Passage à l'échelle de la recherche et de la fouille de contenus multimédia

Michel Crucianu (CEDRIC – CNAM)
michel.crucianu@cnam.fr
avec des contributions de
Laurent Amsaleg (IRISA) et Frédéric Précioso (ENSEA)

Plan de la présentation

1. **Recherche et fouille de données multimédia : exemples**
2. Typologie des besoins et nature des problèmes
3. Passage à l'échelle de la recherche par similarité
4. Passage à l'échelle de la fouille
5. Apports et implications de la distribution

Contexte

- Production de contenus numériques
  - Grand public : démocratisation de l'imagerie numérique
  - Industrie audiovisuelle : mondialisation des programmes
  - Spécialisés : satellite, vidéosurveillance, imagerie médicale
- Stockage numérique
  - Forte diminution du coût par gigaoctet → démocratisation stockage
  - Croissance de la vitesse d'accès
- Transmission / diffusion numérique
  - Augmentation des débits
  - Forte diminution du coût → démocratisation diffusion / partage

→ Très grands volumes de données multimédia à exploiter

→ Distribution à grande échelle des sources de données

Besoins des utilisateurs

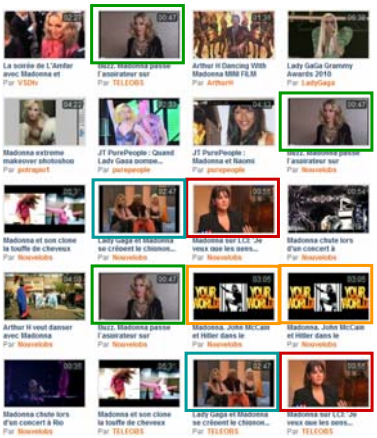
- Recherche par similarité
  - Requête par l'exemple
  - Recherche interactive
- Structurer une base multimédia
  - Partitionnement → résumé
  - Détection de liens
- Développer et appliquer des modèles pour détection / reconnaissance

Surveillance de flux vidéo

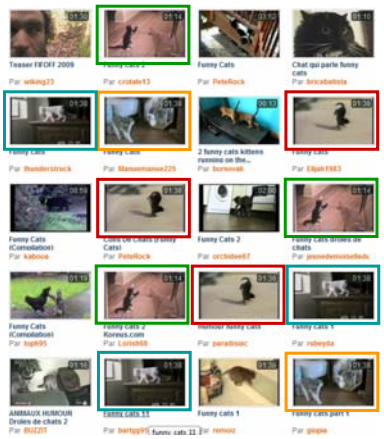
- Détection de copies vidéo : recherche par similarité
- « Copie » : version modifiée d'un contenu original (changements photométriques, géométriques, temporels, post-production)
- Exemple de volumétrie
  - Plus de 400 000* heures vidéo numérisée à l'INA
  - Plus de 400 flux audiovisuels accessibles en France (hors Internet)

[PCS09]

Structuration de collection vidéo



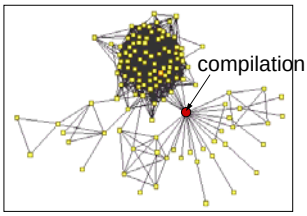
Source : DailyMotion



Source : DailyMotion

Structuration de collection vidéo

- Contexte grande archive de contenu professionnel (exemple : l'INA)
  - Segmentation, étiquetage (détection génériques)
  - Aide à l'annotation
  - Analyse de bruit média et de notoriété
  - Analyse des offres média
- Contexte site partage vidéo
  - Nettoyage (élimination doublons)
  - Structuration : sélection des représentants, identification vidéos caractéristiques
  - Mutualisation / filtrage de mots clés entre versions
  - Nouveaux outils navigation/visualisation



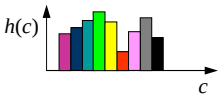
Fouille d'images satellitaires

- Découverte de motifs dans des collections d'images satellitaires
  - Méthodes non supervisées
    ou
  - Apprentissage interactif
- Exemple de volumétrie pour Pleiades (2 satellites)
  - 1 image : env. 1,6 Gpixel (20 x 20 km, résolution 50 cm)
  - Jusqu'à 1 000 000 km^2 par jour

© Google Earth

Description du contenu

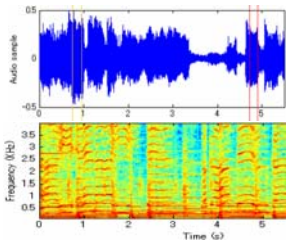
- Séparation et description de caractéristiques
  - Image : couleurs, textures, formes
  - Vidéo : couleurs, textures, formes, mouvement
- Identification de « composantes »
  - Image : régions homogènes, points d'intérêt, configurations ; objets, visages, ...
  - Vidéo : *shots*, régions à mouvement cohérent; scènes, personnes, objets, …

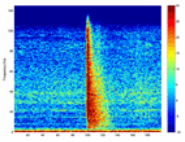


http://www-rocq.inria.fr/imedia

Description du contenu

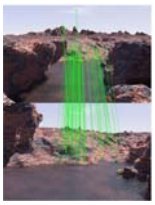
- Séparation et description de caractéristiques
  - Audio : temporelles, spectrales
  - Description résumée ou séquence de descriptions
- Identification de « composantes »
  - Audio : séparation/segmentation voix/musique/bruit, événements sonores (exemples : but, explosion)





Évaluation de la similarité

- Dimension / complexité des descriptions
  - Vecteurs de grande dimension (souvent > 50)
  - Graphes (ex. : image ~ graphe de régions)
- Complexité des opérations de comparaison
  - S'appliquent à des objets composés
    - Ensemble ↔ (sous-)ensemble
    - Séquence ↔ (sous-)séquence
  - Nécessitent la mise en correspondance
    - Images, vidéo : configurations de points d'intérêt, avec transformations géométriques
    - Audio : séquences, avec dilatation/compression

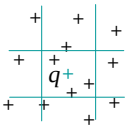


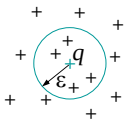
Plan de la présentation

1. Recherche et fouille de données multimédia : exemples
2. **Typologie des besoins et nature des problèmes**
3. Passage à l'échelle de la recherche par similarité
4. Passage à l'échelle de la fouille
5. Apports et implications de la distribution

