Méthodes non-paramétriques

Approche par Exemplification

Méthodes non-paramétriques

k-Nearest Neighbour : KNN

k-plus proches voisins : KPPV

Les k plus proches voisins (kPPV)

Principe :

- La forme à classer est comparée aux autres formes ayant déjà été classées, et on lui affecte la classe la plus représentée parmi les k plus proches d'elle.
- Dans le cas particulier k=1, c'est la classe de la forme la plus proche de la forme à classer qui est affectée à cette dernière.
- La notion de proximité utilisée est quantifiée par une mesure de similarité. La mesure de similarité la plus utilisée est la distance euclidienne.
- Pour que l'algorithme puisse démarrer, il faut utiliser un certain nombre d'exemples dont la classe est connue, auxquels vont être comparés les premiers vecteurs à classer.

Les k plus proches voisins (kPPV)

Paramètres :

Les différents paramètres de l'algorithme, réglables par l'utilisateur, sont :

- le nombre d'exemples dont la classe est connue, utilisés au démarrage de l'algorithme
- la valeur de k
- la mesure de similarité (ex. : distance euclidienne)

Les k plus proches voisins (kPPV)

Algorithme :

Choisir des exemples initiaux, dont la classe est connue

Pour chaque vecteur à classer :

- Mesurer la similarité entre le vecteur à classer et tous les vecteurs déjà classés
- Déterminer les k vecteurs pour lesquels la similarité est la plus grande (=kPPV)
- Déterminer la classe la plus représentée parmi ces k vecteurs et affecter cette classe au vecteur à classer

Les k plus proches voisins (kPPV)

Propriétés (1) :

- Le coût de calcul augmente avec le nombre de vecteurs déjà classés, ce qui représente un inconvénient.
- Le résultat de classification dépend de l'ordre de présentation des exemples, d'où l'importance de faire une initialisation correcte.
- Le choix de k constitue un compromis : une petite valeur permet de définir des frontières compliquées entre les classes, mais présente une grande sensibilité au bruit ; une grande valeur permet de définir des frontières lisses et de présenter une insensibilité au bruit, mais ne permet pas de traiter le cas de classes possédant un effectif réduit (car dans ce cas il faut beaucoup d'exemples initiaux).

Les k plus proches voisins (kPPV)

Propriétés (2) :

- K=1 :
  - frontières des classes très complexes très sensible aux fluctuations des données (variance élevée).
  - risque de sur-ajustement.
  - résiste mal aux données bruitées.
- K=n :
  - frontière rigide moins sensible au bruit.
  - plus la valeur de k est grande plus la résultat d'affectation est bien réalisée

Les k plus proches voisins (kPPV)

Propriétés (3) :

–La règle d'affectation définie dans l'algorithme comporte de nombreux cas indécidables. Par exemple, les cas où il n'y a pas une classe plus représentée que les autres dans les k plus proches voisins (exemple : k=3 et nombre de classes =3 également). Dans de tels cas, on peut soit imposer que la valeur de k respecte quelques contraintes par rapport au nombre de classes (par exemple il ne doit pas être un multiple du nombre de classes), ou choisir une règle de décision supplémentaire, comme par exemple les distances minimales.

–Il peut être démontré, d'une manière théorique, que cette méthode est optimale quand les données sont gaussiennes .

Les k plus proches voisins (kPPV)

Avantages :

- Il n'y a pas de modèle à apprendre : Simplicité, pas d'apprentissage d'un modèle (lazy learning)
- pas d'hypothèse sur les distributions
- Apprentissage rapide, Méthode facile à comprendre, Adapté aux domaines où chaque classe est représentée par plusieurs prototypes et où les frontières sont irrégulières (ex. Reconnaissance de chiffre manuscrits ou d'images satellites)
- Incrémentalité (garder à disposition les individus de la base)
- Bonnes performances en général :
  - Err(1-ppv) < 2 x Err(Modèle bayesien idéal)

Les k plus proches voisins (kPPV)

Inconvénients :

- Nécessité de garder sous la main la base de données : demande le stockage et l'accès à toutes les données (le nombre de données peut être très très grand).
- Lenteur de prédiction (passage en revue de tous les individus de la base à chaque fois) : le coût de calcul peut être très important.
- Paramétrage difficile (choix de la taille du voisinage).
- Impossibilité d'interprétation d'un classement proposé.
- Sensibilité à la dimensionnalité (et aux variables non pertinentes et corrélés).

