Data Mining Final Project

Name: Yijun Zhou

Motivation

- Create 4 model to do recommendation on Netflix Dataset
- Compare the performance of each model based on explicit data and implicit data
- Models:
  - SVD+bias
  - Combination of GMF and MLP
  - Deepwalk + neural network
  - Node2vec + neural network
  - (weight_deepwalk+neural network)

Dataset

- Original netflix dataset has 100 million ratings
- Divide the training dataset and test dataset according to the ratio of 80 to 20
- Created explicit dataset and implicit dataset
    - Explicit dataset: {movie, user, rating_score}
    - Implicit dataset:
        - Train_implicit_dataset: {movie, user, rating_score = 1}
        - Test_implicit_dataset : {movie,user, rating_Score = 1/0}

	movie	user	rating	date
count	4985661	4985661	4985661	4985661
unique	22176	24960	11	2166
top	5317	16272	4	2005-01-19
freq	11499	5900	1635230	37005

Dataset

- Original netflix dataset has 100 million ratings
- Divide the training dataset and test dataset according to the ratio of 80 to 20
- Created explicit dataset and implicit dataset
    - Explicit dataset: {movie, user, rating_score}
    - Implicit dataset:
        - Train_implicit_dataset: {movie, user, rating_score = 1}
        - Test_implicit_dataset : {movie,user, rating_Score = 1/0}

	movie	user	rating	date
count	4985661	4985661	4985661	4985661
unique	22176	24960	11	2166
top	5317	16272	4	2005-01-19
freq	11499	5900	1635230	37005

use leave-one-out method and negative sampling method: for each user in test dataset, keep the latest movie and add 100 movies which are not in the user's watching list

Methods(SVD+bias)

Algorithm: SVD+bias

INPUT: train_df, test_df, dimensional k, steps, gamma, Lambda
Initialize: pu, qi, bu, bi
shuffle(train_df)
Definite pred_rating function
For step = 1 **to** steps **do**
    For all (user, movie, rating) ∈ train_df **do**
        Pred_rating =pred_rating(user,movie)
        Upgrade bu, bi, pu qi
    For all (user, movie, rating) ∈ test_df **do**
        Pred_rating = pred_rating(user, movie)
    Calculate metrics
Return metrics

Methods(GMF+MLP)

Algorithm: GMF+MLP

INPUT: train_df, test_df, dimensional k, steps, gamma, Lambda
Initialize: GMF_model, MLP_model
shuffle(train_df)
Define predict function
For step = 1 **to** steps **do**
    For all (user, movie, rating) ∈ train_df **do**
        Pred_rating =pred_rating(user,movie)
        Upgrade GMF_model, MLP_model
    For all (user, movie, rating) ∈ test_df **do**
        Pred_rating = pred_rating(user, movie)
    Calculate metrics
Return metrics

Predict(user, movie, GMF_model, MLP_model)
Define GMF_model.predict function
Define MLP_model.predict function
Create a layer to concatenate the result of two models
Calculate result
Return result

Methods(Deepwalk + Neural Network)

Algorithm: DeepWalk + Neural Network

```
LearnFeatures(Graph G=(V,E), Dimensions d, Walks per node r, Walk length l, Context size k)
Initialize walks to Empty
For iter = 1 to r do
    For all nodes u ∈ V do
        Walk = RandomWalk(G,u,l)
        Append walk to walks
Return Word2vec(walks)
```

```
RandomWalk(Graph G'=(V,E), Start node u, length l
Initialize walk to [u]
For walk_iter =1 to l do
    Curr = walk[-1]
    Vcurr = GetNeighbors(curr,G')
    S = random.sample(Vcurr)
    Append s to walk
Return walk
```

Methods(Deepwalk + Neural Network)

NeuralNetwork(representations, users, movies,k, batch_size)
Set model = {
Input layer with dimension ((2*k,1));
hidden layer with dimension ((128,48));
ReLU layer();
output layer with dimension ((48,1));}
Return model

Recommendation(model, train_df, test_df)
shuffle(train_df)
For step = 1 **to** steps **do**
    model(train_df)
    model(test_df)
    Calculate metrics
Return metrics

