Examen 1ère session (1h30) - 12 mai 2021

Rappels : **Aucun document n'est autorisé.** Les calculatrices et autres appareils électroniques doivent être éteints et rangés. Le barème (sur 25) n'est donné qu'à titre indicatif.

Exercice 1 *Algorithme des k-moyennes (5 points)*

Soit un ensemble d'exemples $\mathbf{X} \in \mathbb{R}^{n \times d}$

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{pmatrix}$$

Q. 1. Donner l'expression (mathématique) des coordonnées du centre de gravité $c = (c_1, \dots, c_d)$ de $\mathbf{X}$.

Q. 2. Écrire la fonction Python `centroide` qui, étant un ensemble d'exemples, contenant au moins 2 exemples donné sous la forme d'un array rend leur centroide (sous la forme d'un array).

Q. 3. Dans l'algorithme des k-moyenne, qu'appelle-t-on l'inertie intra-cluster d'un cluster ? En considérant que $\mathbf{X}$ est un cluster, donner l'expression mathématique de son inertie.

Q. 4. Écrire la fonction Python `inertie` qui, étant donné un array contenant un ensemble d'exemples contenant au moins 2 exemples, rend la valeur de l'inertie de cet ensemble.

Exercice 2 *Arbres de décision (6pts)*

On considère la base d'apprentissage suivante :

Âge	Prescription	Astigmatisme	Classe
jeune	myope	non	pas de lentilles
jeune	hypermétrope	oui	lentilles
âgé	myope	non	lentilles
âgé	myope	oui	pas de lentilles

Q. 1. On construit un arbre de décision pour prédire si un patient doit porter des lentilles. Indiquer le premier test sélectionné lorsqu'on utilise l'entropie de Shannon.

Q. 2. On souhaite construire un arbre parfait. Indiquer s'il est nécessaire de faire des tests supplémentaires et, le cas échéant, les déterminer et donner l'arbre complet obtenu.

Q. 3. Existe-t-il un arbre parfait de taille inférieure (en profondeur) ?

Exercice 3 *Descente(s) de gradient (6 points)*
Soit des données supervisées et une problématique de régression linéaire.

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{pmatrix}, \forall i\ \mathbf{x}_i \in \mathbb{R}^d \qquad \mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \forall i\ y_i \in \mathbb{R}, \qquad f(\mathbf{x}_i) = \mathbf{w}\mathbf{x}_i$$

On envisage deux fonctions de coût à optimiser, dites L_1 et L_2 et définies de la manière suivante :

$$C_{L_1} = \sum_i \left| f(\mathbf{x}_i) - y_i \right|, \qquad C_{L_2} = \sum_i (f(\mathbf{x}_i) - y_i)^2$$

©2020-2021 (by Sorbonne Université/Licence d'informatique/LU3IN026)

Q. 1. Quelle est la dimension de $\mathbf{w}$?

Q. 2. Vous êtes familier avec C_{L_2} mais pas encore avec C_{L_1}. Dans l'absolu, quelles erreurs seront plus pénalisées par C_{L_1} que par C_{L_2} ? Dans quelle application cela pourrait-il être utile ?

Q. 3. La dérivée de la valeur absolue est difficile mais nous allons néanmoins procéder simplement : nous distinguerons les cas où $f(\mathbf{x}_i) \geq y_i$ et les cas où $f(\mathbf{x}_i) < y_i$. Dans chaque cas, la dérivée devient triviale. Donner l'expression de $\frac{\partial C_{L_1}}{\partial w_j}$ pour un échantillon $\mathbf{x}_i$. (Comme pour le perceptron et ADALINE, on ne considère que le coût en i)

Q. 4. En déduire la valeur de $\nabla_{\mathbf{w}} C_{L_1}$, c'est à dire de l'ensemble des dérivées partielles pour toutes les dimensions. Indiquer la dimension de $\nabla_{\mathbf{w}} C_{L_1}$.

Q. 5. Donner le code Python de la méthode `grad_L1(X,Y,epsilon, niterations)` qui part d'un vecteur nul et réalise `niterations` de descente de gradient stochastique (i.e. en tirant aléatoirement un indice i à chaque itération).
Vous ne traiterez pas le problème de la convergence mais vous réfléchirez à ce que doit retourner cette fonction.

Exercice 4 *Recursive Backward Elimination & régularisation (5 points)*

En conservant les mêmes notations qu'à l'exercice précédent mais en considérant maintenant un problème de classification binaire où la dimension d est très grande, nous risquons de faire face à du sur-apprentissage.

Afin d'éliminer le problème, nous proposons l'algorithme suivant :

a) Considérer tous les sous-ensembles de caractéristiques (= ensembles de colonnes) de taille $d-1$;

b) Apprendre un classifieur sur ces données réduites et l'évaluer ;

c) Conserver le meilleur sous-ensemble de caractéristiques ;

d) Itérer le processus jusqu'à l'élimination de toutes les caractéristiques.

Q. 1. Définir la notion de sur-apprentissage en quelques lignes. Pourquoi une plus grande valeur de d induit-elle plus de risque de sur-apprentissage ?

