K Plus Proches Voisins - Partie 1

Contacts
paul.viallard@univ-st-etienne.fr emilie.morvant@univ-st-etienne.fr
Salon discord "Projet" de l'UE Programmation Impérative 2

1 - Avant-Propos

Le projet doit être réalisé en **binôme**, les binômes doivent nous êtes communiqués sur discord dans le salon "projet" de l'UE programmation impérative 2 avant le 24 mars.

Vous devez implémenter dans le langage C l'application décrite ci-dessous, puis le déposer sur claroline **avant le lundi 10 mai à 23h59** dans le rendu dédié (les soutenances auront lieu le mardi 11 mai, les modalités seront transmises ultérieurement) [1]. Le dépôt doit prendre la forme d'une archive `tar.gz` et doit UNIQUEMENT contenir :

- l'ensemble du code source (fichiers sources, Makefile)
- une notice README expliquant comment créer l'exécutable, principal, l'exécutable de la génération de fichiers des tests et comment générer les fichiers de tests.

2 - Objectif du projet

L'objectif du projet est d'implémenter une application permettant de visualiser en 2D un algorithme simple de prise de décision automatique : les K Plus Proches Voisins (K-PPV) ou K Nearest Neighbors (KNN) [2].

3 - La Méthode des K Plus Proches Voisins

En intelligence artificielle, plus précisément en apprentissage automatique (*machine learning* en anglais), la méthode des K Plus Proches Voisins (K-PVV) est une méthode dîte de classification. Elle consiste à attribuer automatiquement une classe (ou une étiquette ou une catégorie) à un objet (ou un individu) à classer, en se basant sur des données déjà observées (voir la Figure 1 ci-contre).

Pour une tâche de classification particulière (par exemple une tâche de reconnaissance de chiffres manuscrits), les données vont être représentées par un vecteur dans un espace potentiellement de grande dimension. Nous supposons que nous disposons d'un ensemble de données observées *en amont* dont nous connaissons déjà la classe : l'ensemble des données observées *en amont* forme ce qui est appelé l'ensemble d'apprentissage. Étant donné cet ensemble d'apprentissage, lorsque qu'une donnée D (que l'on souhaite classer) est observée, la méthode des K-PPV cherche parmis les données de l'ensemble d'apprentissage les K données les plus proches du point D et renvoie la classe majoritaire parmi ces K voisins : la méthode renvoie la classe qui apparaît le plus de fois dans les voisins (c'est ce que l'on appelle la règle de décision des K-PPV).

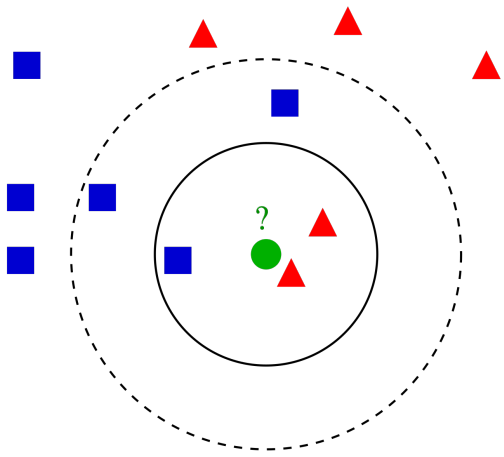

FIGURE 1 – Illustration de la méthode des K-PPV. Pour cela on dispose d'un jeu de données constitué d'objets (représentés dans un espace 2D) dont on connait déjà la classe : carré bleu ou triangle rouge. L'objectif ici est de trouver automatiquement la classe du point vert. La décision prise dépend des plus proches voisins (PPV) de ce point : Le cercle noir montre ses 3 PPV, dans ce cas, le point vert sera classé comme un triangle rouge, en pointillé ses 5 PPV, dans ce cas le point vert sera classé comme un carré bleu. (image par Antti Ajanki AnAj — Travail personnel, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2170282)

Dans ce projet,

- la mesure de distance utilisée pour trouver les plus proches voisins est la distance euclidienne naturelle et classique (on peut évidemment utiliser d'autres mesures de distance, mais cela est une autre histoire) ;
- la règle de décision utilisée correspond au vote de majorité (il existe d'autres méthodes de prise de décision pour les K-PPV.) ;
- l'espace de représentation est un espace à 2 dimensions bornées : les coordonnées sont comprises entre -1 et $+1$;
- dans un premier temps, nous vous conseillons de ne travailler qu'avec 2 classes, puis d'étendre le travail à plus de 2 classes.

