

本科毕业论文（设计）

学 院： 数学与统计学院

专 业： 信息与计算科学

学生姓名： 王圳川

学 号： 20271022

指导教师： 周声龙

**北京交通大学**

2024年5月

学士论文版权使用授权书

本学士论文作者完全了解北京交通大学有关保留、使用学士论文的规定。特授权北京交通大学可以将学士论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名： 指导教师签名：

签字日期： 年 月 日 签字日期： 年 月 日

学士论文诚信声明

本人声明所呈交的毕业论文（设计），题目 深度神经网络的稀疏优化方法 是本人在指导教师的指导下，独立进行研究工作所取得的成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名： 日期：

中文摘要

**摘要：**中文摘要应将论文的内容要点简短明了地表达出来，约400字左右，字体为宋体小四号。内容应包括工作目的、研究方法、成果和结论。要突出本论文的创新点，语言力求精炼。为了便于文献检索，应在本页下方另起一行注明论文的关键词（3-5个），如有可能，尽量采用《汉语主题词表》等词表提供的规范词。图X幅，表X个，参考文献X篇。

**关键词：**

ABSTRACT

**ABSTRACT:**与中文摘要内容要相对应。

**KEYWORDS：**

目 录

[中文摘要 i](#_Toc414268225)

[ABSTRACT ii](#_Toc414268226)

[目 录 iii](#_Toc414268227)

[1 引言 1](#_Toc414268228)

[1.1 论文选题背景与意义 3](#_Toc414268230)

[1.2 计算机领域经典稀疏优化算法 3](#_Toc414268230)

[1.2.1 数据无关类 3](#_Toc414268231)

[1.2.2 数据驱动类 3](#_Toc414268231)

[1.2.3 训练感知类 3](#_Toc414268231)

[1.3 本文的主要工作与章节安排 3](#_Toc414268230)

[2 稀疏优化的最优性条件 2](#_Toc414268229)

[2.1 有稀疏约束条件的优化问题 3](#_Toc414268230)

[2.1.1 基本可行解 3](#_Toc414268231)

[2.1.2 -稳定性 3](#_Toc414268231)

[2.1.1 收敛性 3](#_Toc414268231)

[2.2 稀疏优化算法 3](#_Toc414268230)

[2.2.1 全局稀疏优化算法 3](#_Toc414268231)

[2.2.2 层级稀疏优化算法 3](#_Toc414268231)

[3 实验结果与讨论 4](#_Toc414268232)

[3.1 函数拟合实验 4](#_Toc414268233)

[3.1.1 实验设计 4](#_Toc414268234)

[3.1.1 实验结果 4](#_Toc414268234)

[3.2 双螺旋分类实验 4](#_Toc414268233)

[3.2.1 实验设计 4](#_Toc414268234)

[3.2.2 实验结果 4](#_Toc414268234)

[3.3 基于的文本分类实验 4](#_Toc414268233)

[3.3.1 实验设计 4](#_Toc414268234)

[3.3.2 实验结果 4](#_Toc414268234)

[5 结论 6](#_Toc414268238)

[参考文献 7](#_Toc414268239)

[致 谢 8](#_Toc414268240)

[附 录 8](#_Toc414268240)

目录说明：

（小四、黑体、左对齐、空1字符）

（小四、宋体、首行缩进1字符）

（小四、宋体、首行缩进2字符）

1. 引言
   1. 论文选题背景及意义

深度神经网络在许许多多的领域内（例如：图像分类、机器翻译、语音合成）展现出了最先进的水平。虽然一方面模型的质量在随着模型和训练数据的增大而提升，但是随之而来的训练和部署方面的成本也水涨船高。目前图像分类和机器翻译类的神经网络的参数量是以千万计的，并且通常预测一个结果就需要百亿量级的浮点数运算。这就导致这些领域内的先进模型的部署、预测成本到达一个不可接受的程度，尤其是移动和嵌入式设备上的模型部署、预测。一些近年来大热的巨型模型像目标检测领域的先进模型Inception-V3[1]，一次预测需要进行57亿次运算，拥有2,700,000个参数；NLP领域的GPT-3模型需要175,000,000个参数（假设其参数为16 bits精度的话大约需要350GiB的存储空间）。除了部署与预测，训练这些大模型成本更高，每一次训练大概需要数百万美元。

