Reconnaissance de Formes et Classification

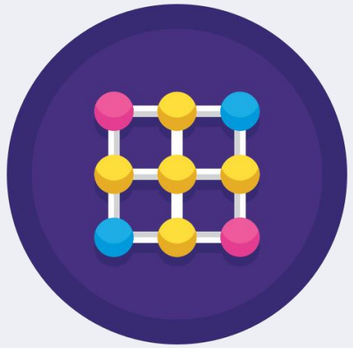
Ce chapitre explore le domaine fascinant de la reconnaissance de formes et de la classification, des concepts fondamentaux dans le traitement du signal et l'apprentissage automatique. Nous découvrirons les principes clés qui sous-tendent la capacité des machines à identifier et à classer des objets, des sons et d'autres formes de données. Au cours de cette exploration, nous examinerons des techniques telles que la programmation dynamique, les k plus proches voisins et les nuées dynamiques, en examinant leurs applications dans des domaines variés tels que la reconnaissance vocale, la vision par ordinateur et l'analyse de données.

INTRODUCTION GENERALE

La reconnaissance des formes (RDF) repose sur le classement des objets ou formes en les comparants à des formes types.

De manière générale, deux types de RDF se distinguent :

- La RDF structurelle qui se base sur une représentation des formes à l'aide des grammaires.
- La RDF statistique qui s'appuie sur une représentation numérique des formes.



Définition d’une Forme

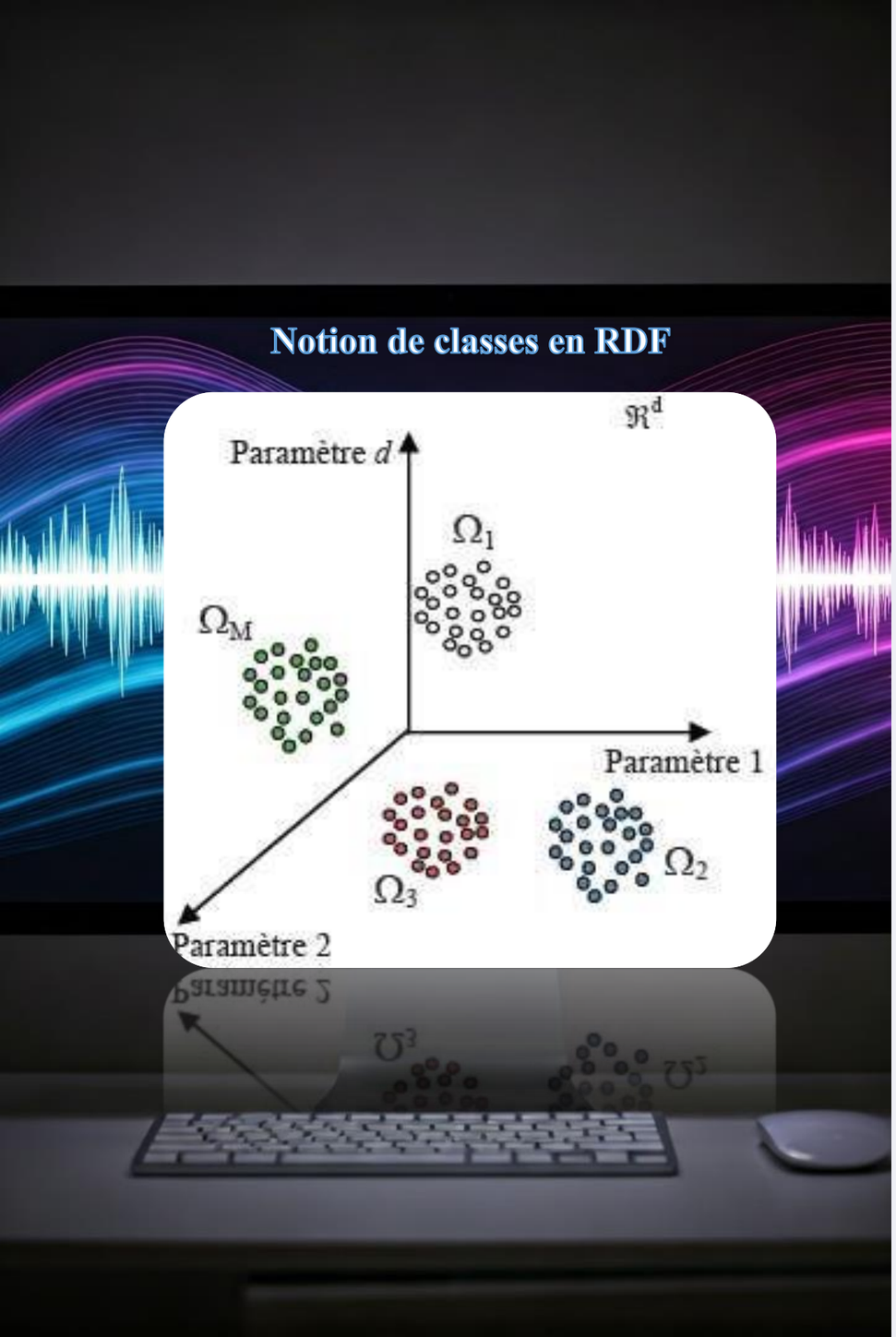
En traitement du signal, une forme est un modèle ou une structure dans les données qui peut être identifiée et caractérisée. Elle peut prendre différentes formes, comme un signal temporel, une image, un texte, ou un ensemble de données numériques. L'idée centrale est de capturer les caractéristiques distinctives qui permettent de distinguer une forme d'une autre.

Signal de parole

Un signal de parole est un signal temporel qui représente les vibrations acoustiques produites par la voix humaine. La forme du signal varie en fonction des sons prononcés, permettant de distinguer les mots, les phrases et les locuteurs.

Image

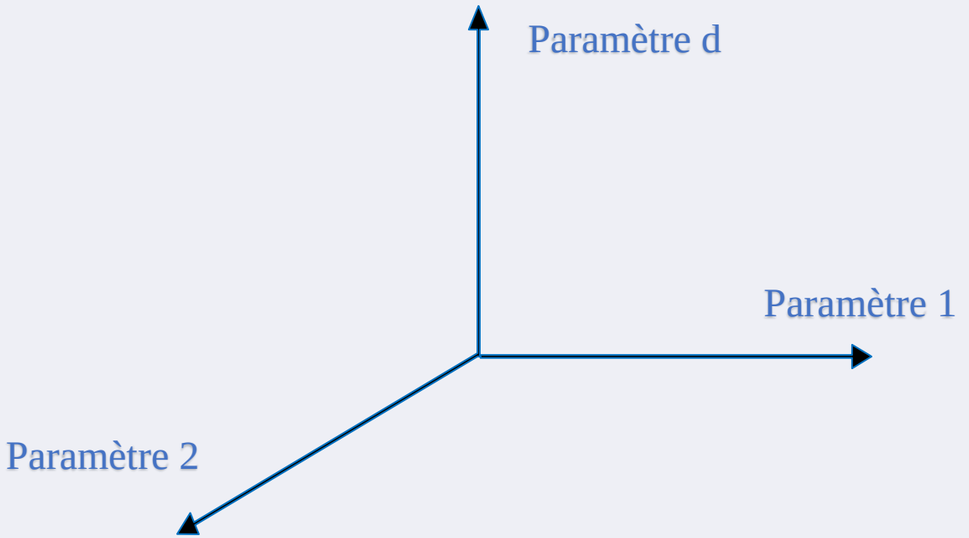
Une image est une représentation bidimensionnelle d'un objet ou d'une scène. La forme de l'image est définie par l'arrangement spatial des pixels, qui reflètent les différentes couleurs et intensités lumineuses de la scène capturée.



