Announcements

- Limited/modified office hours this week (see Ed Discussion)
- HW 6 due Wednesday, 11/29
- Quiz 10 due Thursday, 11/30
- Project Milestone 3 due Wednesday, 12/6
  - https://docs.google.com/document/d/17EAxAYeYB7bfs3YK69p6mPB75MpbyRq0/edit?usp=sharing&ouid=104445367729520435803&rtpof=true&sd=true

Recap

- **Q iteration:** Compute optimal Q function when the transitions and rewards are known

- **Q learning:** Compute optimal Q function when the transitions and rewards are unknown

- **Extensions**
  - Various strategies for exploring the state space during learning
  - Handling large or continuous state spaces

Exploration-Exploitation Tradeoff

- **Question:** How to choose actions?
  - **Exploration:** Try actions to better estimate their rewards
  - **Exploitation:** Use action with the best estimated reward to maximize payoff

Application: Training ChatGPT

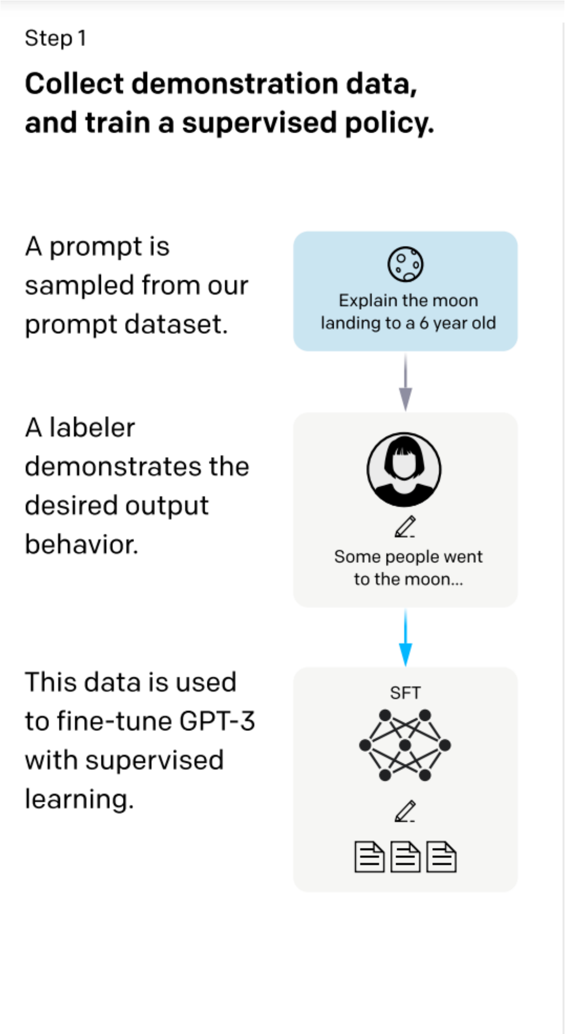
- **Problem**
  - Language models are trained using **unsupervised learning**
  - Generating from these models mimics training data rather than human preferences

- **Solution**
  - **Step 1:** Predict human preferences over possible generations (the reward)
  - **Step 2:** Finetune GPT using reinforcement learning, where it is rewarded for generating content preferred by humans

