Examen ING2 EILCO - Ingénierie Mathématique

January 12, 2026

Nom :

Prénom :

Total: 46 points
Durée: 2h
Consignes: Pas de calculatrice autorisée, aucun document autorisé
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 (23pts)

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

Vrai / Faux	*La dérivée de la fonction sigmoïde est donnée par* $\sigma'(x) = \sigma(x)(\sigma(x) - 1)$
Vrai / Faux	*Soit un ensemble de données défini par des vecteurs caractéristiques* $\boldsymbol{x}$ *comportant 5 caractéristiques et stockés dans la matrice* $\boldsymbol{X}$ *de taille* $N \times 5$*, la ligne* PolynomialFeatures.fit_transform(X, degree=2) *produira une matrice de taille* $N \times 15$
Vrai / Faux	*L'algorithme de rétropropagation pour l'entrainement des réseaux de neurones ne fonctionne que si toutes les fonctions d'activations sont identiques*
Vrai / Faux	*Un modèle entrainé via la minimisation d'une formulation de type Lasso contiendra toujours au moins un coefficient nul*
Vrai / Faux	*En apprentissage supervisé, certaines données d'entrainement ne sont pas annotées.*
Vrai / Faux	*Les pénalités utilisées par les regularisations de type Ridge et Lasso sont deux cas particuliers de normes* ℓ_p
Vrai / Faux	*Dans le cadre de la régression linéaire, l'estimateur de maximum a posteriori peut sécrire* $\beta_{\text{MAP}} = \underset{\beta}{\text{argmax}}\, P(\beta\|t)P(\beta)$

2. [5*pts*] *Pour les questions suivantes, déterminer la ou les réponses correctes:*

   1) *En scikit-learn, quelle fonction est utilisée pour entrainer un modèle de régression linéaire?*
      - *A.* sklearn.linear_model.SimpleLinearRegression()
      - *B.* sklearn.linear_model.OLSRegression()
      - *C.* sklearn.linear_model.LinearRegression()
      - *D.* sklearn.linear_model.Linear_Regression()
      - *E.* sklearn.linear_model.Regression()

   2) *Parmi les propositions suivantes, quelles sont les lignes permettant d'importer correctement la librairie* pandas *et d'enregistrer le fichier* data/ram_price.csv *dans une variable?*

A. import panda as pd
ram_prices = pd.readcsv('data/ram_price.csv')

B. import pandas as pd
ram_prices = pandas.read_csv['data/ram_price.csv']

C. import pandas as pd
ram_prices = pd.read_csv('data/ram_price.csv')

D. import pandas pd
ram_prices = pd.read_csv('data/ram_price.csv')

3) Parmi les propositions suivantes, quelle est la ligne qui va produire le modèle avec le plus grand nombre de coefficients nuls?

A. model = Ridge(alpha=1000)

B. model = Lasso(alpha=0.01, max_iter=5000)

C. model = Lasso(alpha=0.0001)

D. model = Lasso(alpha=50)

4) On entraine un modèle linéaire (représenté par la variable clf*) via la minimisation d'une formulation de type* Ridge *. On souhaite afficher les régions associées à chacune des classes sur une grille de coordonnées* xx *et* yy *générée en utilisation* meshgrid*. Parmi les propositions suivantes, quelles sont celles qui permettent de calculer la prédiction du modèle sur la grille et de représenter le résultat via* contourf*?*

A. plt.contourf(xx, yy, clf.predict(np.hstack((xx.flatten(), yy.flatten())).reshape(xx.shape)))

B. plt.contourf(xx, yy, clf.predict(np.hstack((xx.flatten(), yy.flatten())).T).reshape(xx.shape))

C. plt.contourf(xx, yy, clf.predict(np.vstack((xx.flatten(), yy.flatten())).T).reshape(xx.shape))

D. plt.contourf(xx, yy, clf.predict(np.c_[xx,yy]), alpha=0.5)

E. plt.contourf(xx, yy, clf.predict(np.hstack([xx.flatten(), yy.flatten()])).reshape(xx.shape), alpha=0.5)

5) Parmi les propositions suivantes, quelles sont celles qui décrivent correctement le fonctionnement d'un objet de la classe PolynomialFeatures *en* scikit-learn*?*

A. L'objet PolynomialFeatures *peut être utilisé directement via la ligne* PolynomialFeatures.transform(X)

B. Les lignes PolynomialFeatures(order=2) *et* PolynomialFeatures(degree=2) *sont toutes les deux valides.*

C. PolynomialFeatures *génère de nouvelles caractéristiques consistant en des produits des caractéristiques d'origine jusqu'a un certain degré.*

D. La ligne PolynomialFeatures.transform(X) retire automatiquement les caractéristiques colinéaires afin d'eviter la redondance dans les modèles polynomiaux.