Définition d'une Forme

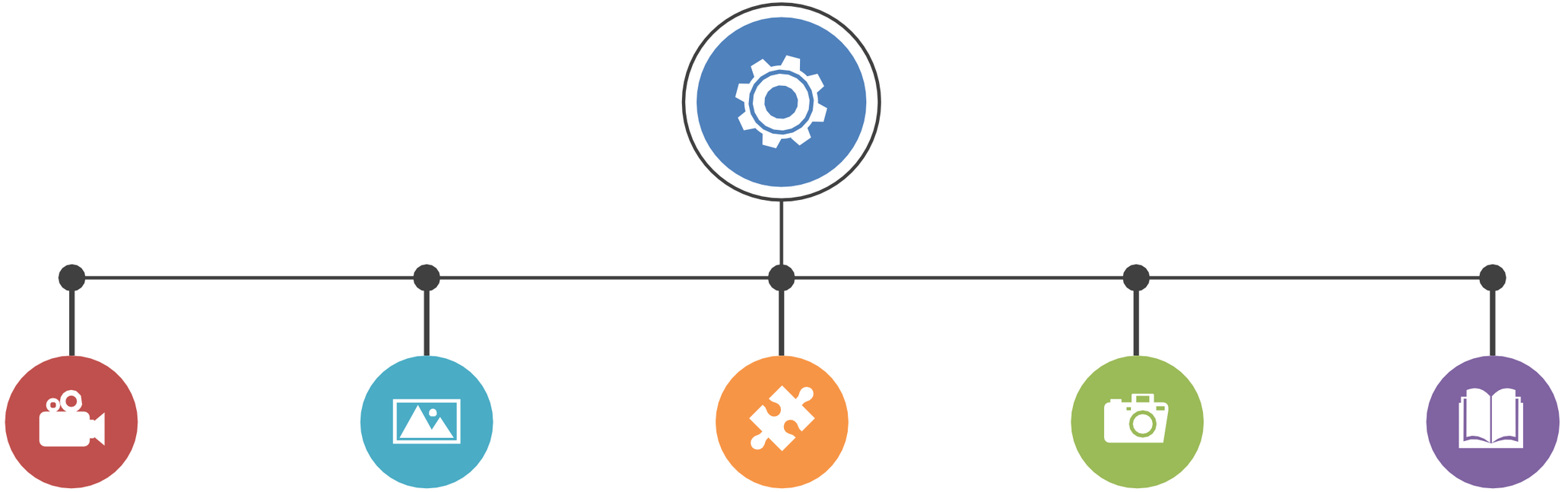
Une forme est une observation réalisée sur le processus. Elle est caractérisée par un ensemble de d paramètres (ou caractère) et représentée par un point dans l'espace de dimension d, défini par les différents paramètres (espace de représentation). Comme les paramètres sont souvent des nombres réels, une forme i peut être définie par un vecteur

$$Xi = [x_i^1, x_i^2, x_i^3, x_i^4, \quad , x_i^d] \in \mathbb{R}^d.$$



Système de RDF

RDF repose sur le classement des objets, plusieurs méthodes ont été développées comme les approches inspirées de Swarm Intelligence (GWO, HHO, GOA, DA, SSA) , de biology (les algorithms génétiques et les réseaux de neurons), les lois de mathématique (SCA, SVM) et physiques (Equilibrium Opttimizer EO) et des modèles probabilistes (Naïve Bayesian), les réseaux de neurones qui cherchent à imiter le cerveau humain ont connu un essor important en commençant par le perceptron multi-couches (MLP) et en concluant par le deep learning ou Convolutional Neural Network (CNN)



Tâche de Reconnaissance de Formes

La tâche de reconnaissance de formes consiste à identifier et à classer des formes à partir de données brutes. Elle implique deux phases essentielles : le prétraitement et la classification.

1

Prétraitement

Le prétraitement vise à nettoyer et à organiser les données brutes afin de les préparer pour la classification. Il peut inclure des étapes comme la réduction du bruit, la normalisation des données et l'extraction de caractéristiques pertinentes.

2

Classification

La classification consiste à attribuer une étiquette ou une catégorie à chaque forme en fonction de ses caractéristiques. Il s'agit de choisir le meilleur modèle qui correspond le mieux à la forme parmi un ensemble de modèles prédéfinis.

Représentation d’une Forme

Avant de pouvoir classer une forme, il faut la représenter sous une forme compréhensible pour les algorithmes de classification. La représentation d'une forme est un ensemble de caractéristiques numériques qui décrivent ses propriétés distinctives.

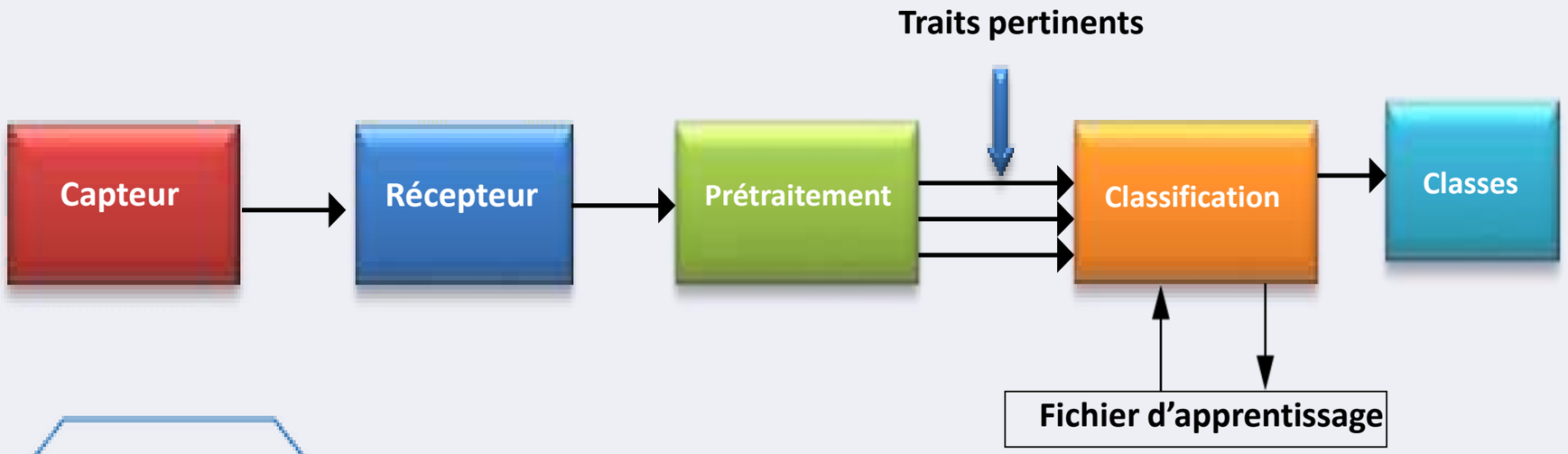
1 Compacité

L'hypothèse de compacité stipule que les formes appartenant à la même classe doivent être similaires les unes aux autres dans l'espace de représentation. En d'autres termes, des formes similaires devraient avoir des représentations proches dans l'espace de caractéristiques.

