10-315 Notes
Recommender Systems, Clustering
v2.0

Carnegie Mellon University
Machine Learning Department

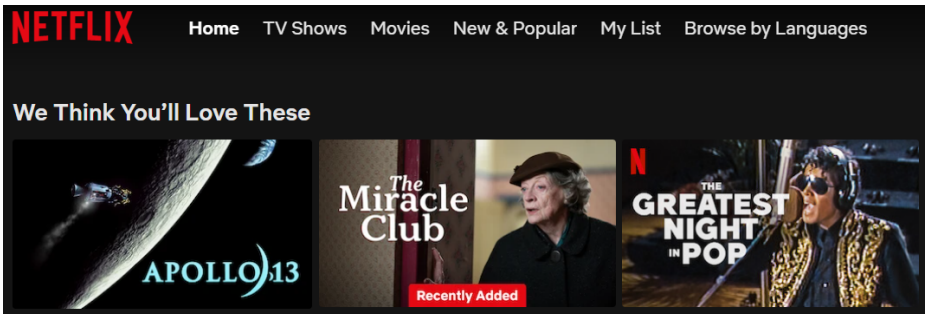
Contents

1 Recommender Systems

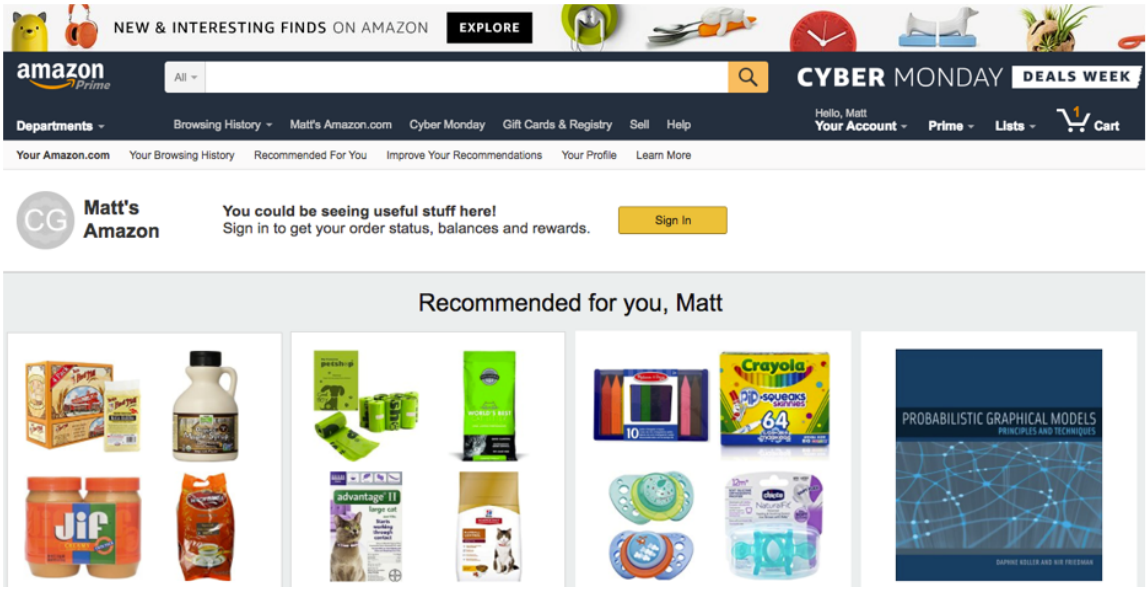
Recommender systems are a class of algorithms that predict which items (products) a user is most likely to be interested in. They are widely used in e-commerce, social media, and other online platforms to help users discover new products and services.

1.1 Examples

Example: Netflix movie ecommendations Netflix uses a recommender system to recommend movies to users to help users quickly find movies they are likely to enjoy (and of course, keep them happy using the product).



Example: Amazon product recommendations Amazon uses a recommender system to recommend products to users based on their browsing and purchasing history. This helps users discover new products and increases the likelihood of a purchase.



