UW DSC.

Recommender Systems Workshop

Slideshow: https://bit.ly/3ji05zy

Notebook: https://bit.ly/3dzMDmM

Presented by Jack Douglas

Introduction

Jack Douglas

(he/him)

2B Software Engineering

Email: jack.douglas@uwaterloo.ca

LinkedIn: https://bit.ly/3gWuXUo

Workshop Overview

Intended Audience: Beginner to Intermediate Data Scientists

Goals:

- Learn about what recommender systems do and the types of data that you will see in the real world
- Learn about the two main types of recommender systems
- Learn how to implement a recommender system on your own

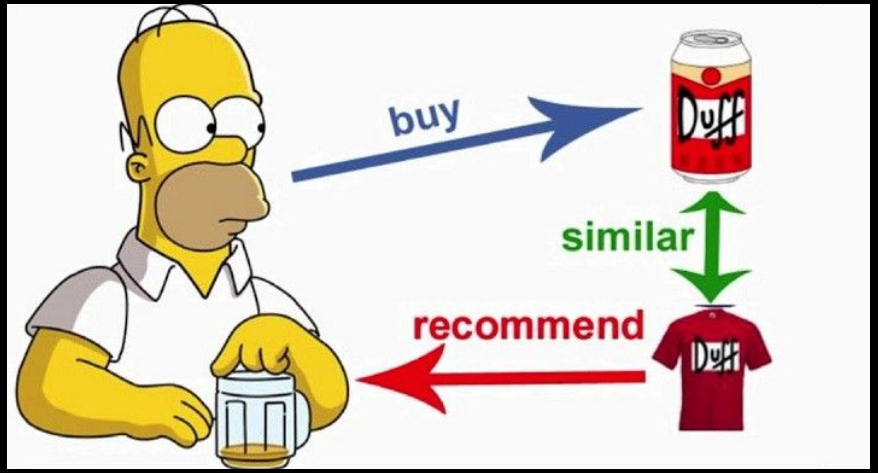
Workshop Outline

1. What is a Recommender System?
2. Motivation
3. How Recommender Systems Work
4. Types of Data
5. Useful Notation
6. Approaches to Recommendations
    a. Content-based Filtering
    b. Collaborative Filtering
7. Movie Recommendations Notebook
8. Deep Learning Recommender Systems

What is a Recommender System?

- **Def'n:** a type of machine learning system that recommend new products and services to users

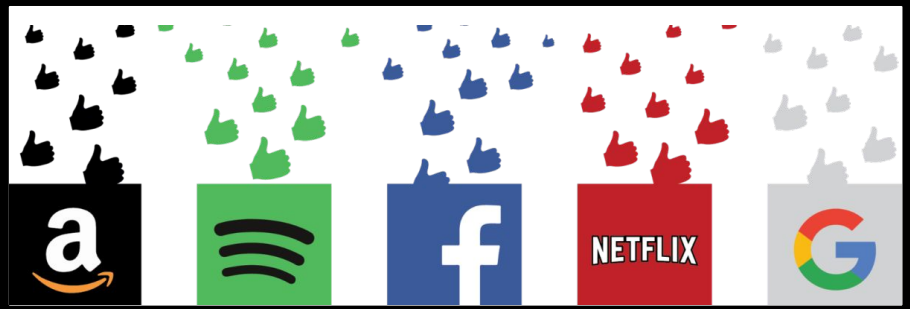
- Recommender system looks at patterns of activities between different users and different products to produce these recommendations



Motivation

- Recommendation systems are used everywhere
  - Advertisements/product recommendations, song recommendations, and movie recommendations are some examples

- Understanding recommender systems is a practical skill that can help make your résumé stand out!



How Recommender Systems Work

UW DSC.

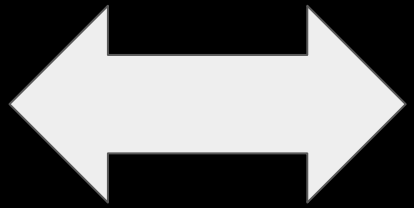
- Recommender systems seek to understand 3 key relationships:
  - **User - Product**: based on users' individual product preferences
  - **Product - Product**: based on similar products, either by appearance or description
  - **User - User**: based on similar users' likely having similar product preferences

- A combination of these relationships is used to sort the best recommendations for users

User - Product Example

If a user rates a movie highly, then the recommender system uses the connection between the user and genre, cast, director, etc. to create recommendations:

User 1



User 1 Profile

- Likes comedy
- Likes Will Ferrell
- Likes John C. Reilly
- etc.

