Embedding技术在推荐系统中的应用（送书）

原创 子墨 搜索与推荐Wiki 2020-11-02

收录于话题
#推荐相关笔记

32个

> 自费送书《深度学习推荐系统》一本，说出你对推荐系统中Embedding的认识、应用场景说明等，对于不敷衍的留言，点赞的第一名可免费获取该书。

编辑：子墨为客
来源：《深度学习推荐系统》笔记，并进行补充和说明

1、Embedding 是什么

Embedding是用一个低维稠密的向量来“表示”一个对象（这里的对象泛指一切可推荐的事物，比如商品、电影、音乐、新闻等），同时表示一词意味着Embedding能够表达相应对象的某些特征，同时向量之间的距离也能够反应对象之间的相似性。

在词向量空间内，甚至完全不知道一个词的向量的情况下，仅靠语义关键加词向量运算就可以推荐出这个词的词向量。

Embedding技术对于深度学习推荐系统的重要性

- 推荐场景中大量使用one-hot编码对类别特征、ID类特征进行编码，导致向量特别稀疏，而深度学习本身不利于处理这些稀疏特征，因此几乎所有的深度学习推荐模型都会由Embedding层负责将稀疏向量转化为稠密向量
- Embedding本身就是极其重要的特征向量，相比MF等传统方法产生的特征向量，Embedding的表达能力更强，特别是Graph Embedding技术被提出后，Embedding几乎可以引入任何信息进行编码
- Embedding对物品、用户相似度的计算是常用的推荐系统召回层技术。在局部敏感哈希（Locality-Sensitive hashing）等快速最近邻搜索技术应用于推荐系统后，Embedding更适用于对海量备选物品进行快速“初筛”，过滤出千级别的物品交与深度学习模型进行排序「局部

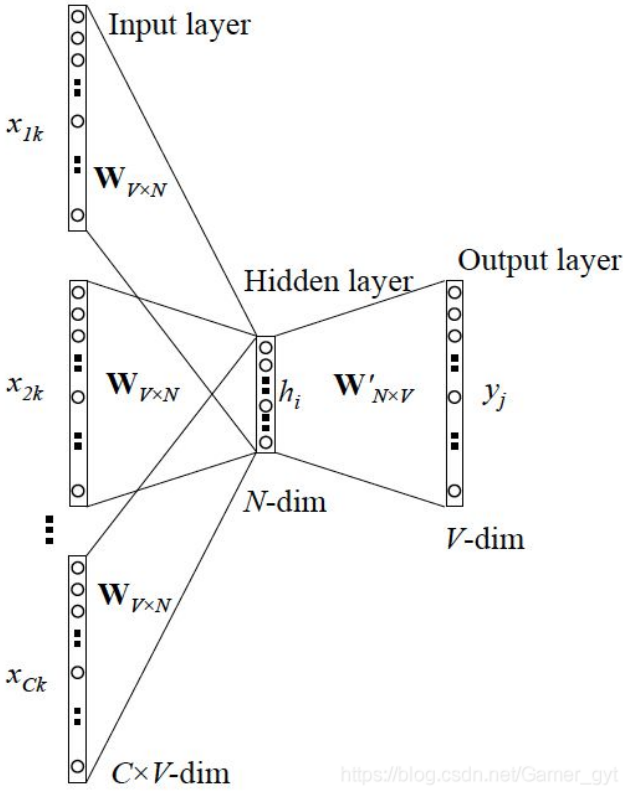
敏感哈希虽然经典，但是**recall**低，现在另外一种常用的快速检索方法为：基于图的检索方法」

2、Word2Vec

Google 2013年提出的经典的Embedding算法，word2vec训练模式分为：CBOW和Skip-gram两种模式「经验上**skip-gram**效果要好一些」

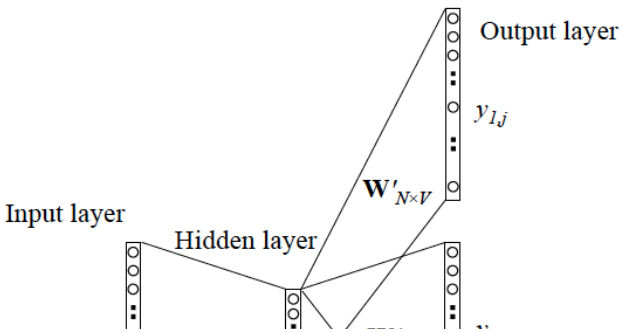
两种模式

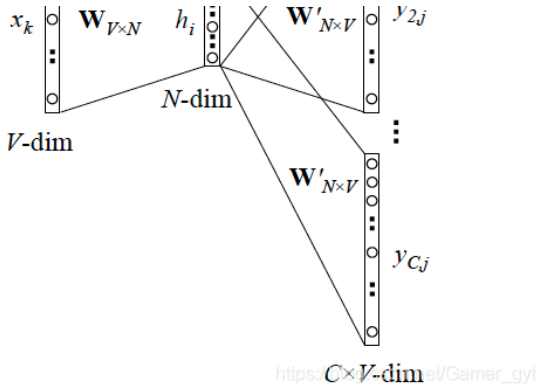
CBOW模式，中心词两边的词预测中心词，如下图所示。



CBOW模式

Skip-gram模式，中心词预测两边的词，如下图所示。





Skip-gram模式

基于skip-gram的训练过程

为了基于语料库生成模型的训练样本，选取长度为$2c+1$的滑动窗口，从语料库中抽取一个句子，将滑动窗口由左至右滑动，每移动一次，窗口中的词组就形成了一个训练样本。

