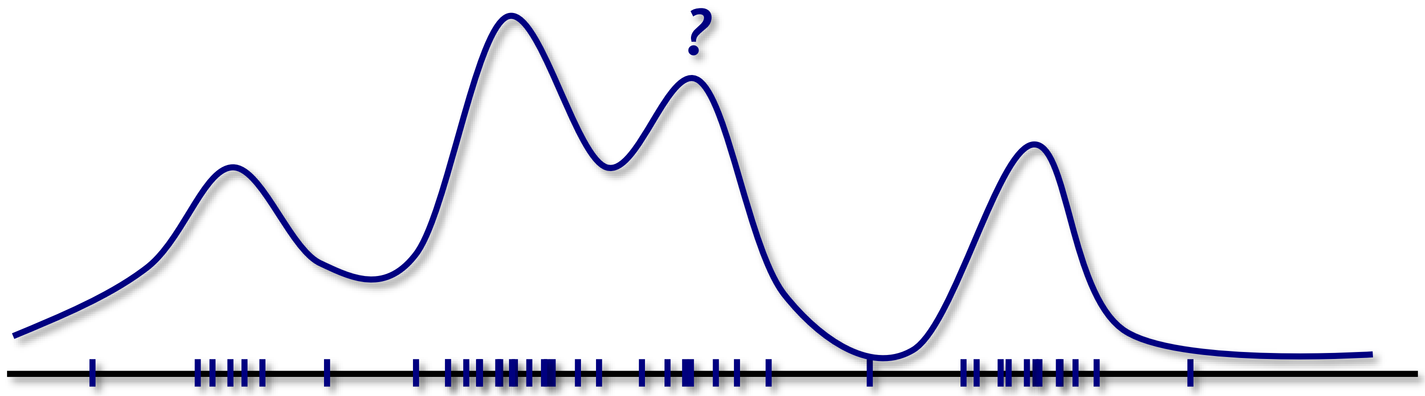
Density estimation

In density estimation problems, we are given a random sample $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ from an unknown density $f(\mathbf{x})$

Our objective is to estimate $f(\mathbf{x})$



Applications

Classification

- If we estimate the density for each class, we can simply plug this in to the formula for the Bayes' classifier
- Density estimation (for each feature) is the key component in Naïve Bayes

Clustering

- Clusters can be defined by the density: given a point $\mathbf{x}$, climb the density until you reach a local maximum

