知乎

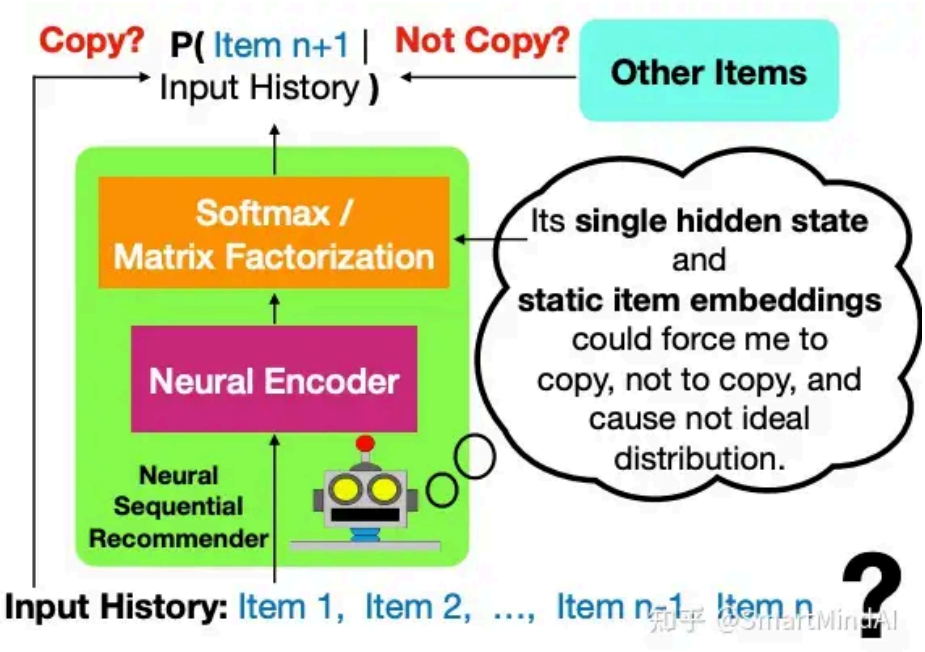
亚马逊2023研究：采用分段softmax函数有效解决序列推荐中重复商品建模难题

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我
已关注

原文《To Copy, or not to Copy; That is a Critical Issue of the Output Softmax Layer in Neural Sequential Recommenders》

Introduction

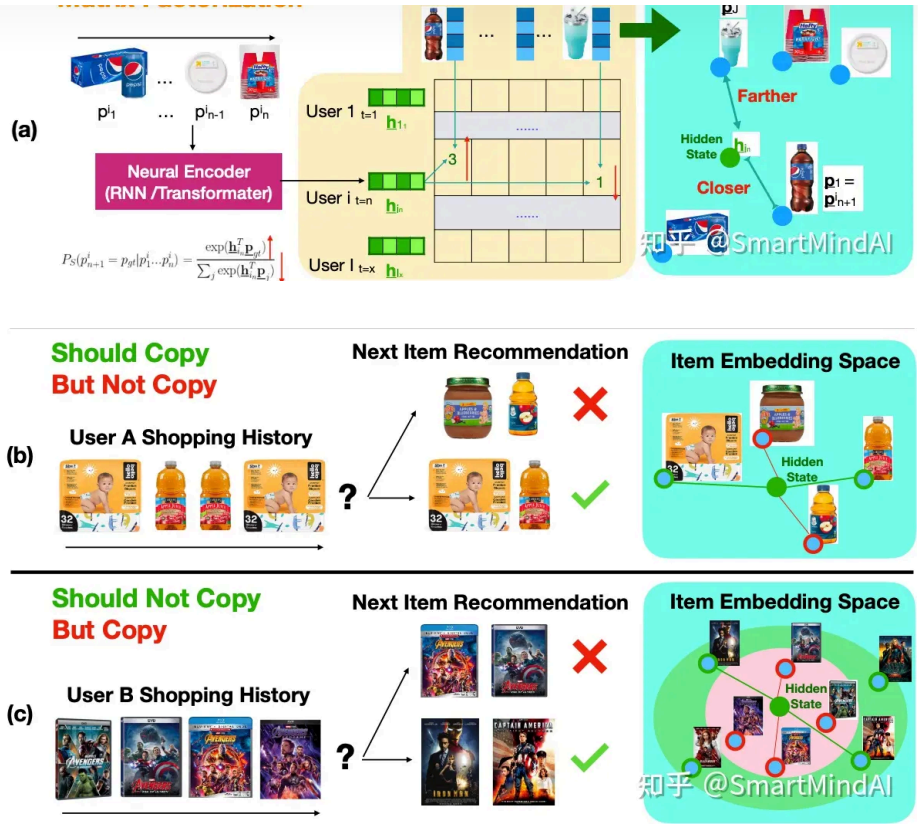
序列推荐是一种常见的推荐问题，旨在根据用户和项目的交互历史向用户推荐下一个可能感兴趣的项目。许多研究证明神经网络能够有效捕捉复杂的交互并获得最先进的性能。如图1所示，这种系统接受用户的物品历史作为输入，并输出下一物品的概率分布，其中预测概率最高的物品将被推荐给用户。此外，系统还可以根据用户过去的互动情况决定是否将这些物品推荐给他们。



