# 基于门控制循环单元的协同推荐系统

**摘 要**：针对用户行为中的时序信息以及深层训练网络中存在的梯度爆炸问题，本文提出一种基于批量归一化门控制循环单元网络(Batch Normalization Gated Recurrent Unit，BN\_GRU)模型。将用户历史行为信息转换成时间序列，结合批量归一化BN算法将GRU门控制循环单元结构进行改进，提升模型鲁棒性。BN\_GRU网络模型最大限度地保证每次正向传播输出在同一分布，消除内部协变量转移的影响，使网络输入分布保持稳定，从而加速深层神经网络的训练速度。通过BN\_GRU模型挖掘用户行为序列中蕴含的信息，动态地捕捉用户的兴趣变化，从而为用户提供精确的推荐服务。实验表明提出的基于循环神经网络的推荐算法性能要优于User-KNN、Markov Chain等传统协同过滤算法。

**关键词：**推荐系统，特征学习，门控制循环单元网络，批量归一化

1. 引言

随着互联网持续稳健发展，网络信息出现了爆炸式增长已造成严重的“信息过载”问题。推荐系统为解决“信息过载”问题提供了十分有效的工具。它极大地降低了用户在搜索和选择商品的时间成本，并提升用户的粘合度以及激发用户的潜在购买意愿，可为商家带来更多的收益。然而，在享受推荐系统带来便利的同时，推荐系统面临着严峻的挑战。在传统协同过滤推荐系统中，当用户的喜好随时间发生改变或者受到外部环境因素影响时，用户行为特征会产生一定的偏差，而用户行为中的时序信息常常被其所忽视。目前，深度学习已经是推荐系统领域一个重要的研究方向。采用深度学习方法对用户行为建模时，当前网络层输入的数据分布会随着隐藏层数的增加越来越受到上一网络层参数变化的影响，使神经网络运算变得复杂且存在梯度爆炸问题。

针对上述问题，本文提出一种基于批量归一化门控制循环单元网络(Batch Normalization Gated Recurrent Unit，BN\_GRU)模型。通过BN\_GRU来刻画用户行为序列所在当前时刻的输入与上一时刻输出信息的关联，挖掘用户行为序列蕴含的时序信息，构建更加符合用户偏好需求的推荐模型。

1. 相关工作

近年来，为了构建一个高效的个性化推荐系统，研究者提出了许多相关技术方案。传统的推荐算法[1]主要包括协同过滤推荐算法、基于内容的推荐算法、混合推荐算法等。基于协同过滤推荐算法[2]的优点在于算法本身易于实现以及推荐原理更容易被理解，但它也面临着在海量的数据库中存在着数据稀疏问题。基于内容的推荐算法[3]是通过提取相关项目的主要属性特征，计算其与用户偏好项目之间的相似性，筛选匹配出用户可能感兴趣的物品。该算法更加侧重于对物品属性内容的挖掘分析，可有效缓解冷启动问题。但是由于不使用其他用户的评分，往往导致推荐项目的多样性和新颖性不断下降。基于混合的推荐算法[4]融合多种推荐技术的优点，可以弥补单一推荐算法带来的弱点和限制。目前应用大多将基于内容的推荐方法融于协同过滤中，利用基于内容的推荐算法弥补协同过滤存在的数据稀疏性和冷启动问题[5]。采用混合推荐方案，可保证推荐准确率的条件下提升推荐结果的多样性，但它也面临着大规模数据的多模态、分布不均匀等复杂特性，且需要依赖丰富的专家经验、专业领域知识和工程技术来进行特征提取。针对以上传统的推荐算法存在的问题，研究人员将深度学习融入推荐系统中，构建更加贴合用户偏好需求的用户模型，以提高推荐系统的性能和用户满意度。

基于深度学习的推荐系统通常将用户和项目的交互信息作为神经网络的输入，利用多层网络模型学习用户和项目更加深层次的特征表示，并基于这种特征表示为用户产生项目推荐。Salakhutdinov等人[6]提出一种基于受限玻尔兹曼机（Restricted Boltzmann Machine，RBM）的协同过滤推荐模型对大型评级数据集进行建模，并为这类建模提供高效的学习和推理过程。当RBM模型和SVD预测模型线性组合时，推荐性能比Netflix系统更加优越。Wu Y等人[7]提出一种协作去噪自动编码器（Collaborative Denoising Auto-Encoders, CDAE）的新方案，使用Auto-Encoder框架制定最佳前N项推荐问题。在公共数据集上的实验结果表明，CDAE模型在各种常用评估指标上始终优于最先进的前N项推荐方法。Gong Y等[8]人提出一种基于注意力的卷积神经网络（Convolutional Neural Networks,CNNs）执行标签推荐任务，将卷积神经网络用作于微博的特征提取，该模型融合了一个全局通道和一个局部注意力通道。从微博真实数据集的实验结果表明，所提出的模型比SVD等现有技术模型有更好的性能。在推荐系统领域，用户的未来行为和历史行为有着密不可分的时序关联。卷积神经网络和其他全连接网络在同一层的网络节点是无连接的，该种网络结构对序列数据建模往往显得无能为力。~~同时，随着隐藏层数的增加，深度神经网络的训练将越来越复杂并且存在梯度爆炸问题。~~

