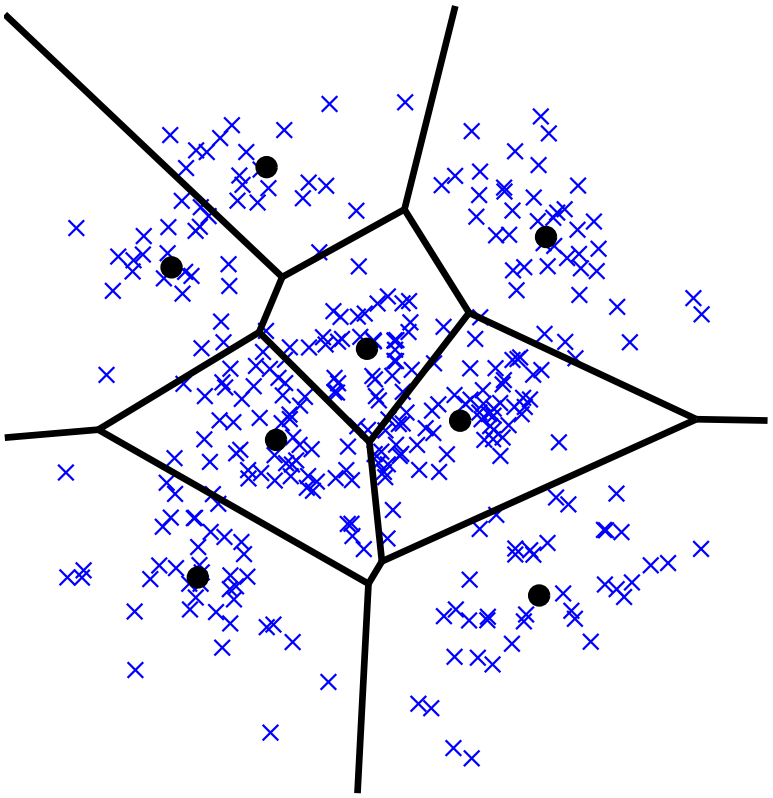
PR414 / PR813 Lecture 5

Clustering



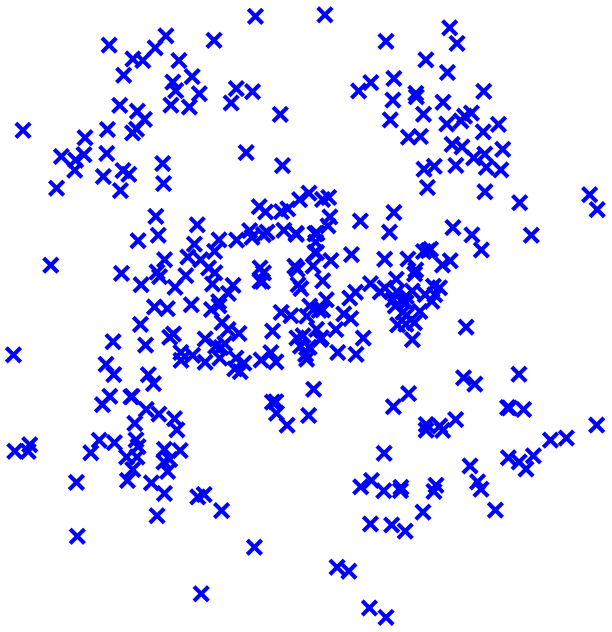
Ludwig Schwardt, Johan du Preez

University of Stellenbosch

17 March 2003

1. Clustering / Vector Quantisation (VQ)

- The idea is to represent a data set by a set of K vectors, called a *vector codebook*
- Code vectors (also known as *cluster centroids*) should ideally lie in areas of feature space with high data density $\Longrightarrow$ VQ is form of density estimation
- Codebook therefore captures *internal structure* of data set
- Example data set below shows evidence of "clumps" of feature vectors $\Longrightarrow$ the idea is to divide the set into groups or *clusters* of points, with a representative code vector at the center of each cluster
- Codebook size K is assumed to be known $\Longrightarrow$ choice of K is *black art* (e.g. how many clusters in data set below?)



