

Méthodes non paramétriques

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professeur: Christian Gagné

Semaine 4



UNIVERSITÉ
LAVAL

4.1 Estimation par histogramme

- Méthodes paramétriques (incluant densité-mélange)
 - Densités de probabilité ($p(\mathbf{x})$) posées à l'avance (typiquement, $\mathbf{x} \sim \mathcal{N}_D(\boldsymbol{\mu}, \boldsymbol{\Sigma})$)
 - Recherche de la paramétrisation de ces densités
- Méthodes non paramétriques
 - Estimer la densité de probabilité directement à partir des données
 - Aucune hypothèse *a priori* sur la distribution des données
- Approches principales
 - Estimation par histogramme
 - Estimation par noyau
 - k -plus proches voisins (k -PPV)

Estimation non paramétrique de densités

- Probabilité que valeur x inférieure ou égale à a

- $P(x \leq a) = \int_{x=-\infty}^a p(x) dx$

- Estimation avec échantillonnage $\{x^t\}_{t=1}^N$: $\hat{P}(x \leq a) = \frac{\#\{x^t \leq a\}}{N}$

- Estimation de valeur x dans l'intervalle $[a, a + h]$

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

- Approximation de densité $p(x)$ dans $[a, a + h]$ par valeur constante
 $\hat{p}(x|x \in [a, (a + h)]) \approx \hat{p}(a)$

$$\hat{P}(a \leq x \leq (a + h)) = \int_{x=a}^{a+h} \hat{p}(x) dx \approx \hat{p}(a)(a + h - a) = h\hat{p}(a)$$

$$\hat{p}(x|x \in [a, (a + h)]) \approx \frac{1}{h} \left[\frac{\#\{x^t \leq (a + h)\} - \#\{x^t \leq a\}}{N} \right]$$

Estimation par histogramme

- Estimation par histogramme (1D)
 - Diviser l'espace d'entrée en compartiments de tailles égales (*bins*)
 - Chaque compartiment est de largeur h et positionné par rapport à une origine x_0

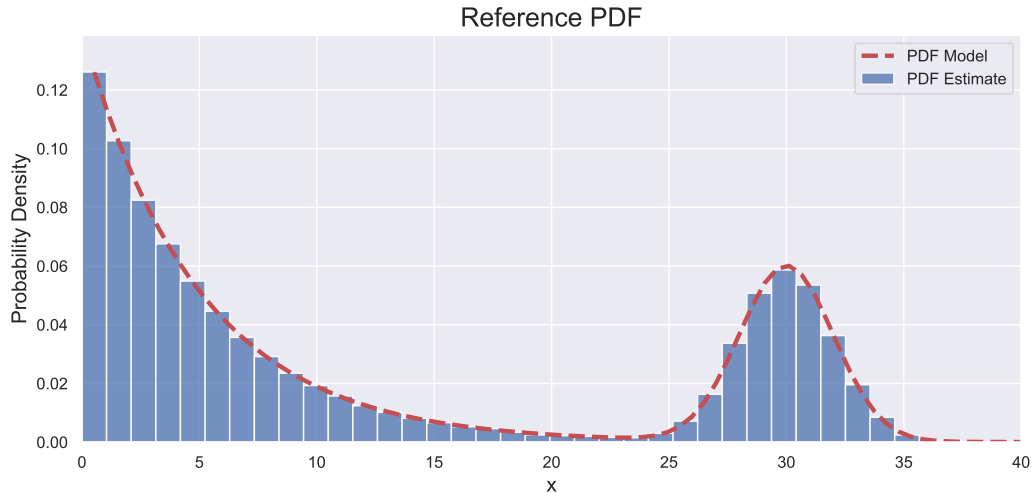
$]x_0 + mh, x_0 + (m + 1)h]$, avec m un nombre naturel

- Estimation en 1D, à partir d'un jeu $\{x^t\}_{t=1}^N$

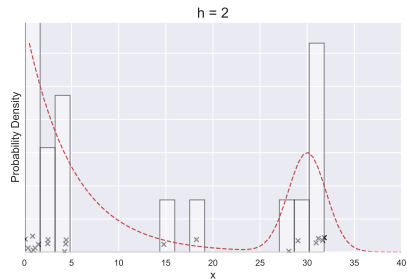
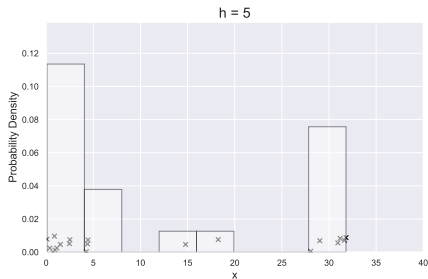
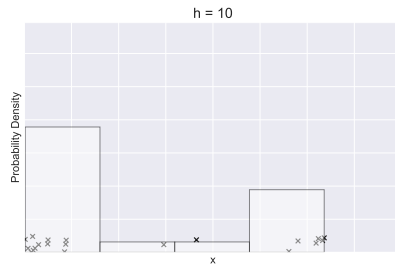
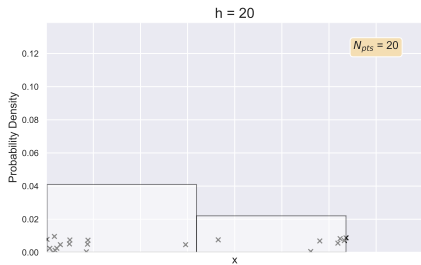
$$\hat{p}(x) = \frac{\#\{x^t \text{ dans même compartiment que } x\}}{Nh}$$

- Choix de l'origine x_0 peut affecter légèrement l'estimateur (discontinuités aux frontières)
- Choix de la largeur h affecte significativement l'estimateur
 - Valeur de h faible, de nombreux pics dans l'estimation
 - Valeur de h grande, estimation plus douce (moins précise)

Densité d'estimation par histogramme



Densité d'estimation par histogramme



Estimation selon plusieurs dimensions

- Estimation en plusieurs dimensions par histogramme
 - Compartiments correspondant à des hypercubes d'hypervolumes égaux
 - Souffre gravement de la malédiction de la dimensionnalité
- Conditions générales pour estimations convergent vers véritables densités de probabilité, $\hat{p}(\mathbf{x}) \rightarrow p(\mathbf{x})$
 - Volume V_n de chaque compartiment réduit

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

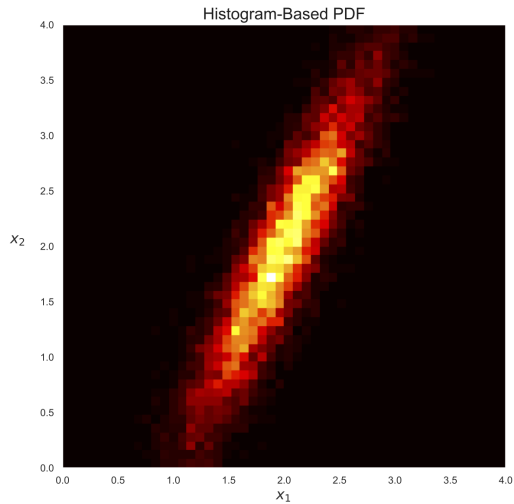
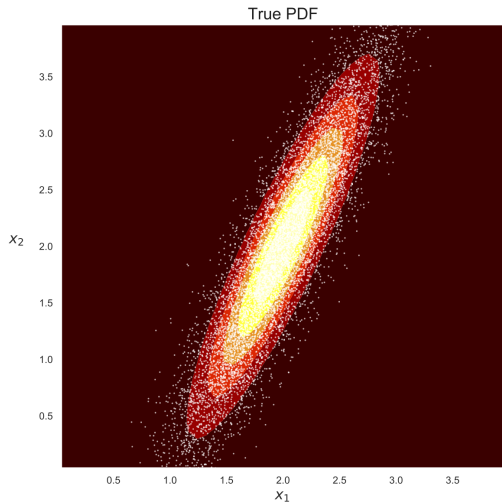
- Nombre d'observations k_n par compartiment très grand

