

MATH60629. Devoir.

Date de remise : 3 novembre 2026

Consignes :

- Veuillez inclure votre nom et votre matricule HEC avec votre remise.
- Le devoir est à remettre au plus tard à 23 h 59 à la date de remise. Veuillez téléverser une version PDF de votre devoir sur ZoneCours.
- Le devoir compte pour 15 % de la note finale du cours.
- Le devoir doit être fait individuellement.

1 Principes de l'apprentissage automatique (3pt)

Vos réponses à ces questions devraient être courtes (au plus 5 lignes).

1. (1pt) Expliquez la différence entre l'erreur d'entraînement et l'erreur de généralisation. Assurez-vous de décrire comment estimer l'erreur de généralisation d'un modèle en pratique, y compris les pièges de cette approche (autrement dit, décrivez les conditions sous lesquelles cette approche est fiable).
2. (1pt) Expliquez l'effet de la variation de la force de régularisation (par exemple avec la régularisation Ridge) sur le compromis biais-variance.
3. (1pt) Un collègue a entraîné un modèle pour prédire le prix médian des maisons à partir de données immobilières de Montréal. Le modèle obtenu fonctionne bien selon ses données de validation. Cependant, ce modèle performe mal lorsqu'il est utilisé pour prédire les prix des maisons sur le marché immobilier de Toronto. Expliquez pourquoi et suggérez une façon d'obtenir un meilleur modèle pour les données de Toronto.

2 Classification (16pt)

- Nous utiliserons un jeu de données synthétique que j'ai créé. Il est disponible [ici](#).

Une fois les données accessibles depuis votre répertoire de travail, vous pouvez les charger avec le code suivant :

```
data = np.load("a24_devoir_q2-classification.npz")
X = data["X"]
y = data["y"]
```

Les données sont encodées dans un *numpy array*.

La tâche consiste à prédire la classe de chaque donnée à partir de ses deux attributs.

1. (4pt) À la suite d'une exploration initiale des données, qu'observez-vous? Que pouvez-vous dire de la performance approximative en test d'un modèle linéaire?
2. (2pt) Divisez votre jeu de données en un ensemble d'entraînement et un ensemble de test. L'ensemble de test doit représenter 20 % du jeu de données original. Assurez-vous d'utiliser ce paramètre lors de l'appel de la fonction sklearn appropriée : `random_state=123`. Nous vous demandons de fournir les quelques lignes de code utilisées pour diviser les données.
3. (1pt) Que vous indique le fait que nous ne conservons pas d'ensemble de validation?

Vous explorerez maintenant le modèle bayésien naïf.

4. (1pt) Quel classifieur bayésien naïf devriez-vous utiliser? Autrement dit, quelle est une distribution raisonnable pour les densités conditionnelles aux classes? Justifiez votre réponse.
5. (2pt) Entraînez le modèle sur les données d'entraînement et obtenez sa performance en entraînement et en test. Veuillez fournir les quelques lignes de code pour ce faire, ainsi que les performances d'entraînement et de test obtenues.
6. (4pt) Ré-entraînez le modèle en utilisant des probabilités a priori uniformes pour les classes (fournissez le code pour ce faire). Que remarquez-vous quant aux performances d'entraînement et de test? Expliquez ce résultat.
7. (2pt) Croyez-vous que cette observation est susceptible de tenir en général?

3 Régression (32pt)

Vous entraînerez maintenant un modèle des k plus proches voisins et un réseau de neurones pour la tâche de prédire la note d'une évaluation écrite (texte).

Les données à télécharger sont [ici](#). Chaque donnée est une évaluation, en format texte, d'un produit Amazon.¹

¹Pour votre information, les jeux de données complets sont disponibles [ici](#). Nous utilisons un sous-ensemble de la catégorie *Instruments*. Notez que les évaluations sont en anglais.

Dans le fichier de données, chaque ligne correspond à une donnée. Chaque donnée contient une cible (y) suivie d'un court texte (x). La variable cible est la note donnée par un utilisateur à un produit. C'est une valeur entière entre 1 et 5. Le texte est l'évaluation. Pour prétraiter les données, vous devrez d'abord séparer les cibles des attributs.

Indice : vous pouvez utiliser la librairie csv pour charger ces données. Par exemple :

```
import csv

with open('instruments.csv', mode='r') as file:
    reader = csv.reader(file)
    for i, row in enumerate(reader):
        print(i, row)
```

1. (2pt) Nous modéliserons cette tâche comme un problème de régression (vous pouvez donc utiliser l'erreur quadratique moyenne pour mesurer la performance). Donnez un avantage et un inconvénient de plutôt modéliser le problème comme un problème de classification à 5 classes.

