La recherche des plus proches voisins

Bibliographie Scientifique

Projet Transversal 2013-2014

Orange – Polytech'Nantes

Fabien Richard Valentin Proust

Février 2014

Introduction

Une partie du programme que nous avons à réaliser pour Orange Labs dans le cadre de notre projet transversal, consiste à mettre en valeur le plus proche voisin pour l'ensemble des films. C'est pourquoi il est important de comparer et d'étudier les différentes techniques de recherche de voisins les plus proches qui existent pour pouvoir choisir celle qui sera la plus adaptée à notre cas. Pour cela, nous commencerons par définir ce que doit faire l'algorithme et nous détaillerons ses différentes applications. Puis nous soulignerons l'intérêt que cette étude a pour notre projet. Enfin, nous étudierons les méthodes de recherche exacte puis les méthodes de recherche approchée.

Sommaire

I. Définition

La recherche du plus proche voisin est un problème d'optimisation pour trouver le point le plus proche d'un point donné.

Considérons :
- un espace *E* de dimension *D* ;
- un ensemble *A* de *N* points dans cet espace ;

La recherche du plus proche voisin consiste, étant donné un point *x* de *E* n'appartenant pas nécessairement à *A*, à déterminer quel est le point de *A* le plus proche de *x*.

II. Applications générales

La recherche de voisinage est utilisée dans de nombreux domaines :
- la reconnaissance de formes
- le clustering (ou regroupement en classes)
- l'approximation de fonctions
- la prédiction de séries temporelles et même les algorithmes de compression (recherche d'un groupe de données le plus proche possible du groupe de données à compresser pour minimiser l'apport d'information).
- détection de plagiat
- systèmes de recommandations

III. Intérêt pour notre projet

Les ingénieurs d'Orange Labs qui travaillent sur les algorithmes pour le système de recommandation du service de vidéo à la demande, veulent pouvoir visualiser, pour chaque film A, le film B qui lui est le plus proche. On peut imaginer que dans un système de recommandation, la première suggestion est la plus importante, puisque c'est celle-ci que l'utilisateur regardera en premier. Si elle n'est pas pertinente, alors il y a des chances que l'utilisateur ne regarde pas les suggestions suivantes. Cette étude bibliographique des techniques existantes pour le calcul des plus proches voisins nous est essentielle. En effet, le nombre important de données peut ralentir considérablement la génération des images.

IV. Algorithme naïf

Recherche linéaire des k plus proches voisins (Wikipédia) :

```
pour i allant de 1 à k
   mettre le point D[i] dans proches_voisins
fin pour
pour i allant de k+1 à N
   si la distance entre D[i] et x est inférieure à la distance d'un des points de proches_voisins à x
      supprimer de proches_voisins le point le plus éloigné de x
      mettre dans proches_voisins le point D[i]
   fin si
fin pour
proches_voisins contient les k plus proches voisins de x
```

V. Algorithmes efficaces

Il existe deux types d'algorithmes de recherche des plus proches voisins :
- les algorithmes de recherche exacte qui ne sont pas beaucoup plus performants que les algorithmes linéaires
- les algorithmes de recherche approchée qui acceptent de manquer des points voisins de la requête, mais qui, en contrepartie, sont beaucoup plus rapides.

Les méthodes que nous présenterons ici sont basées sur une structure de données appelée "arbre kd".

1. Présentation des arbres kd

Un arbre kd est une structure de donnée pour diviser de manière hiérarchique l'espace étudié. Ils sont des cas particuliers des arbres à partitionnement binaire de l'espace.
- Chaque nœud contient un point en dimension k.
- Chaque nœud non terminal divise l'espace en deux demi-espaces.
- Le noeud courant contient les points situés dans chacun des deux demi-espaces dans ses branches gauche et droite.

Afin d'avoir un arbre équilibré, le point inséré dans l'arbre à chaque étape est celui qui a la coordonnée médiane dans la direction considérée.

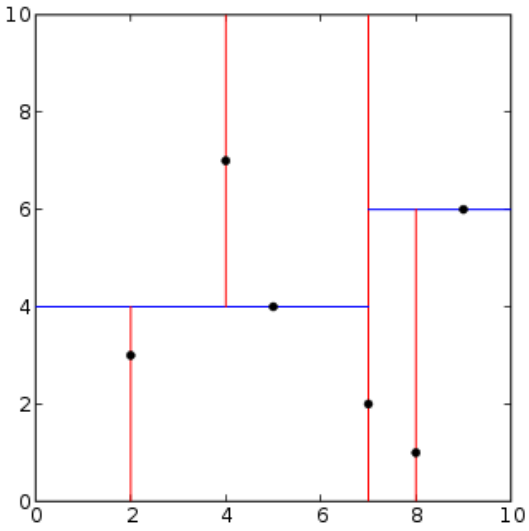

Fig. 1 Decomposition en arbre kd pour le jeu de données : (2,3), (5,4), (9,6), (4,7), (8,1), (7,2).