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

- Ratio du nombre d'observations par compartiment avec le nombre d'observations total élevé

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

Densités d'estimation en 2D



Densités par estimation naïve d'histogramme

- Estimateur naïf d'histogramme (aussi connu comme une *fenêtre de Parzen*)
 - Estimer la densité autour de x dans un hypercube de largeur $2h$
 - Formulation en 1D

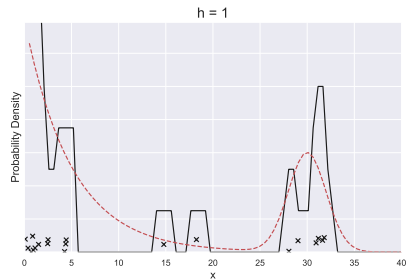
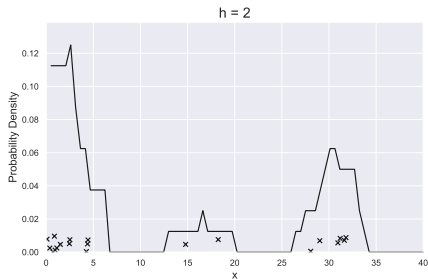
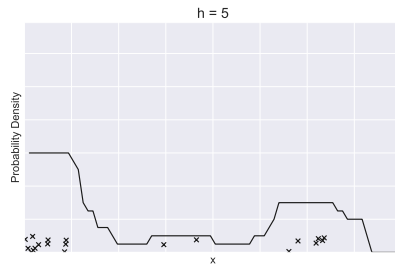
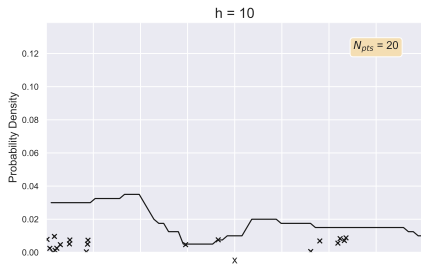
$$\hat{p}(x) = \frac{\#\{(x-h) < x^t \leq (x+h)\}}{2Nh}$$

$$= \frac{1}{2Nh} \sum_{t=1}^N w\left(\frac{x - x^t}{h}\right)$$

$$\text{où } w(u) = \begin{cases} 1 & \text{si } |u| < 1 \\ 0 & \text{autrement} \end{cases}$$

- Évite de devoir poser une origine x_0
- L'estimation n'est pas continue et comporte des marches à $x^t \pm h$

Densités par estimation naïve d'histogramme



4.2 Estimation par noyau

- Estimation par noyau : estimation plus douce que l'estimateur naïf d'histogramme
 - Utiliser un *noyau adoucissant*, typiquement un noyau gaussien

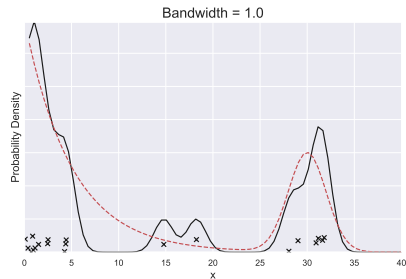
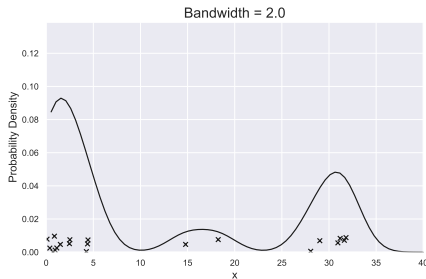
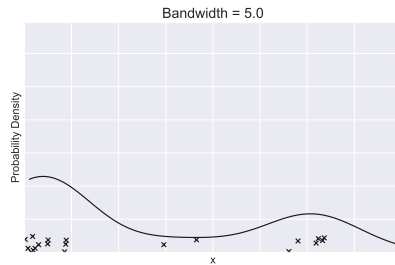
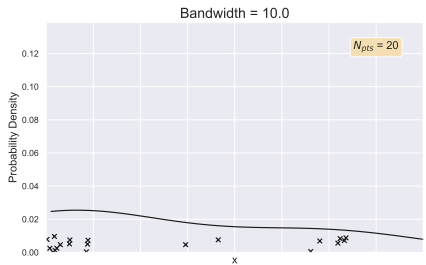
$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{u^2}{2}\right]$$

- Convolution du noyau adoucissant avec les données $\{x^t\}_{t=1}^N$

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

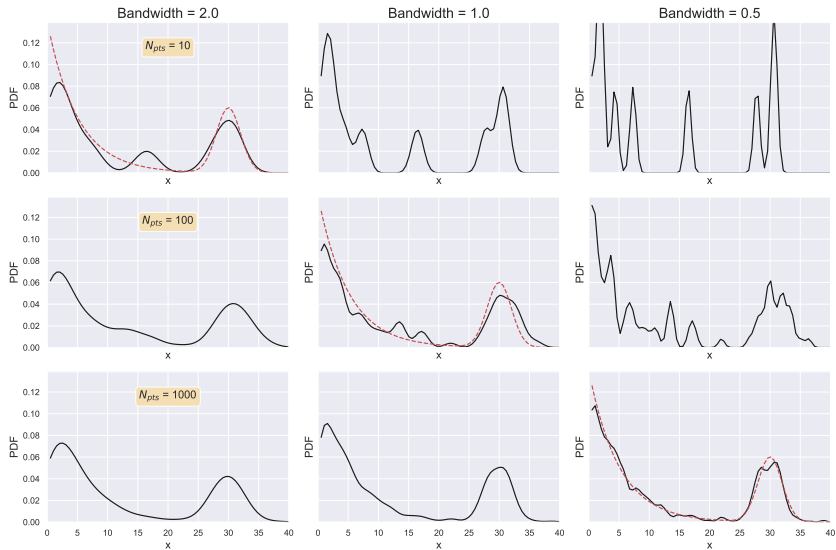
- Noyau $K(\cdot)$ détermine la forme de l'influence des données
- Largeur de fenêtre h détermine la largeur de l'influence des données
- Généralise l'estimation naïve, qui utilise une boîte rectangulaire comme noyau

Densités d'estimation par noyau



- Largeur de la fenêtre influence grandement l'estimation
 - h petit : chaque donnée a un effet local important
 - h grand : estimation plus douce, avec du recoupement entre les noyaux
- Estimation $\hat{p}(x) \rightarrow p(x)$ lorsque $N \rightarrow \infty$
 - Il faut que $h \rightarrow 0$, mais plus lentement que N (c'est-à-dire $Nh \rightarrow \infty$)
 - Typiquement, on pose que $h_N = h_1/\sqrt{N}$, utilisant une fenêtre de h_N pour un jeu de taille N

