# Nonparametric Methods

Introduction to Machine Learning – GIF-7015

Professor: Christian Gagné

Week 4

UNIVERSITÉ LAVAL

## 4.1 Histogram estimation

# Nonparametric methods

- Parametric methods (including mixture models)
  - Probability densities ($p(\mathbf{x})$) selected in advance (typically, $\mathbf{x} \sim \mathcal{N}_D(\boldsymbol{\mu},\boldsymbol{\Sigma})$)
  - Search for the parameterization of these densities
- Nonparametric methods
  - Estimate the probability density directly from the data
  - No hypothesis *a priori* on the distribution of data
- Main approaches
  - Histogram estimation
  - Kernel density estimation
  - $k$-nearest neighbours ($k$-NN)

# Nonparametric density estimation

- Probability that value $x$ is less than or equal to $a$
  - $P(x \leq a) = \int\limits_{x=-\infty}^{a} p(x)\ dx$
  - Estimation by sampling $\{x^t\}_{t=1}^{N}$: $\hat{P}(x \leq a) = \frac{\#\{x^t \leq a\}}{N}$
- Estimated value $x$ in the range $[a, a+h]$

$$\hat{P}(a \leq x \leq (a+h)) = \frac{\#\{x^t \leq (a+h)\} - \#\{x^t \leq a\}}{N}$$

- Approximation of density $p(x)$ in $[a, a+h]$ by constant value $\hat{p}(x|x \in [a,(a+h)]) \approx \hat{p}(a)$

$$\begin{aligned} \hat{P}(a \leq x \leq (a+h)) &= \int\limits_{x=a}^{a+h} \hat{p}(x)\ dx \approx \hat{p}(a)(a+h-a) = h\hat{p}(a) \\ \hat{p}(x|x \in [a,(a+h)]) &\approx \frac{1}{h}\left[\frac{\#\{x^t \leq (a+h)\} - \#\{x^t \leq a\}}{N}\right] \end{aligned}$$

# Histogram estimation

- Histogram estimation (1D)
  - Divide the input space into compartments of equal size (*bins*)
  - Each bin is $h$ wide and positioned with respect to an origin $x_0$

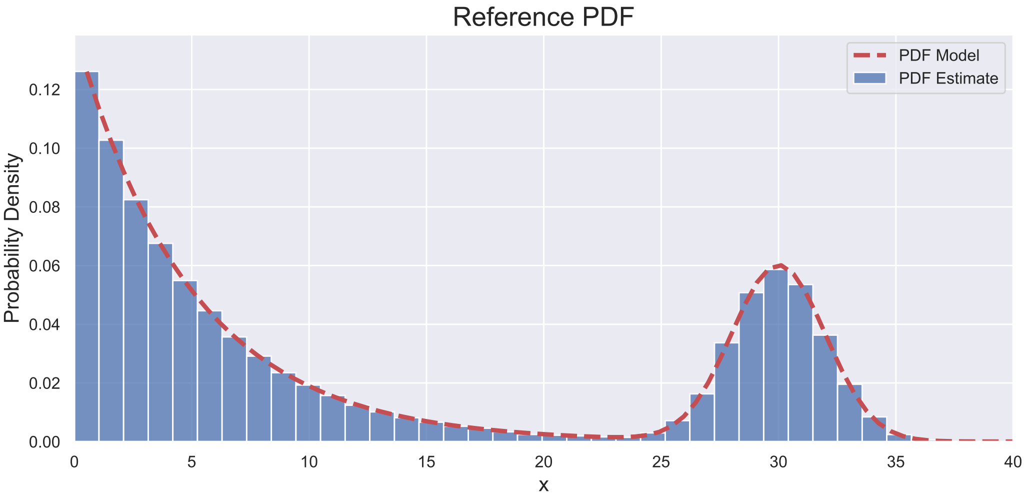
$$]x_0 + mh, x_0 + (m+1)h], \text{ with } m \text{ n natural number}$$

  - Estimation in 1D, from a set $\{x^t\}_{t=1}^N$

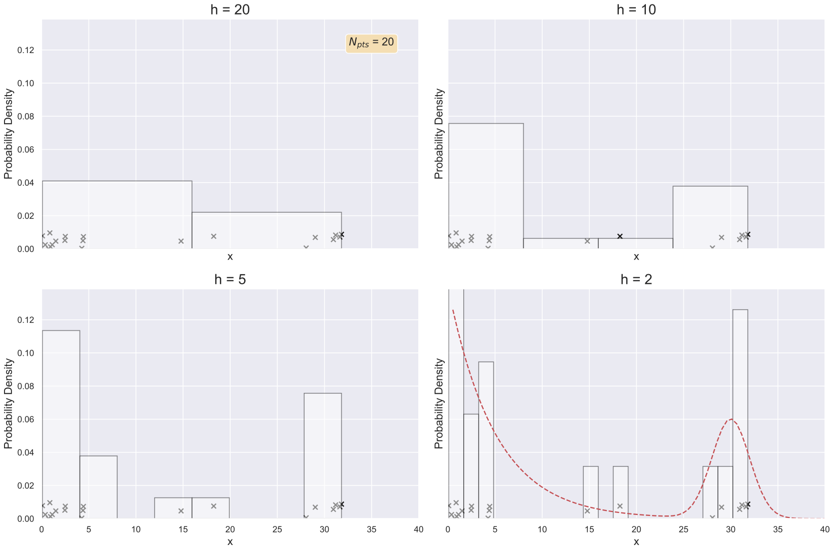
$$\hat{p}(x) = \frac{\#\{x^t \text{ in the same bin than } x\}}{Nh}$$

- Choice of origin $x_0$ may slightly affect the estimator (boundary discontinuities)
- Choice of width $h$ significantly affects the estimator
  - If the value of $h$ is low, many peaks in the estimate
  - If the value of $h$ is high, softer (less accurate) estimate

# Histogram density estimation



# Histogram density estimation



# Estimation in many dimensions

- Histogram estimation in many dimensions
  - Bins corresponding to equal hypervolume hypercubes
  - Highly impacted by the curse of dimensionality
- General conditions for estimates to converge to the true probability density, $\hat{p}(\mathbf{x}) \to p(\mathbf{x})$
  - Volume $V_n$ of each bin reduced

$$\lim_{n\to\infty} V_n = 0$$

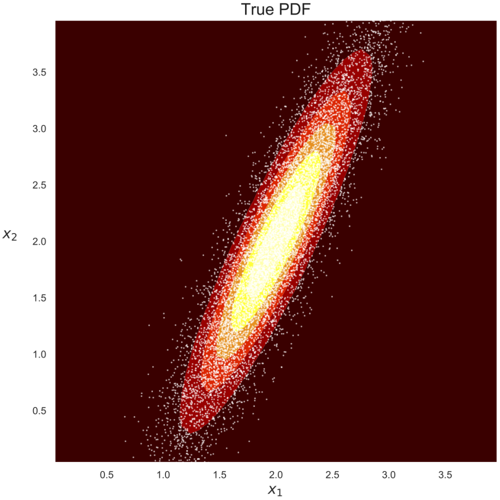
  - Number of observations $k_n$ per bin is very high

