

Embedding

1. 定义(什么是 embedding?)

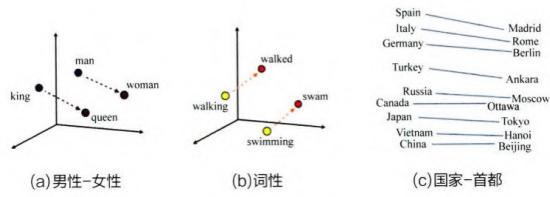


图 4-1 词向量举例

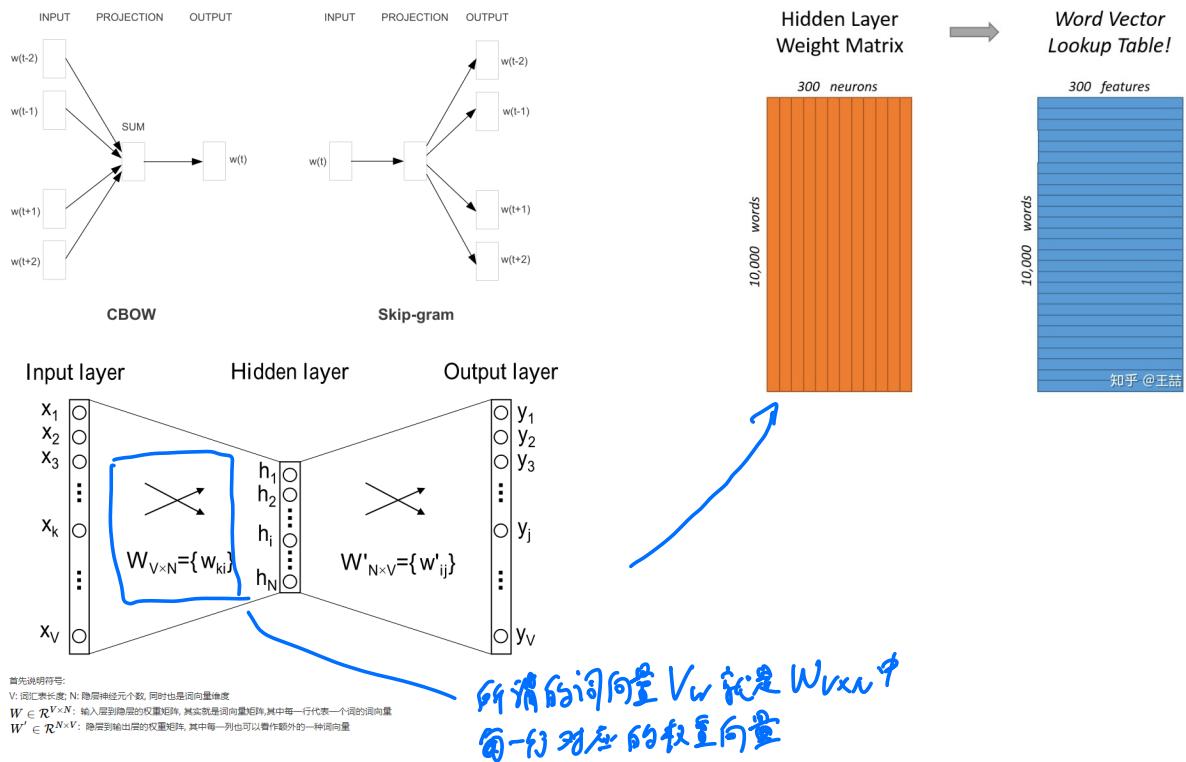
What's embedding?

- ① one-hot 编码高维 \rightarrow embedding 稀疏低维
- ② Embedding 值高，在深度学习中，常与其他特征 concatenate 后输入层/全连接层
- ③ & Locality-Sensitive Hashing 技术，常用于 matching 阶段。

方法

1. Word2Vec

几张重要的图



Word Embedding

one-hot encoding $\xrightarrow{\text{drawback}}$ ignore the relationship among the word

t-SNE $300d \xrightarrow{\text{embedding}} 2d$

to visualize word in 2-d space

distance of similar feature is close

所以 \rightarrow 通过 t-SNE 把 embedding 放一起

Properties

类似物理

$$C_{\text{man}} - C_{\text{woman}} = C_{\text{king}} - (\underline{C_{\text{queen}}})$$

所以 $C_{\text{king}} + C_{\text{woman}} = C_{\text{man}}$

\Rightarrow Find word w_i , $\arg \max_w \text{similarity}(C_w, C_{\text{king}} + C_{\text{woman}} - C_{\text{man}})$