Variation du nombre d'observations



Propriétés des noyaux adoucissants

- Propriétés désirables d'un noyau adoucissant

1. Valeurs positives ou nulles

$$K(x) \geq 0, \forall x$$

2. Aire sous la courbe unitaire

$$\int_{-\infty}^{\infty} K(x) dx = 1$$

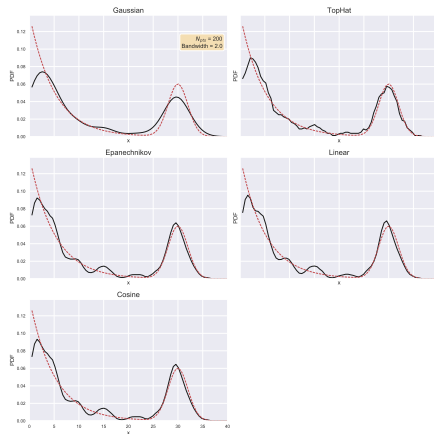
3. Centré sur l'origine

$$\int_{-\infty}^{\infty} x K(x) dx = 0$$

- Si propriétés 1 et 2 sont respectées, $K(u)$ correspond à une fonction de densité valide et donc $\hat{p}(x)$ l'est également
- De plus, si $K(u)$ est continue et différentiable, $\hat{p}(x)$ l'est également
- Support : étalement des valeurs de u pour lequel $K(u)$ est non-nul

Exemples de noyaux adoucissants

- Gaussien
 - Dérivable, mais support n'est pas borné
- Boxcar / TopHat : Estimation naive d'histogramme
 - Support borné, mais fonction non dérivable
- Epanechnikov : $K(u) = (3/4)(1 - u^2)$ pour $u \in [-1,1]$
 - Support borné, fonction non dérivable
- Linéaire / triangle : $K(u) = 1 - |u|$ pour $u \in [-1,1]$
 - Support borné, fonction non dérivable
- Cosinus : $K(u) = \cos(u\pi/2)$ pour $u \in [-1,1]$
 - Support borné, fonction non dérivable



Estimation par noyau, cas multidimensionnel

- Équation générale de l'estimation par noyau en D dimensions

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

- Contrainte sur le noyau : $\int_{\mathbb{R}^D} K(\mathbf{x}) d\mathbf{x} = 1$
- Noyau gaussien multivarié

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

- Sensible à la dimensionnalité et à la normalisation des valeurs selon les différentes dimensions
- Noyau incluant une normalisation selon l'estimation de la covariance Σ

$$K(\mathbf{u}) = \frac{1}{(2\pi)^{0,5D} |\Sigma|^{0,5}} \exp\left[-0,5\mathbf{u}^\top \Sigma^{-1} \mathbf{u}\right]$$

Estimation par noyau pour le classement

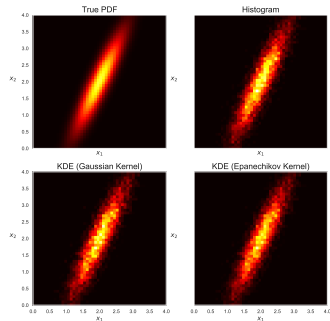
- Estimation de $\hat{p}(\mathbf{x}|C_i)$ par noyau

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

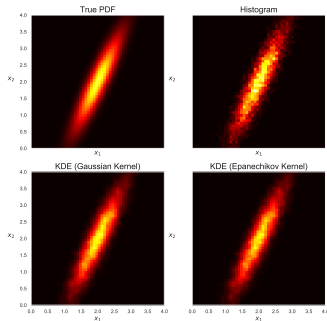
- Fonction discriminante correspondante

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

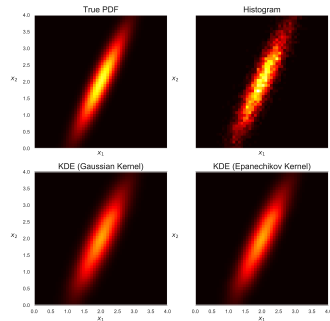
Effet de largeur du noyau pour classement



Étroit



Moyen



Large

4.3 k -plus proches voisins

- k -plus proches voisins (k -PPV)
 - Ensemble de référence $\mathcal{X} = \{x^t\}_{t=1}^N$
 - Adapter la largeur de la fenêtre selon la densité locale des données (k données les plus proches)

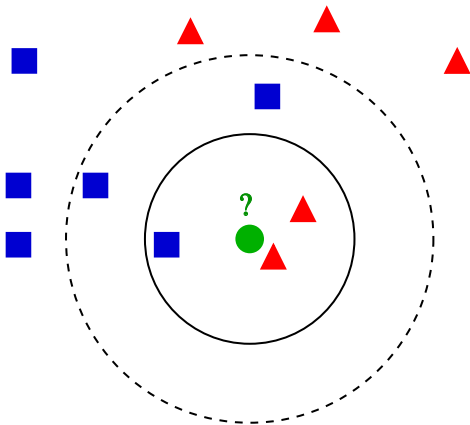
$$\hat{p}(x) = \frac{k}{2Nd_k(x, \mathcal{X})}$$

- $h = d_k(x, \mathcal{X})$: distance du k -ième voisin à la donnée x dans \mathcal{X}
- Estimateur non continu, similaire à l'estimateur naïf par histogramme, avec largeur h adaptative

- Les k -PPV définis par trois paramètres principaux
 - Nombre de voisins k
 - k faible : découpage de l'espace fin selon l'ensemble de référence
 - k élevé : découpage plus doux, moyennage selon le voisinage
 - Mesure de distance $D(\mathbf{x}, \mathbf{y})$
 - Définit la relation de voisinage entre les données
 - Ensemble de données de référence \mathcal{X}
 - Taille de l'ensemble de données
 - Densité de la répartition dans l'espace des données
 - Représentativité des données (filtrage)

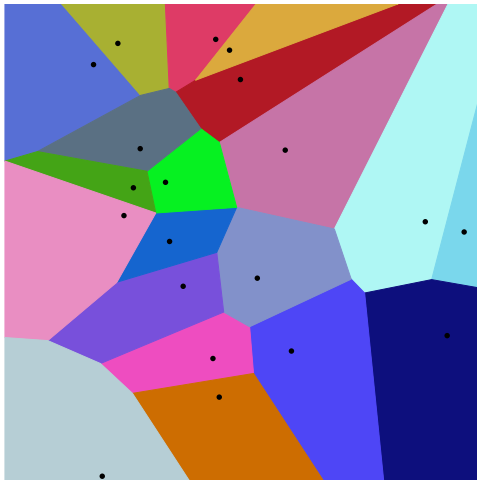
- Classement par k -plus proches voisins (k -PPV)
 - Ensemble de référence (d'entraînement) $\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$
 - Pour classer une donnée inconnue \mathbf{x} , calculer les k -plus proches voisins dans \mathcal{X} en utilisant une mesure de distance (ex. distance euclidienne)
 - Assigner à \mathbf{x} l'étiquette la plus fréquente parmi celles des k -plus proches voisins
- Méthode très simple et directe pour le classement
- Avec $k = 1$, divise l'espace d'entrée selon un diagramme de Voronoï basé sur \mathcal{X}

