Etude Pratique (1) – Master Informatique – IAA – 2023
Université d'Aix-Marseille
The curse of dimensionality (*aka* fléau de la dimension)

Cécile Capponi – Qarma, LIS – AMU
Cecile.Capponi@lis-lab.fr

7 mars 2023

Objectifs de la séance :

1. retour sur la classification supervisée, ici les k-ppv et les arbres de décision, et pratiques expérimentales d'estimation des performances ;
2. observer en pratique les effets conjoints sur les performances de la classification, du nombre de données vectorielles, de la dimension de leur espace de description, de leur expansion (dilatation ou compactage) dans cet espace de description, de la complexité du modèle induit, et du bruit dans les données ;
3. comprendre l'intérêt des jeux de données artificiellement générés pour comprendre (par l'expérimentation contrôlée) le comportement d'un algorithme et le rôle de ses hyper-paramètres.

1 Étude des k-ppv sur des données en damier

1.1 Quand la dimension augmente avec un nombre constant d'exemples

On cherche ici à observer l'impact de la dimension des données sur le calcul et la significativité des distances entre elles.
La commande `np.random.rand(N,d)` renvoie un tableau de dimension `N` $\times$ `d` dont chaque ligne représente les coordonnées d'un point tiré uniformément au hasard dans le cube unité de dimension `d`. Le centre de ce cube unité est le point `0.5*np.ones(d)`. On définit la distance entre deux point comme le maximum de la valeur absolue des différences des coordonnées :

$$d(\mathbf{x}, \mathbf{y}) = \max_i |x_i - y_i|.$$

1. X désigne un échantillon de n points du cube unité de dimension `d`. Écrivez les fonctions `distance_au_centre(X)` et `voisin_le_plus_proche_du_centre(X)`, qui calculent respectivement la moyenne des distances des points de X au centre, et la distance minimale d'un point de X au centre.
2. Que cherche à étudier le programme suivant ? Exécutez le et **commentez** les résultats obtenus au regard de l'objet d'étude. Illustrer le phénomène pour $d = 1, 2, 3$ avec 5 exemples ($N = 5$ au lieu de 100).

```
for d in range(1,21):
    dist = []
    v = []
    for i in range(10):
        X = np.random.rand(100,d)
        dist.append(distance_au_centre(X))
```

```
            v.append( voisin_le_plus_proche_du_centre(X))
        print(np.mean(dist), np.mean(v))
```

1.2 Données en damier

La fonction `damier(dimension, grid_size, nb_examples, noise = 0)` génère `nb_examples` points du cube unité de dimension `dimension` et les étiquettes par 1 ou -1 selon qu'ils appartiennent à une case « blanche » ou « noire » d'un damier possédant `grid_size` cases par dimension, avec un bruit uniforme de classification (échange de labels) de paramètre `noise`.

```
def damier(dimension, grid_size, nb_examples, noise = 0):
    data = np.random.rand(nb_examples,dimension)
    labels = np.ones(nb_examples)
    for i in range(nb_examples):
        x = data[i,:];
        for j in range(dimension):
            if int(np.floor(x[j]*grid_size)) % 2 != 0:
                labels[i]=labels[i]*(-1)
        if np.random.rand()<noise:
            labels[i]=labels[i]*(-1)
    return data, labels
```

L'exercice consiste à mener une étude expérimentale du classifieur des k plus proches voisins, en faisant varier : la dimension, le nombre d'exemples, le nombre de cases du damier et le taux de bruit.