但是，李2023年的一项研究发现，尽管近年来神经推荐系统有了显著的进步，但它们在某些情况下仍然无法正确处理重复行为或从历史中排除项目。为了解决这个问题，我们专注于改进神经推荐系统的输出Softmax层，并实现了显著的性能提升。Softmax层可以视为一种矩阵分解模型。

与传统的协同过滤方法中使用的固定用户嵌入不同，神经推荐系统使用RNN（循环神经网络）或Transformer架构来将历史输入项序列为用户嵌入。然后，通过交叉熵损失和softmax层，系统鼓励生成的用户嵌入与可能的下一件物品的嵌入之间产生较高的点积，如图2(a)所示。点积定义如下： dot product $:=\sum_{i=1}^{n} w_i * v_i$ 它鼓励相似的项目具有相似的项目嵌入，而相似的输入序列会被编码为类似的用户嵌入。这有助于增强系统的泛化能力。

知乎



为了缓解softmax层的问题，我们采用了softmax-CPR。softmax-CPR最初被提出是为了减少语言生成模型[+]的幻觉。它通过为输入序列中的项目使用不同的隐藏状态来计算不同项目的概率，以避免项目的相互影响。例如，对于"苹果汁"和"尿布"这类经常一起购买的产品，可以为其使用一个隐藏状态；而对于其他的项目，则使用另一个隐藏状态。这样就可以将这两个项目之间的关系独立出来，从而避免其他项目的干扰。这种方法在图2中起到了很好的效果。

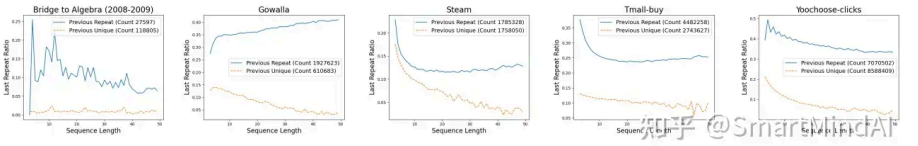
Problems

本节将详细讨论输出 softmax 层中的单个隐藏状态和静态项目嵌入所引发的问题。

Duplicated Items

在许多应用场景中，序列中的项目经常重复，并且这些重复行为可能会在不同的领域、不同的实验设置或仅在序列的不同时间发生改变。例如，在用户购买一条裙子时，用户通常首先探索多个选项（即没有太多的重复），然后反复点击一组选项来比较它们。一旦用户购买了一条裙子，就不太可能再次购买同一条裙子，这说明预测下一个点击和下一次购买的复制模式是非常不同的。

为了展示复制模式的多样性，我们在图中展示了最后一项物品的重复概率与序列长度的关系。蓝色曲线明显高于橙色曲线，这表明，一旦用户开始频繁交互，复制概率就会显著增加。此外，不同数据集中的曲线存在显著差异，说明难以设计出适用于所有领域的通用复制策略。



