Recherche d'Informations

Sylvain Gault

8 octobre 2024

Généralités

Définition

- En anglais *Information Retrieval*
- Trouver du contenu (habituellement des documents)
- Non structuré (habituellement du texte)
- Qui satisfait des besoins (en termes d'informations)
- Depuis une collection de documents (habituellement stockés sur ordinateurs)

Généralités

Exemples

- Recherche web
- Recherche locale (sur le PC de l'utilisateur)
- Recherche dans des e-mails
- Recherche dans des documents légaux
- Recherche dans une base de connaissance locale
- ...

Données structurées et non-structurées

Dans les années 90

- Beaucoup plus de données non-structurées que structurées
- Offres commerciales importante pour la gestion de données structurées
- Peu d'offres commerciales pour les données non-structurées

Aujourd'hui

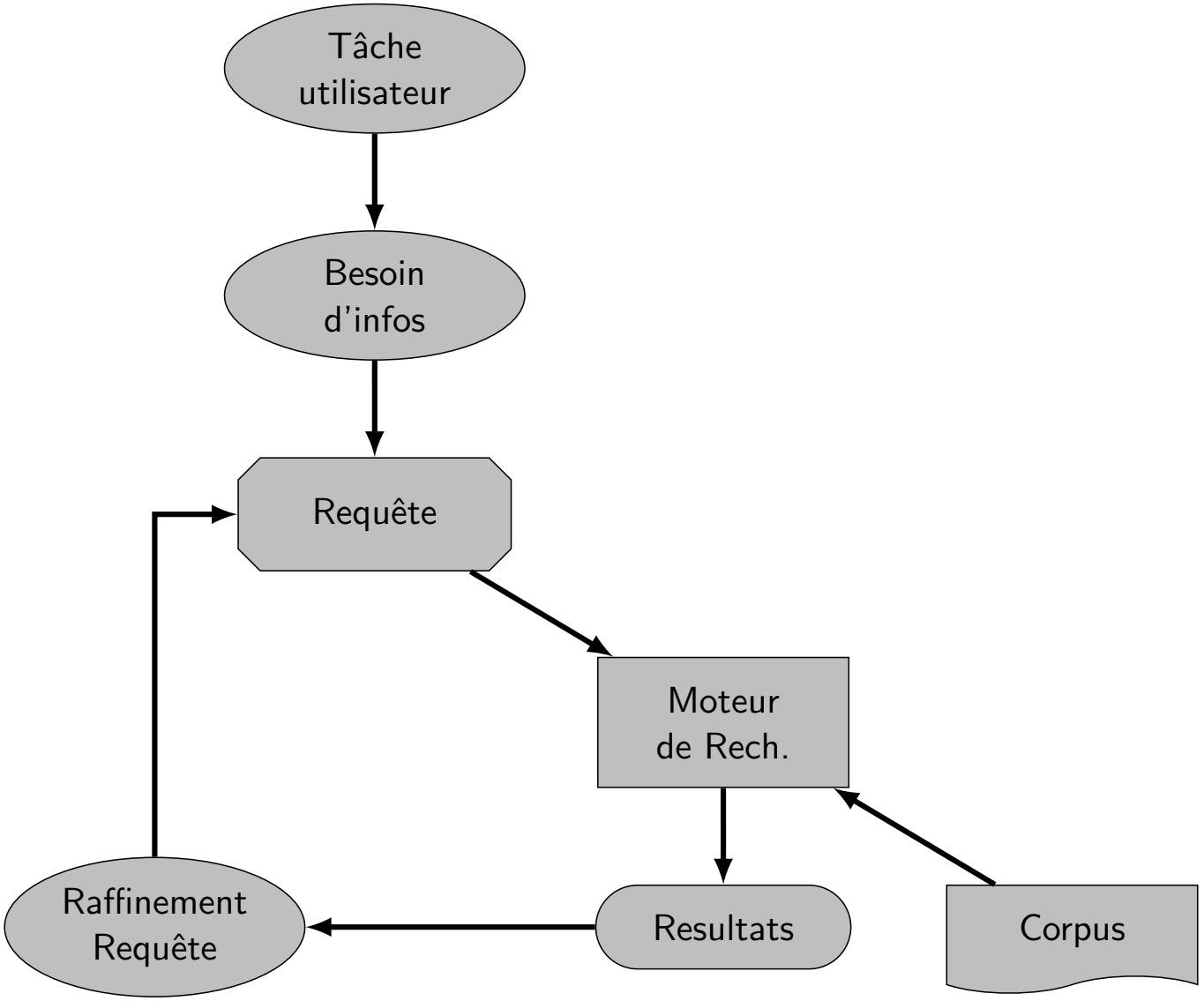
- Beaucoup plus de données non-structurées que structurées
- Beaucoup d'offres commerciales de gestion de données non-structurées
- Un peu moins d'offres commerciales de gestion de données structurées (relativement)

Suppositions de base de la recherche d'information

Suppositions

- Collection de documents
  - Statique pour le moment
- But : retrouver les documents pertinents

Modèle de recherche classique



Exemple : se débarrasser des souris

Tâche

- Se débarrasser des souris d'une manière politiquement correcte

Besoin d'informations

- Enlever les souris sans les tuer

Requête

- Attraper souris vivantes

Exemple : se débarrasser des souris

Tâche

- Se débarrasser des souris d'une manière politiquement correcte
- Idée fausse sur le monde

Besoin d'informations

- Enlever les souris sans les tuer
- Mauvaise formulation

Requête

- Attraper souris vivantes

Exemple : se débarrasser des souris

Tâche

- Se débarrasser des souris d'une manière politiquement correcte
- Idée fausse sur le monde

Besoin d'informations

- Enlever les souris sans les tuer
- Mauvaise formulation

Requête

- Attraper souris vivantes
- Attraper souris sans danger
- Rajouter des guillemets
- ...