Application: Training ChatGPT

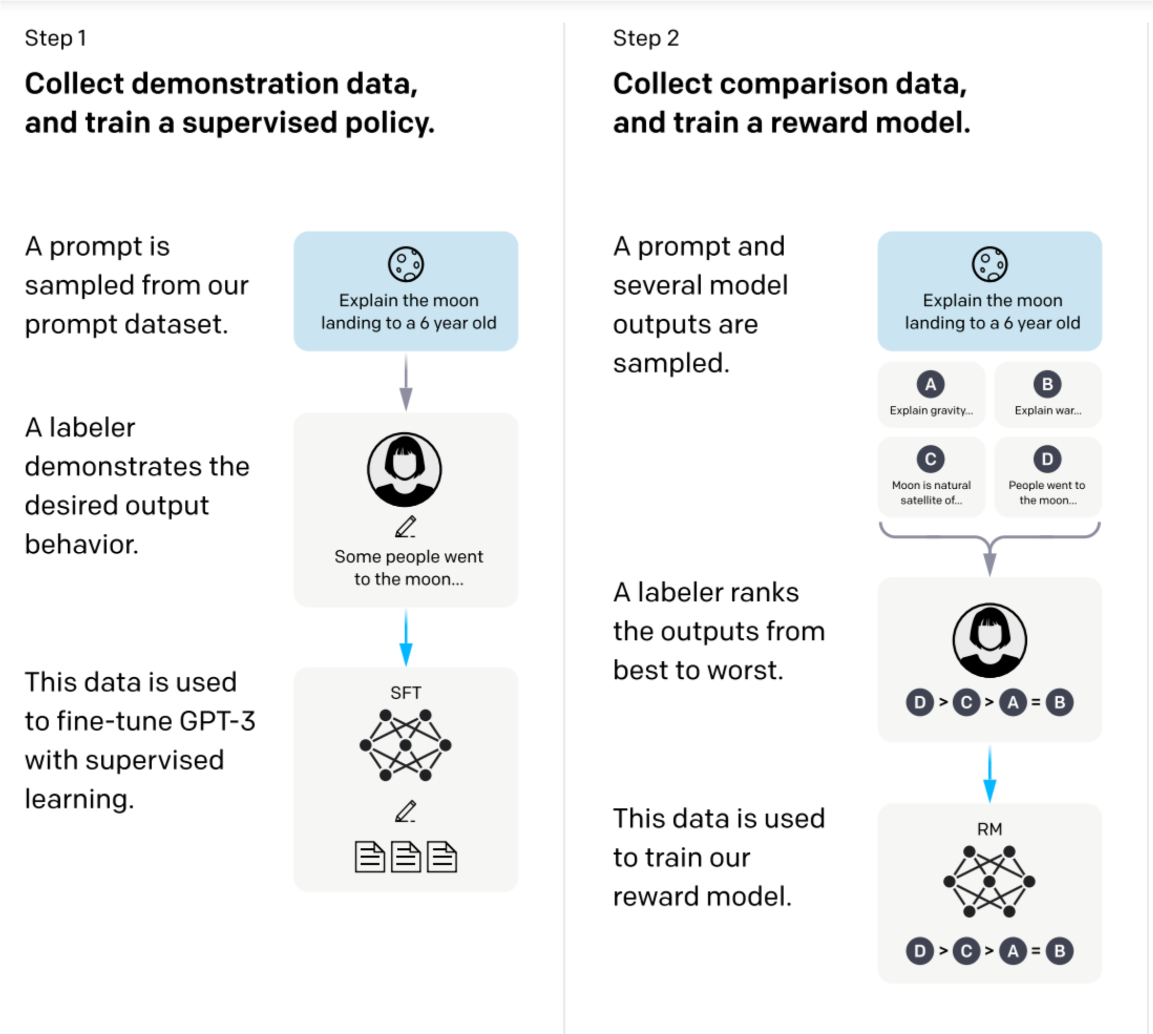
Source: Ouyang et al., Training language models to follow instructions with human feedback.

