期末论文：复现机器学习中的顿悟现象

刘宇扬
School of Mathematical Sciences
Peking University
2401110049
liuyuyang@stu.pku.edu.cn

卢天泽
School of Mathematical Sciences
Peking University
2100010869
2100010869@stu.pku.edu.cn

摘要

本篇论文以Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets中的实验为基础，在多种模型上复现了机器学习中的顿悟（grokking）现象。实验基于 Python 编程语言和 Pytorch 框架实现。我们针对模加法展开研究，在不同的问题设置以及训练算法下，在 Transformer、MLP、ResNet 和 LSTM 模型上均观察到了不同程度的顿悟现象。最后，我们根据大量实验结果的比较，对于顿悟现象的产生归结为了过拟合问题。进一步提出了数据的轮换一致性对顿悟现象可能的影响，并展开了相关实验以辅佐我们的猜想。实验代码等辅助材料可以从如下链接获取：`https://github.com/LiuOnly1121/MIML_FinalProject_Grokking.git`。

1 引言

顿悟现象（grokking）是指，当模型参数量存在大量冗余时，训练集准确率很快达到接近 100%，但验证集上的准确率在很长时间内几乎不增长，随后突然增长到接近 100% 的现象。这一现象表明在某些问题下，过拟合后经过长时间训练依然能够得到泛化能力较好的模型。

本文中学习的目标问题是在模 p 意义下的模加法问题。在第 2 部分，我们会给出这个问题的定义。在第 3 部分，我们会依次展示不同模型实验设置以及实验结果。在第四部分，我们会对实验结果做总结，并分析 grokking 现象产生的原因。

2 问题描述

本文的探究对象是在模 p 意义下的模加法问题，定义如下

2.1 K 个数求和

给定 K 个 $\mathbb{Z}_p$ 中的数，它们在模 p 意义下的求和为

$$(x_1, x_2, \cdots, x_K) \mapsto (x_1 + x_2 + \cdots + x_K) \mod p \qquad \forall x_1, x_2, \cdots, x_K \in \mathbb{Z}_p$$

2.2 问题编码

为了使模型识别输入，我们把问题视作字符串到字符串的“翻译”问题，输入和输出都视为“字母” $0, 1, \cdots, p-1, +, =$ 构成的字符串。将所有合法的输入字符串构成的集合记为 $\mathcal{D}_K$。以 $K = 2$ 为例，如果我们输入的形式为字符串“$0 + 0 =$” $\in \mathcal{D}_2$，希望得到输出“0”。所以我们进行如下编码：

1. 首先我们将 $\mathbb{Z}_p$ 中的元素依次对应于 $\{0, 1, \cdots, p-1\}$ 中的元素，再将“+”、“=”依次对应于 $p, p+1$。对字符串的每个字母使用该映射，我们便得到了：

$$\mathcal{D}_K \to \{0, 1, \cdots, p+1\}^{2K}$$

2. 接着我们使用 Pytorch 中的 embedding 层，将每个“字母”映射到一个 $\mathbb{R}^{d_{model}}$ 中的向量，进而有

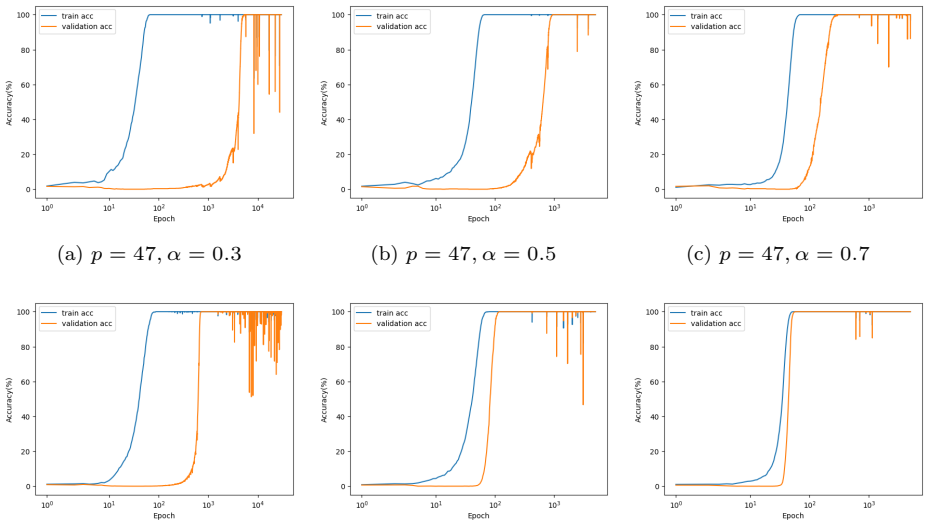
$$\{0, 1, \cdots, p+1\}^{2K} \to \mathbb{R}^{d_{model} \times 2K}$$

