深度学习中的序列模型研究

——基于 2020 腾讯广告算法大赛数据

麦嘉仪　　宋宁宇　　李卓熹

一、背景介绍

信息时代下的数据量快速膨胀，信息过载现象越来越严重。为了缓解信息过载带来的影响，有很多值得称赞的解决方案，其中最有效的处理方法之一就是推荐系统。推荐系统利用用户与项目的交互信息进行推荐。前馈网络和 CNN 在传统的推荐系统中被广泛应用。

众所周知，像用户年龄和性别这样的人口统计学特征是各类推荐系统的重要输入特征，其中自然也包括了广告平台。这背后的假设是，用户对广告的偏好会随着其年龄和性别的不同而有所区别。许多行业的实践者已经多次验证了这一假设。然而，大多数验证所采用的方式都是以人口统计学属性作为输入来产生推荐结果，然后离线或者在线地对比是否使用这些属性特征作为输入变量的推荐效果。2020 腾讯广告大赛的题目尝试从另一个方向来验证这个假设，即以用户在广告系统中的交互行为作为输入来预测用户的人口统计学属性。我们认为这一赛题的“逆向思考”本身具有其研究价值和趣味性，此外也有实用价值和挑战性。例如，对于缺乏用户信息的实践者来说，基于其自有系统的数据来推断用户属性，可以帮助其在更广的人群上实现智能定向或者受众保护。

本文采用的是序列模型（sequence model），它指的是处理语言或者音视频等前后相互关联数据的模型。难发现用户在广告系统中的素材点击序列具有强关联关系，例如用户点击了素材 A, 下一时间点他可能会点击包含素材 A 中介绍到的同款商品的素材 B。前馈网络和 CNN 不能处理这样的数据集吗？为什么我们还需要序列模型呢？这是因为我们希望模型能够考虑到序列中前后元素之间的这些关联。一个典型的例子是，当我们考虑一个英文句子，也就是多个单词构成的序列时，传统的前馈网络和 CNN 在训练和预测时往往是把每个单词视作独立的输入，而不会把单词和单词之间的信息利用进来；相对地，序列模型会在考虑某个单词输入的时候，试图把同一句子中出现在前面或后面的词也给利用上，从而改善建模效果。

互联网时代的到来为传统问题提供了新的解决思路，机器学习算法有不过分依赖模型的优点，可以将更多的因素纳入考虑范畴，同时在大数据的加持下，由模型根据影响因子使用

海量数据进行自主学习，使模型具有更强的范化能力。机器学习的应用方向主要是分类和预测。在分类方面，决策树算法、支持向量机、遗传算法得到广泛应用；在预测方面，20 世纪 80 年代诞生的循环神经网络开启了时间序列预测的新时代。1997 年，Hochreiter S 等人提出了一种新结构—LSTM 递归神经网络，该网络具有长时记忆的特点，在预测较长时间的序列数据中有着良好的应用，2014 年 Cho 等人在此基础上提出 GRU 递归神经网络，目前这两种神经网络已广泛应用于金融预测、交通流预测、文本预测、机器翻译等多个领域。

神经网络的复杂化衍生了深度学习的算法，与浅层神经网络相比，深度学习有多层能够进行非线性变换的隐层。近年来，深度学习在序列模型中的应用和发展越来越快，在多个比赛中呈现出无可比拟的优势，其中应用比较热门的是 LSTMs、RNNs、GRU、Attention 机制等，使用率在私人排行榜前 10 名。

二、模型介绍

在这一部分，我们介绍 RNN 模型、LSTM 模型及其一些拓展模型的基本原理。

2.1 循环神经网络

循环神经网络(Recurrent Neural Networks, RNN)的设想源自 1982 年 Saratha Sathasivam 提出的霍普菲尔德神经网络(Hopefield Networks)。RNN 是具有循环结构的深度神经网络，可以对序列数据进行预测。在语句、段落等逻辑序列和股票走势价格、监控数据等时间序列中，前后数据并不是孤立的，后续词语和数据的预测依赖之前的信息，RNN 能够记忆之前的输出信息，并应用于当前输出的计算。RNN 的记忆特性归功于其特有的网络结构。

2.1.1 RNN 的结构

RNN 与传统神经网络的结构有相似之处，都由输入层、隐层和输出层组成。但两者的神经元连接方式不同，传统神经网络的连接形式采用层与层之间神经元全连接、层内神经元间无连接的形式，RNN 为了实现对之前信息的记忆，其隐层神经元间形成有序连接，这样，在隐层的每个神经元不仅接收输入层的信息，还会接收上一时刻的隐层传递的信息。具体结构如下图所示:

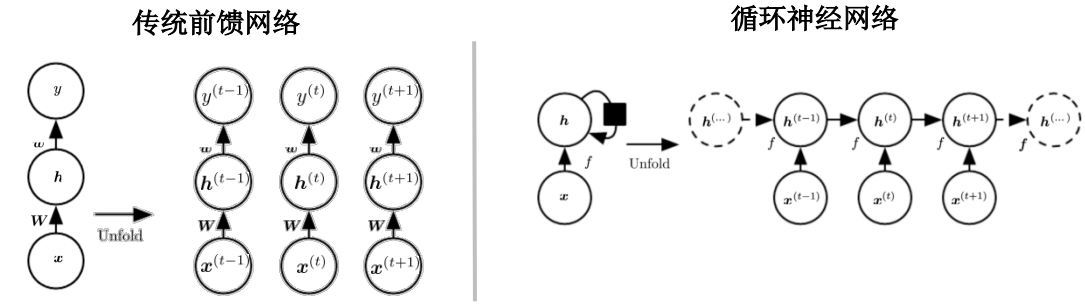

图 1：传统前馈网络（左）与循环神经网络（右）的比较

2.1.2 RNN 的优势与不足

相较 CNN 等其他神经网络模型而言，RNN 的结构链条在理论上可以无限循环，其优势在于能够处理时间序列数据，这对时间序列数据预测的发展具有跨时代的意义。

RNN 的不足之处也很明显，其参数学习方法带来梯度爆炸和梯度消失问题，因而能够处理的时间序列长度较短，无法记忆长时间的数据，也无法进行长时间的序列预测。

RNN的参数训练采用随时间反向传播(back-propagation through time, BPTT)的传导机制，在选用 Sigmoid 函数作为激活函数的链式求导中，由于 Sigmoid 函数的性质，当该函数的自变量很大或很小时，其导数趋于零，在链式法则的乘积传导中导致梯度极小，形成梯度消失，对权重的调整几乎无效。当初始权重很大时，通过多次乘积传导，形成梯度爆炸，导致结果不稳定。梯度爆炸和梯度消失的存在，表明 RNN 只能完成短期记忆，当输出值与很长时间之前的数据有关时，靠前的神经元对应的权重无法进行有效学习，无法得到较优的结果。梯度爆炸梯度消失的解决方案中，常用的有以下三种：一是优化初始化权重，使权重维持在合理区域，以减小梯度消失的可能性；二是设置梯度下降的阈值，在到达阈值时梯度截断，不再进一步执行下降操作；三是更换激活函数，使用自变量为正时导数不为零的 ReLU 函数或其相似变体代替 Sigmoid 函数。

上述方法的使用，在一定程度上缓解了梯度爆炸和梯度消失现象，随之而来的是数据记忆时间长度的限制，无法处理长时数据。直至长短期记忆递归神经网络的提出，通过调整神经元的内部结构，从结构上缓解了梯度消失问题，同时保留循环神经网络长时记忆的优势。

2.2 LSTM 与 GRU 递归神经网络

2.2.1 LSTM 递归神经网络

长短期记忆(Long Short-Term Memory,LSTM)递归神经网络是 Hochreater 和 Schmidhube 在 1997 年提出的，是一种特殊的循环神经网络。该算法能够克服 RNN 不能有效处理长时依赖的问题，并且在一定程度上缓解梯度消失问题。目前，LSTM 在智能翻译、文本生成摘要、文本分类、机器对话等具有长时依赖关系的多个领域都有着不俗的应用。

LSTM 的神经元间拓扑结构和神经元内运算结构与 RNN 均有所区别。从外部结构来看，LSTM 的每个单元接收三个变量，分别是当前状态的原始输入值和上一单元的两个输出量，随后在单元内部进行复杂的运算，输出两个值，这两个值也会作为下一个神经单元的输入值，其中一个还会作为当前状态的输出值。LSTM 递归神经网络通过门控制将短期记忆和长期记忆结合起来。

LSTM 是在 RNN 循环的基础上，优化神经元细胞部分，将每个神经元内部的运算精细化，以达到长时记忆的效果。

（1） LSTM 的前向传播结构

在神经元细胞中，每一时刻接收的数据包含上一节点的输入数据C_{t-1}和h_{t-1}以及当前时刻的输入数据x_t三项。每个神经元内部对输入数据的运算有三个过程，分别依赖遗忘门、输入门和输出门。