Methods(Node2vec + neural network)

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
  π = PreprocessModifiedWeights(G, p, q)
  $G' = (V, E, \pi)$
  Initialize $walks$ to Empty
  for $iter = 1$ **to** r **do**
    for all nodes $u \in V$ **do**
      $walk$ = node2vecWalk(G', u, l)
      Append $walk$ to $walks$
  f = StochasticGradientDescent(k, d, $walks$)
  return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
  Inititalize $walk$ to $[u]$
  for $walk\_iter = 1$ **to** l **do**
    $curr = walk[-1]$
    V_{curr} = GetNeighbors($curr$, G')
    s = AliasSample(V_{curr}, π)
    Append s to $walk$
  return $walk$

(Perozzi, B., Al-Rfou, R., & Skiena, S, 2014)

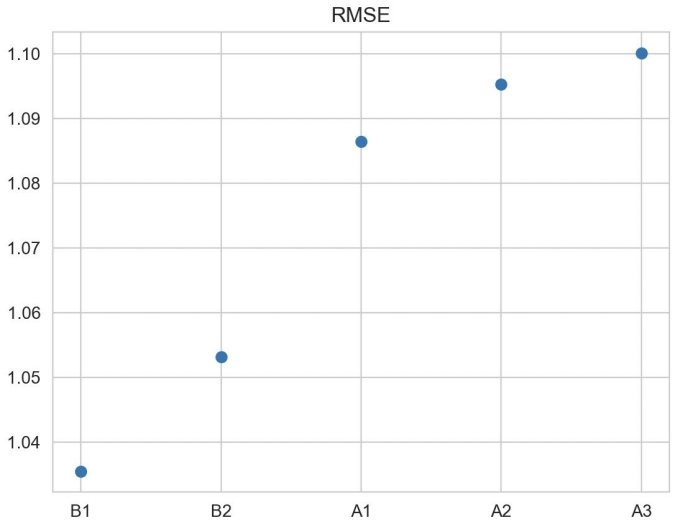
Methods(weight_deepword + Neural Network)

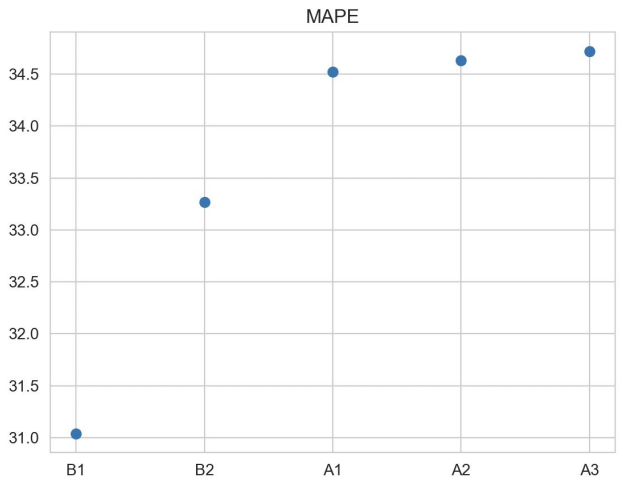
Algorithm: weightDeepwalk + Neural Network

LearnFeatures(Graph G=(V,E), Dimensions d, Walks per node r, Walk length l, Context size k)
Initialize walks to Empty
Calculate degree centralities of each node
For iter = 1 to r do
    For all nodes u ∈ V do
        Walk = WeightWalk(G,u,l,degree_centralities)
        Append walk to walks
Return Word2vec(walks)

