

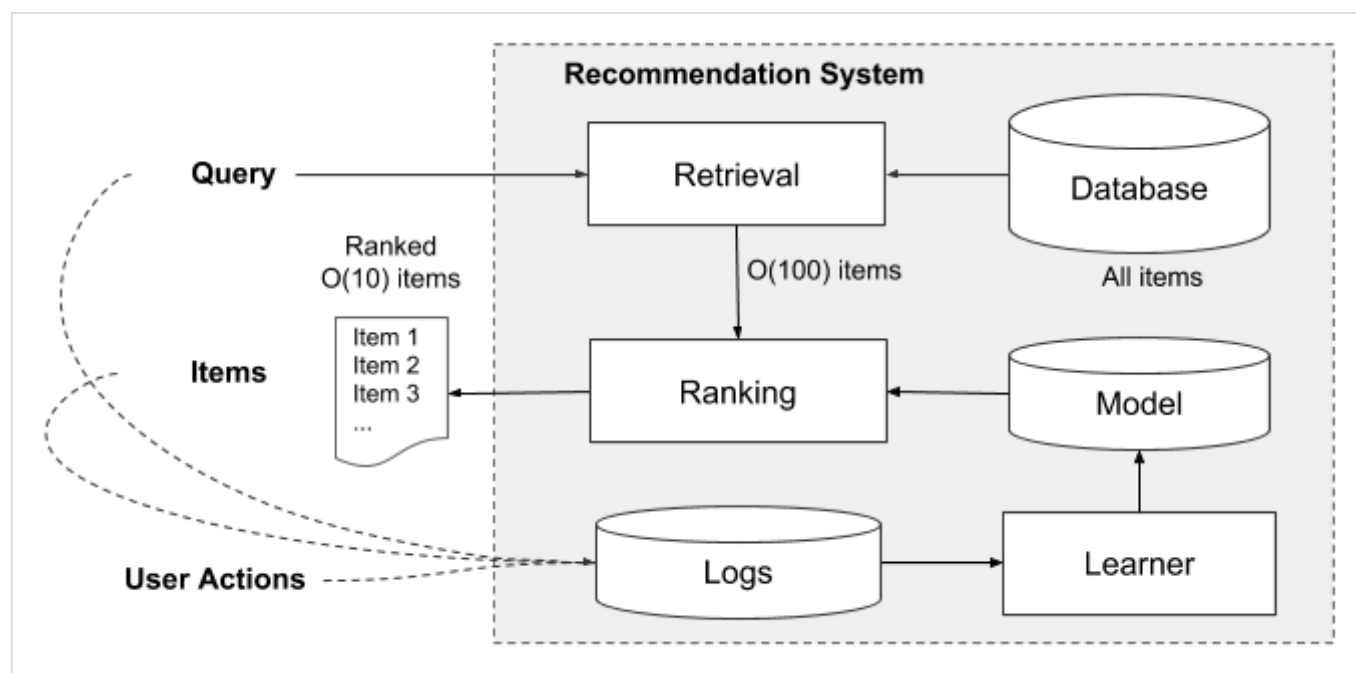
徐阿衡

论文笔记 - Wide and Deep Learning for Recommender Systems

📅 2017-03-13 | 📁 NLP, Recommender Systems | 📄 1045

Google Play 用的深度神经网络推荐系统，主要思路是将 Memorization(Wide Model) 和 Generalization(Deep Model) 取长补短相结合。论文见 [Wide & Deep Learning for Recommender Systems](#)

Overview of System



先来看一下推荐系统的整体架构，由两个部分组成，**检索系统(或者说候选生成系统)** 和 **排序系统(排序网络)**。首先，用 **检索(retrieval)** 的方法对大数据集进行初步筛选，返回最匹配 query 的一部分物品列表，这里的检索通常会结合采用 **机器学习模型(machine-learned models)** 和 **人工定义规则(human-defined rules)** 两种方法。从大规模样本中召回最佳候选集之后，再使用 **排序系统** 对每个物品进行算分、排序，分数 $P(y|x)$ ， y 是用户采取的行动(比如说下载行为)， x 是特征，包括

- **User features**
e.g., country, language, demographics
- **Contextual features**
e.g., device, hour of the day, day of the week

