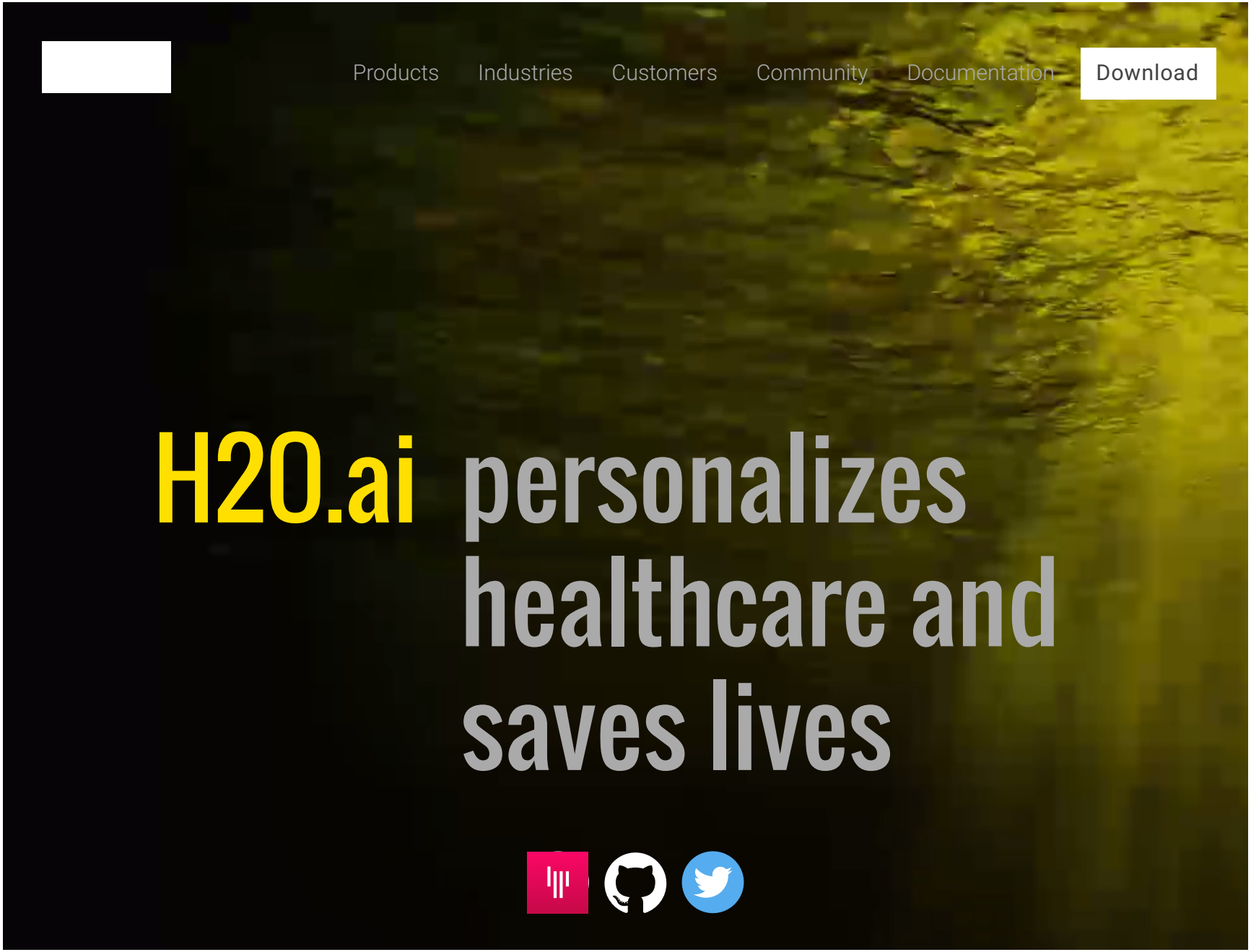
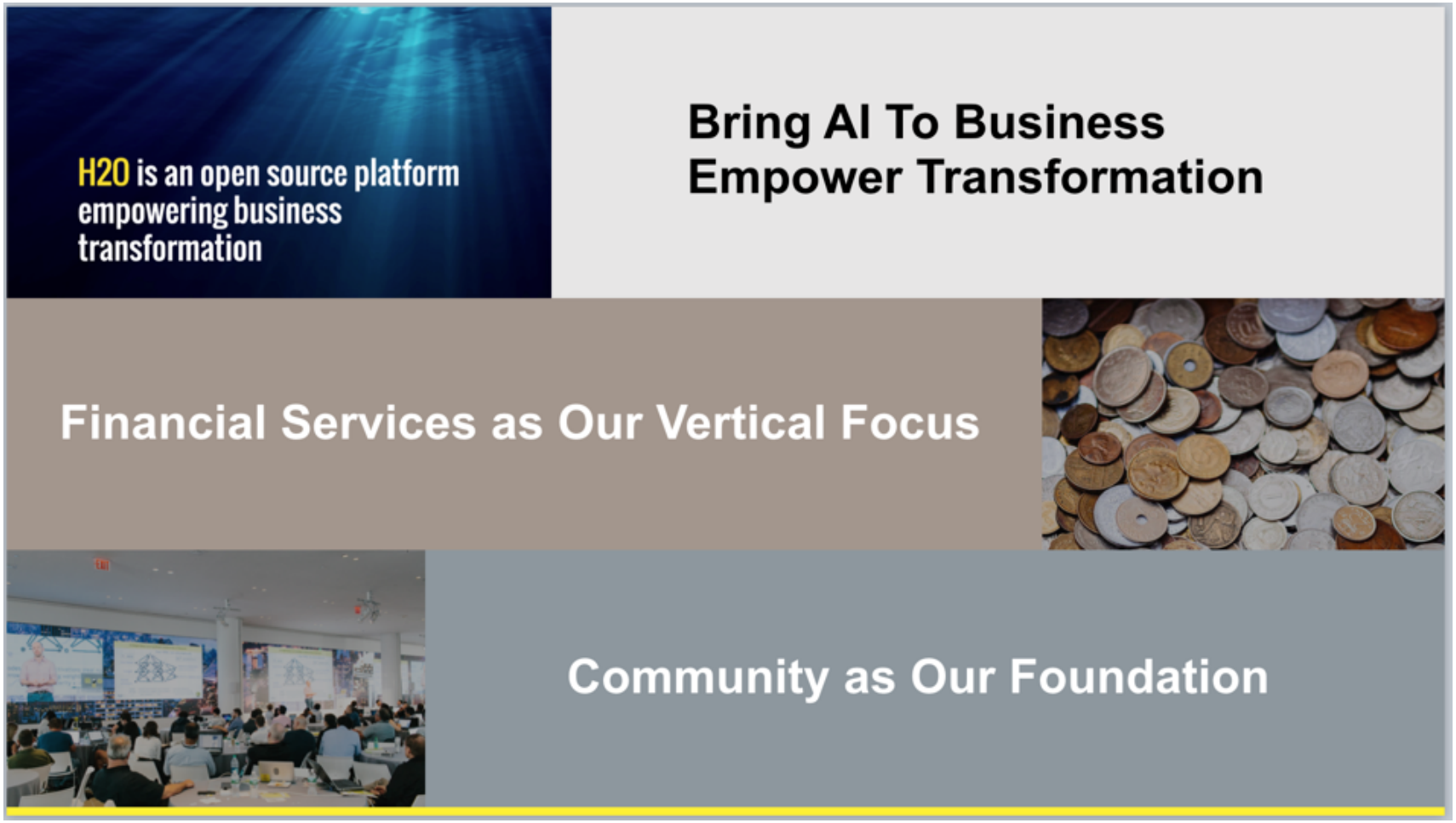
Recommender Systems with H2O

