1 Definitions to Go

1.1 PCA

1. **Dimensionality Reduction:** The transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data. Dimensionality reduction is useful because time and space can be saved by using fewer features, especially when dealing with a large dataset.

2. **Principal Component Analysis:** A dimensionality reduction method that transforms a set of features in a dataset into a smaller number of features, called principal components, while also trying to retain as much information from the original dataset as possible. The principal components are uncorrelated, and the first principal component will explain the most variance in the original dataset.

   The algorithm for PCA is as follows, for some input dataset X_{orig} and integer K:

   (a) Center the data using $X = X_{orig} - \mu$

   (b) $V = \text{eigenvectors}(X^T X)$

   (c) Keep only the top K eigenvectors: V_K

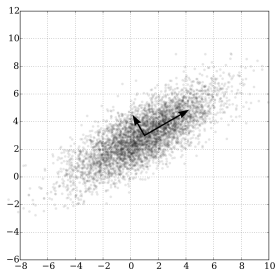
3. **Lagrange Multiplier:** A simple and elegant method of finding the local minima or local maxima of a function subject to equality or inequality constraints. It allows us to wrap the constraint into the objective function.

4. **Eigenvalue & Eigenvector:** For a matrix $A_{N\times N}$ and column vector $\mathbf{v}_{N\times 1}$, we consider $\mathbf{v}$ to be an eigenvector of A and $\lambda \neq 0$ to be an eigenvalue of A iff $A\mathbf{v} = \lambda\mathbf{v}$. Note how for a given eigenvalue, we can have infinite eigenvectors associated with it by taking constant multiples of $\mathbf{v}$. To be consistent, we are usually concerned with the eigenvector that's an unit vector (i.e. $\|\mathbf{v}\|_2 = 1$).

5. **Orthogonal Matrix:** Matrix $A_{N\times N}$ is considered orthogonal if all the columns and rows are orthogonal. A special and a very useful property about orthogonal matrices is that their transpose is equal to their inverse i.e. $A^{-1} = A^T$. We use this property when we unrotate back our transformations to the original data. If the columns are unit length, we call the matrix orthonormal.

6. **Eigen Value Decomposition:** If a matrix $A_{N\times N}$ has a full set of eigenvalues, then we can write $A = X\Lambda X^{-1}$ where $\Lambda_{N\times N}$ is a diagonal matrix with eigenvalues on the diagonal and $X_{N\times N}$ is the matrix whose columns are the eigenvectors corresponding to the eigenvalues in Λ. If A is symmetric, we have that X is orthogonal.

7. **Singular Value Decomposition (SVD):** the factorization of a matrix $A_{M\times N}$ into three matrices U, S, V^T. U is defined as an $M \times N$ matrix of the orthonormal eigenvectors of AA^T. S is defined as an $N \times N$ diagonal matrix of the singular values, which are the square roots of the eigenvalues of $A^T A$. Finally V^T is the transpose of a $N \times N$ matrix containing the orthonormal eigenvectors of $A^T A$. When we have that $M >> N$, we can use a more efficient version where we scale the matrices to $Rank(A) = min(M, N) = N$.



1.2 Recommender Systems

1. **Recommender Systems**: 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. **Latent Factor Model**: A specific type of recommender system algorithm. 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, $\hat{r} = \mathbf{u}_i^\top \mathbf{v}_j$.

   - 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 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:

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

1.3 K-Means

1. **K-Means**: An unsupervised machine learning algorithm used for clustering data. The algorithm aims to partition a given dataset into K clusters, where K is a hyperparameter.

2. **K-Means Algorithm**:
   Iteratively:

   (a) Assign data points to the closest cluster mean

   (b) Update the mean location based on the newly assigned points

   This converges when the cluster assignments and the means no longer change.

3. **K-Means Optimization**: Our goal is to minimize the sum of the distances between each point and the center of its assigned cluster.

   - 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: $\boldsymbol{\mu}_k \in \mathbb{R}^M$, the center of cluster k

   The objective function is then:

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

4. **Alternating Minimization**: The objective above is actually quite difficult to optimize directly. Instead, we can use an alternating minimization approach.

(a) **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.