1. Nous nous réservons le droit de modification des dates en cas de restrictions sanitaires plus fortes (ex. un confinement).
2. Page wikipédia sur la méthode des K-PPV : `https://fr.wikipedia.org/wiki/Methode_des_k_plus_proches_voisins`.

4 - Avancement

Le développement de votre programme doit se faire en 6 étapes. Seules les 3 premières étapes sont décrîtes dans ce document. Les 3 autres étapes seront présentées lors de la séance de TP prévue le 7 avril. Nous vous conseillons donc fortement d'avoir bien avancé dans ces 3 étapes d'ici le 7 avril.

4.1 - Préliminaires

Votre application doit permettre le chargement et l'affichage de différents jeux de données. Ces derniers sont stockés dans des fichiers (certains fichiers exemples seront fournis sur claroline). Le format des fichiers est le suivant. La première ligne contient 3 entiers positifs séparés par un espace : le nombre de points n contenus dans le fichier, la dimension d des données (le projet est à réaliser avec des données en 2 dimensions) et le nombre de classe c. Le n lignes restantes du fichier sont de la forme `classe coord_1 coord_2 ... coord_d`, où `classe` est un entier compris entre 1 et c et les coordonnées sont des réels compris entre -1 et 1.

Exemple d'un fichier simple (en 2D, avec 2 classes) :

```
4 2 2
1 0.5 0.67
1 0.45 0.1111
2 -0.4 -1
2 -0.75 -0.3212
```

Pour tester votre programme tout au long du développement de votre application, vous devez dans un premier temps créer un autre programme (indépendant) de génération de fichiers de jeux de données prenant en argument le nom du fichier, le nombre de points à générer, le nombre de classes et le nombre de dimension (par défaut 2).

4.2 - Étape 1 - Mode Création/Chargement des données (tableau)

La première étape de votre projet vise à implémenter le mode **création/chargement des données** via une interface graphique avec MLV (les éléments qui doivent OBLIGATOIREMENT apparaître dans l'interface graphique sont résumés dans la Figure 2).

1. Vous aurez besoin d'une structure de données `point` pour modéliser les données. Cette structure contiendra trois champs : deux réels `x` et `y` (compris entre $[-1, 1]$) et un entier `classe`.
2. L'ensemble des points sera stockées dans un premier temps dans un tableau de structures `point` (le nombre de points dans le tableau est amené à varier au cours de l'exécution du programme).
3. Vous devez implémenter une fonction de chargement des données à partir d'un fichier de données (suivant le format décrit dans les Préliminaires). Cette fonction doit stocker les points dans le tableau de points, vous nommerez la fonction : `chargement_fichier`. *Cette fonction sera ammenée à évoluer lors de la partie 2 du projet.*
4. Vous devez implémenter l'interface graphique qui permettra d'afficher les points dans un espace en 2 dimensions sachant que $(x, y) \in [-1, +1] \times [-1, +1]$. Dans un premier temps, nous vous conseillons de ne travailler qu'avec 2 classes possibles : 1 et 2. Les points de classe 1 seront représentés par un $-$ bleu, ceux de classe 2 par un $+$ rouge. Lorsque vous considèrerez plus de classes, vous êtes libres de les représenter comme vous le souhaitez.
5. Les données peuvent être "chargées" via deux méthodes différentes :
   - via un fichier : l'interface doit offrir la possibilité de charger un fichier de données,
   - ou manuellement "en cliquant" : l'interface doit également pouvoir vous permettre d'entrer manuellement des points en cliquant dans l'espace (pensez à une solution pour que l'utilisateur précise la classe du point ajouté).
6. Après avoir entré mannuellement des points, l'interface doit offrir la possibilité de sauvegarder les points dans un fichier (de jeu de données) (vous nommerez, la fonction de sauvegarde : `sauvegarde_fichier`) .
7. Un bouton "suppression/annulation" doit permettre d'annuler la dernière saisie.

Pour les étapes suivantes, l'interface graphique doit également :

- contenir une zone de texte permettant de saisir la valeur de K (c'est un entier positif qui ne peut pas dépasser le nombre de points du jeu de données),
- une zone permettant de sélectionner des options d'affichage (voir ci-dessous),
- une zone permettant de selectionner soit le mode création (qui correspond à l'étape 1), soit le mode K-PPV (qui correspond aux étapes 2 et 3).