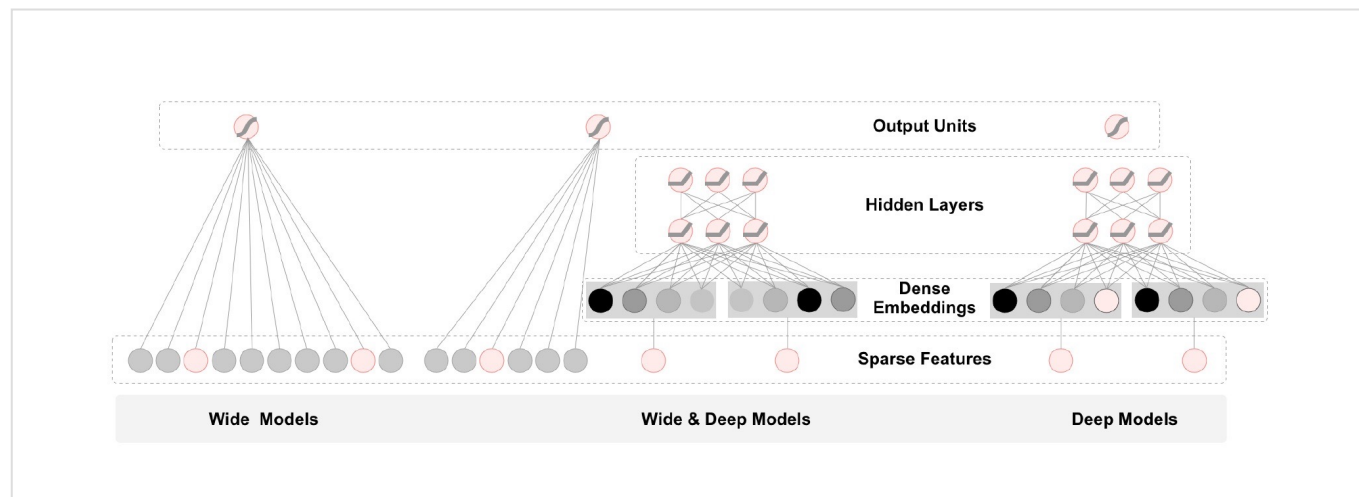
- **Impression features**

e.g., app age, historical statistics of an app

WDL 就是用在排序系统中。

Wide and Deep Learning

简单来说，人脑就是一个不断记忆（memorization）并且归纳（generalization）的过程，而这篇论文的思想，就是将宽线性模型（Wide Model，用于记忆，下图左侧）和深度神经网络模型（Deep Model，用于归纳，下图右侧）结合，汲取各自优势形成了 Wide & Deep 模型用于推荐排序（下图中间）。



Wide Model

Memorization can be loosely defined as learning the frequent co-occurrence of items or features and exploiting the correlation available in the historical data.

要理解的概念是 **Memorization**，主要是学习特征的共性或者说相关性，产生的推荐是和已经有用户行为的物品直接相关的物品。

用的模型是 **逻辑回归(logistic regression, LR)**，LR 的优点就是简单(simple)、容易规模化(scalable)、可解释性强(interpretable)。LR 的特征往往是二值且稀疏的(binary and sparse)，这里同样采用 one-hot 编码，如 “user_installed_app=netflix” 这个特征的值为 1，如果用户安装了 Netflix，否则为 0。

为了达到 Memorization，我们对稀疏的特征采取 cross-product transformation，比如说 AND(user_installed_app=netflix, impression_app=pandora) 这个特征，只有 Netflix 和 Pandora 两个条件都达到了，值才为 1，这类 feature 解释了 co-occurrence 和 target label 之间的关系。一个 cross-product transformation 的局限在于，对于在训练集里没有出现过的 query-item pair，它不能进行泛化(Generalization)

到此，总结一下，宽度模型的输入是用户安装应用(installation)和为用户展示（impression）的应用间的向量积（叉乘），模型通常训练 one-hot 编码后的二值特征，这种操作不会归纳出训练集中未出现的特征对。

Linear model 大家都很熟悉了

$$y = w^T x + b$$

$x = [x_1, x_2, \dots, x_d]$ 是包含了 d 个特征的向量, $w = [w_1, w_2, \dots, w_d]$ 是模型参数, b 是偏置。特征包括了原始的输入特征以及 cross-product transformation 特征, cross-product transformation 的式子如下:

$$\phi_k(x) = \prod_{i=1}^d x_i^{c_{ki}}$$

c_{ki} 是一个布尔变量, 如果第 i 个特征是第 k 个 transformation ϕ_k 的一部分, 那么值就为 1, 否则为 0, 作用:

This captures the interactions between the binary features, and adds nonlinearity to the generalized linear model.

Deep Model

Generalization is based on transitivity of correlation and explores new feature combinations that have never or rarely occurred in the past.

