
INFO0902-1 Structures des données et algorithmes

Projet 3 : Résolution de problèmes

Bachelier en sciences de l'ingénieur, Bloc 2

Samuel Degueldre et Corentin Van Putte

2016 - 2017

Première partie

Analyse Théorique

1 DTW

1.1 Expliquez brièvement le principe de cet algorithme

L'algorithme consiste à calculer un alignement de deux séries temporelles en tenant compte d'éventuelles différences de vitesse entre les échantillons. Une fois cet alignement réalisé, on peut extraire une distance entre les deux échantillons.¹

1.2 Discutez de l'intérêt des contraintes de localité

Le principe de la contrainte de localité est d'éviter que l'algorithme n'associe deux points d'une série s'ils sont trop éloignés, ce qui donnerait une distance DTW effectivement plus faible mais correspondrait à un alignement de séquences qui n'a pas de sens dans la réalité. Cela permet également d'accélérer l'algorithme puisqu'il ne doit alors plus calculer de score d'association entre des points trop éloignés temporellement.¹

1. Sources : https://en.wikipedia.org/wiki/Dynamic_time_warping

**1.3 L'algorithme DTW est basé sur la programmation dynamique
Donnez la formulation récursive correspondante**

$$DTWcosts[i, j] = \begin{cases} 0, & \text{si } i = 0 \text{ et } j = 0 \\ +\infty, & \text{si } i = 0 \text{ ou } j = 0 \\ basecosts + \min(DTWcosts[i-1, j-1], DTWcosts[i, j-1], DTWcosts[i-1, j]), & \text{sinon} \end{cases}$$

1.4 Donnez la complexité de l'algorithme en fonction des longueurs des signaux comparés et de la contrainte de localité

Soit S1 la longueur du premier signal, S2 la longueur du second signal et L la contrainte de localité.

En général, c'est à dire lorsque $L > S2$, l'algorithme est $\Theta(S1 \times S2)$.

En particulier, c'est à dire lorsque $L \leq S2$, l'algorithme est $\Theta(S1 \times L)$.

On notera tout de même que la matrice du DTW doit être dans tous les cas initialisée. Cette matrice ayant une taille $S1 \times S2$, l'initialisation de la matrice est $\Theta(S1 \times S2)$.

2 Découpage optimal

2.1 Formulez M(n) de manière récursive en précisant bien le cas de base

$$M(n) = \begin{cases} +\infty, & \text{si } size \geq lMax \text{ ou } size < 2 \times lMin \\ & \text{et } size < lMin \\ & \text{et } \frac{size}{lMin} < \frac{size}{lMax} \\ predictDigit(n, database, locality), & \text{sinon} \end{cases}$$

où size est la taille du sous-signal.

2.2 Déduez-en le pseudo-code d'un algorithme efficace pour calculer cette découpe

$$BESTSPLIT(\text{signal}, \text{database}, \text{locality}, lMin, lMax)$$

1. **if** signal.length / IMin < signal.length / IMax
2. **return** +∞
3. Let splitScores[i...signal.length, j...signal.length] be a matrix
4. **for** i = 0 **to** signal.length
5. **for** j = 0 **to** signal.length
6. splitScore[i,j].digit = -1
7. returnSequence = BESTSPLITWRAPPER(signal, database, locality,
8. IMin, IMax, splitScores, 0,
9. signal.size - 1)
10. **return** returnSequence

BESTSPLITWRAPPER(signal, database, locality, lMin, lMax,
splitScores, start, stop)

- [illegible]

```

18.                                     splitScores, i, stop)
19.                                     newSplit = UNITE(leftDigit, bestRightSplit, start)
20.                                     if newSplit.score < bSplit.score
21.                                         bSplit = newSplit
22. if size < lMax
23.     if splitScores[start,stop].digit == -1
24.         seqScore = SPLITSCORE(signal, database, locality, start,
25.                                 stop, splitScores)
26.     else
27.         seqScore = splitScore[start,stop]
28.     if bSplit.score > seqScore.score
29.         bSplit.nDigits = 1
30.         bSplit.score = seqScore.score
31.         bSplit.digits[0] = seqScore.digit
32.         bSplit.splits[0] = start
33. return bSplit

```

SPLITSCORE(signal, database, locality, start, stop, splitScores)

```

1. if splitScores[start,stop].digit == -1
2.     Let subsignal be a signal of size [start,stop]
3.     splitScores[start,stop] = PREDICTDIGIT(subsignal, database,
4.                                             locality)
5. return splitScores[start,stop]

```

UNITE(dscore, seq, start)

```

1. retSeq.nDigits = seq.nDigits + 1
2. retSeq.score = dscore.score + seq.score
3. retSeq.digits[0] = dscore.digit
4. retSeq.splits[0] = start
5. for i = 1 to retSeq.nDigits
6.     retSeq.digits[i] = seq.digits[i-1]
7.     retSeq.splits[i] = seq.splits[i-1]
8. return retSeq

```

2.3 Analysez la complexité au pire et au meilleur cas de votre algorithme en fonction des paramètres les plus appropriés

Deuxième partie

Analyse Empirique

1 Vérification de la pertinence du DTW pour l'identification de chiffres isolés

Contrainte de localité	Nombre d'erreurs
0	50
1	40
2	37
3	29
4	21
5	15
6	12
7	11
8	10
9	9
10	3
11	1
12	1
13	1
14 ou plus	0

Tableau 1 – Nombre d'erreurs de reconnaissance en fonction de la contrainte de localité²

2 Vérification du bon fonctionnement de l'algorithme de découpage

2. Cette analyse est basée sur 50 échantillons de tests, 5 pour chaque nombre de 0 à 9