Contents

下载本文 pdf: https://github.com/daiwk/collections/blob/master/pdfs/recommend.pdf

1 推荐系统整体梳理

https://daiwk.github.io/posts/links-navigation-recommender-system.html

https://github.com/Doragd/Algorithm-Practice-in-Industry

王喆的机器学习笔记系列:

https://github.com/wzhe06/Reco-papers

https://github.com/wzhe06/Ad-papers

深度学习传送门系列:

https://github.com/imsheridan/DeepRec

推荐系统遇上深度学习系列:

链接: https://pan.baidu.com/s/1jZkJ2d9WckbZL48aGFudOA 密码:kme3

推荐系统技术演进趋势: 召回-> 排序-> 重排

推荐系统的发展与 2019 最新论文回顾

深度推荐系统 2019 年度阅读收藏清单

推荐工业界实战角度详解 TensorFlow 中 Wide & Deep 源码（三）

2 特征、样本、数据流

浅谈微视推荐系统中的特征工程

推荐系统之数据与特征工程

稠密特征加入 CTR 预估模型的方法汇总

3 预估架构

3.1 HugeCTR

点击率预估的训练传统上存在着几个困扰着广大开发者的问题：巨大的哈希表（Embedding Table），较少的矩阵计算，大量的数据吞吐。

HugeCTR 是首个全部解决以上问题的开源 GPU 训练框架，与现有 CPU 和混合 CPU / GPU 解决方案相比，它的速度提高了 12 倍至 44 倍。HugeCTR 是一种端到端训练解决方案，其所有计算都在 GPU 上执行，而 CPU 仅用于 I / O。GPU 哈希表支持动态缩放。它利用 MPI 进行多节点训练，以支持任意大的嵌入尺寸。它还还支持混合精度训练，在 Volta GPU 及其后续版本上可以利用 Tensor cores 进一步加速。

如何解决点击率预估？英伟达专家详解 HugeCTR 训练框架（二）

Merlin HugeCTR 分级参数服务器简介

3.2 BOX

大规模深度学习广告系统的分布式分层 GPU 参数服务器

Distributed Hierarchical GPU Parameter Server for Massive Scale Deep Learning Ads Systems

4 索引架构

4.1 ANN 索引

annoy hnsw faiss pq

4.2 暴力召回 ANN 加速

https://kexue.fm/archives/9336

大致思想，CUR 分解：query 和 item 的 MxN 打分矩阵，分解成 F(Mxk1), G(k1xk2), H(k2xN) 三个矩阵

- Mxk1 矩阵：原矩阵里搞 k1 列出来，即选出 k1 个种子 item，得到 F
- k2xN 矩阵：原矩阵里搞 k2 列出来，即选出 k2 个种子 query，得到 H
- k1xk2 矩阵：即矩阵 1 和矩阵 2 求交集，比如矩阵 1 是抽的第 1,23,54 列出来，矩阵 2 是抽的第 4,80 行出来，那交集元素就是 (1,4),(1,80),(23,4),(23,80),(54,4),(54,80) 这 6 个点，构成 k1xk2 矩阵，然后算一下伪逆得到 G

建索引：＋挑出种子 query，和所有 item 两两计算相似度，得到 H 矩阵 ＋挑出种子 item，和种子 query 两两计算相似度，再算伪逆，得到 G 矩阵 ＋计算 G*H，存起来

检索：＋输入的 query 和 k1 个种子 item 算一下相似度，得到 1xk1 的矩阵 q ＋q 和 GH 相乘，就能得到 q 和每个 $item$ 的相似度了 ＋【这步可以 ann 化】：GH 就是 $k1$N，按列来看，就是 N 个 k1 维向量，相当于 N 个 item 向量，扔到 annlib 里去就行了，而输入的 q 也是一个 k1 维向量，就可以 ann 了

5 召回

360 展示广告召回系统的演进

推荐场景中深度召回模型的演化过程

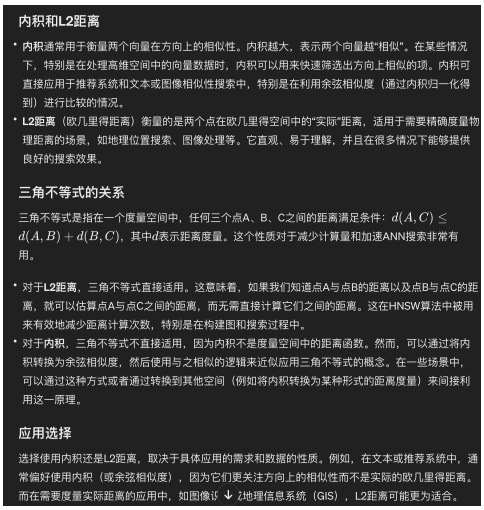
https://github.com/imsheridan/DeepRec/tree/master/Match

精准推荐的秘术：阿里解耦域适应无偏召回模型详解对应Co-training Disentangled Domain Adaptation Network for Leveraging Popularity Bias in Recommenders

谈谈文本匹配和多轮检索

搜索中的深度匹配模型

5.1 内积、余弦和 L2



内积和L2距离

- **内积**通常用于衡量两个向量在方向上的相似性。内积越大，表示两个向量越"相似"。在某些情况下，特别是在处理高维空间中的向量数据时，内积可以用来快速筛选出方向上相似的项。内积可直接应用于推荐系统和文本或图像相似性搜索中，特别是在利用余弦相似度（通过内积归一化得到）进行比较的情况。
- **L2距离**（欧几里得距离）衡量的是两个点在欧几里得空间中的"实际"距离，适用于需要精确度量物理距离的场景，如地理位置搜索、图像处理等。它直观、易于理解，并且在很多情况下能够提供良好的搜索效果。

三角不等式的关系

三角不等式是指在一个度量空间中，任何三个点A、B、C之间的距离满足条件：$d(A,C) \leq d(A,B) + d(B,C)$，其中$d$表示距离度量。这个性质对于减少计算量和加速ANN搜索非常有用。

