Géométrie Algorithmique
Données, Modèles, Programmes

7. Structures de données géométriques

Jean-Daniel Boissonnat

Collège de France
24 mai 2017

Géométrie algorithmique

Données, modèles, programmes

1. Modèles géométriques discrets
   F. Cazals : Modèles géométriques pour la prédiction des interactions macro-moléculaires
2. La puissance de l'aléa : algorithmes randomisés
   P. Calka : Probabilités géométriques
3. Le calcul géométrique
   S. Pion : La bibliothèque logicielle CGAL
4. Génération de maillages
   J-M. Mirebeau : Les deux réductions de Voronoï et leur application aux équations aux dérivées partielles
5. Courbes et surfaces
   P. Alliez : Reconstruction de surfaces
6. Espaces de configurations
   A. de Mesmay : Dessin de graphes
7. Structures de données géométriques
   D. Feldman : Core sets
8. Géométrie des données
   F. Chazal : Analyse topologique des données

1. Requêtes géométriques

2. Recherche de plus proches voisins en petites dimensions
   - Recherche de plus proches voisins exacte
   - Recherche de plus proches voisins approchée

3. Recherche de voisins et dimension intrinsèque des données

4. Recherche de voisins en grandes dimensions

Structures de données

Prétraitement et requêtes

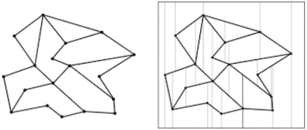
Objectif : Prétraiter les données pour répondre efficacement à de nombreuses requêtes

Exemple : localisation dans une carte planaire

Mesures de complexité

- Taille de l'entrée : n
- Prétraitement
  taille de la structure $S(n)$
  temps de construction $T(n)$
- Temps de réponse : $Q(n)$

Cloisonnement, history graph



$S(n) = O(n)$, $T(n) = O(n \log n)$,
$Q(n) = \log n$

Structures de données

Prétraitement et requêtes

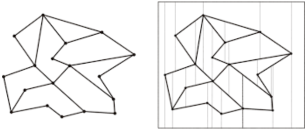
Objectif : Prétraiter les données pour répondre efficacement à de nombreuses requêtes

Exemple : localisation dans une carte planaire

Mesures de complexité

- Taille de l'entrée : n
- Prétraitement
  taille de la structure $S(n)$
  temps de construction $T(n)$
- Temps de réponse : $Q(n)$

Cloisonnement, history graph



$S(n) = O(n)$, $T(n) = O(n \log n)$,
$Q(n) = \log n$

Structures de données

Prétraitement et requêtes

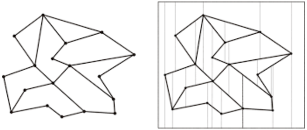
Objectif : Prétraiter les données pour répondre efficacement à de nombreuses requêtes

Exemple : localisation dans une carte planaire

Mesures de complexité

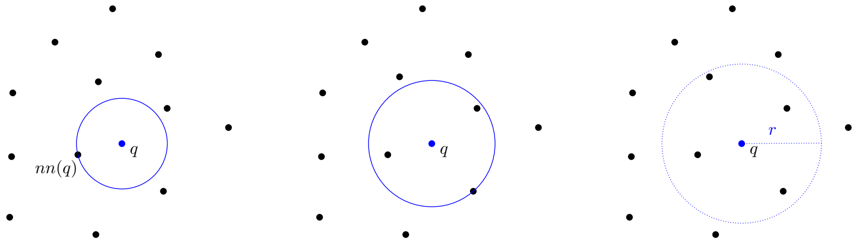
- Taille de l'entrée : n
- Prétraitement
  taille de la structure $S(n)$
  temps de construction $T(n)$
- Temps de réponse : $Q(n)$

Cloisonnement, history graph



$S(n) = O(n)$, $T(n) = O(n \log n)$,
$Q(n) = \log n$

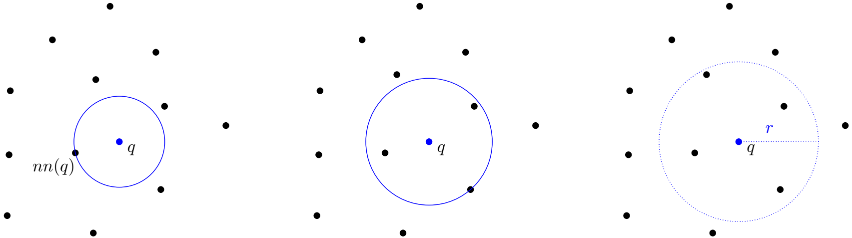
Recherche de plus proches voisins



Variantes

- k plus proches voisins : trouver les k points de P qui sont les plus proches de q
- Voisins à distance r : trouver les points de P à distance $\leq r$ de q
- Différentes métriques : L_2, L_p, L_∞, distance de Hamming etc...

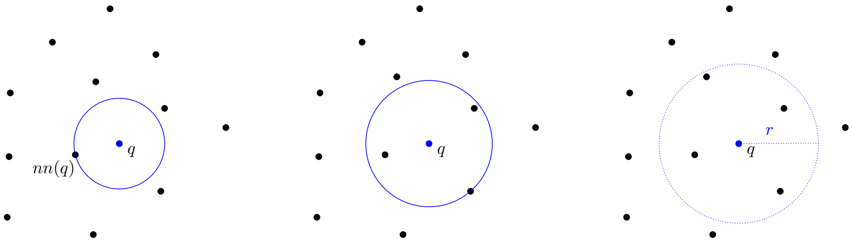
Recherche de plus proches voisins



Variantes

- k plus proches voisins : trouver les k points de P qui sont les plus proches de q
- Voisins à distance r : trouver les points de P à distance $\leq r$ de q
- Différentes métriques : L_2, L_p, L_∞, distance de Hamming etc...

Recherche de plus proches voisins



Variantes

- *k plus proches voisins* : trouver les k points de P qui sont les plus proches de q
- *Voisins à distance r* : trouver les points de P à distance $\leq r$ de q
- Différentes métriques : L_2, L_p, L_∞, distance de Hamming etc...

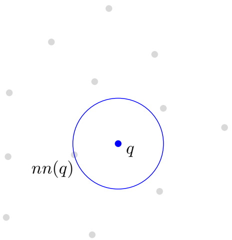
Recherche du plus proche voisin

Prétraiter un ensemble P de n points de $\mathbb{R}^d$

Requête : étant donné un point q, trouver rapidement un point de P plus proche de q

Solution triviale : comparer q à tous les points de P :

$$S(n) = T(n) = 0, \quad Q(n) = dn$$



Applications

- recherche dans des bases de données
- quantification vectorielle, codage (théorie de l'information)
- classification en apprentissage
- reconnaissance d'image, parole, musique

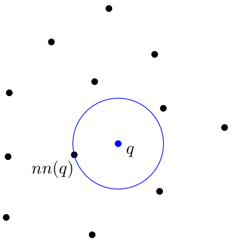
Recherche du plus proche voisin

Prétraiter un ensemble P de n points de $\mathbb{R}^d$

Requête : étant donné un point q, trouver rapidement un point de P plus proche de q

Solution triviale : comparer q à tous les points de P :

$$S(n) = T(n) = 0, \quad Q(n) = dn$$



Applications

- recherche dans des bases de données
- quantification vectorielle, codage (théorie de l'information)
- classification en apprentissage
- reconnaissance d'image, parole, musique

