

投机解码

从接受—拒绝采样到并行验证

投机解码（speculative decoding）的目标，是减少大模型串行解码的次数：先由便宜的草稿模型连续提出若干 token，再由目标模型一次验证多个位置。对于采样式投机解码，接受—拒绝规则和残差分布保证最终结果仍严格服从目标模型，而不是近似草稿模型。

两个核心问题

- 数学上：如何把从草稿分布 q 得到的样本校正为目标分布 p ？
- 计算上：如何用一次昂贵的目标模型前向计算确认多个 token？

目录

1. 普通自回归解码	2
2. 投机解码的基本设定	2
3. 单位置投机采样	2
3.1. 分布与样本不能混淆	2
3.2. 接受规则	2
3.3. 拒绝后的残差分布	3
3.4. 单位置流程	3
4. 为什么最终分布严格等于目标分布	4
4.1. 完整证明	4
4.2. 概率质量的分解	4
4.3. 接受率与总变差距离	4
5. 从一个 token 扩展到多个 token	5
5.1. 草稿模型连续提出候选	5
5.2. 目标模型为什么能一次验证	5
5.3. 必须从左到右验证	5
5.4. 全部接受时的额外 token	5
5.5. 多 token 算法	6
6. 多位置算法为何仍然正确	6
7. 加速从哪里来	6
8. 正确性边界与实现注意事项	7
8.1. “完全一致”具体指什么	7
8.2. 采样解码与贪心解码	7
8.3. 数值与系统实现	7
9. 总结	7
10. 参考资料	8
11. 问题	8

1. 普通自回归解码

设已经确认的前缀为

$$x_{<t} = (x_1, x_2, \dots, x_{t-1}). \quad (1)$$

目标模型在位置 t 给出条件分布

$$p_{t(v)} = P(x_t = v \mid x_{<t}), \quad v \in \mathcal{V}, \quad (2)$$

其中 $\mathcal{V}$ 是词表。模型前向计算先产生 logits, softmax 再将其变成分布; 此时还没有生成新 token。随后执行

$$x_t \sim p_t \quad (3)$$

才得到最终 token。普通解码直接从目标分布采样, 所以不需要接受、拒绝或补偿。

问题在于自回归依赖是串行的: 只有确定 x_t 后才能得到真正的 $p_{t+1}(\cdot)$ 。若每次目标模型前向计算只推进一个 token, 生成延迟会受到内存带宽、调度开销和串行依赖的限制。

2. 投机解码的基本设定

引入一个计算成本较低的草稿模型。对于同一已确认上下文, 它给出

$$q_{t(v)} = Q(\tilde{x}_t = v \mid x_{<t}). \quad (4)$$

草稿模型负责“提议”, 目标模型仍是最终标准。一次投机解码迭代包含两阶段:

1. **提出候选**: 草稿模型自回归地产生 K 个候选 token。
2. **验证候选**: 目标模型对整段候选做一次前向计算, 并从左到右决定接受多少个。

先忽略 $K > 1$ 的情况, 只研究一个位置。这个单位置问题包含了投机采样的全部概率论核心。

3. 单位置投机采样

3.1. 分布与样本不能混淆

假设词表只有 A, B, C , 草稿分布和目标分布分别为

$$q = (0.60, 0.30, 0.10), \quad p = (0.40, 0.35, 0.25). \quad (5)$$

token	q : 草稿	p : 目标
A	0.60	0.40
B	0.30	0.35
C	0.10	0.25

q 只是分布。草稿模型还要执行一次采样

$$\tilde{x} \sim q \quad (6)$$

才产生具体的 draft token。假设本次得到 $\tilde{x} = A$, 后续接受判断只检查这个已经抽中的 A , 不会逐个“比较并选择”整个词表。

3.2. 接受规则

对于草稿样本 $\tilde{x}$, 接受概率定义为

$$\alpha(\tilde{x}) = \min\left(1, \frac{p(\tilde{x})}{q(\tilde{x})}\right). \quad (7)$$

等价的实现方式是独立采样 $u \sim \text{Uniform}(0, 1)$ ，当 $u \leq \alpha(\tilde{x})$ 时接受。

在例子中，若 $\tilde{x} = A$ ，则

$$\alpha(A) = \min\left(1, \frac{0.40}{0.60}\right) = \frac{2}{3}. \quad (8)$$

若接受，直接令最终 token 为 $x = A$ 。一般地，token v 经由“草稿抽中且被接受”这条路径出现的概率为

$$q(v)\alpha(v) = q(v) \min\left(1, \frac{p(v)}{q(v)}\right) = \min(q(v), p(v)). \quad (9)$$

