Clustering

1 Unsupervised Learning: Introduction

Clustering will be our first unsupervised algorithm.

In clustering, we will no longer have labels for the data. We will make the algorithm find some structure in the data, clusters.

Applications:

- Market segmentation.
- Social network analysis.
- Organize computing clusters.
- Astronomical data analysis.

2 K-Means Algorithm

Iterative algorithm for grouping data in clusters:

Input = number of clusters and unlabeled training set.

1. Randomly initialize cluster centroids.
2. Assign each data point to its closest centroid.
3. Move each centroid to the mean of the points assigned to them.
4. Iterate 2 and 3 until no major change in step 3 happens.

A more formal description is given in algorithm 1:

Algorithm 1 Kmeans algorithm

function KMEANS(K, x) ▷ K = number of clusters, x = unlabeled dataset
  Randomly initialize K cluster centroids $\mu_1, \mu_2, ..., \mu_K \in \mathbb{R}$.
  while not convergence **do**
    for $i = 1$ to m **do** ▷ m is the length of the dataset
      $c^{(i)} \leftarrow$ index of cluster centroid closest to $x^{(i)}$
    end for
    for $k = 1$ to K **do**
      $\mu_k \leftarrow$ average (mean) of points assigned to cluster k
    end for
  end while
end function

To find out which cluster is the closest one, we can use any distance (such as the euclidian one).

Convergence is declared once there is no change between two consecutive runs.

If a cluster ends with no points assigned to it, simply eliminate it or reinitialize it randomly if the amount of clusters can't be diminished at all.

3 Optimization Objective

The algorithms we've seen all have a cost function to optimize. Turns out, Kmeans also has a cost function. Let's remember some of the kmean notation first:

- $c^{(i)}$ = index of cluster to which example$x^{(i)}$is currently assigned

- μ_k = cluster centroid $k(\mu_k \in \mathbb{R}^n)$
- $\mu_c(i)$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

Given that, we can now see that kmeans is minimizing eq. (1)

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \left\| x^{(i)} - \mu_c(i) \right\|^2 \tag{1}$$

eq. (1) is sometimes called the distortion formula.

4 Random Initialization

We need to make sure that the kmeans algorithm evades local optimal. Some things that help with that are:

- $K < m$: number of clusters < number of data points.
- Randomly choosing K training examples instead of choosing freely random points for cluster initialization.

Even using that, it might end up in a local optima. We can initialize kmeans several times to solve that as shown in algorithm 2.

Algorithm 2 Kmeans solving local optima problem

```
function KMEANS(K, x, repetitions)      ▷ K = number of clusters, x = unlabeled dataset
    for i = 1 to repetitions do
        Run normal K-means.
    end for
    return the instance that gets the lowest cost function.
end function
```

5 Choosing the number of clusters

The most common thing to do is to explore that data in order to know the optimum amount of clusters.

A method sometimes used is called Elbow number (fig. 1), in which you pick the point in which the cost function no longer decreases very fast. This can only be applied in certain cases in which that value is very clear, which might not be always as shown in the right graph of fig. 1.

Sometimes Kmeans is used to get clusters to use for some later purpose. We might exploit that to select the amount of clusters so that it performs well for that purpose.

For example, if we are clustering t-shirts sizes, to find out the amount of t-shirt sizes we have to produce, as seen in fig. 2. We can think about it from the t-shirt business point of view, and we might conclude that having less t-shirt sizes might be cheaper, so we would prefer an small amount of clusters.

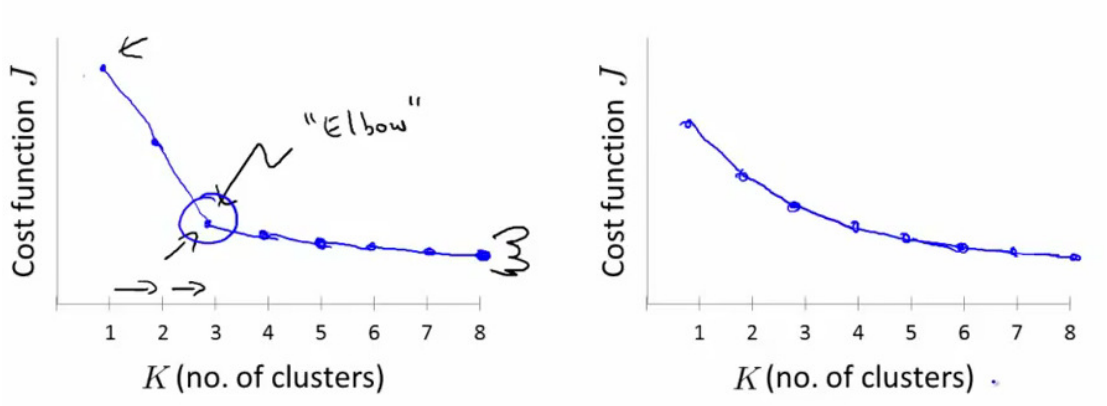

Figure 1: Graphs indicating when to use the elbow method to select the optimal amount of clusters.

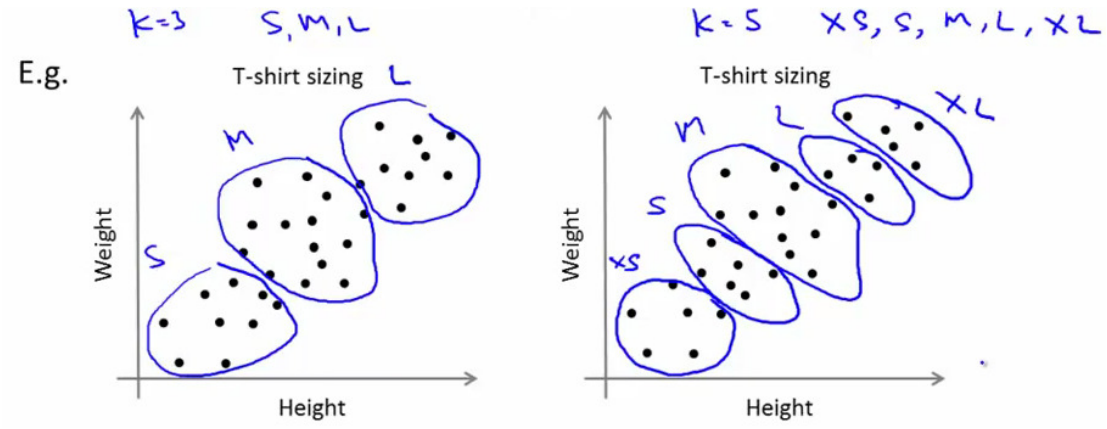

Figure 2: Graphs showing different cluster amounts for the t-shirt industry.