- 对于**L2距离**，三角不等式直接适用。这意味着，如果我们知道点A与点B的距离以及点B与点C的距离，就可以估算点A与点C之间的距离，而无需直接计算它们之间的距离。这在HNSW算法中被用来有效地减少距离计算次数，特别是在构建图和搜索过程中。
- 对于**内积**，三角不等式不直接适用，因为内积不是度量空间中的距离函数。然而，可以通过将内积转换为余弦相似度，然后使用与之相似的逻辑来近似应用三角不等式的概念。在一些场景中，可以通过这种方式或者通过转换到其他空间（例如将内积转换为某种形式的距离度量）来间接利用这一原理。

应用选择

选择使用内积还是L2距离，取决于具体应用的需求和数据的性质。例如，在文本或推荐系统中，通常偏好使用内积（或余弦相似度），因为它们更关注方向上的相似性而不是实际的欧几里得距离。而在需要度量实际距离的应用中，如图像识 ↓ 别地理信息系统（GIS），L2距离可能更为适合。

给定 a，找到和它最像的 b

$$ab = ||a||cos\theta||b||$$

如果用内积，会找 $cos\theta||b||$ 最大的 b 出来，可能是夹角小，也可能是模大的 b，所以可能偏热门

5.2 DSSM

参考Modeling Interestingness with Deep Neural Networks

对应的 ppt：ppt

5.2.1 2013 年 CIKM 的 dssm

Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

相当于一个 q，和每个 d 分别算 cos。

$$R(Q,D) = cosine(y_Q, y_D) = \frac{y_Q^T y_D}{\|y_Q\| \|y_D\|}$$

所以 given Q 点击 D 的概率就是：

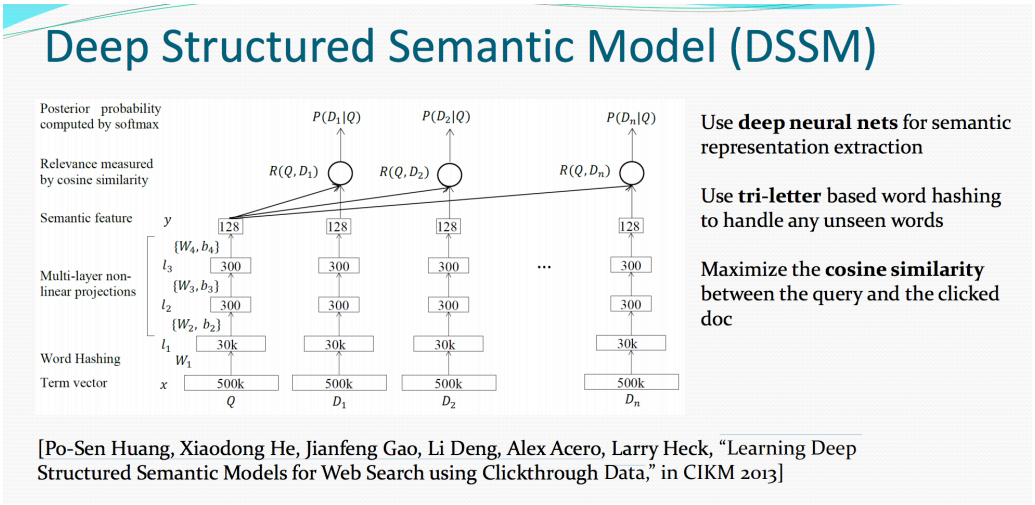
$$P(D|Q) = \frac{exp(\gamma R(Q,D))}{\sum_{D' \in \mathbf{D}} exp(\gamma R(Q,D'))}$$

其中的 γ 是平滑因子。这里的 **D** 是需要 rank 的 Documents 的集合，理想情况肯定是全集了。实际上定义 (Q, D^+) 为一个 query 和点击文档的 pair 对，通过一个 D^+ 和 N 个随机选的未点击的文档 $D_j^-, j = 1, ..., N$ 近似。

所以训练时，在训练集上，给定 query，有点击 doc 的概率最大化就是我们的目标 (其中的 Λ 是网络参数):

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q)$$

Word Hashing：例如，一个英文单词是"good"，那么会先在头尾各补上一个"#"，处理成"#good#"，然后拆成 n-gram（假设 n=3，也就是 tri-gram，那就是"#go"，"goo"，"ood"，"od#" 这么多个" 新" 词)。这样，可以把原来 500k 的词典缩到 300k，可以有效缓解词典太大的问题，而且因为英文就 26 个字母，这样做也解决了新单词的 OOV 问题。



https://blog.csdn.net/zjrn1027/article/details/80170966

相似度衡量可以使用 cos，而最终的 loss 可以用 hinge loss:

设置一个 margin m，query V_Q，正样本 V_{A^+}，负样本 V_{A^-}，如果**正负样本的相似度之差**小于边界值，那就还需要优化，如果**已经大于等于边界值**了，说明**模型已经能区分**了，所以用 hinge loss:

$ L=max(0, m - (cos(V_Q,V_{A^+} - cos(V_Q, V_{A^-})))) $

tf 算 cos

```
def getCosineSimilarity(q, a):
    q1 = tf.sqrt(tf.reduce_sum(tf.multiply(q, q), 1))
    a1 = tf.sqrt(tf.reduce_sum(tf.multiply(a, a), 1))
    mul = tf.reduce_sum(tf.multiply(q, a), 1)
    cosSim = tf.div(mul, tf.multiply(q1, a1))
    return cosSim
```

tf 算 hinge

```
def getLoss(trueCosSim, falseCosSim, margin):
    zero = tf.fill(tf.shape(trueCosSim), 0.0)
    tfMargin = tf.fill(tf.shape(trueCosSim), margin)
    with tf.name_scope("loss"):
        losses = tf.maximum(zero, tf.subtract(tfMargin, tf.subtract(trueCosSim, falseCosSim)))
```

```
            loss = tf.reduce_sum(losses)
        return loss
```