因此：

- 当 $q(v) > p(v)$ 时，只接受草稿给出的部分样本，将概率质量从 $q(v)$ 削减到 $p(v)$ ；
- 当 $q(v) \leq p(v)$ 时，该 token 一旦被草稿抽中就全部接受，但仅能贡献 $q(v)$ ，仍未达到目标所需的 $p(v)$ 。

3.3. 拒绝后的残差分布

若草稿 token 被拒绝，当前位置仍必须产生一个最终 token。此时不能从 p 直接重采，也不能选择 p 中概率最大的 token；正确做法是从残差分布采样：

$$r(v) = \frac{[p(v) - q(v)]_+}{\sum_{w \in \mathcal{V}} [p(w) - q(w)]_+}, \quad [z]_+ = \max(z, 0). \quad (10)$$

在上述例子中，

$$p - q = (-0.20, 0.05, 0.15), \quad (11)$$

$$[p - q]_+ = (0, 0.05, 0.15), \quad (12)$$

所以

$$r = (0, 0.25, 0.75). \quad (13)$$

这说明残差分布只补偿草稿模型低估的 token：A 已经通过接受路径获得目标所需的全部 0.40 概率质量，不应再获得机会；B 和 C 分别缺少 0.05 与 0.15，拒绝时释放的概率质量按这个缺口比例重新分配。

从 r 采出的 token 直接成为当前位置的最终 token，不再进行第二轮接受—拒绝判断。

当 $p = q$ 时，分母为零，但此时所有草稿样本都以概率 1 接受，拒绝分支不可达，因此无需构造 r 。实现中仍应显式处理这一数值边界。

3.4. 单位置流程

步骤	操作
1	草稿模型计算 q ，并采样 $\tilde{x} \sim q$ 。
2	目标模型计算同一上下文下的 p 。
3	以 $\min\left(1, \frac{p(\tilde{x})}{q(\tilde{x})}\right)$ 的概率接受。
4a	若接受，令 $x = \tilde{x}$ 。
4b	若拒绝，构造 $r \propto [p - q]_+$ ，采样 $x \sim r$ 。

4. 为什么最终分布严格等于目标分布

4.1. 完整证明

固定任意 token v 。它成为最终输出有两条互斥路径。

第一条路径是草稿模型抽中 v 且接受：

$$P(\tilde{x} = v, \text{accept}) = q(v)\alpha(v) = \min(q(v), p(v)). \quad (14)$$

所有草稿 token 的总接受概率为

$$P(\text{accept}) = \sum_v \min(q(v), p(v)). \quad (15)$$

因而总拒绝概率为

$$P(\text{reject}) = 1 - \sum_v \min(q(v), p(v)) = \sum_v [p(v) - q(v)]_+. \quad (16)$$

第二条路径是发生拒绝，并从残差分布抽中 v ：

$$P(\text{reject}, x = v) = P(\text{reject})r(v) = [p(v) - q(v)]_+. \quad (17)$$

两条路径相加可得

$$P(x = v) = \min(q(v), p(v)) + [p(v) - q(v)]_+ = p(v). \quad (18)$$

因此，对词表中的每个 v 都有 $P(x = v) = p(v)$ 。投机采样不是对目标分布的近似；在使用相同处理后分布、精确算术和正确随机规则的理想条件下，它与直接从目标模型采样具有相同分布。

4.2. 概率质量的分解

上述证明可以压缩为一个恒等式：

$$p(v) = \underbrace{\min(q(v), p(v))}_{\text{草稿 + 接受}} + \underbrace{[p(v) - q(v)]_+}_{\text{残差补偿}}. \quad (19)$$

它解释了为什么残差分布不能给 $q(v) > p(v)$ 的 token 再分配概率：这个 token 已经在第一项中得到完整的 $p(v)$ 。

4.3. 接受率与总变差距离

利用恒等式 $\min(a, b) = \frac{a+b-|a-b|}{2}$ ，单位位置的平均接受率为

$$P(\text{accept}) = \sum_v \min(p(v), q(v)) = 1 - \frac{1}{2} \sum_v |p(v) - q(v)|. \quad (20)$$

离散分布的总变差距离定义为

$$\text{TV}(p, q) = \frac{1}{2} \sum_v |p(v) - q(v)|, \quad (21)$$

所以

$$P(\text{accept}) = 1 - \text{TV}(p, q). \quad (22)$$

这给出一个直接结论：草稿分布越接近目标分布，接受率越高；若 $p = q$ ，接受率为 1；若两者支撑集完全不相交，接受率为 0。

5. 从一个 token 扩展到多个 token

5.1. 草稿模型连续提出候选

设当前前缀仍为 $x_{<t}$ 。草稿模型自回归地产生候选序列

$$\tilde{x}_t, \tilde{x}_{t+1}, \dots, \tilde{x}_{t+K-1}. \quad (23)$$

第 j 个候选不是独立生成的，而是服从

$$\tilde{x}_{t+j} \sim q_{t+j}(\cdot \mid x_{<t}, \tilde{x}_t, \dots, \tilde{x}_{t+j-1}). \quad (24)$$

