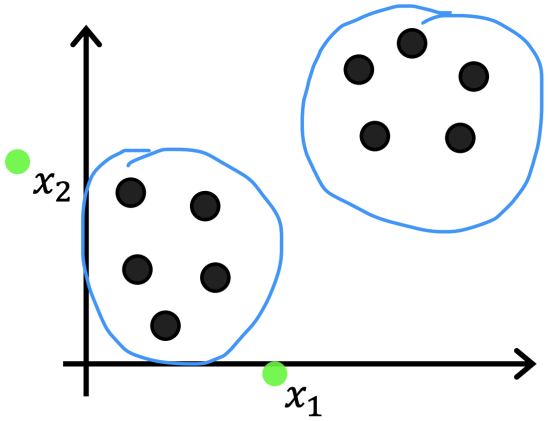
Clustering K-means

Clustering is a method used to identify and group similar data points together without prior knowledge of the group labels. The primary goal of clustering is to find structure or patterns in a dataset where the data points are grouped into clusters. Points in the same cluster are more similar to each other than to those in other clusters.

Unsupervised learning



Clustering

Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(m)}\}$

K-means clustering algorithm

Repeat{

Step 1: Assign each point to its closest centroid

Step 2: Recompute the centroids

}

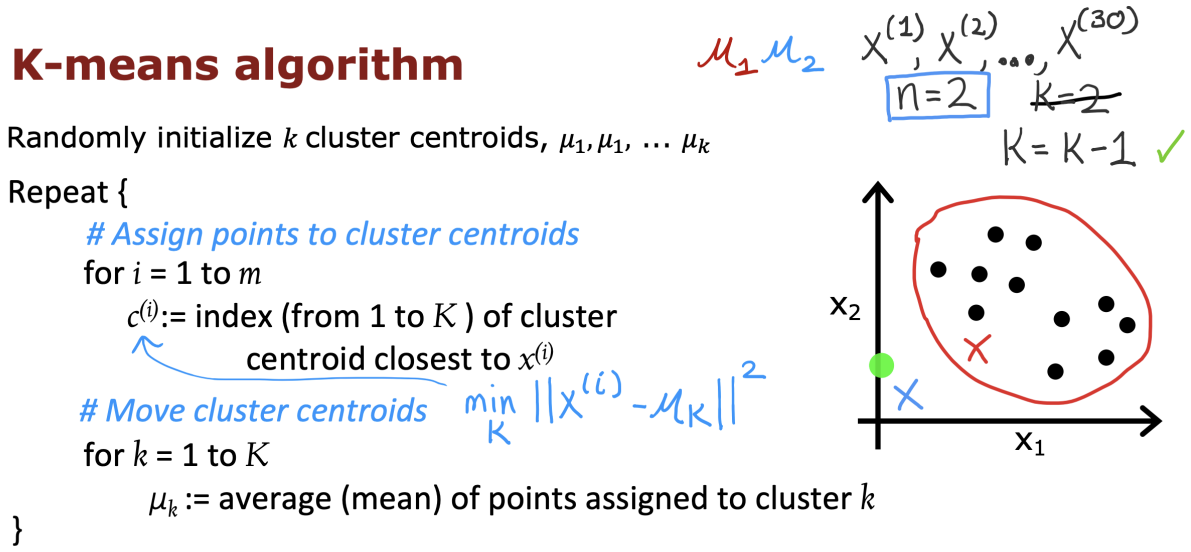
K-means Algorithm Pseudocode

1. **Initialise**: Choose `K` cluster centroids randomly (`μ1, μ2, ..., μK`). These centroids should have the same dimensionality as the feature vectors in your dataset.

2. **Repeat Until Convergence**:
    - **Assignment Step**: For each feature vector `xi` in the dataset, assign it to the nearest cluster centroid. The "nearest" is determined by the Euclidean distance (or squared Euclidean distance for computational efficiency). Mathematically, for each `i`, set:

      ```
      c^i = argmin_k ||xi - μk||^2
      ```

3. **Check for Convergence**: The algorithm has converged when the assignments no longer change. That is, when the points remain in the same clusters, and thus the cluster centroids no longer move significantly.