~~针对以上存在的问题，本文提出一种基于批量归一化门控制循环单元网络BN\_GRU模型，对用户历史行为数据的语义和时序信息进行建模，充分挖掘用户行为次序关联，同时最大限度减少网络内部协变量转移来提升模型泛化能力。~~

~~本文，来解决深度网络训练过程中梯度爆炸问题以及提升模型泛化能力。地保证每次正向传播输出在同一分布，~~

1. BN\_GRU神经网络

推荐系统通过挖掘用户的偏好、兴趣，从海量信息中筛选出重要的信息片段生成用户感兴趣的项目，引导用户发现自身需求信息，为其提供个性化的推荐列表服务。在推荐系统领域中，用户的行为信息具有长时相关性并蕴含着丰富的序列信息，基于矩阵分解(Matrix Factorization, MF)等传统的协同过滤推荐系统更加注重预测出来的列表是否存在于用户的未来行为之中，往往忽略用户行为序列中的时序信息。此类推荐算法不适宜捕捉随时间不断变化的用户品味或兴趣，而且还伴随这庞大而又繁琐的特征提取。因此，需要一个具有记忆能力的神经网络对用户行为信息建模，并得出相应的预测结果。

循环神经网络是一种内部动态展示时序行为的神经网络，是目前最通用的序列预测方法，已经在机器翻译[10]、语音识别[11]、标注生成[12]领域取得的成功，使它成为解决推荐问题的良好候选者。通常在深层网络训练过程中，将网络内部节点产生的分布变化称为内部协变量转移（Internal Covariate Shif）。随着循环神经网络隐藏层数的增加，网络层输入的数据分布严重受到上一网络层参数变化的影响，使得网络模型运算复杂并且存在梯度爆炸问题。门控制循环单元(Gated Recurrent Unit，GRU)[13]是一个通过带自反馈的神经元且在时间维度上权值共享的循环神经网络，通过选择性遗忘之前累计的信息，来达到控制依赖信息的累计速度。GRU网络组合了遗忘门和输入门到一个单独的“更新门”中，直接使用当前的细胞状态与前一个历史状态之间形成依赖关系，不再引入额外的记忆单元门。它是一种比长短记忆的时间递归神经网络(Long Short-Term Memory，LSTM)[14]更加简单、效率更高的循环神经网络。但当训练过程中GRU网络层数增加时，同样会面临内部协变量转移问题。

为了解决训练过程中网络中间层数据分布发生改变的情况，Sergey Ioffe等人[15]提出批量归一化（Batch Normalization，BN）算法。BN算法采用对每个标量特征单独归一化来替代图层输入与输出中特征联合的白化处理，使其具有零均值和单位方差；同时对每个神经元成对引入可学习参数、，使得插入到网络中的变换可以恢复至原始某一层网络所学习的特征分布，从而能够保持输入数据分布的稳定，消除内部协变量转移带来的影响。本文结合批量归一化算法思想，提出批量归一化门控制循环单元网络(Batch Normalization Gated Recurrent Unit，BN\_GRU)模型。其基本思想是对GRU网络内部结构进行适当的改进，将网络中每一层运算出来的数据归一化成均值为0方差为1的标准高斯分布（Gaussian distribution）。BN\_GRU模型进一步降低网络内部协变量转移以及最大限度地保证每次正向传播输出在同一分布，动态的加速深度神经网络的训练速度。除此之外，BN\_GRU模型还减少了对网络模型dropout、L2正则化等参数选择问题，更有效地提升模型泛化能力。其内部结构图如图1所示：

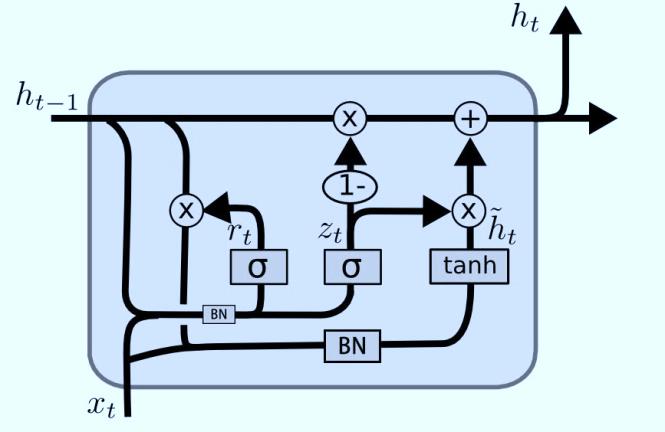


图1 BN\_GRU单元结构

图1中，更新门表示当前状态信息受前一个时刻的状态影响的程度；复位门代表控制遗忘上一时刻的状态信息的程度；候选隐藏层表示当前时刻的新信息的状态； BN\_GRU神经元最终的输出值为。在计算更新门、复位门、候选隐藏层状态时加入批量归一化算法，具体如公式1所示：

