Clustering scalable et distribué

Beck Gaël, Azzag Hanane, Lebbah Mustapha

1. La montée de gradient
2. Le Locality Sensitive Hashing
3. La méthode de clustering
4. Résultats expérimentaux
5. Conclusion et perspectives

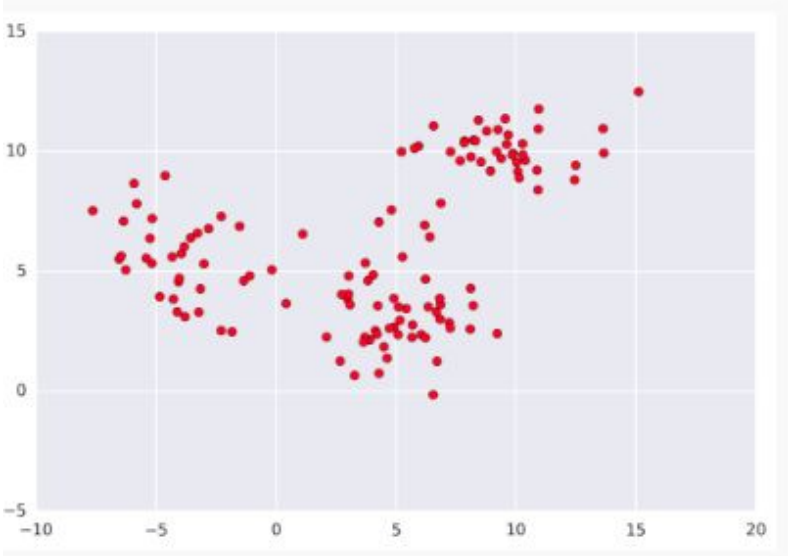
Propriétés

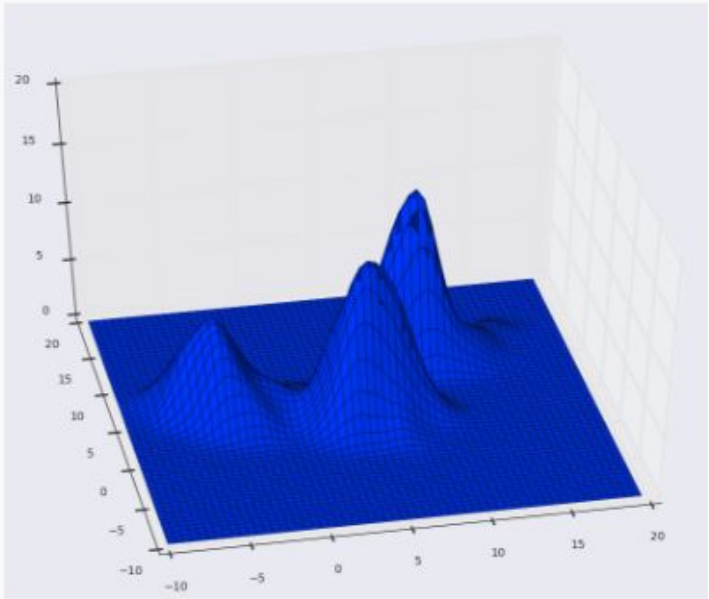
Détection automatique du nombre de clusters