Anomaly detection

- Given a density estimate $\widehat{f}(\mathbf{x})$, we can use the test

$$\widehat{f}(\mathbf{x}) \lessgtr \gamma$$

to detect anomalies in future observations

Kernel density estimation

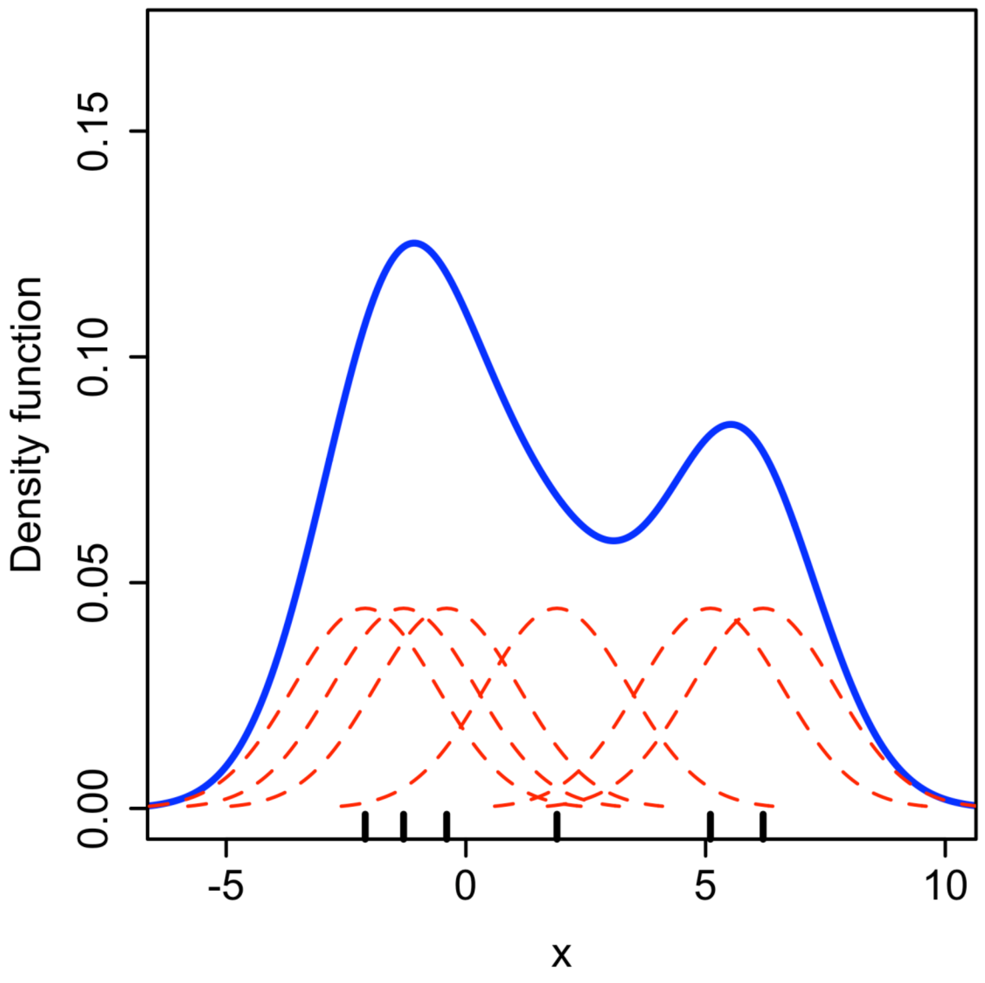
A ***kernel density estimate*** has the form

$$\widehat{f}(\mathbf{x}) := \frac{1}{n} \sum_{i=1}^{n} k_\sigma(\mathbf{x} - \mathbf{x}_i)$$

where k_σ is called a ***kernel***

- A kernel density estimate is ***nonparametric***
- Another name for this is the ***Parzen window method***
- The σ parameter is called the ***bandwidth***
- Looks just like kernel ridge regression, but with equal weights
- Note that k_σ does not necessarily need to be an inner product kernel