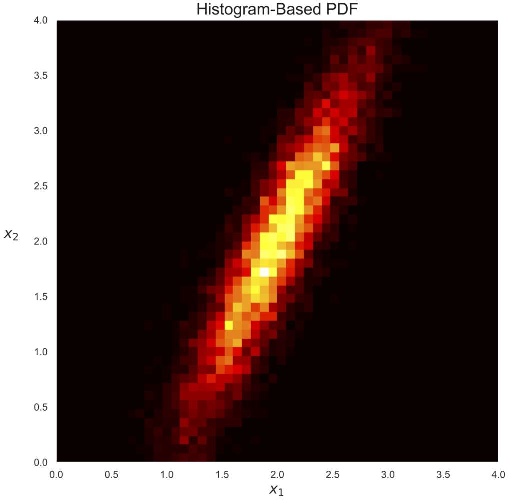
$$\lim_{n\to\infty} k_n = \infty$$

  - Ratio of the number of observations per bin to total number of observations is high

$$\lim_{n\to\infty} \frac{k_n}{n} = 0$$

# 2D density estimations



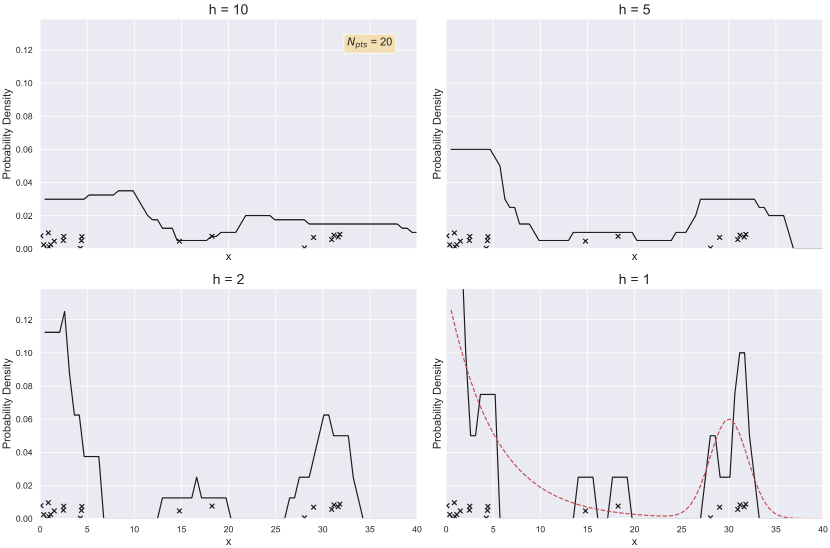
# Naive histogram density estimation

- Naive histogram estimator (also known as a *Parzen window*)
  - Estimate the density around $x$ in a hypercube of width $2h$
  - Formulation in 1D

$$
\begin{aligned}
\hat{p}(x) &= \frac{\#\{(x-h) < x^t \leq (x+h)\}}{2Nh} \\
&= \frac{1}{2Nh}\sum_{t=1}^{N} \mathrm{w}\left(\frac{x-x^t}{h}\right) \\
\text{where } \mathrm{w}(u) &= \begin{cases} 1 & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

  - Removes the origin $x_0$
  - The estimation is not continuous and has steps at $x^t \pm h$

# Naive histogram density estimation



## 4.2 Kernel density estimation

# Kernel density estimation

- Kernel density estimation: softer estimation than the naive histogram estimator
  - Use a *softening kernel*, typically a Gaussian kernel

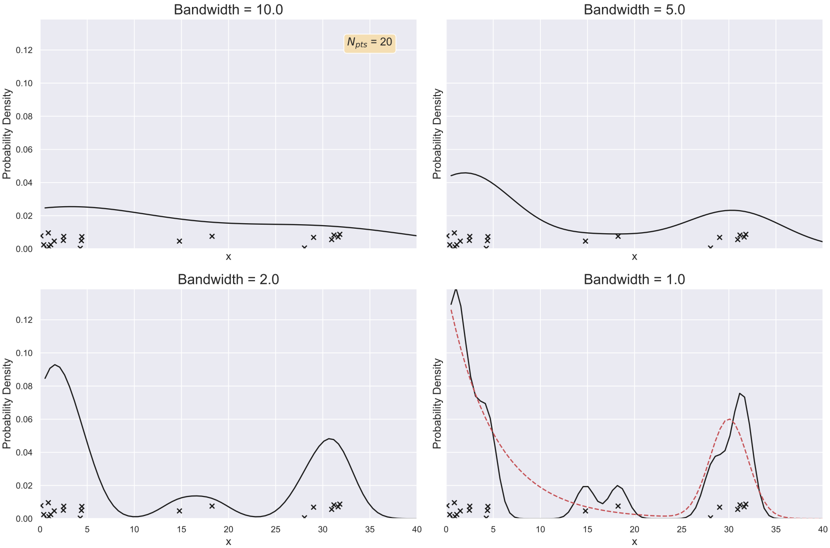
$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right]$$

  - Convolution of the softening kernel with data $\{x^t\}_{t=1}^N$

$$\hat{p}(x) = \frac{1}{Nh} \sum_{t=1}^{N} K\left(\frac{x - x^t}{h}\right)$$

  - Kernel $K(\cdot)$ determines the shape of influence of the data
  - Window width $h$ determines the width of the data influence
  - Generalizes the naive estimation, which uses a rectangular box as a kernel

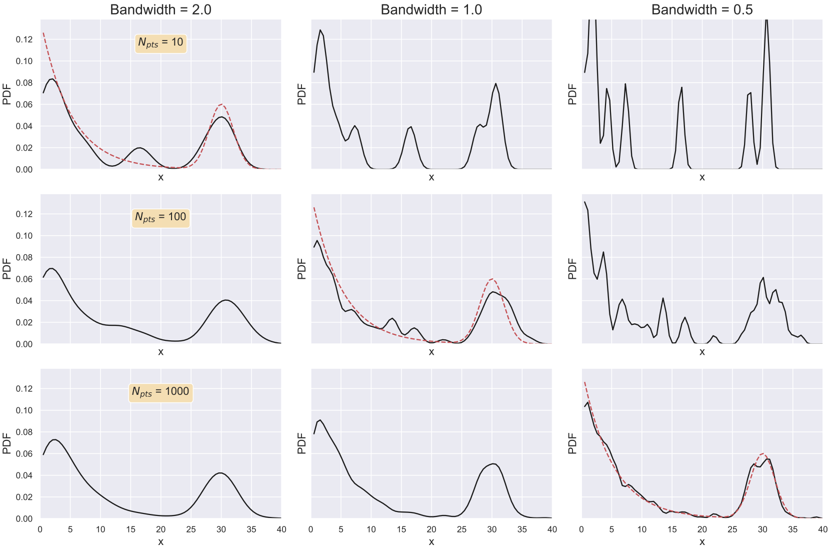
# Kernel density estimation



# Quality of the kernel density estimate

- Window width greatly influences the estimate
  - $h$ small: each data has an important local effect
  - $h$ large: smoother estimation, with overlapping between kernels
- Estimation $\hat{p}(x) \to p(x)$ when $N \to \infty$
  - $h$ has to $\to 0$, but slower than $N$ (i.e. $Nh \to \infty$)
  - Typically, we set $h_N = h_1/\sqrt{N}$, using a window of $h_N$ for a dataset of size $N$

# Varying the number of observations

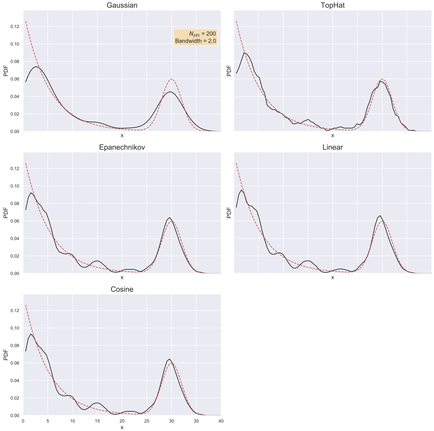


# Properties of softening kernels

- Desirable properties of a softening kernel
  1. Positive (or zero) values: $K(x) \geq 0, \forall x$
  2. Area under the curve is equal to 1: $\int\limits_{-\infty}^{\infty} K(x)\ dx = 1$
  3. Centred on the origin: $\int\limits_{-\infty}^{\infty} x\, K(x)\ dx = 0$
- If properties 1 and 2 are respected, $K(u)$ corresponds to a valid density function and therefore $\hat{p}(x)$ is also valid
- Moreover, if $K(u)$ is continuous and differentiable, $\hat{p}(x)$ also is
- Support: spreading of $u$ values for which $K(u)$ is non-zero