On en déduit l'arbre kd suivant:

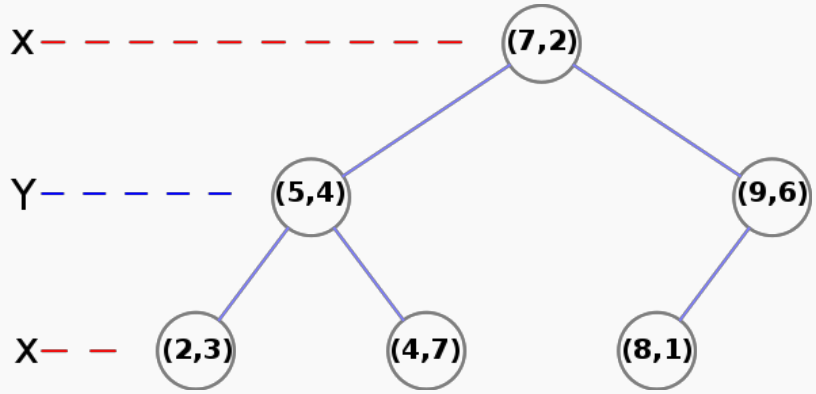

Fig. 2 Arbre kd résultant

Construction de l'arbre dans l'exemple ci-dessus :
- L'espace est divisé en deux parties par le point de coordonnées (7,2) selon un axe orthogonal à (Ox).
- Les deux sous espaces sont ensuite divisés par les points (5,4) et (9,6) selon un axe orthogonal à (Oy).
- Et ainsi de suite jusqu'à ce que tous les points divisent une partie de l'espace.

2. Méthode de recherche exacte des plus proches voisins basée sur les arbres kd.

Cette algorithme privilégie la profondeur (depth-first). En effet, on a plus de chance de trouver les plus proches voisins au début, et donc de limiter le nombre de branches a visiter ensuite.

Les étapes de recherche sont les suivantes :
- Recherche du noeud (c'est à dire la région) dans lequel se trouve le point choisi.
- Sélection des points de la feuille atteinte.(c'est à dire la région)
- A chaque noeud visité précédemment, on regarde si la branche non visitée peut contenir des voisins, si c'est le cas, on visite cette branche.

3. Méthodes approchées

L'Université de Maryland aux États-Unis a développé la librairie ANN (Approximate Nearest Neighbor Searching) qui permet d'utiliser différentes structures de données basées sur les arbres kd. Cette librairie permet l'utilisation de nombreuses distances (comme la distance Euclidienne, la distance de Manhattan ou encore la distance max) et permet de résoudre le problème des plus proches voisins de manière exacte ou approchée.

Source : http://www.cs.umd.edu/~mount/ANN/

3.1 Locality-Sensitive Hashing (LSH)

Cette méthode consiste à utiliser une fonction de hachage afin de regrouper les points qui sont dans des zones similaires dans des mêmes "paquets" (*buckets*). On détaillera ici cette méthode pour la similarité cosinus et la distance Euclidienne.