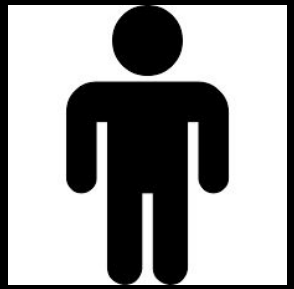
Product - Product Example

If two products share many attributes, then the recommender system forms a connection between the two products:

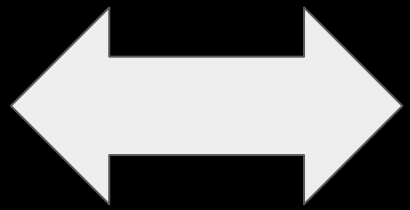
Similar casts, similar genres, same director

User - User Example

If a two users rates a movie highly, then the recommender system uses that connection between the two to help create recommendation:



User 1



User 2

Type of Data: User Behaviour Data

- **Explicit Ratings**: ratings provided by the user
  - Ex. ratings, reviews, likes, following

- **Implicit Ratings:** ratings provided from users interacting with a product
  - Ex. clicks, views, purchases

- Used in every type of recommender system

Movie	User 1	User 2
Step Brothers **(1)**	5	1
Anchorman **(2)**	?	1
The Other Guys **(3)**	?	1
Taxi Driver **(4)**	4	?
Goodfellas **(5)**	5	5

Explicit Ratings Example

Type of Data: User Demographic Data

- Examples: age, location, education, income

- Used in some recommender systems, can help improve the recommendations

NA Adapter

UK Adapter

Type of Data: Product Attributes

- Examples: genre, type of item, company producing item

- Needed in content-based filtering

Attributes: Nike, T-Shirt, Black, Graphic, etc.

Movie Rating Dataset Examples

Movie	User 1	User 2	x_1 (comedy)	x_2 (drama)
Step Brothers **(1)**	5	1	1	0
Anchorman **(2)**	?	1	1	0
The Other Guys **(3)**	?	1	1	0
Taxi Driver **(4)**	4	?	0	1
Goodfellas **(5)**	5	5	0	1

Dataset with explicit ratings and product attributes

Useful Notation

$r(i,j)$: 1 *if user j has rated movie i* (0 *otherwise*)

$x^{(i)}$: *feature vector for movie i* (*first index is always* 1)

$\theta^{(j)}$: *parameter vector for user j*

$y^{(i,j)}$: *actual rating by user j on movie i* (*if defined*)

$(\theta^{(j)})^T x^{(i)}$: *predicted rating by user j on movie i*

Notation Examples

Movie	User 1	User 2	x_1 (comedy)	x_2 (drama)
Step Brothers **(1)**	5	1	1	0
Anchorman **(2)**	?	1	1	0
The Other Guys **(3)**	?	1	1	0
Taxi Driver **(4)**	4	?	0	1
Goodfellas **(5)**	5	5	0	1

$r(i,j)$: 1 *if user j has rated movie i* (0 *otherwise*)

$x^{(i)}$: *feature vector for movie i* (*first index is always* 1)

$\theta^{(j)}$: *parameter vector for user j*

$y^{(i,j)}$: *actual rating by user j on movie i* (*if defined*)

$(\theta^{(j)})^T x^{(i)}$: *predicted rating by user j on movie i*

$$r(1,1) = 1$$

$$r(4,2) = 0$$

$$x^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\theta^{(1)} = \begin{bmatrix} 5 \\ 0 \\ 5 \end{bmatrix}$$

$$y^{(1,1)} = 5$$

$$y^{(1,2)} = 1$$

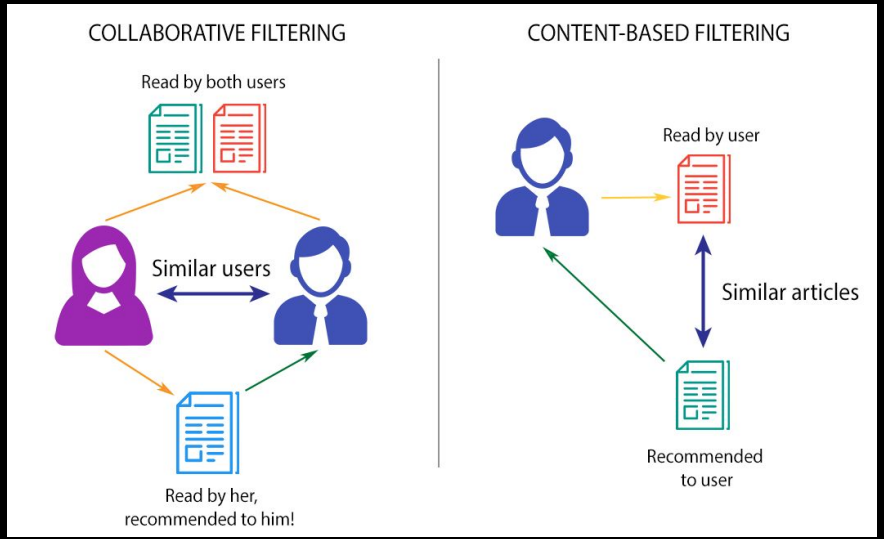
$$y^{(2,1)} = undefined$$

Approaches to Recommendations

- **Content-based**: recommends products that are similar products to the ones that the user has liked in the past

- **Collaborative**: recommends products that similar users have liked in the past

- **Hybrid**: a combination of content-based and collaborative



Content-based Filtering

UW DSC.