其中 similarity of $\hat{U}(\hat{V})$ Cosine similarity $\frac{U^T V}{\|U\|_2 \|V\|_2}$
如果 $\|U\|=1$, $\|V\|=1$, 那么 $U^T V \hat{=} \hat{U}^T \hat{V}$ 即 $\hat{U}^T \hat{V}$ 为相似度

Embedding matrix

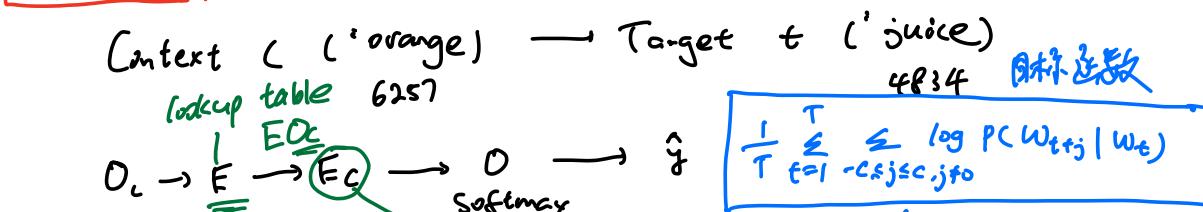
how to select certain feature from embedding matrix? $E \times D$:

$$\begin{matrix} \text{apple} & \text{orange} \\ \vdots & \vdots \\ 300 & \end{matrix} \quad \boxed{E} \quad \left[\begin{array}{c} \text{apple} \\ \vdots \\ 300 \end{array} \right] \times \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{array} \right]_{(300 \times 1)} = \boxed{\text{out-hot vector of certain feature}}$$

★ Learning word embedding

word2vec { skip-gram
negative sampling
CBOW (单向)逐单词作为向量的模型 (双向)
Glove

Skip-gram (用输出单词作为中心单词预测周边单词)



④ V_w 代表词 w , 词与词之间的内积距离 $V \cdot V^T$ 代表词义近似程度.

④ softmax は？

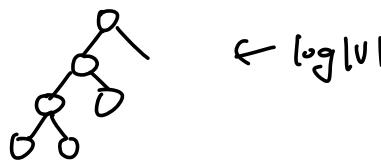
$$\max_{\mathbf{W}_0} \ln \text{softmax} : \quad p_{\text{L}}(\mathbf{W}_0 | \mathbf{W}_1) = \frac{\exp(V_{W_0}^T V_{W_1})}{\sum_{W_2} \exp(V_W^T W_2)}$$

$p(W_{t+3} | W_t)$ 的定义

Problem:

① Computational Speed

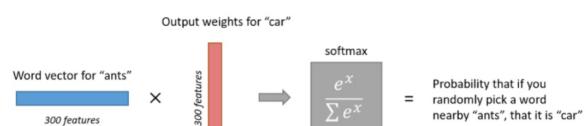
Use Hierarchical softmax (= 分枝 softmax)



② How to sample the context C ?

$\#$ uniform sampling. the. a. an efficient?

\Rightarrow 所以一般不会均匀年样



Negative Sampling

I often eat cereal with my cereals.

Example: I want a glass of orange juice to go along with my cereal

Context	word	target?	
orange	juice	1	← 首个标记为1，其余记为0
orange	king	0	
orange	book	0	
orange	of	0	

$k = 5 \approx 20$
 $k = 2 \approx 5$ large dataset

$$\text{softmax } p(c|t) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{|V|} e^{\theta_j^T e_c}}$$

Context word target
 \uparrow \uparrow \uparrow
 c t y

$$p(c|t) = \sigma(\theta_t^T e_c)$$

A set of binary classifiers using logistic regressions that aim at assessing how a given context and a given target words are likely to appear simultaneously.

