

10-315 Notes

PCA, Recommender Systems, Clustering

Carnegie Mellon University
Machine Learning Department

Contents

1 PCA Math Background	2
1.1 PCA Motivation: Dimensionality Reduction	2
1.2 Projections	2
1.2.1 Scalar Projection	2
1.2.2 Vector Projection	3
1.3 Rotating Data	3
1.3.1 Rotation	3
1.3.2 Projection Matrix	4
1.4 Covariance Matrix	5
2 Recommender Systems	7
2.1 Examples	7
2.2 Latent Factor Model	8
2.2.1 Optimization	8
3 Unsupervised Learning: Clustering	9
3.1 Examples	9
3.2 K-means clustering	10
3.2.1 Algorithm	10
3.2.2 Optimization Formulation	12
3.2.3 Alternating Minimization	13

1 PCA Math Background

1.1 PCA Motivation: Dimensionality Reduction

Dimensionality reduction is a key technique in machine learning and data analysis. It aims to map the original dataset to a smaller dimension, where each sample has fewer features. Dimensionality reduction can be thought of as reducing the dimensions of the data matrix \mathbf{X} from $N \times M$ to $N \times K$ where $K < M$, hence the name dimensionality reduction. By reducing the number of features, we can

- Reduce computational cost and therefore increase the training speed of our models because there are fewer features to train on
- Improve model performance by excluding irrelevant or noisy features
- Reconstruct high-dimensional data in two or three dimensions for better visualization

Principal component analysis is just one dimensionality reduction strategy, which we will cover in detail below.

1.2 Projections

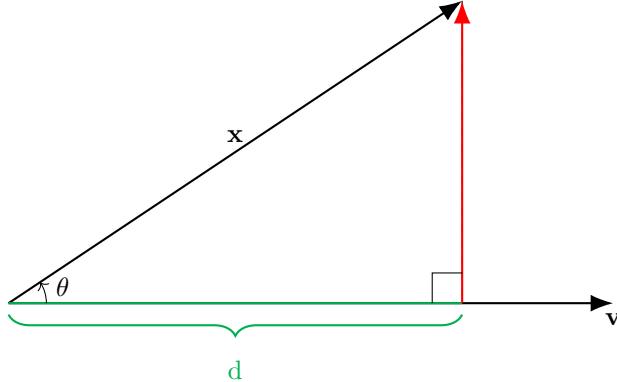
1.2.1 Scalar Projection

Given two vectors in \mathbb{R}^N denoted \mathbf{x} and \mathbf{v} , the scalar projection of \mathbf{x} onto \mathbf{v} is defined as:

$$d = \frac{\mathbf{v}^\top \mathbf{x}}{\|\mathbf{v}\|_2}$$

Note that if we assume that \mathbf{v} is a unit vector, i.e., $\|\mathbf{v}\|_2 = 1$, the projection formula is *much* simpler:

$$d = \mathbf{v}^\top \mathbf{x}$$



The scalar projection, d , is the length of the vector \mathbf{x} projected onto \mathbf{v} . We can prove this geometrically (recall that $\mathbf{v}^\top \mathbf{x} = \mathbf{v} \cdot \mathbf{x} = \|\mathbf{v}\|_2 \|\mathbf{x}\|_2 \cos(\theta)$):

$$\begin{aligned}\cos(\theta) &\triangleq \frac{d}{\|\mathbf{x}\|_2} \\ d &= \|\mathbf{x}\|_2 \cos(\theta) \\ d &= \frac{\mathbf{v}^\top \mathbf{x}}{\|\mathbf{v}\|_2}\end{aligned}$$

1.2.2 Vector Projection

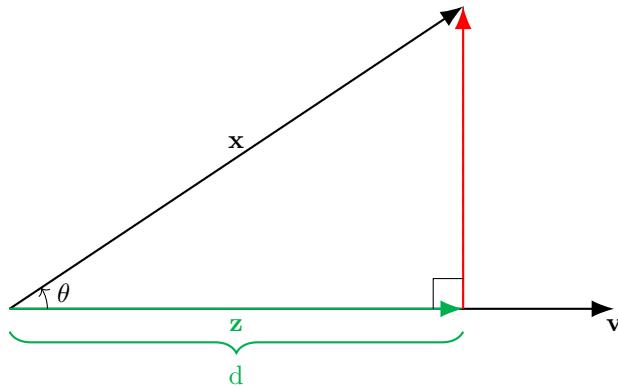
Recall from linear algebra, the definition of vector projection. Given two vectors in \mathbb{R}^N denoted \mathbf{x} and \mathbf{v} , the vector projection of \mathbf{x} onto \mathbf{v} , or $\text{proj}_{\mathbf{v}} \mathbf{x}$, is defined as:

$$\mathbf{z} = \frac{\mathbf{v}^\top \mathbf{x}}{\|\mathbf{v}\|_2} \frac{\mathbf{v}}{\|\mathbf{v}\|_2}$$

