IFT 603-712 : Devoir 2

Travail par équipe de 2 ou de 3

Remettez votre solution aux numéros 1, 2 et 3 en format pdf ou manuscript (et scanné) via **turninweb**. Même chose pour le code.

1- [1.5 point] Prouvez que la régression de Ridge ayant pour objectif de minimiser la fonction d'énergie suivante (voir les notes pdf sur la régression linéaire):

$$E(\vec{w}) = \sum_{n=1}^{N} \left(t_n - \vec{w}^T \vec{\phi}(\vec{x}_n) \right)^2 + \lambda \vec{w}^T \vec{w}$$

a pour solution l'équation suivante :

$$\vec{w} = \left(\Phi^T \Phi + \lambda I \right)^{-1} \Phi^T t \quad .$$

2- [2.5 points] Nous avons vu que l'objectif de la régression logistique est de trouver un vecteur de paramètres $\vec{w}$ pouvant reproduire la probabilité conditionnelle suivante

$$p\left(C_1 \middle| \vec{\phi}(\vec{x}) \right) = \sigma\left(\vec{w}^T \vec{\phi}(\vec{x}) \right).$$

Prouvez que lorsque la fonction de perte est une entropie croisée (*cross-entropy)*, le gradient utilisé pour effectuer la descente de gradient est déterminé par la fonction suivante :

$$\vec{\nabla} E(\vec{w}) = \sum_{n=1}^{N} \left(y(\vec{x}_n) - t_n \right) \vec{\phi}(\vec{x}_n).$$

3- [1.0 point] Soit une variable aléatoire X pouvant prendre 3 valeurs possibles {1, 2, 3} avec probabilités $p(X = 1) = p_1$, $p(X = 2) = p_2$ et $p(X = 3) = p_3$. Démontrez que la loi de probabilité ayant l'entropie la plus élevée et satisfaisant la contrainte $p_1 = 2p_2$ a les probabilités suivantes :

$$p_1 = \frac{2}{2^{2/3}+3} \qquad p_2 = \frac{1}{2^{2/3}+3} \qquad p_3 = \frac{2^{2/3}}{2^{2/3}+3}$$

Pour ce faire, utilisez des multiplicateurs de Lagrange afin de tenir compte de la contrainte de sommation à 1 et de la contrainte $p_1 = 2p_2$.

4-[5 points] Programmez des algorithmes de classification linéaire fonctionnant en 2D. Pour ce faire, vous devez télécharger le fichier **devoir2.zip** du site web du cours.

Les algorithmes doivent être implémentés à l'intérieur de la classe **`ClassifieurLineaire`**, dans le fichier **`solution_classification_lineaire.py`** qui contient déjà une ébauche. Le premier algorithme (**2.25 points**) est l'approche générative vue en classe et présentée à la section 4.2.2 du livre de Bishop. La deuxième méthode est celle du Perceptron (**2.25 points**) également vue en classe et présentée à la section 4.1.7 du livre de Bishop. Pour ce faire, il vous faudra coder la <u>descente de gradient stochastique</u> vue en classe. Vous devez également implémenter la méthode du Perceptron proposée par la librairie `sklearn`[1] (**0.5**

1http://scikit-learn.org/

point). Il est d'ailleurs recommandé de commencer par cette méthode qui est la plus facile à coder (à peine 4 lignes dans la solution du prof!).

Veuillez vous référer aux commentaires sous la signature de chaque méthode de la classe **`ClassifieurLineaire`** afin d'avoir plus de détails quant à leur implantation. À noter que votre code 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 **`classificationLineaire`** s'exécute déjà. Pour vous en convaincre, vous n'avez qu'à taper la commande suivante dans un terminal :

```
python classifieur_lineaire.py 1 280 280 0.001 1 1
```

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 votre code dans un ide tel S*pyder* 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.