Megan Kurka


Products
Industries
Customers
Community
Documentation
Download
H2O.ai personalizes healthcare and saves lives

2/39



H_2O.ai

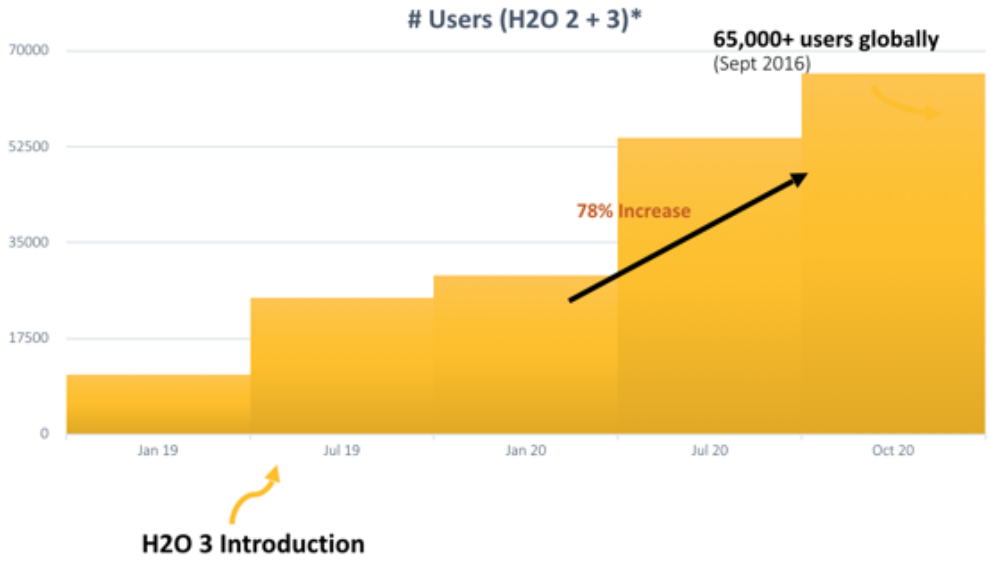
Machine Intelligence

H2O.ai

Machine Intelligence

Community Growth

Tremendous Momentum Globally



* DATA FROM GOOGLE ANALYTICS EMBEDDED IN THE END USER PRODUCT

Large User Circle

- 65,000+ users from ~8,000 companies in 140 countries. Top 5 from:

1. United States
2. India
3. Japan
4. Germany
5. United Kingdom

H2O.ai

H2O.ai

Machine Intelligence

H_2O.ai Makes A Difference as an AI Platform

Open Source	Integration with Big Data Frameworks	Flexible Interface	Smart and Fast Algorithms
• 100% open source	And More...	H2O Flow	

