大模型在强化学习中探索和学习的熵机制

参考论文：《https://arxiv.org/pdf/2505.22617》

🌰 **论文速览**

- **熵崩溃（entropy collapse）现象**

所谓熵崩溃，指的是在不加熵正则（如entropy loss或KL正则）干预的情况下，RL训练过程中**policy entropy会在早期迅速下降，最终趋近于零**，导致模型几乎只会选择极少数确定性的动作，探索能力完全丧失。

这直接带来了下游reward（性能）的上限瓶颈：模型性能在训练初期快速提升，但很快到达平台期，后续难以突破。

作者通过大量实验发现，无论采用何种主流RL算法（GRPO、RLOO、PRIME等），甚至算法细节变化，**策略熵**（$\mathcal{H}$）与**下游reward**（R）都始终呈现出高度一致且可预测的数学关系：

$$R = -a\ exp(\mathcal{H}) + b$$

其中 a, b 为拟合系数，反映了**特定模型**和**训练数据**的内在特性。上式说明，RL在“用熵换reward”，而当policy熵耗尽后，性能的上限也被锁死。因此，如果不主动干预，熵崩溃是必然的，reward上限也完全可以提前预测到：$\mathcal{H} = 0, R = -a + b$。

- **熵崩溃的内在机制**

进一步地推导和实验证明，对于SoftMax策略，**策略熵的单步变化主要由动作的输出概率与其logits变化量之间的协方差 (Covariance)所驱动**。对于基于**策略梯度（Policy gradient）**推导出的算法（如PPO），logits的变化量与该动作的优势成正比。换句话说，高概率且具有高优势的动作会减少策略熵，罕见且高优势的动作会增加策略熵。

- **解决方案：基于协方差的熵控制**

熵控制的关键在于限制高协方差token的更新，避免大幅降低整体的策略熵。提出了两种控制策略熵的方法，鼓励探索从而获得更好的下游性能：

  - **Clip-Cov**: 识别出具有最高协方差的一小部分token，并在策略更新时，对其梯度的影响进行裁剪或直接分离其梯度，防止它们对策略进行过度激进的更新。
  - **KL-Cov**: 在高协方差token上施加一个额外的KL散度惩罚项，旨在约束当前策略与先前策略在这些关键token上的分布差异，从而间接控制熵的下降速度。

