

02-03

Locality-Sensitive Hashing

NOUS ÉCLAIRON.
VOUS BRILLENZ.

FORMATION CONTINUE
ET SERVICES AUX ENTREPRISES



Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

NN: considérations pratiques

- La recherche d'un plus proche voisin par **force brute** ou par l'algorithme **KD-trees** (non couvert par le cours) ne résout pas certaines problématiques:
 - Implémentation efficace difficile (KD-trees)
 - Fléau de la dimension (force brute/KD-trees)
- Pour un grand nombre d'application, il n'est pas nécessaire de trouver le plus proche voisin de manière exacte
 - Sur plusieurs millions d'articles, a-t-on nécessairement besoin du plus proche ou plus simplement d'un article similaire ?
 - Quelle confiance accorder à la mesure de similarité ?

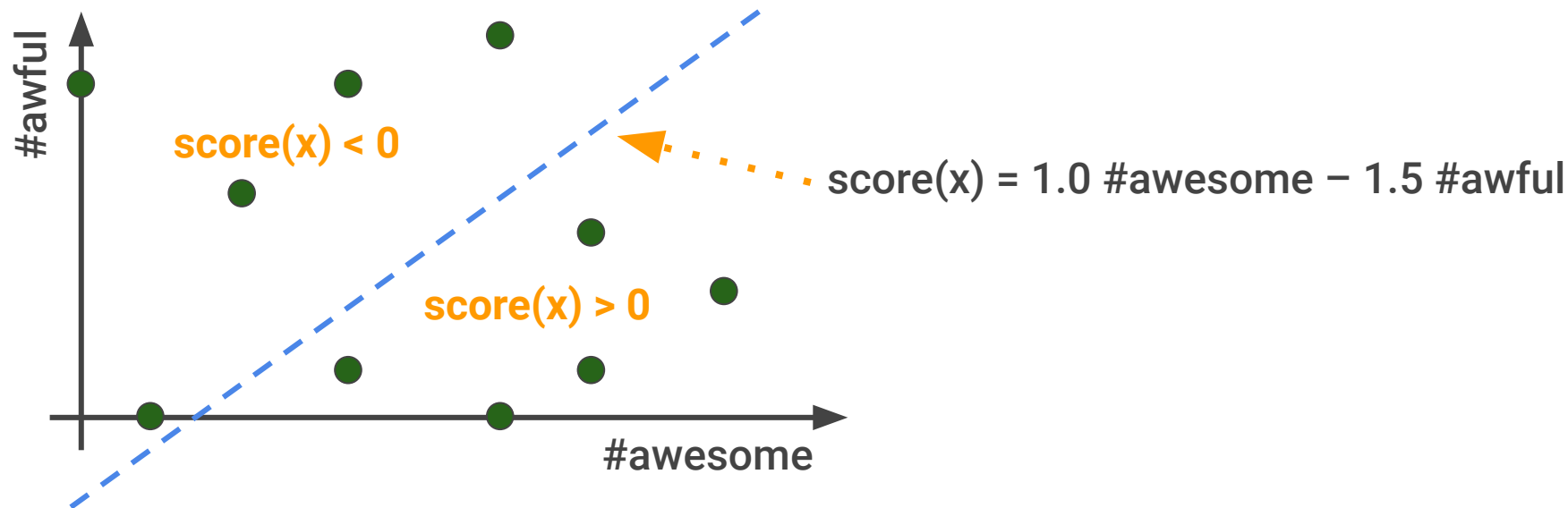
NN: considérations pratiques

- La recherche de plus proche voisin par **force brute** ou par l'algorithme **KD-trees** (non couvert par le cours) ne résout pas certaines problématiques:
 - Implémentation efficace difficile de mise en oeuvre (KD-trees)
 - Fléau de la dimension (force brute/KD-trees)
- **Besoin d'une méthode retournant un résultat approximatif ayant une forte probabilité d'être proche du document "requête"**
- Pour un grand nombre d'applications, il n'est pas nécessaire de trouver le plus proche voisin de manière exacte
 - Sur plusieurs millions d'articles, a-t-on nécessairement besoin du plus proche ou plus simplement d'un article similaire ?
 - Quelle confiance accorder à la mesure de similarité ?

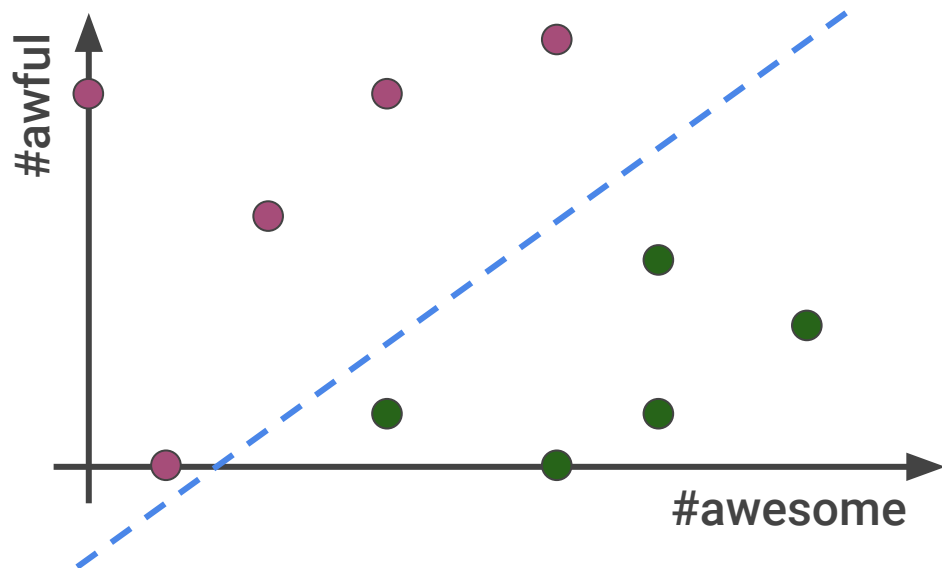
Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