2 Séparabilité

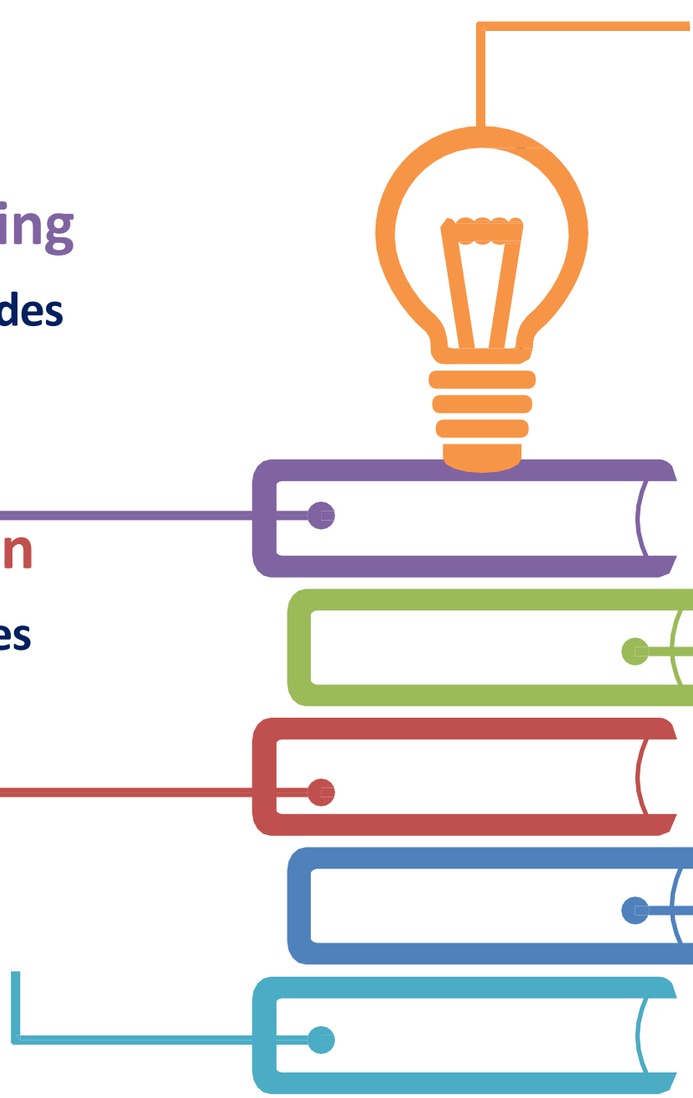
L'hypothèse de séparabilité stipule que les formes appartenant à des classes différentes doivent être distinctes dans l'espace de représentation. En d'autres termes, les formes de classes différentes devraient avoir des représentations éloignées les unes des autres dans l'espace de caractéristiques.

Le schéma de principe d’un système de RDF



Application réelle: classififcation des opinions

Pour réaliser l’étape de prétraitement, il faut convertir le texte en forme numérique au niveau de la phase de prétraitement



5-Stemming

Consiste à représenter l’ensemble des mots dérivés par leurs racines.

4-Elimination des mots vides

consiste à nettoyer le texte des mots vides qui n’ont aucun sens en se basant sur un dictionnaire .

3-Tokenisation

La tokenization consiste à identifier les mots, appelés tokens, à partir de sequence de caractères.

2-Nettoyage

Elimination de tous caractéres qui ne représente pas l’alphabet de la langue étudiée tels que les balises, la ponctuation,les chiffres …

1-Normalisation du texte

Préparation de nos bases de données au format utf-8 Pour faciliter la suite du prétraitement

Distance entre Représentations

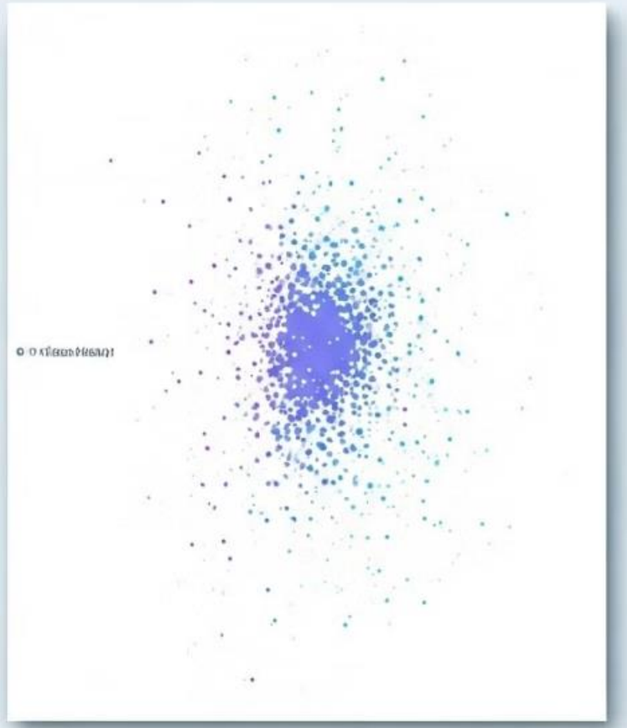
Pour mesurer la similarité ou la différence entre deux formes, on utilise des mesures de distance. La distance entre deux représentations de formes reflète leur dissimilarité. Plus la distance est petite, plus les formes sont similaires.

Distance Métrique

La distance métrique est une mesure de distance qui satisfait à certaines propriétés mathématiques, telles que la symétrie, la positivité et l'inégalité triangulaire. La distance euclidienne est un exemple de distance métrique couramment utilisée dans la reconnaissance de formes.

Distance Binaire

La distance binaire est utilisée pour comparer des représentations de formes binaires. Elle compte le nombre de bits qui diffèrent entre les deux représentations. Cette distance est utilisée pour comparer des images binaires ou des données textuelles binaires.



Les k Plus Proches Voisins (K-ppv)

L'algorithme des k plus proches voisins (K-ppv) est une technique de classification simple et efficace qui se base sur l'idée que des formes similaires devraient être classées de la même manière.