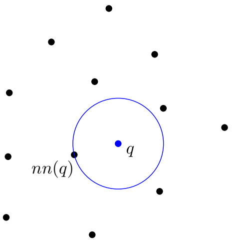
Recherche du plus proche voisin

Prétraiter un ensemble P de n points de $\mathbb{R}^d$

Requête : étant donné un point q, trouver rapidement un point de P plus proche de q

Solution triviale : comparer q à tous les points de P :

$$S(n) = T(n) = 0, \quad Q(n) = dn$$

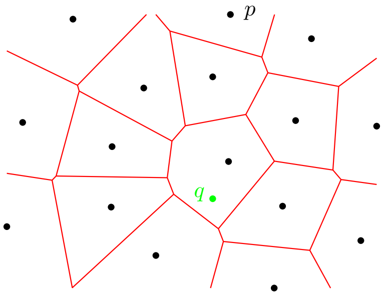


Applications

- recherche dans des bases de données
- quantification vectorielle, codage (théorie de l'information)
- classification en apprentissage
- reconnaissance d'image, parole, musique

Localisation dans le diagramme de Voronoï ($d = 2$)

Solution exacte optimale



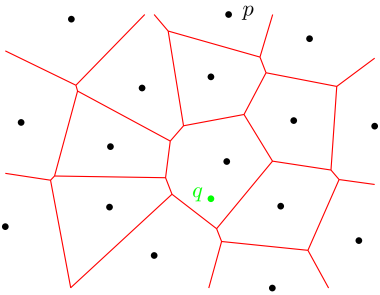
Prétraitement : diagramme de Voronoï $+$ structure de localisation

$$S(n) = O(n), \quad T(n) = O(n \log n)$$

Requête : $Q(n) = O(\log n)$

Localisation dans le diagramme de Voronoï ($d = 2$)

Solution exacte optimale

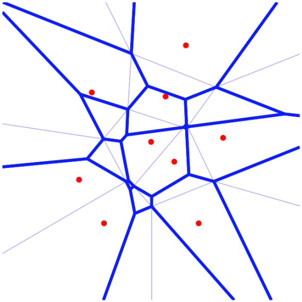


Prétraitement : diagramme de Voronoï $+$ structure de localisation

$$S(n) = O(n), \quad T(n) = O(n \log n)$$

Requête : $Q(n) = O(\log n)$

Recherche des k plus proches voisins ($d = 2$)



Prétraitement : diagramme de Voronoï d'ordre k + structure de localisation

$$S(n) = O(kn), \quad T(n) = O(kn \log n)$$

Requête : $Q(n) = O(\log n)$

Le fléau de la dimension

- La taille des diagrammes de Voronoï croît de manière exponentielle avec la dimension d : $O\left(n^{\lceil \frac{d}{2} \rceil}\right)$
- Toutes les structures de données permettant de chercher un plus proche voisin en temps sous-linéaire ont une taille exponentielle : $O\left((dn)^{O(d)}\right)$

Espace mémoire	Temps de requête	
$O\left(n^{2^{d+1}}\right)$	$O\left(2^d \log n\right)$	Dobkin, Lipton 76
$O\left(n^{\lceil \frac{d}{2} \rceil (1+\delta)}\right)$	$O\left(d^d log n\right)$	Clarkson 88
$O\left(n^{d+\delta}\right)$	$O\left(d^5 \log n\right)$	Meiser 93
		Agarwal, Erickson 98

- La complexité en espace peut être réduite à 2^d si p est un net

Le fléau de la dimension

- La taille des diagrammes de Voronoï croît de manière exponentielle avec la dimension d : $O\left(n^{\lceil \frac{d}{2} \rceil}\right)$
- Toutes les structures de données permettant de chercher un plus proche voisin en temps sous-linéaire ont une taille exponentielle : $O\left((dn)^{O(d)}\right)$

Espace mémoire	Temps de requête	
$O\left(n^{2^{d+1}}\right)$	$O\left(2^d \log n\right)$	Dobkin, Lipton 76
$O\left(n^{\lceil \frac{d}{2} \rceil (1+\delta)}\right)$	$O\left(d^d logn\right)$	Clarkson 88
$O\left(n^{d+\delta}\right)$	$O\left(d^5 \log n\right)$	Meiser 93
		Agarwal, Erickson 98

- La complexité en espace peut être réduite à 2^d si p est un net

1. Requêtes géométriques
2. Recherche de plus proches voisins en petites dimensions
   - Recherche de plus proches voisins exacte
   - Recherche de plus proches voisins approchée
3. Recherche de voisins et dimension intrinsèque des données
4. Recherche de voisins en grandes dimensions

Recherche approchée et partitionnements plus économiques

Un partitionnement exact (diagramme de Voronoï) conduit à un coût mémoire exponentiel, ce qui enlève tout intérêt pratique au delà de la dimension 3

Si on veut faire mieux que la solution naïve linéaire, il faut réduire ses ambitions

- accepter une réponse approximative
- accepter une probabilité d'erreur

Recherche approchée et partitionnements plus économiques

Un partitionnement exact (diagramme de Voronoï) conduit à un coût mémoire exponentiel, ce qui enlève tout intérêt pratique au delà de la dimension 3

Si on veut faire mieux que la solution naïve linéaire, il faut réduire ses ambitions

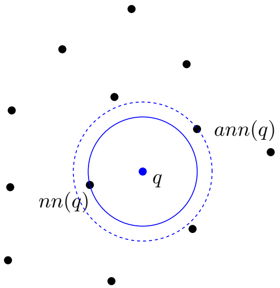
- accepter une réponse approximative
- accepter une probabilité d'erreur

Plus proche voisin approché

Etant donné un ensemble P de n points de $\mathbb{R}^d$
et un point de requête q,

un ϵ-plus proche voisin de q,
est un point $p \in P$ tel que :

$$d(q,p) \leq (1+\epsilon)d(q,P)$$



Recherche d'un voisin RV à distance $\leq r$

Soit $\mathcal{P} \subset \mathbb{R}^d$, $r > 0$ et $\varepsilon > 0$. q : un point de requête

si $d(q, \mathcal{P}) \leq r$ retourner $p \in \mathcal{P}$ s.t. $\|p - q\| \leq (1+\varepsilon)r$

si $d(q, \mathcal{P}) \geq (1+\varepsilon)\, r$ retourner "$d(q, \mathcal{P}) \geq r$"

sinon retourner une des 2 réponses

Recherche du plus proche voisin RV*

Utiliser RV et une recherche binaire sur r

$$\Rightarrow \quad T(RV^*) = T(RV) \times O \log(\tfrac{\Phi}{\varepsilon}) \qquad \text{où } \Phi = \frac{\max_{p,q\in\mathcal{P}} \|p-q\|}{\min_{p,q\in\mathcal{P}} \|p-q\|}$$

(étalement, spread)

Recherche approchée de voisins

Détection et optimisation

Recherche d'un voisin RV à distance $\leq r$

Soit $\mathcal{P} \subset \mathbb{R}^d$, $r > 0$ et $\varepsilon > 0$. q : un point de requête

si $d(q, \mathcal{P}) \leq r$	retourner	$p \in \mathcal{P}$ s.t. $\|p - q\| \leq (1+\varepsilon)r$
si $d(q, \mathcal{P}) \geq (1+\varepsilon)\, r$	retourner	"$d(q, \mathcal{P}) \geq r$"
sinon	retourner	une des 2 réponses