为了解决上述问题，稀疏优化成为了一个可行手段。通过稀疏优化，我们可以使得参数的一个子集置为0，而任何与权重0相乘的运算将被跳过，并且由于参数是稀疏的，可以采取稀疏存储的方式节省内存空间，极大程度上降低了模型的部署和预测成本[2]。比如Minerva模型[3]通过对激活值较小的参数的舍弃，大约节约了训练过程中50％的能源消耗；Eyeriss模型[4]通过将激活值置为0节约计算成本；Cnvlutin[5]也是通过类似的方法达到了与优化之前的模型13倍的速度差距和降低10倍的能源消耗。由此可见，对模型参数的稀疏优化的确能够大幅度降低模型训练、预测、部署的成本。

虽然对模型的稀疏优化能够极大程度上降低成本，但是有许多人认为这种大幅度的减少模型参数的方法势必也会伴随模型性能的大幅度下降。但是事实证明并非如此。有研究已经证明深度神经网络对参数的高稀疏程度具有相当强的容忍度[6]。甚至有研究发现深度神经网络中的95％的参数能够通过剩余的5％的参数预测出来[7]。这些都说明了一定程度的稀疏优化不会对模型的性能造成明显的影响。事实上模型性能与稀疏程度的关系通常如下图所示，一定的稀疏程度非但不会使模型的性能下降，反而会让模型的性能得到提升。一些研究和奥卡姆剃刀原则都能够解释这种现象：提高模型的稀疏程度能够降低模型中存在的噪声。



在过去的几年里，学术界提出了很多稀疏优化的方案，比如变分丢弃法[8]、正则法、结构剪枝。然而这些方法并非对模型参数的直接稀疏优化，而是通过训练过程中产生的梯度或激活值为指标，丢弃掉神经网络中的一些结构，从而达到稀疏优化的目的。或者是采取在损失函数中增加正则项的方式，通过训练隐式地让模型规模减小，间接地达到稀疏优化的目的。这些方案往往以直觉为导向，缺乏足够的数学理论支持。而本论文计划将深度神经网络抽象为最优化问题，从理论出发，提出了神经网络中的全局稀疏优化算法和层级稀疏优化算法，并通过三个实验验证了我们所提出的算法能够大幅度稀疏化模型参数，一定程度上提高模型的泛化效果，并且是一个通用的算法，能够用在不同任务形式和不同大小的模型的训练过程中。

万物之始，大道至简，衍化至繁。——老子《道德经》

* 1. 计算机领域经典稀疏优化方法

目前的稀疏优化方法种类繁多，最主要可以分为结构化的稀疏方法和非结构化的稀疏方法。结构化的稀疏方法比如直接去掉神经网络中的神经元、卷积神经网络中的卷积核[9]、Transformer类模型[10]中的注意力头或者更复杂一些的神经网络中的某一个模块这类具有某种固定模式的参数。结构化的稀疏方法虽然能够减少记录需要稀疏的参数的位置信息的存储开销，但是结构化的稀疏方法往往会导致模型效果的明显下降，这是因为结构化的稀疏方法会大大降低模型稀疏时的自由度。因此，结构化的稀疏往往是一种粗粒度且低开销的稀疏优化方法。相比之下，非结构化的稀疏方法，通常是根据一些衡量参数重要程度的指标来选择重要程度高的参数保留，其他参数置为0来达到稀疏优化的目的，在实际运用中发现，非结构化的稀疏方法往往能够同时带来高稀疏度和较好的模型效果。而我们所设计的全局稀疏优化算法和层级稀疏优化算法也属于非结构化稀疏方法的一种，因此，我会先在接下来的模块简单介绍一下目前的非结构化稀疏优化方法。

* + 1. 训练无关类

最简单也是最直觉性的一种非结构化稀疏优化是移除绝对值最小的参数，这种方法可以追溯到1993年，Hagiwara使用这种方法将75%的参数置为0，同时提高了10%的泛化效果[11]。还有Gale将卷积神经网络中的卷积核稀疏[12]，以及Guiying将这种方法推广到RNN中，在保证模型的泛化性能不明显下降的前提下，去掉了96.84%的连接[13]。虽然对于该方法的研究已经很多，但是大多数研究所讨论的模型参数相对于目前的千亿参数级别的大语言模型来说，规模还比较小。因此，在之前的研究中将全局参数进行排序，选取绝对值最大的前个参数所花费的时间是可以接受的，但是当参数规模到达千亿级别时，排序所花费的时间已经不可忽略了，这也是我们提出层级稀疏优化算法的主要原因。我们注意到在模型参数增加的同时，模型深度也在增加，因此我们提出了模型层之间相互独立的假设，对模型的每一层单独进行稀疏，这样就能够并行稀疏模型的每一层，大幅度缩短稀疏所消耗的时间。