La topologie des clusters peut être aléatoire

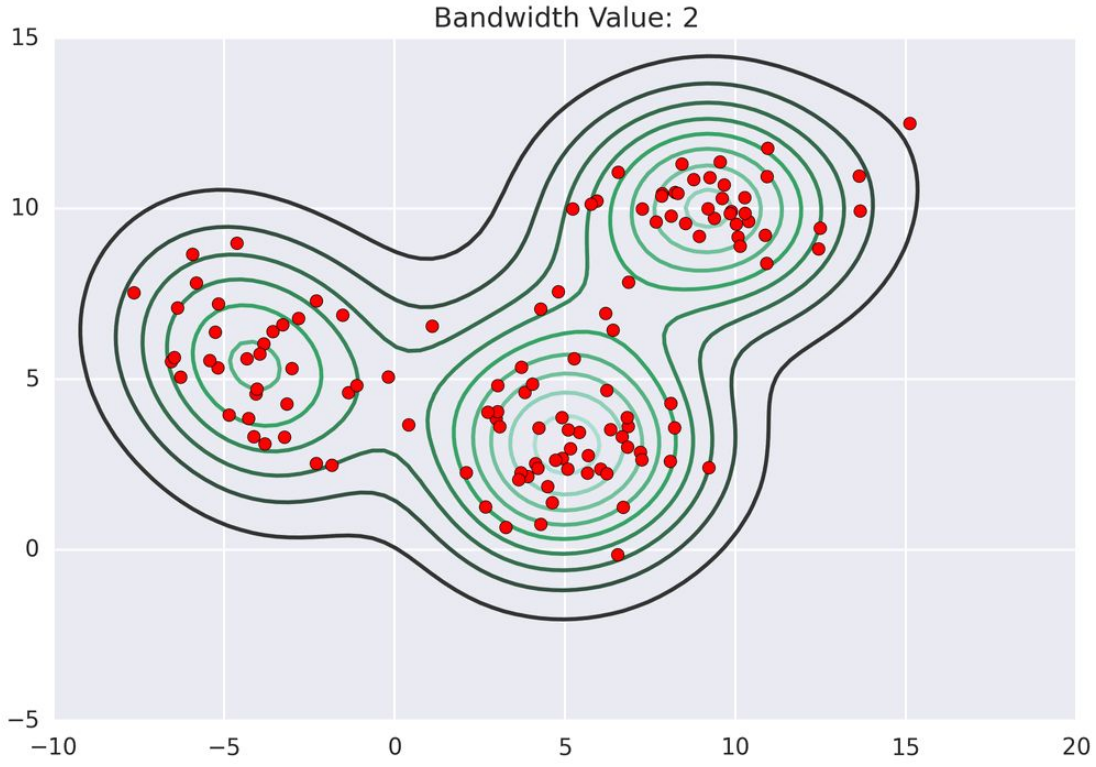
Complexité $O(n^2)$

Mean Shift à noyau





Convergence d’un jeu de données

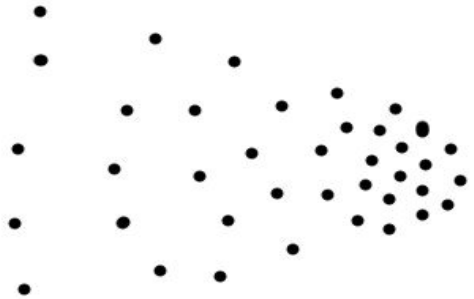


Montée de gradient via les K plus proches voisins

Complexité

O(j.n.log(n)) par point

O(j.n².log(n))



La montée de gradient

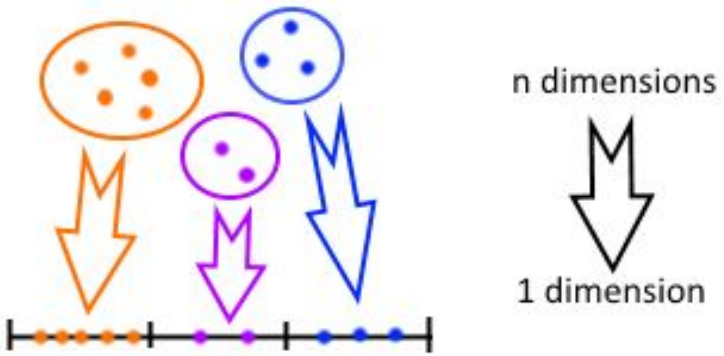
Algorithm 1 NNGA – Nearest Neighbor Gradient Ascent with exact k-nn

Input: $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}, \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, k, \varepsilon_1, j_{\max}, s_{\min}$
Output: $\{\mathrm{c}(\boldsymbol{x}_1), \ldots, \mathrm{c}(\boldsymbol{x}_m)\}$

1: **for** $\ell := 1$ to m **do**
2: $\quad j := 0$; $\boldsymbol{x}_{\ell,0} := \boldsymbol{x}_\ell$;
3: $\quad \boldsymbol{x}_{\ell,1} :=$ mean of k-nn$(\boldsymbol{x}_{\ell,0})$;
4: $\quad$ **while** $\|\boldsymbol{x}_{\ell,j+1}, \boldsymbol{x}_{\ell,j}\| > \varepsilon_1$ **or** $j < j_{\max}$ **do**
5: $\quad\quad j := j + 1$; $\boldsymbol{x}_{\ell,j+1} :=$ mean of k-nn$(\boldsymbol{x}_{\ell,j})$;
6: $\quad \boldsymbol{x}^*_\ell := \boldsymbol{x}_{\ell,j}$;

Le Locality Sensitive Hashing

- Problématique du calcul des plus proches voisins
  - LSH fournit une façon probabiliste d'obtenir les plus proches voisins
  - On transforme les vecteurs du jeu de données en scalaire via une fonction de hashage :
  - $u = \frac{x*v+b}{w}$
    - x : vecteur du jeu de données
    - v : vecteur dont les composantes sont pris sur $N(0,1)$
    - b : constante d'uniformisation pris sur $U(0,w)$
    - w : constante défini par l'utilisateur
  - Les vecteurs sont projetés sur un segment qu'on découpe en blocs
  - On cherche les plus proche voisin d'un vecteur requête exclusivement dans le bloc dans lequel il tombe
  - Chaque bloc est traité de manière indépendante par un exécuteur Spark



