Announcements

- Limited office hours this week (see Ed Discussion)
- Quiz 11 is due **Thursday, December 1 at 8pm**
- HW 6 due **Friday, December 2 at 8pm**
  - 2 day extension

Bayesian Networks

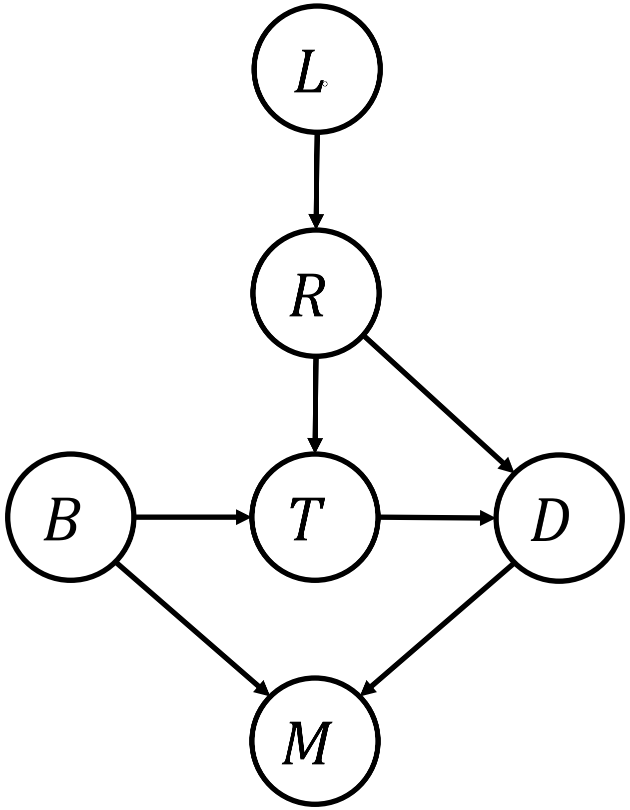
- **Nodes/vertices:** Variables X_k
- **Arcs/edges:** Encode parameter structure
  - **Parameters:** Distribution of each X_i given its parents
  - $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P\big(X_i = x_i \mid \text{parents}(X_i) = \big(x_{i_1}, \ldots, x_{i_k}\big)\big)$
- Graph structure establishes conditional independencies
  - Based on d-separation algorithm
  - Also encodes conditional independence given neighbors; see https://en.wikipedia.org/wiki/Moral_graph for details

Bayesian Networks

- **Nodes/vertices:** Variables X_k
- **Arcs/edges:** Encode parameter structure
  - **Parameters:** Distribution of each X_i given its parents
  - $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P\left(X_i = x_i \mid \text{parents}(X_i) = \left(x_{i_1}, \ldots, x_{i_k}\right) \right)$
- Graph structure establishes conditional independencies
  - Based on d-separation algorithm
  - Also encodes conditional independence given neighbors; see https://en.wikipedia.org/wiki/Moral_graph for details

Example

$$P(L, B, R, T, D, M) =$$
$$P(L)$$
$$P(B)$$
$$P(\,R \mid L\,)$$
$$P(\,T \mid R, B\,)$$
$$P(\,D \mid R, T\,)$$
$$P(\,M \mid B, D\,)$$



