2018-09-02 由ADMIN

中文版《Deep Neural Networks for YouTube Recommendation》

读了《Deep Neural Networks for YouTube Recommendation》，正在实现这个系统的过程中，总觉得得留下点什么来帮助自己加深理解。刚刚用扇贝测试了下英文词汇量3300（呵呵，大学4年学的英语可能是假的），所以水平相当有限，一边抱着google translate，一边试着翻译下吧，如有不通顺的地方敬请指正。

YouTube = You （你）Tube（原意：真空管电视机） 赶脚这里的Tube应该是动词，所以我山寨的翻译成“你来做视频”吧。和YouTube的Logo下方的「Broadcast Yourself!」相呼应，加一起就是“你型你秀”。

《YouTube深度神经网络推荐系统》

摘要

youtube 代表了目前规模最大、最复杂的工业推荐系统之一。在这篇文章里，我们从系统的角度上重点讲述深度学习带来的巨大效果提升。根据经典的信息检索二分法，本文分为2阶段：首先，我们详细描述了一个候选集生成模型和一个深度排序模型。然后，我们还提供了一些从设计、迭代和维护庞大用户量的推荐系统中得到的实战经验和见解。

关键词

推荐系统；深度学习；可扩展性

1. 介绍

youtube是世界上最大的生产、分享、发现视频内容的平台。youtube推荐系统旨在帮助10亿多用户从不断增长的视频集中发现个性化的视频内容。在本文中，我们将重点介绍深度学习最近在YouTube视频推荐系统上的巨大影响。图1显示了YouTube app首页上的推荐。

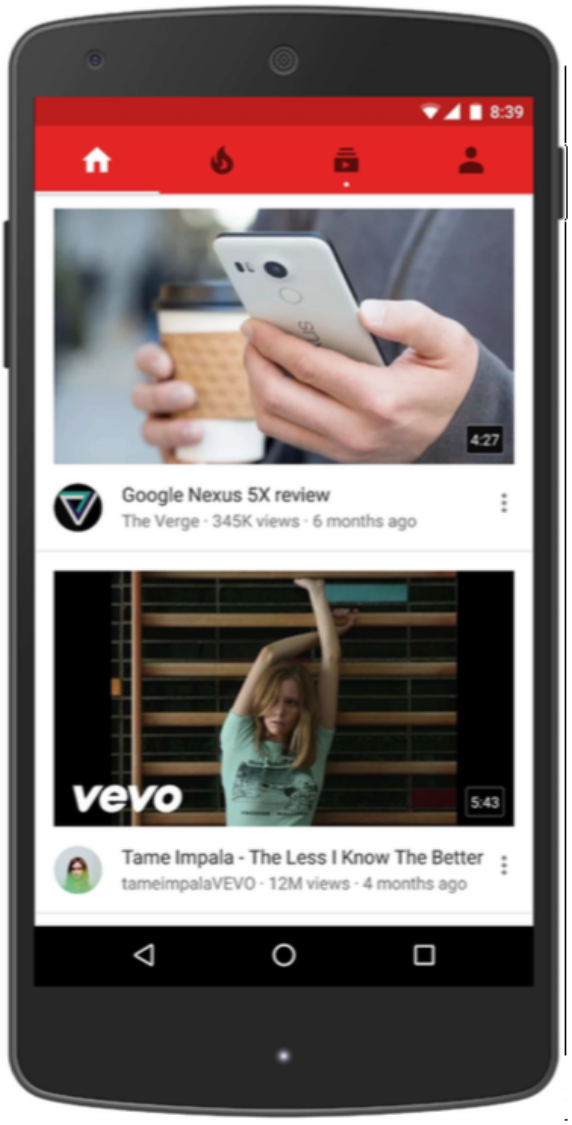


Figure 1: Recommendations displayed on YouTube mobile app home.

（接下来的是吹牛逼阶段：youtube推荐是多么难）

推荐YouTube视频的挑战性从三个主要方面来看：

— 规模：许多已经被证明在小问题上表现良好的推荐算法不能在我们的规模上起作用。 高度专业化的分布式学习算法和高效服务系统对于处理YouTube庞大的用户群和语料库至关重要。

— 新鲜度：YouTube具有非常动态的语料库，每秒钟上传的视频时长很多。 推荐系统应该能在有新视频上传或者用户有新行为的时候及时响应。 平衡新内容与成熟的视频可以从探索/开发的角度理解。

–噪音：YouTube上的历史用户行为由于稀疏性和各种不可观察的外部因素而不可预测。 我们很少能获得基本真实的用户满意度，更多的是隐式反馈噪声信号。 此外，与内容相关联的元数据结构不良，没有良好定义的本体。 基于我们的训练数据的这些特定特性,我们的算法需要是鲁棒的。

（接下来这一段是说：本文提到的推荐系统是实现在tensorflow上的，youtube现在对于学习问题都开始用deep learning了。）

与Google的其他产品领域相结合，YouTube已经经历了一个基本的范式转变，使用深度学习已经成为几乎所有学习问题的通用解决方案。我们的系统建立在Google Brain [4]基础上，最近开源为TensorFlow [1]。 TensorFlow为使用大规模分布式训练的各种深层神经网络架构提供了一个灵活的框架。