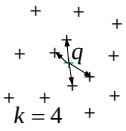
Recherche par l'exemple

- Nature du critère à satisfaire ($\mathbf{q}$ est la requête)
  - Recherche par intervalle (*range query*)
    $Range_{\mathbf{r}}(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} \mid \forall i, |x_i - q_i| \leq r_i\}$
  - Recherche dans un rayon (ε-*sphere query* ou *within distance query*)
    $Sphere_{\varepsilon}(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} \mid d(\mathbf{x},\mathbf{q}) \leq \varepsilon\}$
  - Recherche des *k* plus proches voisins (*k*ppv, *k*NN)
    $kNN(\mathbf{q}) = \{\mathbf{x} \in \mathcal{D} \mid |kNN(\mathbf{q})| = k \wedge \forall \mathbf{y} \in \mathcal{D} - kNN(\mathbf{q}), d(\mathbf{y},\mathbf{q}) \geq d(\mathbf{x},q)\}$





- *N* données ⇒ complexité *O*(*N*) ?

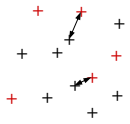


Jointure par similarité

- Jointure avec seuil de distance θ, ensembles $\mathcal{D}_1, \mathcal{D}_2$ avec N_1 et respectivement N_2 éléments

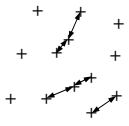
  $K_\theta = \{(\mathbf{x},\mathbf{y}) \mid \mathbf{x} \in \mathcal{D}_1, \mathbf{y} \in \mathcal{D}_2, d(\mathbf{x},\mathbf{y}) \leq \theta\}$

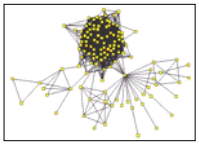
  ⇒ complexité $O(N_1 \times N_2)$?



- Auto-jointure : $\mathcal{D}_1 \equiv \mathcal{D}_2$

  ⇒ complexité $O(N^2)$?

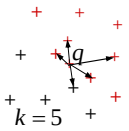




(variante sans seuil de distance : jointure *top-k*)

Décision

- *k* **p**lus **p**roches **v**oisins
  1. Trouver les *k*ppv
  2. Vote majoritaire, avec ou sans seuil d'ambiguïté



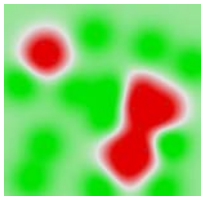
- Machine à noyaux
  - Fonction de décision

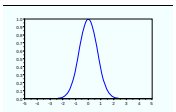
$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* \, y_i \, K(\mathbf{x}, \mathbf{x}_i) + b^*$$

⇒ complexité *O*(*N*) ?

or, noyaux locaux ⇒ seuls comptent les voisins dans un rayon

$$K(-u) = K(u), \quad \int_{-\infty}^{\infty} K(u)\, du = 1$$





Apprentissage

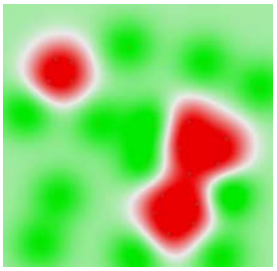
- Supervisé, ν-SVM
  - Problème de minimisation

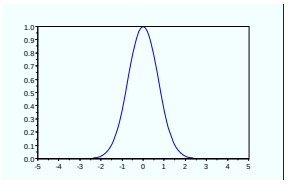
$$\begin{cases} \min\limits_{\boldsymbol{\alpha}} \sum\limits_{i,j=1}^{N} \alpha_i \, \alpha_j \, y_i \, y_j \, K(\mathbf{x}_i, \mathbf{x}_j) \\ 1/N \geq \alpha_i \geq 0, \quad 1 \leq i \leq N \\ \sum_{i=1}^{N} \alpha_i \, y_i = 0 \\ \sum_{i=1}^{N} \alpha_i \geq \nu \end{cases}$$

⇒ complexité $O(N^2)$?

or, noyaux locaux ⇒ seuls comptent les voisins dans un rayon

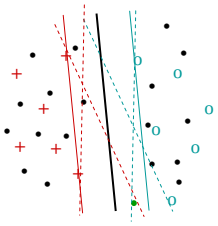
$$K(-u) = K(u), \quad \int_{-\infty}^{\infty} K(u)\, du = 1$$



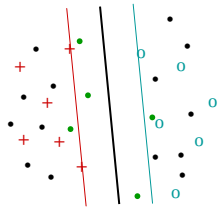


Apprentissage

- Semi-supervisé : tenir compte aussi des données non-étiquetées
  - Transduction avec SVM : trouver l'étiquetage qui maximise la marge
  - Quelles données non-étiquetées peuvent avoir le plus fort impact ?



- Actif : sélectionner pour étiquetage les données « les plus informatives »
  - Approche classique : choisir les données les plus ambiguës (+ non redondantes)



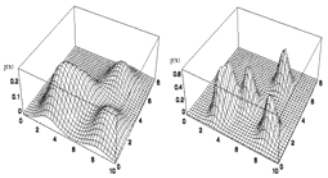
→ Trouver les données proches de la frontière !

