Advanced Machine Learning

Principal Components Analysis
+
Recommender Systems

Today’s Outline

- PCA Examples + Intuition
- Start Recommender Systems

Big and High-dimensional data

- High dimensional = many many features!

- Ex: MovieLens ratings data
  - ~600 users (rows) vs. ~10,000 movies (columns)

Low dim. representations

- Useful to have lower dimensional representations of the data.

- Helps with:
  - Visualization
  - Computing ML models
    - Compressed data → less memory
    - Fewer features → less time taken in training models
  - Statistical generalizability
    - Low dimensional data often generalizes better
  - Signal/noise decoupling

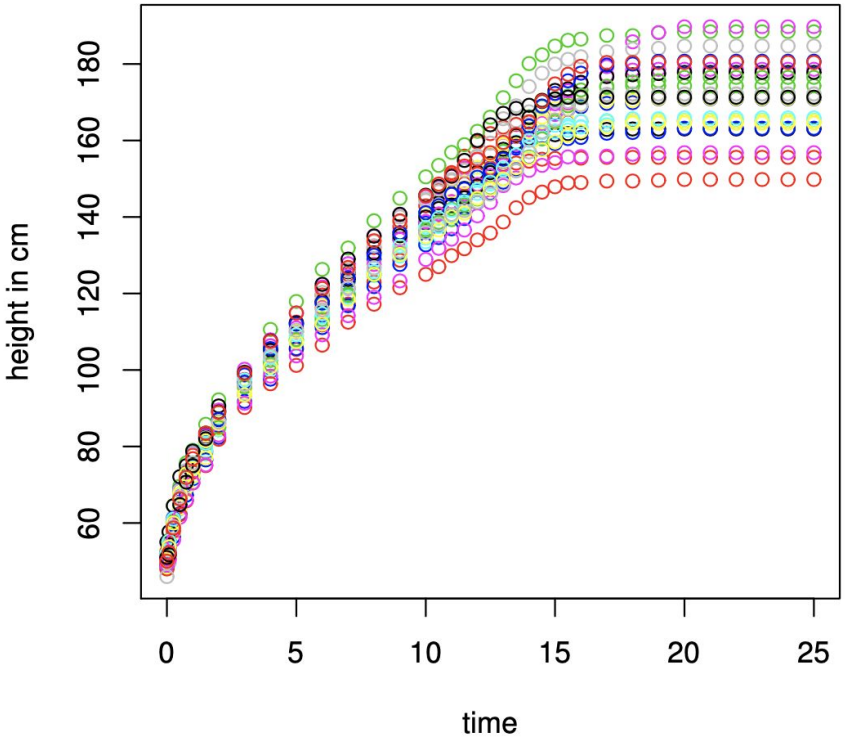
Principal Components Analysis

- PCA = unsupervised learning algorithm for learning *low-dimensional representations*
- Based on the variance structure of the data

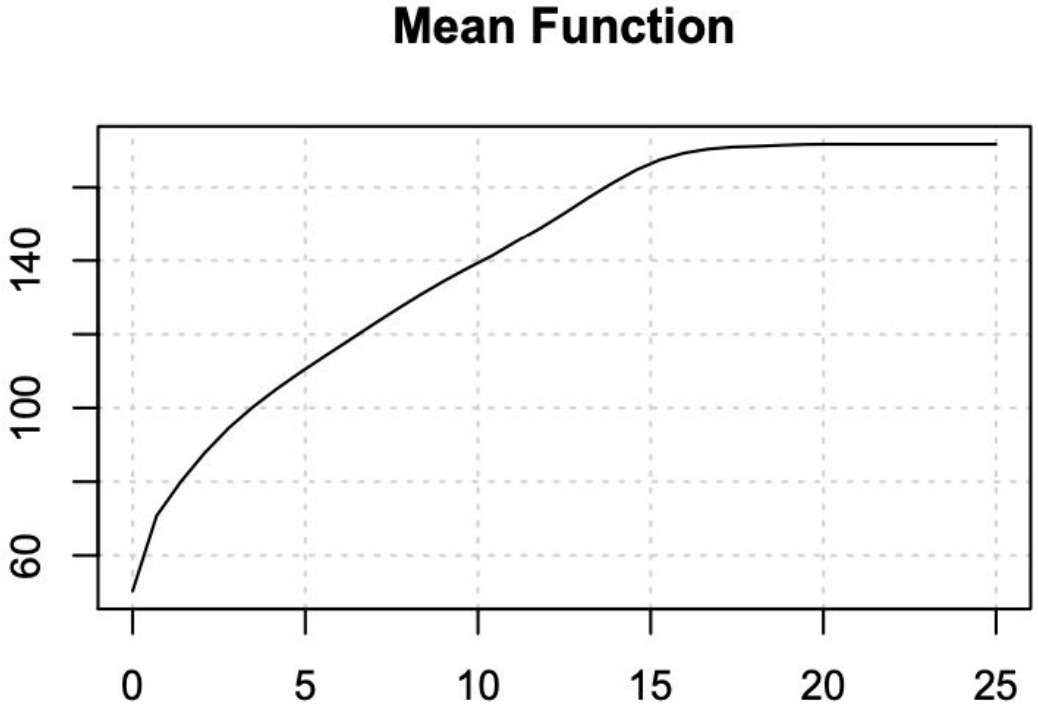
Principal Components Analysis

- Relies on a change of basis idea
- Useful when the datacloud lies in an intrinsically smaller subspace than the “ambient space” in which it was observed

