Projet du module "Algorithmique pour IA"

Master 1 Data Science, UIE, Bamako

22 février 2025

Informations & Consignes sur l'évaluation

- **Type d'évaluation :** Projet
- **Année universitaire :** 2024 - 2025
- **Date de rendu :** Vendredi 28 Mars 2025
- **Format de rendu :** un **notebook Python** pour la partie pratique ; des **copies manuscrites** ou le PDF résultant d'un document **LaTeX** pour la partie théorique.

  NB : la partie théorique peut être aussi traitée directement dans le notebook (Markdown/LaTeX).
- **Consignes particulières :**
  - **Projet à faire en trinôme** (indiquez les noms des membres du trinôme, suivis de M1_DS_UIE dans les noms de vos documents de rendu et dans l'objet de votre mail) **sans plagiat entre trinômes ou ailleurs**, ni sur internet non plus (**citez vos sources**) et sans copier-coller de ChatGPT ou un autre ChatLLM (ce sont des outils utiles, sachez les utiliser ... sans oublier de les citer également en cas d'utilisation).
- **Enseignant :** Dr DIABATE Modibo (modibo.diabate.pro@gmail.com)

Partie I : Machine Learning (Questions de cours pratiques)

Considérons l'ensemble de données sur une maladie cardiaque, `heart_disease.csv` (téléchargé depuis Kaggle), contenant des données médicales sur $n = 303$ patients. Comme le montre la Figure 1, `heart_disease.csv` contient 303 lignes (exemples) et 14 colonnes (variables) qui sont :

— `age` : âge du patient
— `sex` : sexe du patient (0 : féminin, 1 : masculin)
— `cp` : type de douleur angineuse (1 : typique, 2 : atypique, 3 : non angineuse, 4 : asymptomatique)
— `trestbps` : tension artérielle au repos (en mm Hg)
— `chol` : quantité de cholestérol (en mg/dl) récupérée via le capteur IMC
— `fbs` : variable binaire regardant si la glycémie à jeun est > 120 mg/dl (1 : vrai, 0 : faux)
— `restecg` : résultats électrocardiographiques (ECG) au repos
— `thalach` : fréquence cardiaque maximale atteinte (en battement par minute (bpm))
— `exang` : angine induite par l'effort (1 : oui, 0 : non)
— `oldpeak` : dépression ST (anomalie dans un ECG) induite par l'exercice par rapport à l'état de repos
— `slope` : pente du segment ST (partie d'un ECG) d'effort maximal
— `ca` : variable catégorielle sur le nombre de grands vaisseaux
— `thal` : variable catégorielle sur le type de défaut
— `target` : variable binaire sur la maladie du coeur (0 : pas de maladie, 1 : maladie).

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

Figure 1 – Aperçu sur `heart_disease.csv`

Partie I/A : Prédiction de la fréquence cardiaque maximale

Dans cette première partie, nous nous concentrons sur l'étude de la fréquence cardiaque maximale atteinte, la variable `thalach` dans le jeu de données (voir Figure 1).

1. Questions préliminaires
   (a) Quel est le type de la variable `thalach` ? Même question pour `age`, `sex` et `chol`.
   (b) Selon vous, comment peut-on vérifier l'existence d'un lien et la force de ce lien entre la variable `chol` et `thalach` ?
2. A présent, on souhaite mettre en place un modèle qui nous permettra de prédire la fréquence cardiaque maximale (`thalach`) d'un patient à partir de différentes variables identifiées comme ayant un effet significatif sur `thalach`. Dans la suite, on considérera que ces variables identifiées sont: `age`, `sex`, `trestbps`, `chol`, `oldpeak` et `thal` (on les notera dans la suite respectivement par $x^{[1]}, x^{[2]}, x^{[3]}, x^{[4]}, x^{[5]}, x^{[6]}$). Un Data scientist nous suggère un modèle de régression linéaire pour ce problème de prédiction.
   (a) Est-ce que la régression linéaire est réellement le modèle adapté pour ce problème ? Justifiez votre réponse.
   (b) Identifiez la variable étiquette de ce problème. On la notera par y dans la suite.
   (c) Écrivez l'expression mathématique du modèle de régression pour le patient i (avec $i \in \{0, \ldots, n = 302\}$) ayant l'étiquette y_i et les caractéristiques $x_i^{[1]}, \ldots, x_i^{[6]}$, sans oublier le terme d'erreur (résidu) ϵ_i. On note $\beta_0, \beta_1, \ldots, \beta_6$ les paramètres du modèle.
   (d) Afin de pouvoir utiliser notre modèle de régression pour des fins de prédiction, nous devons d'abord l'entraîner.
      i. Quel est le but de l'entrainement selon vous ? En quoi cela consiste concrètement ?
      ii. Quelle fonction perte considérez vous pour ce problème ? Justifiez votre choix.
      iii. Quel algorithme d'optimisation suggérez vous pour l'entrainement de notre modèle ? Comment fonctionne cet algorithme ?
      iv. Quelles peuvent être, selon vous, les limites de cet algorithme d'optimisation ? Connaissez-vous une autre alternative ?