# Examples of softening kernels

- Gaussian
  - Differentiable, but support is not bounded
- Boxcar / TopHat: Naive histogram estimation
  - Bounded support, non-differentiable function
- Epanechnikov: $K(u) = (3/4)\,(1 - u^2)$ for $u \in [-1,1]$
  - Bounded support, non-derivable function
- Linear / triangle: $K(u) = 1 - |u|$ for $u \in [-1,1]$
  - Bounded support, non-derivable function
- Cosinus: $K(u) = \cos(u\,\pi/2)$ for $u \in [-1,1]$
  - Bounded support, non-derivable function



# Kernel estimation, multidimensional case

- General equation of the kernel estimation in $D$ dimensions

$$\hat{p}(\mathbf{x}) = \frac{1}{Nh^D} \sum_{t=1}^{N} K\left(\frac{\mathbf{x} - \mathbf{x}^t}{h}\right)$$

  - Kernel constraint: $\int_{\mathbb{R}^D} K(\mathbf{x})\, d\mathbf{x} = 1$
- Multivariate Gaussian kernel

$$K(\mathbf{u}) = \left(\frac{1}{\sqrt{2\pi}}\right)^D \exp\left[-\frac{\|\mathbf{u}\|^2}{2}\right]$$

- Sensitive to dimensionality and normalization of values in different dimensions
- Kernel including a normalization based on the covariance estimation $\boldsymbol{\Sigma}$

$$K(\mathbf{u}) = \frac{1}{(2\pi)^{0.5D}|\boldsymbol{\Sigma}|^{0.5}} \exp\left[-0.5\mathbf{u}^\top \boldsymbol{\Sigma}^{-1}\mathbf{u}\right]$$

# Kernel estimation for classification

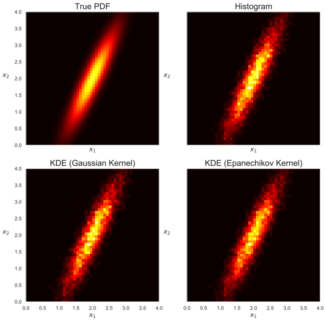
- Kernel estimation of $\hat{p}(\mathbf{x}|C_i)$

$$\hat{p}(\mathbf{x}|C_i) = \frac{1}{N_i h^D} \sum_{t=1}^{N} K\left(\frac{\mathbf{x}-\mathbf{x}^t}{h}\right) r_i^t$$

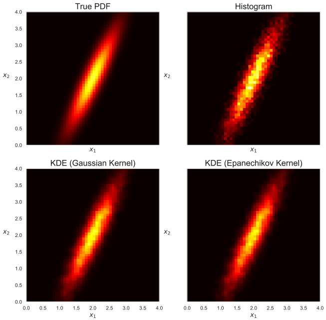
  - Corresponding discriminant function

$$\begin{aligned} \hat{P}(C_i) &= \frac{N_i}{N} \\ \mathrm{h}_i(\mathrm{x}) &= \hat{p}(\mathbf{x}|C_i)\hat{P}(C_i) \\ &= \frac{1}{N h^D} \sum_{t=1}^{N} K\left(\frac{\mathbf{x}-\mathbf{x}^t}{h}\right) r_i^t \end{aligned}$$

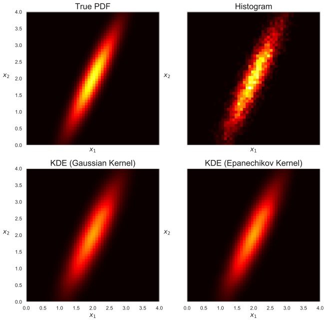
# Kernel width: impact the classification




Narrow




Medium




Large

## 4.3 $k$-nearest neighbours

# $k$-NN density estimation

- $k$-nearest neighbours ($k$-NN)
  - Reference dataset $\mathcal{X} = \{x^t\}_{t=1}^N$
  - Adapt the window width according to the local data density ($k$ closest data)

$$\hat{p}(x) = \frac{k}{2Nd_k(x,\mathcal{X})}$$

  - $h = d_k(x,\mathcal{X})$: distance from the $k$-th neighbour to the $x$ data in $\mathcal{X}$
  - Non-continuous estimator, similar to the naive histogram estimator, with adaptive $h$ width

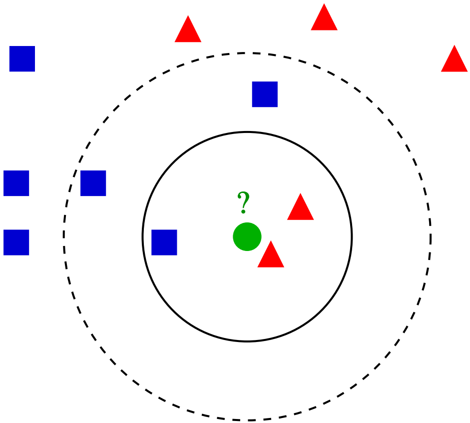
# $k$-NN parameters

- $k$-NN is defined by three main parameters
  - Number of neighbours $k$
    - $k$ low: narrow space division based on the reference dataset
    - $k$ high: smoother, larger divisions, average depending on the neighbourhood
  - Distance measurement $D(\mathbf{x},\mathbf{y})$
    - Defines the neighbourhood relationship between the data
  - Reference dataset $\mathcal{X}$
    - Dataset size
    - Density of distribution in the data space
    - Data representativeness (filtering)

# $k$-NN classification

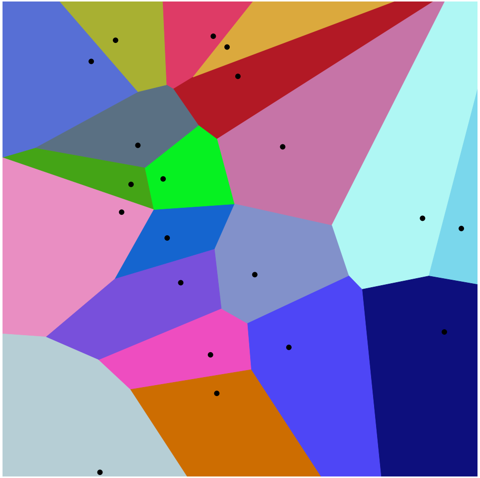
- $k$-nearest neighbours classification
  - Reference (training) dataset $\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$
  - To classify an unknown data $\mathbf{x}$, compute the $k$-closest neighbours in $\mathcal{X}$ using a distance measure (e.g., Euclidean distance)
  - Assign to $\mathbf{x}$ the most frequent label among those of the $k$-nearest neighbours
- Very simple and direct method for classification
- With $k = 1$, divide the input space according to a Voronoi diagram based on $\mathcal{X}$.