接下来要定义优化目标了，每个词w_t 决定了相邻词w_{t+j}，基于极大似然估计的方法，希望所有样本的条件概率$p(w_{t+j}|w_t)$之积最大，这里使用对数据概率，因此word2vec的目标函数为：

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,j\neq 0} log\, p(w_{t+j}|w_t)$$

那么如何定义$p(w_{t+j}|w_t)$，作为一个多分类问题，最直接的方法就是使用softmax，**word2vec的愿景是希望用一个向量v_w表示词w，用词之间的内积距离$v_i^T v_j$表示语义的接近程度**，那么条件概率 $p(w_{t+j}|w_t)$可以表示为：

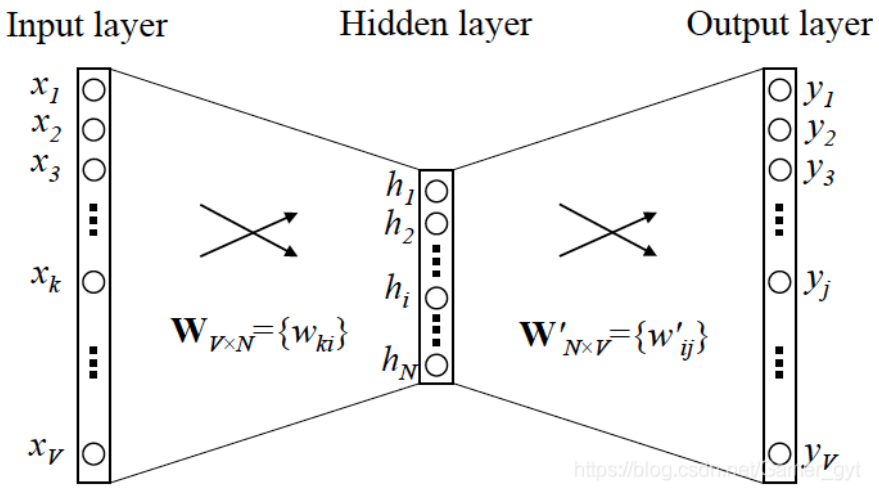
$$p(W_o|W_i)=\frac{exp({V'_{w_o}}^T V_{w_i})}{\sum_{w=1}^{W} exp({V'_w}^T V_{w_i})}$$

w_i 表示的是输入词，w_o 表示的是输出词，V_{w_i} 表示的是输入词的输入向量，$V_{w_o}{}'$ 表示的是输出词的输出向量。

> 「结合文章「**https://www.cnblogs.com/guoyaohua/p/9240336.html**」进行理解」

因此可以看出Skip-gram的本质是**计算输入词的input向量和输出词的output向量之间的余弦相似度，并进行softmax归一化**

那么什么是输入向量，什么是输出向量呢？



word2vec 输入向量 输出向量

上图中输入层导隐藏层的权重矩阵W_{VxN} 就是输入向量，隐藏层到输出层的权重矩阵W_{NxV}'就是输出向量。

Word2vec的“负采样”训练方法

如果按照上面给出的word2vec的模型结构进行模型训练是不现实的，因为输出的的神经元特别大，实际计算中无法承受这样的计算量。

为了减轻计算负担，往往采用“负采样”的方法进行训练，相比原来的需要计算所有字典中所有词的预测误差，负采样方法只需要对采样出的几个负样本计算预测误差，在这种情况下，word2vec模型的优化目标从一个多分类问题退化成了近似二分类问题，如下所示：

$$E = -log\sigma({v'_{w_o}}^T h) - \sum_{w_j \in W_{neg}} log\sigma(-{v'_{w_j}}^T h)$$

其中：

- v'_{w_o} 是输出词向量（及正样本）
- h 是隐层向量
- W_{neg} 是负样本集合
- v'_{w_j} 是样本词向量

由于负样本集合的大小非常有限（在实际应用中通常小于10），在每轮梯度下降迭代中，计算复杂度会降低懂啊原来的很多倍！

实际上加快word2vec训练的方法还有**Hierarchical softmax（层级softmax）** 但实现比较复杂，且最终效果没有明显优于负采样方法，因此很少使用。

负采样算法补充说明（来自上边提到的文章链接）

负采样算法的思想最初来源于一种叫做 Noise-Contrastive Estimation的算法，原本是为了解决那些无法归一化的概率模型的参数预估问题，与改造模型输出概率的Hierarchical Softmax算法不同，NES算法改造的是模型的似然函数。

基于skip-gram的word2vec目标函数中的$p(w_{t+j}|w_t)$ 是一个在整个字典上归一化了的概率。而在NCE算法中，我们构造了这样一个问题：对于一组训练样本，我们想知道，target word的出现，是来自于context的驱动，还是一个事先假定的背景噪声的驱动？显然可以通过一个逻辑回归函数来回答该问题。

$$P(D=1|w,context)=\frac{p(w|context)}{p(w|context)+kp_n(w)}=\sigma(log\,p(w|context)-log\,kp_n(w))$$

这个式子给出了 target word w 来自于 context 驱动的概率，其中，k 是一个先验参数，表明草绳的采样频率。$p(w|context)$ 是一个非归一化的概率分布，这里采用softmax 归一化函数中的分子部分，$p_n(w)$ 则时背景噪声的词分布，通常采用word的unigram分布。

通过对噪声分布的 k 采样，我们得到一个新的数据集，其中，label标记了数据的来源。在这个新的数据集上，我们就可以用最大化上式中逻辑回归的似然函数来求解模型的参数。