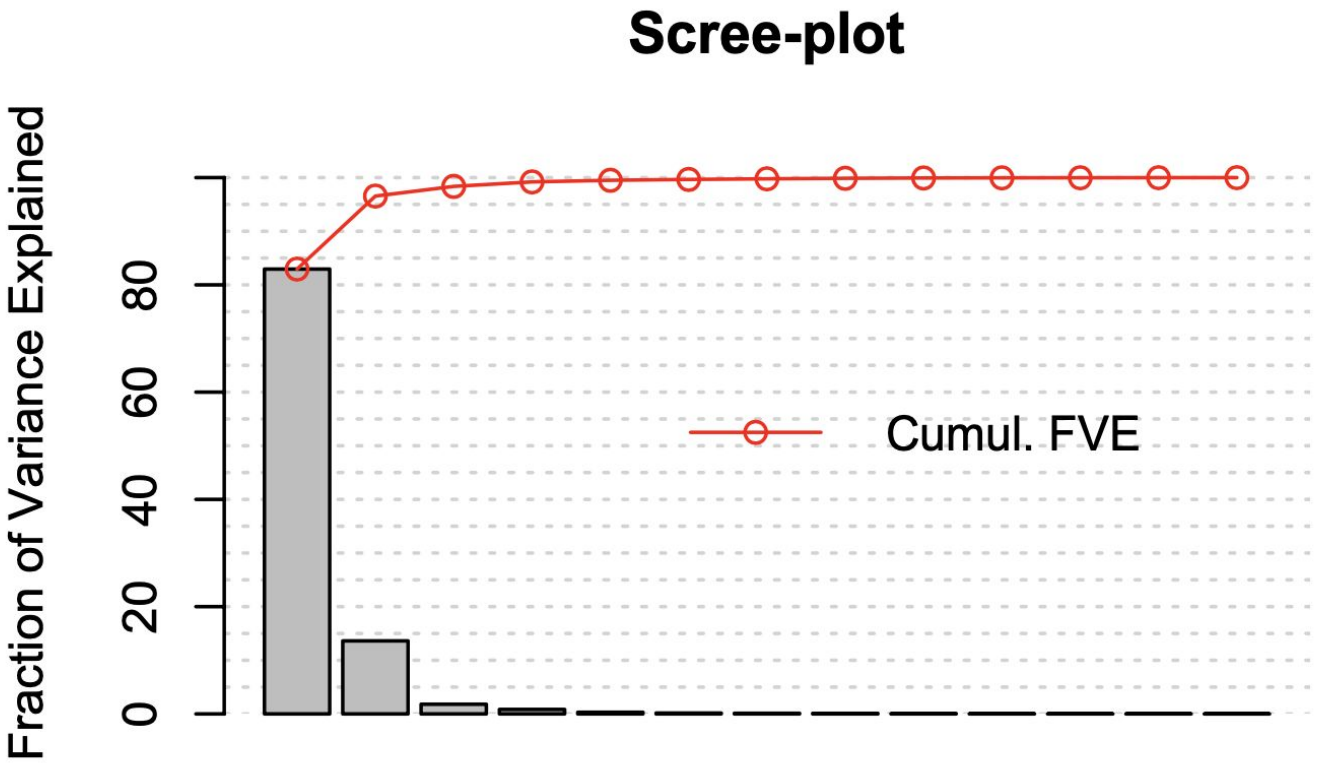
Ex: Growth Curves



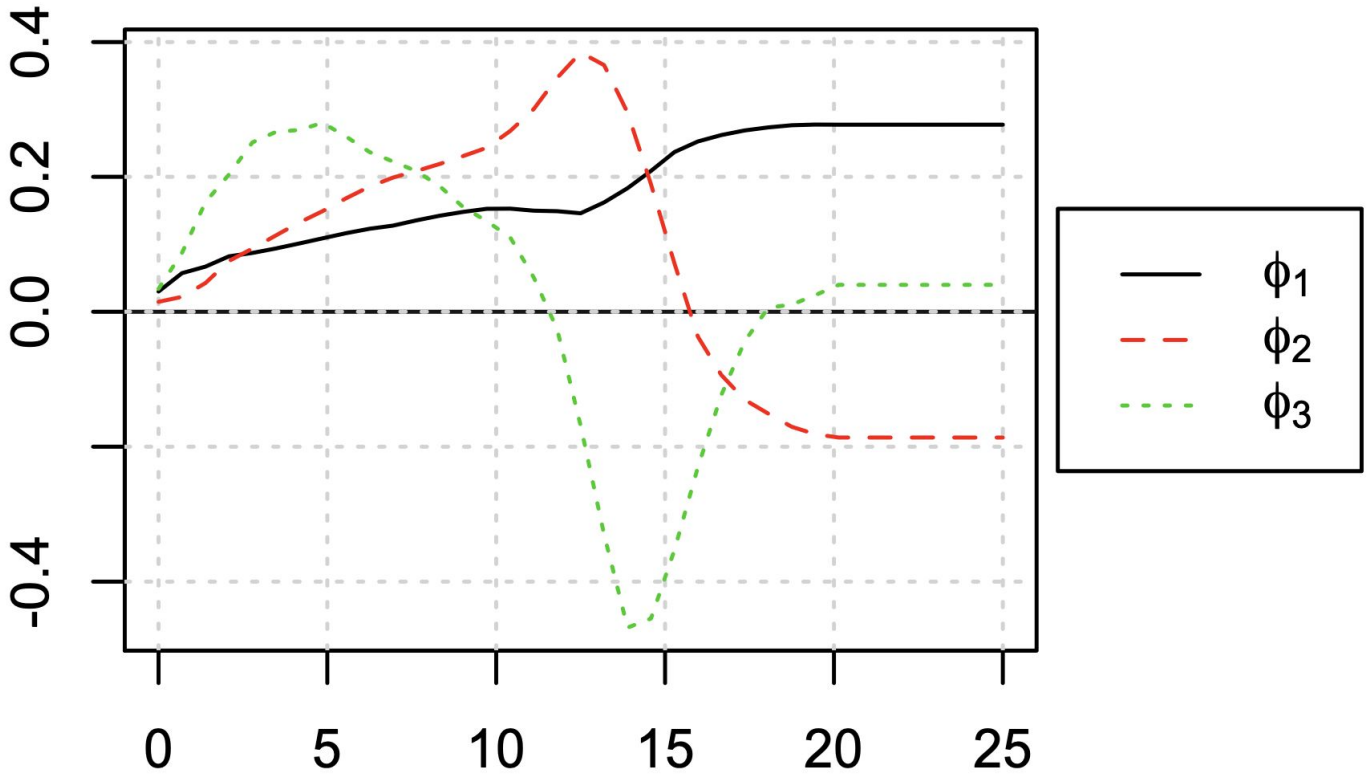
Step 1: Calculate mean & center data



Step 2: Calculate covariance & look at eigenvalue decay

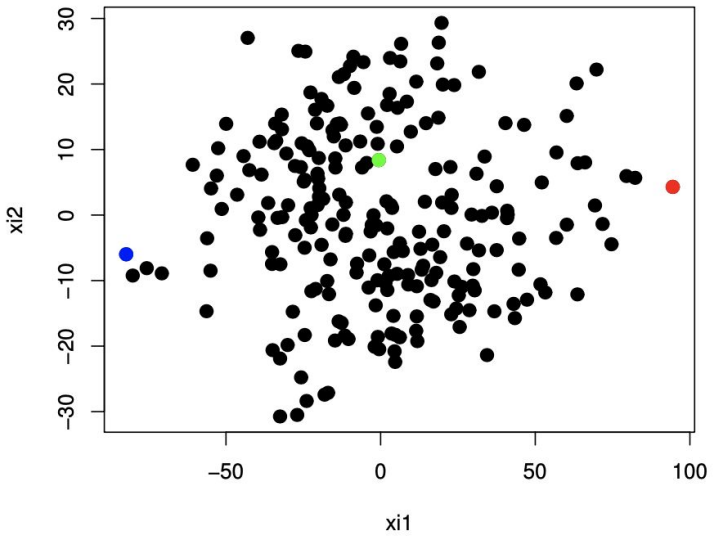


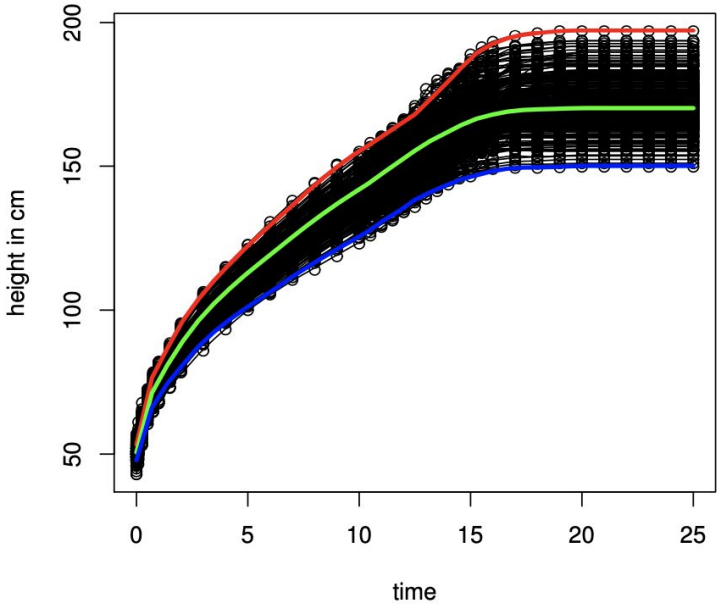
Step 3: Fix K & look at the first K principal directions / eigenvectors