Example

$$P(L, B, R, T, D, M) =$$

$$P(L)$$

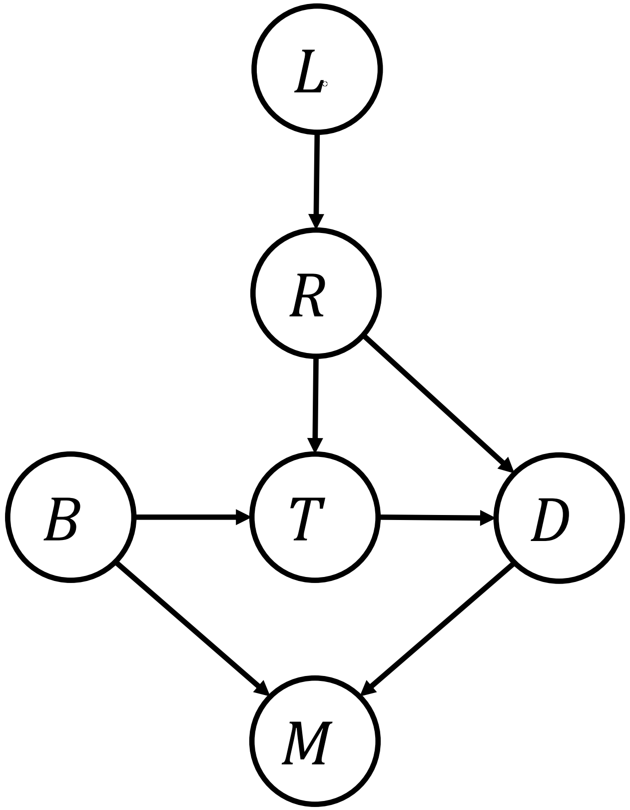
$$P(B)$$

$$P(R \mid L)$$

$$P(T \mid R, B)$$

$$P(D \mid R, T)$$

$$P(M \mid B, D)$$

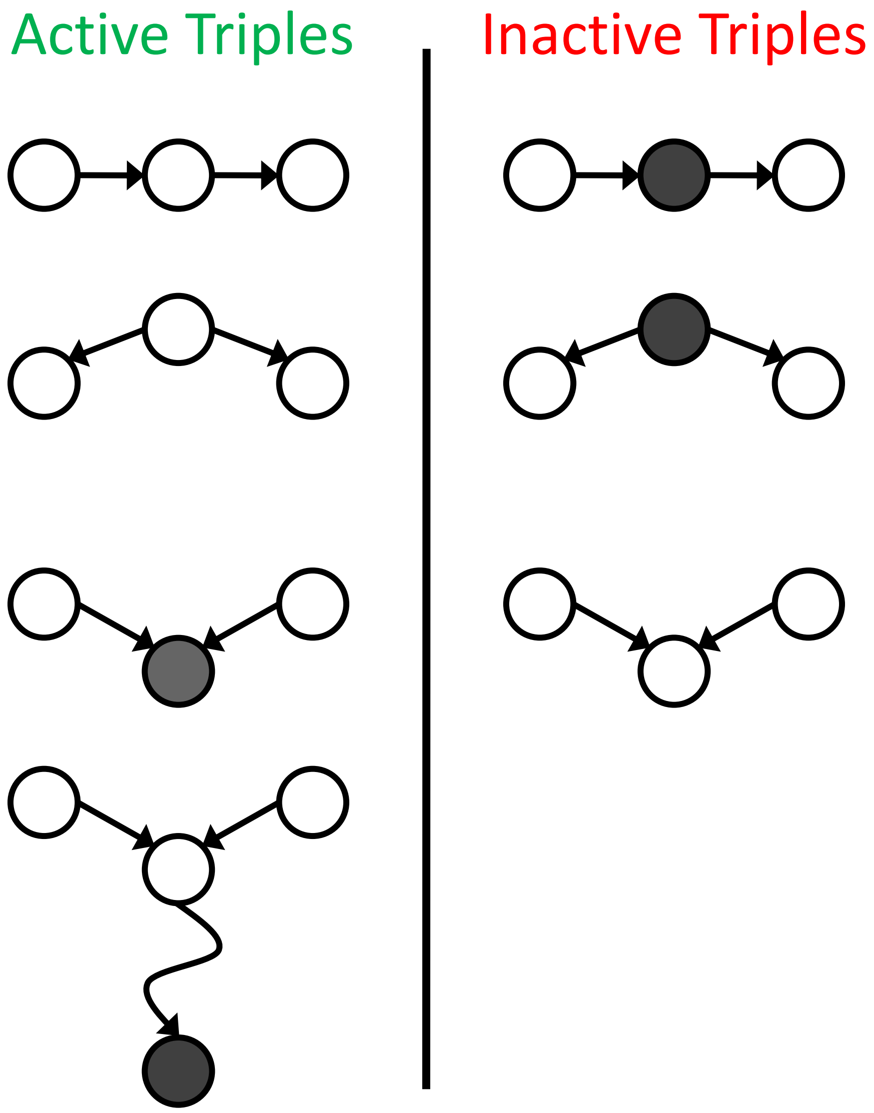


D-Separation

- **Query:** $X \perp\!\!\!\perp Y \mid Z_1, \dots, Z_n$

D-Separation

- **Causal chain**
  - $A \rightarrow B \rightarrow C$
  - Active iff $B \notin \{Z_i\}$
- **Common cause**
  - $A \leftarrow B \rightarrow C$
  - Active iff $B \notin \{Z_i\}$
- **Common effect**
  - $A \rightarrow B \leftarrow C$
  - Active iff $B \in \{Z_i\}$ (or descendant $\in \{Z_i\}$)



D-Separation

- **Query:** $X \perp\!\!\!\perp Y \mid Z_1, \dots, Z_n$

- **for each** (acyclic) path $X = A_0 - A_1 - \cdots - A_n - A_{n+1} = Y$**:**
  - $\text{active} \leftarrow \mathbf{true}$
  - **for each** triple $A_{i-1} - A_i - A_{i+1}$**:**
    - **if** triple is causal chain **and** $A_i \in \{Z_j\}$: $\text{active} \leftarrow \mathbf{false}$
    - **if** triple is common cause **and** $A_i \in \{Z_j\}$: $\text{active} \leftarrow \mathbf{false}$
    - **if** triple is causal effect **and** $\text{descendants}(A_i) \cap \{Z_j\} = \emptyset$: $\text{active} \leftarrow \mathbf{false}$
  - **if** active: **return false**
- **return true**

- **Intuition:** Return false if there is a path where all triples are active

Marginal Inference

- **Input:**
  - **Evidentiary variables:** $E_1 = e_1, \dots, E_k = e_k$ (features)
  - **Query variable:** Q (label)
  - **Hidden variables:** $H_1, \dots, H_m$ (all remaining, “latent” variables)

- **Goal:** For each q, compute

$$P(Q = q \mid E_1 = e_1, \dots, E_k = e_k)$$

- **Equivalently:** Likelihood $p(y \mid x)$

Variable Elimination

- **Step 0:** Initial factors are $P(\,X_i \mid \text{parents}(X_i)\,)$ for each node X_i
  - Immediately drop rows conditioned on evidentiary variables

- **Step 1:** For each H_i:
  - **Step 1a:** Join all factors containing H_i
  - **Step 1b:** Eliminate H_i

- **Output:** Join all remaining factors and normalize

Maximum Likelihood Learning

- Minimize the NLL:

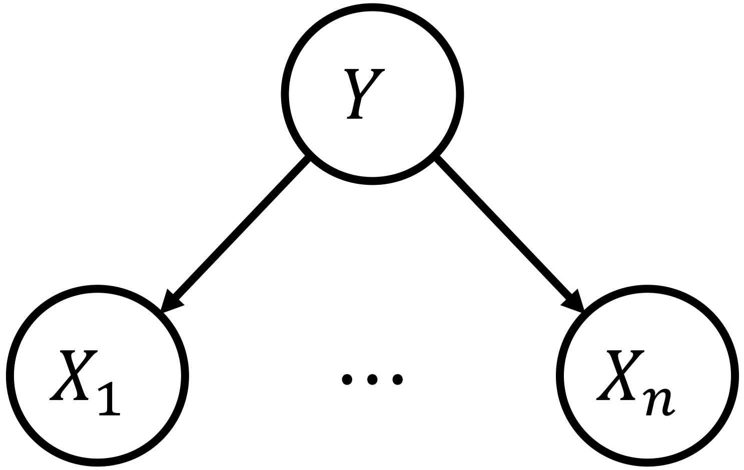
$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} \sum_{j=1}^{d} \log P_{\theta}\left(X_j = x_{i,j} \;\middle|\; \text{parents}(X_j) = \left(x_{i,k_1}, \dots, x_{i,k_j}\right)\right)$$