3. 本文我们根据不同问题选择 $p = 47$ 或 $p = 97$，$d_{model} = 128$

3 实验

在实验 3.1, 3.2, 3.3 中，我们只考虑 $K = 2$ 的情况。

3.1 Transformer 模型

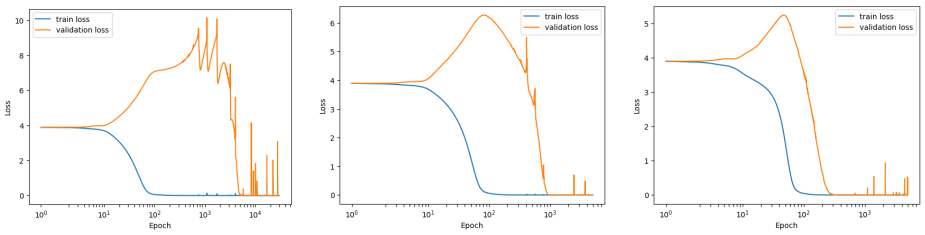


(a) $p = 47, \alpha = 0.3$ (b) $p = 47, \alpha = 0.5$ (c) $p = 47, \alpha = 0.7$

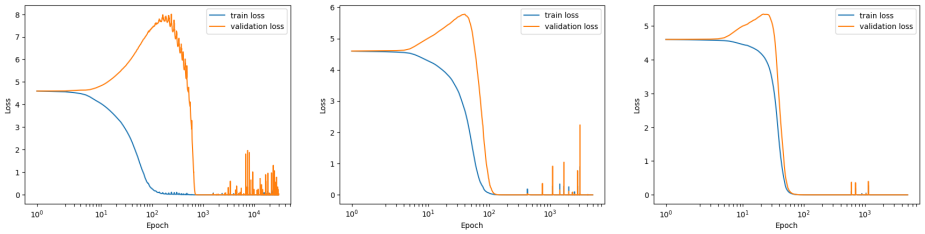
(d) $p = 97, \alpha = 0.3$ (e) $p = 97, \alpha = 0.5$ (f) $p = 97, \alpha = 0.7$

图 1: 不同的 α 和 p 下的预测准确率

实验中使用的是一个编码器和解码器分别为 2 层，隐藏层大小为 $d_{ff} = 512$，4 头注意力机制，$dropout = 0.1$ 的 transformer 模型，编码向量的大小为 $d_{model} = 128$，优