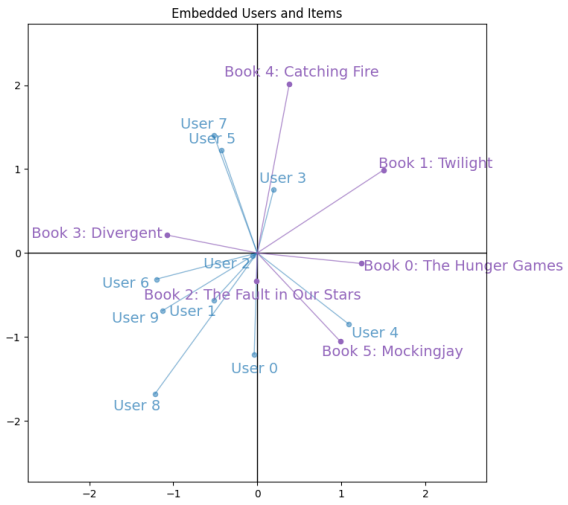
1.2 Latent Factor Model

One specific type of recommender system algorithm is the **latent factor model**. The idea behind latent factor models is to represent both users and items in a K-dimensional embedded space. Each user and item is represented by a K-dimensional vector, and the recommendation is made based on the similarity between the user and item vectors.

In the case where users have given ratings to items, the latent factor model tries to predict the ratings based on the learned user and item vectors. Specifically, we can try to predict the numerical rating of user i on item j as the dot product of the user and item vectors, $\hat{r} = \mathbf{u}_i^\top \mathbf{v}_j$.

This latent factor model can work even when we don't have any information or features about the users or the items other than a dataset of user-item ratings!

Here's a quick example of book items (purple) and users (blue) embedded in a $K = 2$ dimensional space. This is just the initial random parameters in U and V. As we train, it is likely that similar books will move closer to each other, as will readers with similar interests.



1.2.1 Optimization

The specific optimization problem is as follows:

- Input: N users, M items, and a dataset of user-item ratings $\mathcal{S} = \{(i, j, r)^{(1)}, (i, j, r)^{(2)}, \cdots\}$
- Hyperparameter: K, the size of the embedded space
- Parameters: $U \in \mathbb{R}^{N \times K}$ and $V \in \mathbb{R}^{M \times K}$, where the i^{th} row of U, $\mathbf{u}_i$, is the K-dimensional vector learned to represent user i and the j^{th} row of V, $\mathbf{v}_j$ is the K-dimensional vector learned to represent item j

We will use square error as the loss function to measure the different between actual rating from user i on item j, r, and the predicted rating $\hat{r} = \mathbf{u}_i^\top \mathbf{v}_j$. The objective function is then:

$$J(\mathbf{U}, \mathbf{V}) = \sum_{(i,j,r) \in \mathcal{S}} (r - \mathbf{u}_i^\top \mathbf{v}_j)^2$$

Once we compute the gradiens of the objective function with respect each user and item vector, we can run stochastic gradient descent to optimize the user and item vectors, where each data point is a single user-item rating tuple (i, j, r).

$$\mathbf{u}_i^{(t+1)} \leftarrow \mathbf{u}_i^{(t)} - \alpha \nabla_{\mathbf{u}_i} J(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$$
$$\mathbf{v}_j^{(t+1)} \leftarrow \mathbf{v}_j^{(t)} - \alpha \nabla_{\mathbf{v}_j} J(\mathbf{U}^{(t)}, \mathbf{V}^{(t)})$$

2 Unsupervised Learning: Clustering

Supervised learning is the process of learning a function that maps an input to an output based on example input-output pairs. In contrast, **unsupervised learning** is the process of learning a function that maps an input to an output based on input data without example output pairs.