Les k plus proches voisins (kPPV)

Ou encore !!!

- Traitement des variables discrètes (codage 0/1 ou axes factoriels).
- Choix de la distance pèse sur les résultats.
- Attention aux problèmes d'échelle si distance euclidienne utilisée.
- Pondérer l'influence des observations selon leur éloignement dans le voisinage.

Méthodes non-paramétriques

DTW : Dynamic Time Warping

DTW : Dynamic Time Warping

Déformation(Distorsion) temporelle dynamique

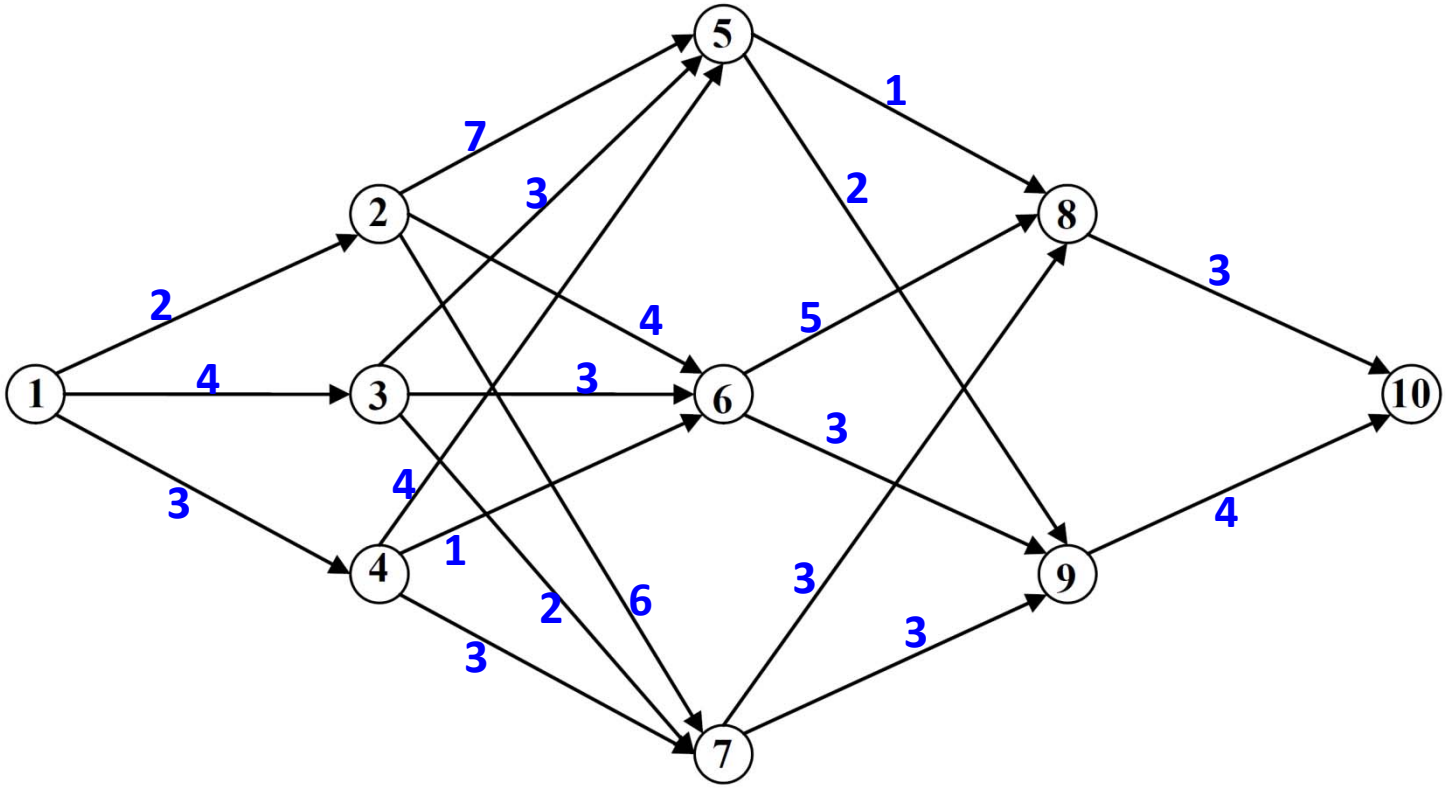
Principe d'optimalité de Bellman

La programmation dynamique: consiste à résoudre un problème en le décomposant en sous-problèmes, puis à résoudre les sous-problèmes, des plus petits aux plus grands en stockant les résultats intermédiaires.

Méthode algorithmique pour résoudre des problèmes d'optimisation, grand succès dès son apparition en 1950 par **Richard Bellman**.

DTW : Dynamic Time Warping

Exemple : Un avion doit faire une séquence de vols le conduisant de la ville 1 à la ville 10 avec 4 escales. Les lieux d'escales possibles sont décrits par le schéma suivant.

