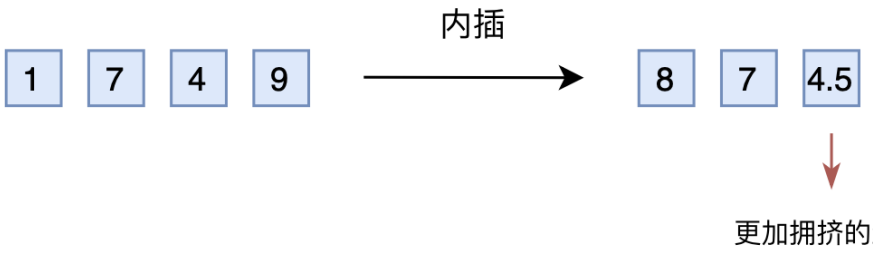
Long context2- 插值

将推理时的位置索引进行下采样或缩放，就是把2k的位置编码对应到1k。通过这种方式，推理时的位置索引被映射回了模型训练时的范围内，从而帮助模型更好地处理这些原本超出其处理能力的输入序列。

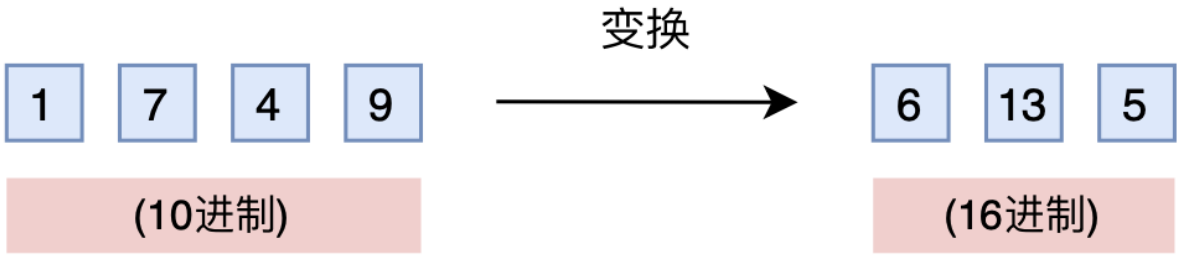
线性内插

比如通过除以2，将4位转成3位，导致的结果是最后一位更加拥挤，相邻数字的差距变成了0.5。虽然经过微调后效果不会明显下降，但是当处理范围进一步增大时，相邻数字差异更小，并且相邻差异只集中在个位数，其他位相邻差异仍是1，导致维度之间分布不一样，增大模型学习难度。



进制转换

可以通过进制转换，既不用新增维度，也可以保持相邻间距。比如采用16进制取代10进制。



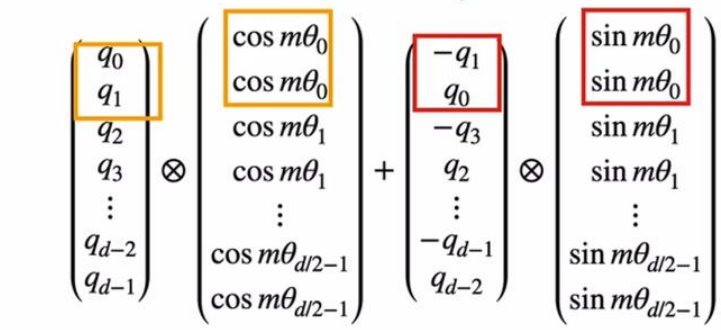
重新思考RoPE

首先给出苏神的定义：位置m的旋转位置编码(RoPE)，本质上就是数字m的 β 进制编码。

举个例子：给定一个10进制的数字m，求其 β 进制的从右往左数的第n位数字，采用如下公式：

$$ceil(\frac{m}{\beta^{n-1}})\ mod\ \beta$$

又知道RoPE的定义中有以下cos序列（sin也同理）：$[cos\ m\theta_0, cos\ m\theta_1, ..., cos\ m\theta_{d/2-1}]$



$$\theta_i = 10000^{-2i/d}, i \in [1,2,...,\frac{d}{2}]$$

d 一定是偶数

$$cos(\frac{m}{10000^{2i/d}}) = cos(\frac{m}{10000^{(2/d)*i}}) = cos(\frac{m}{\beta^i}), \beta = 10000^{(2/d)}$$

cos序列就可以表示为：$[cos\ \frac{m}{\beta^0}, cos\ \frac{m}{\beta^1}, ..., cos\ \frac{m}{\beta^{d/2-1}}]$，将其翻转过来：$[cos\ \frac{m}{\beta^{d/2-1}}, ..., cos\ \frac{m}{\beta^1}, cos\ \frac{m}{\beta^0}]$。