(e) Notre Data scientist nous suggère de séparer nos données en données d'entrainement et données de test avant de procéder à l'entrainement de notre modèle. Expliquez en quoi cette séparation des données est importante et à quoi servent les données d'entrainement et à quoi servent les données de test.

(f) Après l'entraînement, vous devez communiquer à l'hôpital le pouvoir prédictif de votre modèle. Quel indicateur (exprimant la qualité d'un modèle de régression) choisissez-vous et sur quelles données vous le calculez ? Pourquoi ?

(g) L'hôpital vous demande de prédire la fréquence cardiaque maximale d'un nouveau patient $i = 317$ absent des données initiales (Figure 1). Comment procédez-vous (en utilisant votre modèle entrainé dont les paramètres estimés sont notés $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_6$) ?

Partie I/B : Prédiction de la présence de la maladie cardiaque

Dans cette deuxième partie, nous nous concentrons sur la prédiction de la survenue de la maladie du coeur, la variable `target` dans le jeu de données (voir Figure 1).

3. Notre Data scientist nous suggère ici d'utiliser un modèle de classification pour prédire si un nouveau patient est malade du coeur ou non.

   (a) Êtes vous d'accord avec lui ? Pourquoi ?

   (b) Ce problème fait partie de quelle classe de problèmes de Machine Learning ? Justifiez.

   On considère dans la suite qu'on a identifié les variables suivantes comme les caractéristiques pertinentes de notre problème: `age`*,* `sex`*,* `cp`*,* `chol`*,* `thalach`*,* `oldpeak`*,* `thal`*.*

4. Suggérez un algorithme de classification pour ce problème de prédiction de la variable `target`. Pourquoi cet algorithme ? Décrivez brièvement le principe de cet algorithme.

5. Soit $y_{\text{subset}} = (1, 1, 0, 0, 1, 0, 1, 0, 0, 1)$ et $\hat{y}_{\text{subset}} = (1, 0, 1, 0, 1, 1, 1, 1, 0, 1)$ respectivement, une partie des vraies étiquettes et des étiquettes prédites (par votre algorithme de classification).

   (a) Proposez un pseudo code (décrivez la procédure algorithmique) permettant de calculer rapidement la précision ("accuracy") de votre modèle en partant de y_{subset} et $\hat{y}_{\text{subset}}$.

   (b) Reproduisez et Complétez la matrice de confusion ci-dessous à partir des valeurs des vecteurs y_{subset} et $\hat{y}_{\text{subset}}$ ci-dessus.

	True 0	True 1
Predicted 0	TN =	FN =
Predicted 1	FP =	TP =

   où TN, FN, FP et TP correspondent respectivement aux nombres de : vrais négatifs, faux négatifs, faux positifs et vrais positifs.

   (c) Déduisez de la matrice de confusion : "l'accuracy", la sensibilité (taux d'individus positifs bien prédits) et la spécificité (taux d'individus négatifs bien prédits) et $1 -$ spécificité (taux d'individus négatifs mal prédits) de votre modèle de classification.

   (d) Suggérez une méthode de comparaison de plusieurs modèles de classification binaires.

Partie II : Théorie et Pratique de l'algorithme de Descente de gradient dans un réseau de neurones dense (DNN) simple

Partie II/A : Théorie de la Descente de gradient dans un DNN simple

Soit un réseau de neurones multi-couches dense (un Perceptron multi-couches ou MLP) à une seule couche cachée ayant h neurones (ou unités) et une couche de sortie de taille q.

