Structures de données et algorithmes
Projet 3: Résolution de problèmes

Pierre Geurts – Jean-Michel Begon – Romain Mormont

26 avril 2019

On souhaite réaliser une application de recherche et de reconnaissance de croquis ("sketch" en anglais), c'est-à-dire des croquis réalisés rapidement à main levée. Pour ce faire, on a identifié dix classes d'intérêt et on dispose d'une base de données de référence [1] contenant 1000 croquis par classe (voir la figure 1 pour un exemple de croquis pour chacune des 10 classes).

A partir d'un croquis source (la requête), l'objectif est d'identifier le plus rapidement possible dans la base de données les k croquis "les plus proches" de cette requête selon une certaine mesure de distance. Formellement, pour un croquis source Q, on cherche à déterminer l'ensemble $\{S_1^*(Q), \ldots, S_k^*(Q)\}$ où $S_j^*(Q)$ tel que

$$S_j^*(Q) = \min_{S \in LS \setminus \cup_{i=1}^{j-1}\{S_i^*\}} dtw(Q, S) \qquad 1 \leq j \leq k \tag{1}$$

où LS est la base de données de référence et dtw mesure la distance entre deux croquis.

Les croquis ainsi retrouvés peuvent être ensuite utilisés pour faire une prédiction de la classe du croquis source, par exemple, en lui attribuant la classe majoritaire (*i.e.* la plus fréquente) parmi les k croquis retrouvés. Le processus est illustré sur un exemple à la figure 2.

1 Description de l'approche

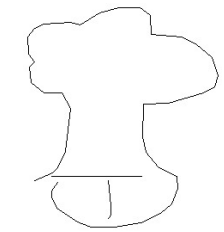
1.1 Représentation des croquis

Un croquis est représenté par un ensemble de *strokes*. Un *stroke* est une polyligne représentant un coup de crayon. Une polyligne est représentée par une succession de points formant des segments de droite. L'ordre des *strokes* et des points à l'intérieur d'un *strokes* suit l'ordre chronologique de tracé du croquis.

1.2 Dynamic time warping

Puisque le nombre de *strokes* et de segments par *stroke* varient d'un croquis à l'autre, on se propose de procéder en deux temps pour calculer la similarité entre deux croquis. D'abord, on concatène les *strokes* d'un croquis en une seule polyligne. On obtient alors une série temporelle décrivant l'évolution de la position du crayon. Ensuite, puisque ces séries ont des longueurs

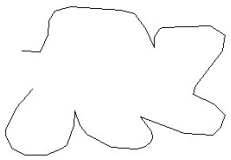
1. Ces croquis sont extraits de la base de données "Quick, Draw !" collectée par Google et disponible à l'adresse suivante : `https://github.com/googlecreativelab/quickdraw-dataset`.



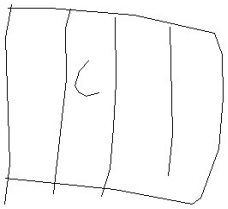
(a) Arbre (classe 0)

(b) Zigzag (classe 1)

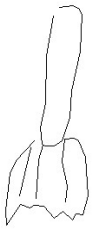
(c) Cellule (classe 2)



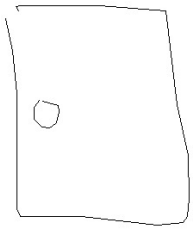
(d) Nuage (classe 3)



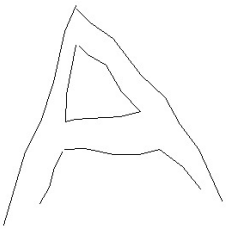
(e) Cellule (classe 4)



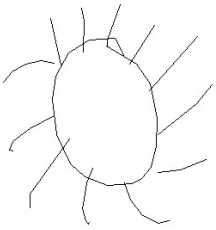
(f) Balais (classe 5)



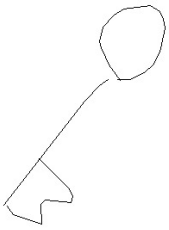
(g) Porte (classe 6)



(h) Tour Eiffel (classe 7)