“Binning” simple des données (1/3)

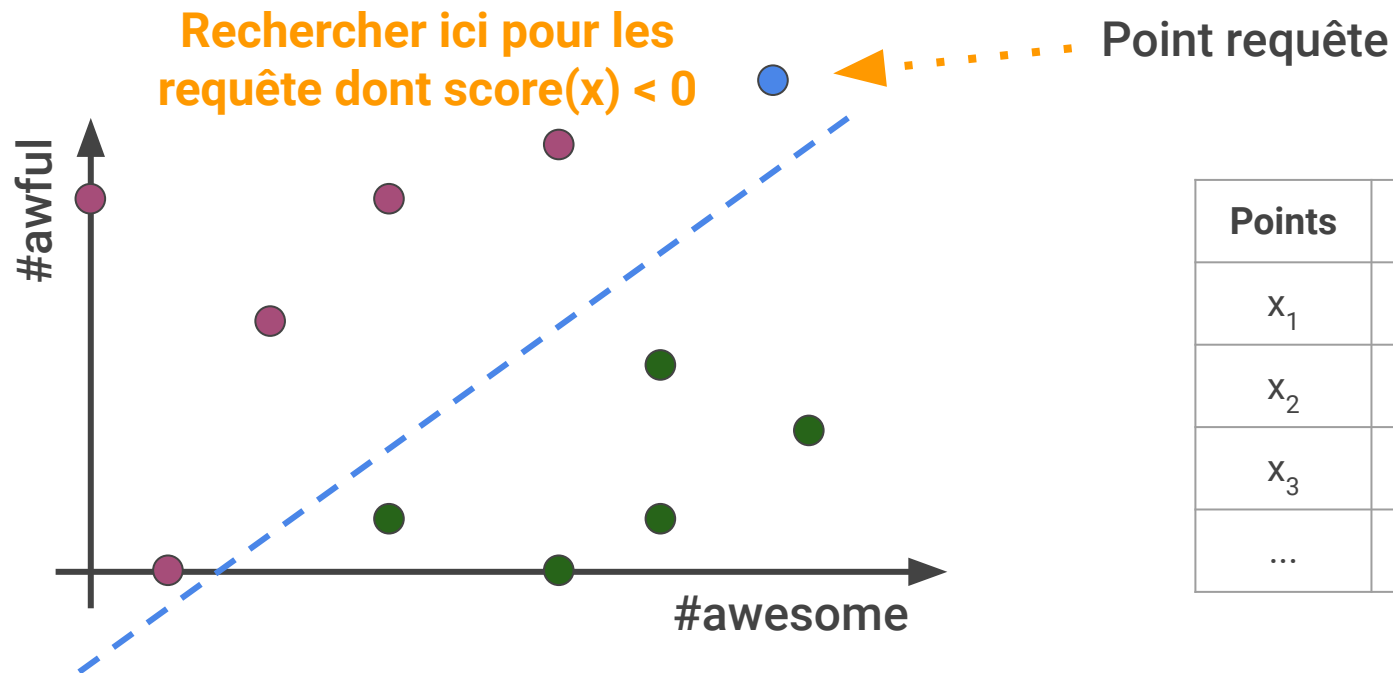


“Binning” simple des données (2/3)

