统中的稀疏性。《离散事件动态系统》，29(1)：163–189，2019年。Zhang, Y., Blusseau, S., Velasco-Forero, S., Bloch, I., and Angulo, J. 最大加算子在神经网络滤波器选择和模型剪枝中的应用。2019年。

A. 附录

A.1. 定理 3.1 的证明：显式构造参数序列

证明。1. **由于 f 在 $\|\cdot\|_\infty$下是 K–利普希茨的，因此我们有**

$$\forall x, y \in X, \qquad |f(x) - f(y)| \le K\|x - y\|_\infty.$$

2. **定义线性变换** $\lambda(x)$ 如下

$$\lambda(x) = [Kx_1, -Kx_1, \ldots, Kx_p, -Kx_p] \in \mathbb{R}^{2p},$$

即,

$$\lambda : \mathbb{R}^p \to \mathbb{R}^{2p}.$$

利用两侧（$\pm Kx_j$），我们确保下一层的（min,+）能够构造一个以任意点为中心的金字塔，这对于逼近利普希茨函数至关重要。

3. **定义**$(\min\ +)$ **函数** $g_i(x)$**，每个神经元** i对应一个，如下所示

$$g_i(x) = \min_{j=1}^{2p} \big(\lambda_j(x) + a_{ij}\big),$$

其中 $a_i \in \mathbb{R}^{2p}$ 是一个可学习偏置向量（每个神经元对应一个）。

每个 g_i 都是一个凹分段线性函数。只要正确设置 a_{ij}，它就会在某个点 x^i处达到峰值。这样，g_i 就在选定的网格点 x^i处定义了一个金字塔函数。

4. **在网格点处构造金字塔：**

令 f： $\mathbb{R}^p \to \mathbb{R}$ 为目标利普希茨函数。定义一个网格点集合

$$\mathcal{M} = \{x^1, x^2, \ldots, x^m\} \subset \mathbb{R}^p,$$

其中我们已知 $f(x^i)$的值。

网格间距记为 $\delta > 0$，并令 $\delta \to 0$。

我们假设每个 $x \in X$ 都接近某个网格点 x^i。对于每个网格点 x^i，我们希望构造一个（$\min\ +$）函数 $g_i(x)$，使得 $g_i(x) = \min_{j=1}^{2d}(\lambda_j(x) + a_{ij})$，并满足

$$g_i(x^i) = f(x^i).$$

这确保金字塔在其中心处与函数 f 相接触。

(a) 固定网格点 $x^i \in \mathbb{R}^p$时偏置 $a_i \in \mathbb{R}^{2p}$ 的构造

我们将偏置定义为

$$a_{ij} := -\lambda_j(x^i) + f(x^i).$$

采用这一选择后,

$$g_i(x) = \min_{j=1}^{2p} \big(\lambda_j(x) + a_{ij}\big) = \min_{j=1}^{2p} \big(\lambda_j(x) - \lambda_j(x^i) + f(x^i)\big).$$

特别地，在 $x = x^i$ 处我们得到

$$g_i(x^i) = f(x^i).$$

因此，a_{ij} 的这个定义确保金字塔函数 $g_i(x)$ 在网格点 x^i 处与 $f(x)$ 相切，且其值恰好等于 $f(x^i)$。

由于 $\lambda(x) = [Kx_1, -Kx_1, \ldots, Kx_d, -Kx_d]$，我们有

$$\lambda_j(x) = Kx_j, \quad j = 1, \ldots, p, \qquad \lambda_j(x) = -Kx_{j-p}, \quad j = p+1, \ldots, 2p.$$

因此

$$a_{ij} = -Kx_j^i + f(x^i), \quad j = 1, \ldots, p, \qquad a_{ij} = Kx_{j-d}^i + f(x^i), \quad j = p+1, \ldots, 2p.$$

代入 $g_i(x)$得到

$$g_i(x) = \min_{j=1}^{2p} \big(\lambda_j(x) + a_{ij}\big) = \min\Big\{ \min_{j=1,\ldots,p} \big(K(x_j - x_j^i) + f(x^i)\big),\ \min_{j=p+1,\ldots,2p} \big(-K(x_{j-p} - x_{j-p}^i) + f(x^i)\big)\Big\}.$$

