TD : K-NN en version matricielle

Il s'agit de se familiariser avec une implémentation "matricielle" de l'algo de K-NN. On suppose avoir T documents d'apprentissage, chaque document est représenté par un vecteur, de taille V = la taille du vocabulaire (cf. TDs KNN précédent).

Pour représenter un document, au lieu d'une représentation de type dictionnaire ne stockant que les composantes non nulles des vecteurs, on utilise cette fois-ci un vrai vecteur, de taille V (le vecteur est donc très **creux**).

L'ensemble d'apprentissage peut alors être représenté comme une matrice où chaque ligne (ou colonne, c'est au choix) contient la représentation vectorielle d'un document.

Il faut cependant connaître la sémantique de chaque position dans le vecteur représentant le document: ici chaque position est associée à un mot du vocabulaire. I.e. par exemple la première composante d'un vecteur correspondra au mot "le", la deuxième composante au mot "maison" etc…

Il faut donc disposer d'une **correspondance** entre mots du vocabulaire et indice de mot.

On gère également de manière vectorielle les classes (prédites ou gold), et on doit pour cela disposer d'une correspondance entre le vocabulaire des classes et indice de classe.

1 Correspondance vocabulaire / indices

Une possibilité est pour chaque correspondance d'avoir 2 structures:
- un dico : du mot vers un numéro de mot (un indice de mot)
- une liste : au rang k de la liste, on stocke le mot d'indice k

On peut ainsi passer efficacement des mots aux indices et vice-versa.

➔ **Construisez** ces correspondances (mots ⇔ indices de mots, et classes ⇔ indices de classes) à la lecture des données
(modifiez par exemple le read_examples en ce sens, qui retournera par exemple les correspondances)

2 Version matricielle des données d'apprentissage

Il s'agit de transformer les données d'apprentissage (les T couples (vecteur, classe gold) en un seul couple:
- Une matrice X_train de taille (T, V) où chaque ligne est le vecteur d'un des exemples
- Un vecteur ou liste Y_train pour les T classes gold correspondantes

On fait de même pour les données de test, pour obtenir X_test et Y_train.

Par exemple, vous pourrez faire une fonction qui prend en entrée
- une liste d'instances de Example
- les index construits précédemment
- et retourne la matrice X et le vecteur Y correspondants
- indication: construire une matrice / vecteur de la taille voulue, rempli(e) de zéros, puis instancier les valeurs non nulles

3 Ré-implémentation de l'algo du KNN via operations entre matrices / vecteurs

A l'aide du mémo numpy, détaillez comment réécrire la classification des exemples de test via l'algo KNN.

Vous supposez ici disposer de X_train , Y_train, X_test, Y_test tels que définis supra.

Ecrivez et testez en mode interactif les opérations numpy pour calculer efficacement:

- la norme des vecteurs contenus dans X_train et X_test
- calculer les matrices contenant les vecteurs normalisés de X_train et X_test
- le cosinus d'un vecteur contenu ds X_test avec tous les vecteurs dans X_train
- **plus efficace**: directement le cosinus de tous les vecteurs des exemples de test avec tous les vecteurs des exemples d'apprentissage

NB: pour éviter les bugs, il vous faut tester les opérations matricielles sur des exemples construits manuellement, en mode interactif.

4 Comparaison des temps d'exécution

Comparez les temps d'exécution de la version "dictionnaire" et la version matricielle, avec
- l'ensemble apprentissage / test medium (2000 docs / 200 docs)
- l'ensemble complet reuters (8806 docs / 2346 docs)

Vous pouvez utiliser un profiler en appelant
python -m cProfile votreprogramme.py … > log