Évaluation de la recherche d'information

Précision

- Fraction du nombre de documents retournés qui sont pertinents pour les besoins d'informations de l'utilisateur
- Nombre de vrai positifs / nombre de documents retournés

Rappel (recall)

- Fraction du nombre de documents pertinents du corpus qui sont retournés
- Vrais positifs / Nombre de pertinents

Évaluation de la recherche d'information

Précision

- Fraction du nombre de documents retournés qui sont **pertinents pour les besoins d'informations de l'utilisateur**
- Nombre de vrai positifs / nombre de documents retournés

Rappel (recall)

- Fraction du nombre de documents pertinents du corpus qui sont retournés
- Vrais positifs / Nombre de pertinents

Matrice d'incidence terme-document

Définition

- *Term-document incidence Matrix*
- Matrice qui indique la présence d'un mot dans un document
- Pour chaque document
- Pour chaque mot d'un vocabulaire

Mot	**Doc 1**	**Doc 2**	**Doc 3**
`the`	1	1	1
`very`	1	0	1
`best`	1	0	0

Requêtes booléennes

Utilisation

- Rechercher d'une expression booléenne
  - Exemple : *(the AND very) OR best*
- Appliquer les opérateurs booléens sur les vecteurs de bits

Mot	**Doc 1**	**Doc 2**	**Doc 3**
`the`	1	1	1
`very`	1	0	1
`best`	1	0	0

Requête	**Doc 1**	**Doc 2**	**Doc 3**
`the AND very`	1	0	1
`(the AND very) OR best`	1	0	1

Limites de cette matrice

Limites

- Taille énorme pour un grand vocabulaire et beaucoup de documents
- Contient majoritairement des 0 si le vocabulaire est significativement plus grand que la longueur moyenne des documents
- Temps de calcul beaucoup trop grand

Solution

Limites de cette matrice

Limites

- Taille énorme pour un grand vocabulaire et beaucoup de documents
- Contient majoritairement des 0 si le vocabulaire est significativement plus grand que la longueur moyenne des documents
- Temps de calcul beaucoup trop grand

Solution

- Représentation *creuse* (*sparse*)

Index inversé

Mot	Doc 1	Doc 2	Doc 3
the	1	1	1
very	1	0	1
best	1	0	0

Mot	Liste de documents
the	1 2 3
very	1 3
best	1

Index inversé

Index inversé

- Stocke pour chaque mot...
- La liste des documents qui contiennent ce mot

Pour aller plus loin

Autres features

- Expressions communes dans les requêtes = nouveau *mots*
- Stocker la liste des positions où le mot est trouvé
  - Rehercher *« d'expressions exactes »* en comparant les positions
- Pondérer les recherches par le nombre d'occurrence des mots dans les documents

Ranking

- Recherche par expressions booléennes sort trop ou trop peu de documents
- Trier par pertinence serait plus judicieux
- $\rightarrow$ Retourner *trop* de document n'est plus un problème

Pour aller plus loin

Autres features

- Expressions communes dans les requêtes = nouveau *mots*
- Stocker la liste des positions où le mot est trouvé
  - Rehercher *« d'expressions exactes »* en comparant les positions
- **Pondérer les recherches par le nombre d'occurrence des mots dans les documents**

Ranking

- Recherche par expressions booléennes sort trop ou trop peu de documents
- Trier par pertinence serait plus judicieux
- $\rightarrow$ Retourner *trop* de document n'est plus un problème

Matrice de comptage terme-document

Matrice de décompte terme-document

- Similaire à la matrice d'incidence terme-document
- Compte le nombre d'occurrences
- Ne distingue pas l'ordre des mots (modèle *Bag of Words*)

Mot	**Doc 1**	**Doc 2**	**Doc 3**
`the`	10	9	20
`very`	5	0	1
`best`	15	0	0

Fréquence des termes

Term-Frequency

- Ici *« fréquence »* désigne le pourcentage d'occurrence dans un document
- ... et pas une régularité temporelle
- La pertinence d'un document par rapport à un mot recherché croit avec la fréquence du mot dans le document
- ... mais pas linéairement

Exemple : *« Écureuil »*

- Un document qui mentionne *« Écureuil »* 10 fois est plus pertinent qu'un document qui le mentionne qu'une fois
- Mais pas 10 fois plus pertinent

Pondération logarithmique

Utiliser le log de la fréquence

- $w_{t,d} = 1 + log(tf_{t,d})$ si $tf_{t,d} > 0$
- $w_{t,d} = 0$ sinon
  - $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$
- Éliminer les mots de la requête qui n'apparaissent pas dans le document

$$score_{q,d} = \sum_{t \in q \cap d} 1 + log(tf_{t,d})$$

- D'autres formules à base de *log* existent

Document Frequency

Description

- Les mots courant devraient compter moins que les mots rares
- Éliminer les *stop words* ?
- Que faire des termes comme *« augmenter »*, *« ligne »*, *« envoyer »* ?
  - Communs mais pas inutiles
  - Ils devraient avoir un poids faible mais > 0
- Le score de *document-frequency* le prend en compte

Document Frequency

Féquence-document ?