⇒ complexité *O*(*N*) ? (*N* données non-étiquetées)

Estimation de densité

- Noyaux *K* centrés sur $\mathbf{x}_i, 1 \le i \le N$
  - Fonction de densité

$$f(\mathbf{x}) \propto \sum_{i=1}^{N} K(\mathbf{x}, \mathbf{x}_i)$$



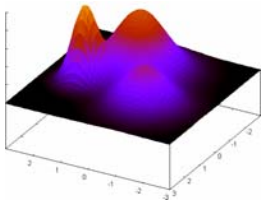
© Duda, Hart, Stork 2001

⇒ complexité *O*(*N*) ?

or, noyaux locaux ⇒ seuls comptent les voisins dans un rayon

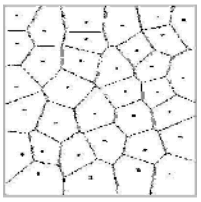
$$K(-u) = K(u), \quad \int_{-\infty}^{\infty} K(u)\, du = 1$$

(valable également pour modèles paramétriques avec *N* composantes)

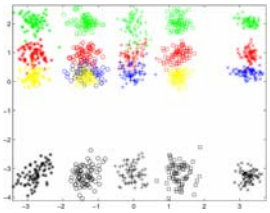


Quantification, classification

- Nombre fixé *c* de centres, *N* données
  ⇒ complexité $O(c \times N)$



- Nombre de centres dépendant des données
  ⇒ ~ $N^{1/3}$ centres
  ⇒ complexité $O(N^{4/3})$, $O(N \log N)$ ou $O(N^2)$?
  (selon la méthode)



Une typologie des problèmes

1. Requête de même nature que les objets de la base
   - Recherche par l'exemple, décision, estimation de densité, etc.
   - Objectif : $O(N) \rightarrow O(\log N)$ ou $O(1)$
2. Regroupement par similarité
   - Jointure par similarité, apprentissage supervisé à noyaux, *N-body problems* [RLWG09], quantification, classification
   - Objectif : $O(N^2) \rightarrow O(N)$ ou $O(N \log N)$
3. Requête = frontière
   - Apprentissage actif, apprentissage semi-supervisé
   - Objectif : $O(N) \rightarrow O(\log N)$ ou $O(1)$

- Remarque générale : ordre de complexité ≥ taille résultat !

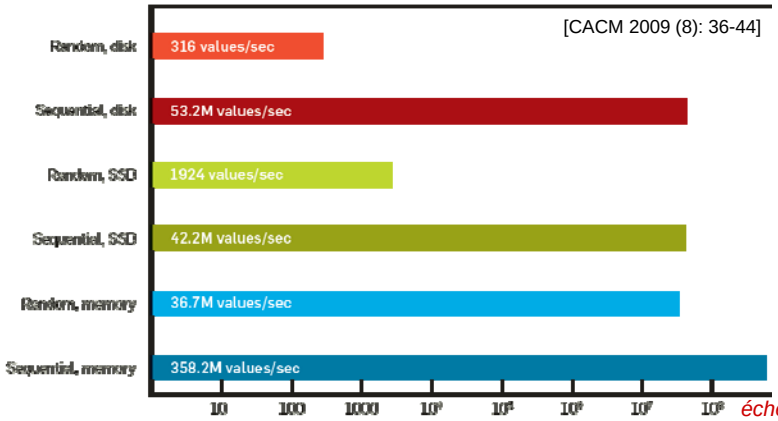
Passage à l'échelle

- Volume de données
  - Coûts à considérer
    - Calculs de distances (ou noyaux)
    - Transferts E/S (coût dominant pour volumes très élevés)
  - → Minimiser : coût total, temps de réponse
- Distribution (solution et problème)
  - Variable spécifique : nombre de sources de données
  - → Minimiser : coût total, temps de réponse, trafic réseau

→ Éviter les comparaisons sans potentiel
→ Distribution : maximiser le parallélisme utile

Hiérarchie de stockage

[CACM 2009 (8): 36-44]



échelle logarithmique !

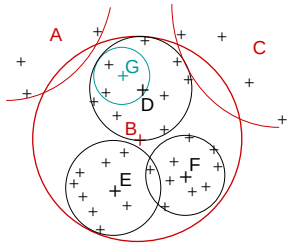
* Disk tests were carried out on a freshly booted machine (a Windows 2003 server with 64GB RAM and eight 15,000RPM SAS disks in RAID5 configuration) to eliminate the effect of operating-system disk caching. SSD test used a latest generation Intel high-performance SATA SSD.

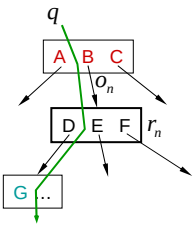
Principe général des approches

(*N* données ⇒ recherche exhaustive O(*N*))

- Hypothèses
  1. Seules les données « proches » sont utiles
  2. Peu de données sont proches
  3. Les données proches sont bien plus proches que les autres
- Illustration : regroupement hiérarchique
  - 1 nœud → 1 page disque
  - Parcours de l'arbre en partant de la racine
  - Idéalement O(log *N*) nœuds traversés
    ⇒ O(log *N*) calculs de distance
    ⇒ O(log *N*) échanges E/S

⇒ 2 *N* données → coût log *N* + cst. !





Plan de la présentation

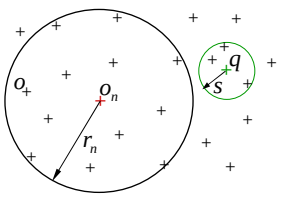
1. Recherche et fouille de données multimédia : exemples
2. Typologie des besoins et nature des problèmes
3. **Passage à l'échelle de la recherche par similarité**
4. Passage à l'échelle de la fouille
5. Apports et implications de la distribution

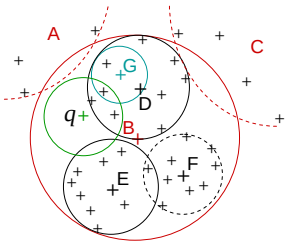
Typologie des structures d'index

- Selon la nature des données
  - Espace vectoriel ⇒ coordonnées, centres de gravité, densités de probabilité, etc.
  - Espace métrique non vectoriel ⇒ distances seulement
- Selon la méthode de réduction de l'espace de recherche
  - Partitionnement de l'espace : k-d-B-*tree*, LSD-*tree*...
  - Partitionnement des données : R-*tree*, SR-*tree*, M-*tree*...
  - Filtrage des données : VA-*file*, LPC-*file*…
  - Mixtes

Illustration : arbre métrique

- Regroupement hiérarchique des données basé sur la métrique *d*
- Rejet de sous-arbre : critère issu de l'inégalité triangulaire
  - Sous-arbre $N(o_n)$: objets *o* tels que $d(o,o_n) \leq r_n$
  - Requête *q* de rayon *s*
  - $d(q,o_n) > r_n + s \Rightarrow d(q,o) > r_n + s - r_n = s \Rightarrow$ rejet sous-arbre $N(o_n)$

