# Nonparametrics and kernels (Part 2, Density estimation)

Daniel Lawson University of Bristol

Lecture 04.1.2 (v1.0.1)

# Signposting

- This is part 2 of Lecture 4.1, which is split into:
  - 4.1.1 covers Transforms
  - 4.1.2 covers Density estimation
  - 4.1.3 covers the Kernel Trick.

# Kernel density estimation (KDE)

- Let $\{\vec{x}_i\}_{i=1}^{N}$ be a dataset on some space (for simplicity taken as $\mathbb{R}^d$).
- Then the Kernel $K$ provides the density estimate for **any point** $\vec{y}$ as:

$$f_{\mathbf{H}}(\vec{y}) = \frac{1}{N}\sum_{i=1}^{N} K_{\mathbf{H}}\left(\vec{y} - \vec{x}_i\right),$$

  where $\mathbf{H}$ is a matrix of bandwidths.
- In other words, its a **sum of independent contributions** from each datapoint.
- It can be written:

$$K_{\mathbf{H}}\left(\vec{y} - \vec{x}_i\right) = \frac{1}{\det(\mathbf{H})} K\left(\mathbf{H}^{-1}(\vec{y} - \vec{x}_i)\right)$$

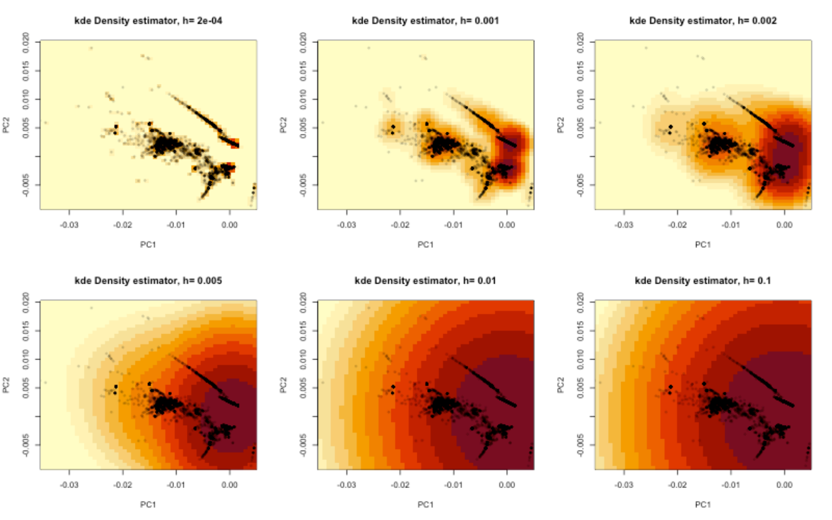
# KDE in 1d

- In 1D:

$$f_h(\vec{y}) = \frac{1}{N}\sum_{i=1}^{N} K\left(\frac{\vec{y}-\vec{x}_i}{h}\right)$$

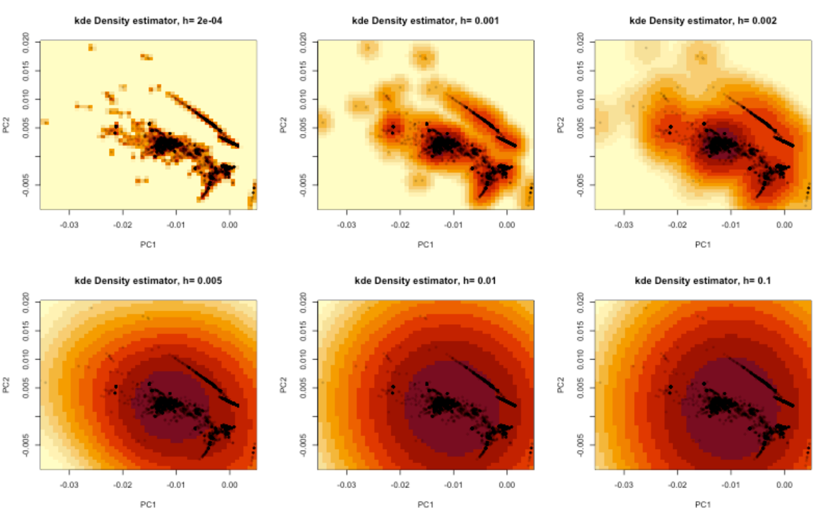
  - Its common to use a Normal kernel $K(x) = \mathrm{Normal}(x; \mu = 0, \sigma = 1)$.
  - $h$ can be chosen by minimising the "Mean Integrated Square Error"...
  - which theoretically suggests a functional form $h \propto N^{-1/5}$.
- Most density tools in packages use a reasonable default (which also depends on dimension).
  - This is appropriate for statistical inference of the density estimate at an unspecified point $x$.
- In practice the **"right" bandwidth** is a function of the **question**, so defaults might work poorly.
  - For **EDA**, we often want a **smaller bandwidth** to reveal potential data features