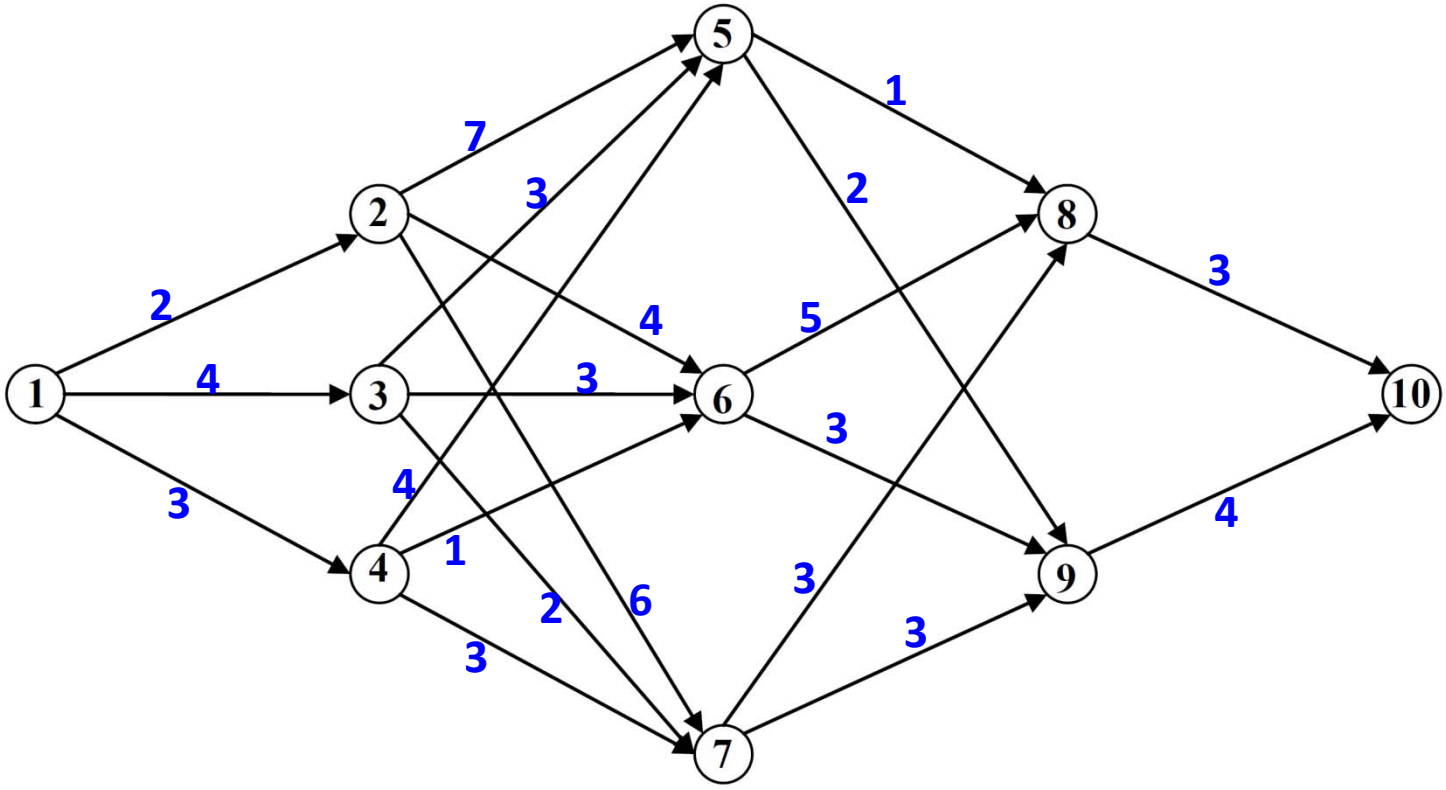


A chaque trajet (i,j) est associé un coût : c_{ij}

DTW : Dynamic Time Warping

Solution : La valeur optimale du critère est donc 11. On retrouve les trajets fournissant cette valeur :

(1, 3, 5, 8, 10) ou (1, 4,6, 9, 10) ou (1, 4, 5, 8, 10).



