11

Assignment 3:Scaling Law

参考资料 11.1

幂律分布：*https://www.zhihu.com/question/312593367*

Scaling Law: https://zhuanlan.zhihu.com/p/671327709

11.1. Scaling Law

Scaling Law 指出，大语言模型的性能（通常以 Test Loss 衡量）与三个主要变量之间存在着严格的**幂律（Power Law）**关系。这三个变量是：

1. **计算量 (Compute, C)**：训练模型所用的 FLOPs。
2. **数据集大小 (Dataset Size, D)**：训练用的 Token 数量。
3. **参数量 (Parameters, N)**：模型神经网络的权重数量。

Extension 11.1 幂律分布

幂律意味着：规模扩大时，性能按指数规律提升，但提升幅度逐渐递减。

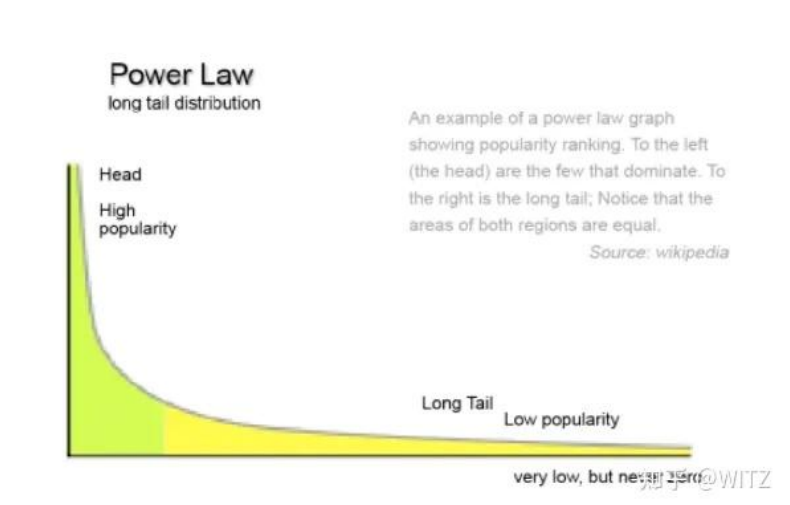

图 11.1: 幂律分布示意图

11.1.1. Scaling Law 相关公式

Chinchilla 被认为是目前最通用的 Scaling Law，它修正了早期过分看重模型大小的误区，强调了**数据量**的重要性。

Chinchilla Loss Formula

这是目前最通用的计算公式。

$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$

参数说明：

- **L(N, D)**：模型在给定参数量 N 和训练数据量 D 下的预估损失值（Loss）。损失越低，性能越好。
- **E**：不可约损失（或称贝叶斯误差）。这是基于自然语言本身的熵，代表了模型性能的**理论极限**。即使用了无限的参数和无限的数据，Loss 也不会低于 E。
- **N**：模型参数数量。
- **D**：训练数据的 Token 数量。
- **A, B, α, β**：这些是根据实验数据拟合出来的常数。
  - $\frac{A}{N^{\alpha}}$：表示由于**模型太小（容量不足）**而导致的额外损失。
  - $\frac{B}{D^{\beta}}$：表示由于**数据太少（训练不足）**而导致的额外损失。

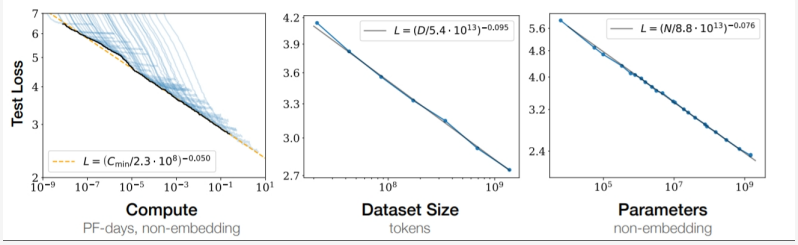

图 11.2: C、D、N 之间的关系

> Keynote 11.1
>
> 关键发现：*Chinchilla* 研究得出 $\alpha \approx 0.34$ 且 $\beta \approx 0.28$。这意味着模型参数量 N 和数据量 D 对性能的贡献是**大致相当的**（指数相近），因此在扩大算力时，应该同比例扩大模型和数据（即 *1:1* 缩放），而不是像以前那样盲目追求巨大的模型参数。