遗忘门(forget gate)用来控制前期信息的遗忘程度。将h_{t-1}和x_t组成拼接向量，使之与权重矩阵相乘，随后通过 Sigmoid 函数激活，得到遗忘门控。遗忘门控的值为 0 到 1 之间的向量，数值越接近 0，表示对过往的信息遗忘越多。在这个阶段会按照重要程度进行选择，将对后续使用意义不大的数据遗忘。遗忘门工作原理图如下:

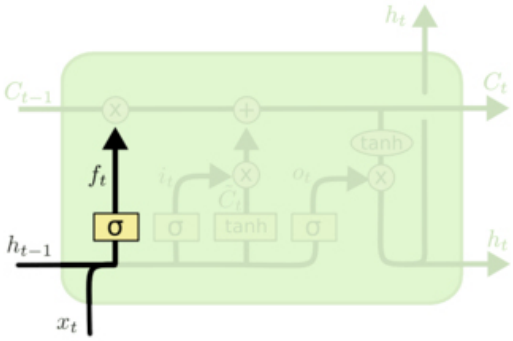

图 2：遗忘门工作原理

遗忘门的控制方式的数学表述如下:

$$f_t = \sigma(W_f \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_f)$$

若h_{t-1}和x_t的维度分别为ξ和η，则权重W_f为ξ行，$\xi+\eta$列的矩阵，偏置项为ξ维向量，激活函数部分无需要训练的参数，因此在 LSTM 层数为λ时，遗忘门部分待训练参数的数量为:

$$num = \lambda(\xi(\xi+\eta)+\xi)$$

输入门(input gate)用来控制前期信息的记忆选择，输入门由两个神经网络层构成。将h_{t-1}和x_t组成拼接向量，使之与权重矩阵相乘，随后通过 Sigmoid 函数激活，得到输入门控，进行选择性记忆。输入门控的值为 0 到 1 之间的向量，数值越接近 1，表示对过往的信息记

忆越多。在这个阶段会按照重要程度进行选择，着重记忆对后续较为重要的数据。第二部分是将h_{t-1}和x_t组成拼接向量，使之与权重矩阵相乘，随后通过 tanh 函数变换，得到$\widetilde{C}_t$值。输入门工作原理图如下:

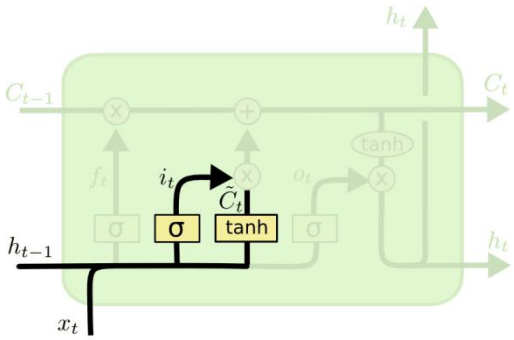

图 3：输入门工作原理

输入门的控制方式的数学表述如下:

$$i_t = \sigma(W_i \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_i)$$

tanh 函数激活层数学表达如下:

$$\widetilde{C}_t = \tanh(W_C \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_C)$$

其中，

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

输入门和遗忘门的权重维度相同，且激活函数和 tanh 函数部分无需要训练的参数，因此在 LSTM 层数为λ时，输入门部分待训练参数的数量为:

$$num = 2\lambda(\xi(\xi + \eta) + \xi)$$

输出门(output gate)控制当前神经元的输出和流入下一个神经元的数据。将h_{t-1}和x_t组成拼接向量，使之与权重矩阵相乘，随后通过 Sigmoid 函数激活，得到输出门控，进行选择性输出。输出门控的值为 0 到 1 之间的向量，数值越接近 1，表示对此类信息输出越多。通过输出门的控制，决定哪些将会被当成当前状态的输出。输出门工作原理图如下:

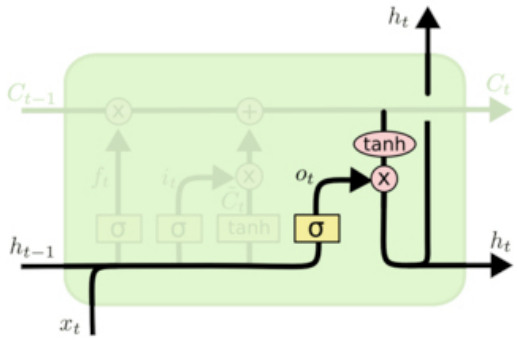

图 4：输出门工作原理

输出门的控制方式的数学表述如下:

$$o_t = \sigma(W_o \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b_o)$$

输出门和遗忘门的权重维度相同，且激活函数部分无需要训练的参数，因此在 LSTM 层数为λ时，输出门部分待训练参数的数量为:

$$num = \lambda(\xi(\xi + \eta) + \xi)$$

（2） LSTM 的优势与不足

LSTM 递归神经网络能够通过门控控制传输信息，遗忘不重要的信息，处理需要长期记忆的数据结果良好。

LSTM 递归神经网络有其不足之处，由于每个神经单元内含有 4 个全连接神经网络层，多次使用激活函数和矩阵运算，导致参数变多，训练难度增大。为缓解算力问题，提出了多种 LSTM 的变体，其中 GRU 递归神经网络的使用最为广泛。