Scalability and Performance	Rapid Model Deployment	GPU Enablement	Cloud Integration
• Distributed In-Memory Computing Platform • Distributed Algorithms • Fine-Grain MapReduce	• Highly portable models deployed in Java (POJO) • Automated and streamlined scoring service deployment with Rest API		

H_2O.ai

Machine Intelligence

H_2O.ai Offers AI Open Source Platform

Product Suite to Operationalize Data Science with Visual Intelligence

Visual Intelligence and UX Framework For Beautiful Data Products

100% Open Source

H_2O.ai

In-Memory, Distributed Machine Learning Algorithms with H_2O Flow GUI

Deep Water

Best in class Deep Learning on GPUs with TensorFlow, MXNet or Caffe with the ease of use of H2O

Spark + H_2O SPARKLING WATER

H2O AI Open Source Engine Integration with Spark

Steam

Operationalize and Streamline Model Building, Training and Deployment Automatically and Elastically

H_2O.ai

Machine Intelligence

H2O.ai

Machine Intelligence

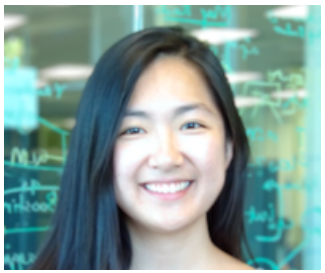
Products Industries Customers Community Documentation

Team

Join us and help change how

the world discovers insights from data

JOIN US ☐ (HTTP://H2OAI-WEB.HEROKUAPP.COM/CAREERS/)


9/39

Chris Mascioli

Data Scientist

Michal Kurka

Senior Software Engineer

Kerry O'Shea

Director of Customer Experience

Megan Kurka

Customer Data Scientist

H_2O.ai

Machine Intelligence

What is a Recommender System?

aims to suggest new items to users

build a model that predicts how much a user will like an item

use this information for personalized product recommendations, website personalization, and personalized loyalty programs and offers

What data is used?

User Behavior Data

Item Features

User Features

Contextual Information

Content Based Filtering

model uses the user's preference of item features to predict how much they will like an item

In the past I've liked comedy movies, recommend a comedy movie.

