Année 2021-2022
Emmanuel Franck Mathématiques Appliquées L3

Informatique S6 - CC2

Exercice 1 (**Gradient** (2.0 points))**.**
On va construire une suite de classe décrivant les méthodes de gradients.

On se donne un cas simple qui peut ressembler à celui des réseaux de neurones en apprentissage profond. On considère la fonction :

$$f(x) = \phi(wx + b)$$

avec ϕ *une fonction,* $w, b \in \mathbb{R}$ *de points. On souhaite ici trouver* w *et* b *qui minimise :*

$$J(w, b) = \sum_{i=1}^{n} f(x_i).$$

Ajoutez à la classe ***gradient_method*** *le constructeur par copie.*

Exercice 2 (**Simple Gradient** (7.0 points))**.**
On va maintenant écrire une classe qui correspond à la méthode de gradient simple qui consiste à itérer, sur les poids à déterminer, avec :

$$w_{t+1} = w_t - \eta \nabla_w J(w_t, b_t) \tag{1}$$

et

$$b_{t+1} = b_t - \eta \nabla_b J(w_t, b_t) \tag{2}$$

Codez une classe gradient simple.

Question (2.0 points)**.** *Creez une classe héritée de la classe précédente. Elle aura comme attribut* η*. Codez le constructeur par défaut, le constructeur par copie et un destructeur. Utilisez les constructeurs de la classe mère.*

Question (1.5 points)**.** *Codez un constructeur qui prend* n*, les fonctions* `f` *et* `df` *(attributs de la classe mère) et* η*. Il devra initialiser* η *et les autres attributs à partir du constructeur de la classe mère.*

Question (2.5 points)**.** *Codez la méthode "solve_gradient" qui prend un entier* n *et qui rèpete* n *fois l'itéré de gradient (1) et (2). A chaque itération on calcul le gradient et on update les poids* w *et* b*.*

Exercice 3 (**Gradient avec moment** (4.5 points))**.**
On va maintenant écrire une classe qui correspond à la méthode de gradient avec moment qui consiste à itérer, sur les poids à déterminer, avec :

$$v_t^w = \alpha w_t + (1 - \alpha) \nabla_w J(w_t, b_t) \tag{3}$$

$$w_{t+1} = w_t - \eta v_t^w \tag{4}$$

et

$$v_t^b = \alpha b_t + (1-\alpha)\nabla_b J(w_t, b_t) \tag{5}$$

$$b_{t+1} = b_t - \eta v_t^b \tag{6}$$

Codez une classe gradient avec moment. ***N'hésitez pas à recopier des choses de l'exo précédent, la structure est la même****.*

Question (1.5 points)**.** *Creez une classe héritée de la classe précédente. Elle aura comme attributs* η *et* α*. Codez le constructeur par défaut, le constructeur par copie et un destructeur. Utilisez les constructeurs de la classe mère.*

Question (1.0 points)**.** *Codez un constructeur qui prend* n*, les fonctions* f *et* df *(attribut de la classe mère),* α *et* η*. Il devra initialiser* η *et* α *et les autres attributs à partir du constructeur de la classe mère.*

Question (2.0 points)**.** *Codez la méthode "solve_gradient" qui prend un entier* n *et qui rèpete* n *fois l'itéré de gradient (3)-(5) et (4)-(6). On initilise les* v_t^w *et* v_t^b *avec les gradients.*

Exercice 4 (**Adagrad** (4.0 points))**.**
On va maintenant écrire une classe qui correspond à la méthode adagrad qui consiste à itérer, sur les poids a déterminer, avec :

$$s_{t+1} = s_t + \nabla_w J(w_t, b_t) * *2.0 \tag{7}$$

$$w_{t+1} = w_t - \frac{\eta}{\epsilon + s_{t+1}} \nabla_w J(w_t, b_t) \tag{8}$$

et

$$r_{t+1} = r_t + \nabla_b J(w_t, b_t) * *2.0 \tag{9}$$

$$b_{t+1} = b_t - \frac{\eta}{\epsilon + r_{t+1}} \nabla_b J(w_t, b_t) \tag{10}$$

Codez une classe adagrad. ***N'hésitez pas à recopier des choses de l'exo précédent la structure est la même****.*

Question (1.0 points)**.** *Creez une classe héritée de la classe précédente. Elle aura comme attributs* η *et* ϵ*. Codez le constructeur par défaut, le constructeur par copie et un destructeur. Utilisez les constructeurs de la classe mère.*

Question (0.75 points)**.** *Codez un constructeur qui prend* n*, les fonctions* f *et* df *(attribut de la classe mère),* ϵ *et* η*. Il devra initialiser* η *et* ϵ *et les autres attributs à partir du constructeur de la classe mère.*

Question (2.25 points)**.** *Codez la méthode "solve_gradient" qui prend un entier* n *et qui rèpete* n *fois l'itéré de gradient (7)-(9) et (8)-(10).*

Exercice 5 (**Template** (4.0 points))**.**
On regarde les suites :

$$u_{n+1} = f(u_n)$$

avec U_n *de type* T*.*

Question (2 points)**.** *Ecrivez un template de classe de paramètre : une classe T avec les attributs :*

- $u0$ *et* un *de type* T
- f *un pointeur de fonction qui prend un objet de type T et renvoit un objet de type T.*

Faire les constructeurs par défaut, copie et prenant en paramètre $u0$ *et* f*.*

Question (2 points)**.** *Faire une méthode dont le corps est écrit en dehors de la classe qui prend un entier* n *et calcul le résultat de la suite après* n *iterations.*