

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)



A visualization of a movie rating matrix. At the top, there is a row of movie posters: SHERLOCK, HOUSE OF CARDS, THE AVENGERS, ALIENS, BREAKING BAD, and THE WALKING DEAD. Below this row is a 6x6 grid representing user ratings. The rows are labeled with user icons, and the columns are labeled with movie icons. The numerical ratings are as follows:

	SHERLOCK	HOUSE OF CARDS	THE AVENGERS	ALIENS	BREAKING BAD	THE WALKING DEAD
User 1	2			4	5	
User 2	5		4			1
User 3			5		2	
User 4		1		5		4
User 5			4			2
User 6	4	5		1		

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

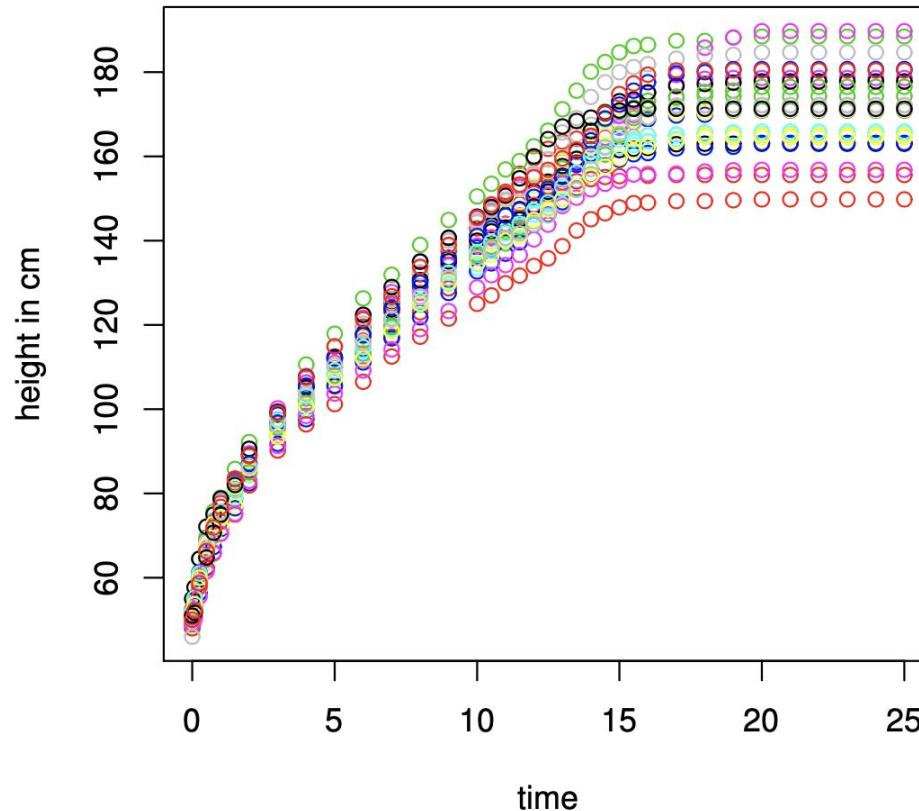
Principal Components Analysis

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

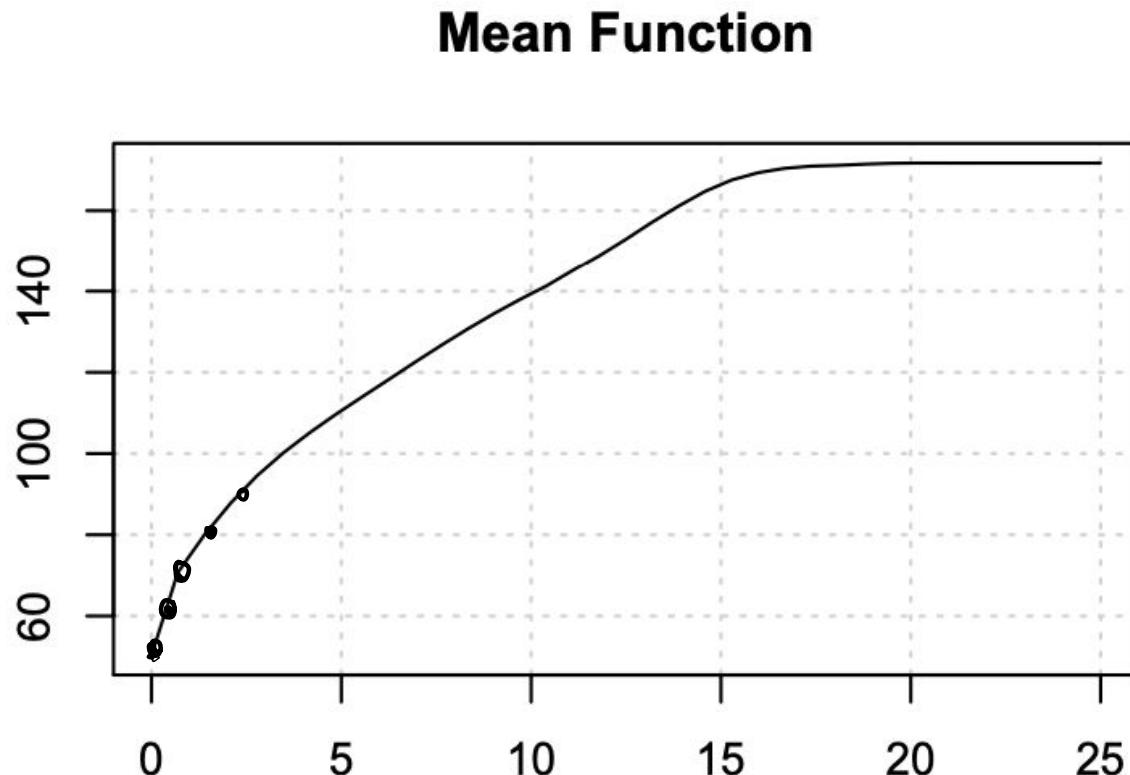
Principal Components Analysis

- 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

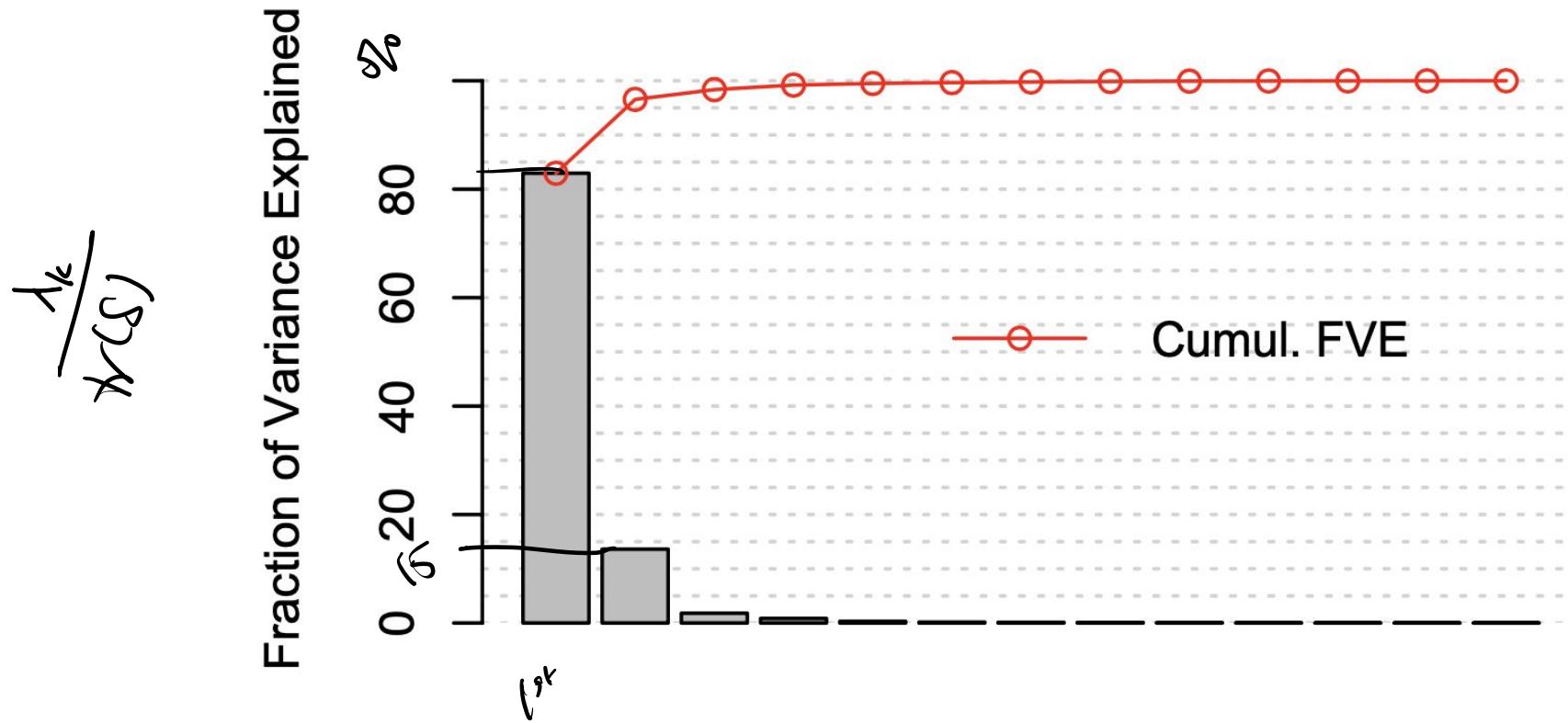


Step 1: Calculate mean & center data



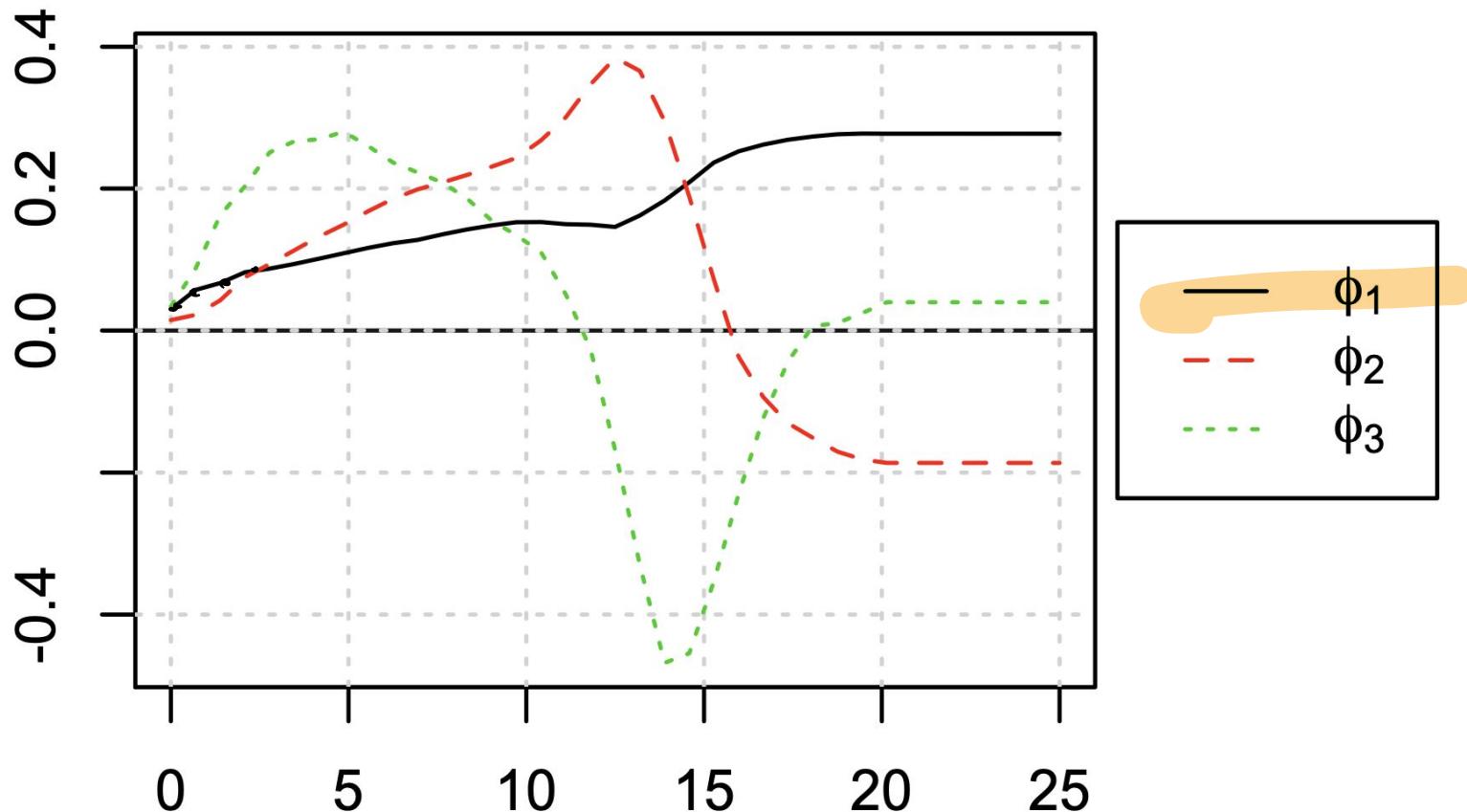
Step 2: Calculate covariance & look at eigenvalue decay

Scree-plot



\leftarrow

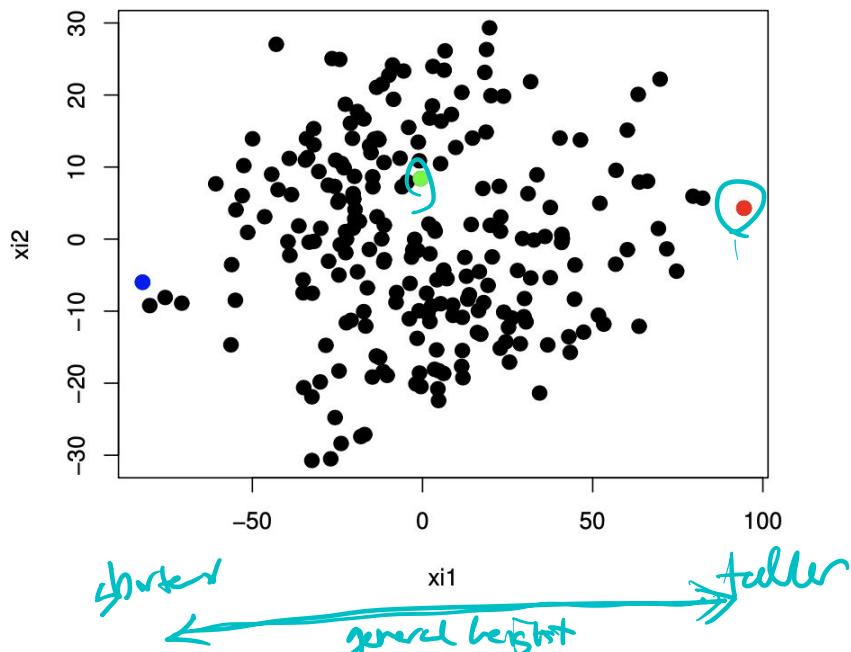
Step 3: Fix K & look at the first K principal directions / eigenvectors