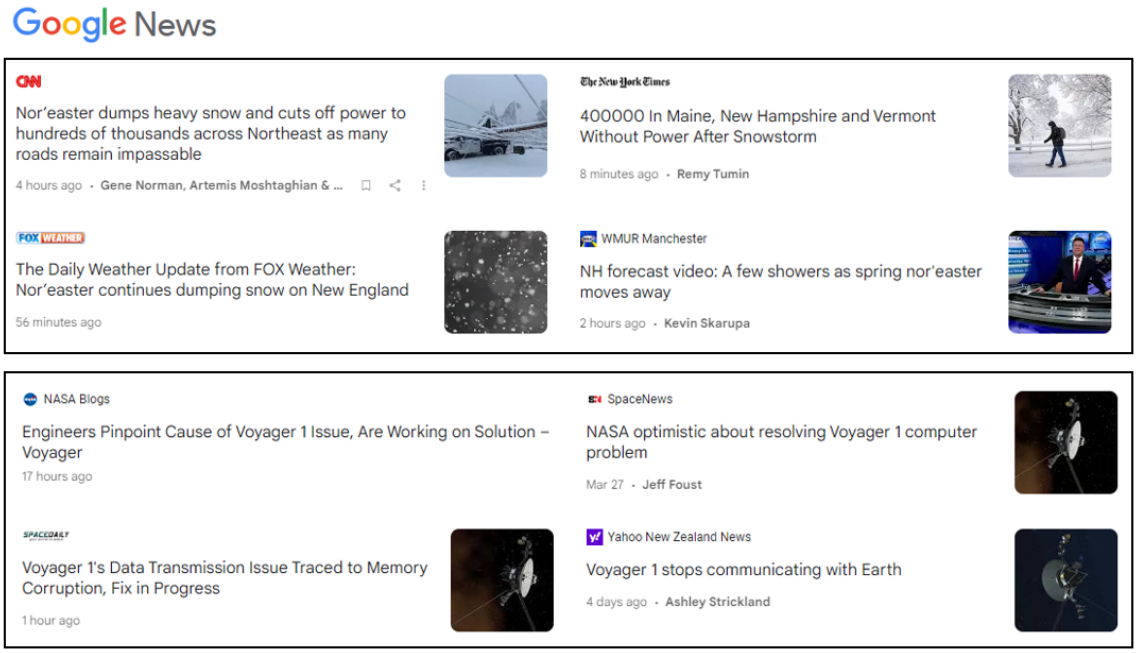
Clustering is a type of unsupervised learning that groups similar data points together. It is essentially classification when we don't have labeled data.

Even though we don't have labeled classes for our data points, clustering can be incredibly valuable. For example, clustering can be used to segment customers based on their purchasing behavior, to group similar documents together, or to identify patterns in data.

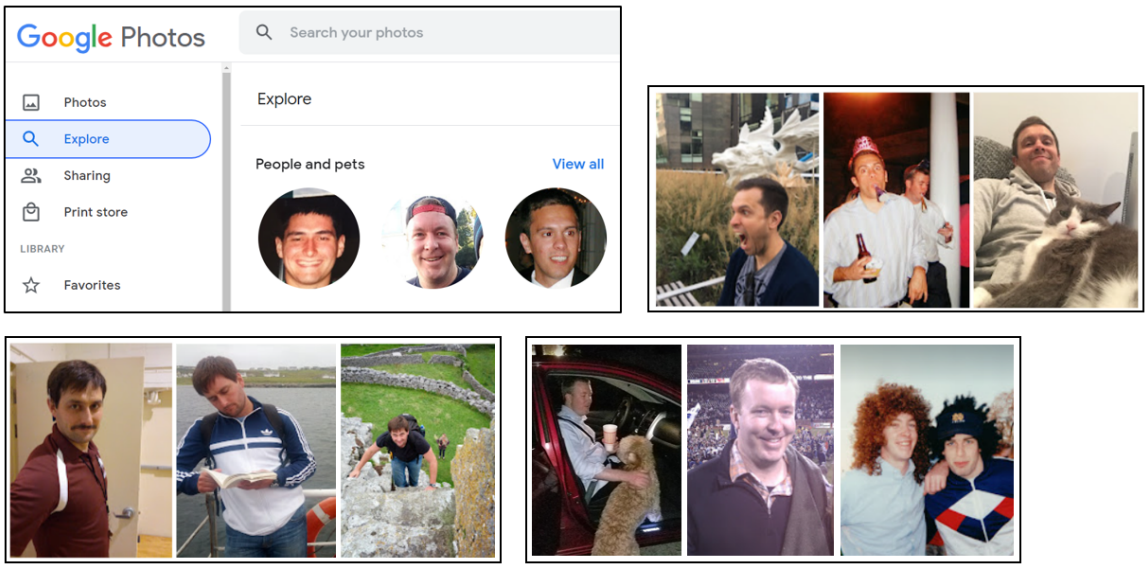
Clustering can also be use to analyze the underlying structure of the data or as a preprocessing step for a later task. For example, Google Photos can cluster photos without any labels, and then prompt a the user to label the cluster with the name of the person that appears in the cluster photos.

2.1 Examples

Example: Google News Google News uses clustering to group similar news articles together. When a new article is published, Google News uses clustering to determine which cluster the article belongs to. This allows Google News to group similar articles together and present them to the user.