# $k$-NN classification

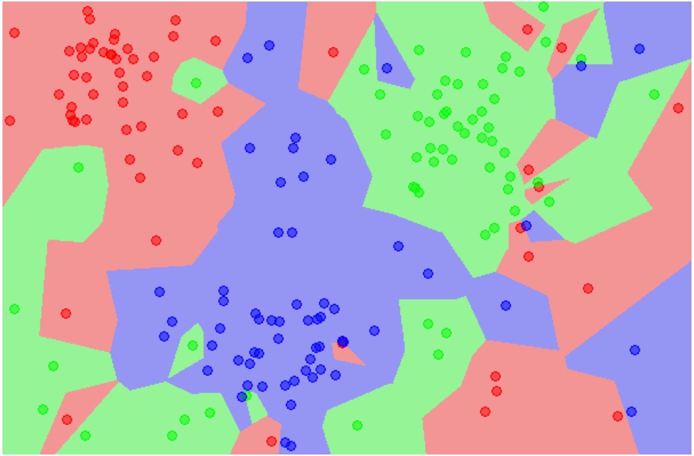



By Antti Ajanki, CC-BY-SA 3.0, `https://en.wikipedia.org/wiki/File:KnnClassification.svg`

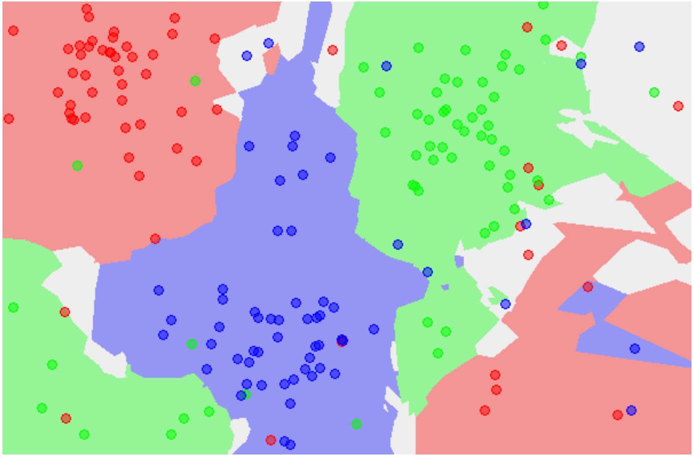
# Voronoï diagram (1-NN)



# Regions and borders for 1-NN



# Regions and borders for 5-NN

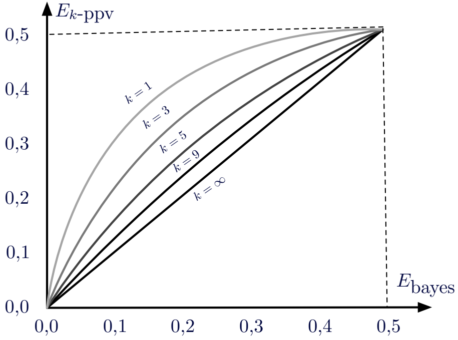


By Agor153, CC-BY-SA 3.0, `https://en.wikipedia.org/wiki/File:Map5NN.png`

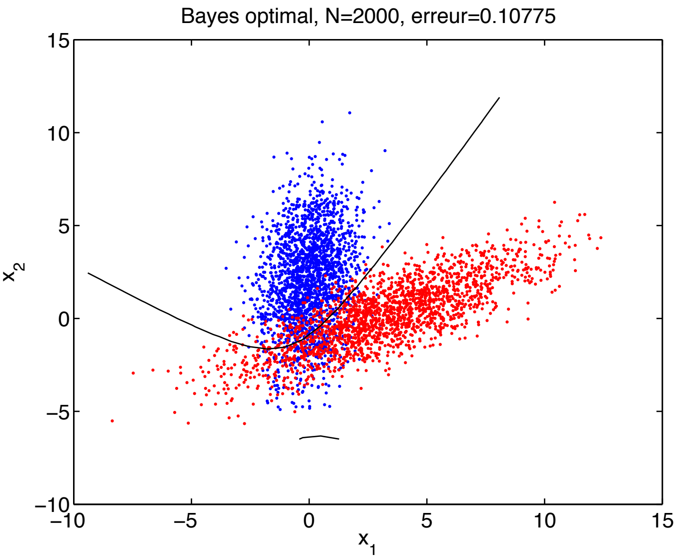
## 4.4 Notions about $k$-NN

# Bounds of the $k$-NN classifier

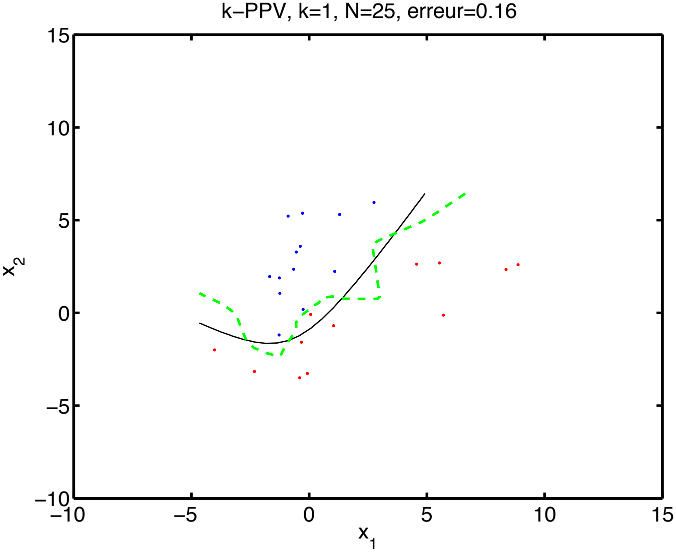
- Optimal Bayesian error rate ($E_{\text{bayes}}$)
  - Error rate when true class probability densities are known
  - Optimal, impossible to do better in generalization
- Two bounds on the $k$-NN error rate
  - With $k = 1$ and $N \to \infty$ then $E_{1\text{-ppv}} \leq 2E_{\text{bayes}}$
  - With $k \to \infty$ and $N \to \infty$ then $E_{k\text{-ppv}} \to E_{\text{bayes}}$



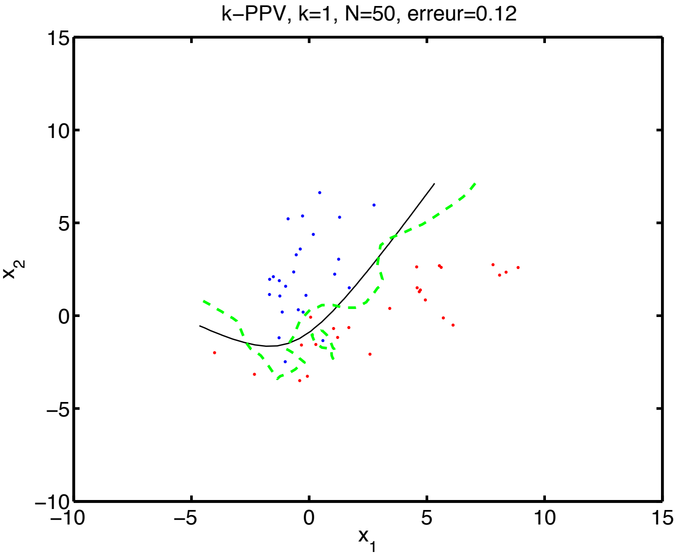
# Optimal Bayesian classification ($N = 2000$)



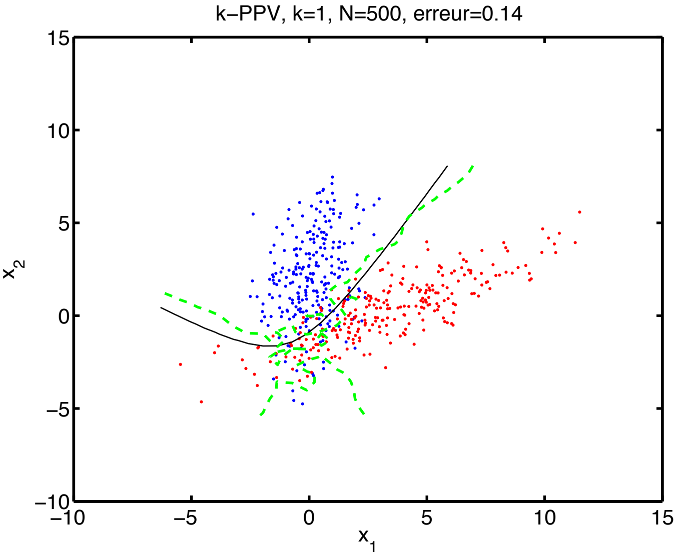
# Varying the number of observations, $k = 1$, $N = 25$