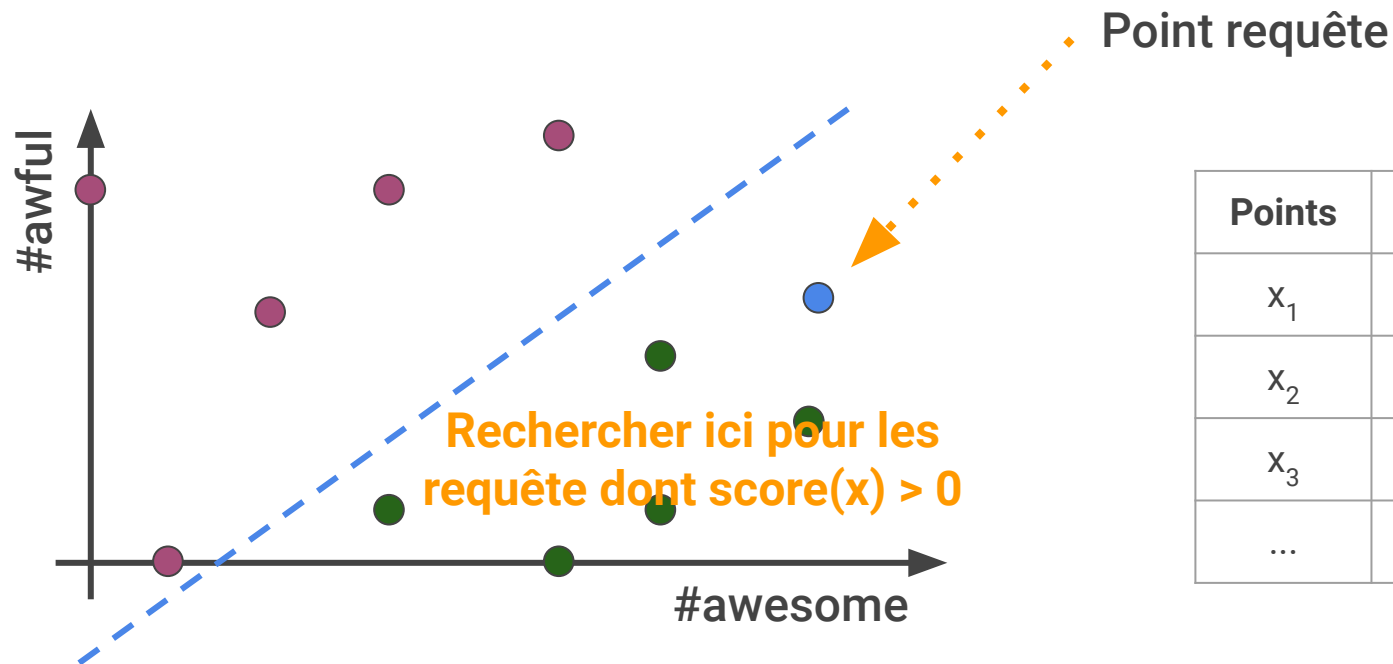


Points	Score
x_1	-1
x_2	-1
x_3	1
...	...

“Binning” simple des données (2/3)




“Binning” simple des données (2/3)



Points	Score	Bin
x_1	-1	0
x_2	-1	0
x_3	1	1
...

“Binning” simple des données (3/3)

Index du bin



Bin	0	1
Liste des indices des points	{1, 2, 3, 7, ...}	{3, 5, 6, 8, ...}

Points	Score	Bin
x_1	-1	0
x_2	-1	0
x_3	1	1
...

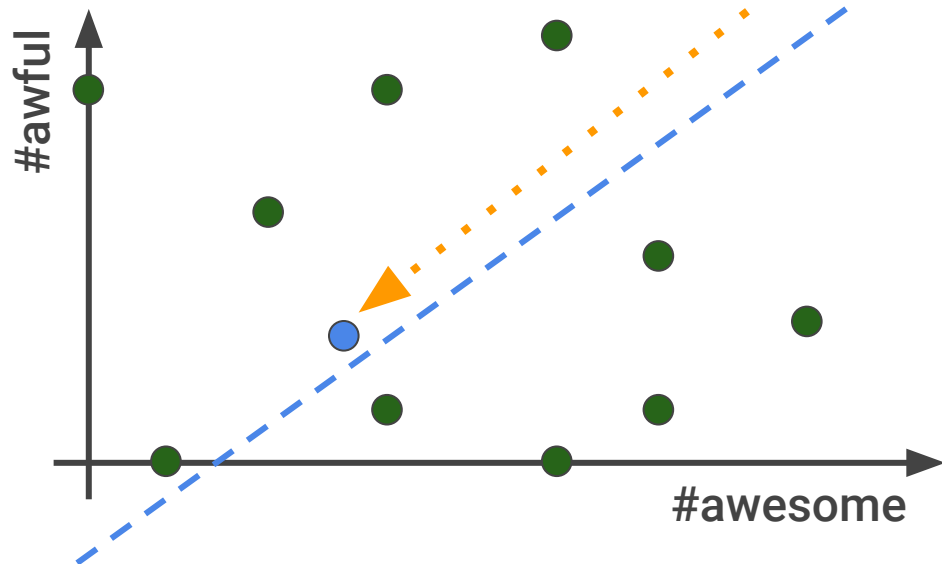
“Binning” simple des données (3/3)

Index du bin

Bin	0	1
Liste des indices des points	{1, 2, 3, 7, ...}	{ 3 , 5, 6, 8, ...}

Points	Score	Bin
x_1	-1	0
x_2	-1	0
x_3	1	1
...

Approximation NN



Avec cette partition, est-on capable de trouver “le” plus proche voisin de ce point requête? Non !

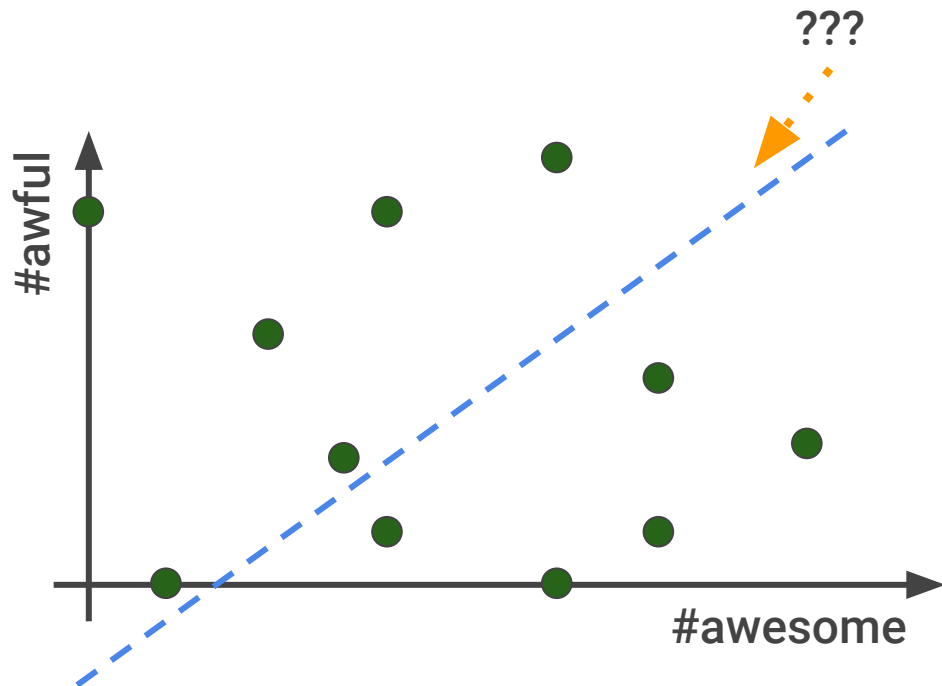
Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