使用

```
        self.trueCosSim = self.getCosineSimilarity(question2, trueAnswer2)
        self.falseCosSim = self.getCosineSimilarity(question2, falseAnswer2)
        self.loss = self.getLoss(self.trueCosSim, self.falseCosSim, self.margin)
```

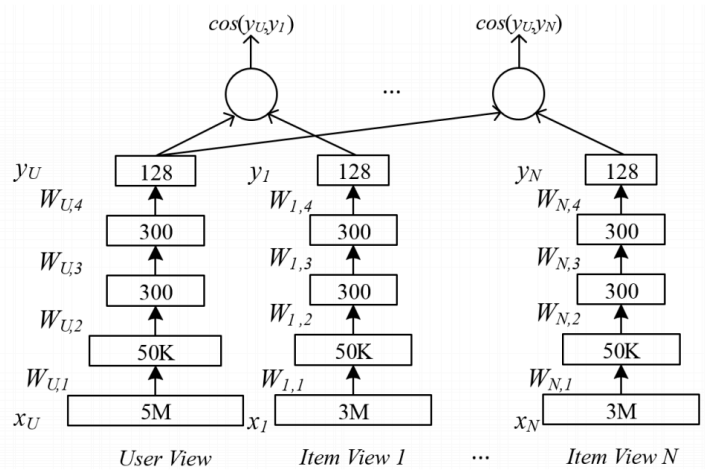
5.2.2 multiview dssm

A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems

https://blog.csdn.net/shine19930820/article/details/78810984

现在很多公司都不仅仅只有一个产品，而是有**多个产品线**。比如微软可能就有搜索、新闻、appstore、xbox 等产品，如果将用户在这些产品上的行为（反馈）统一在一起训练一个深度学习网络，就能很好的解决**单个产品上**（用户）**冷启动**、**稀疏**等问题。

5.2.2.1 概述



一个产品线就是一个 view，一条训练样本只有 user+1 个 view，其他 view 置 0。

总体的优化目标是保证在所有视图上 user 和正向反馈的 item 的相似度大于随机选取的无反馈或者负向反馈的相似度，并且越大越好。

$$p = \underset{W_u, W_1, \ldots W_v}{argmax} \sum_{j=1}^{N} \frac{exp(\alpha_a cos(Y_u, Y_{a,j})}{\sum_{X' \in R^{d_a}} exp(\alpha cos(Y_u, f_a(X', W_a)))}$$

其中的 $f_i(X_i, W_i)$ 是 X_i 到 Y_i 的变换。有一个用户 view，加上 v 个辅助 (auxiliary) 的 item view。$X_i \in R^{d_i}$，即每个 view 有自己的特征空间。对于第 j 条样本，它只有第 i 个 auxiliary view 是有值的，其他 view 都是 0。

tf 代码：https://github.com/InsaneLife/dssm/blob/master/multi_view_dssm_v3.py

5.2.3 降维

5.2.3.1 top features

对于 user features，选取 top-k 最频繁的 features。并通过 TF-IDF 过滤掉最常用的特征。

5.2.3.2 k means

kmeans 的公式如下：

$$\underset{C_1,...,C_k}{argmin} \sum_{i=1}^{N} \min_{C_j \in \{C_1,...,C_k\}} distance(X_i, C_j)$$

通过 K-means 对相似的特征群分组为同一个 cluster 并生成新的特征，共生成 k 个新特征。生成的特征向量 Y_i 是一个 K 维的向量，第 i 维是第 i 个 cluster 中 features 的出现数。

具体地，使用一个长度为 U 的 vector f_i，U 是训练集中的用户数，$f_i(j)$ 表示用户 i 有 feature j 的次数。将每个 f_i 进行归一化。这样，对于每一个用户向量 f_i，可以产生它的降维了的用户向量 Y_i（假设 feature a 分配给了 cluster $Cls(a)$，$1 \leq Cls(a) \leq K$）：

$$Y_i(j) = \sum_{a:X_i(a)>0 \& Cls(a)=j} f_i(a)$$

想要抽出 reasonable 的 feature 数目，需要比较大的 cluster 数 K。因为如果 cluster 数比较少，那一个 cluster 里就会有非常多的用户 feature，很难学到有用的 pattern。在本文中，设置 $K = 10000$，平均每个 cluster 里大概有 350 个 feature。因为 cluster 数比较大，所以用 mr 版的 kmeans。

5.2.3.3 Local sensitive Hashing

通过一个随机的矩阵将数据映射到低维向量空间上，并且**保持原始空间上的 pairwis cos 距离**在**新的空间上**仍然获得**保留**。

原始矩阵 d 维，降到 k 维，所以对应的矩阵是 $A \in R^{d \times k}$。所以 A 中有 k 个映射，每个映射 A_i 都将 X 映射到 Y_i，输出的 $Y_i \in R^k$。计算方式如下：

$$Y_i = \begin{cases} 1, & if\ A_i X \geq 0 \\ 0, & else \end{cases}$$

计算 X_1，X_2 的 cos 相似度是 $cos(\frac{H(Y_1,Y_2)}{k}\pi)$，其中，$X_1, X_2 \in R^d$。$H(Y_1, Y_2)$ 是 LSH 输出向量的汉明距离。为了保持 cos 相似度的高准确率，需要把 k 设得比较大，这里和 k-means 一样，$k = 10000$。

因为对每个向量算 LSH 是相互独立的，所以这一步是可以高度并行化的。但是，在我们的 case 里，有 10000 的 k，有 3.5M 的 d，所以相当于要把包括了 $3.5M \times 10^4$ 个服从 $N(0,1)$ 浮点数的 A 存到每个节点的内存里，也就是 300G 的内存消耗，这是肯定不行的。有很多解决方法，大部分方法是生成一个 sparse 的矩阵 A【kdd06 的Very sparse random projections】。