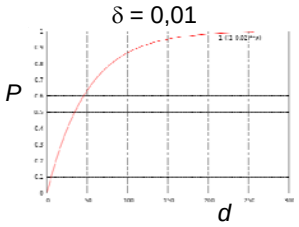




Malédiction de la dimension

(curse of dimensionality)

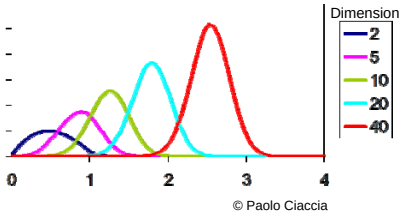
- *N* données $\in [0, 1]^d$, distribution uniforme, division de l'espace
  - ⇒ Nombre de cellules augmente exponentiellement avec *d*
  - ⇒ Nombre de données par cellule diminue exponentiellement avec *d*, nombreuses cellules sont vides
- Données proches des hypersurfaces externes : probabilité d'être à moins de δ de la frontière est $1- (1 - 2\ \delta)^d$
  - ⇒ 1 requête coupe de nombreuses cellules
  - ⇒ Séparation difficile entre cellules



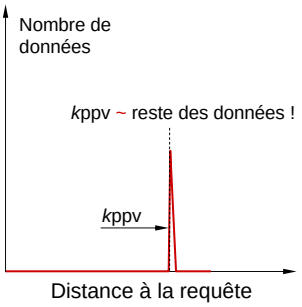
Malédiction de la dimension

- Concentration des mesures : avec l'augmentation de la dimension, la variance de la distribution des distances diminue
  - ⇒ Pour partitionnement des données : plus d'intersections entre partitions ⇒ intérêt d'un index diminue \ recherche exhaustive !
  - ⇒ Cas extrême : *k*ppv encore significatifs ? Classification significative ?

Données uniformes : variation de la distribution des distances avec la dimension



© Paolo Ciaccia



Recherche approximative ?

- L'approximation convient très bien à certaines applications
  - Illustration : recherche par l'exemple (similarité calculée ↔ perçue)
  - *k*ppv approximatifs : soit $r_k = \max_{\mathbf{x} \in kNN(\mathbf{q})} d(\mathbf{x},\mathbf{q})$ alors

    $kNN_{a(\beta)}(\mathbf{q}) = \left\{ \mathbf{x} \in \mathcal{D} \,\middle|\, |kNN_{a(\beta)}(\mathbf{q})| = k \wedge \forall \mathbf{x} \in kNN_{a(\beta)}(\mathbf{q}),\ d(\mathbf{x},\mathbf{q}) \le (1+\beta) r_k \right\}$
  - Avantage : gain potentiellement très significatif en efficacité !
- Structures d'index ↔ recherche approximative
  - Définies pour recherche exacte, admettant recherche approximative
  - Dédiées à la recherche approximative : LSH, PP-index, SASH...

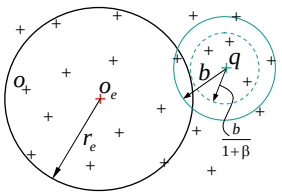
Recherche approximative

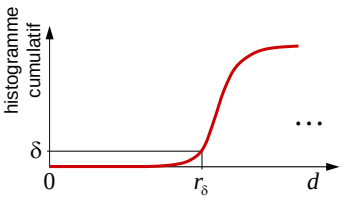
- *k*ppv approximatifs : arrêt précoce car toute amélioration limitée à β est inutile
- Bases difficiles (faible variance de la distribution des distances)
  - ppv sont éloignés de la requête
  - Nombreuses partitions en intersection avec la requête, mais intersections (presque) vides

  ⇒ Estimer $r_{q,\delta}$ tel que

  $P\left\{ \exists o : d(q,o) \le r_{q,\delta} \right\} \le \delta$

  arrêt dès que $b \le (1+\beta)\, r_{q,\delta}$





Locality Sensitive Hashing *(LSH)*

(hachage sensible à la similarité)

- Domaine $\mathcal{D}$, ensemble de clés $\mathcal{Q}$, métrique $d_{\mathcal{H}} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}^+$
- $\mathcal{H} = \{h : \mathcal{D} \to \mathcal{Q}\}$ fonctions de hachage $(r, \varepsilon, p_1, p_2)$-sensibles si

$$\forall x, y \in \mathcal{D}, \quad d_{\mathcal{H}}(x,y) \le r \Rightarrow P_{h \in \mathcal{H}}(h(x) = h(y)) \ge p_1$$
$$d_{\mathcal{H}}(x,y) > r \Rightarrow P_{h \in \mathcal{H}}(h(x) = h(y)) < p_2$$

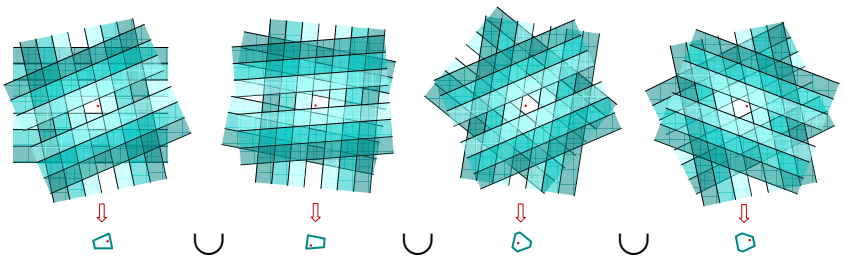
(pour $r, \varepsilon > 0$, $p_1 > p_2 > 0$)

- Recherche par similarité avec LSH
  1. Hachage de chaque objet de la base et stockage ensemble (même case) des objets de même clé
  2. Hachage de l'objet-requête, lecture de la case associée à la clé
  3. Retour comme réponse des objets de cette case (avec ou sans sélection ultérieure par calcul des distances)

vertigo 2 juillet 2010 Assises GDR I3 31 Cédric

LSH : utilisation pratique

1. Concaténation de fonctions = intersection des cases issues des différentes fonctions ($\to$ 1 table de hachage) $\Rightarrow$ meilleure précision (meilleure sélectivité, moins de faux positifs)
2. Plusieurs tables de hachage indépendantes, réunion des cases issues de tables différentes dans lesquelles la requête se trouve $\Rightarrow$ meilleur rappel (moins de faux négatifs)

