TD1: Modèles probabilistes – concepts généraux

1 Contenu d'une urne – analyse théorique

Soit une urne contenant m balles de couleur blanche ou orange. On souhaite inférer la composition de cette urne à partir de n tirages successifs avec remise.

Question 1

Rappelez les formules du calcul des probabilités (négation, produit, probabilités totales, théorème de Bayes).

Question 2

Soient $b \in \{0, 1\}$, $x, y \in \mathbb{R}$.
1. Calculer $bx + (1 - b)y$ pour les différentes valeurs de b ?
2. Même question pour $x^b y^{1-b}$?
3. Quelle concept informatique retrouve-t-on dans ces formules ?

Question 3

Rappeler la définition de la loi de Bernoulli et de la loi uniforme sur un ensemble fini.

Pour réaliser l'inférence nous allons construire un modèle probabiliste faisant intervenir la grandeur recherchée, ici le nombre de balles oranges dans l'urne, et la grandeur observée, le résultat des n tirages. Ce modèle décrira une loi de probabilité jointe sur les deux grandeurs, appelée loi *a priori*. L'inférence à proprement parler consistera à calculer la loi du nombre de balles oranges sachant le résultat des n tirages. Cette loi est appelée loi *a posteriori*.

Question 4

Construire un modèle probabiliste adapté en problème en :
1. introduisant les v.a. utiles
2. en spécifiant une loi de probabilité pour chacune

On notera ce modèle M_{urne}.

Question 5

1. Déterminer une formule pour la probabilité d'une observation (les n tirages) sachant le nombre de balles oranges dans l'urne.
2. Montrer que votre résultat fait intervenir le nombre de tirages ayant donné une boule orange.

Question 6

Déterminer une formule pour la probabilité de l'observation (les n tirages).

Question 7

1. À quelle famille de lois la loi *a posteriori* appartient-elle ?
2. Déduire des questions précédentes la loi *a posteriori* du nombre de balles oranges dans l'urne.
3. Le nombre de boules oranges tirées est un exemple de **statistique suffisante**. En faisant preuve d'imagination ou en recherchant rapidement sur le web, expliquez ce que cela signifie.

L'analyse théorique du modèle est terminée, nous allons implémenter quelques fonctions en Python pour réaliser l'application numérique confortablement. Les prochains paragraphes vont nous y préparer.

2 Mise en route pratique

Nous allons maintenant implémenter sous Python le résultat trouvé en question 7, en utilisant l'environnement Jupyter. Il s'agira de produire un *notebook* dans lequel pour chaque question, vous écrirez une fonction sans argument dont l'exécution offre une réponse à la question concernée ; on les appelera tout naturellement `questionN` où `N` est le numéro de la question.

Pour installer Jupyter et les différentes dépendances requises, il est fortement recommandé d'utiliser le gestionnaire de paquets `uv`. Si vous ne l'utilisez pas déjà il faut donc commencer par l'installer, en suivant les instructions officielles. Une fois cette étape passée, créez un répertoire `mpb-tp1` pour le projet et créez les fichiers de configuration pour `uv` :

```
$ mkdir mpb-tp1
$ cd mpb-tp1
$ uv init
$ uv python pin 3.14.2
$ uv add jupyter matplotlib numpy scipy
```

Pour lancer Jupyter

```
$ uv run jupyter lab
```

Avant de rentrer dans le vif du sujet, nous allons (re?)voir une notion importante de calcul numérique appelé **vectorisation** avec l'utilisation de la bibliothèque `numpy`.

3 Tenseurs

Un tenseur (en informatique) est un tableau multi-dimensionnel dont les éléments sont des types scalaires (càd simples) comme des nombres à virgule flottante, des entiers ou des booléens. On distingue les tenseurs en fonction de leur **ordre** : les scalaires sont des tenseurs d'ordre 0, les vecteurs des tenseurs d'ordre 1, les matrices des tenseurs d'ordre 2. L'ordre est simplement le nombre d'indices nécessaires pour désigner un élément dans le tenseur.

La bibliothèque `numpy` implémente une structure de données pour représenter les tableaux multidimensionnels appelée `ndarray` (pour *n-dimensional array*). Voyons pour commencer quelques constructeurs utiles :

```
import numpy as np

np.array([1., 2., 3.])  # construction d'un vecteur à partir d'une liste
np.zeros((3, 4))        # matrice de zéros, on spécifie les dimensions
np.ones((2, 3, 4))      # un tableau de uns, mais à triple entrée !
np.arange(10)           # 10 premiers entiers
np.linspace(2, 3, 10)   # 10 nombres également espacés entre 2 et 3
np.random.normal(size = 10) # un vecteur de 10 nombres normalement distribués
```

Les tenseurs ainsi créés peuvent être combinés à l'aide des opérateurs habituels :

```
x = np.array([1., 2., 3.])
y = np.array([4., 5., 6.])
2 * x
x ** 2
x + y
x - y
x * y
x / y
```