4.3 - Étape 2 - Mode K-PPV - sans prise de décision

Cette étape vise à implémenter le mode K-PPV sans prise de décision où les Plus Proches Voisins sont cherchés dans le tableau décrit ci-dessus : vous devez donc trouver dans le tableau les K plus proches voisins (au sens de la distance euclidienne) d'un point donné et les mettre en avant dans la zone d'affichage. Plus précisément, si ce mode est actif, lorsque l'utilisateur clique dans l'espace, votre programme doit

1. afficher le point cliqué (x, y) (en noir),
2. trouver les K Plus Proches Voisins du point (x, y),
3. mettre en avant les K-PPV trouvés (en les affichant dans une autre couleur par exemple).

Vous devrez également ajouter l'option de "dessin du voisinage" du point (x, y) : lorsque cette option est activée l'interface doit afficher le cercle de centre (x, y) passant par son plus proche voisin le plus éloigné ($i.e.$, le K-ième).
Pour passer à un autre point, l'utilisateur doit cliquer sur le bouton suppression/annulation (qui doit effacer (x, y), ainsi que la mise en avant de ses K-PPV).
Tant que le point (x, y) n'a pas été supprimé l'utilisateur ne doit pas pouvoir ajouter de nouveaux points. Il peut néanmoins modifier les options d'affichage, ainsi que la valeur de K (avec rafraichissement de l'affichage dès lors qu'une modification a été détectée).

4.4 - Étape 3 - Mode K-PPV - avec prise de décision

Cette étape vise à implémenter le mode K-PPV avec prise de décision : vous devez donc implémenter l'option de l'interface "avec prise de décision". Lorsque cette option est activée, le point (x, y) sera représenté par un rond plein dont la couleur correspond à la classe calculée par la règle de décision des K-PPV, c'est-à-dire la classe majoritaire [3] parmi les K-PPV de (x, y). *Soyez curieux, pensez à regarder ce qu'il se passe avec différentes valeurs de K.*

5 - Remarques importantes

1. En fonction du nombre de points qui compose votre jeu de données, le calcul des K-PPV peut être long, cela est normal. Les étapes 4 à 6 auront pour objectif d'accélérer le calcul.
2. La Figure 2 résume les éléments qui doivent obligatoirement apparaître dans votre interface. L'organisation et le design de l'interface sont libres, vous pouvez également rajouter des fonctionnalités qui vous semblent utiles.
3. N'oubliez pas d'organiser votre code en modules
4. N'oubliez pas de commenter votre code et de faire un rendu "propre" ! !
5. Votre programme devra compiler sans erreur ni warning sur les machines de l'université via la commande `make`.

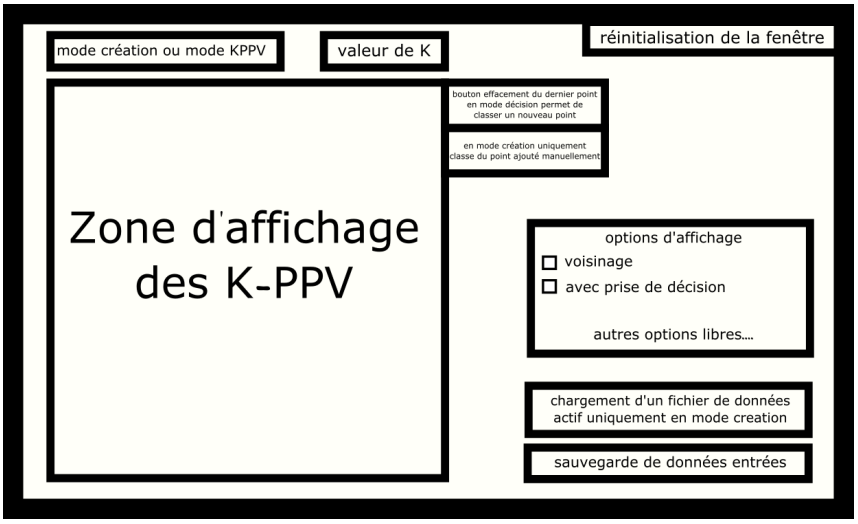

FIGURE 2 – Les éléments qui doivent **obligatoirement** apparaitre dans votre interface. Attention votre interface doit permettre de sélectionner deux modes d'affichage : le mode création/chargement des données et le mode KPVV.

3. À vous de décider de la méthode de décision à prendre en cas d'égalité dans le vote de majorité.