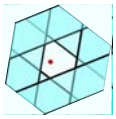


vertigo 2 juillet 2010 Assises GDR I3 32 Cédric

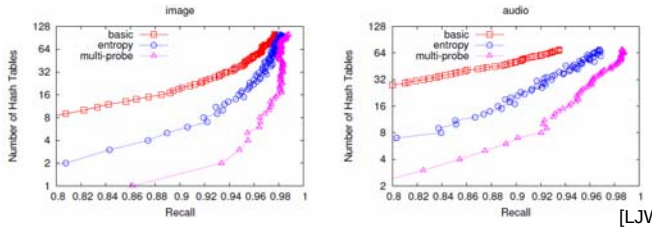
Multi-probe *LSH*

- Bon rappel avec LSH ⇐ nombreuses tables : occupation mémoire

- *Multi-probe* [LJW07] : dans chaque table échantillonner aussi les cases voisines, avec probabilité ∝ proximité



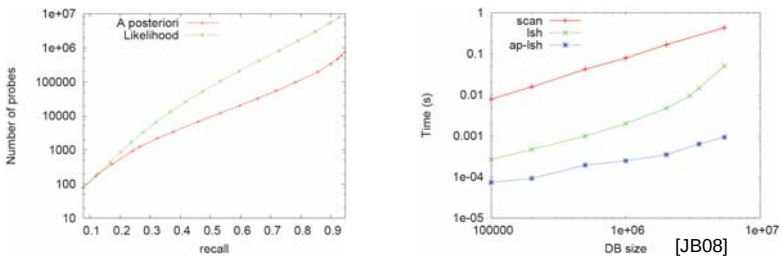
- Comparaison



[LJW07]

A posteriori multi-probe *LSH*

- *Multi-probe* LSH : voisinage d'une case B définie par les fonctions de hachage, indépendamment des données !
- *A posteriori multi-probe* LSH [JB08]
  - Estimer probabilité *a posteriori* de trouver des réponses pertinentes dans chaque case voisine
  - Choisir le minimum de cases telles que Somme de leurs probabilités > seuil de rappel α



[JB08]

LSH : conclusion

- Avantages
  - Complexité recherche ~ *O*(1)
  - Métriques variées (entre vecteurs, ensembles, séquences…)
  - Adaptable au fonctionnement avec stockage externe
  - Accès optimisé par rapport à la distribution (*a posteriori multi-probe*)
  - Hachage peut être optimisé par rapport à une distribution conditionnelle (hachage « sémantique », hachage spectral,…)
- Difficultés
  - Recherche approximative
  - Peu adapté aux distributions très non uniformes
  - Filtrage peu efficace si nombre fonctions de hachage << dimension intrinsèque des données
  - Coût spatial très élevé si dimension intrinsèque élevée

Recherche : conclusion

- Très nombreuses structures d'index [Sam06]
- Études comparatives systématiques et compréhensives ?
  - Vaste ensemble de structures d'index
  - Grandes bases diverses, distributions variées (et représentatives ?)
  - Plusieurs types de recherche exacte et approximative

Dimension faible, données uniformes ou non	Dimension élevée, données non uniformes	Dimension élevée, données uniformes
Cas simple, résolu	Cas difficile, intéressant	Cas peu pertinent ?

difficulté → *difficulté* →

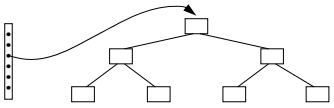
- Autres critères à prendre en compte
  - Fonctionnement avec un haut débit de requêtes
  - Facilité d'exécution parallèle

Plan de la présentation

1. Recherche et fouille de données multimédia : exemples
2. Typologie des besoins et nature des problèmes
3. Passage à l'échelle de la recherche par similarité
4. **Passage à l'échelle de la fouille**
5. Apports et implications de la distribution

Jointure par similarité

- Application directe de la recherche par similarité
  - Chaque objet devient une requête
  - Arbre → $O(N \log N)$; LSH → $O(N)$



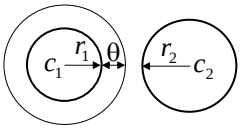
- Regrouper le travail pour objets-requête similaires
  - Parcours en parallèle de deux arbres → $O(N)$
    mais construction d'arbre $O(N \log N)$!
  - Partitionnement par LSH et jointure intra-cases
    méthode approximative !

(rappel : ordre de complexité ≥ taille résultat !)

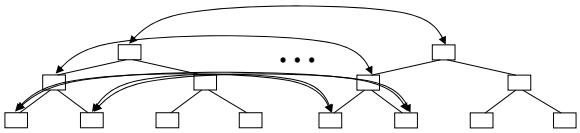
Algorithme à base de deux arbres

- Idée : parcourir en parallèle deux arbres de recherche
  - Test d'exclusion de 2 sous-arbres

$$d(c_1, c_2) > r_1 + r_1 + \theta$$



- Jointure : algorithme doublement récursif avec 2 arbres

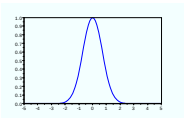


- Auto-jointure : algorithme doublement récursif avec 1 seul arbre

Algorithme à base de deux arbres

- *N-body problems* [RLWG09]

$$\forall \mathbf{x}_i,\ f(\mathbf{x}_i) = \sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j)$$



- Principe : approximation par algorithme doublement récursif
  - Approximation : différences finies (Gregory-Newton)
  - Borne sur l'approximation ↔ borne sur la distance
  - À chaque étape, les calculs de noyau sont faits s'il n'y a pas exclusion des noeuds