Application: Training ChatGPT



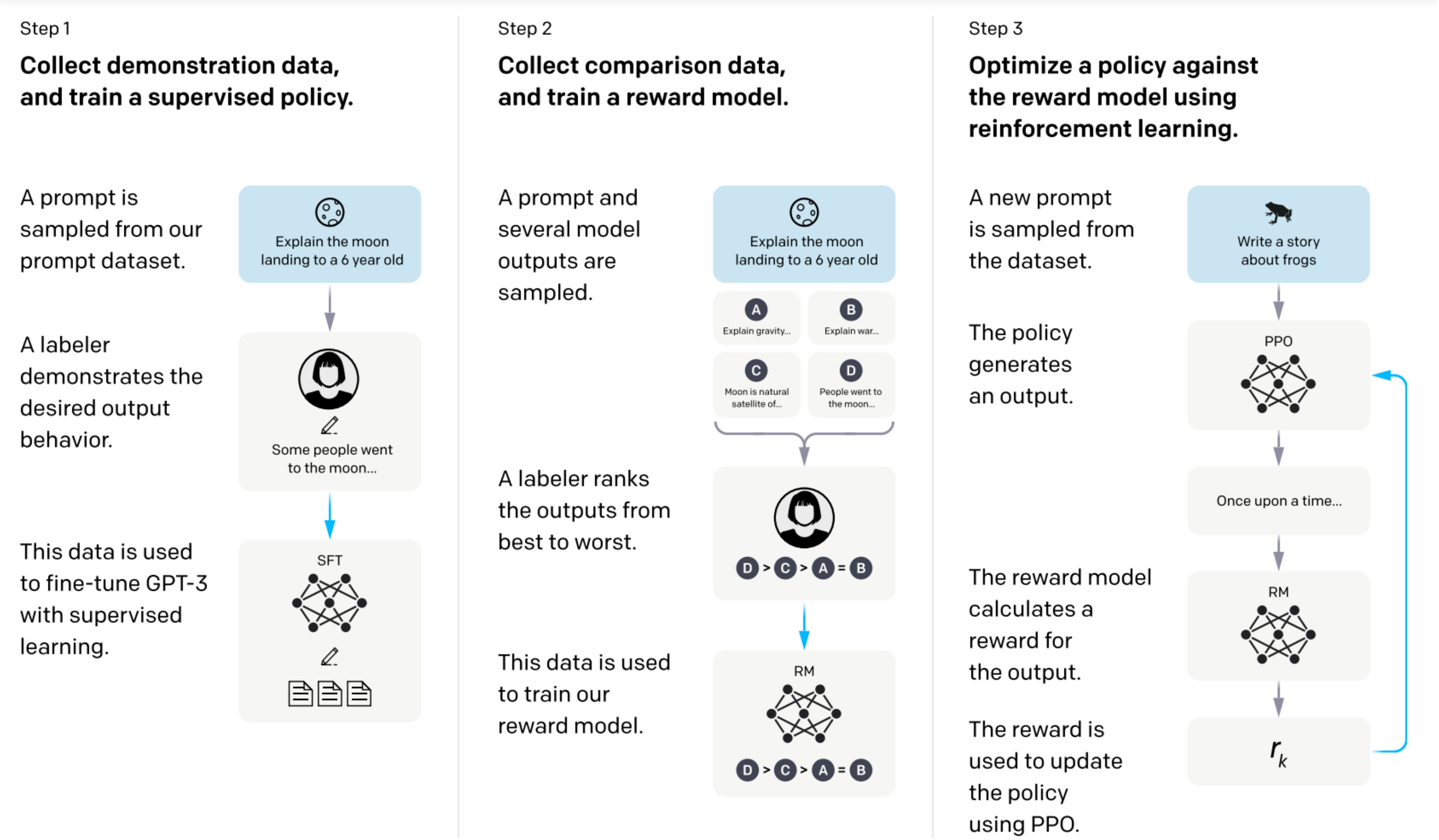
Source: Ouyang et al., Training language models to follow instructions with human feedback.

Application: Training ChatGPT



Source: Ouyang et al., Training language models to follow instructions with human feedback.

Application: Training ChatGPT

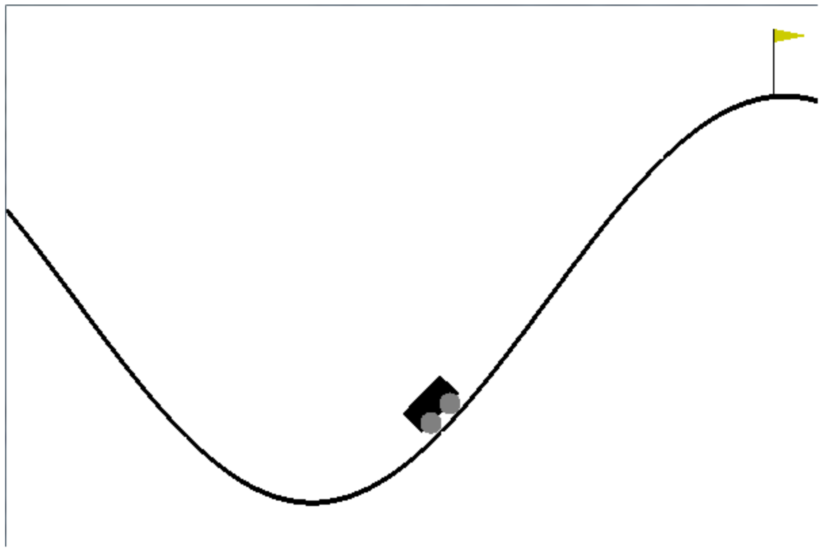


Source: Ouyang et al., Training language models to follow instructions with human feedback.

Exploration in Reinforcement Learning

- ϵ-greedy suffers additional issues due to state space

- Policy learning is an effective practical solution
  - No theoretical guarantees due to local minima



Exploration in Finite MDPs

- **Upper confidence bound (UCB)**
  - Choose action $a_t = \arg\max_{a \in A} \left\{ Q_t(s,a) + \frac{\text{const}}{\sqrt{N_t(s,a)}} \right\}$
  - $N_t(s,a) = \sum_{i=1}^{t-1} 1(s_i = s, a_i = a)$ is the number of times action a has been played in state s

- **Thompson sampling**
  - Choose action $a_t = \arg\max_{a \in A} \{ Q_t(s,a) + \epsilon_{t,s,a} \}$, where $\epsilon_{t,s,a} \sim N\left(0, \frac{\text{const}}{\sqrt{N_t(s,a)}}\right)$