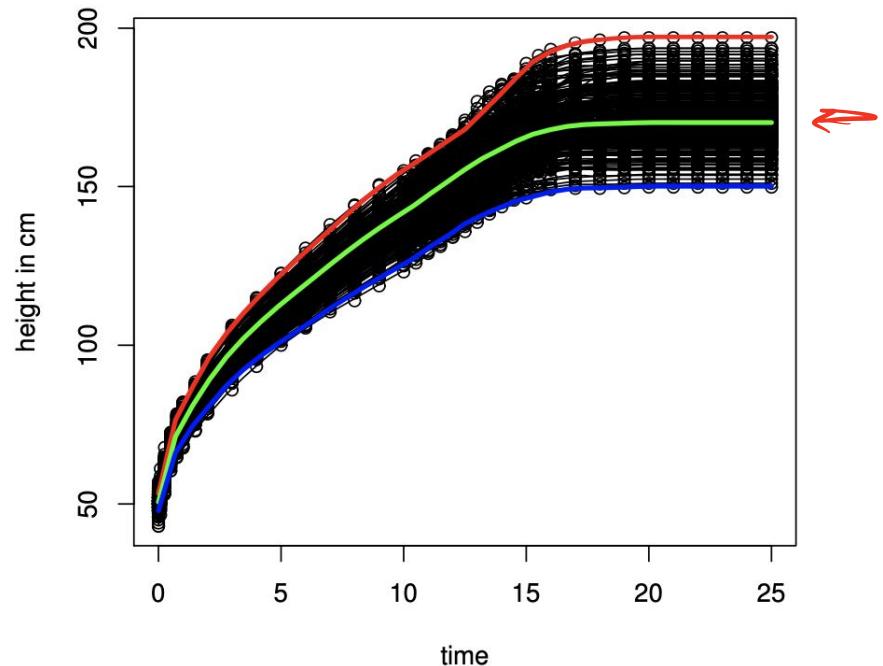
$$\underline{x_i \approx \bar{x} + \xi_{ii} \phi_1}$$

Step 4: Project centered data onto principal directions

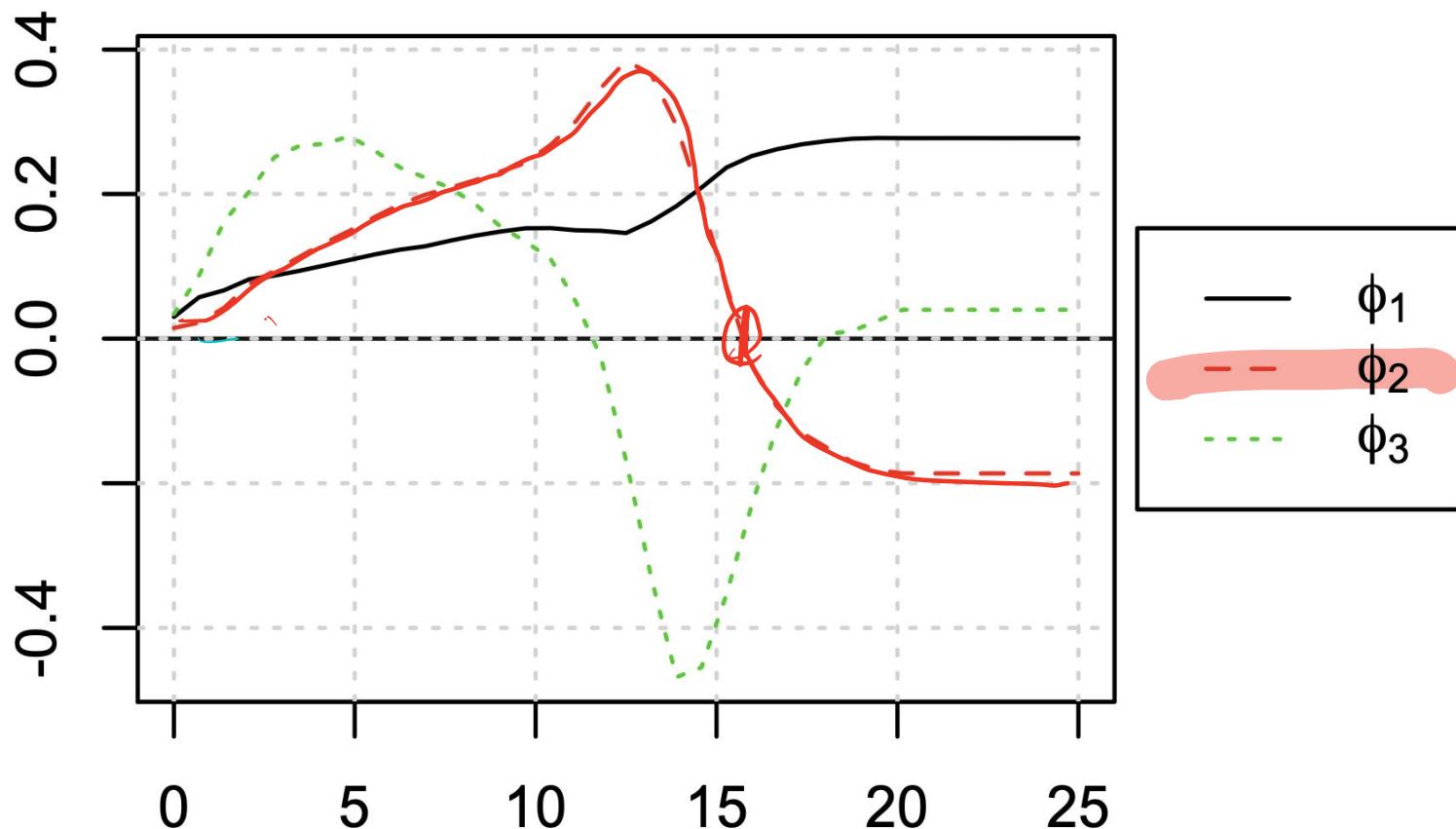
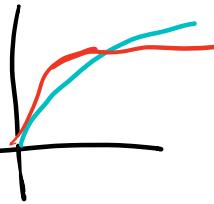
We can think of principal directions as ***modes of variation!***