因此,

$$g_i(x) = f(x^i) - K\|x - x^i\|_\infty.$$

这表明 $g_i(x)$是一个帐篷（或金字塔）函数，即一个翻转并向上平移的绝对值锥 :
i. 它在 x^i处有一个峰值，ii. 它达到高度
$f(x^i)$，iii. 它在两侧以斜率 $\pm K$下降。

在每个 $x^i \in \mathcal{M}$ 处，我们构建一个金字塔 $g_i(x)$，使得 $g_i(x^i) = f(x^i)$。这种构造迫使近似在网格点处与 f 的真实值相匹配。

5. **构造最终近似函数** $h(x)$

我们构造了一族帐篷（金字塔）函数 $g_i(x)$，每个函数都以一个网格点 $x^i \in \mathcal{M}$为中心，并满足

$$g_i(x^i) = f(x^i).$$

现在，我们将近似函数 $h(x)$定义为所有这些金字塔函数的最大值:

$$h(x) = \max_{j=1,\ldots,2p} g_j(x).$$

我们取最大值，是因为每个金字塔函数 $g_i(x)$ 都是局部化的：它只在中心 x^i附近才能很好地近似 f。对所有金字塔函数取最大值，就能结合它们各自的优势。

- 每个点 x 都靠近某个网格中心 x^i。
- 最近的金字塔函数能为 $f(x)$提供最佳的局部估计。
- 最大值确保 $h(x)$在每个点都选取最高（最佳）的局部近似。

在网格点 x^i**处:**

$$h(x^i) = \max_{j=1,\ldots,2p} g_j(x^i) = g_i(x^i) = f(x^i), \qquad \text{for } j = i, \tag{1}$$

and

$$g_j(x^i) \le f(x^i), \qquad \forall j \ne i. \tag{2}$$

验证。 由于 f 是 K–利普希茨的,

$$|f(x^j) - f(x^i)| \le K\|x^j - x^i\|_\infty.$$

特别地,

$$f(x^j) \le f(x^i) + K\|x^j - x^i\|_\infty.$$

根据构造,

$$g_j(x^i) = -K\|x^j - x^i\|_\infty + f(x^j).$$

与 (1) 结合可得

$$g_j(x^i) \le -K\|x^j - x^i\|_\infty + f(x^i) + K\|x^j - x^i\|_\infty = f(x^i).$$

因此，对所有 $j \ne i$， $g_j(x^i) \le f(x^i)$。

范数之间的关系。对于所有 $v \in \mathbb{R}^n$,

$$\|v\|_\infty \ \le \ \|v\|_1 \ \le \ n\|v\|_\infty.$$

因此，如果 f 在 $\|\cdot\|_\infty$下是利普希茨连续的，那么它在 $\|\cdot\|_1$ 下也同样是利普希茨连续的（反之亦然），其常数相差的因子仅取决于 n。
因此，所有金字塔函数的上包络满足

$$h(x^i) = \max_j g_j(x^i) = g_i(x^i) = f(x^i), \quad \forall x^i \in \mathcal{X}.$$

这意味着，近似 $h(x)$ 在每个网格点上都与函数 $f(x)$ 完全匹配 .

6. **网格点之外的近似误差**
我们知道，近似 $h(x)$ 在网格点上与函数 $f(x)$ 完全匹配。现在我们要证明，即使在网格点之外，我们的近似 $h(x)$ 仍然与真实的 $f(x)$ 非常接近。

