IFT 603 : Devoir 2
Travail individuel

Remise : 26 février 2014, 17h00 (**au plus tard**).

Remettez votre solution aux numéros 1 et 2 en format papier et au numéro 3 via *turnin*.

1. **[2 points]** Démontrez que si $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, i.e. que $\mathbf{x}$ est la concaténation des sous-vecteurs $\mathbf{x}_a$ et $\mathbf{x}_b$, et que si $k_a(\mathbf{x}_a, \mathbf{x}'_a)$ et $k_b(\mathbf{x}_b, \mathbf{x}'_b)$ sont des noyaux valides et applicables aux sous-vecteurs $\mathbf{x}_a$ et $\mathbf{x}_b$ respectivement, alors le noyau $k(\mathbf{x}, \mathbf{x}')$ suivant est également valide :

$$k(\mathbf{x}, \mathbf{x}') \quad = \quad k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \ .$$

2. **[2 points]** En général, pour un problème de classification binaire, on suppose que la perte (erreur) $L(t, \hat{y}(\mathbf{x}))$ de classifier dans la classe $\hat{y}(\mathbf{x})$ une entrée de la classe t est simplement 1 si $\hat{y}(\mathbf{x}) \neq t$ et 0 sinon.

   Imaginez maintenant un problème de classification où la perte n'est pas symétrique. Ainsi, $L(1,0) \neq L(0,1)$, i.e. la perte de classifier une entrée de la classe 1 dans la classe 0 n'est pas la même que la perte de classifier une entrée de la classe 0 dans la classe 1. La perte de classifier dans la bonne classe demeure 0, i.e. $L(1,1) = L(0,0) = 0$.

   Démontré que la décision optimale est alors :

$$\hat{y}(\mathbf{x}) = \begin{cases} 1 & \text{si } p(\mathcal{C}_1|\mathbf{x}) \geq \frac{L(0,1)}{L(1,0)+L(0,1)} \\ 0 & \text{sinon} \end{cases}$$

3. **[6 points]** Programmez l'algorithme de la classification probabiliste générative. Pour ce faire, vous devez télécharger et décompresser le fichier `devoir_2.zip` du site web du cours.

   L'algorithme doit être implémenté sous la forme d'une classe `ClassifieurGeneratif`. Votre implémentation de cette classe doit être placée dans le fichier `solution_classifieur_generatif.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 `ClassifieurGeneratif` afin de savoir comment les implémenter. Votre implémentation doit être efficace et utiliser les fonctionalités de la librairie Numpy (e.g. vous devez éviter les boucles `for`).

   À noter que, pour ce problème, le calcul de la matrice de covariance Σ telle que spécifiée par l'équation 4.78 du livre de Bishop résultera en une matrice qui n'est pas inversible. Ainsi, on vous demande d'ajouter une constante λ à sa diagonale (argument `lamb` du constructeur de ClassifieurGeneratif), ce qui la rendra inversible.

   De plus, vous devez implémenter une méthode `pretraitement_donnees` prenant comme argument un nom de fichier texte contenant des données (d'entraînement ou de test) et retournant la matrice des entrées $\mathbf{X}$ (i.e. un tableau Numpy 2D où chaque rangée est une entrée $\mathbf{x}_n$) et le vecteur de cibles associées $\mathbf{t}$ (i.e. un tableau Numpy 1D). Veuillez également vous référer à la "docstrings" de cette fonction, dont la signature se trouve déjà dans le fichier `solution_classifieur_generatif.py`.

Le fichier `solution_classifieur_generatif.py` sera importé par le script `classifieur_generatif.py`, qui exécute votre code sur les données d'entraînement et mesure la performance du modèle de classification sur les ensembles d'entraînement et de test. Ce script nécessite également que les fichiers suivants soient présents dans le même répertoire :

- Données d'entraînement et de test (en format texte) :
  - `adult.data`
  - `adult.test`
- Fichiers de comparaison avec une implémentation correcte :
  - `solution_pretraitement_entrainement.pkl`
  - `solution_predictions_entrainement.pkl`
  - `solution_erreurs_entrainement.pkl`

Les données utilisés dans ce numéro proviennent du *Adult Data Set*[1]. La tâche est de déterminer, à partir de diverses informations sur une personne (âge, niveau d'éducation, état civil, pays de naissance, etc.), si cette personne a un salaire annuel dépassant les 50 000$.

Une implémentation correcte obtiendra une erreur d'entraînement de 0.1578 (15.78%) et une erreur de test de 0.1562 (15.62%).

Vous devez remettre votre solution via l'outil *turnin*, comme suit :

```
turnin -c ift603 -p devoir_2 solution_classifieur_generatif.py
```

1. Pour en savoir plus, voir `http://archive.ics.uci.edu/ml/datasets/Adult`.