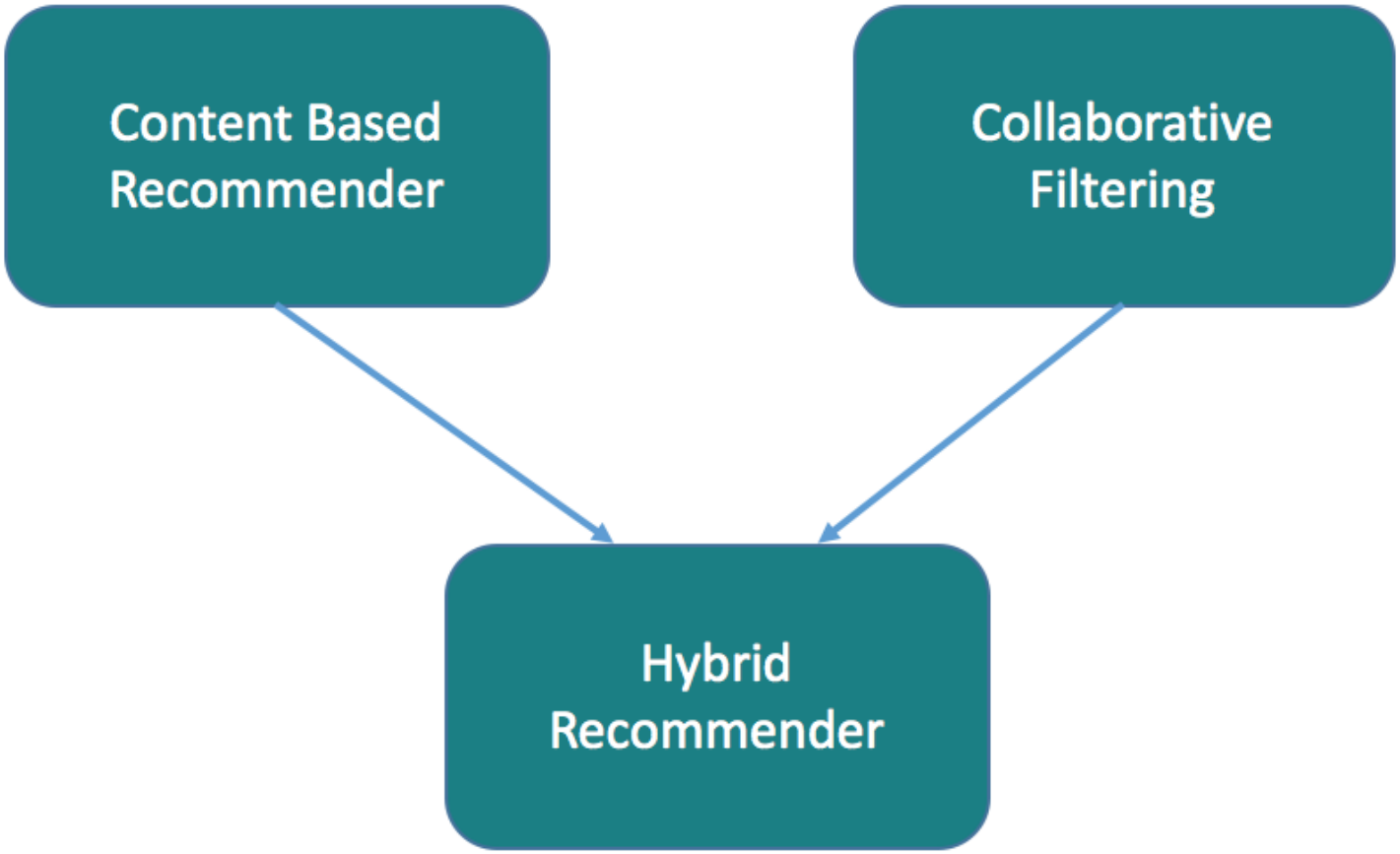
Collaborative Filtering

model uses the behavior of many users to predict how much a user will like an item

My friend and I have liked the same movies in the past. Recommend a new movie to me that my friend likes.

Hybrid

combining content based filtering and collaborative filtering


Content Based Recommender
Collaborative Filtering
Hybrid Recommender

The Problem

Generate 10 "good" movie recommendations for each user.

