IFT 603-712 : TP 3

Travail par équipe de 2 ou 3

Remettez votre solution aux numéros 1 et 2 en format pdf ou manuscrit (et scanné) via **turnin**. Remettez aussi votre solution du numéro 3 via **turnin**.

→ **DATE LIMITE DE REMISE : 24 mars 2022, 23h59.**

1- [2 points] Prouvez qu'à l'aide de la représentation duale, la solution à l'équation de la régression linéaire de type maximum a posteriori (voir Eq.(1.67) et (6.2) de Bishop)

$$J(\vec{w}) = \frac{1}{2}\sum_{n=1}^{N}(\vec{w}^T\vec{\phi}(x) - t_n)^2 + \frac{\lambda}{2}\vec{w}^T\vec{w}$$

est donnée par l'équation (6.9) du livre de Bishop. **NOTE** : la simple énumération des équations (6.2) à (6.9) est insuffisante. Vous devez expliciter le contenu de chaque variable et clairement exprimer la transition entre chaque équation.

2- [2 points] Dites ce qu'est un vecteur de support et en quoi il contribue à la fonction de prédiction $y_w(\vec{x})$. Expliquez également le lien qu'il y a entre l'erreur de *Hinge* (la *Hinge loss*) et la machine à vecteurs de support.

3- [6 points] Programmez un algorithme de classification non linéaire à noyaux et dont la fonction d'erreur est celle du maximum a posteriori exprimée à l'équation (6.2) du livre de Bishop. Pour ce faire, vous devez télécharger le fichier **ift603_tp3_prog.zip** du site web du cours.

L'algorithme doivent être implémenté à l'intérieur de la classe **`MAPnoyau`**, dans le fichier **`map_noyau.py`** qui contient déjà une ébauche. Vous devez également calculer l'erreur de classification et de validation dans la fonction **`main()`**. Aussi, afin de réduire au maximum l'erreur de validation, il vous faut faire une sélection de paramètres de type « *grid search* ». Veuillez-vous référer aux commentaires sous la signature de chaque méthode de la classe **`MAPnoyau`** afin d'avoir plus de détails quant à leur implantation. Vous devez implémenter quatre noyaux, soient

$$\text{Linéaire} : k(x, x') = x^T x'$$
$$\text{Rbf} : k(x, x') = \exp\left(\frac{-\|x-x'\|^2}{2\sigma^2}\right)$$
$$\text{Polynomial} : k(x, x') = (x^T x' + c)^M$$
$$\text{Sigmoidal} : k(x, x') = \tanh(bx^T x' + d).$$

À noter que votre implémentation doit être efficace et utiliser les fonctionnalités de la librairie Numpy (**évitez les boucles `for` inutiles**).

Note 1 : bien que vide, le code de la classe **`MAPnoyau`** s'exécute déjà. Pour vous en convaincre, vous n'avez qu'à taper la commande suivante dans un terminal :

```
python non_lineaire_classification.py rbf 100 200 0 0
```

Note 2 : le code des devoirs (ainsi que des notebooks) a été testé avec python 3.5.2. Pour faire fonctionner votre code sur les ordinateurs du département, vous devez

1. démarrer une session linux (ubuntu)
2. démarrer un terminal
3. normalement, si vous démarrez une session ipython (tappez `ipython` dans le terminal) vous verrez « `Python 3.5.2` » (ou plus récent).

Note 3 : il est recommandé de rédiger son code dans un ide comme `spyder` ou `pycharm`.

Note 4 : pour exécuter le code des notebooks disponibles sur le site web du cours, vous devez tapper la commande « `jupyter notebook` » dans un terminal.