Additional Considerations

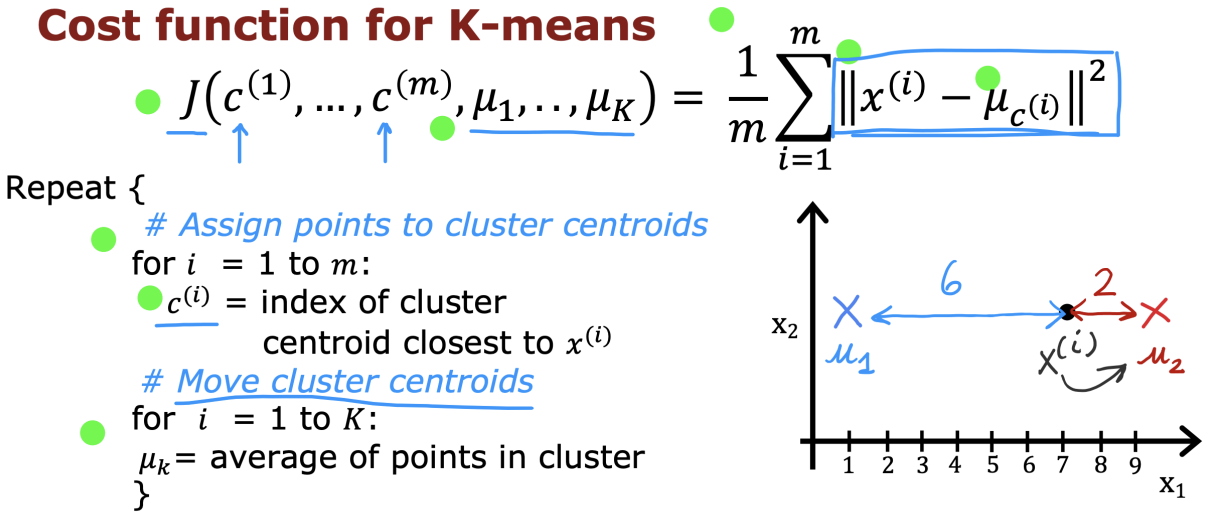
- **Choosing K**: The number of clusters `K` is a parameter that needs to be chosen beforehand. Methods like the Elbow Method can help determine a suitable `K` by analyzing the trade-off between the number of clusters and the within-cluster sum of squares (WCSS).
- **Initialization Sensitivity**: The initial choice of centroids can significantly affect the final clusters. Techniques like K-means++ can be used for smarter initialization to improve the chances of finding a good solution.

- **Convergence Criterion**: Besides checking for changes in assignments, convergence can also be determined based on changes in the positions of the centroids or improvements in a cost function (e.g., minimizing the within-cluster sum of squares).
- **Handling Empty Clusters**: If a cluster ends up with no points assigned to it during the algorithm's execution, one common approach is to remove that cluster or reinitialize its centroid randomly.

Implementation Tips

- **Vectorization**: When implementing K-means, leveraging vectorized operations in languages like Python (using libraries such as NumPy) can significantly speed up the computation, especially for large datasets.
- **Iterative Refinement**: K-means is an iterative refinement algorithm. Visualizing the clustering process over iterations can provide insights into how the algorithm converges and can help debug or understand the clustering dynamics.

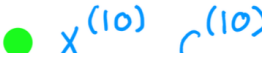
This pseudocode and the additional considerations provide a roadmap for implementing the K-means clustering algorithm, from initialization through to convergence, including practical tips for ensuring a robust implementation.



$c^{(i)}$ = index of cluster $(1, 2, \ldots, K)$ to which example $x^{(i)}$ is currently assigned

μ_k = cluster centroid k

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned



💡 The goal of K-means is to minimize this cost function, which represents the average squared distance between each training example and the centroid of the cluster to which it has been assigned.