而本文用了Online Generation of Locality Sensitive Hash Signatures里提到的 pooling trick。

保存一个 size 是 m 的 pool B，每个元素是从 $N(0,1)$ 中随机出来的浮点数，m 远小于 transition matrix A 的 size。要获取 A_{ij} 的一条记录，就是使用关于 i, j 的一致性 hash 的方法在 B 中找到一个 index，然后查出它的值。在本文中，设置 $m = 1000000$，将单节点的内存损耗从 100G 缩减到 10M，可以直接用 mr 搞啦。

5.2.3.4 减少训练样本数

每个用户在每个域都有大量的日志数据，将每个用户在每个域只选取一个 user-item 对，具体为用户特征-用户在**此域**喜欢的所有 item 的**平均分数**。

5.3 YoutubeDNN

https://daiwk.github.io/posts/dl-youtube-video-recommendation.html

参考http://www.sohu.com/a/155797861_465975

参考https://zhuanlan.zhihu.com/p/25343518

Deep neural networks for youtube recommendations

YouTube 是世界上最大的视频上传、分享和发现网站，YouTube 推荐系统为超过 10 亿用户从不断增长的视频库中推荐个性化的内容。整个系统由两个神经网络组成：**候选生成网络**和**排序网络**。候选生成网络从**百万量级**的视频库中生成**上百个**候选，排序网络对候选进行打分排序，输出**排名最高的数十个结果**。

5.3.1 候选生成网络（Candidate Generation Network）

候选生成网络将推荐问题建模为一个**类别数极大的多分类问题**：对于一个 Youtube 用户，使用其观看历史（视频 ID）、搜索词记录（search tokens）、人口学信息（如地理位置、用户登录设备）、二值特征（如性别，是否登录）和连续特征（如用户年龄）等，对视频库中所有视频进行多分类，得到每一类别的分类结果（即每一个视频的推荐概率），最终输出概率较高的几百个视频。===> 即，【使用**用户特征**，对所有视频进行分类，得到**和这个用户最相关的几百个候选结果**。】

将推荐看成分类问题，用户 U 在上下文 C 中，选择视频 i 的概率是：

$$P(w_t = i|U, C) = \frac{e^{v_i u}}{\sum_{j \in V} e^{v_j u}}$$

其中，$v_i \in R^N$ 是第 i 个视频的 emb，$u \in R^N$ 是用户的 emb，两个 emb 都是 N 维的，这里看起来是用内积 $v_i u$ 把它们变成一个数。

由于视频有百万量级，所以做这么一个超大规模的分类，需要，并且使用的是到的样本**通过 importance sampling 进行负采样** (参考On using very large target vocabulary for neural machine translation)。对每一个 example 而言，他的 cross-entropy 是在 true label 和采样的负类中求 min。在实践中，采样了数千个负类，比传统的 softmax 有将近 100 倍的速度提升。

另一种做法是 hierarchical softmax(参考Hierarchical probabilistic neural network language model)，但实践中效果没那么好。因为在 hsoftmax 中，遍历树里的每一个节点时，会引入对经常是毫无联系的类别的分辨，使得分类问题变得更困难以至于效果变差。

在线 serving 时，由于低延迟的要求，需要有对类别数是 sublinear 的时间复杂度的近邻检索方法，之前 youtube 的系统使用的是 hashing 的方法，即Label partitioning for sublinear ranking。因为在线的时候，softmax 的输出没啥用，所以打分问题就变成了一个在点积的空间上进行最近邻检索的问题，有很多通用库可以用，例如基于 LSH 的 ann 算法：An Investigation of Practical Approximate Nearest Neighbor Algorithms。

注：

item-embedding 也可以参考https://zhuanlan.zhihu.com/p/24339183?refer=deeplearning-surfing里说的Item2vec: Neural Item Embedding for Collaborative Filtering的想法，把 item 视为 word，用户的行为序列视为一个集合，item 间的共现为正样本，并按照 item 的频率分布进行负样本采样，相似度的计算还只是利用到了 item 共现信息，缺点是：忽略了 user 行为序列信息; 没有建模用户对不同 item 的喜欢程度高低。

5.3.2 Modeling Expected Watch Time

训练用的是 logistic regression 加上 cross-entropy，

假设第 i 个正样本的播放时长是 T_i，使用 weighted logistic regression，将正样本的权重设为播放时长，而负样本的权重设为 1，这样，假设总共有 N 个样本，有 k 个被点击了，就相当于有了 $\sum T_i$ 个正样本，N-k 个负样本。所以 odds（注：一个事件的几率 odds 指该事件发生与不发生的概率比值）就是正样本数/负样本数 $=\frac{\sum T_i}{N-k}$。

而实际中，点击率 P 很低，也就是 k 很小，而播放时长的期望是 $E(T) = \frac{\sum T_i}{N}$，所以 $E(T)$ 约等于 $E(T)(1+P)$，约等于 odds，即 $\frac{\sum T_i}{N-k}$

最后在 inference 的 serving 中，直接使用 e^{Wx+b} 来产出 odds，从而近似 expected watch time。

参考https://en.wikipedia.org/wiki/Logistic_regression

odds 是平均时长，训练时输入给 sigmoid 的是 logit，

$ wx+b = logit = log odds $

所以，infer 的时候：

$$E(T) = odds = e^{logit} = e^{logodds} = e^{wx+b}$$

参考 tf 的 weighted sigmoid：`weighted_cross_entropy_with_logits`: https://www.tensorflow.org/api_docs/python/tf/nn/weighted_cross_entropy_with_logits

正常的 sigmoid：

```
labels * -log(sigmoid(logits)) +
    (1 - labels) * -log(1 - sigmoid(logits))
```

weighted sigmoid 只对正样本加权：

```
labels * -log(sigmoid(logits)) * pos_weight +
    (1 - labels) * -log(1 - sigmoid(logits))
```

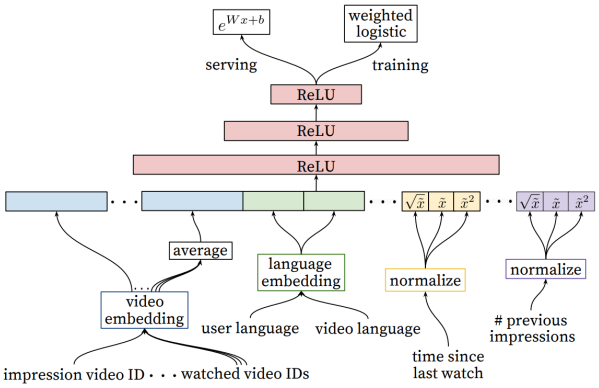


Figure 7: Deep ranking network architecture depicting embedded categorical features (both univalent and multivalent) with shared embeddings and powers of normalized continuous features. All layers are fully connected. In practice, hundreds of features are fed into the network.

