1. 论文介绍

1.1 背景介绍

推荐系统中的冷启动问题是指在系统刚开始运行或者新物品或新用户加入时，由于缺乏足够的用户-物品交互数据，导致无法进行有效的个性化推荐。这是一个普遍存在的问题，因为推荐系统需要足够的数据才能准确地预测用户偏好并提供有用的建议。

解决冷启动问题的方法主要包括利用辅助信息来提高推荐系统的性能，例如基于内容的推荐系统和跨领域推荐系统。此外，还有一些方法使用元学习或元优化来预测用户偏好。

1.2 论文方法

《MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation》

这篇论文提出了一种名为Memory-Augmented Meta-Optimization (MAMO)的方法，用于解决推荐系统中的冷启动问题。该方法使用元优化来预测用户偏好，即使是对于新用户或新物品也能够进行有效的推荐。

本文提出的方法（MAMO）具有以下优势：

- MAMO使用元学习和元优化来预测用户偏好，可以在只有少量交互数据的情况下进行个性化推荐，从而有效地解决了冷启动问题；
- MAMO使用记忆增强机制来捕捉用户的历史偏好，并将其用于初始化本地参数，从而提高了模型的泛化能力和鲁棒性；
- MAMO在两个广泛使用的数据集上进行了大量实验，并展示了相对于现有技术的优越性能。实验结果表明，MAMO可以显著提高推荐系统的准确性和效率；

1.3 数据集介绍

本文使用的数据集是MovieLens。MovieLens数据集是由明尼苏达大学的GroupLens实验室收集和维护的一组电影评分数据集。这些数据集包含了来自不同用户对多部电影的评分，时间跨度大。MovieLens数据集被广泛用于推荐系统、协同过滤和个性化推荐等领域的研究。本文使用的数据集规模是1M，包含100万个评分，来自6040名用户对3952部电影的评分数据。

