概述

这篇论文是对YouTube中基于DNN的推荐系统的整体描述。本文旨在对论文进行总结。

整个系统主要分 Candidate Generation 和 Ranking，也就是召回和排序两部分。召回模块从数百万的视频集合中挑选出数百个候选视频；排序模块则是从数百个中挑选出几十个视频，并排序后推送给用户。

下图是整体框架：

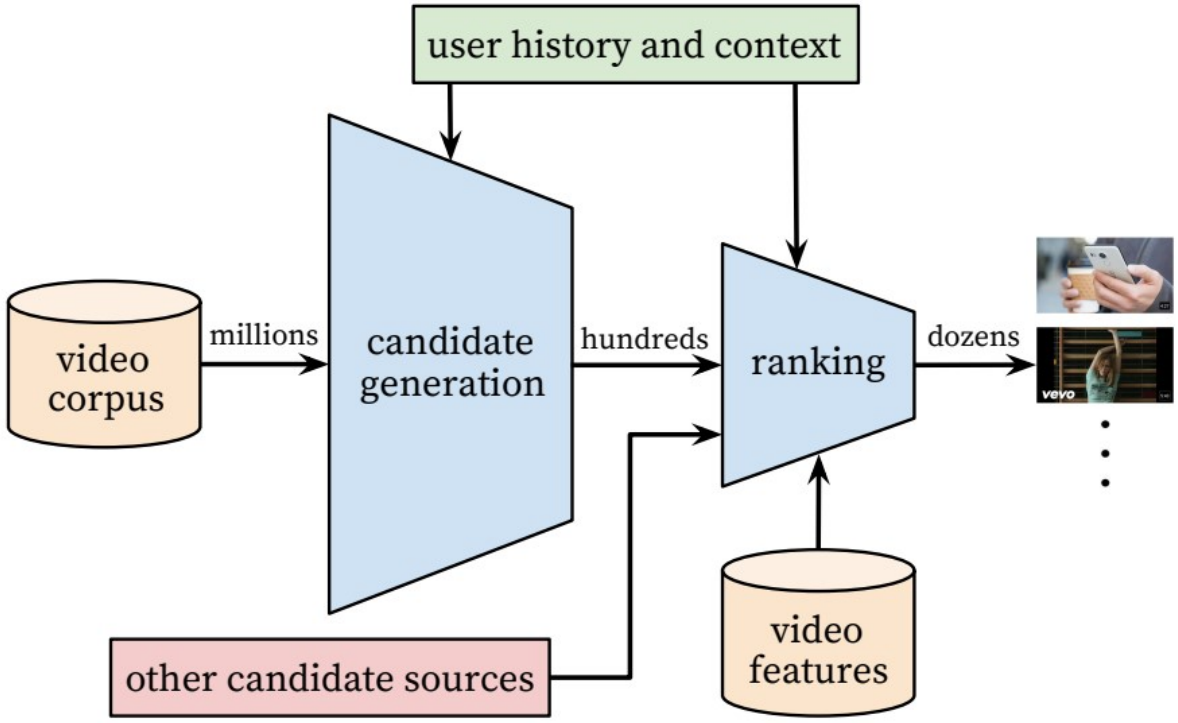


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

一、召回

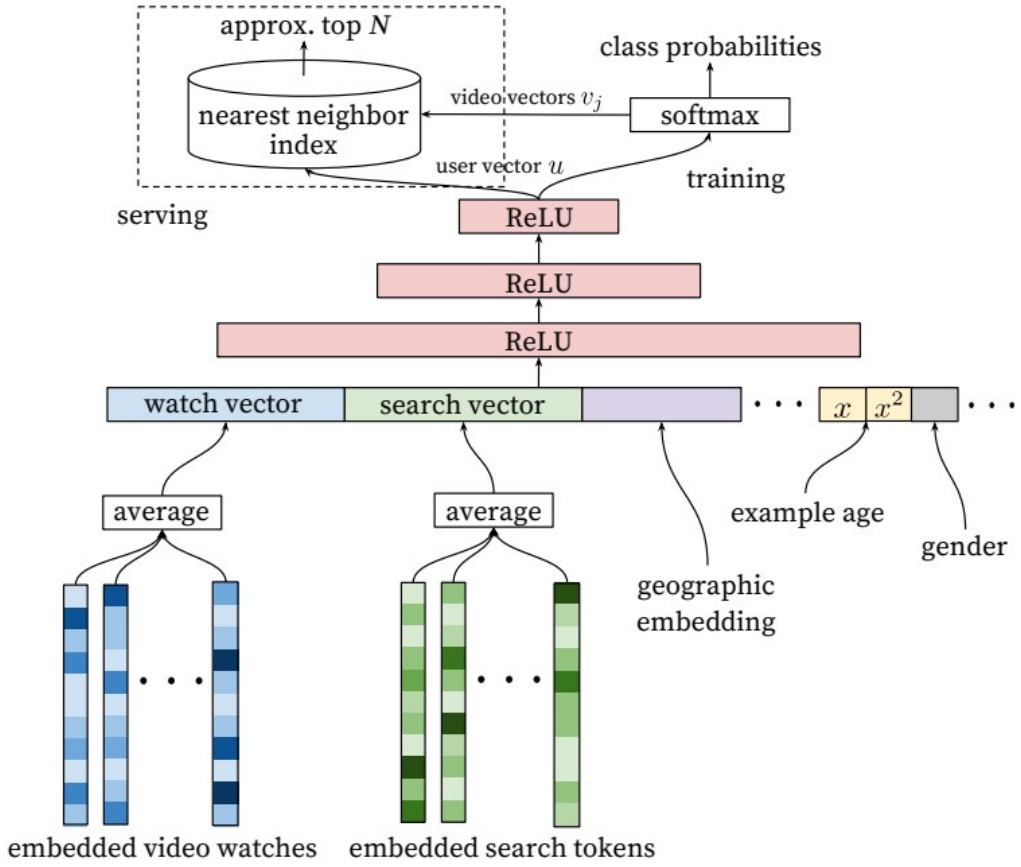


Figure 3: Deep candidate generation model architecture showing embedded sparse features concatenated with dense features. Embeddings are averaged before concatenation to transform variable sized bags of sparse IDs into fixed-width vectors suitable for input to the hidden layers. All hidden layers are fully connected. In training, a cross-entropy loss is minimized with gradient descent on the output of the sampled softmax. At serving, an approximate nearest neighbor lookup is performed to generate hundreds of candidate video recommendations.

1. 问题定义

YouTube将召回问题转化为一个多分类问题去处理，建模以预测：在t时刻发生的某次视频观看事件w_t中，具体观看的是视频集合V中的哪个视频。