5.3.3 代码实现

https://github.com/ogerhsou/Youtube-Recommendation-Tensorflow/blob/master/youtube_recommendation.py

关于数据集：

https://github.com/ogerhsou/Youtube-Recommendation-Tensorflow/commit/e92bac1b8b5deb0e93e996b490561baaea60bae8

使用的是https://github.com/facebookresearch/fastText/blob/master/classification-example.sh

在 init_data 函数中，给每个 __label__xx 编了个号，如：

```
__label__6 0
__label__12 1
__label__14 2
__label__7 3
```

然后 read_data 的时候，y 就用这个编号来表示（假装是时长）：

```
y.append(label_dict[line[0]])
```

而使用的是 nce_loss(参考https://daiwk.github.io/posts/knowledge-tf-usage.html#tfnnnceloss)：

```
ce_weights = tf.Variable(
    tf.truncated_normal([n_classes, n_hidden_1],
                        stddev=1.0 / math.sqrt(n_hidden_1)))
nce_biases = tf.Variable(tf.zeros([n_classes]))

loss = tf.reduce_mean(tf.nn.nce_loss(weights=nce_weights,
                     biases=nce_biases,
                     labels=y_batch,
                     inputs=pred,
                     num_sampled=10,
                     num_classes=n_classes))

cost = tf.reduce_sum(loss) / batch_size
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
out_layer = tf.matmul(pred, tf.transpose(nce_weights)) + nce_biases
```

5.4 采样

batch 内 shuffle 采样（有放回）

On Sampling Strategies for Neural Network-based Collaborative Filtering

浅谈个性化推荐系统中的非采样学习

Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations

https://www.tensorflow.org/extras/candidate_sampling.pdf

下载了一份：https://github.com/daiwk/collections/blob/master/assets/candidate_sampling.pdf

推荐系统遇上深度学习 (七十二)-[谷歌] 采样修正的双塔模型

5.5 召回常用 Loss

表示学习中的 7 大损失函数梳理

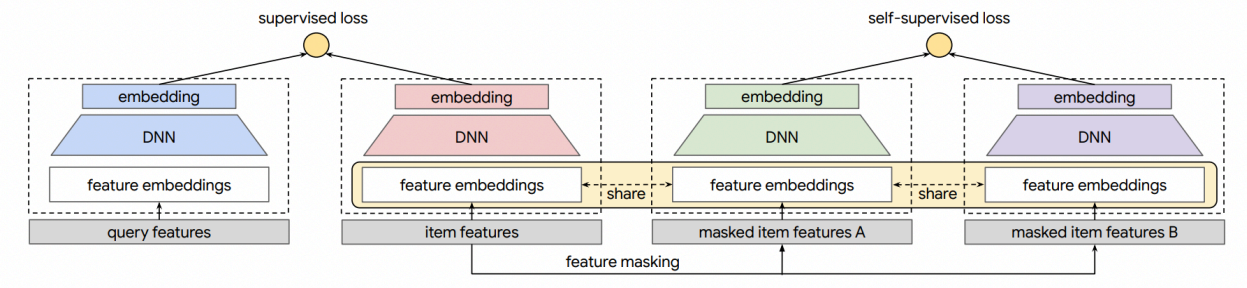
5.5.1 Contrastive Loss

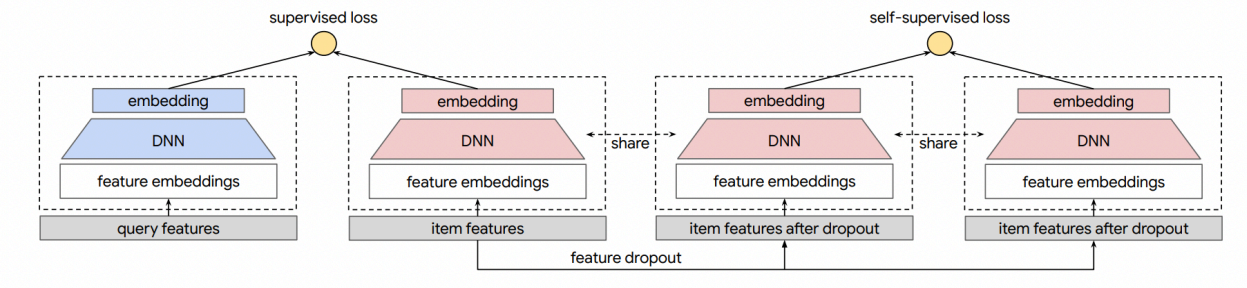
5.5.2 Triplet Loss

5.5.3 InfoNce Loss

Self-supervised Learning for Large-scale Item Recommendations

v3 有两个图：https://arxiv.org/pdf/2007.12865v3.pdf





5.5.4 Focal Loss

5.5.5 Circle Loss

5.5.6 qalign

Spherical Graph Embedding for Item Retrieval in Recommendation System

自己下载了

代码：https://github.com/WNQzhu/Q-align

自己的注释：https://github.com/daiwk/llms_new/blob/main/demos/qalign.py

假设 $N_K(u)$ 是节点 u 的 K 跳邻居，那么目标函数是最大化这些邻居的概率，即

$\max_f \sum_{u \in \mathcal{V}} \log \Pr\left(N_K(u) \mid f(u)\right)$

5.6 突破双塔

5.6.1 TDM->JTM

下一代深度召回与索引联合优化算法 JTM

5.6.2 二向箔

xx

5.6.3 DR

字节最新复杂召回模型，提出深度检索 DR 框架解决超大规模推荐系统中的匹配问题

Deep Retrieval: An End-to-End Learnable Structure Model for Large-Scale Recommendations