Example



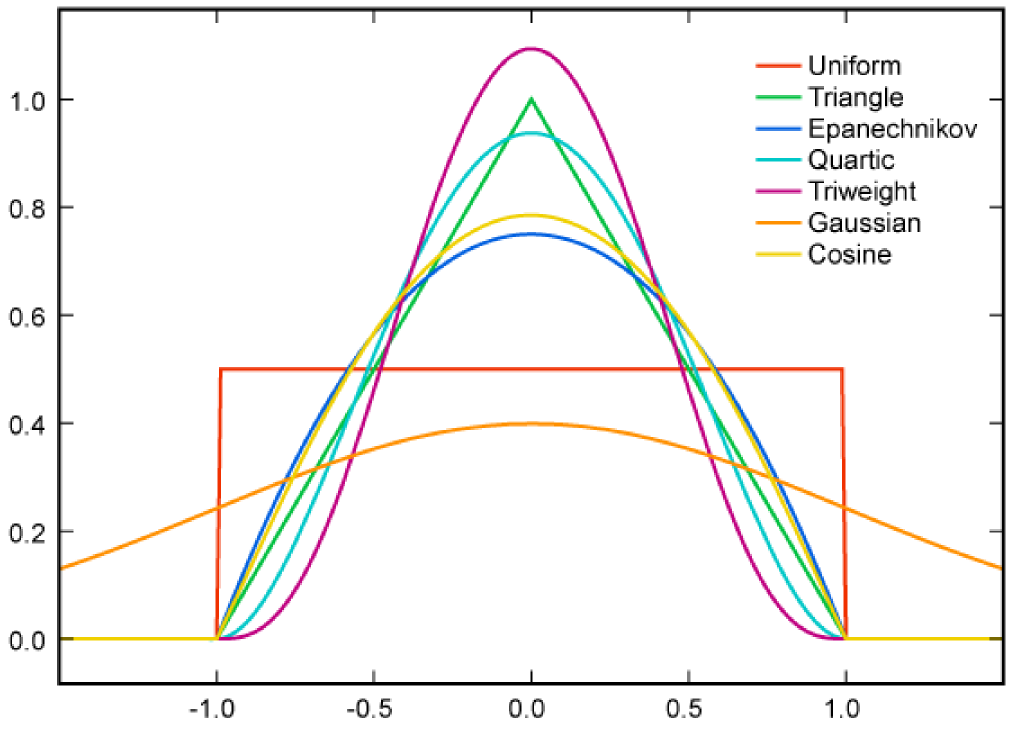
Kernels

In the context of density estimation, a kernel should satisfy

1. $\int k_\sigma(y)dy = 1$
2. $k_\sigma(y) \geq 0$
3. $k_\sigma(y) = \frac{1}{\sigma^d} D(\frac{\|y\|}{\sigma})$ for some D

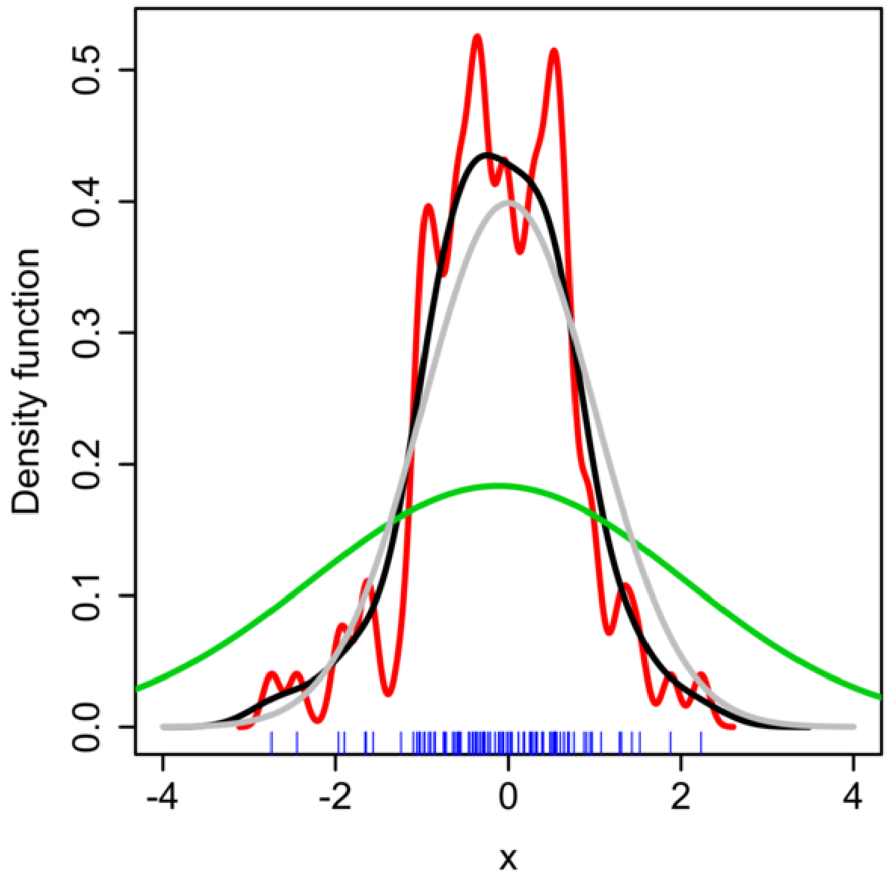
Examples (in $\mathbb{R}$)

- Uniform kernel
- Triangular kernel
- Epanichnikov kernel
- Gaussian
- …



Kernel bandwidth

The accuracy of a kernel density estimate depends critically on the bandwidth



Setting the bandwidth - Theory

Theorem

Let $\widehat{f}_\sigma(\mathbf{x})$ be a kernel density estimate based on the kernel k_σ

Suppose $\sigma = \sigma_n$ is such that

- $\sigma_n \to 0$ as $n \to \infty$
- $n\sigma_n^d \to \infty$ as $n \to \infty$