(a) $p = 47, \alpha = 0.3$ (b) $p = 47, \alpha = 0.5$ (c) $p = 47, \alpha = 0.7$



(d) $p = 97, \alpha = 0.3$ (e) $p = 97, \alpha = 0.5$ (f) $p = 97, \alpha = 0.7$

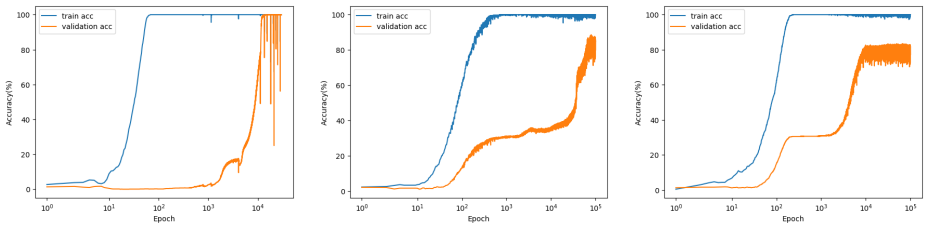
图 2: 不同的 α 和 p 下的损失函数

化器为 AmamW，训练的超参数为 $lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.98, WeightDecay = 1.0$。我们选取 $p = 47, 97$，对于 $\alpha = 0.3, 0.5, 0.7$ 进行了实验，记录了个 epoch 后模型在训练集和验证集上的准确率和损失，实验结果参见图1与图2。从图中可以看到，这些实验均出现了明显的 grokking 现象，并且当训练集占比较小以及 p 更小时，grokking 现象更加明显。

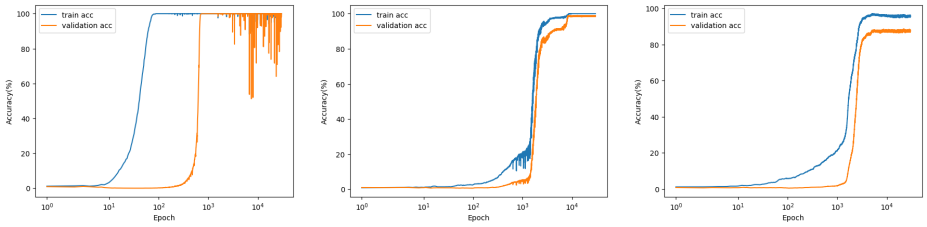
3.2 其他模型

我们选取了长短期记忆网络 (LSTM)、多层感知机 (MLP)、残差网络 (ResNet) 三种模型进行实验。其中 LSTM 模型中编码器和解码器分别是两层长短期记忆网络，隐藏层宽度为 128，输出层为解码器后的一个线性全连接层，最后，仅保留序列输出的第 1 位作为最终的分类概率。MLP 模型中，首先将编码后的序列数据取平均，得到 MLP 的输入，之后经过四个宽度为 128 的隐藏层，最后到 $p + 2$ 维的输出，每个维度大小代表分类概率。ResNet 与 MLP 类似，但是增加了残差层：在每个隐藏层经过 ReLU 函数后，与原输入相加再经过一次 ReLU 函数。

训练的优化器均选为 AdamW，结果见图3。可以发现，在这些模型中都出现了不同程度的 grokking 现象。其余 α 下的实验也出现了与实验 3.1 一致的结果。值得注意的是，在 MLP 和 ResNet 模型中，随着 p 的增大，问题复杂度增加，训练集的收敛速度有了明显下降（这里为了展示依然存在 grokking 现象，在这两个模型 $p = 47$ 时取的步长更长，但是此时只会加快收敛速度，不影响相关结论）。但 Transformer 和 LSTM 更擅长处理长距离序列中的相互关系，将该问题视作自然语言问题，在训练集上，它们对于更大的素数并没有表现出收敛速度的区别。



(a) LSTM, $p = 47, \alpha = 0.3$ (b) MLP, $p = 47, \alpha = 0.3$ (c) ResNet, $p = 47, \alpha = 0.3$



(d) LSTM, $p = 97, \alpha = 0.3$ (e) MLP, $p = 97, \alpha = 0.3$ (f) ResNet, $p = 97, \alpha = 0.3$

