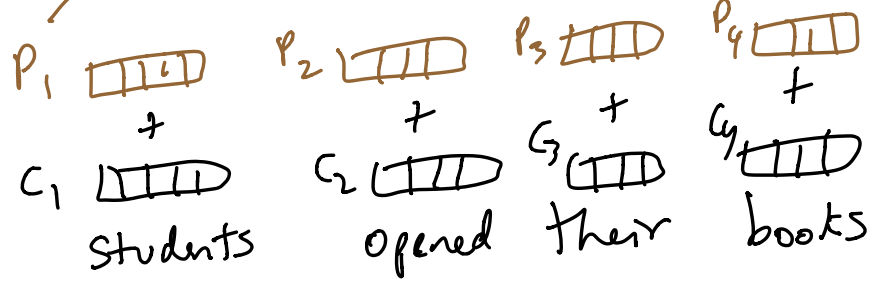
Positional encoding:

↳ self-attn doesn't have any notion of order

fixed vectors (orig. Transformer paper)
learned pos. embs (BERT)



↳ inject positional info via absolute and additive embs

$$q_{\text{students}} = W_q \cdot (c_{\text{students}} + p_1)$$

↳ simple, helps w/ tasks that depend on absolute position

↳ hard to generalize to seq. longer than seen during training

↳ what about relative position

students	opened	their	books
1	2	3	4

noisy	students	opened	their	books
1	2	3	4	5

absolute vs. relative position embs

↳ encode relative distance between two tokens

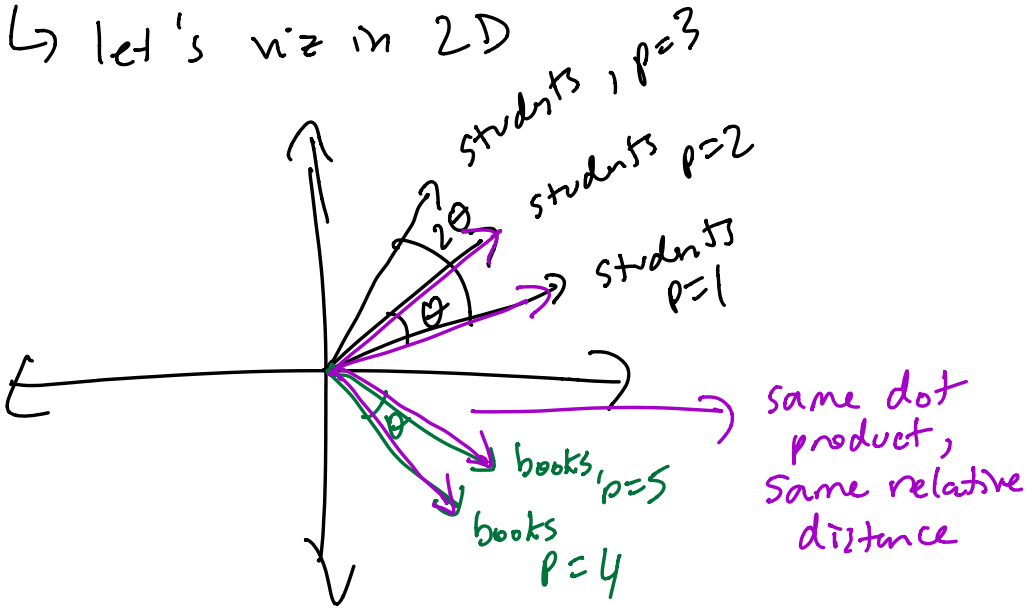
e.g. books and students are 3 words apart
4-1, 5-2

↳ lose info about absolute positions

RoPE: rotary position encoding

↳ don't add, rotate!

↳ let's viz in 2D



how do we rotate a vector?

↳ multiply by a rotation matrix

$$W_{R_{\theta,p}} = \begin{bmatrix} \cos(p\theta) & -\sin(p\theta) \\ \sin(p\theta) & \cos(p\theta) \end{bmatrix}$$

rotation freq, position in seq

how do we integrate this into self-attn?

absolute: $q_{student} = W_q(c_{student} + p_1)$

RoPE:

$$q_{student} = \underbrace{W_{R_{\theta,p=1}} \cdot W_q} \cdot c_{student}$$

$$k_{books} = \underbrace{W_{R_{\theta,p=4}} \cdot W_k} \cdot c_{books}$$

attn score = $q_{student} \cdot k_{books}$

due to properties of the rotation matrix

$$= \left(W_q c_{student}\right)^T \cdot W_{R_{\theta, 4-1}} \cdot \left(W_k c_{books}\right)$$

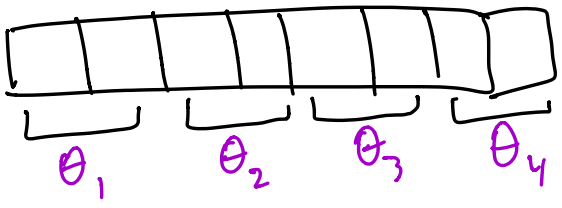
$= 3$

depends only on diff, in the two positions, not absolute position

↳ same distance, same dot product regardless of abs. position

---

how do we generalize to > 2d embeddings



each of these θ_i is a constant that controls rotation freq

↳ higher θ = faster spinning

↳ more sensitive to position changes

↳ lower θ = slow spinning
  ↳ less sensitive to position
  ↳ encode more semantic content

→ mix both high and low θ
to benefit from abs. vs. relative pos. embeddings

What if we want to extend RoPE to longer seqs than observed in training?

↳ train on 2K tokens
↳ test on 4K tokens
↳ can we just use $W_{R\theta,4K}$
↳ empirically, no, model does not generalize

↳ simple trick: position interpolation

$$p_{new} = p \cdot \left[\frac{\text{training length}}{\text{test length}} \right] \frac{1}{2}$$

p_{new}: 2000; p: 4000

↳ basically squeeze the long test seq into the trained range