假设用U表示这次事件中的用户，用C表示上下文，用u表示对U、C一起进行embedding后的特征向量，用v_i表示对视频i进行embedding后的特征向量。则我们需要预测的分类到视频i的概率可以形式化如下：

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

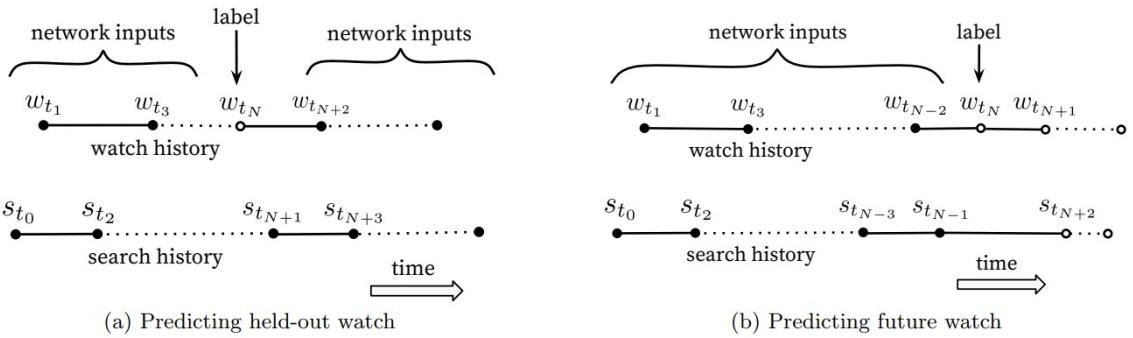
通过这样的问题转化以后，理想情况下，我们能预测到在当前用户、当前情景下，每个视频被观看的概率。取概率最高的前M个视频即可作为召回模块的输出。

2. 数据准备

在准备数据样本时，需要注意：

1. 数据源方面，采用所有YouTube视频的观看事件（包含如嵌在其他网站的视频等），而不仅仅是YouTube主站上的。
2. 对每个用户带来的训练样本数进行了限制，从而避免高活跃用户对模型的过度影响。

3. 注意避免样本数据中掺入未来信息，模型的输入应该始终只有打标签以前的数据。



(a) Predicting held-out watch (b) Predicting future watch

Figure 5: Choosing labels and input context to the model is challenging to evaluate offline but has a large impact on live performance. Here, solid events • are input features to the network while hollow events ○ are excluded. We found predicting a future watch (5b) performed better in A/B testing. In (5b), the example age is expressed as $t_{\max} - t_N$ where $t_{\max}$ is the maximum observed time in the training data.