步骤 6.1：利普希茨控制。由于 f 是 K–利普希茨的，对于任意 x, $x^i \in X$ 我们有

$$|f(x^i) - f(x)| \le K\|x^i - x\|_\infty.$$

步骤 6.2：网格点附近的误差界。固定 $x \in X$，令 x^i 为距离 x 最近的网格点，则

$$\|x - x^i\|_\infty \le \delta.$$

考虑以 x^i 为中心的金字塔函数:

$$g(x) = -K\|x - x^i\|_\infty + f(x^i).$$

由于 $h(x) = \max_j g_j(x)$，我们有

$$h(x) \ge g(x).$$

因此,

$$h(x) - f(x) \ \ge \ g(x) - f(x) = f(x^i) - K\|x - x^i\|_\infty - f(x).$$

利用利普希茨不等式（A），

$$f(x^i) - f(x) \ \ge \ -K\|x^i - x\|_\infty,$$

因此

$$h(x) - f(x) \ \ge \ -K\|x - x^i\|_\infty - K\|x - x^i\|_\infty = -2K\|x - x^i\|_\infty \ge -2K\delta.$$

步骤 6.3：上界 界。 我们还知道 $h(x) \le f(x)$，因为根据构造，每个金字塔都位于 f 下方。因此 s,

$$-2K\delta \ \le \ h(x) - f(x) \ \le \ 0.$$

等价地,

$$|h(x) - f(x)| \le 2K\delta.$$

结论。 近似 $h(x)$ 在网格点上与 $f(x)$ 完全相等，而在其他所有位置与 $f(x)$ 的差异至多为 $2K\delta$。随着网格间距 $\delta \to 0$，我们得到

$$\sup_{x \in X} |h(x) - f(x)| \to 0,$$

这就证明了一致收敛。

□

A.2. 分类扩展

在.....的详细证明中　　orem 3.1 该构造针对标量值函数给出 $f\colon\ \mathbb{R}^p \to \mathbb{R}$。其中，每个金字塔函数定义为

$$g_i(x) = \min_j \big(\lambda_j(x) + a_{ij}\big), \qquad h(x) = \max_i g_i(x),$$

这为 f 提供了标量近似。

然而，在我们的实验设置中，我们处理的是向量值函数 $f\colon\ \mathbb{R}^p \to \mathbb{R}^C$，对应于多分类问题中的 C 输出类别。直接扩展该证明需要构建 $C \times m$ 个金字塔，每个类别和每个网格点各一个，之后再添加一个输出层。随着 C（类别数）和m（样本数）的增加，这很快就会变得难以承受。

相反，我们采用一种受该证明启发、但专为分类量身定制的高效策略。我们首先构建一组由 m 个共享金字塔组成的 g_j（x），与标量情形完全相同，每个金字塔都以一个网格点 x^j为中心。在一个真实标签为 y_i的训练点 x^i 处，这些金字塔满足

$$g_j(x^i) = \mathbf{1}_{ij},$$

从而每个金字塔函数都编码其中心。

为了获得类别特定分数，我们重用这些共享金字塔函数，但施加由置信度参数 $k > 0$ 控制的简单常数偏移。对于每个类别 $d \in \{1, \ldots, d\}$，我们定义

$$h^d(x) = \max_j \Big(g_j(x) + k\,(2 \cdot \mathbf{1}_{\{d=y_j\}} - 1)\Big),$$

这意味着，如果 d 是样本 x^j 的真实类别，则分数会增加 $+k$；而如果 $d \neq y_j$，分数则会降低 $-k$。

在网格点 x^i 处，这种构造保证

$$h^{y_i}(x^i) \geq k, \qquad h^d(x^i) \leq -k \quad \text{for all } d \neq y_i,$$

因此，预测类别恰好是真实标签：$\arg\max_d h^d(x^i) = y_i$。

这一修改至关重要：我们无需为每个 C 个类别分别复制 m 个金字塔函数（总计 $C \times m$ 个神经元），只需使用 m 个共享金字塔函数，再加上 C 个类别特定偏移。因此，最终复杂度为 $C + m$，大幅降低，使该方法具备可扩展性，同时保留了证明的构造性思路。具体而言，实验模型继承了理论构造的几何直觉，即以样本为中心的金字塔函数，但实现这一点所需的运算要少得多。

