IFT 603 : Devoir 3
Travail individuel

Remise : 20 mars 2015, 17h00 (**au plus tard**).

Remettez votre solution au numéro 1 en format papier et au numéro 2 via *turnin*.

1. **[4 points]** Supposons que les entrées $\mathbf{x}_n = (x_{n1}, \ldots, x_{nD})^{\mathrm{T}}$ à modéliser soient binaires, c'est-à-dire que chaque élément $x_{ni} \in \{0, 1\}$. Pour ce type d'entrée, l'utilisation de gaussiennes n'est pas appropriée, puisqu'une gaussienne peux générer d'autres valeurs que 0 ou 1.

   Un meilleur choix de distribution est la produit de Bernoulli. Spécifiquement, un produit de Bernouilli suppose que chacun des éléments x_{ni} de $\mathbf{x}_n$ a été généré indépendamment d'une Bernoulli, et est égal à 1 avec probabilité μ_i (et 0 avec probabilité $1 - \mu_i$).

   Supposons qu'on souhaite utiliser un mélange de K produits de Bernoulli. Le k^{e} produit utilise les probabilités $\boldsymbol{\mu}_k = (\mu_{k1}, \ldots, \mu_{kD})^{\mathrm{T}}$ pour modéliser les probabilités que chacun des éléments $x_{n1}, \ldots, x_{nD}$ soit 1. En d'autres mots, la probabilité que le k^{e} produit ait généré $\mathbf{x}_n$ est

$$p(\mathbf{x}_n | z_{nk} = 1) = \prod_{i=1}^{D} \mu_{ki}^{x_{ni}} (1 - \mu_{ki})^{(1 - x_{ni})} \tag{1}$$

   Vous devez dériver l'algorithme EM pour entraîner un tel mélange de K produits de Bernoulli.

   Spécifiquement, vous devez :

   (a) **Étape E :** Donner la formule correspondant aux probabilités d'appartenance $\gamma(z_{nk}) = p(z_{nk} = 1 | \mathbf{x}_n)$, pour le cas spécifique d'un mélange de produits de Bernoulli.

   (b) **Étape M :** Montrer que, étant données les probabilités d'appartenance $\gamma(z_{nk})$ fixes, l'étape M correspond à mettre à jour les probabilités π_k comme suit

$$\pi_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk})}{N} \tag{2}$$

   et les vecteurs $\boldsymbol{\mu}_k$ de probabilités du produit de Bernoulli comme suit

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \quad . \tag{3}$$

   Pour ce faire, vous devez montrer que les équations ci-haut correspondent au maximum de la borne inférieur à la log-probabilité d'entraînement ($\ln p(\mathbf{X}) = \sum_n \ln p(\mathbf{x}_n)$)

$$\mathcal{L}(q, \theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln(p(\mathbf{x}_n | z_{nk} = 1) p(z_{nk} = 1)) + \text{const} \; . \tag{4}$$

   Vous devez donc calculer les gradients de cette borne par rapport à π_k et $\boldsymbol{\mu}_k$, les fixer à 0 et résoudre le système d'équations associé. Dans le cas de π_k, n'oubliez pas de respecter la contrainte de sommation à 1, à l'aide d'un multiplicateur de Lagrange.

2. **[6 points]** Programmez l'algorithme EM pour l'entraînement d'un mélange de gaussiennes. Pour ce faire, vous devez télécharger et décompresser le fichier `devoir_3.zip` du site web du cours.

L'algorithme doit être implémenté sous la forme d'une classe `MelangeGaussiennes`. Votre implémentation de cette classe doit être placée dans le fichier `solution_melange_gaussiennes.py`, qui contient déjà une ébauche de la classe. Veuillez vous référer aux "docstrings" (la chaîne de caractères sous la signature de chaque méthode) des méthodes de la classe `MelangeGaussiennes` afin de savoir comment les implémenter.