3. 特征处理

特征方面，抽象来看，主要涉及用户属性、用户行为与事件时间特征三大块。作者在论文中给出了不同特征组合的效果对比：

- Depth 0: A linear layer simply transforms the concatenation layer to match the softmax dimension of 256
- Depth 1: 256 ReLU
- Depth 2: 512 ReLU → 256 ReLU
- Depth 3: 1024 ReLU → 512 ReLU → 256 ReLU
- Depth 4: 2048 ReLU → 1024 ReLU → 512 ReLU → 256 ReLU

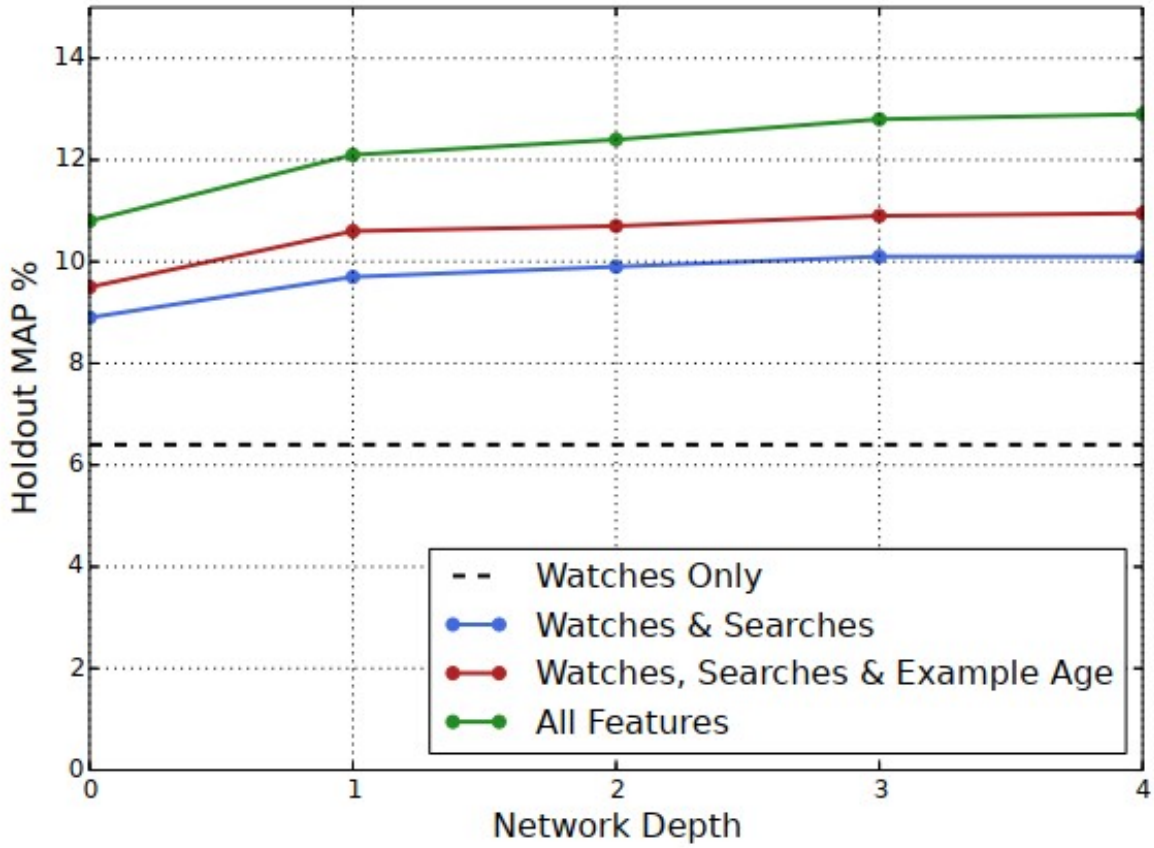


Figure 6: Features beyond video embeddings improve holdout Mean Average Precision (MAP) and layers of depth add expressiveness so that the model can effectively use these additional features by modeling their interaction.