1

Principe

Le principe du K-ppv est de classer une nouvelle forme en fonction de la classe majoritaire parmi ses k plus proches voisins dans l'espace de représentation.

2

Notions de Voisinage

Le voisinage est quantifié par une mesure de similarité, généralement une distance métrique. On cherche les k points les plus proches de la nouvelle forme dans l'espace de représentation.

3

Algorithme

L'algorithme du K-ppv consiste à calculer la distance entre la nouvelle forme et tous les points d'apprentissage. Ensuite, on identifie les k points les plus proches et on détermine la classe majoritaire parmi ces k voisins.

K-ppv

Les k plus proches voisins (K-ppv)

Les k plus proches voisins (K-ppv)

Principe :

Une forme à classer est à comparer à d'autres formes déjà classées (formes de référence, cood bo ok, protot-ype) et on lui affecte la classe la plus représentée parmi les k plus proches.

K=1→ Classe de la forme la plus proche de la forme à classer qui lui est affectée.

Notions de voisinage quantifiées par mesure de similarité :

$$d_E(v_1, v_2) = \sqrt{\sum_{i=1}^{n} (x_{2i} - x_{1i})^2} \ \% \ distance\ euclidienne$$

$$d_H(v_1, v_2) = \sigma_{i=1}^{n} |x_{2i} - x_{1i}| \ \% \ distance\ de\ Hamming\ k = \frac{1}{d}\ [si\ d \searrow \Rightarrow k \nearrow\ k\ croissant]$$

Algorithme :

Initialisation de l'algorithme : choix de :

Nombre de classes.

Valeur de k.

Exemples initiaux.

Mesure de similarité ($d_H(v_1, v_2)$).

Pour chaque vecteur (formes) à classer :

Mesurer la distance du vecteur à classer avec tous les vecteurs déjà classés.

Déterminer les k vecteurs pour lesquels la distance est la plus petite.

Déterminer les classes (qui contiennent le plus de k) et affecter ce vecteur

K-ppv

Les k plus proches voisins (K-ppv)

Propriétés :

- Le cout de calcul augmente avec le nombre de vecteurs déjà classés.
- L'algorithme est sensible à l'initialisation, exemple si les exemples initiaux déjà classés

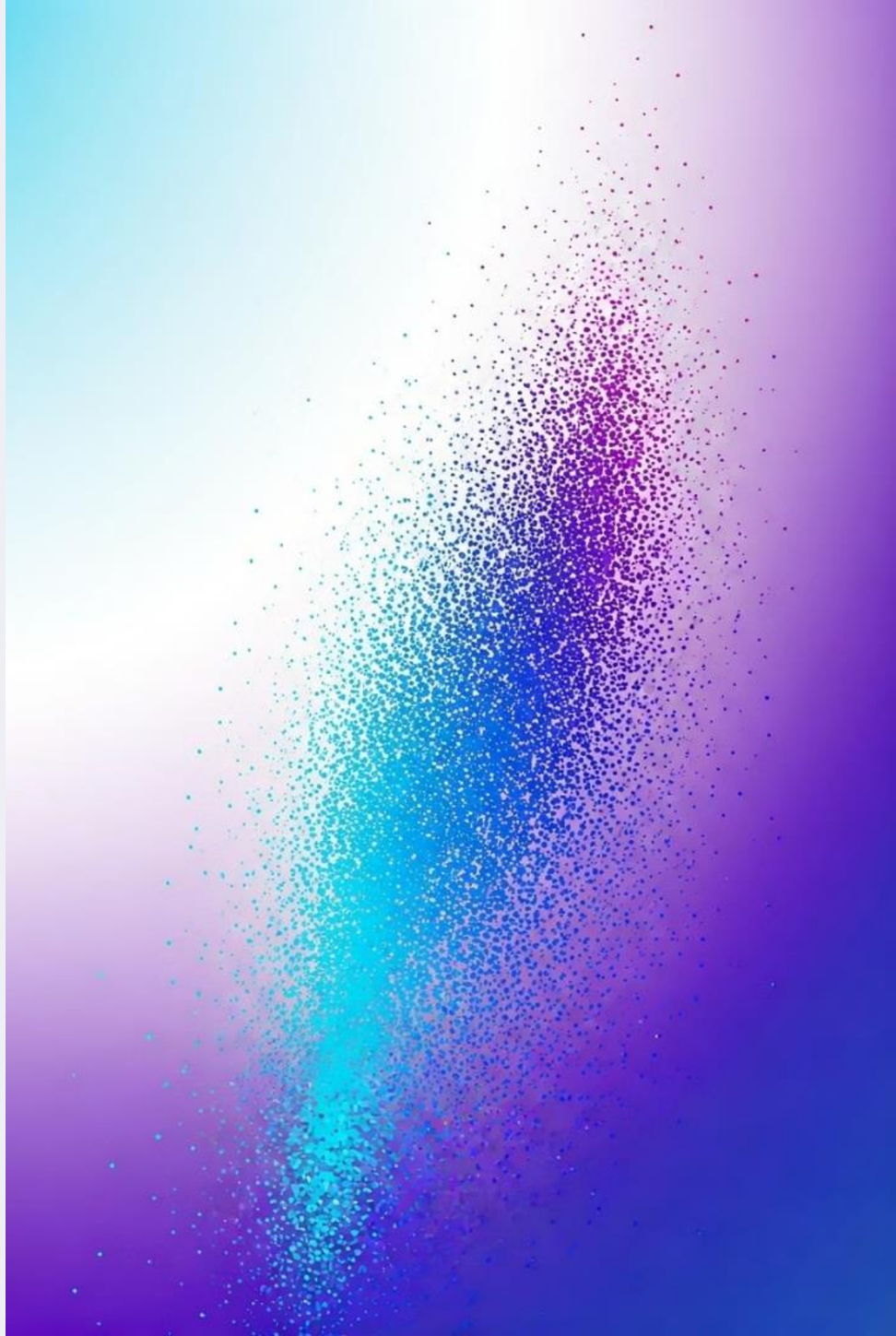
  se situent près de frontières des classes, l'algorithme risque de diverger.
- Le choix de k (Si k est choisi petit : frontières compliquées entre classes définies mais

  sensible au bruit ; si k est choisi grand : frontières lisses entre classes définies et insensible au bruit)

Les Nuées Dynamiques (K-means)

L'algorithme des nuées dynamiques (K-means) est un algorithme de classification non supervisée qui vise à partitionner un ensemble de données en k clusters.

Algorithme	Propriétés
L'algorithme commence par initialiser k centroïdes aléatoirement.	Simple à implémenter.
À chaque itération, chaque point est attribué au cluster dont le centroïde est le plus proche.	Sensible à l'initialisation des centroïdes.
Les centroïdes sont ensuite recalculés en tant que moyenne des points de chaque cluster.	Peut converger vers des solutions locales.