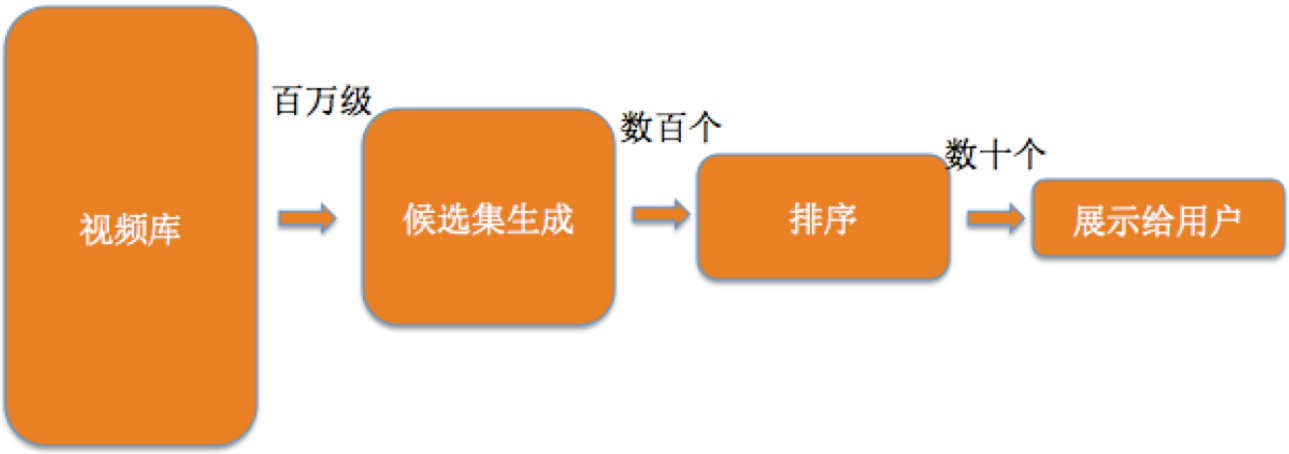
（接下来这一段是讲：深度学习在推荐系统中的研究现状，衬托一下本文在推荐领域还是挺有开创性的，老外写论文也都是套路，无非还是赋比兴这一套，哈哈）

与大量的矩阵分解方法研究[19]相比，很少有研究是关于使用深层神经网络做推荐系统的。 神经网络在[17]中用于推荐新闻，在[8]中被引用和在[20]中的评论。 协作过滤在[22]中被建立为深层神经网络，在[18]中被自动编码。 Elkahky 等使用深层学习进行跨域用户建模[5]。 在基于内容方面，Burges等人使用深层神经网络进行音乐推荐[21]。

（接下来是：剧透一下具体要讲啥）

本文的组织结构如下：第2节介绍了一个简要的系统概述。第3节更详细地描述了候选集生成模型，包括如何对其进行训练并用于提供推荐服务。 实验结果将显示模型如何从深层的隐藏单元和其他的异质信号中受益。 第4节详细说明了排序模型，包括如何修改经典逻辑回归以训练预测预期观察时间（而不是点击概率）的模型。 实验结果表明，隐层深度在这种情况下也是有帮助的。 最后，第5节介绍了我们的结论和经验教训。

2. 系统概述



我们的推荐系统的整体结构如图2所示。该系统由两个神经网络组成：一个用于候选集的生成，一个用于排序。

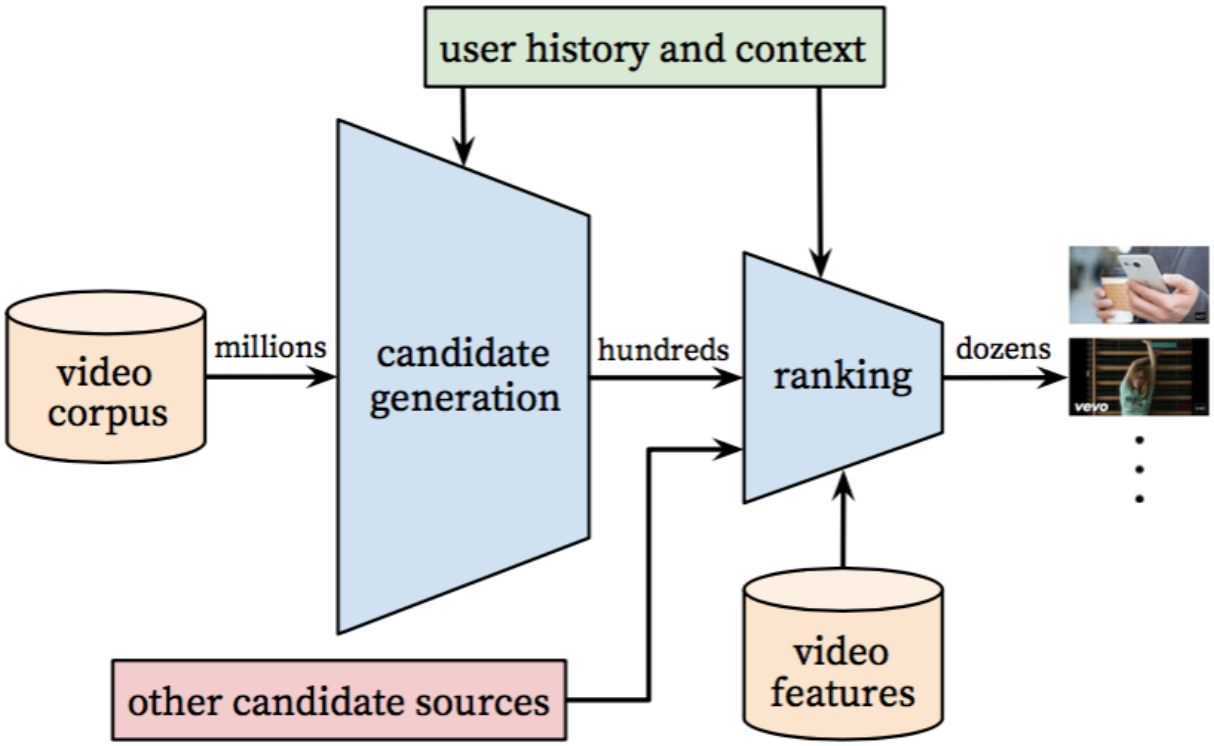


Figure 2: Recommendation system architecture demonstrating the "funnel" where candidate videos are retrieved and ranked before presenting only a few to the user.