图 3: 不同模型下的 grokking 现象

3.3 不同优化器和正则化方法的对比

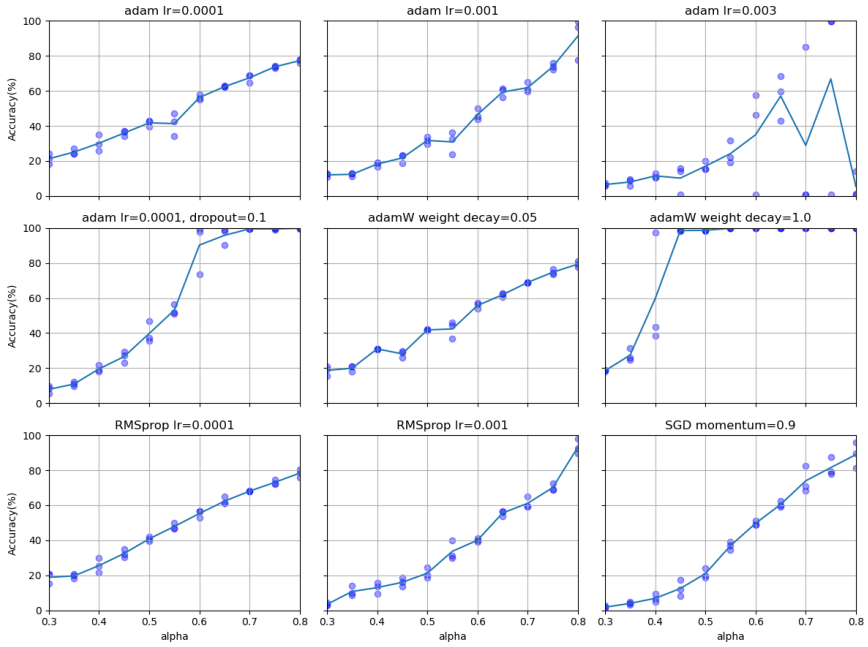


图 4: 不同优化器以及正则化方法下，1000epoch 后验证集准确率的收敛情况

在这个实验中我们分析了不同优化器和正则化方法对于 grokking 现象的影响。我们选取了 Adam 优化器（学习率分别为 $1 \times 10^{-4}, 1 \times 10^{-3}, 3 \times 10^{-3}$），使用 Dropout

正则化方法的 Adam 优化器（学习率为 1×10^{-4}，Dropout 取 0.1），AdamW 优化器（Weight Decay 分别为 $0.05, 1.0$），RMSprop 优化器（学习率为 1×10^{-3}），随机梯度下降（SGD）优化器（分别为不使用 Momentum，使用 Momentum 为 0.9）。在不同优化器上，在 0.3 至 0.8 之间以 0.05 为间隔取 α，每个优化器和 α 上以 1000epoch 训练 3 次，用散点表示出每次训练中在验证集上的最大准确度，并将这三次取平均值后用折线连接。结果如图4所示。

对比 Adam 优化器是否使用 Dropout 的情形，可知正则化方法对于提高模型在该问题上的的泛化能力有重要的作用。对比 AdamW 和 Adam 优化器也能发现，AdamW 能够更好地正则化；因为该问题并不复杂，参数冗余多，所以选择较大的 Weight Decay 能尽快提高模型的泛化能力。SGD 没有显性的正则化方法，所以在 α 较小时表现不佳，但提供充分训练样本后也表现出了很好的泛化能力。RMSprop 没有引入动量，所以在该问题上收敛较慢。

3.4 对于 $K = 2, 3, 4$ 的分析

为了分析加法元素数目 K 对于 grokking 现象的影响，我们分别选取 $K = 2, 3, 4$，在 $p = 17, \alpha = 0.5$ 的条件下进行实验，记录了每个 epoch 后模型在训练集和验证集上的准确率，结果如图5所示。

当 K 增大时，问题的复杂度会明显增加。实验结果可以见图5。由于 p 的取值非常小，当 $K = 2$ 时，会出现十分严重的过拟合现象，导致 1000 个 epoch 内验证集上无法泛化；$K = 3$ 时，有明显的 grokking 现象，此时能够在 1000epoch 内收敛；$K = 4$ 时，即使在训练集上也无法正常收敛，推测是问题过于复杂，现有模型参数量和训练样本数不足以找到问题较好的表示。