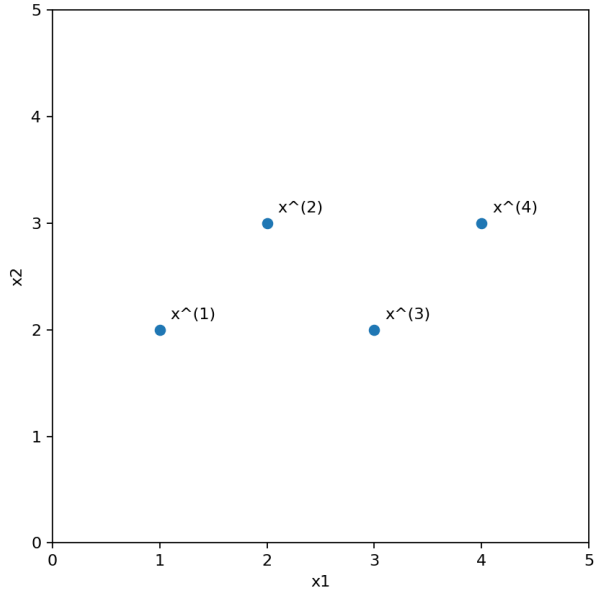
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \mathbf{x}^{(i)} \; \mathbb{I}\{z^{(i)} = k\}$$

(b) **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.

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

2 PCA Walkthrough

Consider dataset $\mathcal{D} = \{\mathbf{x}^{(1)} = \begin{bmatrix}1\\2\end{bmatrix}, \mathbf{x}^{(2)} = \begin{bmatrix}2\\3\end{bmatrix}, \mathbf{x}^{(3)} = \begin{bmatrix}3\\2\end{bmatrix}, \mathbf{x}^{(4)} = \begin{bmatrix}4\\3\end{bmatrix}\}$. A visualization of the dataset is as below.



1. Centering is crucial for PCA. We must preprocess data so that all features have zero mean before applying PCA, i.e.

$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} = \vec{0}$$

Compute the centered dataset $\mathcal{D}' = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}\}$.

First, note that $\mathbb{E}[x_1] = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_1^{(i)} = 2.5$ and $\mathbb{E}[x_2] = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_2^{(i)} = 2.5$. To center the data, we subtract the vector $\begin{bmatrix}2.5\\2.5\end{bmatrix}$ from each of the points in the original dataset.

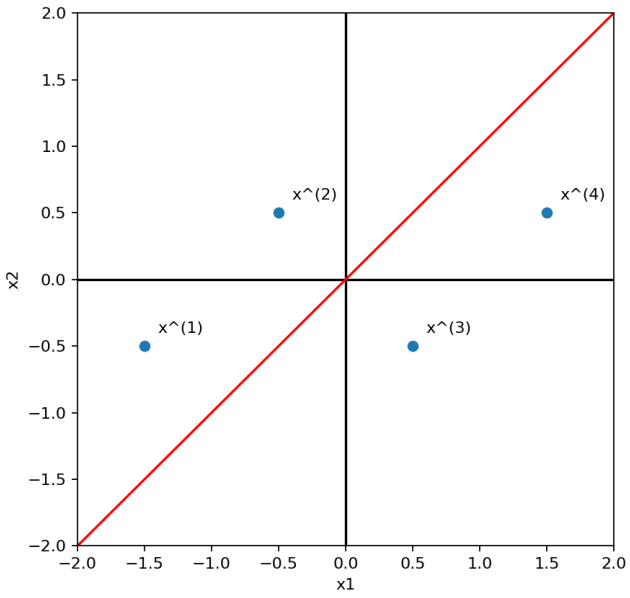
$$\mathbf{x}^{(1)} = \begin{bmatrix}-1.5\\-0.5\end{bmatrix} \qquad \mathbf{x}^{(2)} = \begin{bmatrix}-0.5\\0.5\end{bmatrix}$$
$$\mathbf{x}^{(3)} = \begin{bmatrix}0.5\\-0.5\end{bmatrix} \qquad \mathbf{x}^{(4)} = \begin{bmatrix}1.5\\0.5\end{bmatrix}$$

2. In order to easily compute the projected coordinates of data, we need to make the projected directions unit vectors. Suppose we want to project our data onto the vector $\mathbf{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Normalize $\mathbf{v}$ to be a unit vector.