2.2.2 GRU 递归神经网络

门控循环单元(Gated Recurrent Unit, GRU)递归神经网络是 cho 等人在 2014 年提出的 LSTM 的变体形式。GRU 递归神经网络比 LSTM 少一个门控设置，因此参数比 LSTM 少，相对易于训练，且能够达到与 LSTM 相似的效果。

（1）GRU 的结构

GRU 的结构兼具 RNN 和 LSTM 的特色。从外部结构来看，GRU 的输入输出结构与普通循环神经网络相同，每个单元接收两个变量，分别是当前状态的原始输入值和上一单元的

输出值，经过单元内部复杂运算后，输出信息向量，该向量也会作为输入值输入下一神经单元。GRU 是在 LSTM 循环的基础上，优化门控装置，将神经元内部的运算简化，以达到有效训练的效果。LSTM 和 GRU 的结构对比图如下:

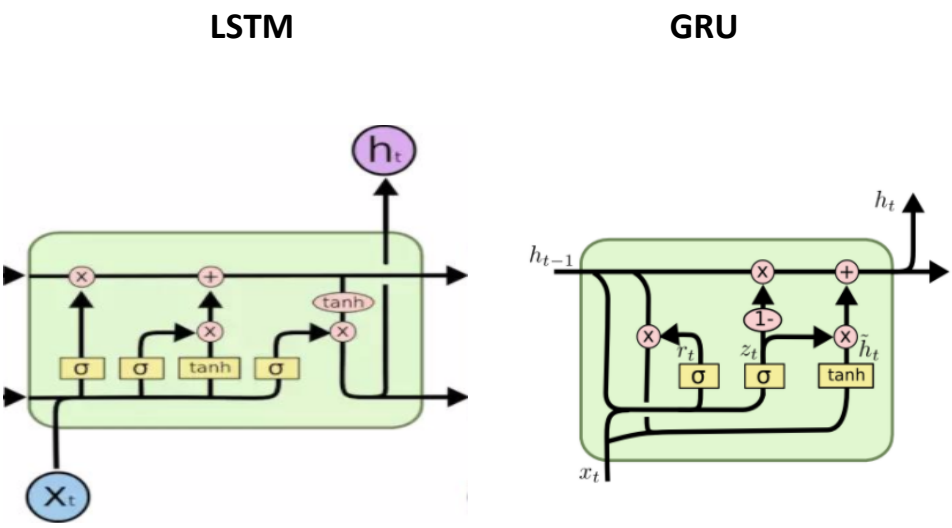

图 5：LSTM 与 GRU 示意图

在 LSTM 中，遗忘门、输入门、输出门控制记忆、输入和输出。GRU 模型将其简化为两个门控结构:更新门和重置门。更新门用于控制前一时刻状态的信息被带入到当前状态的程度，更新门的值越大，前一时刻的状态信息存留越多。重置门控制前一状态有多少信息被写入到当前的候选集 h 上，重置门的值越小，前一状态的信息被写入的越少。

（2） GRU 的优势与不足

GRU 模型是 LSTM 模型的一种变体，相较 LSTM 模型，GRU 模型的结构更加简单，同时效果也很好，因此称为当前流行的模型。由于 GRU 是 LSTM 的变体，因此同样能够解决 RNN 网络中的长依赖问题。与 LSTM 相比，在海量数据集情形下，GRU 更容易进行训练，提高效率。

2.3 Transformer 的编码模块

（1）Transformer 的编码结构

由于本文的算法采用了 Transformer 的编码模块，且编码模块也是预训练模型的核心构建，理解编码模块的结构和机理至关重要。Transformer 编码部分的基本架构如下图所示:

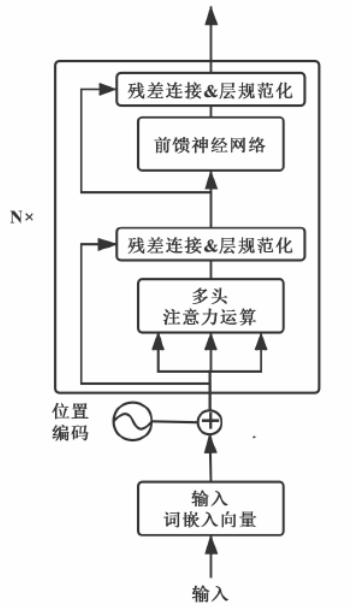

图 6：Transformer 编码部分基本架构

编码模块的数量为模型超参，而每一个特定编码模块的作用都是发现输入文本序列中各个词之间的某种关系，越靠近底层的编码模块学习到的内容更为一般越靠近上层的编码模块学习到的内容更为具体，更偏向特定输入序列内部的细节信息。编码模块中最重要的模块是多头注意力子模块，其基本架构如下图所示:

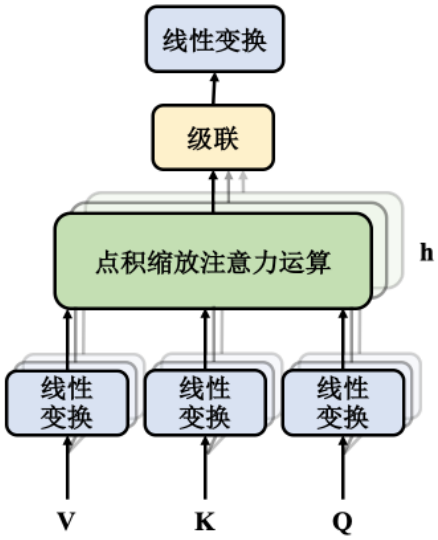

图 7：Transformer 多头自注意力运算子模块

它使用了多头注意力机制，意思是它会用不同的权重矩阵进行 h 次注意力运算，这 h 次注意力运算是并行的，每一个注意力运算的结果都被称为一个头部(head)，这 h 个结果，即 h 个头部最后会级连在一起得到一个(input_lenth) x (h * d_v)规格的级连矩阵。这个级连矩阵会通过一个拥有 emb_dim 个神经元的线性变换层，输出一个(input lenth) x (emb_ dim)规格的矩阵。抽象成数学公式可以表示如下:

$$MultiHead(Q,K,V) = Concat(head_1, head_2, ..., head_h)W^o$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \quad (i = 1,2,...,h)$$

其中 Concat 表示级连运算，一般情况下会令 Q=K=V，统一为各个编码模块的输入矩阵，例如对于第一个编码模块而言，Q, K 和 V 均为经过位置编码的输入矩阵 X，如果存在第二个或更多的编码模块，则 Q, K 和 V 为上一个编码模块的输出矩阵。

残差连接和层规范化子模块的输出矩阵会再经过两层的前馈神经网络，中间的隐藏层会用 ReLu 函数做激活处理，抽象成数学公式为:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

其中 x 为残差连接和层规范化子模块的输出矩阵。W_1和W_2分别为(emb_ dim)x(d_F)和(d_F) x (emb_ dim)规格的权重向量，d_F为隐层的神经元个数。可以看出经过前馈神经网络子模块的处理，输出数据仍然为一个与原始输入 X 同样规格，即(input length) x (emb_ dim)的矩阵，这个矩阵会进一步作为输入交给编码模块的第二个残差连接和层规范化子模块作处理。

（2）Transformer 的编码模块小结

在自注意力机制中，每个 token 都和周围上下文的所有 token 作 attention,因此任意两个位置的 token 都相当于有直连线路，因此，无论两个 token 之间相距多远，它们之间都有直连线路，从而可以捕获长距离的依赖信息，这样就解决了传统的 RNN 或者 Seq2Seq 神经网络的长期记忆问题。另外，自注意力机制的可解释性更好，根据注意力分数(Attention Score)能够知道一个 token 和上下文中的哪个 token 的关系相对更大。最后，自注意力机制便于实现并行化运算，这同样克服了传统循环神经网络只能串行运算的缺点。

三、比较实验

3.1 数据集介绍

数据集采用 2020 腾讯广告算法大赛初赛初复赛数据，以用户在广告系统中的交互行为作为输入来预测用户的人口统计学属性。

具体而言，数据集是一组用户(300 万)在长度为 91 天(3 个月)的时间窗口内的广告点击历史记录作为训练数据集。每条记录中包含了日期(从 1 到 91)、用户信息 (年龄，性别)，被点击的广告的信息(素材 id、广告 id、产品 id、产品类目 id、广告主 id、广告主行业 id 等)，以及该用户当天点击该广告的次数。原始数据如图 8 所示。测试数据集将会是另一组用户(100 万)的广告点击历史记录。提供给参赛者的测试数据集中不会包含这些用户的年龄和性别信息。

time	user_id	creative_id	click_times	user_id	age	gender
9	30920	567330	1	1	4	1
65	30920	3072255	1	2	10	1
56	30920	2361327	1	3	7	2
6	309204	325532	1	4	5	1
59	309204	2746730	1	5	4	1
12	309204	726402	1	6	6	1
79	309204	2851451	1	7	6	2
32	309204	1569716	1	8	5	1
5	309204	71956	1	9	5	1
8	309204	322354	1	10	9	2