候选集生成网络将用户的YouTube活动历史记录中的事件作为输入，并从大型语料库中检索一小部分（数百个）视频。 这些候选集通常旨在以高精度与用户相关。 候选集生成网络通过协作过滤只能提供广泛的个性化。 用户之间的相似性以诸如视频观看的ID、搜索查询词、人口统计的粗略特征来表示。

在列表中呈现几个“最佳”推荐结果需要精细级别表示来区分具有高召回率的候选集之间的相对重要性。 排序网络通过使用描述视频和用户的丰富的特征集合，根据期望的目标函数向每个视频分配分数来完成这个任务。 最高得分的视频呈现给用户（按它们的得分排名）。

两阶段的推荐方法允许我们从非常大的语料库（数百万）视频中做推荐，同时还要保证设备上出现的少量视频是个性化的，可以为用户服务的。 此外，该设计使得能够混合由其它数据源（例如在早期工作[3]中描述的那些数据源）产生的候选集。

在开发过程中，我们广泛使用离线数据（精度，召回率，排序损失等）来指导我们的系统的迭代改进。 然而，对于算法或模型的有效性的最终确定，我们依赖于通过线上实验的A / B测试。 在线上实验中，我们可以衡量点击率，观看时间和衡量用户投入的许多其他指标的细微变化。 这是很重要的，因为线上A / B测试结果并不总是与离线实验正相关。

3. 候选集生成

embedding这个术语着实不太好翻译，有的人翻译成“嵌入”，我觉得是不合适的、完全不搭边的，翻译成“映射”可能还好一些，中文的精髓在于见形知义嘛。embedding其实就是把一个ID类特征 或者 离散特征 映射成一个n维向量，而这个向量初始化后，在训练过程中和模型的权重一样是作为一个变量来训练的。

在候选集生成阶段，大量的YouTube语料被砍掉，到最后可能只剩下与用户相关的数百个视频。 这里描述的推荐器的前身是在最小化秩损失的目标下训练的矩阵分解方法[23]。 我们的神经网络模型的早期迭代中通过浅层网络模拟了这种用户行为矩阵分解，但是只对用户的之前的观看记录做了embedding。 从这个角度来看，我们的方法可以被看作是因式分解技术的非线性泛化。

3.1 用分类做推荐

【下面这段话意思是：对于用户u，在看了一些视频之后（这就是“上下文”），最有可能看哪个视频？这原本是个推荐问题，本文把这个推荐问题转化成分类问题，假设网站上有100万的视频，即有100万种类别，这个推荐问题就变成了：用户u看过一些视频后，我们需要预测用户下一个要看的视频是100万分类中的哪一个分类？这是个典型的多分类问题，但是类别的数目非常大。】

我们提出用极端多类分类做推荐，预测问题就变成了：对于用户U和上下文C，把语料库V中的数百万个视频（分类）i，在时间t处做准确的分类， u代表了user & context对儿的高维

embedding，vi 代表各个候选视频的embedding。

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

在这个设定中，embedding仅仅是稀疏实体（单个视频，用户等）到实数密集向量的映射。 深层神经网络的任务是根据用户的历史和上下文来学习用户的embedding，这对于用softmax分类器来区分视频是有用的。

虽然YouTube上存在明确的反馈机制（大拇指向上/向下，产品内置调查等），但是我们这里使用观看记录的隐式反馈[16]来训练模型，其中看完了视频的用户是一个正例。 这个选择是基于在数量级上可用的隐式用户历史更多，允许我们在尾部进行深度推荐，而显式反馈是非常稀疏的。

有效的极端多分类

重要！！！

这里作者对负样本的抽样一笔带过，可能是作者觉得这里很容易，其实这里是非常重要的一个细节，涉及到noise contrasive estimate 和negative sampling，对于训练速度提升至关重要，这里不重视的话可能连模型都训练不出来。具体可以搜索Word2Vec的相关原理祥解。

为了有效地训练这种具有数百万类的模型，我们依靠一种技术从背景分布（“候选抽样”）中抽样负类，然后通过重要性权重校正这个抽样[10]。 对于每个例子，对于真实标签和样本负类，交叉熵损失被最小化。 在实践中，对几千个负样本进行采样，速度相当于传统softmax的100倍以上。一个流行的替代方法是分层softmax [15]，但我们不能实现与之可比的准确性。 在分层softmax中，遍历树中的每个节点包括区分通常不相关的类集合，使得分类问题更加困难并且性能更低下。

在提供推荐服务时，我们需要计算最可能的N个类（视频），以便选择前N个呈现给用户。 在数十毫秒的严格服务延迟下对数百万个视频进行评分，需要使用和分类数量呈亚线性的近似评分方案。以前的YouTube推荐系统依赖hashing[24]，这里描述的分类器使用类似的方法。 由于在服务时间不需要对来自softmax输出层进行似然校验，所以评分问题减少到可以使用通用库的点积空间中的最近邻搜索[12]。 我们发现A / B结果对最近邻搜索算法的选择不是特别敏感。

3.2 模型架构

受连续词袋语言模型（CBOW）的启发[14]，我们以固定词汇表学习每个视频的高维映射（embedding），并将这些高维映射（embedding）喂给前馈神经网络。用户的观看历史由稀疏的视频ID构成的长度可变的序列表示，其通过embedding被映射成稠密向量表示。神经网络的输入需要是固定大小的稠密矩阵，简单的将用户的所有历史观看视频ID的embedding做平均，其效果在几种策略（相加、分量最大等等）中效果最好。

重要的是，embedding通过正常的梯度下降反向传播更新与所有其他模型参数一同学习。 特征被连接成一个宽的第一层，随后是几层完全连接的修正线性单元（ReLU）[6]。 图3显示了包含其他非视频特征的一般网络架构。