Word2vec 论文中提出的负采样算法，时NCE的一个简化版本，在这个算法里，作者抛弃了NCE似然函数中对噪声分布的依赖，直接用原始的softmax函数里的分子定义了逻辑回归函数，进一步简化了计算：

$$p(D=1|w_0,w_i)=\sigma(U_0\cdot V_i)$$

此时模型相应的目标函数为：

$$J(\theta)=log\,\sigma(U_0\cdot V_i)+\sum_{j=1}^{k}E_{w_j\sim p_nw}[log\,\sigma(-U_j\cdot V_i)$$

「对比上式和书中的公式只不过是将负号提取出来的区别，本质是一样的。」

在spark中和python（gensim）中都有实现好的word2vec供使用，感兴趣的可以参考：

- spark：ML中的Word2vec实现，MLlib中的word2vec实现
- gensim：Gensim word2vec 官方文档

「构建序列数据一般会划定时间，常用的是一个小时，但也分场景，对于**item**比较多的业务下，可以，但是对于**item**不是那么多的业务，一个小时的时间窗口就不太合适了」

3、Item2vec

Item2vec 是word2vec在其它领域的扩展，于2016年由微软提出。其和word2vec的区别试：Item2vec摒弃了时间窗口的概念，认为序列中任意两个物品都相关，因此在item2vec的目标函数中可以看到，其是两两物品的对数概率的和，而不仅试时间窗口内物品的对数概率之和。

其目标函数为：

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

万物届可**Embedding**，理论上Embedding对物品进行向量化的方法都可以称为Item2vec。在推荐系统中应用比较成功的是双塔模型，具体可以参考文章：论文丨从**DSSM**语义匹配到**Google**的双塔深度模型召回和广告场景中的双塔模型思考

「**Item2vec**缺点」

- 用户的行为序列时序性缺失
- 用户行为序列中的item强度是无区分性的
- 没有融合item的其他信息

「在**Item2vec**中，样本数据的两种方式」

- 基于时序，认为item之间存在强时序的关系，即前面item对后面item的产生有很大影响，这样和Word2vec的训练数据构造就没什么区别了，可以直接按照Word2vec的方式进行训练

- 基于集合，认为item之间有非常弱的时序关系，这时候需要放弃考虑item间的时空信息，用集合取代序列（比如购物场景中的订单集合），此种情况下，有两种代码实现方式：
  - 1、把word2vec的上下文窗口由定长改为变长，窗口长度就是集合长度，其实也就是对整个item集合中的item进行两两组合构成正样本，此方法需要修改网络结构
  - 2、不修改网络结构，在训练时，对物品集合进行shuffle操作，变相的起到忽略序列带来的影响
  - 说明：在item论文中指出两种方法的实验效果基本一致，论文中采用的是第一种方法，即将目标函数改为了上述的那种。

「关于**Item2vec**实现的代码可参考」

- https://github.com/ctjoy/item2vec
- https://github.com/lujiaying/MovieTaster-Open
- https://zhuanlan.zhihu.com/p/55960239

4、Graph Embedding——引入更多的结构信息的图嵌入

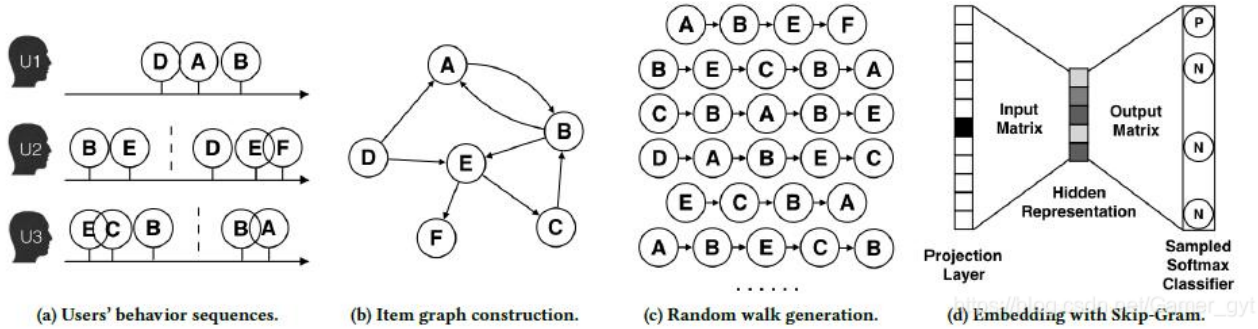
word2vec 和 item2vec都是建立再序列的样本上的，但是在互联网的场景下，数据对象之间更多呈现的是图结构，在面对图结构的数据时，传统的序列Embedding方法就显得力不从心了，这时候Graph Embedding成为了新的研究方向。

Graph Embedding 是一种对图结构中的节点进行Embedding编码的方法，最终生成的节点Embedding向量一般包含图的结构信息及附近节点的局部相似性信息。不同的Graph Embedding方法原理不尽相同，对于图信息的保留方式也有所区别，下面简单介绍几种。

DeepWalk-基础graph Embedding的方法

DeepWalk于2014年提出，其主要思想时在由物品组成的图结构上进行随机游走，产生大量的物品序列，然后将这些物品序列作为训练样本输入Word2Vec进行训练，继而得到物品的Embedding信息。DeepWalk可以被看作时序列Embedding和Graph Embedding的过渡方法。

DeepWalk的算法流程为：



(a) Users' behavior sequences. (b) Item graph construction. (c) Random walk generation. (d) Embedding with Skip-Gram.