1.4 pipeline

本作业将基于论文官方代码仓库实现，将pytorch版本的网络模型转换成mindspore版本的模型。

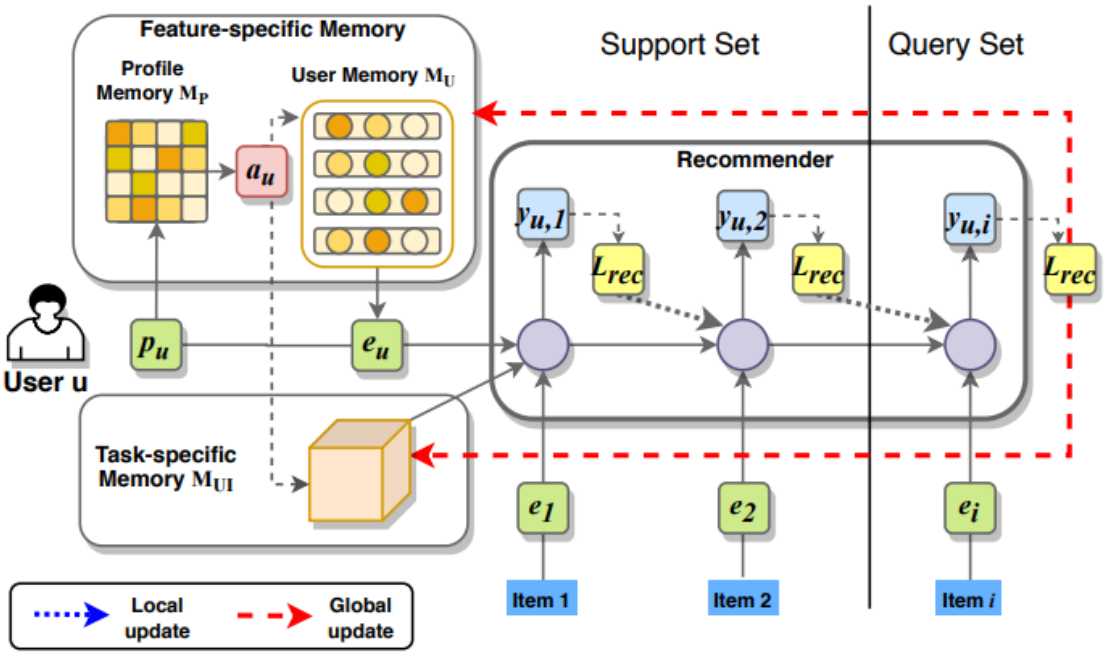

Figure 1: The training phase of MAMO

2. pytorch实现版本

2.1 准备工作

创建环境：

```
conda create -n mamo python=3.7
```

安装依赖包：

```
# Name                    Version                   Build  Channel

ca-certificates           2022.12.7            h5b45459_0
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
certifi                   2022.12.7                 pypi_0    pypi

charset-normalizer        3.1.0                     pypi_0    pypi

colorama                  0.4.6                     pypi_0    pypi
idna                      3.4                       pypi_0    pypi
libsqlite                 3.40.0               hcfcfb64_1
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
numpy                     1.21.6                    pypi_0    pypi
openssl                   3.1.0                hcfcfb64_2
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
pandas                    1.3.5                     pypi_0    pypi
pillow                    9.5.0                     pypi_0    pypi
pip                       23.1.1             pyhd8ed1ab_0
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
python                    3.7.12          h900ac77_100_cpython
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
python-dateutil           2.8.2                     pypi_0    pypi
pytz                      2023.3                    pypi_0    pypi
requests                  2.28.2                    pypi_0    pypi
setuptools                67.7.2             pyhd8ed1ab_0
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
six                       1.16.0                    pypi_0    pypi
sqlite                    3.40.0               hcfcfb64_1
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
torch                     1.12.1+cu113              pypi_0    pypi
torchaudio                0.12.1+cu113              pypi_0    pypi
torchvision               0.13.1+cu113              pypi_0    pypi
tqdm                      4.65.0                    pypi_0    pypi
typing-extensions         4.5.0                     pypi_0    pypi
ucrt                      10.0.22621.0         h57928b3_0
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
urllib3                   1.26.15                   pypi_0    pypi
vc                        14.3                h3d8a991_11
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
vs2015_runtime            14.34.31931         h4c5c07a_11
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
wheel                     0.40.0             pyhd8ed1ab_0
http://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
```

数据集下载链接：https://files.grouplens.org/datasets/movielens/ml-1m.zip，下载后的数据集放在`data_raw`文件夹下。

创建`data_processed`文件夹，运行`prepareDataset.py`脚本得到处理后的数据。

该项目的文件目录树如下：

```
D:.
|  configs.py
|  LICENSE
|  mamoRec.py
|  models.py
|  prepareDataset.py
|  README.md
|  tree.txt
|  utils.py
|
├─.idea
|  |  .gitignore
|  |  Code-for-MAMO.iml
|  |  misc.xml
|  |  modules.xml
|  |  vcs.xml
|  |  workspace.xml
|  |
|  └─inspectionProfiles
|          profiles_settings.xml
|          Project_Default.xml
|
├─data_processed
|  └─movielens
|      |  item_dict.p
|      |  item_state_ids.p
|      |  ratings_sorted.p
|      |  user_dict.p
|      |  user_state_ids.p
|      |
|      └─raw
├─data_raw
|  ├─book_crossing
|  |      Download the dataset into this folder.txt
|  |
|  └─ml-1m
|          List_director.txt
|          List_genre.txt
|          movies.dat
|          movies_extrainfos.dat
|          ratings.dat
|          README
|          users.dat
|
├─modules
|  |  info_embedding.py
|  |  input_loading.py
|  |  memories.py
|  |  rec_model.py
```

```
|
|
├─prepare_data
|   |  prepareList.py
|   |  prepareMovielens.py
```