Prétraitement des données (5pt)

2. (2pt) Divisez d'abord le jeu de données en ensembles d'entraînement (80 % des données), de validation (10 %) et de test (10 %). Pour ce faire, fixez le germe aléatoire à 1234 (`random_state=1234`).

Fournissez les quelques lignes de code utilisées pour diviser les données.

3. (3pt) Vous devez maintenant obtenir une représentation sac de mots (*bag-of-words*) des attributs. sklearn fournit plusieurs [fonctions](#) pour ce faire.

Pour limiter le temps d'entraînement requis, veuillez utiliser un maximum de 2 000 mots dans votre vocabulaire (`max_features=2000`) et la liste de mots outils anglais de sklearn (`stop_words="english"`). Les mots de cette liste seront automatiquement retirés des données puisqu'ils sont, a priori, moins prédictifs pour la tâche. Utilisez la valeur par défaut pour tous les autres paramètres de la fonction.

Fournissez les quelques lignes de code sklearn utilisées pour encoder (et seulement celles-ci) les données d'entraînement, de validation et de test.

K plus proches voisins (13pt)

4. (3pt) Parmi les trois fonctions de distance suivantes, « cosine », « euclidean » et « manhattan », laquelle jugez-vous la plus appropriée pour ce problème? Veuillez justifier.
5. (5pt) Étant donné votre réponse précédente, entraînez un modèle des k plus proches voisins adéquat pour cette tâche. Nous vous demandons d'entraîner des modèles avec 1, 10, 50, 100 et 1000 voisins.

Quelle est la performance de chaque modèle sur les ensembles d'entraînement et de validation?

Fournissez les quelques lignes de code utilisées pour entraîner ce modèle et évaluer sa performance.

6. (5pt) Quelle valeur de l'hyperparamètre donne les meilleurs résultats? Expliquez.

Réseaux de neurones (10pt)

Lors de l'instanciation de vos réseaux de neurones, fixez le germe aléatoire à 1234 (c'est-à-dire `random_state=1234`).

7. (5pt) Vous entraînerez maintenant une série de réseaux de neurones avec différents hyperparamètres. Utilisez l'option `early_stopping=True` et trouvez les hyperparamètres qui donnent les meilleurs résultats sur l'ensemble de validation (pour vous donner une idée, entraîner environ 20 modèles différents est suffisant). Je vous suggère d'explorer les trois hyperparamètres suivants : le taux d'apprentissage, la taille du réseau et la force du terme de régularisation L2.

Dans votre réponse, veuillez inclure le code utilisé pour entraîner ces réseaux et obtenir la performance sur l'ensemble de validation (et seulement cette partie du code).

Quelle est la meilleure combinaison d'hyperparamètres et quelle est sa performance sur l'ensemble de validation?

8. (5pt) Qu'avez-vous appris sur l'importance des différents hyperparamètres?

Comparaison (2pt)

9. (2pt) Quelle est la performance finale de chaque modèle (k plus proches voisins et réseau de neurones)? *Veuillez fournir les quelques lignes de code utilisées pour ce faire.*

4 Représentations préentraînées (6pt)

La représentation sac de mots de la section 3 ignore l'ordre des mots et traite les mots comme des symboles sans lien entre eux. Les systèmes modernes d'apprentissage automatique réutilisent plutôt souvent des représentations apprises par des modèles préentraînés sur de grands corpus (rappelez-vous la capsule sur l'apprentissage de représentations).

Nous fournissons un *plongement de phrase* (*sentence embedding*) précalculé pour chaque évaluation : [instruments_embeddings.npz](#). La ligne i de la matrice est un plongement de dimension 384 de la i -ième évaluation de

`instruments.csv`, calculé avec le modèle préentraîné `all-MiniLM-L6-v2`.² Vous pouvez les charger avec :

```
E = np.load("instruments_embeddings.npz")["E"]
```

Utilisez la même division entraînement/validation/test qu'à la section 3 (`random_state=1234`) afin que les lignes demeurent alignées avec vos cibles.

1. (3pt) Ré-entraînez votre modèle des k plus proches voisins (vous pouvez re-sélectionner k sur la même grille qu'auparavant et revisiter brièvement votre choix de fonction de distance) ainsi que votre meilleure configuration de réseau de neurones de la section 3, cette fois en utilisant les plongements comme attributs. Rappelez les erreurs quadratiques moyennes de validation et de test. *Fournissez les quelques lignes de code utilisées.*
2. (3pt) Comparez ces résultats avec ceux obtenus avec la représentation sac de mots et expliquez la différence (ou son absence). En particulier : comment une représentation préentraînée de dimension 384 peut-elle rivaliser avec, voire surpasser, une représentation sac de mots de dimension 2 000? Reliez votre réponse à l'idée d'apprentissage de représentations vue en classe.