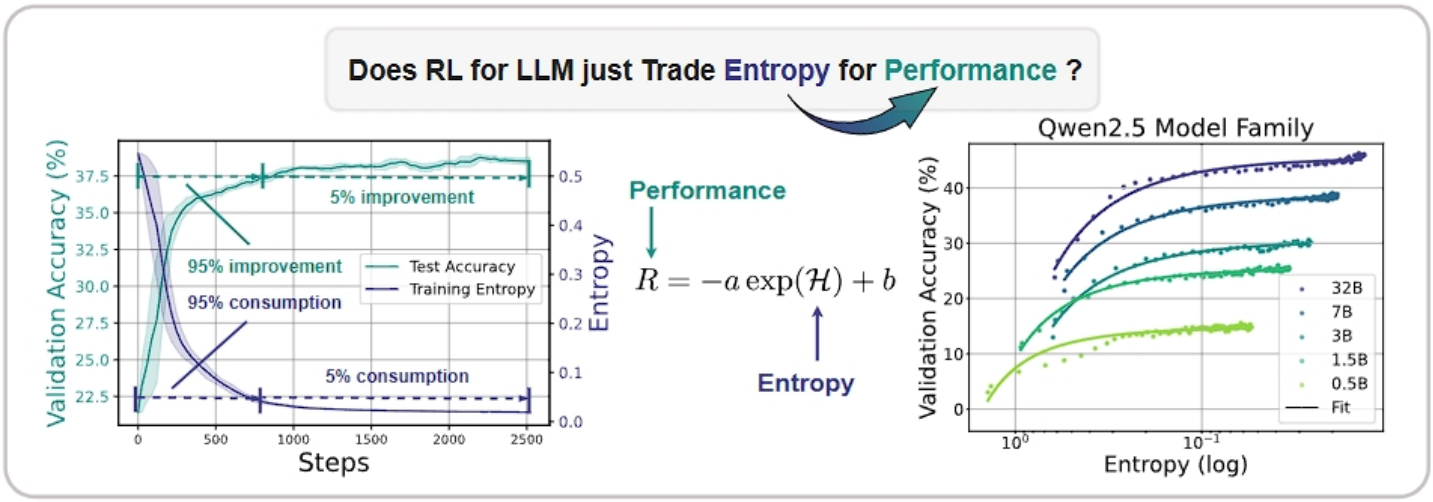
策略熵可预测崩溃

强化学习的核心挑战就是exploration-exploitation之间的平衡。**策略熵**是衡量exploration能力的关键因素，它衡量了模型在行动选择时的不确定性。在llm而言，策略熵在训练前期迅速下降至接近0，即策略变得很稳定，从而限制了模型性能提升。且**下游任务性能**（R）与**策略熵**（$\mathcal{H}$）之间的关系可以预测为：

$$R = -a\ exp(\mathcal{H}) + b$$

> 基于该公式，可以用scaling law，在RL早期阶段就能预测策略模型性能，并根据小模型预测大模型性能。

如左图所示，随着训练进行，熵崩溃的同时性能达到饱和，并且超过95%的熵下降/性能增加发生在RL训练的早期阶段。右图说明模型性能和策略熵之间的关系可以预测，在不加干预的情况下，模型性能显示出明显的上限，即当熵值耗尽时也就达到了策略模型性能上限：$\mathcal{H} = 0, R = -a + b$。



背景知识

首先回顾RL在可验证任务（如数学和代码，以避免奖励劫持）上优化LLM的设置。给定输入prompt x，LLM π_θ 自回归的生成输出序列 y，包含 $y_1, ..., y_T$ token。RL的目的是最大化从验证器获得的累积奖励 $r(y)$，其中 $\mathcal{D}$ 是数据分布。

$$\max_{\theta \to \infty} J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(x)}[r(y)]$$

策略梯度算法（policy gradient）的梯度估计为：

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(x)}[\sum_{t=0}^{T} \nabla_\theta\ log\ \pi_\theta(y_t|y_{<t}) A_t]$$

其中 A_t 表示当前动作的优势值，**REINFORCE算法**定义优势值为 $A_t = r(y)$；**GRPO**做了组归一化 $A_t = \frac{r(y) - mean(r(y^{1:K}))}{std(r(y^{1:K}))}$；**PPO**通过优化一个代理目标函数来约束策略更新的幅度：

$$E_t[min(\frac{\pi_\theta(y_t|y_{<t})}{\pi_{old}(y_t|y_{<t})}) A_t, clip(\frac{\pi_\theta(y_t|y_{<t})}{\pi_{old}(y_t|y_{<t})}, 1-\epsilon, 1+\epsilon) A_t]$$

策略熵 $\mathcal{H}(\pi_\theta, \mathcal{D})$ 量化动作选择的随机性，衡量策略模型在训练数据上的平均token级别的熵：

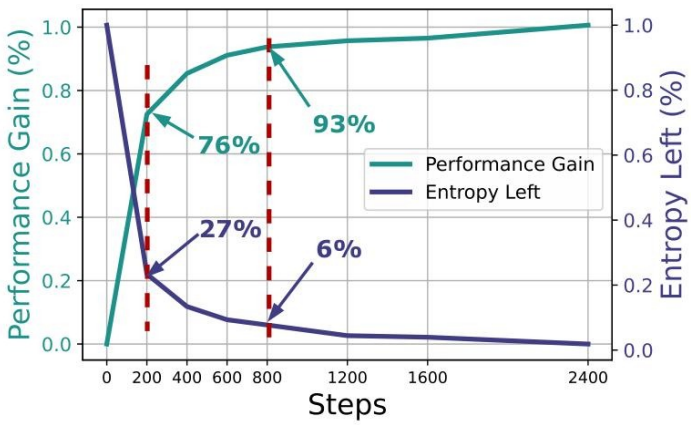
$$\mathcal{H}(\pi_\theta, \mathcal{D}) = -\mathbb{E}_{\mathcal{D},\pi_\theta}[log\ (\pi_\theta(y_t|y_{<t}))] = -\frac{1}{\mathcal{D}}\sum_{x\in\mathcal{D}}\frac{1}{|y|}\sum_{t=1}^{|y|}\mathbb{E}_{y_t\in\pi_\theta}[log\ (\pi_\theta(y_t|y_{<t}, x))]$$