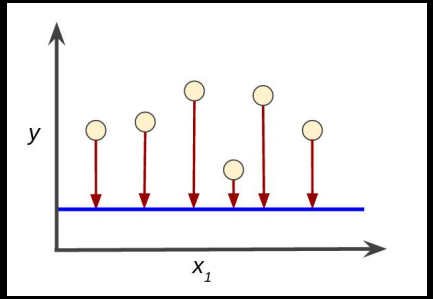
- **Content-based**: recommends products that are similar products to the ones that the user has liked in the past
  - Using the **User - Product** and **Product - Product** relationships

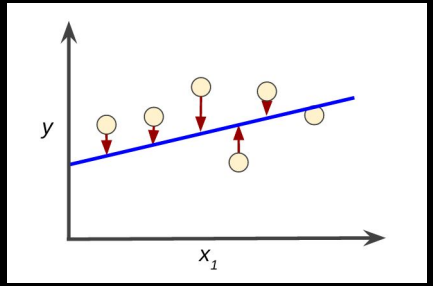
- Types of data used
  - **Product attribute** (necessary)
  - **User behaviour** (at least one “rating”)

- Assume you are given the **feature vector**, we must learn the **user/parameter vector** using linear regression!

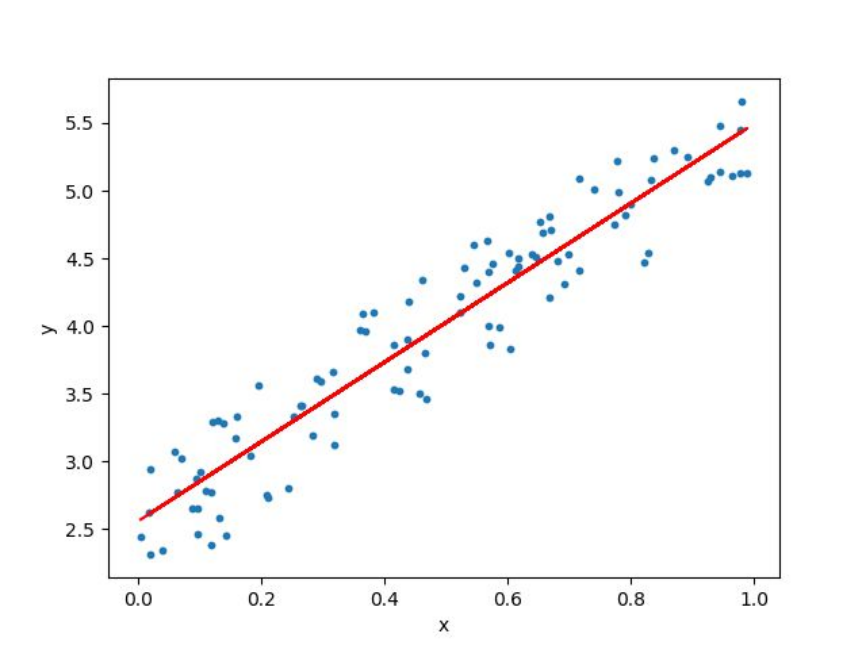
Loss & Optimizers

1. **Loss**
   a. An indicator of how bad the model's prediction was on a single sample
2. **Loss/Cost Function**
   a. A function which calculates the error in the model across all samples
3. **Optimizers**
   a. Algorithms which aim to minimize loss by changing the training parameters