- 1、(a)为用户行为序列
- 2、(b) 基于这些用户行为序列构建了物品的关系图。可以看出 物品A和B之间的边产生的原因是用户U_1先后购买了物品A和物品B，如果后续产生了多条相同的有向边，则有向边的权重被加强。再将所有用户行为序列都转化成物品关系图中的边之后，全局的物品关系图就建立起来了
- 3、(c) 采用随机游走的方式随机选择起始点，重新产生物品的序列
- 4、(d) 将这些序列输入导Word2Vec模型中，生成最终的物品Embedding向量

在上述介绍的DeepWalk算法流程中，唯一需要形式化定义的是随机游走的跳转概率，即到达节点v_i 后，下一步遍历v_i的邻接点v_j的概率，如果物品关系图是有向有权图，那么从节点v_i跳转到v_j到概率定义为：

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

其中ε 是物品关系图中所有边的集合，$N_+(v_i)$ 是节点v_i所有的出边集合，M_{ij} 是节点v_i 到节点v_j边的权重，即DeepWalk的跳转率就是跳转边的权重占所有相关出变的权重之和的比例。

如果物品关系图是无向无权图，那么上述公式中 M_{ij} 应该是常数1，$N_+(v_i)$ 应是节点v_i所有边的集合，而不是所有出边的集合。

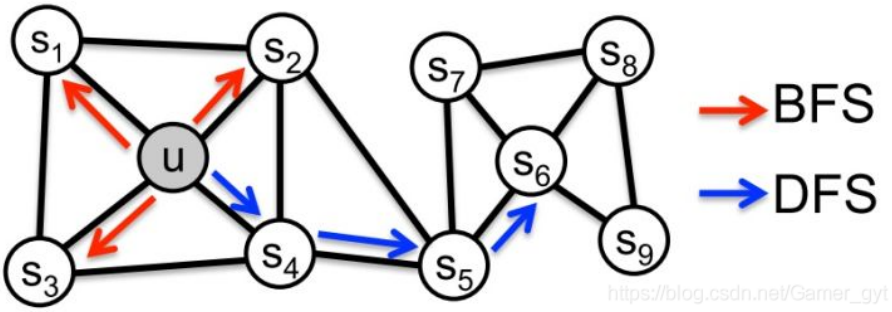
「 DeepWalk补充说明 」

- DeepWalk原文：http://www.perozzi.net/projects/deepwalk/
- DeepWalk论文：https://arxiv.org/pdf/1403.6652.pdf
- DeepWalk代码：https://github.com/phanein/deepwalk

Node2vec-同质性和结构性的权衡

2016年斯坦福大学的研究人员在DeepWalk的基础上，提出了Node2vec模型，它通过调整随机游走权重的方法使Graph Embedding的结果更倾向于体现网络的**同质性（homophily）** 和 **结构性（structural equivalence）**

1、同质性和结构性



node2vec 同质性和结构性

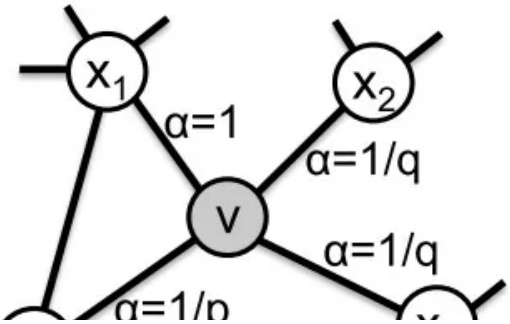
网络的同质性指的是距离相近的节点的Embedding应尽量近似，如上图所示，节点U与其相连的节点s_1, s_2, s_3, s_4 的Embedding表达应该是相近的，这就是网络的同质性体现。

结构性指的是结构上相似的节点的Embedding应尽量近似，如上图节点U 和节点 s_6 都是各自局域网络的中心点，结构上相似，其Embedding的表达也应该近似，这是结构性的体现。

如果倾向于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会更多的再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」。

如果倾向于Graph Embedding的结果能够表达网络的同质性，需要让随机游走的过程更倾向于DFS，因为DFS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，从而更多的表达网络的同质性。

2、node2vec的跳转概率

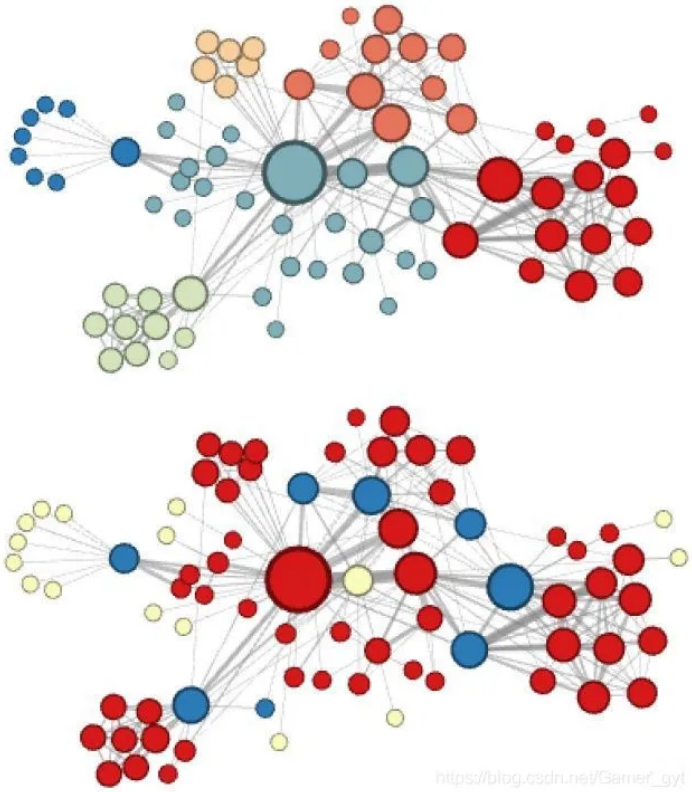


node2vec的跳转概率

从节点v跳转到下一个节点x的概率$\pi_{vx}=\alpha_{pq}(t,x)\cdot w_{vx}$，其中$w_{vx}$是边$vx$的权重，$\alpha_{pq}(t,x)$的定义如下所示：