初步观察：熵崩溃与性能饱和

- **模型**：Qwen2.5（0.5B, 1.5B, 3B, 7B, 32B），Mistral（7B，24B)，LLaMA（3B，8B），DeepSeek-Math-7B-Base。
- **训练任务**：可验证的数学和代码任务。
- **训练数据**：根据模型推理能力，选择不同难度的数据进行训练，包括（数学：GSM8K，Eurus-2-RL-Math，，代码：AceCode，Eurus-2-RL-Code 和 KodCode）。
- **测评数据**：数学：ATH500，AIME 2024，AMC，OlypiadBench，OMNI-MATH，代码：Eurus-2-RL-Code， KodCode
- **算法：GRPO，REINFORCE++，PRIME。**
- **超参数设置**：采用veRL框架训练，学习率 5×10^{-7}，PRIME中隐式PRM学习 10^{-6}，batch256，micro-batch128，每个prompt采样8个响应，参考KL散度系数设置为0，$\epsilon = 0.2$。过滤掉所有响应都正确或者错误的提示（也就是太难或者太简单的prompt）。

右图呈现了2400步RL运行中，11个不同模型的平均归一化熵消耗/性能提升的百分比。

- 200步出现了73%的熵消耗和76%的性能提升。
- 800步出现了93%的熵消耗和94%的性能提升。



实验结果表明模型训练初期策略熵就会快速下降，伴随着模型性能快速提升至饱和，模型变得高度自信且确定。

定量关系：策略熵与模型性能曲线

下游任务性能（R）与**策略熵**（$\mathcal{H}$）之间的关系可以预测为：

$$R = -a\ exp(\mathcal{H}) + b$$

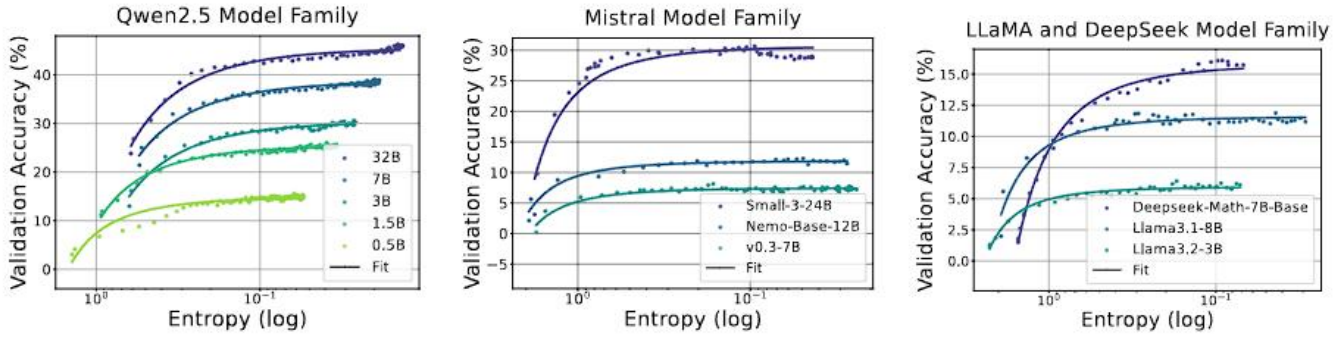


Figure 3: Fitting curves between policy entropy and validation performance on math task. We conduct validation every 4 rollout steps until convergence.