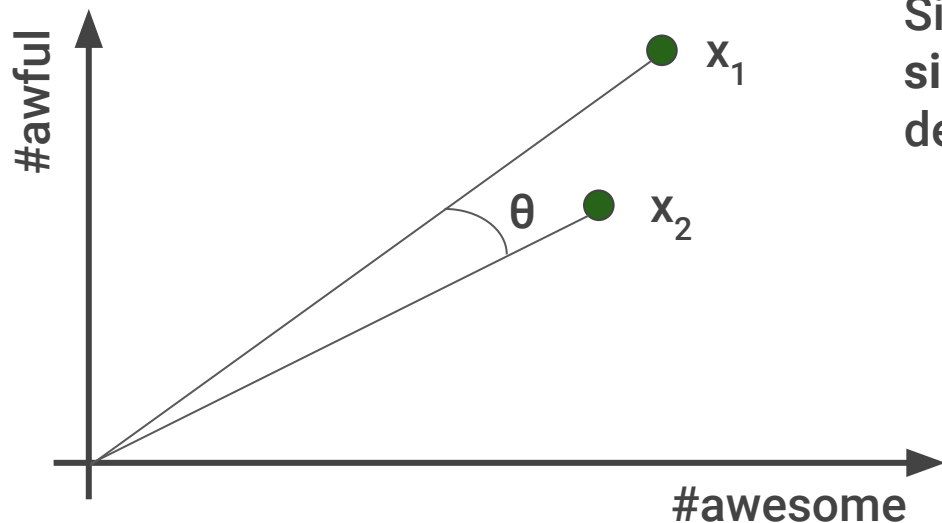
3 problématiques persistent ...

- Comment trouver une **ligne de partition** ?
- Qualité de la solution (des points proches peuvent être placés dans des bins différents)
- Coût de calculs: les bins contiennent encore trop de points, espace de recherche trop grand pour NN

Comment trouver une ligne de partition ?

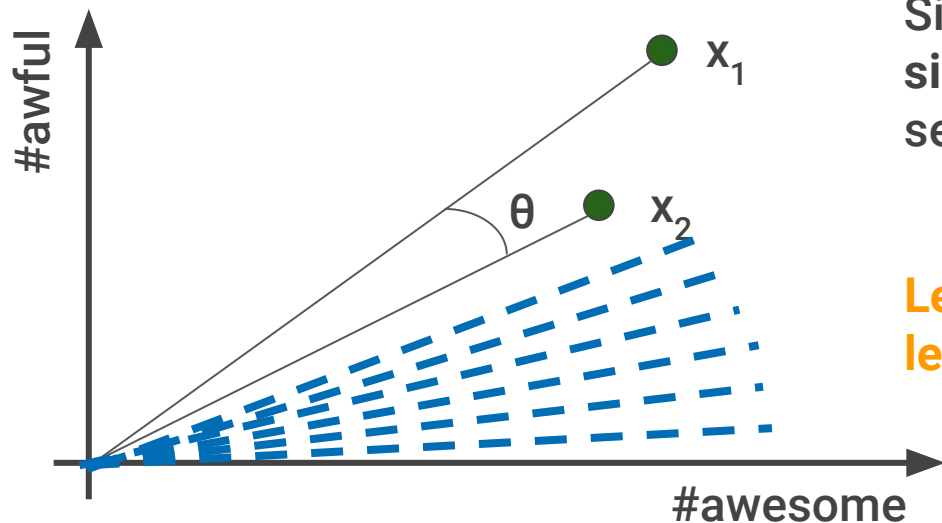


Comment trouver une ligne de partition ?



Si x_1 et x_2 sont proches (d'un point de vue similarité cosinus), alors ceux-ci devraient se retrouver dans le même bin

Comment trouver une ligne de partition ?

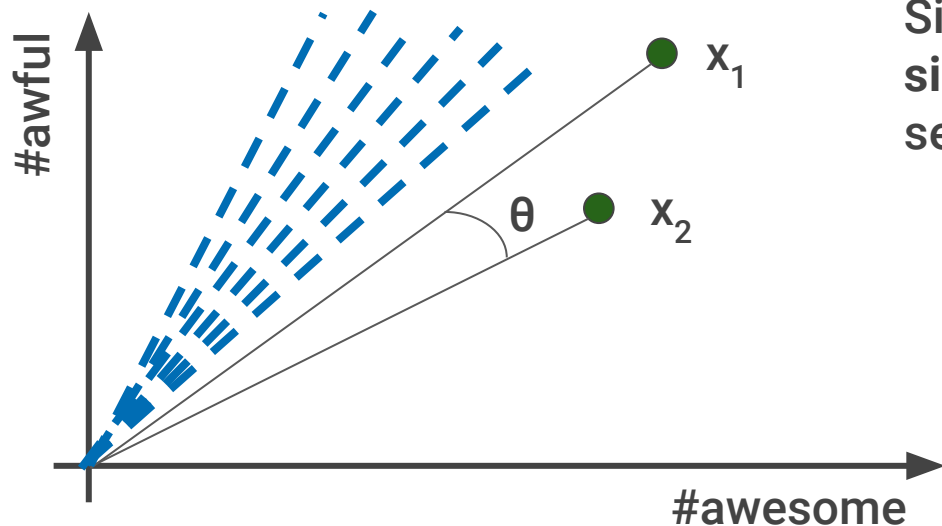


Si x_1 et x_2 sont proches (d'un point de vue similarité cosinus), alors ceux-ci doivent se retrouver dans le même bin