图 8：原始数据

本文只运用其最细粒度的素材 id 进行实验，因为没有比赛中测试集的真实标签，将用户的训练集重新划分为两部分来进行序列模型的对比实验。评价指标为年龄的准确率和性别的准确率之和。

3.2 实验设计

3.2.1 预处理

最细粒度的素材 id 一共约有 400 万种，同时用户行为序列稀疏和广告投放稀疏，如图 9，在分布较为稀疏时，往往基于低频特征无法很好的学习到 id 的具体信息；同时，部分测

试集的广告在训练集没有出现，部分训练集的广告在测试集中也没有出现，所以需要稠密化转化。

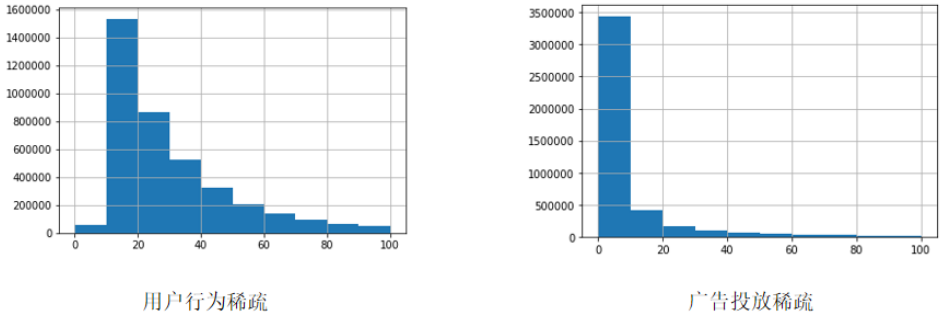

图 9 :用户与广告分布情况

稠密转化的方式我们使用 Word2Vec 进行预训练，被点击的素材可以理解成句子中的词语，用户点击的广告序列作为句子，由于对于用户的点击每条的广告之间存在一定的关联，因此用中间词去预测周围词是有意义的，训练素材 id 序列的词向量表征。具体 Word2Vec 的训练参数为我们选取维度为 512，窗口选取 16，采取 skip-gram 模型，负采样数为 5，轮数选择 8 轮，从而得到的 400 万广告的稠密转化表征。

由于用户点击序列长度不一，点击序列只截取前 128 个，不足补 0（训练时会被 mask 掉）。

对于分类标签处理，由于分别预测建模效率不高，我们将其考虑成 20 分类问题（年龄 10 分类 × 性别 2 分类），并采取概率聚合预测。

$$P(\texttt{gender}=1)=\sum_{i=1}^{10}P(\texttt{gender}=1,\texttt{age}=i)$$

$$P(\texttt{age}=1)=\sum_{j=1}^{2}P(\texttt{gender}=j,\texttt{age}=1)$$

3.2.2 模型

实验中我们的网络模型如图 10，首先通过 word2vec 预训练进行稠密转换，从而可以将素材 id 转为 512 维的词向量表示，将用户的长度为 128 的词向量序列作为模型输入，利用 LSTM 或 Transformer 等，建立 20 分类模型。

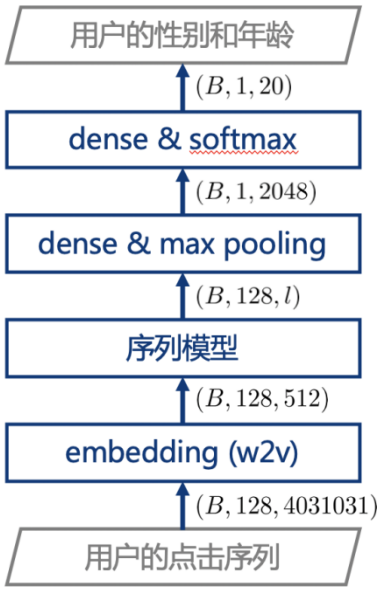

图 10 ： 网络模型

在实验中，我们分别使用以下不添加序列模型和添加非序列模型的网络作为对比，最后一列为预设学习率 (学习率调小，表现较好，可能因为 batch size 较小不稳定)：