在实际训练中，我们通常受限于计算约束。计算量 C 与 N 和 D 的关系近似为：

$C \approx 6 \times N \times D$

- **C (FLOPs)**：总浮点运算次数。
- **6**：这是一个经验系数（在 Transformer 架构中，每个 Token 的训练大约需要 6N 次浮点运算（处理 D 个 Token 自然就 6ND），其中前向传播 2N，反向传播 4N）。

有了这个公式，我们的好处就是**可以通过训练小模型，精准预测大模型的表现**，从而避免为了验证想法而浪费几百万美元训练一个无用的巨型模型。

具体来说，就是已知小模型的 Loss 后，根据幂律曲线来估算大模型的 Loss。

Kaplan 的单变量幂律公式

他认为模型性能主要取决于规模的指数级增长。

$L(N) \approx (\frac{N_c}{N})^N$

$L(D) \approx (\frac{D_c}{D})^D$

$L(C) \approx (\frac{C_c}{C})^C$

早期的 Kaplan 认为**参数量 (N)** 是王道。于是拼命把模型做大，哪怕数据量不够。这就使得总体模型是欠拟合的。

“IsoFLOPs 方法”(等计算量分析法)

讲义里 Chinchilla 关于推导 Scaling Law 的方法

在固定计算量 C （比如设置为 10e20 FLOPs）的前提下，衡量模型参数 N 和数据量大小 D 的关系。

当他们把实验结果画成图（横轴是模型大小 N，纵轴是最终 Loss）时，发现曲线呈现出一个 **“U 型”**

1. **左端：模型 (N) 太小**
   - **解释**：虽然你给小模型喂了海量的数据（因为 N 小，D 就可以很大），但它的“脑容量”太小了，根本记不住也学不会这么复杂的知识。导致**欠拟合**。
   - **结果**：Loss 很高。
2. **右端：数据 (D) 太小**
   - **解释**：你造了一个超级巨大的模型，但因为预算 C 是固定的，你剩下的钱只够让它读几页书（D 非常小）。模型还没来得及收敛，计算预算就花光了。这叫 **训练不充分**。
   - **极端的例子**：原文提到，如果 N 无穷大，你的预算可能连做一次梯度下降（算一步）都不够，训练直接结束。
   - **结果**：Loss 也很高。
3. 底部：甜点区 (Sweet Spot)
   - **解释**：在这里，N 近似等于 D，Loss 最低。

实操步骤:

1. **找最低点**：对于每一个预算等级（比如 C_1, C_2, C_3），他们都画出那个 U 型曲线，并找到曲线最低点对应的**最优模型大小**，记为 $N_{opt}(C)$。
2. **连点成线**：现在有了一系列的数据点对：$\langle C_1, N_{opt1} \rangle, \langle C_2, N_{opt2} \rangle, \ldots$

3. **拟合幂律**：他们用数学公式去拟合这些点，得出结论：

$N_{opt} \propto C_a$

$D_{opt} \propto C_b$

实际意义与应用

- **预测未来新模型**：给定目标计算预算 C，直接用公式估算最优 N_opt 和 D_opt，避免盲目超大或超小。
- **小模型外推**：通过训练一系列小模型拟合 A、B、α、β、E 等参数，即可可靠预测百倍、千倍规模模型（比如 GPT-4 的预算）的性能。
- **避免浪费**：过去常花数百万美元训练一个配置不佳的巨型模型；现在可用十分之一预算的小实验验证想法。