3.1 用户属性特征

用户属性特征在论文中只是简单的一笔带过，包括：用户处理的地理位置、设备、性别、登录状态、年龄。直觉来看，这类特征应该对新用户的推荐效果有着重要影响。

尽管没有细讲，从最后对不同特征组合的实验来看，却似乎带有很大的提升："All Features"相对于"Watches, Searches & Example Age"有显著提升。

3.2 用户行为特征

对用户行为的特征挖掘，主要从用户的视频观看历史（"watch vector"）与搜索历史（"search vector"）着手。

"watch vector"

从用户的视频观看历史中挖掘特征主要分两步：

1. 通过单独的模型预训练好每个视频的embedding。
2. 取出用户历史（*All or Top-k?*）观看的视频的embedding取均值，作为 "watch vectors"。

具体而言，是如何做embedding的呢？文中只是简单的提了一下：

> Inspired by continuous bag of words language models, we learn high dimensional embeddings for each video in a fixed vocabulary

从我的理解来看，应该是将每个用户历史观看的视频ID序列，看作一个“句子”，所有用户的“句子”汇聚成一个语料集合；进而参考Word2Vec中基于CBOW的训练方式来做训练，从而获得每个视频的embedding。

作者还提到了一句：

> Importantly, the embeddings are learned jointly with all other model parameters through normal gradient descent backpropagation updates

"search vector"

从用户的搜索历史中挖掘特征的步骤，与前面相似：

1. 将每个query分词成unigrams跟bigrams，而token又是被embedding好的；
2. 汇总所有的这些embedding求均值，作为 "search vector"；

> Search history is treated similarly to watch history - each query is tokenized into unigrams and bigrams and each token is embedded. Once averaged, the user’s tokenized, embedded queries represent a summarized dense search history

从作者的描述来看，应该就是基于用户的搜索预料来训练Word2Vec模型，从而得到embedding向量。

3.3 事件时间特征

"Example Age" 是个较为特殊的特征。引入这个特征，是因为作者观察到，用户更偏好新产的视频。

> we feed the age of the training example as a feature during training. At serving time, this feature is set to zero (or slightly negative) to reflect that the model is making predictions at the very end of the training window.

论文在一张插图的描述中提到：

> the example age is expressed as $t_{max} - t_N$ where t_{max} is the maximum observed time in the training data.