- 非序列模型基准：
  - No Sequence Model（null）0.001
  - Deep Neural Networks （dnn）0.001
- 序列模型：
  - GRU （gru）0.0002
  - BiLSTM （lstm1）0.0005
  - BiLSTM + BiLSTM （lstm2）0.0005
  - RNN （rnn）0.0002
  - Transformer (tr1) 0.0001
  - Transformer + Transformer （tr2） 0.0001
  - Transformer + BiLSTM （trlstm） 0.0001

我们在将 300 万用户随机划分为 200 万用户作为训练集和 100 万用户作为测试集，然后在测试集上进行预测，并计算预测结果的准确率求和的值。同时为了评估模型的计算效率，我们还统计了模型的训练时间和测试时间。我们重复 5 次以上过程，报告每个模型的准确率和(均值、方差)、训练时间、测试时间的平均值。

关于模型训练的具体细节。我们使用 PyTorch 1.4.0 搭建模型，训练时 batchsize 为 100

(比较小是由于显存有限)，总共训练了 4 个 epoch（4 个 epoch 就能让模型收敛），优化器为 Adam，学习率在第一个 epoch 的从 0 线性递增到预设学习率，在第二、三轮维持预设学习率，再在最后一个轮学习率线性递减到 0。

对于预设学习率，对于含有 transformer 结构的模型，预设学习率设置为比默认小 10 倍效果较好，即 0.0001，同时对于 rnn、lstm、gru 均在 0.0001、0.0002、0.0003、0.0005、0.001 中选取效果最优的学习率，dnn、null 采取默认学习率 0.001。

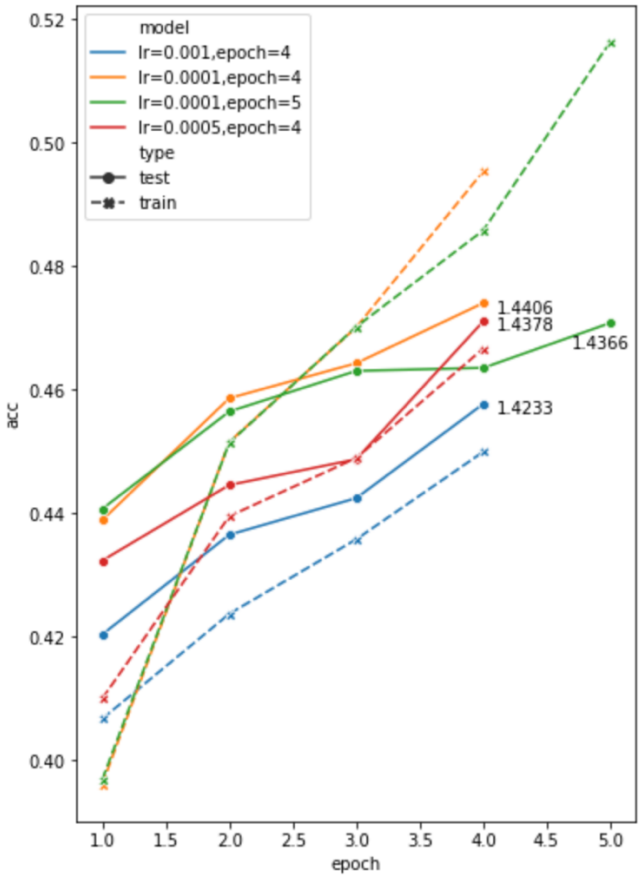

图 11 ： trlstm 模型结果

- 对于预设学习率，trlstm 模型，学习率为 0.001 是效果较差，随学习率降低，效果慢慢变好，可能是 batch_size 较小，以及含有 transformer 结构较难训练造成的，采用较小的预设学习率效果较好。
- 对于轮数 epoch，可以发现 trlstm 五个 epoch 已经过拟合训练集，甚至在训练集可以达到很高的准确率，可能的解释是：对于 trlstm 模型，由于其结构较为复杂，可以学到更多的知识，但这些知识也可能会造成对训练集的过拟合，该实验仅采用素材 id 序列，在实际比赛中会采用更多的序列信息，比赛时线下测试也是该模型效果较好，但在实验中对于测试集准确率来说，训练四个 epoch 可以达到最优。