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

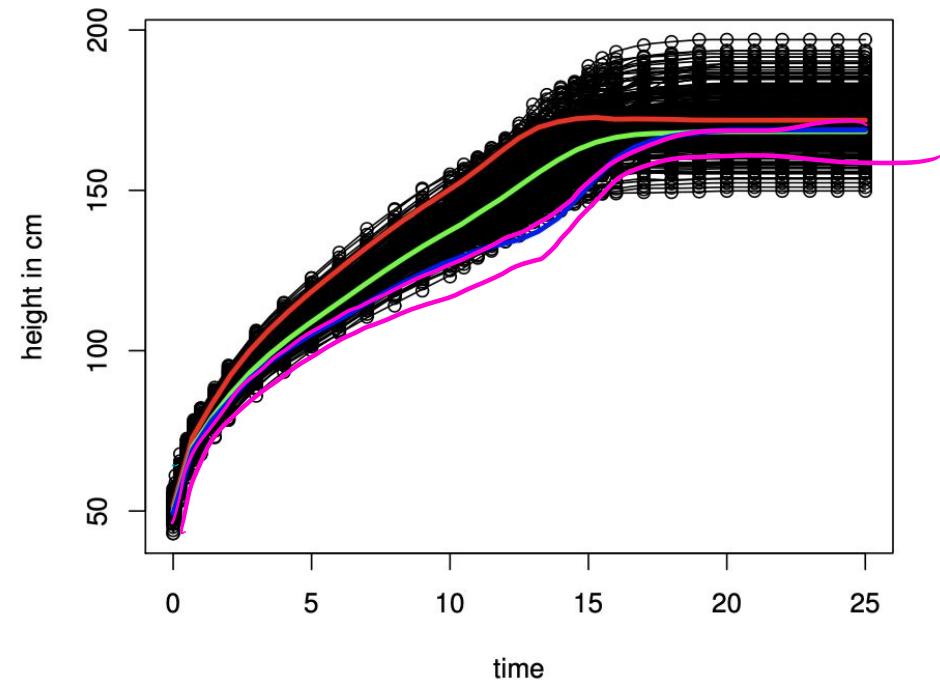
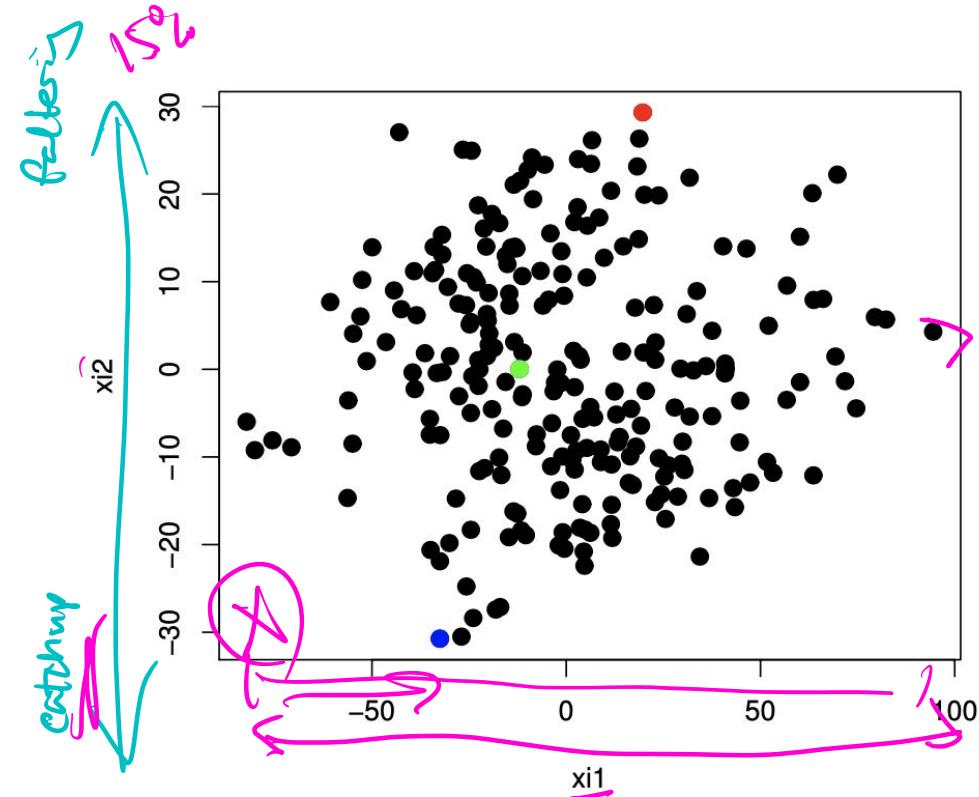


Look at PC 2 now: second mode of variation



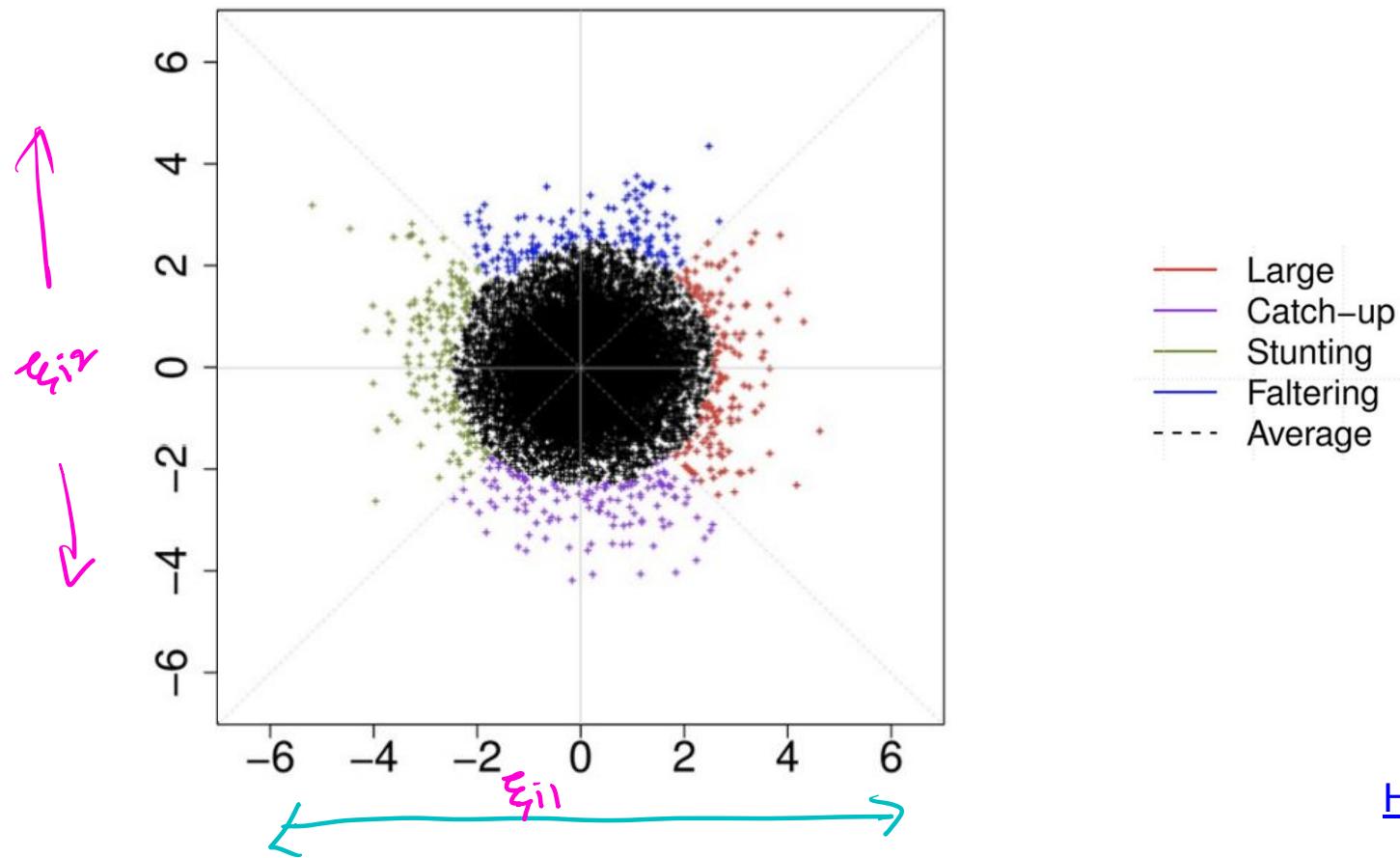
$$x_i \approx \bar{x} + \xi_{i1} \phi_1 + \xi_{i2} \phi_2$$

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](#)