DTW : Dynamic Time Warping

Introduction :

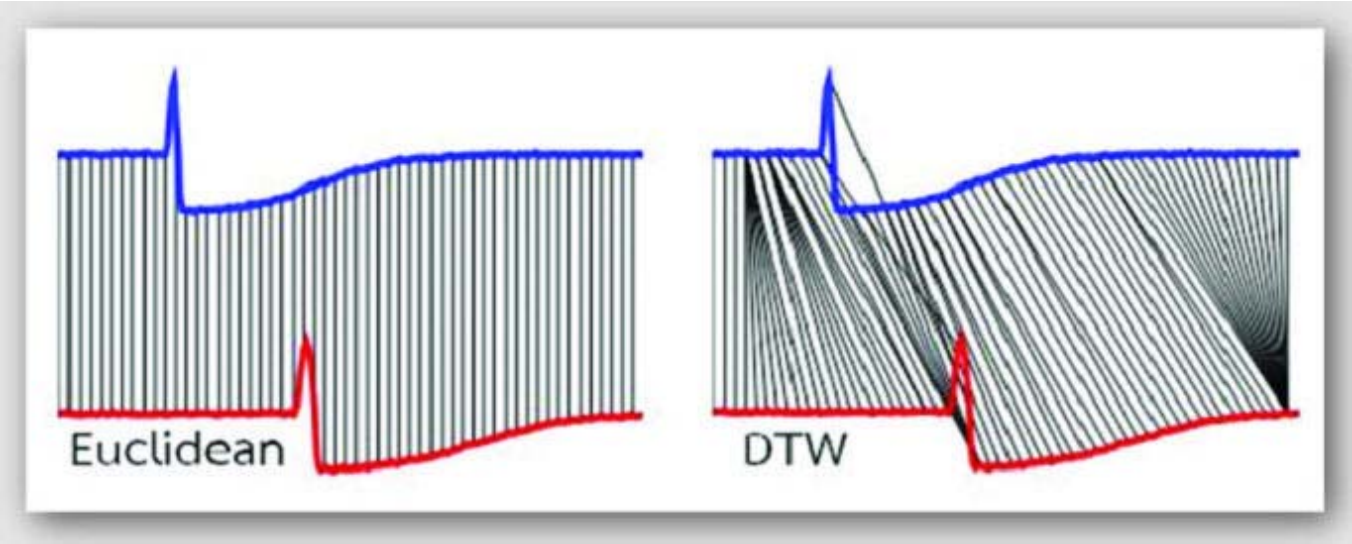
—Soient deux mots T et R sous formes d'une suite de vecteur : t(0),t(1),…,t(I) et r(0),r(1),…,r(J)

—Si la longueur de I est différente de J, il est impossible d'utiliser une formule de distance entre ces deux vecteurs.