Note that if we assume that \mathbf{v} is a unit vector, i.e., $\|\mathbf{v}\|_2 = 1$, the projection formula is *much* simpler:

$$\mathbf{z} = (\mathbf{v}^\top \mathbf{x}) \mathbf{v}$$

The projected vector \mathbf{z} lies in the direction of \mathbf{v} and represents the component of \mathbf{x} in the direction of \mathbf{v} . We can think of the projection \mathbf{z} as a restriction of \mathbf{x} to the \mathbf{v} -axis. The vector projection is visualized below:



Geometrically, the vector projection can be thought of as the vector \mathbf{z} above, when both vectors are anchored at the origin.

To develop a better understanding of how projections work and what they mean, feel free to use this [Desmos](#) link. Moving around the u and v points in the Desmos link will change the projected vector.

1.3 Rotating Data

Now that we have established an intuition for projections, we can use this concept for rotating data. Recall that a basis for \mathbb{R}^N is a set of unit vectors $\mathcal{V} = \{\mathbf{v}_1, \dots, \mathbf{v}_N\}$ such that any vector $\mathbf{x} \in \mathbb{R}^N$ can be written as a linear combination of $\mathbf{v}_1, \dots, \mathbf{v}_N$. When the basis vectors are perpendicular to each other, they can also be thought of as the axes of our data. For example, the basis vectors $\mathbf{v}_1 = [1, 0]^\top$ and $\mathbf{v}_2 = [0, 1]^\top$ represent the x -axis and y -axis in the standard Cartesian coordinate system. You should observe that vector $\mathbf{x} \in \mathbb{R}^2$ can be written as $z_1 \mathbf{v}_1 + z_2 \mathbf{v}_2$ where z_1 and z_2 are scalars.

1.3.1 Rotation

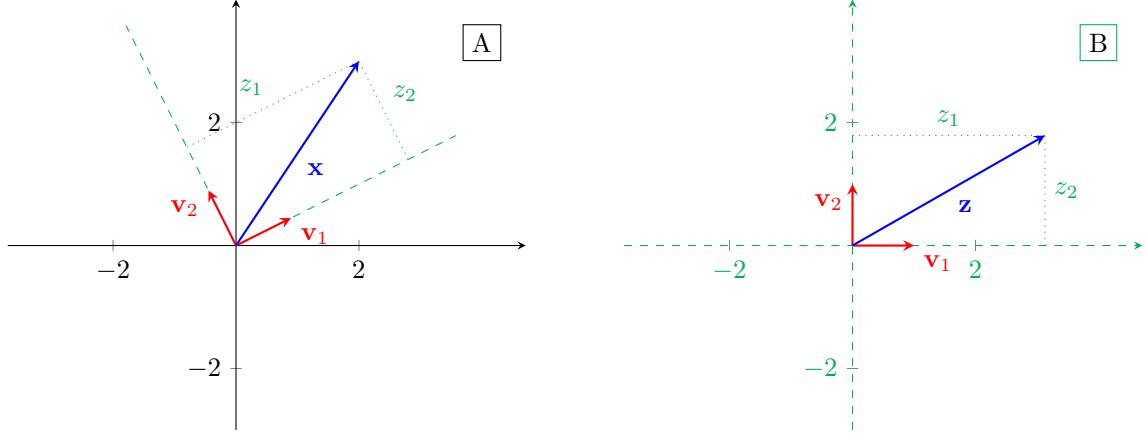
We can transform the data to be aligned to any set of axes by projecting onto the corresponding set of basis vectors. Consider the $\mathbf{x} = [2, 3]^\top$. Now we will project \mathbf{x} onto the basis vectors

$$\mathbf{v}_1 = \left[\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right]^\top \text{ and } \mathbf{v}_2 = \left[\frac{-1}{\sqrt{5}}, \frac{2}{\sqrt{5}} \right]^\top$$