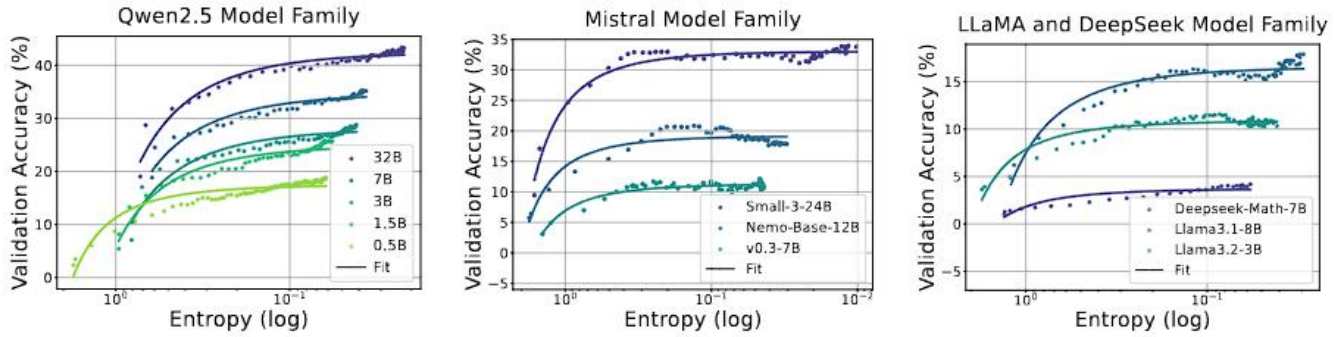


Figure 4: Fitting curves between policy entropy and validation performance in coding task. We conduct validation every 4 rollout steps until convergence.

基于以上的拟合，可以在训练初期（高熵）预测模型最终性能（低熵），例如在Qwen2.5家族上，只需要用前36步来拟合 a和b，就可以预测之后的模型性能，最终预测的RMSE不超过2%。

> **超参数观察**
>
> - **超参数与算法无关**：使用不同的RL算法（包括GRPO, RLOO, PRIME）对同一模型进行训练，拟合出的 a, b 基本一致，这表明 a, b 与使用的RL算法无关。
> - **超参数的含义**：对拟合方程求导，$dR/d\mathcal{H} = -a\ exp(\mathcal{H})$, a 表示模型将熵转化成下游性能的速率，$-a + b$ 决定了模型的性能上限。
> - **超参数与模型大小和数据集相关**：a和b 都随着策略模型大小以对数线性速率平滑变化，可以根据较小模型的训练情况推测大模型的超参数（类似于Scaling law）。

策略熵动态分析

总结：对于使用Softmax函数输出概率的策略模型，包括LLM在内，**策略熵的单步变化主要由动作的对数概率与其logits变化量之间的协方差所驱动，高协方差的动作会加速策略熵的下降**。而对于基于**策略梯度**的强化学习算法，logits的变化量与该动作的优势成正比。

Softmax策略的熵变化

LLM的输出层通常采用Softmax函数将logits转换为概率分布：

$$\pi_\theta(a|s) = \frac{exp(z_{s,a})}{\sum_{s' \in \mathcal{A}} exp(z_{s,a'})}$$

其中 $s \sim d_{\pi_\theta}$ ，即已经生成的token $y_{<t}$ ； $a \sim \pi_\theta^k(\cdot|s)$ ，即当前要生成的token； $z_{s,a}$ 表示在状态-动作对 (s,a) 输出的logits值。有以下引理：

> 🏆 **引理1：**
>
> $$\mathcal{H}(\pi_\theta^{k+1}|s) - \mathcal{H}(\pi_\theta^k|s) \approx -Cov_{a\sim\pi_\theta^k(\cdot|s)}(log\pi_\theta^k(a|s), z_{s,a}^{k+1} - z_{s,a}^k)$$

$\mathcal{H}(\pi_\theta^k|s) = -\sum_a \pi_\theta^k(a|s)\ log\pi_\theta^k(a|s)$ ， $z_{s,a}^{k+1} - z_{s,a}^k$ 表示前后两步之间的logits变化， $Cov_{a\sim\pi_\theta^k(\cdot|s)}$ 表示在第k步策略 $\pi_\theta^k(\cdot|s)$ 下，对所有动作计算的协方差。引理1表明熵的变化，与动作和logits变化之间协方差项负相关。

对于策略梯度算法，其策略模型的梯度估计为：

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(x)}[\sum_{t=0}^{T} \nabla_\theta\ log\ \pi_\theta(y_t|y_{<t})A_t]$$