 (1)

1. 基于BN\_GRU神经网络的推荐算法

本文基于批量归一化门控制循环单元网络推荐模型更加注重每个行为次序之间的关联，利用数据驱动的方式学习行为序列数据内部蕴含的特征，动态揭示用户的喜好变化。

4.1 BN\_GRU神经网络推荐算法架构

BN\_GRU神经网络推荐算法框架如图2所示。首先对原始数据集进行预处理，生成用户行为时间序列；然后通过构建模型样本模块，构造BN\_GRU神经网络所需样本格式，并将数据集拆分成训练集、验证集、测试集；其次，利用训练集和验证集构建BN\_GRU网络推荐模型；最终，对模型中的超参数进行调优，提高模型的精确度。

C:\Users\yang\Desktop\图\002.emf

图2 推荐系统模型架构图

* 1. 构造BN\_GRU模型样本

用户历史行为数据中蕴含着丰富的时序信息，将其构建成BN\_GRU模型训练所需的样本格式。主要分为以下两个步骤：

步骤一，数据集进行预处理。对原数据集的缺失值填充等常规预处理，数据集进行时间戳排序并按用户分组，对比前后*userId*作为判断是否为同一个用户，为每个用户按行为数据发生的时间顺序生成行为序列[*userId,1st-itemId,2nd-itemId,…,n-itemId*]添加到*preFormatList*集合集合中。

步骤二，构建训练样本。首先遍历*preFormatList*集合中的每一项元素，如果该用户历史行为总次数少于最大序列长度*max\_length*，则过滤掉该用户，以免影响训练精度。然后将每个用户历史行为序列构建成len(*userSequence*) - *max\_length*个长度为*max\_length*+1的子序列，该子序列前*max\_length*个元素为BN\_GRU模型的输入*X*，最后一个元素作为模型的输出标签*Y*。

算法1 构造训练样本

**INPUT:** *rawDataSet* [[*userId::itemId::rating::timestamp*]…]; *max\_length*

**OUTPUT:** *generatorSample*[ [*1\_ itemId, 2\_itemId,…, max\_length\_ itemId*]…]

**BEGIN**

// Step1: Sample pretreatment

1. **IF**(*data.rating* == NULL)
2. *data. rating* = 0
3. **END IF**
4. *data.*sort(*timestamp*) **AND** *data*.groupby(*userId*)
5. *preFormatList =* []; *userItemList* = []; *preUserId* =-1;
6. **FOR** **EACH** *item* **IN** *data:*
7. **IF**(*item.userId* != *preUserId*)
8. *preFormatList.*add(*userItemList*)
9. *userItemList* = [];
10. *preUserId = item.userId*
11. *userItemList*.add(*item.userId*)
12. **CONTINUE**；
13. **END IF**
14. *userItemList*.add(*ItemId*)
15. **END FOR**

// Step2: Build train sample data

1. *generatorSample = []*
2. **FOR** **EACH** user*Sample* **IN** *generatorSample:*
3. *user\_id, userSequence = userSample*
4. **IF**(*userSequence*.size() < *max\_length*)
5. **CONTINUE**;
6. **END IF**
7. **FOR** (*i* = *max\_length* < *userSequence*.size()):
8. *generatorSample*.add(*userSequence*[*i-* *max\_length*]*, i*)
9. *i++*
10. **END FOR**
11. **END FOR**
12. **RETURN** *generatorSample*

**END**

* 1. 构建BN\_GRU推荐模型

BN\_GRU网络推荐算法模型的基本思想是将用户行为预测问题类比于神经网络中的多分类问题。本文利用深层次的网络模型来提升RNN的提取特征功能[16]如图3所示，构建2层BN\_GRUCell循环网络与一层全连接网络（FullConnectCell）结构。在BN\_GRU网络中选取Softplus函数作为模型的激活函数，Softplus函数相比ReLU函数更加平滑同时相对保留小于0的输出值。在超参数优化方面，采取一种学习率自适应的优化算法Adam (Adaptive Moment Estimation)[17]。Adam是扩展式的随机梯度下降算法，本质上结合了Adagrad[18]处理稀疏梯度的优点以及RMSprop[19]善于非平稳目标处理的特性，利用梯度的一阶矩估计和二阶矩估计对每个参数进行动态的调整。它可以保证经过偏置校正后，学习率在每一次迭代过程中都会有一个明确的范围，使得参数整体波动不大，内存也没有额外的需求。