The Data

1 million user ratings of movies on a 1-5 scale

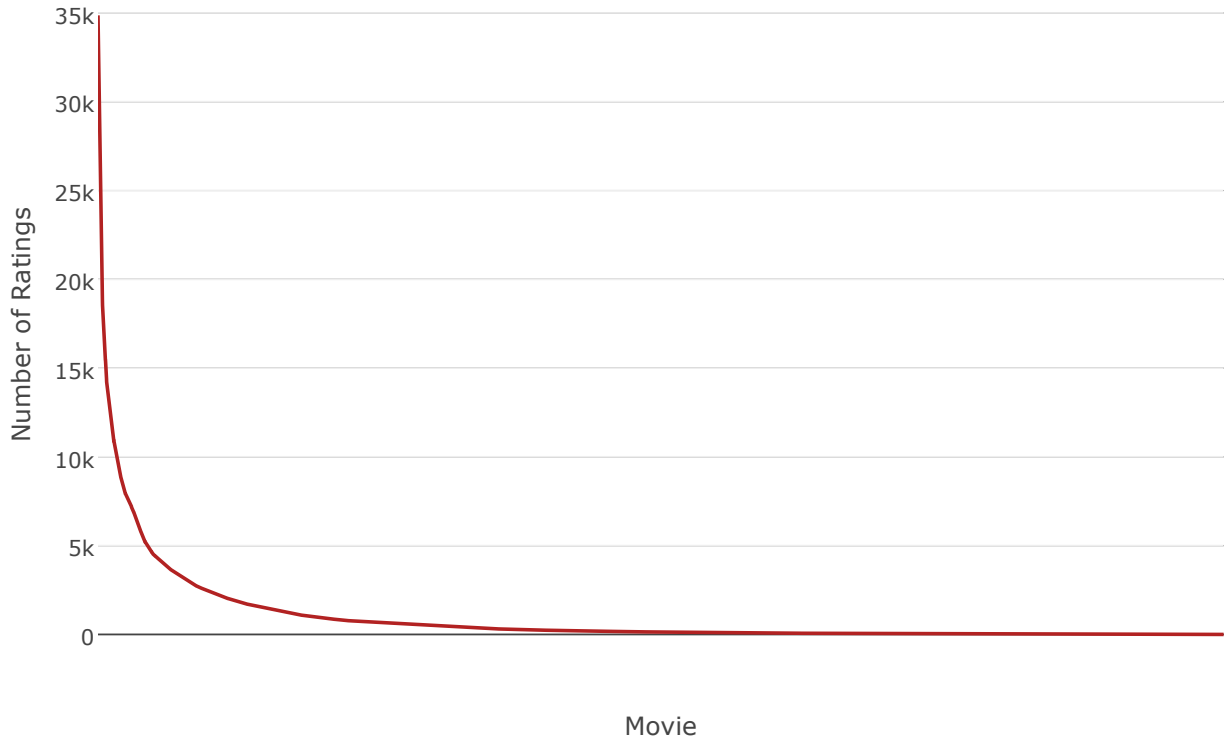
movie features

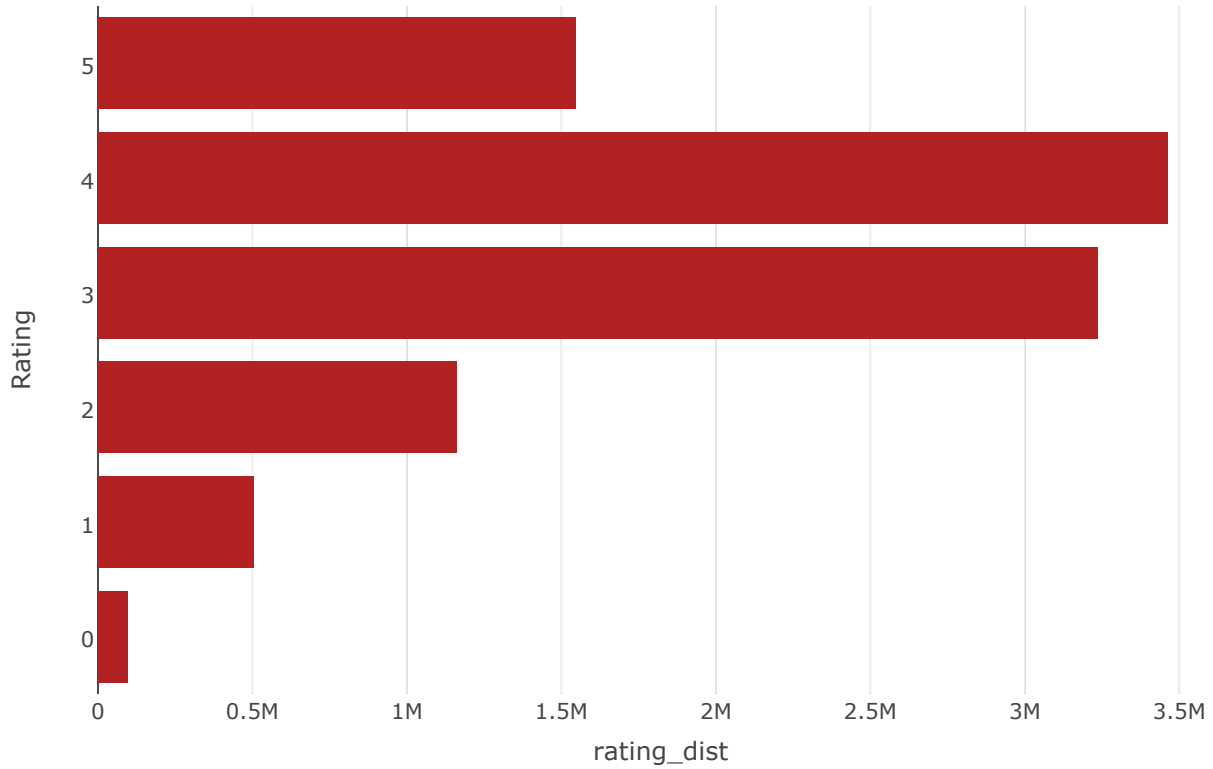
The Data

USERID	TITLE	RATING	TIMESTAMP
1	Jumanji (1995)	3.5	2005-04-02 18:53:47
1	Rob Roy (1995)	4	2004-09-09 23:08:54
1	Clerks (1994)	4	2005-04-02 18:46:13
1	Pulp Fiction (1994)	4	2005-04-02 18:32:47

Table: Table continues below

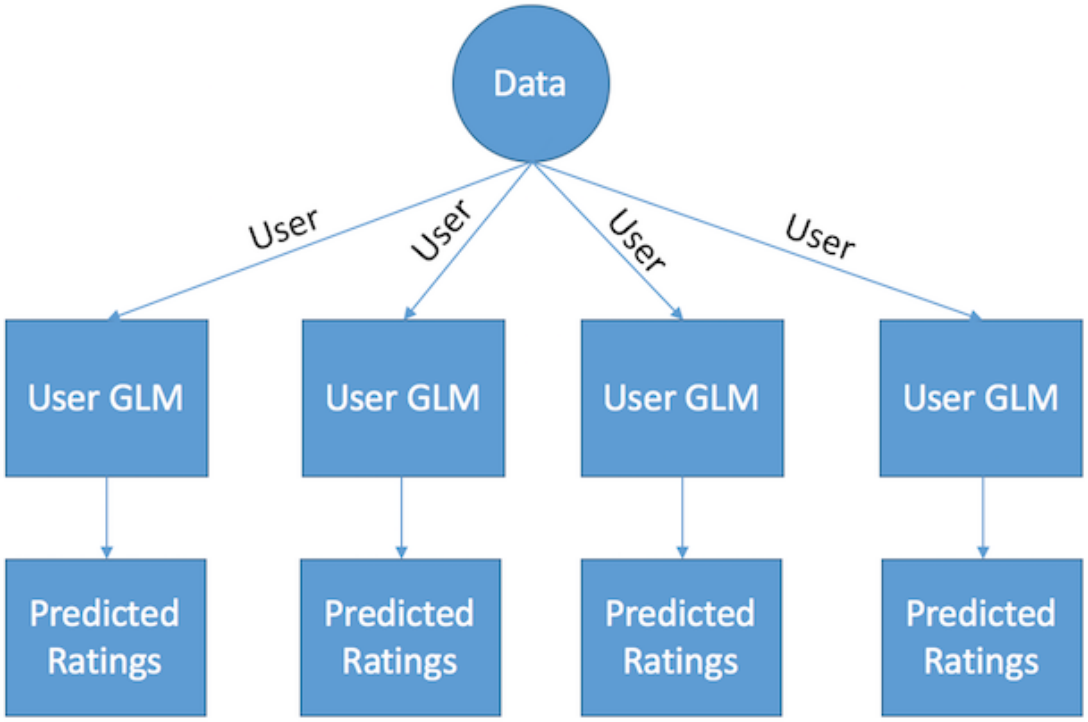
GENRES
Adventure,Children,Fantasy
Action,Drama,Romance,War
Comedy
Comedy,Crime,Drama,Thriller


35k
30k
25k
20k
15k
10k
5k
0
Number of Ratings
Movie


5
4
3
2
1
0
Rating
0
0.5M
1M
1.5M
2M
2.5M
3M
3.5M
rating_dist

We will create a model based on user's preferences of movie features to predict ratings.