Classement par les k -PPV



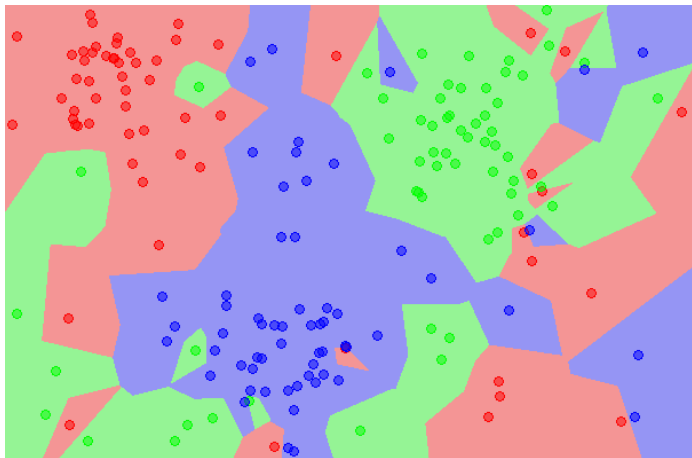
Par Antti Ajanki, CC-BY-SA 3.0, <https://en.wikipedia.org/wiki/File:KnnClassification.svg>

Diagramme de Voronoï (1-PPV)



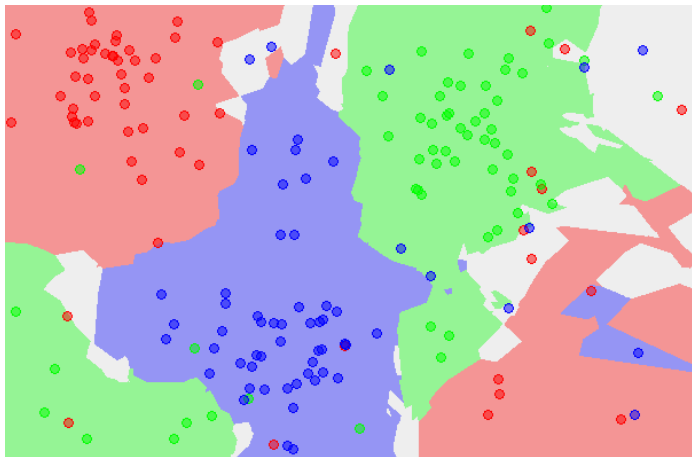
Par Balu.ertl, CC-BY-SA 4.0, https://commons.wikimedia.org/wiki/File:Euclidean_Voronoi_diagram.svg

Régions et frontière pour 1-PPV



Par Agor153, CC-BY-SA 3.0, <https://en.wikipedia.org/wiki/File:Map1NN.png>

Régions et frontière pour 5-PPV

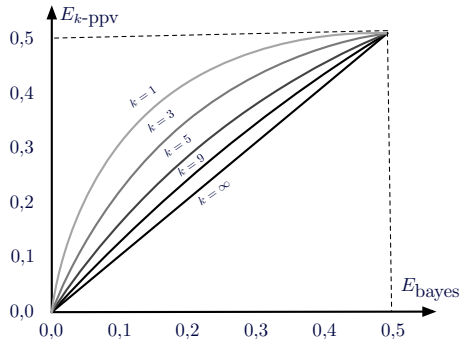


Par Agor153, CC-BY-SA 3.0, <https://en.wikipedia.org/wiki/File:Map5NN.png>

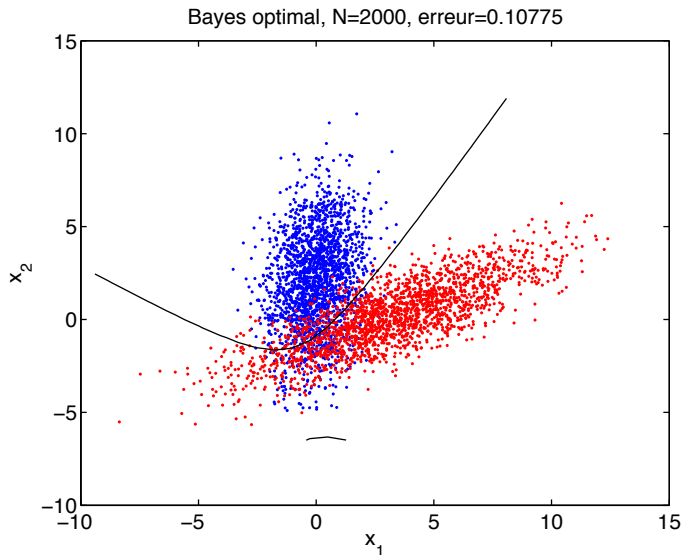
4.4 Notions sur les k -PPV

Bornes du classifieur k -PPV

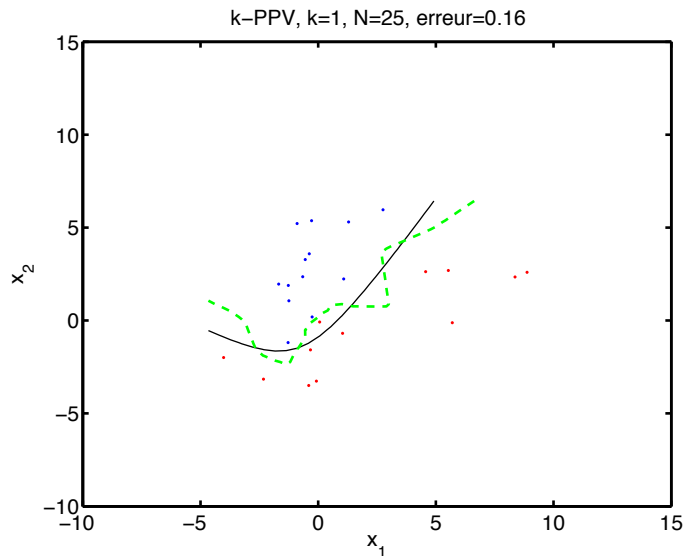
- Taux d'erreur bayésien optimal (E_{bayes})
 - Taux d'erreur lorsqu'on connaît les véritables densités de probabilité par classe
 - Optimal, impossible de faire mieux en généralisation
- Deux bornes sur le taux d'erreur du k -PPV
 - Avec $k = 1$ et $N \rightarrow \infty$ alors
$$E_{1\text{-ppv}} \leq 2E_{\text{bayes}}$$
 - Avec $k \rightarrow \infty$ et $N \rightarrow \infty$ alors
$$E_{k\text{-ppv}} \rightarrow E_{\text{bayes}}$$



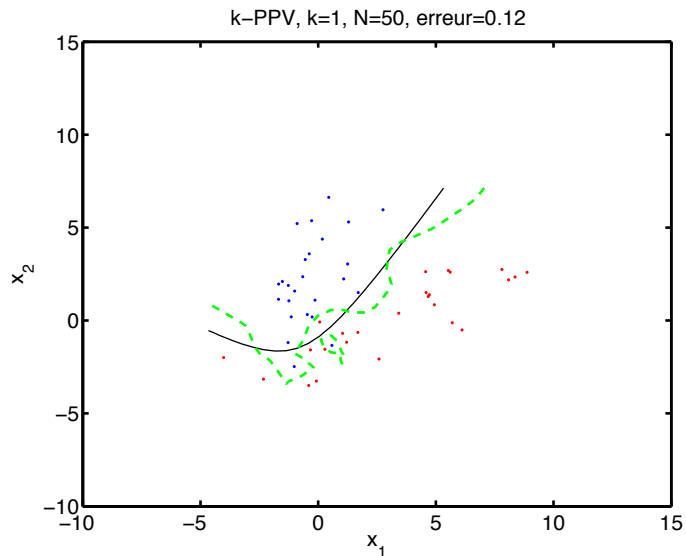
Classement bayésien optimal ($N = 2000$)