- On considère un mini-batch $\mathbf{X} \in \mathbb{R}^{n \times d}$ (n exemples ayant chacun d caractéristiques) pour constituer sa couche d'entrée.
- Pour établir les connexions entre la couche d'entrée et la couche cachée, nous définissons la matrice de poids $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times h}$ et le vecteur de biais $\mathbf{b}^{(1)} \in \mathbb{R}^{1 \times h}$.
- Pour établir les connexions entre la couche cachée et la couche de sortie, nous définissons la matrice de poids $\mathbf{W}^{(2)} \in \mathbb{R}^{h \times q}$ et le vecteur de biais $\mathbf{b}^{(2)} \in \mathbb{R}^{1 \times q}$.

Les calculs ci-dessous entre la couche d'entrée et la couche cachée, puis entre la couche cachée et la couche de sortie ont été faits :

- Pre-activation et activation au niveau de la couche cachée :

  $\mathbf{Z}^{(\text{cachée})} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}$ et $\mathbf{H} = a^{(1)}(\mathbf{Z}^{(\text{cachée})})$, ici $a^{(1)}(z) = \dfrac{1}{1+\mathrm{e}^{-z}}$ (**sigmoïde**)

- Pre-activation et activation au niveau de la couche de sortie :

  $\mathbf{Z}^{(\text{sortie})} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$ et $\mathbf{Y} = a^{(2)}(\mathbf{Z}^{(\text{sortie})})$, ici $a^{(2)}(z) = \dfrac{1}{1+\mathrm{e}^{-z}}$

- **Fonction de perte :**

  $$E(\mathbf{W}, \mathbf{b}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \left(\hat{y}^{(i)} - y^{(i)}\right)^2,$$

  où $\{\mathbf{W}, \mathbf{b}\} = \{\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}\}, \{\mathbf{b}^{(1)}, \mathbf{b}^{(2)}\}\}$.

En vous basant : sur les calculs différentiels ("dérivées") vus dans le cours dans différents algorithmes d'optimisation et sur les recherches que vous êtes invités à faire **(sans plagiat)** :

1. Montrez que la dérivée $a'(z)$ de la fonction sigmoïde est égale à : $a'(z) = a(z)(1 - a(z))$.
2. Décrivez le rôle / effets d'une fonction d'activation et, spécifiquement, celui de la sigmoïde.
3. Calculez les dérivées partielles de la fonction de perte $E(\mathbf{W}, \mathbf{b})$ par rapport aux poids et biais de la couche cachée et de la couche de sortie d'un MLP ayant une seule couche cachée.
4. Déduisez de ces résultats, les formules de mise à jour d'un algorithme de descente de gradient stochastique.
5. Expliquez la différence entre la descente de gradient classique et stochastique.

Partie II/B : Pratique avec un DNN simple (dans un notebook Python)

1. Implémentez *from scratch* (uniquement grâce à vos précédents calculs et le code Python fourni dans le projet sur le neurone mono-couche avec l'activation sigmoïde) un réseau de neurones dense ayant une couche cachée (avec 3 neurones) utilisant l'activation sigmoïde sur la couche cachée et la couche de sortie.
   (a) Combien d'unités (neurones) comporte votre couche d'entrée ?
   (b) Combien d'unités (neurones) comporte votre couche de sortie ?
2. Générez aléatoirement des données binaires non séparables linéairement (configuration **XOR**) comme dans le code fourni sur le neurone mono-couche.
   (a) Expliquez la particularité de ces données (configuration **XOR**). Expliquez aussi, de façon générale, l'utilité des données générées (synthétique) en Data Science.

(b) Représentez graphiquement vos données de sorte qu'on puisse distinguer facilement les deux classes (malades, non malades).

(c) Rappelez les précautions à prendre pour la séparation des données *iid* (indépendantes et identiquement distribuées) en données d'apprentissage et données de test.

(d) Écrivez un code Python *from scratch* (en utilisant, au plus, la librairie *Numpy*) permettant de **bien** séparer vos données en données d'entrainement et données de test.

(e) Pour information, il existe des outils / librairie Python permettant de faire cette séparation des données en données d'entrainement et données de test de façon plus automatisée. Citez en un et expérimentez son utilisation dans la séparation de vos données.

(f) A quoi sert la graine (*seed* en anglais) ? Expérimentez/illustrez son rôle dans votre code de séparation de données.

3. Entrainez, puis testez (évaluez) sur vos données préparées, votre modèle de réseau de neurones mis en place précédemment.
4. En parallèle, utilisez *sklearn* pour entrainer et tester un modèle de régression logistique (en spécifiant l'architecture) sur vos données générées / préparées.
5. Comparez (de façon sensée statistiquement) les résultats de cet algorithme de *sklearn* à ceux de votre algorithme codé *from scratch* ? Commentez.