-2015

Examen 2014-2015

Documents autorisés : 1 feuille A4 recto verso
Barême donné à titre indicatif

Notations Nous considérerons $\mathbb{R}^d$ l'espace de représentation des exemples, un ensemble de labels $\mathcal{Y} = \{-1, 1\}$ dans le cas de la classification binaire, $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ un vecteur décrivant un exemple, un ensemble de n exemples d'apprentissage $X_{app} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ et leur label associé $Y_{app} = \{y^1, \ldots, y^n\}$. On notera $\mathbf{1}(.)$ la fonction qui renvoie 1 si la condition est vraie et 0 sinon.

Exercice 1 (7 points) – Echauffement - Questions indépendantes

Rappel : un noyau, fonction $K : X \times \mathcal{X} \to \mathbb{R}$, est dit admissible s'il existe une fonction $\phi : X \to \mathcal{B}$ (fonction de projection ou *feature map*) et $K(x, x') =< \phi(x), \phi(x') >$.

Q 1.1 Montrez que si K et K' sont deux noyaux en explicitant la fonction de projection ϕ :

1. cK est un noyaux pour $c \in \mathbb{R}^+$
2. $K + K'$ est un noyau ;
3. $Q(x, y) = K(z, x)K'(z, y)$ est un noyau pour z fixé.

Q 1.2 Soit le problème en 2D suivant : la région qui est dans le triangle de coordonnées $\{(0, 0), (0, 1), (1, 1)\}$ est négative, celle en dehors positive. Proposer un réseau de neuronnes qui permettent de séparer les deux régions. Est-il possible d'utiliser un perceptron ?

Q 1.3 Répondre par vrai ou faux **en justifiant** en 1 ligne la réponse :

1. Plus l'ensemble d'apprentissage est grand, plus le risque de sur-apprentissage est grand.
2. Pour les k-plus proches voisins, de grandes valeurs pour K augmentent le risque de sur-apprentissage.
3. La regression logistique apprend des frontières non-linéaires car la fonction logistique est non linéaire.
4. Soit des données séparables linéairement, l'algorithme du perceptron converge vers des minimas locaux du fait de l'initialisation aléatoire des poids.

Soit X un vecteur aléatoire de $\mathbb{R}^d$, Y une variable aléatoire dans $\{-1, 1\}$ et f une fonction de $\mathbb{R}^d \to \mathbb{R}$, et les coûts suivants :

1. $L^1_{\alpha,\beta}(f) = \mathbb{E}(\alpha \mathbf{1}[Y = 1]\mathbf{1}[\text{signe}(f(X)) = -1)] + \beta \mathbf{1}[Y = -1]\mathbf{1}[\text{signe}(f(X)) = 1]$, $\alpha, \beta > 0$
2. $L^2(f) = \mathbb{E}((Y - \text{signe}(f(X)))^2)$
3. $L^3(f) = \mathbb{E}((Y - f(X))^2)$, $f : \mathbb{R}^d \to \mathbb{R}$

Q 1.4 Commenter en quelques mots chacun des coûts (ce qu'il représente, cas d'utilisation, problème ou avantage). A quel coût correspond le coût $0 - 1$?

Q 1.5 Version empirique

Q 1.5.1 En considérant une base d'apprentissage $X_{app} = \{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ avec les labels $Y_{app} = \{y^1, \ldots, y^n\}$, donner l'expression du coût (par exemple de L^1) empirique sur cette base d'exemple.

Q 1.5.2 On suppose f linéaire, paramétrisée par le vecteur $\mathbf{w} = (w_1, \cdots, w_d)$ (sans biais pour simplifié). Que vaut $f(x)$? Quelle propriété mathématique sur L caractérise un vecteur w qui optimise le coût ?

Q 1.5.3 Que vaut le gradient de L^1 et L^2 par rapport à $\mathbf{w}$? Que peut-on en conclure ?

Q 1.5.4 Calculer le vecteur $\mathbf{w}$ optimal pour L^3.