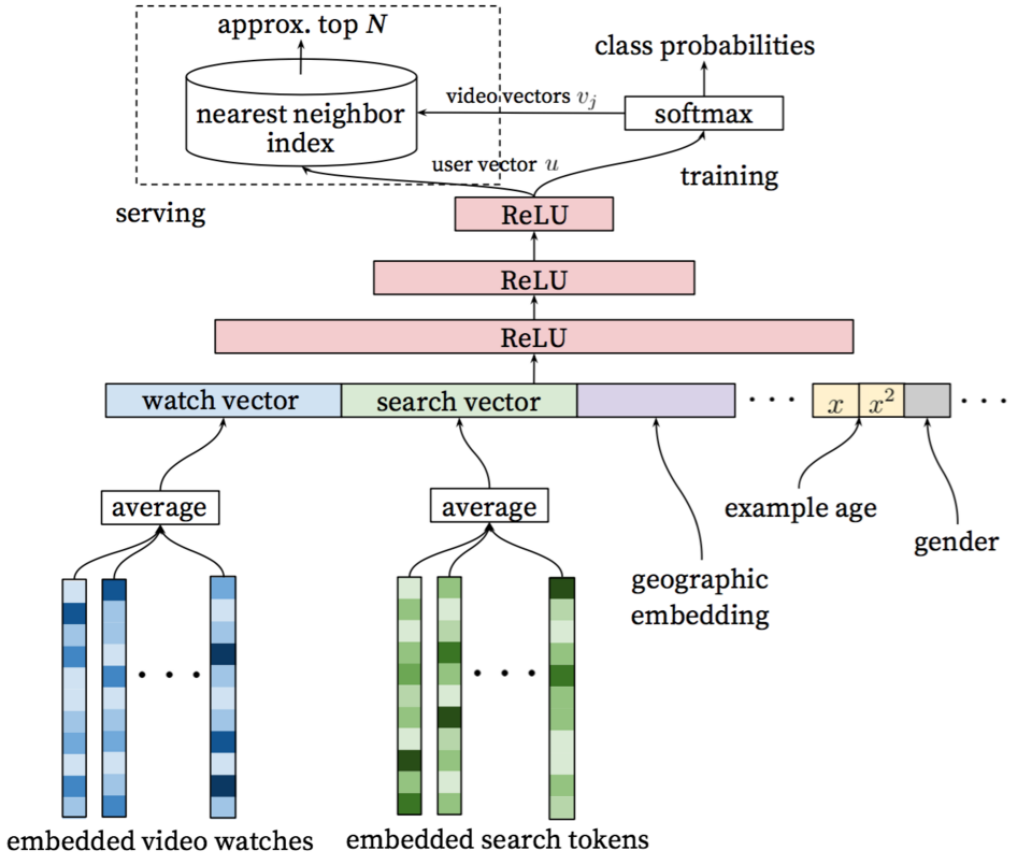


Figure 3: Deep candidate generation model architecture showing embedded sparse features concatenated with dense features. Embeddings are averaged before concatenation to transform variable sized bags of sparse IDs into fixed-width vectors suitable for input to the hidden layers. All hidden layers are 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.

3.3 异构信号

使用深层神经网络作为矩阵分解的一般化的一个关键优点是任意连续和分类特征可以容易地添加到模型中。搜索历史与观看历史记录被同等对待——每个查询词都被切分为一元和二元的，并且每个字都被embedded。一旦平均之后，用户的切分并embedded的查询就代表了一个概

括的稠密搜索历史。 人口统计特征对于提供先验是重要的，使得推荐对于新用户来说是合理的。 用户的地理区域和设备被embedded和连接。简单的二值和连续特征，例如用户的性别，登录状态和年龄作为归一化为[0,1]的实数值直接输入到网络中。

“样本年龄”特征 （视频已上传的时长）

每秒有许多长达若干小时的视频被上传到YouTube。 推荐这些最近上传（“新鲜”）的内容对于作为产品的YouTube极为重要。 我们经常发现用户喜欢新鲜的内容，虽然没有损失相关性。除了要第一优先保障简单的推荐给用户想要观看的新视频的之外，还存在引导绑定和传播病毒内容的关键次要现象[11]。机器学习系统往往表现出对过去的隐含偏差，因为他们用历史样本训练模型来预测未来的行为。视频流行度的分布是非常不稳定的，但是由我们的推荐器产生的语料库的多项分布将反映在几周的训练窗口中的平均观看可能性。 为了纠正这一点，在训练的时候我们将训练样本的年龄作为一个特征。 在推荐系统提供服务的时候，这个特征设置为零（或绝对值比较小的负数），以反映模型正在训练窗口的末端进行预测。