Note that \mathbf{v}_1 and \mathbf{v}_2 are once again unit vectors. Performing the projection calculation, we have that

$$z_1 = \mathbf{v}_1^\top \mathbf{x} = \frac{7}{\sqrt{5}} \approx 3.13$$

$$z_2 = \mathbf{v}_2^\top \mathbf{x} = \frac{4}{\sqrt{5}} \approx 1.79$$



1.3.2 Projection Matrix

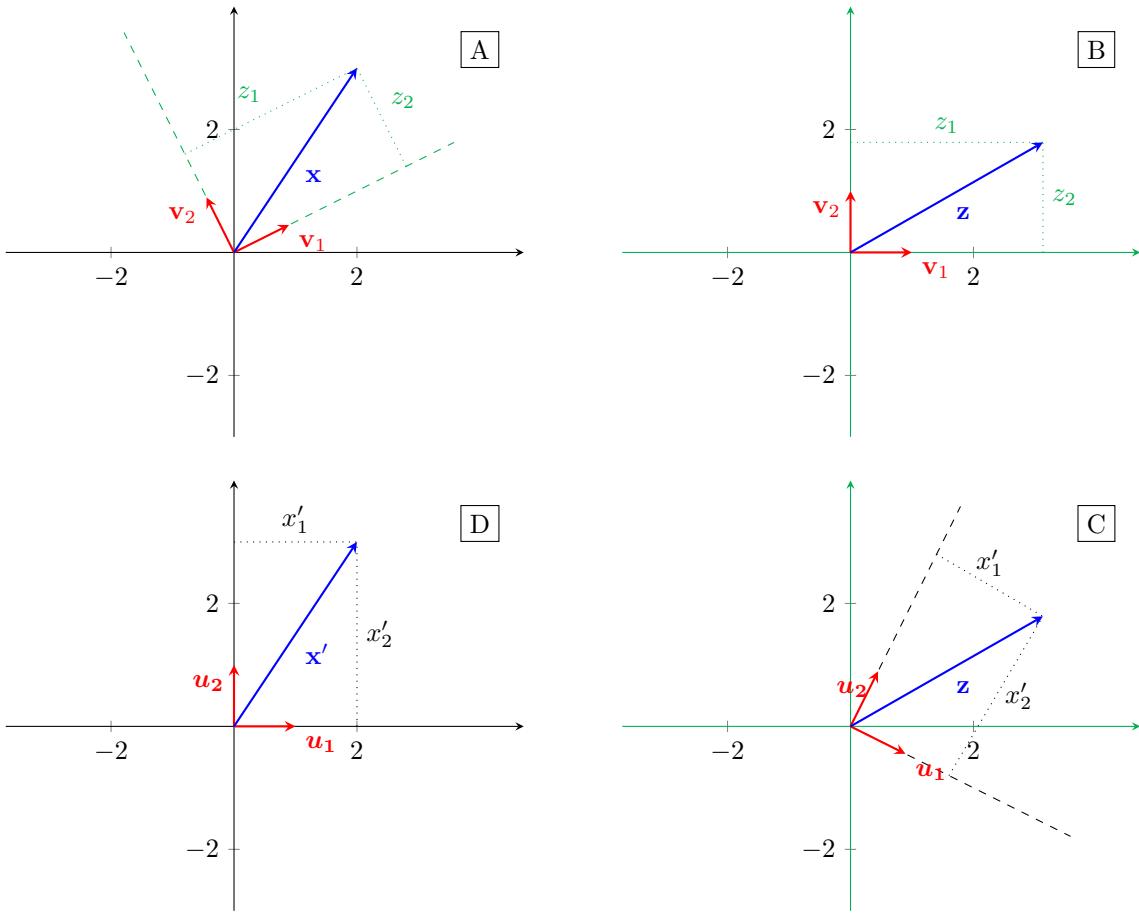
You may have noticed that when the \mathbf{v}_i are unit vectors, z_i is just the dot product of \mathbf{v}_i and \mathbf{x} . Let us now define V as:

$$V = \begin{bmatrix} -\mathbf{v}_1^\top & - \\ \vdots & \\ -\mathbf{v}_N^\top & - \end{bmatrix}$$