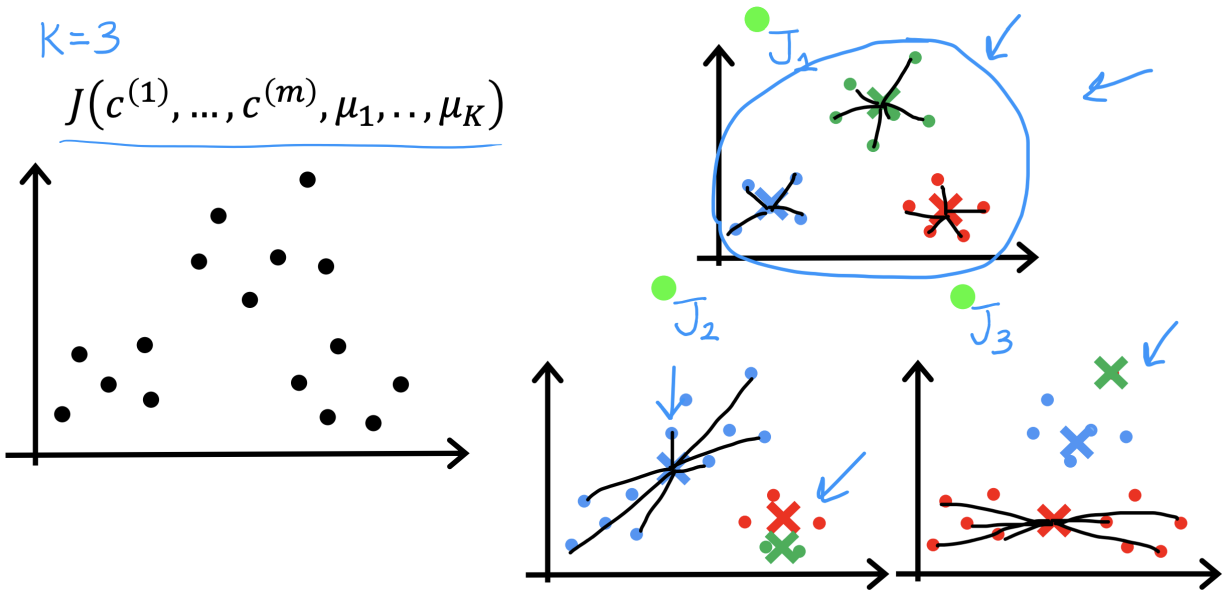
Initialising K-means

Random initialisation

For initialising cluster centroids (`μ1, μ2, ..., μK`) in K-means involves:

- Randomly selecting `K` distinct training examples from the dataset. (These selected points serve as the initial guesses for the centroids of the clusters.)
- Set (`μ1, μ2, ..., μK`) equal to these `K` training examples.

The K-means algorithm aims to minimise the cost function. However, this objective function can have multiple local minima—configurations where any small change increases the cost, but which are not the overall best (global minimum) configuration. Depending on the initial centroids, the algorithm might converge to one of these local minima rather than the global minimum.



Random initialization

better than running K-means once

For $i = 1$ to 100 {

50-1000

pick K random examples
Set as cluster centroids

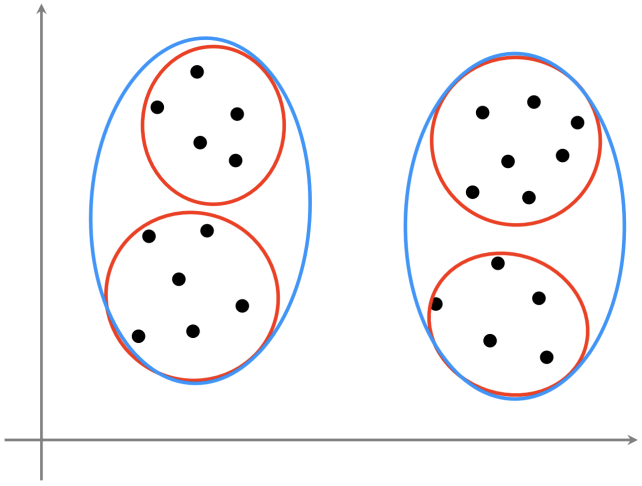
Randomly initialize K-means.
Run K-means. Get $c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_1, \ldots, \mu_k$
Compute $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_1, \ldots, \mu_k)$

distortion
(cost function)