(a) Math Exercises (b) Locations (c) Video Games (d) Purchases (e) E-commerical Clicking

Softmax Bottleneck

输出概率可以通过一个softmax层表示为：

$$p_i = \frac{\exp(a_i)}{\sum_{j=1}^{n} \exp(a_j)}$$

知乎

$$P_{\mathcal{S}}(p_{n+1}^{i}=p_{x}|i_{n})=\frac{\exp(\text{Logit}(x,i_{n}))}{\sum_{j}\exp(\text{Logit}(j,i_{n}))}=\frac{\exp(\underline{\boldsymbol{h}}_{i_{n}}^{i}\underline{\boldsymbol{p}}_{x})}{\sum_{j}\exp(\underline{\boldsymbol{h}}_{i_{n}}^{T}\underline{\boldsymbol{p}}_{j})}$$

其中i_n表示用户i的输入项目序列的长度为n的形式$\underline{\boldsymbol{h}}_{i_n}$是由神经编码器编码的序列的隐藏状态，而$\underline{\boldsymbol{p}}_{x}$是用于商品 / 产品$x$的输出项嵌入。 在训练过程中，最大似然估计会通过最大化 $\underline{\boldsymbol{h}}_{i_n}^{T}\underline{\boldsymbol{p}}_{gt}$ 和最小化 $\underline{\boldsymbol{h}}_{i_n}^{T}\underline{\boldsymbol{p}}_{j}$

来增加实际观察到的产品$\underline{\boldsymbol{p}}_{gt}$的概率

$$P_{\mathcal{S}}(p_{n+1}^{i}=p_{gt}|i_{n})$$

这意味着隐藏状态$\underline{\boldsymbol{h}}_{i_n}$将被拉近 $\underline{\boldsymbol{p}}_{gt}$

并远离其他项嵌入$\underline{\boldsymbol{p}}_{j}$。这种隐式矩阵分解过程在图中进行了说明。 softmax 层和矩阵分解的主要局限性是其静态的项嵌入$\underline{\boldsymbol{p}}_{\cdot}$每个项目的含义对于不同的用户都是不同的，但是 softmax 层中的项嵌入是全局的，独立于输入项目序列。例如，大多数用户常常一起买 A 和 B 商品，所以他们的全局嵌入倾向于相似。然而，如果有一种类型的用户只买 A 因为他/她喜欢 A 但不喜欢 B，那么这些两个项目对于这种类型的用户来说根本不是相似的。然而，即使用户已经多次展示出他们购买 B 的低概率，推荐系统也可能被迫继续推荐 B ，因为它们的相似全球嵌入。

如图所示，当用户有很强的复制或排除历史项目的偏好时，这种差异尤其严重。在工业推荐系统中，因为嵌入模型的参数很大，所以往往不会使用太大的隐藏状态大小。另外，由于每个用户可能对不同产品组合有偏好，所以在训练数据中并不能观察到所有可能的产品分布。因此，在许多序列推荐任务中，softmax层难以通过移动项目嵌入来减少多模态分布，这也凸显了softmax瓶颈的问题。