a. Similarité cosinus

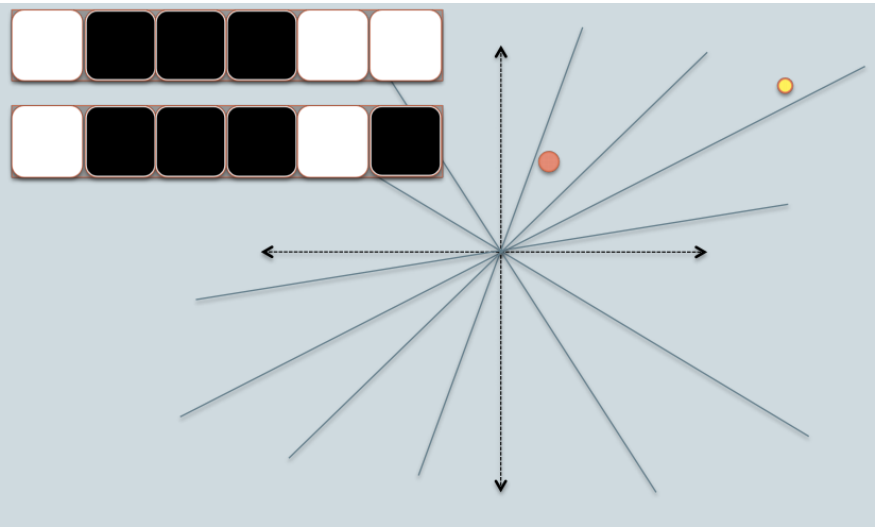

Fig. 3 Illustration de la clé de hachage

Sur la figure ci-dessus, les deux points (jaune et rouge) représentent deux données dans un espace à deux dimensions. L'objectif est de trouver leur similarité cosinus avec la méthode de LSH. Les axes gris sont pris au hasard uniformément. En fonction que le point se trouve au dessus ou en dessous de l'axe, il est marqué par un 1 (blanc), sinon par un 0 (noir) dans la signature – *sketch* – (les blocks en haut à gauche sur la figure). Pour être précis dans le calcul de la différence entre les deux points, un grand nombre d'axes est nécessaire puisque seuls les axes situés entre les deux points expliqueront la différence.

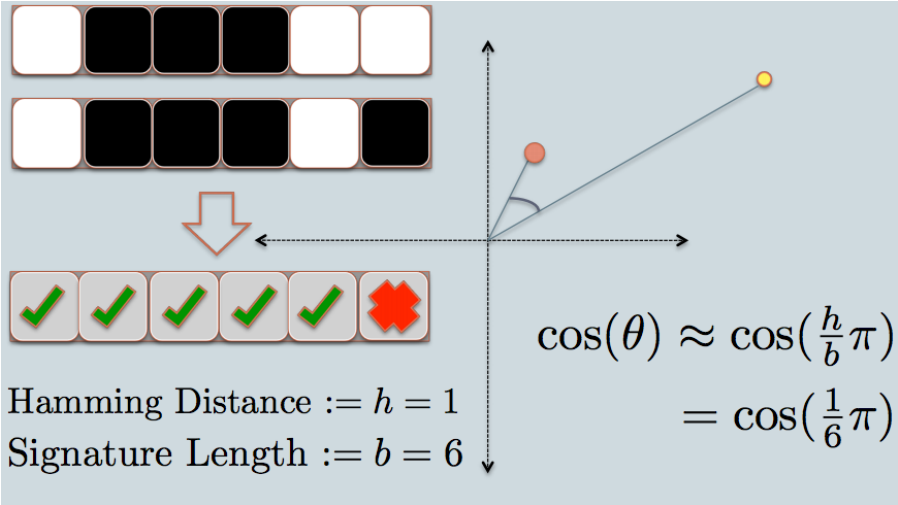

Fig. 4 Explication du calcul de la similarité cosinus

Dans cet exemple, 6 bits sont utilisés pour représenter les points. Ils correspondent au hachage LSH des données de départ. Les points qui ont une clé de hachage identique ont une forte probabilité de se trouver dans la même région de l'espace de recherche. La similarité cosinus est calculée en faisant :

$cos(\theta) \approx cos(h/b * \pi)$

avec h la distance de Hamming entre les deux clés (nombre de bits différents, ici 1)
et b la longueur de la signature (ici 6 bits)

On peut alors trouver les plus proches voisins d'un point de manière plus rapide en ne cherchant que parmi les points qui ont des clés de hachage similaires. La précision du résultat dépend de la longueur de la clé :

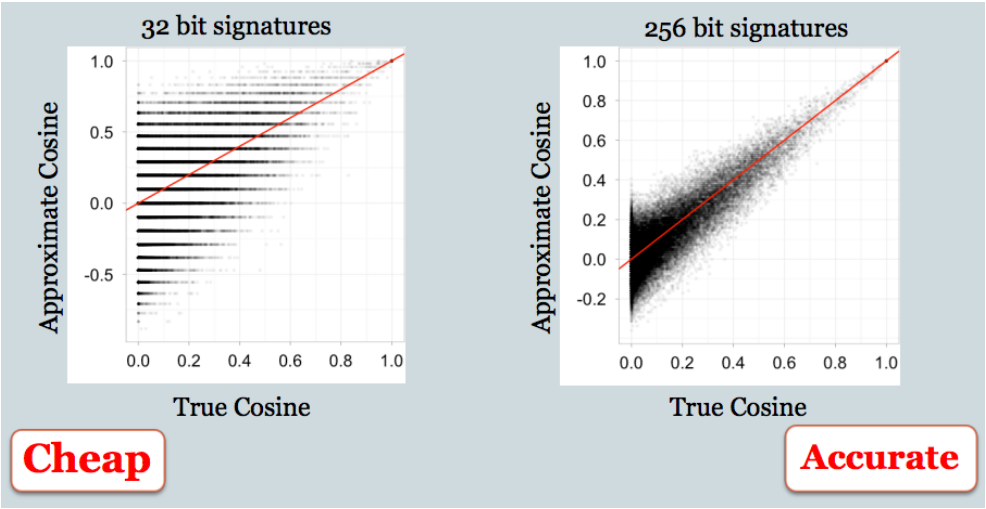

Fig. 5 Précision de la méthode LSH pour la similarité cosinus en fonction de la taille de la clé de hachage