Step 4: Project centered data onto principal directions

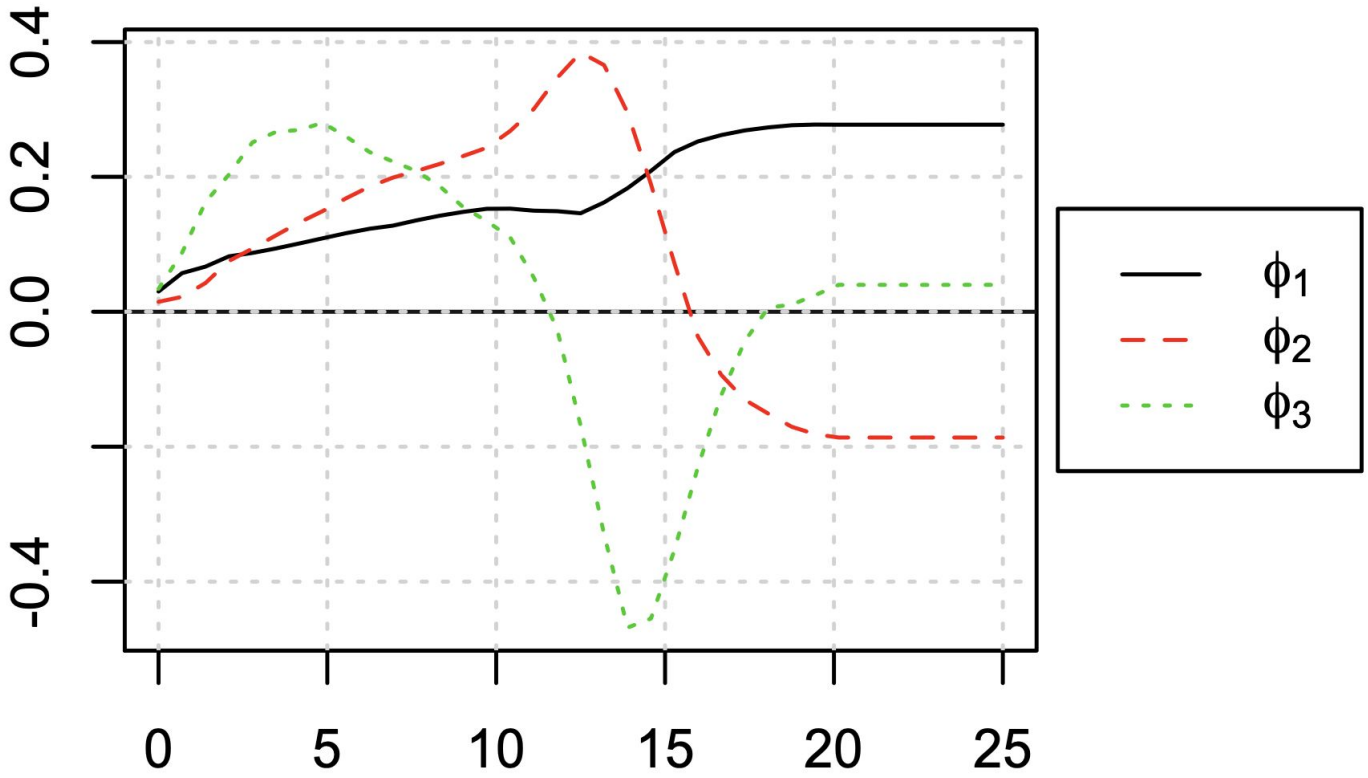
We can think of principal directions as ***modes of variation***!



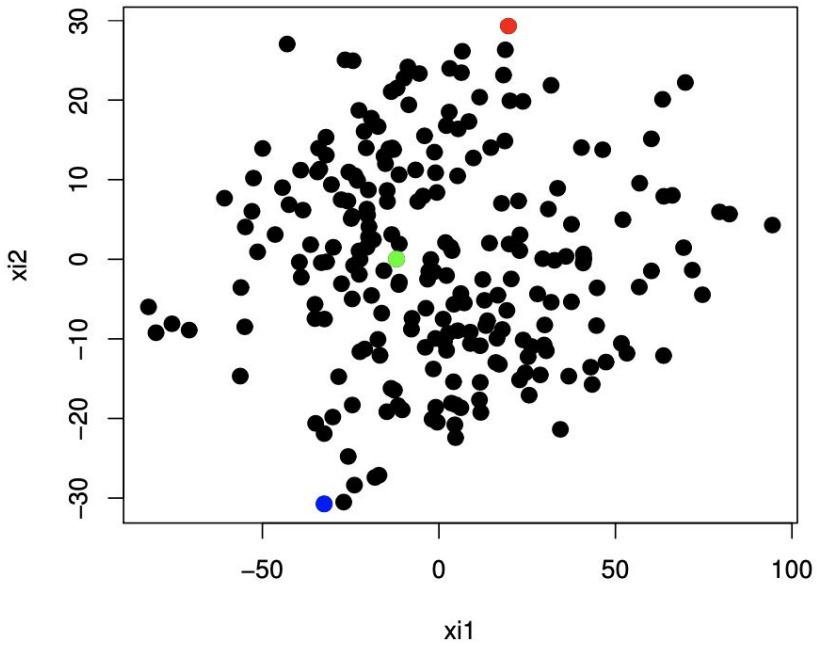


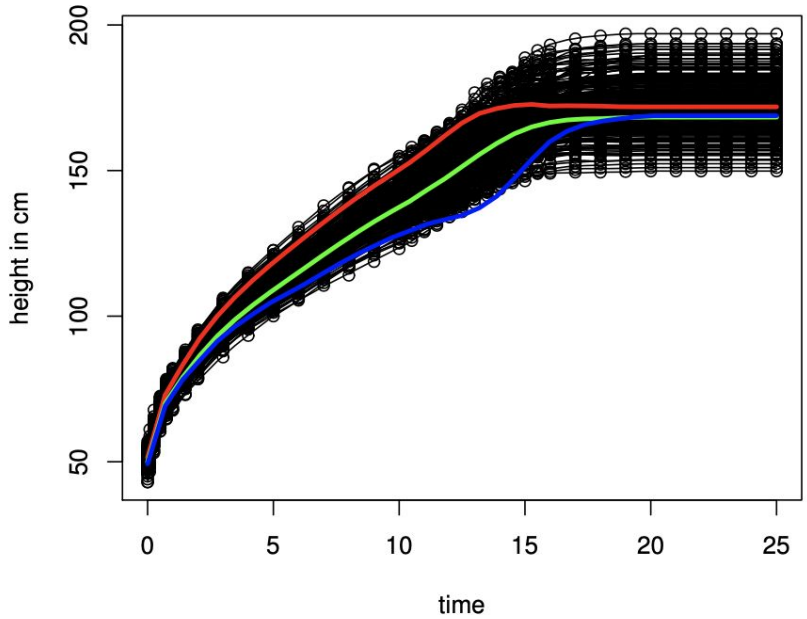
First mode of variation: general height overall. High PC1 = taller kid, low PC1 = shorter kid

Look at PC 2 now: second mode of variation



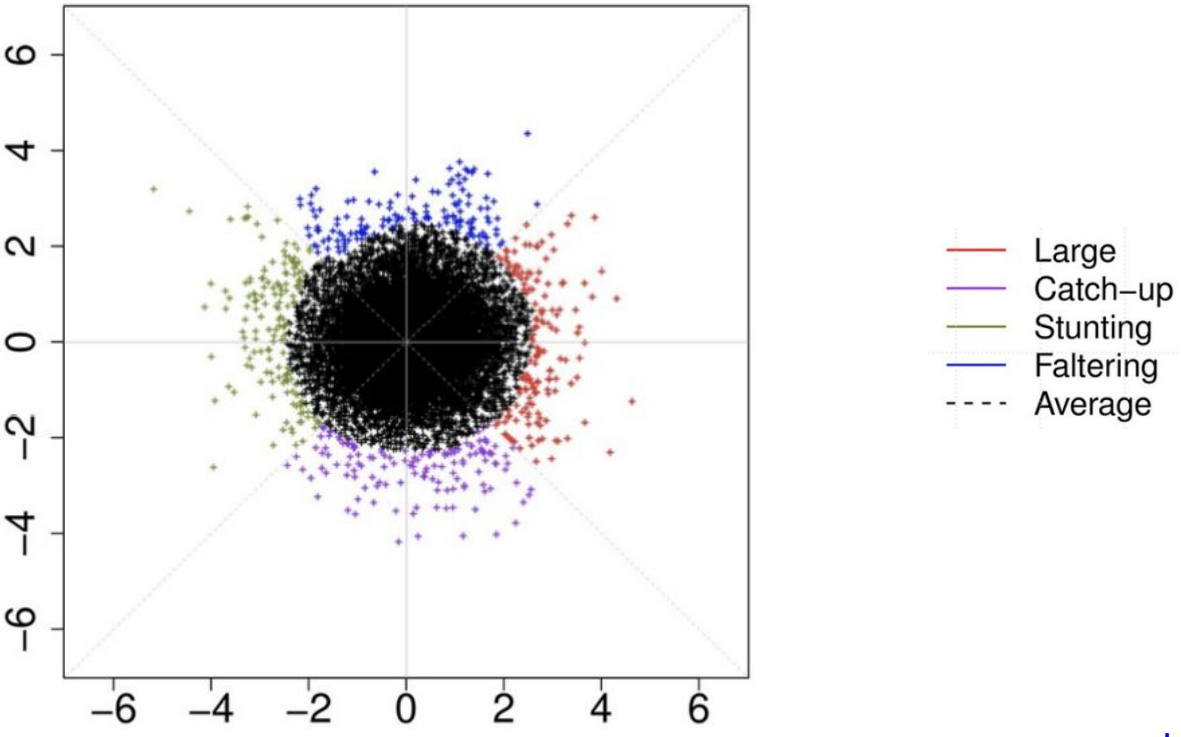
Second mode of variation





Second mode of variation: Contrast in pre-adolescent vs. post-adolescent heights.
High PC1 = less of a growth spurt, Low PC1 = big growth spurt!