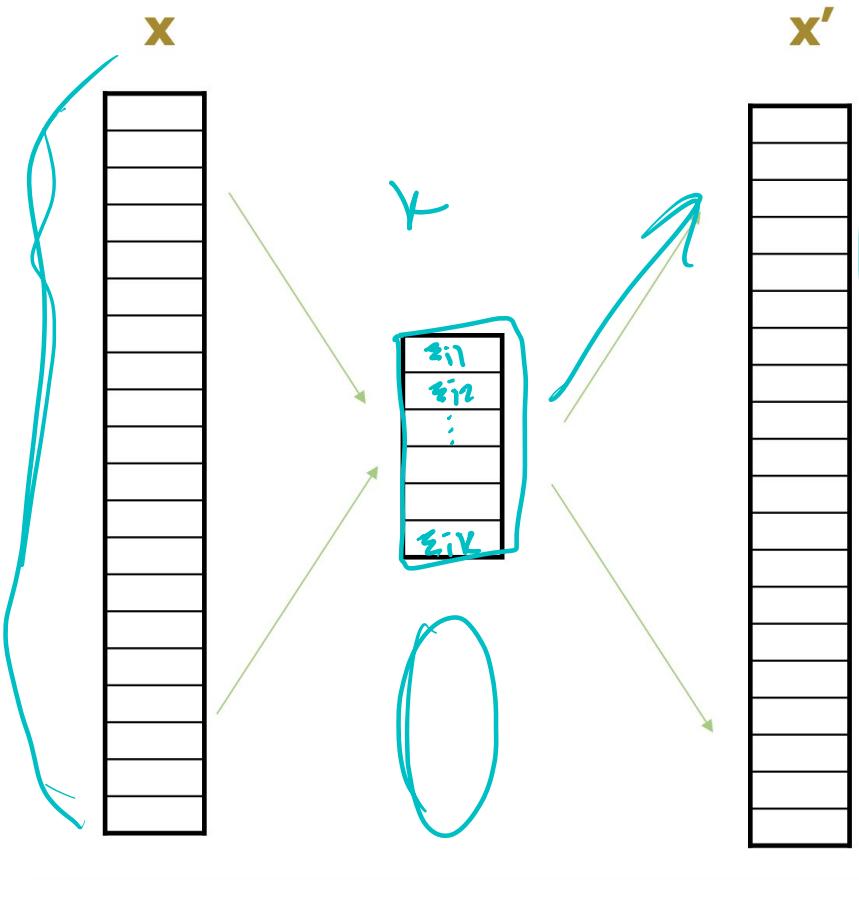
Noise Filtering Ex for Images

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

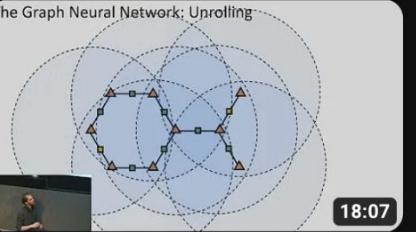


1:53:46

 Bedouin live at Petra, Jordan for Cercle

Cercle

1.8M views • 8 months ago



18:07

 The Graph Neural Network: Uprolling

Microsoft Research

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



32:52

 The Cinematography That Changed Cinema

The Cinema Cartography

970K views • 2 years ago

Motivation

Wildlife

Machine learning



1:47



Raven approaches me and makes clicky sound

Travel Wonders

8K views • 1 month ago



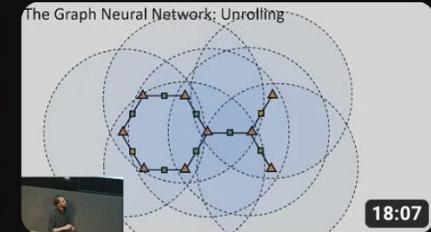
1:53:46



Bedouin live at Petra, Jordan for Cercle

Cercle

1.8M views • 8 months ago



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



32:52



The Cinematography That Changed Cinema

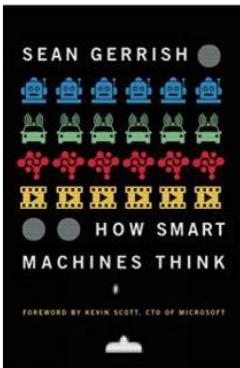
The Cinema Cartography

970K views • 2 years ago

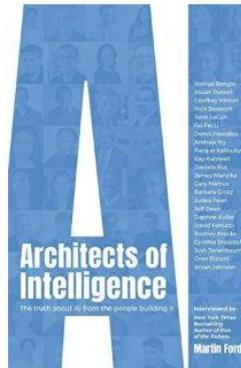
Motivation

Customers who bought this item also bought

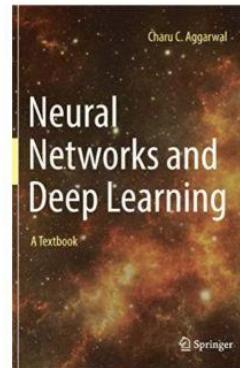
Page



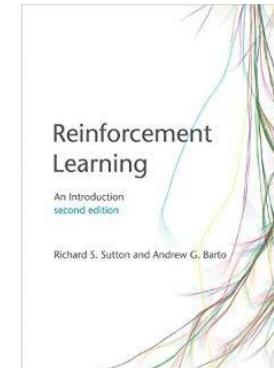
[How Smart Machines Think \(The MIT Press\)](#)
Sean Gerrish
 1
Hardcover
\$25.15 



[Architects of Intelligence: The truth about AI from the people building it](#)
Martin Ford
 23
Paperback
\$22.49 



[Neural Networks and Deep Learning: A Textbook](#)
Charu C. Aggarwal
 12
Hardcover
\$55.92 



[Reinforcement Learning: An Introduction \(Adaptive Computation and Machine Learning series\)](#)
Richard S. Sutton
 6
Hardcover
\$51.43 



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

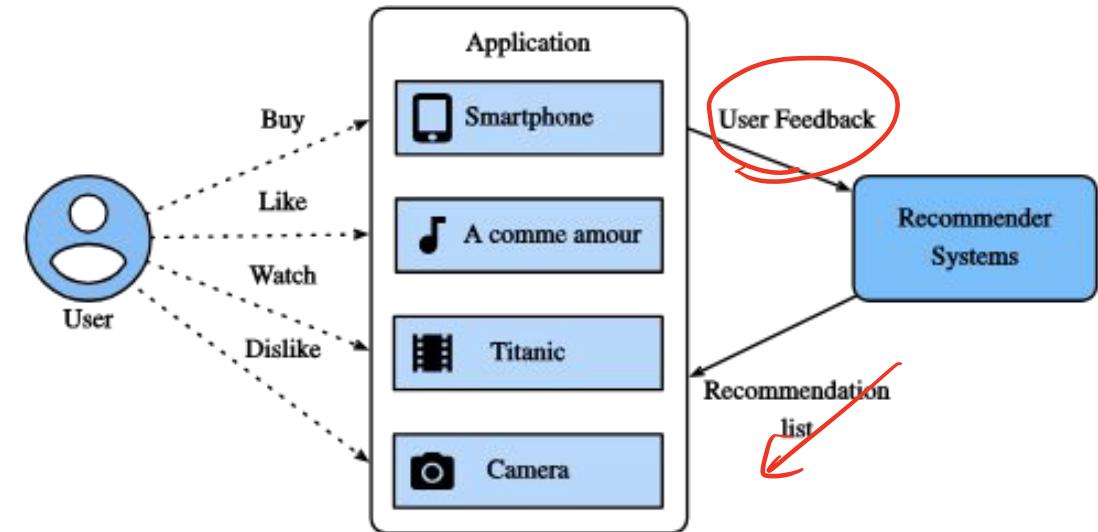
Flood fears rise as drenching rains and storms on tap threaten Sonoma County and North Coast

Yesterday • Local coverage

Full Coverage

Recommender System:

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



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

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. *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

Content-based or Collaborative Filtering?

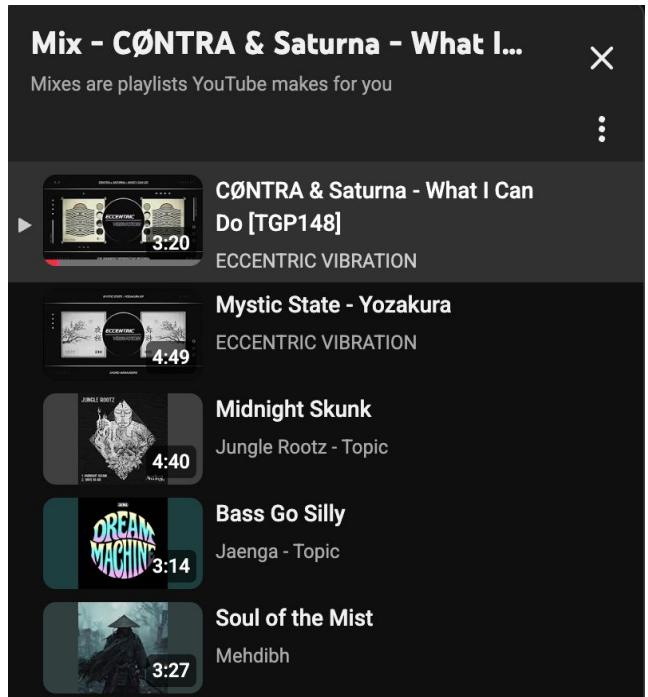
- The owner of a new bookstore in San Francisco wants to recommend books to users. This recommendation will be made on the receipt, right after the a customer purchased a book. **CB**
- You run an online furniture store and have item ratings from many users. You want to use this to identify what items 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%)