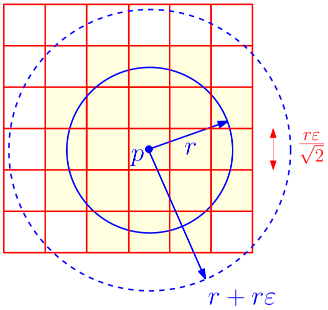
Recherche du plus proche voisin RV*

Utiliser RV et une recherche binaire sur r

$$\Rightarrow \quad T(RV^*) = T(RV) \times O \log\left(\frac{\Phi}{\varepsilon}\right) \qquad \text{où } \Phi = \frac{\max_{p,q \in \mathcal{P}} \|p-q\|}{\min_{p,q \in \mathcal{P}} \|p-q\|}$$

(étalement, spread)

Recherche approchée sur une grille



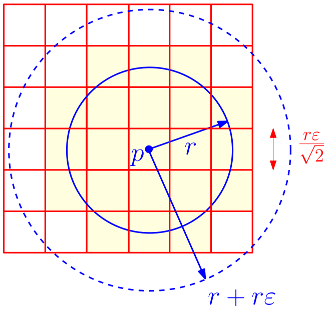
Pour chaque $p \in \mathcal{P}$, recouvrir $B(p, r)$ avec des cellules de la grille de côté $\frac{r\varepsilon}{\sqrt{d}}$

Nombre de cellules intersectant $B(p, r)$

Autour d'un point $\leq \left(\frac{C}{\varepsilon}\right)^d$ (argument de volume, $C < 5$)

au total $\leq n \left(\frac{C}{\varepsilon}\right)^d$

Recherche approchée sur une grille



Pour chaque $p \in \mathcal{P}$, recouvrir $B(p, r)$ avec des cellules de la grille de côté $\frac{r\varepsilon}{\sqrt{d}}$

Nombre de cellules intersectant $B(p, r)$

Autour d'un point $\leq \left(\frac{C}{\varepsilon}\right)^d$ (argument de volume, $C < 5$)

au total $\leq n \left(\frac{C}{\varepsilon}\right)^d$

Recherche pour r fixé

Pour un point q de requête, tester si q appartient à une des cellules intersectant $B(p, r)$ pour un point $p \in \mathcal{P}$

si oui : retourner p (p est à distance $\leq r + r\varepsilon$ de q)

sinon : q n'a aucun voisin dans $\mathcal{P}$ à distance $\leq r$

Complexité

$$S(n, r) = T(n, r) = n \left(\frac{C}{\varepsilon}\right)^d, \quad Q(n, r) = O(d)$$

Recherche du plus proche voisin

Construire t grilles pour les $r = r_0, r_0(1+\varepsilon), ..., r_0(1+\varepsilon)^t$

$r_0 = \min_{p,q \in \mathcal{P}} \|p - q\|$, t est le plus petit entier t.q. $r_0(1+\varepsilon)^t \geq \max_{p,q \in \mathcal{P}} \|p - q\|$

$$t \leq \log \Phi, \quad S(n) = T(n) = \log \Phi \times n \left(\frac{C}{\varepsilon}\right)^d, \quad Q(n) = \log\log \Phi \times O(d)$$

Recherche pour r fixé

Pour un point q de requête, tester si q appartient à une des cellules intersectant $B(p, r)$ pour un point $p \in \mathcal{P}$

si oui : retourner p (p est à distance $\leq r + r\varepsilon$ de q)

sinon : q n'a aucun voisin dans $\mathcal{P}$ à distance $\leq r$

Complexité

$$S(n, r) = T(n, r) = n \left(\frac{C}{\varepsilon}\right)^d, \quad Q(n, r) = O(d)$$

Recherche du plus proche voisin

Construire t grilles pour les $r = r_0, r_0(1+\varepsilon), ..., r_0(1+\varepsilon)^t$

$r_0 = \min_{p,q \in \mathcal{P}} \|p - q\|$, t est le plus petit entier t.q. $r_0(1+\varepsilon)^t \geq \max_{p,q \in \mathcal{P}} \|p - q\|$

$$t \leq \log \Phi, \quad S(n) = T(n) = \log \Phi \times n \left(\frac{C}{\varepsilon}\right)^d, \quad Q(n) = \log \log \Phi \times O(d)$$

Recherche approchée sur une grille

Structure de données et algorithme

Recherche pour r fixé

Pour un point q de requête, tester si q appartient à une des cellules intersectant $B(p, r)$ pour un point $p \in \mathcal{P}$

si oui : retourner p (p est à distance $\leq r + r\varepsilon$ de q)

sinon : q n'a aucun voisin dans $\mathcal{P}$ à distance $\leq r$

Complexité

$$S(n, r) = T(n, r) = n \left(\frac{C}{\varepsilon}\right)^{d}, \quad Q(n, r) = O(d)$$

Recherche du plus proche voisin

Construire t grilles pour les $r = r_0, r_0(1+\varepsilon), ..., r_0(1+\varepsilon)^t$

$r_0 = \min_{p,q \in \mathcal{P}} \|p - q\|$, t est le plus petit entier t.q. $r_0(1+\varepsilon)^t \geq \max_{p,q \in \mathcal{P}} \|p - q\|$

$$t \leq \log \Phi, \quad S(n) = T(n) = \log \Phi \times n \left(\frac{C}{\varepsilon}\right)^{d}, \quad Q(n) = \log \log \Phi \times O(d)$$

Recherche approchée sur une grille

- La recherche approchée permet de beaucoup réduire la complexité en espace

$$O(n^d) \quad \searrow \quad O\left(\log \Phi \times n \left(\frac{C}{\varepsilon}\right)^d\right)$$

- Peut-on supprimer la dépendance en d ?

Recherche approchée sur une grille

- La recherche approchée permet de beaucoup réduire la complexité en espace

$$O(n^d) \quad \searrow \quad O\left(\log \Phi \times n \left(\frac{C}{\varepsilon}\right)^d\right)$$

- Peut-on supprimer la dépendance en d ?

Lemme (Corollaire de la concentration des fonctions Lipschitz [Lévy])

Soit U un sous-espace affine de dimension k de $\mathbb{R}^d$ avec $k = \Omega(\frac{1}{\varepsilon^2} \log n)$

$$\forall p \in P,\ f(p) = \sqrt{\frac{d}{k}}\, \pi_U(p)$$

$$\forall p, q \in P,\ \text{proba}\left[\frac{\|f(p) - f(q)\|^2}{\|p - q\|^2} \notin [(1-\epsilon), (1+\epsilon)]\right] \leq \frac{2}{n^2}$$

Lemme de Johnson-Lindenstrauss

Soit P un ensemble de n points de $\mathbb{R}^d$ et $\epsilon \in (0,1)$. Si on projette P sur un plan aléatoire de dimension $k = \Omega(\frac{1}{\varepsilon^2} \log n)$, avec probabilité $\frac{1}{n}$

$$\forall p, q \in P,\quad (1-\epsilon)\|p-q\|^2 \leq \|f(p) - f(q)\|^2 \leq (1+\epsilon)\|p-q\|^2$$

Démonstration : Proba $\geq 1 - \binom{n}{2} \frac{2}{n^2} = \frac{1}{n}$. Projeter $O(n)$ fois donne une proba cst

Lemme (Corollaire de la concentration des fonctions Lipschitz [Lévy])

Soit U un sous-espace affine de dimension k de $\mathbb{R}^d$ avec $k = \Omega(\frac{1}{\varepsilon^2} \log n)$

$$\forall p \in P,\, f(p) = \sqrt{\frac{d}{k}}\, \pi_U(p)$$

$$\forall p, q \in P,\ \text{proba}\left[\frac{\|f(p) - f(q)\|^2}{\|p - q\|^2} \notin [(1-\epsilon), (1+\epsilon)]\right] \leq \frac{2}{n^2}$$