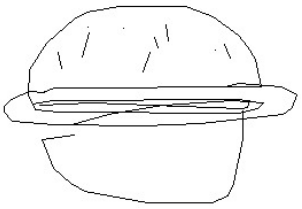


(i) Soleil (classe 8)



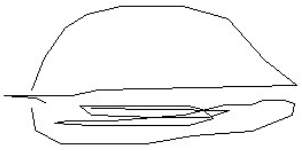
(j) Clé (classe 9)

FIGURE 1 – Exemples de croquis

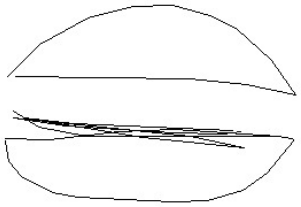


(a) Requête

(b) Plus proche voisin



(c) Deuxième plus proche voisin



(d) Troisième plus proche voisin

FIGURE 2 – Requête et plus proches voisins

différentes, on propose de calculer la distance entre elles à l'aide d'un algorithme appelé *dynamic time warping (DTW)*. Cet algorithme générique calcule la distance entre deux séries temporelles en prenant en compte les variations en termes de vitesses et de longueurs entre les deux séries. Il semble donc tout à fait approprié pour prendre en compte les variabilités en termes de déplacements du crayon entre personnes.

L'idée de l'algorithme DTW est de rechercher l'appariement des points des deux séries temporelles (respectant l'ordre temporelles de ces points) qui minimisent la somme des distances entre points appariés. Dans le cadre de ce projet, nous utiliserons comme mesure de distance entre deux points la distance absolue moyenne :

$$d(P_1, P_2) = \frac{1}{2}\Big(|x_2 - x_1| + |y_2 - y_1|\Big) \tag{2}$$

1.3 Recherche des k croquis les plus proches

La détermination des k croquis les plus proches d'un croquis source Q dans la base de données de référence LS sera effectuée par une fonction NearestNeighbours(LS, Q, k). Cette fonction devra calculer les distances selon l'algorithme DTW entre le croquis Q et successivement tous les croquis de l'ensemble LS. Afin de maintenir efficacement les k croquis les plus proches lors du parcours de la base de données de référence, nous vous demandons d'utiliser une file à priorité dont la capacité sera limitée à k éléments maximum et utilisant la distance à l'exemple Q comme valeur de priorité. Lors du parcours de la base de données de référence, un croquis sera ajouté à la file, soit si cette file contient moins que k croquis, soit si ce croquis est plus proche de Q que le croquis de la file le plus éloigné de Q. Dans ce dernier cas, le nouveau croquis devra remplacer dans la file le croquis le plus éloigné de Q.

Lorsque l'ensemble Q est grand, la recherche des k plus proches croquis d'une requête peut être relativement lente. Dans le cas où la file contient déjà k croquis, il est possible d'éviter d'avoir à effectuer tout le calcul de la distance DTW pour un nouveau croquis en prenant en compte la distance maximale courante, notée d_{max}, entre un croquis de la file et la requête. L'algorithme DTW calcule en effet une matrice de coût colonne par colonne ou ligne par ligne. Dès que le minimum d'une colonne ou d'une ligne [2] est supérieur à la distance d_{max}, on a la garantie que la distance DTW ne pourra pas être inférieure à d_{max} et il est alors possible d'arrêter prématurément le calcul de la distance.

2 Implémentation

Les fichiers sources suivants vous sont fournis :

- `Sketch.h/c` : une librairie définissant les structures utilisées pour représenter les croquis et les bases de données et implémentant des fonctions permettant de charger en mémoire un ensemble de croquis à partir d'un fichier, d'afficher leur classe et de générer une image (au format ppm) du croquis. Cette dernière fonction utilise une librairie graphique (fichiers `easyppm.h/c`) également fournie.
- `main.c` : un fichier main d'exemple utilisant les fonctions à implémenter pour retrouver les k plus proches croquis d'une requête.

2. auquel on pourra ajouter la distance entre les deux derniers points des deux séries temporelles, sauf pour la dernière ligne/colonne.