图4演示了这种方法对任意选择的视频的有效性[26]。

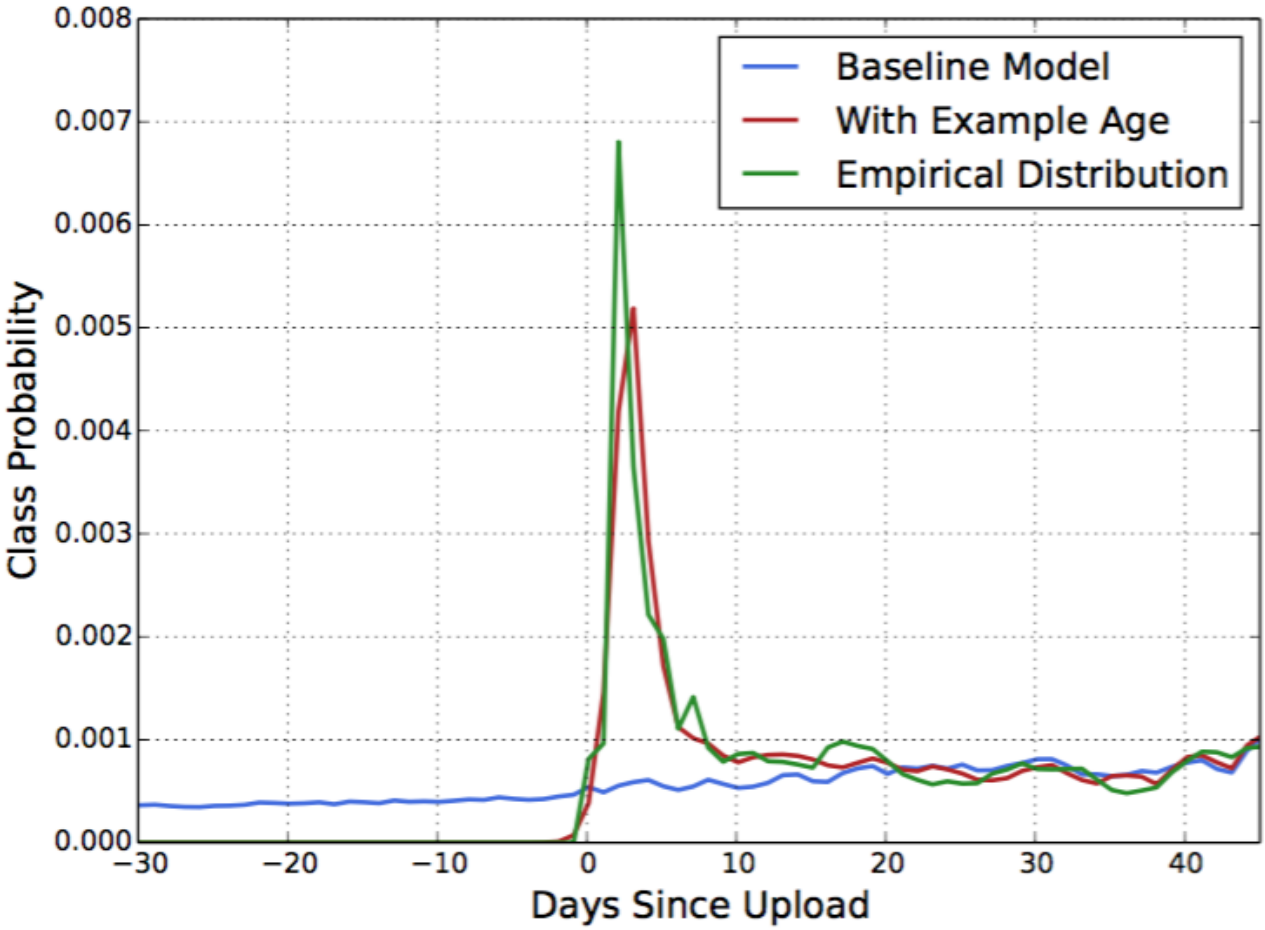


Figure 4: For a given video [26], the model trained with example age as a feature is able to accurately represent the upload time and time-dependant popularity observed in the data. Without the feature, the model would predict approximately the average likelihood over the training window.

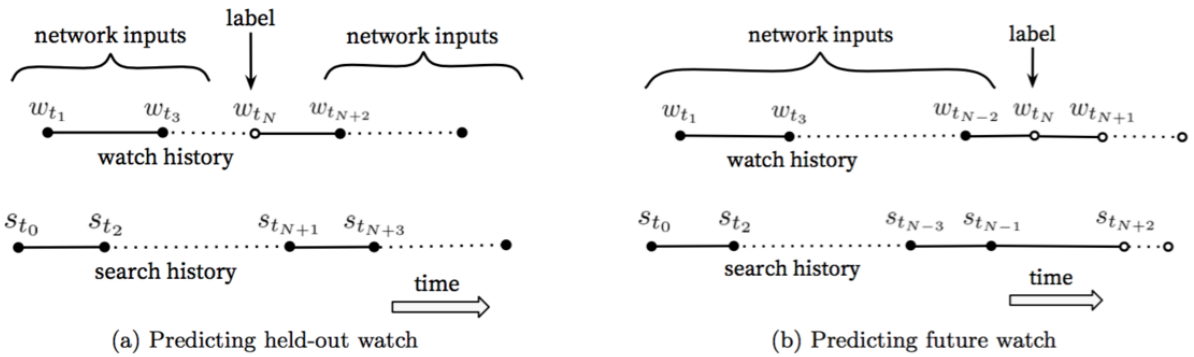
3.4 正负样本和上下文选择

强调一下，推荐通常涉及解决替代问题并将结果转移到特定上下文。 一个典型的例子：我们通常假设准确的预测分数才能达成高效的电影推荐[2]。 我们发现，这种替代学习问题的选择在A / B测试中的性能具有极大的重要性，但是很难用离线实验来衡量。

训练样本要用youtube上的所有视频观看记录，而不只是我们的推荐的视频的观看记录。否则，面对新视频的时候很难推荐，并且推荐器会过度偏向exploitation（这里的exploitation是针对exploration来讲的）。如果用户不是在我们的推荐场景发现新视频，我们希望能够快速的

通过协同过滤将这种发现传播给其他人。另外一个提升线上衡量指标的感悟是为每个用户生产固定数量的训练样本，在损失函数中所有用户的权重一样。这能防止一部分非常活跃的用户主导损失函数值。

有点反直觉的是，必须非常小心地保留来自分类器的信息，以防止模型利用网站的结构和过度拟合替代问题。 想象一下这种情况，一个用户刚刚搜索查询了一个词“taylor swift”。因为我们的问题是预测下一个观看的视频，一个分类器拿到这个搜索关键词信息的之后，会预测最有可能被观看的视频肯定是出现在“taylor swift”搜索结果页上的视频。想都不用想，用用户最后一次搜索的结果页作为首页的推荐结果效果非常差。通过丢弃搜索信息的顺序，用无序的词袋表示搜索查询，分类器就无法直接感知正负样本的来源了。视频的自然消费模式通常导致非常不对称的共同观看概率。 剧集系列通常是顺序地观看，并且用户经常是先发现最广泛流行的流派的艺术家的视频，然后才专注于较小众的。因此，我们发现预测用户的下一个观看视频的效果要好得多，而不是预测随机推出的视频（图5）。 许多协同过滤系统隐式地从历史观看记录中随机拿出来一个作为正样本来预测它，其余的历史观看记录作为上下文（5a）。这样其实泄露了未来的信息，并且忽略了任何非对称的消费模式。相反，我们是从用户的历史视频观看记录中随机拿出来一个作为正样本，然后只用这个视频之前的历史观看记录作为输入（5b）。



(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.