Lemme de Johnson-Lindenstrauss

Soit P un ensemble de n points de $\mathbb{R}^d$ et $\epsilon \in (0,1)$. Si on projette P sur un plan aléatoire de dimension $k = \Omega(\frac{1}{\varepsilon^2} \log n)$, avec probabilité $\frac{1}{n}$

$$\forall p, q \in P, \quad (1-\epsilon)\|p-q\|^2 \leq \|f(p) - f(q)\|^2 \leq (1+\epsilon)\|p-q\|^2$$

Démonstration : Proba $\geq 1 - \left(\begin{array}{c} n \\ 2 \end{array}\right) \frac{2}{n^2} = \frac{1}{n}$. Projeter $O(n)$ fois donne une proba cst

Le lemme de J&L appliqué dans l'espace de dimension k donne une structure de données

$$S(n) = \left(\frac{1}{\varepsilon}\right)^{O(\frac{\log n}{\varepsilon^2})} \log \Phi = n^{O\left(\frac{\log \frac{1}{\varepsilon}}{e^2}\right)} \log \Phi$$

$$\begin{aligned} Q(n) &= \text{coût de la projection} + \text{coût de la recherche dans } \mathbb{R}^k \\ &= O\left(\frac{d \log n}{\varepsilon^2}\right) + O(\log n \, \log\log \Phi) \end{aligned}$$

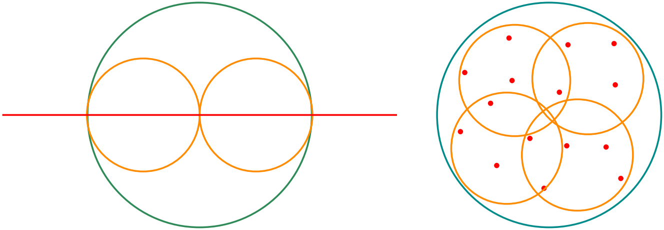
1. Requêtes géométriques
2. Recherche de plus proches voisins en petites dimensions
   - Recherche de plus proches voisins exacte
   - Recherche de plus proches voisins approchée
3. Recherche de voisins et dimension intrinsèque des données
4. Recherche de voisins en grandes dimensions

Espaces doublants

Définition

Un ensemble X est de dimension doublante $\mathrm{dbd}(X) = d$ si d est le plus petit entier t.q.

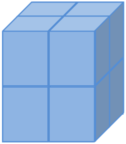
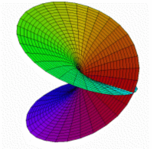
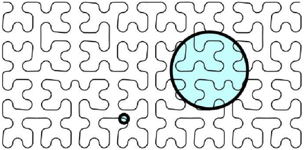
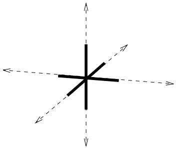
$\forall B$, $B \cap X$ peut être recouvert par 2^d boules de rayon moitié



Pour tout sous-ensemble Y de X : $\mathrm{dbd}(Y) \leq \mathrm{dbd}(X)$

Espaces doublants

Exemples

- Espace affine : si X est sous-ensemble d'un espace affine de dimension k, $\mathrm{dbd}(X) = O(k)$
- Sous-variétés : si X est une sous-variété de dimension k de $\mathbb{R}^d$ *de portée positive*, $\mathrm{dbd}(X) = O(k)$
- Ensembles creux : si $\forall x \in X$, x n'a que k coordonnées non nulles, $\mathrm{dbd}(X) = O(k \log d)$

Si $X \subset \mathbb{R}^d$ est un espace de dimension doublante $\mathrm{dbd}(X)$

- $\boxed{Z}$: il n'existe pas de plongement de X dans un espace $\mathbb{R}^m$ avec distortion $1+\varepsilon$ si on impose que m et ε dépendent de $\mathrm{dbd}(X)$ et pas de n [Baral et al. 2015]
- Trouver une représentation de X qui s'adapte à $\mathrm{dbd}(X)$: partitions de l'espace arborescentes

Si $X \subset \mathbb{R}^d$ est un espace de dimension doublante $\mathrm{dbd}(X)$

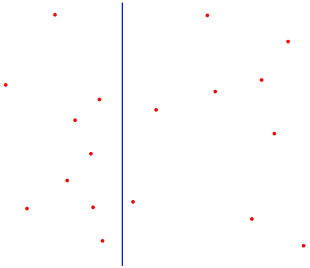
- $\boxed{Z}$: il n'existe pas de plongement de X dans un espace $\mathbb{R}^m$ avec distortion $1+\varepsilon$ si on impose que m et ε dépendent de $\mathrm{dbd}(X)$ et pas de n [Baral et al. 2015]
- Trouver une représentation de X qui s'adapte à $\mathrm{dbd}(X)$: partitions de l'espace arborescentes

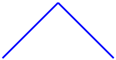
Les arbres kd

Découpage récursif en hyper-rectangles

Les arbres kd

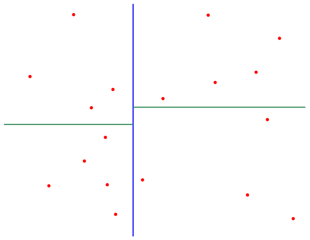
Découpage récursif en hyper-rectangles

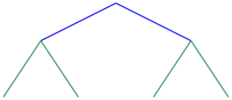




Les arbres kd

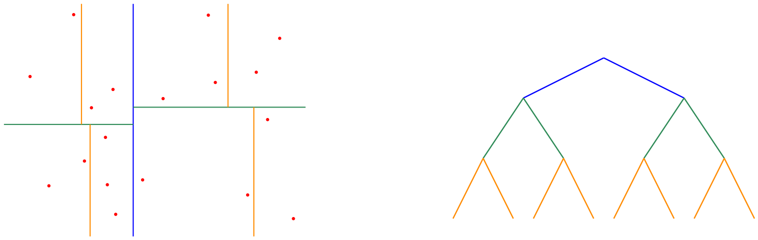
Découpage récursif en hyper-rectangles





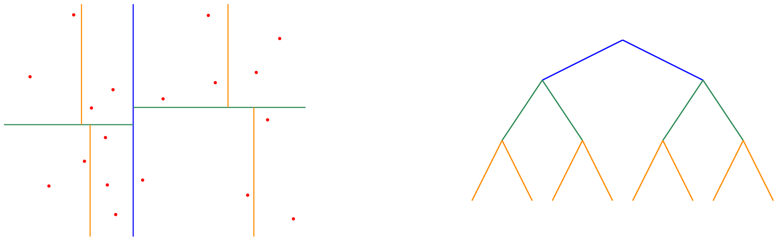
Les arbres kd

Découpage récursif en hyper-rectangles



Découpage récursif en hyper-rectangles

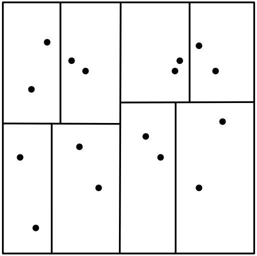
Les arbres *kd*



- Chaque nœud N de l'arbre kd représente une cellule $C(N)$ de la subdivision
- Les descendants d'un nœud N représentent les cellules obtenues en coupant $C(N)$ par un hyperplan orthogonal à un axe de coordonnées
- Toutes les coupes à un niveau donné sont parallèles
- Les points de $\mathcal{P}$ sont affectés aux feuilles de l'arbre