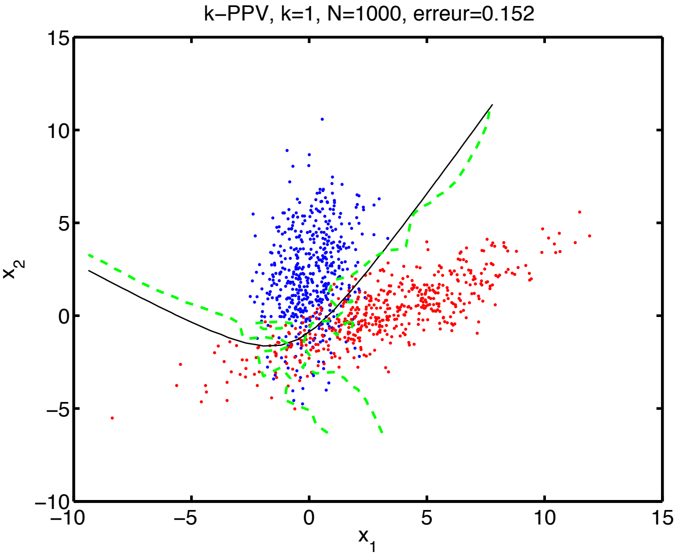
# Varying the number of observations, $k = 1$, $N = 50$



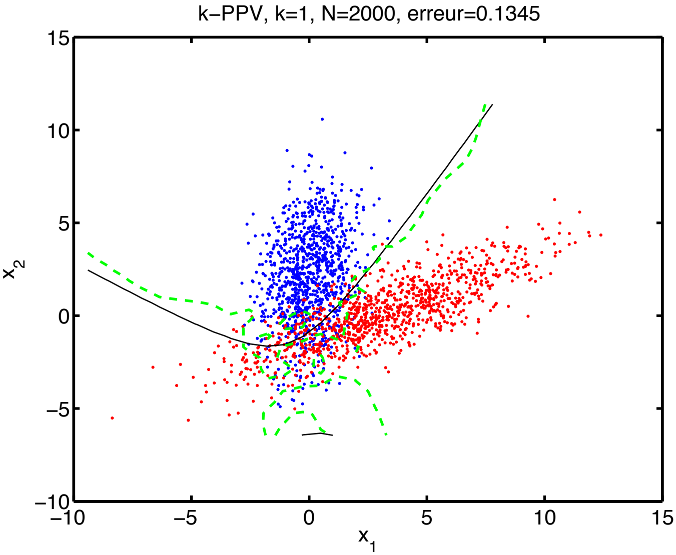
# Varying the number of observations, $k = 1$, $N = 500$



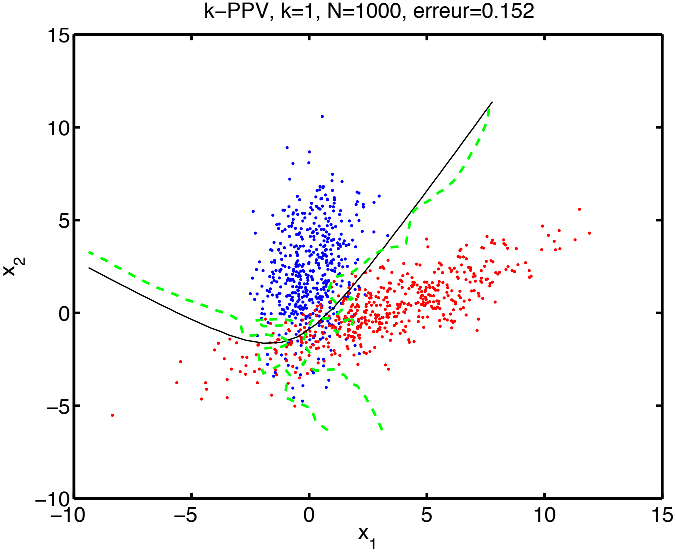
# Varying the number of observations, $k = 1$, $N = 1000$



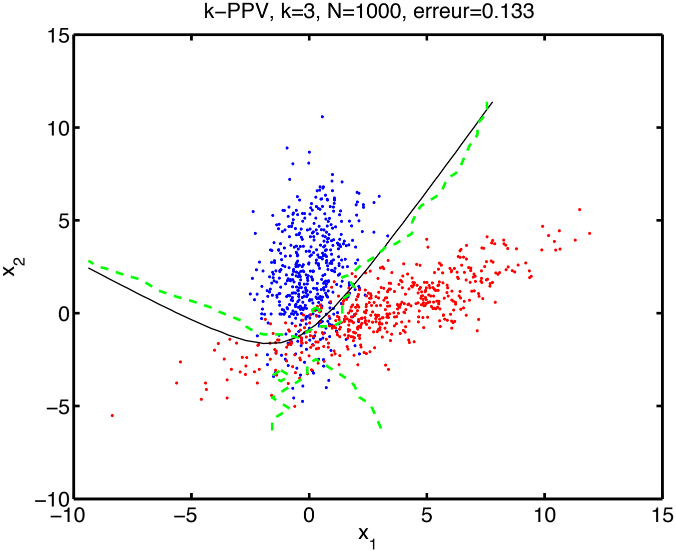
# Varying the number of observations, $k = 1$, $N = 2000$



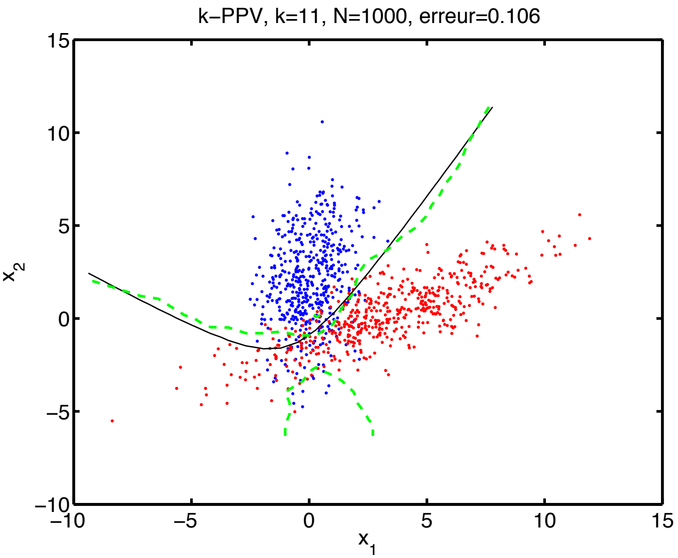
# Varying the number of neighbours, $k = 1$, $N = 1000$



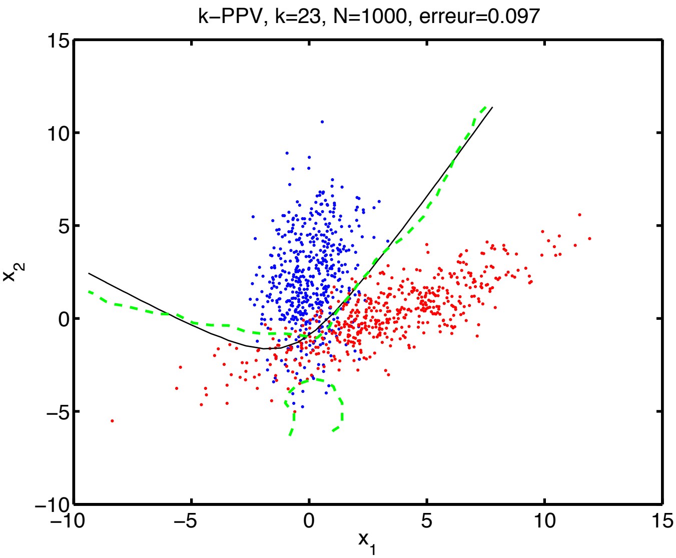
# Varying the number of neighbours, $k = 3$, $N = 1000$



