–FAN LI & TINA LIU

PREDICTING IMPLICIT RATINGS

NEGATIVE SAMPLING

Users: Give each user weight based on their rating frequency

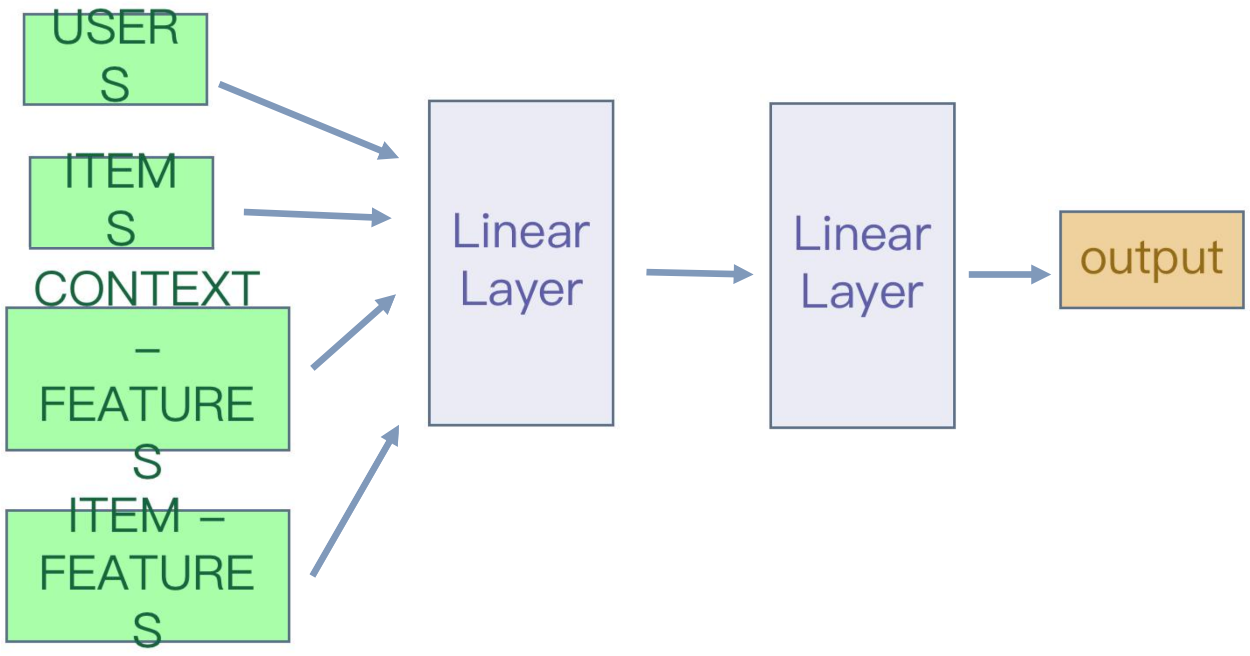
Items: Give each item weight based on their inverse popularity (Method: softmax the inverse)

Cold–Start Problem:

1. 'Cold user' we gave a fixed amount of negative samples

2. 'Cold item' we gave the same weight as itmes who have frequency of 1

NEURAL NETWORK




```
class NeuralNet(nn.Module):
    def __init__(self, num_users, num_items,num_item_fea, num_context_fea,emb_size=100, n_hidden=10):
        super(CollabFNet, self).__init__()
        self.user_emb = nn.Embedding(num_users, emb_size)
        self.item_emb = nn.Embedding(num_items, emb_size)
        self.item_fea_emb = nn.Embedding(num_item_fea, emb_size)
        self.context_fea_emb = nn.Embedding(num_context_fea, emb_size)

        self.lin1 = nn.Linear(emb_size*4, n_hidden)
        self.lin2 = nn.Linear(n_hidden, 1)
        self.drop1 = nn.Dropout(0.3)
    def forward(self, u, v,a,b):

        U = self.user_emb(u)
        V = self.item_emb(v)
        A = self.item_fea_emb(a)
        B = self.context_fea_emb(b)

        x = F.relu(torch.cat([U, V,A,B], dim=1))
        x = self.drop1(x)
        x = F.relu(self.lin1(x))
        x = self.lin2(x)
        return x
```