2.2 运行代码

执行脚本`python mamoRec.py`输出结果如下：

```
start train
train finished
0 4461
1 4580
2 1009
3 5429
...
...
1204 4481
1205 2795
1206 2207
test finished
start train
train finished
0 4461
1 4580
2 1009
3 5429
...
...
1204 4481
1205 2795
1206 2207
test finished
start train
train finished
0 4461
1 4580
2 1009
3 5429
...
...
1204 4481
1205 2795
1206 2207
test finished

Process finished with exit code 0
```

3. mindspore实现版本

代码仓库：https://github.com/XiShuFan/MAMO_mindspore

3.1 mindspore框架介绍

MindSpore是华为推出的一款人工智能计算框架，主要用于开发AI应用和模型。它的特点如下:

- 框架设计：MindSpore采用静态计算图设计，PyTorch采用动态计算图设计。静态计算图在模型编译时确定计算过程，动态计算图在运行时确定计算过程。静态计算图通常更高效，动态计算图更灵活；
- 设备支持：MindSpore在云端和边缘端都有较好的支持，可以在Ascend、CPU、GPU等硬件上运行；
- 自动微分：MindSpore提供自动微分功能，可以自动求导数，简化模型训练过程；
- 运算符和层：MindSpore提供丰富的神经网络层和运算符，覆盖CNN、RNN、GAN等多种模型；
- 训练和部署：MindSpore提供方便的模型训练和部署功能，支持ONNX、CANN和MindSpore格式的模型导出，可以部署到Ascend、GPU、CPU等硬件；

3.2 环境准备

使用华为HECS(云耀云服务器)，操作系统Ubuntu 22.04。

安装anaconda环境：

```
wget https://mirrors.bfsu.edu.cn/anaconda/archive/Anaconda3-2022.10-Linux-x86_64.sh --no-check-certificate
bash Anaconda3-2022.10-Linux-x86_64.sh
```

创建虚拟环境并且切换到环境：

```
conda create -n mamo python=3.7
conda activate mamo
```

克隆已经实现好的mindspore版本mamo代码：

```
git clone https://github.com/XiShuFan/MAMO_mindspore.git
```

下载依赖包：

```
cd MAMO_mindspore
pip install mindspore==2.0.0a0
```

3.3 模型迁移

将Pytorch的API替换成mindspore的API，官方给出了文档说明。

另外mindspore还提供了MindConverter工具，方便从pytorch迁移模型。

下面是我在模型迁移过程替换的API以及Class：

pytorch API / Class	mindspore API/Class	说明	两者差异
torch.from_numpy	mindspore.tensor.from_numpy	从numpy得到tensor	无
torch.tensor.to	mindspore.tensor.to_device	将tensor传入指定的设备	无
torch.utils.data.Dataset	mindspore.dataset.GeneratorDataset	数据集类	PyTorch：自定义数据集的抽象类，自定义数据子类可以通过调用`__len__()`和`__getitem__()`这两个方法继承这个抽象类。 MindSpore：通过每次调用Python层自定义的Dataset以生成数据集。
torch.zeros_like	mindspore.ops.ZerosLike	获得指定shape的全零元素tensor	无
torch.nn.Sigmoid	mindspore.nn.Sigmoid	激活函数	无
torch.nn.Tanh	mindspore.nn.Tanh	激活函数	无
torch.nn.ReLU	mindspore.nn.ReLU	激活函数	无
torch.nn.Softmax	mindspore.nn.Softmax	归一化	无
torch.nn.LeakyReLU	mindspore.nn.LeakyReLU	激活函数	无
torch.nn.Sequential	mindspore.nn.SequentialCell	整合多个网络模块	无
torch.argmax	mindspore.ops.argmax	返回最大值下标	PyTorch：沿着给定的维度返回Tensor最大值所在的下标，返回值类型为torch.int64。 MindSpore：MindSpore此API实现功能与PyTorch基本一致，返回值类型为int32.
torch.abs	mindspore.ops.abs	计算tensor绝对值	PyTorch：计算输入的绝对值。 MindSpore：MindSpore此API实现功能与PyTorch一致，仅参数名不同。
torch.mean	mindspore.ops.ReduceMean	计算均值	无
torch.optim.Adam	mindspore.nn.Adam	优化器	无