A.3. 命题的证明

证明。 设 $\{(x_n, y_n)\}_{n=1}^N$ 是一个带标签数据集，其中 $y_n \in \{1, \ldots, C\}$，并假设模型对每个类别 d都生成 logit $Z_d(x_n)$。softmax 概率为

$$\hat{Y}_d(x_n) = \frac{\exp(Z_d(x_n))}{\sum_{d'=1}^C \exp(Z_{d'}(x_n))},$$

$$\mathcal{L}(x_n, y_n) = -\log \hat{Y}_{y_n}(x_n)$$

是稀疏分类交叉熵损失。

反设模型未达到完美分类准确率。那么，存在一个样本 x_n和一个错误类别 $d' \neq y_n$，使得 $Z_{d'}(x_n) \geq Z_{y_n}(x_n)$，因此 $e^{Z_{d'}(x_n)} \geq e^{Z_{y_n}(x_n)}$。

因此，softmax 分母满足：

$$\sum_{d=1}^C e^{Z_d(x_n)} \geq e^{Z_{y_n}(x_n)} + e^{Z_{d'}(x_n)} \geq 2e^{Z_{y_n}(x_n)}.$$

因此，真实类别的预测概率为：

$$\hat{Y}_{y_n}(x_n) = \frac{e^{Z_{y_n}(x_n)}}{\sum_{d=1}^{C} e^{Z_d(x_n)}} \leq \frac{e^{Z_{y_n}(x_n)}}{2e^{Z_{y_n}(x_n)}} = \frac{1}{2}.$$

取负对数得：

$$\mathcal{L}(x_n, y_n) = -\log \hat{Y}_{y_n}(x_n) \geq \log 2.$$

这与 $\max_n \mathcal{L} < \log 2$ 矛盾。
因此，对于每个训练样本，我们必须有

$$\arg\max_d Z_d(x_n) = y_n,$$

这意味着模型能够正确预测每个训练标签，即训练准确率为 100%。 □

A.4. 0 隐藏层的次梯度公式证明

证明。 稀疏分类交叉熵损失定义为：

$$\text{Loss}_n(W) = -\log\left(\hat{Y}_{n,y_n}\right)$$

ˆ其中代入$Y_{n,\ y_n}$的公式（1）后，得到：

$$\text{Loss}_n(W) = -\log\left(\frac{\exp\Big(\max_p\left(X_{n,p} + W_{d,p}\right)\Big)}{\sum_{d=0}^{9}\exp\Big(\max_p\left(X_{n,p} + W_{d,p}\right)\Big)}\right)$$

我们希望计算一个次梯度

$$\mathcal{L}(W) = \max_n \text{Loss}_n(W) = \max_n\left(-\max_p(X_{n,p} + W_{y_n,p}) + \log\sum_{d=0}^{9}\exp\left(\max_p(X_{n,p} + W_{d,p})\right)\right)$$

记

$$n^* = \arg\max_n\left(-\max_p(X_{n,p} + W_{y_n,p}) + \log\sum_{d=0}^{9}\exp\left(\max_p(X_{n,p} + W_{d,p})\right)\right),$$

