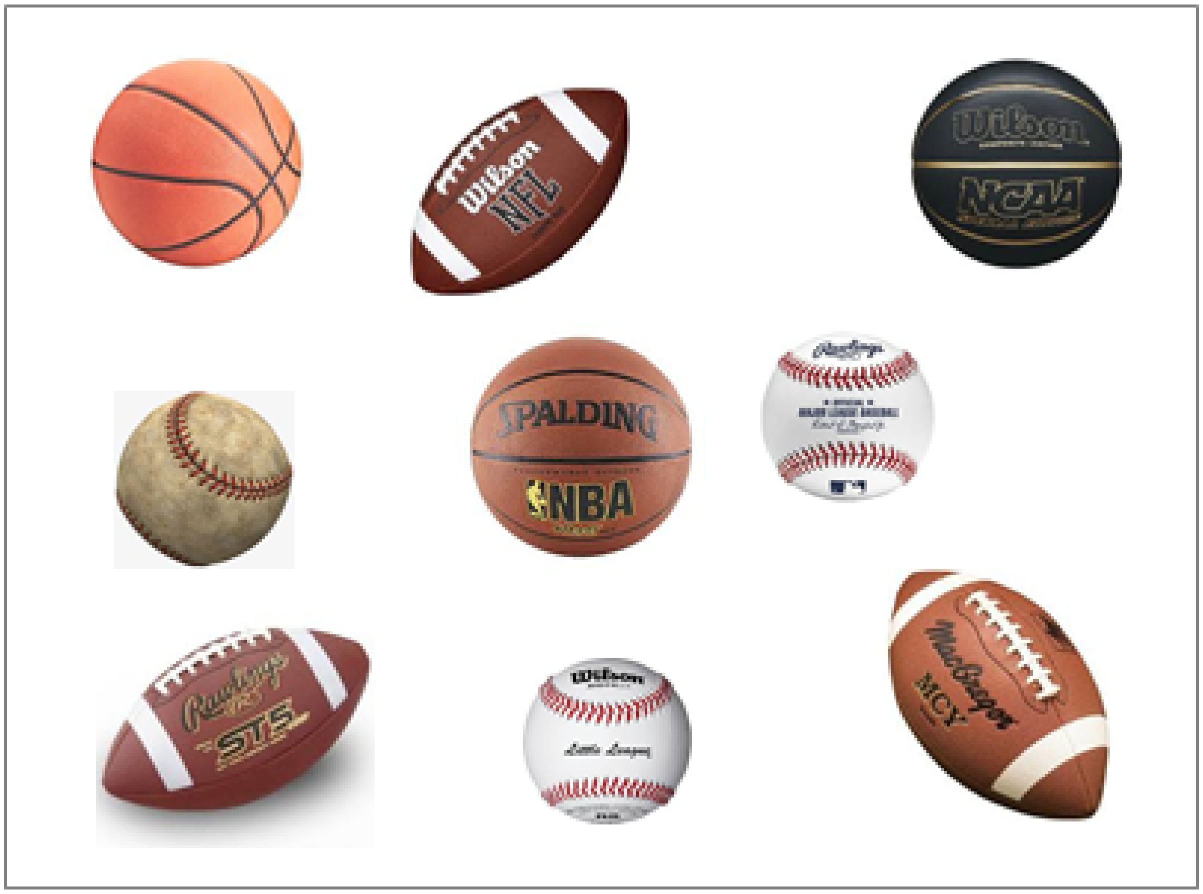
Lecture 14 Part 2 - Introduction to Clustering with K-Means Clustering

Clustering Clustering is a type of *unsupervised* Machine Learning technique. Clustering algorithms seek to learn, from the properties of the data, an optimal division or discrete labeling of groups of points.

Suppose you collect pictures of the following objects:

```
In [2]: from IPython.display import Image
        Image('figures/ClusteringExample.png', width=600)
```

Out[2]:



The goal in clustering is to partition the data into groups.

- How many groups would you partition this data into?

We will first need to collect/extract features that characterize each object in each image.

Then we can represent each image as a collection of features (in our polynomial regression problem, we use polynomial features, but we can use *any* basis function to characterize the data).

In the feature space, we want to find partitions of the data. (In some cases we want to partition the original space - the input space of the images.)

- We have seen before that we can use **Gaussian Mixture Models (GMMs)** to characterize the data and then consider each Gaussian as distribution representation of each group.

Another popular clustering algorithm that takes a **non-parametric** form is called the **K-Means Algorithm**.

Types of Clustering

There are many different types of clustering:

There are 5 types of clustering:

1. Centroid clustering
    - Example: K-Means
2. Distribution clustering
    - Example: Mixture Models
3. Density clustering
    - Example: DBSCAN
4. Hierarchical clustering
    - Example: Dendrogram
5. Graph clustering
    - Example: Spectral Algorithm

They differ from one another in how they assess *similarity* between points to be assigned in the same cluster.

K-Means Clustering Algorithm

K-Means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**.