Then it follows that $\mathbf{z} = V\mathbf{x}$ where the matrix V is called the projection matrix. Sometimes we want to reconstruct the original \mathbf{x} from the projection \mathbf{z} . Let $\mathbf{x}' = V^\top \mathbf{z} = V^\top V\mathbf{x}$. When the projection matrix V is a square matrix, we are projecting \mathbf{x} into a space with the same number of dimensions. Then $V^\top V = I \implies \mathbf{x}' = \mathbf{x}$, so we can reconstruct \mathbf{x} perfectly. When we project into a lower-dimensional space, V is not a square matrix and thus we cannot reconstruct \mathbf{x} perfectly.

Now, we can consider reversing that rotation through a different rotation matrix. Define $U = V^\top$. We can see that by applying the rotation U onto \mathbf{z} , we get $\mathbf{x}' = U\mathbf{z}$. Note that this is simply another rotation; however, because we have specifically chosen $U = V^{-1} = V^\top$, we get that $\mathbf{x}' = \mathbf{x}$.

Intuitively, if \mathbf{x} represents the data in terms of the standard basis $[1, 0]^\top$ and $[0, 1]^\top$, then \mathbf{z} represents the data in terms of $\mathbf{v}_1 = \left[\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right]^\top$ and $\mathbf{v}_2 = \left[\frac{-1}{\sqrt{5}}, \frac{2}{\sqrt{5}} \right]^\top$. We can visualize what is happening below. Moving from plot A to plot B represents applying the rotation matrix V onto \mathbf{x} to get \mathbf{z} . Then, by moving from plot C to D, we can see the reverse step by applying the rotation matrix U onto \mathbf{z} to get \mathbf{x}' .



1.4 Covariance Matrix

You may be wondering what we mean by the variance of our dataset. In machine learning, when our dataset \mathcal{D} has multiple features, each sample is represented as a column vector $\mathbf{x}^{(i)} \in \mathbb{R}^M$. Then we can represent the entire dataset as a matrix $\mathbf{x} = [\mathbf{x}^{(1)} \dots \mathbf{x}^{(N)}]^\top$.

In this context, variance measures the variability of each feature in the dataset, as well as the relationships between different features. We will now introduce the covariance matrix.

The covariance matrix $\Sigma \in \mathbb{R}^{M \times M}$ is a square matrix that summarizes the covariance between each pair of features in the dataset. The size of the matrix is determined by the number of features.

- **Diagonal Elements:** The diagonal elements of the covariance matrix represent the variances of each feature. In other words, they represent how spread out the values are for a specific feature. Mathematically, the variance of feature m is $\frac{1}{N} \sum_{i=1}^N (x_m^{(i)} - \mu_m)^2$ where $x_m^{(i)}$ is the value of feature m for the i -th data point and μ_m is the mean of feature m .
- **Off-Diagonal Elements:** The off-diagonal elements of the covariance matrix represent the covariance between different features. Covariance is a measure of how two features vary together. If the covariance is positive, it means that when one feature increases, the other tends to increase as well. If it's negative, it means that when one feature increases, the other tends to decrease. Mathematically, the covariance between feature j and feature k is $\frac{1}{N} \sum_{i=1}^N (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$.

Generally, we assume that our data is centered and scaled for each feature, in other words $\mu_m = \sum_{i=1}^N x_m^{(i)} = 0$.

Because each feature of our dataset has a mean of zero, and $\Sigma_{jk} = \frac{1}{N} \sum_{i=1}^N (x_j^{(i)} - \mu_j)(x_k^{(i)} - \mu_k)$, the covariance matrix simplifies to

