IFT714: Traitement automatique des langues naturelles

Dû, le 6 février 2023
23h59 heure de l'est

Devoir #2

Hiver 2023

Enseignant: Amine Trabelsi

- Ce devoir contient 4 problèmes, trois problèmes écrits et un exercice de programmation. Les réponses aux problèmes doivent inclure les étapes de calcul intermédiaires qui vous ont amené à arriver à la réponse suggérée. Aucun crédit ne sera accordé aux réponses qui ne répondent pas à ce critère.

- Pour la question de programmation, c'est-à-dire la question 4., nous fournissons un notebook ".ipynb" que vous pourriez ouvrir avec Google Colab et les fichiers du jeu de données téléchargeables à partir de la page du cours sur Moodle (les détails peuvent être trouvés ci-dessous dans les instructions de l'exercice).

- Dans le fichier PDF que vous soumetterez, veuillez indiquer à la fois la précision (accuracy) d'entrainement et de développement/validation sur le jeu de données pour la question de programmation, c'est-à-dire la question 4, ainsi que les réponses aux problèmes.

- Vous devrez également soumettre ".ipynb" et ".py" de votre notebook Colab terminé.

- L'utilisation de LaTeX est préférable mais pas obligatoire. Vous pouvez consulter un tutoriel LaTeX et ce cheatsheet. Dans la plupart des cas, si vous voulez écrire quelque chose en LaTeX, vous pouvez simplement Google "how to do {X} in latex" et les premiers liens devraient fournir la syntaxe que vous recherchez.

- Incluez votre nom et CIP avec votre soumission.

- Toutes les soumissions doivent être au format PDF suivant ce format: "prenom_nom_Devoir2_CIP".

- Si vous souhaitez soumettre des réponses manuscrites, vous pouvez les numériser et les soumettre au format PDF.

- Le devoir doit être remis sur le site du cours sur Moodle avant 23h59 heure de l'est à la date d'échéance.

- Selon la politique de jours de retard énoncée sur le plan de cours, un devoir soumis 24 heures après la date limite sera pénalisé de 3%. Un devoir soumis deux jours (24 à 48 heures) après la date limite sera pénalisé de 10%, et un devoir soumis trois jours sera pénalisé de 20%. Soumettre une livrable 72 heures (3 jours) après la date limite ne sera pas accepté.

- Les soumissions incomplètes ou en retard seront évaluées en tant que telles et aucun accommodement ou réévaluation ne seront faits.

---

1. (0.5 + 1.5 + 1 + 1 = 4 Points)

Considérons le problème de la classification binaire où $\boldsymbol{x} = (x_1, ..., x_d)^T \in \mathbb{R}^d$, $y \in \{0, 1\}$ et $\boldsymbol{w} = (w_1, ..., w_d)^T \in \mathbb{R}^d$ sont le vecteur d'attributs (features) d'entrée, la cible de sortie et le vecteur de poids, respectivement. Le modèle de régression binaire est paramétré de la manière suivante: $y \sim Bernoulli(\phi_y)$ avec $\phi_y = p(y = 1|\boldsymbol{x}, \boldsymbol{w}) = \sigma(z) = \frac{1}{1+e^{-z}}$ où $z = \boldsymbol{w}^T\boldsymbol{x} = \sum_{j=1}^{d} w_j x_j$.

(a) Pour un seul exemple $(\boldsymbol{x}^{(i)}, y^{(i)})$, la perte est définie comme la perte de log vraisemblance négative ou la perte d'entropie croisée:
$L_{CE}^{(i)}(\boldsymbol{w}) = -\log p(y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{w})$.
Rappelez-vous que, $p(y^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{w}) = \sigma(z^{(i)})^{y^{(i)}}(1 - \sigma(z^{(i)}))^{1-y^{(i)}}$, $y^{(i)} = 0, 1$.
Montrez que
$L_{CE}^{(i)}(\boldsymbol{w}) = -y^{(i)} \log \sigma(z^{(i)}) - (1 - y^{(i)}) \log\big(1 - \sigma(z^{(i)})\big)$

(b) En utilisant la règle de dérivation en chaîne, montrez que $\frac{\partial}{\partial w_j} L_{CE}^{(i)}(\boldsymbol{w}) = (\sigma(z^{(i)}) - y^{(i)})x_j^{(i)}, j = 1...d$.