3.5 特征和深度实验

添加特征和深度显着提高了对保留数据的准确性，如图6所示。

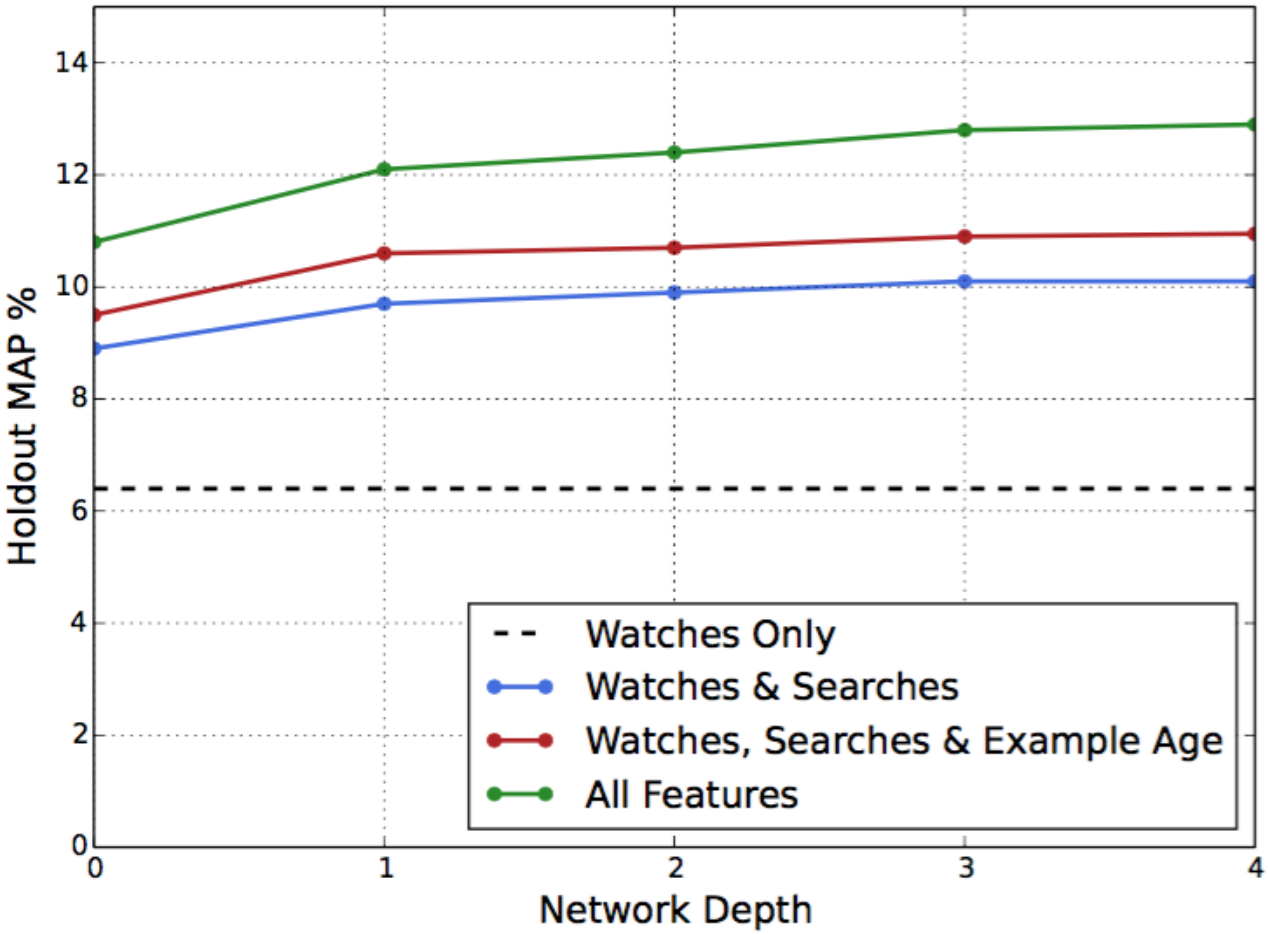


Figure 6: Features beyond video embeddings improve holdout Mean Average Precision (MAP) and layers of depth add expressiveness so that the model can effectively use these additional features by modeling their interaction.