Les deux points sont dans le bin d'index 0

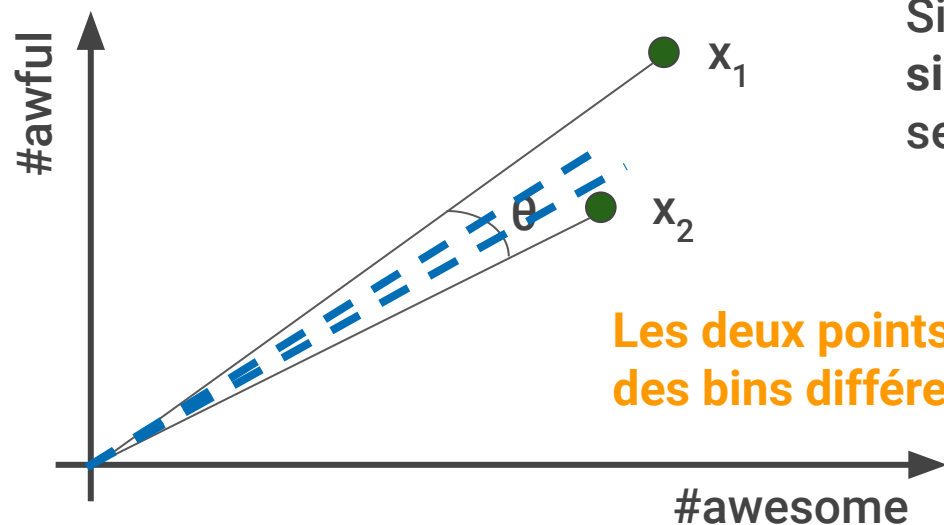
Comment trouver une ligne de partition ?

Les deux points sont dans
le bin d'index 1



Si x_1 et x_2 sont proches (d'un point de vue similarité cosinus), alors ceux-ci doivent se retrouver dans le même bin

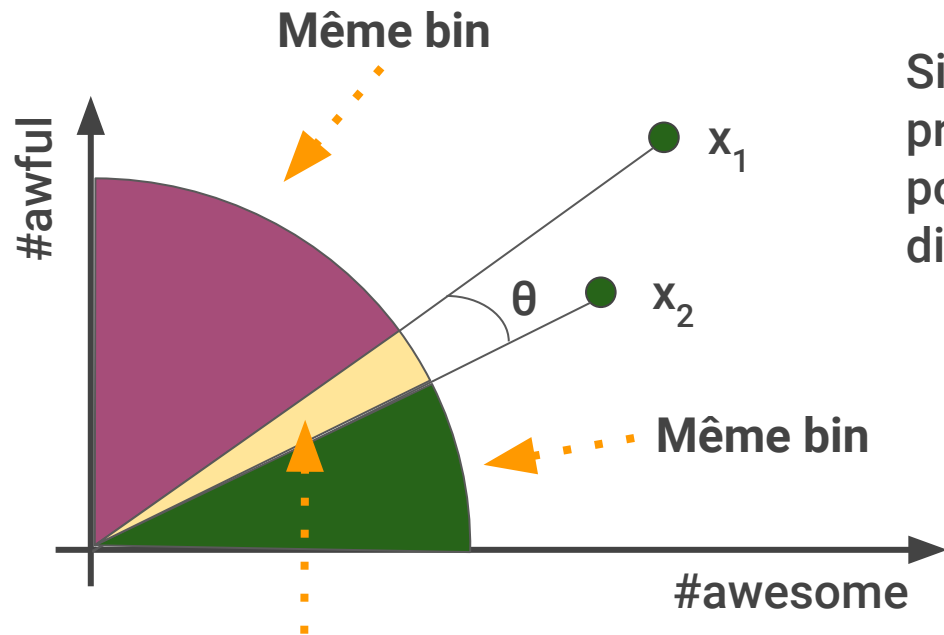
Comment trouver une ligne de partition ?



Si x_1 et x_2 sont proches (d'un point de vue similarité cosinus), alors ceux-ci doivent se retrouver dans le même bin



Les deux points sont dans des bins différents

Comment trouver une ligne de partition ?



Si θ est suffisamment petit (x_1 et x_2 sont proches), alors la probabilité que les points soient placés dans des bins différents est faible

3 problématiques persistent ...

- Comment trouver une **ligne de partition** ? 
- Qualité de la solution (des points proches peuvent être placés dans des bins différents) 
- Coût de calculs: les bins contiennent encore trop de points, espace de recherche trop grand pour NN