至于模运算，它的最重要特性是周期性，cos刚好也是周期函数。所以，除掉取整函数这个无关紧要的差异外，RoPE其实就是数字m的 β 进制编码！

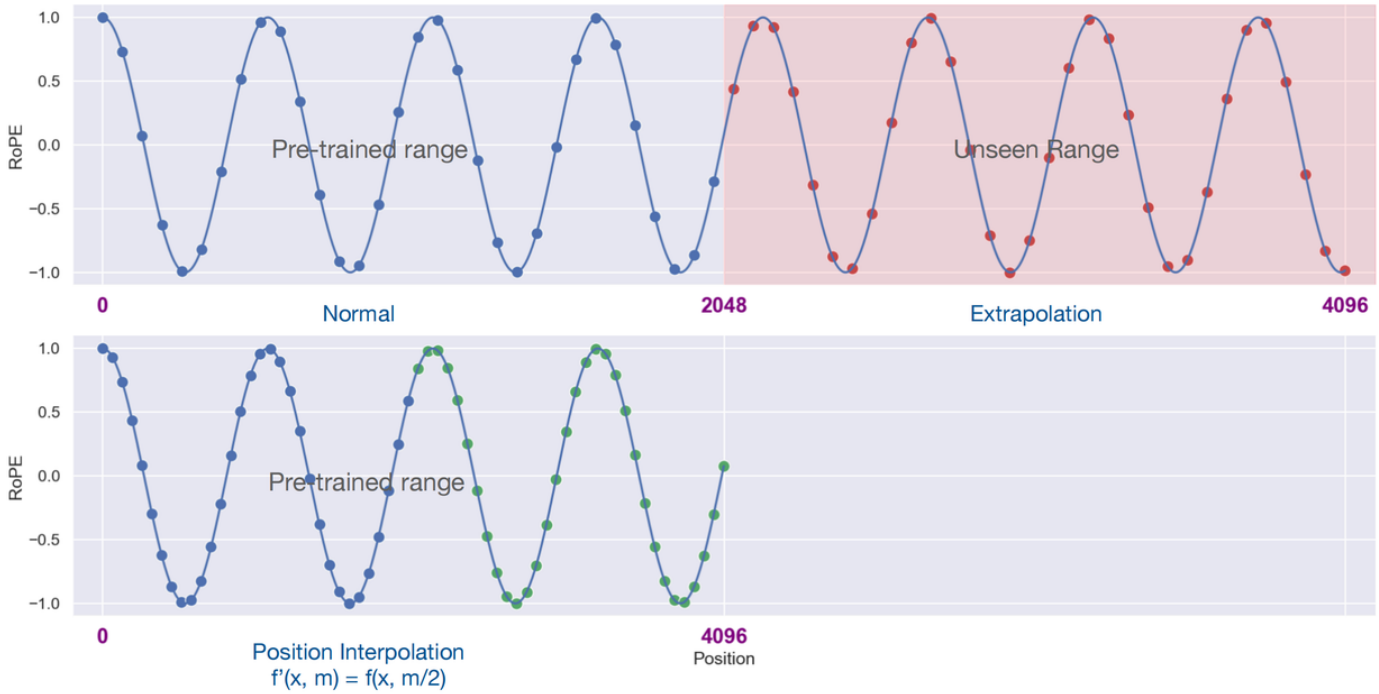
基于以上结论，后续介绍的内插就是将m换成m/k，k是推理时需要扩大的倍数；NTK插值则是将10000换成10000k。

Position Interpolation 位置插值

- 参考文章：Extending context window of large language models via positional interpolation

PI通过直接将位置索引缩小，这对于RoPE等位置编码更合适，并且可能需要较少的训练，因为没有添加可训练参数，使得最大位置索引与预训练阶段的上下文窗口限制相匹配。其本质就是在相邻的整数位置上插值位置索引，因为位置索引可以应用在非整数的位置上(而非在训练位置外进行外推)。

- 区别于线性内插，PI仍保留了4096个位置索引，只不过索引之间距离变成0.5，而前者只有2048个索引，多出来的索引被压缩到了最后一位。



使用PI后，位置m缩放成了 $\frac{mL}{L'}$ ，L是训练的上下文长度(2048)，L'是推理时需要扩展到的长度(4096)，对应的q和k计算时变成了 $f(x, \frac{mL}{L'})$ 。使用PI后的微调只需要少量用例且对用例不敏感，原因在于模型在微调阶段仅适应新的上下文窗口，从良好的初始化开始，而不是获取新的知识。只需要进行1000步对微调就能显著降低ppl。

Model Size	Context Window	Number of fine-tuning steps 0	200	400	600	800	1000
7B	8192	16.10	7.12	7.10	7.02	6.99	6.95
7B	16384	112.13	7.05	6.93	6.88	6.84	6.83

插值法存在的问题：与RoPE一起使用时，RoPE中的每个维度 $sin\ m\theta_j, \theta_j = 10000^{-\frac{2j}{d}}, j \in [0, 1, ..., d/2-1]$ ，其周期为 $\frac{2\pi}{m}10000^{\frac{2j}{d}}$ ，对于维度较低的j，其对应的周期比较小，频率较高。对于这种维度，插值后会变得很拥挤（本来一个周期包含10个值，但是内插之后能包含20个值）。