Solutions

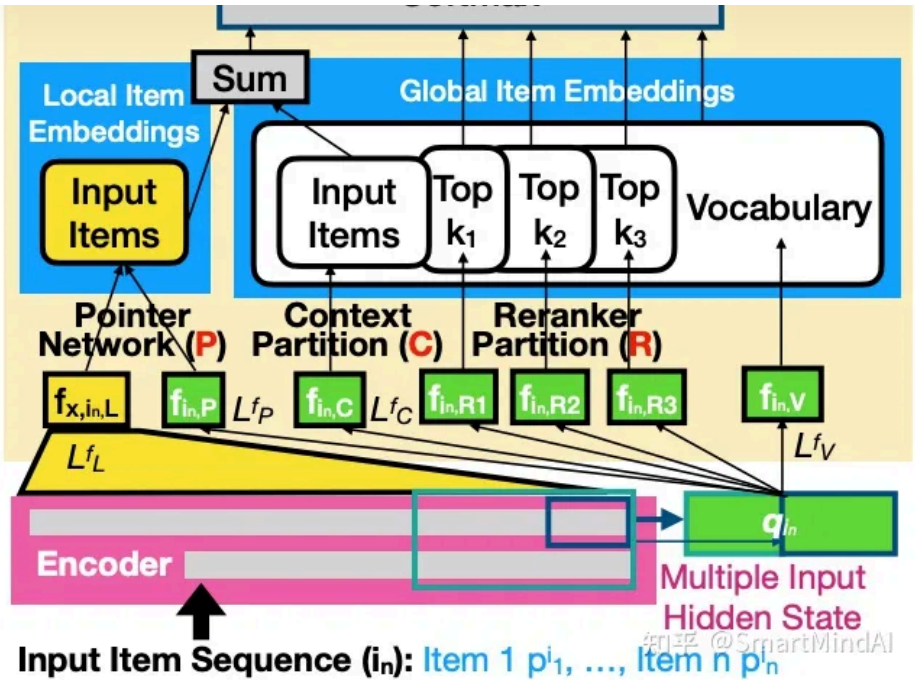
在本节中，将介绍多种解决softmax瓶颈问题的方法。

Post Processing

在工业环境中，常见问题可以通过业务洞察和启发式规则得到解决。例如，产品经理可能会发现多数用户不会连续观看同部电影，故只需增加一个后处理步骤，从推荐列表中剔除已观看电影。然而，《图2重复率统计》所显示的简单统计结果表明寻找通用规则并不容易。此外，在某些领域，这类规则可能很快变得过于复杂难以管理。例如，一位用户可能希望外卖应用推荐新菜品，但有时又喜欢购买尝试过超过m次的菜品（以免筛选出糟糕餐厅）。在这种情况下，学习大型训练集中复制模式或许较手动设计的启发式规则更容易及更有效。

Softmax-CPR

知乎



Softmax-CPR通过结合上下文分区、指针网络以及重排分区三种方法来优化输出softmax层。首先，引入上下文分区方法，使得softmax层的logit计算发生变化。在softmax公式中，softmax层使

$$\mathbf{Logit}(x, i_n) = \underline{\boldsymbol{h}}_{i_n}^T \underline{\boldsymbol{p}}_x$$

在上下文分区中，令项x的logit⁺表示为$\mathbf{Logit}_1(x, i_n)$和$\mathbf{Logit}_2(x, i_n)$，其中第1个logit对应前一个词，而第2个logit对应当前词。然后，通过指针网络连接上下文分区，实现更好的语义关联。最后，通过重排分区方式，使得临近词的上下文分区内距离更近的项能获得更高的权重。这些改进能有效地提高模型性能。

$$\begin{array}{ll} \underline{\boldsymbol{f}}_{i_n,C}^T \underline{\boldsymbol{p}}_x & \text{if } x \in i_n \\ \underline{\boldsymbol{f}}_{i_n,V}^T \underline{\boldsymbol{p}}_x & \text{O/W} \end{array}$$

其中

$$\underline{\boldsymbol{f}}_{i_n,C} = L_C^f(\underline{\boldsymbol{h}}_{i_n})$$

和

$$\underline{\boldsymbol{f}}_{i_n,V} = L_V^f(\underline{\boldsymbol{h}}_{i_n})$$

是隐藏状态的线性投影。 在这篇论文中

$$L_\cdot^f(\underline{\boldsymbol{h}}_{i_n}) = W_\cdot \underline{\boldsymbol{h}}_{i_n} + \underline{\boldsymbol{b}}_\cdot$$

（例如

$$L_C^f(\underline{\boldsymbol{h}}_{i_n}) = W_C \underline{\boldsymbol{h}}_{i_n} + \underline{\boldsymbol{b}}_C$$