K-means

Les nuées dynamiques (K-means)

Les nuées dynamiques (K-means)

La principale différence avec kppv est que chaque vecteur à classer n'est pas comparé à tous les exem ples déjà classés, mais à **un seul représentatif de chaque classe**.

Chaque classe est représentée par un vecteur (noyau, représentant, prototype).

Représentant égale à la moyenne des vecteurs de cette classe.

Chaque vecteur à classer est comparé à tous les noyaux (distance de Hamming).

Ce vecteur est associé à la classe dont le noyau est le plus proche.

$$m_{n+1} = \frac{n * m_n + x_{n+1}}{n+1}$$

Algorithme :

Initialisation de l'algorithme : choix de :

Nombre de classes.

Représentant initial.

Mesure de similarité ($d_{\boldsymbol{H}}(v_1, v_2)$).

Pour chaque vecteur à classer :

Calcul de la distance entre ce vecteur et les noyaux des autres classes.

Déterminer le représentant correspondant à la distance minimale.

Modifier le noyau pour intégrer le nouveau vecteur affecté à cette classe.

K-means

Les nuées dynamiques (K-means)

Les nuées dynamiques (K-means)

La principale différence avec kppv est que chaque vecteur à classer n'est pas comparé à tous les exem ples déjà classés, mais à **un seul représentatif de chaque classe**.

Chaque classe est représentée par un vecteur (noyau, représentant, prototype).

Représentant égale à la moyenne des vecteurs de cette classe.

Chaque vecteur à classer est comparé à tous les noyaux (distance de Hamming).

Ce vecteur est associé à la classe dont le noyau est le plus proche.

$$m_{n+1} = \frac{n * m_n + x_{n+1}}{n+1}$$

Algorithme :

Initialisation de l'algorithme : choix de :

Nombre de classes.

Représentant initial.

Mesure de similarité ($d_{\boldsymbol{H}}(v_1, v_2)$).

Pour chaque vecteur à classer :

Calcul de la distance entre ce vecteur et les noyaux des autres classes.

Déterminer le représentant correspondant à la distance minimale.

Modifier le noyau pour intégrer le nouveau vecteur affecté à cette classe.

Propriétés :

Méthode moins couteuse en calcul, car chaque vecteur à classer n'est comparé qu'au noyau.

Sensible à l'ordre d'arrivée des données à classer.

Exemple

On suppose que la base contient deux classes contenant 3 documents des opinions positives et 2 documents négatives caractérisées par 2 variables,

1 classifier la forme T à l'aide de 1-NN,

2 Classifier la forme T à l'aide de Kmeans

Classe A : les opinions P+	Classe B: les opinions N-	Données T
$A1=(2, 3)^t$	$B1=(5, 8)^t$	$T1=(5, 5)^t$
$A2=(3, 3.5)^t$	$B2=(6, 8)^t$	$T2=(6, 4)^t$
$A3=(4, 4)^t$	-	-

distH	$A1=(2, 3)^t$	$A2=(3, 3.5)^t$	$A3=(4, 4)^t$	$B1=(5, 8)^t$	$B2=(6, 8)^t$	T1
$T1=(5, 5)^t$	|5-2|+|5- 3|=5	3,5	**2**	3	4	- $T1 \in A$
$T2=(6, 4)^t$	5	3,5	**2**	5	4	**2**

Alors la classe T contient des documents positifs

Exemple

On suppose que la base contient deux classes contenant 3 documents des opinions positifs et 2 documents négatifs caractérisés par 2 variables,

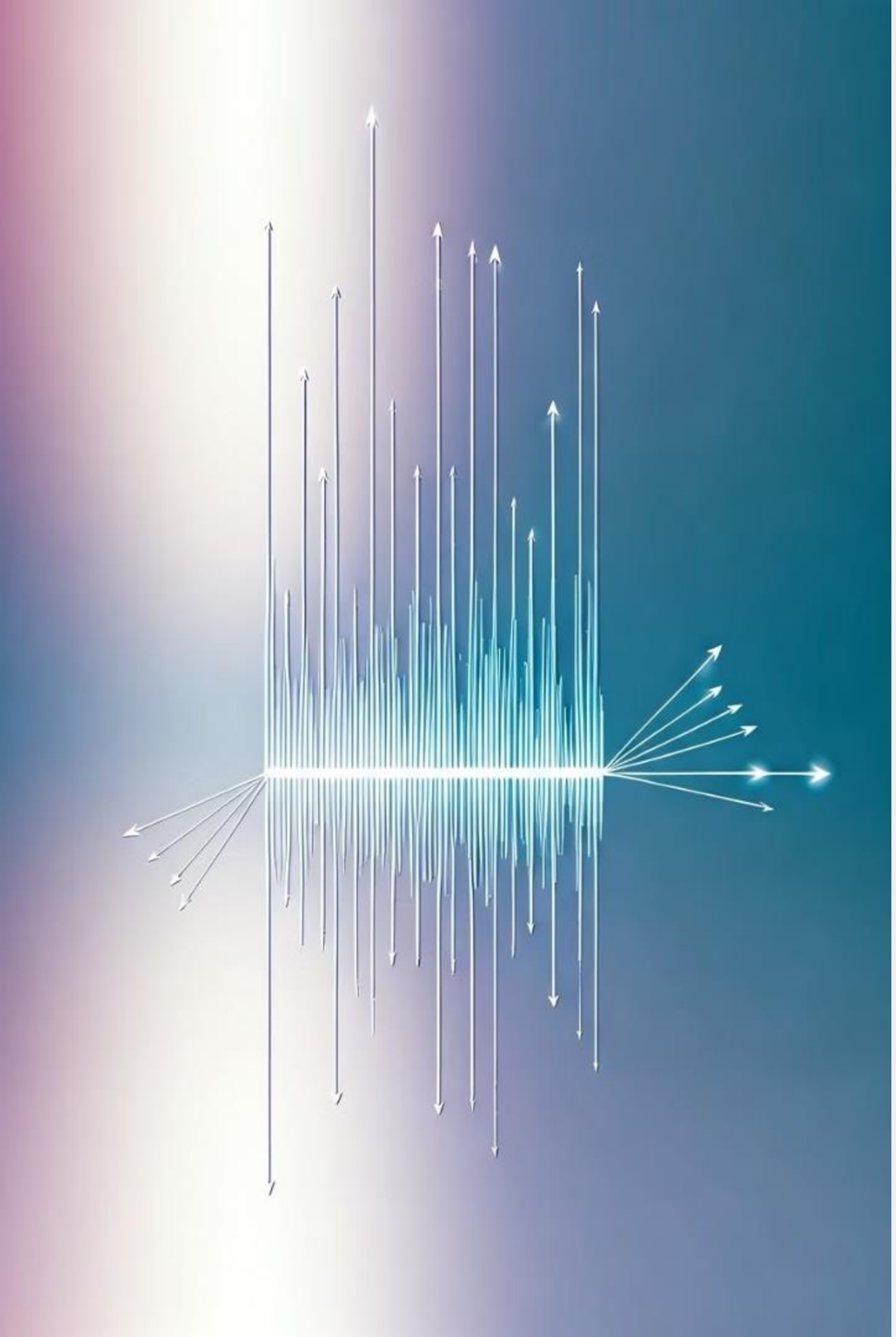
1 classifier la forme T à l'aide de **1-NN**,

