Let's Play!
Video Game Recommenders

Based on Collaborated Filtering and Content-Based Model

PRESENTER: 王維瀚 WEI-HAN WANG
ADVISOR: 王士豪 教授 PROF. SHYH-HAU WANG

Introduction

- **What is Steam?**
  - Largest distribution platform for PC gaming.
  - Allows the public to access their games and software from any computer or mobile device around the world.
  - Allows independent game developers to sell their games

Stakeholder & Objectives

- **Main Stakeholders:**
  - Valve Corporation
  - Steam customers

- **Objectives:**
  - Build a model based on detailed information and user reviews of games.
  - Use model to gain deeper understanding of data set and the game industry.
  - Provide recommendations for specific users.

Motivation

- Companies like YouTube, Netflix, 博客來 use recommender systems to suggest items to users.

- Effective models are needed to recommend items that can both match users' interests and potentially yield profits to these companies.

DATASET

Dataset Information

	User - Item	Game Item
Features	Steam ID, User ID	Game ID, Game Name, Game Title
	User URL	Genres, Tags, Specifications
	Owned Item ID, Item Name	Price, Discount Price
	Total Owned Item Count	Developer, Publisher
	Total Playtime for each Item	Release Dates, Early Access
	Total Playtime in recent 2 weeks for each Item	Review URL, Sentments, Metascore
Source	https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data	

Data Preprocessing

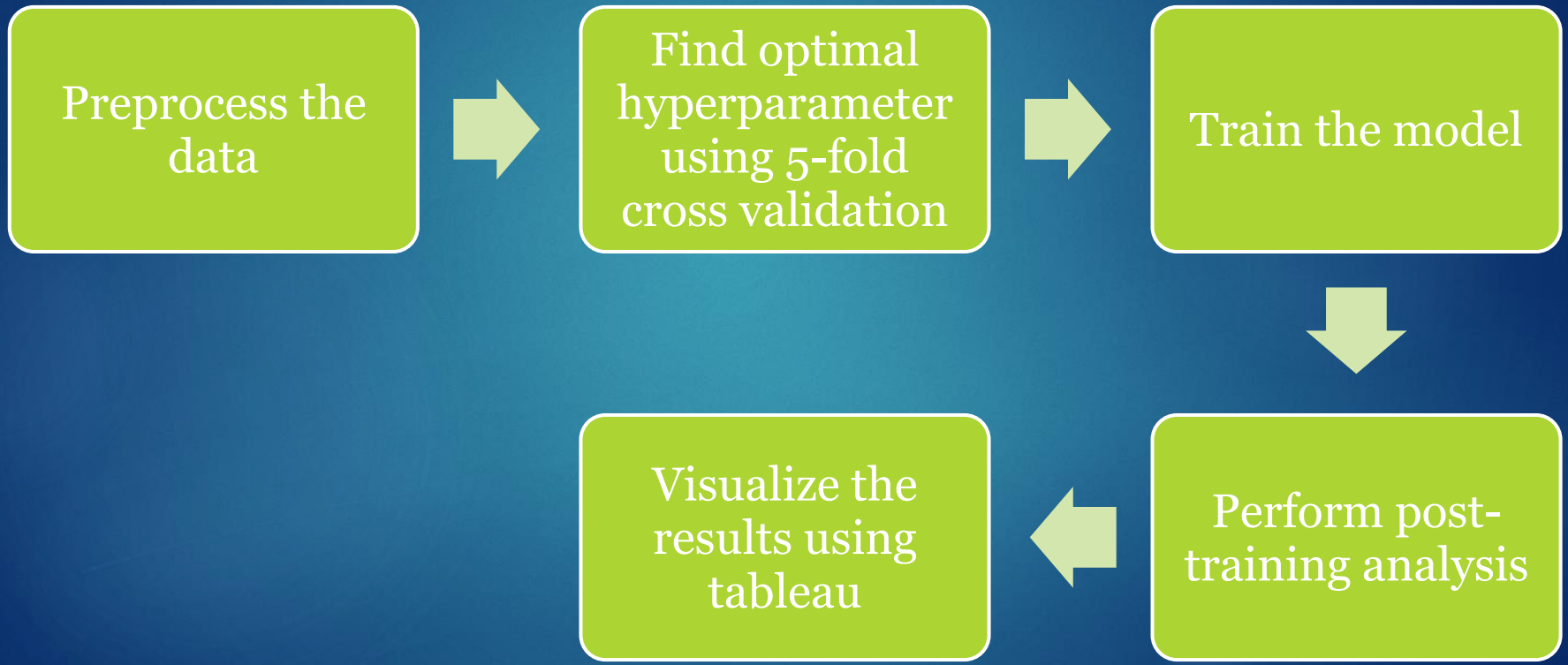
1. Clean game metadata and join them to user_items
2. Build interaction matrix