Les arbres *kd*

Complexité



function MakeTree(S)
  If $|S| < n_o$: return (Leaf)
  Rule $=$ **ChooseRule**(S)
  LeftTree $=$ **MakeTree**($\{x \in S : \text{Rule}(x) = \text{true}\}$)
  RightTree $=$ **MakeTree**($\{x \in S : \text{Rule}(x) = \text{false}\}$)
  return (Rule, LeftTree, RightTree)

function ChooseRule(S)
  Choose a coordinate direction i
  $\text{Rule}(x) = (x_i \leq \text{median}(\{z_i : z \in S\}))$
  return (Rule)

Profondeur de l'arbre et temps de localisation d'un point x

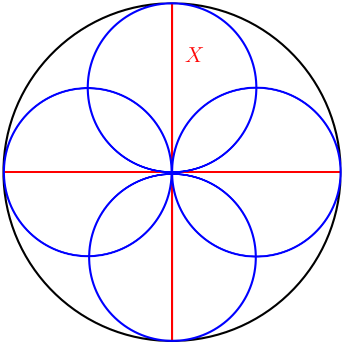
$$Q(n) = O(\log \frac{n}{n_0}) = O(\log n) \quad \text{si} \quad n_0 = O(1)$$

Taille et temps de construction

$$S(n) = O(n) \qquad T(n) = n \log n$$

(médiane en temps $O(n)$)

Les arbres kd ne s'adaptent pas à la dimension doublante de $\mathcal{P}$



$X = \bigcup_{i=1}^{d} \{te_i,\ -1 \leq t \leq 1\}$

$X \subset B(O, 1)$

$X \subset \bigcup_{i=1}^{d} B(\pm\frac{e_i}{2}, \frac{1}{2})$ ($2d$ balls)

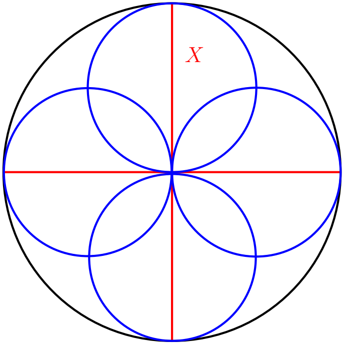
$\mathrm{dbd}(X) = \log 2d = 1 + \log d$

Z L'arbre kd a besoin de d niveaux pour diviser par 2 le diamètre de ses cellules

Décroissance du diamètre des cellules

Une borne inférieure [Dasgupta & Freund 2008]

Les arbres kd ne s'adaptent pas à la dimension doublante de $\mathcal{P}$



$$X = \bigcup_{i=1}^{d} \{te_i,\ -1 \leq t \leq 1\}$$
$$X \subset B(O,1)$$
$$X \subset \bigcup_{i=1}^{d} B(\pm \tfrac{e_i}{2}, \tfrac{1}{2}) \qquad (2d \text{ balls})$$

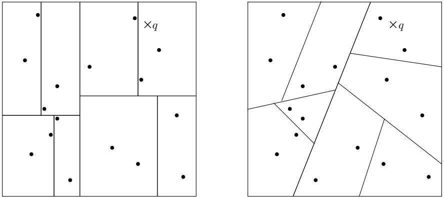
$$\text{dbd}(X) = \log 2d = 1 + \log d$$

Z L'arbre kd a besoin de d niveaux pour diviser par 2 le diamètre de ses cellules

Décroissance du diamètre des cellules

La randomisation aide

Arbres RP (projections aléatoires) [Dasgupta & Sinha 2012]



1. Choisir une direction v au hasard sur S^{d-1}
2. Choisir une perturbation δ au hasard dans un intervalle I
3. Calculer la médiane perturbée $m = \mathrm{mediane}(p \cdot v, p \in \mathcal{P}) + \delta$
4. Couper à la médiane selon $H \perp v$

Rotation aléatoire des axes de coordonnées [Vempala 2012]

Théorème

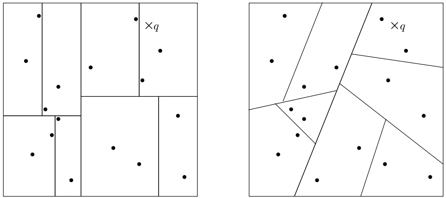
$\mathcal{P}$ un ensemble fini de points de $\mathbb{R}^d$ de dimension doublante k. Il existe une cst A t.q., avec probabilité $> 1/2$, toute cellule C et toute cellule C' qui est au moins $Ak \log k$ niveaux en dessous de C vérifient

$$\mathrm{diam}(C' \cap \mathcal{P}) \leq \frac{1}{2} \mathrm{diam}(C \cap \mathcal{P})$$

Décroissance du diamètre des cellules

La randomisation aide

Arbres RP (projections aléatoires) [Dasgupta & Sinha 2012]



1. Choisir une direction v au hasard sur S^{d-1}
2. Choisir une perturbation δ au hasard dans un intervalle I
3. Calculer la médiane perturbée $m = \text{mediane}(p \cdot v, p \in \mathcal{P}) + \delta$
4. Couper à la médiane selon $H \perp v$

Rotation aléatoire des axes de coordonnées [Vempala 2012]

Théorème

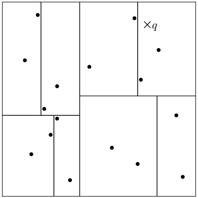
$\mathcal{P}$ un ensemble fini de points de $\mathbb{R}^d$ de dimension doublante k. Il existe une cst A t.q., avec probabilité $> 1/2$, toute cellule C et toute cellule C' qui est au moins $Ak \log k$ niveaux en dessous de C vérifient

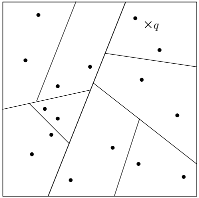
$$\text{diam}(C' \cap \mathcal{P}) \leq \frac{1}{2} \text{diam}(C \cap \mathcal{P})$$

Décroissance du diamètre des cellules

La randomisation aide

Arbres RP (projections aléatoires) [Dasgupta & Sinha 2012]





1. Choisir une direction v au hasard sur S^{d-1}
2. Choisir une perturbation δ au hasard dans un intervalle I
3. Calculer la médiane perturbée $m = \mathrm{mediane}(p \cdot v, p \in \mathcal{P}) + \delta$
4. Couper à la médiane selon $H \perp v$

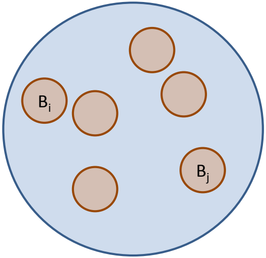
Rotation aléatoire des axes de coordonnées [Vempala 2012]

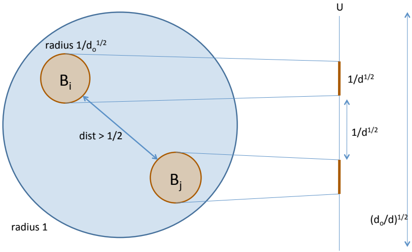
Théorème

$\mathcal{P}$ un ensemble fini de points de $\mathbb{R}^d$ de dimension doublante k. Il existe une cst A t.q., avec probabilité $> 1/2$, toute cellule C et toute cellule C' qui est au moins $Ak \log k$ niveaux en dessous de C vérifient

$$\mathrm{diam}(C' \cap \mathcal{P}) \leq \frac{1}{2}\, \mathrm{diam}(C \cap \mathcal{P})$$