../../毕业论文/3-5%20本文中推荐算法中的门控制循环单元网络模型-修改.emf

图3 BN\_GRU推荐网络结构

构建BN\_GRU网络模型主要分为定义网络结构和网络训练两个部分见算法2所示。定义2个占位符，*X*作为输入序列，*Y*为对应的标签，并设置样本序列最大时间长度（max\_length）。由于采用批处理（mini\_batch）操作且*X*的原始数据是逐个输入网络，固按最大时间长度将*X*转化成max\_length个数组，每个数组包含batch\_size个元素。将输入序列*X*首先输入dynamic\_rnn函数；然后将其结果接入全连接Full\_ConnectCell层得到预测序列*pred*；最后该预测序列*pred*进行softmax分类生成n\_items个元素的*result*数组，网络的正向传播结束。紧接着模型采用softmax交叉熵损失函数来衡量预测值*pred*与真实值*Y*之间的概率分布距离。通过随时间反向传播算法和梯度下降算法不断调整循环神经网络中权重参数取值，直到符合停止准则为止。

算法2 构建BN\_GRU推荐模型

**INPUT:** *generatorBatchSample*; n\_hidden = [100, 50], maxIterations; batch\_size; *max\_length*;

**OUTPUT:** *BN\_GRUModel;*

**BEGIN**

// Step1: build a dynamic BN\_GRU

1. *X* = placeholder([batch\_size, *max\_length*, n\_items])
2. *Y =* placeholder([batch\_size,n\_items])
3. multi*BnGruCell* = MultiBN\_GRUCell(*activation =* softplus (), *num\_units* = n\_hiddens)
4. *rnn\_outputs, \_states* = dynamic\_rnn(multi*BnGruCell*, *X, Sequence\_length*)
5. *pred =* fullConnectCell(*rnn\_outputs(*-1*),* n\_events)
6. *loss*= softmaxCrossEntropy(*pred, Y* )
7. *optimizer =* Adam().minimize(*loss*)

// Step2: model: train start

1. *Iterations=0;*
2. **WHILE**(*Iterations<* maxIterations)
3. *batchSample* **=** next(*generatorSample*)
4. *batch\_X* = *batchSample[:-1];*
5. *batch\_Y* = *batchSample[-1]*
6. *\_, cost=* Session*.*run([*optimizer, loss*]*,* feed(*batch\_X, batch\_Y*))
7. *Iterations++;*
8. **END WHILE**
9. *BN\_GRUMode =*Save(Session)

**END**

1. 实验设计与结果分析

实验的数据集来源于GroupLens Research提供的MovieLens 1M电影评级数据集，包括6040个用户对象信息，100多万条用户对电影的评价信息，以及近4000个电影描述信息。本文将构建BN\_GRU网络模型对这*Max\_length*次观看电影记录数据进行训练学习，然后对紧接着后续观看电影名称进行预测。实验平台是基于Tensorflow[20]深度学习框架搭建，整个训练过程是基于具有16GB的内存、Intel Core i7处理器的MacBook Pro (Retina, 15-inch, Mid 2015)进行模型训练。对构建完成之后的实验样本进行3:1:1拆分，其中数据样本的60%用于训练模型，20%的样本用作模型的验证数据，剩下的20%用来做测试样本。BN\_GRU推荐模型的默认设置如表1所示：

表1 BN\_GRU模型默认配置

|  |  |
| --- | --- |
| 参数 | 默认值 |
| RNN类型cell | BN\_GRU |
| 隐藏层规模n\_hiddens | 100，50 |
| Batch-size | 32 |
| Max\_length | 25 |
| Optimizer | Adam |
| 学习率 | 0.001 |
| loss | SoftmaxCrossEntropy |

5.1 实验评价标准

评估系统性能设计对于了解各种推荐算法的有效性至关重要。对于推荐系统的评估往往是多个方面的，单一指标无法衡量在推荐系统的多目标性，需要多种评估策略从不同的角度进行评判。本文主要从召回率、用户覆盖率、短期预测精度三个评价指标来衡量推荐系统的性能。

（1）召回率

它表示在系统预测的推荐项目列表中的符合用户期望的项目数占用户所有喜欢的项目的比例。

 （2）

其中表示测试集的总数，表示用户在预测时间段内消费期望项目的总数，表示推荐系统提供的*k*个推荐项目和用户在预测时间段内消费期望项目的交集。

（2）用户覆盖率

用户在系统提供的*k*个推荐项目中，至少接受到一个正确的推荐项目的人数占测试总人数的比例。

 （3）

其中m表示测试集的总人数，表示推荐系统提供的的*k*个推荐项目集合，表示测试用户i在预测时间内消费项目的总数。

（3）短期预测精度

 （4）

其中表示推荐系统提供的的*k*个推荐项目集合，表示测试集的总数。受到用户覆盖率的启发，本文在此稍做改进，将测试用户*i*期望的项目定义为用户在预测时间段内前三个消费项目。