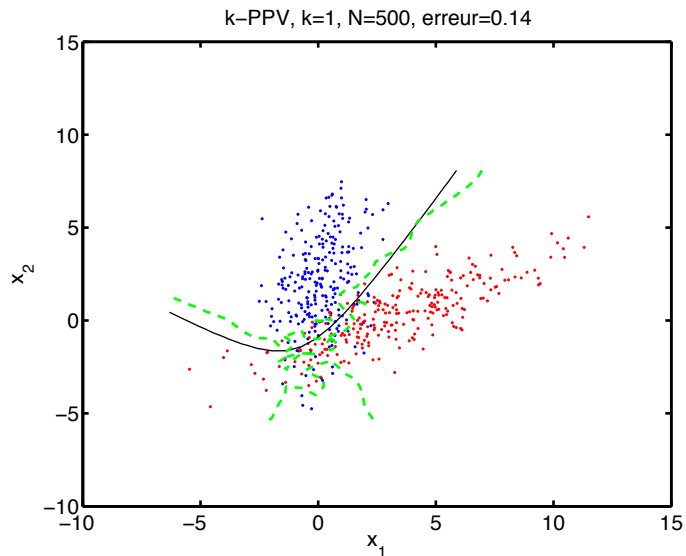
Variation du nombre d'observations, $k = 1$, $N = 25$



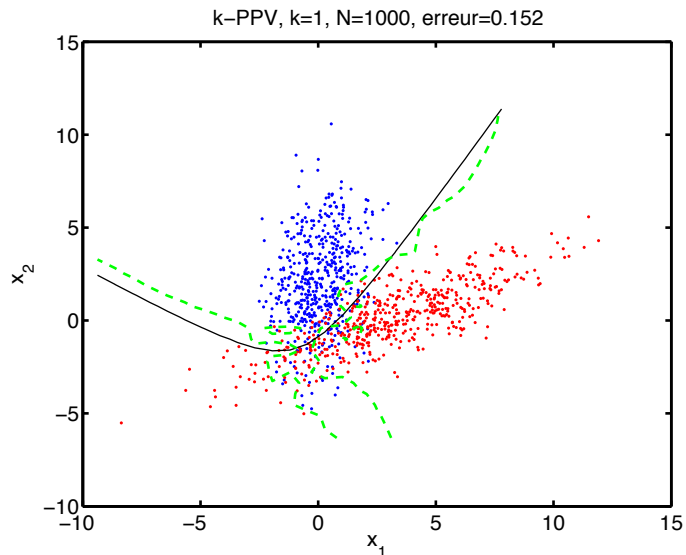
Variation du nombre d'observations, $k = 1$, $N = 50$



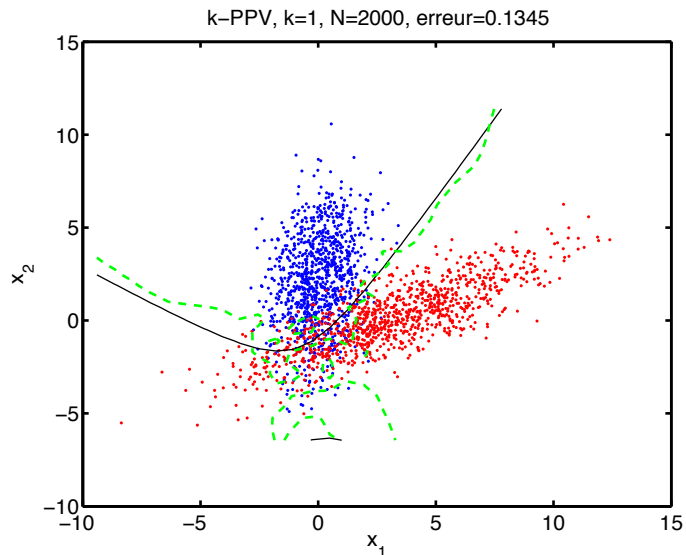
Variation du nombre d'observations, $k = 1$, $N = 500$



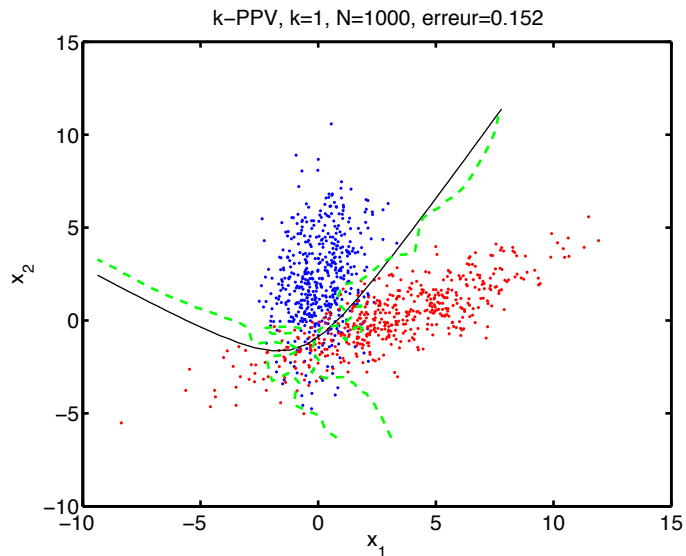
Variation du nombre d'observations, $k = 1$, $N = 1000$



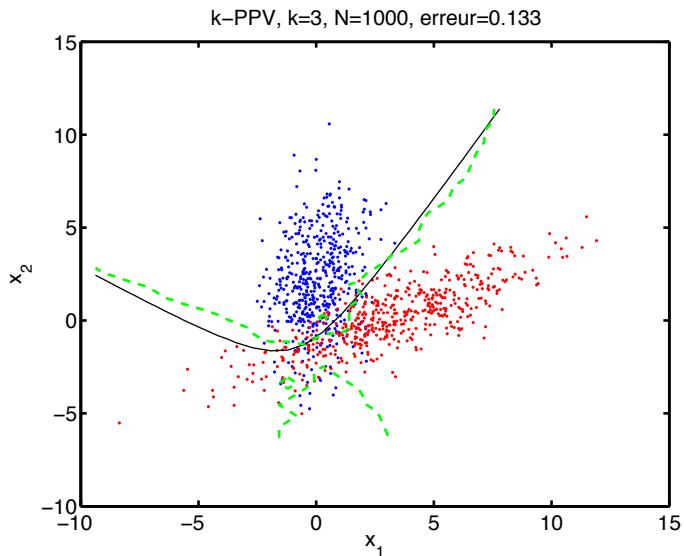
Variation du nombre d'observations, $k = 1$, $N = 2000$