* + 1. 数据驱动类

这一类稀疏优化方法主要考虑模型神经元或整个模型的输出的数值敏感度。在这一类方法中，通常存在一个训练数据的集合来确定哪些参数需要被移除。举例来说，输入该集合中的不同例子，激活值变化最小或者几乎不变的参数将被移除，因为它们对模型判别不同的输入基本没有贡献。因此，这个敏感度指标可以用来衡量参数的重要性，而重要性较低的参数将被移除。在移除后，这些移除参数的激活值会被加入到下一层神经元的偏置中，以维持整个神经网络的不变性[14]。Castellano[15]泛化了这一方法，并在最后移除参数之后，将根据与移除之前的激活值变化最小为依据，重新计算参数。上述方法同样也可以运用在卷积神经网络中，比如Luo[16]以卷积层输出的激活值为依据，删除在整个小批量训练数据中对激活值改变最小的卷积层参数。

* + 1. 训练感知类

除了上述两种稀疏优化方法以外，我们还可以使用一些训练感知的方法。其中一种最简单的方法就是考虑训练过程中的总参数变化量。我们可以存储所有参数在整个训练中的更新量的大小，然后移除更新量最少的参数[17]。直觉上的解释是如果参数自随机初始化开始到训练结束，其改变量很小的话，那么这些参数可以被认为是“不重要的”。

这些训练感知类方法中比较著名的一种是在损失函数中加入正则项，举例来说，其中是原始损失函数，而是关于模型参数的惩罚项。惩罚项的设计可以是关于任何神经元本身的[18]，也可以是关于一些指标的，比如所需浮点数运算[19]，添加惩罚项能够间接让模型在训练过程中变得稀疏，同时降低了复杂度。

1. 理论依据

最优性条件在最优化问题的研究中有着重要的地位，从实际运用的角度来说，最优性条件是许多数值解法的基础，所以我们希望从解的最优性条件出发去支撑我们的算法。然而神经网络中的最优化问题通常是非凸的，所以不存在充分必要条件来确保找到最优解，因此我们参考了Amir Beck所提出的理论[20]，将其推广到了神经网络中，在下面的板块列举出最优解的一些必要条件，并且分析它们之间的关系，并在2.2中详细介绍这些必要条件如何引导我们设计算法，找到满足这些条件的解的。

* 1. 稀疏优化的必要最优性条件
     1. 记号与假设

我们先假设数据集记数据集的标签，模型的预测值为。

对于单层神经网络，我们要优化的问题可以写做：

这里的，是该单层神经网络所有参数的范数，即参数矩阵*w*中的非零参数的个数。

对于多层神经网络来说，如果令，为神经网络的层数，则原优化问题可写做：

**假设2.1.** 假设神经网络每一层参数之间相互独立。

由假设，我们可以推得原问题在前层上每一层是一个有稀疏约束条件的优化问题，不失其一般性，我们假定为第层：

可以看到实际上与相同，为记号简便，后续讨论中直接讨论。

我们记为最多-稀疏的参数矩阵集合：

。

我们记为参数矩阵中非零参数的位置集合：

对应零参数的位置集合：

设，是指中绝对值最大的第个元素：

* + 1. 基本可行解

**定义2.1.** 是的一个基本可行解，如果有：

**定理2.1.** 如果令是的最优解，那么是一个基本可行解。

* + 1. -

对于问题

这里是闭凸可行域，是一个连续可导的函数，可能是非凸的。那么该问题的一个解可以被称为是稳定点，当：

如果是该问题的最优解，那么显然也是稳定点，因此，稳定点是一个必要最优性条件。许多解决上述非凸问题的优化方法都只能保证收敛到稳定点。

实际上述性质可以转化为，解是一个稳定点，当且仅当

这里假设，记作上的正交投影，即

上式推广至有稀疏约束的问题*P*中，记为-稳定性。

**定义2.2.** 一个解是问题的一个*L*-稳定点，如果满足关系

**引理2.1.** ，满足条件当且仅当和

**假设2.2.** 目标函数的梯度是李普希茨连续，设其李普希茨常数，即

该假设对目标函数成立，取，。

**引理2.2. (Decent Lemma)** 假设连续可导且满足假设3.1，

**引理 2.3.** 如果假设2.2成立，。对于任意，y是一个参数矩阵，满足

我们有

**定理2.2.** 假设2.2成立的前提下，，假设是问题的最优解，那么

1. 是一个-稳定点
2. 只包含一个元素
   * 1. 理论总结

通过上述定理我们发现这几个最优解的必要条件之间的关系如图所示。【插入图片】

* 1. 算法设计

现在我们根据上述提供的理论结果，设计两个稀疏优化算法，分别是全局稀疏优化算法和层级稀疏优化算法。

* 全局稀疏优化算法主要使用-稳定性必要最优性条件，该算法与Amir Beck在其论文[20]中所提出的IHT硬阈值算法比较类似，但是我们将该算法推广至了多层神经网络之中，我们会证明全局稀疏优化算法所产生的极限点是-稳定点。
* 层级稀疏优化算法
  + 1. 全局稀疏优化算法
    2. 层级稀疏优化算法