3.2 实验结果

表 1 ： 模型对比实验结果

模型	训练时长 (s)	预测时长 (s)	测试准确率之和(std)	训练参数个数	预设学习率
null	2009	232	1.2893 (0.000132)	278548	0.001
dnn	5661	390	1.4343 (0.000499)	9489940	0.001
gru	9130	474	1.4371 (0.000433)	13690388	0.0002
lstm1	**9506**	**492**	**1.4415 (0.000355)**	**14741012**	**0.0005**
lstm2	12311	584	1.4382 (0.000452)	21040660	0.0005
rnn	8458	466	1.4338 (0.000343)	11589140	0.0002
tr1	9423	540	1.4388 (0.000330)	12642324	0.0001
tr2	13079	666	1.4400 (0.000580)	15794708	0.0001
trlstm	12944	615	1.4400 (0.000326)	17893396	0.0001

通过上表我们有如下结论：

- dnn 相较于 null，已经有了很大的提升，可能是由于其参数数量有提升较大，对于 null 模型，模型可以学到的知识有限，而加入 dnn 模块有了更多参数便可以学到更多的知识。
- 相较于非序列模型，序列模型均能够取得更好的测试准确率，但提升程度不同。这一定程度上说明了广告点击序列确实具有某种前后的依赖性，而序列模型能够把这些信息利用起来改善建模效果，通俗理解就是对于假设用户点击了过一条体育类型的广告，如果再次点击体育类的广告，序列模型会加强该用户点击体育类广告的效应，而如果用户如果点击其他各类广告，序列模型会减弱这种的效应。
- 对于考虑先后顺序的 BiLSTM (lstm1) 与不考虑先后顺序的 Transformer (tr2)，均超过了 1.44，可以得出广告点击序列为弱时间序列的结论。同时，比赛中采取了对序列进行打乱的序列增强操作，准确率依然波动不大，这可能说明广告点击序列内部各个广告有一定的依赖性，但并不是一个根据时间先后顺序的强依赖的关系。
- 表现最好的是单层 BiLSTM 模型，这表明 LSTM 模型作为最常用的序列模型之一，有着很强的实用性，值得在比赛或业务中作为首选的尝试方案。而双层 BiLSTM 模型效果不如单层，可能是仅采用素材 id 以及部分用户进行训练，数据较为简单，过于复杂的序列模型反而可能降低模型的泛化能力。
- 使用到 Transformer 的几个模型 (tr1, tr2, trlstm) 虽然在我们的实验结果中表现略输于单层 BiLSTM，但其实表现也都还不错，和单层 BiLSTM 的差距都在几个千分点左右。实际比赛中采用 trlstm 模型，其学习潜力比较大，比赛还结合了其他序列并且有更多的用户以及一些序列增强的方法，可能更复杂的模型学习效果更好。
- 在训练时长上，随参数量的增加，训练时长也是增加的。

四、总结

通过对比实验，我们发现序列模型确实能够对广告点击序列这种具有前后关联的数据实现更好的建模效果。

同时我们也发现，在序列模型的应用上有一些细节值得留意。比如在应用序列模型前有必要考虑序列前后依赖性的强弱，对于弱序列不能一味地追求复杂的序列模型，否则可能导致建模效果不佳；此外还有 Transformer 的表现可能比较依赖于调参的好坏，这需要我们进一步思考如何在应用中更好地发挥 Transformer 的威力，比如改进调参的策略或者尝试自动调参等。

五、参考文献

[1] Cho, B.van Merrienboer, D.Bahdanau, and YBengio. On the properties of neural machinetranslation: Encoder-decoder approaches. arXiv preprint arXiv: 1409. 1259, 2014.

[2] Graves A. Supervised Sequence Labelling with Recurrent Neural Networks[M]. Springer Berlin Heidelberg, 2012: 5-13.

[3] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[4] Ian Goodfellow, Yoshua Bengio. Deep learning[M].2017.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling[EB/OL]. 2014-12-11.

[6] Rongxi Wang, Caiyuan Peng, Jianmin Gao, Zhiyong Gao, Hongquan Jiang. A dilated convolution network-based LSTM model for multi-step prediction of chaotic time series[J]. Computational and Applied Mathematics,2019,39(9).

[7] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, Kaiming He. Aggregated Residual Transformations for Deep Neural Networks[C]. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017:1492-1500

[8] S.Karimi, X.Dai, H.Hassanzadeh, A.Nguyen. Automatic Diagnosis Coding of Radiology, Reports: A Comparison of Deep Learning and Conventional Classification Methods, 2017.

[9] M. T. Luong, H. Pham, C. D. Manning. Effective approaches to attention-based neural machine translation. arXiv preprint. arXiv:1508.04025, 2015.

[10] W. Yin, H. Schiitze, B. Xiang, B. Zhou. Abcnn: Attention-based convolutional neural network for modeling sentence pairs[J]. Transactions of the Association for Computational Linguistics,2016, vol. 4:259-272.

[11] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.