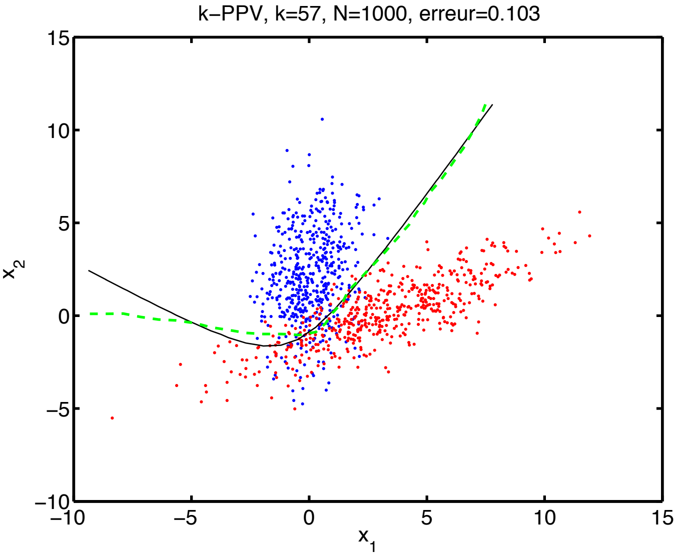
# Varying the number of neighbours, $k = 11$, $N = 1000$



# Varying the number of neighbours, $k = 23$, $N = 1000$



# Varying the number of neighbours, $k = 57$, $N = 1000$



# Distances

- The distance measurement gives the neighbourhood relationship between the data
- Mathematical definition of a metric $\mathrm{D} : X \times X \mapsto \mathbb{R}$
  - Non-negativity: $\mathrm{D}(\mathbf{x},\mathbf{y}) \geq 0$
  - Reflexivity: $\mathrm{D}(\mathbf{x},\mathbf{y}) = 0$ iff $\mathbf{x} = \mathbf{y}$
  - Symmetry: $\mathrm{D}(\mathbf{x},\mathbf{y}) = \mathrm{D}(\mathbf{y},\mathbf{x})$
  - Inequality of the triangle: $\mathrm{D}(\mathbf{x},\mathbf{z}) \leq \mathrm{D}(\mathbf{x},\mathbf{y}) + \mathrm{D}(\mathbf{y},\mathbf{z})$
- Different distance measurements are possible
  - Euclidean distance
  - Minkowsky distance
  - Tanimoto distance (distance between sets)
  - Tangent distance

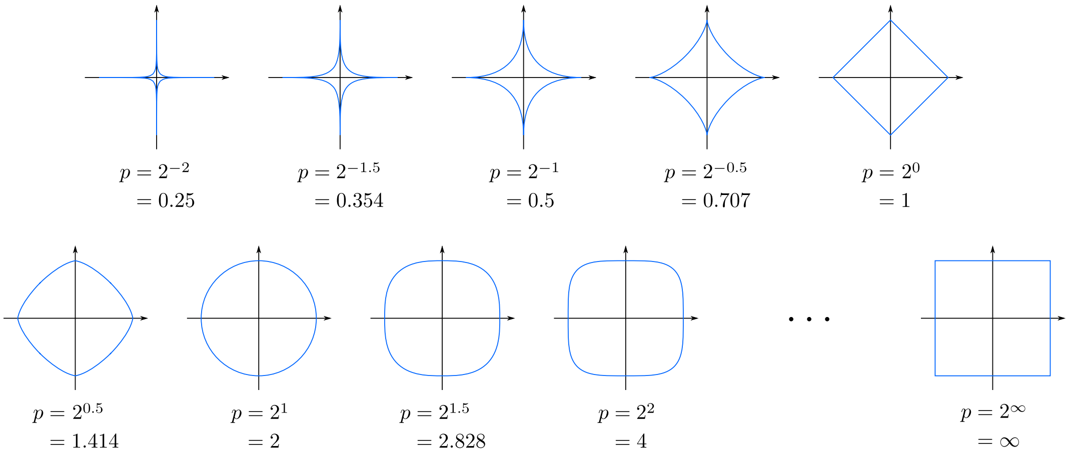
# Minkowsky distance

- Minkowsky distance

$$\mathrm{D}_p(\mathbf{x},\mathbf{y}) = \left( \sum_{i=1}^{D} |x_i - y_i|^p \right)^{1/p}$$

  - Parameter $p$ controls the emphasis on the dimensions where the magnitude is greatest
- Manhattan distance ($p = 1$), equal weight for all the dimensions: $\mathrm{D}_1(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{D} |x_i - y_i|$
- Distance $D_\infty$, using only the dimension where the difference is of maximum magnitude: $\mathrm{D}_\infty(\mathbf{x},\mathbf{y}) = \max_{i=1}^{D} |x_i - y_i|$
- Euclidean distance ($p = 2$), trade-off between these extremes: $\mathrm{D}_2(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{D} (x_i - y_i)^2}$

# Illustration of the Minkowsky distance



# Data normalization

- Distance measurement sensitive to data scale of all the dimensions
  - Values in a dimension where the scale is large relative to the other dimensions will absorb the value of the other dimensions

$$|x_j - y_j| \gg |x_i - y_i|, \forall i \neq j \quad \Rightarrow \quad \mathrm{D}(\mathbf{x},\mathbf{y}) \approx |x_j - y_j|$$

- Standardization of the data is necessary if the scales are different according to the dimensions
  - Standardization according to the meaning of the data (physical units)
  - Standardization according to max-min value of each dimension
  - Whitening transformation

# Performance evaluation leave-one-out

- No training required with $k$-NN
  - Training simply consists in storing the dataset $\mathcal{X}$
- *Leave-one-out* performance evaluation
  - Takes advantage of zero cost training
  - Corresponds to $K$-folds cross validation, with $K = N$

1. For each data $\mathbf{x}^t \in \mathcal{X}$:
   1.1 Calculate the $k$-NN of $\mathbf{x}^t$ among the $\mathcal{X} \backslash \{\mathbf{x}^t\}$ set
   1.2 Determine the most common label of these $k$ closest neighbours as a classification label of $\mathbf{x}^t$
2. For computing the error rate, return the ratio between the number of misclassified data in $\mathcal{X}$ and the size of $\mathcal{X}$

## 4.5 Computational efficiency of $k$-NN

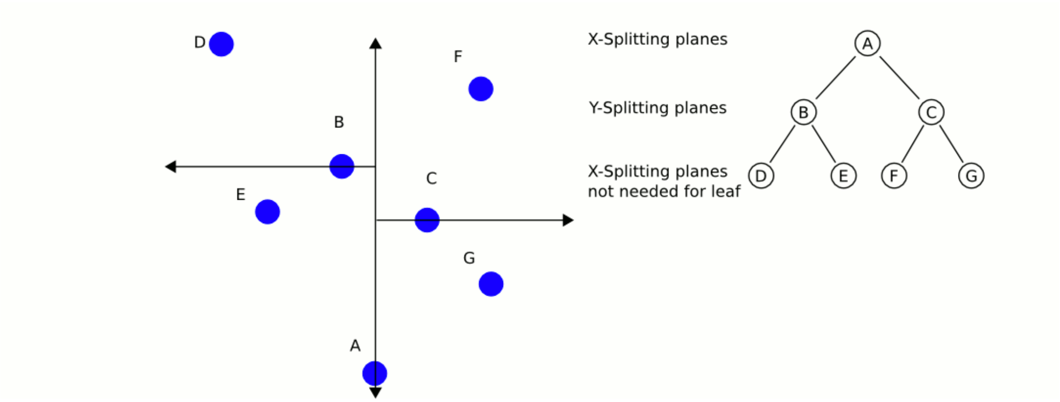
# Algorithmic complexity of $k$-NN

- Training: data storage in memory
  - Complexity in time and memory: $O(N)$
- Data processing (test/validation): get the $k$ neighbours
  - Get the $k$ closest neighbours of $\mathbf{x}$ in $\mathcal{X}$: complexity in time $O(N \log N)$ (naive algorithm)
  - Classifying $M$ data: complexity in time $O(MN \log N)$
- It is possible to optimize the calculations by using more sophisticated methods