Question: how to choose negative example?

$$p(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{|V|} f(w_j)^{3/4}}$$

heuristic algorithm

GloVe

define $X_{ij} =$ the number of times i appear in the context of j
 $X_{ij} = X_{ji}$

Model.

$$\text{minimize } \sum_{i=1}^V \sum_{j=1}^V f(X_{ij}) (\theta_i^T e_j + b_i + b_j - \log X_{ij})^2$$

$$e_w^{(\text{final})} = \frac{e_w + \theta_w}{2}$$

2. Item2Vec

假设 Item2Vec 中一个长度为 K 的用户历史记录为 w_1, w_2, \dots, w_k . 类似 Word2Vec, Item2Vec 的优化目标为

$$\frac{1}{K} \sum_{i=1}^K \sum_{j \neq i}^K \log p(w_j | w_i)$$

双塔模型

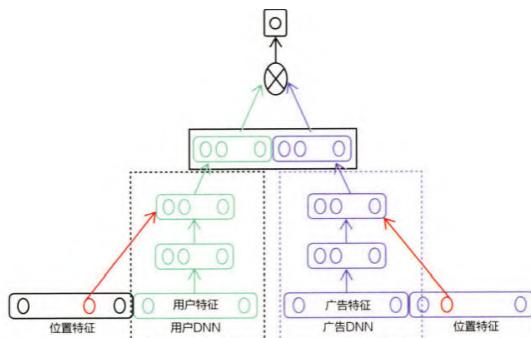


图 4-5 双塔模型

局限性：只能利用序列型数据。

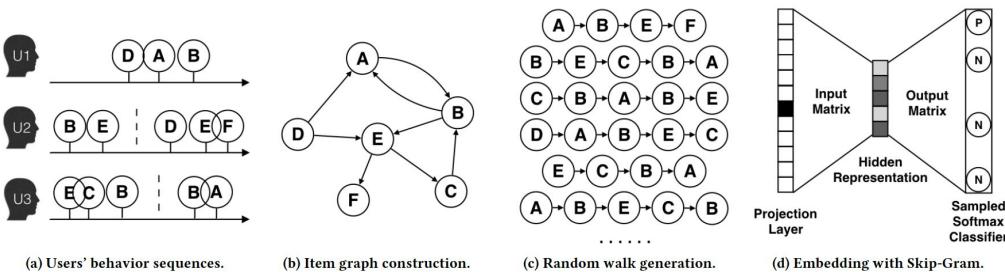
3. Graph Embedding

Intuition: 在互联网场景下，数据对象之间更多为图结构。Word2Vec 和 Item2Vec 却是化简为“序列”。

What: 对图结构中的节点进行 Embedding，获得生成的图上 Embedding 后能一定程度地保留图的结构信息和语义信息的局部相似性。

经典方法

i. DeepWalk



分为四步

(a) 用原始行为序列

(b) 基于(a)的序列，构造 Item 的图。如有重叠的，可设置权重表示程度。

(c) 在(b)的图上随机游走，得到一个物品序列

(d) 将这些物品序列放入 skip-gram 中训练，生成物品的 embedding。

其中，和(c)是，如果有向有权图，从节点 V_i 跳转到 V_j 的概率：

$$P(V_j | V_i) = \begin{cases} \frac{M_{ij}}{\sum_{j \in N_+(V_i)} M_{ij}} & , V_j \in N_+(V_i) \\ 0 & , e_{ij} \notin E \end{cases}$$

其中 E 是物品关系图所有边的集合

$N_+(V_i)$ 是节点 V_i 所有出边的集合

M_{ij} 是节点 V_i 到 V_j 的权重

$P(V_j | V_i)$ 即 跳转边权重 / 所有出边权重之和

2. Node2Vec — 同质性(homophily) 和 结构性(structural equivalence) 算法

同质性：相近节点的 embedding 应尽量相似

结构性：跨网上相似节点的 embedding 应尽量相似

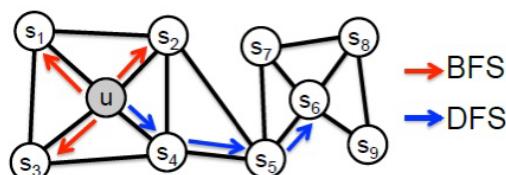


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

e.g. 用同质性， s_1, s_2, s_3, s_4 for embedding 应相似

用结构性， u 和 s_6 for embedding 应该相似

为了使 Graph Embedding 的结果能够表达网络的“结构性”，在随机游走的过程中，需要让游走的过程更倾向于 BFS，因为 BFS 会更多地在当前节点的邻域中游走遍历，相当于对当前节点周边的网络结构进行一次“微观扫描”。当前节点是“局部中心节点”，还是“边缘节点”，或是“连接性节点”，其生成的序列包含的节点数量和顺序必然是不同的，从而让最终的 Embedding 抓取到更多结构性信息。

结构性 \rightarrow BFS
同质性 \rightarrow DFS

另外，为了表达“同质性”，需要让随机游走的过程更倾向于 DFS，因为 DFS 更有可能通过多次跳转，游走到远方的节点上，但无论怎样，DFS 的游走更大概率会在一个大的集团内部进行，这就使得一个集团或者社区内部的节点的 Embedding 更为相似，从而更多地表达网络的“同质性”。

Question: 如何控制什么层级 DFS 和 BFS?