**假设2.1.** 假设神经网络每一层参数之间相互独立。

由假设

1. 实验

在本章中，我们设计了三个实验来证明我们所设计的算法的有效性。第一个实验是拟合Sinc函数的回归实验，第二个实验是双螺旋数据的分类实验，第三个实验是基于Transformer架构的实际文本分类实验。通过前两个实验我们可以验证算法的通用性。因为目前几乎所有大模型都是以Transformer为基础架构，所以第三个实验验证了我们的算法对大模型进行稀疏优化的潜力。

实验环境为Python3.8，显卡配置为NVIDIA RTX A6000

* 1. 拟合Sinc函数的回归实验

我们考虑使用神经网络从有噪声的数据中拟合Sinc函数，其中噪声指的是均值为0，方差的高斯噪声。

假设训练数据集样本数量，我们将在该数据集上训练一个隐藏层为2的神经网络即

神经网络示意图如图所示。

超参数设置为为保留的参数占模型总参数量的比例，，这里当时相当于没有对模型进行稀疏优化。优化器为Adam优化器，学习率设置为0.001，算法终止条件为更新前后模型所有参数之间的距离小于。

* + 1. 稀疏优化中的参数更新可视化

为了了解在不同稀疏度下，神经网络中参数的实际更新情况和层内与层间参数重要性的差异。我们先在一个小型的神经网络上使用了全局稀疏优化算法，画出了参数保留频次热力图和算法停止时模型最终的参数情况。如图所示。

这里以，即保留40%的参数时举例。假设以参数的绝对值大小决定参数重要性。首先在同一层中，可以看到参数矩阵第一层保留频次都在16000次左右，不同位置的参数保留频次相差很小。而第二层和第三层中，同一层内一部分参数保留频次在16000次左右，而大量参数保留频次在2000次左右，相差约8倍。所以在同一层中不同位置的重要性在第一层中相差不大而在第二层和第三层中相差很大。其次不同层之间，第一层平均保留频次远高于第二层和第三层，所以可以推测第一层的重要性远高于第二层和第三层。

下面是算法停止后，最终模型中的参数情况，其中白色位置的参数为0，非白色位置的参数非0。

这三张图直观反映了我们的算法确实能够将模型内部的参数矩阵变得稀疏，能够达到我们的目的。通过对比三张图我们可以发现最终模型第一层没有值为0的参数，这进一步印证了我们前文关于不同位置参数重要性的猜想。并且通过对比图和图，我们发现算法停止后，模型内部非0参数的位置和参数保留频次热力图中颜色较深的位置重合度较高，这启示我们模型权重在训练后期进行稀疏优化的位置是相对固定的，这启示我们在后续的研究中可以重新设计算法，在训练中期挑选出更新频次比较固定的参数位置，在训练后期无需再计算全局参数的梯度，只计算目标位置的梯度，更新目标位置的参数，其他参数直接置为0即可。这样就能够大大降低模型训练过程中的计算量。

* + 1. 实验结果

3.1.1中的实验为了能够可视化参数矩阵因而故意将隐藏层神经元数目降低。在正式实验中，我们同样使用隐藏层数目为2的神经网络，其中第一个隐藏层的神经元数目为200个，第二个隐藏层的神经元数目为300个，则该神经网络的总参数量为60500（不计算偏置项），激活函数为Sigmoid函数。为减少随机性带来的影响，我们进行了多次实验，最终结果取其平均值。

对于该回归任务，我们的评价指标是RMSE，即。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数保留比例s | 全局稀疏优化 | | 层级稀疏优化 | |
|  |  |  |  |
| 100% | 0.0701 | 0.0931 | 0.0701 | 0.0931 |
| 90% | 0.0724 | 0.1329 | 0.0706 | 0.0971 |
| 80% | 0.0715 | 0.1156 | 0.0704 | 0.0851 |
| 70% | 0.0720 | 0.1111 | 0.0711 | 0.0943 |
| 60% | 0.0713 | 0.1032 | 0.0705 | 0.0887 |
| 50% | 0.0703 | 0.1123 | 0.0694 | 0.0932 |
| 40% | 0.0743 | 0.1626 | 0.0695 | 0.0901 |
| 30% | 0.0842 | 0.2829 | 0.0855 | 0.2756 |
| 20% | 0.0805 | 0.2184 | 0.0875 | 0.3064 |
| 10% | 0.0941 | 0.3885 | 0.1037 | 0.6664 |
| 5% | 0.1766 | 0.8886 | 0.1067 | 0.6575 |