- Can use gradient descent to optimize
  - There is a nice formula for the gradient

Simplest Example: Naïve Bayes

- Model:

$$P(Y, X_1 \ldots, X_n) = P(Y) \prod_{i=1}^{n} P(\, X_i \mid Y \,)$$



Inference in Naïve Bayes

- **Step 1:** For each $y \in D_Y$, compute joint probability distribution

$$P(y, x_1, \dots, x_n) = P(y) \prod_{i=1}^{n} P(\, x_i \mid y \,)$$

- **Step 2:** Normalize distribution:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(\, x_i \mid y \,)}{Z}$$

  - Here, $Z = \sum_{y' \in D_Y} P(y') \prod_{i=1}^{n} P(\, x_i \mid y' \,)$

Naïve Bayes for Spam Detection

- Bag of words model

- Parameter sharing via “tied” distribution: For all i, j, constrain

$$P(X_i = x \mid Y) = P\left(X_j = x \mid Y\right)$$

- Encodes invariant structure in bag of words models

Maximum Likelihood Learning

- Minimize the NLL for Naïve Bayes for text:

number of words in example i

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} \left\{ \log P_\theta(y_i) + \log \sum_{j=1}^{d_i} P_\theta(\, x_{i,j} \mid y_i \,) \right\}$$

- Can show that parameters are counts:

$$P_\theta(\, x \mid y \,) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y \wedge x_{i,j} = x)}{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y)}$$

Maximum Likelihood Learning

number of words in example i

- Minimize the NLL for Naïve Bayes for text:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} \left\{ \log P_\theta(y_i) + \log \sum_{j=1}^{d_i} P_\theta(\, x_{i,j} \mid y_i \,) \right\}$$

number of times the word x
occurs in examples labeled y

- Can show that parameters are counts:

$$P_\theta(\, x \mid y \,) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y \wedge x_{i,j} = x)}{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y)}$$

Maximum Likelihood Learning

number of words in example i

- Minimize the NLL for Naïve Bayes for text:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} \left\{ \log P_\theta(y_i) + \log \sum_{j=1}^{d_i} P_\theta(\, x_{i,j} \mid y_i \,) \right\}$$

number of times the word x occurs in examples labeled y

- Can show that parameters are counts:

$$P_\theta(\, x \mid y \,) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y \wedge x_{i,j} = x)}{\sum_{i=1}^{n} \sum_{j=1}^{d_i} 1(y_i = y)}$$

number of words in examples labeled y

Naïve Bayes for Spam Detection

$P(y)$

```
not spam: 0.66
spam:     0.33
```

$P(x \mid \text{spam})$

```
the :  0.0156
to  :  0.0153
and :  0.0115
of  :  0.0095
you :  0.0093
a   :  0.0086
with:  0.0080
from:  0.0075
...
```

$P(x \mid \text{not spam})$

```
the :  0.0210
to  :  0.0133
of  :  0.0119
2002:  0.0110
with:  0.0108
from:  0.0107
and :  0.0105
a   :  0.0100
...
```

Reinforcement Learning

- Sequential decision-making

- **Planning:** Known transitions/rewards
  - Optimization

- **Reinforcement learning:** Unknown transitions/rewards
  - Learning + optimization

Q Iteration

- Initialize $Q_1(s,a) \leftarrow 0$ for all s, a
- For $i \in \{1,2,\dots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow \sum_{s' \in S} P(\, s' \mid s, a\,) \cdot \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s', a') \right)$$

Q Learning

- Initialize $Q_1(s, a) \leftarrow 0$ for all s, a
- For $i \in \{1, 2, \ldots\}$ until convergence:

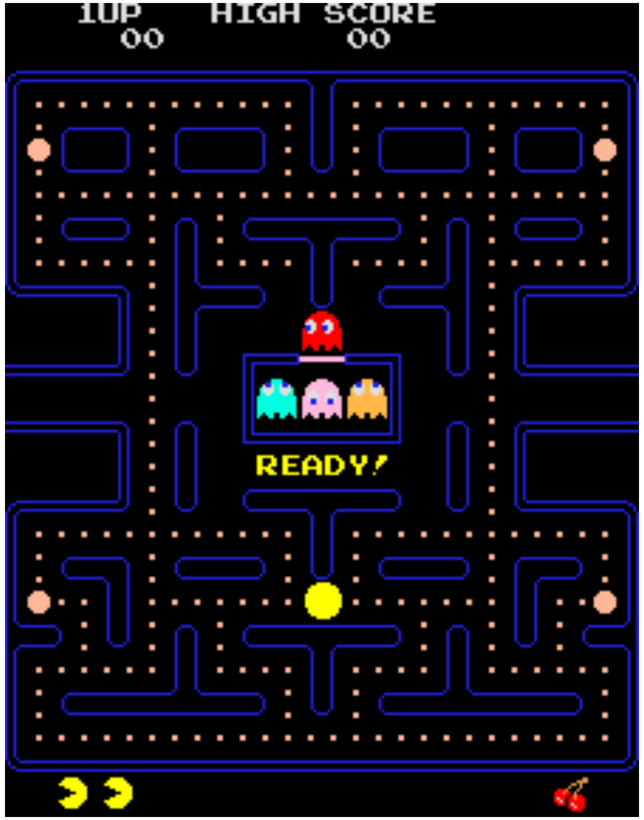
$$Q_{i+1}(s, a) \leftarrow (1 - \alpha) \cdot Q_i(s, a) + \alpha \cdot \left(R(s, a, s') + \gamma \cdot \max_{a' \in A} Q_i(s', a')\right)$$

Curse of Dimensionality

- How large is the state space?
  - **Gridworld:** One for each of the n cells
  - **Pacman:** State is $(\text{player}, \text{ghost}_1, \dots, \text{ghost}_k)$, so there are n^k states!