Montée de gradient approximative

Algorithm 2 NNLSHGA – Nearest Neighbors Gradient Ascent Approximate k-NN with LSH

Input: $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}, \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, k, M_1$

Output: $\{k\text{-}\widetilde{\text{nn}}(\boldsymbol{x}_1), \ldots, k\text{-}\widetilde{\text{nn}}(\boldsymbol{x}_m)\}$

/* Create hash table with M_1 buckets */

1: **for** $i := 1$ to n **do** $H_i := H(\boldsymbol{X}_i)$;

/* Search for approx nn in adjacent buckets */

2: **for** $\ell := 1$ to m **do**

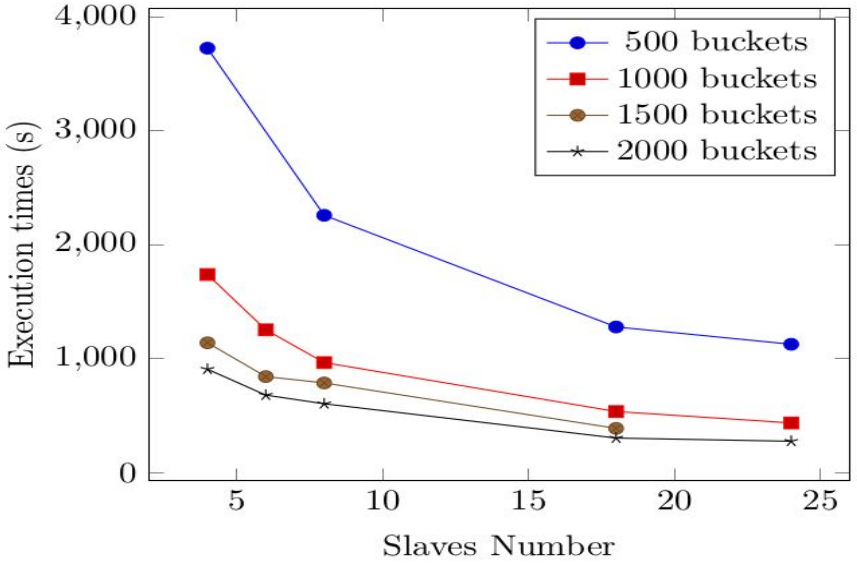
3: $\quad R(\boldsymbol{x}_\ell) := \{\boldsymbol{X}_i : H_i = H(\boldsymbol{x}_\ell), i \in \{1, \ldots, n\}\}$

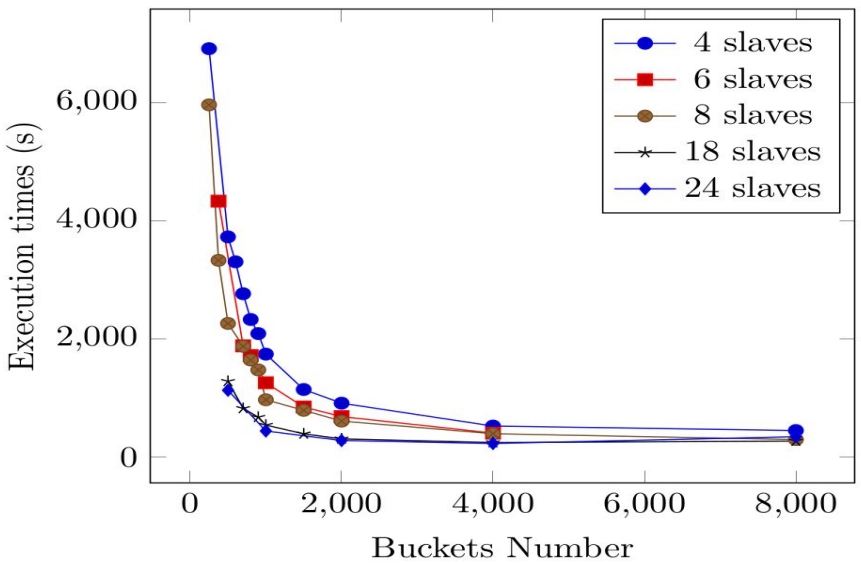
4: $\quad$ **while** $\text{card}(R(\boldsymbol{x}_\ell)) < k$ **do**

5: $\quad\quad R(\boldsymbol{x}_\ell) := R(\boldsymbol{x}_\ell) \cup$ adjacent bucket;