MATRIX FACTORIZATION

10	-1	8	10	9	4
8	9	10	-1	-1	8
10	5	4	9	-1	-1
9	10	-1	-1	-1	3
6	-1	-1	-1	8	10

```python
class MF(nn.Module):
    def __init__(self, num_users, num_items, emb_size=100):
        super(MF, self).__init__()
        self.user_emb = nn.Embedding(num_users, emb_size)  # .cuda()
        self.item_emb = nn.Embedding(num_items, emb_size)  # .cuda()
        #self.item_feature_emb = nn.Embedding(num_item_feature, emb_size)
        self.user_bias = nn.Embedding(num_users, 1)
        self.item_bias = nn.Embedding(num_items, 1)

        # initlializing weights
        self.user_emb.weight.data.uniform_(0, 0.05)  # .cuda()
        self.item_emb.weight.data.uniform_(0, 0.05)  # .cuda()
        #self.item_feature_emb.weight.data.uniform_(0, 0.01)
        self.user_bias.weight.data.uniform_(-0.01, 0.01)
        self.item_bias.weight.data.uniform_(-0.01, 0.01)

    def forward(self, u, v):
        U = self.user_emb(u)  # .cuda()
        V = self.item_emb(v)  # .cuda()
        #T = self.item_feature_emb(t)

        b = self.user_bias(u).squeeze()
        c = self.item_bias(v).squeeze()
        return (U*V).sum(1)+b+c
```

HYPERPARAMETER SEARCH

Learning Rate	[0.1, 0.01, 0.001]
Embedding Sizem	[1, 10, 100, 1000]
Epoch	[5, 10, 15, 20]
Weight Decay	[0, 0.01, 0.001]

- Note: During the fine tuning session, parameters were adjusted accordingly.

FINE TUNE

Fine tuning was based on train loss and kaggle score:

NUM OF NEGATIVE SAMPLING FOR COLD USER: 250

- train loss: 0.186 score: 0.44632 Improvement: NO
- train loss: 0.199 score: 0.45339 Improvement: NO
- train loss: 0.171 score: 0.44005 Improvement: YES
- train loss: 0.125 score: 0.43416 Improvement: YES

NUM OF NEGATIVE SAMPLING FOR COLD USER: 500

- train loss: 0.183 score: 0.47949 Improvement: NO
- train loss: 0.205 score: 0.50414 Improvement: NO
- train loss: 0.124 score: 0.43281 Improvement: YES
- train loss: 0.065 score: 0.44438 Improvement: NO
- train loss: 0.098 score: 0.42703 Improvement: YES
- train loss: 0.075 score: 0.43514 Improvement: NO
- train loss: 0.086 score: 0.42929 Improvement: NO
- train loss: 0.093 score: 0.42714 Improvement: NO
- train loss: 0.096 score: 0.42774 Improvement: NO

EXPERIMENT RESULT

- Best Parameters:

	Embedding	Epochs	Learning rate	Weight decay
Round 1	100	20	0.1	1e–5
Round 2	100	16	0.01	0
Round 3	100	2	0.001	0

- Mean of Predicted Ratings:

Train	Test
0.1060	0.5075475

- Best Ratio of 1:0

Existed users	Cold users
~ 1:5	fixed 500 negative samples

LESSON LEARNED

- Negative Sampling – Cold start (both user and item) is a very challenging issue in real world redommandation system. Negative sampling cold starters is essential.
- Write things in Function or Class format – can be reused easily
- Understand the pro/cons of different ML models — use the most efficient and effective one
- Hyperparameter Tuning is Very Important — thoroughly search parameters

"GOOD DATA BEATS GOOD MODELS!"