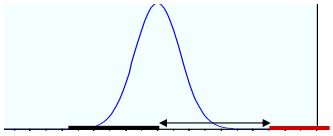


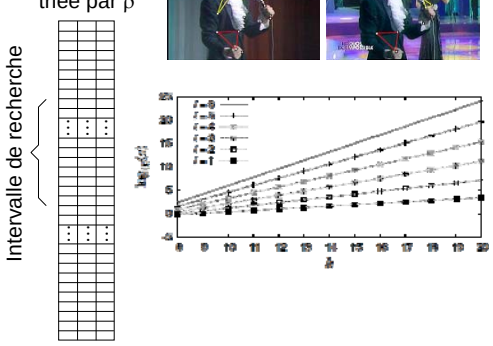
Illustration LSH pour jointure

- Quantification de l'espace de description en cellules → chaque case est définie par un triplet de cellules [PCS09]
- Triplets de points définis par les *k*ppv dans l'*image*
- Accélération théorique

$$a = \frac{C^l_{2^h}}{\left(C^l_L\right)^2}$$

- Information géométrique pour plus de sélectivité

$$\rho = \frac{d\left(q, 1^{\text{er}}\ \text{ppv}\right)}{d\left(q, 2^{\text{nd}}\ \text{ppv}\right)}$$



Jointure : conclusion

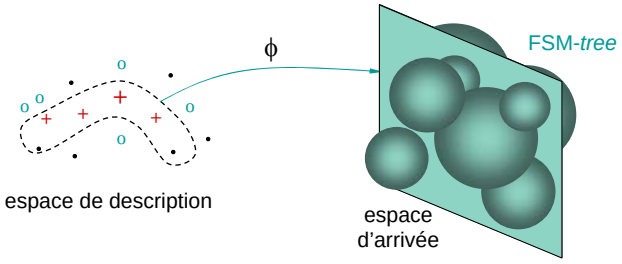
- Nombreux travaux sur le passage à l'échelle de la classification automatique, peu sur la jointure par similarité, *N-body problems*
- Diverses méthodes exploitent des spécificités des données (distribution particulière, données creuses)
- Extension possible aux ordres supérieurs
- Tenir compte dès le départ de la totalité des critères définissant la similarité (exemple : configurations géométriques)

Requête = frontière

- Potentiel : apprentissage actif, apprentissage semi-supervisé
- Données les plus proches de la frontière
  - Indexation dans l'espace de description : la frontière est complexe
    - Classification hiérarchique (méthode approximative)
    - Index spatial (méthode exacte)
  - *Feature space* M-*tree* : dans l'espace d'arrivée, frontière = hyperplan
- Problème lié : données les plus éloignées de la frontière
  - *Kernel* VA-*file* : construction fichier d'approximation (VA-*file*) dans l'espace d'arrivée
  - *Kernel indexer* (KDX) : indexation des données sur une sphère (dans l'espace d'arrivée) pour noyaux $\forall \mathbf{x}, K(\mathbf{x}, \mathbf{x}) = \text{cst}$.

FSM-tree et requête hyperplan

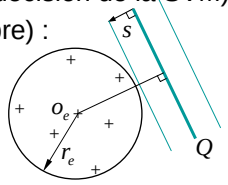
- Méthode
  - Discrimination par machine à noyaux (SVM, KDA…)
  - Construction M-*tree* dans l'espace d'arrivée
  - Frontière = hyperplan → requête *k*NN avec un hyperplan
  - *Kernel trick* : calculs dans l'espace de description !
  - Large classe de noyaux conditionnellement définis positifs



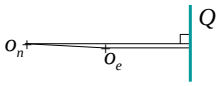
FSM-tree et requête hyperplan

- Distance à l'hyperplan : $d(Q,o_e)=\frac{\left|\sum_i \alpha_i\, y_i\, K(o_e,x_i)+b\right|}{\sqrt{\sum_{i,j}\alpha_i\,\alpha_j\, y_i\, y_j\, K(x_i,x_j)}}$

  ($f(o_e)=\sum_i \alpha_i\, y_i\, K(o_e,x_i)+b$ étant la fonction de décision de la SVM)
- Principe d'élagage (test de rejet d'un sous-arbre) :
  - Si $d(Q,o_e)>r_e+s$ alors le nœud n'est pas conservé pour exploration ultérieure



- Comment éviter plus de calculs de distances :

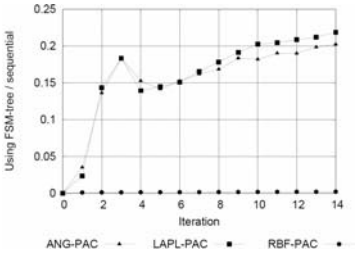


$d(Q,o_e)+d(o_e,o_n)\geq d(Q,o_n)$ mais $d(Q,o_e)+d(Q,o_n)\not\geq d(o_e,o_n)$

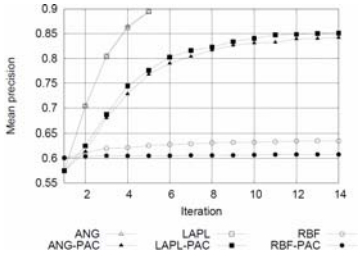
$\Rightarrow d(Q,o_e)\geq d(Q,o_n)-d(o_e,o_n)$ ($d(Q,o_e)\not\geq |d(Q,o_n)-d(o_e,o_n)|$), donc si $d(Q,o_n)-d(o_e,o_n)>r_e+s$ alors $d(Q,o_e)>r_e+s$

FSM-tree : évaluation

- Base : *Amsterdam Library of Object Images* (110 000 images)



Calculs de distance avec PAC-NN + FSM-*tree* vs. recherche exhaustive



Évolution précision avec recherche exacte et avec PAC-NN + FSM-*tree*

Requête frontière : conclusion

- Difficultés spécifiques
  - Espace de description : frontière complexe
  - Espace d'arrivée : concentration des mesures → moindre sélectivité
  - Requête frontière moins sélective que requête objet

    du moins dans l'espace des données !

→ Résultats médiocres de la recherche exacte avec index, la recherche approximative peut améliorer sensiblement l'efficacité

- Problème encore largement ouvert !