- df_t est la fréquence du terme t dans le corpus
- Le nombre de documents qui contiennent t
- Mesure l'inverse de l'informativité
  - Plus df_t est faible, plus t est informatif
- IDF *« inverse document frequency »* :

$$idf_t = log(N/df_t)$$

- On utilise un *log* pour les mêmes raisons que pour *tf*
  - Un mot qui appraît dans 10 fois moins de documents est plus intéressant, mais pas 10 fois plus intéressant

Inverse-Document Frequency

Remarques sur *idf*

- La valeur idf_t est constante entre requêtes
  - Chaque terme a un *idf*, et c'est tout
- Ce facteur n'a aucune influence sur les requêtes à un seul mot
  - Tous les résultats sont pondérés par la même constante

Fréquence de corpus ?

- Pourquoi ne pas compter le nombre d'occurrence des mots dans tout le corpus ?
- Exemple : Deux mots de même nombre d'occurrences mais distribution différentes (uniforme, ou localisées dans peu de documents)
- Le mot avec une distribution uniforme est moins informatif

Term-Frequency Inverse-Document-Frequency

Définition

- $w_{t,d} = (1 + log(tf_{t,d})) \times log(N/df_t)$
- Valeur croissante avec le nombre d'occurrences dans un document
- Valeur croissante avec la rareté du terme dans le corpus

Term-Frequency Inverse-Document-Frequency

TF-IDF

$$score_{q,d} = \sum_{t \in q \cap d} w_{t,d}$$

$$score_{q,d} = \sum_{t \in q \cap d} tfidf_{t,d}$$

Documents en tant que vecteurs

Description de l'espace

- Représenter les documents comme un vecteur de valeurs $tfidf_{t,d}$ pour tous les mots du vocabulaire
- Les termes sont des dimensions de cet espace
- Les documents sont des points dans cet espace
- Espace de très haute dimensions sur le web (dizaines de millions)
- Avec beaucoup de zéros

Requêtes en tant que vecteurs

Idée

- Représenter les requêtes comme des vecteurs du même espace
- Trier les documents par rapport à leur *proximité* au vecteur correspondant à la requête
- Proximité $=$ similarité des vecteurs
- Proximité $\approx$ inverse de la distance

Requêtes en tant que vecteurs

Idée

- Représenter les requêtes comme des vecteurs du même espace
- Trier les documents par rapport à leur *proximité* au vecteur correspondant à la requête
- Proximité = similarité des vecteurs
- Proximité $\approx$ inverse de la distance
- Rappel On veut s'éloigner du *tout ou rien* des requêtes booléennes

Formuler la proximité

Idée

- Prendre la distance entre les points

Formuler la proximité

Idée

- Prendre la distance entre les points
- Distance euclidienne ?
- Les vecteurs de longueur différentes ont une distance très différente
- *Schéma « Ragot Jalousie » au tableau*

Formuler la proximité

Idée

- Prendre la distance entre les points
- Distance euclidienne ?
- Les vecteurs de longueur différentes ont une distance très différente
- *Schéma « Ragot Jalousie » au tableau*
- Les angles sont une meilleure mesure

Similarité cosinus

Similarité cosinus

- Prendre le cosinus de l'angle entre les deux vecteurs
- Fonction de valeur décroissante quand l'angle augmente
- $\rightarrow$ deux vecteurs quasiment colinéaires auront une valeur proche de 1
- $\rightarrow$ deux vecteurs très différents auront une valeur proche de 0
- Très rapide à calculer sans utiliser de fonction *cos*

Similarité cosinus

Calcul

$$\vec{q} \cdot \vec{d} = ||\vec{q}|| \times ||\vec{d}|| \times cos(\widehat{\vec{q}, \vec{d}})$$

$$cos(\widehat{\vec{q}, \vec{d}}) = \frac{\vec{q} \cdot \vec{d}}{||\vec{q}|| \times ||\vec{d}||} = \frac{\vec{q}}{||\vec{q}||} \cdot \frac{\vec{d}}{||\vec{d}||}$$

Similarité cosinus

Calcul

$$\vec{q} \cdot \vec{d} = ||\vec{q}|| \times ||\vec{d}|| \times cos(\widehat{\vec{q}, \vec{d}})$$

$$cos(\widehat{\vec{q}, \vec{d}}) = \frac{\vec{q} \cdot \vec{d}}{||\vec{q}|| \times ||\vec{d}||} = \frac{\vec{q}}{||\vec{q}||} \cdot \frac{\vec{d}}{||\vec{d}||}$$

- Normaliser les vecteurs avant pour éviter la division

Normalisation

Normalisation des vecteurs

- Diviser tous leurs coefficients par leur longueur

$$d'_i = \frac{d_i}{\sqrt{\sum_{j=1}^{|V|} d_j^2}}$$

- Idem pour q

Similarité cosinus

Calcul

$$\begin{aligned} cos(\widehat{\vec{q}, \vec{d}}) &= \vec{q} \cdot \vec{d} \\ &= \sum_i q_i d_i \end{aligned}$$