$$\alpha_{pq}(t,x)=\begin{cases}\frac{1}{p} & ,if\ d_{tx}=0\\ 1 & ,if\ d_{tx}=1\\ \frac{1}{q} & ,if\ d_{tx}=2\end{cases}$$

其中d_{tx}指节点t到节点x的距离（即经过的节点数），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，随机游走回节点t的可能性越大，Node2vec就更关注表达网络的结构性。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就越大，Node2vec就更注重表达网络的同质性；反之，则当前节点更可能再附近节点游走。



node2vec实验结果

node2vec这种灵活表达同质性和结构性的特点也得到了实验的证实，通过调整参数p和q产生了不同的Embedding结果，如上图的上半部分就是Node2vec更注重同质性的体现，下半部分就是node2vec更注重结构性的体现。

node2vec所体现的同质性和结构性再推荐系统中可以被狠直观的解释，同质性相同的物品很可能是同品类、同属性、或者经常被一起购买的物品，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商品。

同时对于Node2vec产出的不同倾向性的Embedding，可以一起输入深度神经网络中，以保留物品的不同图特征信息。

「 Node2vec的优化目标 」

设$f(u)$ 是将顶点u 映射未embedding向量得映射函数，对于图中每个顶点u，定义$N_S(u)$为通过采样策略S 采样出得顶点u的近邻顶点集合。

node2vec优化的目标是给定每个顶点条件下，令其近邻顶点（如何定义近邻顶点很重要）出现的概率最大。

$$max_f \sum_{u \in V} log\, P_r(N_S(U)|f(u))$$

为了将上述最优化问题可解，文章提出两个假设：

1、条件独立性假设

假设给定源顶点情况下，其近邻顶点出现的概率与近邻集合中其余顶点无关

$$P_r(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} P_r(n_i|f(u))$$

2、特征空间对称性假设

这里是说一个顶点作为源顶点和作为近邻顶点的时候共享同一套Embedding向量。（对比LINE中的2阶相似度，一个顶点作为源点和近邻点的时候是拥有不同的embedding向量的）在这个假设下，上述条件概率分布可以表示未：

$$P_r(n_i|f(u)) = \frac{expf(n_i) \cdot f(u)}{\sum_{v \in V} expf(v) \cdot f(u)}$$

根据以上两个假设条件，最终的目标函数表为：

$$max_f \sum_{u \in V} [-logZ_u + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u)]$$

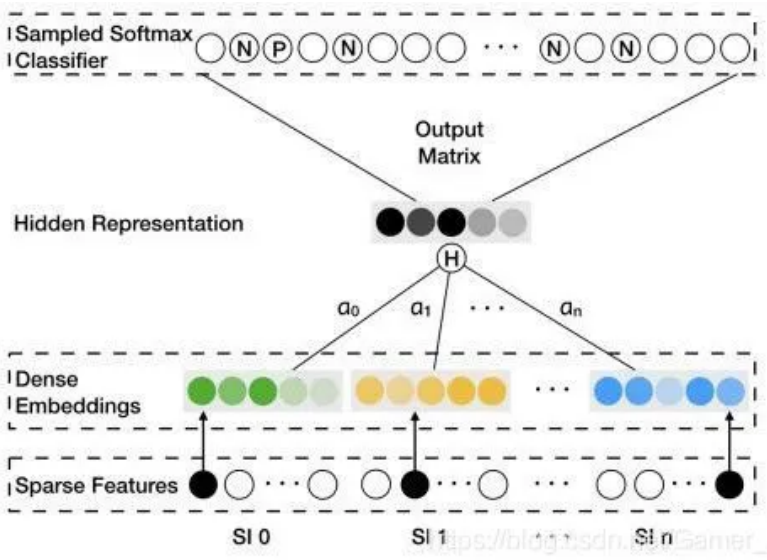
由于归一化因子$Z_u = \sum_{n_i \in N_s(u)} exp(f(n_i) \cdot f(u))$ 的计算代价高，所以采用负采样技术优化。

5、EGES-阿里巴巴的综合性Graph Embedding方法

2018年，阿里巴巴公布了其在淘宝应用的Embedding方法EGES（Enhanced Graph Embedding with side Infomation），其基本思想是在DeepWalk生成的Graph Embedding基础上引入补充信息。

单纯的使用用户行为构建的Graph Embedding可以生成Embedding信息，但是对于新物品或者没有过多「互动」信息的「长尾物品」，推荐系统会表现出很严重的冷启动问题。为了解决这个问题，阿里巴巴引入更多的补充信息来丰富Embedding的来源。**在构建物品关系图时，不仅依赖用户的交互行为，也可以利用用户的属性信息建立联系，从而生成基于内容的知识图谱，急于知识图谱生成的向量可以称为补充信息Embedding向量**