Example: Google Photos Google Photos uses clustering to group similar photos together. Despite going out of your way not to identify your college roommates, Google Photos can still group photos containing them together.



2.2 K-means clustering

K-means clustering is a popular clustering algorithm that groups data points into K clusters.

Given a dataset of N unlabeled M-dimensional data points, $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^M$, the goal is to assign each point to a cluster such that we minimize the sum of the distances between each point and the center of its assigned cluster. The center of each cluster is the mean of all data points assigned to that cluster.

2.2.1 Algorithm

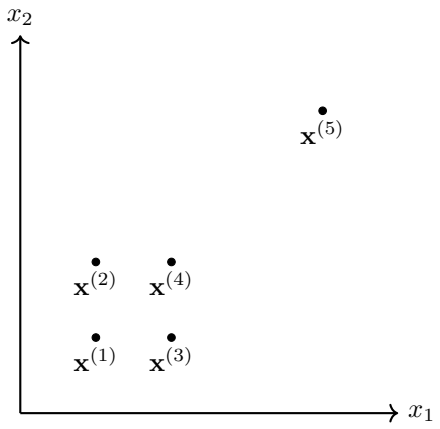
The K-means clustering algorithm is as follows:

Plain english version:

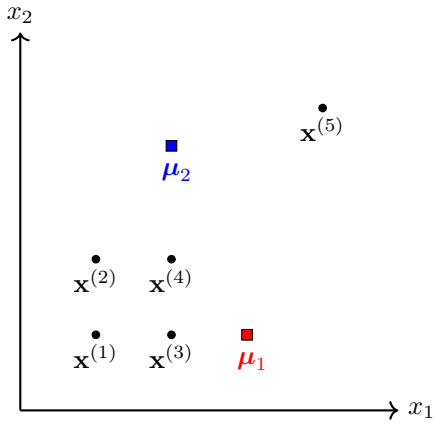
1. Randomly initialize K cluster centers
2. Assign each point in the training data to the cluster with the closest center
3. Update the cluster centers by computing the mean of all data points assigned to each cluster
4. Repeat steps 2 and 3 until the cluster centers no longer change

Repeated with a few more details:

1. Randomly initialize K M-dimentional points to represent the initial cluster means, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_K$.
2. Assign each data point a cluster index, $z^{(i)} \in \{1, 2, \ldots, K\}$, by finding the cluster centroid that is closest to the data point. That is, $z^{(i)} = \text{argmin}_k \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
3. Update the cluster means by computing the mean of all data points assigned to each cluster, $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \mathbf{x}^{(i)} \mathbb{I}\{z^{(i)} = k\}$, where N_k is the number of data points assigned to cluster k.
4. Repeat steps 2 and 3 until the cluster means no longer change

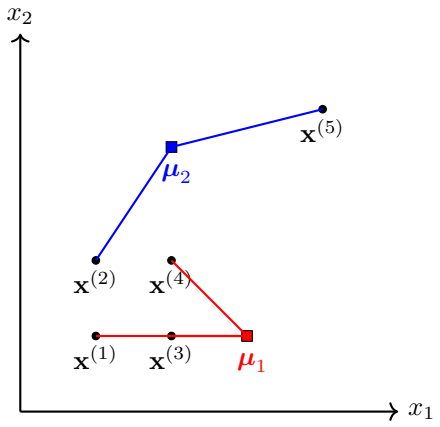


Step 1: Random Initialization We randomly initialize the cluster centers. For example, we might initialize the cluster centers as follows:



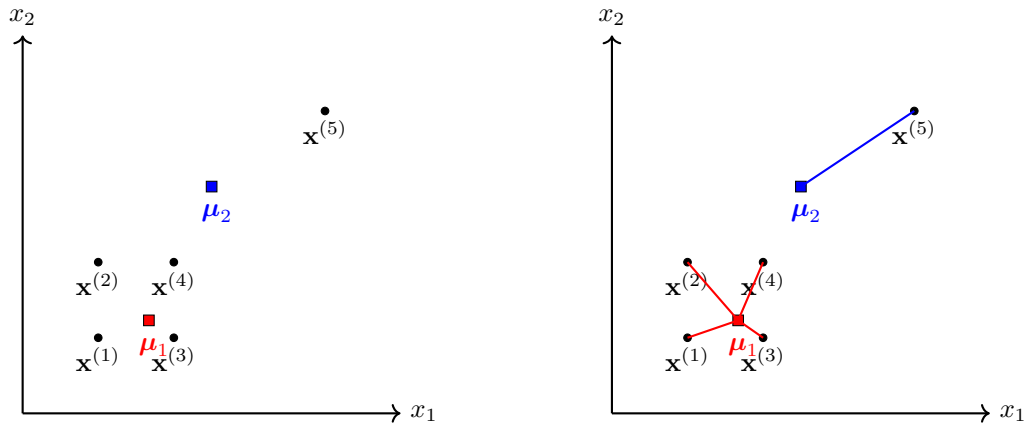
Where the 2 cluster centers are $\boldsymbol{\mu}_1 = [3, 1.5]^\top$ and $\boldsymbol{\mu}_2 = [1, 3]^\top$.

Step 2: Assign Data Points to Clusters We assign each data point to the cluster with the closest center. For example, we would assign the data points as follows:

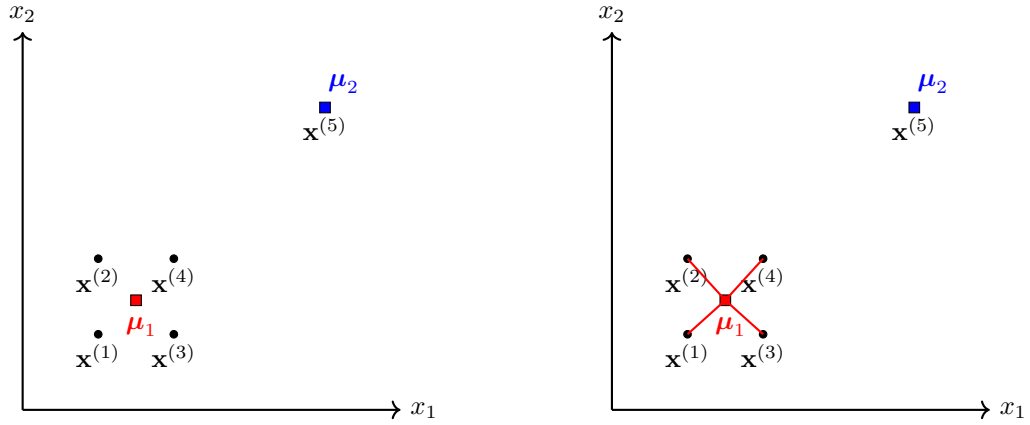


Where the data points are assigned to clusters as follows: $\{z^{(1)} = 1, z^{(2)} = 2, z^{(3)} = 1, z^{(4)} = 1, z^{(5)} = 2\}$.

Step 3: Update Cluster Centers We update the cluster centers by computing the mean of all data points assigned to each cluster. For example, we would update the cluster centers as follows:



Repeat Steps 2 (above, right) and 3 (below, left):



Repeat Step 2 again (above, right):

We could repeat step three again, but the cluster assignments didn't change, so the cluster means will stay the same.

2.2.2 Optimization Formulation

Our goal is to minimize the sum of the distances between each point and the center of its assigned cluster. We can formulate this as an optimization problem:

- Input: $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^M$
- Output: $z^{(i)} \in \{1, 2, \ldots, K\}$, the cluster assignment for each data point i
- Output: $\boldsymbol{\mu}_k \in \mathbb{R}^M$ be the center of cluster k

$$\min_{\{z^{(i)}\},\{\boldsymbol{\mu}_k\}} \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_{z^{(i)}}\|_2^2$$

2.2.3 Alternating Minimization

The objective is actually quite difficult to optimize directly. Instead, we can use an **alternating minimization** approach. We alternate between two steps: (1) fixing the cluster assignments and updating the cluster centers, and (2) fixing the cluster centers and updating the cluster assignments.

1. **Fix cluster assignments, update cluster centers:** Given the cluster assignments, we can update the cluster centers by computing the mean of all data points assigned to each cluster.

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \mathbf{x}^{(i)} \; \mathbb{I}\{z^{(i)} = k\}$$

2. **Fix cluster centers, update cluster assignments:** Given the cluster centers, we can update the cluster assignments by assigning each data point to the cluster with the closest center.

$$z^{(i)} = \text{argmin}_k \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|_2^2$$