- **Problem:** Learning in one state does not tell us anything about the other states!

- Many states → learn very slowly



State-Action Features

- Can we learn **across** state-action pairs?

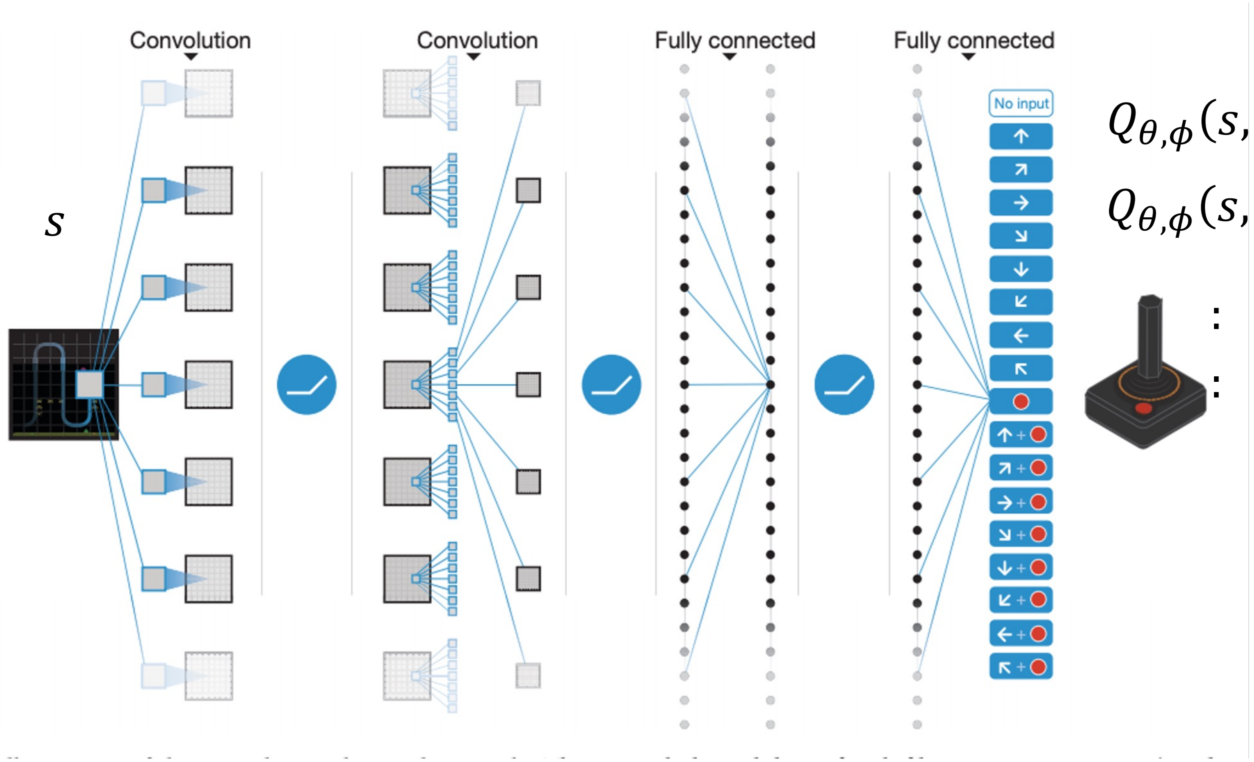
- Yes, use features!
  - $\phi(s, a) \in \mathbb{R}^d$
  - Then, learn to predict $Q^*(s, a) \approx Q_\theta(s, a) = f_\theta\big(\phi(s, a)\big)$
  - Enables generalization to similar states

- **Examples:** Distance to closest ghost, distance to closest dot, etc.

Neural Network Q Function

- Can also use neural networks to **learn** features (e.g., represent Pacman game state as an image and feed to CNN)!

s



$Q_{\theta,\phi}(s, a_1)$

$Q_{\theta,\phi}(s, a_2)$

$\vdots$

Deep Q Learning

- For $i \in \{1,2,\ldots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow (1-\alpha) \cdot Q_i(s,a) + \alpha \cdot \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s',a') \right)$$

Deep Q Learning

- For $i \in \{1, 2, \ldots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) - \alpha \cdot \left(Q_i(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s',a') \right) \right)$$

Deep Q Learning

- For $i \in \{1,2,\dots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) - \alpha \cdot \left(Q_i(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s',a')\right)\right)$$

- **Learning:** Gradient descent with the squared Bellman error loss:

$$\Bigg(\underbrace{Q_\theta(s,a)}_{\text{“Predicted Label” } \hat{y}} - \underbrace{\left(R(s,a,s') + \gamma \cdot \max_{a'} Q_\theta(s',a')\right)}_{\text{“Label” } y}\Bigg)^2$$

Based on slide by Sergey Levine

Deep Q Learning

- For $i \in \{1,2,\ldots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) - \alpha \cdot \left(Q_i(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s',a')\right)\right)$$

- **Learning:** Gradient descent with the squared Bellman error loss:

$$\theta_{i+1} \leftarrow \theta_i - \alpha \cdot \left(Q_{\theta_i}(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a'} Q_{\theta_i}(s',a')\right)\right) \nabla_\theta Q_{\theta_i}(s,a)$$

assume constant when computing gradient

Based on slide by Sergey Levine

Deep Q Learning

- For $i \in \{1,2,\dots\}$ until convergence:

$$Q_{i+1}(s,a) \leftarrow Q_i(s,a) - \alpha \cdot \left(Q_i(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a' \in A} Q_i(s',a') \right) \right)$$

