

Rapport KNN Final

Afin de développer l'algorithme KNN pour le Classification Challenge nous nous sommes beaucoup inspirés de nos précédents travaux sur le KKN.

Tout d'abord nous avons créé 2 versions de l'algorithme KNN, basées toutes deux sur le même système de prédiction.

La première version utilise un premier fichier .csv comme base d'informations et prend un élément aléatoire dans un second csv contenant des points et leurs labels. Après avoir procédé à la détermination du label de l'élément sélectionné, la prédiction est comparée avec le label véritable. En répétant cette opération 1000 fois (afin de pouvoir confirmer nos résultats avec la loi des grands nombres), nous pouvons construire une matrice de confusion fiable nous permettant d'ajuster notre code et d'en voir le résultat.

La seconde version du KNN s'appuie sur un premier fichier csv contenant des points avec leurs labels et un second contenant des points dont nous voulons prédire le label. L'algorithme de détermination des labels est similaire au précédent, à la différence que cette fois ci les labels prédits sont entrés dans un fichier .txt de réponse. Afin de tester nos résultats, nous avons parfois indiqué comme second csv un fichier dont les labels nous étaient donnés. Dans ce cas les résultats de notre fichier de prédictions sont comparés avec les véritables labels et on obtient notre pourcentage de réussite (qui tourne généralement entre 85% et 90%).

Nous allons maintenant détailler les caractéristiques de notre système de prédiction.

Afin de déterminer le label d'un élément inconnu, notre algorithme détermine d'abord les 8 voisins connus les plus proches en basant sa recherche sur une distance euclidienne pondérée. Les pondérations servaient à pouvoir donner plus de poids à une dimension si celle-ci se révélait significativement plus importante que les autres. Nous avons finalement décidé de fixer toutes les pondérations à 1 car les résultats semblaient satisfaisants.

Une fois les k voisins les plus proches déterminés, nous passons à la prédiction de l'élément à tester qu'on appellera X. Nous avons voulu mettre en avant la proximité des k voisins par rapport X. Si l'on considère X et ses k voisins, on peut dire que X est au centre d'un cercle contenant tous les k voisins, le cercle ayant de rayon égal à la distance du k voisin le plus X. Nous utilisons cette distance comme référence de proximité plutôt que la distance maximum parmi tous les points afin d'éviter que les différences entre les k voisins soient gommées. La différence entre cette référence et la distance de chaque k voisin est ensuite ajoutée à la valeur de probabilité générale du groupe. Ainsi plus un voisin est proche du centre plus il va

ajouter à la valeur de probabilité de son groupe, le groupe ayant la plus grande valeur de probabilité étant désigné comme celui auquel appartient l'élément X.

Nous avons à l'origine créé une fonction de normalisation des valeurs des axes mais celle-ci ne s'est pas révélée utile. En effet elle allongeait considérablement l'exécution de notre programme et n'améliorait pas la qualité des prédictions (voire la dégradait). Nous avons du mal à comprendre pourquoi cette normalisation n'apporte rien à notre prédiction, peut être est-elle mal implémentée ou alors fait doublon.

Enfin afin d'augmenter notre précision lors de la détermination des éléments de `finalTest.csv`, nous avons créé un nouveau fichier d'apprentissage `Ensemble_base.csv` composé des éléments de `data.csv` et ceux de `preTest.csv`. On augmente ainsi la taille de notre ensemble de valeurs d'apprentissage.