Archetypes from PCA



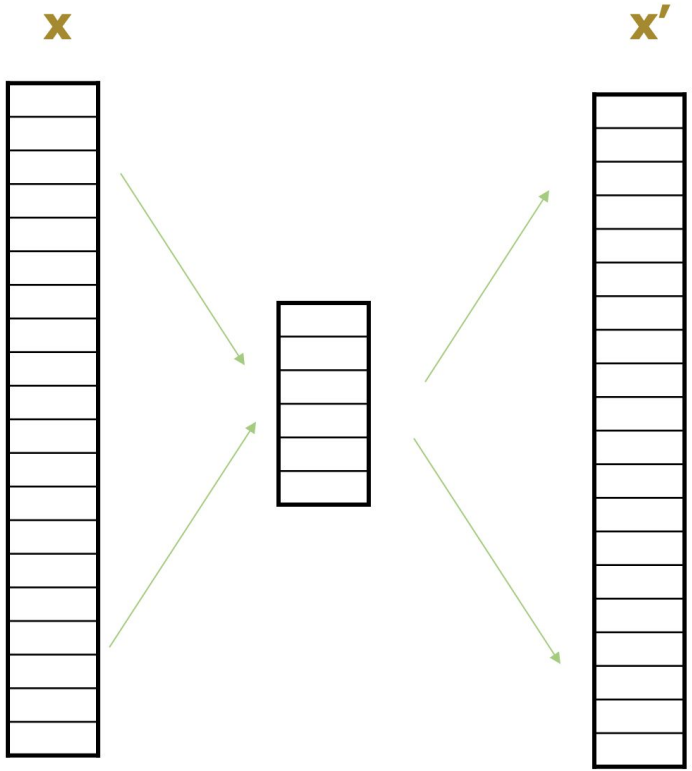
Han et al. 2018

Noise Filtering Ex for Images

Noisy Image

Ex. from Matt Gormley, CMU

Idea of an Autoencoder



Ex. from Matt Gormley, CMU

Denoised Image
(15 principal components)

Ex. from Matt Gormley, CMU

What is gained/lost by using the low rank approximation?

+ : got rid of the noisy pixels

- : lost some of the resolution given by the PCs which had small eigenvalues

Recommender Systems

Recommender Systems

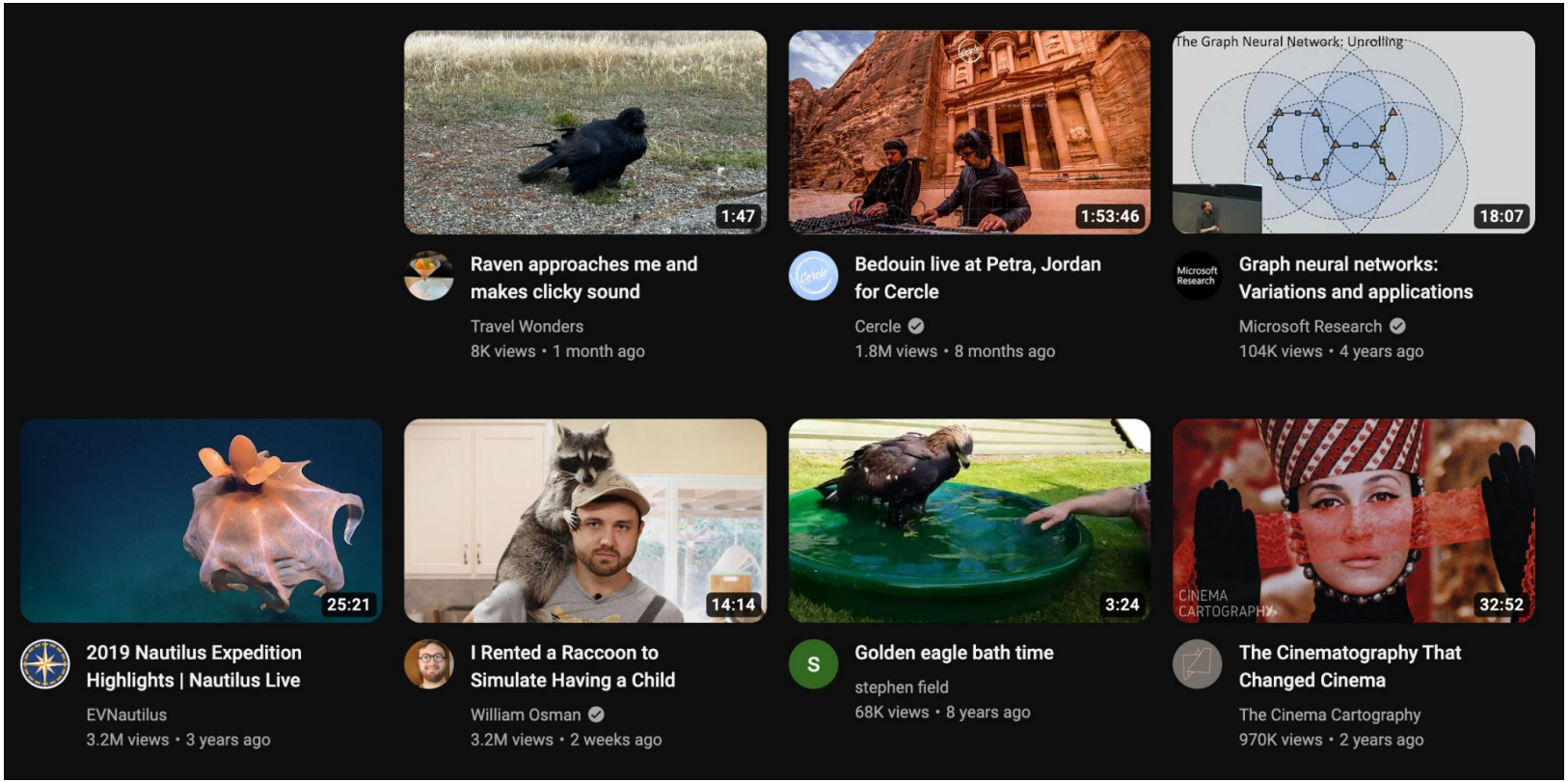
Required reading:

- Chapter 9 from “Mining of Massive Datasets." Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman
  - (focus on Collaborative Filtering part)

Optional:

- https://engineering.quora.com/Machine-Learning-at-Quora
- https://www.youtube.com/watch?v=bLhq63ygoU8
- https://www.youtube.com/watch?v=zzTbptEdKhY