Vous pouvez constater que les opérateurs s'appliquent indépendamment à tous les éléments de leurs opérandes. On parle d'**opérations vectorisées**. Cela fonctionne pour des tenseurs de tout ordre. Cette notion est essentielle : avec les opérations vectorisées on peut réaliser des algorithmes raisonnablement performants malgré la piètre efficacité de l'interpréteur Python (c'est pareil en R). *Evitez les boucles tant que faire se peut et remplacez-les par des opérations vectorisées.*

Les fonctions usuelles sont également disponibles :

```
np.exp(x)
np.log(y)
np.cos(x + np.sin(y))
np.sqrt(x)    # racine carrée
```

On peut également sommer les éléments d'un tenseur :

```
np.sum(y)
```

Question 8

Calculez la somme des carrés des 100 premiers entiers.

Question 9

Leonard Euler a démontré en 1741 que :

$$\sum_{k=1}^{+\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

Servez-vous de ce résultat pour calculer une valeur approchée de π.

(🌶, optionnel) Représentez graphiquement l'erreur d'approximation en fonction du nombre de termes considérés dans la somme.

Dans la suite de ce TP, nous allons avoir besoin de représenter des fonctions. Voici un extrait de code dont vous pourrez vous inspirer et qui utilise les bibliothèques `numpy` et `matplotlib` :

```
import matplotlib.pyplot as plt

xs = np.linspace(-1, 1, 100)
ys = np.cos(xs)
plt.plot(xs, ys, label="f", color = "red")
plt.title("Cosinus")
plt.legend(loc='upper right')
```

4 Contenu d'une urne – calcul numérique

Question 10

1. Implémenter le calcul de la loi a posteriori pour M_{unif}. La fonction prendra en argument le nombre de tirages n, le nombre de boules oranges tirées s et le nombre de boules dans l'urne m.
2. Implémenter une fonction pour représenter graphiquement cette loi (voir la fonction `scatter` dans `pyplot`).

Il est temps d'exploiter notre travail : on passe à l'expérience en effectuant plusieurs tirages successifs.

Question 11

1. Observer la distribution *a priori*. Cela correspond-il à nos hypothèses de départ ?
2. Observer la distribution *a posteriori* après le premier tirage. Pour estimer une proportion à partir de comptages, il est habituel de dire qu'il faut utiliser une formule du type

$$\frac{\text{nombre de succès}}{\text{nombre de tentatives}}$$

En quoi l'approche d'inférence que nous avons suivie est-elle préférable lorsque le nombre d'observations est faible ?
3. Décrivez ce qu'il se passe au fur et à mesure que nous accumulons les tirages.

5 Estimation de fréquence allélique

Nous passons maintenant à un nouveau problème : on considère une population naturelle dans laquelle on souhaiterait évaluer la fréquence d'un allèle particulier. Pour cela on met en place une procédure de génotypage que l'on applique à un échantillon d'individus prélevés dans la population. On suppose que la procédure détecte sans erreur la présence de l'allèle chez un individu.

Question 12

1. Ce problème ressemble au précédent, mais il y a deux différences (dont une que l'on peut raisonnablement ignorer). Lesquelles ?
2. Proposer un modèle probabiliste adapté au problème.

On appelera ce modèle M_{unif}.

Pour tester la correction et la qualité d'une méthode d'inférence, il est de bon ton d'utiliser des simulations. Cela ne garantit pas que l'inférence se passera bien sur des données empiriques, mais ça rassure déjà de pouvoir vérifier son implémentation sur des exemples où la bonne réponse est connue.

Question 13

Implémentez en Python un simulateur produisant un résultat expérimental pour un grand nombre d'allèles. On devra pouvoir fournir en entrée le nombre d'allèles souhaités et le nombre d'individus testés (ce sera le même pour chaque allèle). En sortie on obtiendra un comptage pour chaque allèle ainsi que la véritable fréquence de chaque allèle.

Passons maintenant au calcul de la loi *a posteriori*. Nous sommes dans un cas plus difficile ici puisque la grandeur que nous souhaitons déterminer est continue. Cela signifie que la loi *a posteriori* est une loi continue. Une première possibilité consiste à essayer d'en déterminer la densité de probabilité en espérant retomber sur une forme connue. La deuxième consiste à utiliser un algorithme approché. Ici les deux sont possibles.

5.1 Analyse théorique du modèle (🌶🌶)

Note : cette approche est plus exigeante d'un point de vue mathématique, vous pouvez la passer si cela semble trop difficile.

Lorsque l'on travaille avec des lois continues, on remplace les probabilités par des densités de probabilité. Il faut cependant être très prudent·e avec leur manipulation parce qu'il y a des ressemblances fortes avec les formules de probabilité mais aussi de grosses différences.

Question 14

> Une probabilité est un nombre compris entre 0 et 1. Est-ce le cas pour une densité ?