t_N指的是样本打标签的时间，也就是当前的事件的时间戳，这个好理解。

虽然说得比较模糊，但结合前面的描述：在serving时，该特征被置为零。所以t_{max}应该是指全体训练样本中的最大观测时间。

至于具体是用秒？分钟？小时？还是天？则没有提及，考虑到不同量纲之间可以通过线性变换来相互切换，所以这个问题的影响不大。

作者通过统计分析表明，模型在加入了"Example Age"之后，能比较好的捕捉到视频上传时间的影响。

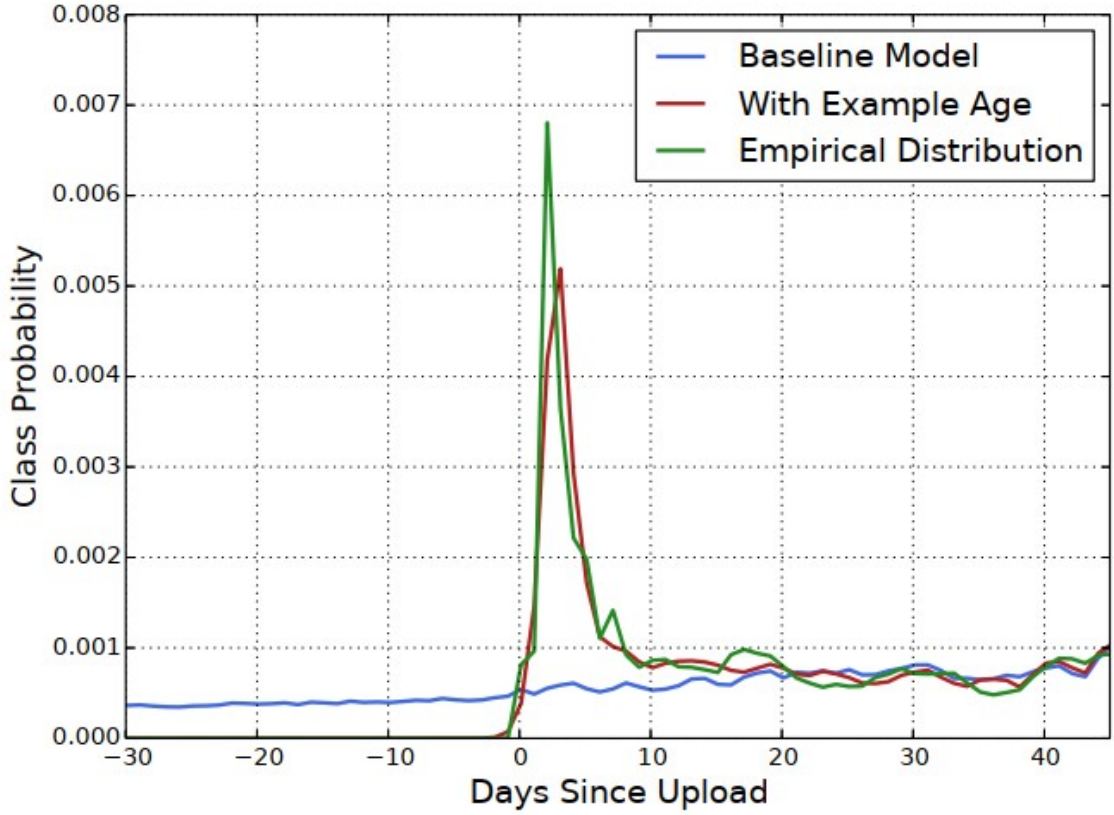


Figure 4: For a given video [26], the model trained with example age as a feature is able to accurately represent the upload time and time-dependant popularity observed in the data. Without the feature, the model would predict approximately the average likelihood over the training window.

4. 模型训练与线上服务

4.1 训练技巧： Negative Sampling

一般情况下，基于 SoftMax 的 Cross-Entropy Loss 形式如下：

$$logit(i) = \frac{exp(w_i x)}{\sum_j^M exp(w_j x)}$$

$$loss = -log(logit(i)) = -(w_i x) + log(\sum_j^M exp(w_j x))$$

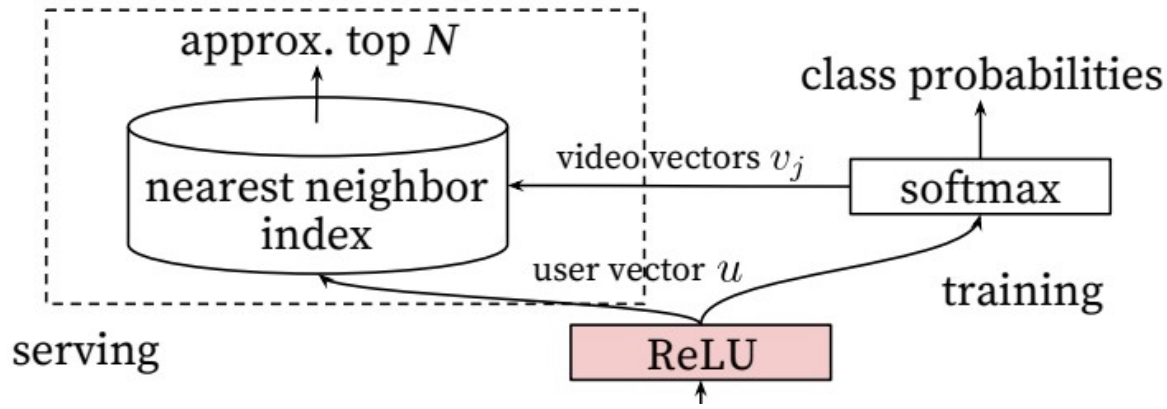
可以看到，当类别数M多达数百万的时候，损失函数的后半部分$log(\sum_j^M exp(w_j x))$的计算量将会特别大。

而 Negative Sampling 的思路则是，通过采样指定K个类别，从而把计算量从$O(M) \rightarrow O(K)$控制了下来。作者在论文中指出，一般K取数千。

这里有几个细节：

1. K是否把类别i包含在内?
2. 具体如何进行随机采样？均匀采样?
3. 是每个训练样本都做一次采样？还是每个batch做一次采样?
4. 每次负采样、训练时，并不会更新K个被选中的类别以外的类别权重。那么如果存在某个类别的样本数量相对较大，会不会对模型效果有影响?