5.2 实验设计与分析

5.2.1 BN\_GRU与RNN、GRU的性能对比

RNN最主要的特性是它具有记忆的功能，可以充分利用历史的信息来辅助当前的决策。在MovieLens 1M数据集上对BN\_GRU网络、GRU网络、RNN网络进行对比实验，观察各自的性能差异，如图4所示。

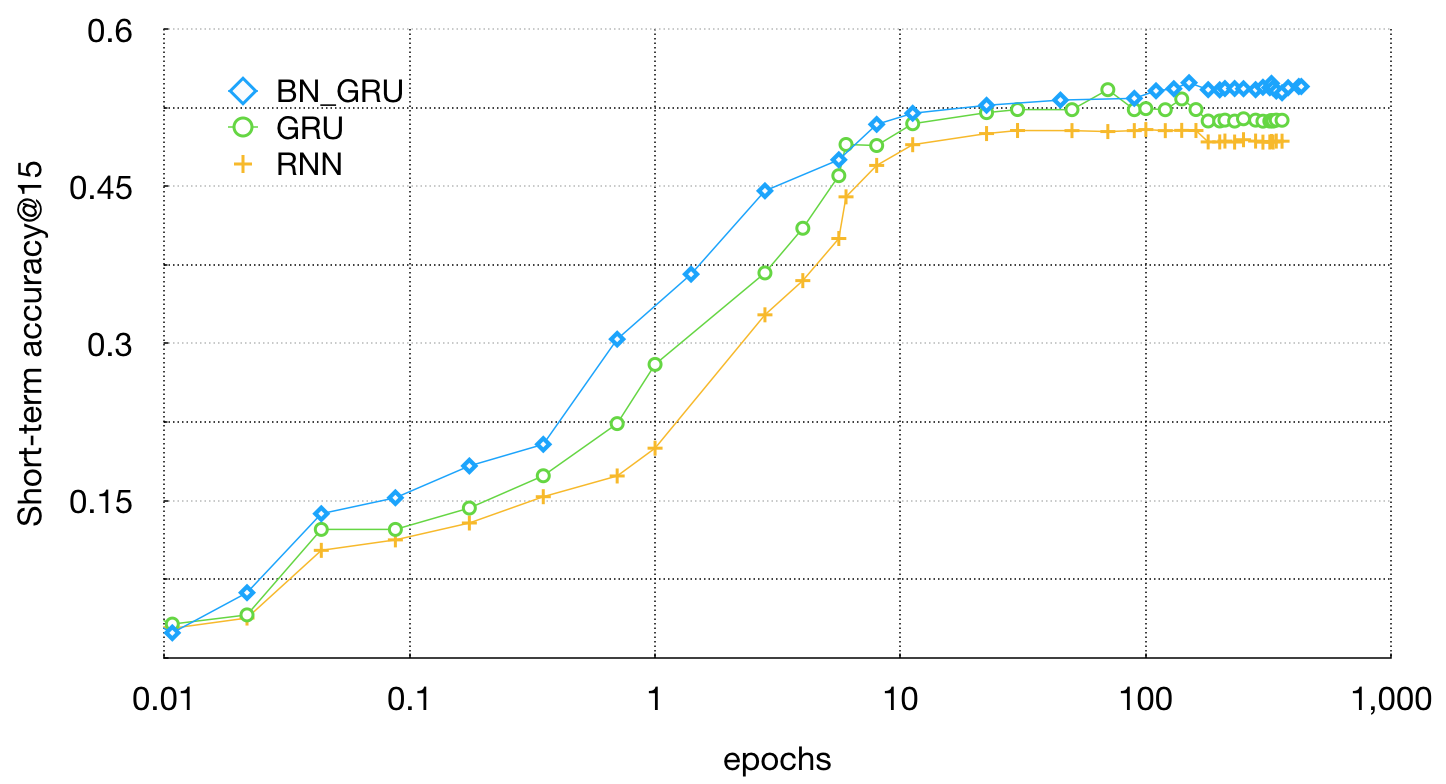


图4 RNN体系结构模型对比

从图中1 epochs的时候，BN\_GRU算法模型比RNN、GRU基本模型精度高9%、3%左右。可得出在相同的epochs上，批量归一化门控制循环单元BN\_GRU模型对于门控制循环单元GRU模型、RNN基本模型提高了精确度并且拥有更快的训练速度，尤其是在模型前期训练的情况下，这种速度优势更为明显。

5.2.2 学习速率对模型的影响

学习速率是控制损失函数的梯度调整网络权重大小的超参数，它决定了权重在一个小批量样本中沿着梯度方向更新的距离。学习率设置越低，则损失函数的变化速度也将越慢，模型的性能将更加准确。但其缺点是优化需要花费更长的时间才能达到收敛。相反，学习率设置过于偏高且训练过程中容易发生震荡甚至趋向发散，这便导致权重更新偏大模型优化数据超越最小值，使得损失函数变得更糟。如图5所示，本文将采用Adam调度器在MovieLens 1M数据集上观察不同的学习率参数对模型的影响。