3. Filter 1000 most owned games
   - Faster training
   - Game ownership becomes sparse beyond top 1000

4. Remove all free games from user-item dataset
   - Focus on games that cost money
   - From experimentation, model performs better

item	r	s	u	w	x	y	z
user							
-	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
v	0.0	0.0	0.0	0.0	0.0	0.0	0.0
w	0.0	0.0	0.0	0.0	0.0	0.0	0.0
x	0.0	0.0	0.0	0.0	0.0	0.0	0.0
y	0.0	1.0	0.0	0.0	0.0	0.0	0.0
z	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Data Pipeline



MODEL

Design Approach

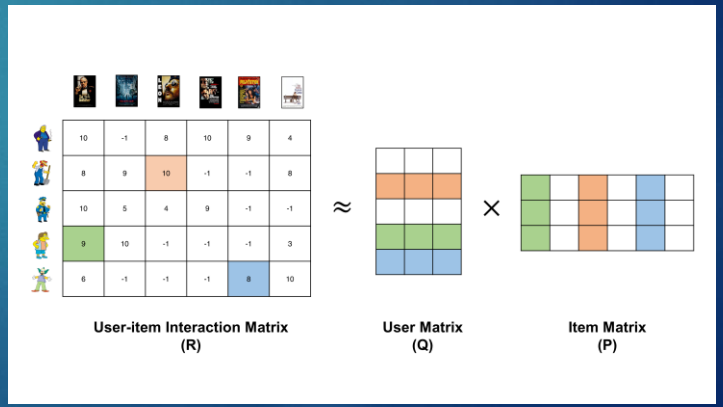
- Content-Based Filtering
  - Prediction based on user's past item metadata.
  - More like traditional classification/regression.
  - Recommend items based on user profile and item features
- Collaborative Filtering
  - Prediction based on past ratings and preferences of other similar users
  - Build item-item matrix to determine relationships between pairs of items
  - Recommendation based on similarity between users
- Hybrid recommender system
  - Collaborative Filtering + Content-Based Filtering
  - Popular to address weaknesses of single approach.

Collaborative Filtering

- Make predictions about a user's interests by considering what similar users also like.
  - Common problems: Cold Start, Sparsity, Scalability

- Interaction matrix constructed from the items owned by users.
  - Usually use single feature as the interaction.
  - Interaction matrix used as input by learning algorithm.
  - Matrix factorization takes interaction matrix and decomposes it to two

item	Europa Universalis IV	Hurtworld	Interstellar Marines	ORION	Overlord: Raising Hell	Strike Suit Zero	Terraria
user							
76561198040806617	0	0	0	0	0	0	0
76561198071059521	0	0	1	0	0	0	1
76561198077795250	0	0	0	0	0	0	0
76561198093085776	0	0	0	0	0	0	0
76561198096850767	0	0	0	1	0	0	1
Hueheueeheuee	0	0	0	0	0	0	0
Vladimirputinisgreatman	0	1	0	0	0	0	1
pinkie10	0	0	0	0	0	0	1
sonnymack	0	0	0	0	0	0	1
urami	0	0	0	0	0	0	0



Users/ Items	1	2	3	4	5	6	7	8	9
A	4.5	4				2.8		1	
C	1.9	2		4.1			3.9		
D	2			4		5	5		1
J	1.5	5	2	4		3.4			
T			3			3.5		2	3

score

C is most similar to D;
A is most similar to T;
J is most similar to T

Calculate cosine similarity between each item.
For example, the cosine similarity of C and D is:

$$f(similarity) = \frac{1}{\sqrt{(C_1-D_1)^2 + (C_4-D_4)^2 + (C_7-D_7)^2}}$$

$$= \frac{1}{\sqrt{1.9-2)^2 + (4.1-4)^2 + (3.9-5)^2}}$$

$$= 0.90$$

Users/ Users	A	C	D	J	T
A		0.30	0.30	0.31	0.82
C	0.30		0.90	0.33	-
D	0.30	0.90		0.60	0.40
J	0.31	0.33	0.60		0.99
T	0.82	-	0.40	0.99	

