Année 2020-2021
Emmanuel Franck Mathématiques Appliquées L3

Informatique S6 - CC4

Le but de ces exercices est d'étudier le problème de régression à noyau en grande dimension. On se donne des données $(x_1, ..., x_n)$ avec $x_i \in \mathbb{R}^d$ et $(y_1, ..., y_n)$ avec $y_i \in \mathbb{R}$. Le problème de régression à noyau consiste à déterminer le paramètre $\theta \in \mathbb{R}^n$ qui minimise la fonction

$$J(\theta) = \| A\theta - b \|_2^2 + R(\theta)$$

où A une matrice de taille $n \times n$ définie par

$$A_{ij} = k(x_i, x_j)$$

où $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ une une fonction appelée noyau, $b = (y_1, ..., y_n)^T \in \mathbb{R}^n$ et $R : \mathbb{R}^n \to \mathbb{R}_+$ est une fonction dite de régularisation utile en grande dimension, lorsque la dimension d est bien plus grande que le nombre de données n. Le problème de minimisation :

$$\min_{\theta \in \mathbb{R}^n} J(\theta)$$

est résolu par une méthode de gradient.

Exercice 1 (**Classe Matrice** (3.25 points, sur papier))**.**

Un template de classe `matrice` *vous est fourni.*

Question (1.0 points)**.**
Expliquez comment est stockée une matrice : structures de données utilisées, tableau 1d ou 2d, taille etc. Justifiez les choix effectués.

Question (0.75 points)**.**
Expliquez pourquoi on n'a pas eu besoin de faire de boucles (comme en cours) pour la copie de la matrice dans le "="

Question (1.5 points)**.**
Donner les types de retour des fonctions `transpose`*,* `produit`*,* `+`*,* `*`*,* `-` *et expliquez pourquoi certaines fonctions renvoies des données extérieures à la classe.*

Exercice 2 (**Régularisation** (7.5 points))**.**

On programme tout d'abord la partie régularisation.

Question (1.25 points)**.**
Ecrire un template de classe abstraite `regularisation` *avec deux paramètres de template : une classe K et un entier* n*. Elle condiendra 2 fonctions virtuelles pures :*

— *une fonction* `reg` *qui prend une matrice de type K et de taille* $n \times 1$ *et renvoie un nombre de type K*
— *une fonction* `derivate_reg` *qui prend une matrice de type K et de taille* $n \times 1$ *et renvoie une matrice de même taille*

Ces fonctions génériques contiendront $R(\theta)$ *et* $\nabla_\theta R(\theta)$.

Question (3.0 points)**.**
Ecrire un template de classe `regularisation_ridge` *qui hérite de régularisation.*

Elle contient un attribut α*. Ecrivez les constructeurs par défaut, par copie, un constructeur prenant* α *en entrée et le destructeur. Définissez la fonction* `reg` *qui calcule :*

$$R(\theta) = \frac{\alpha}{2} \parallel \theta \parallel_2^2 = \frac{\alpha}{2} \sum_{i=1}^{n} \theta_i^2$$

en utilisant les fonctions de la classe matrice. Ecrivez la fonction `derivate_reg` *:*

$$R(\theta) = \alpha\theta$$

en utilisant les fonctions de la classe matrice.

Question (3.25 points)**.**
Ecrire un template de classe `regularisation_lasso` *qui hérite de régularisation.*

Ecrivez les constructeurs par défaut, par copie, un constructeur prenant α *en entrée et le destructeur. Définissez la fonction* `reg` *qui calcule :*

$$R(\theta) = \frac{\alpha}{2} \parallel \theta \parallel_1 = \alpha \sum_{i=1}^{n} \mid \theta_i \mid$$

en utilisant les fonctions de la classe matrice. Ecrivez la fonction `derivate_reg` *qui calcule* $\nabla_\theta R(\theta) = (\partial_{\theta_i} R(\theta))_i$ *avec*

$$\partial_{\theta_i} R(\theta) = \begin{cases} 1, & si\ \theta_i > 0, \\ -1, & si\ \theta_i \leqslant 0. \end{cases}$$

Question (Sur papier (0.5 points))**.**
Rappeler les priopriétés d'une classe abstraite et expliquer pourquoi cela convient à la classe `regularisation`.

Exercice 3 (**Régression** (8.25 points))**.**

On va maintenant coder la régression

Question ((2.25 points))**.**
Ecrire un template de classe `regression` *avec trois paramètres de template : une classe K, un entier* n *(taille de l'échantillon) et un entier* d *(taille des données* x*). Elle contiendra 5 attributs :*

— *une matrice* A *de type K et de taille* $n \times n$
— *une matrice* b *de type K et de taille* $n \times 1$
— *une matrice* θ *de type K et de taille* $n \times 1$
— *un pointeur de fonction* `kernel` *de signature : sortie : K, entrée : un entier et deux* `valarray<K>`.
— *un pointeur sur un objet de type* `regularisation`

Coder le constructeur par défaut, par copie, un constructeur qui prend un pointeur de fonction et un pointeur sur un objet `régularisation` *en entrée et le destructeur.*

Question (Sur papier (1.0 points)).
Pourquoi dans la classe ci-dessus est-on obligé d'utiliser un pointeur sur un objet `régularisation` *au lieu de déclarer directement un objet ?*

Question (1.0 points).
En utilisant les mutateurs/accesseurs de matrice, codez un accesseur pour les composantes de θ *(qui prend en entrée l'indice du coefficient) et un mutateur pour pour les composantes de* b *(qui prend en entrée l'indice du coefficient et la valeur).*

Question (1.0 points).
En utilisant les mutateurs/accesseurs de matrice, codez un mutateur pour les composantes de A. *Il prend en entrée* i, j *ainsi que les* `valrray<K>` x_i, x_j *et construit*

$$A_{ij} = \mathrm{kernel}(x_i, x_j)$$

Question (2.0 points).
En utilisant les fonction de la classe matrice, écrire une méthode qui calcule le gradient de J *donné par :*

$$\nabla_\theta J(\theta) = (AA^t)\theta - Ab + \nabla_\theta R(\theta)$$

avec $\nabla_\theta R(\theta)$ *donnée par* `derivate_reg`. *Le gradient de* J *est donc une matrice de taille* $n \times 1$.

Question (1.0 points).
Ecrire une fonction `descente_gradient` *qui prend en entrée un entier* m *et un paramètre* λ *de type de double. Cette fonction contiendra une boucle avec* m *itérations. A chaque itération, on calculera le gradient de* J *puis on mettra à jour la matrice* θ *:*

$$\theta \leftarrow \theta - \lambda \nabla_\theta J(\theta)$$

On utilisera le $*$ *de la classe matrice (attention au sens de la multiplication).*

Exercice 4 (**Régression** (3.0 points)). *Complétez le main.cpp en :*
— *choisissant une valeur pour le paramètre de template* K,
— *créant une régularisation ridge (taille* n*) avec comme paramètre* $\alpha = 0.1$,
— *créant une régression (taille* n, d*) en donnant comme paramètre la fonction* `gaussian` *et la régularisation créée précédemment,*
— *Remplissant A et b de la régression avec les accesseurs et x et y (donnés dans le main.cpp)*
— *appelant la fonction qui effectue la descente de gradient,*
— *affichant* θ.