Examen

Deux feuilles A4 autorisées, calculette autorisée, barème indicatif

Exercice 1 (7 points) – Questions indépendantes

Q 1.1 Expliquer en quelques lignes comment adapter au cas multi-classes (a) le modèle de réseau de neurones, (b) le modèle du perceptron.

Q 1.2 SVM et noyaux

Q 1.2.1 Rappeler le principe de l'algorithme SVM, ses principaux avantages et inconvénients.

Q 1.2.2 Qu'est ce qu'un noyau ?

Q 1.2.3 Pour $x, x' \in \mathbb{R}^d$, préciser si les fonctions suivantes sont des noyaux : (a) la similarité cosine définie par $s(x, x') = \frac{<x,x'>}{||x||||x'||}$, (b) la fonction $m(x, x') = max(0, x - x')$.

Q 1.2.4 Soit S une séquence de mots sur un alphabet $\mathcal{A}$ fini et la fonction $K : S \times S \to \mathbb{R}$, $K(x, x') = 1$ si x et x' ont au moins une sous-chaîne de longueur 2 en commun, 0 sinon. Donner la matrice $\mathcal{K} := k_{i,j} = k(x_i, x_j)$ pour $x_1 = 1100$, $x_2 = 1122$, $x_3 =, 1000$. Montrer que cette fonction ne peut pas être un noyau.

Q 1.3 Décrire en quelques lignes le problème dit du dilemme exploration vs exploitation et quelques algorithmes pour le traiter.

Q 1.4 Répondre par oui ou non en justifiant en quelques mots :

(a) les k-nns sont adaptés aux problèmes de grande dimension ;

(b) avec un algorithme de boosting, il vaut mieux utiliser des SVMs que des perceptrons ;

(c) la solution d'un clustering est généralement unique à nombre de clusters fixés ;

(d) la VC-dimension est une notion utile pour déterminer le bruit dans les données.

Exercice 2 (6 points) – Apprentissage d'ordonnancement

L'objectif du *ranking* (ou ordonnancement) est de prédire des préférences sur les items considérés. L'exemple classique de machines d'ordonnancement sont les moteurs de recherche (tels que Google par exemple) qui retournent des listes ordonnées de résultats en réponse à un besoin d'information. La tâche d'un moteur de recherche est alors de décider quel document placer au dessus de quel autre document.
Soient N échantillons étiquetés $X = \{\mathbf{x}^i\}_{i=1,\ldots,N}$ codés sous forme de vecteurs à d dimensions : $\mathbf{x}^i \in \mathbb{R}^d$. Les étiquettes sont réelles : $Y = \{y^i\}_{i=1,\ldots,N}, y^i \in \mathbb{R}$. Plutôt que d'apprendre directement à associer les étiquettes aux vecteurs d'entrée, ce qui peut dans certains cas se révéler une tâche difficile, on se

contente de respecter des ordres de préférence sur les items : plus y_i est grand, plus il doit être préféré selon notre prédicteur $f(\mathbf{x}^i)$.
Nous proposons d'optimiser la fonction coût suivante :

$$L = \sum_{(i,j),y^i>y^j} (y^i - y^j)\max(0, \gamma - (f(\mathbf{x}^i) - f(\mathbf{x}^j))), \text{ avec } \gamma = 1$$

Dans toute la suite, nous proposons d'utiliser un prédicteur linéaire :

$$f(x^i) = \sum_j x^i_j w_j = \langle \mathbf{x}^i, \mathbf{w} \rangle \in \mathbb{R} \text{ (produit scalaire classique)}$$

Q 2.1 Interprétation de la fonction de coût.

Q 2.1.1 A quoi correspond une valeur nulle du coût pour une paire d'échantillons ? une valeur positive de coût ? Comment se comporte le coût en fonction de l'écart entre y_i et y_j ?

Q 2.1.2 Que se passe-t-il si on augmente la valeur de γ ? Et pour une valeur nulle de γ ?

Q 2.2 Optimisation : nous utiliserons une technique de gradient stochastique. Le gradient est calculé à chaque étape sur seulement une paire de points tirés au hasard.

Q 2.2.1 Calculer le gradient du coût par rapport à $\mathbf{w}$ pour un couple (i, j).