$$p^*(n,d) = \arg\max_p X_{n,p} + W_{d,p}$$

且当 e 为真时， $\mathbb{I}(e)$
$= \begin{cases} 1 \\ 0 \end{cases}$；当 e 为假时。

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W_{i,j}}(W) &= \frac{\partial}{\partial W_{i,j}}\left[\max_n\left(-\max_p(X_{n,p}+X_{y_n,p})+\log\sum_{d=0}^{9}\exp\left(\max_p(X_{n,p}+W_{d,p})\right)\right)\right] \\
&= \frac{\partial}{\partial W_{i,j}}\left[-\max_p\left(X_{n^*,p}+W_{y_{n^*},p}\right)\right]+\frac{\partial}{\partial W_{i,j}}\left[\log\sum_{d=0}^{9}\exp\left(\max_p X_{n^*,p}+W_{d,p}\right)\right] \\
&= -\frac{\partial}{\partial W_{i,j}}\left[X_{n^*,p^*(n^*,y_{n^*})}+W_{y_{n^*},p^*(n^*,y_{n^*})}\right]+ \\
&\quad \frac{\sum_{d'=0}^{9}\frac{\partial}{\partial x_{i,j}}\left[\max_p(X_{n^*,p}+W_{d',p})\right]*\exp\left(X_{n^*,p^*(n^*,d')}+W_{d',p^*(n^*,d')}\right)}{\sum_{d=0}^{9}\exp\left(X_{n^*,p^*(n^*,d)}+W_{d,p^*(n^*,d)}\right)} \\
&= -\mathbb{I}(i=y_{n^*})\mathbb{I}(j=p^*(n^*,y_{n^*}))+ \\
&\quad \frac{\sum_{d'=0}^{9}\frac{\partial}{\partial W_{i,j}}\left[X_{n^*,p(n^*,d')}+W_{d',p(n^*,d')}\right]*\exp\left(X_{n^*,p^*(n^*,d')}+W_{d',p^*(n^*,d')}\right)}{\sum_{d=0}^{9}\exp\left(X_{n^*,p^*(n^*,d)}+W_{d,p^*(n^*,d)}\right)} \\
&= -\mathbb{I}(i=y_{n^*})\mathbb{I}(j=p^*(n^*,y_{n^*}))+ \\
&\quad \frac{\sum_{d'=0}^{9}\mathbb{I}(i=d')\mathbb{I}(j=p^*(n^*,d'))*\exp\left(X_{n^*,p^*(n^*,d')}+W_{d',p^*(n^*,d')}\right)}{\sum_{d=0}^{9}\exp\left(X_{n^*,p^*(n^*,d)}+W_{d,p^*(n^*,d)}\right)} \\
&= -\mathbb{I}(i=y_{n^*})\mathbb{I}(j=p^*(n^*,y_{n^*}))+ \\
&\quad \frac{\exp\left(X_{n^*,p^*(n^*,i)}+W_{i,p^*(n^*,i)}\right)\mathbb{I}(j=p^*(n^*,i))}{\sum_{d=0}^{9}\exp\left(X_{n^*,p^*(n^*,d)}+W_{d,p^*(n^*,d)}\right)}
\end{aligned}
$$

总之，稀疏次梯度矩阵将有 10 个非零元素：对于每个 $i\in\{0,\dots,9\}$，元素 $(i,p^*(n^*,i))$ 和元素 $(y_{n^*},p^*(n^*,y_{n^*}))$，后者已在前面的列表中。 □

A.5. LMM 模型的次梯度公式证明

证明。 设 $\text{Loss}_n(W)=-\log(\hat{Y}_{y_n}(x_n))$ 为稀疏分类交叉熵损失，其中预测概率为

$$
\hat{Y}_d(x)=\frac{\exp(Z_d(x))}{\sum_{d'=1}^{C}\exp(Z_{d'}(x))}
$$

and

$$
Z_d(x)=\max_h\left(g_h(x)+W^2_{h,d}\right),\quad g_h(x)=\min_i\left(\lambda_i(x)+W^1_{i,h}\right),\quad \lambda(x)=W^0x.
$$

我们定义训练目标:

$$
\mathcal{L}(W)=\max_n\text{Loss}_n(W)=\max_n\left(-\log\hat{Y}_{y_n}(x_n)\right).
$$

设 $n^\star\in\arg\max_n\text{Loss}_n(W)$ 为一个最坏情况样本。我们定义:

$$
h^\star_d\in\arg\max_h\left(g_h(x_{n^\star})+W^2_{h,d}\right),\quad i^\star_d\in\arg\min_i\left(\lambda_i(x_{n^\star})+W^1_{i,h^\star_d}\right),
$$

并使用简写 $\mathbb{I}(e)=1$若 e 为真，否则为 0。

现在我们将损失函数完全展开:

$$
\text{Loss}_{n^\star}(W)=-\log\left(\frac{\exp(Z_{y_{n^\star}}(x_{n^\star}))}{\sum_{d=1}^{C}\exp(Z_d(x_{n^\star}))}\right)=-Z_{y_{n^\star}}(x_{n^\star})+\log\sum_{d=1}^{C}\exp(Z_d(x_{n^\star})).
$$

So:

$$\mathcal{L}(W) = -Z_{y_{n^\star}}(x_{n^\star}) + \log \sum_{d=1}^{C} \exp(Z_d(x_{n^\star})).$$