Les densités admettent des formules analogues pour les formules que l'on a utilisé plus haut afin de déterminer la loi *a posteriori*. Il y a la formule des probabilités totales

$$\mathbb{P}(A) = \int_{-\infty}^{\infty} \mathbb{P}[A \mid \theta] p(\theta) \mathrm{d}\theta$$

et le théorème de Bayes

$$p(\theta \mid X = x) = \frac{\mathbb{P}(X = x \mid \theta) p(\theta)}{\mathbb{P}(X = x)}$$

Appliquons cela à M_{unif}.

Question 15

> Déterminer une formule pour la probabilité d'une mesure pour un allèle conditionnellement à la fréquence de l'allèle

Question 16

> Déterminer une formule pour la probabilité d'une mesure pour un allèle. Il sera sans doute utile de parcourir la page Wikipedia de la fonction Beta.

Question 17

> Donner l'expression de la densité de la loi *a posteriori* et montrer qu'il s'agit de la densité d'une loi Beta dont vous préciserez les paramètres.

5.2 Calcul approché de la loi *a posteriori*

Il n'est pas très courant de pouvoir dériver une forme analytique de la loi *a posteriori* d'un modèle probabiliste. Nous allons ici explorer une deuxième possibilité consistant à calculer une approximation de la loi *a posteriori*. Il existe pour cela des quantités d'algorithmes, par exemple les méthodes Monte Carlo (MCMC, *Importance Sampling*, Monte Carlo séquentiel, *etc*) ou les méthodes variationnelles. Ici il n'y a qu'une variable non observée donc on peut recourir à une méthode très simple : si on discrétise l'ensemble des valeurs possibles pour la fréquence allélique, on retombe tout simplement sur le modèle M_{urne}.

Question 18

> Visualisez la distribution *a posteriori* obtenue pour une observation de votre choix en faisant varier M et commentez.

5.3 Estimation ponctuelle

Quand il est trop compliqué ou trop coûteux de calculer la distribution *a posteriori* d'une variable d'intérêt, on peut se contenter d'une estimation ponctuelle, c'est-à-dire de prédire une seule valeur.

Question 19

> Donner au moins trois résumés statistiques fournissant une valeur "centrale" ou "typique" d'une distribution ou d'un échantillon.

Le modèle M_{unif} est de la forme

$$\theta \sim \mathcal{D}_\theta$$
$$X \sim \mathcal{D}_X(\theta)$$

où θ est un paramètre continu. Il existe une version continue du théorème de Bayes :

$$p(\theta \mid X = x) = \frac{\mathbb{P}(X = x \mid \theta) p(\theta)}{\mathbb{P}(X = x)}$$

qui permet de calculer la **densité** de la loi *a posteriori*. Une stratégie calculatoirement favorable pour fournir un estimateur ponctuel est de calculer le mode de cette densité, c'est-à-dire la valeur de θ qui maximise la fonction $\theta \mapsto p(\theta \mid X = x)$. On parle d'estimateur MAP (pour Maximum A Posteriori).

Question 20

> 1. Expliquer pourquoi il est inutile de connaître $\mathbb{P}(X = x)$ pour calculer l'estimateur MAP et pourquoi c'est une bonne nouvelle.
> 2. Donner dans le cas du modèle M_{unif} l'expression du terme $\mathbb{P}(X = x \mid \theta) p(\theta)$.
> 3. Plutôt que de travailler directement sur la densité *a posteriori* on préfère son logarithme.
>   - expliquer pourquoi cela est équivalent
>   - expliquer pourquoi cela est préférable d'un point de vue numérique
>   - expliquer en quoi cela peut-il faciliter les calculs
> 4. Donner l'expression du logarithme de la densité *a posteriori* (on la notera f).
> 5. Déterminer analytiquement la valeur de fréquence allélique qui maximise f.

5.4 Choix de la distribution *a priori*

Nous avons énoncé à travers M_{unif} que la distribution *a priori* sur la fréquence allélique est uniforme entre 0 et 1. C'est une description possible de notre état de connaissance des fréquences allélique mais on peut le remettre en cause avec un peu de bagage en génétique des population. Sous certaines hypothèses on peut démontrer que la fréquence d'un allèle neutre est distribuée selon une loi Beta, plus précisément

$$\mathrm{B}(\alpha, \alpha)$$

où $\alpha = 4N\mu$ où N est la taille de population, μ le taux de mutation. Pour la population considérée on prendra $N = 10^5$ et $\mu = 2,5 \times 10^{-8}$.

Question 21

> Comment modifier M_{unif} pour prendre en compte cette information ?

On appelle ce nouveau modèle M_{beta}.

Question 22

> Implémenter un simulateur de M_{beta} et visualiser la loi *a priori* pour les fréquences alléliques. Quelle est qualitativement la différence ?

Question 23

> (🌶🌶) Déterminer de manière analytique la distribution *a posteriori* de M_{beta}.

Question 24

> Déterminer l'estimateur MAP sous M_{beta}. Comparer à l'estimateur MAP sous M_{unif} et commentez.

Question 25

> En supposant les données générées sous M_{beta}, comparez par des simulations la performance des estimateurs MAP des deux modèles.