- Both come with theoretical guarantees

Exploration in Continuous MDPs

- Can we adapt these ideas to continuous MDPs?
  - Thompson sampling is more suitable

- **Bootstrap DQN**
  - Train ensemble of k different Q-function estimates $Q_{\theta_1}, \ldots, Q_{\theta_k}$ in parallel
  - Original idea was to use online bootstrap, but training from different random initial θ's worked as well
  - In each episode, act optimally according to Q_{θ_i} for $i \sim \mathrm{Uniform}(\{1, \ldots, k\})$

Exploration in Continuous MDPs

- Can we adapt these ideas to continuous MDPs?
  - Thompson sampling is more suitable

- **Soft Q-learning**
  - Sample actions according to $a \sim \mathrm{Softmax}\left(\left[\beta \cdot \hat{Q}_\theta(s,a)\right]_{a \in A}\right)$

Curiosity

- **Intuition:** Rather than focus on optimism with respect to reward, focus on exploring where we are uncertain

- **How to determine uncertainty?**

- **Candidate strategy**
  - Train a **dynamics model** to predict $s' = f(s, a)$
  - Take actions where $f(s, a)$ has high variance (e.g., use bootstrap)

- **Problems?**
  - What if s' includes spurious components, like a TV screen playing a movie?

Curiosity

- Learn a feature map $\phi(s) \in \mathbb{R}^d$
- **Model 1:** Train a model to predict state transitions:

$$\hat{\phi}(s') = f_\theta(\phi(s), a)$$

  - Feature map lets the model "ignore" spurious components of s such as a TV
  - **Problem:** We could just learn $\phi(s) = \vec{0}$?

Curiosity

- Learn a feature map $\phi(s) \in \mathbb{R}^d$

- **Model 1:** Train a model to predict state transitions:

$$\hat{\phi}(s') = f_\theta(\phi(s), a)$$

- **Model 2:** Train a model to predict action to achieve a transition:

$$\hat{a} = g_\theta\big(\phi(s), \phi(s')\big)$$

  - "Inverse dynamics model" that avoids collapsing ϕ

Curiosity

- Curiosity reward is

$$R(s, a, s') = \left\| \hat{\phi}(s') - \phi(s') \right\|_2^2$$

- In other words, reward agent for exercising transitions that f cannot yet predict accurately

Offline Reinforcement Learning

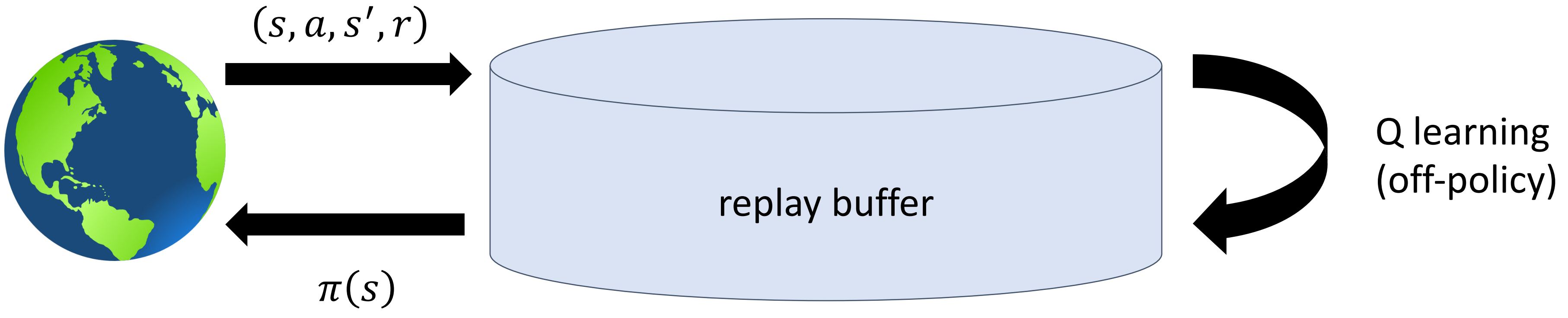
- **Offline reinforcement learning:** How can we learn **without** actively gathering new data?
  - E.g., learn how to perform a task from videos of humans performing the task
  - Also known as **off-policy** or **batch** reinforcement learning

- **Recall:** Drawback of Q learning was we need an exploration strategy

- However, this also enables us to use Q learning with offline data!