> 🥛 **命题1：**
>
> 连续两步之间的logit差值为： $z_{s,a}^{k+1} - z_{s,a}^k = -\eta \cdot \nabla_\theta J(\theta)$ ，其中 $J(\theta)$ 表示强化学习的目标函数，对于表格型Softmax策略，在一次梯度上升计算后，logit的变化为：
>
> $$z_{s,a}^{k+1} - z_{s,a}^k = \eta\pi_\theta^k(a|s)A(s,a)$$

命题1表明在策略梯度算法中，一个动作导致的logit增加正比于学习率、当前策略选择该动作的概率、该动作的优势值。将命题1和引理1结合，就有：

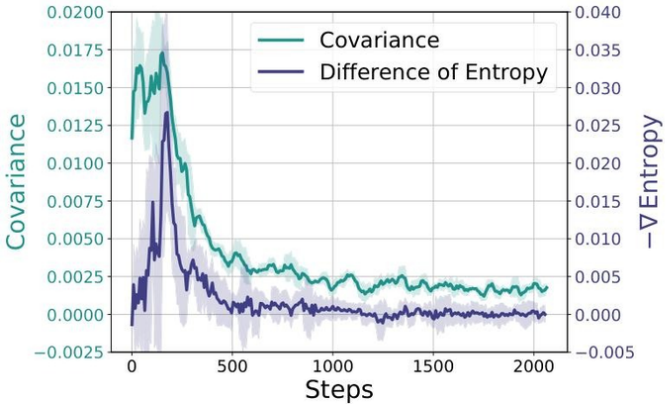
> 🌰 **定理1：**
>
> 对于表格softmax策略，连续两步之间的策略熵满足：
>
> $$\mathcal{H}(\pi_\theta^{k+1}|s) - \mathcal{H}(\pi_\theta^k|s) \approx -\eta Cov_{a\sim\pi_\theta^k(\cdot|s)}(log\pi_\theta^k(a|s), \pi_\theta^k(a|s)A(s,a))$$

根据定理1，高概率且高优势的动作会导致协方差为正，从而导致熵减。在自然策略梯度算法中也有类似的结论。

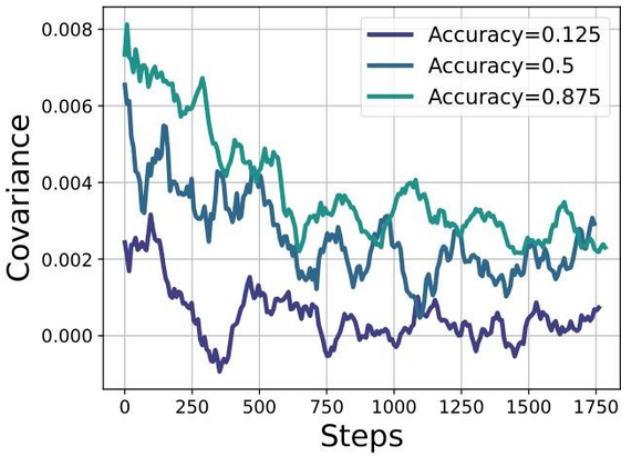
> 📚 **定理2：**
>
> 对于自然梯度算法，logit的变化与优势值 $A(s,a)$ 成正比： $z_{s,a}^{k+1} - z_{s,a}^k = \eta A(s,a)$ ，连续两步之间的策略熵满足：
>
> $$\mathcal{H}(\pi_\theta^{k+1}|s) - \mathcal{H}(\pi_\theta^k|s) \approx -\eta Cov_{a\sim\pi_\theta^k(\cdot|s)}(log\pi_\theta^k(a|s), A(s,a))$$

实验验证

下图展示出了在Qwen2.5-7B模型上，采用GRPO算法的结果。采用bandit设定，即将prompt x 作为状态，整条回复 y 作为动作。



每一步策略熵的实际变化量 $-(\mathcal{H}(\pi_\theta^{k+1}|s) - \mathcal{H}(\pi_\theta^{k}|s))$ 与协方差在训练过程中呈现高一致性，并且协方差基本都是正的，导致策略熵持续下降直至耗尽。