Q 2.2.2 Rappeler la définition d'un algorithme de descente de gradient (cas général). Donner ensuite l'algorithme pour ce cas particulier. Rappeler les entrées (hyper-paramètres de la méthode) et les sorties de cet algorithme.

Q 2.2.3 Cet algorithme étant itératif, proposer au moins deux critères d'arrêt en justifiant succinctement les points forts et faibles de ces critères (en terme de facilité d'implémentation ou de coût de calcul par exemple).

Q 2.3 Preuve de bon fonctionnement. Soit $y^i > y^j$:

Q 2.3.1 Exprimer $\mathbf{w}^{t+1}$ en fonction de $\mathbf{w}^t$ lorsqu'il y a mise-à-jour (application de la descente de gradient sur une itération en considérant que l'on a choisi un exemple mal classé)

Q 2.3.2 Montrer que $f^{t+1}(\mathbf{x}^i) - f^{t+1}(\mathbf{x}^j) = f^t(\mathbf{x}^i) - f^t(\mathbf{x}^j) - \epsilon(y^i - y^j)\|\mathbf{x}^i - \mathbf{x}^j\|^2$, avec ϵ le pas de gradient

Q 2.3.3 Comparer $f^{t+1}(\mathbf{x}^i) - f^{t+1}(\mathbf{x}^j)$ et $f^t(\mathbf{x}^i) - f^t(\mathbf{x}^j)$. Le résultat obtenu vous semble-t-il logique ? Expliquer rapidement pourquoi.

Exercice 3 (4 points) – Reinforcement learning

Rappel : opérateur de Bellman : $(T^\pi V)(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s))V(s')$
On considère le jeu de pacman : pacman se balade dans un labyrinthe 2D, composé de murs, de cases vides et de pastilles ; de 4 fantômes ont pour objectif de manger pacman ; pacman a pour objectif de ramasser toutes les pastilles sans mourir (rencontrer un fantôme).

Q 3.1 Rappeler en quelques lignes ce qu'est un état, une action, une politique, une récompense et V la fonction valeur des états.

Q 3.2 On considère le problème simple où il n'y a ni pastilles ni fantôme. Le but dans ce cas est uniquement d'atteindre une case précise du labyrinthe (la sortie). Soit un labyrinthe de 4 colonnes et

2 lignes, avec un mur en case $(3, 1)$ et la sortie en case $(4, 1)$. On suppose une récompense de 100 pour la case de sortie et une récompense de -1 pour chaque autre case. Faites tourner l'algorithme de value iteration sur cet exemple (avec un discard $\gamma = 1$). Quel effet si la récompense pour une case vide est de 0 ?

Q 3.3 On considère maintenant que les fantômes et les pastilles existent, le but étant de manger toutes les pastilles sans se faire manger par un fantôme.

Q 3.3.1 Quelle modélisation proposer vous pour un état (indication : considérer qu'une sous-grille du plateau de jeu, centré autour de pacman) ? Quelles récompenses ? Combien d'états possibles ?

Q 3.3.2 On ajoute des pastilles spéciales qui quand elles sont mangées, permettent à pacman d'être invincible et de manger les fantômes. Quel(s) changement(s) apporter à la modélisation précédente ? Combien d'états ?

Q 3.4 L'apprentissage sur cette modélisation vous semble-t-elle réaliste ? Avec quel(s) algorithme(s) ? Comment faire baisser la complexité en temps de l'apprentissage ?