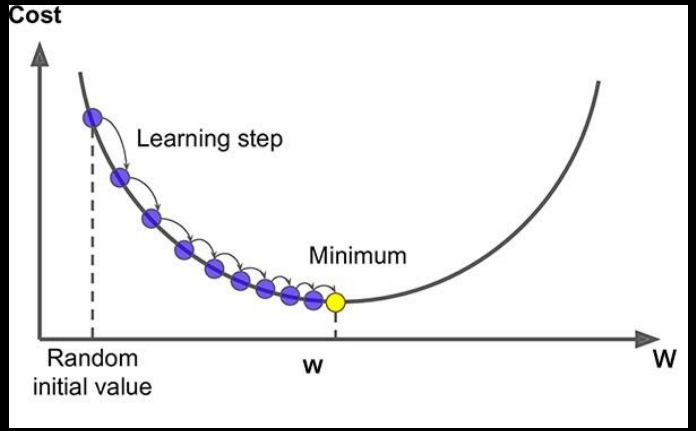


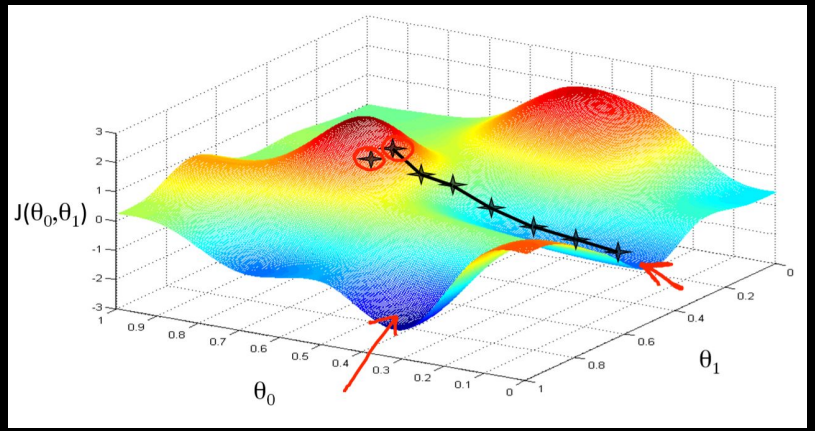
Mean Squared Error (Loss Function)



$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y}_i)^2$$

Gradient Descent (optimizer)





repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

Content-based Filtering Optimization

UW DSC.

Movie	User 1	User 2	x_1 (comedy)	x_2 (drama)
Step Brothers **(1)**	5	1	1	0
Anchorman **(2)**	?	1	1	0
The Other Guys **(3)**	?	1	1	0
Taxi Driver **(4)**	4	?	0	1
Goodfellas **(5)**	5	5	0	1

Dataset with explicit ratings and product attributes

$r(i,j)$: 1 *if user j has rated movie i* (0 *otherwise*)

$x^{(i)}$: *feature vector for movie i* (*first index is always* 1)

$\theta^{(j)}$: *parameter vector for user j*

$y^{(i,j)}$: *actual rating by user j on movie i* (*if defined*)

$(\theta^{(j)})^T x^{(i)}$: *predicted rating by user j on movie i*

$$\min_{\theta^{(j)}} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

Given feature vectors, we are learning the user/parameter vectors

Collaborative Filtering

UW DSC.

- **Collaborative**: recommends products that similar users have liked in the past
  - Using the **User - Product** and **User - User** relationships

- Types of data used
  - **User behaviour** (need many “ratings”)

- Assume you are given the **user/parameter vector**, we must learn the **feature vector** using linear regression!

Collaborative Filtering Optimization

UW DSC.

Movie	User 1	User 2	x_1 (comedy)	x_2 (drama)
Step Brothers **(1)**	5	1	?	?
Anchorman **(2)**	?	1	?	?
The Other Guys **(3)**	?	1	?	?
Taxi Driver **(4)**	4	?	?	?
Goodfellas **(5)**	5	5	?	?

Dataset with explicit ratings and without product attributes

$r(i,j)$: 1 *if user j has rated movie i* (0 *otherwise*)

$x^{(i)}$: *feature vector for movie i* (*first index is always* 1)

$\theta^{(j)}$: *parameter vector for user j*

$y^{(i,j)}$: *actual rating by user j on movie i* (*if defined*)

$(\theta^{(j)})^T x^{(i)}$: *predicted rating by user j on movie i*

$$\min_{x^{(i)}} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2$$

Given user/parameter vectors, we are learning the feature vectors

Advantages & Disadvantages

Approach	Advantages	Disadvantages
Content-based	• Doesn't need many ratings from user before recommendations are usable	• Different products that haven't been rated by the user or aren't related to other products don't get much exposure • More data is needed
Collaborative	• Easy to implement • Don't need product attributes/feature vectors	• User's preferences change, so things they may have liked before may not apply • Needs many ratings from user before recommendations are usable

UW
DSC.

Movie Recommendation Notebook

Deep Learning Recommender Systems

UW DSC.

- Deep learning (DL) refers to machine learning techniques which use artificial neural networks (ANNs)
  - UW DSC Neural Networks Workshop: https://bit.ly/3B5vgo1

- DL recommender systems aim to understand the stand relationships discussed and learns the **feature vector** inside the neural networks

- TensorFlow Recommenders Library: https://www.tensorflow.org/recommenders

Resources

- An In-Depth Guide to How Recommender Systems Work: https://builtin.com/data-science/recommender-systems
- Machine Learning by Andrew Ng: https://www.coursera.org/learn/machine-learning/home/welcome
- Movie Recommender Systems by Memphis Meng: https://towardsdatascience.com/movie-recommendation-system-based-on-movielens-ef0df580cd0e
- UWaterloo Data Science Channel: https://www.youtube.com/channel/UCknY88pglf2xz_S72WHIDxg