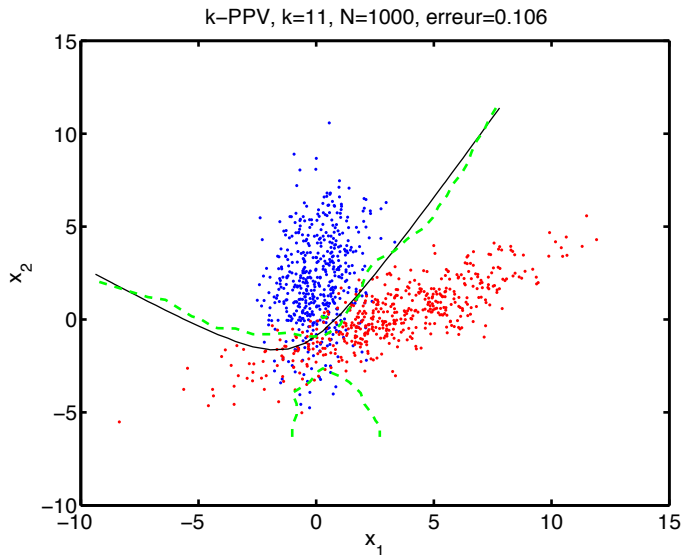
Variation du nombre de voisins, $k = 1$, $N = 1000$



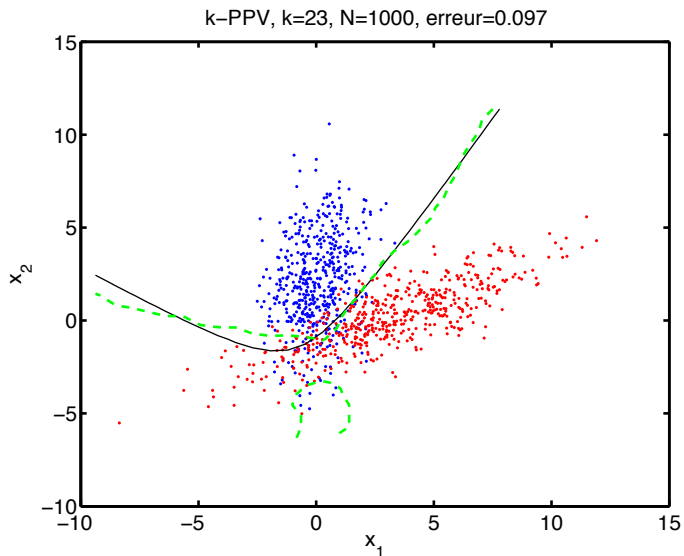
Variation du nombre de voisins, $k = 3$, $N = 1000$



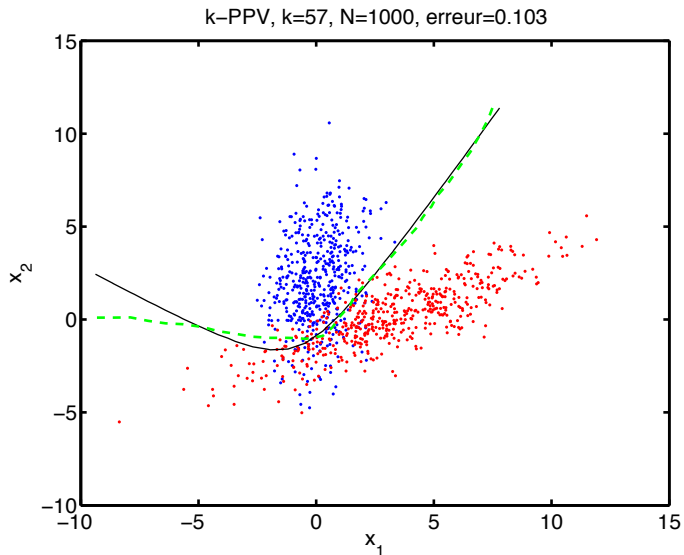
Variation du nombre de voisins, $k = 11$, $N = 1000$



Variation du nombre de voisins, $k = 23$, $N = 1000$



Variation du nombre de voisins, $k = 57$, $N = 1000$



- La mesure de distance donne la relation de voisinage entre les données
- Définition mathématique d'une métrique $D : X \times X \mapsto \mathbb{R}$
 - Non-négativité : $D(\mathbf{x}, \mathbf{y}) \geq 0$
 - Réflexivité : $D(\mathbf{x}, \mathbf{y}) = 0$ ssi $\mathbf{x} = \mathbf{y}$
 - Symétrie : $D(\mathbf{x}, \mathbf{y}) = D(\mathbf{y}, \mathbf{x})$
 - Inégalité du triangle : $D(\mathbf{x}, \mathbf{z}) \leq D(\mathbf{x}, \mathbf{y}) + D(\mathbf{y}, \mathbf{z})$
- Différentes mesures de distance possibles
 - Distance euclidienne
 - Distance de Minkowsky
 - Distance de Tanimoto (distance entre ensembles)
 - Distance tangente

Distance de Minkowsky

- Distance de Minkowsky

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

- Paramètre p contrôle l'accent sur les dimensions où la magnitude est la plus grande
- Distance de Manhattan ($p = 1$), pondération égale de toutes les dimensions :

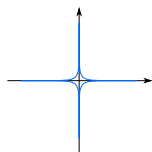
$$D_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^D |x_i - y_i|$$

- Distance D_∞ , uniquement la dimension où la différence est de magnitude maximale : $D_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^D |x_i - y_i|$

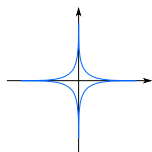
- Distance euclidienne ($p = 2$), compromise dans ces extrêmes :

$$D_2(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^D (x_i - y_i)^2}$$

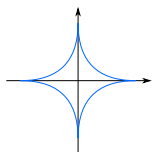
Illustration de la distance de Minkowsky



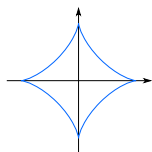
$$p = 2^{-2} \\ = 0.25$$



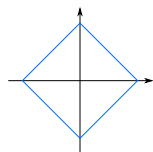
$$p = 2^{-1.5} \\ = 0.354$$



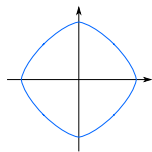
$$p = 2^{-1} \\ = 0.5$$



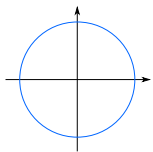
$$p = 2^{-0.5} \\ = 0.707$$



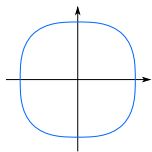
$$p = 2^0 \\ = 1$$



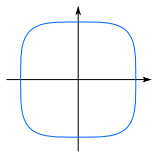
$$p = 2^{0.5} \\ = 1.414$$



$$p = 2^1 \\ = 2$$

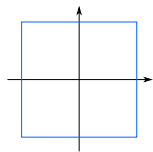


$$p = 2^{1.5} \\ = 2.828$$



$$p = 2^2 \\ = 4$$

...



$$p = 2^\infty \\ = \infty$$

Par Waldir, CC-BY-SA 3.0, https://commons.wikimedia.org/wiki/File:2D_unit_balls.svg

- Mesure de distance sensible à l'échelle des données selon les différentes dimensions
 - Valeurs selon une dimension où l'échelle est grande relativement aux autres dimensions vont « absorber » la valeur selon ces dimensions

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

- Normalisation des données nécessaire si les échelles sont différentes selon les dimensions
 - Normalisation selon le sens des données (unités physiques)
 - Normalisation selon valeur max-min de chaque dimension
 - Transformation blanchissante