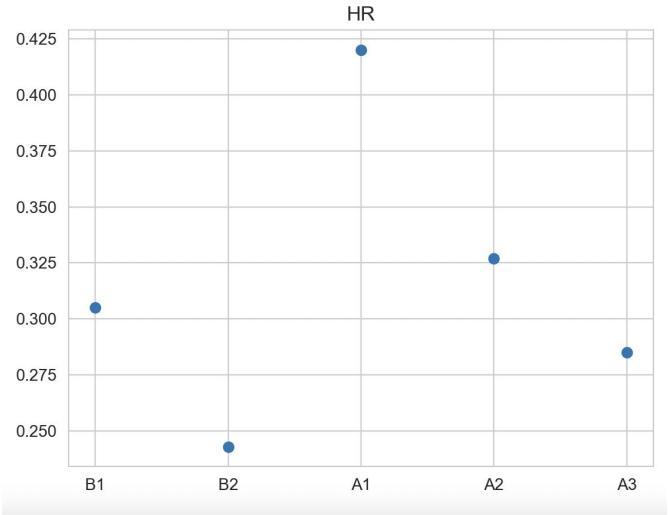
WeightWalk(Graph G'=(V,E), Start node u, length l, degree_centralities
Initialize walk to [u]
For walk_iter =1 **to** l **do**
    Curr = walk[-1]
    Vcurr = GetNeighbors(curr,G')
    S = node for node ∈ Vcurr with highest centrality
    Append s to walk
Return walk

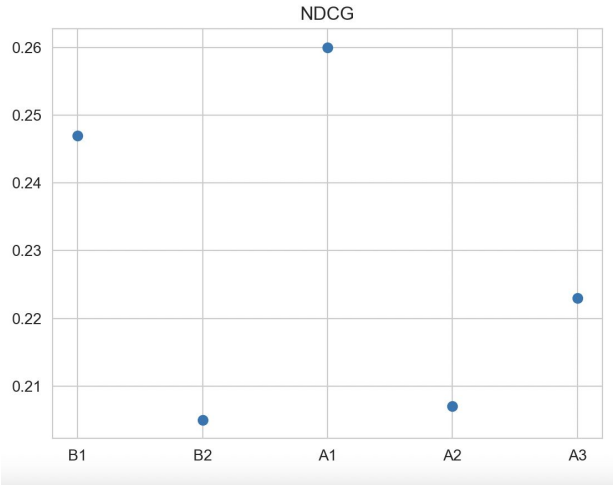
Results


RMSE
1.10
1.09
1.08
1.07
1.06
1.05
1.04
B1
B2
A1
A2
A3


MAPE
34.5
34.0
33.5
33.0
32.5
32.0
31.5
31.0
B1
B2
A1
A2
A3

Results


HR
0.425
0.400
0.375
0.350
0.325
0.300
0.275
0.250
B1
B2
A1
A2
A3


NDCG
0.26
0.25
0.24
0.23
0.22
0.21
B1
B2
A1
A2
A3

Problems

1. The performances of graph neural network models are not as well as baseline. (possible reason:
    a. Embedding isn’t good
        i. Distance of walk should be longer
        ii. Epoch to run each walk should be larger(one-time walk is stomastically)

Problems

1. The performances of graph neural network models are not as well as baseline. (possible reason:
    a. Embedding isn't good
        i. Distance of walk should be longer
        ii. Epoch to run each walk should be larger(one-time walk is stomastically)
    b. Structure of neural network
        i. Should try more different hyper-parameters
        ii. Also need to add epochs
    c. Reduce the dataset due to computing resources limited, might cause important information loss

Problems

2. The weight_deepwalk doesn’t work well

(possible reason: nodes tend to walk to hub nodes, limits the probability of personal recommendation)

Problems

2. The weight_deepwalk doesn’t work well

(possible reason: nodes tend to walk to hub nodes, limits the probability of personal recommendation)

Take-away

1. Implemented SVD+bias model, GMF+MLP model, deepwalk+neural network model, node2vec + neural network model and compared the results of these models
2. Added weight to deep walk based on degree centralities of nodes, and compared the result with deepwalk and node2vec models

Take-away

1. Implemented SVD+bias model, GMF+MLP model, deepwalk+neural network model, node2vec + neural network model and compared the results of these models
2. Added weight to deep walk based on degree centralities of nodes, and compared the result with deepwalk and node2vec models
3. Find some problems when implementing the models
4. Get familiar with pytorch

References

1. Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).
2. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182).
3. Mnih, A., & Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. Advances in neural information processing systems, 20.
4. https://github.com/pyy0715/Neural-Collaborative-Filtering

Thanks