如何融合一个物品的多个Embedding向量？



EGES

最简单的方法是在深度神经网络中加入平均池化层，将不同的Embedding平均起来。但该种方法会导致有效Embedding信息的丢失。

EGES的做法是，对于每类特征对应的Embedding向量，分别赋予权重$a_0, a_1, \ldots, a_n$，图中的隐层表达（Hidden Representation层）就是对不同Embedding进行加权平均操作的

层，将加权平均后的Embedding向量输入softmax层，通过剃度反向传播，求得每个Embedding的权重$a_i(i=0,1,\ldots,n)$

- 上图的Sparse Features代表 item 和 side information 的ID信息；
- Dense Embeddings 表示 item 和 side information 的 embedding 信息；
- $a_0, a_1, \ldots, a_n$ 分别代表 item 和 side information 的 embedding 权重；
- Sampled Softmax Classifier中的N代表采样的负样本（见论文中的Algorithm 2 Weighted Skip-Gram描述的第8行），P代表正样本（某个item周边上下n个item均为正样本，在模型中表示时不区分远近）

在实际的模型中，阿里采用的是e^{a_j} 而不是a_j作为相应的Embedding的权重，王喆老师认为主要原因有两点，（一）：避免权重为0，（二）：因为e^{a_j}在梯度下降过程中有较好的数学性质

「EGES论文：https://arxiv.org/pdf/1803.02349.pdf，网友提供的代码（貌似有点小问题，未验证）：https://github.com/wangzhegeek/EGES」

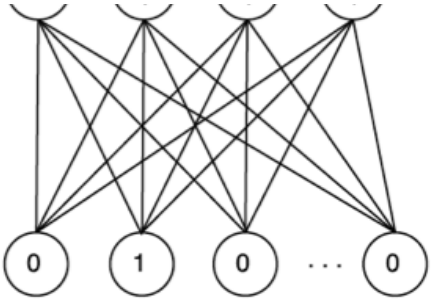
6、Embedding与深度学习推荐系统的结合

Embedding技术在深度学习推荐系统中主要应用在三个方法：

- 在深度神经网络中作为Embedding层，完成从高纬稀疏特征向量到低维稠密特征向量的转换
- 作为预训练的Embedding特征项链，与其他特征向量连接后，一同输入深度神经网络进行训练
- 通过计算用户和物品的Embedding相似度，Embedding可以直接作为推荐系统的召回层或者召回策略之一

DNN中的Embedding层

高维稀疏特征不适合做多层复杂的神经网络训练，因此在DNN中会在输入层和全连接层之间加入Embedding层，完成高维稀疏特征向量到低维稠密特征向量的转换。比如在Deep Crossing、FNN、Wide&Deep 三个典型的深度学习模型的Embedding层，这三个模型的Embedding层接收的都是类别型特征的one-hot向量，转换的目标是低维的Embedding向量。



矩阵形式表达Embedding层

用矩阵形式表达Embedding层，本质是求解一个m（输入高维西蜀向量的维度） * n（输出稠密向量的维度）的权重矩阵过程，如果输入向量是one-hot特征向量，则权重矩阵中的「 列向量 」为相应维度one-hot特征的Embedding向量。

Embedding层与整个深度神经网络进行整合一起训练是理论上的最优选择，因为上层梯度可以直接反向传播到输入层，模型是闭合自治的。但缺点是：**Embedding的维度往往很大，进行训练是参数也很大，会拖慢整体的训练速度**，因此在很多工程上放弃了Embedding层的端到端的训练，用预训练的方式代替。