# kDE Example



kde Density estimator, h= 2e-04
kde Density estimator, h= 0.001
kde Density estimator, h= 0.002
kde Density estimator, h= 0.005
kde Density estimator, h= 0.01
kde Density estimator, h= 0.1
PC1
PC2
-0.03
-0.02
-0.01
0.00
-0.005
0.005
0.010
0.015
0.020

# KDE with unique points



# KDE kernels

- Some **important multivariate kernels**:
  - Spheroid Gaussian ($\mathbf{H}$ and $\Sigma$ are diagonal)
  - Rectangular ($\mathbf{H}$ is diagonal, Uniform kernel)
  - Product Gaussian ($\mathbf{H}$ off-diagonals are products, $\Sigma$ is diagonal)
- $\mathbf{H}$ is a parameter. It can be estimated by Cross-Validation but it is high dimensional so this is hard.

# Applications of KDE

- Kernel density estimates are considered important in many applications, including:
  - Smoothing
  - Clustering
  - Topological Data Analysis
  - Level set estimation
  - Feature Extraction
  - … etc!

# Signposting

- This is the half-way point for this lecture.
- The next lecture covers k-Nearest Neighbours and the Kernel Trick.

# K-Nearest neighbours

- Measuring neighbourhoods is a very important component of many applications.
- A fast way to do this is by computing for each point, their k-Nearest neighbours (k-NN).
- Note the requirement for a **distance measure** (metric or otherwise).
- Algorithms to do this are called **nearest neighbour search**:
  - **Linear algorithms**: Check all distances for all points. $O(N^2)$ to compute the structure.
  - **Space partitioning**: KD-trees etc partition the space. $O(N\log(N))$ but are less good in high dimensions…
  - **Approximate methods**: there are many great methods for this problem, which are often nearly perfect and much faster. Locality Sensitive Hashing is popular.

# k-NN density estimation

- A **Density estimate** using **k-NN**:

$$\hat{p}_{kNN}(x) = \frac{k}{N} \cdot \frac{1}{V_d R_k^d(x)}$$

- where:
  - $d$ is the dimension of the space,
  - $k$ is the number of neighbours,
  - $N$ is the sample size,
  - $R_k^d(x)$ is the "radius", i.e. the distance to the $k$-th closest neighbour of $x$, and
  - $V_d$ is the volume of a unit ball:

$$V_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)}$$

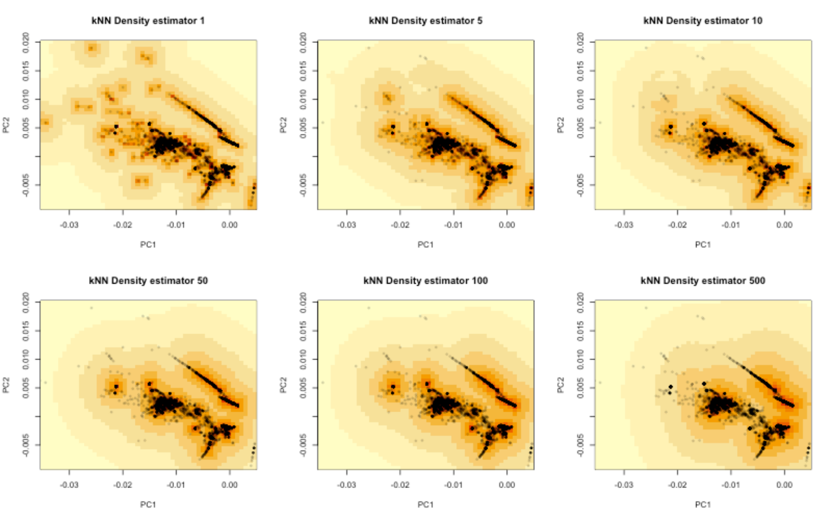
- so $V_1 = 2$, $V_2 = \pi$, $V_3 = \frac{4}{3}\pi$.
- NB $k$ is a **parameter**!

# k-NN density estimation

```
library("TDA")
Xseq <- seq(-0.035, 0.0046, length.out=50)
Yseq <- seq(-0.009, 0.02, length.out=50)
Grid <- expand.grid(Xseq, Yseq)

klist=c(1,2,5,10,20,50)
knnlist=lapply(klist,function(k){
    KNN <- knnDE(testdata_all.svd$u[,1:2], Grid, k)
    KNNm=matrix(KNN,nrow=length(Xseq),ncol=length(Yseq))
})
```

# k-NN density estimation



# Reflection

- When is regular KDE appropriate? How does it compare to nearest-neighbour approaches?
- When might neither be appropriate?
- What does the density estimate at a point mean?
- How could it be used in classification?
- What are its other uses?
- By the end of the course, you should:
  - Be able to implement kernel density estimation
  - Be able to reason about it's use for classification

# Signposting

- Next up: The Kernel Trick
- Further reading for Kernel Density Estimation:
  - Kernel Smoothing: Chapter 6 of The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Friedman, Hastie and Tibshirani).
  - For kNN Yen-Chi Chen's notes on kNN and the Basis