5.7 多兴趣召回

推荐系统多兴趣召回论文解读

模型	年份	会议	公司	多兴趣提取阶段	训练阶段	多兴趣聚合阶段（线上阶段）	亮点	不足
MIND	2019	CIKM	阿里	胶囊网络	label-aware attention	K*N召回 取topN	• 使用胶囊网络来提取用户多兴趣表示。	• 基于target label的训练方式存在训练测试不一致的问题 • 没有考虑兴趣组合
ComiRec	2020	KDD	阿里	ComiRec-DR ComiRec-SA	使用与target item最近的兴趣	相关性和多样性权衡	• 使用胶囊网络和self-attentive来提取用户多兴趣表示。 • 线上serving时，同时考虑相关性和多样性。	• 基于target label的训练方式存在训练测试不一致的问题 • 用户都使用固定的兴趣数量（文中是4）
SINE	2021	WSDM	阿里	概念激活 + self-attentive	intention-aware attention	K*N召回 取topN	• 分别提出了新的兴趣聚类算法和兴趣聚合算法。 • 使用协方差正则化来引导概念池的学习	• 用户都使用固定的兴趣数量（文中是8）
Octopus	2020	SIGIR	MSRA	信道激活 +attention	使用与target item最近的兴趣	1.K*N召回 取topN 2. 额外构建一个多分类任务	• 自适应选取兴趣数量	• 非端到端 • 多兴趣提取比较粗暴

5.8 transformer+ 召回

ICLR2020 cmu+google:

Pre-training Tasks for Embedding-based Large-scale Retrieval

6 粗排

6.1 COLD

6.2 CAN

6.3 poly-encoder

https://zhuanlan.zhihu.com/p/119444637

7 精排

7.1 传统 ctr

https://daiwk.github.io/posts/dl-traditional-ctr-models.html

7.1.1 lr for ctr

Simple and scalable response prediction for display advertising

Online Models for Content Optimization

7.1.2 gbdt for ctr

gbdt 基础知识:

https://zhuanlan.zhihu.com/p/86263786

bagging 全称叫 bootstrap aggregating，每个基学习器都会对训练集进行有放回抽样得到子训练集，比较著名的采样法为 0.632 自助法。每个基学习器基于不同子训练集进行训练，并综合所有基学习器的预测值得到最终的预测结果。bagging 常用的综合方法是投票法，票数最多的类别为预测类别。

boosting 训练过程为阶梯状，基模型的训练是有顺序的，每个基模型都会在前一个基模型学习的基础上进行学习，最终综合所有基模型的预测值产生最终的预测结果，用的比较多的综合方式为加权法。

stacking 是先用全部数据训练好基模型，然后每个基模型都对每个训练样本进行的预测，其预测值将作为训练样本的特征值，最终会得到新的训练样本，然后基于新的训练样本进行训练得到模型，然后得到最终预测结果。

bagging 和 stacking 中的基模型为强模型（偏差低，方差高），而 boosting 中的基模型为弱模型（偏差高，方差低）。

bagging 的特点：

- 整体模型的期望等于基模型的期望，这也就意味着整体模型的偏差和基模型的偏差近似。
- 整体模型的方差小于等于基模型的方差，当且仅当相关性为 1 时取等号，随着基模型数量增多，整体模型的方差减少，从而防止过拟合的能力增强，模型的准确度得到提高。

所以，bagging 中的基模型一定要为强模型，如果 bagging 使用弱模型则会导致整体模型的偏差提高，而准确度降低。

boosting 的特点：

- 整体模型的方差等于基模型的方差，如果基模型不是弱模型，其方差相对较大，这将导致整体模型的方差很大，即无法达到防止过拟合的效果。因此，boosting 框架中的基模型必须为弱模型。
- boosting 框架中采用基于贪心策略的前向加法，整体模型的期望由基模型的期望累加而成，所以随着基模型数的增多，整体模型的期望值增加，整体模型的准确度提高。

gbdt 与 Adaboost 对比

相同：

- 都是 boosting，使用弱分类器；
- 都使用前向分布算法；

不同：

- 迭代思路不同：adaboost 是通过提升错分数据点的权重来弥补模型的不足（利用错分样本），而 GBDT 是通过算梯度来弥补模型的不足（利用残差）；
- 损失函数不同：adaBoost 采用的是指数损失，GBDT 使用的是绝对损失或者 Huber 损失函数；

Learning the click-through rate for rare/new ads from similar ads

Using boosted trees for click-through rate prediction for sponsored search

Improving Ad Relevance in Sponsored Search

Stochastic Gradient Boosted Distributed Decision Trees

https://zhuanlan.zhihu.com/p/148050748

7.2 深度学习 ctr

https://daiwk.github.io/posts/dl-dl-ctr-models.html

7.3 序列建模

一文看懂序列推荐建模的最新进展与挑战

从 MLP 到 Self-Attention，一文总览用户行为序列推荐模型

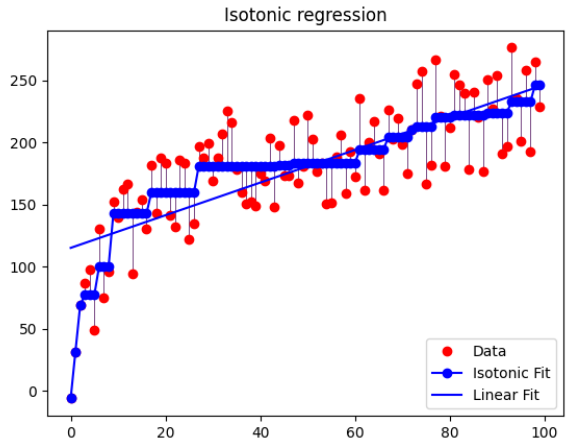
Transformer 在推荐模型中的应用总结

阿里妈妈点击率预估中的长期兴趣建模

DCN V2：Google 提出改进版 DCN，用于大规模排序系统中的特征交叉学习 (附代码)