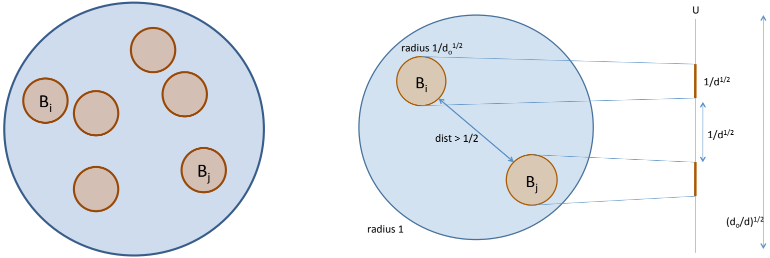
Idée de la preuve





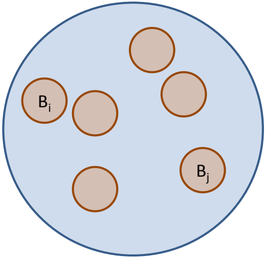
- Considérer les points dans une cellule C (rayon 1) et la recouvrir de $k^{k/2}$ boules B_i de rayon $r = 1/\sqrt{k}$
- Considérer toutes les paires (B_i, B_j) à distance $\geq 1/2r$. Une coupe aléatoire sépare B_i et B_j avec proba cst
- Il y a k^k paires (B_i, B_j). Après $k \log k$ coupes, les paires éloignées ont été séparées avec proba cst

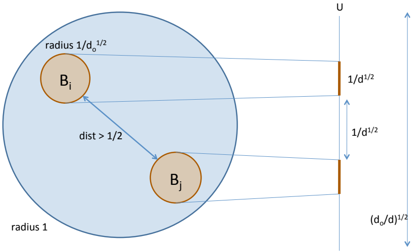
Idée de la preuve



- Considérer les points dans une cellule C (rayon 1) et la recouvrir de $k^{k/2}$ boules B_i de rayon $r = 1/\sqrt{k}$
- Considérer toutes les paires (B_i, B_j) à distance $\geq 1/2r$. Une coupe aléatoire sépare B_i et B_j avec proba cst
- Il y a k^k paires (B_i, B_j). Après $k \log k$ coupes, les paires éloignées ont été séparées avec proba cst

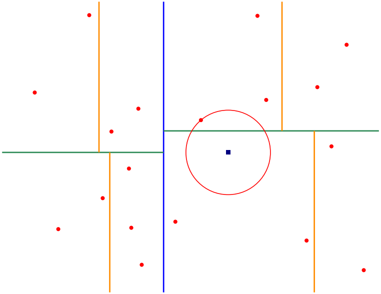
Idée de la preuve





- Considérer les points dans une cellule C (rayon 1) et la recouvrir de $k^{k/2}$ boules B_i de rayon $r = 1/\sqrt{k}$
- Considérer toutes les paires (B_i, B_j) à distance $\geq 1/2r$. Une coupe aléatoire sépare B_i et B_j avec proba cst
- Il y a k^k paires (B_i, B_j). Après $k \log k$ coupes, les paires éloignées ont été séparées avec proba cst

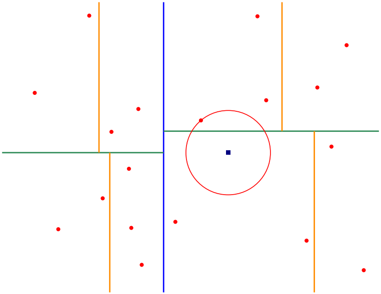
Des arbres kd à la recherche de proches voisins



$\boxed{\text{Z}}$ *le point de $\mathcal{P}$ le plus proche de x n'appartient pas toujours à la cellule qui contient x*

Plusieurs heuristiques

Des arbres kd à la recherche de proches voisins



Z *le point de $\mathcal{P}$ le plus proche de x n'appartient pas toujours à la cellule qui contient x*

Plusieurs heuristiques

Analyse

- $F(q, \{p_1, \ldots, p_n\}) = \frac{1}{n} \sum_{i=2}^{n} \frac{\|q-p_1\|}{\|q-p_i\|}$ (les p_i sont triés par dist. $\nearrow$ à q)
- Probabilité d'échec $= O(F \log \frac{1}{F})$ (ne retourne pas le ppv (q))
  (sur la randomisation dans la construction de l'arbre)
- si $q \in \mathcal{P}$ est choisi au hasard dans $\mathcal{P}$

  $E(F) \leq C^k \log \Phi(\mathcal{P})$ où $k = \mathrm{dbd}(\mathcal{P})$

Analyse

- $F(q, \{p_1, \ldots, p_n\}) = \frac{1}{n} \sum_{i=2}^{n} \frac{\|q-p_1\|}{\|q-p_i\|}$ (les p_i sont triés par dist. $\nearrow$ à q)
- Probabilité d'échec $= O(F \log \frac{1}{F})$ (ne retourne pas le ppv (q))
  (sur la randomisation dans la construction de l'arbre)
- si $q \in \mathcal{P}$ est choisi au hasard dans $\mathcal{P}$

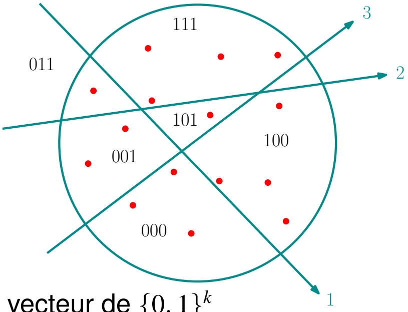
  $$E(F) \leq C^k \log \Phi(\mathcal{P}) \quad \text{où} \quad k = \mathrm{dbd}(\mathcal{P})$$

Analyse

- $F(q, \{p_1, \ldots, p_n\}) = \frac{1}{n} \sum_{i=2}^{n} \frac{\|q-p_1\|}{\|q-p_i\|}$ (les p_i sont triés par dist. $\nearrow$ à q)
- Probabilité d'échec $= O(F \log \frac{1}{F})$ (ne retourne pas le ppv (q))
  (sur la randomisation dans la construction de l'arbre)
- si $q \in \mathcal{P}$ est choisi au hasard dans $\mathcal{P}$

$$E(F) \leq C^k \log \Phi(\mathcal{P}) \quad \text{où } k = \mathrm{dbd}(\mathcal{P})$$

Partitionnement aléatoire de l'espace



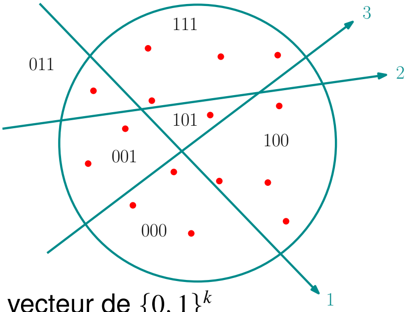
$\mathcal{P}$ un ensemble de n points de $\mathbb{R}^d$

- k hyperplans définissent 2^k cellules
- chaque cellule est caractérisée par un vecteur de $\{0,1\}^k$
- en moyenne chaque cellule contient $\frac{n}{2^k}$ points

- Localisation : trouver la cellule $C(x)$ qui contient un point x : $O(dk)$
- Trouver le point de $C(x)$ le plus proche de x : coût moyen : $O(d\,\frac{n}{2^k})$
- Coût de recherche du ppv (x) : $O(dk + d\,\frac{n}{2^k}) = O(\log n)$ si $k = O(\log n)$

Probabilité d'échec ?