   To make $\mathbf{v}$ into a unit vector, we divide $\mathbf{v}$ by its magnitude. The magnitude of a vector is given by the L2 norm.

   $$||\mathbf{v}|| = \sqrt{1^2 + 1^2} = \sqrt{2}$$
   $$\mathbf{v} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

3. The centered data should now look like the following:



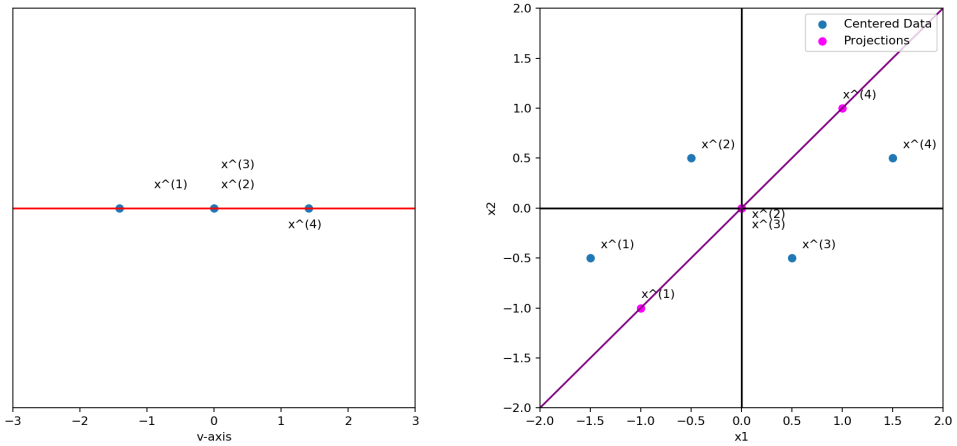
   Suppose we want to project the centered data onto $\mathbf{v}$, where $\mathbf{v}$ goes through the origin. Compute the magnitude of the projections, i.e. compute $z^{(i)} = \mathbf{v}^T\mathbf{x}^{(i)}, \forall 1 \leq i \leq N$.

$$z^{(1)} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \begin{bmatrix} -1.5 \\ -0.5 \end{bmatrix} = \frac{-3}{2\sqrt{2}} - \frac{1}{2\sqrt{2}} = -\frac{4}{2\sqrt{2}} = -\sqrt{2}$$
$$z^{(2)} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} = \frac{-1}{2\sqrt{2}} + \frac{1}{2\sqrt{2}} = 0$$
$$z^{(3)} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} = \frac{-1}{2\sqrt{2}} + \frac{1}{2\sqrt{2}} = 0$$
$$z^{(4)} = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}] \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix} = \frac{3}{2\sqrt{2}} + \frac{1}{2\sqrt{2}} = \frac{4}{2\sqrt{2}} = \sqrt{2}$$

4. Let $\mathbf{x}^{(i)'}$ be the projected point of $\mathbf{x}^{(i)}$. Note that $\mathbf{x}^{(i)'} = \mathbf{v}^T\mathbf{x}^{(i)}\mathbf{v} = z^{(i)}\mathbf{v}$. Compute the projected coordinates $\mathbf{x}^{(1)'}, \mathbf{x}^{(2)'}, \mathbf{x}^{(3)'}$, and $\mathbf{x}^{(4)'}$.

$$\mathbf{x}^{(1)'} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \qquad \mathbf{x}^{(2)'} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\mathbf{x}^{(3)'} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathbf{x}^{(4)'} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Below is a visualization of the projections:



5. One of the two goals of PCA is to find new directions to project our dataset onto such that it **minimizes the reconstruction error**, where the reconstruction error is defined as following:

$$\text{Reconstruction Error} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}^{(i)'} - \mathbf{x}^{(i)}\|_2^2$$

What is the reconstruction error in our case?