4.2 线上服务



模型框架图中的这个细节，是我一开始没有留意到的。

当时只是想当然的认为，在做serving时，每次用户来到时，跑一遍模型预测，然后取出概率值Top N的视频来召回。而从YouTube的框架图来看，实际做serving时是以下步骤：

1. 从最后一层ReLU层获取用户向量$\vec{u}$（256维）；
2. 从SoftMax层获取视频向量$\overrightarrow{v_j}$；
3. 通过最近邻搜索来找到近似的Top N视频。

以上的简要描述可能仍然不好理解。

我们知道，在ReLU和SoftMax两层之间存在一个大小为$(256, V)$的权重矩阵$\vec{W}$，V表示视频总数；$\vec{W}$通过训练学习到。

来看常规的feedward流程：

1. 计算至最后的ReLU层得到$\vec{u}$；
2. 进行矩阵乘法$\vec{z} = \vec{u}^T \vec{W}$；
3. 进行指数运算$exp(\vec{z})$；
4. 归一化$\vec{y} = exp(\vec{z}) / |||exp(\vec{z})|||_1$；
5. 按y_j进行倒序取Top-N视频作为召回结果；

观察到，由于指数运算具有单调性，且在进行召回时只关注模型输出的相对值，而不关注绝对值；我们发现3、4两步可以省略掉，直接在计算出$\vec{z}$之后，取z_j的值来作为排序的依据即可。

由于视频数量巨大，$\vec{z} = \vec{u}^T \vec{W}$这一步仍然存在高昂的计算成本。为了提升效率，在完成了模型训练之后，可以提前把$\vec{W}$拆成一个个列向量$\overrightarrow{v_j}$。

线上serving时，计算出用户向量$\vec{u}$之后，下一步就变成了寻找与$\vec{u}$内积最大的N个列向量$\overrightarrow{v_j}$的问题。而这可以转化为最近邻搜索问题（作者引用论文：An investigation of practical approximate nearest neighbor）。

二、排序

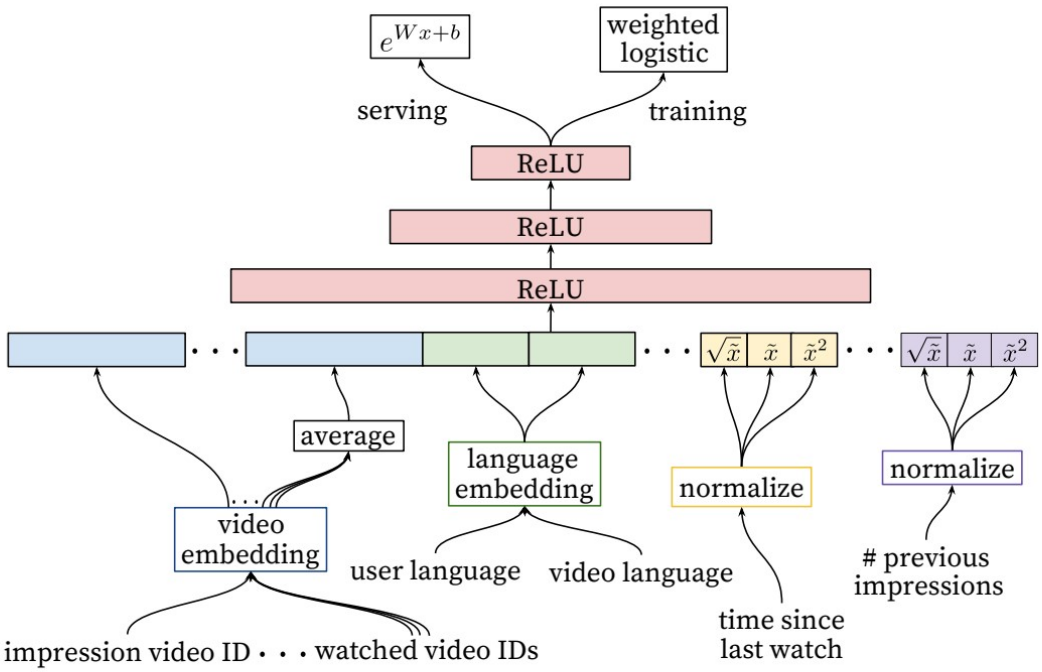


Figure 7: Deep ranking network architecture depicting embedded categorical features (both univalent and multivalent) with shared embeddings and powers of normalized continuous features. All layers are fully connected. In practice, hundreds of features are fed into the network.