Évaluation des performances leave-one-out

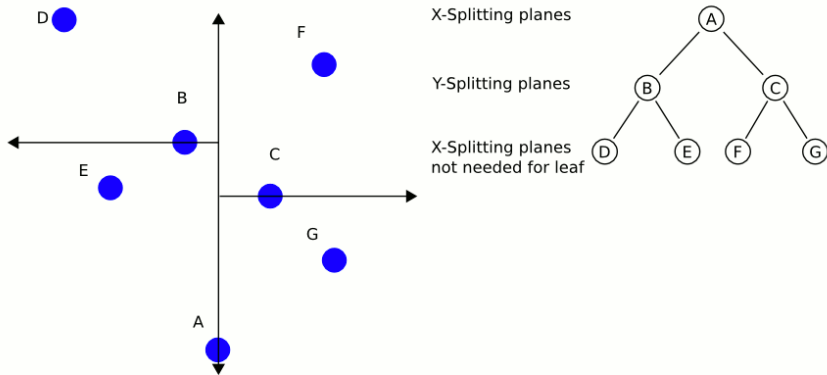
- Pas d'entraînement nécessaire avec k -PPV
 - Entraînement correspondant à stocker le jeu de données \mathcal{X}
 - Évaluation des performances de type *leave-one-out*
 - Tirer avantage du coût d'entraînement nul
 - Correspond à la validation croisée à K plis, avec $K = N$
1. Pour chaque donnée $\mathbf{x}^t \in \mathcal{X}$:
 - 1.1 Calculer les k -PPV de \mathbf{x}^t parmi l'ensemble $\mathcal{X} \setminus \{\mathbf{x}^t\}$
 - 1.2 Déterminer l'étiquette majoritaire de ces k plus proches voisins comme étiquette de classement de \mathbf{x}^t
 2. Retourner comme taux d'erreur le ratio entre le nombre de données mal classées dans \mathcal{X} et la taille de \mathcal{X}

4.5 Efficacité computationnelle de k -PPV

- Entraînement : stockage des données en mémoire
 - Complexité en temps et en mémoire : $O(N)$
- Traitement des données (test/validation) : obtenir les k voisins
 - Obtenir les k plus proches voisins de \mathbf{x} dans \mathcal{X} : complexité en temps $O(N \log N)$ (algorithme naïf)
 - Classer M données : complexité en temps $O(MN \log N)$
- Optimisation des calculs possibles par méthodes plus sophistiquées

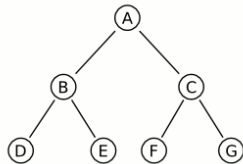
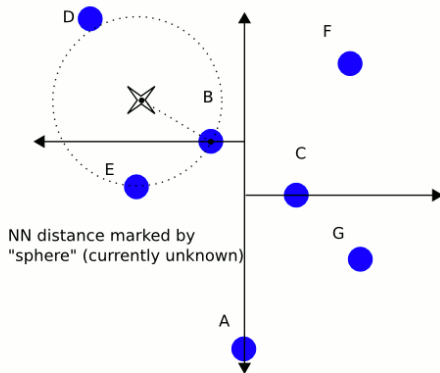
- Structure/topologie des données dans l'espace peut être exploitée pour la recherche des k -PPV
 - Éviter de calculer la distance avec certaines données, qui sont de toute façon trop loin de la donnée à tester
- KD-tree : structure de données en arbre capturant la topologie des données dans un espace euclidien
 - Construction du KD-Tree pour N données : $O(N \log N)$
 - Espace de stockage nécessaire du KD-Tree : $O(N)$
 - Requête des k -PPV d'une donnée dans un KD-Tree
 - $O(N^{\frac{D-1}{D}} + k)$ en D dimensions
 - $O(\sqrt{N} + k)$ en 2D
 - $O(\log N)$ avec $k = 1$
 - Traitement de M données : $O(M(N^{\frac{D-1}{D}} + k))$
- Implémentations efficaces de KD-tree disponibles (ex. CGAL en C++, `scipy.spatial.KDTree` en Python)

Illustration du KD-tree



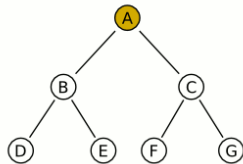
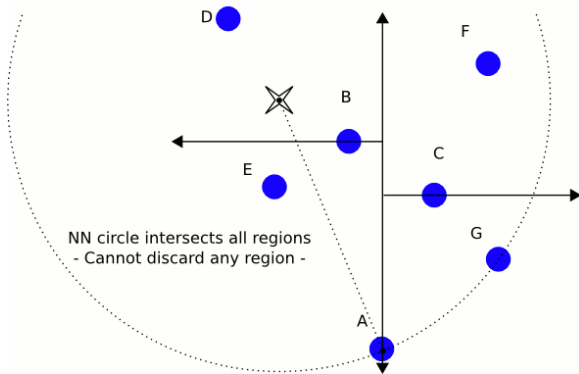
Par User.A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

Illustration du KD-tree



Par User.A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

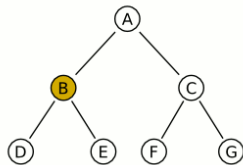
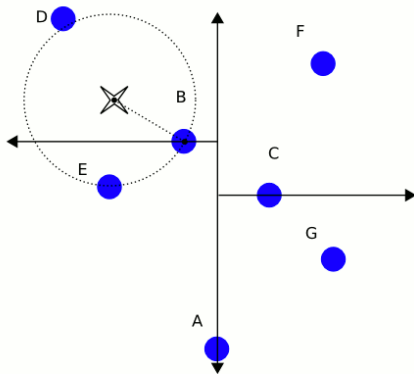
Illustration du KD-tree



Start at A, then proceed in depth-first search (maintain a stack of parent-nodes if using a singly-linked tree). Set best estimate to A's distance. Then examine left child node

Par User_A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

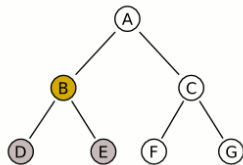
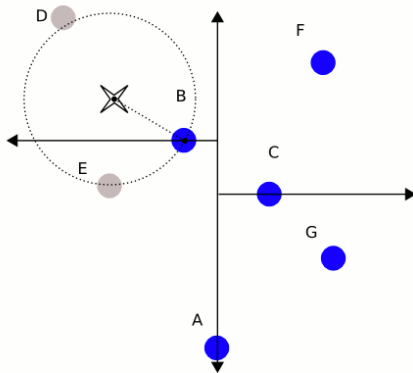
Illustration du KD-tree



Calculate B's distance and
compare against best estimate
- It is smaller distance, so update
best estimate. Examine children (left then right)

Par User.A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

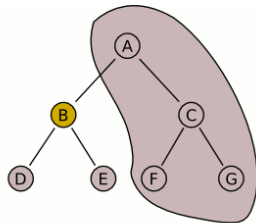
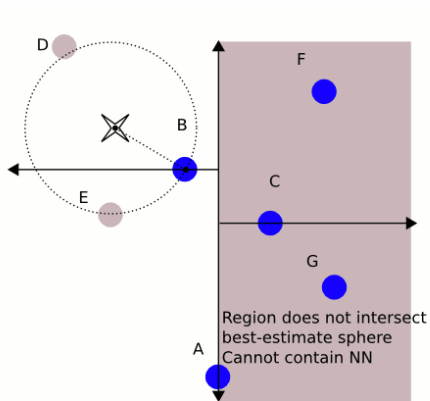
Illustration du KD-tree



D & E Discarded as B
(already visited) is closer.
B is the best estimate for B's sub-branch
Proceed back to parent node