2. Metrics, centroids and clusters

- **VQ intimately tied to *distance measure* or *metric*** $d(\boldsymbol{x}, \boldsymbol{y})$ **defined between feature vectors**
- **Standard choice is *Euclidean* metric given by** $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| \triangleq \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T(\boldsymbol{x} - \boldsymbol{y})}$**, but other measures might be more appropriate for specific feature spaces**
- **Metric determines calculation of *centroid*** $\boldsymbol{c}(\boldsymbol{X})$ **of set of vectors** $\boldsymbol{X}$**, defined as that point which minimises sum of distances to members of set (i.e. "center" of set)**
- **For Euclidean metric, centroid is simply *mean* of set** $\boldsymbol{c}(\boldsymbol{X}) = 1/N \sum_{n=1}^{N} \boldsymbol{x}_n$**, where** $\boldsymbol{X} = \{\boldsymbol{x}_n | n = 1 \ldots N\}$
- **Three important questions for VQ:**
  1. ***Codebook design ("training"):*** **find best codebook for given data set**
  2. ***Quantisation:*** **find closest code vector to given vector**
  3. ***Codebook distance ("testing"):*** **find closest codebook to given vector, i.e. one with closest code vector**
- **We will examine three codebook design algorithms:**
  1. K**-means**
  2. **Non-uniform binary split**
  3. **Binary split followed by** K**-means**

3. The K-means algorithm

- K-means is probably simplest example of an ***Expectation-Maximisation (EM)*** optimisation algorithm
- EM algorithms very useful for optimising problems containing ***hidden or incomplete*** information
- For example, in clustering problem, if each data point belonged to a known set or cluster, the optimum code vectors would simply be the centroids of these clusters
- Unfortunately we do not know which data points group together $\Longrightarrow$ this is ***hidden*** information
- EM approach suggests ***estimating*** hidden information from best solution so far (the ***E-step***), and then using this to improve solution (the ***M-step***)
- EM alternates between E-step and M-step until solution converges $\Longrightarrow$ guaranteed to converge to ***local optimum***
- Choice of initial solution crucial to get to ***good*** optimum
- K-means algorithm starts with initial estimate of centroids, and then alternately determines ***clusters from centroids*** in E-step, and ***centroids from clusters*** in M-step, until convergence

4. Calculating K-means

1. **Start with initial set of K centroids $[\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K]$, or choose K vectors at random from data set $[\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$**
2. ***E-step:*** **Split data set into K clusters $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_K\}$, where data point $\boldsymbol{x}_n$ belongs to cluster $\boldsymbol{X}_i$ if $\boldsymbol{y}_i$ is closest centroid**

$$\boldsymbol{X}_i = \left\{ \boldsymbol{x}_n \;\middle|\; d(\boldsymbol{x}_n, \boldsymbol{y}_i) \leq d(\boldsymbol{x}_n, \boldsymbol{y}_j),\;\; j = 1 \ldots K \right\}$$

3. ***M-step:*** **Update the centroids based on new clusters**

$$\boldsymbol{y}_i = \boldsymbol{c}(\boldsymbol{X}_i),\;\; i = 1 \ldots K$$

4. **Determine *total distortion* as sum of distances to closest centroids**

$$D = \sum_{n=1}^{N} d(\boldsymbol{x}_n, \boldsymbol{y}_{i(n)}), \;\; \textbf{where} \;\; i(n) = k \;\; \textbf{if} \;\; \boldsymbol{x}_n \in \boldsymbol{X}_k$$

5. **Repeat steps 2 to 4 until total distortion doesn't change appreciably or limit on number of iterations is reached**

- **Useful test to see if distortion stabilised is to check relative drop in distortion from last iteration and stop if**

$$\Delta D = \frac{D_{\text{prev}} - D_{\text{curr}}}{D_{\text{prev}}} < 10^{-4}$$