1. 问题定义

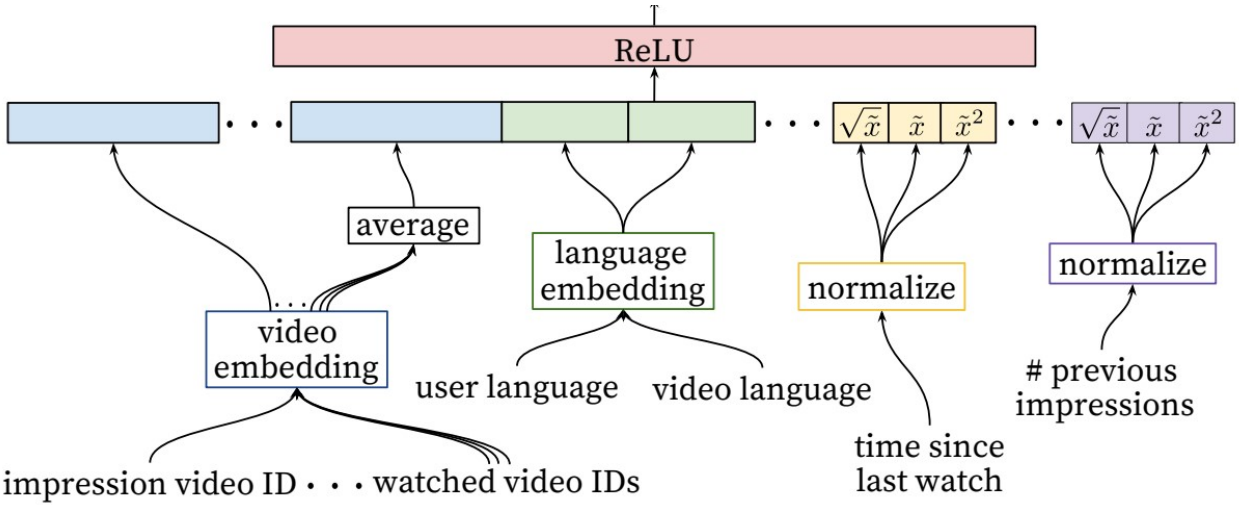
YouTube的推荐系统中，将排序问题转化为预测：给用户u_i曝光视频v_j后，用户的观看时长。

为什么不转化为预测CTR？因为光看CTR容易使模型偏好“标题党”或者“封面党”，进而影响用户体验、商业变现等。

2. 数据准备

对于每一次视频曝光事件（给用户u_i曝光视频v_j），如果用户点击观看了视频，则取视频观看时长T_i作为预测值；如果没有点击，则取单位值作为预测值。

3. 特征处理



论文中简单的按照特征值类型分别展开论述。

3.1 离散值特征

离散值特征需要进行embedding，在图中也展示了主要的两种：对视频ID的embedding，以及对文本的embedding。

"video embedding"

对于视频ID，先按照召回模块中相似的处理方式（是否完全一样？），单独训练得到video embedding，维度约为$klog(V)$。**注意：作者提到，对于点击次数较少的长尾视频，直接采用零向量作为embedding。**

- 对于模型输入的视频ID（有且仅有一个），直接取相应的embedding输入到网络中；
- 对于用户观看过的视频ID序列（全部或者最近K个？），获取相应的embedding取均值输入到网络中；

"language embedding"

图中说得很模糊，按我理解应该是指文本相关的特征，包括对"user language"、"video language"两块的embedding。

3.2. 连续值特征

对于连续值特征，YouTube采用了颇为特别的处理方式。

首先是对连续值特征进行正则化：假设x的分布函数是f，则通过$\tilde{x} = \int_{-\infty}^{x} df$进行正则化。式中的积分，通过基于特征值分位数的线性插值进行估计。更具体的操作论文中没有展开说。

> A continuous feature x with distribution f is transformed to $\tilde{x}$ by scaling the values such that the feature is equally distributed in [0, 1) using the cumulative distribution, $\tilde{x} = \int_{-\infty}^{x} df$. This integral is approximated with linear interpolation on the quantiles of the feature values computed in a single pass over the data before training begins.

其次是在正则化后的值基础上，还通过取平方$\tilde{x}^2$与开根号$\sqrt{\tilde{x}}$引入了两种特征值，进而引入了非线性特征。

架构图中明确指出进行了正则化的特征有两个：

1. "time since last watch"，也就是“距离上一次观看的时间”。但具体来讲，“上一次观看”是指“该视频上一次被任意用户观看的时间”？还是“该用户上一次观看任意视频的时间”？还是“该用户对该视频的上一次观看的时间”？不得而知。
2. "# previous impressions"，也就是“此前曝光的数量”。但具体来讲，“曝光”是指“给该用户的该视频的曝光次数”？还是“给该用户的任意视频的曝光次数”？还是“给任意用户的该视频的曝光次数”？这里我认为是第一种，因为论文在其他地方提到，如果已经给用户曝光过某视频但用户没有点击，那后面应该逐渐减少这个视频的推荐，进而从用户的角度看，推荐列表是在逐渐变化的。

3.3 其他

其他一些论文中提到了，但是没有放到图中的，大概有这些：

- 用户看过多少同频道的视频?
- 用户上一次看同频道或同主题的视频是什么时候?
- 用户过往与相似视频的交互特征特别重要
- 来自召回模块的特征
- 用户是否登录

4. 模型训练与线上服务

直觉来看，既然将排序问题转化为预测问题，似乎应该和常见的回归模型一样，用均方差等作为损失函数才对，而YouTube并没有这样做，而是用 Cross-Entropy 结合 Logistic Regression，为什么可以这么做呢?