Then

$$\mathbb{E}\left[\int |\widehat{f}_\sigma(\mathbf{x}) - f(\mathbf{x})| d\mathbf{x}\right] \to 0$$

as $n \to \infty$, regardless of the true density $f(\mathbf{x})$

Proof: See Devroye and Lugosi, *Combinatorial Methods in Density Estimation* (1987)

Setting the bandwidth - Practice

Silverman's rule of thumb

If using the Gaussian kernel, a good choice for σ is

$$\sigma \approx 1.06\widehat{\sigma}n^{-1/5}$$

where $\widehat{\sigma}$ is the standard deviation of the samples

How can we apply what we know about model selection to setting σ?

- Randomly split the data into two sets
- Obtain a kernel density estimate for the first
- Measure how well the second set fits this estimate
  - e.g., compute another kernel density estimate on the second set and calculate the KL divergence between the two
- Repeat over many random splits and average

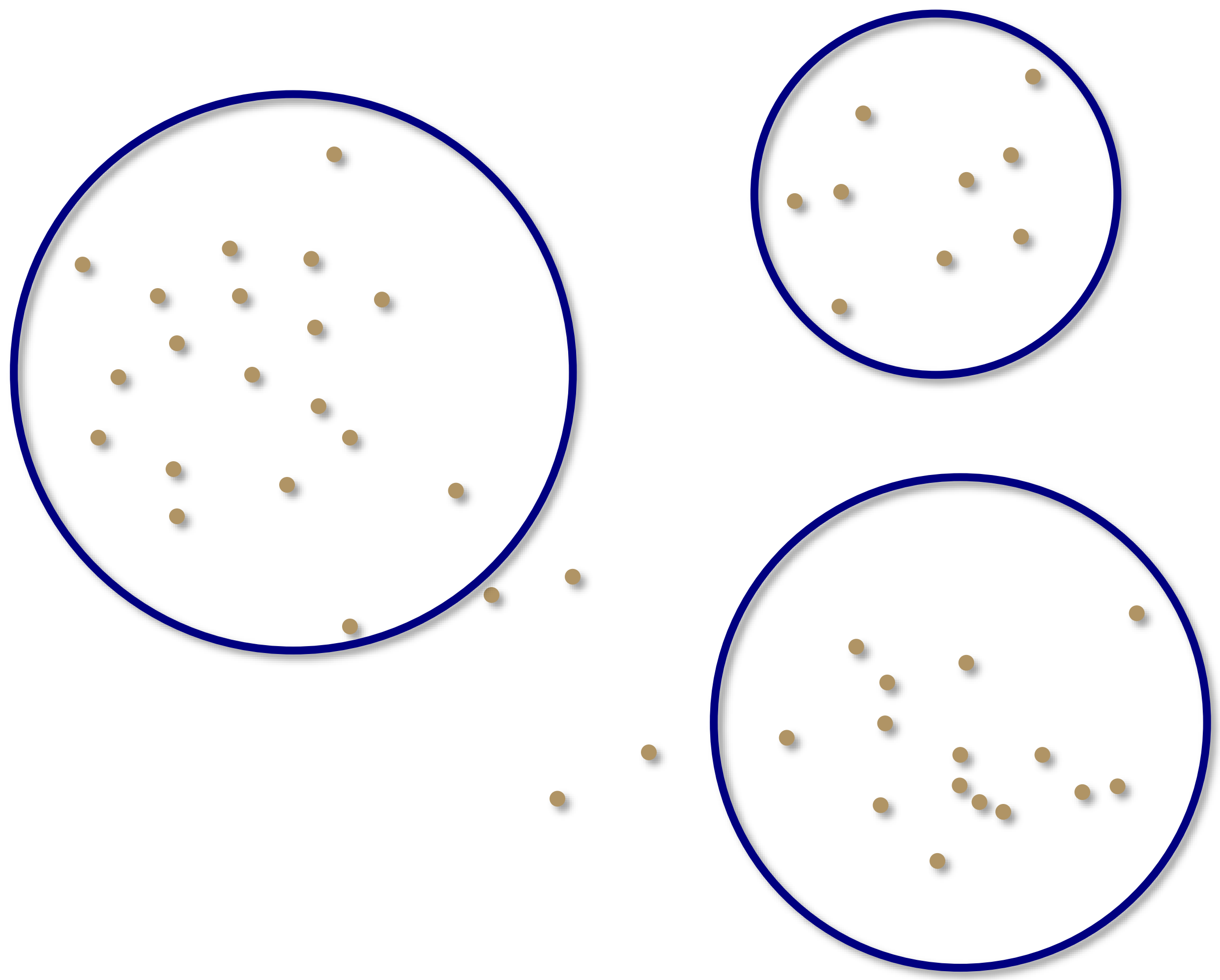
Density estimation is hard…

Kernel density estimation works fairly well if you have lots of data in extremely low-dimensional data