$$\Sigma = \frac{1}{N} \mathbf{x}^\top \mathbf{x}$$

2 Recommender Systems

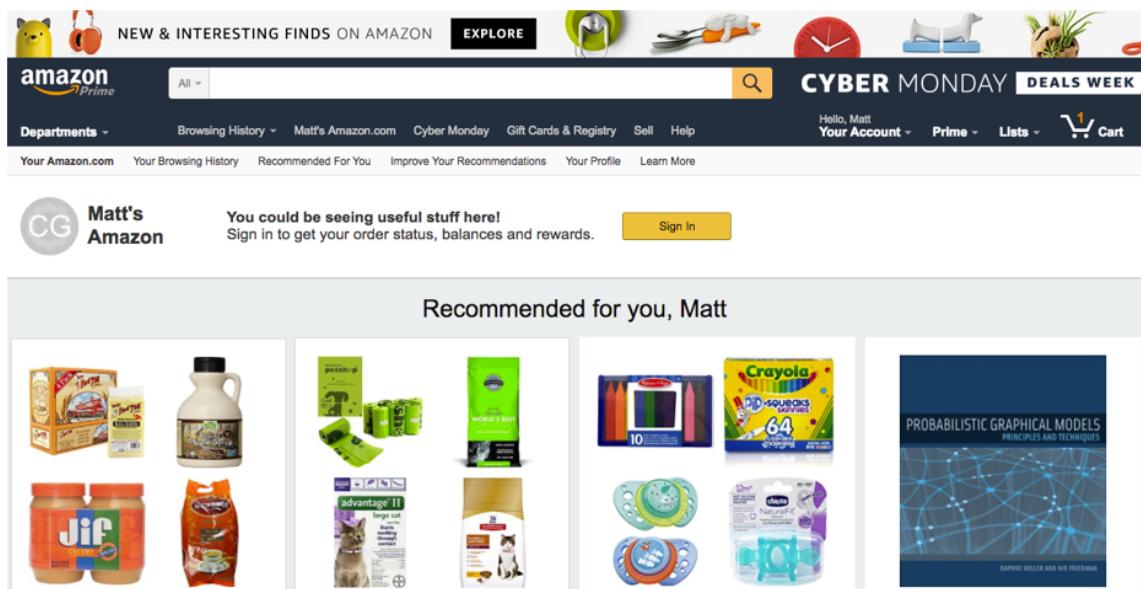
Recommender systems are a class of algorithms that predict which items (products) a user is most likely to be interested in. They are widely used in e-commerce, social media, and other online platforms to help users discover new products and services.

2.1 Examples

Example: Netflix movie recommendations 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.



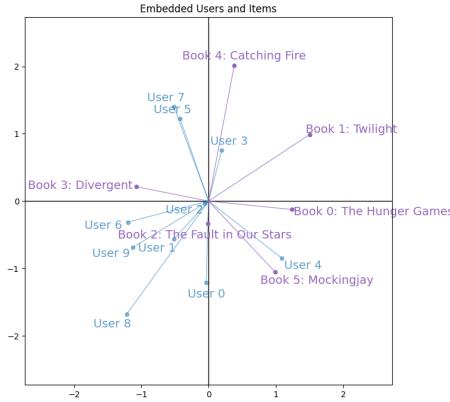
2.2 Latent Factor Model

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. These are 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.



2.2.1 Optimization

The specific optimization problem is as follows:

- Input: N users, M items, and a dataset of user-item ratings $\mathcal{S} = \{(i, j, r)^{(1)}, (i, j, r)^{(2)}, \dots\}$
- 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 difference between an 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 gradients of the objective function with respect to 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) .

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

3 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 used 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 the user to label the cluster with the name of the person that appears in the cluster photos.