—nous avons besoin d'une méthode plus flexible permettant de trouver la meilleure correspondance entre les éléments de T et ceux de R.

DTW : Dynamic Time Warping

Introduction :

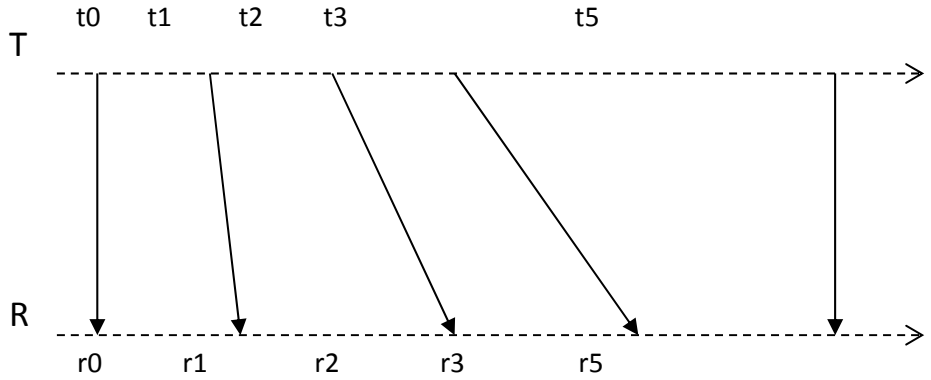


DTW est un algorithme permettant de mesurer la similarité entre deux suites ***<u>qui peuvent varier au cours du temps.</u>***

DTW : Dynamic Time Warping

Recalage Temporel : propose d'effectuer une synchronisation de deux formes à comparer.

Soient deux mots T et R sous formes d'une suite de vecteur : t(0),t(1),…,t(I) et r(0),r(1),…,r(J)



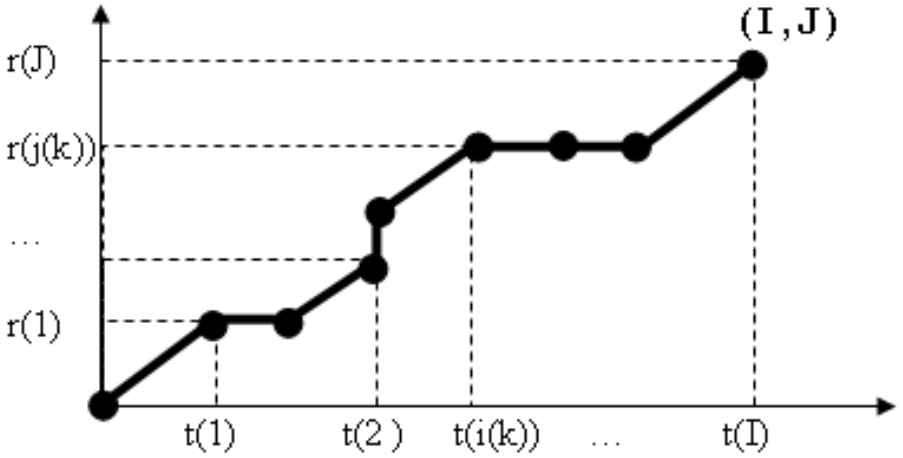
Le recalage temporel établie et optimise la correspondance entre ces vecteurs.

DTW : Dynamic Time Warping

Le recalage temporel établie et optimise la correspondance entre ces vecteurs.

$$(..., I+J) \longrightarrow (1.., I) \times (1.., J)$$

$$K \xrightarrow{w} w(k) = (i(k),\ j(k))$$



Le problème consiste à trouver $\hat{w}$ optimal qui met en coïncidence le *i(k)* prélèvement de T avec le j(k) prélèvement de R.