- **Learning:** Gradient descent with the squared Bellman error loss:

$$\theta_{i+1} \leftarrow \theta_i - \alpha \cdot \left(Q_{\theta_i}(s,a) - \left(R(s,a,s') + \gamma \cdot \max_{a'} Q_{\theta_i}(s',a') \right) \right) \nabla_\theta Q_{\theta_i}(s,a)$$

assume constant when computing gradient

Based on slide by Sergey Levine

Deep Q Learning

- **Iteratively perform the following:**
  - Take an action a_i and observe (s_i, a_i, s_{i+1}, r_i)
  - $y_i \leftarrow r_i + \gamma \cdot \max_{a' \in A} Q_\theta(s_{i+1}, a')$
  - $\theta \leftarrow \theta - \alpha \cdot \frac{d}{d\theta}(Q_\theta(s_i, a_i) - y_i)^2$

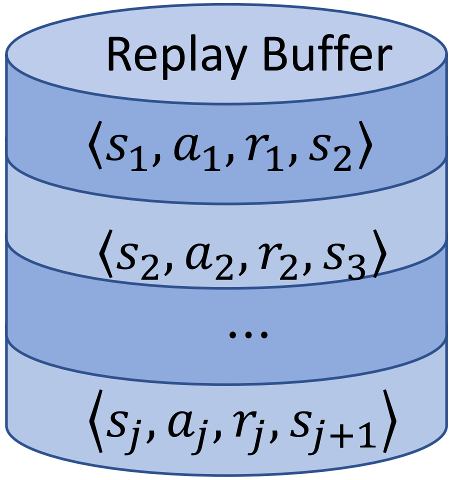
- **Note:** Pretend like y_i is constant when taking the gradient

- For finite state setting, recover incremental update if the "parameters" are the Q values for each state-action pair

Based on slide by Sergey Levine

Experience Replay Buffer

- **Problem**
  - Sequences of states are highly correlated
  - Tend to overfit to current states and forget older states
- **Solution**
  - Keep a **replay buffer** of observations (as a priority queue)
  - Gradient updates on samples from replay buffer instead of current state
- **Advantages**
  - Breaks correlations between consecutive samples
  - Can take multiple gradient steps on each observation

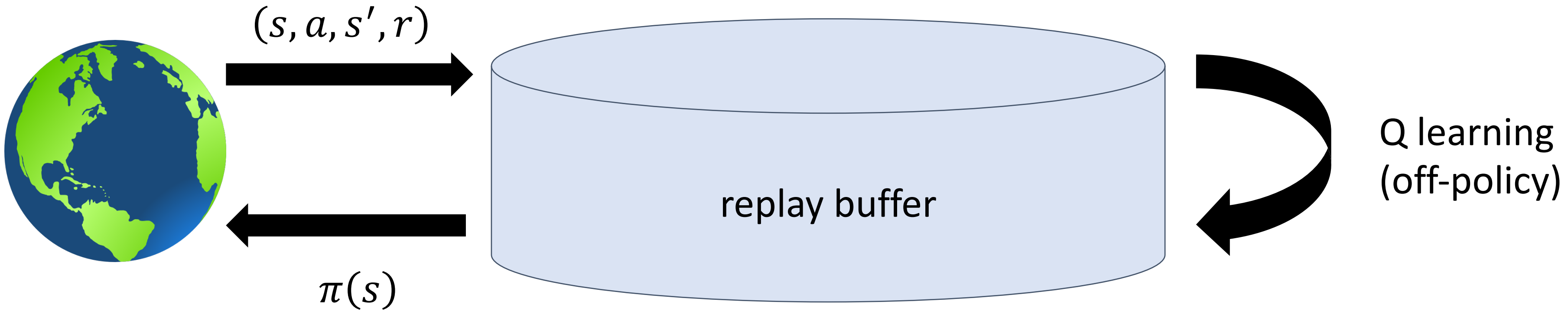


Priority Queue

Based on slide by Sergey Levine

Deep Q Learning with Replay Buffer

- **Iteratively perform the following:**
  - Take an action a_i and add observation (s_i, a_i, s_{i+1}, r_i) to replay buffer D
  - For $k \in \{1, \dots, K\}$:
    - Sample $(s_{i,k}, a_{i,k}, s_{i+1,k}, r_{i,k})$ from D
    - $y_{i,k} \leftarrow r_{i,k} + \gamma \cdot \max_{a' \in A} Q_\theta(s_{i+1,k}, a')$
    - $\phi \leftarrow \phi - \alpha \cdot \frac{d}{d\theta}\left(Q_\theta(s_{i,k}, a_{i,k}) - y_{i,k}\right)^2$



Based on slide by Sergey Levine

Target Q Network

- **Problem**
  - Q network occurs in the label y_i!
  - $\theta \leftarrow \theta - \alpha \cdot \frac{d}{d\theta}\left(Q_\theta(s_i, a_i) - r_i + \gamma \cdot \max_{a' \in A} Q_\theta(s_{i+1}, a')\right)^2$
  - Thus, labels change as Q network changes

- **Solution**
  - Use a separate **target Q network** for the occurrence in y_i
  - Only update target network occasionally
  - $\theta \leftarrow \theta - \alpha \cdot \frac{d}{d\theta}\left(\underbrace{Q_\theta(s_i, a_i)}_{\text{Original Q Network}} - r_i + \gamma \cdot \max_{a' \in A} \underbrace{Q_{\theta'}(s_{i+1}, a')}_{\text{Target Q Network}}\right)^2$

Based on slide by Sergey Levine

Deep Q Learning with Target Q Network

- **Iteratively perform the following:**
  - Take an action a_i and add observation (s_i, a_i, s_{i+1}, r_i) to replay buffer D
  - For $k \in \{1, \dots, K\}$:
    - Sample $(s_{i,k}, a_{i,k}, s_{i+1,k}, r_{i,k})$ from D
    - $y_{i,k} \leftarrow r_{i,k} + \gamma \cdot \max_{a' \in A} Q_{\theta'}(s_{i+1,k}, a')$
    - $\theta \leftarrow \theta - \alpha \cdot \frac{d}{d\theta}\left(Q_\theta(s_{i,k}, a_{i,k}) - y_{i,k}\right)^2$
  - Every N steps, $\theta' \leftarrow \theta$