E. PolynomialFeatures *ne peut prendre que des vecteurs (i.e tableaux 1D) en entrée.*

6) On souhaite entrainer un modèle de classification de type "un contre un" pour K classes en utilisant le modèle LinearRegression() *de* Scikit-learn*. Parmi les propositions suivantes, indiquer quelles sont les lignes permettant d'itérer correctement sur les paires de classes (sans tenir compte de l'ordre) en entrainant un modèle linéaire pour chaque paire.*

A.
```
for i in range(K):
  for j in range(i+1, K):
    LinearClassifier().fit(X[y==i], y[y==i])
```

B.
```
for i in range(K):
  for j in range(i, K):
    LinearClassifier().fit(X[(y==i)|(y==j)], y[(y==i)|(y==j)])
```

C.
```
for i in range(K):
  for j in range(i+1, K):
    LinearClassifier().fit(X[(y==i)|(y==j)], y[(y==i)|(y==j)])
```

D.
```
for i in range(K):
  for j in range(i+1, K):
    LinearRegression().fit(X[(y==i)|(y==j)], y[(y==i)|(y==j)])
```

7) Parmi les propositions suivantes, quelles sont celles qui permettent d'instantier un réseau de neurones à 3 couches contenant chacune 5 neurones en Scikit-learn*?*

A. clf = MLPClassifier(hidden_layer_sizes=(3, 5), solver="adam")
B. clf = MLPClassifier(hidden_layer_sizes=(5,5,5), solver="adam")
C. clf = MLPClassifier(hidden_layers=(5, 5,5), solver="sgd")
D. clf = MLPClassifier(hidden_layers=(5, 5,5), solver="rprop")
E. clf = MLPClassifier(hidden_layer_sizes=[3, 5], solver="lbfgs")

3. [8*pts*] *On considère le modèle linéaire* $y(\boldsymbol{x}) = \boldsymbol{\beta}^\intercal \boldsymbol{x} + \varepsilon = \beta_1 x_1 + \beta_2 x_2 + \varepsilon$ *(dont le biais* β_0 *est égal à* 0*). Pour ce modèle on se donne les données suivantes:*

$\boldsymbol{x}^{(i)}$	$t^{(i)}$
$(1, 1)$	2
$(1, -1)$	0

(1)

   (a) [2*pts*] *Rappeler la forme générale de la solution des équations normales pour la fonction de coût donnée par la somme des carrés des résidus ainsi que pour la régularisation de type Ridge (sans substituer les données)*

   (b) [2*pts*] *En utilisant les formulations dérivées au point précédent, calculer l'estimateur* $\hat{\beta}_{OLS}$ *(on pourra utiliser le fait que l'inverse d'une matrice* 2×2 *est donnée par* $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad-bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$*)*

   (c) [2*pts*] *Calculer l'estimateur* $\hat{\beta}_{\text{Ridge}}$ *pour un* λ *quelconque.*

   (d) [1*pt*] *Calculer l'estimateur* $\hat{\beta}^{\text{lasso}}$ *pour un* λ *donné en utilisant le fait que*

$$\hat{\beta}_j^{\text{lasso}} = \text{sign}(z_j) \max\left(|z_j| - \lambda, 0\right), \quad \boldsymbol{z} = \beta_{OLS} \tag{2}$$

   où $\boldsymbol{X}$ *est la matrice caractéristique (utilisée aux points précédents) et* $\boldsymbol{t}$ *est le vecteur cible.*

   (e) [1*pt*] *Déterminer les valeurs de* λ *pour lesquelles* $\hat{\beta}_1^{\text{lasso}} = 0$ *d'une part et* $\hat{\beta}_2^{\text{lasso}} = 0$ *d'autre part.*