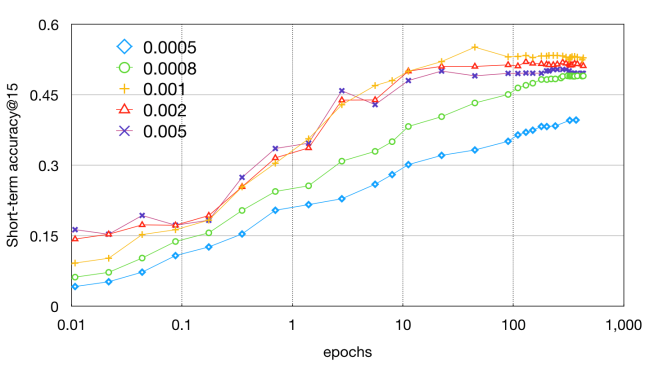


图5 Adam调速器学习率的影响

从图5可以发现，当=0.0005时，模型的训练速度明显降低，但总体趋势比较平稳，而当=0.005时，模型的学习效率最高，但与此同时，在模型训练过程中会出现小幅度的震荡。经过一系列不同学习率之间的对比, =0.002左右模型展现了较佳的效果。

5.2.3 神经元数量对模型的影响

神经元数量是影响模型训练的另一重要参数。理论上，单隐层神经元个数与神经网络逼近连续函数的能力成正比。通常神经网络的神经元数越多，模型对特征的抽象程度将会越高，推荐预测的精度也将会随之提升。但随之而来的过度地添加神经元数目将会造成训练时间开销剧增问题。本节采用MovieLens 1M数据集进行相关对比实验，评估不同的神经元数对模型产生的影响。