Les différents fichiers à fournir sont les suivants :

— `DynamicTimeWarping.c` contenant l'implémentation du *dynamic time warping* en répondant à l'interface définie dans `DynamicTimeWarping.h`.
— `BoundedPriorityQueue.c` regroupant les différentes fonctions liées à la file à priorité dont l'interface est précisée dans le fichier `BoundedPriorityQueue.h`.
— `NearestNeighbours.c` implémentant la fonction de recherche des k croquis les plus proches selon l'interface définie dans `NearestNeighbours.h`.

En outre, nous vous fournissons les bases de données suivantes :

— `testset.txt` la base de données de test.
— `trainingsetverylarge.txt` la base de données de référence contenant 1000 exemples par classe.
— `trainingset.txt` un sous-ensemble de la base de données de référence contenant 100 exemples par classe.

3 Analyse théorique et empirique

Répondez aux questions suivantes dans votre rapport :

1. Renseignez vous sur l'algorithme DTW (citez vos sources !) et expliquez brièvement son principe.
2. L'algorithme DTW est basé sur la programmation dynamique. Donnez la formulation récursive correspondante.
3. Donnez la complexité en temps et en espace de l'algorithme en fonction des longueurs des signaux comparés.
4. Donnez le pseudo-code de la fonction NearestNeighbours(LS, Q, k), qui retourne les k plus proche voisin de l'exemple Q dans la base de données de référence LS. Cette fonction devra utiliser la fonction Replace de la structure de file à priorité (voir le fichier `BoundedPriorityQueue.h`). Donnez également le pseudo-code de cette fonction.
5. Donnez et justifiez la complexité de la fonction NearestNeighbours au pire et au meilleur cas en fonction de $p = |Q|$, le nombre de points du croquis Q, de $n = |LS|$, le nombre d'échantillons de référence, de l, la longueur des croquis de LS (qu'on supposera constante) et de k, le nombre de voisins.
6. Pour $k \in \{1, 3, 7\}$, calculez le taux d'erreur de reconnaissance des échantillons de tests en utilisant les deux bases de données de référence fournies. Le taux d'erreur est la fraction du nombre de fois où la vraie classe d'un croquis de l'ensemble de test ne correspond pas à la classe majoritaire parmis les k croquis les plus proches de l'ensemble de référence.
7. Reportez dans le rapport les temps nécessaires au calcul de ces taux d'erreur en fonction de k, sans la technique d'accélération.
8. Pour la grande base de données de référence et pour la meilleure valeur de k trouvée à la question 6, calculez et reportez dans le rapport la matrice de confusion des prédictions, c'est-à-dire la matrice M dont l'élément $M_{i,j}$ est le nombre de croquis de l'ensemble de test de classe i dont la classe prédite est la classe j. Sur base de cette matrice, déterminez quelles sont les erreurs de prédiction les plus souvent commises par l'approche.
9. Expliquez et justifiez la correction de la technique d'accélération mentionnée dans la section 1.3. Implémentez la et mesurez son impact sur les temps de calcul.

4 Deadline et soumission

Le projet est à réaliser **par groupe de deux** pour le **16 mai 2019, 23h59** au plus tard. Il doit être soumis via la plateforme de soumission (`http://submit.montefiore.ulg.ac.be/`).

Le projet doit être rendu sous la forme d'une archive `tar.gz` contenant :

1. Votre rapport (7 pages maximum) au format PDF. Soyez bref mais précis et respectez bien la numération des (sous-)questions.
2. Les fichiers `DynamicTimeWarping.c`, `NearestNeighbours.c` et `BoundedPriorityQueue.c`.

Vos fichiers seront évalués avec les flags habituels (`--std=c99 --pedantic -Wall -Wextra -Wmissing-prototypes - DNDEBUG`) sur les machines ms8xx. Ceci implique que :

— Les noms des fichiers doivent être respectés.
— Le projet doit être réalisé dans le standard C99.
— La présence de *warnings* impactera négativement la cote finale.
— Un projet qui ne compile pas sur ces machines recevra une cote nulle (pour la partie code du projet).

Un projet non rendu à temps recevra une cote globale nulle. En cas de plagiat avéré, l'étudiant se verra affecter une cote nulle à l'ensemble des projets.

Les critères de correction sont précisés sur la page web des projets.

Bon travail !