(c) Soit $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ un ensemble d'entraînement de m exemples identiquement indépendamment distribués. La perte d'entropie croisée correspondant à D est $L_{CE}^{D}(\boldsymbol{w}) = \sum_{i=1}^{m} L_{CE}^{(i)}(\boldsymbol{w})$.
Dérivez l'expression pour $\frac{\partial}{\partial w_j} L_{CE}^{(D)}(\boldsymbol{w}), j = 1...d$, et fournissez ainsi les règles de mise à jour de la descente de gradient stochastique et de la descente de gradient par lot (batch).

(d) Donnez une interprétation des résultats trouvés dans (c).

2. (0.5 + 1 + 1 + 1.5 = 4 Points)

Considérons un problème de classification dans lequel la variable de sortie y peut prendre l'un des K valeurs, $y = 1, ..., K$, qui représentent les K classes $C_1, .., C_K$. Soit $\phi_1, ..., \phi_K$ soient les probabilités de chacun des résultats possibles. La variable y est modélisée comme une distribution Multinomiale sur les K résultats possibles avec $\phi_k = p(y = k), k = 1, ..., K$, et $\sum_{k=1}^{K} \phi_k = 1$.

(a) Montrer que $p(y) = \phi_1^{\mathbb{1}\{y=1\}}\phi_2^{\mathbb{1}\{y=2\}}...\phi_K^{\mathbb{1}\{y=K\}}$, où $\mathbb{1}\{y = k\} = 1$ lorsque $y = k$, sinon $\mathbb{1}\{y = k\} = 0$.

(b) Soit $\eta_k = \log \frac{\phi_k}{\phi_K}$, $k = 1, ..., K$. Montrer que $\phi_K = \frac{1}{\sum_{l=1}^{K} e^{\eta_l}}$, et ensuite $\phi_k = \frac{e^{\eta_k}}{\sum_{l=1}^{K} e^{\eta_l}}$ pour $k = 1, ..., K$. ϕ_k apparaît comme la k^{eme} composante de la fonction Softmax.

(c) Soit $\boldsymbol{x} = (x_1, ..., x_d)^T \in \mathbb{R}^d$ est un vecteur d'attributs d'entrée à classer dans l'une des K catégories. Supposons que η_k est lié de manière linéaire à $\boldsymbol{x}$, c'est-à-dire, $\eta_k = \boldsymbol{w}_k^T \boldsymbol{x}$, où $\boldsymbol{w}_k = (w_{k1}, ..., w_{kd})^T$.
Exprimez $p(y|\boldsymbol{x}, \boldsymbol{w}_1, .., \boldsymbol{w}_K)$.

(d) Soit $D = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^m$ un ensemble d'entraînement de m exemples identiquement indépendamment distribués. Dérivez l'expression de la perte d'entropie croisée correspondante en termes de $\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_K$.

3. (0.5 + 1.5 + 2 = 4 Points)

Dans la plupart des tâches de classification, il est préférable d'avoir une sortie normalisée telle que les valeurs retournées puissent être interprétées comme des probabilités. Voici la fonction softmax qui mappe un vecteur d'entrée $\boldsymbol{z} \in \mathbb{R}^K$ et le normalise en une distribution de probabilité, ce qui implique que les valeurs du vecteur mappé somment à 1, c'est-à-dire:

$$Softmax(\boldsymbol{z}) = \boldsymbol{s} = (s_1, ..., s_K)$$

, où $s_k = \frac{e^{z_k}}{\sum_{l=1}^K e^{z_l}}$.

(a) Montrez que $Softmax$ est invariant par rapport à la translation du vecteur d'entrée par un vecteur constant $\boldsymbol{a} = (a, ..., a)^T \in \mathbb{R}^K$, c'est-à-dire $Softmax(\boldsymbol{z} + \boldsymbol{a}) = Softmax(\boldsymbol{z})$.