Exercice 4 (8 points) – Filtrage collaboratif

Le but de l'exercice est de proposer un modèle pour la recommandation (ou filtrage collaboratif) : il s'agit de prédire l'avis d'un utilisateur sur un item (qui peut être un film, un produit, un restaurant, ...) en fonction de votes connus entre n utilisateurs (ensemble $\mathcal{U}$) et m items (un ensemble $\mathcal{V}$). Les votes (ou scores) que les utilisateurs ont donné aux items sont représentés par une matrice de votes (ratings) R de taille $n \times m$, avec r_{ij} le vote de l'utilisateur i pour l'item j. On suppose tous les votes strictement positif (par exemple entre 1 et 5), une valeur nulle pour r_{ij} signifiera donc que le score de l'utilisateur i pour l'item j n'est pas connu.
Une modélisation courante de ce contexte est de considérer un espace *latent* - non explicite mais exprimé dans les données - de facteurs explicatifs de l'avis des utilisateurs sur les items. Par exemple, dans le cas de la recommandation de films, on peut imaginer que cet espace latent est composé des différents genres de films (comédie, thriller, romance, ...). Un film peut être alors représenté par un ensemble de poids sur les différents genres - son vecteur de poids - par affinité de chaque genre avec le film. De même, un utilisateur peut être représenté par son vecteur de poids sur les différents genres. Le score prédit pour un utilisateur sur un film est alors le produit scalaire entre les deux vecteurs de poids. L'erreur au moindres carrés entre le score prédit et le score observé est généralement utilisée comme fonction de coût.

Q 4.1 On suppose un espace latent de dimension 4 : (comédie, romance, thriller, policier) et les données suivantes :

	comédie	romance	thriller	policier
films				
v_1 : Angry Bird	0.4	0.2	0.	0.
v_2 : The Big Lebowski	0.8	0.2	0.6	0.4
v_3 : Fargo	0.4	0.1	0.3	0.2
v_4 : Saw	0	0	0.6	0.1
users				
u_1	0.5	0.5	0.5	0.5
u_2	0.2	0.2	0.6	0.8
u_3	0.1	0.1	0.3	0.8

user	film	score
u_1	v_1	3
u_1	v_3	10
u_2	v_2	9
u_2	v_3	4
u_3	v_3	3
u_3	v_4	3

Q 4.1.1 Calculer la matrice de scores R pour ces données.

Q 4.1.2 Comparer les scores obtenus avec les scores de supervision, donner le coût de cette approximation. Quel opération élémentaire permet d'améliorer grandement le coût ? Est-ce important ?

Q 4.1.3 Que remarquez vous pour les scores et la représentation des films v_2 et v_3 ? Sont-ils proches ou éloignés ? Expliquez le phénomène observé en tenant compte de la moyenne des votes pour chaque film. La distance euclidienne vous semble-t-elle adaptée pour calculer la proximité entre films (resp. entre utilisateurs) ? Quelle autre distance utilisée ?

Q 4.2 Soit la matrice des scores R de taille $n \times m$, chaque ligne $r_{i.}$ correspondant aux scores donnés par un utilisateur et chaque colonne aux scores reçus $r_{.j}$ par un item. On considère que toute la matrice est connue. Formaliser le problème de recommandation décrit dans l'énoncé comme la recherche de deux matrices U et V de tailles $n \times d$ et $d \times m$ qui minimisent une expression faisant intervenir R également. A quoi correspondent U, V et d ?

Q 4.3 Au terme trouvé à la question précédente, on ajoute le terme $\lambda(\sum_i ||u_{i,.}||^2 + \sum_j ||v_{.,j}||^2)$ pour établir la quantité finale à optimiser. Que représente les sommes de ce second terme ? Quel intérêt ?

Q 4.4 En réalité, nous disposons rarement de tous les scores : la matrice R est parcimonieuse, avec très peu de valeurs connues. Comment adapter le problème d'optimisation ?

Q 4.5 Le problème n'est pas directement optimisable : pourquoi ? Une solution est d'optimiser par alternance les deux matrices U et V : à un pas de temps, U est optimisée en considérant V fixée, à celui d'après V est optimisée en considérant U fixée.

Q 4.5.1 A quelle famille classique de problèmes d'apprentissage statistique correspond chacune des deux phases d'optimisation ?

Q 4.5.2 Pour un rating $r_{i,j}$, donner les formules de mise-à-jour associées. Donner l'algorithme complet d'optimisation (en considérant une optimisation stochastique).

Q 4.5.3 Donner l'implémentation python.

Q 4.6 Tous les utilisateurs n'ont pas la même moyenne d'avis (un utilisateur peut être plus sévère qu'un autre mais avoir les mêmes gouts), de même pour les items. Que manque-t-il au modèle pour pouvoir représenter ces biais ? Donner les modifications à apporter à l'algorithme.