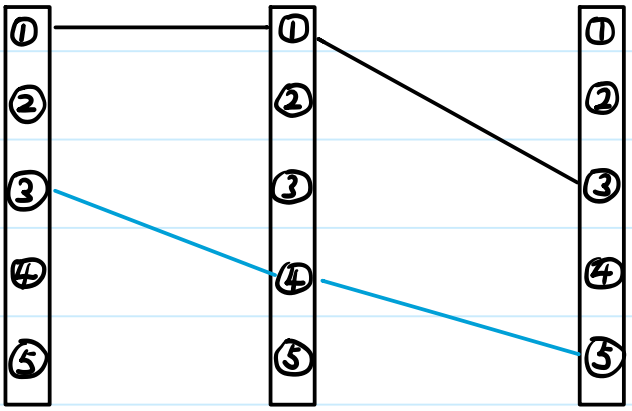
# Deep Retrieval

Sunday, March 17, 2024 4:34 PM

**Index**: item is represented as a path.



depth = 3

width = 5

item = {1, 1, 3}

item = {3, 4, 5}

idex 1: item → List<path>: an item is represented by a path.

index 2: path → List<item>: a path may correspond multiple items.

**Estimated like score**:

① path = [a, b, c]

② given user feature $\vec{x}$, estimate like on node a: $P_1(a|\vec{x})$

③ given $\vec{x}$ and a, estimate like score on node b: $P_2(b|a;\vec{x})$

④ given $\vec{x}$, a, b, estimate like score on node c. $P_3(c|a;b;\vec{x})$

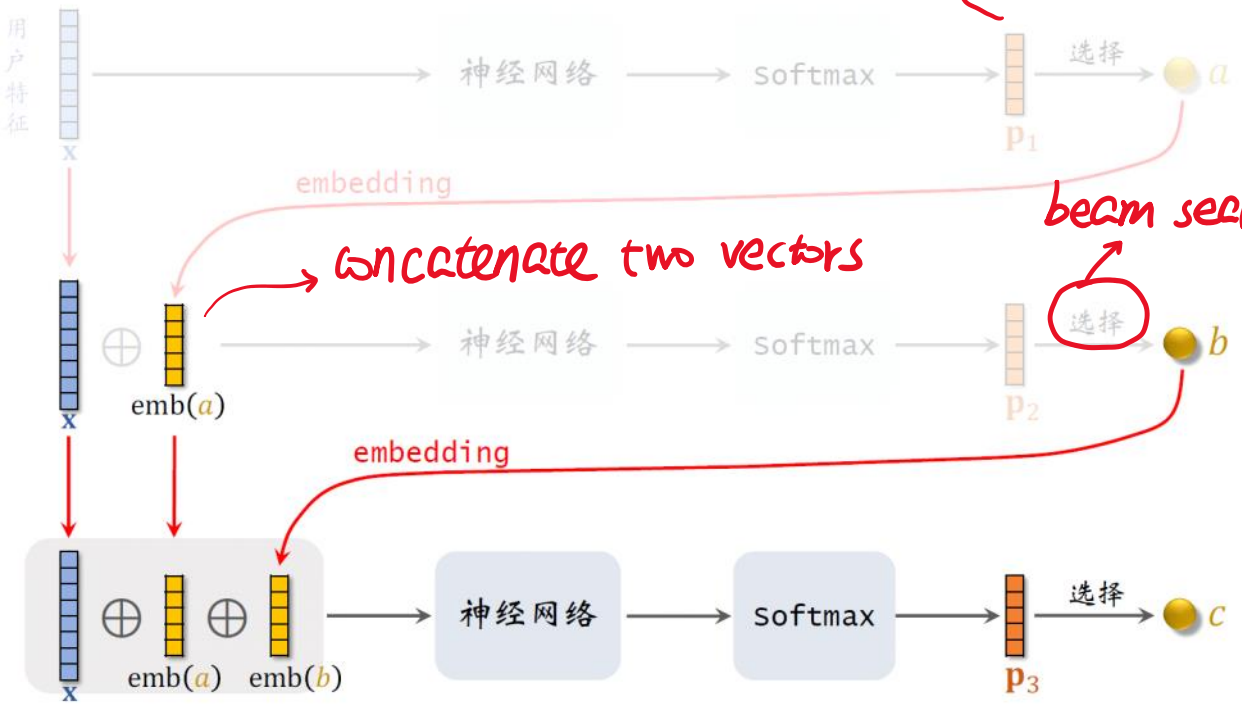
⑤ overall, like score on [a, b, c]:

$$P(a,b,c|\vec{x}) = P_1(a|\vec{x}) \cdot P_2(b|a;\vec{x}) \cdot P_3(c|a;b;\vec{x})$$

# Deep Retrieval

Sunday, March 17, 2024 4:34 PM

same length as width of path




(Picture from Shusen Wang on YouTube/Bilibili)

concatenate three vectors

Online Retrieval: user → path → item

① given user feature, use beam search to get a set of paths

② use index "path → List<item>" to find 1st batch of items

③ compute like score and rank: get a subset.

# Deep Retrieval

Sunday, March 17, 2024 4:34 PM

## Beam Search:

① Path breadth = $K$; depth = 3 ⟶ total $K^3$ paths

② using neural network to score $K^3$ is too expensive

③ use beam search to reduce computation load

hyper-parameter: beam size

## How does beam search work?

① set beam size = 1; path breadth = $K$.

② from $j^{th}$ layer to $(j+1)^{th}$ layer:

calculate $K$ scores

select 1 node whose score is highest

③ repeat step ② for every layer

optimal path:

$$[a^*, b^*, c^*] = \underset{a,b,c}{\operatorname{argmax}} \; P(a, b, c \mid \vec{x})$$

beam size = 1 is actually greedy algorithm

it cannot generate global optimal path

More generally, if beam size = $B$ ⟶ path depth

Then total calculation is $O(B^3)$

# Deep Retrieval

Sunday, March 17, 2024 4:34 PM

## Offline Training:

① learn parameters of NN for $P(a,b,c|\vec{x})$

② learn item representation:

item → List <path>

path → List <item>

③ Only rely on postive samples: user clicked item

④ item representation: an item → J paths

$[a_1, b_1, c_1], [a_2, b_2, c_2], \cdots [a_j, b_j, c_j]$

⑤ if user clicks the item:

make $\sum_{j=1}^{J} P(a_j, b_j, c_j|x)$ as large as possible

⑥ loss function:

$$loss = -\log\left(\sum_{j=1}^{J} P(a_j, b_j, c_j|x)\right)$$

⑦ user like score on patch $[a,b,c]$

$$P(path|user) = P(a,b,c|x)$$

⑧ item and path correlation:

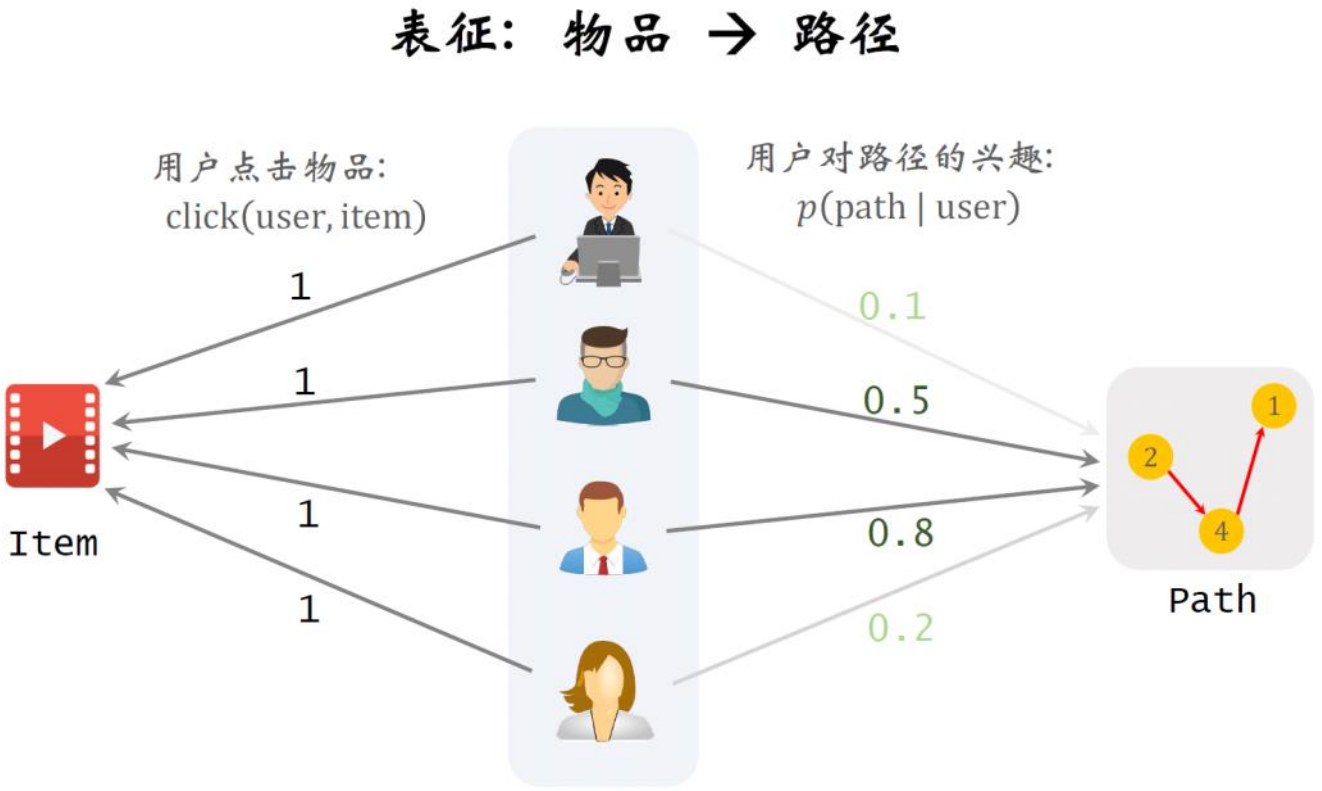
$$score(item, path) = \sum_{user} P(path|user) \cdot click(user, item)$$

user: like score to path; click(user, item): whether click (0 or 1)

⑨ select J paths as item represenation from score

# Deep Retrieval

Sunday, March 17, 2024 4:34 PM




(Picture from Shusen Wang on YouTube/Bilibili)

⑩ J paths : $\Pi = \{path_1, \cdots path_J\}$

⑪ loss function :

$$loss(item, \Pi) = -\log\left[\sum_{j=1}^{J} score(item, path_j)\right]$$

⑫ regularization :

$$reg(path_j) = (\#\ of\ items\ on\ path_j)^4$$

# Deep Retrieval

Sunday, March 17, 2024 4:34 PM

## Summary of item representation:

① assume items are represented as J paths $\pi$

② every time fix $\{path_i\}_{i \neq l}$; select a new $path_l$

$$path_l \leftarrow \underset{path_l}{\text{argmin}}\ loss(item, \pi) + \alpha \cdot reg(path_l)$$

③ the path selected has high score

meanwhile the path does not have too many items.

## Summary of Training:

### update Neural Network

① like score: $p(path|x)$

② data: (a) item → path

(b) clicked items

③ if user clicked item and item correspond to path

update NN parameters to increase $p(path|x)$

### update item representation

① correlation between item & path

item ← user → path

clicked item like score from NN

② let each item correlates J paths

(a) item and path are highly correlated

(b) single path cannot have too many items.