Build a GLM model for each user predicting the rating they will give of a movie based on the movie's attributes.



User's GLM

$$\text{rating} = 3.85 + 3.14 * \text{Romance} - 0.4 * \text{Action} - 0.38 * \text{Comedy}$$

The Code

```
glm_model <- h2o.glm(x = genres, y = "rating",
                     training_frame = user_train,
                     validation_frame = user_test,
                     remove_collinear_columns = TRUE)
```

Pros:

handles the cold start problem: can provide recommendations on new movies.

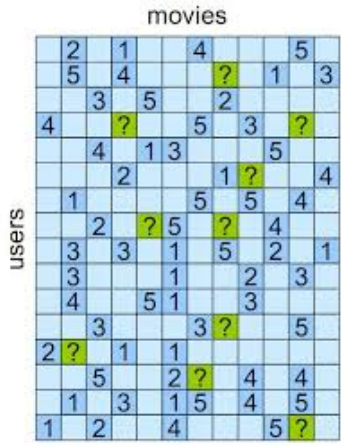
Cons:

sparse data: some users may not have provided many ratings

recommendations only as good as the movie attributes

We will use a Low Rank Model to impute the missing values in the matrix. If we know the ratings a user would give all movies, we can produce recommendations.

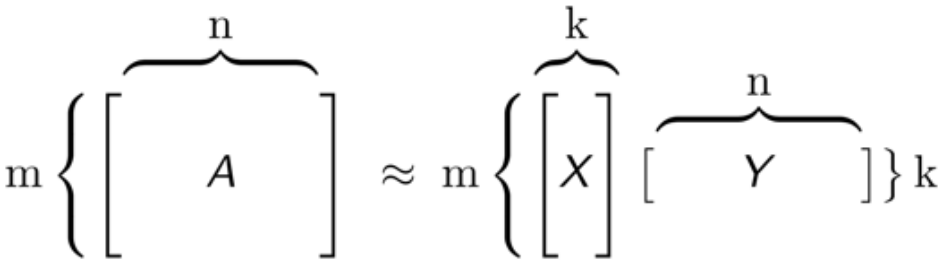
Can be viewed as a low-rank matrix completion problem with noise, i.e., approximate a given noisy data matrix featuring missing entries with a low-rank matrix.



Data table with m rows and n columns

Compress representation as numeric tables and , where # cols in X = # rows in Y = small user-specified k << max(m, n)

cols in is d = (total dimension of embedded features in) >= n



= archetypal features created from columns of

= row of in reduced feature space

GLRM can approximately reconstruct from product

minimize

$$||A - XY||^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (A_{i,j} - x_i y_j)^2$$

with variables

$$X \in R^{mxk}, Y \in R^{kxn}$$

minimize

$$L_j(x_i y_j, A_{ij}) + \sum_{i=1}^{m} r_i(x_i) + \sum_{j=1}^{n} \hat{r}_j(y_j)$$

general loss function (doesn't have to be squared error)