- **Algorithm should converge before about 100 iterations**

5. Non-Uniform Binary Split

- K-means algorithm produces good quality codebooks, but can be slow $\Longrightarrow$ finding closest centroid (*quantisation*) also involves a linear search through all K code vectors
- *Hierarchical clustering* recursively splits data set into clusters and subclusters $\Longrightarrow$ suboptimal but much faster
- Once boundary is drawn between two clusters, it remains in place $\Longrightarrow$ example of "hard" decision approach compared to "soft" style of K-means
- *Binary split* algorithm splits data set into two clusters, and continues to split clusters in half until K clusters left
- This allows binary search of order $\log K$ to find closest centroid, which is large saving if codebook is huge
- Training of codebook also faster due to rigid boundaries $\Longrightarrow$ price paid is lower quality codebook
- *Non-uniform* version splits cluster with biggest distortion on each round, as opposed to *uniform* version that splits each cluster in turn, regardless of distortion $\Longrightarrow$ improves quality of codebook

6. Calculating Binary Split

1. **Start with all data points x_n in one cluster $\boldsymbol{X}_1$ with associated centroid $\boldsymbol{y}_1 = \boldsymbol{c}(\boldsymbol{X}_1)$, and let cluster counter $k = 1$**
2. **Repeat steps 3 to 6 $(K-1)$ times to obtain K centroids**
3. **Choose cluster $\boldsymbol{X}_i$ with largest *distortion*, as measured by average distance of points in cluster to centroid**

   $$D_j = \frac{1}{N_j} \sum_{n=1}^{N_j} d(\boldsymbol{x}_n^{(j)}, \boldsymbol{y}_j), \quad \text{with } \boldsymbol{X}_j = \left\{ \boldsymbol{x}_n^{(j)} \;\middle|\; n = 1 \ldots N_j \right\},$$

   and $D_i \geq D_j$ for $j = 1 \ldots k$
4. **Split selected cluster $\boldsymbol{X}_i$ into two subclusters $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$, by choosing one of following methods:**
   (a) **(Any metric) Do K-means on set $\boldsymbol{X}_i$ with $K = 2$**
   (b) **(Euclidean metric only and less accurate but very fast) Determine principal eigenvector $\boldsymbol{v}_i$ of set $\boldsymbol{X}_i$, then let subcluster $\boldsymbol{X}_a$ be those points in $\boldsymbol{X}_i$ closest to $\boldsymbol{y}_i + \boldsymbol{v}_i$, and $\boldsymbol{X}_b$ those closest to $\boldsymbol{y}_i - \boldsymbol{v}_i$**
5. **Replace centroid $\boldsymbol{y}_i$ and add new centroid by letting**

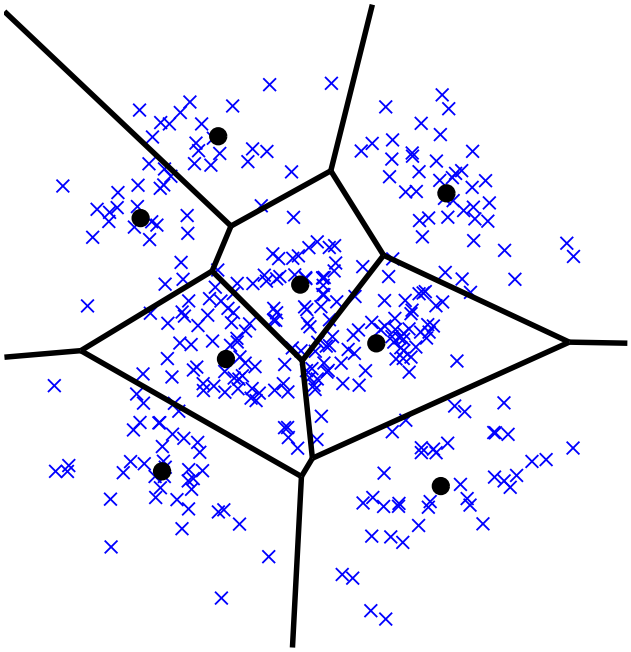
   $$\boldsymbol{y}_i = \boldsymbol{c}(\boldsymbol{X}_a) \quad \text{and} \quad \boldsymbol{y}_{k+1} = \boldsymbol{c}(\boldsymbol{X}_b)$$