(a) 关于 W^2 的次梯度：我们计算：

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial W^2_{h,d}} &= \frac{\partial}{\partial W^2_{h,d}} \left[-Z_{y_{n^\star}}(x_{n^\star}) + \log \sum_{d'=1}^{C} \exp(Z_{d'}(x_{n^\star})) \right] \\
&= -\frac{\partial Z_{y_{n^\star}}(x_{n^\star})}{\partial W^2_{h,d}} + \frac{\partial}{\partial W^2_{h,d}} \left[\log \sum_{d'=1}^{C} \exp(Z_{d'}(x_{n^\star})) \right] \\
&= -\mathbb{I}(d = y_{n^\star}) \cdot \mathbb{I}(h = h^\star_d) \quad + \quad \frac{\exp(Z_d(x_{n^\star}))}{\sum_{d'} \exp(Z_{d'}(x_{n^\star}))} \cdot \mathbb{I}(h = h^\star_d) \\
&= \mathbb{I}(h = h^\star_d) \cdot \left(\hat{Y}_d(x_{n^\star}) - \mathbb{I}(d = y_{n^\star}) \right)
\end{aligned}$$

因此，

$$\frac{\partial \mathcal{L}}{\partial W^2_{h,d}} = \mathbb{I}(h = h^\star_d) \cdot \left(\hat{Y}_d(x_{n^\star}) - \mathbb{I}(d = y_{n^\star}) \right)$$

(b) 关于 W^1的次梯度：我们现在应用链式法则，经由

$$g_h(x) = \min_i \big(\lambda_i(x) + W^1_{i,h} \big).$$

我们计算：

$$\frac{\partial \mathcal{L}}{\partial W^1_{i,h}} = \sum_{d=1}^{C} \frac{\partial \mathcal{L}}{\partial Z_d(x_{n^\star})} \cdot \frac{\partial Z_d(x_{n^\star})}{\partial g_h(x_{n^\star})} \cdot \frac{\partial g_h(x_{n^\star})}{\partial W^1_{i,h}}.$$

首先，利用softmax交叉熵表达式，我们有

$$\frac{\partial \mathcal{L}}{\partial Z_d(x_{n^\star})} = \hat{Y}_d(x_{n^\star}) - \mathbb{I}_{\{d = y_{n^\star}\}}.$$

其次，由定义 $Z_d(x) = \max_h (g_h(x) + W^2_{h,\,d})$，我们得到

$$\frac{\partial Z_d(x_{n^\star})}{\partial g_h(x_{n^\star})} = \mathbb{I}_{\{h = h^\star_d\}}.$$

第三，由 $g_h(x) = \min_i (\lambda_i(x) + W^1_{i,\,h})$，我们有

$$\frac{\partial g_h(x_{n^\star})}{\partial W^1_{i,h}} = \mathbb{I}_{\{i = i^\star_d\}}.$$

结合这三个表达式可得

$$\frac{\partial \mathcal{L}}{\partial W^1_{i,h}} = \sum_{d=1}^{C} \left[\left(\hat{Y}_d(x_{n^\star}) - \mathbb{I}_{\{d = y_{n^\star}\}} \right) \mathbb{I}_{\{h = h^\star_d\}} \mathbb{I}_{\{i = i^\star_d\}} \right].$$

等价地，

$$\frac{\partial \mathcal{L}}{\partial W^1[i,h]} = \begin{cases} \hat{Y}_d(x_{n^\star}) - \mathbb{I}_{\{d = y_{n^\star}\}} & \text{for some } d \text{ with } h = h^\star_d \text{ and } i = i^\star_d, \\ 0, & \text{otherwise.} \end{cases}$$

特别地，由于 $\frac{\partial \mathcal{L}}{\partial W^2[h,d]} = \mathbb{I}_{\{h=h_d^\star\}}(\hat{Y}_d(x_{n^\star}) - \mathbb{I}_{\{d=y_{n^\star}\}})$，我们也可以写成

$$\frac{\partial \mathcal{L}}{\partial W^1[i,h]} = \sum_{d=1}^{C} \mathbb{I}_{\{h=h_d^\star\}} \mathbb{I}_{\{i=i_d^\star\}} \frac{\partial \mathcal{L}}{\partial W^2[h,d]}$$