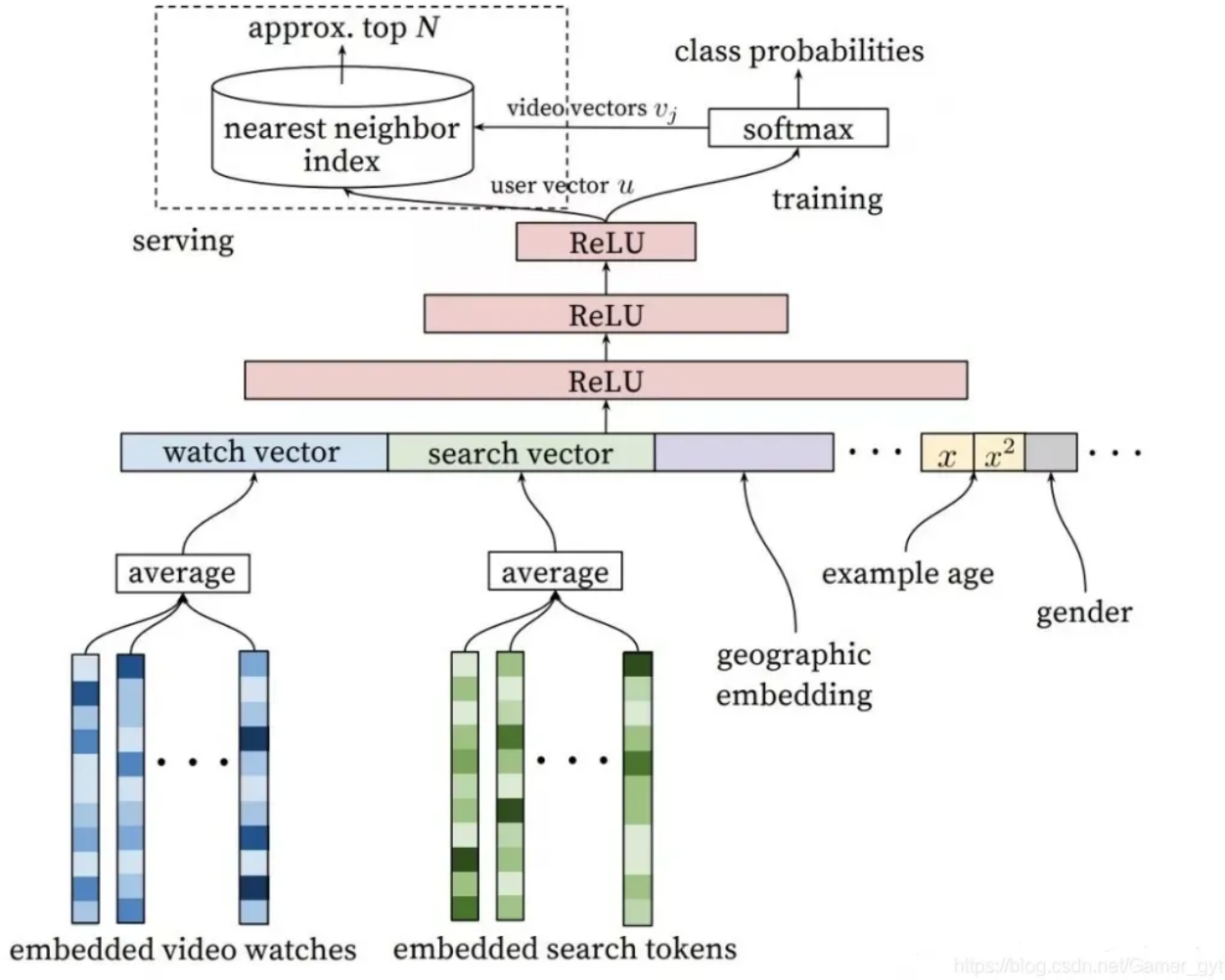
Embedding的预训练方法

典型的采用Embedding预训练的方法是采用FNN模型（《传统机器学习和前沿深度学习推荐模型演化关系》），它将FM模型训练得到的各特征隐向量作为Embedding层的初始化权重，从而加快了整个网络的收敛速度。FNN模型的原始实现中，整个梯度下降过程中还是会更新Embedding的权重，如果希望进一步加快网络的收敛速度，可以将其固定。

将Embedding过程与深度神经网络的训练过程割裂会损失一定的细腻些，但训练过程的独立也带来了训练灵活性的提升。比如，物品或者用户的Embedding是比较稳定的（因为用户的兴趣、物品的属性不可能在几天内发生巨大的变化），Embedding的训练频率其实不需要很高，甚至可以降低到周的级别，但上层神经网络为了尽快抓住最新的数据整体趋势信息，往往需要更高频率甚至实时训练。使用不同训练频率更新Embedding模型和神经网络模型，是训练开销和模型效果二者之间权衡后的最优方案。

Embedding作为推荐系统召回层的方法

Embedding自身表达能力的增强使得直接利用Embedding生成推荐列表成了可行的选择。因此利用Embedding向量的相似性，将Embedding作为推荐系统找回层的方案逐渐被推广开来，其中比较典型的是YouTube推荐系统召回层的解决方案，如下图所示：



YouTube深度召回

其中模型的输入层特征全部都是用户的相关特征，从左到右依次是用户观看历史视频的Embedding向量、用户搜索词Embedding向量、用户地理位置属性特征Embedding向量、用户（样本）年龄、性别相关特征。

模型的输出层为softmax层，该模型本质伤是一个多分类模型，预测目标是用户观看了哪个视频，因此softmax层的输入层是经过三层ReLU全连接层生成的用户Embedding，输出向量是用户观看每一个视频的概率分布，由于输出向量的每一维对应了一个视频，该维对应的softmax层列向量就是物品Embedding。通过模型的离线训练，可以最终得到每个用户的Embedding和物品Embedding。

模型部署的经验是：**将用户和物品的Embedding向量存储到线上数据库，使用时，只需要将用户Embedding和物品Embedding存储到线上内存数据库，通过内积运算再排序的方法就可以得到物品的排序，再通过取序列中Top N的物品即可得到召回的候选集合**

「但是再几百万量级的互联网场景下，即使是遍历内积运算这种$O(n)$级别的操作，也会消耗大量的计算时候，这时候就需要「局部敏感哈希技术来解决了」」

7、局部敏感哈希——Embedding的快速搜索法

Embedding技术凭借其能够综合多种信息和特征的能力，相比传统的给予规则的召回方法，更适于解决推荐系统的召回问题。

快速Embedding 最近邻搜索

传统的Embedding相似度的计算方法是Embedding向量间的内积运算，这就意味着为了筛选某个用户的候选物品，需要对候选集合中的所有物品进行遍历。假设embedding维度为k, 物品总数是n，那么遍历一遍的时间复杂度为$O(kn)$，在线计算会导致巨大的延迟。

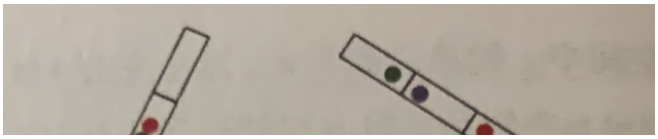
由于用户和物品的Embedding同处于一个向量空间内，所以召回与用户最相似的物品Embedding向量的过程其实是一个在向量空间内搜索最近邻的过程，如果能够找到高维空间快速搜索最近邻点的方法，那么相似Embedding的快速搜索问题就能够迎刃而解了。