- K-Means Clustering is a centroid-based clustering.

- It tries to make the *inter-cluster* data points as similar as possible while also keeping the clusters as different (far) as possible.

- It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all data points that belong to that cluster) is at the minimum.

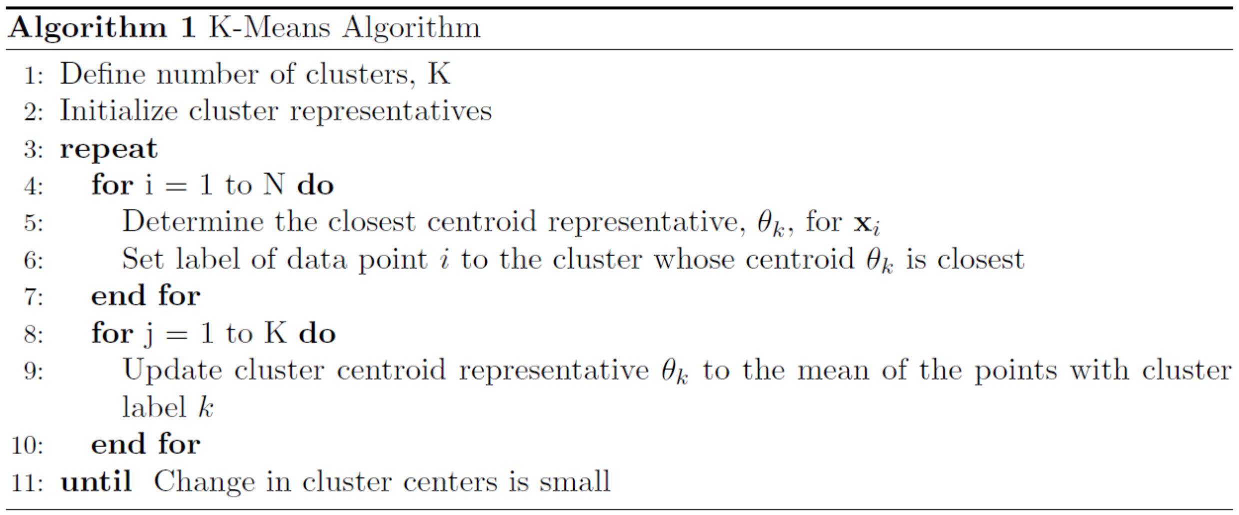
- The less variation we have within clusters, the more homogenous (similar) the data points are within the same cluster.

The **pseudo-code** is summarized as follows:

1. Specify number of clusters K
2. Initialize centroids by first shuffling the dataset amd then randomly selecting K data points for the centroids without replacement. (There are different ways of initializing the centroids)
3. Keep iterating until there is no change to the centroids, i.e., assignment of data points to clusters isn't changing.
    - Compute the sum of the squared distance between data points and all centroids
    - Assign each data pint to the closest cluster (centroid)
    - Compute the centroids for the clusters by taking the average of all data points that belong to each cluster

```
In [4]: Image('figures/KMeans.png',width=800)
```

Out[4]:



Algorithm 1 K-Means Algorithm

```
1:  Define number of clusters, K
2:  Initialize cluster representatives
3:  repeat
4:      for i = 1 to N do
5:          Determine the closest centroid representative, $\theta_k$, for $\mathbf{x}_i$
6:          Set label of data point $i$ to the cluster whose centroid $\theta_k$ is closest
7:      end for
8:      for j = 1 to K do
9:          Update cluster centroid representative $\theta_k$ to the mean of the points with cluster
            label $k$
10:     end for
11: until Change in cluster centers is small
```

The K-Means algorithm uses **Expectation-Maximization (EM) as the optimization approach**:

- The E-Step assigns (hard) membership
- The M-step updates the representatives (centroids) for each cluster

As we learned before, optimization with EM is called **Alternating Optimization** and therefore the final solution will be **dependent** on the initialization (of cluster centroids). Therefore the final solution may not be the *optimal* solution.

- The objective function for the K-Means algorithm is:

$$
\begin{aligned}
J(\Theta, U) &= \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} d^2\left(x_i, \theta_k\right) \\
&= \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} \|x_i - \theta_k\|_2^2 \\
\text{such that } u_{ik} \in \{0,1\} &\text{ and } \sum_{k=1}^{K} u_{ik} = 1
\end{aligned}
$$

where u_{ij} are cluster assignments, θ_j is the j^{th} cluster representative and $d\left(x_i, \theta_k\right)$ is the distance between data point x_i and cluster centroid θ_k.

- In K-Means, we want to optimize:

$$\arg_{\Theta, U} \min J(\Theta, U)$$

- If the **Euclidean distance** metric is used, does the K-means algorithm make any assumptions on cluster shape?
- Given a data set with an unknown number of clusters, can you come up with a strategy for determining the *right* number of clusters?
- Can we use other distance metrics in objective function $J(\Theta, U)$?