Deep Q Learning for Atari Games

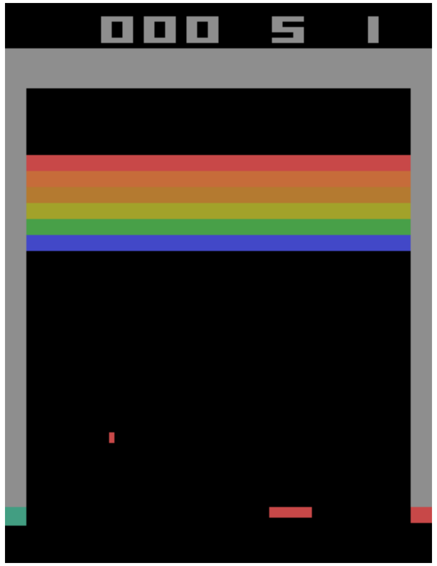


Image Sources:
https://towardsdatascience.com/tutorial-double-deep-q-learning-with-dueling-network-architectures-4c1b3fb7f756
https://deepmind.com/blog/going-beyond-average-reinforcement-learning/
https://jaromiru.com/2016/11/07/lets-make-a-dqn-double-learning-and-prioritized-experience-replay/

Lecture 22: Recommender Systems

CIS 4190/5190

Fall 2022

Recommender Systems

- **Media recommendations:** Netflix, Youtube, etc.
- **News feed:** Google News, Facebook, Twitter, Reddit, etc.
- **Search ads:** Google, Bing, etc.
- **Products:** Amazon, ebay, Walmart, etc.
- **Dating:** okcupid, eharmony, coffee-meets-bagel, etc.

Recommender Systems

- **Account for:**
  - 75% of movies watched on Netflix [1]
  - 60% of YouTube video clicks [2]
  - 35% of Amazon sales [3]

[1] McKinsey & Company (Oct 2013): https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers [Note: non-authoritative source; estimates only]

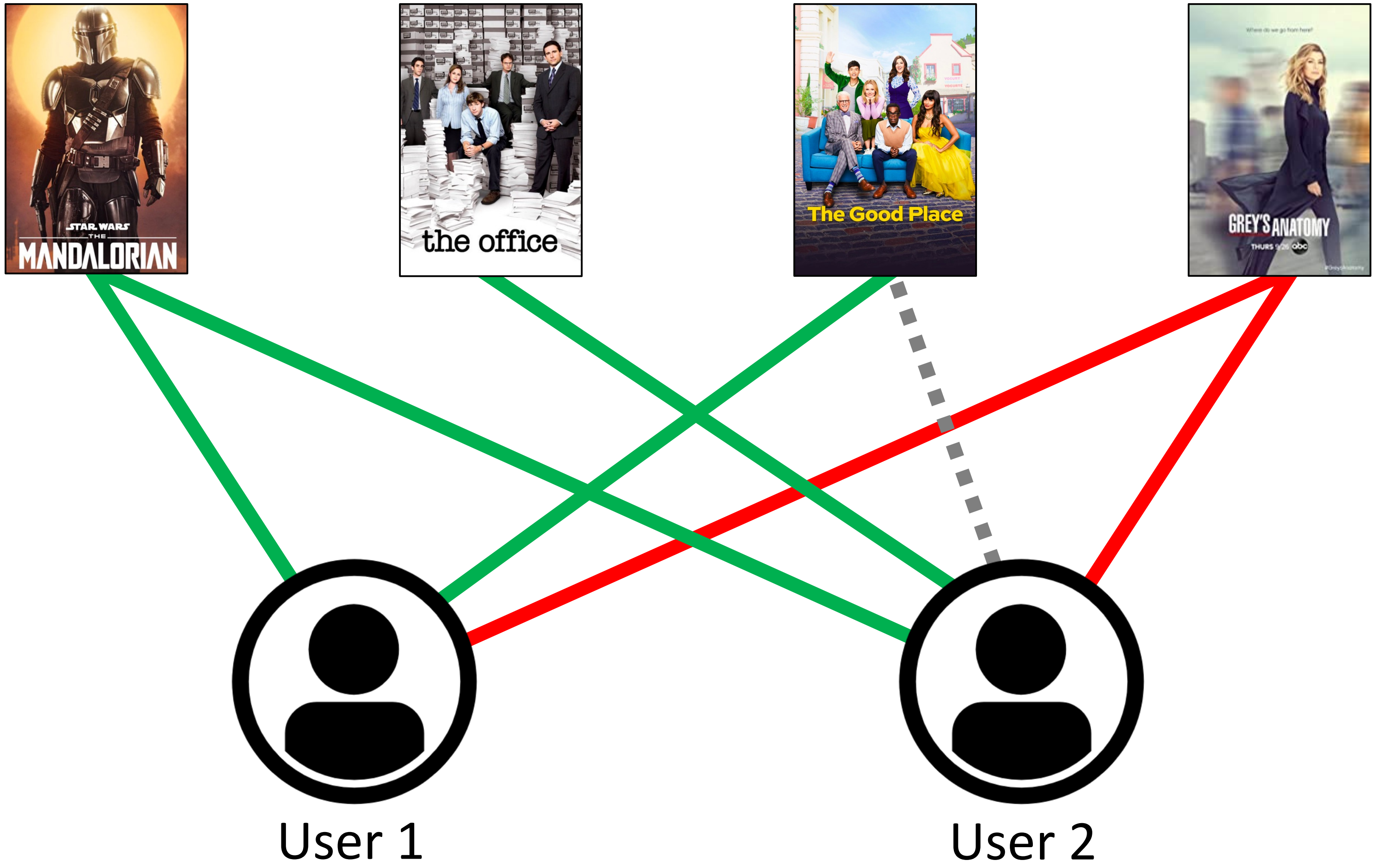
[2] J. Davidson, et al. (2010). The YouTube video recommendation system. Proc. of the 4th ACM Conference on Recommender systems (RecSys). doi.org/10.1145/1864708.1864770

[3] M. Rosenfeld, et al. (2019). Disintermediating your friends: How online dating in the United States displaces other ways of meeting. Proc. National Academy of Sciences 116(36).