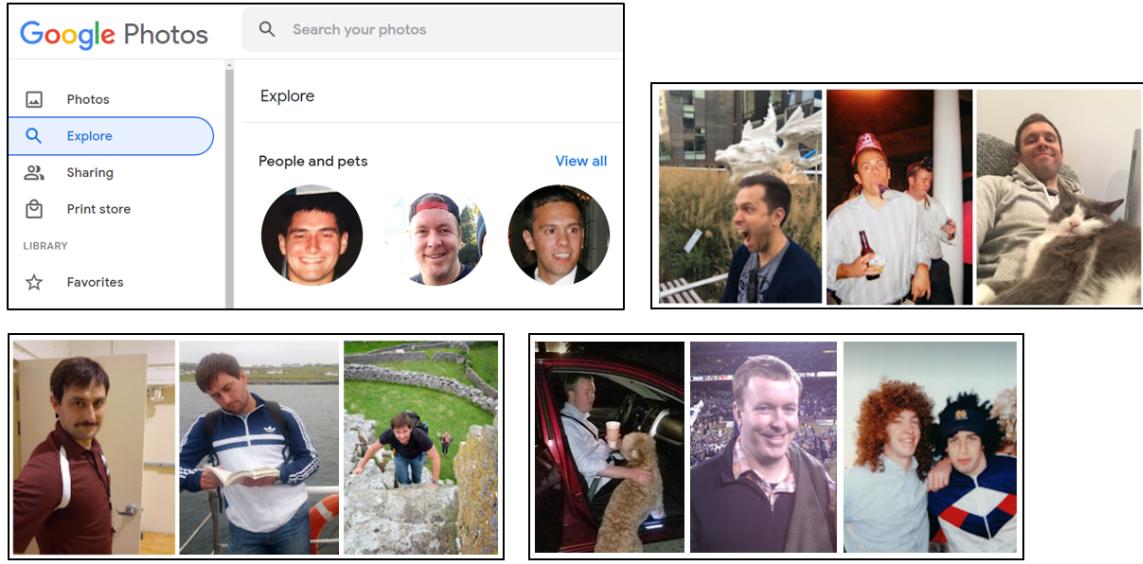
3.1 Examples

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.

The screenshot shows the Google News homepage. At the top, the "Google News" logo is visible. Below it, there are two main sections of news cards:

- Top Section:** Contains four news items:
 - ON**: "Nor'easter dumps heavy snow and cuts off power to hundreds of thousands across Northeast as many roads remain impassable" by Gene Norman, Artemis Moshtaghian & ... (4 hours ago). Includes a small image of a snow-covered street.
 - The New York Times**: "400000 In Maine, New Hampshire and Vermont Without Power After Snowstorm" by Remy Tumin (8 minutes ago). Includes a small image of a person walking in a snowy area.
 - FOX WEATHER**: "The Daily Weather Update from FOX Weather: Nor'easter continues dumping snow on New England" by Kevin Skarupa (56 minutes ago). Includes a small image of falling snow.
 - WMUR Manchester**: "NH forecast video: A few showers as spring nor'easter moves away" by Jeff Foust (2 hours ago). Includes a small image of a weather forecast screen.
- Bottom Section:** Contains two news items:
 - NASA Blogs**: "Engineers Pinpoint Cause of Voyager 1 Issue, Are Working on Solution - Voyager" by Jeff Foust (17 hours ago). Includes a small image of the Voyager 1 probe.
 - SpaceNews**: "NASA optimistic about resolving Voyager 1 computer problem" by Jeff Foust (Mar 27). Includes a small image of the Voyager 1 probe.

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.



3.2 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.

3.2.1 Algorithm

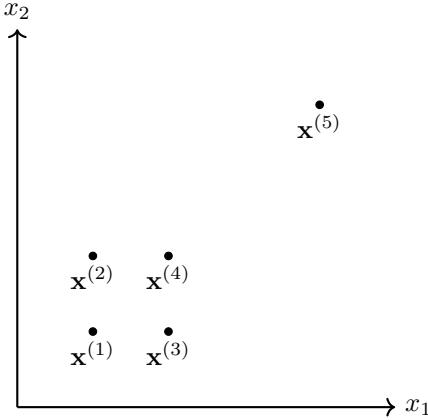
The K-means clustering algorithm is as follows:

Plain english version:

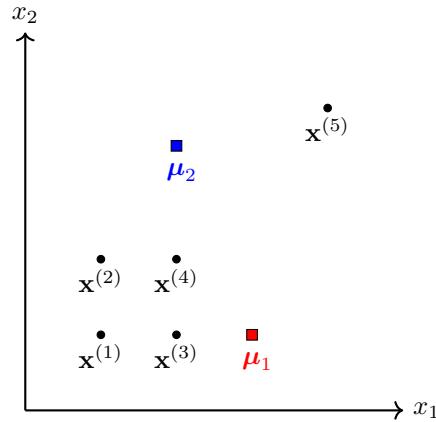
1. Randomly initialize K cluster centers
2. Assign each point in the training data to the cluster with the closest center
3. Update the cluster centers by computing the mean of all data points assigned to each cluster
4. Repeat steps 2 and 3 until the cluster centers no longer change

Repeated with a few more details:

1. Randomly initialize K M -dimensional points to represent the initial cluster means, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_K$.
2. Assign each data point a cluster index, $z^{(i)} \in \{1, 2, \dots, K\}$, by finding the cluster centroid that is closest to the data point. That is, $z^{(i)} = \operatorname{argmin}_k \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2$
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 (i.e. cluster centers converge)