2 Classifier la forme T à l'aide de **Kmeans**

Classe A : les opinions P+	Classe B: les opinions N-	Données T
A1=(2, 3)^t	B1=(5, 8)^t	T1=(5, 5)^t
A2=(3, 3.5)^t	B2=(6, 8)^t	T2=(6, 4)^t
A3=(4, 4)^t	-	-

$dist_{H7/}$	NA=((2+3+4)/3, (3+3,5+4)/3)^t =(3 , 3,5) ^{t}	NB=((5+6)/2, (8+8)/2)^t =(5,5 , 8) ^{t}	T1
T1=(5, 5)^t	**|5-3|+|5-3,5|=3,5**	**3,5**	- T1∈A
Mise à jour des centres	N'A=[3*(3 , 3,5) ^{t} +(5, 5)^t] /4= (14/4, 15,5/4) ^{t}	NB=((5+6)/2, (8+8)/2)^t =(5,5 , 8) ^{t}	
T2=(6, 4)^t	|6-3,5|+|4-3,875|=**2,625**	4,5	T2∈A

Alors la classe T contient des documents positifs



Programmation Dynamique

La programmation dynamique est une technique efficace pour résoudre des problèmes d'optimisation qui peuvent être décomposés en sous-problèmes imbriqués. En reconnaissance de formes, la programmation dynamique est utilisée pour le recalage temporel, qui consiste à aligner deux signaux temporels en trouvant le meilleur décalage temporel entre eux.

Exemple

Le recalage temporel est utilisé dans la reconnaissance vocale pour aligner un signal de parole inconnu avec un modèle de mot connu. En trouvant le meilleur alignement temporel, on peut déterminer si le mot prononcé correspond au modèle.

Recalage Temporel

Le recalage temporel vise à trouver le meilleur décalage temporel entre deux signaux temporels en minimisant une fonction de coût, qui mesure la différence entre les deux signaux.

Chemins de Recalage

Les chemins de recalage représentent les différentes combinaisons possibles de décalages temporels entre les deux signaux. Le but est de trouver le chemin de recalage optimal qui minimise la fonction de coût.

Algorithme de la Programmation Dynamique

L'algorithme de la programmation dynamique pour le recalage temporel consiste à calculer récursivement le coût optimal pour aligner les deux signaux jusqu'à un certain point temporel. En utilisant une matrice de coûts, on calcule le coût optimal pour chaque paire de points temporels en tenant compte des coûts des sous-problèmes précédents.

L'algorithme se termine en trouvant le chemin de recalage optimal qui correspond au coût minimal global. Ce chemin représente l'alignement temporel optimal entre les deux signaux.

Algorithme de la programmation dynamique

Algorithme de la programmation dynamique :

1) **Initialisation :** $D_{pp}(1,1) = 2 * d(1,1)$

$$D_{pp}(1,j) = \infty \equiv pour\ j = 2\ à\ J$$

2) **Programmation dynamique**

Pour i de 2 à I Faire

Pour j de 1 à J faire

$$D_{pp}(i,j) = min \begin{cases} D_{pp}(i-1,j) + d(i,j) \\ D_{pp}(i-1,j-1) + 2 * d(i,j) \\ D_{pp}(i,j-1) + d(i,j) \end{cases}$$

3) **Détermination du taux de dissemblance**

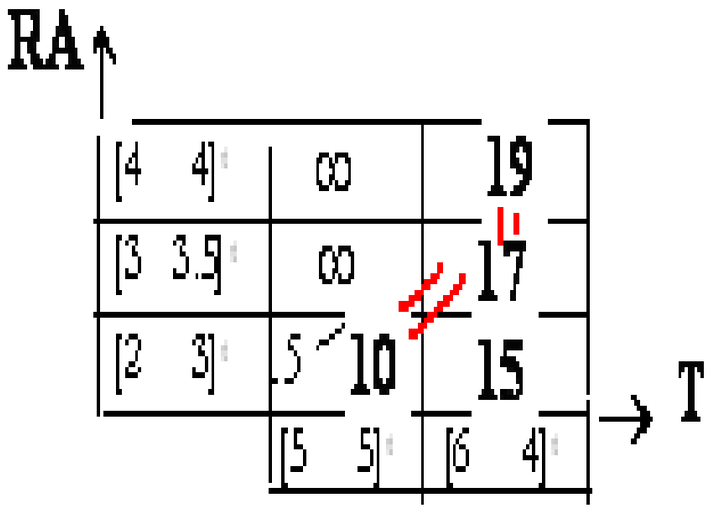
$$D(T,R) = \frac{D_{pp}(I,J)}{I+J}$$

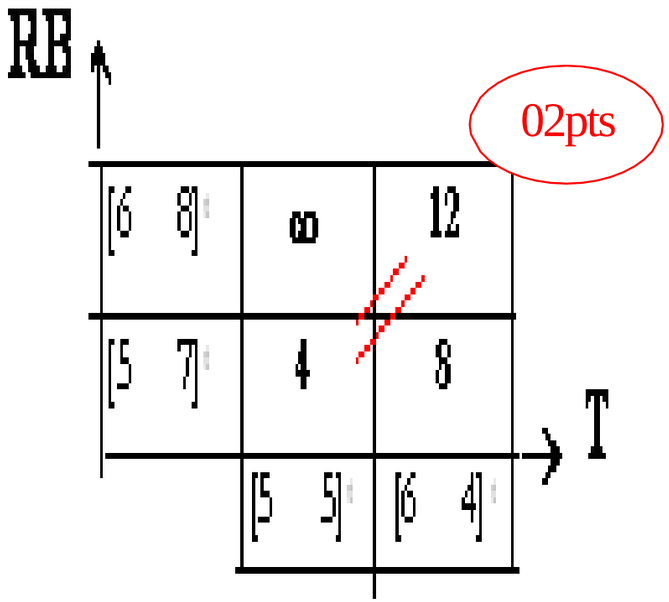
Considérons un problème à deux ensembles de données (A, B) et l'ensemble du test T représentés par le nuage de point suivant dans le plan (x, y) :

$$A=\begin{bmatrix}2 & 3 & 4\\ 3 & 3.5 & 4\end{bmatrix},\ B=\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix} \text{ et } T=\begin{bmatrix}5 & 6\\ 5 & 4\end{bmatrix}$$

1. Par l'algorithme DTW, rechercher et tracer le chemin de recalage optimal entre l'ensemble du test T et l'ensemble d'apprentissage (A, B).