pytorch API / Class	mindspore API/Class	说明	两者差异
torch.nn.CrossEntropyLoss	mindspore.nn.CrossEntropyLoss	损失函数	PyTorch：计算预测值和目标值之间的交叉熵损失。 MindSpore：MindSpore此API实现功能与PyTorch基本一致，而且目标值支持两种不同的数据形式：标量和概率。
torch.nn.Module	mindspore.nn.Cell	神经网络的基本构成单位	
torch.nn.Linear	mindspore.nn.Dense	全连接层	PyTorch：全连接层，实现矩阵相乘的运算。 MindSpore：MindSpore此API实现功能与PyTorch基本一致，而且可以在全连接层后添加激活函数。
torch.cat	mindspore.ops.concat	tensor按照指定维度拼接	无
torch.randn	mindspore.ops.StandardNormal	获得正态分布数据的tensor	无
torch.mm	mindspore.ops.MatMul	矩阵乘法	无
torch.sqrt	mindspore.ops.Sqrt	开根号	无
torch.sum	mindspore.ops.ReduceSum	求和	无
torch.Tensor.mul	mindspore.ops.Mul	相乘	无
torch.div	mindspore.ops.div	除法	无
torch.nn.Embedding	mindspore.nn.Embedding		PyTorch：支持使用`_weight`属性初始化embedding，并且可以通过`weight`变量获取当前embedding的权重。 MindSpore：支持使用`embedding_table`属性初始化embedding，并且可以通过`embedding_table`属性获取当前embedding的权重。除此之外，当`use_one_hot`为True时，你可以得到具有one-hot特征的embedding。
torch.tensor.repeat	mindspore.ops.tile	对tensor进行重复叠加	无
torch.tensor.view	mindspore.ops.Reshape	重新排列tensor的维度	无
Adam.zero_grad	Adam.clear_grad	清除梯度	无

pytorch API / Class	mindspore API/Class	说明	两者差异

3.4 详细迁移代码

数据集实现

```
import mindspore.dataset as ds

class UserDataLoader:
    def __init__(self, x1, x2, y, y0, transform=None):
        self.x1 = x1
        self.x2 = x2
        self.y = y
        self.y0 = y0
        self.transform = transform

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
        if isinstance(idx, mindspore.tensor):
            idx = idx.asnumpy()

        user_info = self.x1[idx]
        item_info = self.x2[idx]
        ratings = self.y[idx]
        cold_labels = self.y0[idx]

        return user_info, item_info, ratings, cold_labels

# 数据加载方法
dataset_generator = UserDataLoader()
dataset = ds.GeneratorDataset(dataset_generator, ["user_info", "item_info",
"ratings", "cold_labels"], shuffle=False)

for data in dataset.create_dict_iterator():
    print(data["user_info"], data["item_info"], data["ratings"],
data["cold_labels"])
```