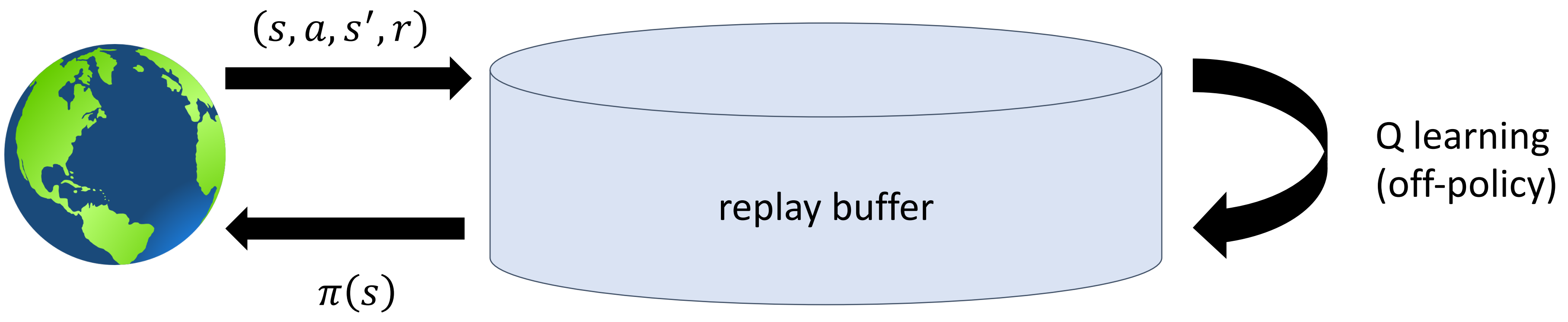
Offline Reinforcement Learning

- **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
    - $y_{i,k} \leftarrow r_{i,k} + \gamma \cdot \max_{a' \in A} Q_\theta(s_{i+1,k}, a')$
    - $\phi \leftarrow \phi - \alpha \cdot \frac{d}{d\theta}\left(Q_\theta(s_{i,k}, a_{i,k}) - y_{i,k}\right)^2$



Offline Reinforcement Learning

- **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
    - $y_{i,k} \leftarrow r_{i,k} + \gamma \cdot \max_{a' \in A} Q_\theta(s_{i+1,k}, a')$
    - $\phi \leftarrow \phi - \alpha \cdot \frac{d}{d\theta}\left(Q_\theta(s_{i,k}, a_{i,k}) - y_{i,k}\right)^2$



Lecture 23: Recommender Systems

CIS 4190/5190

Fall 2023

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).