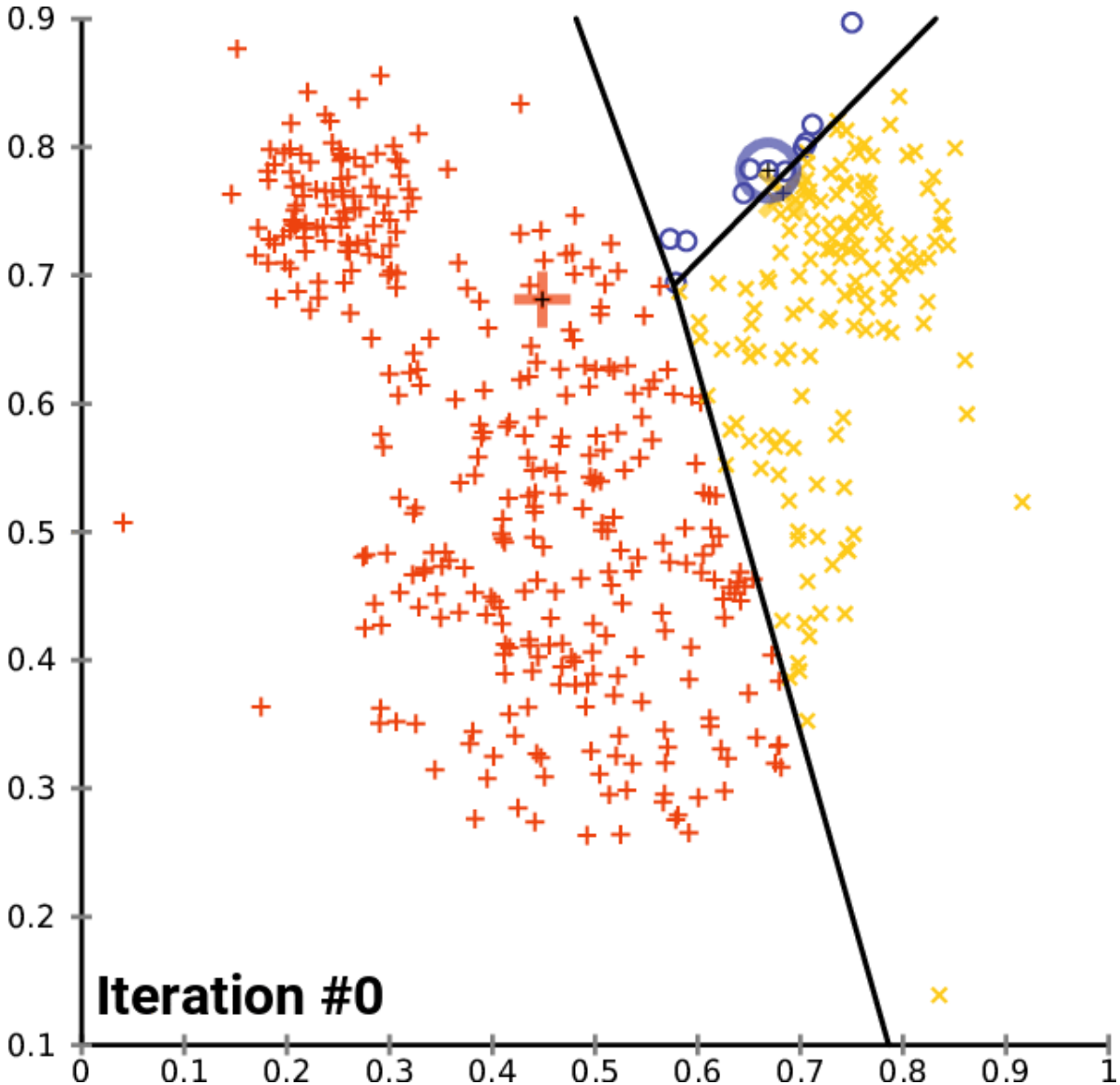
- Beaucoup plus rapide à calculer

Récap

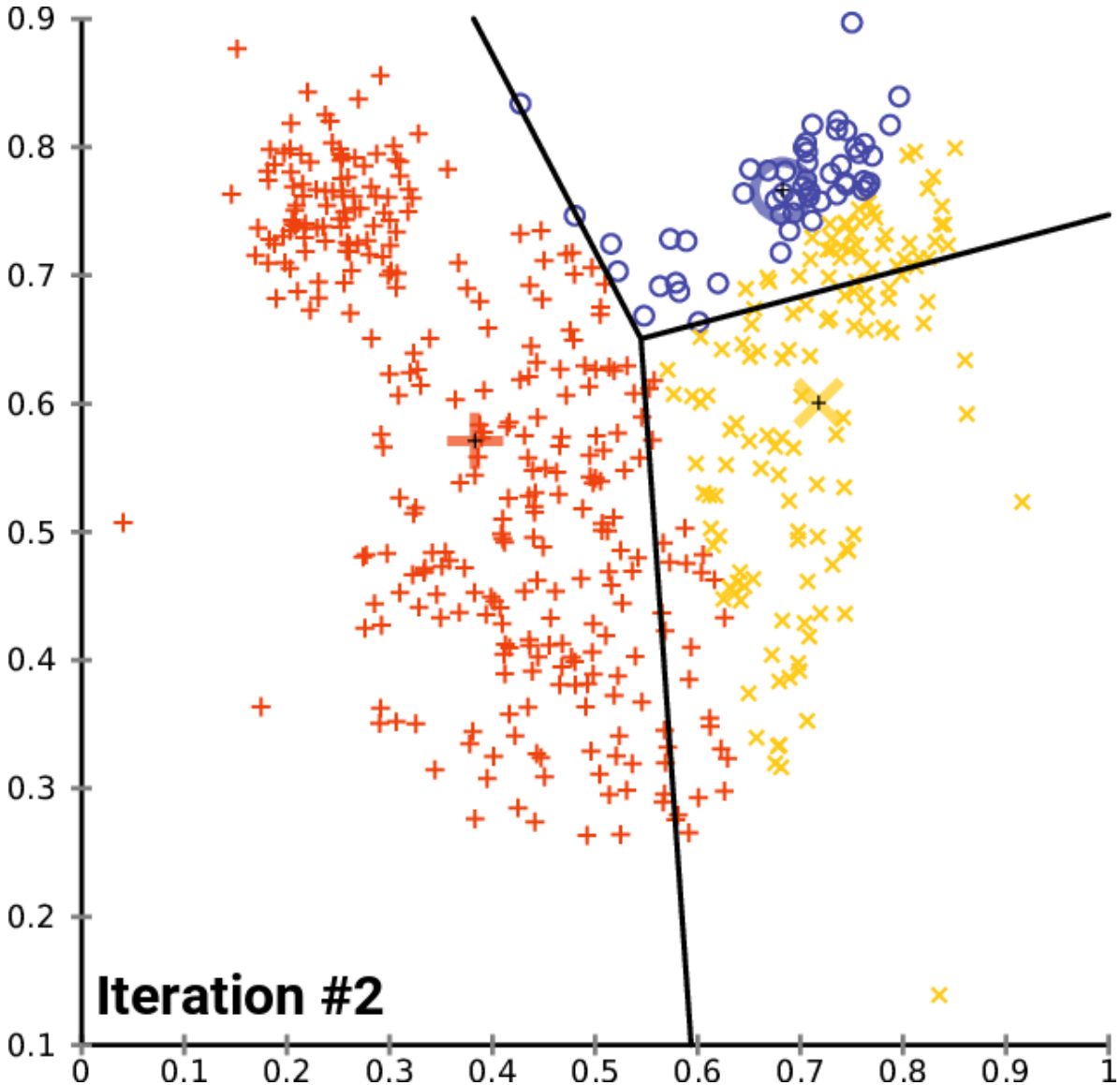
Récap

- Les espaces vectoriels sont très utiles
- Permettent de sortir de la vue binaire des recherches booléennes
- Il existe d'autres variantes de TF-IDF, autre trick que *log* (ou autres avec *log*)
- D'autres distances que cosinus et d'autres normalisations