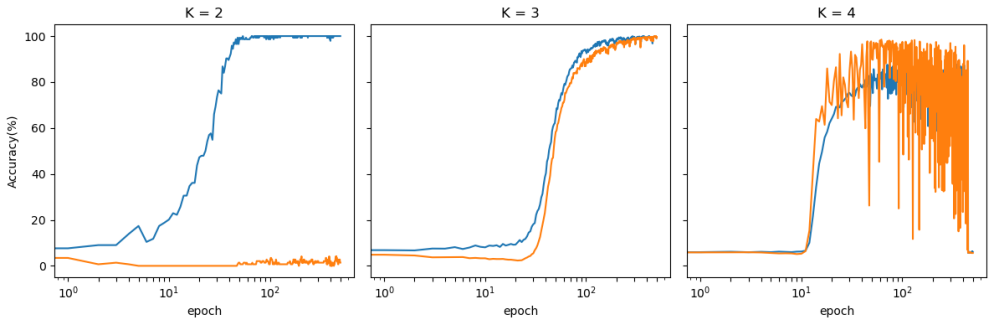


图 5: $K = 2, 3, 4$ 时的问题复杂度

为了进一步体现 grokking 现象的程度，我们又选取了 $K = 2, 3$，在 $p = 31, \alpha = 0.5$ 下实验

结果如图6所示。对于相同的 p, α，$K = 2$ 时的 grokking 十分明显，但 $K = 3$ 时就没有 grokking 现象。但是如图8我们在后续实验 4.2 中，通过对数据集的处理，实现了 $K = 3$ 的 grokking 现象。

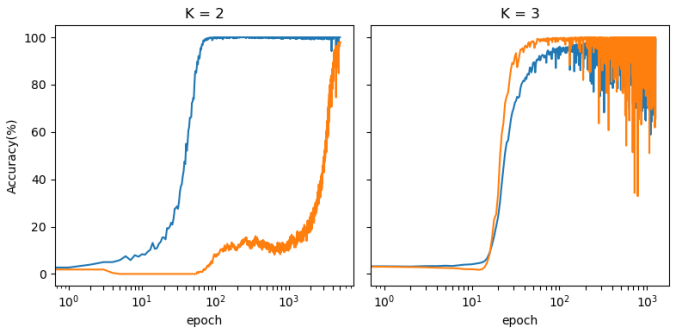


图 6: $K = 2, 3$ 对 grokking 现象的影响

4 对于 grokking 现象的解释与进一步实验

4.1 过拟合与正则化过程

grokking 现象分为三个阶段。第一个阶段，模型拟合训练集，训练集上的准确率迅速上升，验证集能力很差；第二阶段，模型在训练集上的准确率接近 100%，但验证集上的准确率长时间保持较低状态；第三阶段，验证集上的准确率迅速上升至 100%。

我们有充分的理由相信，第二阶段是严重的过拟合表现。这在我们的以上许多实验结果中可以得到验证：

- 在图1中，对于相同的 α，$p = 47$ 时的 grokking 现象要显著比 $p = 97$ 时更明显，对于相同的 p，α 更小时的 grokking 现象更明显。图2中测试集损失函数在第二阶段仍增大的现象也说明了这一点。

- 在图3中，所有模型都在 $p = 47$ 时的表现出了更明显的 grokking 现象。对于表达能力相对较弱的 MLP 和 ResNet 模型，当 $p = 97$ 时（也就是图3e和图3f），训练集上的收敛较慢，而且经过测试，此时如果需要出现 grokking 现象，需要更小的步长与更小的批处理数量。说明模型对于问题的参数冗余并不大，这时几乎没有 grokking 现象。

- 在图5中，更小的 K 表现出更明显的 grokking 现象，更大的 K 甚至不收敛。

总结以上实验结果，对于相同的模型和训练方法，模型在更简答的问题上会出现更明显的 grokking 现象，当每次提供的训练集规模更小时会出现更明显的 grokking 现象。对于相同问题，表达能力更强的模型会出现更明显的 grokking 现象。

