Graph-based approaches for non-binary rating prediction

Anand Dhoot, Vivekkumar Patel
Stanford University

Introduction

Perform Collaborative Filtering in non-binary rating settings using graph-based approaches.

Our contributions:

- Adapt Random Walk for rating prediction
- Average over Random Walks as an improvement
- Predicting ratings using Shortest Paths on the bipartite graph

Traditional methods

Cosine similarity measure:

- Rating vectors for each user
- Define similarity:

$$Sim(u_p, u_q) = \frac{\mathbf{u}_p.\mathbf{u}_q}{||\mathbf{u}_p||.||\mathbf{u}_q||}$$

- Predict rating using:

$$\hat{r}_{pq} = \bar{r_p} + \alpha \sum_{j=1}^{n} sim(u_p, u_j)(r_{jq} - \bar{r}_j)$$

Matrix Factorization:

User/Item	A	B	C	D
1	1	2	0	0
2	0	4	0	0
3	0	4	4	5

Figure: The user-item matrix

- Feature vectors of size k for each user and item.
- Predicted rating $= \hat{r}_{ij} = p_i^T q_j$.
- Total error $= \Sigma(r_{ij} - \hat{r}_{ij})^2$
- Use gradient descent to decrease total error.

Challenges

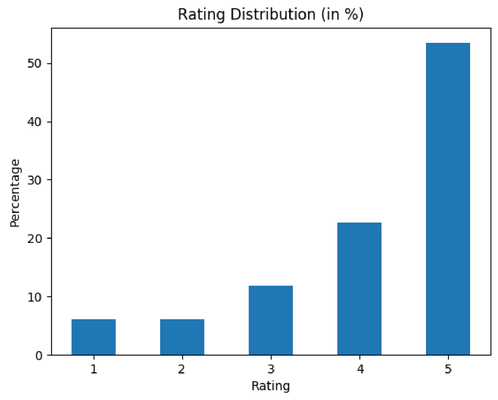
- Scalability
- Extending to non-binary use cases

Dataset

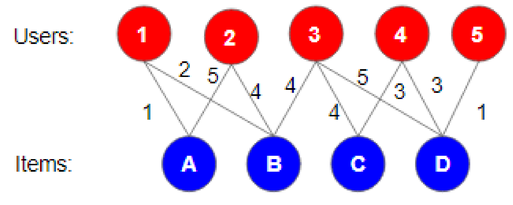
Amazon review dataset of 'Movies & TV'. We use 5-core version. Details:

- 1.75 million ratings
- 125 thousand users
- 50 thousand products

Skewed distribution of ratings. More than half are 5/5, less than 15% below 3.



We construct user-item bipartite graph for our algorithms as below. Degree distribution follows power-law. Small diameter: 5.8 for the bipartite graph.



Evaluation

RMSE over ratings in the test set. Important to model nodes rated with small ratings.

Random Walk with Restarts (and variations)

Basic Random Walk with Restarts commonly used to generate relevance rankings. Modified version:

Input : Weighted bipartite graph G, number of random walks to perform N, source node to start the random walk src
Output: Predicted ratings for every node $n \in G$

```
for i ∈ 1, 2, ··· N do
    Node u = src
    sumWalkWeights = 0
    numSteps = 0
    for s ∈ 1, 2, ··· numSteps do
        v ← randomNeighbor(u)

        sumWalkWeights += weight(u, v)
        numSteps++

        sumAvgWts[v] +=
         sumWalkWeights/numSteps
        visitCount[v] ++

        u ← v
        if random() < β then
            break
        end
    end
end
for n ∈ Nodes(G) do
    predictions[n] =
     sumAvgWts[n]/visitCount[n]
end
return predictions
```

Average over shortest paths

Aim to model flow of influence from one user to another. To predict rating for user u and product i, find average ratings across all shortest paths between u and i. Motivation:

- Helps to capture low ratings.
- Small diameter of graph $\Rightarrow$ small shortest paths. This makes the solution scalable.

Results

The shortest path approach performs the best among those sampled. Several extensions possible.

No.	Method	RMSE
0	Random	2.62
1	Cosine Similarity	1.69
2	Matrix Factorization	1.29
3	Weighted RWR	1.49
4	Average over RWR	1.21
5	Average Shortest Paths	1.05

Interpretations

Our hypotheses:

- Matrix factorization performance likely due to rich information in the 5-core.
- Weighted RWR counts very small for most nodes.

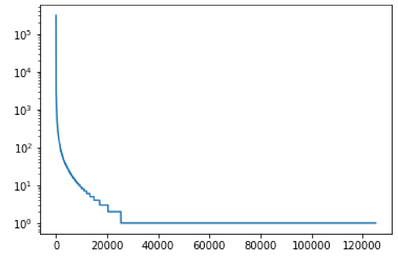


Figure: Random Walk visit counts

- Naive ranking to rating conversion in RWR. 'Average over RWR' improves on this.
- High performance of Shortest Path mainly from better capturing of low-rated products.
- Considering all shortest paths help to capture varied types of interactions and average out noise.