EXAMPLE





1. Evaluer les taux de dissemblances D(T ,A) ,D(T, B) et déduire dans ce cas les résultats de la classification.

D(T ,A)=19/5=3.8

D(T, B)=12/4=3

$D(T,B)<D(T,A)$ alors **T≈B**

2. En utilisant les deux algorithmes : 1-means, déterminer la nouvelle composition pour chaque classe (En traitant par ordre les vecteurs du Test T).

EXAMPLE

1-means

[5 5]^t	NA=[3 3.5]^t	NB=[5.5 7.5]^t
	5 NA=[3 3.5]^t	**3**
[6 4]^t	3.5	NB'=[16/3 20/3]^t **10/3**

T≈B 0.5pt

L'Algorithme LBG

L'algorithme **LBG (**L**inde-**B**uzo-**G**ray)** est une méthode utilisée principalement pour le vecteur de quantification dans le traitement du signal, notamment dans la compression de données ou la reconnaissance de la parole. Cet algorithme permet de partitionner l'espace de données en clusters et de trouver les **centroïdes** optimaux pour minimiser l'erreur de quantification. Voici une description détaillée de l'algorithme avec les termes epsilon et d0.

Principe de L'Algorithme LBG

L'algorithme LBG est basé sur un processus itératif de division et de raffinement des **centroïdes**, ou "codebooks", jusqu'à ce que la différence entre deux itérations successives devienne négligeable, selon un seuil donné.

1 Itérations

L'algorithme LBG fonctionne en itérant sur plusieurs étapes, en affinant progressivement les centroïdes et en réduisant l'erreur de quantification.

2 Division

À chaque itération, les centroïdes existants sont divisés en deux nouveaux centroïdes, créant ainsi une partition plus fine de l'espace de données.

3 Raffinement

Après la division, les centroïdes sont raffinés en recalculant leur position en fonction des vecteurs de données qui leur sont associés.

4 Convergence

L'algorithme converge lorsque la différence entre les distorsions de deux itérations successives devient inférieure à un seuil donné.

Étapes de L'Algorithme LBG

L'algorithme LBG est composé de plusieurs étapes clés, qui sont répétées itérativement jusqu'à ce que la convergence soit atteinte.

1. **Initialisation**

   Calculer le **centre de gravité** C0 des données, qui est la moyenne de tous les vecteurs d'entraînement. Ce centre de gravité est initialement le premier centroïde du codebook.

2. **Division des centroïdes**

   Un petit facteur ϵ est utilisé pour perturber les centroïdes à chaque étape afin de créer une nouvelle partition de l'espace. Par exemple, si le centroïde est Ci, on génère deux nouveaux centroïdes Ci+=Ci+ϵ et Ci−=Ci−ϵ. Cela permet de doubler le nombre de clusters à chaque itération.

3. **Quantification**

   Attribuer chaque vecteur d'entraînement au centroïde le plus proche selon une métrique donnée (comme la distance euclidienne).

4. **Mise à jour des centroïdes**

   Pour chaque cluster, recalculer les centroïdes en prenant la moyenne des vecteurs qui lui ont été assignés.

5. **Convergence**

   Calculer la **distorsion moyenne** (l'erreur de quantification), généralement définie comme la somme des distances entre chaque vecteur et son centroïde assigné. Si la différence entre la distorsion actuelle Dn et la distorsion précédente Dn−1 est inférieure à **un seuil donné** (ou si un nombre maximum d'itérations est atteint), l'algorithme converge.

Explication des Termes ε Et D0

Les termes epsilon (ϵ) et distorsion initiale (D_0) jouent un rôle crucial dans l'algorithme LBG. Ils permettent de contrôler la précision et la convergence de l'algorithme.

Epsilon ϵ

ϵ est un petit facteur perturbateur, typiquement un nombre très petit, qui est utilisé pour légèrement modifier les centroïdes à chaque étape de la division. Il permet d'éviter que tous les centroïdes ne convergent vers le même point au départ. Par exemple, ϵ=0.01 est couramment utilisé dans les implémentations de l'algorithme LBG. Le choix de ϵ influence la précision et la rapidité de la convergence de l'algorithme. Un ϵ trop grand pourrait entraîner une mauvaise convergence, tandis qu'un ϵ trop petit ralentit le processus.

Distorsion Initiale D_0

La **distorsion** est une mesure de l'erreur de quantification, définie comme la somme des distances entre les vecteurs d'entraînement et les centroïdes qui leur sont assignés. D0 est la distorsion calculée pour le premier centroïde initial (avant toute division). Elle sert de référence pour suivre la diminution de l'erreur de quantification au cours des itérations. L'algorithme LBG essaie de minimiser cette distorsion, et il s'arrête lorsque la différence de distorsion entre deux itérations est inférieure à ϵ.

Algorithme LBG - Résumé des Étapes Avec ε Et D_0

L'algorithme LBG est un processus itératif qui utilise les paramètres ϵ et D0 pour contrôler la division des centroïdes et la convergence de l'algorithme.

1

Initialisation

Calculez C0 (centre de gravité) et la distorsion initiale D0.

2

Division

Perturbez C0 pour générer deux nouveaux centroïdes C1=C0+ϵet C2=C0−ϵ.

3

Quantification et réaffectation

Attribuez les vecteurs aux nouveaux centroïdes en minimisant la distance.

4

Mise à jour des centroïdes et distorsion

Recalculez les centroïdes des nouveaux clusters. Calculez la nouvelle distorsion Dn.

5

Convergence

Répétez les étapes 2 à 4 jusqu'à ce que |Dn−Dn−1|

Applications de L'Algorithme LBG

L'algorithme LBG est largement utilisé dans diverses applications de traitement du signal, notamment la compression de données, la reconnaissance de la parole et la classification d'images.

Compression de données

L'algorithme LBG est utilisé pour réduire la taille des fichiers de données, tels que les images, les audios et les vidéos, en quantifiant les données et en réduisant le nombre de bits nécessaires pour les représenter.

Reconnaissance de la parole

L'algorithme LBG est utilisé pour segmenter et quantifier les signaux vocaux, ce qui permet de créer des modèles acoustiques pour la reconnaissance de la parole.

Classification d'images

L'algorithme LBG est utilisé pour segmenter et quantifier les images, ce qui permet de créer des modèles visuels pour la classification d'images.

Avantages De L'Algorithme LBG

L'algorithme LBG présente plusieurs avantages par rapport à d'autres méthodes de quantification vectorielle.

Vitesse

L'algorithme LBG est relativement rapide et efficace, ce qui le rend adapté aux applications en temps réel.

Précision

L'algorithme LBG peut atteindre une précision élevée dans la quantification vectorielle, ce qui permet de minimiser la perte de données.

Efficacité

L'algorithme LBG est efficace en termes de ressources informatiques, ce qui le rend adapté aux applications avec des contraintes de mémoire et de puissance de calcul.

Limites de L'Algorithme LBG

Malgré ses avantages, l'algorithme LBG présente également certaines limites.