Model Description

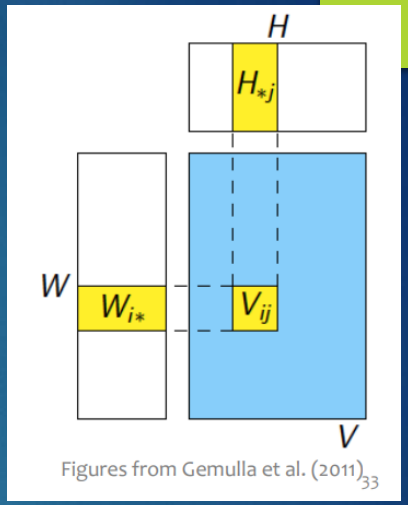
- **Bayesian Personalized Ranking (BPR):**

✓ A matrix factorization algorithm for ranking items of a given user given binary interaction.

✓ The algorithm generates embeddings, ɣ_u and ɣ_i for every user and item. Score is then calculated by taking dot product of ɣ_u and ɣ_i.

✓ Embeddings found using iterative optimization like SGD.

✓ Sort scores to generate top n recommendation list.

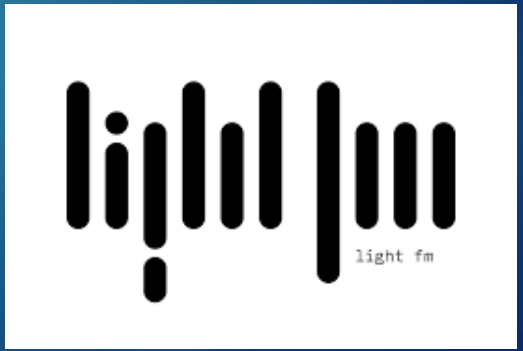


Figures from Gemulla et al. (2011)

- **LightFM:**

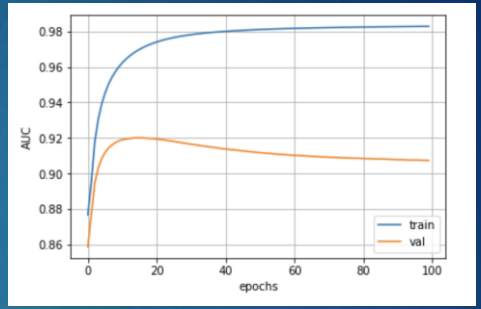
✓ used to implement model

✓ Fast, easy to use, quick for retrieving results



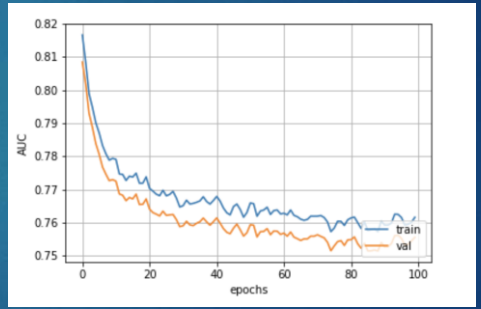
Model Evaluation

- Model performance evaluated by AUC-ROC.
  - As usual, split data into train and test sets
  - AUC = probability that a positive example in test ranks higher than negative example



WARP

- Consider BPR and WARP loss functions.
  - Weighted Approximate-Rank Pairwise (WARP) only updates embeddings if violation found.
  - BPR prone to overfitting with more epochs
  - WARP provides across the board better results.



BPR

Model Tuning

- Random hyperparameter search
  - Searched over 500 sets of hyperparameters.
  - 5-Fold Cross Validation used to reduce overfitting.
  - Saved results to PostgreSQL database to later compare and retrieve.