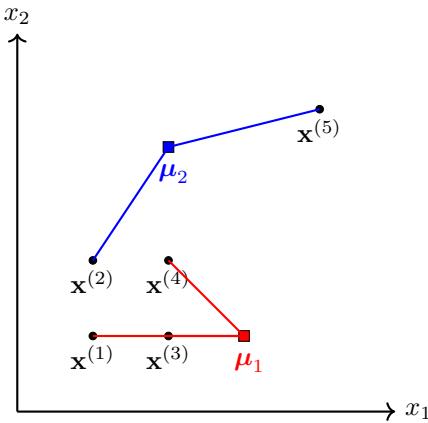


Step 1: Random Initialization We randomly initialize the cluster centers. For example, we might initialize the cluster centers as follows:



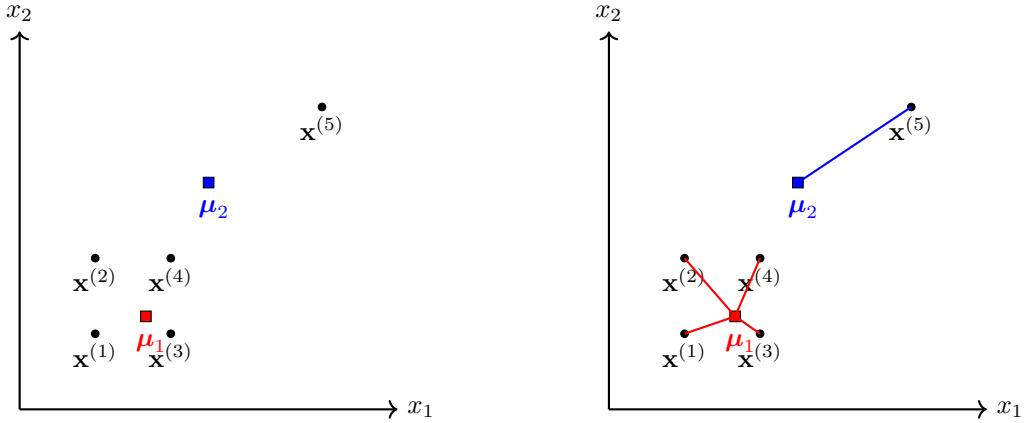
Where the 2 cluster centers are $\mu_1 = [3, 1.5]^\top$ and $\mu_2 = [1, 3]^\top$.

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:

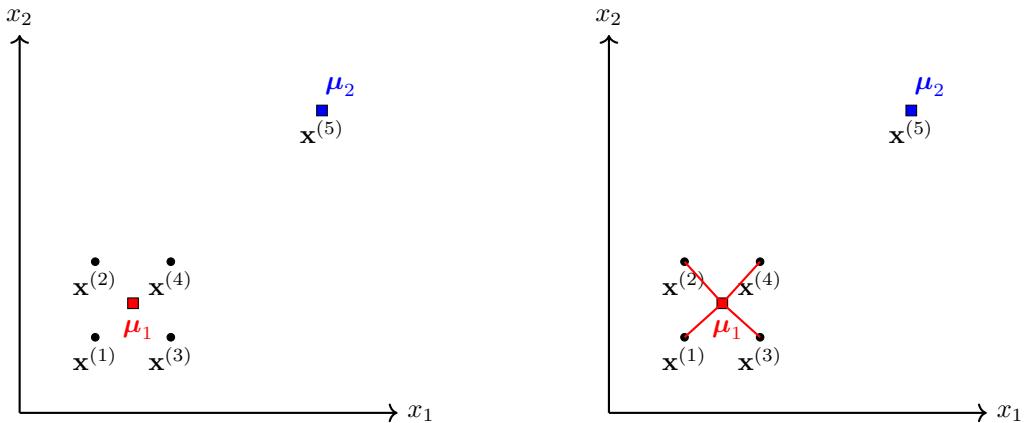


Where the data points are assigned to clusters as follows: $\{z^{(1)} = 1, z^{(2)} = 2, z^{(3)} = 1, z^{(4)} = 1, z^{(5)} = 2\}$.

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.

3.2.2 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: $z^{(i)} \in \{1, 2, \dots, K\}$, the cluster assignment for each data point i
- Output: $\boldsymbol{\mu}_k \in \mathbb{R}^M$ be the center of cluster k

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

3.2.3 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.

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