每一步都要保存该位置草稿分布中已抽中 token 的概率；若发生拒绝并需要构造完整残差分布，还需要可访问该位置的完整 q_{t+j} 。

5.2. 目标模型为什么能一次验证

得到整段草稿后，目标模型以

已确认前缀 | d1 d2 ... dK

作为输入。因果注意力掩码保证每个位置只能看到左侧上下文，所以一次前向计算可以同时得到

$$p_t(\cdot \mid x_{<t}), p_{t+1}(\cdot \mid x_{<t}, d_1), \dots, p_{t+K}(\cdot \mid x_{<t}, d_1, \dots, d_K). \quad (25)$$

前 K 个分布用于验证 $d_1, d_2, \dots, d_K$ ；最后一个分布可在所有草稿 token 都接受时额外采样一个 token。实际实现必须正确处理 logits 与 token 的位置偏移。

这里并没有破坏自回归依赖：各位置的条件上下文仍然正确，只是候选 token 已知后，不同位置的 logits 能在加速器上并行计算。

5.3. 必须从左到右验证

验证顺序为 $d_1, d_2, \dots, d_K$ 。对第 j 个位置，使用与其草稿生成上下文一致的 p_j 和 q_j ，并以

$$\alpha_j = \min \left(1, \frac{p_j(d_j)}{q_j(d_j)} \right) \quad (26)$$

的概率接受。

- 若 d_j 被接受，它成为正式 token，继续验证 d_{j+1} 。
- 若 d_j 被拒绝，从 $r_j \propto [p_j - q_j]_+$ 采出替代 token，然后停止本轮验证。

第一次拒绝之后，所有更靠后的草稿 token 都必须丢弃。若 d_j 被替换为 d'_j ，后续草稿原本依赖的上下文包含 d_j ，而真实上下文已经变为 d'_j ；继续使用后续草稿会违反条件分布。

例如草稿序列为

A	B	C	D
✓	✓	✗	

则 A, B 被确认，在 C 所在位置从残差分布采出 C' ，本轮最终推进 A, B, C' ； D 作废，下一轮从新前缀重新开始。

5.4. 全部接受时的额外 token

若 K 个草稿 token 全部接受，目标模型已经计算出跟在整段草稿之后的分布 p_{t+K} ，因此可直接从该分布额外采样一个 token。此时一次目标模型验证最多推进 $K + 1$ 个 token。

这个额外 token 必须直接来自目标分布，不需要草稿校正。它也使每轮至少产生一个新 token，并简化算法的推进逻辑。

5.5. 多 token 算法

下面的伪代码省略 KV cache 管理和批处理细节：

```
while not finished:
    prefix = 已确认序列

    # Draft 阶段：串行提出 K 个候选
    for j in 1..K:
        q[j] = Draft(prefix + d[1:j-1])
        d[j] ~ q[j]

    # Target 阶段：一次前向计算得到 K+1 个位置的分布
    p[1:K+1] = Target(prefix + d[1:K])

    for j in 1..K:
        u ~ Uniform(0, 1)
        if u <= min(1, p[j](d[j]) / q[j](d[j])):
            append d[j]
        else:
            r = normalize(max(p[j] - q[j], 0))
            x ~ r
            append x
            discard d[j+1:K]
            begin next iteration

    # 只有 K 个候选全部接受才执行
    x ~ p[K+1]
    append x
```

6. 多位置算法为何仍然正确

单位置证明是在一个固定上下文上成立的。多 token 验证从左向右进行：

- 第一个位置经校正后严格服从目标条件分布；
- 条件于第一个位置被接受，第二个位置所用的上下文正是目标模型所需的上下文，单位置证明再次成立；
- 依此类推，直到第一次拒绝或全部接受。

因此，每个真正提交的 token 都服从目标模型在已提交前缀下的条件分布。根据概率的链式法则，整段输出序列服从目标模型的联合分布：

$$P(x_t, \dots, x_{t+m} \mid x_{<t}) = \prod_{j=0}^m p_{t+j}(x_{t+j} \mid x_{<t+j}). \quad (27)$$