Q. 2. A chaque itération, une caractéristique est donc éliminée du problème. Imaginez quelle forme générale aura la courbe de mesure des performances au cours des itérations.

Q. 3. Quelle procédure d'évaluation suggérez-vous dans le cas présent ? Si on utilise une procédure d'évaluation naïve sur les données d'entraînement, quel ensemble de caractéristiques optimal va forcément émerger ?

Q. 4. Si l'apprentissage d'un classifieur (quel qu'il soit et quel que soit d) coûte une unité de calcul et que les autres coûts sont négligeables, combien coûte l'ensemble de la procédure ?

Q. 5. Ce coût étant très élevé, les chercheurs en IA préfèrerait une procédure qui élimine les dimensions inutiles au cours de l'apprentissage. Cette approche s'appelle la régularisation. Un exemple d'implémentation consiste à optimiser la fonction de coût suivante :

$$C_{L_2R} = \underbrace{\sum_{i=1}^{n} (\mathbf{w}\mathbf{x}_i - y_i)^2}_{\text{Coût classique}} + \lambda \underbrace{\sum_{j=1}^{d} w_j^2}_{\text{Régularisation}}$$

où λ est un hyper-paramètre fixé par un expert.

a) Si $\lambda \to 0$, nous revenons au problème original... Mais si $\lambda \to \infty$, quelle est la solution évidente du problème de minimisation du coût ?

b) En vous focalisant uniquement sur le terme de régularisation, calculer le gradient de ce terme. Analyser l'impact du terme de régularisation sur la mise à jour d'un poids w_j au cours de la descente de gradient, dans les cas où $w_j > 0$ et $w_j < 0$. Conclure sur l'intérêt de la procédure.

©2020-2021 (by Sorbonne Université/Licence d'informatique/LU3IN026)

Exercice 5 *Recommandation de films & construction de caractéristiques discriminantes points) (3 points)*

Nous disposons d'une grande base de données de films avec différentes caractéristiques :

— tous les acteurs principaux,

— le box-office,

— les nominations aux oscars et aux différents festivals européens,

— la catégorie (comédie, policier, ...).

Q. 1. En considérant que chaque film i est décrit par des caractéristiques sous la forme d'un vecteur $\mathbf{x}_i$ et que l'on possède des avis (binaires) d'utilisateurs sur certains films ($y_{u,i} \in \{+1, -1\}$), comment construire un algorithme de recommandation personnalisé ? Expliquer les algorithmes et les données que vous utiliseriez.

Q. 2. En considérant que la base de données compte 10 catégories de films, comment coder ces informations pour que l'algorithme ci-dessus puisse en tirer parti ?

Q. 3. Les acteurs représentent un grand facteur d'attraction pour les spectateurs. Par défaut, on considère un système où chaque acteur est une caractéristique binaire. Le problème est qu'un certain nombre d'acteurs n'apparaît que dans un seul film de la base. Quel est le risque avec une telle caractéristique ? Faut-il supprimer ces acteurs ou existe-t-il moyen de conserver ces informations sous une certaine forme ?

Q. 4. Certains acteurs donnent l'impression d'être interchangeables, présents toujours dans les mêmes types de film (comédie ou action par exemple), film d'auteur ou blockbuster... Comment tester cette hypothèse ? Quelles caractéristiques pouvez-vous proposer pour en tenir compte et comment les construire ?

Q. 5. Le fait d'être nominé dans n'importe quel festival vous semble-t-il être une caractéristique unique ou opteriez-vous pour un codage séparé des différentes nominations ?

Annexes

n / d		dénominateur (d)									
		1	2	3	4	5	6	7	8	9	10
numérateur (n)	1	1	0,50	0,33	0,25	0,20	0,17	0,14	0,13	0,11	0,10
	2		1	0,67	0,50	0,40	0,33	0,29	0,25	0,22	0,20
	3			1	0,75	0,60	0,50	0,43	0,38	0,33	0,30
	4				1	0,80	0,67	0,57	0,50	0,44	0,40
	5					1	0,83	0,71	0,63	0,56	0,50
	6						1	0,86	0,75	0,67	0,60
	7							1	0,88	0,78	0,70
	8								1	0,89	0,80
	9									1	0,90
	10										1

-(n/d) * log(n/d)		dénominateur (d)									
		1	2	3	4	5	6	7	8	9	10
numérateur (n)	1	0	0,50	0,53	0,50	0,46	0,43	0,40	0,38	0,35	0,33
	2		0	0,39	0,50	0,53	0,53	0,52	0,50	0,48	0,46
	3			0	0,31	0,44	0,50	0,52	0,53	0,53	0,52
	4				0	0,26	0,39	0,46	0,50	0,52	0,53
	5					0	0,22	0,35	0,42	0,47	0,50
	6						0	0,19	0,31	0,39	0,44
	7							0	0,50	0,28	0,36
	8								0	0,15	0,26
	9									0	0,14
	10										0

©2020-2021 (by Sorbonne Université/Licence d'informatique/LU3IN026)