	2			4	5	
	5		4			1
			5		2	
		1		5		4
			4			2
	4	5		1		



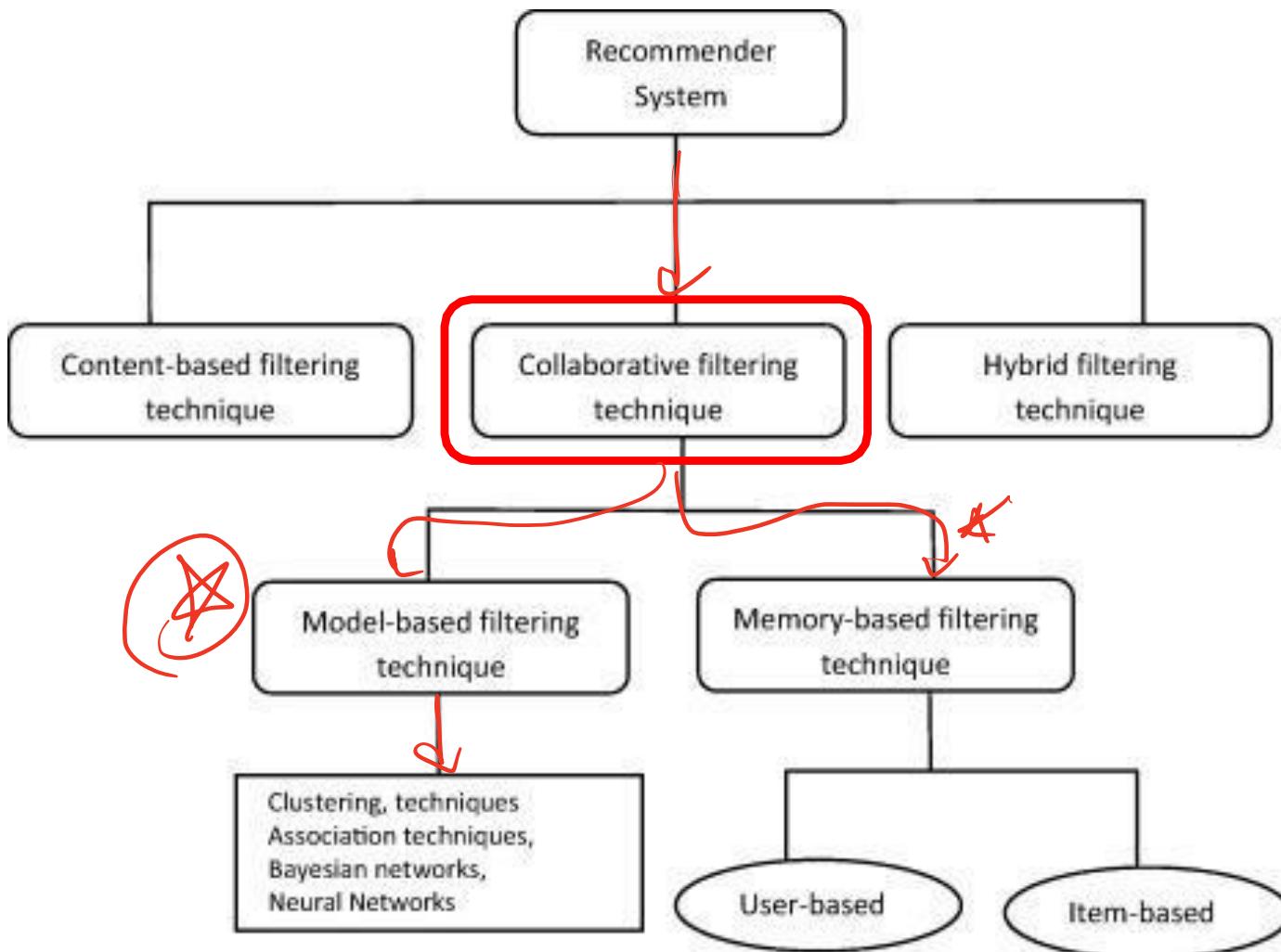
Example in tabular form

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	?



Collaborative filtering methods

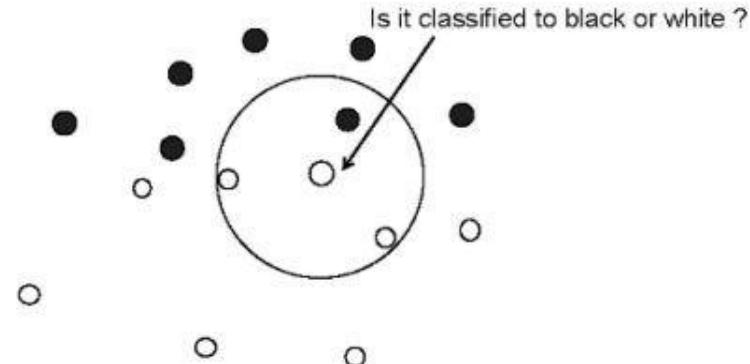
- **Memory based methods**

- Memorizes the utility matrix
- K-nearest neighbor approach
- Tends to be slow

- **Model based method**

- Fits a model
- Tends to have better performance

3-Nearest Neighbor



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)

$$Y = UDV^T \rightarrow$$

$$\hat{Y} = UV^T = PQ^T \leftarrow \text{matrix form}$$

$n_u \times n_m$ $n_u \times k$ $k \times n_m$

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

$V^{=2}$

Q_k^T	(Movie 1)	(Movie 2)	(Movie 3)	(Movie 4)	(Movie 5)
Dim1	-0.44	-0.57	0.06	0.38	0.57
Dim2	0.58	-0.66	0.26	0.18	-0.36

$K^{=2}$

$$\begin{aligned}\hat{r}_{ij} &= u_i \cdot v_j \\ &= 0.47 \cdot 0.38 + (-3) \cdot 0.18 =\end{aligned}$$

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”

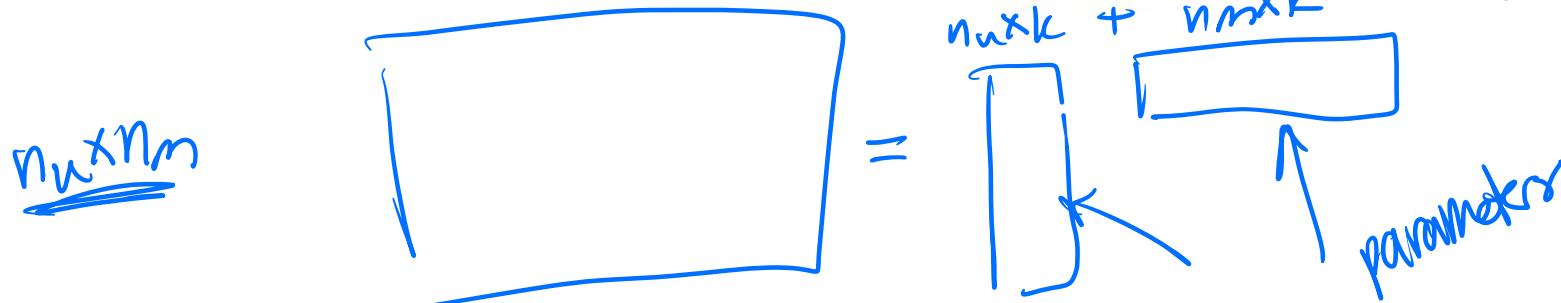


Matrix Factorization model

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 $i \in 1, \dots, n_u$ and $j \in 1, \dots, n_m$

User embedding: i^{th} row of \mathbf{U}

Item embedding: j^{th} row of \mathbf{V}

Predicted ratings:

$$\hat{y}_{ij} = (UV^T)_{ij} = \sum_{s=1}^K u_{is} v_{js}$$

$= \mathbf{u}_i \cdot \mathbf{v}_j$

Matrix Factorization model (Example)

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

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

Exercise: Compute the prediction of the two blank elements.