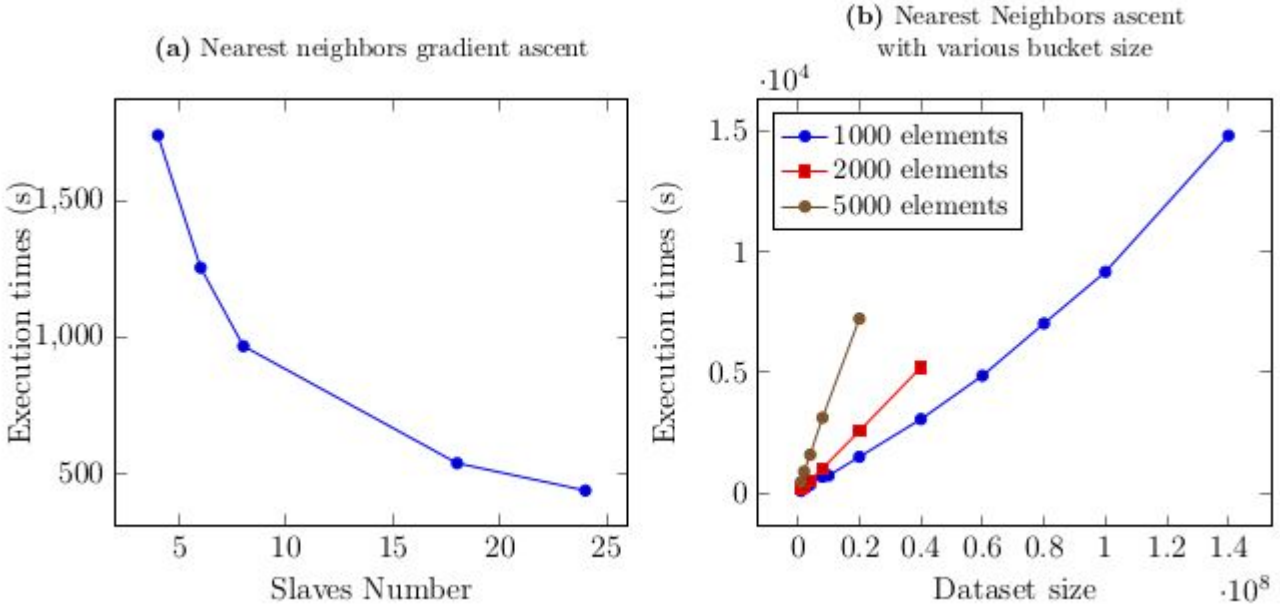
6: $\quad k\text{-}\widetilde{\text{nn}}(\boldsymbol{x}_\ell) := k$-nn from $R(\boldsymbol{x}_\ell)$ to $\boldsymbol{x}_\ell$;

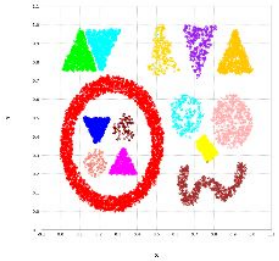
Scalabilité du Mean Shift



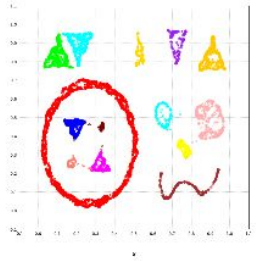


Scalabilité du Mean Shift

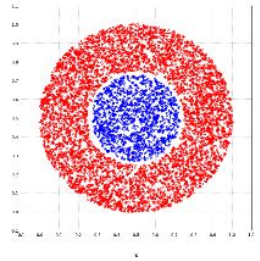




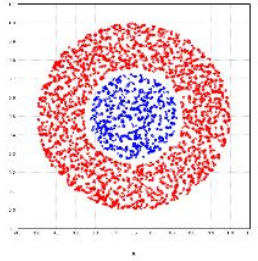
(a) DS1



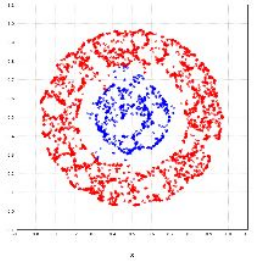
(b) DS1 [$k_1 = 50$]



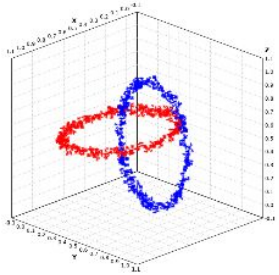
(c) Disk6000



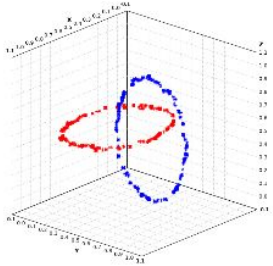
(d) Disk6000 [$k_1 = 5$]



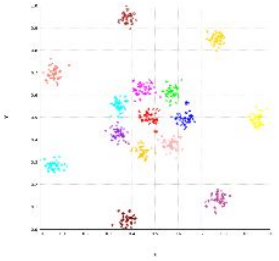
(e) Disk6000 [$k_1 = 50$]