要理解的概念是 **Generalization**, 可以理解为相关性的传递(transitivity), 会学习新的特征组合, 来提高推荐物品的多样性, 或者说提供泛化能力(Generalization)

泛化往往是通过学习 low-dimensional dense embeddings 来探索过去从未或很少出现的新的特征组合来实现的, 通常的 embedding-based model 有 **Factorization Machines(FM)** 和 **Deep Neural Networks(DNN)**。特殊兴趣或者小众爱好的用户, query-item matrix 非常稀疏, 很难学习, 然而 dense embedding 的方法还是可以得到对所有 query-item pair 非零的预测, 这就会导致 over-generalize, 推荐不怎么相关的物品。这点和 LR 正好互补, 因为 LR 只能记住很少的特征组合。

为了达到 **Generalization**, 我们会引入新的小颗粒特征, 如类别特征 (安装了视频类应用, 展示的是音乐类应用, 等等) AND(user_installed_category=video, impression_category=music), 这些高维稀疏的类别特征 (如人口学特征和设备类别) 映射为低维稠密的向量后, 与其他连续特征 (用户年龄、应用安装数等) 拼接在一起, 输入 MLP 中, 最后输入逻辑输出单元。

一开始嵌入向量(embedding vectors)被随机初始化, 然后训练过程中通过最小化损失函数来优化模型。每一个隐层(hidden-layer)做这样的计算:

$$a^{(l+1)} = f(W^{(l)} a^{(l)} + b^{(l)})$$

f 是激活函数(通常用 ReLU), l 是层数。

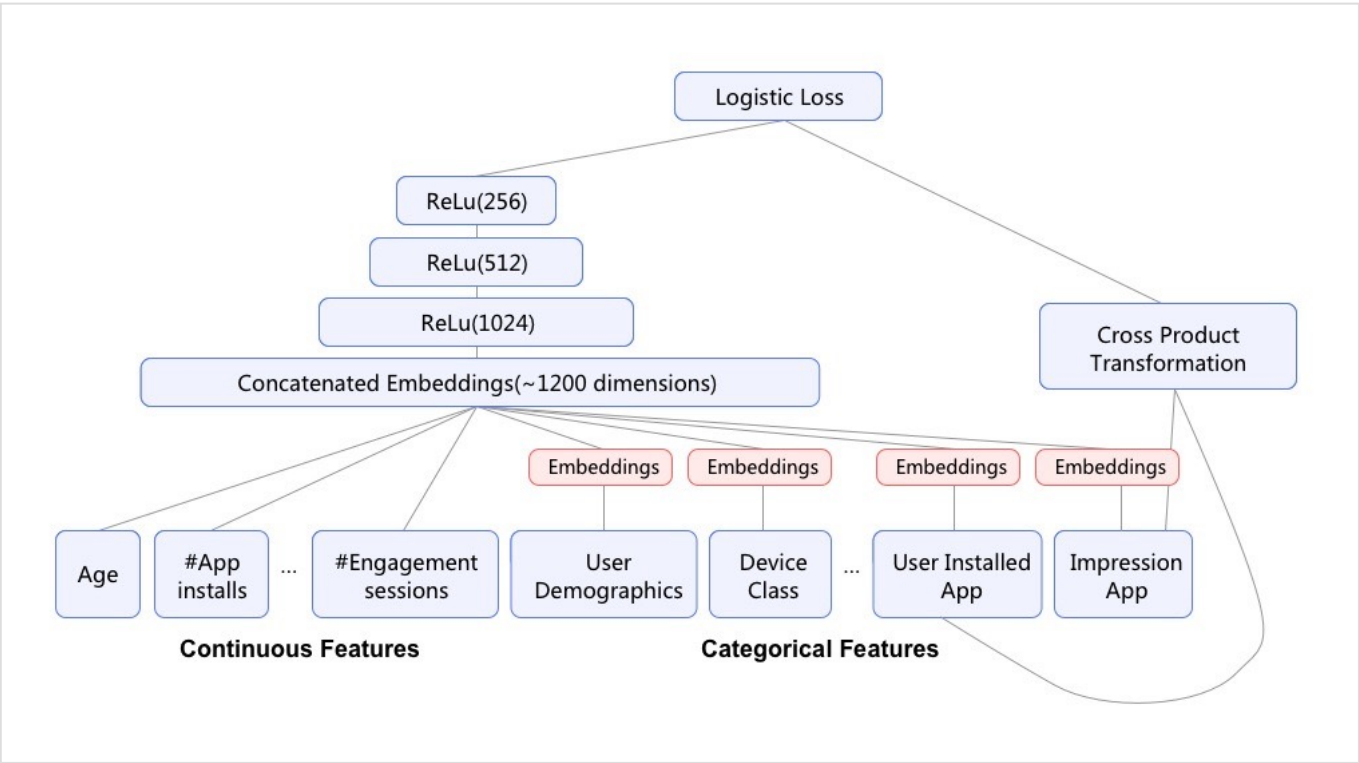
总结一下，基于 embedding 的深度模型的输入是 类别特征(产生embedding)+连续特征。