}

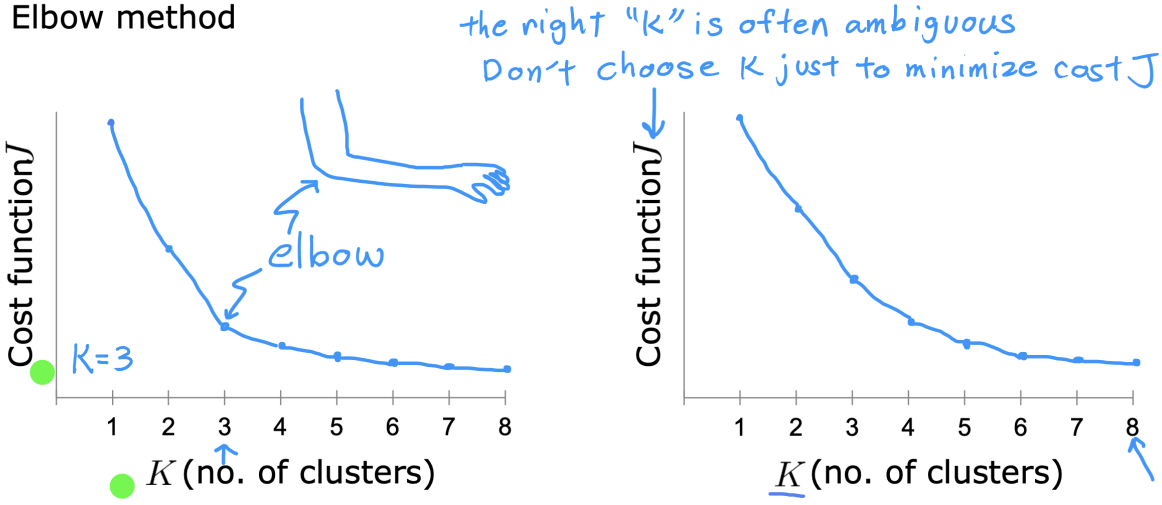
Pick set of clusters that gave lowest cost J

What is the right value of K?



Choosing the value of K

Elbow method

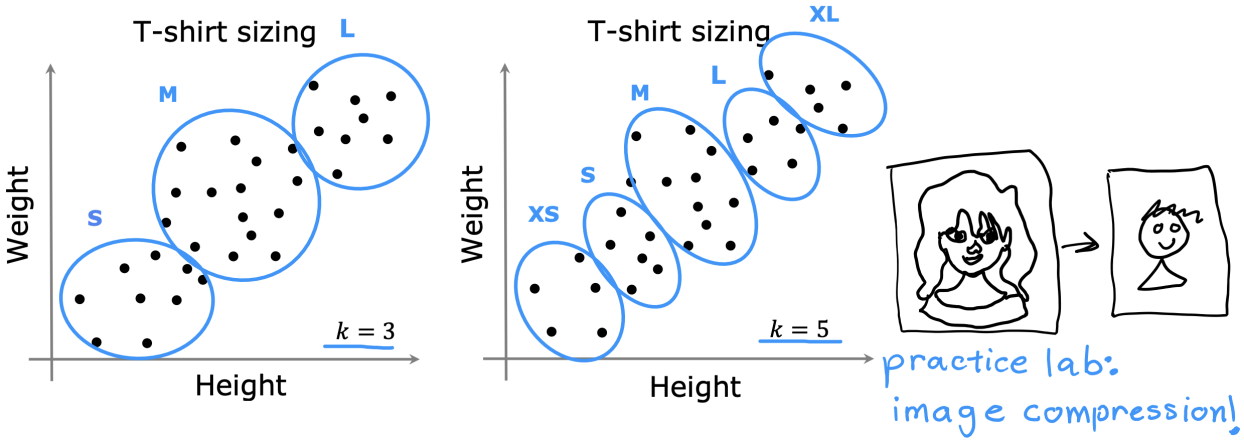


▼ Why methods like elbow should not be used?

Choosing the optimal number of clusters, (K), in K-means clustering using methods like the Elbow Method can be subjective and not always definitive. These methods rely on identifying a point where improvements in variance explained (or a reduction in the cost function) by adding more clusters diminish significantly. However, in many real-world datasets, the "elbow" point may not be clear or pronounced, leading to ambiguity in selecting (K).

Choosing the value of K

Often, you want to get clusters for some later (downstream) purpose.
Evaluate K-means based on how well it performs on that later purpose.



Considerations for Choosing (K)

- **Domain Knowledge**: Sometimes, the choice of \(K\) can be guided by domain knowledge or the specific requirements of the application.
- **Purpose of Clustering**: Consider what the clusters are being used for. If they are used for downstream tasks, such as customer segmentation in marketing, the choice of \(K\) might be influenced by business considerations.
- **Trade-offs**: As with the t-shirt sizing example, there are often trade-offs between having more clusters (potentially more tailored solutions) and the complexity or cost associated with them. Evaluating these trade-offs in the context of your specific application is crucial.