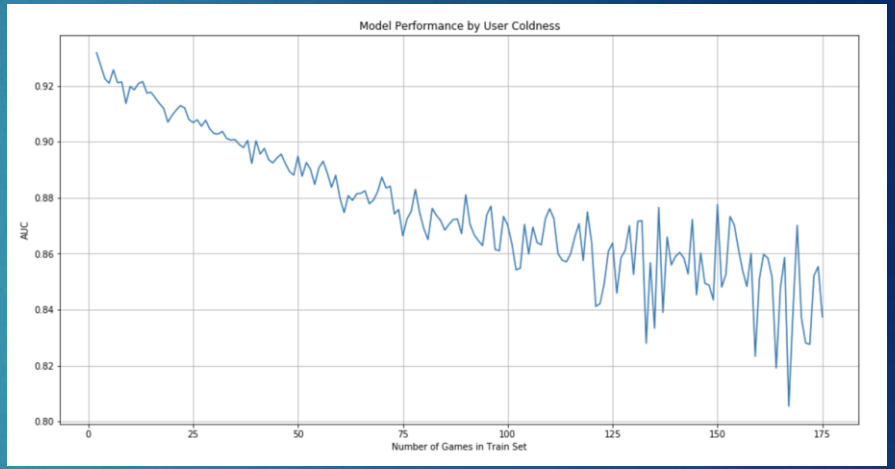
- Main hyperparameters to tune:
  - Loss function - BPR vs WARP
  - Learning rate + choice of optimizer - AdaGrad
  - Item L2 regularization - Regularization for item embedding weights
  - User L2 regularization - Regularization for user embedding weights
  - No_components - Dimension of embeddings

FINDINGS

Cold Start Problem

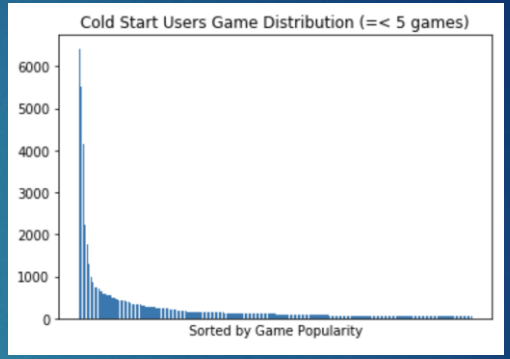
- Cold users normally harder to give recommendations.
  - Common problem in recommendation
  - Fewer interactions to train on

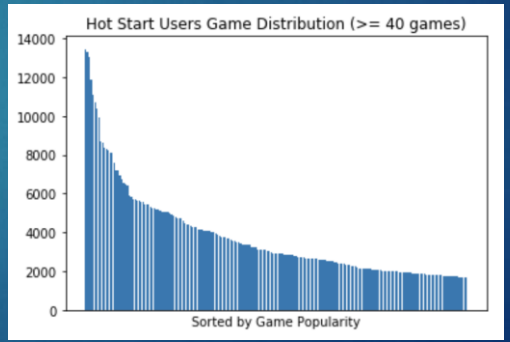
- Model performed better on cold start users than hot start.
  - Collaborative model only
  - Counterintuitive results. Normally not the case. Less data = better results?



Cold Start Problem

- Cold users own mostly popular games
  - Model performs well on popular games.
  - Testing on popular games return high AUC.
- Hot users' library is much more spread out.
  - Model must dig deeper to find a good recommendation.
- Video games different from fashion or film. Data follows a trickle down structure.





Training on Bought vs Played

- 40% of games bought are played less than 30 minutes.
- 21% of games bought and never opened.

- Train four models on different interaction matrices.
  - No filter data set emphasizes games that users **bought**
  - Other models emphasize games that user bought and **played**.

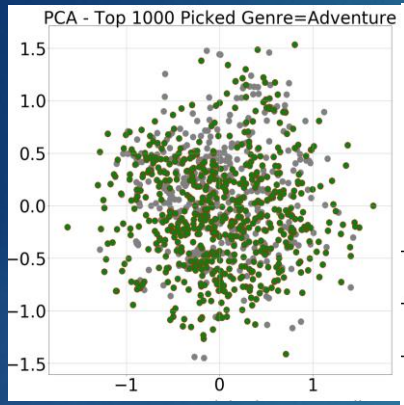
		Test Set			
		No Filter	Only games played > 0 minutes	Only games played > 30 minutes	Only games played > 10th percentile/game
Model	No Filter	0.9151	0.9076	0.9106	0.9061
	Only games played > 0 minutes	0.881	0.9102	0.9179	0.9084
	Only games played > 30 minutes	0.8623	0.9015	0.9168	0.8995
	Only games played > 10th percentile/game	0.8812	0.9105	0.9185	0.9089