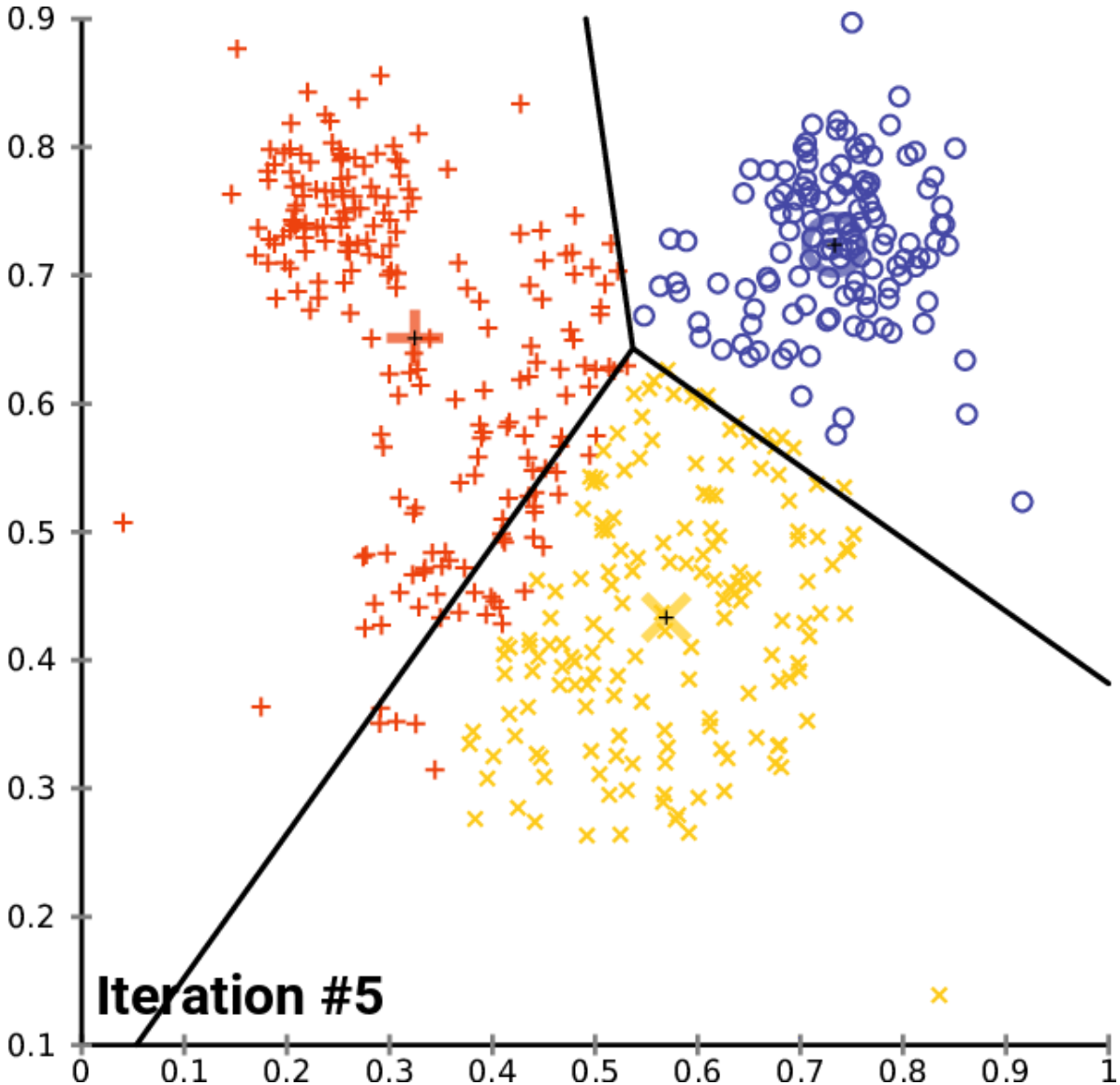
Exemple



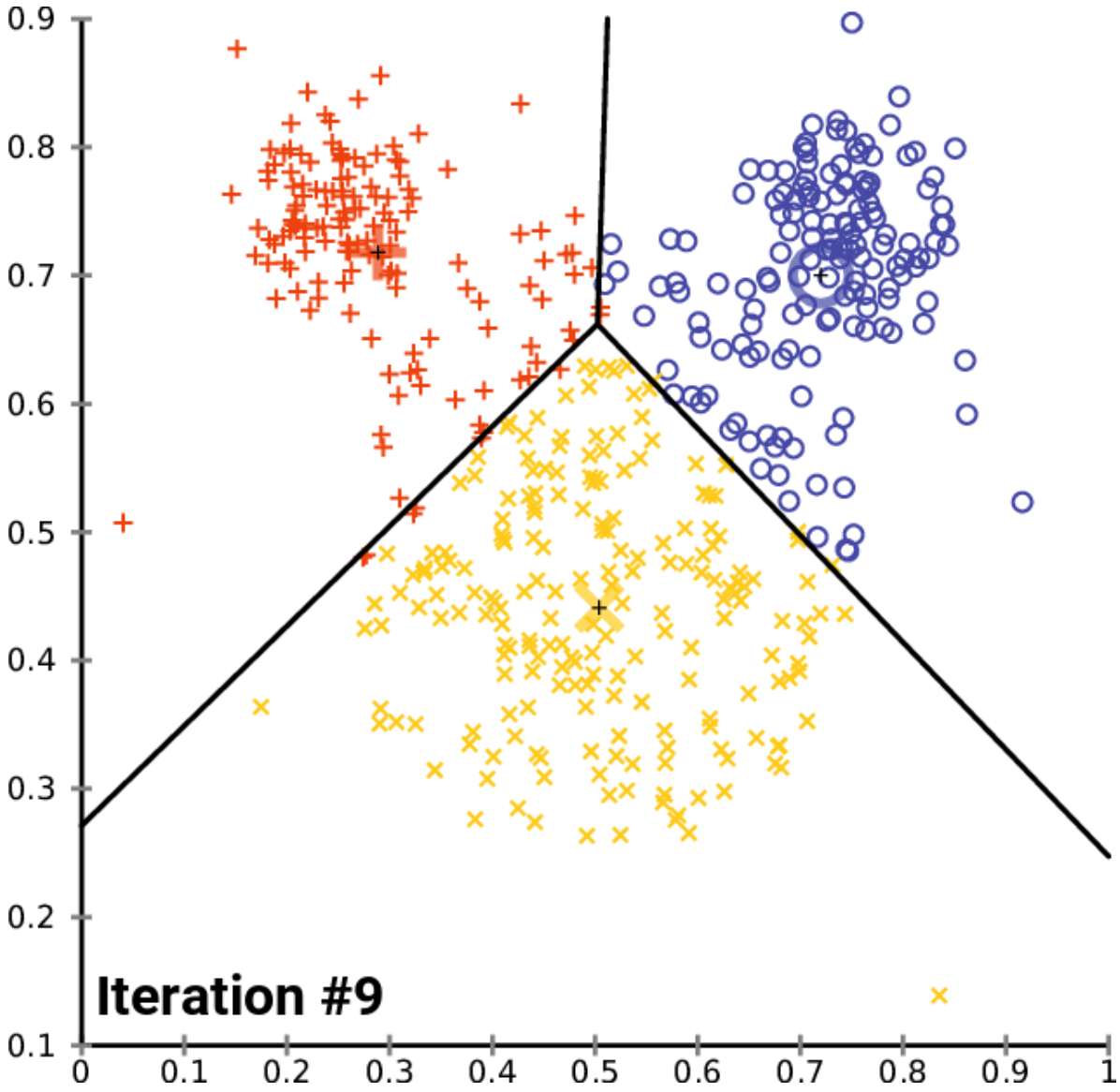
Exemple



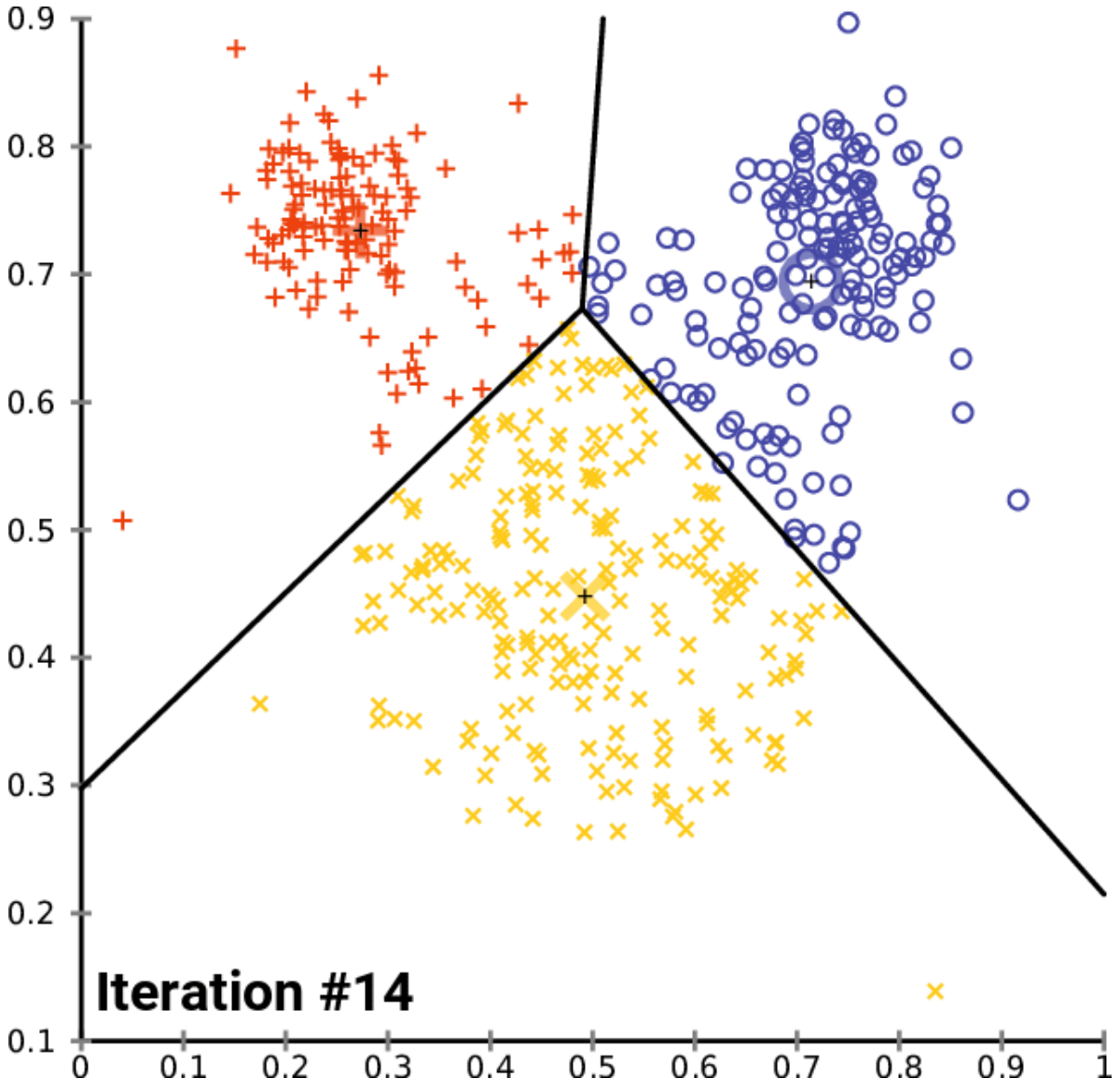
Exemple



Exemple

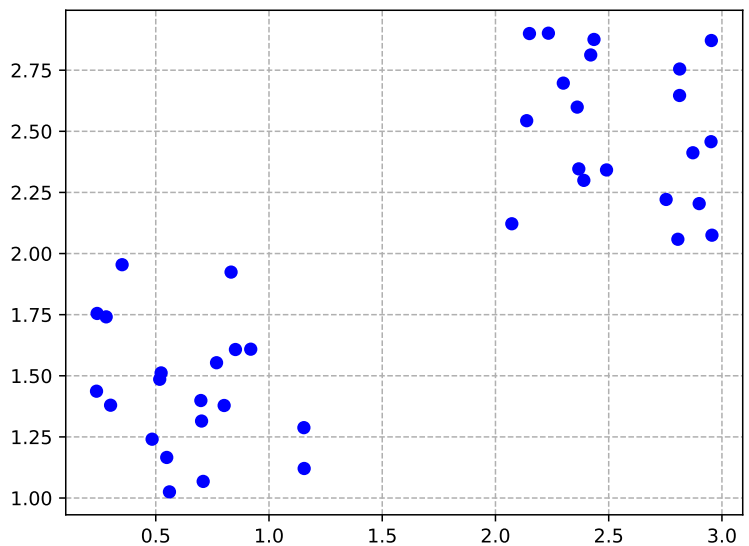


Exemple



Présentation de K-Means *(K-Moyennes)*

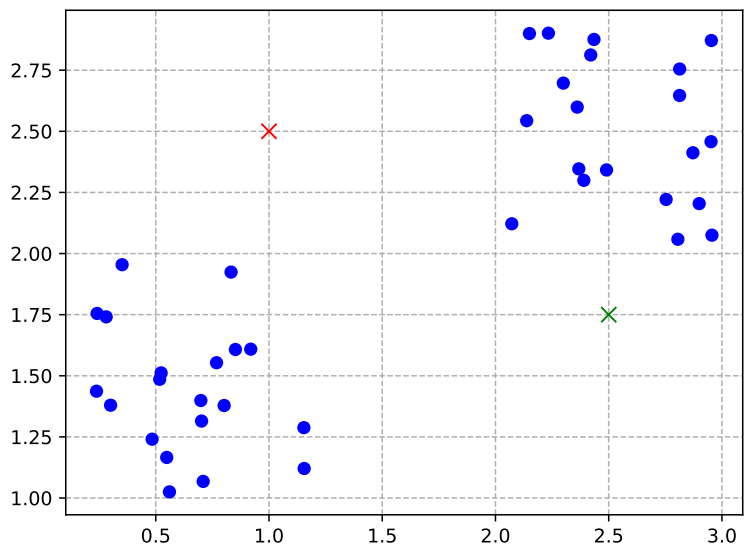
Étapes



Présentation de K-Means *(K-Moyennes)*

Étapes

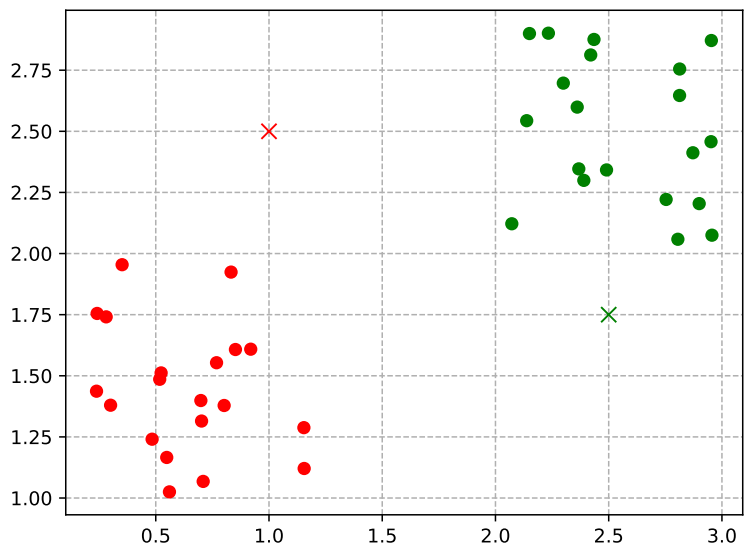
- Choisir des centroïdes aléatoirement



Présentation de K-Means *(K-Moyennes)*

Étapes

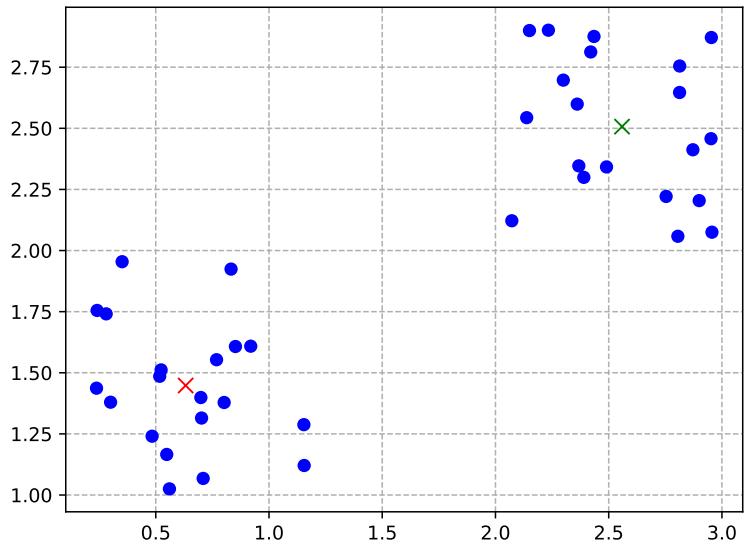
- Choisir des centroïdes aléatoirement
- Affecter les points au centroïde le plus proche



Présentation de K-Means *(K-Moyennes)*