Q 1.6 Version théorique

Q 1.6.1 Exprimer L^1 sous sa forme intégrale en fonction de $p(x, y)dxdy$.

Q 1.6.2 Soit $\eta(x) = p(Y = 1|x)$. Donner pour L^1 la fonction f optimale en fonction de η. Pour cela, vous envisagerez les coûts trouvés précédement à un $\mathbf{x}$ fixé.

Q 1.7 Soit le coût $L^4(f) = \mathbb{E}(log(1 + e^{-Yf(X)}))$, $f : \mathbb{R}^d \to R$. En examinant les valeurs extremes possibles, comparer et commenter l'utilité de ce coût.

Exercice 2 (8 points) – K-means et descente de gradient

L'algorithme K-means est un algorithme de classification non-supervisé. Il prend en entrée un ensemble d'apprentissage $\{\mathbf{x}^i\}, \mathbf{x}^i \in \mathbb{R}^d$ d'exemples en d dimensions et un entier k et produit en sortie un clustering $K = \{\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^k\}, \mathbf{w}^i \in \mathbb{R}^d$. Les $\mathbf{w}^i$ sont appellés prototypes. On notera par la suite $s^i(K)$ la fonction d'affectation de l'exemple i pour un clustering K qui associe à l'exemple $\mathbf{x}^i$ le numéro du cluster associé, i.e. $s^i(K) = argmin_j(\mathbf{x}^i - \mathbf{w}^j), \mathbf{w}^j \in K$. On notera également K_j l'ensemble des exemples d'apprentissage associés au j-ème prototype : $K_j = \{\mathbf{x}^i | s^i(K) = j\}$.
L'algorithme fonctionne de la manière suivante :

- l'ensemble des prototypes K est initialisé aléatoirement ;
- Répéter jusqu'à stabilisation :
  - on estime les nouveaux barycentres en fonction du clustering : $w'^j = \frac{1}{|K_j|} \sum_{\mathbf{x}_i \in K_j} \mathbf{x}^i$;
  - le nouveau clustering est donné par les fonctions d'affectations $s^i(K')$ sur l'ensemble des nouveaux barycentres $K' = \{\mathbf{w}'^1, \ldots, \mathbf{w}'^k\}$

Q 2.1 Soit $C(K)$ le coût de reconstruction : la moyenne des distances euclidiennes au carré entre chaque exemple $\mathbf{x}^i$ et son prototype associé $\mathbf{w}^i$.

Q 2.1.1 Donner l'expression de $C(K)$ en fonction des $\mathbf{x}^i$, des $\mathbf{w}^i$ et des s^i.

Q 2.1.2 Que cherche-t-on à optimiser ? Proposer un algorithme par descente de gradient pour optimiser le coût. Converge-t-il vers la solution optimale ?

Q 2.2 Soit $K' = \{\mathbf{w}'^1, \ldots, \mathbf{w}'^k\}$ et $K = \{\mathbf{w}^1, \ldots, \mathbf{w}^k\}$ deux clusterings, et $Q(K', K) = \sum_i \frac{1}{n} \|\mathbf{x}^i - \mathbf{w}'^{s^i(K)}\|^2$.

Q 2.2.1 Que représente $Q(K, K)$?

Q 2.2.2 Supposons K fixe, on fait varier K'. A quoi correspond $Q(K', K)$? A quoi correspond le clustering K' qui minimise Q ?

Q 2.2.3 Que vaut le gradient de Q par rapport à w'^j ?

Q 2.2.4 Proposer une version batch de l'algorithme de descente de gradient pour optimiser Q.

Q 2.3 Soit K' le minimum de $Q(\bullet, K)$. Quel est le signe de $Q(K', K') - Q(K', K)$? Montrer que $Q(K', K') - Q(K, K) \leq 0$ en exprimant cette quantité en introduisant $Q(K', K') - Q(K', K)$ dans l'équation.