） 并且每个线性投影层在训练过程中都会学习不同的参数（权重$W_\cdot$和偏置$\underline{\boldsymbol{b}}_\cdot$）。 上下文分区允许推荐系统学习何时复制输入项何时排除输入项。例如，在图《》（b）中，推荐系统可以在不被婴儿食品干扰的情况下将 $\underline{\boldsymbol{f}}_{i_n,C}$

放置到苹果汁和尿布之间，因为 $\underline{\boldsymbol{f}}_{i_n,C}$ 只对应于输入项的隐藏状态。 类似地，在图中，推荐系统可以学习输出

知乎

来计算输入项的 logit，其中

$$\underline{\boldsymbol{f}}_{i_n,P} = L_P^f(\underline{\boldsymbol{h}}_{i_n})$$

和

$$\underline{\boldsymbol{f}}_{x,i_n,L}$$

是对应于物品x的线性投影隐藏状态嵌入平均值。 这意味着指针网络允许推荐器预测输入项的局部和上下文相关的嵌入。在上下文划分或指针网络中，我们只使用投影后的隐藏状态单独计算输入项的 logits[+]。然而，也可能存在未被探索的项。为了解决这个问题，重排序分区使用另一个新的隐藏状态分别计算最可能的k项的 logits $\underline{\boldsymbol{f}}_{i_n,R1}^T \underline{\boldsymbol{p}}_x$

Softmax-CPR将上述所有方法结合在一起。如图所示。

$$\begin{cases} \underline{\boldsymbol{f}}_{i_n,C}^T \underline{\boldsymbol{p}}_x + \underline{\boldsymbol{f}}_{i_n,P}^T \underline{\boldsymbol{f}}_{x,i_n,L} & \text{if } x \in i_n \\ \underline{\boldsymbol{f}}_{i_n,R1}^T \underline{\boldsymbol{p}}_x & \text{if } x \in P(k_1) - i_n \\ \underline{\boldsymbol{f}}_{i_n,R2}^T \underline{\boldsymbol{p}}_x & \text{if } x \in P(k_2) - P(k_1) - i_n \\ \underline{\boldsymbol{f}}_{i_n,R3}^T \underline{\boldsymbol{p}}_x & \text{if } x \in P(k_3) - P(k_2) - P(k_1) - i_n \\ \underline{\boldsymbol{f}}_{i_n,V}^T \underline{\boldsymbol{p}}_x & \text{O/W} \end{cases}$$

在这个方法中，$P(k_3)$是最高 $\underline{\boldsymbol{f}}_{i_n,V}^T \underline{\boldsymbol{p}}_x$

的前k_3个词$P(k_2)$是最高 logits 的前k_2个词，同样地，对于$P(k_1)$。

$$\underline{\boldsymbol{f}}_{i_n,Ry} = L_{Ry}^f(\underline{\boldsymbol{h}}_{i_n})$$

请注意，这种方法可以很容易地与最大内积搜索相结合。我们可以仅使用 $\underline{\boldsymbol{f}}_{i_n,V}$

来搜索可能的下一个项，并使用其他隐藏状态来调整这些可能的下一个项和输入项的 logits。由于输入项的数量i_n和k_3小于总物品数，因此计算开销应相对较小。

Multiple Input Hidden States (Mi)

在这个方法中，所有的新的隐藏状态 $\underline{\boldsymbol{f}}_{i_n,}$.

都是$\underline{\boldsymbol{h}}_{i_n}$ 的投影。$\underline{\boldsymbol{h}}_{i_n}$ 的有限维度会迫使 $\underline{\boldsymbol{f}}_{i_n,}$.