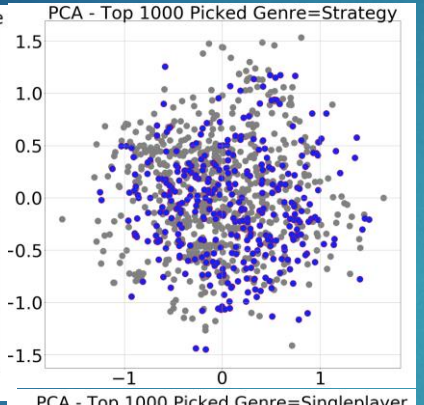
Training on Bought vs Played

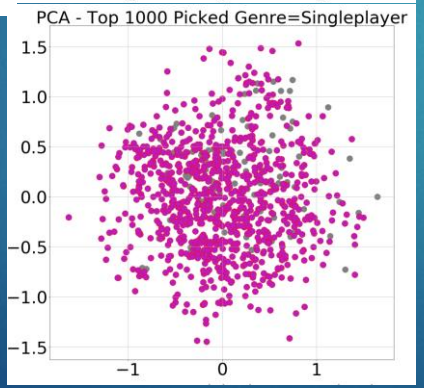
- Is filtering good idea?
  - Filtered data sets is slightly better for played games
  - No filter has best all around performance.
- Roughly speaking, quality of training decreases as size of data set decreases.
- Use no filter set finally.

		Test Set			
		No Filter	Only games played > 0 minutes	Only games played > 30 minutes	Only games played > 10th percentile/game
Model	No Filter	0.9151	0.9076	0.9106	0.9061
	Only games played > 0 minutes	0.881	0.9102	0.9179	0.9084
	Only games played > 30 minutes	0.8623	0.9015	0.9168	0.8995
	Only games played > 10th percentile/game	0.8812	0.9105	0.9185	0.9089

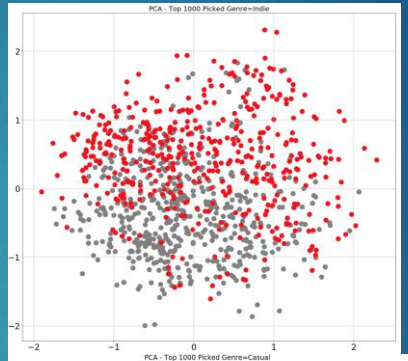
Principal Component Analysis - Game Genres