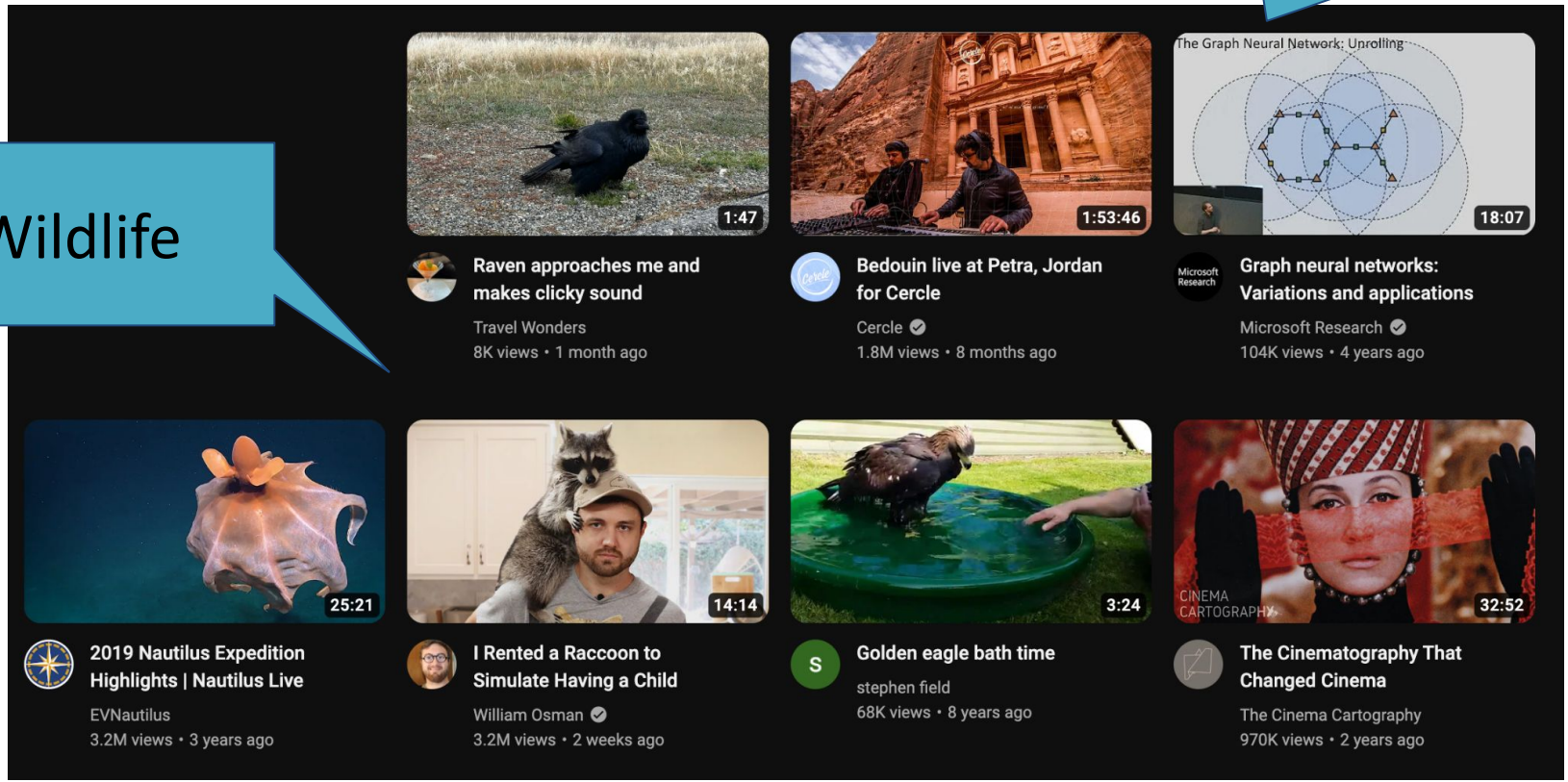
Motivation


1:47
Raven approaches me and makes clicky sound
Travel Wonders
8K views • 1 month ago
1:53:46
Bedouin live at Petra, Jordan for Cercle
Cercle
1.8M views • 8 months ago
The Graph Neural Network: Unrolling
18:07
Graph neural networks: Variations and applications
Microsoft Research
104K views • 4 years ago
25:21
2019 Nautilus Expedition Highlights | Nautilus Live
EVNautilus
3.2M views • 3 years ago
14:14
I Rented a Raccoon to Simulate Having a Child
William Osman
3.2M views • 2 weeks ago
3:24
Golden eagle bath time
stephen field
68K views • 8 years ago
CINEMA CARTOGRAPHY
32:52
The Cinematography That Changed Cinema
The Cinema Cartography
970K views • 2 years ago

Motivation

Machine learning

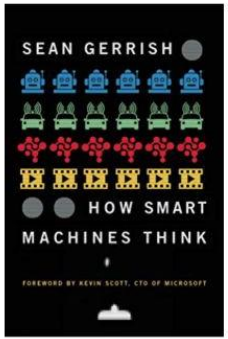
Wildlife



Motivation

Customers who bought this item also bought

Page



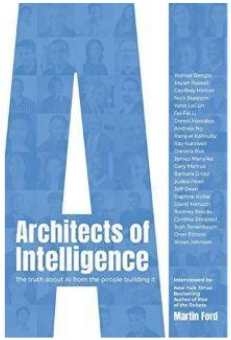
How Smart Machines Think (The MIT Press)

Sean Gerrish

★★★★★ 1

Hardcover

$25.15 ✓prime



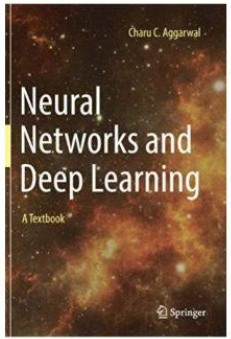
Architects of Intelligence: The truth about AI from the people building it

› Martin Ford

★★★★½ 23

Paperback

$22.49 ✓prime



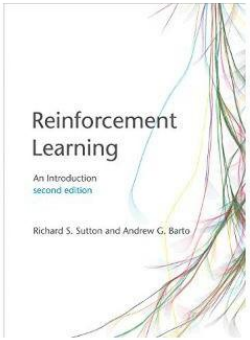
Neural Networks and Deep Learning: A Textbook