Étapes

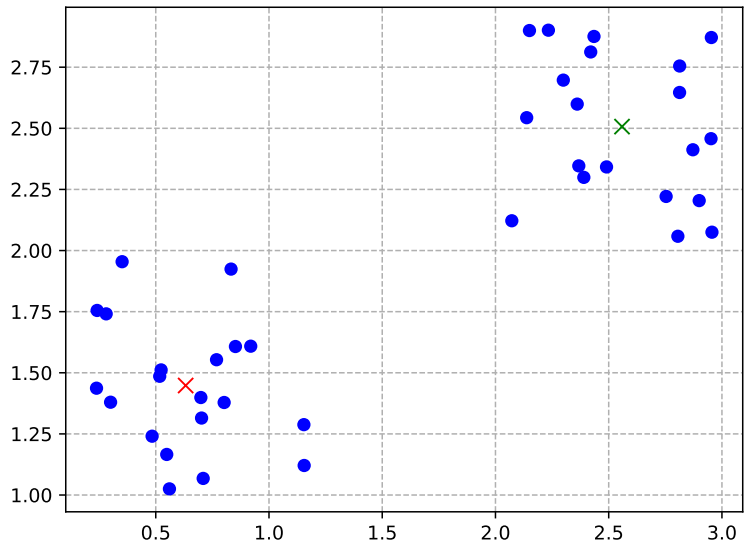
- Choisir des centroïdes aléatoirement
- Affecter les points au centroïde le plus proche
- Déplacer les centroïdes au barycentre de leurs clusters



Présentation de K-Means *(K-Moyennes)*

Étapes

- Choisir des centroïdes aléatoirement
- Affecter les points au centroïde le plus proche
- Déplacer les centroïdes au barycentre de leurs clusters
- Recommencer tant qu'il y a du changement



Algorithme K-Means

Entrées

- K Le nombre de clusters
- Le jeu d'entraînement $x^{(1)}, x^{(2)}, x^{(3)}, \ldots$
- $x^{(i)} \in \mathbb{R}^n$ (pas de $x_0 = 1$ nécessaire ici)

Algorithme K-Means

Algorithme