由于GRPO算法对每个prompt采样多个响应，可以按照这些响应的准确率将prompt划分为不同难度等级。观察到难样本的协方差值更小，因为模型对任务不确定，高概率动作不一定能带来高优势，反之简单任务熵策略模型置信度高，高概率动作与高优势对齐更好，导致策略熵快速下降。

熵控制：协方差正则化

基于以上分析，高协方差是导致熵崩溃的主要原因，因此可以通过限制高协方差token的更新，缓解熵衰减过快。

传统的熵正则化方法主要有以下两种：

- **熵损失**：$L_{ent} = L - \alpha H(\pi_\theta)$，即在原始损失上减去一个策略熵损失。这种方法受 α 影响很大，且性能并未超过不加熵损失的baseline。
- **KL惩罚**：即在原始损失上加策略模型和参考模型之间的KL-散度，$L_{KL} = L + \beta D_{KL}(\pi_\theta||\pi_{ref})$。这种方法虽然能稳定策略熵，但是牺牲了模型性能，因为强制把策略模型拉向参考模型，限制了强化学习的探索能力。

限制高协方差token

通过分析训练过程中token的协方差分布发现，少数token的协方差特别高，贡献了绝大多数的熵值减少。通过识别并限制这些高协方差token，可以有效缓解熵崩溃。参考PPO中为了稳定训练的方法，即clip和KL惩罚，提出了**clip-cov**和**KL-cov**。

首先定义每个token的协方差，从定理2出发，对于采样到的一个batch中的N个token，其中token y_i 对数概率与优势之间的的中心叉积为：

$$Cov(y_i) = (log\pi_\theta(y_i) - \frac{1}{N}\sum_{j=1}^{N} log\pi_\theta(y_j)) \cdot (A(y_i) - \frac{1}{N}\sum_{j=1}^{N} A(y_j))$$

- **Clip-Cov**：不让协方差高的token全部参与梯度计算中，设定一个协方差上下界 $[w_{low}, w_{high}]$ (远大于平均协方差)，协方差落在这个区间的token，随机裁剪掉很小一部分(比如总token数的 2×10^{-4})的token，在计算策略损失时，这些token的梯度置为0。
- **KL-Cov**：对所有token的协方差进行排序，选择Top k%(例如 2×10^{-4})的token添加KL惩罚，惩罚当前策略与上一轮策略（采样时用的策略）之间的差异。

实验验证

可以看到Clip-Cov和KL-Cov都带来了显著的性能提升，对于更大的模型，缓解熵崩溃能带来更大的性能提升。

Method	AIME24	AIME25	AMC	MATH-500	OMNI-MATH	OlympiadBench	Minerva	Avg.
Qwen2.5-7B								
GRPO	21.2	9.6	58.7	78.8	27.9	40.7	36.7	38.6
w. Clip-higher	18.1	11.5	56.6	79.2	29.8	43.3	40.4	38.8
w. CLIP-Cov	22.1	**15.8**	58.2	80.4	**30.5**	**44.1**	**41.1**	40.4
w. KL-Cov	**22.6**	12.9	**61.4**	**80.8**	29.1	42.6	38.2	**40.6**
Qwen2.5-32B								
GRPO	21.8	16.2	69.7	84.2	35.2	43.6	45.5	45.8
w. Clip-higher	35.6	22.3	69.5	77.2	35.1	42.5	43.0	47.2
w. CLIP-Cov	32.3	22.7	67.2	87.0	**42.0**	**57.2**	46.0	50.3
w. KL-Cov	**36.8**	**30.8**	**74.5**	**84.6**	39.1	49.0	**46.3**	**52.2**

- 左图表示，Clip-Cov和KL-Cov都能够将策略熵维持在一个更高的水平。
- 中图表示，模型生成的响应长度稳步增加，模型能够更自由的探索。
- 右图表示，Clip-Cov和KL-Cov在测试集上的性能始终更优，这表明通过主动控制熵，模型不仅探索能力增强，而且能将探索到的知识有效地转化为下游任务的性能提升。