› Charu C. Aggarwal

★★★★★ 12

Hardcover

$55.92 ✓prime



Reinforcement Learning: An Introduction (Adaptive Computation and…

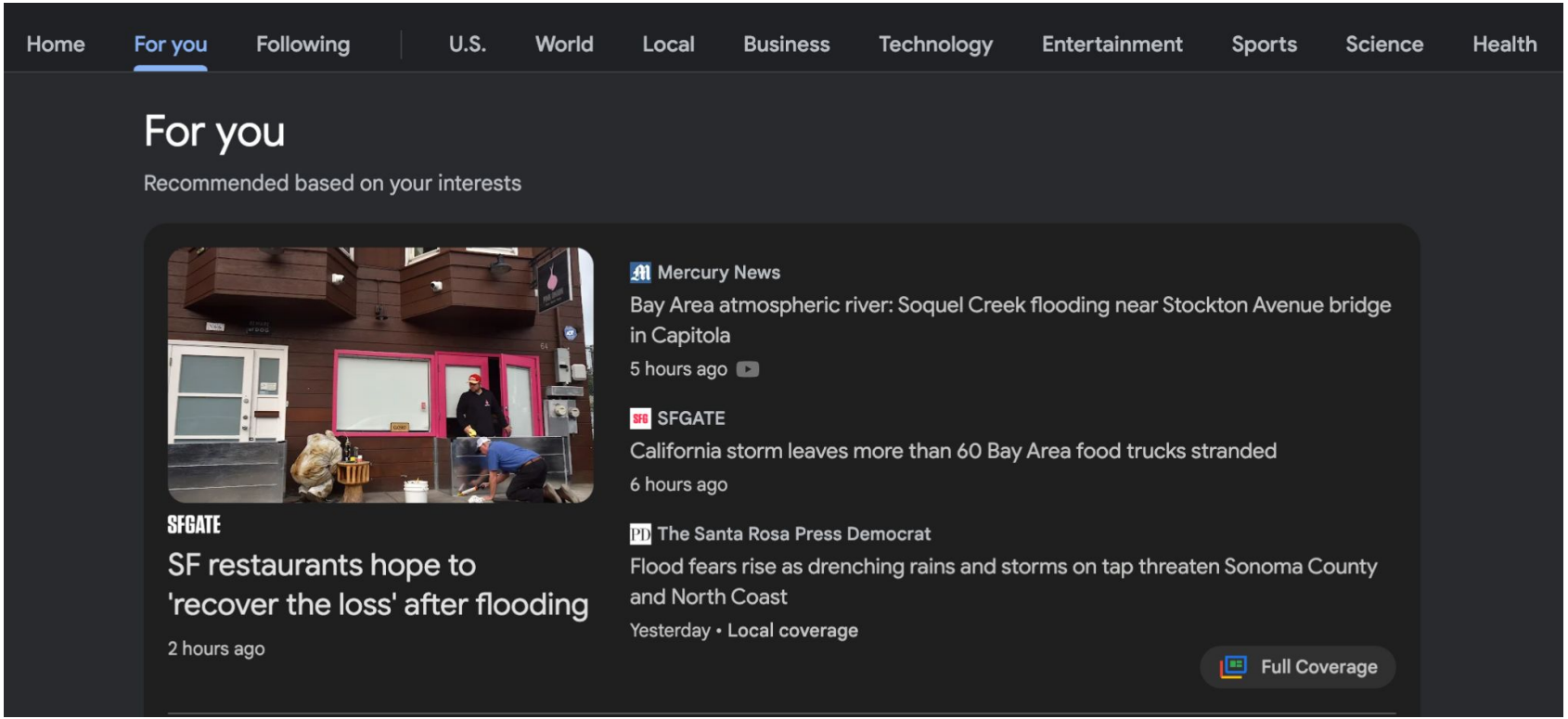
Richard S. Sutton

★★★★★ 6

Hardcover

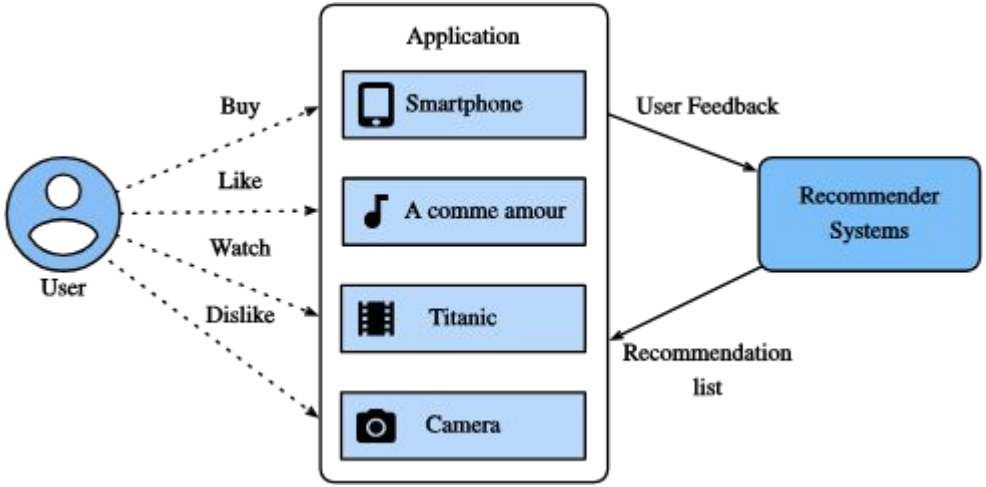
$51.43 ✓prime

Motivation


Home
For you
Following
U.S.
World
Local
Business
Technology
Entertainment
Sports
Science
Health
For you
Recommended based on your interests
SFGATE
SF restaurants hope to 'recover the loss' after flooding
2 hours ago
Mercury News
Bay Area atmospheric river: Soquel Creek flooding near Stockton Avenue bridge in Capitola
5 hours ago
SFGATE
California storm leaves more than 60 Bay Area food trucks stranded
6 hours ago
The Santa Rosa Press Democrat
Flood fears rise as drenching rains and storms on tap threaten Sonoma County and North Coast
Yesterday • Local coverage
Full Coverage

Recommender System:

Application that provides personalized recommendations to users about content they may be interested in.



https://towardsdatascience.com/build-your-own-recommender-system-within-5-minutes-30dd40388fbf

Recommender Systems:

- **Netflix** - shows/movies
- **Amazon -** products
- **YouTube -** videos
- **Google Scholar** - articles
- **Spotify** - songs
- **Hinge** - people… lol

What other companies/ use recommender systems?

What are they recommending?

Value of recommendations

- Netflix: 75% of what people watch is from some sort of recommendation
- YouTube: reports that 60% of the clicks on the home screen are on the recommendations
- Google play: 40% of app installs come from recommendations

Measuring the Business Value of Recommender Systems. Jannach and Jugovac.

Types of Recommender Systems

- **Content-based systems:**
  - uses similarity between items to recommend items similar to what a user likes
  - analysis of content attributes of the ***items***

- **Collaborative filtering:**
  - uses similarities between users and items simultaneously to provide recommendations
  - uses feedback of ***multiple users*** (e.g. ratings)

Content-based or Collaborative Filtering?

- The owner of a new bookstore in San Francisco wants to recommend books to users. This recommendation will be made on the receipt, right after the a customer purchased a book.

- You run an online furniture store and have item ratings from many users. You want to use this to identify what items are "similar" to each other.

Content-based or Collaborative Filtering?

- The owner of a new bookstore in San Francisco wants to recommend books to users. This recommendation will be made on the receipt, right after the a customer purchased a book. **CB**

- You run an online furniture store and have item ratings from many users. You want to use this to identify what items are "similar" to each other. **CF**

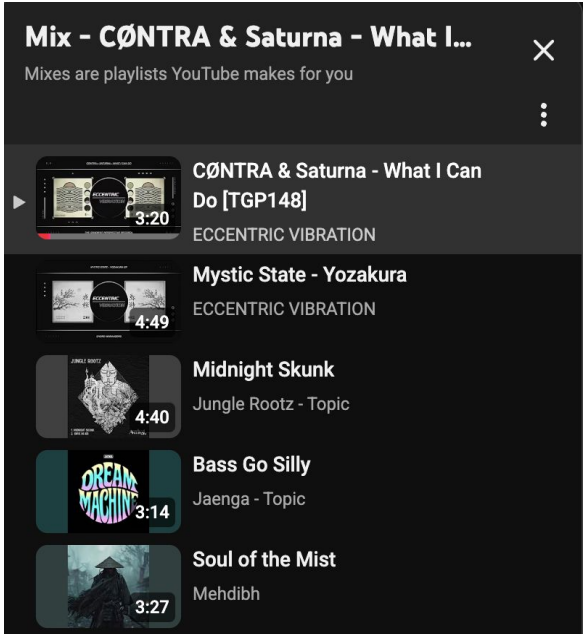
Types of user feedback

- Explicit feedback:
  - Ratings provided by users
  - E.g. 1-5 star system, thumbs up/down

Types of user feedback

- Implicit feedback:
  - Infer users preferences by monitoring actions
  - Purchases, clicks, navigation history, time spent on page

- Questions of quality?



The recommendation problem

- **Users** (request content)
- **Items** (movies, news stories)
- Context (device, location, time)
- Interface (phone, tablet, computer)

Users and Items, Utility Matrix

Items: movies, videos, stories, songs, books

Utility matrix:

- Degree of preference that a user has for an item.
  Example: 1-5 scale ratings of movies
- Very sparse (~1%)

http://www.slideshare.net/xamat/recommender-systems-machine-learning-summer-school-2014-cmu