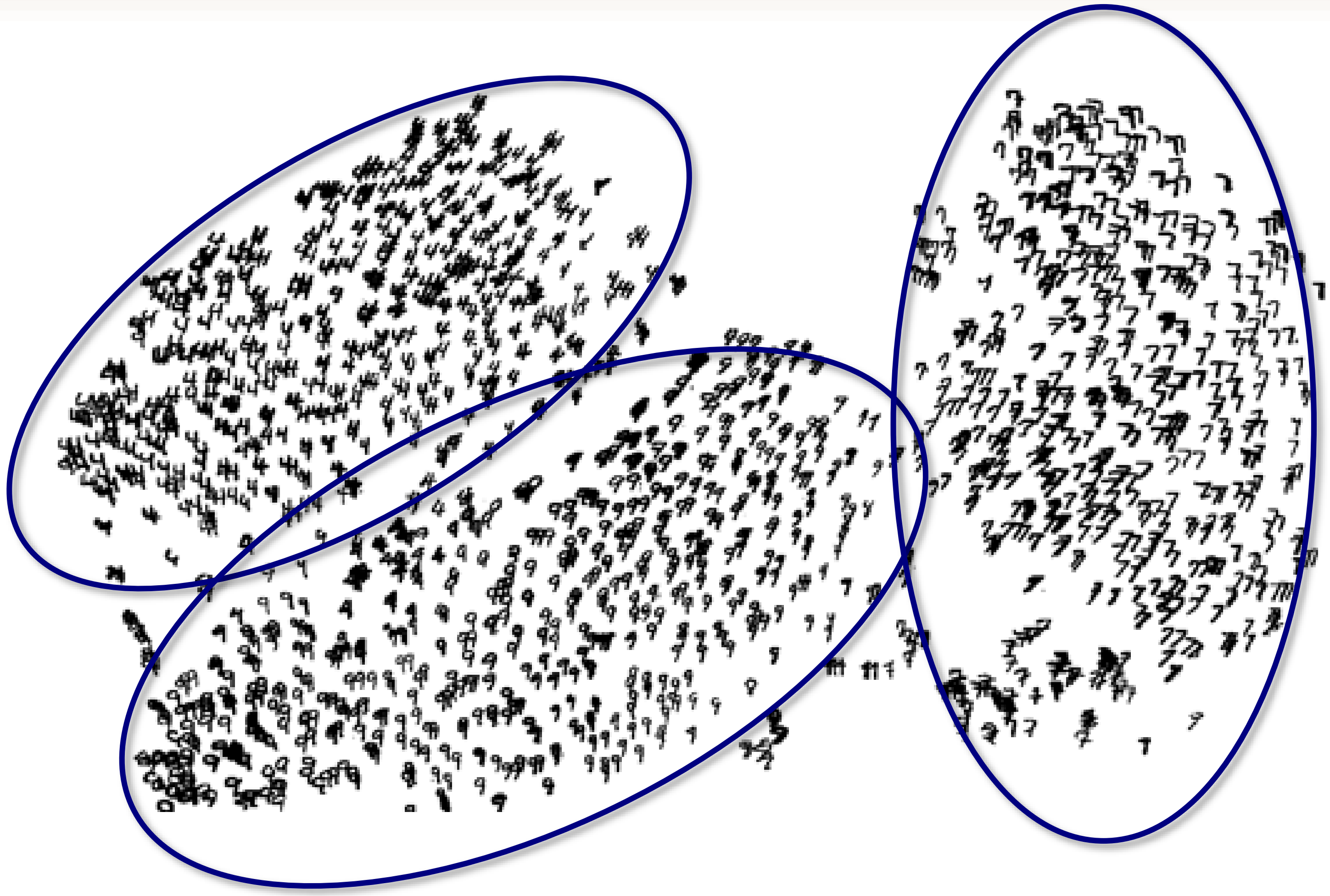
- e.g., 1 or 2 dimensions

Fortunately, it is not strictly necessary in many applications…

Clustering



Example



Formal definition

Suppose $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

The goal of ***clustering*** is to assign the data to disjoint subsets called ***clusters***, so that points in the same cluster are more similar to each other than points in different clusters

A clustering can be represented by a cluster map, which is a function

$$C : \{1, \ldots, n\} \to \{1, \ldots, K\}$$

where K is the number of clusters

K-means criterion

Choose C to minimize

$$W(C) = \sum_{k=1}^{K} \sum_{\underline{i:C(i)=k}} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$$

where

$$\underline{\boldsymbol{\mu}_k := \frac{1}{n_k} \sum_{i:C(i)=k} \mathbf{x}_i} \qquad n_k = |\{i : C(i) = k\}|$$

Note that K is assumed fixed and known

$W(C)$ is sometimes called the "***within-cluster scatter***"

Within-cluster scatter

It is possible to show that

$$W(C) = \sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$$

$$= \frac{1}{2} \sum_{k=1}^{K} \sum_{i:C(i)=k} \underbrace{\left[\frac{1}{n_k} \sum_{j:C(j)=k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \right]}_{\text{average distance between } \mathbf{x}_i \text{ and all other points in the same cluster}}$$

How many clusterings?

How many possible cluster maps C do we need to consider?

$$
\begin{aligned}