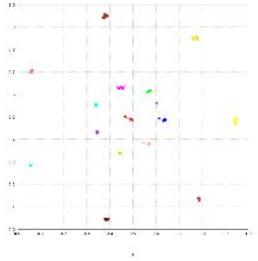
(f) Chainlink



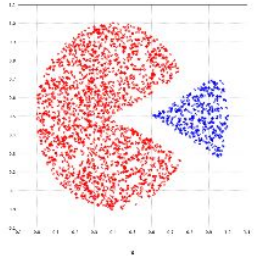
(g) Chainlink [$k_1 = 10$]



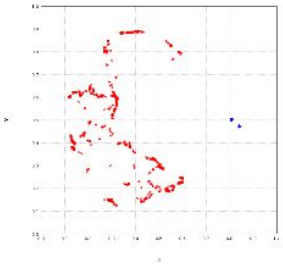
(h) R15



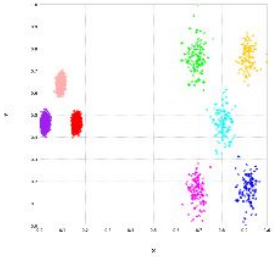
(i) R15 [$k_1 = 20$]



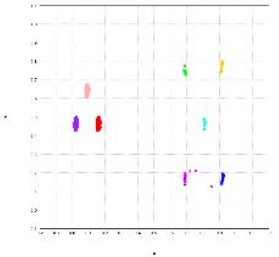
(j) Pie



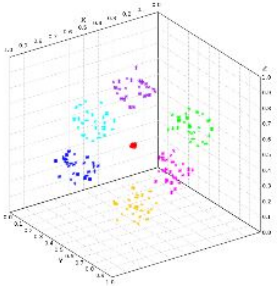
(k) Pie [$k_1 = 200$]



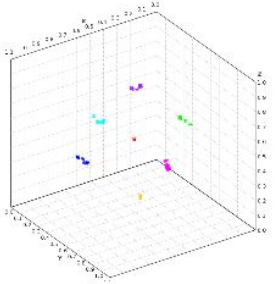
(l) Unbalance



(m) Unbalance [$k_1 =$



(n) Hepta



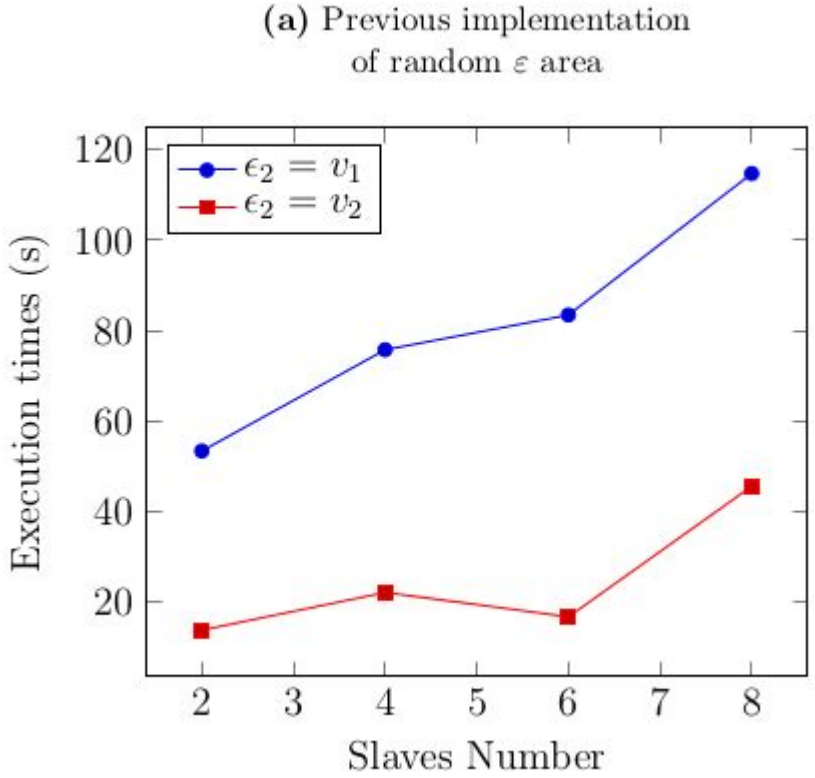
(o) Hepta [$k_1 = 20$]