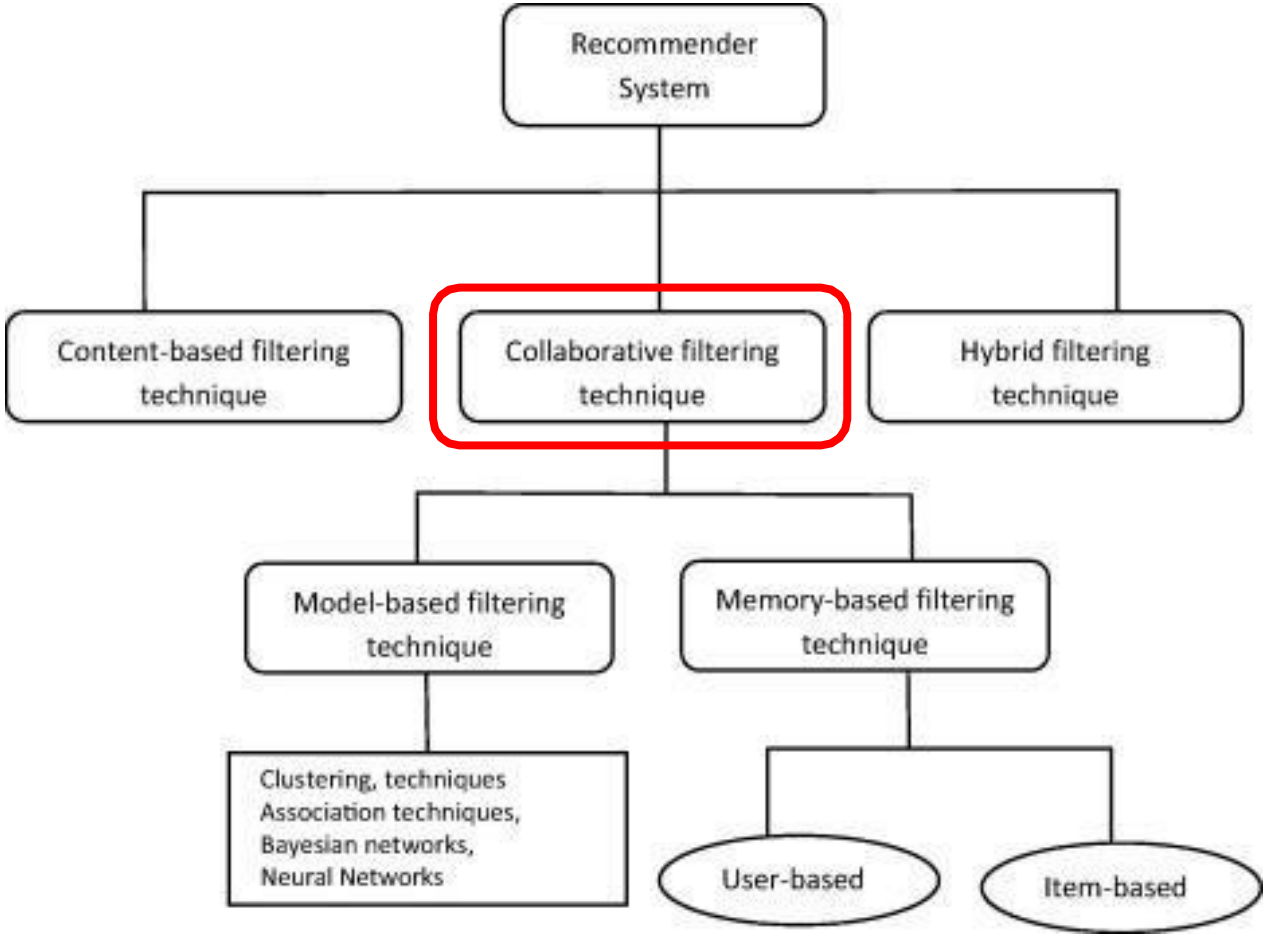
Example in tabular form

Training data

user	movie	date	score
1	21	5/7/02	1
1	213	8/2/04	5
2	345	3/6/01	4
2	123	5/1/05	4
2	768	7/15/02	3
3	76	1/22/01	5
4	45	8/3/00	4
5	568	9/10/05	1
5	342	3/5/03	2
5	234	12/28/00	2
6	76	8/11/02	5
6	56	6/15/03	4

Test data

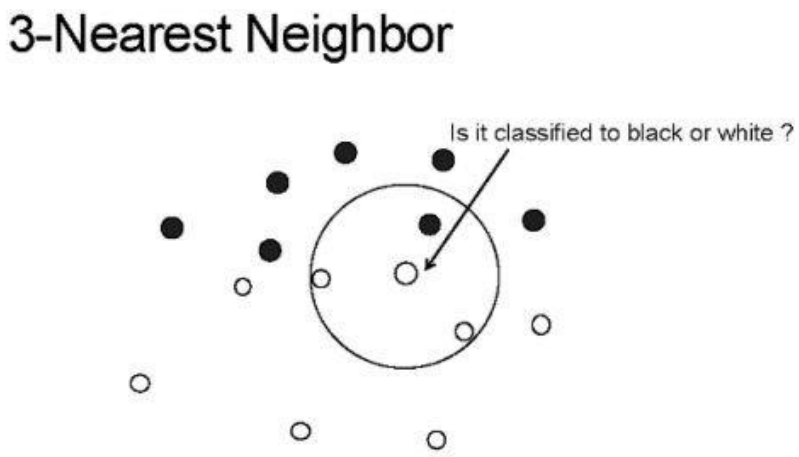
user	movie	date	score
1	62	1/6/05	?
1	96	9/13/04	?
2	7	8/18/05	?
2	3	11/22/05	?
3	47	6/13/02	?
3	15	8/12/01	?
4	41	9/1/00	?
4	28	8/27/05	?
5	93	4/4/05	?
5	74	7/16/03	?
6	69	2/14/04	?
6	83	10/3/03	?



https://ars.els-cdn.com/content/image/1-s2.0-S1110866515000341-gr2.jpg

Collaborative filtering methods

- **Memory based methods**
  - Memorizes the utility matrix
  - K-nearest neighbor approach
  - Tends to be slow
- **Model based method**
  - Fits a model
  - Tends to have better performance



Nearest Neighbor / Memory-based

- User-based vs. Item-Based
- Uses all data to make predictions
  - The data is the model
- Oldest CF method
- Doesn’t scale well

Matrix Factorization (Model-based)

- Netflix Prize competition in 2006
- Funk SVD
- Check out Section 2.2 of this paper for retrospective context
- Optional: blog on SVD's connection to quantum entanglement

Matrix Factorization (Model-based)

P_k	Dim1	Dim2
Alice	0.47	-0.30
Bob	-0.44	0.23
Mary	0.70	-0.06
Sue	0.31	0.93

Q_k^T					
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

Matrix Factorization (Model-based)

- Each user is represented by a K-dimensional vector (embedding)
- Each movie (item) is represented by a K-dimensional vector
- Hypothetical: Dim1 = “Romance”, Dim2 = “Themes of destiny”

http://slideplayer.com/slide/9865426/

Matrix Factorization model

User matrix: U is $n_u \times K$

Item matrix: V is $n_m \times K$

Predicted ratings: $\hat{Y} = UV^T$

Matrix Factorization model

Ratings: y_{ij} where $i \in 1, \ldots n_u$ and $j \in 1, \ldots n_m$

User embedding: i^{th} row of U

Item embedding: j^{th} row of V

Predicted ratings: $$\hat{y}_{ij} = (UV^T)_{ij} = \sum_s u_{is} v_{js}$$

Matrix Factorization model (Example)

Here is a toy example in witch we have 7 users and 5 movies $K = 2$.

$$\begin{bmatrix} 5 & 1 & 4 & 5 & 1 \\ & 5 & 2 & 1 & 4 \\ 1 & 4 & 1 & 1 & 2 \\ 4 & 1 & 5 & 5 & 4 \\ 5 & 3 & 3 & & 4 \\ 1 & 5 & 1 & 1 & 1 \\ 5 & 1 & 5 & 5 & 4 \end{bmatrix} \approx \begin{bmatrix} 0.2 & 3.4 \\ 3.6 & 1.0 \\ 2.6 & 0.6 \\ 0.9 & 3.7 \\ 2.0 & 3.4 \\ 2.9 & 0.5 \\ 0.8 & 3.9 \end{bmatrix} \times \begin{bmatrix} 0.0 & 1.5 & 0.1 & 0.0 & 0.7 \\ 1.3 & 0.0 & 1.2 & 1.4 & 0.7 \end{bmatrix}$$

Exercise: Compute the prediction of the two blank elements.

$$\hat{y}_{ij} = u_i \cdot v_j$$

$$\begin{bmatrix} 5 & 1 & 4 & 5 & 1 \\ y_{21} & 5 & 2 & 1 & 4 \\ 1 & 4 & 1 & 1 & 2 \\ 4 & 1 & 5 & 5 & 4 \\ 5 & 3 & 3 & y_{54} & 4 \\ 1 & 5 & 1 & 1 & 1 \\ 5 & 1 & 5 & 5 & 4 \end{bmatrix} \approx \begin{bmatrix} 0.2 & 3.4 \\ 3.6 & 1.0 \\ 2.6 & 0.6 \\ 0.9 & 3.7 \\ 2.0 & 3.4 \\ 2.9 & 0.5 \\ 0.8 & 3.9 \end{bmatrix} \times \begin{bmatrix} 0.0 & 1.5 & 0.1 & 0.0 & 0.7 \\ 1.3 & 0.0 & 1.2 & 1.4 & 0.7 \end{bmatrix}$$

How to learn model parameters?

- How many parameters do we have?
- How can we learn them?

$$\hat{y}_{ij} = u_i \cdot v_j$$

y_{ij} where $i \in 1, \ldots n_u$ and $j \in 1, \ldots n_m$

How to learn model parameters?

- How many parameters do we have?
  - $(n_u + n_m) \times K$

- How can we learn them?
  - Gradient descent!

Loss function

$$\frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2$$

N is the number ratings in the training set.
r_{ij} be 1 if user i rated item j, 0 otherwise.

What problem do we want to solve?