DTW : Dynamic Time Warping

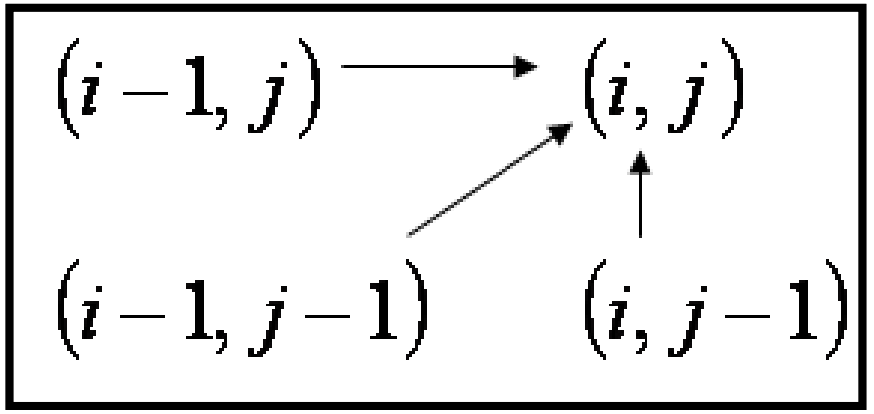
Contraintes sur le chemin de recalage : doit vérifier certaines contraintes :

monotonie : fonction de recalage doit être monotone et croissante :

- $i(k-1) \leq i(k)$
- $j(k-1) \leq j(k)$

un chemin de recalage ne peut aboutir au $pt(i, j)$ qu'en passant obligatoirement par les chemins locaux :

$(i-1, j)$
$(i, j-1)$
$(i-1, j-1)$



DTW : Dynamic Time Warping

Utilisation de la programmation dynamique :

Trouver **ŵ** se résume à une problème de minimisation $D(w)$.

$$D_N(w) = D(i(k), j(k)) = \frac{\sum_{1}^{k} d(i(k), j(k)).P(k)}{N(P)}$$ où :

k : nombre de point du chemin de recalage.

$d(i, j)$: distance locale entre ième vecteur de T et jème vecteur de R.

$P(k)$: pondération qui diffère selon la transition.

$N(P)$: facteur de normalisation $\frac{1}{I + J}$

DTW : Dynamic Time Warping

Pour effectuer une reconnaissance, il est plus important de déterminer le taux de dissemblance $D(T,R)$.

$$D(T,R) = Min\left[D(i(k), j(k))\right] = Min \frac{\sum_{1}^{k} d(i(k), j(k)).P(k)}{N(P)}$$

$$D(T,R) = \frac{1}{I+J} D_p(i(k), j(k)) \quad \text{où} \quad D_p(i(k), j(k)) = Min \sum_{1}^{k} d(i(k), j(k)).P(k)$$

cette équation peut être résolue par programmation dynamique(à l'aide du principe d'optimalité de Bellman). On désigne D_{pp} la distance cumulée :

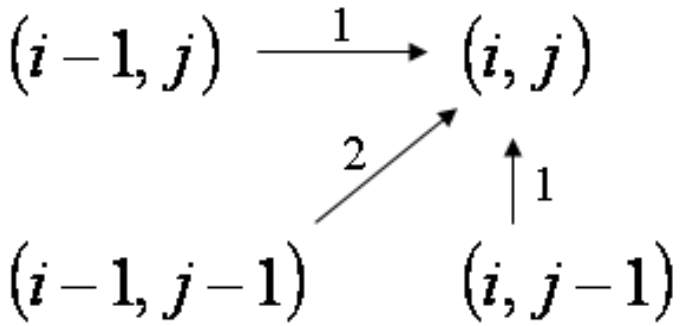
$$D_{pp}(i, j) = Min\left[D_{pp}(i', j') + D_p((i', j'), (i, j))\right]$$

DTW : Dynamic Time Warping

$$D_{pp}(i, j) = Min\left[D_{pp}(i', j') + D_p((i', j'), (i, j))\right]$$

(i', j') appartient au voisinage du (i, j)

$D_p((i', j'), (i, j))$ distance locale pondérée entre $(i', j')(i, j)$



Algorithme :

1) Initialisation :

$$D_{pp}(1,1) = d(1,1) \times 2$$

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

2) Programmation dynamique :

$$pour \quad i = 2....I \quad faire$$

$$pour \quad j = 1....J \quad faire$$

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

3) Détermination du taux de dissemblance :

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