Clustering par zones aléatoires

Algorithm 3 NNMS – Labeling step

/* Create clusters by merging near final iterates */
1: **for** $\ell_1, \ell_2 := 1$ to m **do**
2: **if** $\|\boldsymbol{x}^*_{\ell_1} - \boldsymbol{x}^*_{\ell_2}\| \leq \varepsilon_2$ **then** $\mathrm{c}(\boldsymbol{x}^*_{\ell_1}) := \mathrm{c}(\boldsymbol{x}^*_{\ell_2})$;
/* Merge small clusters */
3: $C^* :=$ cluster with minimum cardinality;
4: **while** $\mathrm{card}(C^*) < c_{\min}$ **do**
5: $C' :=$ nearest other cluster to C^*;
6: **for** $\boldsymbol{x}^*_\ell \in C^*$ **do** $\mathrm{c}(\boldsymbol{x}^*_\ell) := \mathrm{c}(C')$;
7: $C^* :=$ cluster with minimum cardinality;
8: **for** $\ell := 1$ to m **do** $\mathrm{c}(\boldsymbol{x}_\ell) := \mathrm{c}(\boldsymbol{x}^*_\ell)$;

Problème de l'approche



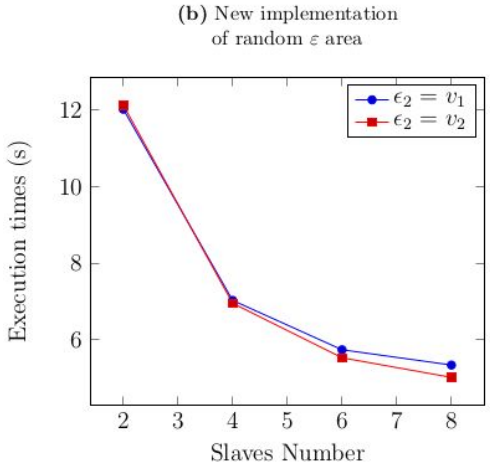
Clustering par zone aléatoire approximé

Algorithm 4 LLSH – Approximate cluster labeling with LSH

Input: $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}, M_2, \varepsilon_2, \varepsilon_3, c_{min}$
Output: $\{\tilde{\mathrm{c}}(\boldsymbol{x}_1), \ldots, \tilde{\mathrm{c}}(\boldsymbol{x}_m)\}$
/* Create hash table with M_2 buckets */
1: **for** $i := 1$ to m **do** $H_i := H(\boldsymbol{x}_i)$;
/* Create initial cluster labels */
2: **for** $\ell := 1$ to m **do**
3: $R(\boldsymbol{x}_\ell) := \{\boldsymbol{x}_i : H_i = H(\boldsymbol{x}_\ell), i \in \{1, \ldots, m\}\}$
4: **for** $\ell_1, \ell_2 := 1$ to $\mathrm{card}(R(\boldsymbol{x}_\ell))$ **do**
5: **if** $\|\boldsymbol{x}_{\ell_1} - \boldsymbol{x}_{\ell_2}\| \leq \varepsilon_2$ **then** $\tilde{\mathrm{c}}(\boldsymbol{x}_{\ell_1}) := \tilde{\mathrm{c}}(\boldsymbol{x}_{\ell_2})$;
/* Compute cluster centroids */
6: **for** $\ell := 1$ to L **do** $\mathrm{cen}_\ell :=$ cluster centroid
/* Merge close clusters */
7: **for** $\ell_1, \ell_2 := 1$ to L **do**
8: **if** $\|\mathrm{cen}_{\ell_1} - \mathrm{cen}_{\ell_2}\| \leq \varepsilon_3$ **then** assign same cluster label $\tilde{\mathrm{c}}(\boldsymbol{x}_\ell)$ for all $\boldsymbol{x}_\ell$ associated with cen_{ℓ_1} or cen_{ℓ_2};
9: **while** it exist $card(cluster_{l_i}) \leq c_{min}$ **do**
10: assign same cluster label than closest cluster