4. [3*pts*] *On considère un ensemble de données d'entrainement* $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}, t^{(i)}\right\}_{i=1}^{N}$ *pour lesquelles la relation entre la caractéristique* x *et la valeur cible* t *est donnée par une fonction* $t = f(x) + \varepsilon$*,* $\varepsilon \sim \mathcal{N}(0, \sigma^2)$*. On utilise* $\mathcal{D}_i$ *pour représenter un sous-ensemble de données, i.e.* $\mathcal{D}_i \subset \mathcal{D}$ *et on considère une famille de modèles paramétrés par* β *et notés* $h_\beta : x \mapsto h_\beta(x)$*.*

   (a) [1*pt*] *Donner l'expression générale du biais pour la famille de modèles* h_β

   (b) [1*pt*] *Donner l'expression générale de la variance.*

   (c) [1*pt*] *Représenter, à la figure 1, la partie de la courbe pour laquelle le biais domine la variance ainsi que la partie de la courbe pour laquelle la variance domine le biais.*

5. [2*pts*] *Donner l'expression de la fonction objectif dans le cas d'une régularisation de type Ridge.*

Partie 2 (21pts)

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

   Vrai / Faux *Soit un réseau de neurones combiné à une pénalité de type Ridge (de paramètre λ). Lorsque $\lambda \to \infty$, les poids du réseau tendent vers 0, et le réseau est équivalent à une fonction constante.*

   Vrai / Faux *Dans l'interprétation statistique de la régularisation Ridge, le paramètre λ de régularisation est inversément proportionnel à la variance de l'apriori Gaussien*

   Vrai / Faux *Dans les équations de rétro-propagation, le gradient de la fonction de coût par rapport à la préactivation de la couche ℓ peut être obtenu à partir du gradient de la couche $\ell - 1$ via l'expression* $\delta_i^{(\ell)} = \sum_{j=1}^{N_\ell} \delta_j^{(\ell-1)} w_{ji}^{(\ell)} \sigma'(a_i^{(\ell)})$ *où* $\delta_i^{(\ell)} = \dfrac{\partial L}{\partial a_i^{(\ell)}}$ *et L est la fonction de coût.*

   Vrai / Faux *Dans la validation croisée à K compartiments, si une donnée est utilisée pour l'entrainement d'un modèle, alors elle n'est pas utilisée pour la validation de ce modèle.*

   Vrai / Faux *Ajouter $\lambda \boldsymbol{I}$ à la matrice $\boldsymbol{X}^\intercal \boldsymbol{X}$ modifie les vecteurs propres de la matrice*

   Vrai / Faux *Dans la validation croisée à K compartiments, toutes les données sont utilisées pour l'entrainement et la validation.*

   Vrai / Faux *Une formulation de type Lasso peut être minimisée via une descente de gradient.*

   Vrai / Faux *Dans l'interprétation statistique des modèles de type Ridge et Lasso, les coefficients β_j sont supposés indépendants.*

2. [8*pts*] *On dispose d'un jeu de données binaires*

$$\boldsymbol{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} \\ \vdots & \vdots \\ x_1^{(N)} & x_2^{(N)} \end{bmatrix}, \quad \boldsymbol{t} = \begin{bmatrix} t^{(1)} \\ \vdots \\ t^{(N)} \end{bmatrix}, \quad t^{(i)} \in \{0, 1\} \tag{3}$$

   On souhaite implémenter un modèle de classification linéaire <u>entièrement</u> en numpy *et représenter le résultat via la fonction* meshgrid

   (a) [4*pts*] *Compléter la fonction* descente_gradient *ci dessous afin qu'elle retourne les coefficients d'un modèle linéaire permettant de séparer les données de chacune des deux classes (on supposera que les deux classes sont linéairement séparables).*

```
def descente_gradient(beta_init, max_iter, lr, X, t):
    #beta_init: estimé initial
    #max_iter: nombre max. d'itérations
    #lr: taux d'apprentissage
    # X, t: Matrice caractéristique et vecteur cible









    return beta_optimal
```

(b) [2*pts*] *On souhaite à présent représenter la frontière de décision du modèle sur la grille. Donner le code python permettant de générer une grille de points équidistants de taille* 100×100 *et telle que toutes les données d'entrainement appartiennent à la zone couverte par la grille.*

(c) [2*pts*] *Finalement, en utilisant les données de la grille ainsi que la fonction* matmul() *de* numpy*, compléter le code ci-dessous afin qu'il affiche la frontière de décision du modèle entrainé au point 1.*

```
prediction = _____

plt.contourf(xx, yy,_____,alpha =.2)
plt.scatter(_____, _____, c='r')
plt.scatter(_____, _____, c='b')
plt.show()
```