Sensibilité aux conditions initiales

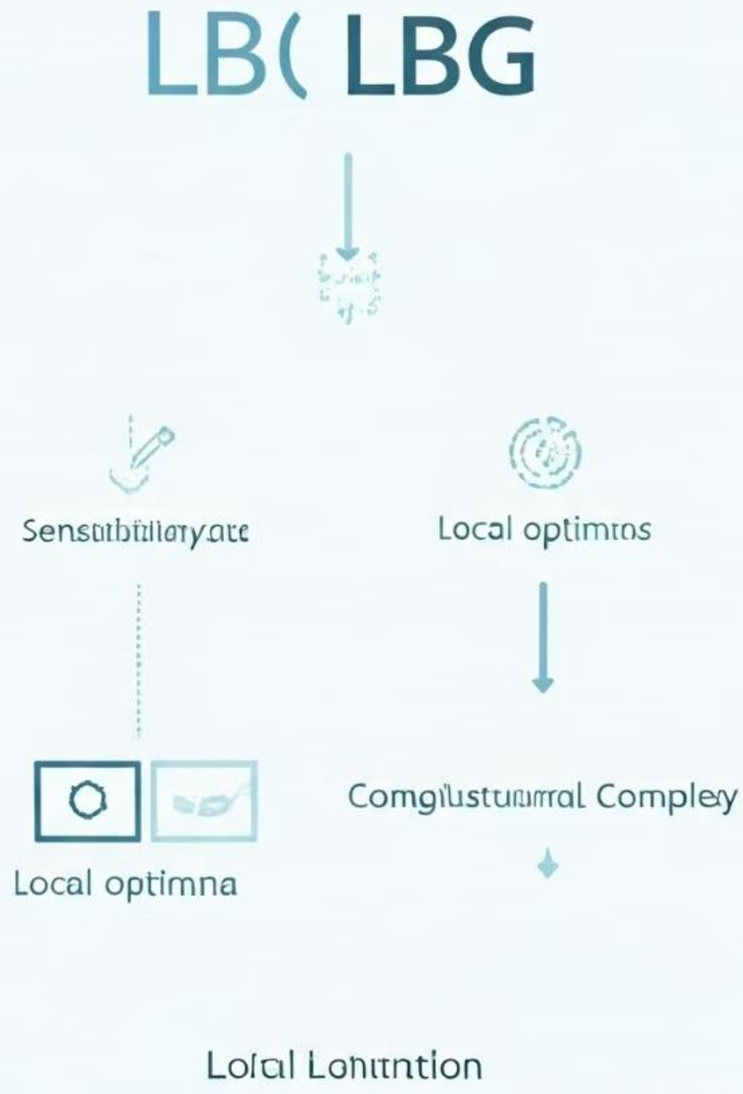
La convergence de l'algorithme LBG peut dépendre des conditions initiales, ce qui peut entraîner des résultats différents pour des initialisations différentes.

Complexité computationnelle

L'algorithme LBG peut être coûteux en termes de calcul pour des ensembles de données volumineux, en particulier pour des dimensions élevées.

Optima locaux

L'algorithme LBG peut converger vers un optimum local plutôt qu'un optimum global, ce qui peut entraîner une erreur de quantification plus élevée.



Conclusion

L'algorithme LBG est un outil puissant pour la quantification vectorielle, offrant une solution efficace et précise pour la compression de données, la reconnaissance de la parole et d'autres applications de traitement du signal. Cependant, il est important de tenir compte de ses limites, telles que la sensibilité aux conditions initiales et la possibilité de converger vers des optima locaux.

On considère l'ensemble suivant :

$$E = \left\{ \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}; \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}; \begin{bmatrix} -0.8 \\ -0.2 \end{bmatrix}; \begin{bmatrix} -0.2 \\ -0.8 \end{bmatrix}; \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}; \begin{bmatrix} 0.6 \\ -0.2 \end{bmatrix} \right\}$$

- Appliquer l'algorithme **LBG**, ***en considérant la valeur d'éclatement***

$$\varepsilon = 0.03 \quad \textbf{\textit{et le seuil}} \quad s = 0.1$$

EXAMPLE

	Centre de gravité global	***Centre de gravité +ε***	***Centre de gravité -ε***	
	$N_g = (0.13 \quad 0.03)^t$	$N_{g+\varepsilon} = (0.16 \quad 0.06)^t$	$N_{g-\varepsilon} = (0.1 \quad 0)^t$	
$T_1 = (0.2 \quad 0.8)$	$\|0.2-0.13\|+\|0.8-0.03\|=0.84$	$\|0.2-0.16\|+\|0.8-0.06\|=0.78$	$\|0.2-0.1\|+\|0.8-0\|=0.9$	**T1∈C1**
$T_2 = (0.8 \quad 0.2)$	$\|0.8-0.13\|+\|0.-0.03\|=0.84$	$\|0.8-0.16\|+\|0.2-0.06\|=0.78$	$\|0.8-0.1\|+\|0.2-0\|=0.9$	**T2∈C1**
$T_3 = (-0.8 \quad -0.2)^t$	$\|-0.8-0.13\|+\|-0.2-0.03\|=1.16$	$\|-0.8-0.16\|+\|-0.2-0.06\|=1.22$	$\|-0.8-0.1\|+\|-0.2-0\|=1.1$	**T3∈C2**
$T_4 = (-0.2 \quad -0.8)^t$	$\|-0.2-0.13\|+\|-0.8-0.03\|=1.16$	$\|-0.2-0.16\|+\|-0.8-0.06\|=1.22$	$\|-0.2-0.1\|+\|-0.8-0\|=1.1$	**T4∈C2**
$T_5 = (0.2 \quad 0.4)^t$	$\|0.2-0.13\|+\|0.4-0.03\|=0.44$	$\|0.2-0.16\|+\|0.4-0.06\|=0.38$	$\|0.2-0.1\|+\|0.4-0\|=0.5$	**T5∈C1**
$T_6 = (0.6 \quad -0.2)^t$	$\|0.6-0.13\|+\|-0.2-0.03\|=0.7$	$\|0.6-0.16\|+\|-0.2-0.06\|=0.7$	$\|0.6-0.1\|+\|-0.2-0\|=0.7$	**T6∈C1 ou T6∈C2**

EXAMPLE

Calcul de D_0

$$D_0 = \frac{\sum d(t_i, N_g)}{nbrTotald'individus} = \frac{0.84+0.84+1.16+1.16+0.44+0.7}{6} = 0.86$$

Calcul de D_1

$$D_1 = \frac{\sum d_{\min}}{nbrTotald'individus} = \frac{0.78+0.78+1.1+1.1+0.38+0.7}{6} = 0.81$$

$$\frac{D_0 - D_1}{D_1} = \frac{0.86 - 0.81}{0.81} = 0.06 \cdot 0.1$$ ----→ ***Arrêt***

Alors C1 contient T1,T2,T5, T6 et C2 contient T2,T4 ou

Alors C1 contient T1,T2,T5 et C2 contient T2,T4,T6