$$\begin{aligned}
\text{Recon. Error} &= \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{x}^{(i)'} - \mathbf{x}^{(i)}\|_2^2 \\
&= \frac{1}{N}(\| \begin{bmatrix}-1\\-1\end{bmatrix} - \begin{bmatrix}-1.5\\-0.5\end{bmatrix} \|_2^2 + \| \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}-0.5\\0.5\end{bmatrix} \|_2^2 + \| \begin{bmatrix}0\\0\end{bmatrix} - \begin{bmatrix}0.5\\-0.5\end{bmatrix} \|_2^2 + \| \begin{bmatrix}1\\1\end{bmatrix} - \begin{bmatrix}1.5\\0.5\end{bmatrix} \|_2^2) \\
&= \frac{1}{4}((\frac{1}{2})^2 + (-\frac{1}{2})^2 + (\frac{1}{2})^2 + (-\frac{1}{2})^2 + (-\frac{1}{2})^2 + (\frac{1}{2})^2 + (-\frac{1}{2})^2 + (\frac{1}{2})^2) \\
&= \frac{1}{4}(\frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \frac{1}{2}) \\
&= \frac{1}{2}
\end{aligned}$$

6. Another goal is to find new directions to project our dataset onto such that it **maximizes the variance of the projections**, where the variance of projections is defined as following:

$$\begin{aligned}
\text{variance of projection} &= \frac{1}{N}\sum_{i=1}^{N}(z^{(i)} - \mathbb{E}[z])^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}^T\mathbf{x}^{(i)} - \frac{1}{N}\sum_{j=1}^{N}\mathbf{v}^T\mathbf{x}^{(j)})^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}^T\mathbf{x}^{(i)} - \mathbf{v}^T(\frac{1}{N}\sum_{j=1}^{N}\mathbf{x}^{(j)}))^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}^T\mathbf{x}^{(i)} - \mathbf{v}^T\vec{0})^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}(\mathbf{v}^T\mathbf{x}^{(i)})^2
\end{aligned}$$

What is the variance of the projections?

$$\begin{aligned}\text{variance} &= \frac{1}{N}\sum_{i=1}^{N}(z^{(i)})^2 \\ &= \frac{1}{4}((-\sqrt{2})^2 + 0^2 + 0^2 + (\sqrt{2})^2) \\ &= \frac{1}{4}(2+0+0+2) \\ &= 1\end{aligned}$$

7. What methods can we use to find the principal components?

Eigenvalue decomposition and SVD

8. What is the first principal component of $X = [x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}]^T$ using the eigenvalue decomposition?

$$X = \begin{bmatrix} -1.5 & -0.5 \\ -0.5 & 0.5 \\ 0.5 & -0.5 \\ 1.5 & 0.5 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 5 & 1 \\ 1 & 1 \end{bmatrix}$$

Solve the eigenvalues and eigenvectors for $X^T X$.

$$X^T X v_i = \lambda_i v_i$$

$$\lambda_1 = 3 + \sqrt{5}, \quad v_1 = \begin{bmatrix} 2+\sqrt{5} \\ 1 \end{bmatrix}$$

$$\lambda_2 = 3 - \sqrt{5}, \quad v_2 = \begin{bmatrix} 2-\sqrt{5} \\ 1 \end{bmatrix}$$

So the first principal component is $v_1/||v_1||_2 = \begin{bmatrix} 0.973 \\ 0.230 \end{bmatrix}$.

3 Fun With Lagrange and PCA

Consider the following optimization where A is a symmetric matrix:

$$\hat{\mathbf{v}} = \underset{\mathbf{v}}{\text{argmax.}} \quad \mathbf{v}^T A \mathbf{v}$$
$$\text{s.t.} \quad \|\mathbf{v}\|_2^2 = 1$$

1. Formulate the Lagrangian, $\mathcal{L}(\mathbf{v}, \lambda)$.

$$\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^T A \mathbf{v} - \lambda(\|\mathbf{v}\|_2^2 - 1)$$

2. Write the gradient of the Lagrangian with respect to $\mathbf{v}$, $\nabla_{\mathbf{v}} \mathcal{L}(\mathbf{v}, \lambda)$.

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \mathbf{v}^T A \mathbf{v} - \lambda(\mathbf{v}^T \mathbf{v} - 1)$$

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \lambda) = 2A\mathbf{v} - 2\lambda\mathbf{v}$$

3. After setting $\nabla_{\mathbf{v}} \mathcal{L}(\mathbf{v}, \lambda)$ equal to zero, what can you say about the relationship between λ, $\mathbf{v}$, and the eigenvalues and eigenvectors of A?

$$2A\mathbf{v} - 2\lambda\mathbf{v} = 0$$
$$A\mathbf{v} = \lambda\mathbf{v}$$

$\mathbf{v}$ is an eigenvector of A and λ is the corresponding eigenvalue of A!