NTK-aware插值

核心思想是：高频外推，低频内插。

对于 $[cos\ \frac{m}{\beta^0}, cos\ \frac{m}{\beta^1}, ..., cos\ \frac{m}{\beta^{d/2-1}}]$ ， $\beta = 10000^{(2/d)}$ ，将最后面的低频项引入 λ 变成 $\frac{m}{(\beta\lambda)^{d/2-1}}$ ，为了与内插法一致（内插就是将n换成n/k，其中k是要扩大的倍数），有 $\frac{m}{(\beta\lambda)^{d/2-1}} = \frac{m/k}{\beta^{d/2-1}}$ ，解得 $\lambda = k^{2/(d-2)}$ 。（即上文提到的“NTK插值则是将10000换成10000k”）

而对于最高频项 $cos\ \frac{m}{\beta}$ ，引入 λ 变成 $\frac{m}{\beta\lambda}$ ，但由于d一般很大，λ很接近于1，所以还是接近于 $\frac{m}{\beta}$ ，等价于外推。

NTK插值存在的问题

1. 由于它不仅仅是一种插值方案，一些维度被轻微外推到“超出边界”的值，因此使用“NTK-aware”插值进行微调的结果不如PI。
2. 此外，由于存在“越界”值，理论尺度因子 k 并不能准确描述真实的上下文扩展尺度。在实践中，对于给定的上下文长度扩展，尺度值 k 必须设置得高于预期尺度

NTK-by-parts插值

波长：维度d上嵌入的RoPE，执行完整旋转(2Π)所需的token长度：

$$\lambda_d = \frac{2\pi}{\theta_d} = 2\pi b^{\frac{2d}{D}}, \theta_d = 10000^{-\frac{2d}{D}}$$

- [ ] “盲”插值方法不关心不同维度对应的不同波长，比如像PI和“NTK-aware”插值，对所有RoPE维度的没有做针对性的处理(因为它们对网络有相同的影响)，而其他方法(如YaRN)，定义为“有针对性的”插值方法。

对RoPE有以下观察：

- 给定上下文大小L，有一些维数d的波长长于预训练期间看到的最大上下文长度($\lambda > L$)，这表明一些维数的嵌入可能在旋转域中不均匀分布。当波长很长时，这些维度上的嵌入几乎不变，可以认为它们保持了绝对位置信息，即每个位置的嵌入不因相对位置变化而变化；当波长较短时，嵌入会在较短的距离内完成多次旋转，这使得这些维度上的嵌入反映的是相对位置信息，即它们可以捕捉到标记之间的相对距离变化。
- 采用RoPE进行拉伸时，所有的token变得更彼此接近，因为 $a \cdot b = ||a||\ ||b||cos(\theta)$ ， θ 减小会导致两个向量的内积变大，变得更加接近。从而损害模型处理邻近token位置时的性能。

为了解决以上问题，对高频率的维度不插值，对更低频率的维度插值。

- 如果波长 λ 比上下文长度 L 小得多，此时不插值
- 如果波长 λ 等于或大于上下文长度 L ，此时只做插值，不做任何外推
- 两者之间的维数可以兼备

定义比率 $r_d = \frac{L}{\lambda_d}$ ，

比率 r=Lλ，且维数为d时，比率 r 以如下方式依赖于 d：

- 如果 $r(d) < \alpha$，比如$\alpha = 1$，意味着**波长大于上下文长度**，则将**线性插入一个尺度** s (完全像PI，避免任何外推)
- 至于如果是 $r(d) > \beta$，则不插值

接下来，定义斜坡函数 γ：$\gamma(r) = \begin{cases} 0, & \text{if } r < \alpha \\ 1, & \text{if } r > \beta \\ \frac{r-\alpha}{\beta-\alpha}, & \text{otherwise} \end{cases}$

NTK-by-parts插值是对RoPE的一种修改，$h(\theta_d) = (1 - \gamma(r(d)))\frac{\theta_d}{s} + \gamma(r(d))\theta_d, s = \frac{L'}{L}$

动态插值

固定缩放因子s可能会导致，模型在长度小于 L 时可能出现性能折扣，当序列长度大于 L′ 时可能出现突然退化。因此提出动态缩放，在每次前向传递中，位置嵌入更新缩放因子 s=max(1,l′/L)，其中 l′ 是当前序列的序列长度。

参考：

- [大模型长度扩展综述：从直接外推ALiBi、插值PI、NTK-aware插值(对此介绍最详)、YaRN到S2-Attention(https://www.cnblogs.com/mudou/p/18309199#321-ntk-by-parts-%E6%8F%92%E5%80%BC%E6%AD%A5%E9%AA%A4)]
- [Long-Context LLM综述(https://blog.csdn.net/Cyril_KI/article/details/139573263)]