Answer: 通过节点间的跳转概率。

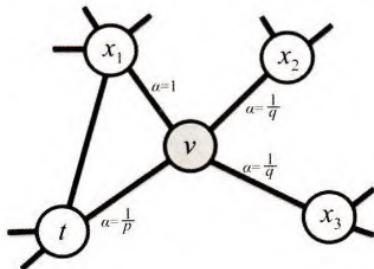


图 4-9 Node2vec 的跳转概率

从 v 跳转到下一个结点 x 的概率

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

其中 $\alpha_{pq} = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$, w_{vx} 是边 vx 的权重,

其中, d_{tx} 指节点 t 到节点 x 的距离, 参数 p 和 q 共同控制着随机游走的倾向性。参数 p 被称为返回参数 (return parameter), p 越小, 随机游回节点 t 的可能性越大, Node2vec 就更注重表达网络的结构性。参数 q 被称为进出参数 (in-out parameter), q 越小, 随机游走到远方节点的可能性越大, Node2vec 就更注重表达网络的同质性; 反之, 则当前节点更可能在附近节点游走。

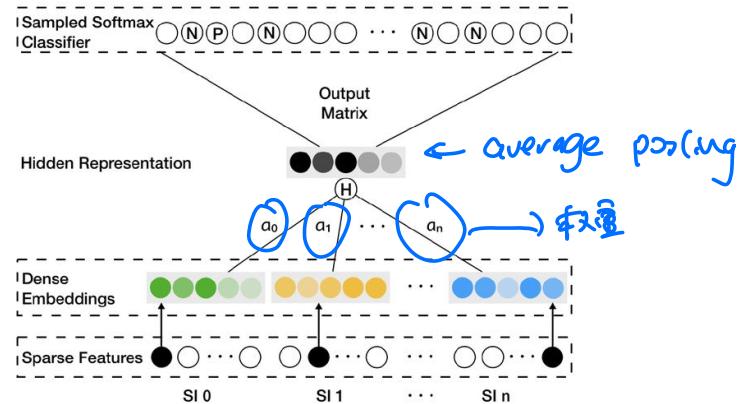
node2vec 所体现的网络的同质性和结构性在推荐系统中也是可以被很直观的解释的。同质性相同的物品很可能是同品类、同属性、或者经常被一同购买的物品, 而结构性相同的物品则是各品类的爆款、各品类的最佳凑单商品等拥有类似趋势或者结构性属性的物品。毫无疑问, 二者在推荐系统中都是非常重要的特征表达。由于 node2vec 的这种灵活性, 以及发掘不同特征的能力, 甚至可以把不同 node2vec 生成的 embedding 融合共同输入后续深度学习网络, 以保留物品的不同特征信息。

3. EGES

Intuition: 解决 Node2vec 的冷启动问题

How: 除了像之前一样生成物品的兴趣图以外, 再利用物品属性、相同类别信息
生成老子内容的兴趣图。然后将一个物品的多个 Embedding 合成一个。

↑
即大 Embedding 信息



→ 常常是将稀疏 embedding 转化为稠密。然后进行 average pooling。

4. LINE / SVDNE (线性)

Embedding 层与特征层结合的优缺点。

主要方向：

1. 在浅层特征向量上加 Embedding 层，生成高维稀疏特征向量供神经网络训练
2. 将浅层特征的 Embedding 特征向量，与其他特征向量连接后，一同进入深度学习网络进行训练
3. 通过计算用户和物品 Embedding 的相似度，Embedding 可以直接作为推荐系统的召回层

策略之一。

1.