通过表格数据可知，对于全局稀疏优化算法来说，能够在模型泛化性能不发生显著下降的前提下，将模型参数减少到原来的50%。拟合曲线对比如图所示。

对于层级稀疏优化算法来说，能够在模型泛化性能不发生显著下降的前提下，将模型参数减少到原来的40%。并且能够在参数量为原来的60%时获得优于不进行稀疏优化时的泛化效果。

对比全局稀疏优化算法和层级优化算法，两者在保留同一比例的参数时，层级稀疏优化算法的泛化效果往往优于全局稀疏优化算法，这是因为全局稀疏优化算法倾向于将最后一层的大部分参数置为0，这会导致在前两层学习到的参数变相丢失，更直观的示意图如图所示。

* 1. 双螺旋数据分类实验

我们考虑使用神经网络将双螺旋数据进行分类，其中两类点各1000个样本按80%、20%的比例分为训练集和验证集。其中数据集的构建方式参考论文【插入论文】，对于单个数据，详细构建方式参考下列公式：

其中为服从标准正态分布的高斯噪声。所生成数据点如图所示。

超参数设置为为保留的参数占模型总参数量的比例，，这里当时相当于没有对模型进行稀疏优化。优化器为Adam优化器，学习率设置为0.001，算法终止条件为更新前后模型所有参数之间的距离小于。

* + 1. 稀疏优化中的参数更新可视化

该部分结论与3.1.1中方法和结论基本相同。为了可视化稀疏优化算法中权重保留过程，我们使用了小型的神经网络，并使用全局稀疏优化算法，画出了参数保留频次热力图和算法停止时模型最终的参数情况。如图所示。

这里以时为例，即保留神经网络中20%的权重参数。权重保留频数图和最终模型内权重情况可视化结果如下图所示，结论与3.1.1相同。

* + 1. 实验结果

3.2.1中的实验为了能够可视化参数矩阵因而故意将隐藏层神经元数目降低。在正式实验中，我们同样使用隐藏层数目为2的神经网络，其中第一个隐藏层的神经元数目为200个，第二个隐藏层的神经元数目为300个，则该神经网络的总参数量为60500（不计算偏置项），激活函数为Sigmoid函数。为减少随机性带来的影响，我们进行了多次实验，最终结果取其平均值。

对于该回归任务，我们的评价指标是分类正确率。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数保留比例s | 全局稀疏优化 | | 层级稀疏优化 | |
|  |  |  |  |
| 100% | 99.72% | 99.75% | 99.72% | 99.75% |
| 90% | 99.66% | 99.50% | 99.64% | 99.50% |
| 80% | 99.75% | 99.75% | 99.85% | 99.92% |
| 70% | 99.88% | 100.00% | 99.79% | 99.71% |
| 60% | 99.81% | 100.00% | 99.88% | 99.75% |
| 50% | 99.84% | 100.00% | 99.82% | 99.58% |
| 40% | 99.81% | 100.00% | 99.49% | 99.75% |
| 30% | 80.25% | 75.63% | 99.28% | 99.71% |
| 20% | 59.12% | 51.13% | 99.79% | 99.75% |
| 10% | 50.25% | 49.00% | 63.89% | 60.04% |
| 5% | 50.25% | 49.00% | 63.40% | 59.79% |

通过表格数据可知，对于全局稀疏优化算法来说，能够在模型泛化性能不发生显著下降的前提下，将模型参数减少到原来的40%。分类结果如图所示。

对于层级稀疏优化算法来说，能够在模型泛化性能不发生显著下降的前提下，将模型参数减少到原来的20%。并且能够在参数量为原来的80%时获得优于不进行稀疏优化时的泛化效果。

* 1. 基于Transformer架构的实际文本分类实验

为了验证我们的算法除了在一般规模的神经网络和简单的模拟数据上，在更大规模和实际应用任务上也具有良好的稀疏优化效果，我们建立了有约1,800,000参数量的神经网络对实际网络服务的描述文本进行风险等级预测。并且我们还希望验证我们的算法对大语言模型也能够进行稀疏优化的潜力。但是在1.1中我们已经提到目前大模型的巨大参数规模，由于硬件设施的限制，我们不能够直接将我们的算法应用在大语言模型中。所以我们建立的模型以Transformer为基础架构，间接展示我们的算法对大语言模型的稀疏优化的潜力。

