TP 2
Introduction aux modèles génératifs

Elie Azeraf

Ce TP necessite l'installation de python 3 avec la librairie numpy et matplolib. Aucune autre librairie n'est autorisé.
Il est à rendre, au choix, soit dans un fichier Jupyter notebook, soit un PDF avec les codes joints en fichiers .py.
Pour être réaliser, il necessite le dossier *TP2_functions*.
Il est à rendre pour le vendredi 22/01/2021 avant 12h00, les seules commentaires écrits requis sont ceux demandés. Tant que le code est clair, vous n'avez pas besoin d'expliquer ce que vous faites.

1 Exercice 1 (8 points)
Classification de texte avec un modèle Naive Bayes

1.1 Les données

Dans le cadre de cet exercice nous allons utiliser le dataset AG News. Ce dataset, en anglais, est parmi les plus populaires pour la classification de texte. Il est composé d'un corpus d'entraînement composé de 120 000 textes, et d'un corpus de test composé de 7 900 textes. Les textes sont séparés en quatre catégorie:

- World
- Sport
- Business
- Sci/Tech

Chaque texte possède un titre et un contenu, ils sont concaténés au préalable pour plus d'aisance lors de l'exercice.

Dans les ficher disponibles, vous disposez du fichier *load_ag_news.py*. Vous pouvez l'utiliser ainsi:

```
from load_ag_news import load_ag_news

path = "path/to/TP2_functions/"
train_set, test_set = load_ag_news(path)
```

L'élément train_set est une liste des différents textes du corpus d'entraînement. Chaque élément de cette liste est composé de deux éléments:

- La classe du texte, prenant les valeurs 0, 1, 2, ou 3
- Le texte lui-même, il s'agit d'une liste des différents mots, par exemple, ["My", "name", "is", "Bond", ",", "James", "Bond", "."].

1.2 Développer un Naive Bayes !

L'objectif de cette exercice est de coder le Naive Bayes exactement de la même manière que vue en cours.

Votre code suivra les étapes suivantes:

- Estimation des paramètres sur le corpus d'entraînement, vous afficherai clairement les valeurs du paramètre π et $\forall i \in \{0, 1, 2, 3\}, b_i('London')$.
- Coder un modèle Naive Bayes. Veuillez à avoir un code clair, lisible, et ergonomique.
- Prédiction pour la phrase ["Arsenal", "is", "a", "londonian", "team", "."], avec affichage des différentes probabilités pour chaque classe, en prenant $\varepsilon = 10^{-5}$.
- Prédiction sur l'ensemble du corpus de test, vous indiquerai le pourcentage de textes bien prédit. Vous ferez ces prédictions trois fois en prenant ε dans $\{10^{-10}, 10^{-5}, 1\}$. Commentez les résultats en fonction des différentes valeurs.

Attention, il est possible que vous passiez au logarithme pour éviter les erreurs d'underflows.

1.3 Question bonus

Le lien suivant https://paperswithcode.com/sota/text-classification-on-ag-news classe les meilleurs algorithmes sur le dataset AG News, en classification de texte. Au vue des résultats obtenus avec le Naive Bayes, que pouvez-vous dire selon vous ?

Cette question n'est pas noté mais une réponse judicieuse et bien construite rapportera un point bonus !

2 Exercice 2 (8 points) Étiquetage morpho-syntaxique avec une chaîne de Markov cachée

2.1 Introduction

L'étiquetage morpho-syntaxique, Part-Of-Speech Tagging (POS Tagging) en anglais, consiste à étiqueter chaque mot d'un texte avec sa fonction grammaticale. Par exemple, la phrase *("Batman", "is", "the", "superhero", "of", "Gotham", ".")* possède les labels *(NOUN, VERB, DET, NOUN, PREP, NOUN, PUNCT)*.

Le dataset utilisé est CoNLL 2000 (qui est une référence pour la décomposition syntaxique, Chunking en anglais), vous pouvez le charger avec la fonction *load_conll2000.py* comme ci-après:

```
from load_ag_news import load_ag_news

```

```
path = "path/to/TP2_functions/"
train_set, test_set = load_conll2000(path)
```

2.2 Développer une chaîne de Markov cachée !

L'objectif de cet exercice est de coder une chaîne de Markov cachée tel que vu en cours.

Votre code suivra les étapes suivantes:

- Estimation des paramètres sur le corpus d'entraînement
- Coder l'algorithme Foward-Backward.
- Afficher les labels prédits sur la phrase: ["I", "am", "a", "student", "in", "a", "French", "school", "."] (vous n'avez pas besoin d'en indiquer les probabilités), avec $\varepsilon = 10^{-5}$.
- Appliquer la chaîne de Markov cachée avec Forward-Backward sur tout le corpus de test, et donnez le pourcentage de mots bien labélisés, avec $\varepsilon = 10^{-5}$.

3 Exercice 3 (5 points) Filtrage de Kalman

3.1 Introduction

Vous disposez d'une série temporelle $y_{1:T}$ de taille $T = 10000$. Vous pouvez charger les signaux ainsi:

```
import numpy as np

path = "path/to/TP2_functions/"
X = np.loadtxt(path + "signal_X.txt")
Y = np.loadtxt(path + "signal_Y.txt")
```

X étant le signal original, et Y sa version bruitée.

3.2 Le filtre de Kalman !

Votre objectif est de coder un filtre de Kalman !

Votre code, en python, devra être organisé de la façon suivante:

- Dans un premier temps, vous devrez estimer la valeur des paramètres a et b de manière **non supervisée**.
- Vous coderez un filtre de Kalman afin de restaurer le signal $\hat{x}_{1:T}$ à partir de $y_{1:T}$.
- Évaluez votre filtre en calculant l'erreur quadratique moyenne entre x et votre signal restauré $\hat{x}$. Calculez également l'erreur quadratique moyenne entre x et y, que pouvez-vous conclure ?
- Affichez sur un graphique les 100 premières itérations de votre signal restauré et le signal d'origine.