7.4 保序回归

参考https://zhuanlan.zhihu.com/p/88623159的代码，能画出下面的图



对于二分类问题，参考https://zhuanlan.zhihu.com/p/101766505

对 lr+gbdt 的负采样校准的方法

Practical Lessons from Predicting Clicks on Ads at Facebook

7.5 cvr 预估

ecpc：用户给定一个粗粒度出价，模型可以在一定的范围内调价 ocpc：完全以模型出价为准

delay feedback https://zhuanlan.zhihu.com/p/555950153

7.6 时长预估

快手 kdd 2022

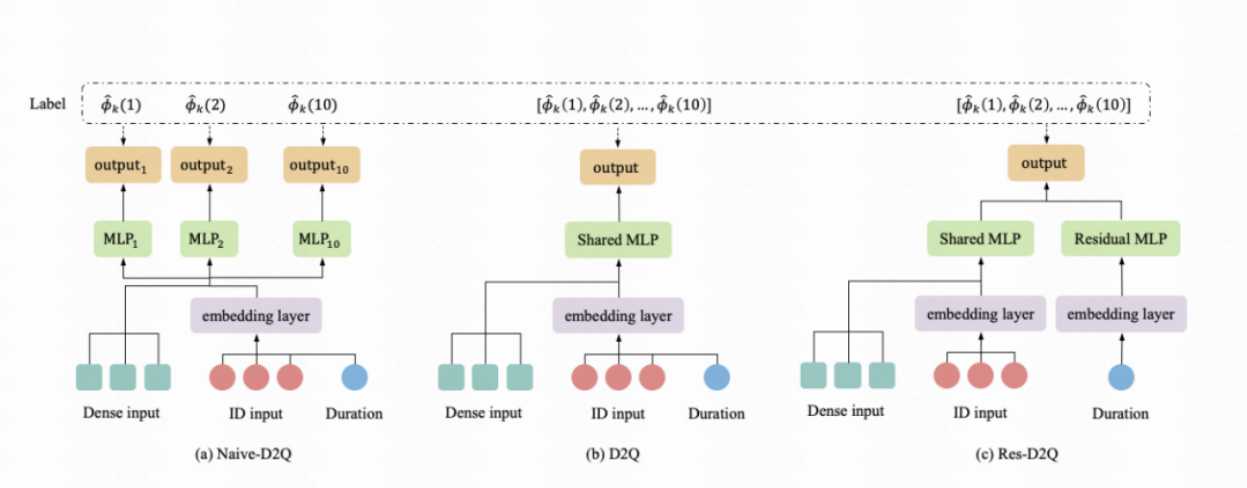
Deconfounding Duration Bias in Watch-time Prediction for Video Recommendation

短视频推荐视频时长 bias 问题

拿物理时长（duration）分桶

D2Q 算法的具体做法如下：

- 统计训练样本的 duration 分布，得到等频分桶分位点；
- 将样本按照等频分桶分位点分成 k 个相互独立的分桶 D_k；
- 对不同 duration 分桶的样本，在组内统计时长分位数作为 label，得到 Duration-Aware Watchtime-Distribution label；
- 分别在上述的分桶上训练时长预估模型 f_k；



- 图 a：M 个网络完全独立，分别学习各自的 label，不共享特征 embedding，特征 embedding 空间随着分桶维度扩大线性增加，存储、训练的资源开销随之增加，实现成本较高，不符合工业界场景的要求；
- 图 b：M 个网络共享底层特征，如果采用多输出的训练方式，则 batch 内样本分布不均的问题会导致子塔训练不稳定，收敛到局部最优。**单塔单输出**的训练方式在实际训练时效果稳定，收敛速度较快，是 D2Q 实现的基线版本。
- 图 c：单塔单输出模型中引入 Duration bias 模块，用于建模不同分桶下的样本差异（Res-D2Q），离线训练指标得到进一步的提升。

论文使用 XAUC、XGAUC 以及 MAE 等指标对时长回归效果进行评估。MAE 表示短视频预估时长与观看时长 label 的误差绝对值，表示模型回归精度，是回归任务的常用评估指标。

- XAUC：将测试集中的样本两两组合，若组合的标签和预估值的序一致则为正序，否则为逆序，XAUC 是正序对数与总组合数的比值；
- XGAUC：用户维度计算的 XAUC。

由于推荐系统主要优化候选集的排序，评估指标 XAUC 能够更加直观的反映预估时长序的好坏，与论文的优化目标更加适配。

8 多目标

8.1 多目标 + 推荐综述

Multi-task 多任务模型在推荐算法中应用总结 1

Multi-task 多任务学习在推荐算法中应用 (2)

多任务学习在推荐算法中的应用

8.2 阿里多目标

阿里提出多目标优化全新算法框架，同时提升电商 GMV 和 CTR

8.3 Youtube 多目标——MMoE

YouTube 多目标排序系统：如何推荐接下来收看的视频

https://daiwk.github.io/posts/dl-youtube-multitask.html

8.4 CGC

cgc 参考 paddle 代码：cgc_demo.py

9 多场景

9.1 APG

APG: 面向 CTR 预估的自适应参数生成网络

摘要：目前基于深度学习的 CTR 预估模型（即 Deep CTR Models）被广泛的应用于各个应用中。传统的 Deep CTR Models 的学习模式是相对静态的，即所有的样本共享相同的网络参数。然而，由于不同样本的特征分布不尽相同，这样一种静态方式很难刻画出不同样本的特性，从而限制了模型的表达能力，导致次优解。在本文中，我们提出了一个高效率、高效果的通用模块，称为自适应参数生成网络 (APG)。其可以基于不同的样本，动态的为 CTR 模型生成不同的模型参数。大量的实验表明，APG 能够被应用于各种 CTR 模型，并且显著的提升模型效果，同时能节省 38.7% 的时间开销和 96.6% 的存储。APG 已在阿里巴巴搜索广告系统部署上线，并获得 3% 的点击率增长和 1% 的广告收入增长。

