Assignment 3

齐子墨12111609

Q1

```
(dspractice) xiaomangguo@xiaomangguodeMacBook-Air 数据科学实践 % conda list pytorch
# packages in environment at /Users/xiaomangguo/miniforge3/envs/dspractice:
#
# Name                    Version                   Build  Channel
pytorch                   2.3.0                   py3.11_0    pytorch
```

Q2

- Refer to a rough preliminary implement on github, and implement a more integrated model which can assign the network according to different mode, say `GMF`, `MLP`, `NMF`. Also implements two metrics and the test-negative data processing. Below is the main part for models.

```
class NeuralCollaborativeFiltering(torch.nn.Module):
    def __init__(self, field_dims, user_field_idx, item_field_idx, embed_d
        super().__init__()
        #  mlp_dims: 中间层的维数，比如(16, 16)表示有两个中间层，每个中间层的维数是
        self.mode = mode
        self.user_field_idx = user_field_idx # 0
        self.item_field_idx = item_field_idx # 1
        self.embedding = FeaturesEmbedding(field_dims, embed_dim) # 索引到e
        self.embed_output_dim = len(field_dims) * embed_dim # 对于mlp是先做
        if self.mode == "NCF":
            self.mlp = MultiLayerPerceptron(self.embed_output_dim, mlp_dim
            self.fc = torch.nn.Linear(mlp_dims[-1] + embed_dim, 1)
        elif self.mode == "MLP":
            self.mlp = MultiLayerPerceptron(self.embed_output_dim, mlp_dim
        elif self.mode == "GMF":
            self.fc = torch.nn.Linear(embed_dim, 1)

    def forward(self, x):
        index = x
        x = self.embedding(x) # 获得embedding，维度为embed_output_dim
        user_x = x[:, self.user_field_idx].squeeze(1) # 按照0取出user的embe
        item_x = x[:, self.item_field_idx].squeeze(1) # 按照1取出item的embe
        gmf = user_x * item_x # generalized matrix factorization
        if self.mode == "NCF":
                x = self.mlp(x.view(-1, self.embed_output_dim)) # 将embedd
            x = torch.cat([gmf, x], dim=1) # 将gmf和mlp的结果拼接
            x = self.fc(x).squeeze(1)
        elif self.mode == "MLP":
                x = self.mlp(x.view(-1, self.embed_output_dim)) # 将embedd
```

```
            x = x.squeeze(1) # 最后一层直接输出
        elif self.mode == "GMF":
            x = self.fc(gmf).squeeze(1)
        return torch.sigmoid(x), index
```

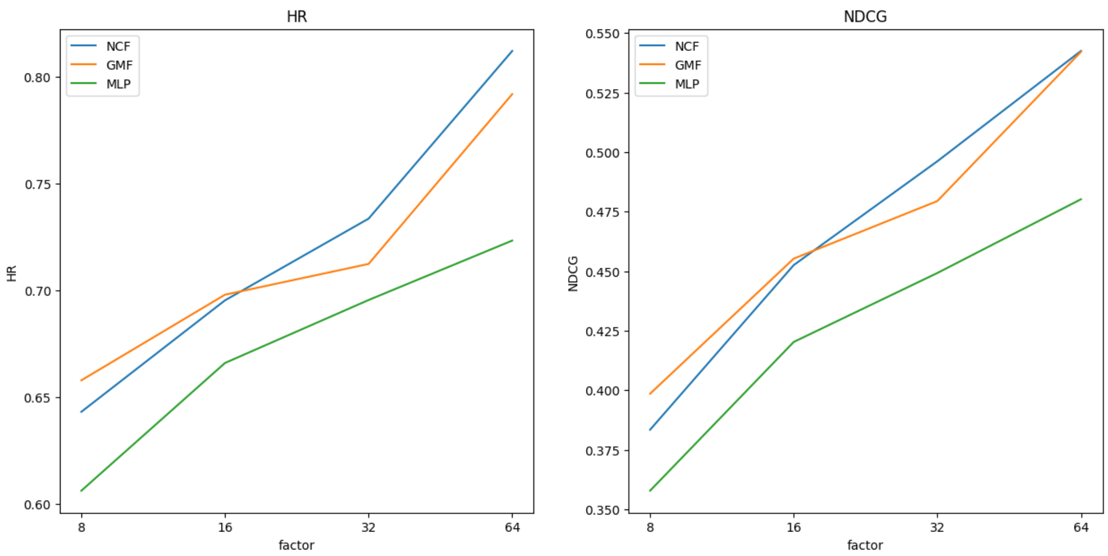
- See the attached code for details.