5 Vérification d'une solution générée (6pt)

Une étudiante a demandé à un LLM de résoudre une tâche similaire à celle de la section 3 (prédire la note d'une évaluation à partir de son texte) et a reçu la solution dont un extrait figure ci-dessous. Le code s'exécute sans erreur et l'explication se lit bien. L'extrait contient néanmoins **trois erreurs conceptuelles**.

```
texts, y = load_reviews("reviews.csv")

vectorizer = CountVectorizer(max_features=2000, stop_words="english")
X = vectorizer.fit_transform(texts) # encoder toutes les donnees

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=1234)

best_k, best_mse = None, np.inf
for k in [1, 10, 50, 100]:
    model = KNeighborsRegressor(n_neighbors=k).fit(X_train, y_train)
    mse = mean_squared_error(y_test, model.predict(X_test))
    if mse < best_mse:
```

²Les plongements ont été générés avec : `model = SentenceTransformer("all-MiniLM-L6-v2"); E = model.encode(texts); np.savez("instruments_embeddings.npz", E=E).`

```
best_k, best_mse = k, mse

print("Modele final : k =", best_k, "avec MSE =", best_mse)
```

L'explication qui accompagne le code se lit comme suit :

« Nous sélectionnons la valeur de k ayant la plus faible erreur sur l'ensemble de test et rapportons cette erreur comme la performance finale du modèle. Parmi les valeurs essayées, $k = 1$ performe souvent le mieux, puisque l'utilisation d'un seul voisin minimise la variance du prédicteur. »

Pour chacune des trois erreurs : (a) identifiez-la, (b) expliquez sa conséquence sur les résultats rapportés ou sur les conclusions, et (c) expliquez comment la corriger. (2pt par erreur)

6 Sonder un système en ligne (12pt)

Vous devez étudier un système en ligne qui pourrait utiliser l'apprentissage automatique pour faire des prédictions (bien sûr, vous ne savez peut-être pas avec certitude si c'est le cas ou non). Plutôt que de décrire le système en termes généraux, vous allez le *sonder* : formuler des hypothèses sur les situations où il devrait échouer, puis les tester avec un petit ensemble systématique d'interactions.

Choisissez un système doté d'une interface publique avec laquelle vous pouvez interagir directement. Choix préapprouvés : [DeepL](#) (traduction); un outil de reconnaissance optique de caractères (p. ex. Google Lens); un outil de dictée / reconnaissance de la parole; Shazam (reconnaissance musicale); une démo d'étiquetage d'images (p. ex. sur Hugging Face Spaces); la démo de l'[API Perspective](#) (détection de toxicité); la prédiction du prochain mot du clavier de votre téléphone. Vous pouvez proposer un autre système, pourvu que vous puissiez interagir avec lui directement (en cas de doute, demandez-nous). **Exclus** : [Google Translate](#) (l'exemple fil rouge du cours), les agents conversationnels généralistes (ChatGPT, Claude, Gemini, ...) et les systèmes que vous ne pouvez pas sonder directement (p. ex. les fils de recommandation comme ceux de Netflix ou de TikTok).

1. (3pt) Décrivez la tâche d'apprentissage automatique que le système tente de résoudre (entrées et sorties), les données (ou leur type) qui seraient nécessaires pour entraîner ce modèle, ainsi qu'une mesure de performance appropriée (vous n'avez pas nécessairement besoin d'utiliser une mesure qui existe).
2. (3pt) Formulez deux hypothèses de modes d'échec, chacune ancrée dans un concept du cours (p. ex. changement de distribution, entrées rares / rareté des données, corrélations fallacieuses, inadéquation entre l'objectif d'entraînement et les besoins des utilisateurs). Pour chacune, justifiez brièvement pourquoi vous vous y attendez pour ce système.

3. (4pt) Concevez et réalisez de 6 à 10 interactions qui testent vos hypothèses. Rapportez-les dans un tableau : entrée, sortie du système, hypothèse testée, et si le résultat la soutient ou la contredit. Si le système restreint l'accès d'une façon qui bloque l'un de vos tests, documentez la restriction : c'est en soi une observation sur les systèmes d'apprentissage automatique déployés.
4. (2pt) Pour chaque hypothèse, concluez (confirmée, rejetée ou non concluante). Pour les modes d'échec confirmés, étant donné vos connaissances en apprentissage automatique, comment tenteriez-vous de les corriger?