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

$$\begin{bmatrix} 5 & 1 & 4 & 5 & 1 \\ \hat{y}_{21} & 5 & 2 & 1 & 4 \\ 1 & 4 & 1 & 1 & 2 \\ 4 & 1 & 5 & 5 & 4 \\ 5 & 3 & 3 & \hat{y}_{54} & 4 \\ 1 & 5 & 1 & 1 & 1 \\ 5 & 1 & 5 & 5 & 4 \end{bmatrix} \approx \begin{bmatrix} 0.2 & 3.4 \\ 3.6 & 1.0 \\ 2.6 & 0.6 \\ 0.9 & 3.7 \\ 2.0 & 3.4 \\ 2.9 & 0.5 \\ 0.8 & 3.9 \end{bmatrix} \times \begin{bmatrix} u_2 \\ v_1 \end{bmatrix} = \begin{bmatrix} 0.0 & 1.5 & 0.1 & 0.0 & 0.7 \\ 1.3 & 0.0 & 1.2 & 1.4 & 0.7 \end{bmatrix}$$

$u_2 \cdot v_1 = 3.6(0) + 1 \cdot 1.3 = 1.3 = \hat{y}_{21}$

$\hat{y}_{54} = 2.0(0) + 3.4(1.4) \approx 4.7$

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 $i \in 1, \dots n_u$ and $j \in 1, \dots n_m$

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

$$E(u, v) = \frac{1}{N} \sum_{(i, j): r_{ij} = 1} (\underline{y_{ij}} - \underline{\hat{y}_{ij}})^2$$

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.

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

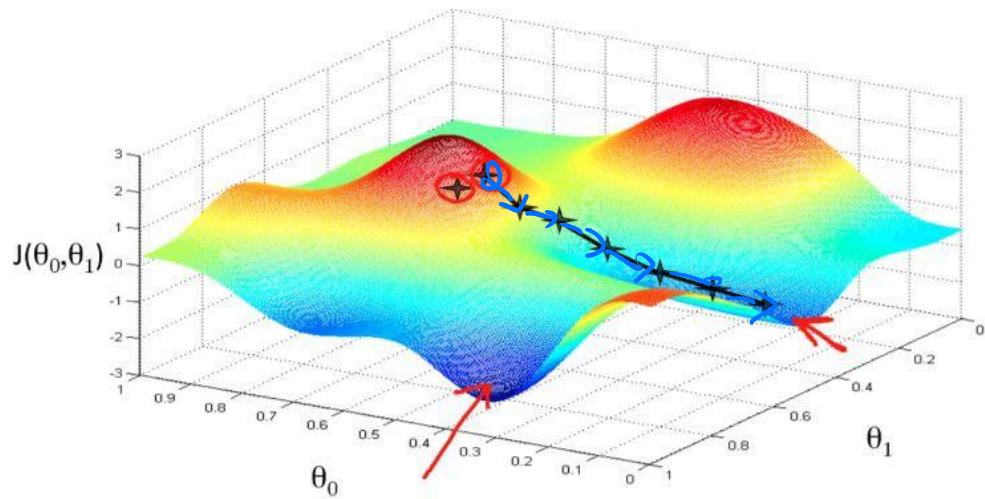


This minimization is over **all** users and items (movies)

Gradient Descent

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

$$\theta \leftarrow \theta - \eta \frac{\partial E}{\partial \theta}$$



Notation

$$\underline{\underline{E}}(w_1, \dots, w_n)$$

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

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

Gradient Descent

$$\min_w E(w)$$

$w \leftarrow$ random initialization;

for $i = 1 : max_iter$ **do**

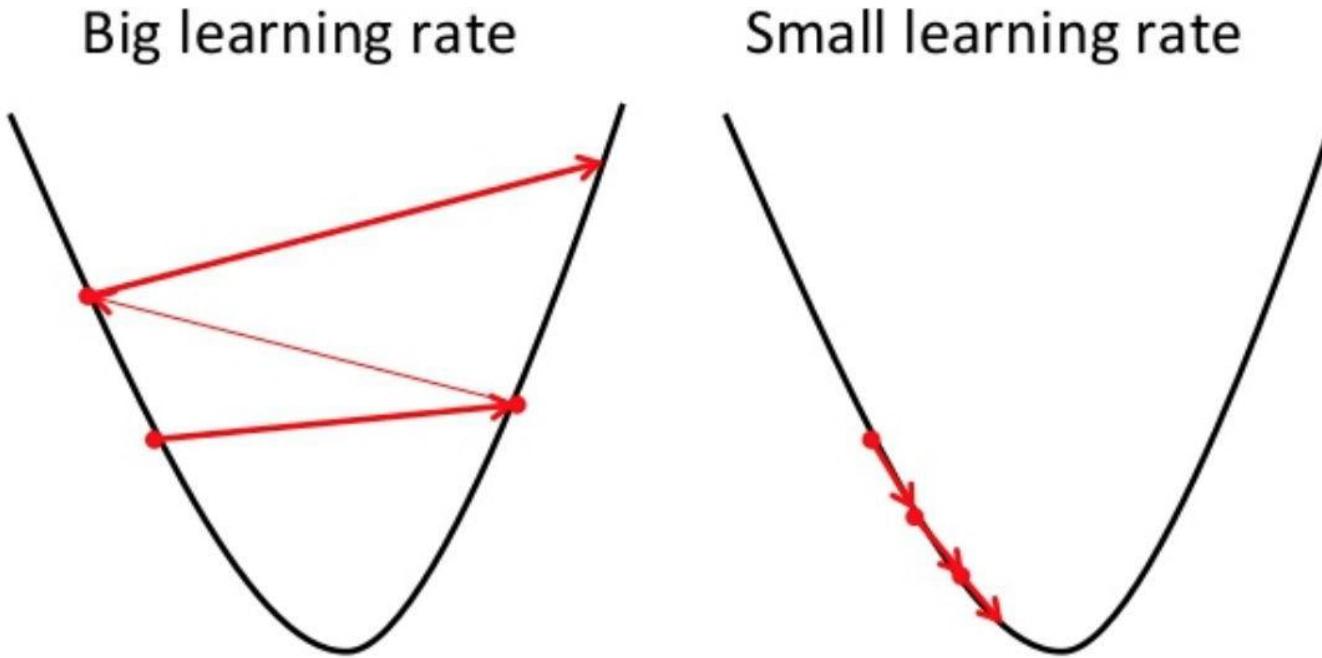
$w \leftarrow w - \eta \nabla E(w);$

end



Learning rate

Gradient Descent - Learning rate



Exercise:

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

$$\hat{y} = \underbrace{a \cdot x}_{\textcolor{blue}{z}} + \underbrace{b}_{\textcolor{blue}{z}}$$