Q3

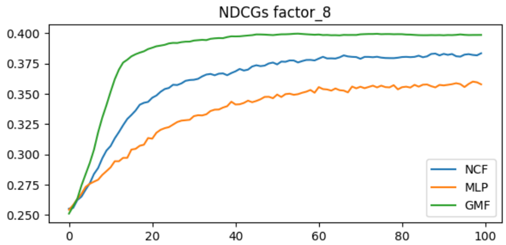
- `batch size` = 512
- `lr` = 0.001
- `predictive factors` = [8, 16, 32, 64]
- `neural CF layers:` 32 → 16 → 8 (default setting)
- `embedding size` = 16

Q4

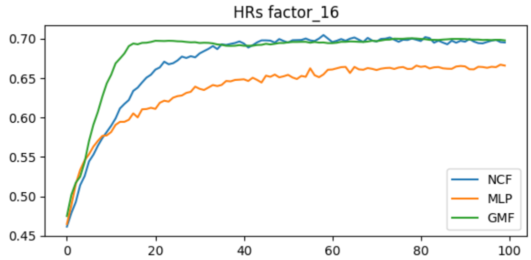
- In the paper, the **factor** means the dimension of latent representation.
    - The **larger factor** means larger capability of representation, and **the results should be better**.
    - Generally speaking, **NCF outperforms other two**, with larger factor specially. However, for small factors, GMF is enough. `It shows a little contradiction with origin paper, which states the superiority of NCF in all settings.`

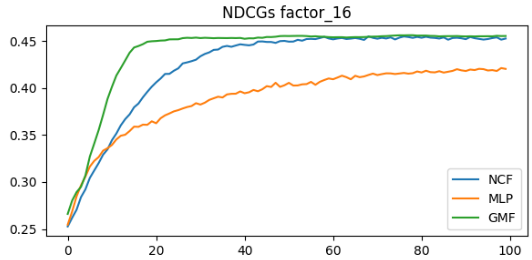


- For more meticulous observation, I plot the epoch-level change of metrics. With different **factor setting**, the result seems different: I trained the three models with 100 epoch, test the metrics with **HR@10** and **NDCG@10**
    - Factor 8, with hidden dimension **(32, 16, 8) in MLP layers**

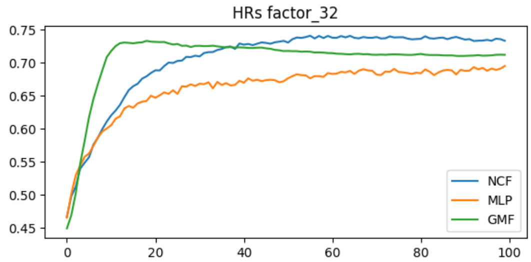


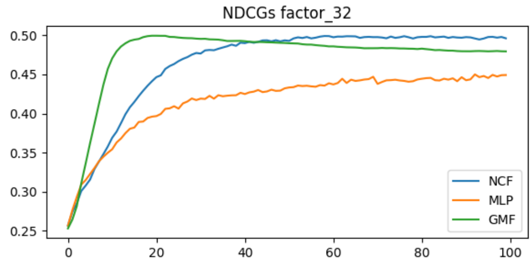
- Factor 16, with hidden dimension **(64, 32, 16) in MLP layers**



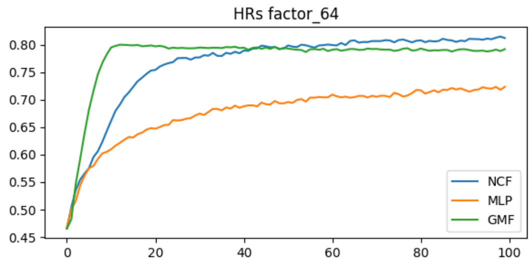


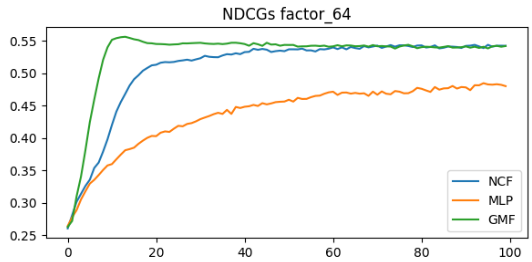
- Factor 16, with hidden dimension **(128, 64, 32) in MLP layers**





- Factor 16, with hidden dimension **(128, 64, 64) in MLP layers**





- There are two main observations:
  - With small quantity of parameters, GMF is good enough
  - Quicker converging for GMF(because of fewer training parameters.
  - MLP performs bad `(contradict with origin papers)`, which shows that simple **GMF guarantees a basic effectiveness and more parameters in MLP improves it a little.**

Q5: Ablation experiment, MLP with different layers

- Show the result for 32-dimension embedding and 32-factor output experiment, training for 30 epochs.

- The hidden dimensions are **[ ], [32], [64, 32], [64, 64, 32], [64, 128, 64, 32]** for MLP-0 → MLP-4
- **HR@10**

	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
Factor-32	0.44	0.58	0.62	**0.63**	0.62

- **NDCG@10**

	MLP-0	MLP-1	MLP-2	MLP-3	MLP-4
Factor-32	0.24	0.33	0.36	**0.37**	0.36

- Besides, I have tried other factor dimension and `only factor-32 shows deeper network helps` and `increasing the hidden dimension does not help`. From my understanding, that's because
  - The training data is not suitable for deep training(considering the normal initialization for embeddings but not contains specialty)
  - More hidden parameters leads to overfitting.
  - One may tries ResNet for deeper network.