Partitionnement aléatoire de l'espace



$\mathcal{P}$ un ensemble de n points de $\mathbb{R}^d$

- k hyperplans définissent 2^k cellules
- chaque cellule est caractérisée par un vecteur de $\{0,1\}^k$
- en moyenne chaque cellule contient $\frac{n}{2^k}$ points

- Localisation : trouver la cellule $C(x)$ qui contient un point x : $O(dk)$
- Trouver le point de $C(x)$ le plus proche de x : coût moyen : $O(d\,\frac{n}{2^k})$
- Coût de recherche du ppv (x) : $O(dk + d\,\frac{n}{2^k}) = O(\log n)$ si $k = O(\log n)$

Probabilité d'échec ?

Hâchage sensible à la localité (LSH)

Un domaine toujours actif depuis l'article fondateur [Indyk & Motwani 1998]

Définition

Une famille de fonctions $\mathcal{F}$ est dite (r,R)-sensible si, pour tous $p,q \in \mathcal{P}$, il existe $p_1, p_2 \in [0,1]$, $p_1 > p_2$ t.q. si f est pris au hasard dans $\mathcal{F}$

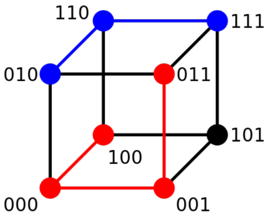
1. if $d(p,q) \leq r \quad \Rightarrow \quad \text{proba}(f(p) = f(q)) \geq p_1$
2. if $d(p,q) > R \quad \Rightarrow \quad \text{proba}(f(p) = f(q)) \leq p_2$

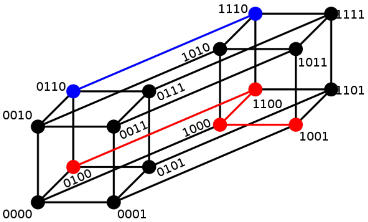
Recherche de voisins sur l'hypercube

Hypercube et distance de Hamming

d-hypercube : $\mathcal{H}^d = \{0,1\}^d$

Mots binaires : $p \in \mathcal{H}^d$: $p = (p_1, ..., p_d)$ où $p_i = 0 \;\text{ ou }\; 1$





Distance de Hamming $d_H(p,q)$ entre $p, q \in \mathcal{H}^d$

Nombre de coordonnées i t.q. $p_i \neq q_i$

$$d_H((0,1,0),\ (1,1,1)) = 2$$

Recherche approchée de voisins sur l'hypercube

Recherche d'un voisin à distance $\leq r$

Soit $\mathcal{P} \subset \mathcal{H}^d$, $r > 0$ et $\varepsilon > 0$, et q un point de requête.

si $d_H(q, \mathcal{P}) \leq r$	retourner $p \in \mathcal{P}$ s.t. $\|p - q\| \leq (1 + \varepsilon)r$
si $d_H(q, \mathcal{P}) \geq (1 + \varepsilon)\, r$	retourner "$d_H(q, \mathcal{P}) \geq r$"
sinon	retourner une des 2 réponses

Recherche du plus proche voisin

Utiliser la recherche de voisins et une recherche binaire

$\rightarrow$ $\times$ le coût du pb de décision par $O \log(\frac{d}{\varepsilon})$

Recherche approchée de voisins sur l'hypercube

Recherche d'un voisin à distance $\leq r$

Soit $\mathcal{P} \subset \mathcal{H}^d$, $r > 0$ et $\varepsilon > 0$, et q un point de requête.

si $d_H(q, \mathcal{P}) \leq r$	retourner $p \in \mathcal{P}$ s.t. $\|p - q\| \leq (1+\varepsilon)r$
si $d_H(q, \mathcal{P}) \geq (1+\varepsilon)\, r$	retourner "$d_H(q, \mathcal{P}) \geq r$"
sinon	retourner une des 2 réponses

Recherche du plus proche voisin

Utiliser la recherche de voisins et une recherche binaire

$$\rightarrow \times \text{ le coût du pb de décision par } O \log(\tfrac{d}{\varepsilon})$$

$\mathcal{F} = \{f_1, ..., f_d\}$ où $f_i(p) = i$-ième coordonnée de p

Lemme

$\forall r > 0, \varepsilon > 0, \quad \mathcal{F}$ est $(r, (1+\varepsilon)r)$-sensible

Démonstration

1. Si $d_H(p,q) \leq r$, p et q ont $\leq r$ bits différents, et $\mathrm{proba}(f_i(p) = f_i(q)) \geq p_1 = 1 - \frac{r}{n}$ (f_i tiré au hasard dans $\mathcal{F}$)

2. Si $d_H(p,q) \geq (1+\varepsilon)\, r$, $\mathrm{proba}(f_i(p) = f_i(q)) \leq p_2 = 1 - \frac{(1+\varepsilon)\, r}{n}$ $\square$

$\mathcal{F} = \{f_1, \dots, f_d\}$ où $f_i(p) = i$-ième coordonnée de p

Lemme

$\forall r > 0, \varepsilon > 0,$ $\mathcal{F}$ est $(r, (1+\varepsilon)r)$-sensible

Démonstration

1. Si $d_H(p,q) \leq r$, p et q ont $\leq r$ bits différents, et $\mathrm{proba}(f_i(p) = f_i(q)) \geq p_1 = 1 - \frac{r}{n}$
(f_i tiré au hasard dans $\mathcal{F}$)

2. Si $d_H(p,q) \geq (1+\varepsilon)\, r$, $\mathrm{proba}(f_i(p) = f_i(q)) \leq p_2 = 1 - \frac{(1+\varepsilon)\, r}{n}$ $\square$

Combiner k fonctions : $\mathcal{G}_k = \{g \mid g(p) = (f^1(p), \ldots, f^k(p)),\ \text{où}\ f^i \in \mathcal{F}\}$

Opérateur $\triangleq$: $g(p) \triangleq g(q) \quad \Leftrightarrow \quad g^i(p) = g^i(q),\ \ \forall\, i \in [1, k]$

Lemme

Si $\mathcal{F}$ est une famille (r, R)-sensible avec p_1 et p_2, alors $\mathcal{G}_k$ est (r, R)-sensible avec p_1^k et p_2^k

Démonstration

$\forall\, p, q \in \mathcal{H}^d, \quad d_H(p, q) \leq r$:

$$\begin{aligned} \text{proba}(g(p) \triangleq g(q)) &= \text{proba}\left(f^i(p) = f^i(q)\ \ \forall i \in [1, n]\right) \\ &= \prod_{i=1}^{k} \text{proba}(f^i(p) = f^i(q)) \geq p_1^k \end{aligned}$$

De même, $\forall\, p, q \in \mathcal{H}^d, \quad d_H(p, q) > R$: $\text{proba}(g(p) \triangleq g(q)) \leq p_2^k$

Combiner k fonctions : $\mathcal{G}_k = \{g \mid g(p) = (f^1(p), ..., f^k(p)),\ \text{où}\ f^i \in \mathcal{F}\}$

Opérateur $\triangleq$: $g(p) \triangleq g(q) \quad \Leftrightarrow \quad g^i(p) = g^i(q), \quad \forall\ i \in [1, k]$

Lemme

Si $\mathcal{F}$ est une famille (r, R)-sensible avec p_1 et p_2, alors $\mathcal{G}_k$ est (r, R)-sensible avec p_1^k et p_2^k

Démonstration

$\forall\ p, q \in \mathcal{H}^d, \quad d_H(p, q) \leq r$:

$$\begin{aligned} \text{proba}(g(p) \triangleq g(q)) &= \text{proba}\left(f^i(p) = f^i(q) \ \ \forall i \in [1, n]\right) \\ &= \prod_{i=1}^{k} \text{proba}(f^i(p) = f^i(q)) \geq p_1^k \end{aligned}$$