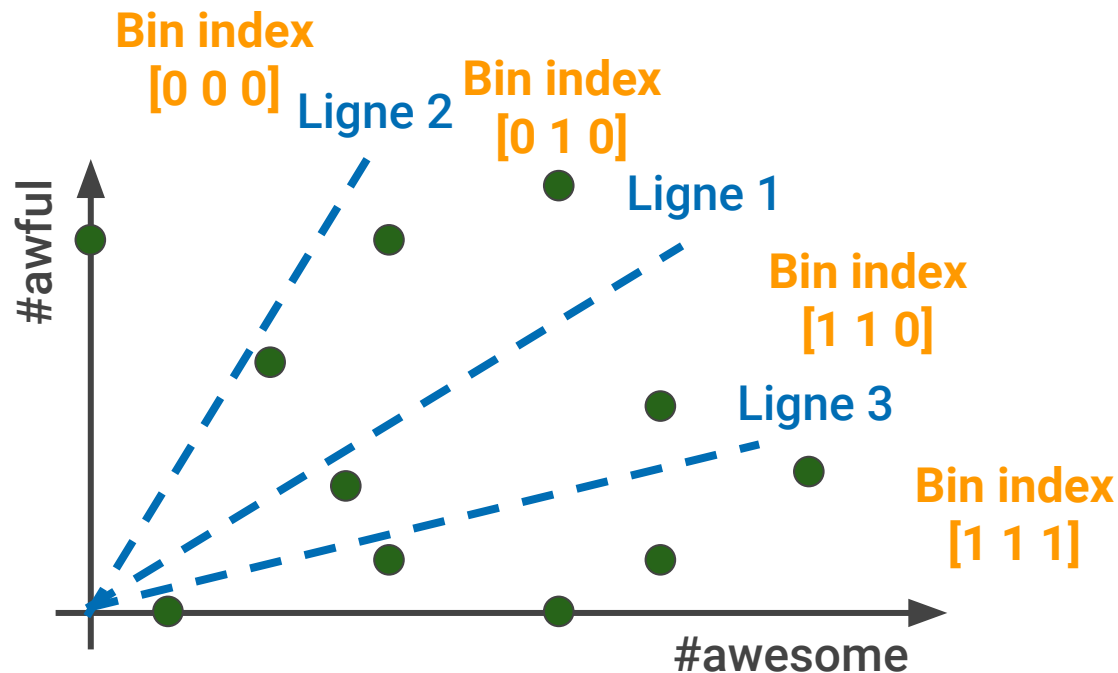


Bin	0	1
Liste des indices des points	{1, 2, 3, 7, ...}	{3, 5, 6, 8, ...}

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. **Nombre de bins**
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

Réduction du coût de recherche avec plusieurs bins

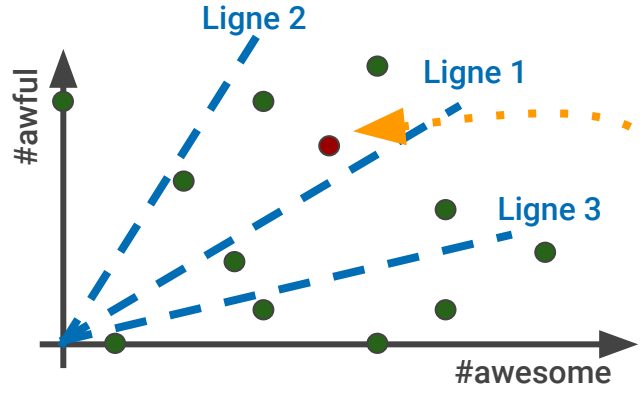


Réduction du coût de recherche avec plusieurs bins

Points	Score 1	Index bin 1	Score 2	Index bin 2	Score 3	Index bin 3
x_1	-1	0	-1	0	-1	0
x_2	-1	0	-1	0	-1	0
x_3	1	1	1	1	1	1
...

Bin	$[0\ 0\ 0]$ 0	$[0\ 0\ 1]$ 1	$[0\ 1\ 0]$ 2	$[0\ 1\ 1]$ 3	$[1\ 0\ 0]$ 4	$[1\ 0\ 1]$ 5	$[1\ 1\ 0]$ 6	$[1\ 1\ 1]$ 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

Réduction du coût de recherche avec plusieurs bins



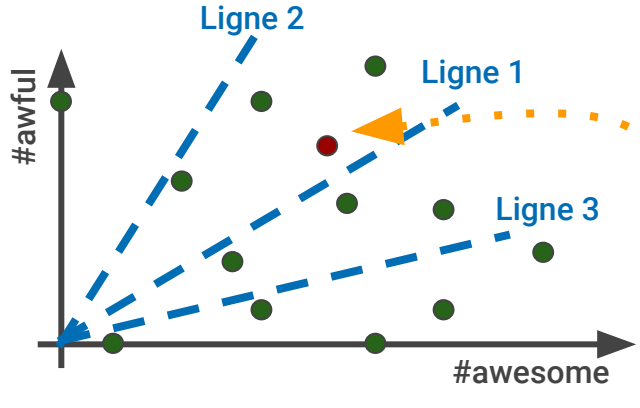
Recherche des plus proches voisins
parmi les éléments du bin d'indice 2

Bin	[0 0 0] 0	[0 0 1] 1	[0 1 0] 2	[0 1 1] 3	[1 0 0] 4	[1 0 1] 5	[1 1 0] 6	[1 1 1] 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