这也是为什么“第一次拒绝后停止验证”不是实现细节，而是正确性条件。

7. 加速从哪里来

普通解码用一次昂贵的目标模型调用推进一个 token。投机解码希望做到

$$\text{一次目标模型验证} \rightarrow \text{确认多个 token.} \quad (28)$$

草稿阶段仍是串行的，但草稿模型更小、更快；目标模型对较短候选段的验证通常能更充分地利用并行硬件。如果一轮在第一次拒绝前接受了 L 个草稿 token，那么该轮通常推进 $L + 1$ 个 token：拒绝时包含一个残差 token，全部接受时包含一个额外的目标 token。

收益取决于以下因素：

- 草稿模型的运行成本；
- 目标模型验证 $K + 1$ 个位置相对单步解码的成本；
- 各位置接受率以及接受率随位置的变化；
- 候选长度 K ；
- KV cache 的构造、回滚和内存开销；
- 硬件、batch size、并行策略和采样实现。

若粗略假设每个位置接受概率都为常数 a ，则一轮提交 token 数的期望为

$$E[N] = 1 + a + a^2 + \dots + a^K = \begin{cases} \frac{1-a^{K+1}}{1-a} & \text{if } a \neq 1 \\ K + 1 & \text{if } a = 1. \end{cases} \quad (29)$$

这里要求接受事件可用同一个常数 a 近似，实际模型并不严格满足这一假设，但该公式能说明权衡： K 太小，单轮推进有限； K 太大，后半段候选可能经常因较早拒绝而白算。

8. 正确性边界与实现注意事项

8.1. “完全一致”具体指什么

严格一致指的是**输出概率分布一致**，并不保证使用相同随机种子时逐 token 得到完全相同的样本路径。投机算法消耗随机数的方式不同，通常不会复现普通解码的同一条具体序列。

结论还依赖以下前提：

- p 与 q 定义在同一词表上；
- 验证时使用的 p_j 、 q_j 与候选生成时的上下文严格对应；
- 两个分布经过相容的约束和处理，例如 temperature、top- k 、top- p 、禁止 token 掩码；
- 接受概率和残差分布按算法计算；
- 浮点误差、量化误差和并行归约差异没有造成有意义的偏移。

如果先对 logits 做截断或重归一化，那么公式中的 p 、 q 应当是处理后的合法分布，而不是未经处理的原始 softmax 分布。

8.2. 采样解码与贪心解码

本文的残差分布证明针对随机采样。贪心解码的目标是 $\arg \max_v p(v)$ ，可以采用更简单的验证规则：草稿 token 与目标模型的贪心 token 相同则接受，否则采用目标 token 并停止本轮。两种模式的正确性目标不同，不能把随机采样的概率比规则与贪心匹配规则混用。

8.3. 数值与系统实现

- 计算 $\frac{p(d)}{q(d)}$ 时应避免除零并控制低精度下的上溢、下溢；草稿实际抽中的 token 理论上满足 $q(d) > 0$ 。
- 对 $[p - q]_+$ 归一化时要处理舍入导致的极小负数或零和。
- 第一次拒绝后要回滚未提交草稿对应的 KV cache 或逻辑长度。
- EOS 也属于词表 token；一旦被正式提交，生成应按普通解码的终止规则结束。
- 是否获得墙钟时间加速必须通过目标硬件和真实负载测量，接受率高本身并不等价于吞吐或延迟一定更好。

9. 总结

投机解码可以归纳为三层：

1. **提议**：草稿模型从 q 连续采样多个候选。
2. **校正**：目标模型以 $\min(1, \frac{p(d)}{q(d)})$ 的概率接受候选；拒绝时从 $[p - q]_+$ 的归一化分布补偿。
3. **并行验证**：因果掩码使目标模型一次得到多个位置的分布，从而让一次昂贵调用推进多个 token。

数学核心是

$$p(v) = \min(p(v), q(v)) + [p(v) - q(v)]_+, \quad (30)$$

工程核心是

$$\text{便宜的串行草稿} + \text{昂贵但并行的整段验证}. \quad (31)$$

前者保证输出分布与目标模型一致，后者才提供潜在加速。

10. 参考资料

- Leviathan, Kalman, and Matias, Fast Inference from Transformers via Speculative Decoding, 2023.
- Chen et al., Accelerating Large Language Model Decoding with Speculative Sampling, 2023.
- [投 k 机解码视频讲解](#)。

11. 问题

1. kv cache 为什么不去缓存 $Q * K$ 的结果？
2. 可以思考下，如果当前计算到了 A 了，本来就是用 A 跑一次，然后获取到下一个，但是如果同时跑 B C D E，中间多获取的矩阵的结果是什么？