TME 2 - Estimation de densité

L'objectif de ce TME est de prendre en main les algorithmes d'estimation de densité : estimation par histogramme et par méthode à noyaux. Nous utiliserons deux classes (dont les prototypes sont donnés dans le code fourni) - `Histogramme` et `KernelDensity` - qui héritent de la classe (abstraite) `Density`. La fonction `fit(x)` permet d'apprendre l'estimateur sur les données passées en argument et la fonction `predict(data)` permet de prédire pour chaque point de `data` la densité.

Complétez la classe `Density` avec une méthode `score(data)` qui permet de renvoyer la log-vraisemblance des données `data` de l'estimateur. Pour gérer les points correspondant à une densité nulle, il est usuel d'ajouter une très petite valeur (10^{-10} par exemple) à chaque vraisemblance avant de passer au log (pourquoi ?).

Données : API Google Places, Points d'intérêt de Paris

Téléchargez l'archive des données sur le site de l'ue. Elle contient un dictionnaire avec des informations sur différents types de points d'intérêt (POI) de Paris. Chaque clé du dictionnaire correspond à un type de POI (restaurant, bar, ...) et la valeur associée est un dictionnaire avec l'identifiant google map comme clé, et les informations sur le POI : les coordonnées GPS, sa notation, son nom, ses types, le niveau des prix.

Le code fourni vous permet de charger la matrice des coordonnées GPS et de visualiser les données.
Affichez sur la carte différents POIs, les restaurants et les bars par exemple.

Méthode par histogramme

Codez les méthodes de la classe Histogramme. Dans l'idéal, votre classe devra gérer des données de dimension quelconque. Pour les besoins du TME, nous n'utiliserons que des données 2D, vous pouvez donc vous contentez de méthodes qui fonctionnent en 2D. Le paramètre `steps` du constructeur doit permettre de spécifier le nombre d'intervalles de discrétisation de chaque dimension
Indications :

- Pour estimer l'histogramme, vous pouvez utiliser `np.histogramdd` ;
- Pour la prédiction, il n'y a pas de fonction `numpy` qui permet de connaître pour un point donné les indices correspondant dans l'histogramme. Vous pouvez passer par une fonction intermédiaire `to_bin(x)` qui calcule ces indices à partir du pas de discrétisation et des bornes min et max de l'histogramme.

Estimez la loi de densité géographique d'un type de POIs par la méthode des histogrammes pour un pas de discrétisation donné.
Vérifiez que vous avez bien une loi de densité : la fonction `get_density2D(f,data,steps)` permet d'évaluer un estimateur f d'un problème en 2 dimensions sur une grille dont les bornes sont définies par les minimums et maximums de `data` sur chaque dimension et où chaque dimension est divisée en `steps` intervalles. Elle renvoie l'estimation en chaque point de la grille, la discrétisation utilisée selon le premier axe et selon le second axe. Que doit vérifier votre estimateur sur cette grille s'il n'y a pas d'erreur ?
Pour quelques valeurs de pas de discrétisation de l'histogramme, affichez les densités obtenues à l'aide de la fonction `show_density(f,data,steps,log)`. Quelle discrétisation semble la meilleure ?

Utilisez un split en ensembles d'apprentissage et de test (ou une validation croisée) afin d'estimer la vraisemblance en apprentissage et en test. Quelle est le meilleur hyper-paramètre empirique pour le pas de discrétisation ?
Faire les mêmes expériences sur un autre type de POI plus rare ("night_club" par exemple). Le meilleur hyper-paramètre est-il du même ordre que précédemment ?

Méthodes à noyaux

On rappelle la définition de la méthode à noyaux :

$$f(\mathbf{x}_0) = \frac{1}{|\mathcal{X}|\sigma^d} \sum_{\mathbf{x} \in \mathcal{X}} \phi\left(\frac{\mathbf{x}_0 - \mathbf{x}}{\sigma}\right)$$

avec ϕ un noyau, σ un hyper-paramètre, $\mathcal{X}$ l'ensemble d'apprentissage et d la dimension de $\mathcal{X}$.

Implémentez le noyau uniforme : la fonction `kernel_uniform(x)` qui renvoie 1 si $|x^i| \leq \frac{1}{2}$ pour toutes les dimensions i, 0 sinon. Votre implémentation doit pouvoir prendre une matrice d'exemples `x` en entrée (et non pas une seule donnée) et renvoyer la valeur de la fonction pour chaque exemple.

Implémentez le noyau gaussien : la fonction `kernel_gaussian(x)` qui renvoie $(2\pi)^{-d/2} e^{-\frac{1}{2}\|x\|^2}$.

Complétez la classe `KernelDensity` : le constructeur prend en argument la fonction noyau qui sera appliquée, la fonction `fit(x)` ne fait qu'enregistrer les données d'apprentissage passées en paramètre, la fonction `predict(data)` doit renvoyer l'estimation de densité en chaque point de `data`.

Faites les mêmes expériences que pour les histogrammes :

- vérifiez que vous obtenez bien une densité
- faites varier les noyaux et les hyper-paramètres, visualisez les résultats
- calculez la vraisemblance en test selon le paramètrage et tracez les courbes de vraisemblance en apprentissage et en test selon le paramètrage.

Régression par Nadaraya-Watson

L'estimateur de Nadaraya-Watson permet de faire de la classification et de la régression en utilisant une méthode très similaires aux méthodes à noyaux. Soit un ensemble d'apprentissage $\{\mathbf{x}_i, y_i\}_{i=1}^n \in \mathbb{R}^d \times \mathbb{R}$, l'estimateur de Nadaraya est défini par $f(\mathbf{x}) = \sum_{i=1}^n \frac{y_i \phi(\frac{\mathbf{x}-\mathbf{x}_i}{\sigma})}{\sum_{i=j}^n \phi(\frac{\mathbf{x}-\mathbf{x}_j}{\sigma})}$.
Codez la classe `Nadaraya` correspondant à cet estimateur. Testez votre implémentation pour prédire la note d'un POI en fonction de son emplacement. Faites varier le paramètrage et mesurez les performances en utilisant l'erreur aux moindres carrés.

Bonus : lissage, covid et pandas

L'estimateur de Nadaraya-Watson est très utilisé pour "lisser" des données séquentielles (temporelles par exemple, la moyenne glissante est un cas particulier de l'estimateur avec un noyau uniforme).
Télécharger en open data les données du COVID en France au format CSV. Utilisez le module `pandas` pour extraire le nombre de contaminations par jour depuis le début de la crise sanitaire (vous pouvez le faire en 1 ligne quasiment, c'est un bon exercice simple d'utilisation de pandas). Tracez les données brutes, puis testez différents noyaux et paramètrages pour visualiser le lissage opéré sur la courbe.