我们知道，Sigmoid函数可以通过对对数几率进行线性回归推导得到：

$$odds = \frac{\hat{y}}{1-\hat{y}}$$

$$log(odds) = log(\frac{\hat{y}}{1-\hat{y}}) = \vec{w}^T\vec{x} + b$$

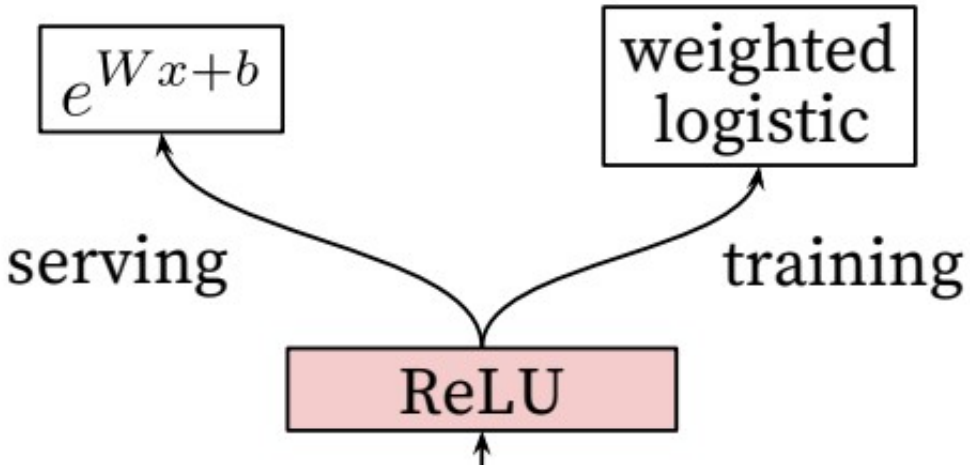
$$\hat{y} = \frac{1}{1+exp(-(\vec{w}^T\vec{x}+b))}$$

上面的推导中，出发点是对$odds$的定义，我们将其定义为正样本概率与负样本概率的比例，值越大说明正负样本概率之间差距越大。对第二个式子做简单变换，得到： $odds = exp(\vec{w}^T\vec{x}+b)$。

前面的$odds$我们可以认为是对点击事件的几率进行计算，也就是$odds(Click)$。下面来考虑基于视频观看时长计算几率。我们假设样本总数为N，其中正样本（点击并观看视频）的数量为K，正样本中视频观看时长记为T_i，负样本的视频观看时长统一认为是1，则$odds(WatchTime)$如下：

$$
\begin{aligned}
odds &= \frac{E[T|Clicked]}{E[T|NotClicked]} = \frac{\frac{\sum_i^K T_i}{N-K+\sum_i^K T_i}}{\frac{N-K}{N-K+\sum_i^K T_i}} \\
&= \frac{\sum_i^K T_i}{N-K} \\
&= \frac{\sum_i^K T_i}{N} * \frac{N}{N-K} \\
&= \frac{\sum_i^K T_i}{N} * \frac{1}{1-K/N} \\
&= \frac{E[T]}{1-ctr}
\end{aligned}
$$

于是，当ctr较小时，$odds$是接近于$E[T]$的；而YouTube框架图中的这看似诡异的部分，背后思想则源于此：



serving时采用几率$odds$，而不是$sigmoid$来作为对视频观看时长的近似。

而训练时，采用 Weighted Logistic Regression：对正样本按T_i加权，对负样本按1加权。

结语

把这篇论文读下来，零零散散花了我三四天；整理成脑图的过程中，陆续发现了很多细节问题，又花了一天；写博客的过程中，抠细节、查资料、补知识点，花了三天时间。这一番折腾下来实在太累，好在YouTube的这篇论文也完全值得我这番精读。

目前文中还是不得已的留下了很多未找到答案的疑问，留着后续慢慢填坑了。

参考

1. 论文地址
2. 王喆-整体介绍
3. 王喆-十个工程问题
4. 王喆-模型Serving
5. 工程再现