S(n, K) &= \text{\# of clusterings of } n \text{ objects into } K \text{ clusters} \\
&= S(n-1, K-1) + K\underline{S(n-1, K)}
\end{aligned}
$$

Solutions to this recurrence (with the natural boundary conditions) are called ***Stirling's numbers of the second kind***

$$S(n, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} \underline{k^n}$$

Examples

- $S(10, 4) = 34,105$
- $S(19, 4) \approx 10^{10}$

Minimizing the K-means criterion

There is no known efficient search strategy for this space

Can be solved (exactly) in time $O(n^{dK+1} \log n)$

Completely impractical unless both d and K are extremely small

- e.g., $d = 2, K = 3$ already results in $O(n^7 \log n)$

More formally, minimizing the K-means criterion is a ***combinatorial*** optimization problem (NP-hard)

Instead, we resort to an ***iterative, suboptimal algorithm***

Another look at K-means

Recall that we want to find

$$C^* = \arg\min_{C,\mu_k} \sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$$

Note that for fixed C

$$\underbrace{\boldsymbol{\mu}_k = \arg\min_{\mathbf{m}} \sum_{i:C(i)=k} \|\mathbf{x}_i - \mathbf{m}\|_2^2}$$

Therefore, we can equivalently write

$$C^* = \arg\min_{C,\{\mathbf{m}_k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2$$

An iterative algorithm

$$C^* = \arg\min_{C,\{\mathbf{m}_k\}_{k=1}^K} \underbrace{\sum_{k=1}^{K} \sum_{i:C(i)=k} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2}_{W(C,\{\mathbf{m}_k\}_{k=1}^K)}$$