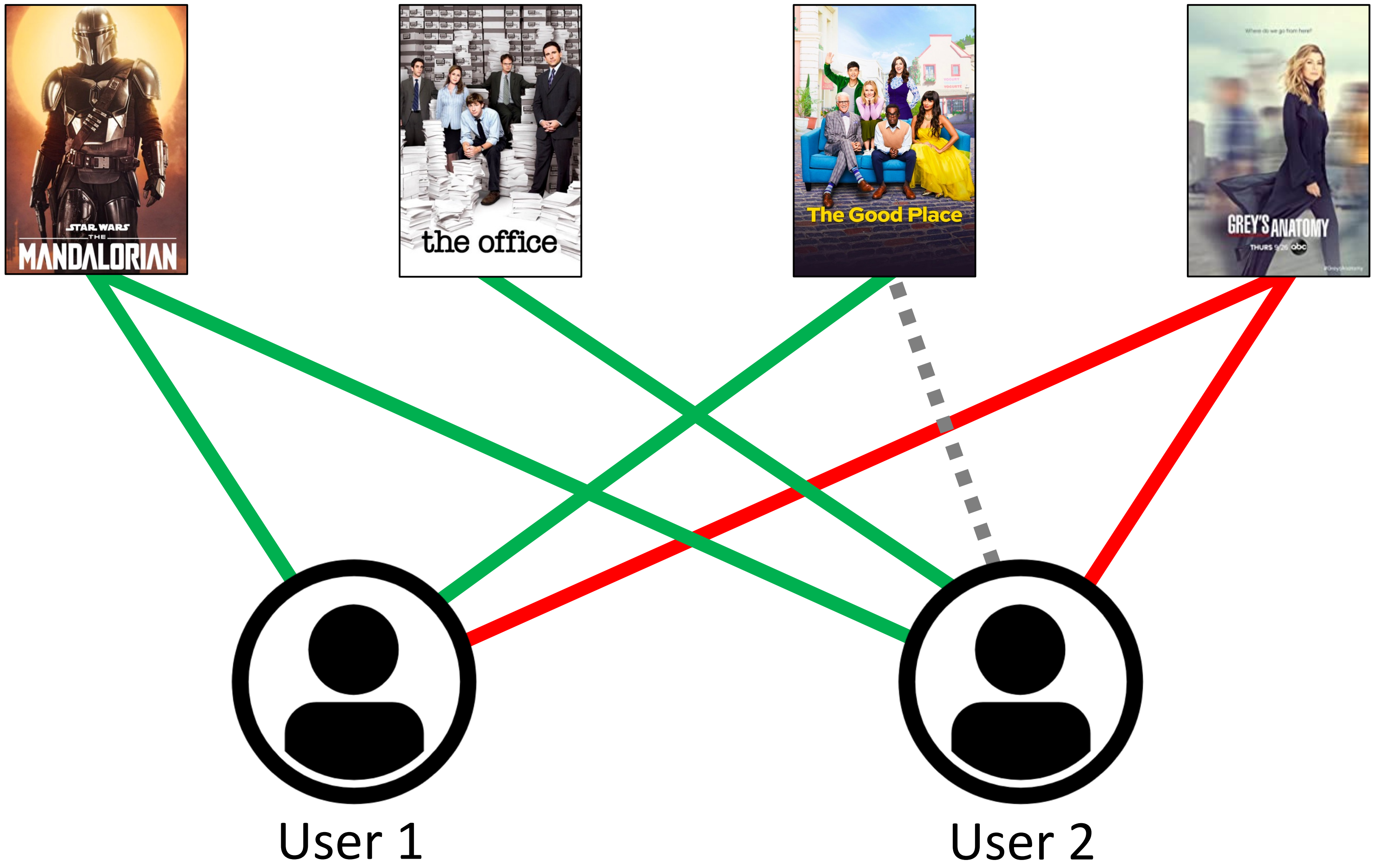
Popularity-Based Recommendation

- Just recommend whatever is currently popular

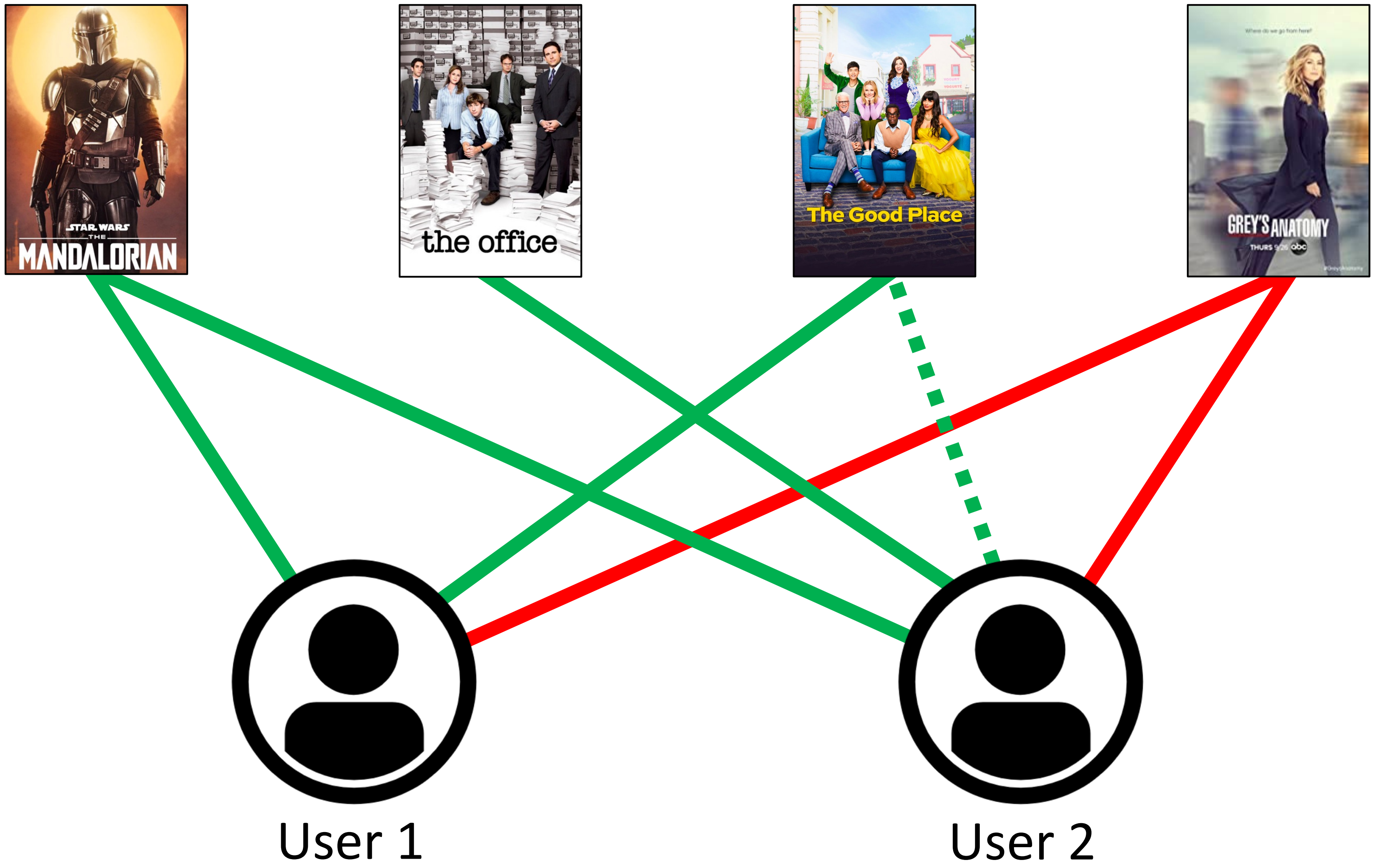
- Simple and effective, always try as a baseline