网络实现

```
import mindspore

# 继承父类 Cell
class BASEModel(mindspore.nn.Cell):
    def __init__(self, input1_module, input2_module, embedding1_module,
embedding2_module, rec_module):
        super(BASEModel, self).__init__(auto_prefix=False)
```

```
        self.input_user_loading = input1_module
        self.input_item_loading = input2_module
        self.user_embedding = embedding1_module
        self.item_embedding = embedding2_module
        self.rec_model = rec_module

    # 对应于forward
    def construct(self, x1, x2):
        pu, pi = self.input_user_loading(x1), self.input_item_loading(x2)
        eu, ei = self.user_embedding(pu), self.item_embedding(pi)
        rec_value = self.rec_model(eu, ei)
        return rec_value

    def get_weights(self):
        u_emb_params = get_params(self.user_embedding.parameters())
        i_emb_params = get_params(self.item_embedding.parameters())
        rec_params = get_params(self.rec_model.parameters())
        return u_emb_params, i_emb_params, rec_params

    def get_zero_weights(self):
        zeros_like_u_emb_params =
get_zeros_like_params(self.user_embedding.parameters())
        zeros_like_i_emb_params =
get_zeros_like_params(self.item_embedding.parameters())
        zeros_like_rec_params =
get_zeros_like_params(self.rec_model.parameters())
        return zeros_like_u_emb_params, zeros_like_i_emb_params,
zeros_like_rec_params

    def init_weights(self, u_emb_para, i_emb_para, rec_para):
        init_params(self.user_embedding.parameters(), u_emb_para)
        init_params(self.item_embedding.parameters(), i_emb_para)
        init_params(self.rec_model.parameters(), rec_para)

    def get_grad(self):
        u_grad = get_grad(self.user_embedding.parameters())
        i_grad = get_grad(self.item_embedding.parameters())
        r_grad = get_grad(self.rec_model.parameters())
        return u_grad, i_grad, r_grad

    def init_u_mem_weights(self, u_emb_para, mu, tao, i_emb_para, rec_para):
        init_u_mem_params(self.user_embedding.parameters(), u_emb_para, mu, tao)
        init_params(self.item_embedding.parameters(), i_emb_para)
        init_params(self.rec_model.parameters(), rec_para)

    def init_ui_mem_weights(self, att_values, task_mem):
        # init the weights only for the mem layer
        u_mui = task_mem.read_head(att_values)
        init_ui_mem_params(self.rec_model.mem_layer.parameters(), u_mui)

    def get_ui_mem_weights(self):
        return get_params(self.rec_model.mem_layer.parameters())
```

3.5 训练结果

下面是将pytorch模型转为mindspore模型后的训练测试结果：

```
CPU
Model parameters:
Param name: input_user_loading.embedding_gender.embedding_table, shape: (2, 100)
Param name: input_user_loading.embedding_age.embedding_table, shape: (7, 100)
Param name: input_user_loading.embedding_occupation.embedding_table, shape: (21,
100)
Param name: input_item_loading.emb_rate.embedding_table, shape: (6, 100)
Param name: input_item_loading.emb_genre.weight, shape: (100, 25)
Param name: input_item_loading.emb_genre.bias, shape: (100,)
Param name: input_item_loading.emb_director.weight, shape: (100, 2186)
Param name: input_item_loading.emb_director.bias, shape: (100,)
Param name: input_item_loading.emb_year.embedding_table, shape: (81, 100)
Param name: user_embedding.fc.0.weight, shape: (150, 300)
Param name: user_embedding.fc.0.bias, shape: (150,)
Param name: user_embedding.final_layer.0.weight, shape: (100, 150)
Param name: user_embedding.final_layer.0.bias, shape: (100,)
Param name: item_embedding.fc.0.weight, shape: (200, 400)
Param name: item_embedding.fc.0.bias, shape: (200,)
Param name: item_embedding.final_layer.0.weight, shape: (100, 200)
Param name: item_embedding.final_layer.0.bias, shape: (100,)
Param name: rec_model.mem_layer.weight, shape: (200, 200)
Param name: rec_model.mem_layer.bias, shape: (200,)
Param name: rec_model.fc.0.weight, shape: (100, 200)
Param name: rec_model.fc.0.bias, shape: (100,)
Param name: rec_model.final_layer.0.weight, shape: (5, 100)
Param name: rec_model.final_layer.0.bias, shape: (5,)
4775
1462
4858
...
...
1991
5714
2774
start train
train finished
0 711
1 5728
2 1277
3 4060
...
...
1204 2362
1205 5416
1206 5934
test finished

Process finished with exit code 0
```

华为云服务器运行截图如下：