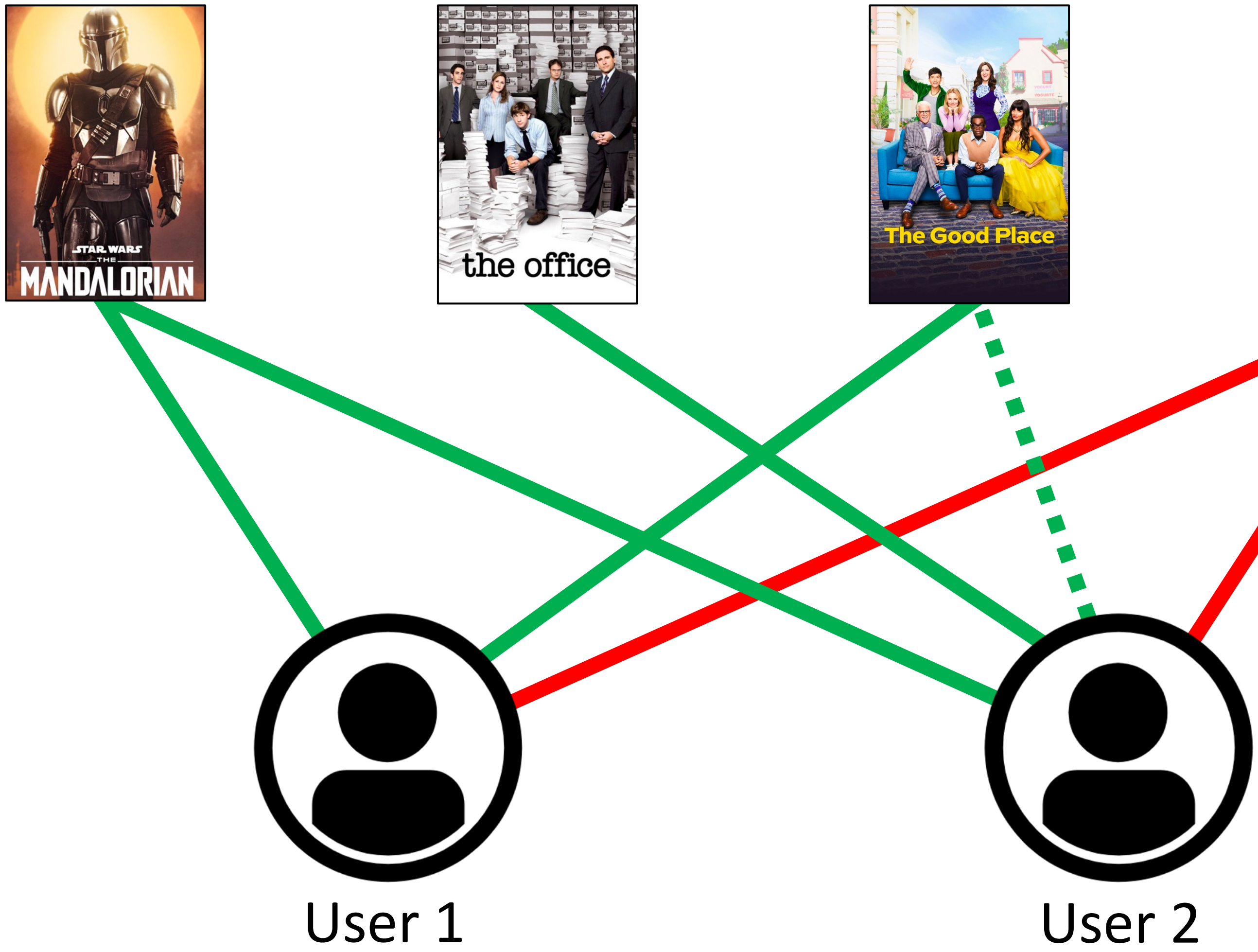
Popularity-Based Recommendation

- Just recommend whatever is currently popular
- Simple and effective, always try as a baseline
- Can be combined with more sophisticated techniques

Collaborative Filtering



Collaborative Filtering



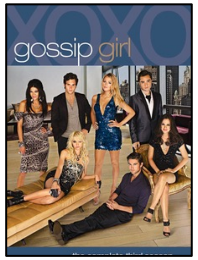
Collaborative Filtering

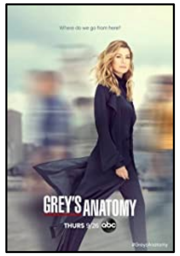
- **Given:**
  - Matrix $X_{i,k} = \begin{cases} \text{rating}_{i,k} & \text{if user}_i \text{ rated product}_k \\ \text{N/A} & \text{otherwise} \end{cases}$
  - Assume fixed set of n users and m products
  - **Not given any information about the products!**

- **Problem:** Predict what $X_{i,k}$ would be if it is observed
  - Not quite supervised or unsupervised learning!

Collaborative Filtering

Missing entries!

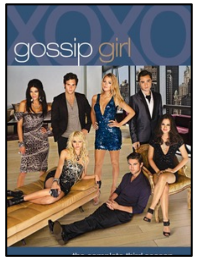


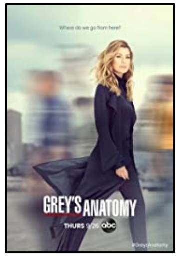


	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	…
Grace	4	5	4	1	5	3	…
Eric	1	4	5	1	5	3	…
Haren	5	5	5	1	3	4	…
Sai	1	2	5	4	3	5	…
Siyan	3	1	1	3	4	5	…
Nikhil	2	3	4	2	2	2	…
Felix	1	1	1	5	2	2	…

Collaborative Filtering

Missing entries!