Par User.A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

Illustration du KD-tree



A's children have all been searched,
B is the best estimate for entire tree

Par User.A1, CC-BY-SA 3.0, <https://commons.wikimedia.org/wiki/File:KDTree-animation.gif>

4.6 Sélection de prototypes

Taille de la banque d'entraînement

- Compromis à faire sur la taille de la banque d'entraînement
 - Avec $N \rightarrow \infty$, algorithme tend vers des performances optimales
 - Mais avec $N \rightarrow \infty$, temps de traitement et besoins en stockage faramineux
- Selon la position, la densité des données peut varier
 - Loin des frontières de décision, la densité des points peut être réduite
 - Données aberrantes ou bruitées dans une région de classe différente pourraient être retirées
- Approximation des frontières de décision par la sélection de quelques *prototypes* représentatifs des données

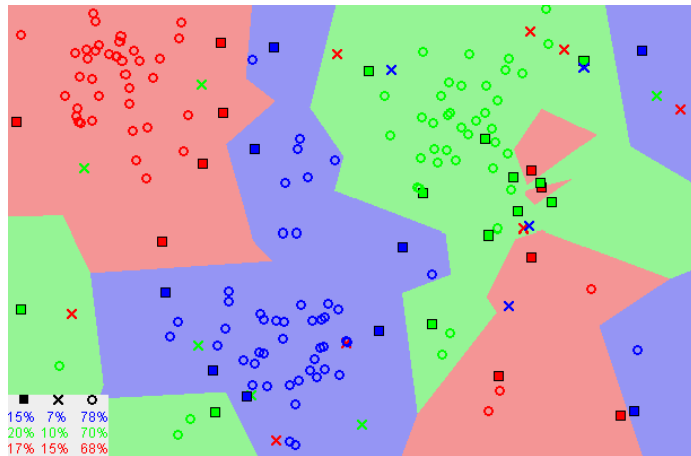
- Condensation de Hart
 - Objectif : sélectionner seulement des données de \mathcal{X} contribuant au classement
 - Heuristique faisant une construction incrémentale de l'ensemble de prototypes
- Approche suivie
 - Démarrer avec un ensemble quasi vide de prototypes (une donnée choisie au hasard)
 - Ajouter les données seulement si elles sont mal classées selon le PPV
 - Répéter tant qu'il y a des données non sélectionnées mal classées

1. Créer un ensemble de prototypes sélectionnés à partir d'une donnée \mathbf{x} choisie aléatoirement dans \mathcal{X} , $\mathcal{Z} = \{\mathbf{x}\}$
2. Tant que l'ensemble \mathcal{Z} est modifié relativement à l'itération précédente :
 - 2.1 Pour chaque donnée $\mathbf{x}^t \in \mathcal{X}$, traitée dans un ordre aléatoire :
 - 2.1.1 Déterminer le plus proche voisin de \mathbf{x}^t dans \mathcal{Z}
$$\mathbf{x}' = \operatorname{argmin}_{\mathbf{x} \in \mathcal{Z}} \|\mathbf{x}^t - \mathbf{x}\|$$
 - 2.1.2 Si l'étiquette de classe de \mathbf{x}' ne correspond pas à celle de \mathbf{x}^t ($r' \neq r^t$), alors sélectionner la donnée comme prototype, $\mathcal{Z} = \mathcal{Z} + \{\mathbf{x}^t\}$
3. Retourner l'ensemble \mathcal{Z} comme les prototypes sélectionnés dans \mathcal{X}

- Édition de Wilson
 - Heuristique pour retirer les données mal classées de \mathcal{X} selon *leave-one-out*
 - Permet d'éliminer les données que l'on croit aberrantes ou bruitées
1. Créer l'ensemble de prototypes \mathcal{Z} à partir de toutes les données, $\mathcal{Z} = \mathcal{X}$
 2. Pour chaque donnée $\mathbf{x}^t \in \mathcal{Z}$, traitée dans un ordre aléatoire :
 - 2.1 Déterminer \mathcal{V} , soit les k -PPV de \mathbf{x}^t dans $\mathcal{Z} \setminus \{\mathbf{x}^t\}$
 - 2.2 Établir l'étiquette de classement $r_{\mathcal{V}}^t$ de \mathbf{x}^t selon l'étiquette majoritaire des données dans \mathcal{V}
 - 2.3 Si l'étiquette $r_{\mathcal{V}}^t$ est différent de l'étiquette r^t de \mathbf{x}^t , alors retirer la donnée de \mathcal{Z} ,
 $\mathcal{Z} = \mathcal{Z} \setminus \{\mathbf{x}^t\}$
 3. Retourner l'ensemble \mathcal{Z} comme les prototypes sélectionnés dans \mathcal{X}

- Sélection agressive de prototypes : édition de Wilson suivie d'une condensation de Hart
 - Filtrer \mathcal{X} en éliminant d'abord les données aberrantes ou bruitées (édition de Wilson)
 - Sélectionner seulement les données contribuant au classement (condensation de Hart)
- Construction de prototypes
 - Déterminer des prototypes qui ne sont pas des données dans \mathcal{X}
 - Approche possible (non supervisé) : K -means de \mathcal{X} avec valeur de K élevée

Illustration de Wilson + Hart



Par Agor153, CC-BY-SA 3.0, <https://en.wikipedia.org/wiki/File:Map1NNReducedDataSet.png>

× : données retirée par Wilson ($k = 3$) □ : prototypes sélectionné par Hart; ○ : données retirées par Hart

4.7 Méthodes non paramétriques dans scikit-learn

- `neighbors.KernelDensity` : estimation par noyau de densités
 - Paramètres
 - `bandwidth` (float) : largeur du noyau
 - `algorithm` (string) : algorithme de voisinage à utiliser, peut être `'kd_tree'`, `'ball_tree'` ou `'auto'` (défaut : `'auto'`)
 - `kernel` (string) : noyau utilisé, peut être `'gaussian'`, `'tophat'`, `'epanechnikov'`, `'exponential'`, `'linear'` ou `'cosine'` (défaut : `'gaussian'`)
 - Méthodes
 - `fit(X)` : apprendre la densité sur les données
 - `sample(n_samples=1)` : génère des échantillons de la distribution (seulement pour noyaux gaussien et tophat)
 - `score(X)` : retourne la log-vraisemblance des données
 - `score_samples(X)` : retourne la densité des données

Scikit-learn : k -plus proches voisins

- `neighbors.KNeighborsClassifier` : classement par les k -plus proches voisins
 - Paramètres
 - `n_neighbors` (int) : nombre de voisins utilisés (défaut : 5)
 - `algorithm` (string) : algorithme de voisinage à utiliser, peut être `'kd_tree'`, `'ball_tree'`, `'brute'` ou `'auto'` (défaut : `'auto'`)
 - `metric` (string ou objet `neighbors.DistanceMetric`) : métrique de distance utilisée. Par défaut `'minkowski'` avec $p = 2$, qui revient à une distance euclidienne. Pour autres métriques, voir documentation de `neighbors.DistanceMetric`.
 - `p` (int) : valeur de p pour la distance de Minkowski (défaut : 2)
 - Méthodes
 - `fit(X,y)` : apprendre modèle de classement sur les données
 - `kneighbors(X, n_neighbors)` : retourne les k -plus proches voisins aux données
 - `predict(X)` : fait le classement des données
- `neighbors.KNeighborsRegressor` : régression par les k -plus proches voisins