Joint Training

对两个模型的输出算 log odds ratio 然后加权求和，作为预测。

Joint Training vs Ensemble

- Joint Training 同时训练 wide & deep 模型，优化的参数包括两个模型各自的参数以及 weights of sum
- Ensemble 中的模型是分别独立训练的，互不干扰，只有在预测时才会联系在一起



用 mini-batch stochastic optimization 来进行训练，可以看下这篇论文[Efficient Mini-batch Training for Stochastic Optimization](#)。

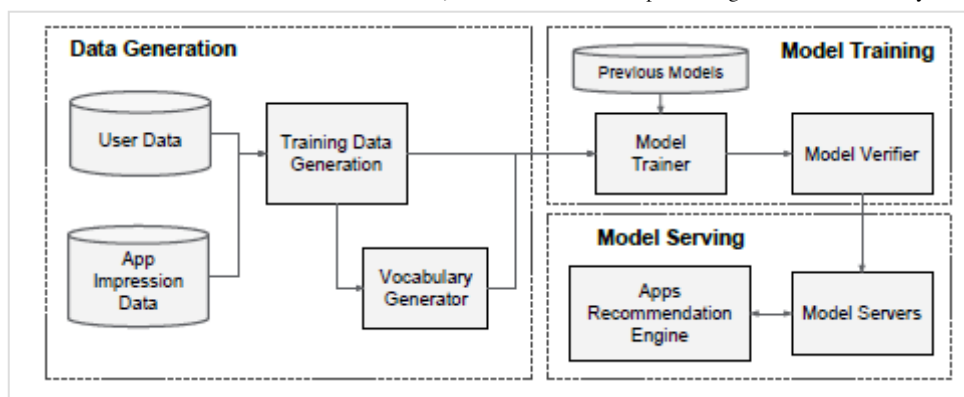
在论文提到的实验中，训练时 Wide Model 部分用了 [Follow-the-regularized-leader\(FTRL\)](#)+ L1 正则，Deep Model 用了 [AdaGrad](#)，对于逻辑回归，模型预测如下：

$$P(Y = 1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^T[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^T \mathbf{a}^{(l_f)} + b) \quad (3)$$

where Y is the binary class label, $\sigma(\cdot)$ is the sigmoid function, $\phi(\mathbf{x})$ are the cross product transformations of the original features \mathbf{x} , and b is the bias term. \mathbf{w}_{wide} is the vector of all wide model weights, and \mathbf{w}_{deep} are the weights applied on the final activations $\mathbf{a}^{(l_f)}$.

System Implementation

pipeline 如下图



Data Generation

Label: 标准是 app acquisition, 用户下载为 1, 否则为 0

Vocabularies: 将类别特征(categorical features)映射为整型的 id, 连续的实值先用累计分布函数CDF归一化到 [0,1], 再划档离散化。

Continuous real-valued features are normalized to [0, 1] by mapping a feature value x to its cumulative distribution function $P(X \leq x)$, divided into n_q quantiles. The normalized value is $\frac{i-1}{n_q-1}$ for values in the i -th quantiles.

Model Training

训练数据有 500 billion examples, Input layer 会同时产生稀疏(sparse)的和稠密(dense)的特征, 具体的 Model 上面已经讨论过了。需要注意的是, 当新的训练数据来临的时候, 我们用的是热启动(warm-starting)方式, 也就是从之前的模型中读取 embeddings 以及 linear model weights 来初始化一个新模型, 而不是全部推倒重新训练。

Model Serving

当模型训练并且优化好之后, 我们将它载入服务器, 对每一个 request, 排序系统从检索系统接收候选列表以及用户特征, 来为每一个 app 算分排序, 分数就是前向传播的值(forward inference)啦, 可以并行训练提高 performance。

参考链接

[《Wide & Deep Learning for Recommender Systems》笔记](#)

[深度学习第二课：个性化推荐](#)

坚持原创技术分享, 您的支持将鼓励我继续创作!

赏

#Recommender Systems

论文笔记 - LightRNN - Memory and Computation-
Efficient Recurrent Neural Networks

◀ 深度学习知识框架

© 2016 - 2017 ♥ 徐阿衡

Powered by [Hexo](#) | Theme - [NexT.Mist](#)

👤 14104 | 👁 29668