This suggests an iterative algorithm

1. Given C, choose $\mathbf{m}_k$ to minimize $W(C,\{\mathbf{m}_k\}_{k=1}^K)$

2. Given $\mathbf{m}_k$, choose C to minimize $W(C,\{\mathbf{m}_k\}_{k=1}^K)$

K-means clustering algorithm

The solutions to each sub-problem are given by

1. $\mathbf{m}_k^* = \frac{1}{n_k} \sum_{i:C(i)=k} \mathbf{x}_i$

2. $C^*(i) = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2$

Algorithm

Initialize $\mathbf{m}_k, k = 1, \ldots, K$

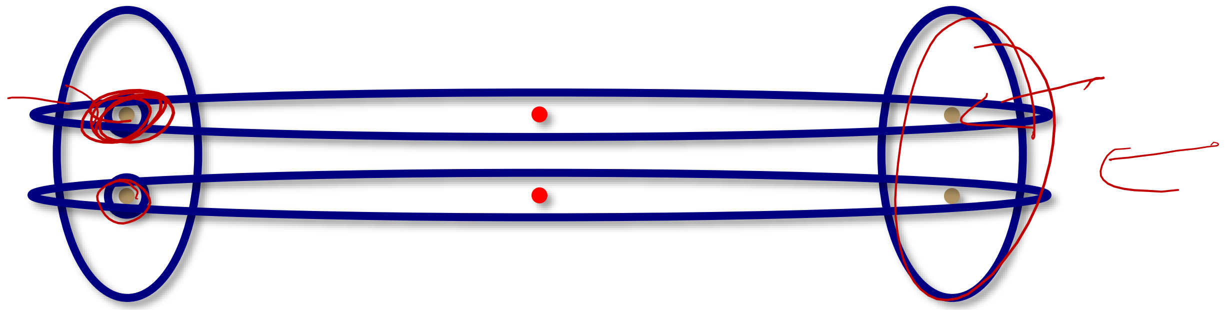
Repeat until clusters don't change

- $C(i) = \arg\min_k \|\mathbf{x}_i - \mathbf{m}_k\|_2^2$
- $\mathbf{m}_k = \frac{1}{n_k} \sum_{i:C(i)=k} \mathbf{x}_i$

Initialization

Traditionally, the algorithm is typically initialized by setting each $\mathbf{m}_k$ to be a ***random*** point in the dataset

However, depending on the initialization, the algorithm can get stuck in a local minimum



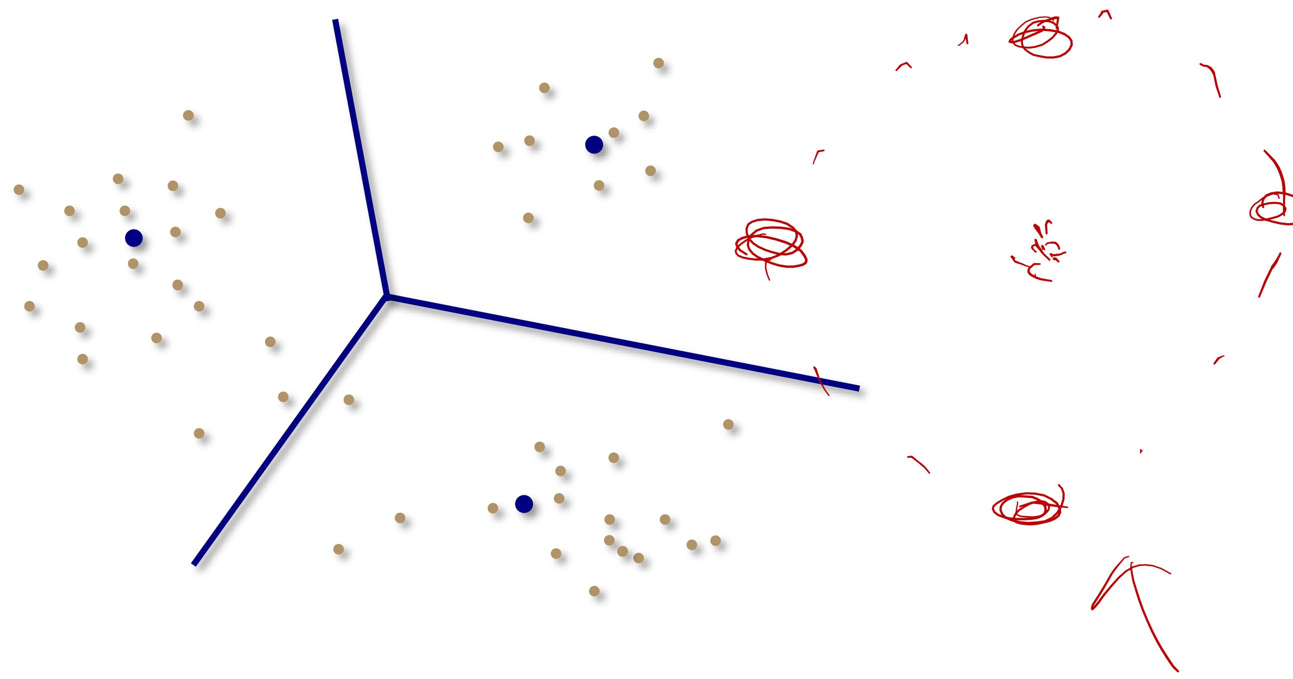
One can avoid this by:

- repeating for several random initializations
- initialize by ***sequentially*** selecting random points in the dataset, but with a probability depending on how far the point is from the already selected $\mathbf{m}_k$: *K-means ++*

Cluster geometry

Clusters are "nearest neighbor" regions or ***Vornoi cells*** defined with respect to the cluster means

Cluster boundaries are formed by the intersections of ***hyperplanes***



K-means will "fail" if clusters are ***nonconvex***

Remarks

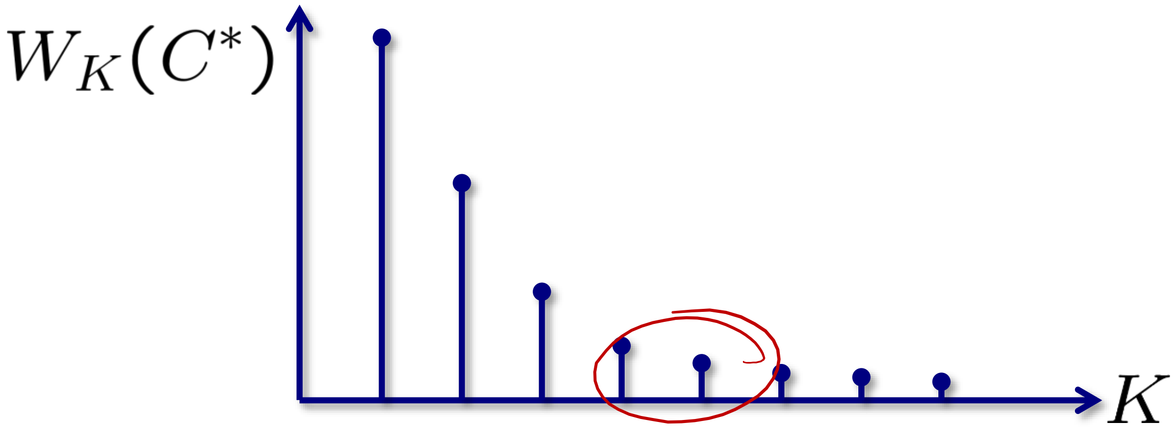
- Algorithm originally developed at Bell Labs as an approach to vector quantization

- If we replace the ℓ_2 norm with the ℓ_1 norm in our function $W(C)$, then
  - the geometry of our Vornoi regions will change
  - the "center" of each region is actually calculated via the median in each dimension
  - results in *K-medians clustering*

Model selection for K-means

How to choose K ?

Let $W_K(C^*)$ be the within-cluster scatter based on K clusters



If the "right" number of clusters is K^*, we expect

- for $K < K^*$, $W_K(C^*) - W_{K-1}(C^*)$ will be ***large***
- for $K > K^*$, $W_K(C^*) - W_{K-1}(C^*)$ will be ***small***

This suggests choosing K to be near the "knee" of the curve

Another take on K-means

I have followed the standard development of the K-means clustering algorithm, but there is another way to view this algorithm…

as simply another instance of structured matrix factorization

$$\mathbf{X} \approx \mathbf{B}\mathbf{C}$$

where

$$\mathbf{B} = \begin{bmatrix} | & & | \\ \mathbf{m}_1 & \cdots & \mathbf{m}_k \\ | & & | \end{bmatrix}$$

and $\mathbf{C}$ has exactly one “1” per column (the rest being zero)