在这些实验中，有1M个视频和1M个搜索词的词表，最多50个视频和50个最近的搜索被映射成256 维的float数组。softmax层输出了一个针对1M个视频分类的多项式分布，每个视频分类对应一个256维的向量。模型基于所有youtube用户数据进行训练，直至收敛。网络结构呈一个常见的塔状，底部的网络最宽，每往上一层节点数减半。深度为0的网络和之前的推荐系统非常相似，是一个高效的线性分解公式。直到收益消失或者很难收敛之前，不断的增加宽度和深度。

- 深度为0：这时网络就是一个把连接起来的输入层转换一下，和softmax的256维输出对应起来

- 深度为1：第一层 256个节点，激活函数 是ReLU （rectified linear units 修正线性单元）

- 深度为2：第一层512个节点，第二层256个节点，激活函数都是ReLU

– 深度为3：第一层1024个节点，第二层512个节点，第三层256个节点，激活函数都是ReLU

……

4.排序

排序的任务是用展现数据，校正给特定用户生产的候选集预测顺序。比如说，一个用户本来很可能会看一个给定的视频，但是因为首页上点赞图片的展现而不会去点这个视频。在排序阶段，我们使用了视频和用户及其关系的很多其他的特征，因为排序阶段只有几百个候选视频，而候选集生成阶段有上百万个。不同数据源产生的候选集的分数是不能直接相比的，在这一点上来说排序是非常重要的。

我们用一个和候选集生成相同的神经网络结构，最后一层用逻辑回归给每个视频展现打个分数。候选集列表按照这个分数排序之后返回给用户。我们的最终排名目标是根据线上A / B测试结果不断调整的，但通常是每次展示的预期观看时间的简单函数。按点击率排名容易造成一些用户看了但是没看完的视频排到前面，而观看时间就很好的反映了用户的偏好程度。

4.1 特征表示

我们的特征与分类和连续/序列特征的传统分类方法是不一样的。我们使用的离散特征在它们的基数中广泛变化——一些是二值的（例如，用户是否登录），而其他具有数百万可能的值（例如，用户的最后搜索查询）。根据它们是仅贡献单个值（“单值”）还是一组值（“多值”）来进一步划分特征。单值分类功能的示例是正在评分的展示的视频ID，而相应的多价特征可能是用户观看过的最后N个视频ID的视频集合。我们还根据它们描述项目的属性（“印象”）还是用户/上下文（“查询”）的属性来对特征进行分类。 每个请求计算一次查询特征，同时为每个已评分项目计算展现特征。

特征工程

我们通常在排序模型中使用数百个特征，离散和连续特征大致各占一半。尽管深度学习可以减轻人工特征工程的压力，但是由于我们的原始数据的特质，原始数据也没那么容易就能输入到前馈神经网络中。我们依然需要花不少特征工程资源把用户和视频转化成有用的特征。主要的挑战在于如何表示用户动作的时间序列以及这些动作如何与正被评分的视频展现相关。我们观察到，最重要的信号是那些描述用户以前与item本身和其他相似item的互动，这一点和其他人在广告排序中的经验是一致的[7]。比如在用户过去有行为的频道中，用户从这个频道里看过多少视频？用户上次观看这个主题的视频是什么时候？这些连续特征在刻画相关item上的用户行为上是非常强的特征，因为他们在不同的item之间泛化能力很强。我们还发现，把这种信息以

特征的方式从候选集生成扩展到排序阶段是非常关键的，比如那些数据源生成了这个候选集？每个数据源给这个候选集打了多少分？

描述视频展现次数的特征对于引入推荐中的“流失”也是非常重要的。如果最近给一个用户推荐了一个视频，但是用户并没有看，那么当下次页面加载的时候，模型会自然的对这个视频展示的概率降低。提供秒级的视频展现和用户观看历史不在本文的探讨范围内，但是对进行实时交互的推荐至关重要。

embedding 离散特征

和候选集生成一样生成，我们使用embedding来将稀疏离散特征映射到适合于神经网络的稠密矩阵形式。每个唯一的ID（词汇）都对应一个单独学习出来的embedding，它的维度大小和整个词汇表的ID个数的对数呈正比增加。这些词汇是通过在训练之前扫描一次数据建立的简单查找表。**对词汇表按照ID在点击展现中出现的频率进行倒序排列，仅保留频率最高的topN个ID，其他的ID就被从词汇表中去掉了**。不在词汇表中的值，简单的被embedding成值为0的向量。像在候选集生成阶段一样，多值离散特征映射成embedding之后，在输入网络之前需要做一下加和平均。

重要的是，离散特征对应的ID一样的时候，他们的底层embedding也是共享的。比如存在一个全局的embedding对应很多不同的特征（曝光的视频ID，用户最后一次浏览的视频ID，推荐系统的种子视频ID等等）。embedding虽然是共享的，但是每个特征在训练的时候是单独输入到网络的，以便上层网络可以学习到每个特征的特殊表示。embedding共享对于提升泛化能力、加速训练、减小内存占用是非常重要的。绝大多数参数模型是在这种高维embedding中的，例如，100万个ID映射成32维的空间，其参数是完全连接的2048个单元宽网络参数的7倍多。

连续特征归一化

神经网络对其输入的缩放和分布非常敏感[9]，而诸如融合了决策树的模型对于各个特征的缩放是不会受什么影响的。我们发现连续特征的正确归一化对于收敛是至关重要的。一个符合f分布的特征x，等价转化成x~,用微积分使其均匀的分布在[0,1）区间上。在训练之前，扫描一次数据，用线性插值的方法，基于特征值的分位数近似的计算出该积分。

除了输入归一化的特征之外，我们还输入归一化特征x~的平方、x~的平方根，特征的超线性和子线性的函数使得网络有更强的表达能力。输入连续特征的幂值，被证明是能提高离线精度的。

4.2 观看时长建模

我们的目标是给定正负样本（展现了没有点是负样本，展现了并点击了是正样本）后，预测用户的观看时长。正样本记录了用户观看视频消耗的时间。为了预测观看时长，我们使用了专门为此而开发的加权逻辑回归技术。