can specify different loss functions for each column

regularizers: prevent overfitting

Alternating Minimization

$$X^t = argmin_X(L_j(x_i y_j^{t-1}, A_{ij}) + \sum_{i=1}^{m} r_i(x_i))$$

$$Y^t = argmin_Y(L_j(x_i^t y_j, A_{ij}) + \sum_{i=1}^{m} \hat{r}_j(y_j))$$

for t = 1, ... until a stopping criterion is satisfied

Gradient Method

$$g = \sum \nabla L_j(x_i y_j, A_{ij}) y_j + \nabla r_i(x_i)$$

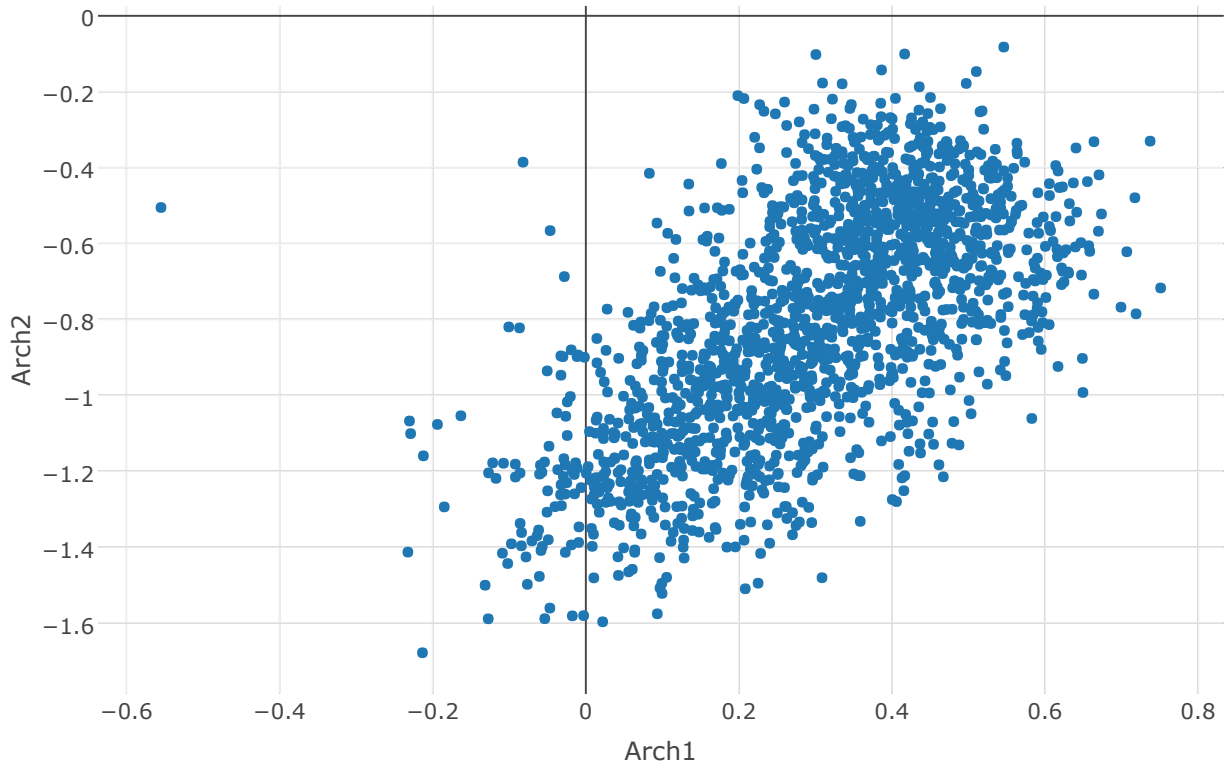
$$x_i^t = x_i^{t-1} - \alpha_t g$$

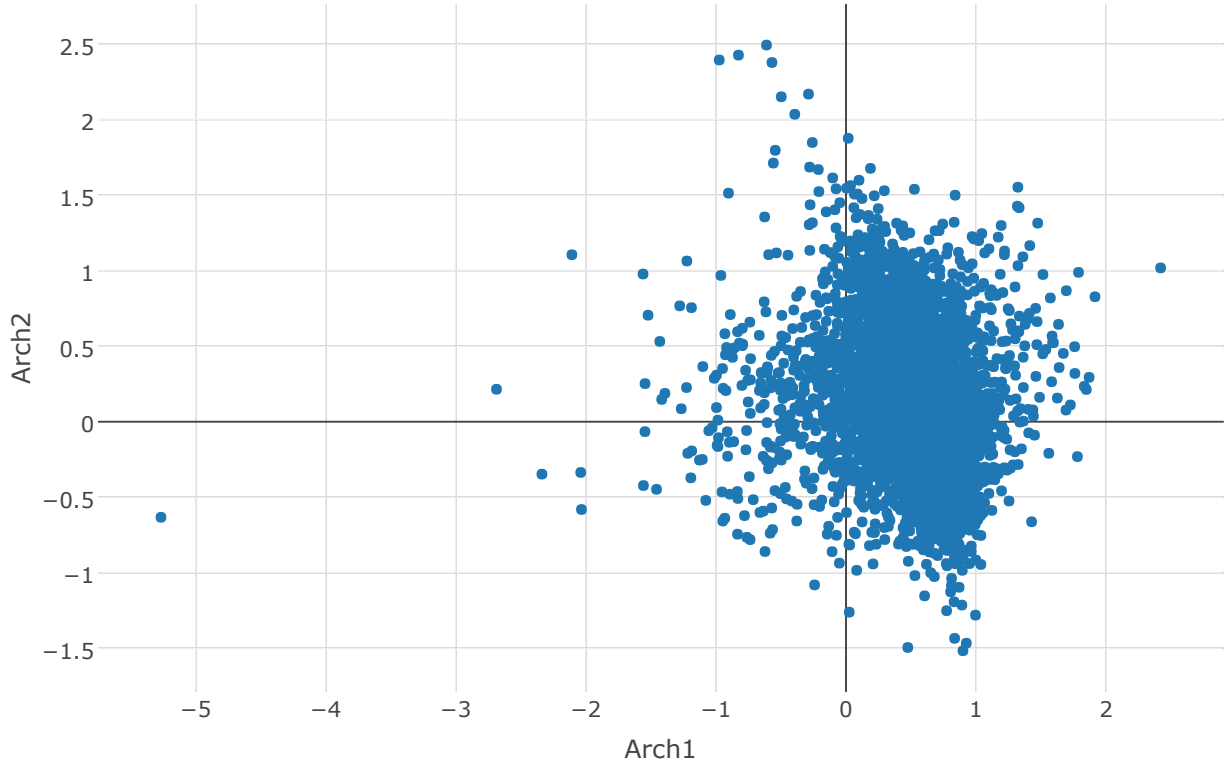
```
glrm_rank <- 2
glrm_cols <- colnames(sparse_train)[c(2:ncol(sparse_train))]

base_model <- h2o.glrm(sparse_train, cols = glrm_cols, k = glrm_rank,
                       validation_frame = sparse_test, seed = 1,
                       regularization_x = "Quadratic",
                       regularization_y = "Quadratic",
                       gamma_x = 1, gamma_y = 1,
                       transform = "DEMEAN", impute_original = TRUE)
```

In building a GLRM model, we decompose our original matrix into two smaller matrices: X and Y.