线性依赖于彼此，无法自由移动到物品嵌入空间。为了解决这个问题，@chang2022softmax 合并了多个输入隐藏状态$\underline{\boldsymbol{h}}_{i_n}$ 并将它们投影到一个新的隐藏状态 $\underline{\boldsymbol{q}}_{i_n}$

以扩大其维度。我们在 中将他们的方法与 Softmax-CPR 结合起来，通过在 中替换$\underline{\boldsymbol{h}}_{i_n}$ 为 $\underline{\boldsymbol{q}}_{i_n}$

Mixture of Softmax (MoS)

图中的一种可能的解决方案是在 尿布 处放置一个隐藏状态，在 苹果汁 处放置另一个隐藏状态来模拟其多模态分布。混合的softmax（MoS）被提出以实现这个目标 。MoS 和 softmax-CPR 都使用多个隐藏状态，但它们每个隐藏状态的作用不同。在 MoS 中，我们需要计算每个隐藏状态与所有商品嵌入之间的点积；而在 softmax-CPR 中，我们将商品集分为不同的分区，并且每个隐藏状态只确定分区中的每个项目的 logits/概率（例如，仅在上下文分区中输入项目）。与 softmax-CPR 相比，MoS 更费时并且没有明确地模型用户的重复行为。

RepeatNet

害模型的一般性。第三， RepeatNet 没有解决 softmax 瓶颈问题对于不在输入序列中的项。

Experiments

所有实验都在 RecBole 中进行，这是一个提供了各种推荐模型和数据集的库。我们从 RecBole 中选择了12个足够大且在过去的工作中广泛使用的数据集，以便使结果更具代表性，并且对超参数设置和随机种子不敏感。这些数据集来自不同的领域，并具有不同的大小。我们在中报告了他们的统计信息。

Dataset		Item Type	Dataset size (k)			Config	
			#User	#Item	# Inter	$\lvert h_{i_n} \rvert$	bsz
Amazon-2014 [26]	Beauty	Products	1210	249	2023	64	[64,128]
	Books		8026	2330	22507	32	32
	Video Games		827	50	1325	64	[64,128]
Movie Lens [11]	10m	Movies	70	11	10000	64	128
	1m		6	4	1000	64	[64,128]
Twitch-100k [29]		Videos	100	740	3052	48	32
Yelp-2018[2]		Stores	1326	175	5262	48	32
Bridge to Algebra (2008-2009) [34]		Exercises	3	1259	8918	48	32
Gowalla [8]		Locations	107	1281	6443	32	32
Steam [15]		Games	2568	32	7793	64	[64,128]
Tmall-buy [35]		Products	886	1144	9349	32	64
Yoochoose-clicks [2]		Products	9250	53	33004	64	128

Models and Baselines

我们通过修改SASRec 和GRU4Rec 的模型代码来实现以下softmax替代方案[+]。

Setup

我们使用NDCG\@10（归一化的折扣累积收益）、HR\@10（命中率）和MRR\@10（平均反向排名）作为评价指标。为接近现实世界，我们在测试性能时未进行负例采样，导致部分数据集得分较低。为概括每种方法性能，我们报告每个方法的几何均值。我们遵循RecBole的默认协议及设置，如GRU4Rec和SASRec使用相同输入和输出项嵌入。除SASRec使用较小的丢弃率外，其它默认参数[+]表现良好。总体而言，我们的性能改善对超参数不敏感。

为在有限计算资源下优化超参数，我们在验证集上使用网格搜索对Amazon Beauty, Games, MovieLens 1m和Steam数据集的NDCG\@10分数进行超参数调整。在网格搜索中，我们设定学习率5e-4至1e-3，批大小64至128。针对GRU4Rec，我们设定丢弃率0至0.5；对于SASRec，我们设定隐藏状态丢弃率0至0.1。其他8个较大数据集，学习率设为1e-3，丢弃率为0。所有其他8个较大数据集的其它超参数或搜索范围与GRU4Rec相同。

为适应GPU内存，我们调整隐藏状态大小和训练批次大小。具体请参考表格。使用网格搜索研究了学习率、Dropout及批大小的超参数敏感性。探究了隐藏状态大小的影响，尝试了$16, 32, 64, 128$的隐藏状态大小以及$64, 128$的批大小。优化了Softmax+Mi基准在各个小数据集上的表现。所有实验均在Nvidia TESLA M40上进行，通过计算所有项的概率而不使用任何最近邻搜索来度量时间。RecBole中的模型代码未优化运行时间，因此时间比较更有意义。未报告RepeatNet的时间，以免产生不公平的比较。