Q 2.4 Lorsque K est proche d'une solution optimale de Q, que vaut $\partial s_i(K)/\partial w_j$? Concluer sur la convergence de l'algorithme k-means.

Q 2.5 Proposer une version stochastique de l'algorithme de descente de gradient pour optimiser Q à partir du résultat de la question 2.2.3.

Exercice 3 (5 points) – Clustering par voisinage local

Les algorithmes de clustering sont très dépendants de la distance utilisée. Dans le cas d'une distance euclidienne utilisée dans un K-Means les clusters sont des "patatoïdes" convexes, ce qui n'est pas toujours pertinent. Dans la figure suivante par exemple on aurait naturellement envie qu'un K-means lancé avec $K = 2$ partitionne les données en deux parties homogènes, ce que ne fait pas un K-means lancé avec $K = 2$ et une distance euclidienne.

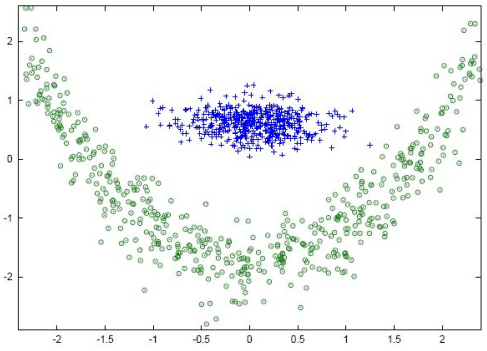

FIGURE 1 –

Une possibilité consiste à définir une distance à partir du voisinage local entre élements de la base d'exemples. A partir d'une base de données X de N exemples, on commence par identifier les p plus proches voisins (au sens de la distance euclidienne) de chacun des exemples. On note $V(n)$ l'ensemble des voisins du noeud n, c'est un ensemble de cardinal p.
L'ensemble des exemples peut être vu comme un graphe de voisinage, dont les noeuds correspondent aux exemples et il existe un arc (orienté) entre deux noeuds n_1 et n_2 si l'exemple correspondant au noeud n_2 est un des p plus proches voisins de l'exemple correspondant au noeud n_1. La distance entre deux noeuds non voisins est supposée infinie. Cette distance (la distance euclidienne ou l'infini suivant que les deux noeuds sont voisins ou pas) est notée la distance de niveau 0, notée d^0, entre deux noeuds. A noter que cette distance n'en est en fait pas une, car elle n'est pas symétrique.

A partir de ce graphe et de cette première distance, de niveau 0, on définit une distance de voisinage de la façon suivante. On commence par définir la distance "de niveau 1" entre deux noeuds n_i et n_j, c'est à dire la distance pour aller d'un noeud à un autre dans le graphe en passant par un seul noeud intermédiaire. Elle est est égale à :

$$d^1(i,j) = argmin_{k/n_k \in V(n_i) \bigcap V(n_k)} d^0(n_i, n_k) + d^0(n_k, n_j)$$

Cette distance est également calculable par $d^1(i,j) = argmin_k d^0(n_i, n_k) + d^0(n_k, n_j)$ si on a posé que $n_v \notin V(n_u) \Rightarrow d^0(n_u, n_v) = \infty$.

Puis on définit la distance de niveau 2, via deux noeuds intermédiaires : $d^2(i,j) = argmin_k d^0(n_i, n_k) + d^1(n_k, n_j)$
On peut ensuite définir la distance de niveau 3, 4, etc. Enfin, la distance de voisinage est au final définie par : $d_{Vois}(n_i, n_j) = min_{l=0,\ldots,\infty} d^l(n_i, n_j)$

Q 3.1 On vous demande d'écrire le code python permettant de calculer une matrice de distances de voisinage pour un ensemble d'exemples passé en paramètres. Le nombre de voisins à considérer (p) est également passé en paramètres. Vous pouvez considérer que vous disposez d'une classe *knn* comme celle utilisée en TME ou toute autre fonction auxiliaire qui peut vous aider.

Q 3.2 On vous demande d'écrire le code d'un algorihme des K-Means exploitant la distance précédente.