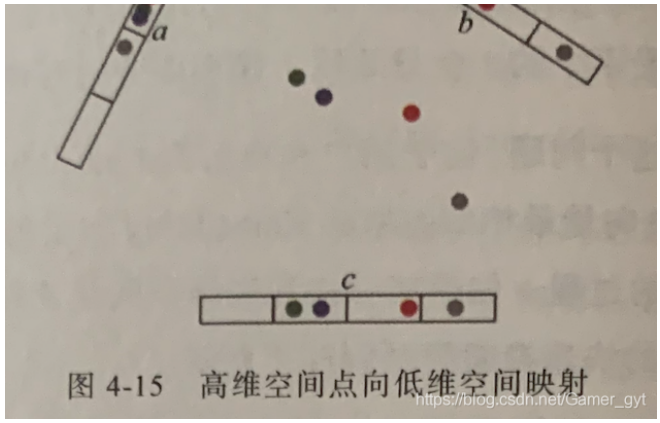
通过建立 kd(k-dimension) 树索引结构进行最近邻搜索是常用的快速最近邻搜索方法，时间复杂度可以降低到$O(log_2 n)$。但kd树结构复杂，而且再进行最近邻搜索时往往还要进行回溯，确保最近邻的结果，导致时间复杂度更低，另外，$O(log_2 n)$的时间复杂度并不是完全理想的状态。

因此推荐系统工程伤主流的快速Embedding 向量最近邻搜索方法-局部敏感哈希（Locality Sensitive Hashing，LSH）。

局部敏感哈希的基本原理

局部敏感哈希的基本思想是让相邻的点落在同一个“桶”，这样在进行最近邻搜索时，只需要在一个桶内进行搜索即可，如果保持每个桶中的元素个数在一个常数附近，就可以将最近邻搜索的时间复杂度降低到常数级别，那么如何构建局部敏感哈希中的桶呢？





高维空间点向低维空间映射

以「基于欧式距离的最近邻搜索」为例。如上图所示,中间的彩色点处在二维空间中，当把二维空间的点通过不同角度映射到a, b, c三个一维空间时，可以看到原本相近的点，在一维空间中都保持着相近的距离，而原本远离的绿色点和红色点在一维空间a中处于相近的位置，却在空间b中处于远离的位置，因此可得出一个结论：**在欧式空间中，将高维空间的点映射到低维空间，原本相近的点在低维空间中肯定依然相近，但原本远离的点则有一定概率变成相近的点**

利用该结论，就可以构造局部敏感哈希桶。

对于Embedding向量来说，也可以用内积操作构建局部敏感哈希桶。假设v是高维空间中的k维Embedding向量，x是随机生成的k维映射向量。如下所示，内积操作可将v映射到一维空间，成为一个数值。

$$h(v) = v \cdot x$$

由上面的结论可知，即使一维空间也会部分保存在高维空间的近似距离信息，因此，可以使用下面的哈希函数$h(v)$进行分桶：

$$h^{x,b}(v) = [\frac{x \cdot v + b}{w}]$$

其中[]是向下取整操作，w是分桶宽度，b是0到w的一个均匀分布随机变量，避免分桶边界固化。

映射操作会损失部分距离信息，因此常见的解决办法是使用m个哈希函数进行同时分桶操作，同时掉进m个哈希函数的同一个桶的亮点，是相似点的概率将会大大增加，通过分桶找到相邻点的候选集合后，就可以再有限的候选集合中通过遍历找到目标点真正的K近邻。

局部敏感哈希多桶策略

采用多个哈希函数进行分桶，存在一个待解决的问题：到底是通过“与（And）”操作还是“或（Or）”操作生成最终的候选集。

- 如果通过与操作，那么候选集中的近邻点的准确率将提高，候选集的规模减少使得需要遍历计算的量降低，但有可能会漏掉一些近邻点。
- 如果通过或操作，那么候选集中近邻点的召回率提高，但候选集的规模变大，计算开销升高。

到底使用几个哈希函数，是用「与」还是用「或」操作来生成近邻点的候选集，需要在准确率和召回率之间权衡，才能得出结论。

不同的距离度量函数，使用的局部敏感哈希方法也不同，但是通过分桶方式保留局部距离信息，大规模降低近邻点候选集得本质思想是通用的

> 在知乎上王老师发的帖子下有个回复点说的比较好：快速搜索的方法有很多，局部敏感哈希比较经典，但一般recall 太低，现在主流的基于图的搜索方法，可以参考文章：基于Delaunay图的快速最大内积搜索算法

8、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，利用单层隐层神经网络获得词的Embedding向量	经典的Embedding方法	仅能针对词序列样本进行训练
Item2vec	把Word2vec的思想扩展道任何序列数据上	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	随机游走进行抽样的针对性不强

Embedding方法	基本原理	特点	局限性
Node2vec	在DeepWalk的基础上，通过调整随机游走的方法使Graph Embedding的结果在网络的同质性和结构性之间进行权衡	可以有针对性的挖掘不同的网络特征	需要较多的人工调参工作
EGES	将不同信息对应的Embedding加权融合生成最终的Embedding向量	融合多种补充信息，解决Embedding的冷启动为你	没有较大的学术创新，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

「精彩推荐」

- 搜索和推荐系统中的深度匹配模型
- 传统机器学习和前沿深度学习推荐模型演化关系
- 论文丨从DSSM语义匹配到Google的双塔深度模型召回和广告场景中的双塔模型思考