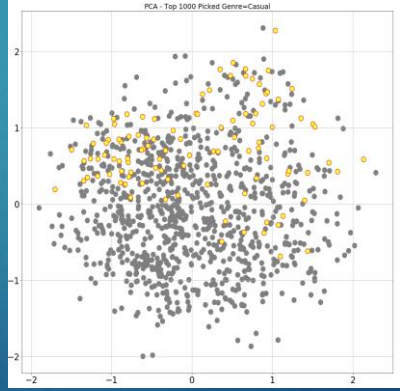


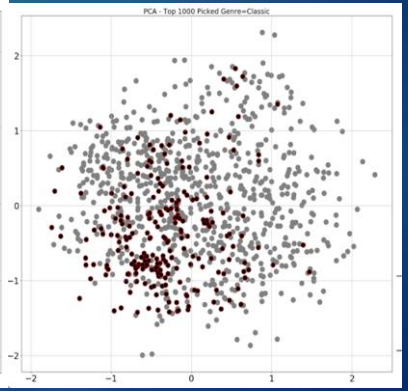




Games with generic genres:
hard to determine pattern or cluster

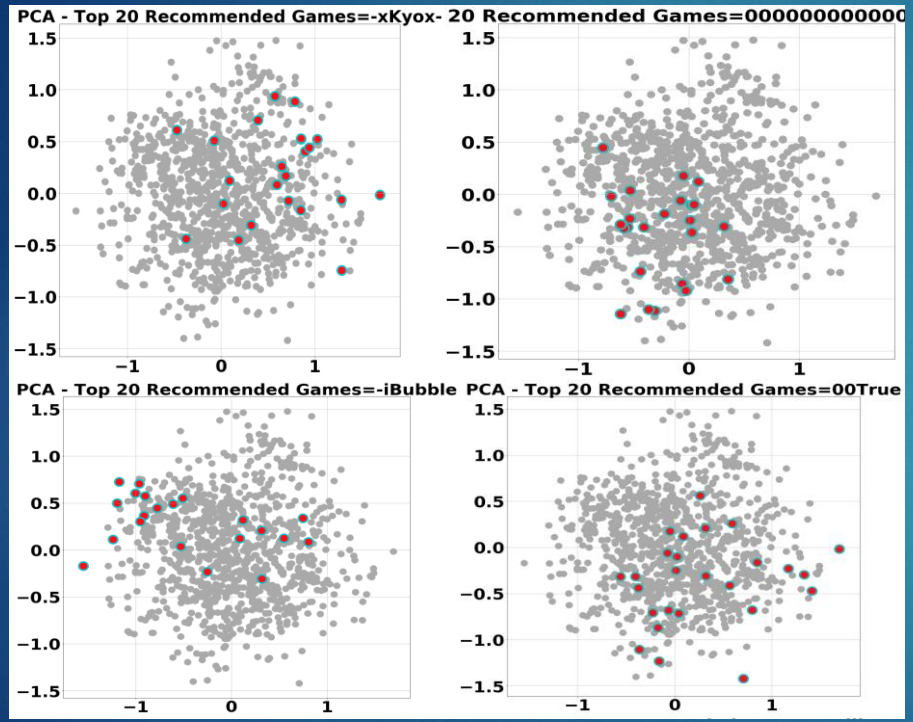






Games with unique genres:
clusters distributed in certain pattern

PCA - Recommended Games



- PCA used on item embeddings to visualize how items relate on top two principal components.
- PCA results when plotting recommended games for a selected user against top 1000 games
- Users with their purchased games located in 1 or 2 clusters indicate likelihoods for same type of game.
- Users with their purchased games diversified indicate games belongs to different and varied genres.

Hybrid Model

- Hybrid = Collaborative Filtering + Content-based Filtering

  = User-Item Interaction Matrix + User Item Feature Matrices

- Feature Selection

User Features	Item Features
No Profile Information	Genres
	Price
	Released Year
Total Playtime →	Median Total Playtime
Recent Playtime →	Median Recent Playtime

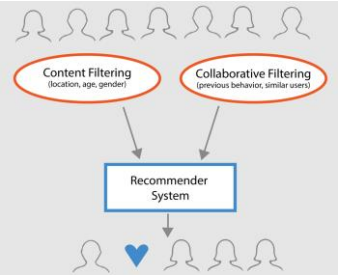
Hybrid Model

- Item Feature Data Preparation
  - Feature Matrix = Number of Items x Number of Features
  - Feature weights calculation
    - Normalization by each feature item separately
    - Normalization by each game item
- Performance
  - Applied with Several configuration parameters combinations
  - Similar AUC with Training and Test Datasets
- Future work
  - Hyperparameter Search and Model Tuning
  - Data Filtering

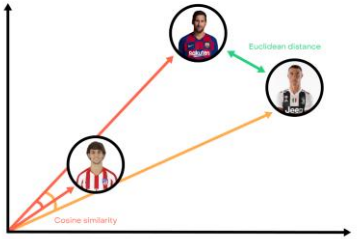
CONCLUSION

Model Outputs

- **User-Item Recommendation**
  - Determined from collaborative filtering model
  - Score and rank items for each to return top recommendations



- **Item-Item Recommendation**
  - Cosine similarity is used to compute distances between items in interaction matrix
  - Measures similarity between games
  - Rank items based on top distances



Dashboard

- Built with Tableau
- Dashboard Elements
  - ✓ Top 20 Recommendations for Each User
  - ✓ Top 20 Recommendations for Each Item
  - ✓ Principal Component Analysis of Recommended Games
- Built with Game Title Images to present the model outputs

Dashboard

Steam User IDs

Game selection highlight

Game selection highlight

PCA Plot of Top 20 Recommendations and Top 1000 games

Top 20 Game Recommendations for each user

Top 20 Game Recommendations for each item selection

Conclusion

- Video games are less susceptible to cold start problem compared to other data sets normally used for collaborative filtering. Cold start users can sometimes be given better recommendations due to their smaller library.
- PCA is a powerful way to visualize natural grouping among BPR embeddings. Artificial groupings such as genre coincide with natural groupings based on collaborative filtering.
- Data filtration can sometimes improve performance of model. BPR and WARP performs well even after removing data. However, keep all data is still better in here.
- Future work include obtaining more data and scaling model across entire Steam system (millions of users) and improve hybrid recommender modeling.

DEMO