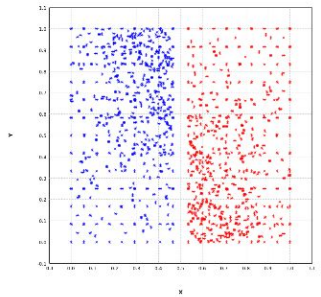
Scalabilité de la méthode de clustering

Clustering par zones aléatoires approximées

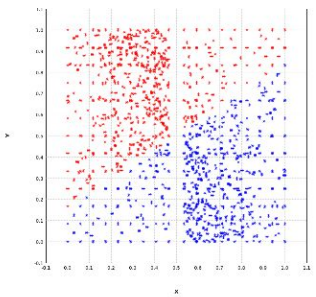


Effets de la montée de gradient sur le clustering

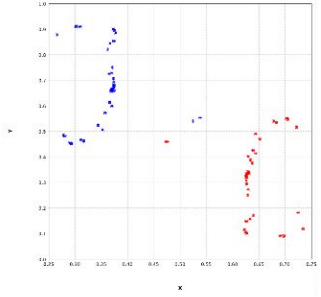
K-Means



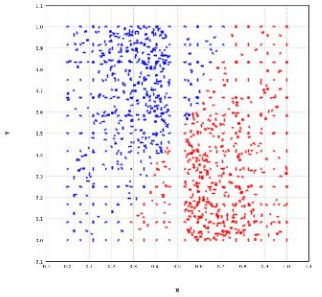
(a) Original WingNut



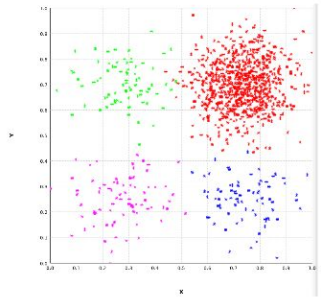
(b) WingNut $k = 2$



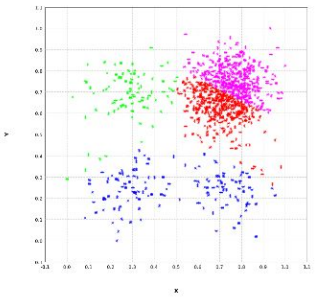
(c) WingNut $k = 2$ $[k_1 = 100]$



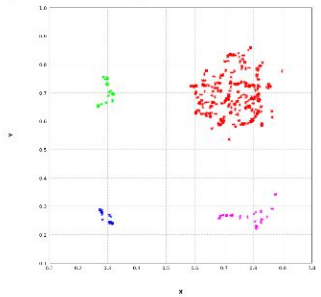
(d) WingNut $k = 2$ $[k_1 = 100, \text{o}]$



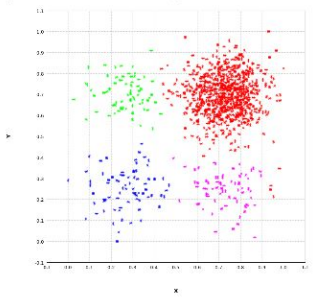
(e) Original Sizes5



(f) Sizes5 $k = 4$

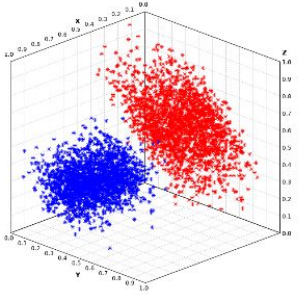


(g) Sizes5 $k = 4$ $[k_1 = 20]$

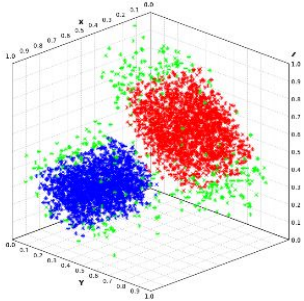


(h) Sizes5 $k = 4$ $[k_1 = 20, \text{o}]$

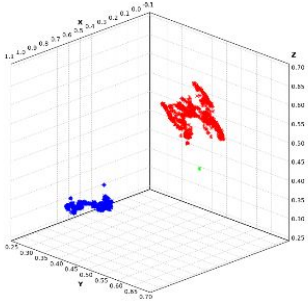
DBSCAN



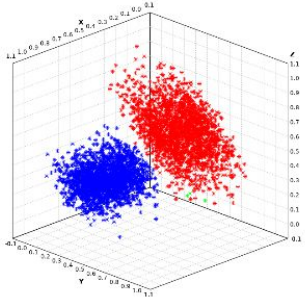
(a) Original EngyTime



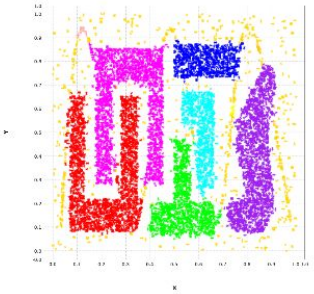
(b) EngyTime



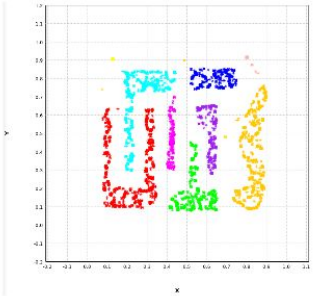
(c) EngyTime [$k_1 = 200$]



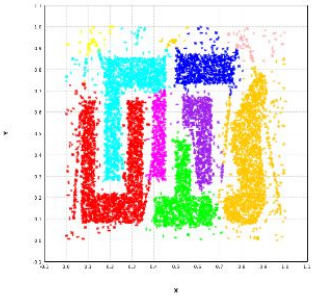
(d) EngyTime [$k_1 = 200$, o]



