Examen ING3-GEE EILCO - Intelligence Artificielle

Automne 2023

Nom :

Prénom :

Total: 27 points
Durée: 2h
Instructions générales: L'examen comprend 2 parties (Chacune de ces parties reprenant différentes sous-questions). Vous êtes libres de rédiger vos réponses sur des pages supplémentaires en veillant toutefois à bien indiquer le numéro de chaque question. Une fois l'examen terminé, Assurez vous de bien écrire votre nom (de façon lisible) sur chacune des pages. Répondez à un maximum de questions, en commençant par les questions qui vous semblent les plus abordables.

Partie 1 (15pts)

1. [5*pts*] *Indiquer si les affirmations suivantes sont vraies ou fausses*

   Vrai / Faux *Les techniques de régularisation sont principalement appliquées aux modèles de régression logistique et ne sont pas pertinentes pour la régression linéaire*

   Vrai / Faux *L'objectif de la régularisation en régression linéaire est de prévenir le surajustement en pénalisant les modèles excessivement complexes avec des coefficients importants*

   Vrai / Faux *La régression Ridge peut être utilisée même quand le nombre d'observations est inférieur au nombre de caractéristiques, contrairement au modèle de régression linéaire standard*

   Vrai / Faux *La fonction de vraissemblance correspondant au modèle linéaire pour un ensemble de n données* $(\boldsymbol{x}^{(i)}, t^{(i)})$ *et un a priori de type Laplace peut sécrire*

   $$p\left(\left\{t^{(i)}\right\}_{i=1}^{n} \mid \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{n}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t^{(i)} - \boldsymbol{\beta}^T \boldsymbol{x}^{(i)})^2}{2\sigma^2}\right) \prod_{j=1}^{D} \exp\left(-\frac{|\beta_j|^2}{2\lambda}\right)$$

   Vrai / Faux *Le modèle de régression Ridge ajoute un terme de pénalité à la fonction de coût de la régression linéaire basé sur les valeurs absolues des coefficients, encourageant la parcimonie dans le modèle*

   Vrai / Faux *La régression Ridge correspond à supposer une distribution Gaussienne pour les résidus*

   Vrai / Faux *La régression Lasso est efficace pour la sélection de caractéristiques car elle a tendance à annuler exactement certains coefficients, excluant ainsi ces caractéristiques du modèle*

2. [7*pts*] *On considère le réseau de neurones représenté à la figure 1. On supposera que toutes les fonctions d'activation (indiquées par la lettre 'σ' sur le diagramme) sont données par la fonction sigmoïde. Bien qu'ils ne soient pas explicitement indiqués, on supposera également que chaque neurone est muni d'un biais, i.e. la sortie de chaque neurone s'écrira donc* $\sigma\left(\sum_j w_{ij}^{(\ell)} z_j^{(\ell-1)} + w_{i0}^{(\ell)}\right)$

   (a) [*1pt*] *Représenter la fonction sigmoïde*

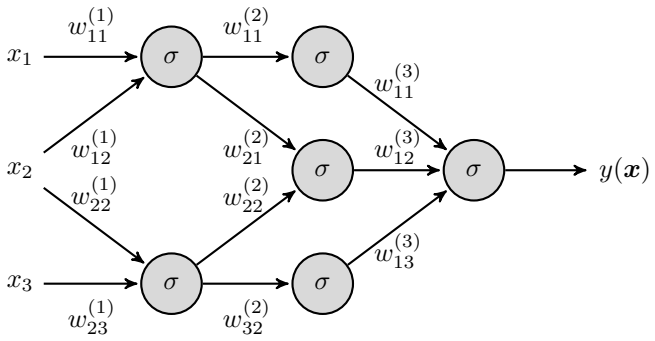

Figure 1: Réseau de neurones utilisé pour la question 1.2

(b) [*2pts*] *Donner l'expression détaillée de la fonction* $y(\boldsymbol{x})$ *correspondant à la sortie du réseau*

(c) [*1pt*] *Donner l'expression de l'entropie binaire croisée* $L(y, t^{(i)})$ *pour une donnée d'entrainement* $\left\{\boldsymbol{x}^{(i)}, t^{(i)}\right\}$.

(d) [*3pts*] *On souhaite utiliser l'algorithme de backpropagation pour calculer le gradient par rapport au poids* $w_{11}^{(1)}$, $\frac{\partial L}{\partial w_{11}^{(1)}}$. *Détaillez votre raisonnement en veillant bien à développer chaque étape.*

3. [*3pts*] *On considère un jeu de données comprenant n vecteurs caractéristiques* $\boldsymbol{x}^{(i)} \in \mathbb{R}^D$ *répartis en K classes. Donner le pseudo-code permettant d'entrainer un modèle de classification linéaire de type "un contre tous".*

Partie 2 (12pts)