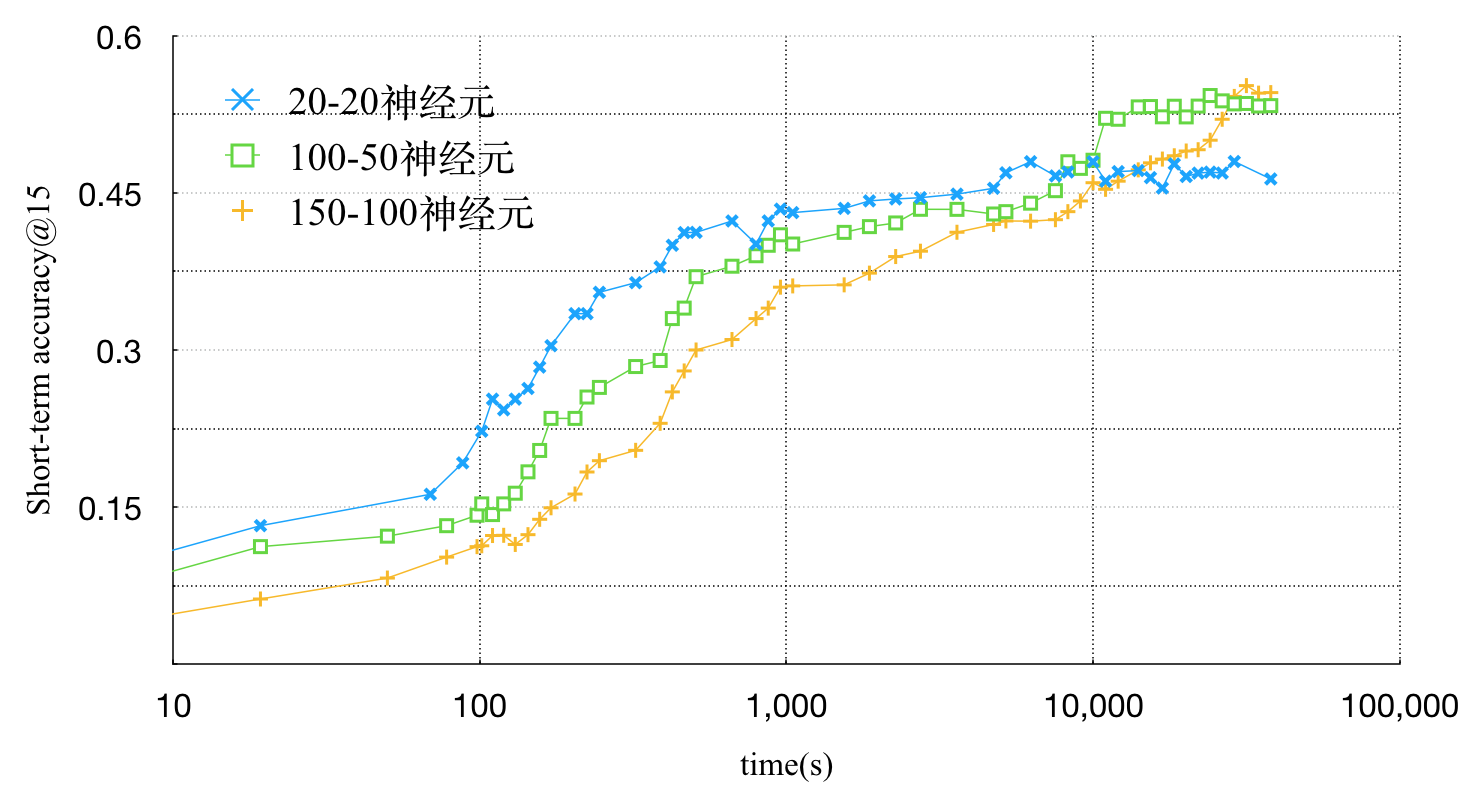


图6 隐藏层神经元数量对模型产生的影响

从图6可以观察到，由20-20个神经元组成的隐藏层网络在训练过程中，模型性能受到了严重的限制。当模型短期预测精度达到45%时，具有20-20个神经元的隐藏层需要耗时1.3小时；100-50个神经元的隐藏层需要2个小时；而具有150-100个神经元的隐藏层需要2.7小时才可达到相同的精度。当模型趋于收敛时，由150-100个神经元组成隐藏层的模型在短期预测精度要比由20-20个神经元构成的隐藏层模型高出5个百分点。由此可以看出，随着神经元的数量的增加，模型的训练时间和预测精度也随着增加。这说明了在合理的范围内，神经元的数量越多，模型则可以学到更加丰富的信息，但其训练速度将受到影响。

5.2.4 batch\_size对模型的影响

在梯度下降的过程中，需要对训练样本进行求损失值处理，然后再决定下降的方向。在深度学习中，一般将全部数据集分成等量的子集，使用其中的子集进行一次梯度下降，继而更新其参数。这些子集称为mini batch，并用batch\_size超参数来描述mini batch中样本的个数，遍历所有子集称为echo。在MovieLens 1M数据集上针对不同的batch\_size做相关实验结果如图7所示，可以观察到选取不同的batch-size对模型性能的影响。

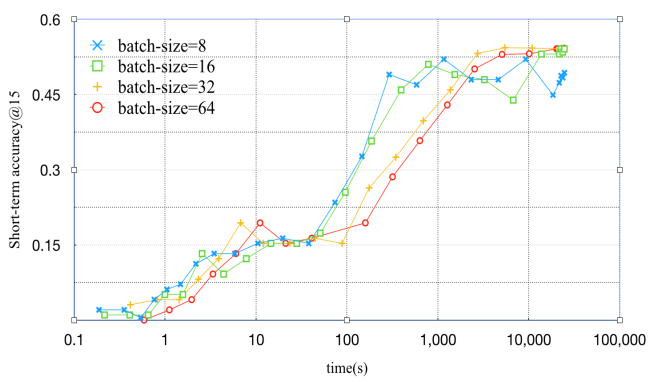


图7 不同的batch\_size对模型的影响

当batch\_size=8时，模型在训练过程中明显呈现的震荡，模型精度提升很快；当batch\_size=64时，达到相同的精度所需要更长的时间，但模型总体趋势变得更加稳定；当batch\_size=32时，模型达到一个比较良好的状态。实验表明在合理的范围内，batch\_size值越大其确定的梯度下降将会越准确，训练过程中引起的震荡将会越小。

5.2.5 对比实验

在MovieLens 1M数据集上从召回率，用户覆盖率、短期预测精度指标方面观察POP、User–KNN、Markov Chain、FISM（Factored Item Similarity Methods）[21]、BN\_GRU推荐算法的性能对比，横向衡量BN\_GRU网络模型的性能。

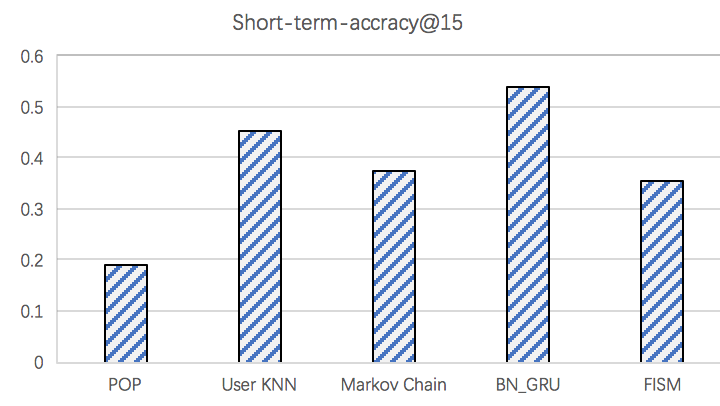


图8 相关推荐算法模型对比

表2 相关推荐算法模型性能对比

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据集 | 方法 | 指标@15 | | |
| Short-term-accracy | Recall | User coverage |
| MovieLens 1M  数据集 | POP | 0.1919 | 0.0561 | 0.7475 |
| User KNN | 0.4545 | 0.0886 | 0.8687 |
| Markov Chain | 0.3737 | 0.0664 | 0.8182 |
| FISM | 0.3532 | 0.0543 | 0.8500 |
| BN\_GRU | 0.5394 | 0.0774 | 0.9082 |