	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

General Strategy

- **Step 1:** Construct user-item ratings
- **Step 2:** Identify similar users
- **Step 3:** Predict unknown ratings

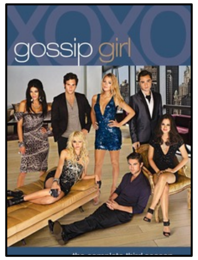
Step 1: Constructing User-Item Ratings

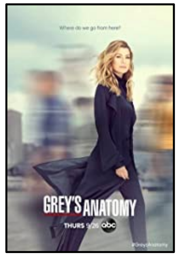
- Can use explicit ratings (e.g., Netflix)

- Can be implicitly inferred from user activity
  - User stops watching after 15 minutes
  - User repeatedly clicks on a video

- Feedback can vary in strength
  - **Weak:** User views a video
  - **Strong:** User writes a positive comment

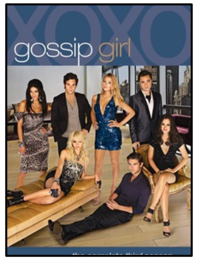
Step 2: Identifying Similar Users

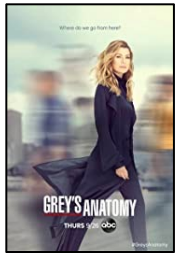




	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey’s Anatomy	…
Grace		5		1	5		…
Eric		4	5		5	3	…
Haren	5		5		3	4	…
Sai		2					…
Siyan	3	1		3		5	…
Nikhil				2	2		…
Felix	1		1		2		…

Step 2: Identifying Similar Users

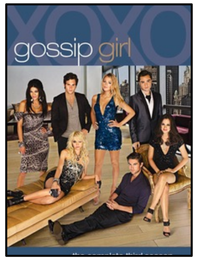


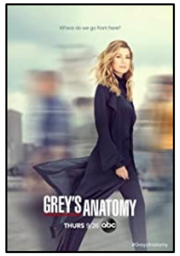


	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

similar

Step 2: Identifying Similar Users





	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

not similar

Step 2: Identifying Similar Users

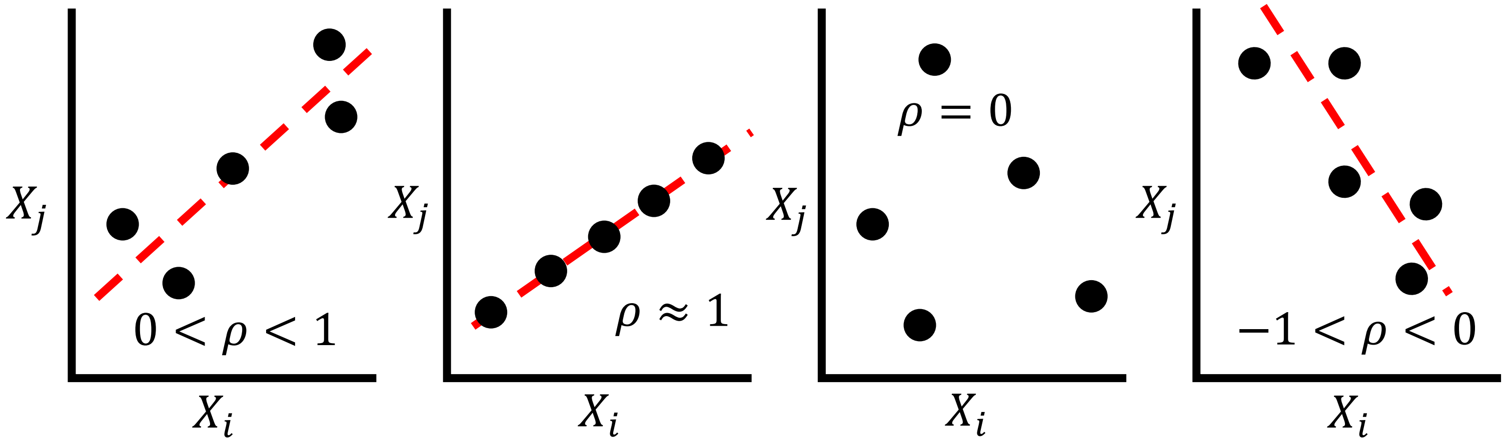
- **How to measure similarity?**
  - Distance $d(X_i, X_j)$, where X_i is vector of ratings for user i

- **Strategy 1:** Euclidean distance $d(X_i, X_j) = \|X_i - X_j\|_2$
  - Ignore entries where either X_i or X_j is N/A
  - **Shortcoming:** Some users might give higher ratings everywhere!

- Similar issues with other distance metrics such as cosine similarity

Step 2: Identifying Similar Users

- **Strategy 2:** Pearson correlation: $\rho = \frac{\sum_{k=1}^{m}(X_{i,k}-\bar{X}_i)(X_{j,k}-\bar{X}_j)}{\sqrt{\sum_{k=1}^{m}(X_{i,k}-\bar{X}_i)^2 \sum_{k=1}^{m}(X_{j,k}-\bar{X}_j)^2}}$
  - Here, $\bar{X}_i = \frac{1}{m}\sum_{k=1}^{m} X_{i,k}$
  - Normalization by variance deals with differences in individual rating scales



Step 3: Predict Unknown Ratings

- **Weighted averaging strategy**
  - Compute weights $w_{i,j} = g\left(d\left(X_i, X_j\right)\right)$ based on the distances
  - Normalize the weights to obtain $\bar{w}_{i,j} = \frac{w_{i,j}}{\sum_{j=1}^{n} w_{i,j}}$
  - For user i rating item k, predict

$$X_{i,k} = \bar{X}_i + \sum_{j=1}^{n} \bar{w}_{i,j} \cdot \left(X_{j,k} - \bar{X}_j\right)$$

Step 3: Predict Unknown Ratings

- **Variations**
  - Instead of weights, choose a neighborhood (e.g., threshold based on similarity, top-k based on similarity, or use k-means clustering)
  - Instead of subtracting the mean, normalize by standard deviation