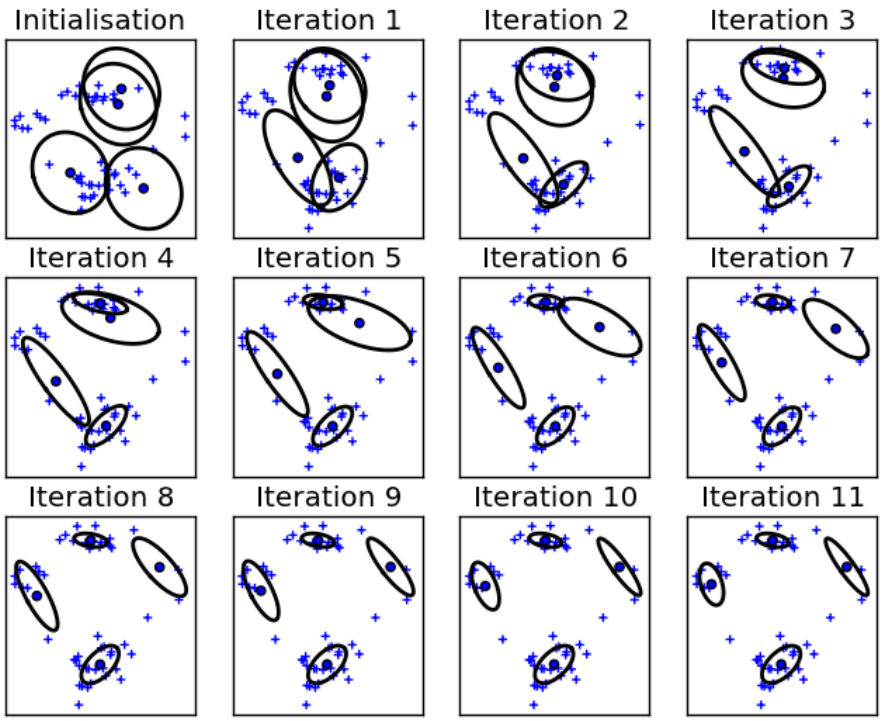
À noter que, lors de la mise à jour de chacune des matrices de covariance Σ_k, vous devez ajouter une constante λ à sa diagonale (argument `lamb` du constructeur de `MelangeGaussiennes`), afin d'assurer qu'elles soient inversibles.

Le fichier `solution_melange_gaussiennes.py` sera importé par le script `melange_gaussiennes.py`, qui exécute votre code sur les données d'entraînement et mesure la performance du modèle de densité sur les ensembles d'entraînement et de test. Ce script nécessite également que les fichiers de comparaison avec une implémentation correcte suivants soient présents dans le même répertoire :

- `solution_p_appartenance.pkl`
- `solution_p_gaussienne_k.pkl`
- `solution_predictions_entrainement.pkl`
- `solution_erreurs_entrainement.pkl`

Les données utilisées dans ce numéro ont été générées artificiellement, d'un mélange de 4 gaussiennes. Une implémentation correcte obtiendra une erreur (log-probabilité moyenne négative) d'entraînement de -0.7755 et une erreur de test de 0.3055.

Le script `melange_gaussiennes.py` entraîne un mélange de $K = 4$ gaussiennes pour 11 itérations. Il générera une illustration de l'initialisation et du résultat des 11 itérations de EM. Une implémentation correcte de l'algorithme EM donnera la visualisation suivante :



Le script `melange_gaussiennes.py` peut également être appelé afin d'entraîner un mélange de gaussiennes avec d'autres valeurs de K et du nombre d'itérations. Par exemple, l'exécution de la commande suivante entraînera un mélange avec $K = 8$ gaussiennes et 5 itérations :

```
python melange_gaussiennes.py 8 5
```

Dans votre implémentation, vous aurez besoin de calculer des déterminants de matrice. Pour ce faire, vous pouvez utiliser la fonction `numpy.linalg.det` de NumPy. La constante π (3.14159...) est également accessible via `numpy.pi`.
Vous devez remettre votre solution via l'outil *turnin*, comme suit :

```
turnin -c ift603 -p devoir_3 solution_melange_gaussiennes.py
```