从表2和图8中可得出以下结论：BN\_GRU网络模型在召回率、用户覆盖率、短期预测精度指标方面有着明显的优势。尤其是在用户覆盖率方面和短期预测精度指标上，优于POP、User–KNN、Markov Chain、FISM四种算法性能。经过多次的试验BN\_GRU网络模型在用户覆盖率方面均达到88%以上，相对比其他算法平均高6%左右，而在短期预测精度指标方面则拥有更好的表现。

1. 总结

针对如何构建高效的推荐算法，本文主要研究内容包括以下几个方面：将批量归一化算法于门控制循环单元网络结构相结合，提出BN\_GRU推荐算法模型。研究BN\_GRU算法对用户行为序列进行建模，更加注重每个行为次序之间的关联，深度挖掘用户行为信息时间序列上的丰富信息，以及捕捉用户的兴趣偏移。重点研究如何构造模型训练样本以及对BN\_GRU门控单元网络推荐算法的建模与设计，如选择不同学习率、神经元数量等对模型调优，使之具有更好的鲁棒性。通过在MovieLens 1M数据集上实验表明，本文提出基于循环神经网络的推荐算法在用户覆盖率、短期预测精度、召回率指标上面相比于User-KNN算法、FISM算法等四种传统推荐算法有着显著的提升。值得注意的是，虽然本文仍存在大量不足，接下来的工作还将有许多需要改进的地方。如循环神经网络的输入除了用户观看电影行为序列作为之外，还可以尝试考虑加入额外的特征信息去提升循环神经网络的推荐系统的精度，如电影自身的热度、流行度、用户的性别、年龄等基本特征。

# 参 考 文 献

1. Adomavicius G, Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. IEEE transactions on knowledge and data engineering, 2005, 17(6): 734-749.
2. Gong S. A collaborative filtering recommendation algorithm based on user clustering and item clustering[J]. JSW, 2010, 5(7): 745-752.
3. Pazzani M J, Billsus D. Content-based recommendation systems[M]//The adaptive web. Springer, Berlin, Heidelberg, 2007: 325-341.
4. 黄璐, 林川杰, 何军, 等. 融合主题模型和协同过滤的多样化移动应用推荐[J]. 软件学报, 2017, 28(3): 708-720.
5. Miranda T, Claypool M, Gokhale A, et al. Combining content-based and collaborative filters in an online newspaper[C]//In Proceedings of ACM SIGIR Workshop on Recommender Systems. 1999.
6. Salakhutdinov R, Mnih A, Hinton G. Restricted Boltzmann machines for collaborative filtering[C]//Proceedings of the 24th international conference on Machine learning. ACM, 2007: 791-798.
7. Wu Y, DuBois C, Zheng A X, et al. Collaborative denoising auto-encoders for top-n recommender systems[C]//Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. ACM, 2016: 153-162.
8. Gong Y, Zhang Q. Hashtag Recommendation Using Attention-Based Convolutional Neural Network[C]//IJCAI. 2016: 2782-2788.
9. Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533.
10. Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[C]//Advances in neural information processing systems. 2014: 3104-3112.
11. Graves A, Jaitly N. Towards end-to-end speech recognition with recurrent neural networks[C]//International Conference on Machine Learning. 2014: 1764-1772.
12. Karpathy A, Fei-Fei L. Deep visual-semantic alignments for generating image descriptions[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2015: 3128-3137.
13. Cho K, Van Merriënboer B, Bahdanau D, et al. On the properties of neural machine translation: Encoder-decoder approaches[J]. arXiv preprint arXiv:1409.1259, 2014.
14. Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.
15. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. arXiv preprint arXiv:1502.03167, 2015.
16. Liu W, Wang Z, Liu X, et al. A survey of deep neural network architectures and their applications[J]. Neurocomputing, 2017, 234: 11-26.
17. Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.
18. Hadgu A T, Nigam A, Diaz-Aviles E. Large-scale learning with AdaGrad on Spark[C]//Big Data (Big Data), 2015 IEEE International Conference on. IEEE, 2015: 2828-2830.
19. Tieleman T, Hinton G. RMSprop gradient optimization[J]. URL http://www. cs. toronto. edu/tijmen/csc321/slides/lecture\_slides\_lec6. pdf, 2014.
20. Abadi M, Barham P, Chen J, et al. TensorFlow: A System for Large-Scale Machine Learning[C]//OSDI. 2016, 16: 265-283.
21. Kabbur S, Ning X, Karypis G. Fism: factored item similarity models for top-n recommender systems[C]//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013: 659-667.