- Initialiser aléatoirement K centroïdes de clusters $\mu_1, ..., \mu_K \in \mathbb{R}^n$
- Faire
  - Calculer $D_{i,j}$ la distance entre $x^{(i)}$ et μ_j pour tous les points et tous les centroïdes
  - Calculer $C^{(i)}$ l'indice du cluster dont $x^{(i)}$ est le plus proche
  - Calculer les nouveaux μ_k, moyenne des points affectés au cluster k
- Répéter tant qu'au moins un centroïde a bougé

Fonction de coût de K-Means

Définition

- L'algorithme présenté minimise une certaine fonction

$$Cost(C, \mu) = \frac{1}{N} \sum_{i=1}^{N} \left|\left| x^{(i)} - \mu_{C^{(i)}} \right|\right|^2$$

Explications

- Minimise la moyenne...
- ... du carré de la distance...
- ... entre les points et le centroïde qui leur est affecté.

Intuition de la fonction de coût

Intuition

- Bouger les centroïdes au barycentre de leurs clusters fait décroître la moyenne des distances
- Affecter les points au centroïde le plus proche fait décroite la moyenne des distances
- Donc la fonction de coût doit décroître à chaque étape.

K-Means pour TF-IDF

Spécialisation pour TF-IDF : Spherical K-Means

- Les points sont sur une hypersphère
- Utiliser la distance cosinus pour affecter les points aux centroïdes
- Re-normaliser les vecteurs de centroïdes calculé

Questions ?

Questions ?

- Questions ?

TP

TP

- TP