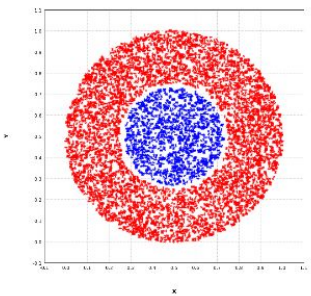
(e) t48k



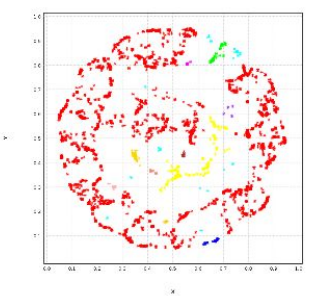
(f) t48k [$k_1 = 30$]



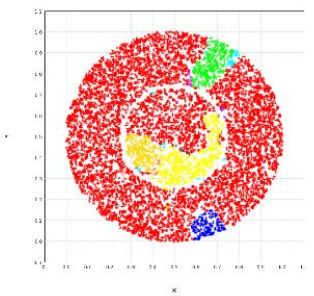
(g) t48k [$k_1 = 30$, o]



(h) Disk6000

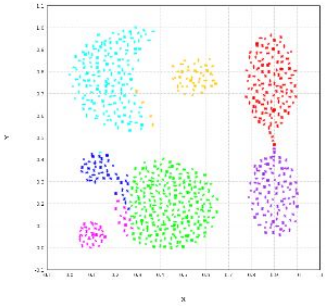


(i) Disk6000 [$k_1 = 100$]

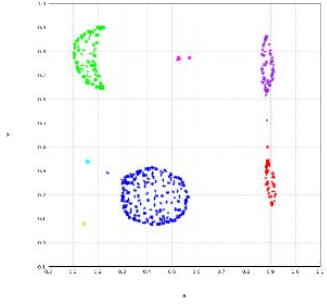


(j) Disk6000 [$k_1 = 100$, o]

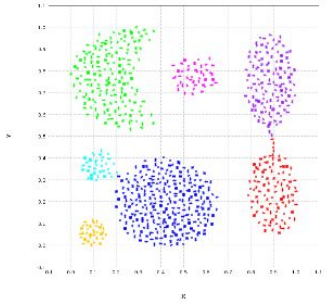
Clustering par zones aléatoires



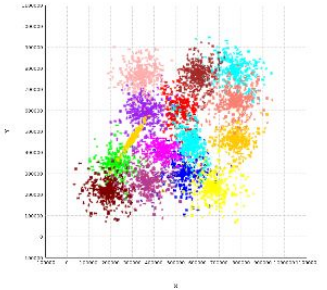
(a) Aggregation



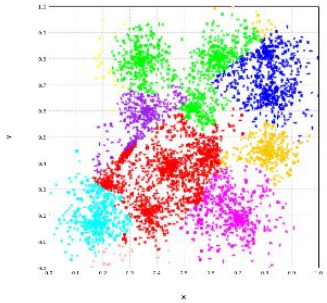
(b) Aggregation $[k_1 = 50]$



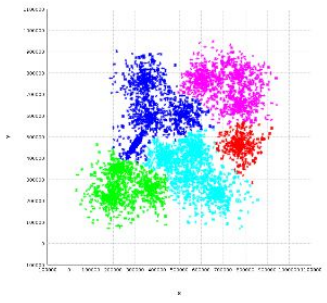
(c) Aggregation $[k_1 = 50, \text{o}]$



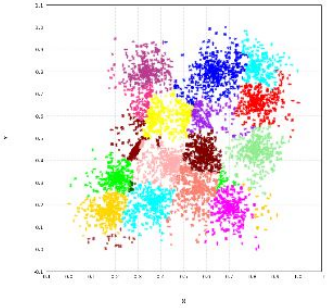
(d) S3 original



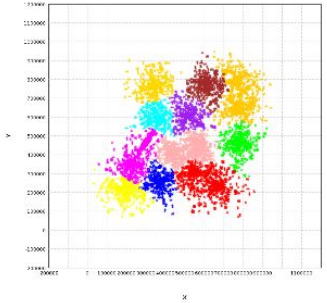
(e) S3 [s1]



(f) S3 $[k_1 = 40, \text{o}, \text{s1}]$



(g) S3 [s2]



(h) S3 $[k_1 = 40, \text{o}, \text{s2}]$

Conclusion et perspectives

Augmenter la valeur de K

Etudier l’impact sur des problèmes de régression

Améliorer la qualité avec le nombre de blocs

Une montée de gradient et un clustering scalable et distribué

Clustering4Ever

Une API Open Source de clustering distribué

https://github.com/Clustering4Ever/Clustering4Ever

Merci de votre attention