# KD-Tree

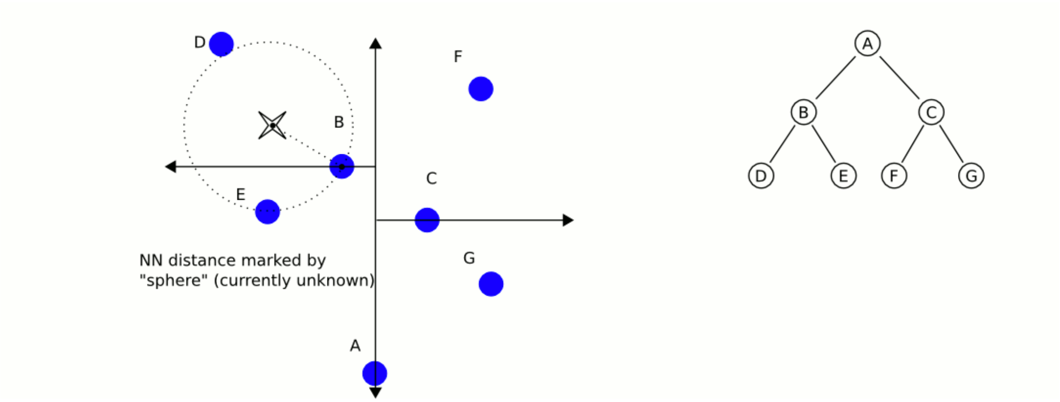
- Structure/topology of data in space can be exploited for the search of the $k$-NN
  - Avoid calculating the distance with some data, which are anyway too far from the data under test
- KD-tree: tree-like data structure capturing data topology in a Euclidean space
  - Construction of the KD-Tree for $N$ data: $O(N \log N)$.
  - Required storage space of KD-Tree: $O(N)$.
  - Querying the $k$-NN of a data in a KD-Tree
    - $O(N^{\frac{D-1}{D}} + k)$ in $D$ dimensions
    - $O(\sqrt{N} + k)$ in 2D
    - $O(\log N)$ with $k = 1$
  - Processing of $M$ data: $O(M(N^{\frac{D-1}{D}} + k))$
- Efficient implementations of KD-tree are available (e.g., CGAL in C++, `scipy.spatial.KDTree` in Python)

# Illustration of KD-tree



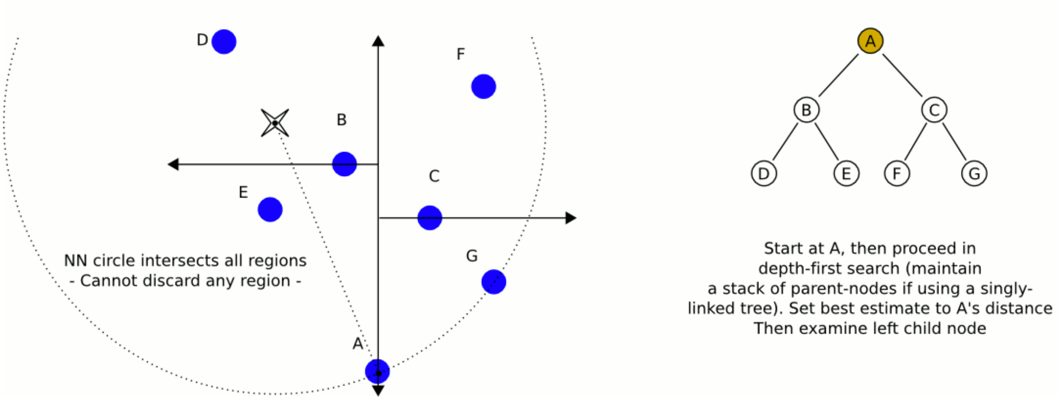

By User_A1, CC-BY-SA 3.0, `https://commons.wikimedia.org/wiki/File:KDTree-animation.gif`

# Illustration of KD-tree

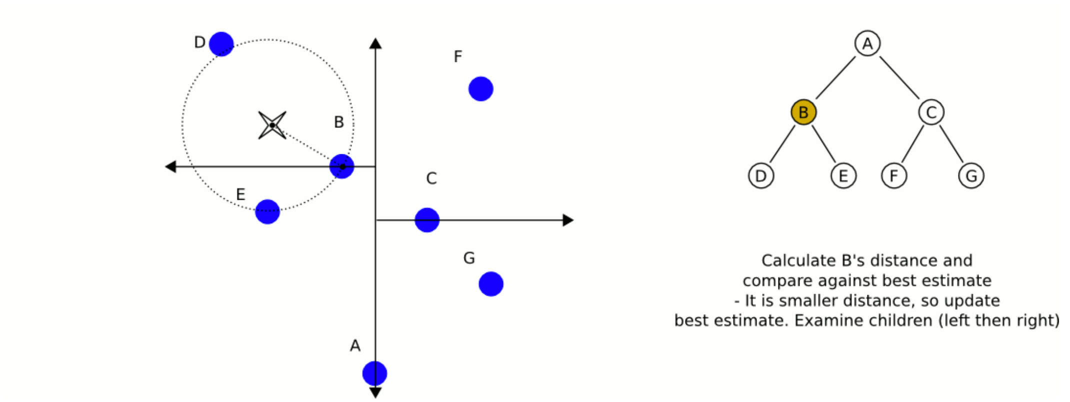



By User_A1, CC-BY-SA 3.0, `https://commons.wikimedia.org/wiki/File:KDTree-animation.gif`