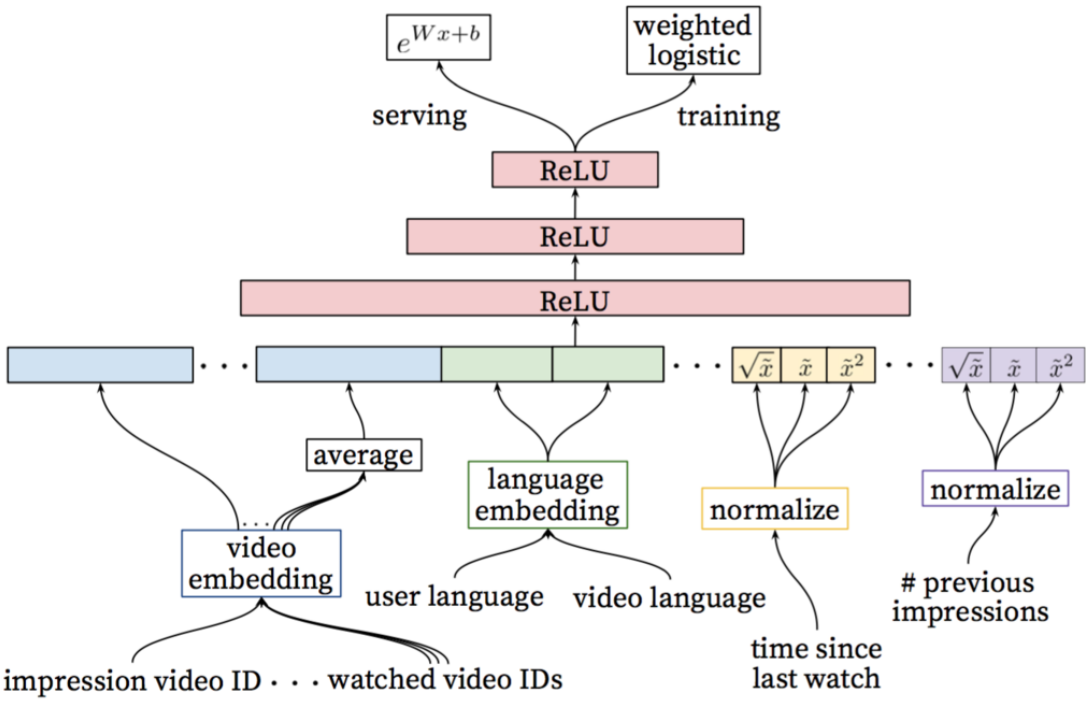


Figure 7: Deep ranking network architecture depicting embedded categorical features (both univalent and multivalent) with shared embeddings and powers of normalized continuous features. All layers are fully connected. In practice, hundreds of features are fed into the network.

这个模型是基于交叉熵损失函数的逻辑回归模型训练的。但是我们用观看时长对正样本做了加权，负样本都用单位权重（即不加权）。这样，我们通过逻辑回归学到的优势就是(ΣTi)/(N–k),其中N是样本数量，k是正样本数量，Ti是观看时长。假设正样本集很小，那么我们学到的优势就近似E[T](1+P),P是点击概率，E[T]是观看时间的期望值。因为P很小，那么这个乘积就约等于E[T]。我们用指数函数e^x作为最终的激活函数来产生近似观看时长的估计值。

4.3 隐层实验

Hidden layers	weighted, per-user loss
None	41.6%
256 ReLU	36.9%
512 ReLU	36.7%
1024 ReLU	35.8%
512 ReLU → 256 ReLU	35.2%
1024 ReLU → 512 ReLU	34.7%
1024 ReLU → 512 ReLU → 256 ReLU	34.6%

Table 1: Effects of wider and deeper hidden ReLU layers on watch time-weighted pairwise loss computed on next-day holdout data.

表1显示了我们在保留数据集上用不同的隐层配置得到的结果。每个配置得到的值是通过在同一个页面上展示给一个用户的正负样本而来的。我们首先用我们的模型对两种样本进行打分。如果负样本的得分比正样本的得分高，就认为这个正样本的观看时长是错误预测。每个用户的损失值就是所有错误预测的数量。

这些结果表明增加隐层的宽度提升了效果，增加深度也是一样。然而考虑到服务器CPU所需的计算时间，需要做个折衷。配置一个1024宽的ReLU，紧跟着一个512宽的ReLU，然后配置一个256宽的ReLU，使我们拿到最好的结果的同时，也使我们能够保持在我们的服务器CPU预算之内。

对于1024–>512–>256的模型我们尝试了只输入归一化的连续特征，不输入它们的幂值（使损失值只增加了0.2%）。使用同样的隐层配置，我们也训练了一个正负样本同时加权的模型。不出意料，这使得观看时长的损失值增加了4.1%之多。

5 总结

我们已经描述了我们的深度神经网络架构，用于推荐YouTube视频，分为两个不同的问题：候选集生成和排序。

我们的深层协作过滤模型能够有效地吸收许多信号，并且模拟它们与层的深度的相互作用，超越了以前在YouTube上使用的矩阵分解方法[23]。选择推荐的替代问题，更多的是门艺术而不是科学，我们发现，捕捉不对称的共同观看行为和防止未来信息的泄露，使得对未来的观看进

行的分类在线上指标表现良好。从分类器中保留不同的信号，对取得好的结果也是很重要的，否则模型会对替代问题过拟合，但是不能将效果转移到首页的推荐上去。我们证明使用训练样本的年龄作为输入特征消除了对过去的固有偏差，并允许该模型表示流行的视频的时间依赖行为。 这提高了离线测试集的精度，并且在最近上传的视频上的A / B测试中大大增加了观看时间。

排序是一个更经典的机器学习问题，但我们的深度学习方法胜过以前的基于线性和树的观看时间预测方法。 推荐系统特别受益于描述过去用户对item的行为的特定特征。深层神经网络需要分类和连续特征的特殊表示，我们分别用embedding和分位数归一化变换实现了这种表示。通过网络层的深度将几百个特征的非线性相互作用有效的进行了建模。对正样本用观看时长进行加权，对负样本不加权，这样修改之后的逻辑回归使得我们能够学习出接近观看时间的模型。与直接预测点击率相比，这种方法在观看时间加权排序评估指标上表现得更好。