Recommender System Algorithm Building

Group 3
April 9

Table of Contents

Prediction Results

Method		Movie dataset	MS dataset
Model-based	EM Clustering	1.095278	44.43615139
Memory-based	Pearson	1.079188	33.96518
	Spearman	1.080625	33.96518
	Cosine Vector	1.087882	33.96518
	Entropy	1.089574	34.00835
	Mean Squared Difference	1.084046	35.08129
	SimRank	1.125395	34.81273
	Significance Weighting + Pearson	1.106849	33.77379
	Variance Weighting + Pearson	1.066502	33.8495
		Lower is better	Higher is better
		Best is 0	

Method Comparison

Model-Based

- No need to keep the data after training, only need the parameters
- Latent cluster assignments can give some insights about the users
- A little complex to implement but a lot of parts can be optimized with matrix computations instead of for loops
- Need to make assumptions and choose hyperparameters (number of clusters, types of distributions for cluster assignment and ratings…)
- Significant improvement in calculating Expected Utility Score for MS data

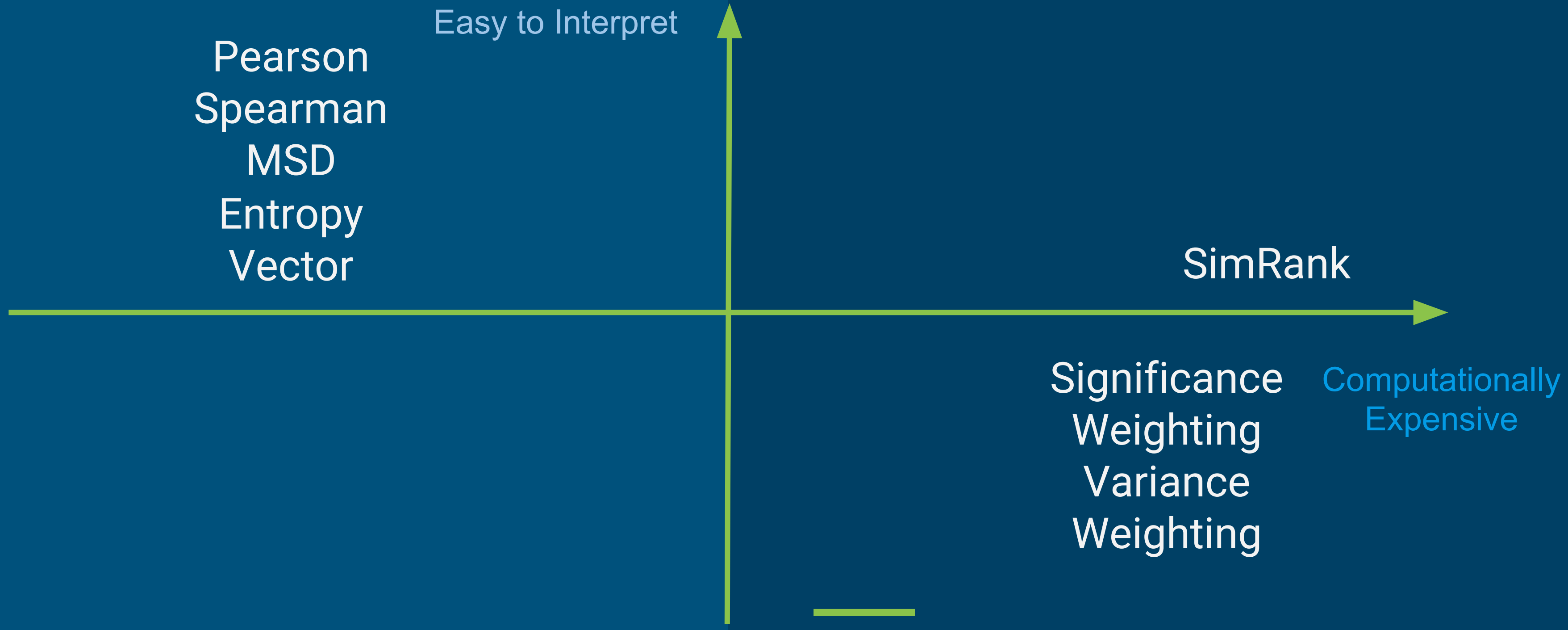
Memory-Based

- Easy to understand and implement initially on baseline
- Computational time varies on different similarity weights; long time to run on SimRank and Significance/Variance Weighting
- Have to keep the dataset for everytime you want to compute predictions

Timing Comparison

Method	Time
EM Model based	< 1 min to train, < 1 min for predictions
Pearson, Spearman, MSD	Less than 1 hour for MS, 2 hrs for movie
Simrank	2.5 hr + 5 hr for one iteration for movie
Significance weighting	30min (in addition to similarity matrix))
Variance weighting	4 hrs
Entropy	~4 hrs for both MS and movie
Cosine Vector	~3 hrs for both MS and movie

Similarity Weight Comparison



Evaluation Criteria

For Movie Data:

The most intuitive way is to calculate the Mean Absolute Error.

Find the columns by name to filter out only the movies test data has on prediction matrix.

For every user,
MAE=sum(|pred-actual|)/number_of_movies

For the whole dataset,

MAE_movie=mean(MAE_user)

For MS Data:

Expected Utility Measure: Half-Life Utility

Probability of an item being viewed decays as it goes down the list

Utility is either 1 or 0, depending on the test dataset

Sorted each row of the prediction matrix to obtain a ranked list of items for each user

Thanks for Watching