1. Générer un damier D_1 à deux dimensions, avec 4 cases par ligne, sans bruit, et 16 points. Affichez les données de ce damier D_1, en colorant en rouge les points de classe -1, et en bleu les autres. De même, générer et affichez le damier D_2 similaire au précédent, mais contenant 160 points.
2. Écrivez un programme estimant le score d'un classifieur des k-plus proches voisins sur D_1 (moyenne de 5 TTS), et celui d'un tel classifieur sur D_2. Expliquez les résultats, éventuellement en vous appuyant sur l'affichage des damiers.
3. Écrivez une fonction calculant les scores d'un classifieur des k-plus proches voisins sur un ensemble d'exemples, en réservant 70% de l'échantillon pour apprendre et 30% pour tester le classifieur appris (répétition de 5 TTS), avec en paramètres de la fonction hyper-paramètres suivants :
   — `k in {1, 5, 10, 50, 100, 500}`
   — `dim in {2, 5, 10, 50, 100}`
   — `nbcases in {2, 5, 10, 20}`
   — `noise in {0, 0.2, 0.3, 0.5}`
   — `nbex in {10, 100, 1000, 10000}`

   Tracez 5 courbes qui, à 4 hyper-paramètres fixés, indique en ordonnée le score du classifieur au fil des valeurs possibles du cinquième hyper-paramètre. Expliquez les tendances observées (par exemple : à 100 exemples en dimension 5, avec 0 bruit et 5 cases, comment évolue le score en fonction de k, et pourquoi ?)

   Pour les plus curieux : vous pouvez aussi faire varier deux hyper-paramètres à la fois (visualisation sur courbe 3D).
4. Écrivez une fonction (sans la grid search de sklearn) qui, sur un ensemble X de n exemples, (a) le répartit en un échantillon d'apprentissage X_train et un échantillon test X_test comprenant respectivement 70% et 30% des exemples, (b) sélectionne la meilleure valeur k_opt de k par validation croisée sur l'échantillon d'apprentissage, (c) ré-entraîne un classifieur des k_opt plus proches voisins sur X_train, et (d) affiche son score sur X_test. Exécutez ce programme pour une dizaine d'instances d'hyper-paramètres que vous choisirez pertinemment en fonction des résultats de la question précédente. Produire un tableau récapitulatif des expérimentations menées.

5. Pour `dimension = 2`, `nb_examples = 1000`, faites varier `grid_size` de 2 à 10 et calculez la meilleure valeur de k sélectionnée par validation croisée (sur l'échantillon d'apprentissage) et le score correspondant (sur l'échantillon test). Chaque résultat doit être la moyenne de 10 expériences indépendantes. Expliquez les résultats obtenus. Refaites ces expériences avec un taux de bruit de 0.2. La tendance observée est-elle la même ? Vous pouvez utiliser le programme `plot_classification.py` disponible à l'url `http://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html` pour dessiner les frontières de décision.
6. On peut démontrerqu'asymptotiquement, si l'on fait tendre k vers $+\infty$ de telle manière que k/n tende vers 0, où n est le nombre d'exemples, l'erreur du classifieur des k plus proches voisins tend vers l'erreur de Bayes. Écrivez un programme permettant d'étudier cette stratégie dans un cas difficile : par exemple, `dimension = 8` et `grid_size=2`, `noise=0` pour un nombre d'exemples variant de 1000 à 50.000 par pas de 1000 et $k = \log n$. Comparez avec $k = 1$. Commentaire ? Observe t-on un comportement analogue si l'on rajoute un bruit uniforme de 0.2 ?

1.3 Impact de la distance

Observe-t-on les mêmes phénomènes avec la distance euclidienne ?

2 A rendre avant le vendredi 10/03/2023 – 20h00

Un rapport PDF de 2 à 4 pages sur les expérimentations menées, qui indiquera une synthèse des résultats expérimentaux obtenus (sous forme de tableaux et/ou de courbes), et une conclusion générale sur ce que vous avez observé expérimentalement sous la perspective de ce qui a été vu en cours (lien entre dimension, nombre d'exemples, complexité de l'espace d'hypothèses, etc.

3 Pour aller plus loin et/ou pour mieux comprendre

— Les magnifiques cours de Stéphane Mallat `https://www.college-de-france.fr/site/stephane-mallat/course-2017-2018.htm` sur ce vaste sujet
— En pratique `https://www.mygreatlearning.com/blog/understanding-curse-of-dimensionality/`
— Liens avec autres sciences (systèmes dynamiques, théorie de l'information, etc.) `https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_133?noAccess=true`