本章的安排是，我们首先在3.3.1中简单介绍Transformer架构，然后在3.3.2中介绍所使用的数据集，最后在3.3.3中展示实验结果和结论。

* + 1. Transformer架构

Transformer架构最早是由Vaswani等在2017年提出的。【插入文献】Transformer架构目前已在自然语言处理领域获得了非常广泛的应用，特别是在大语言模型领域，基本所有的领域内效果领先的大语言模型都采用了Transformer架构，比如Chat-GPT[21]、Chat-GLM[22]、Baichuan[23]和LLaMA[24]等等。Transformer架构通常包含编码器和解码器两部分，由于我们的任务形式是文本分类任务，因此只需要用到编码器部分，即图中左侧红色圈出的部分。【插入图片】

首先对于要分类的文本，需要使用文本分词工具，例如jieba分词将文本分成Tokens，而每个Token将对应词汇表中的一个独热向量，将独热向量与词嵌入矩阵相乘即可得到该Token的词嵌入，进而能够得到整个文本的词嵌入。得到词嵌入以后由于Transformer编码器是对整个词嵌入并行编码，因此还需要给文本的词嵌入加上位置编码，这样才能让Transformer得到文本的完整语义信息。得到上述处理过后的词嵌入之后，需要使用多头注意力机制对词嵌入进行特征提取，多头注意力机制的处理方式可以由下列公式进行简单概括，其中Q、K、V分别是词嵌入与三个方阵相乘所得，而即Transformer中的一部分参数组成。

多头注意力机制得出的结果还需要与原输入残差连接和进行归一化以后输入线性层，并重复残差连接和进行归一化最终得到Transformer编码器的输出，然后将输出再作为下一个编码器的输入再进行编码，重复上述操作。在堆叠多个编码器后，将最后一个编码器的输出连接一个线性层，即可完成分类任务的训练。所以本章所构建的模型结构图如图所示。

我们建立的模型中，序列长度取256，分类类别为4，模型维度取256，嵌入维度取256，隐藏层维度取512，编码器的数目取4，则可以估算出模型的总参数量为1,800,000。

* + 1. 实验数据集

本实验中使用的数据集是从国家信息安全漏洞共享平台上爬取的，包含146,514条关于网络服务漏洞的文本描述，以及相关的风险等级评定。其中几条展示如下表。

|  |  |
| --- | --- |
| 描述文本 | 风险等级 |
| IBM InfoSphere Information Server 8.1，8.5 FP3之前版本以及8.7版本和InfoSphere Business Glossary 8.1.1，8.1.2版本中的Information Services Framework (ISF)中存在漏洞，该漏洞源于程序没有将登录页面中密码字段的自动填表功能关闭。通过利用无人看守的工作站，远程攻击者可利用该漏洞获得访问权限。 | 低危 |
| IBM Informix Dynamic Server (idS) 10.00.xC8之前的10.x版本中的多个未明程序允许本地用户通过列出环境变量中的target文件创建任意文件，该变量的所有权被转移到应用这些程序的用户。 | 中危 |
| Restlet 1.1.10版本中存在代码问题漏洞。远程攻击者可利用该漏洞获取敏感信息。 | 高危 |
| WordPress Icegram Email Subscribers & Newsletters插件4.1.7及之前版本中存在SQL注入漏洞，该漏洞源于基于数据库的应用缺少对外部输入SQL语句的验证，攻击者可利用该漏洞执行非法SQL命令。 | 超高危 |