- Can be combined with more sophisticated techniques

Collaborative Filtering



Collaborative Filtering



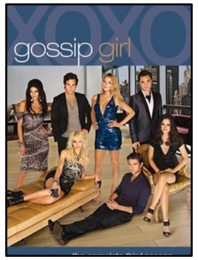
Collaborative Filtering

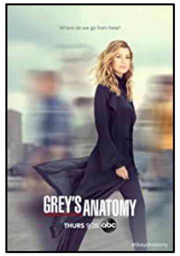
- **Given:**
  - Matrix $X_{i,k} = \begin{cases} \text{rating}_{i,k} & \text{if user}_i \text{ rated product}_k \\ \text{N/A} & \text{otherwise} \end{cases}$
  - Assume fixed set of n users and m products
  - **Not given any information about the products!**

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

Collaborative Filtering

Missing entries!

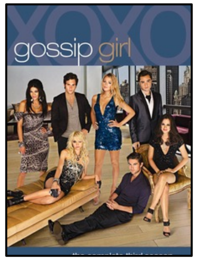


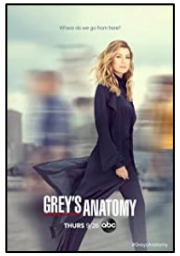


	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	…
Grace	4	5	4	1	5	3	…
Eric	1	4	5	1	5	3	…
Haren	5	5	5	1	3	4	…
Sai	1	2	5	4	3	5	…
Siyan	3	1	1	3	4	5	…
Nikhil	2	3	4	2	2	2	…
Felix	1	1	1	5	2	2	…

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

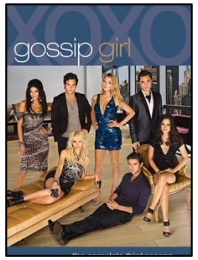
Step 1: Constructing User-Item Ratings

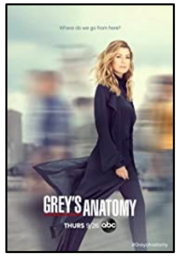
- Can use explicit ratings (e.g., Netflix)
- Can be implicitly inferred from user activity
  - User stops watching after 15 minutes
  - User repeatedly clicks on a video
- Feedback can vary in strength
  - **Weak:** User views a video
  - **Strong:** User writes a positive comment

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

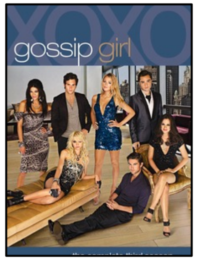


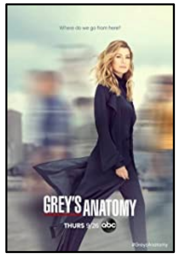


	Gossip Girl	The Office	The Mandalorian	Criminal Minds	The Good Place	Grey's Anatomy	...
Grace		5		1	5		...
Eric		4	5		5	3	...
Haren	5		5		3	4	...
Sai		2					...
Siyan	3	1		3		5	...
Nikhil				2	2		...
Felix	1		1		2		...