Results

知乎

	Softmax + Mi	1.18	2.20	3.23	5.77	3.79	7.48	15.80	26.69	16.67	29.06	8.08	15.03	1.67	3.36
	Softmax + C	1.41	2.41	3.83	6.46	4.41	8.27	19.12	31.13	20.70	34.19	9.14	16.39	1.94	3.82
	Softmax + CP	**1.45**	**2.52**	3.94	6.71	4.54	8.59	18.62	30.51	20.69	**34.67**	**9.45**	**16.93**	2.04	3.91
SASRec	Softmax + CPR:100	1.38	2.42	4.15	6.89	4.57	**8.69**	**19.32**	**31.32**	20.79	34.25	9.11	15.94	**2.22**	4.24
	Softmax + CPR:100 + Mi	1.37	2.41	**4.30**	**7.20**	**4.47**	8.40	18.90	30.73	**20.82**	34.49	9.06	15.91	2.21	4.24
	Softmax + CPR:20,100,500 + Mi	1.39	2.43	3.93	6.60	4.46	8.58	19.19	30.93	20.48	33.61	8.58	14.88	2.20	**4.27**
	Mixture of Softmax (MoS)	1.19	2.24	3.24	5.75	3.74	7.35	15.88	26.82	17.05	29.83	8.17	15.19	1.69	3.42
	Softmax w/o Duplication [22]	1.34	2.42	3.73	6.27	4.42	8.35	18.35	30.19	20.06	33.81	9.01	16.13	1.85	3.64
	Softmax	1.43	2.67	3.09	5.70	4.45	8.64	14.19	24.17	16.05	28.03	8.36	15.55	1.68	3.42
	Softmax + Mi	1.47	2.69	3.30	5.92	4.58	8.79	14.58	25.04	16.55	28.94	8.03	14.98	1.76	3.52
	Softmax + C	1.59	2.88	3.97	6.66	4.95	9.36	17.78	29.24	20.01	32.86	9.25	16.50	2.02	3.92
	Softmax + CP	1.61	2.94	4.07	6.83	**5.10**	9.41	17.46	28.64	19.63	32.91	9.14	16.09	2.00	3.85
GRU4Rec	Softmax + CPR:100	1.78	**3.22**	4.28	7.06	5.05	**9.49**	17.78	29.01	20.35	33.73	9.04	15.82	2.27	4.35
	Softmax + CPR:100 + Mi	1.72	3.15	**4.42**	**7.23**	5.07	9.43	**18.09**	**29.43**	**21.00**	**34.52**	**9.32**	**16.20**	**2.37**	**4.51**
	Softmax + CPR:20,100,500 + Mi	**1.73**	3.11	4.37	7.14	5.02	9.33	17.87	29.09	20.44	33.63	8.80	15.20	2.31	4.39
	Mixture of Softmax (MoS)	1.46	2.73	3.15	5.76	4.06	8.00	14.40	24.50	16.14	28.06	7.90	14.51	1.72	3.50
	Softmax w/o Duplication [22]	1.60	2.91	3.71	6.26	4.83	9.09	16.85	27.68	18.54	31.72	8.94	16.03	1.94	3.80
RepeatNet	-	1.75	2.88	3.94	6.36	4.47	8.36	18.09	29.20	18.71	31.08	8.52	14.91	2.02	3.88

结果分别在表格中呈现。可以发现上下文分区（**Softmax + C**）相较于**Softmax**有显著改进。添加指针网络（P），重新排序分区（R）和多个输入隐藏状态（Mi）后，**Softmax + CPR:100 + Mi**在中实现了最佳的整体性能。与语言模型相比，多个重排分区（**Softmax + CPR:20,100,500 + Mi**），在推荐模型中并没有带来更好的性能。GRU4Rec的改善稍微大于SASRec的改善。这表明softmax-CPR和神经编码器中的自我注意之间的收益存在一定的重叠。需要注意的是SASRec和GRU4Rec的性能并不是可以直接进行比较的，因为我们在超参数调整方面并不细致。重复网络可以显著提高无重复项数据集的性能，尽管它最初是为了处理有重复项的数据集设计的。