Plan de la présentation

1. Recherche et fouille de données multimédia : exemples
2. Typologie des besoins et nature des problèmes
3. Passage à l'échelle de la recherche par similarité
4. Passage à l'échelle de la fouille
5. **Apports et implications de la distribution**

Distribution

- Pourquoi ?
  - Multiplier les ressources pour traiter des problèmes difficiles
    - Puissance de calcul
    - Capacité mémoire !
  - Les données sont distribuées (et doivent le rester)
- Cas extrêmes de systèmes distribués
  - *Cluster* : architecture bien identifiée, réseau haut débit, contrôle centralisé
  - Pair-à-pair : inhomogène, autonomie connexion/déconnexion, contrôle distribué

Quelques faits

- Hiérarchie des bandes passantes
  - Mémoire vive ~ 2,5 – 20 Go/s
  - Réseau (très) local ~ 125 Mo/s – 1,25 Go/s
  - Ligne dédiée haut débit ~ 1,25 Go/s
  - Via Internet ~ 0,1 – 1 Mo/s
  - Disque (accès séquentiel) ~ 50 Mo/s
- Augmentation bande passante ~ (réduction latence)^2

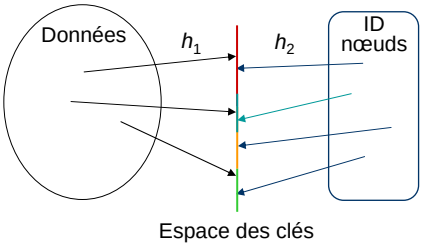
Recherche distribuée

- Quelques questions
  1. Comment localiser des données dans un système distribué ?
     → Index distribué, à construire et maintenir !
  2. Comment équilibrer la charge ?
     → En tenir compte dans le choix de la structure d'index
  3. Comment maximiser le parallélisme ?
     → En tenir compte dans le choix de la structure d'index
- Quelques exigences
  - Disponibilité : départ de pairs, pannes locales de réseau
  - Fraîcheur de l'index : arrivées / départs pairs, données
  - Qualité de recherche : temps de réponse, précision, couverture
  - Efficacité système : utilisation parcimonieuse des ressources, coût maintenance / coût recherches

Distributed Hash Tables

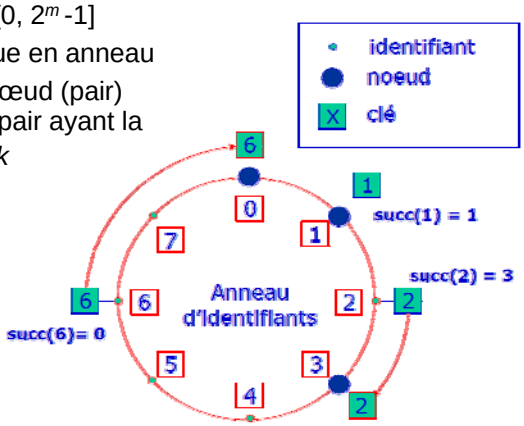
- Hachage consistant : arrivée / départ d'un nœud engendre peu de changements dans les associations clés ↔ nœuds
- Double hachage
  - Donnée → clé
  - ID nœud → clé

  } même espace de clés !



Chord

- Solution DHT
  - Clés sur *m* bits → [0, 2^m -1]
  - Organisation logique en anneau
  - Clé *k* affectée au nœud (pair) **successeur**(*k*) = pair ayant la clé la plus faible ≥ *k*



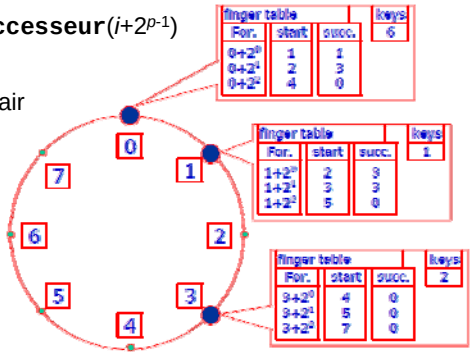
© Dan Vodislav

Chord *: recherche*

- Naïve : chaque pair (*p* au total) connaît uniquement son successeur et lui transmet la clé *k* s'il ne la stocke pas → O(*p*)
- Optimisée : chaque pair *i* a une table
  - Entrée *j* = adresse pair **successeur**($i+2^{p-1}$)

  → Taille table *O*(log *p*)
  - Chaque pair transmet au pair (qu'il connaît) de clé aussi élevée que possible ≤ *k*

  → Recherche *O*(log *p*)



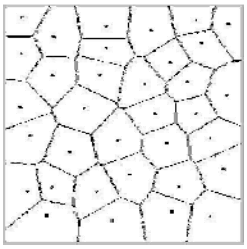
© Dan Vodislav

M-Chord

- Principe [NZ06]
  - Linéarisation de l'espace métrique (variante *iDistance*)
  - Distribution sur un support *Chord*
- Linéarisation : partitionnement des données en *n* groupes C_i de « centres » p_i , calcul clé

$$iDist(x) = d(p_i, x) + i \cdot c$$

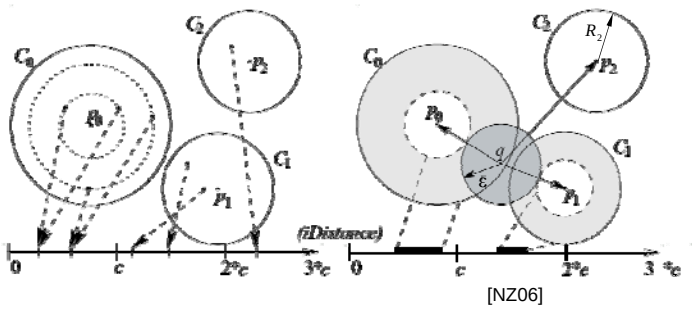
  - Pour c assez grand, les clés des points de C_i sont dans $[i \cdot c,\ (i+1) \cdot c]$
  - Les données sont stockées dans un B⁺-*tree* suivant les clés



M-Chord : iDistance

- Recherche dans un rayon : sélection clusters C_i pour lesquels $d(p_i, q) - \varepsilon \le R_i$, calcul de $d(x, q)$ pour tout point des intervalles

$$\left[i \cdot c + d(p_i, q) - \varepsilon,\ i \cdot c + \min\left\{ R_i, d(p_i, q) + \varepsilon \right\} \right]$$



[NZ06]