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

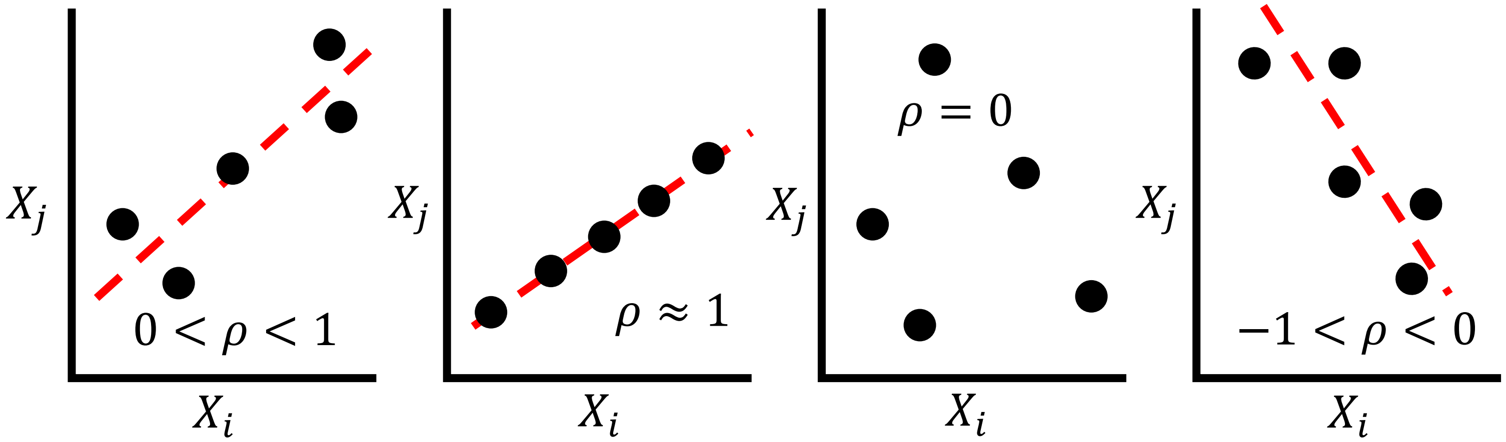
- **How to measure similarity?**
  - Distance $d(X_i, X_j)$, where X_i is vector of ratings for user i

- **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

Matrix Factorization

- **Model family:** Consider parameterization

$$X_{i,k} \approx U_i^\top V_k$$

  - Both $U_i \in \mathbb{R}^d$ and $V_k \in \mathbb{R}^d$ are parameters
  - U_i represents "features" for user i
  - V_k represents "features" for product k

Matrix Factorization

- **Loss function:**

$$L(U, V; X) = \sum_{i=1}^{n} \sum_{k=1}^{m} 1\left(X_{i,k} \neq \mathrm{N/A}\right) \cdot \left(X_{i,k} - U_i^\top V_k\right)^2$$

- **Optimizer:**
  - Can be minimized using gradient descent
  - **"Alternating" least squares:** Hold U fixed, then optimizing V is linear regression (and vice versa), so alternate between the two

Koren, et al. (2009) Matrix factorization techniques for recommender systems. *Computer* 42 (8), ACM.
https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf

Collaborative Filtering

- **Pros**
  - No domain knowledge needed, only user behavior
  - Captures that users may have diverse preferences

- **Cons**
  - Suffers when data is sparse
  - Does not consider item content, so cannot generalize to new items
  - Does not consider user features, so cannot generalize to new users

Content-Based Approaches

- **Step 1:** Manually construct feature vector U_i for item

- **Step 2:** Manually construct feature vector V_k for user

- **Step 3:** Train a model using supervised learning to predict the user's rating for the given item:

$$X_{i,j} \approx f_\beta(U_i, V_k)$$

Content-Based Approaches

- **Pros**
  - Incorporates external sources of knowledge on items/users to generalize
  - More explainable since recommendations are based on handcrafted features

- **Cons**
  - Requires domain knowledge and feature engineering
  - Narrow recommendations

Hybrid Approaches

- **Combine collaborative filtering with content-based approaches**
  - Ensemble different predictions
  - Concatenate collaborative filtering features with handcrafted features

- **Deep-learning based approaches**
  - Can be used with both approaches (or a combination)
  - Active area of research

Other Considerations

- **Challenges measuring utility**
  - Ratings can be misleading
  - Fake reviews/ratings are commonplace

- **Time-varying preferences**
  - User preferences change, item popularities change
  - Can upweight recent data (e.g., exponentially weighted moving average)

- **Evaluation**
  - **Offline:** Split users into train/test, and evaluate model on test users
  - **Online:** Split users into train/test, and run separate algorithms for each

What About New Users?

- Called the “cold start” problem

- **Feature-based approach**
  - Just featurize the user!

- **Collaborative filtering**
  - Need to collect ratings from the user!
  - Use multi-armed bandits