按照训练集占80%，测试集占20%的比例划分总数据集。最终训练集包含117,193条数据，测试集包含29,321条数据。

* + 1. 实验结果

我们使用分类正确率作为评估标准，这里我们发现对于层级稀疏优化算法，在参数保留比例降至10%以下时，其泛化效果相比于不进行稀疏优化时都没有大幅度的下降，因此我们将参数保留比例的选择范围拓展至。此外，由于模型参数量太大并且过拟合问题严重，如果以稀疏优化前后模型参数的距离作为算法停止的标志，训练时间将是不可接受的，因此我们在该实验中采用早停策略控制算法的停止，即模型在测试集上连续五轮正确率不变时，算法终止。最终实验结果如下表所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数保留比例s | 全局稀疏优化 | | 层级稀疏优化 | |
|  |  |  |  |
| 100% | 71.70% | 66.63% | 71.70% | 66.63% |
| 90% | 65.25% | 65.46% | 95.05% | 65.69% |
| 80% | 74.36% | 66.95% | 94.34% | 65.68% |
| 70% | 88.10% | 66.89% | 93.54% | 65.75% |
| 60% | 86.54% | 66.61% | 92.00% | 66.14% |
| 50% | 90.12% | 66.44% | 88.56% | 66.21% |
| 40% | 85.72% | 66.29% | 83.89% | 66.45% |
| 30% | 84.93% | 66.36% | 79.36% | 66.11% |
| 20% | 81.18% | 65.97% | 74.72% | 66.00% |
| 10% | 60.55% | 28.81% | 78.36% | 65.71% |
| 5% | 62.03% | 36.46% | 74.59% | 64.95% |
| 4% | 47.86% | 30.03% | 74.51% | 64.62% |
| 3% | 61.62% | 50.02% | 75.09% | 64.19% |
| 2% | 60.42% | 50.02% | 71.74% | 62.70% |
| 1% | 59.85% | 50.02% | 64.75% | 53.94% |

针对上述实验结果，有以下四点讨论：

* 首先从全部实验结果来看，该文本分类实验存在非常严重的过拟合问题，基本对于不同的稀疏优化算法以及不同的参数保留比例，最终训练集上的正确率都远远高于测试集上的正确率，比如对于层级稀疏优化算法，当参数保留比例为90%时，模型在训练集上的正确率比在测试集上的正确率高出了约30%，这验证了我们在1.1中提到的目前的模型具有非常严重的参数冗余的问题。
* 对于全局稀疏优化算法来说，保留参数在20%以上时，模型的泛化效果可以近似看做不变，在保留参数比例70%、80%时，模型的泛化效果超过了不进行稀疏优化的时候，这再次验证了模型参数的大量冗余以及我们的算法的有效性。对于保留参数比例从10%到5%的结果，从直觉上来说，对于四分类问题，如果模型未从训练集上学习到任何知识，其在验证集以及训练集上的正确率都应该趋近于25%，即随机猜测。我们原本对模型效果变化趋势的预测是，随着模型保留参数比例下降，模型效果应该随之下降并且正确率应趋近于25%。而实际在保留参数比例10%时，模型在测试集上的正确率最低。这可能是早停策略的原因，模型训练终止的条件由收敛变成了测试集上正确率连续五轮不升高。从宏观角度来说，模型在训练轮次趋于无穷时，模型在测试集上的正确率变化应呈现我们预测的趋势，因此这是一个正常的现象。
* 对于层级稀疏优化算法来说，参数保留比例在2%及以上时，模型的泛化效果可以看做不发生明显下降，验证了参数的冗余和我们的算法的有效性。而我们发现层级稀疏优化算法的实验结果中没有出现全局稀疏优化算法中的反常现象，这可能是由于层级稀疏优化的模型对稀疏算法的容忍度较高，训练过程中在测试集上的正确率呈稳定上升的趋势，因此并未出现全局稀疏优化算法中的反常现象。于是我们分别画出了当保留参数为10%时，模型在测试集上的正确率随训练轮次变化的曲线图，左图是全局稀疏优化算法，右侧是层级稀疏优化算法，可以看到全局稀疏优化算法的测试集正确率波动很大，而层级稀疏优化算法的测试集正确率则非常稳定，这就验证了我们的假设。
* 对比全局稀疏优化算法和层级稀疏优化算法，我们发现层级稀疏优化算法能够进一步降低保留参数比例，造成这个结果的原因除了有第三点中阐述的层级稀疏优化算法更加稳定以外，我们还检查了保留参数比例为10%时，该模型的最后一个线性层中非零参数的比例，假设为，。即最后一个线性层共有262144个参数，而最终算法停止后，最后一个线性层中非零参数共2个，与3.1.2拟合Sinc函数的回归实验结果中提到的结论相同，即全局稀疏优化算法会让最后一层参数基本全部置为0，这也是导致全局稀疏优化算法不如层级稀疏优化算法的原因之一。
  1. 实验小结

这些实验结果有力地验证了我们提出的层级稀疏优化算法在降低模型部署难度，提高模型泛化性能上的能力，并且还证明了我们提出的层级稀疏优化算法是一个通用的稀疏优化算法，它能在不同的任务形式、不同的模型参数量下取得良好的稀疏优化效果。

1. 结论

论文的结论是最终的、总体的结论，不是正文中各段的小结的简单重复。结论应该准确、完整、明确、精练。如果不可能导出应有的结论，也可以没有结论而进行必要的讨论。可以在结论或讨论中提出建议、研究设想、仪器设备改进意见以及尚待解决的问题等。

参考文献

[1] SZEGEDY C, VANHOUCKE V, IOFFE S, et al. Rethinking the inception architecture for computer vision; proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, F, 2016 [C].