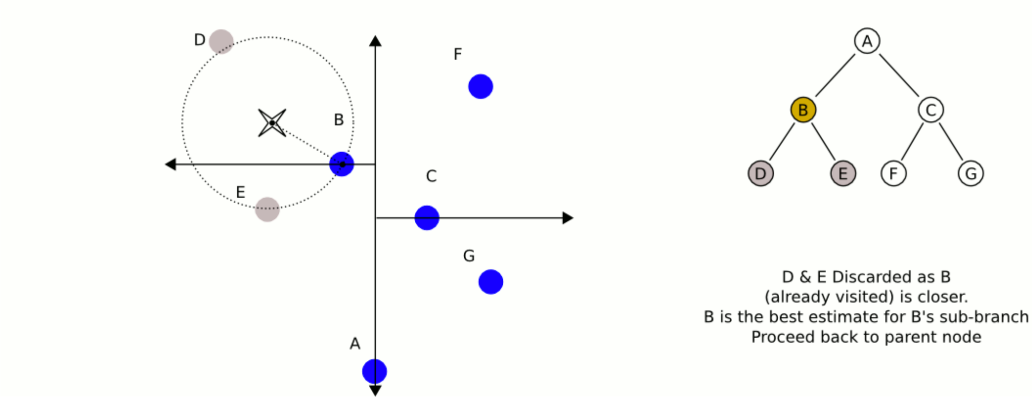
# Illustration of KD-tree



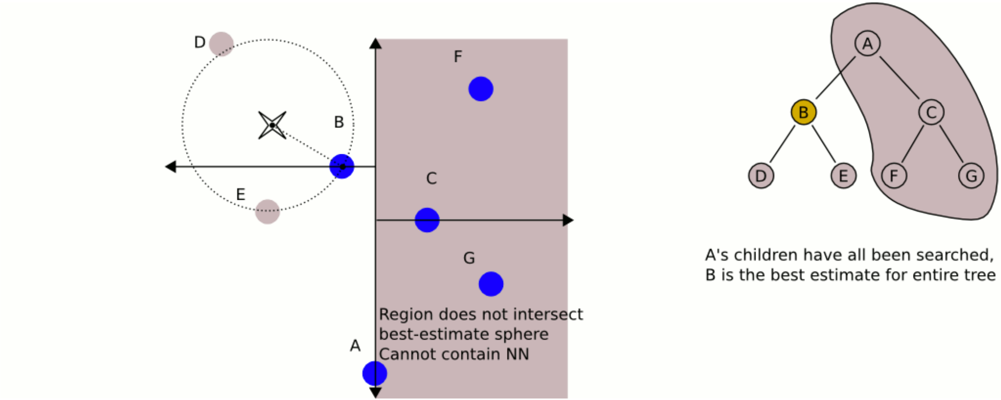
# Illustration of KD-tree



# Illustration of KD-tree



# Illustration of KD-tree



## 4.6 Prototype selection

# Size of training dataset

- Trade-off to make on the size of the training set
  - With $N \to \infty$, the algorithm tends toward optimal performance
  - But with $N \to \infty$, processing time and storage needs are huge
- Depending on the position, data density may vary
  - Far from decision boundaries, point density can be reduced
  - Outliers or noisy data in a different class region could be removed
- Approximation of decision boundaries by selecting a few representatives

# Hart condensation

- Hart condensation
  - Objective: select only $\mathcal{X}$ data contributing to the classification
  - Heuristics making an incremental construction of the set of prototypes
- Approach
  - Start with an almost empty set of prototypes (a randomly chosen data)
  - Add data only if they are misclassified according to the NN
  - Repeat as long as there are misclassified unselected data

# Hart condensation

1. Create a set of prototypes selected from an $\mathbf{x}$ data randomly chosen in $\mathcal{X}$, $\mathcal{Z} = \{\mathbf{x}\}$
2. As long as the $\mathcal{Z}$ set is modified relative to the previous iteration:
   2.1 For each data $\mathbf{x}^t \in \mathcal{X}$, processed in random order:
      2.1.1 Determine the closest neighbour of $\mathbf{x}^t$ in $\mathcal{Z}$

      $$\mathbf{x}' = \operatorname*{argmin}_{\mathbf{x} \in \mathcal{Z}} \|\mathbf{x}^t - \mathbf{x}\|$$

      2.1.2 If the class label of $\mathbf{x}'$ does not match the class label of $\mathbf{x}^t$ ($r' \neq r^t$), then select the data as a prototype, $\mathcal{Z} = \mathcal{Z} + \{\mathbf{x}^t\}$
3. Return the set $\mathcal{Z}$ as the prototypes selected in $\mathcal{X}$

# Wilson edition

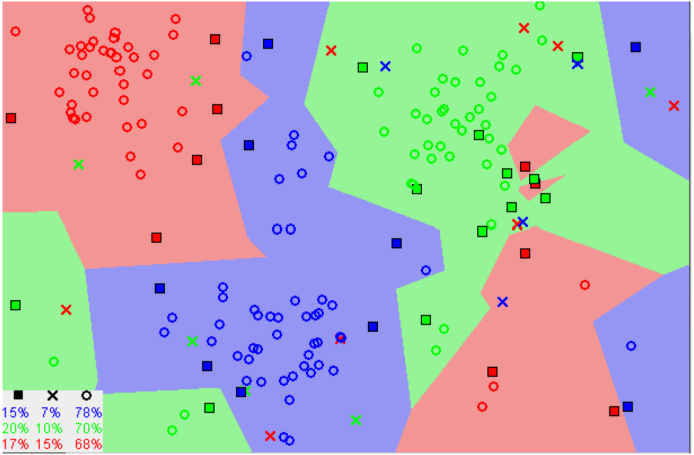
- Wilson edition
  - Heuristics to remove misclassified data from $\mathcal{X}$ according to *leave-one-out*
  - Eliminates data that is thought to be aberrant or noisy

1. Create the set of prototypes $\mathcal{Z}$ from all the data, $\mathcal{Z} = \mathcal{X}$
2. For each data $\mathbf{x}^t \in \mathcal{Z}$, processed in random order:
   2.1 Determine $\mathcal{V}$, which are the $k$-NN of $\mathbf{x}^t$ in $\mathcal{Z} \backslash \{\mathbf{x}^t\}$
   2.2 Determine the classification label $r^t_{\mathcal{V}}$ of $\mathbf{x}^t$ according to the most common label of the data in $\mathcal{V}$
   2.3 If the $r^t_{\mathcal{V}}$ label is different from the $r^t$ label of $\mathbf{x}^t$, then remove the data from $\mathcal{Z}$, $\mathcal{Z} = \mathcal{Z} \backslash \{\mathbf{x}^t\}$
3. Return the set $\mathcal{Z}$ as the prototypes selected in $\mathcal{X}$

# Other approaches to generate prototypes

- Aggressive selection of prototypes: Wilson's edition followed by Hart's condensation
  - Filter $\mathcal{X}$ by first eliminating aberrant or noisy data (Wilson edition)
  - Select only the data contributing to the classification (Hart condensation)
- Prototype building
  - Determine prototypes that are not data in $\mathcal{X}$
  - Possible approach (unsupervised): $K$-means of $\mathcal{X}$ with high $K$ value

# Wilson + Hart illustration






×: data removed by Wilson ($k = 3$) □: prototypes selected by Hart ○: data removed by Hart

## 4.7 Nonparametric methods in scikit-learn

# Scikit-learn: density estimation

- `neighbors.KernelDensity`: kernel density estimation
  - Parameters
    - `bandwidth` (float): kernel width
    - `algorithm` (string): neighbourhood algorithm to use, can be `'kd_tree'`, `'ball_tree'` or `'auto'` (default: `'auto'`)
    - `kernel` (string): noyau utilisé, peut être `'gaussian'`, `'tophat'`, `'epanechnikov'`, `'exponential'`, `'linear'` ou `'cosine'` (default: `'gaussian'`)
  - Methods
    - `fit(X)`: learn density from data
    - `sample(n_samples=1)`: generates samples of the distribution (only for Gaussian and tophat kernels)
    - `score(X)`: returns the log-likelihood of the data
    - `score_samples(X)`: returns the density of data

# Scikit-learn: $k$-nearest neighbours

- `neighbors.KneighborsClassifier`: classification with the $k$-nearest neighbours method
  - Parameters
    - `n_neighbors` (int): number of neighbours used (default: 5)
    - `algorithm` (string): neighbourhood algorithm to use, can be `'kd_tree'`, `'ball_tree'`, `'brute'` or `'auto'` (default: `'auto'`)
    - `metric` (string or object `neighbors.DistanceMetric`): distance metric used. By default `'minkowski'` with $p = 2$, which returns to a Euclidean distance. For other metrics, see documentation of `neighbors.DistanceMetric`.
    - `p` (int): value of $p$ for the Minkowski distance (default: 2)
  - Methods
    - `fit(X,y)`: learn classification model from data
    - `kneighbors(X, n_neighbors)`: returns the $k$-nearest neighbours to the data
    - `predict(X)`: does the data classification
- `neighbors.KneighborsRegressor`: regression by $k$-nearest neighbours