6. **Increment cluster counter $k \leftarrow k + 1$**

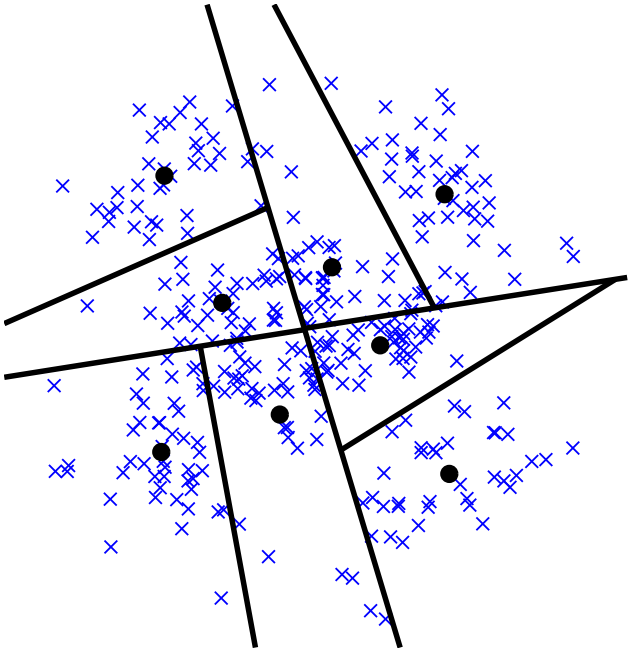
7. Comparing K-means and Binary Split

- Use of Euclidean metric leads to piecewise linear cluster boundaries
- K-means clusters have optimum boundaries which perpendicularly bisect lines connecting neighbouring centroids $\Longrightarrow$ each cluster therefore contains all points closest to its centroid
- Binary split clusters are clearly suboptimal $\Longrightarrow$ some points actually closer to neighbouring centroid than assigned one

K-means

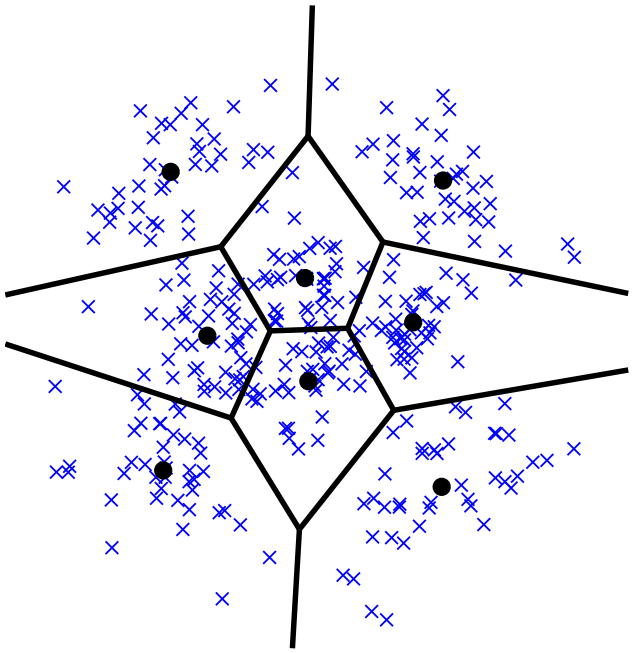


Binary split



8. Improvements on standard K-means

- K-means is sensitive to choice of initial centroids
- Instead of initialising K-means with random data points, use centroids found by binary split $\Longrightarrow$ this serves as coarse clustering to be refined by K-means
- Combined algorithm is faster than standard K-means and also produces higher quality codebooks, as seen below



- It is possible that a cluster $\boldsymbol{X}_i$ can become empty during E-step of K-means, especially if ratio K/N is large
- One solution is to reassign centroid $\boldsymbol{y}_i$ to random data point, thereby shaking up the clustering process
- Another option is to discard $\boldsymbol{y}_i$ and continue with the remaining $K-1$ clusters instead $\Longrightarrow$ this compensates for choosing K too high and is very useful in practice