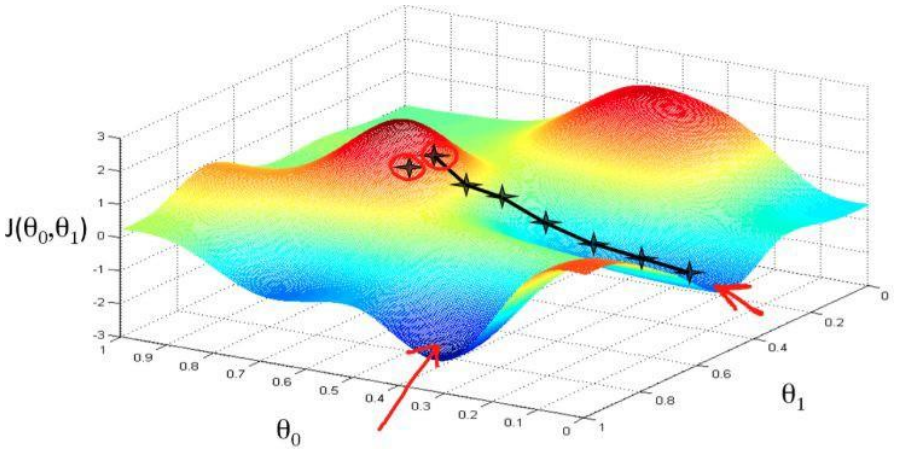
We want to find parameters that minimize the cost (error) function.

$$\min_{u_i, v_j} \frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2$$

This minimization is over **all** users and items (movies)

Gradient Descent

Gradient descent is a iterative optimization algorithm for finding the minimum of a function.



https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

Notation

$$E(w_1, \ldots, w_n)$$

$$\nabla E = \left(\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}\right)$$

Where E is the cost function and $(w_1, \ldots, w_n)$ are the parameters of the model

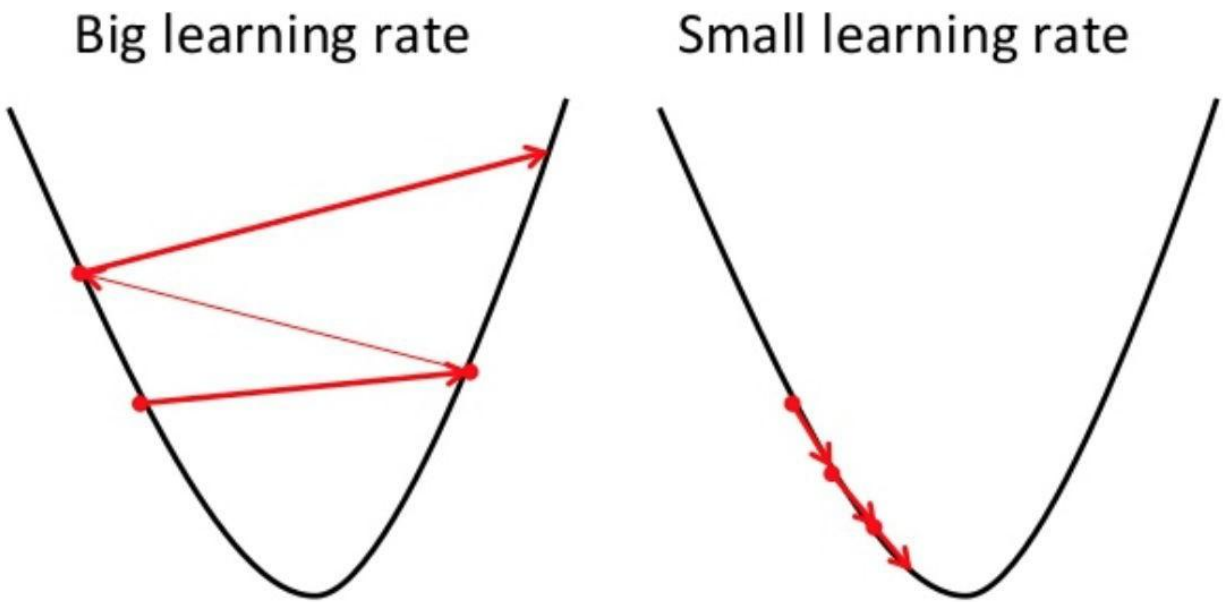
Gradient Descent

$$\min_{w} E(w)$$

$w \leftarrow$ random initialization;
for $i = 1 : max\_iter$ **do**
  $w \leftarrow w - \eta \nabla E(w)$;
end

Learning rate

Gradient Descent - Learning rate



https://www.linkedin.com/pulse/gradient-descent-simple-words-parth-jha/

Exercise:

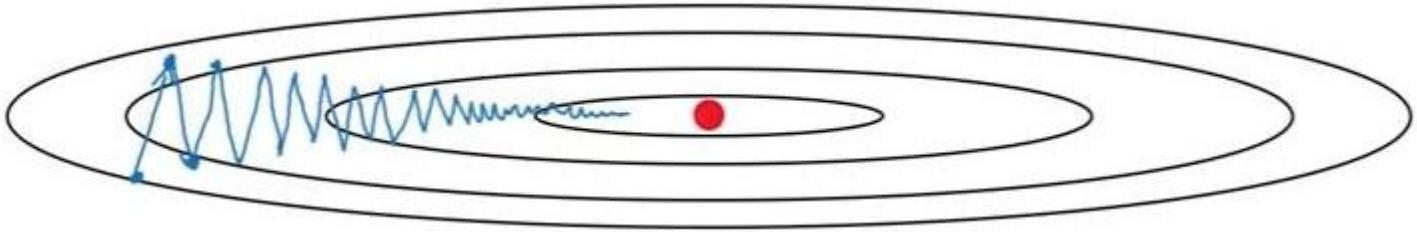
Given a training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$ where $x^{(i)}, y^{(i)} \in \mathbb{R}$, derive the gradient descent equations for linear regression.

$$\hat{y} = a \cdot x + b$$

Gradient descent equations for MF

Gradient Descent with Momentum

Motivation: Gradient descent tends to oscillate and prevent you from using larger learning rates.



https://www.coursera.org/learn/deep-neural-network/lecture/y0m1f/gradient-descent-with-momentum (Andrew Ng)

Gradient Descent with Momentum

$w \leftarrow$ random initialization;

$v \leftarrow 0$;

for $i = 1 : max\_iter$ **do**

$v \leftarrow \beta v + (1 - \beta)\nabla E(w)$;

$w \leftarrow w - \eta v$;

end

momentum term

Gradient Descent with Momentum

$w \leftarrow$ random initialization;

$v \leftarrow 0$;

for $i = 1 : max\_iter$ **do**

  $v \leftarrow \beta v + (1 - \beta)\nabla E(w)$;

  $w \leftarrow w - \eta v$;

end

$\beta = 0$

momentum term

Gradient Descent with Momentum

$w \leftarrow$ random initialization;

$v \leftarrow 0$;

for $i = 1 : max\_iter$ **do**

  $v \leftarrow \beta v + (1 - \beta)\nabla E(w)$;

  $w \leftarrow w - \eta v$;

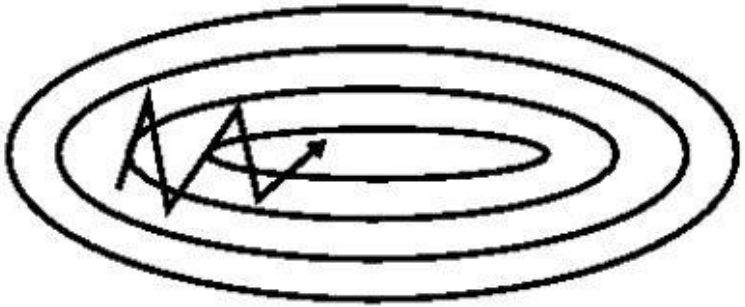
end

$\beta = 0.9$

momentum term

Momentum

(a) SGD without momentum



(b) SGD with momentum

$\beta = 0$

$\beta = 0.9$

https://arxiv.org/pdf/1609.04747.pdf

Avoiding Overfitting

- Similar to regression
- L2 or L1 regularization
- How do we find a good lambda value?
- How would this change **GD** equations?

$$\frac{1}{N} \sum_{(i,j):r_{ij}=1} (y_{ij} - u_i v_j)^2 + \lambda \left(\sum_{i=1}^{n_u} \sum_{k=1}^{K} u_{ik}^2 + \sum_{i=1}^{n_m} \sum_{k=1}^{K} v_{ik}^2 \right)$$

Helpful References

- https://www.sciencedirect.com/science/article/pii/S1110866515000341
- https://research.googleblog.com/2018/01/the-google-brain-team-looking-back-on_12.html
- https://nlp.stanford.edu/manning/talks/Simons-Institute-Manning-2017.pdf
- http://www.cp.jku.at/tutorials/mrs_recsys_2017/slides.pd f