4 Recommender Systems

4.1 Matrix Factorization

1. When doing PCA, given a dataset X, we are able to perform SVD to find the eigenvectors and eigenvalues of the covariance matrix $\frac{1}{N}X^TX$. If dealing with a user/item matrix $R \in \mathbb{R}^{N\times M}$, can we also use SVD to find the matrix decomposition R? If yes, write out the formula for the decomposition; if no, explain why not.

   We cannot use SVD to find the decomposition of the user/rating matrix, because the R matrix has missing values. SVD cannot be performed on a dataset with missing values.

2. Suppose we have N users and M items, with a partially observed ratings matrix $R \in \mathbb{R}^{N\times M}$. For matrix factorization, our goal is to find user matrix $U \in \mathbb{R}^{N\times K}$ and item matrix $V \in \mathbb{R}^{M\times K}$ such that $\hat{R} = UV^T$ accurately reconstructs our observed ratings and predicts our unobserved ratings.

$$U = \begin{bmatrix} \mathbf{u}_1^\top \\ \mathbf{u}_2^\top \\ \vdots \\ \mathbf{u}_N^\top \end{bmatrix}$$

$$V = \begin{bmatrix} \mathbf{v}_1^\top \\ \mathbf{v}_2^\top \\ \vdots \\ \mathbf{v}_M^\top \end{bmatrix}$$

   Our reconstruction of the rating of item j by user i is given by

$$\hat{r}_{ij} = \mathbf{u}_i^T\mathbf{v}_j$$

   , where $\mathbf{u}_i \in \mathbb{R}^K$ is our learned user vector and $\mathbf{v}_j \in \mathbb{R}^K$ is our learned item vector.

   Let $\mathbb{I} = \{(i,j) : r_{ij} \text{ is observed}\}$. Describe the objective function $J(U,V)$, using a squared error loss.

$$J(U,V) = \sum_{(i,j)\in\mathbb{I}} J_{ij}(U,V) = \frac{1}{2}\sum_{(i,j)\in\mathbb{I}} (r_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2$$

3. We can learn our decomposition by using an optimization method like SGD. Describe below the steps applying SGD.

   - sample (i,j) from $\mathbb{I}$
   - step opposite gradient of $J_{ij}(U,V)$

4. What are the gradients $\nabla_{\mathbf{u}_i} J_{ij}(U,V)$ and $\nabla_{\mathbf{v}_j} J_{ij}(U,V)$ for our gradient update?

$$\nabla_{\mathbf{u}_i} J_{ij}(U,V) = -(r_{ij} - \mathbf{u}_i^T\mathbf{v}_j)\mathbf{v}_j$$
$$\nabla_{\mathbf{v}_j} J_{ij}(U,V) = -(r_{ij} - \mathbf{u}_i^T\mathbf{v}_j)\mathbf{u}_i$$

5 K-Means

In KMeans, we aim to minimize the objective function:

$$J(K) = \min_{\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_K, z_1,...,z_N} \sum_{i=1}^{N} ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_{z_i}||_2^2$$

Recall the KMeans algorithm:
Let there be a set of N observations $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ be the set of input examples that each have d features.

Initialize K cluster centers $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_K\}$ where $\boldsymbol{\mu}_r \in \mathbb{R}^d$.

Repeat until convergence:

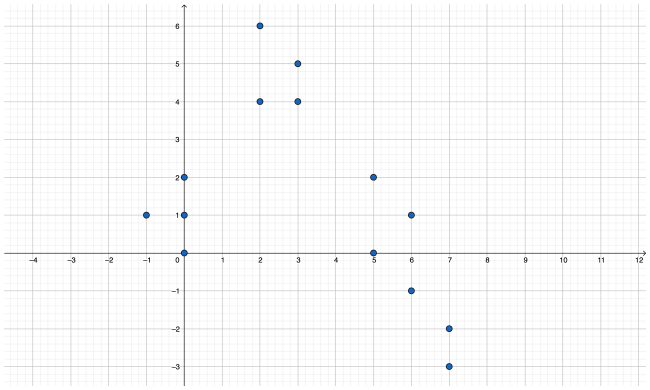
1. Assign each point $\mathbf{x}^{(i)}$ to a cluster $\boldsymbol{\mu}_{z_i}$ where $z_i = \text{argmin}_{1 \le r \le K} ||\mathbf{x}^{(i)} - \boldsymbol{\mu}_r||_2^2$
2. Recompute each $\boldsymbol{\mu}_r$ as the mean of points assigned to it from the previous step.

