TME 10 : Processus Gaussien

Gaussiennes en folie

Le but de cette partie est de comprendre comment la matrice de covariance d'une gaussienne influe sur la distribution des dimensions. Nous allons étudier tout d'abord le cas 2D, puis le cas en N dimensions. Vous aurez besoin des fonctions : `multivariate_normal` de `scipy.stats` qui permet de calculer entre autre la densité d'une gaussienne et de `multivariate_normal` de `numpy.random` qui permet plus facilement que la fonction précédente de tirer des points aléatoirement d'une gaussienne. Si par la suite un message vous indique que votre matrice de covariance est singulière (i.e. non inversible), ajoutez un petit bruit aléatoire sur la diagonale vous permettra de la rendre inversible.
Dans un premier temps, considérez trois gaussiennes en 2 dimensions telles que :

- la première ait ses axes indépendants (matrice de covariance diagonale avec un $\sigma = 1$ sur la diagonale)
- la deuxième ait ses 2 dimensions très peu corrélées (par exemple $cov(x_1, x_2) = 0.1$)
- la troisième ait ses 2 dimensions très corrélées (par exemple $cov(x_1, x_2) = 0.9$.)

Pour chacune d'entre elles :

- représentez la densité de la distribution sur l'espace 2d (en utilisant `np.meshgrid` et `plt.contour`)
- représentez sur la même figure un millier de points tirés selon la distribution
- tracez quelques points de la distribution selon la représentation suivante : pour un point $\mathbf{x} = (x_1, x_2)$ de la gaussienne, le tracé correspond au segment $[(1, x_1), (2, x_2)]$.

Observez la différence entre chaque schéma selon la valeur de la matrice de covariance. Comment s'illustre une covariance faible ? élevée ?
Pour quelques dimensions D (10, 100, 200), engendrez quelques matrices de covariances telles que toutes les covariances aient la même valeur sauf la diagonale fixée à 1. Représentez quelques échantillons selon la dernière représentation en 1D : pour un point échantillonné $\mathbf{x} = (x_1, \ldots, x_D)$, tracez la courbe $[(1, x_1), (2, x_2), \ldots, (D, x_D)]$ (qu'on appellera par la suite *vue 1D* de l'échantillon). Qu'observez vous selon la valeur de la covariance ?

Fonction de covariance (Kernel)

Implémentez les trois noyaux suivants :

- la covariance linéaire : $k_{lin}(\mathbf{x}^1, \mathbf{x}^2) = \sigma^2 \langle \mathbf{x}^1, \mathbf{x}^2 \rangle$
- le noyau exponentiel : $k_{rbf}(\mathbf{x}^1, \mathbf{x}^2) = \sigma^2 e^{-\frac{1}{2l^2}\|x^2 - x^1\|^2}$
- le noyau périodique : $k_{per}(\mathbf{x}^1, \mathbf{x}^2) = \sigma^2 e^{-\frac{2}{l^2} sin^2(\pi \|\mathbf{x}^2 - \mathbf{x}^1\|/p)}$

Vos fonctions doivent admettre des matrices de données à chaque fois (de $N \times D$ nombre d'échantillons par nombre de dimension). Vous pouvez utiliser la fonction `pairwise_distances` de `sklearn.metrics`. Tracez la matrice de covariance pour une discrétisation d'un domaine fixe en 1 dimension (par exemple $[-5, 5]$) à l'aide de la fonction `plt.imshow`. Observez l'effet de la paramétrisation et du noyau. Tracez également la vue $1D$ de quelques échantillons pour chaque matrice de covariance.

Processus gaussien et régression

On suppose que l'on dispose d'un jeu de données $D = \{\mathbf{x}^i, y^i\}_{i=1}^N$ où chaque $y^i = f(\mathbf{x}) + \epsilon$ avec f inconnue et $\epsilon \sim \mathcal{N}(0, \sigma^2)$. En notant $\mathbf{x}$ les points d'entraînement, $\mathbf{y}$ les étiquettes d'entraînement, $\mathbf{x}^t$ les points test et $\mathbf{y}^t$ les labels à inférer, la régression par processus gaussien établit par marginalisation que $p(\mathbf{y}^t|\mathbf{x}, \mathbf{y}, \mathbf{x}^t) \sim \mathcal{N}(\boldsymbol{\mu}_t, K_t)$ avec

$$\boldsymbol{\mu}_t = K_*^t[K + \sigma^2 I]^{-1}\mathbf{y}, \quad K_t = K_{**} - K_*^t(K + \sigma^2 I)^{-1}K_*$$

avec K la matrice de covariance entre points d'entraînement, K_* la matrice de covariance entre points d'entraînement et points test et K_{**} la matrice de covariance entre points test (en supposant les données centrées). La matrice de covariance de $\begin{bmatrix}\mathbf{x}\\ \mathbf{x}^t\end{bmatrix}$ s'écrit $\begin{bmatrix}K & K_*\\ K_*^t & K_{**}\end{bmatrix}$.

Ecrivez une fonction qui prend en entrée les données et les points tests, le noyau et σ le bruit estimé et qui renvoie l'estimation des valeurs de $\mathbf{y}^t$ et de la covariance associée. Testez votre fonction sur un des exemples précédents en ne donnant que quelques points en apprentissage. Vous tracerez sur un graphe la vue 1 d de l'échantillon, les points passés en apprentissage, l'estimation $\boldsymbol{\mu}^t$ de votre fonction et une fenêtre à $\pm 2 * \sqrt{var(\mathbf{x})}$ (avec la fonction `plt.fill_between`) par exemple. Faites varier le nombre de points passé en apprentissage et faites évoluez les hyper-paramètres des noyaux.

Essayez de régresser la fonction $x * sin(x)$. Etudiez l'effet des hyper-paramètres et du choix de noyau. Essayez sur un vrai jeu de données (par exemple boston housing).