(c) 关于 W^0 的次梯度：我们应用完整的链式法则：

$$\frac{\partial \mathcal{L}}{\partial W^0[i,p]} = \sum_{d=1}^{C} \frac{\partial \mathcal{L}}{\partial Z_d(x_{n^\star})} \cdot \frac{\partial Z_d(x_{n^\star})}{\partial g_{h_d^\star}(x_{n^\star})} \cdot \frac{\partial g_{h_d^\star}(x_{n^\star})}{\partial \lambda_i(x_{n^\star})} \cdot \frac{\partial \lambda_i(x_{n^\star})}{\partial W^0[i,p]}$$

我们知道：

$$\frac{\partial \mathcal{L}}{\partial Z_d(x)} = \frac{\partial \mathcal{L}}{\partial W^2[h_d^\star, d]} \qquad \frac{\partial Z_d}{\partial g_{h_d^\star}} = 1$$

$$\frac{\partial g_{h_d^\star}}{\partial \lambda_i} = \mathbb{I}(i = i_d^\star) \qquad \frac{\partial \lambda_i(x)}{\partial W^0[i,p]} = x_{n^\star,p}$$

So:

$$\frac{\partial \mathcal{L}}{\partial W^0[i,p]} = \begin{cases} x_{n^\star,p} \cdot \frac{\partial \mathcal{L}}{\partial W^2[h_d^\star,d]}, & \text{if } i = i_d^\star \\ 0, & \text{otherwise} \end{cases}$$

总结。 上述推导表明，次梯度沿活跃路径稀疏地传播：

- $\frac{\partial \mathcal{L}}{\partial W^2[h,d]} = \mathbb{I}_{\{h=h_d^\star\}} \left(\hat{Y}_d(x_{n^\star}) - \mathbb{I}_{\{d=y_{n^\star}\}}\right)$.
- $\frac{\partial \mathcal{L}}{\partial W^1[i,h]} = \frac{\partial \mathcal{L}}{\partial W^2[h,d]}$ if $h = h_d^\star$ and $i = i_d^\star$.
- $\frac{\partial \mathcal{L}}{\partial W^0[i,p]} = x_{n^\star,p} \frac{\partial \mathcal{L}}{\partial W^2[h_d^\star,d]}$ if $i = i_d^\star$,

□

A.6. 置信度直方图

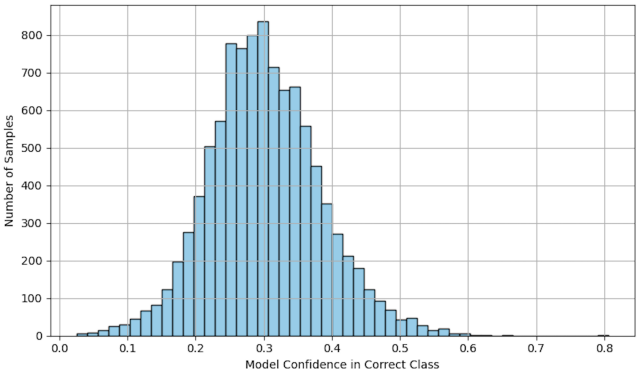

图 *3.* 使用 LMM 模型在 MNIST 上得到的真实标签预测概率分布。

A.7. 多层感知机与 LMM 在 MNIST 上的比较

表 *6.* **训练**性能（MNIST 数据集）。

模型	多层感知机（$\frac{1}{N}$）	LMM（$\frac{1}{N}$）	LMM（最大$_n$）
最大损失	27.63	5.02	1.64
平均损失	0.05	1.23	1.21
准确率	99.15%	69%	91.54%

表*7.* **测试**MNIST 数据集上的性能。

模型	多层感知机（$\frac{1}{N}$）	LMM（$\frac{1}{N}$）	LMM (max$_n$)
最大损失	**27.63**	4.55	3.59
平均损失	0.047	1.22	1.23
准确率	88%	69%	88.75%