1. [5*pts*] *Indiquer si les affirmations suivantes sont vraies ou fausses*

Vrai / Faux	*En classification linéaire, la frontière de décision est un hyperplan qui sépare l'espace des caractéristiques en régions correspondant aux différentes classes*
Vrai / Faux	*En classification linéaire, un déséquilibre entre les classes n'a aucun impact sur la performance du modèle de régression logistique*
Vrai / Faux	*En classification linéaire, un déséquilibre entre les classes n'a aucun impact sur la performance du modèle entrainé en minimisant la somme des carrés des résidus*
Vrai / Faux	*Le modèle de régression logistique ne peut pas être utilisé pour des classes qui ne sont pas linéairement séparables.*
Vrai / Faux	*La fonction de log-vraissemblance pour le modèle de régression logistique à deux classes correspond à la fonction d'entropie binaire croisée*
Vrai / Faux	*La fonction d'activation dans un réseau neuronal introduit une non-linéarité, permettant au réseau d'apprendre des motifs complexes*
Vrai / Faux	*L'initialisation des poids dans un réseau neuronal peut avoir un impact significatif sur la convergence et la performance du modèle*

2. [*3pts*] *On considère un ensemble de donnée* $\left\{t^{(i)}, \boldsymbol{x}^{(i)}\right\}_{i=1}^{n}$ *générées via le modèle suivant*

$$t^{(i)} = \beta_0 + \beta_1 x^{(i)} + \varepsilon^{(i)} \tag{1}$$

$\varepsilon^{(i)}$ *représente un bruit Gaussien. En utilisant le fait que pour une matrice de taille* 2×2,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{2}$$

Donner la solution (β_0, β_1) *en fonction des* $x^{(i)}$ *et* $t^{(i)}$ *(Indice: utiliser les équations normales).*

3. [*4pts*] *Vous disposez de différents extraits de code qui ont été endommagés lors de leur transmission.*

   (a) [*2pts*] *On commence par considérer l'extrait donné à la figure 2. On souhaite compléter les parties manquantes de l'extrait de façon à obtenir le résultat donné à la figure 3. Compléter le tableau 1 avec les lignes manquantes*

   (b) [*2pts*] *On souhaite maintenant compléter un second extrait de code, de façon à obtenir le résultat donné à la figure 5. Compléter le tableau 2 en fournissant les lignes manquantes (les lignes marquées [*] correspondent aux lignes fournies à la question précédente).*

[1]	
[2]	
[3]	
[4]	

Table 1: À compléter avec les lignes manquantes du code 1 (Figure 2)

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import ??????? [1]

xtrain = np.linspace(0,1,10)

beta0 = 1
beta1 = 1

t = beta0 + beta1*xtrain
tnoisy = t +np.random.normal(0,.1, len(xtrain))


model = ????????[2]

model.??????[3](x.reshape(-1,1), tnoisy)

xtest = np.linspace(0,1,100)

prediction = model.???????[4](xtest.reshape(-1,1))

plt.plot(xtest, prediction)
plt.scatter(xtrain, tnoisy, c= 'r')
plt.show()
```

Figure 2: Extrait 1

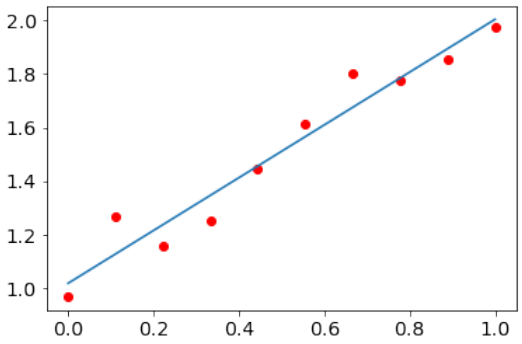

Figure 3: Résultat désiré pour le code donné à la Figure 2

```python
import numpy as np
import matplotlib.pyplot as plt

xtrain = np.linspace(-1,1,10)

t = .1 + .1*xtrain + xtrain**2
tnoisy = t +np.random.normal(0,.1, len(xtrain))

from sklearn.preprocessing import ???????? [1]

poly = ????????? [2]
Xtilde = poly.????????? [3]

from sklearn.linear_model import ??????? [*]

model = ???????? [*]

model.???????? [*](Xtilde, tnoisy)

xtest = np.linspace(-1,1,100)

Xtilde_test = poly.????????? [4]

prediction = model.???????[*](Xtilde_test)
```

Figure 4: Extrait 2

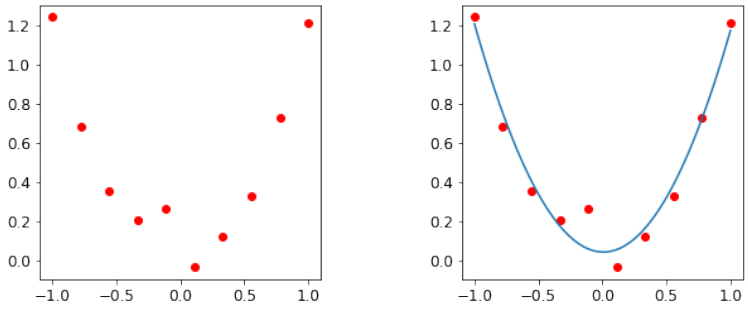

Figure 5: Résultat désiré pour le code donné à la Figure 4

[1]	
[2]	
[3]	
[4]	

Table 2: À compléter avec les lignes manquantes du code 2 (Figure 4)