De même, $\forall\ p, q \in \mathcal{H}^d, \quad d_H(p, q) > R$: $\text{proba}(g(p) \triangleq g(q)) \leq p_2^k$

On peut construire une structure de données qui permet de trouver l'ensemble des points de $\mathcal{P}$ en collision avec un point de requête q

$$X = \{p \in \mathcal{P} : g(p) = g(q)\}$$

$$S(n,k) = O(nk), \quad T(n,k) = O(nk), \quad Q(n,k) = O(k + |X|)$$

Réamplification

$\mathcal{H}_t = \{g \mid g(p) = (g^1(p), ..., g^t(p)),$ où $g^i \in \mathcal{G}_k\}$

Opérateur $\stackrel{\vee}{=}$: $g(p) \stackrel{\vee}{=} g(q) \quad \Leftrightarrow \quad \exists\, i \in [1, t] \mid g^i(p) = g^i(q)$

Lemme

Si $\mathcal{G}_k$ est une famille (r, R)-sensible pour l'opérateur $\stackrel{\wedge}{=}$ avec p_1^k et p_2^k, alors $\mathcal{H}_t$ est (r, R)-sensible pour l'opérateur $\stackrel{\vee}{=}$ avec probabilités

$$\phi = 1 - (1 - p_1^k)^t \quad \text{et} \quad \psi = 1 - (1 - p_2^k)^t$$

Démonstration

$\forall\, p, q \in \mathcal{H}^d, \quad d_H(p, q) \leq r$:

$$\text{proba}(g(p) \stackrel{\vee}{=} g(q)) \quad = \quad 1 - \prod_{i=1}^{t} \text{proba}(g^i(p) \neq g^i(q)) \ \geq \ 1 - (1 - p_1^k)^t$$

De même, $\forall\, p, q \in \mathcal{H}^d, \quad d_H(p, q) > R : \text{proba}(g(p) \stackrel{\vee}{=} g(q)) \ \leq \ 1 - (1 - p_2^k)^t$

Réamplification

$$\mathcal{H}_t = \{g \mid g(p) = (g^1(p), \ldots, g^t(p)), \;\; \text{où} \;\; g^i \in \mathcal{G}_k\}$$

Opérateur $\overset{\vee}{=}$: $\;g(p) \overset{\vee}{=} g(q) \;\; \Leftrightarrow \;\; \exists\, i \in [1, t] \mid g^i(p) = g^i(q)$

Lemme

Si $\mathcal{G}_k$ est une famille (r, R)-sensible pour l'opérateur $\overset{\wedge}{=}$ avec p_1^k et p_2^k, alors $\mathcal{H}_t$ est (r, R)-sensible pour l'opérateur $\overset{\vee}{=}$ avec probabilités

$$\phi = 1 - (1 - p_1^k)^t \quad \text{et} \quad \psi = 1 - (1 - p_2^k)^t$$

Démonstration

$\forall\, p, q \in \mathcal{H}^d, \;\; d_H(p, q) \leq r$:

$$\text{proba}(g(p) \overset{\vee}{=} g(q)) \;\; = \;\; 1 - \prod_{i=1}^{t} \text{proba}(g^i(p) \neq g^i(q)) \;\geq\; 1 - (1 - p_1^k)^t$$

De même, $\forall\, p, q \in \mathcal{H}^d, \;\; d_H(p, q) > R$: $\text{proba}(g(p) \overset{\vee}{=} g(q)) \;\leq\; 1 - (1 - p_2^k)^t$

$t = \lceil \frac{4}{p_1^k} \rceil$

$$\Rightarrow \quad \phi = 1 - (1 - p_1^k)^t \;\geq\; 1 - \exp(-p_1^k t) \;\geq\; 1 - \exp(-4) \;\geq\; \frac{3}{4}$$
$$\Rightarrow \quad \psi = 1 - (1 - p_2^k)^t \;\leq\; t\, p_2^k \;\leq\; 8 \left(\frac{p_2}{p_1}\right)^k$$

Lemme

L'espérance du nombre de points de $\mathcal{P} \setminus B(q, r(1+\varepsilon))$ en collision avec q est $L = O\left(n \left(\frac{p_2}{p_1}\right)^k\right)$

Structure de données : construire t structures de données précédentes : $S(n) = T(n) = O(nkt)$

Temps de requête

Extraire les L points et calculer la distance de q à ces points prend un temps $Q(n) = O(kt + Ld) = O\left(kt + n\left(\frac{p_2}{p_1}\right)^k\right)$

Choisir k pour équilibrer les 2 termes $\rightarrow$ $t, L = O(n^{\frac{1}{1+\varepsilon}})$ et $k = O(\log n)$

Probabilité de succès : $cst \rightarrow cst\,(1 - \frac{1}{n})$ en construisant $O(\log n)$ structures de données et en les interrogeant toutes

Structure de données : construire t structures de données précédentes : $S(n) = T(n) = O(nkt)$

Temps de requête

Extraire les L points et calculer la distance de q à ces points prend un temps $Q(n) = O(kt + Ld) = O\left(kt + n\left(\frac{p_2}{p_1}\right)^k\right)$

Choisir k pour équilibrer les 2 termes $\rightarrow$ $t, L = O(n^{\frac{1}{1+\varepsilon}})$ et $k = O(\log n)$

Probabilité de succès : $cst \rightarrow cst\,(1 - \frac{1}{n})$ en construisant $O(\log n)$ structures de données et en les interrogeant toutes

Structure de données : construire t structures de données précédentes : $S(n) = T(n) = O(nkt)$

Temps de requête

Extraire les L points et calculer la distance de q à ces points prend un temps $Q(n) = O(kt + Ld) = O\left(kt + n\left(\frac{p_2}{p_1}\right)^k\right)$

Choisir k pour équilibrer les 2 termes $\rightarrow$ $t, L = O(n^{\frac{1}{1+\varepsilon}})$ et $k = O(\log n)$

Probabilité de succès : $cst \rightarrow cst\,(1 - \frac{1}{n})$ en construisant $O(\log n)$ structures de données et en les interrogeant toutes

Théorème

La structure de données résoud le problème de décision avec grande probabilité

$$\begin{aligned} T(n) &= O(n^{1+1/(1+\varepsilon)} \log^2 n) \\ S(n) &= O(d\,n + n^{1+1/(1+\varepsilon)} \log^2 n) \\ Q(n) &= O(d\,n^{1/(1+\varepsilon)} \log n) \end{aligned}$$

Utile pour ε assez grand : $\varepsilon = 10 \Rightarrow Q(n) = O(d\,n^{1/11})$

Théorème

La structure de données résoud le problème de décision avec grande probabilité

$$\begin{aligned} T(n) &= O(n^{1+1/(1+\varepsilon)} \log^2 n) \\ S(n) &= O(d\,n + n^{1+1/(1+\varepsilon)} \log^2 n) \\ Q(n) &= O(d\,n^{1/(1+\varepsilon)} \log n) \end{aligned}$$

Utile pour ε assez grand : $\varepsilon = 10 \ \Rightarrow\ Q(n) = O(d\,n^{1/11})$

Conclusions

- Le problème de la recherche de voisins a suscité beaucoup de recherches
- Pour obtenir des structures de données qui ne dépendent pas exponentiellement de la dimension ambiante, il faut se limiter à des approximations et/ou utiliser des algorithmes randomisés
- Les projections aléatoires jouent un rôle central
- D'autres problèmes qui n'ont pas été abordés
  - k plus proches voisins
  - tous les plus (k) proches voisins
  - plus petite paire (2ième cours)
  - Variantes de LSH (autres métriques)