Qualité de la recherche

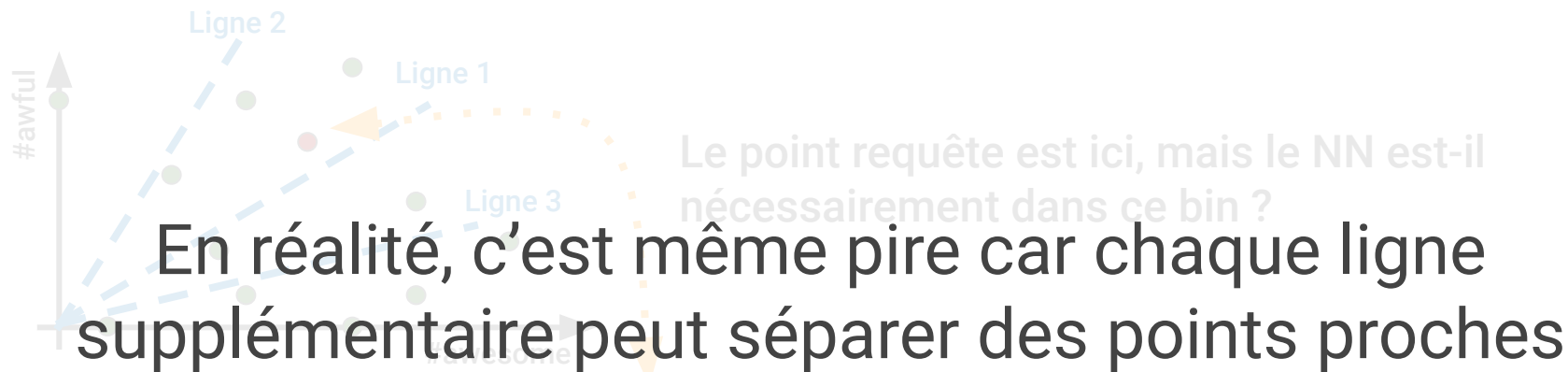


Le point requête est ici, mais le NN est-il nécessairement dans le bin d'index 2 ?

Pas forcément ...

Bin	[0 0 0] 0	[0 0 1] 1	[0 1 0] 2	[0 1 1] 3	[1 0 0] 4	[1 0 1] 5	[1 1 0] 6	[1 1 1] 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

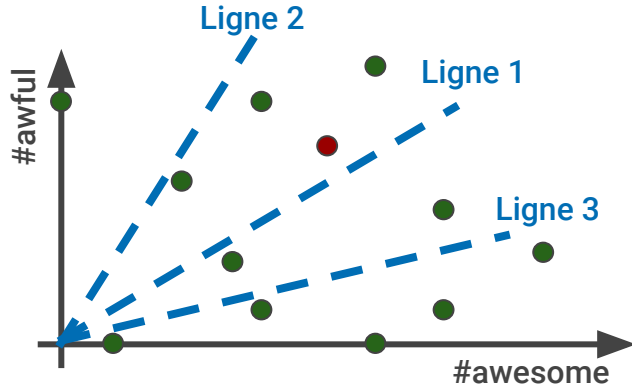
Qualité de la recherche



Compromis vitesse - précision

Bin	[0 0 0] 0	[0 0 1] 1	[0 1 0] 2	[0 1 1] 3	[1 0 0] 4	[1 0 1] 5	[1 1 0] 6	[1 1 1] 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

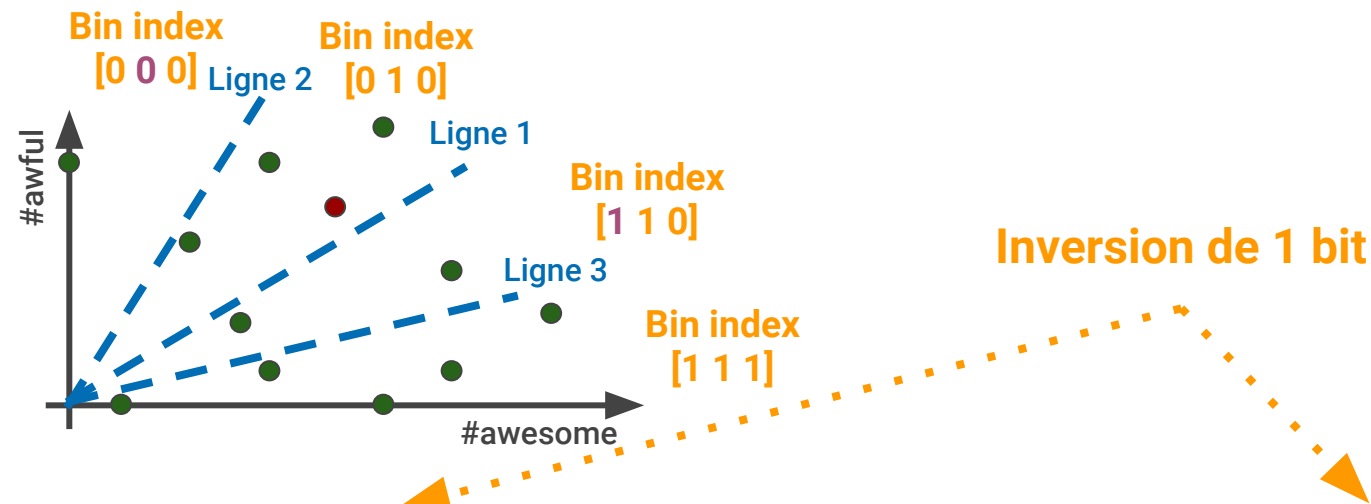
Qualité de la recherche



Dans ce cas, il peut être intéressant de rechercher dans les bins les plus proches. Par quelle opération peut-on les trouver?