Need 2 eqs: one for a & one for b

Loss:

$$E(a, b) = \sum_{i=1}^n (y_i - ax_i - b)^2$$

Update slope (a):

$$a^{(k+1)} = a^{(k)} - \eta \cdot \frac{\partial E}{\partial a}$$

$$\frac{\partial E}{\partial a} = \frac{\partial}{\partial a} \sum_{i=1}^n (y_i - ax_i - b)^2$$

$$= \sum_{i=1}^n \frac{\partial}{\partial a} (y_i - ax_i - b)^2$$

$$= \sum_{i=1}^n 2(y_i - ax_i - b) \cdot (-x_i)$$

chain

$$= 2 \sum_{i=1}^n (y_i - ax_i - b)(x_i)$$

Plug back in:

$$a^{(k+1)} = a^{(k)} + 2\eta \sum_{i=1}^n (y_i - a^{(k)}x_i - b^{(k)})x_i$$

Updating Eq for Intercept (b):

$$b^{(k+1)} = b^{(k)} - \eta \frac{\partial E}{\partial b}$$

$$\begin{aligned} \frac{\partial E}{\partial b} &= \frac{\partial}{\partial b} \sum_{i=1}^n (y_i - ax_i - b)^2 \\ &= \sum_{i=1}^n \frac{\partial}{\partial b} (y_i - ax_i - b)^2 \end{aligned}$$

$$= \sum_{i=1}^n 2(y_i - ax_i - b) \stackrel{\text{chain}}{\overbrace{(-1)}} = -2 \sum_{i=1}^n (y_i - ax_i - b)$$

$$b^{(k+1)} = b^{(k)} + 2\eta \sum_{i=1}^n (y_i - ax_i - b)$$

Gradient descent equations for MF

$$\hat{y} = u \cdot v^T$$

$$\begin{bmatrix} 1 \\ -\hat{r}_{ij} \end{bmatrix} = \begin{bmatrix} -n_i - \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ v_j \end{bmatrix} \Rightarrow \hat{y}_{ij} = u_i \cdot v_j$$

$$E(u, v) = \frac{1}{N} \sum_{\{(ij) : ij=1\}} (r_{ij} - u_i \cdot v_j)^2$$

↑
i is a user index
↑ j is an item "n"

Gradient descent equations for MF

Updating eq for an arbitrary element of u :

u_{dk}

d is a user index

k is an index for the embedding

$$- u_{dk} \leftarrow u_{dk} - \eta \frac{\partial E}{\partial u_{dk}}$$

$$\boxed{u_{dk}^{(z+1)} = u_{dk}^{(z)} - \eta \frac{\partial E}{\partial u_{dk}}}$$

$$\frac{\partial E}{\partial u_{ik}} = \frac{\partial}{\partial u_{ik}} \frac{1}{N} \sum_{\substack{(i,j) : r_{ij}=1}} (y_{ij} - \hat{y}_{ij})^2$$

$$= \frac{1}{N} \sum_{(i,j)} \frac{\partial}{\partial u_{ik}} (y_{ij} - \hat{y}_{ij})^2$$

$$= \frac{1}{N} \sum_{(i,j)} 2(y_{ij} - \hat{y}_{ij}) \left(-\frac{\partial \hat{y}_{ij}}{\partial u_{ik}} \right)$$

↗

claim:

$$\frac{\partial \hat{y}_{ij}}{\partial u_{ik}} = \begin{cases} 0 & \text{if } l \neq i \\ v_{jk} & \text{if } l = i \end{cases}$$

EX: $K=2$

$$\hat{y}_{34} = u_3 \cdot v_4 = (u_{31} \ u_{32}) \cdot (v_{41} \ v_{42})$$

$$= \underbrace{u_{31} v_{41} + u_{32} v_{42}}_l$$

If $l=5, k=1$,

then $\frac{\partial \hat{y}_{34}}{\partial u_{51}} = \frac{\partial}{\partial u_{51}} (u_{31} v_{41} + u_{32} v_{42}) = 0$

If $\ell=3$, $k=1$, $(\ell=i)$

$$\frac{\partial \hat{y}_{24}}{\partial u_{31}} = \frac{\partial}{\partial u_{31}} (u_{31} v_{41} + u_{32} v_{42}) = v_{41} + 0 = v_{41}$$

" v_{jk} " //

$$\text{II} = \frac{1}{N} \sum_{i,j} 2(y_{ij} - \hat{y}_{ij})^2 \underbrace{\left(-\frac{\partial \hat{y}_{ij}}{\partial u_{ik}} \right)}_{z.}$$

$$= \frac{1}{N} \left(2 \sum_{\substack{i \\ \{(i,j) \in r_{ij}=1\}}} \underbrace{(y_{ij} - \hat{y}_{ij})(-v_{jk})}_{z.} \right)$$

means:

$$l=3 \quad \left. \begin{array}{c} y_{24} \\ y_{31} \\ y_{35} \\ y_{512} \\ y_{510} \\ y_{106} \end{array} \right\} = \frac{-2}{N} \sum_j (y_{ej} - \hat{y}_{ej})(v_{jk})$$

for a full matrix

$$E(u, v) = \frac{1}{N} \sum_{i=1}^{n_u} \sum_{j=1}^{n_v} (y_{ij} - \hat{y}_{ij})^2$$

can you derive the 60 equations?

Gradient descent equations for MF

Claim: Updating Eqs for MF are:

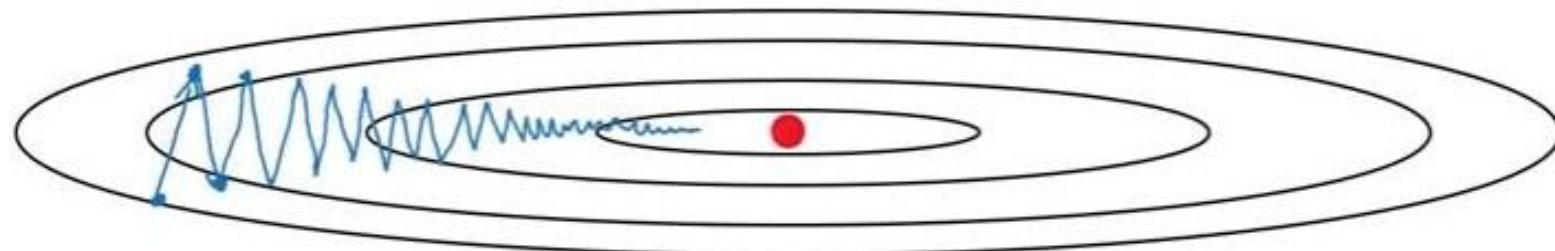
$$u_{ik} \leftarrow \underline{u_{ik}} + \frac{2\eta}{N} \sum_{j:r_{ij}=1} (y_{ij} - u_i \cdot v_j) v_{jk}$$

$$v_{jk} \leftarrow v_{jk} + \frac{2\eta}{N} \sum_{i:r_{ij}=1} (y_{ij} - u_i \cdot v_j) u_{ik}$$

See Matrix factorization notes pdf!

Gradient Descent with Momentum

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



Gradient Descent with Momentum

```
w ← random initialization;  
v ← 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;

$v \leftarrow 0;$

$$\beta = 0$$

for $i = 1 : max_iter$ **do**

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

$w \leftarrow w - \eta v;$



momentum term

end

Gradient Descent with Momentum

$w \leftarrow$ random initialization;

$v \leftarrow 0;$

$$\beta = 0.9$$

for $i = 1 : max_iter$ **do**

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

$w \leftarrow w - \eta v;$



momentum term

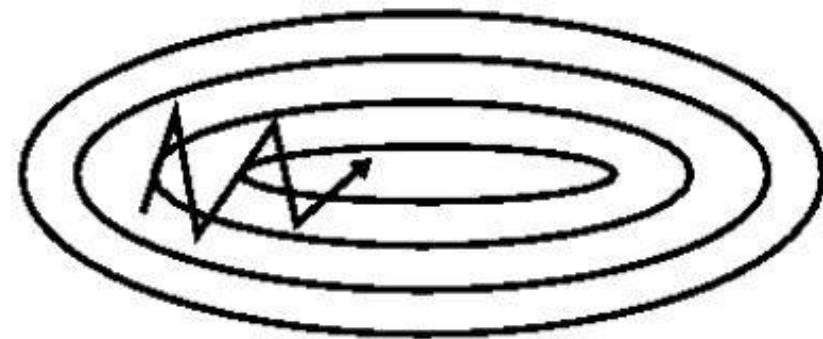
end

Momentum



(a) SGD without momentum

$$\beta = 0$$



(b) SGD with momentum

$$\beta = 0.9$$

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.pdf