[2] KALCHBRENNER N, ELSEN E, SIMONYAN K, et al. Efficient neural audio synthesis; proceedings of the International Conference on Machine Learning, F, 2018 [C]. PMLR.

[3] REAGEN B, WHATMOUGH P, ADOLF R, et al. Minerva: Enabling low-power, highly-accurate deep neural network accelerators [J]. 2016, 44(3): 267-78.

[4] CHEN Y-H, KRISHNA T, EMER J S, et al. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks [J]. 2016, 52(1): 127-38.

[5] ALBERICIO J, JUDD P, HETHERINGTON T, et al. Cnvlutin: Ineffectual-neuron-free deep neural network computing [J]. 2016, 44(3): 1-13.

[6] HAN S, POOL J, TRAN J, et al. Learning both weights and connections for efficient neural network [J]. 2015, 28.

[7] DENIL M, SHAKIBI B, DINH L, et al. Predicting parameters in deep learning [J]. 2013, 26.

[8] MOLCHANOV D, ASHUKHA A, VETROV D. Variational dropout sparsifies deep neural networks; proceedings of the International conference on machine learning, F, 2017 [C]. PMLR.

[9] POLYAK A, WOLF L J I A. Channel-level acceleration of deep face representations [J]. 2015, 3: 2163-75.

[10] MICHEL P, LEVY O, NEUBIG G J A I N I P S. Are sixteen heads really better than one? [J]. 2019, 32.

[11] HAGIWARA M. Removal of hidden units and weights for back propagation networks; proceedings of the Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), F, 1993 [C]. IEEE.

[12] ELSEN E, DUKHAN M, GALE T, et al. Fast sparse convnets; proceedings of the Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, F, 2020 [C].

[13] LI G, YANG P, QIAN C, et al. Stage-wise magnitude-based pruning for recurrent neural networks [J]. 2022.

[14] SIETSMA, DOW. Neural net pruning-why and how; proceedings of the IEEE 1988 international conference on neural networks, F, 1988 [C]. IEEE.

[15] CASTELLANO G, FANELLI A M, PELILLO M J I T O N N. An iterative pruning algorithm for feedforward neural networks [J]. 1997, 8(3): 519-31.

[16] LUO J-H, WU J, LIN W. Thinet: A filter level pruning method for deep neural network compression; proceedings of the Proceedings of the IEEE international conference on computer vision, F, 2017 [C].

[17] YANG D, GHASEMAZAR A, REN X, et al. Procrustes: a dataflow and accelerator for sparse deep neural network training; proceedings of the 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), F, 2020 [C]. IEEE.

[18] ZHUANG T, ZHANG Z, HUANG Y, et al. Neuron-level structured pruning using polarization regularizer [J]. 2020, 33: 9865-77.

[19] MOLCHANOV P, TYREE S, KARRAS T, et al. Pruning convolutional neural networks for resource efficient inference [J]. 2016.

[20] BECK A, ELDAR Y C J S J O O. Sparsity constrained nonlinear optimization: Optimality conditions and algorithms [J]. 2013, 23(3): 1480-509.

[21] KOCOŃ J, CICHECKI I, KASZYCA O, et al. ChatGPT: Jack of all trades, master of none [J]. 2023, 99: 101861.

[22] DU Z, QIAN Y, LIU X, et al. Glm: General language model pretraining with autoregressive blank infilling [J]. 2021.

[23] YANG A, XIAO B, WANG B, et al. Baichuan 2: Open large-scale language models [J]. 2023.

[24] TOUVRON H, MARTIN L, STONE K, et al. Llama 2: Open foundation and fine-tuned chat models [J]. 2023.

致 谢

放置在参考文献页后，对象包括：1）国家科学基金，资助研究工作的奖学金基金，合同单位，资助或支持的企业、组织或个人。2）协助完成研究工作和提供便利条件的组织或个人。3）在研究工作中提出建议和提供帮助的人。4）给予转载和引用权的资料、图片、文献、研究思想和设想的所有者。5）其他应感谢的组织和个人。

附 录

附录A 程序代码

附录是作为论文主体的补充项目，并不是必须的。

论文的附录依序用大写正体英文字A、B、C……编序号，如：附录A。

附录B 工程

图纸