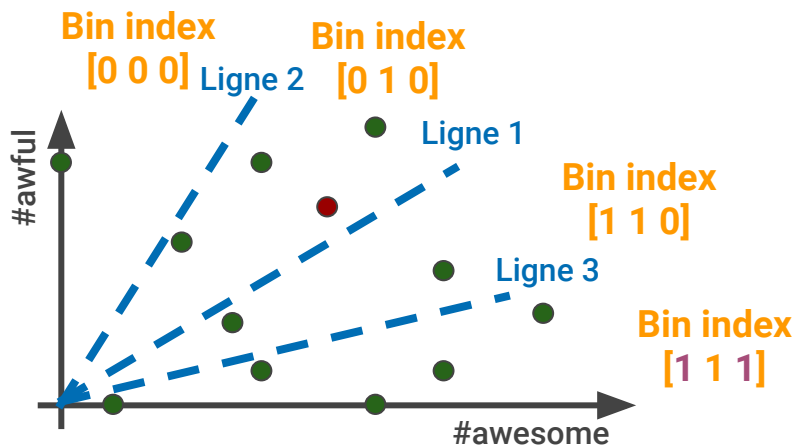
Bin	$[0\ 0\ 0]$ 0	$[0\ 0\ 1]$ 1	$[0\ 1\ 0]$ 2	$[0\ 1\ 1]$ 3	$[1\ 0\ 0]$ 4	$[1\ 0\ 1]$ 5	$[1\ 1\ 0]$ 6	$[1\ 1\ 1]$ 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

Qualité de la recherche

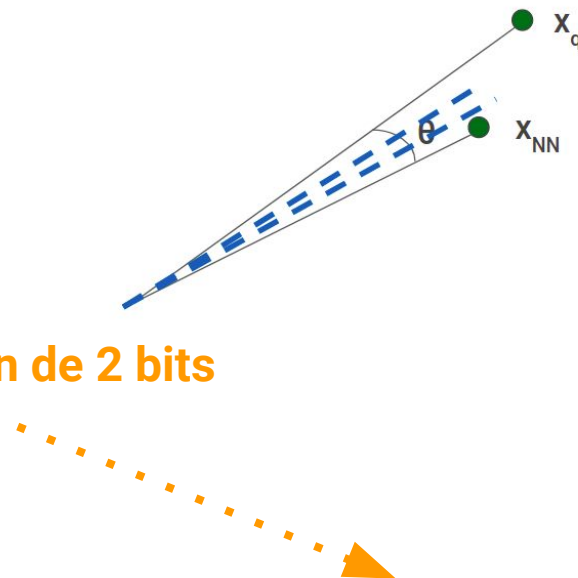


Bin	[0 0 0] 0	[0 0 1] 1	[0 1 0] 2	[0 1 1] 3	[1 0 0] 4	[1 0 1] 5	[1 1 0] 6	[1 1 1] 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

Qualité de la recherche



Inversion de 2 bits



Bin	[0 0 0] 0	[0 0 1] 1	[0 1 0] 2	[0 1 1] 3	[1 0 0] 4	[1 0 1] 5	[1 1 0] 6	[1 1 1] 7
Liste des indices des points	{1, 2}	--	{4, 8, 11}	--	--	--	{7, 9, 10}	{3, 5, 6}

Qualité de la recherche

- La qualité du NN trouvé ne peut qu'augmenter avec recherchant dans plus de bins
- Algorithme: **continuer la recherche du NN tant que le budget de calcul n'est dépassé ou que la qualité du meilleur NN trouvé soit suffisamment satisfaisante**

LSH - Résumé

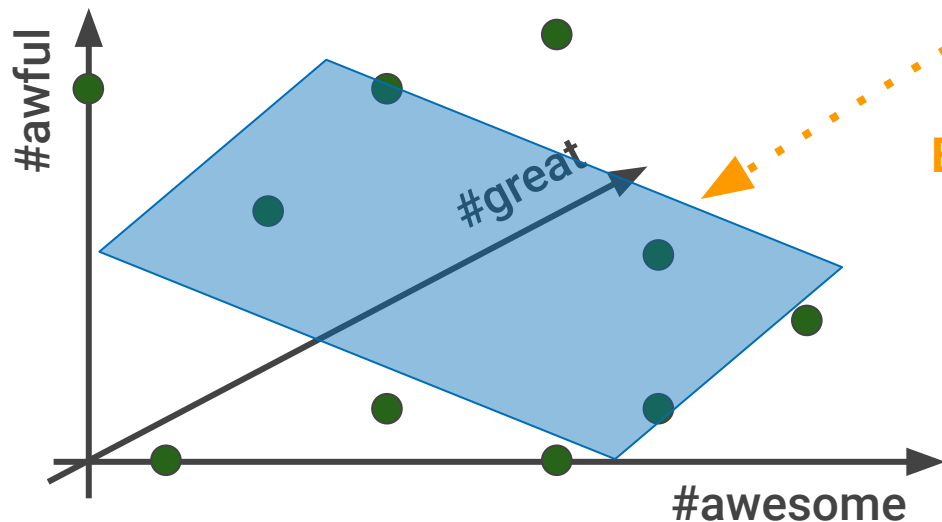
- Échantillonner h lignes de partition ?
- Calculer le score pour chaque point et convertir en index binaire
- Utiliser un vecteur binaire de h bits par point comme index de bin
- Créer la table de hachage
- Pour chaque point requête \mathbf{x}_q , rechercher $\text{bin}(\mathbf{x}_q)$, puis les bins voisins jusqu'à un critère d'arrêt prédéfini

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

LSH et dimensionnalité

$$\text{score}(x) = v_1 \# \text{awesome} + v_2 \# \text{awful} + v_2 \# \text{great}$$



Échantillonner des plans aléatoires

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références



<https://github.com/mswawola-cegep/420-a58-sf-gr-12060.git>

02-03

Sommaire

1. NN: considérations pratiques
2. Algorithme LSH
3. Implémentation
4. Nombre de bins
5. Qualité de la recherche
6. LSH et dimensionnalité
7. Atelier
8. Lectures et références

Références

[1] Machine Learning: Clustering and Retrieval - Emily Fox & Carlos Guestrin -
University of Washington