APG: Adaptive Parameter Generation Network for Click-Through Rate Prediction

10 item 冷启

poso

Personalized Cold Start Modules for Large-scale Recommender Systems

https://zhuanlan.zhihu.com/p/534056942

11 用户冷启

11.1 PeterRec

仅需少量视频观看数据，即可精准推断用户习惯：腾讯、谷歌、中科大团队提出迁移学习架构 PeterRec

Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation

https://github.com/fajieyuan/sigir2020_peterrec

搞一个 pretrain-finetune 的架构，学好一套用户的表示，可以给各种下游任务用。

采用如下方式：

- **无监督**地学习用户表示：使用**序列模型**，**预测**用户的**下一次点击**。为了能建模**超长**的 u-i 交互序列，使用类似 NextItNet（A Simple Convolutional Generative Network for Next Item Recommendation）的模型
- 使用预训练好的模型去**有监督**地 finetune 下游任务
- 在各个下游任务间，想要尽可能共享更多的网络参数：参考 learning to learn，即一个网络的大部分参数可以其他参数来预测（一层里 95% 的参数可以通过剩下的 5% 的参数来预测）。文章提出了 model patch(模型补丁)，每个模型补丁的参数量不到原始预训练模型里的卷积层参数的 10%。通过加入模型补丁，不仅可以保留原来的预训练参数，还可以更好地适应下游任务。模型补丁有串行和并行两种加入方式。

序列推荐模型：

- RNN：强序列依赖
- CNN：可并行，能比 RNN 叠更多层，所以准确率更高。难以建模长序列是因为卷积核一般都比较小（如 3x3），但可以通过空洞 (dilated) 卷积来解决，可以使用不变的卷积核，指数级地扩充表示域。
- 纯 attention：可并行，例如 SASRec（Self-attentive sequential recommendation）。但因为时间和存储消耗是序列长度的平方的复杂度。

考虑到用户的点击序列往往成百上千，所以使用类似 NextItNet 的 casual 卷积，以及类似 GRec（Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation）的双向 encoder 的这种 non-casual 卷积。

与推荐系统现有的 transfer learning 对比：

- DUPN：
  - 训练的时候就有多个 loss。如果没有相应的 loss 和 data，学好的用户表示效果就会很差。而本文只有一个 loss，却能用在多个 task 上，所以算是一种 multi-domain learning(Efficient parametrization of multi-domain deep neural networks)
  - DUPN 在用户和 item 特征上需要很多特征工程，并没有显式地对用户的行为序列建模
  - DUPN 要么 finetune 所有参数，要么只 finetune 最后一个分类层。PeterRec 则是对网络的一小部分进行 finetune，效果并不比全 finetune 差，比只 finetune 最后一个分类层要好很多

- CoNet：杨强提出的Conet: Collaborative cross networks for cross-domain recommendation
  - cross-domain 用于推荐的一个网络。同时训练 2 个目标函数，一个表示 source 网络，一个表示 target 网络。
  - pretrain+finetune 效果不一定好，取决于预训练的方式、用户表示的表达能力、预训练的数据质量等

预训练时没有 [TCL]，fintune 时加上。

- 原 domain$S$：有大量用户交互行为的图文或视频推荐。一条样本包括 $(u, x^u) \in \mathcal{S}$，其中，$x^u = \{x_1^u, ..., x_n^u\}$ $(x_i^u \in X)$ 表示用户的点击历史
- 目标 domain$T$：可以是用户 label 很少的一些预测任务。例如用户可能喜欢的 item、用户性别、用户年龄分桶等。一条样本包括 $(u, y) \in \mathcal{T}$，其中 $y \in \mathcal{Y}$ 是一个有监督的标签。

12 GNN+ 推荐

https://zhuanlan.zhihu.com/p/323302898

Graph Neural Networks in Recommender Systems: A Survey

Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions

https://daiwk.github.io/posts/dl-graph-representations.html

13 bias v.s. debias

推荐系统炼丹笔记：推荐系统 Bias 大全 | Debias 方法综述

13.1 position bias

搜索、推荐业务中 - position bias 的工业界、学术界发展历程 - 系列 1(共计 2)

推荐系统遇上深度学习 (七十一)-[华为] 一种消除 CTR 预估中位置偏置的框架

PAL: A Position-bias Aware Learning Framework for CTR Prediction in Live Recommender Systems

推荐系统之 Position-Bias 建模

14 工业界的一些推荐应用

14.1 dlrm

Facebook 深度个性化推荐系统经验总结 (阿里内部分享 PPT))

14.2 instagram 推荐系统

Facebook 首次揭秘：超过 10 亿用户使用的 Instagram 推荐算法是怎样炼成的?

https://venturebeat.com/2019/11/25/facebook-details-the-ai-technology-behind-instagram-explore/

Instagram 个性化推荐工程中三个关键技术是什么?

14.3 微信读书推荐系统

微信读书怎么给你做推荐的?

14.4 youtube 推荐梳理

一文总览近年来 YouTube 推荐系统算法梳理

15 其他

15.1 混合推荐架构

混合推荐系统就是多个推荐系统“大杂烩”吗?

15.2 认知推荐

NeurIPS 2019 | 从感知跃升到认知，这是阿里在认知智能推荐领域的探索与应用

Learning Disentangled Representations for Recommendation