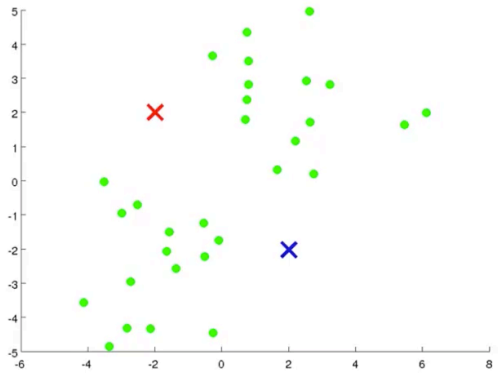
Clustering (Unsupervised Learning)

In unsupervised learning, we give the data to the model and trying to look for some structures in our data. If we group the data, we have clustering algorithm.

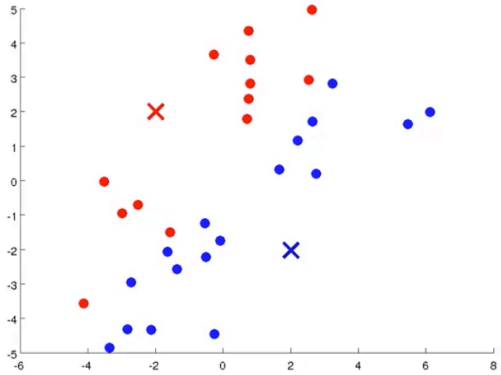
K-mean algorithm

It's the mostly wide used algorithm in unsupervised machine learning. Suppose we have a dataset and we would like to cluster them into two groups. Following is the procedure we follow for k-mean algorithm:

- Randomly allocate two points as the cluster centroids (if you are grouping into k clusters, you need k random points)



- Cluster assignment step: goes through each dataset (green dots) and assign them to one of the two centroid points based on distance.



- Moving centroid step: Take each centroid and move to the average of the correspondingly assigned data-points. In other word, take the mean of red cluster and move centroid to the average value and do the same for the blue cluster. We repeat step 2 and 3 till it converges.

K-mean for non-separated clusters

Sometimes K-means is applied to datasets where there aren't well defined clusters. For example, t-shirt sizes:

This is an example of market segmentation. We can separate smaller height and weight to t-shirt size small and medium height and weight to medium and larger values to large size.

Optimization objective

In k-mean, we try to optimize the cost function as well. Let's use following syntax:

Clustering (Unsupervised Learning)

c^i: the index of clusters {1,2, ..., K} to which x^i is currently assigned
μ_k: cluster centroid
$\mu_c{}^i$: is the cluster centroid of the cluster to which example x^i has been assigned to
The optimization objective is to minimize the (distortion) cost function:

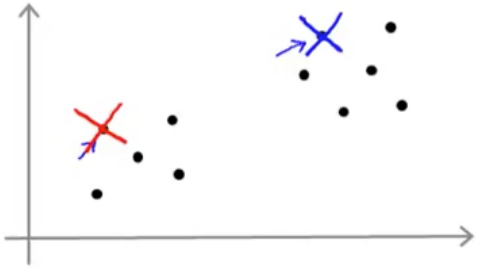
$$J(c^1, c^2, \dots, c^m, \mu_1, \mu_2, \dots, \mu_K) = \frac{1}{m}\sum_{i=1} \left\| x^{(i)} - \mu_c(i) \right\|^2$$
$$min_{c^1, c^2, \dots, c^m, \mu_1, \mu_2, \dots, \mu_K} J(c^1, c^2, \dots, c^m, \mu_1, \mu_2, \dots, \mu_K)$$

So, the algorithm will be as follow:

```
Repeat {
        For i=1 to m
                c^i:=index (from 1 to K) of cluster centroid closest to x^(i)
        for k=1=1 to K
                μ_k:= average (mean) of points assigned to cluster k
}
```

The first part is the clustering assignment that min cost. The second part is move centroid step that chooses J with respect to the location (μ). *When you check J vs number of iterations, remember that it's not possible the line increases sometimes and decreases sometimes*.

Random initialization

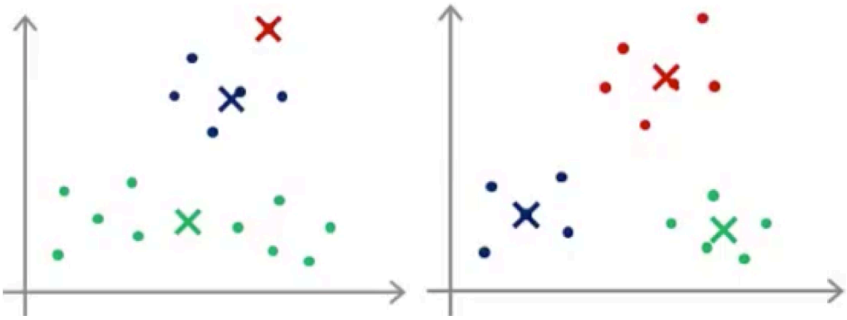


There are different ways to initialize k cluster centroids, μs. Note that number of clustering must be smaller than training examples (K<m).
The best way to do the initialization is:

- Randomly pick K training example
- Set μs equal to these K samples

Depending on how the random training set is chosen, the problem may converge to different values and end up at local optima, not the global one.

Clustering (Unsupervised Learning)

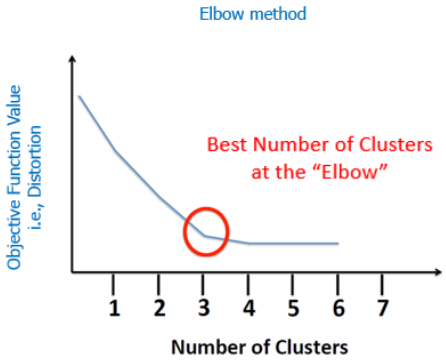


What's the solution then? Try multiple random initialization (around 50-1000 times) and make sure to get the best solution. Finally Pick the clustering which gave the lowest distortion. If K>10 then multiple random initializations are less likely to be necessary. First solution is probably good enough.

Choosing the number of clusters

There are several methods, currently people are using.

- Elbow method: In this method, we vary the number of cluster and calculate the cost function. The elbow at the plot, where distortion happens rapidly and then slowdown is the best number of clustering to choose. This method is not used quiet often, since the plot most of the time is smooth and cannot see the elbow.



- Eyeballing the value: like the t-shirt size example we mentioned above, using human insight