		Bridge to Algebra		Gowalla		Steam		Tmall-buy		Yoochoose-clicks	
		NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR
	Softmax	85.66	90.42	29.28	40.39	15.67	20.28	22.44	26.60	35.74	57.28
	Softmax + Mi	85.68	89.72	29.72	40.72	15.77	20.47	22.64	26.80	36.62	57.93
	Softmax + C	86.25	91.15	32.23	45.15	16.32	21.13	25.29	30.36	37.26	58.93
	Softmax + CP	85.60	89.75	32.88	45.68	16.30	21.05	25.58	30.50	37.43	59.02
SASRec	Softmax + CPR:100	87.40	91.09	33.03	46.17	16.43	21.31	25.73	**30.70**	37.79	59.15
	Softmax + CPR:100 + Mi	88.19	**92.19**	33.41	46.29	**16.48**	**21.39**	**25.74**	30.58	39.03	**59.69**
	Softmax + CPR:20,100,500 + Mi	**88.81**	92.07	**33.92**	**46.64**	16.34	21.15	25.58	30.22	**39.26**	59.68
	Mixture of Softmax (MoS)	84.77	89.78	29.74	40.87	15.90	20.49	23.07	27.28	35.59	57.07
	Softmax w/o Duplication [22]	80.13	82.89	3.92	7.00	4.89	9.15	4.29	6.28	17.00	27.84
	Softmax	85.10	89.23	28.37	39.48	15.35	19.88	22.06	26.42	36.19	56.97
	Softmax + Mi	84.68	89.01	27.99	39.06	15.69	20.26	21.76	26.05	36.39	57.15
	Softmax + C	85.86	89.75	32.23	45.18	16.29	21.04	25.18	30.25	37.46	58.54
	Softmax + CP	86.24	91.06	32.48	45.43	16.32	21.06	25.45	30.36	37.90	58.76
GRU4Rec	Softmax + CPR:100	88.56	**92.35**	33.01	46.08	16.36	21.15	**25.77**	30.34	38.35	59.15
	Softmax + CPR:100 + Mi	88.81	92.19	**33.22**	**46.09**	**16.49**	**21.35**	25.54	30.01	**38.72**	**59.42**
	Softmax + CPR:20,100,500 + Mi	**89.46**	92.29	33.18	45.93	16.41	21.19	25.72	**30.43**	38.54	59.20
	Mixture of Softmax (MoS)	86.11	90.30	27.91	38.60	15.89	20.41	21.50	25.75	36.39	56.97
	Softmax w/o Duplication [22]	79.06	81.67	3.93	7.05	4.65	8.72	4.24	6.32	16.80	27.44
RepeatNet	-	77.44	81.70	33.83	45.88	16.28	20.50	25.67	30.11	38.09	58.55

Table 3: The test performance (%) in 5 datasets with duplicated items. The notations are the same as Table 2.

与Softmax+C相比，重复网络的主要改进来源于单独计算重复项的概率，而不是它的自注意力机制*或额外参数。Softmax+C可以达到与之相同程度的改进，但模型大小只有Softmax+C的一半，Softmax+C:100+CPR:100+Mi还可以进一步扩大这种改进，通过更好地克服softmax瓶颈。Softmax+C:100+CPR:100+Mi引入的额外参数可以忽略不计。在特定的超参数值下，所有方法都对数据集大小和学习率非常敏感。在GRU4Rec中，较低的数据集如Amazon Video Games的性能会在没有dropout的情况下恶化。隐藏状态大小小于**64**时，性能会开始下降。在RepeatNet中，项目嵌入大小是隐藏状态大小的两倍。

发布于 2024-03-28 16:25 · IP 属地北京

深度学习（Deep Learning） softmax 机器学习

理性发言，友善互动

1 条评论 默认 最新

hallucination

好硬核

04-05 · 湖南

回复 喜欢

推荐阅读