(b) Expliquez pourquoi, en pratique, nous choisissons la constante $a = -\max_i(z_i)$.

(c) Dérivez $\frac{\partial s_k}{\partial z_l}$.

4. Question Programmation (8 Points)

Ce problème nécessitera un programme en Python 3. Le but est de créer un modèle de régression logistique que vous avez appris en cours sur un jeu de données réel de classification de sentiments.

Le jeu de données que vous utiliserez est collecté à partir de critiques de films en ligne. Tous les exemples ont déjà été tokenisés et rendu en minuscule et chaque exemple est étiqueté comme 1 (positif) ou 0 (négatif). Le jeu de données fourni a un ensemble d'entraînement et un ensemble de développement/validation.

Pour commencer, vous devriez charger et ouvrir le fichier fourni pour le devoir ***H23_Devoir-2_Question_Programmation.ipynb*** de préférence dans Google Colab (qui contient l'environnement nécessaire), à travers votre compte Google Drive. Téléchargez les fichiers ***train.txt*** et ***dev.txt*** (disponibles aussi sous Devoir 2 dans la Section Semaine 4 sur Moodle). Téléversez-les sur votre Google Drive, puis chargez-les dans votre notebook Colab en montant votre lecteur. Le code pour monter et charger est fourni pour vous dans le notebook, vous devez simplement modifier le chemin des fichiers si nécessaire.

Pour toutes les parties de codage, vous n'aurez pas besoin de créer de nouveaux fichiers ou notebooks. Le notebook a des sections où vous pouvez remplir le code pour tous les sous-problèmes. Recherchez le mot-clé "TODO" et remplissez votre code dans l'espace vide. Sentez-vous libre d'ajouter et de supprimer des arguments dans les signatures de fonctions, mais **faites attention** que vous pourriez avoir besoin de les changer dans les appels de fonction déjà présents dans le notebook.

Vous allez construire les parties clés d'un modèle de régression logistique pour l'analyse de sentiments. Vous devez implémenter la classe `LogisticRegressionClassifier` et la fonction `train_lr`.

- `train_lr`: elle prend une liste d'exemples d'entraînement (classe `SentimentExample`) et un extracteur d'attributs ou features (classe `FeatureExtractor`) en entrée et doit renvoyer une instance de `LogisticRegressionClassifier`.
- `LogisticRegressionClassifier`: vous devez au moins implémenter les fonctions `__init__`, `train`, et `predict`. Sentez-vous libre d'ajouter plus d'arguments à `__init__` et `train`, mais **NE changez pas les arguments dans** `predict` - qui doit simplement prendre un exemple `ex` (classe `SentimentExample`) et prédire une étiquette 1 ou 0.
- Vous voudrez probablement jeter un coup d'œil à la classe `UnigramFeatureExtractor` déjà implémentée pour vous !

Implémentez un modèle de régression logistique puis entraînez votre modèle avec des attributs/features Unigram (un mot) Sac-à-mots ou Bag-of-words (le code pour créer des attributs/features a déjà été écrit pour vous). Pour l'optimisation, utilisez la Descente de gradient stochastique (SGD).

Exécutez la partie d'évaluation (implémentée pour vous). Dans le fichier PDF que vous rendrez, veuillez rapporter à la fois la précision (accuracy) d'entrainement et de développement/validation sur le jeu de données pour cette question de programmation, en plus des réponses aux questions théoriques écrites.

Vous devez également soumettre le fichier " .ipynb" (avec la dernière cellule d'évaluation déjà exécutée) et le fichier " .py", tous deux extraits de votre notebook Colab une fois terminé (allez dans Fichier → Télécharger → Télécharger .ipynb OU Télécharger .py).

Notez qu'il n'est pas autorisé d'utiliser un quelconque outil ("toolkit") d'apprentissage automatique pour ce devoir, à l'exception des librairies scientifiques ou d'algèbre linéaire (par exemple, `numpy`). Les paquets explicitement interdits incluent `sklearn`, `keras`, `tensorflow`, `pytorch`, et `theano`.