所以，此时为了增强模型的快速泛化能力，需要依靠正则化，也就是降低模型参数的复杂度。我们在图4中可以发现使用了 dropout 方法后，验证集的收敛速度明显加快。而关于为什么没有使用正则化方法的模型，最后也会具有泛化能力，这是因为许多算法有隐性泛化能力。例如用 SGD 方法做线性回归，最后总能收敛到 l_2 范数最小的系数，在这些问题中也是如此，这些算法总能收敛到一个复杂度较低的模型，进而最终使模型有良好的泛化性能。

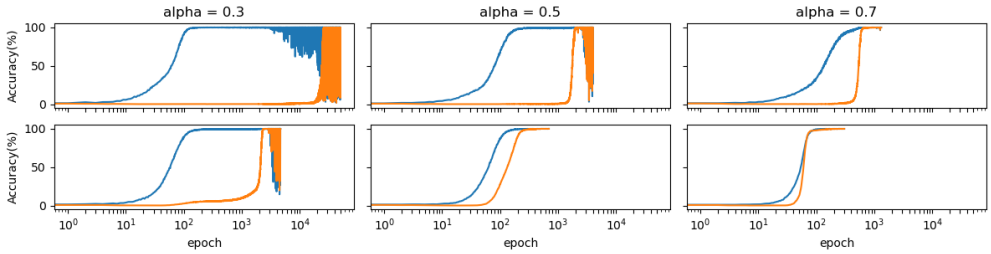


图 7: 有序数据集与原数据集的对比的对比

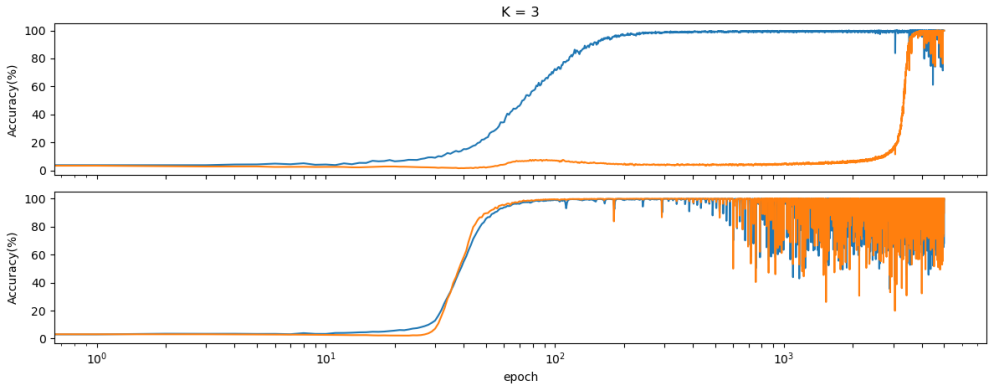


图 8: 有序数据集与原数据集的对比的对比

4.2 轮换对称性

我们注意到该问题具有轮换对称性，我们合理推测，能否学习到这一性质对于验证集上的收敛有很大的作用。我们在原数据集的基础上，选取了所有满足 $x_1 \leq x_2 \leq \cdots \leq x_K$ 的数据，构成有序数据集。$K = 2$ 时，在该数据集上，对于 $\alpha = 0.3, 0.5, 0.7$ 进行了实验，并且与原数据集进行对比。

结果如图7所示（第一行为有序数据集上的结果，第二行为原数据集上的结果）。可以看到相对于混杂的数据，提供轮换对称的数据的 grokking 现象明显增强。

在 $K = 3$ 时，互为轮换对称的数据对更多，这一结果变得更加明显，结果见图8。在全数据集上已经看不到 grokking 现象，但是在遮挡轮换对称性的筛选过的有序数据集上依然有特别明显泛化缓慢的 grokking 现象。

References

[1] Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. arXiv preprint arXiv:2201.02177.

[2] Humayun, A. I., Balestriero, R., & Baraniuk, R. (2024). Deep networks always grok and here is why. arXiv preprint arXiv:2402.15555.