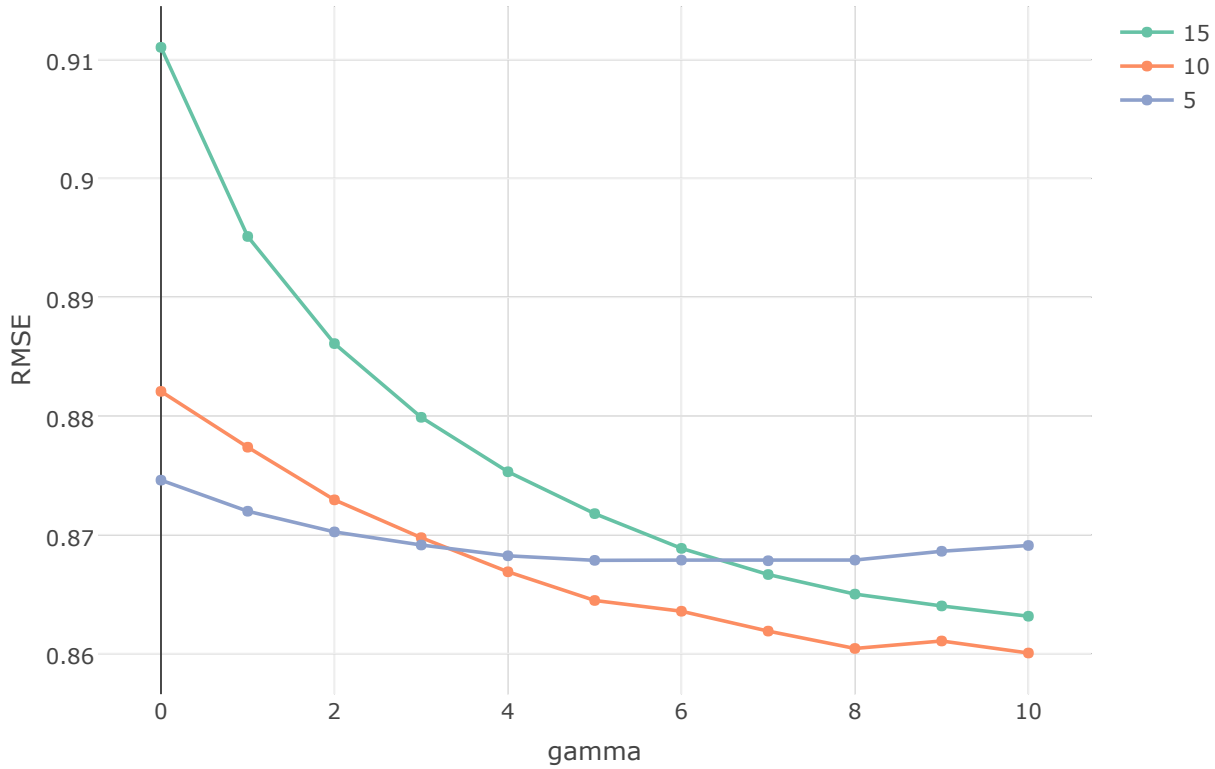
each row of Y = archetypal feature formed from the movies

each row of X = the user projected into a reduced dimension feature space (dimension 2)


0
−0.2
−0.4
−0.6
−0.8
−1
−1.2
−1.4
−1.6
Arch2
−0.6
−0.4
−0.2
0
0.2
0.4
0.6
0.8
Arch1


2.5
2
1.5
1
0.5
0
−0.5
−1
−1.5
Arch2
−5
−4
−3
−2
−1
0
1
2
Arch1

What should the Rank and Regularization Strength be of the Low Rank Model?



Combine Predictions

- Average Predictions: average predictions from Content Based and Collaborative Filtering recommendations.
- Stacking: use a super learner to combine the predictors from multiple algorithms.

Add Content Based features to Collaborative Filtering

- Ex: Add movie attributes to the user-movie matrix and generate a GLRM.

Add Collaborative Filtering features to Content Based Recommender

- Ex: Add latent factors to GLM.

Joint training of both Content Based and Collaborative Filtering

- Ex: Tensorflow's Wide and Deep Model: Builds a Content Based Recommender (GLM) and Collaborative Filtering Recommender (Neural Network) simultaneously.

Generalized Low Rank Models

https://web.stanford.edu/~boyd/papers/pdf/glrm.pdf

Wide and Deep Model

https://arxiv.org/abs/1606.07792

A Hybrid Approach to Recommender Systems based on Matrix Factorization

http://www.dai-labor.de/fileadmin/files/publications/DiplomaThesisStephanSpiegel.pdf