3. [5*pts*] *On considère le problème de classification représenté à la figure 2. On souhaite entrainer un réseau de neurones de façon à séparer la classe rouge (triangles) de la classe bleue (disques).*

(a) [3*pts*] *Donner l'expression (coefficients de régression, biais et fonctions d'activations) de chacun des neurones du réseau représenté à la figure 3 de façon à ce que ce réseau permette de séparer les deux classes.*

(b) [2*pts*] *Représenter la frontière de décision de chacun des neurones de la première couche sur la figure 2*

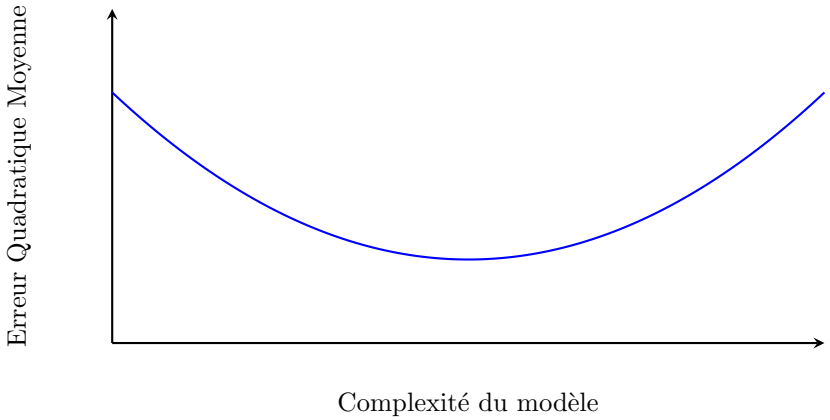

Figure 1: Erreur quadratique moyenne pour la question 1.4

4. [3*pts*] *Donner l'expression de la fonction d'entropie binaire croisée.*

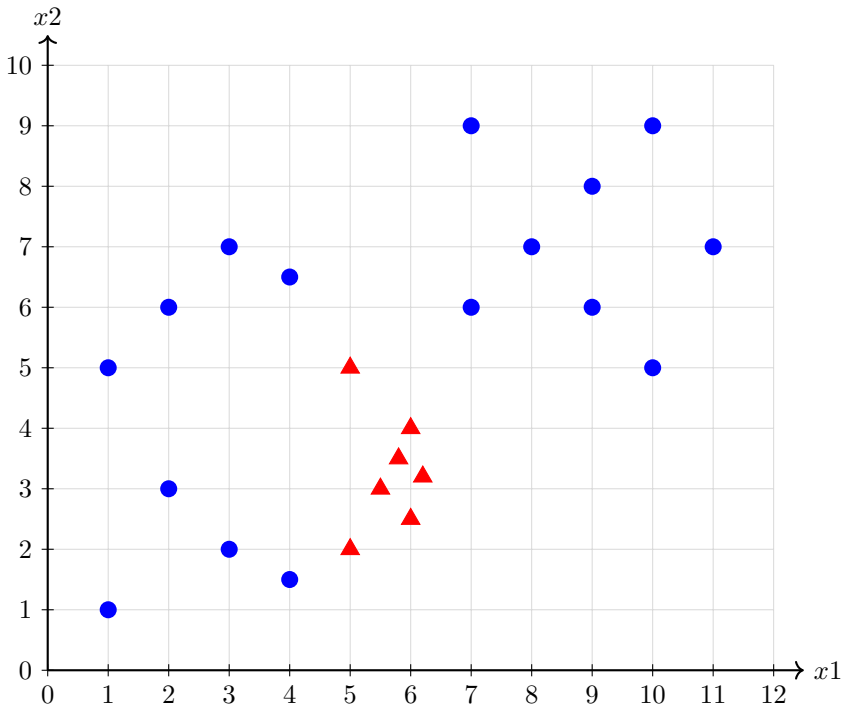

Figure 2: Jeu de données utilisé à la question 3

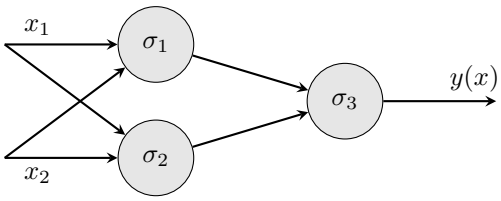

Figure 3: Schéma de réseau utilisé à la question 3.