Source des images : http://www.cs.jhu.edu/~vandurme/papers/VanDurmeLallACL10-slides.pdf

b. Distance Euclidienne

Nous expliquerons cette méthode seulement pour un espace à deux dimensions, mais elle fonctionne aussi pour des espaces de plus grande dimension.

Le principe est le suivant : chaque fonction de hachage f dans notre famille F sera associée à une ligne prise au hasard dans l'espace de recherche. On choisit une constante *a* et on divise la ligne en segments de longueur *a*. Ces segments correspondent aux "packets" (*buckets*) des fonctions de hachage par leur projection sur la ligne. Une fois les fonctions de hachage appliquées, pour trouver les plus proches voisins d'un point, on peut restreindre l'espace de recherche aux points qui ont une clé de hachage identique pour accélérer la recherche.

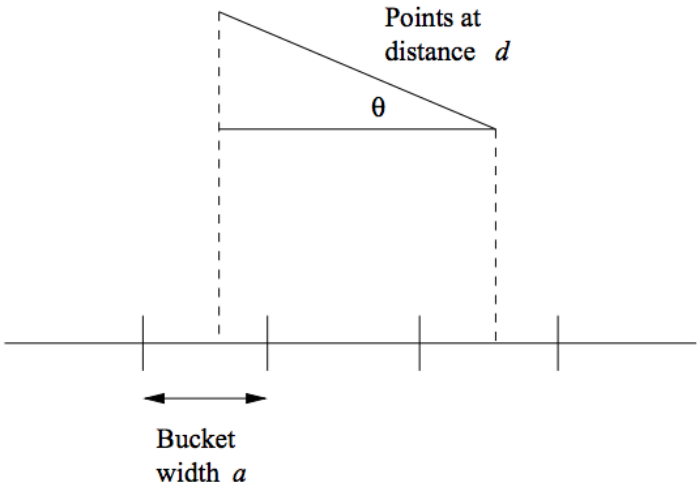

Fig. 6 Les points éloignés (d>>a) sont rarement dans le même packet.

Source de l'image : http://infolab.stanford.edu/~ullman/mmds/ch3.pdf

3.2 Best Bin First (BBF)

La méthode de Best Bin First est une variante des arbres kd. Le principe est le suivant : comme pour la méthode LSH, on recherche seulement dans des sous espaces de recherche (*bins*) par ordre croissant de la distance au point interrogé. La distance entre le sous espace et le point interrogé est la distance minimale entre ce point et tout point situé sur la limite du sous espace de recherche.

Source : http://image.ntua.gr/iva/files/ann.pdf

Conclusion

Nous avons pu comparer et étudier les différentes méthodes de recherche des plus proches voisins pour pouvoir choisir une technique adaptée à notre projet. L'objectif de cette comparaison était de trouver une méthode de recherche qui donne un résultat précis en un temps raisonnable. Nous avons opté pour un algorithme linéaire. En effet, la génération des mosaïques implique déjà de faire un tri des films en fonction de leur position (coordonnées *x,y*). On peut alors réduire l'espace de recherche aux films appartenant à la même mosaïque (voire aux mosaïques voisines pour les cas où le film le plus proche se trouve sur une mosaïque différente).

Références

GORISSE David, K Nearest Neighbours Search, Janvier 2009, http://perso-etis.ensea.fr/~davigori/master/knn.pdf

FLEURY Cédric, Le kd-Tree : une méthode de subdivision spatiale, novembre 2007, http://cedric.fleu.free.fr/media/rapportCTR_cfleury.pdf

RAJARAMAN Anand and ULLMAN Jeff, Mining of Massive Datasets, juillet 2012, http://infolab.stanford.edu/~ullman/mmds.html

VAN DURME Benjamin & LALL Ashwin, Online Generation of Locality Sensitive Hash Signatures, 2010, http://www.cs.jhu.edu/~vandurme/papers/VanDurmeLallACL10-slides.pdf

KYBIC Janand and VNUCKO Ivan, Approximate Best Bin First k-d Tree All Nearest Neighbor Search with Incremental Updates, juillet 2010, ftp://cmp.felk.cvut.cz/pub/cvl/articles/kybic/Kybic-CAK-2010-40.pdf