What is the benefit of using k-means algorithm when solving a partitioning problem?

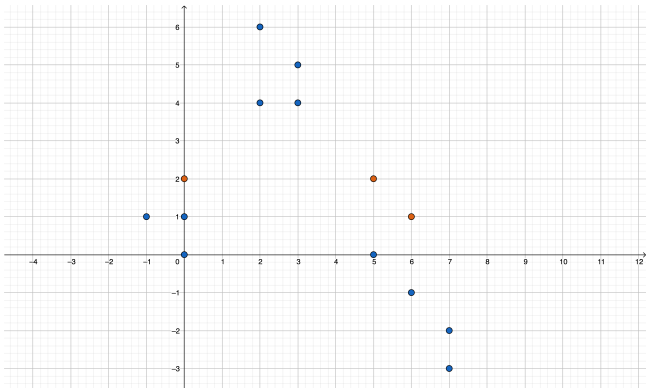
> Brute-forcing (exhaustively enumerating all partitions) is NP-hard. Using K-means can solve the problem in $O(lKN)$, where l is the number of iterations, K is the number of cluster centers, and N is the number of points.

5.1 Walking through an example

Lets walk through an example of KMeans with $k = 3$ using the following dataset for the first iteration:



Let the cluster centers be initialized to $\mathbf{c}^{(1)} = (0, 2)$, $\mathbf{c}^{(2)} = (5, 2)$, $\mathbf{c}^{(3)} = (6, 1)$ as depicted below in the orange:



Perform one iteration of the KMeans algorithm:

1. What are the cluster assignments?

$\mathcal{C}^{(1)} = \{(0,0), (-1,1), (0,1), (0,2), (2,4), (2,6)\}$
$\mathcal{C}^{(2)} = \{(3,4), (3,5), (5,2)\}$
$\mathcal{C}^{(3)} = \{(5,0), (6,1), (6,-1), (7,-2), (7,-3)\}$

2. What are the recomputed cluster centers?

$\mathbf{c}^{(1)} = (0.5, 2.33)$
$\mathbf{c}^{(2)} = (3.67, 3.67)$
$\mathbf{c}^{(3)} = (6.2, -1)$

6 Warm-up for Gaussian Mixture Models

6.1 MLE for Categorical/Gaussian Generative Model

Consider the following dataset and model:

$$\mathcal{D} = \left\{x^{(i)}, y^{(i)}\right\}_{i=1}^{N}$$
$$Y \sim Categorical(\pi_1, \pi_2, \pi_3)$$
$$X_{Y=k} \sim \mathcal{N}\left(\mu_k, \sigma_k^2\right)$$

Formulate the maximum likelihood optimization for parameters $\theta = \{\pi_k, \mu_k, \sigma_k\}_{k=1}^{3}$.

$$\begin{aligned}
\hat{\theta}_{MLE} &= \underset{\theta}{\text{argmax}}\, p(D \mid \theta) \\
&= \underset{\theta}{\text{argmax}} \prod_{i=1}^{N} p(x^{(i)}, y^{(i)} \mid \theta) \\
&= \underset{\theta}{\text{argmax}} \log \prod_{i=1}^{N} p(x^{(i)}, y^{(i)} \mid \theta) \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)} \mid \theta) \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log p(x^{(i)}, y^{(i)} \mid \theta) \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log p(y^{(i)} \mid \theta) p(x^{(i)} \mid y^{(i)}, \theta) \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \log \prod_{k=1}^{k} \pi_k^{y_k^{(i)}} f_{\mathcal{N}}(x^{(i)} \mid \mu_k, \sigma_k^2)^{y_k^{(i)}} \\
&= \underset{\theta}{\text{argmax}} \sum_{i=1}^{N} \sum_{k=1}^{k} y_k^{(i)} \log(\pi_k f_{\mathcal{N}}(x^{(i)} \mid \mu_k, \sigma_k^2))
\end{aligned}$$