# TP : Théorie Bayesienne

Le but de ce TP est de vous familiariser avec l'environnement de calcul numérique Python/Numpy en travaillant sur un problème de reconnaissance de formes. Vous allez appliquer dans ce TP les méthodes de reconnaissance de formes vues dans le cours "Théorie bayesienne de la décision".

## 1 Présentation des données

Les données sur lesquelles vous allez travailler sont des données biomédicales d'aide au diagnostic du diabète. Les données ont été obtenues à partir d'un échantillon de 709 femmes amérindiennes Pimas. Les Pimas sont connus pour avoir le plus important taux de diabétiques et d'obésité au monde, il sont à ce titre un sujet d'étude pour les scientifiques.

Le but est de prédire si un sujet est diabétique à partir d'un certain nombre de variables mesurées pour chaque sujet :

1. Nombre de grossesse.
2. Concentration en glucose du plasma 2 heures après un test de tolérance au glucose oral.
3. Pression sanguine diastolique (mm Hg)
4. Épaisseur de la peau sur le triceps (mm)
5. 2-heures d'insuline en sérum ( $\mu$U/ml)
6. Indice de masse corporelle (poids en kg/(taille en m)$^2$)
7. Fonction de pedigree du diabète
8. Age (années)

Bien que dans la pratique, l'utilisation de toutes les variables permet d'obtenir de meilleurs taux de reconnaissance, dans ce TP, nous nous concentrerons sur deux variables : la concentration en glucose (2) et l'indice de masse corporelle (6). Ceci nous permettra de visualiser à la fois les données et les fonctions de classification.

Les données sont originaires du site web de l'université de Californie [1]. Elles doivent être téléchargées sur le site du cours [2]. Le fichier "pima.npz" contient trois variables :

— **xall** : matrice de taille $709 \times 8$ contenant les données biomédicales.
— **y** : vecteur de taille 709 contenant la classe de chaque sujet (-1 : non diabétique, +1 : diabétique).
— **n** : variable égale à 709, le nombre d'exemples d'apprentissage.

Notez que nous avons au préalable retiré un certain nombre de sujets ayant des données manquantes.

1. http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes
2. https://remi.flamary.com/cours/rdf_icm.fr.html

## 2 Analyse exploratoire des données

**Chargement des données et pré-traitement**

- Télécharger le fichier "pima.npz".
- Charger ce fichier sous matlab en utilisant la fonction `np.load`.
- Extraire les variables (2) et (6). Stocker le résultat dans une matrice d'apprentissage **xtemp**.

**Visualisation du problème**

- Visualiser des histogrammes de chaque classe pour chaque variables (fonction `pl.hist`).
- Visualiser les exemples de chaque classe en 2D en leur mettant des couleurs différentes (fonction `pl.plot,pl.scatter`).

## 3 Décision Bayésienne

**Classifieur Bayesien : Cas 1** Premièrement, vous allez implémenter le classifieur Bayésien correspondant au cas 1 du cours.

- Estimer les probabilités empiriques $P(w_i)$ d'appartenir à chaque classe(`np.mean`).
- Extraire les variables (2) et (6) et les normaliser (centrer et réduire, `np.mean,np.std`) en conservant leur moyenne et leur écart-type. Stocker le résultat dans la matrice d'apprentissage $\mathbf{x}$. La variance $\sigma$ est connue et égale à un car les données ont été normalisée.
- Calculer les moyennes $\mu_i$ de chaque classe (`np.mean`).
- Calculer les valeurs des fonctions $g_i(\mathbf{x})$ pour chaque exemple de test en utilisant la formule du cours (`np.dot`).
- Calculer la valeur de la fonction de prédiction $f(\mathbf{x}) = g_1(\mathbf{x}) - g_{-1}(\mathbf{x})$ pour les exemples de test.
- Évaluer le taux de bonne reconnaissance du classifieur. Discuter des performances. Interpréter le classifieur (`==,np.mean`).

**Classifieur Bayesien : Cas général** Finalement, vous aller coder le classifieur Bayésien correspondant au cas 5 du cours. Ce dernier peut être utilisé quelque soit votre *a priori* sur la matrice de covariance.

- Estimer les probabilités empiriques $P(w_i)$ d'appartenir à chaque classe.
- Calculer les moyennes $\mu_i$ de chaque classe.
- Estimer les covariances $\Sigma_i$ de chaque classe avec la fonction *cov*. La matrice de covariance $\Sigma$ est estimée empiriquement sur une matrice $\mathbf{X}$ par la formule (`np.cov`) :

$$\Sigma_{i,j} = \frac{1}{n}\sum_{k=1}^{n}(X_{k,i} - \bar{x}_i)(X_{k,j} - \bar{x}_j) \tag{1}$$

  où $\Sigma_{i,j}$ est le terme générale de $\Sigma$ contenant la covariance entre les variables $i$ et $j$, et $\bar{x}_i$ la valeur moyenne de la variable $i$ (0 dans notre cas car les données sont normalisées).
- Calculer les valeurs des fonctions $g_i(\mathbf{x})$ pour chaque classe et chaque exemple de test en utilisant la formule du cours (`for,np.dot`).
- Calculer la valeur de la fonction de prédiction $f(\mathbf{x}) = g_1(\mathbf{x}) - g_{-1}(\mathbf{x})$ pour les exemples de test.

BONUS : Visualiser la frontière de décision du classifieur 2D sur le plan et la frontière de décision correspondante (utilisation de `np.meshgrid` et `pl.contour`).