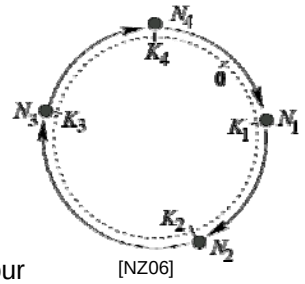
M-Chord : support Chord

- Transposition des clés *iDistance* dans le domaine *Chord* [0, 2^m) avec *h*

$$mchord(x) = h\left[d(p_i, x) + i \cdot c \right]$$

- Distribution sur le support *Chord*
- Stockage des distances avec les objets pour filtrage supplémentaire : $d(x,q)$ n'est pas calculée si

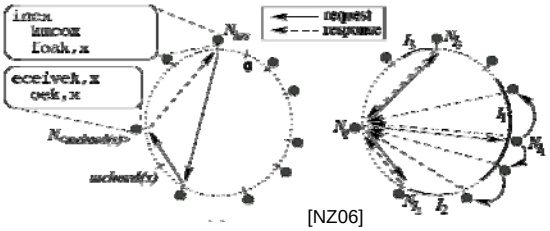
$$\left| d(p_i, x) - d(p_i, q) \right| > \varepsilon$$



[NZ06]

M-Chord : construction, recherche

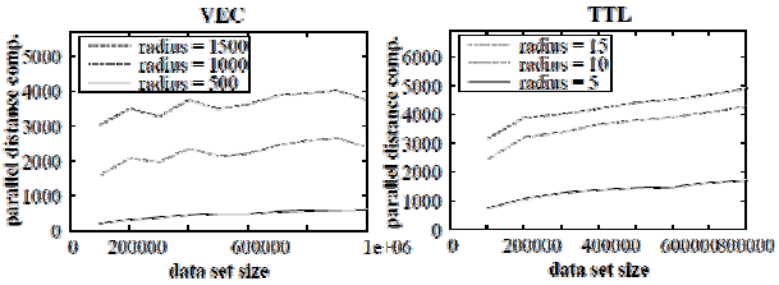
- Construction
  - Initialisation : sur un seul nœud, on trouve les centres et la fonction *h*
  - Activation d'un nouveau nœud : *split* à partir d'un nœud actif
- Insertion donnée : calcul de la clé et insertion *Chord*
- Recherche dans un rayon : sélection intervalles avec *iDistance*, identification nœuds suivant *Chord*
- Recherche *k*NN
  1. Trouver (heuristique) *k* objets proches
  2. ε-*query* avec le rayon du $k^{ième}$ objet



[NZ06]

M-Chord : performances

- Évaluation : 200 000 objets, 300 machines sur réseau local…
  1. VEC : objets = vecteurs de dimension 45 ; distance euclidienne
  2. TTL : objets = titres, sous-titres livres et périodiques ; distance *edit*



[NZ06]

Recherche distribuée : conclusion

- $O(\log p)$ x latence réseau ↔ interactivité ?

→ tables de routage $O(1)$ en temps ⇒ espace $O(p)$

- 10^6 pairs → ~ 10 Mo, c'est peu !

  mais en pair-à-pair coût excessif de la maintenance des tables !

  → Approche exploitable uniquement si arrivées / départs rares

- Couverture (rappel) dans un contexte pair-à-pair ?

→ Pair-à-pair : inutile de s'intéresser à la recherche exacte !

Distribution : conclusion

- Recherche interactive
  - Peu compatible avec un environnement pair-à-pair ?
  - Réalisable si arrivées / départs rares !
- Fouille : pas de contre-indication
- Évolution des approches
  - Cluster/*grid* : un processeur qui se libère reçoit des données et un traitement (→ transport des données)
  - *Cloud* : données persistantes traitées localement ; réplication pour garantir la persistance

Conclusion générale

- Passage à l'échelle : la prise de conscience est récente dans une grande partie de la communauté

→ Emploi souvent peu critique de méthodes « prises sur l'étagère »

→ Résultats rarement conformes aux attentes

→ Focalisation sur un ensemble très restreint de problèmes

- Résultats théoriques ? Oui, sous hypothèses très restreintes…

→ Évaluation sur « données représentatives »

→ Obtenir de telles données

→ Expérimentalement : caractéristiques données ↔ résultats méthodes

⇒ Compréhension des liens, résultats théoriques

Conclusion générale

- Principaux besoins de la communauté à court terme
  - Données : corpus de très grande taille (mais pb. de droits, d'accès…)
  - Campagnes d'évaluation : critères incluant le passage à l'échelle
- Quelques initiatives
  - En Europe : CHORUS2 (action de coordination, WP Évaluation)
  - Internationale : TRECVID commence à y être sensible

Merci pour votre attention !

Bibliographie

[BGP10] P. Bruneau, M. Gelgon, and F. Picarougne. Parsimonious reduction of Gaussian mixture models with a variational-Bayes approach. *Pattern Recognition*, vol. 43, pp. 850-858, 2010.

[DII04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on *p*-stable distributions. In SCG'04: *Proc. 20th annual Symposium on Computational Geometry*, pages 253–262, New York, NY, USA, 2004. ACM.

[KKZ09] H. Kriegel, P. Kröger, and A. Zimek. Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Trans. Knowl. Discov. Data* 3, 1 (Mar. 2009), 1-58.

[NZ06] D. Novak, and P. Zezula. M-Chord: a scalable distributed similarity search structure. *Proc. InfoScale'06*, pp. 19–28, 2006.

[PCS09] S. Poullot, M. Crucianu, and S. Satoh. Indexing local configurations of features for scalable content-based video copy detection. In LS-MMRM: *1st Workshop on Large-Scale Multimedia Retrieval and Mining*, with *ACM Multimedia 2009*, pages 43–50, New York, NY, USA, 2009. ACM.

[RLWG09] P. Ram, D. Lee, W. March, A. Gray. Linear-time Algorithms for Pairwise Statistical Problems. In *NIPS 2009*, pp. 1527-1535.

Bibliographie

[Sam06] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, San Francisco, CA, USA, 993 p. 2006.

[SZ03] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proc. 9th IEEE Intl. Conf. on Computer Vision* (ICCV'03), pp. 1470-1477, Washington, DC, USA, 2003. IEEE Computer Society.