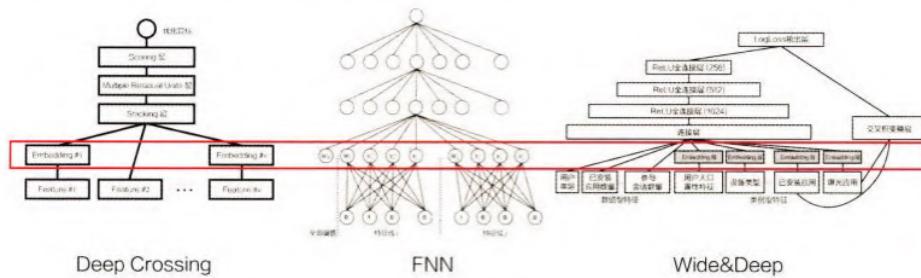
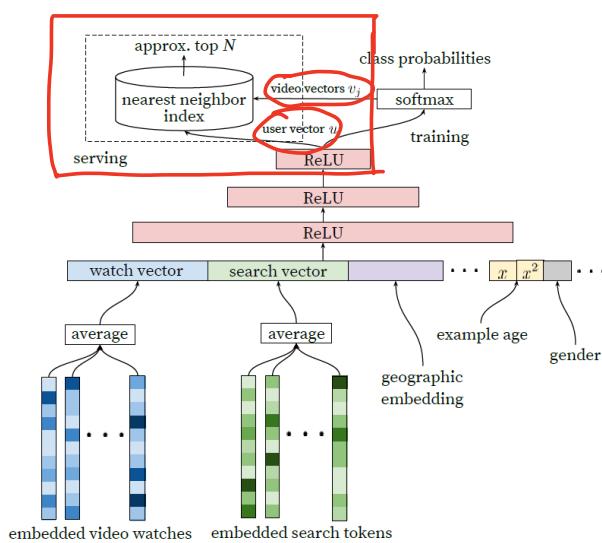


图 4-12 Deep Crossing、FNN、Wide&Deep 模型的 Embedding 层

2. 1. 深度过大，所以 Embedding 层的特征维度只适合于深度学习网络运行。在得到稀疏特征的稠密表达以后，再与其他特征一起输入神经网络训练。☆

Embedding 层的维度不高且很高，但上层神经网络为了层次拟合需要的参数量庞大
的信息，往往需要高歌密至实时训练。所以需要使用不同的训练方式对 Embedding
和神经网络模型

3.



局部敏感哈希

Intuition: 在召回时，如何使用embedding技术快速处理上百万级别的数据？

传统计算相似度方法 → embedding向量的内积计算 → 运算太慢

局部敏感哈希原理：

LSH的基本思想是：将原始数据空间中的两个相邻数据点通过相同的映射或投影变换（projection）后，这两个数据点在新的数据空间中仍然相邻的概率很大，而不相邻的数据点被映射到同一个桶的概率很小。也就是说，如果我们对原始数据进行一些hash映射后，我们希望原先相邻的两个数据能够被hash到相同的桶内，具有相同的桶号。对原始数据集合中所有的数据都进行hash映射后，我们就得到了一个hash table，这些原始数据集被分散到了hash table的桶内，每个桶会落入一些原始数据，属于同一个桶内的数据就有很大可能是相邻的，当然也存在不相邻的数据被hash到了同一个桶内。因此，如果我们能够找到这样一些hash functions，使得经过它们的哈希映射变换后，原始空间中相邻的数据落入相同的桶内的话，那么我们在该数据集合中进行近邻查找就变得容易了，我们只需要将查询数据进行哈希映射得到其桶号，然后取出该桶号对应桶内的所有数据，再进行线性匹配即可查找到与查询数据相邻的数据。换句话说，我们通过hash function映射变换操作，将原始数据集合分成了多个子集合，而每个子集合中的数据间是相邻的且该子集合中的元素个数较小，因此将一个在超大集合内查找相邻元素的问题转化为了在一个很小的集合内查找相邻元素的问题，显然计算量下降了很多。

总结：

Embedding 方法	基本原理	特 点	局限性
Word2vec	利用句子中词的相关性建模，利用单隐层神经网络获得词的 Embedding 向量	经典 Embedding 方法	仅能针对词序列样本进行训练
Item2vec	把 Word2vec 的思想扩展到任何序列数据上	将 Word2vec 应用于推荐领域	仅能针对序列样本进行训练
DeepWalk	在图结构上进行随机游走，生成序列样本后，利用 Word2vec 的思想建模	易用的 Graph Embedding 方法	针对性不强
Node2vec	在 DeepWalk 的基础上，通过调整随机游走权重的方法使 Embedding 的结果在网络的性质和结构性之间进行权衡	可以有针对性地挖掘不同网络特征	需要较多的人工调参工作
EGES	将不同信息对应的 Embedding 向量融合后生成最终的 Embedding 向量	解决 Embedding 的冷启动问题	新，更多是从工程角度解决多 Embedding 融合问题
局部敏感哈希	利用局部敏感哈希的原理进行快速的 Embedding 向量最近邻搜索	解决利用 Embedding 作为推荐系统召回层的快速计算问题	存在小概率的最近邻遗漏的可能，需要进行较多的人工调参