-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 $\mathbf{1}_{\text{cond}}$ la fonction qui renvoie 1 si la condition est vraie et 0 sinon.

Exercice 1 (7 points) – Questions indépendantes

Q 1.1 Définir en quelques phrases les notions suivantes : 1) Sur-apprentissage et sous-apprentissage, 2) Apprentissage non supervisé, 3) Descente de gradient, 4) Hinge loss

Q 1.2 Soit un perceptron de poids $(w_0, w_1, w_2) = (2, 1, 1)$.

Q 1.2.1 Quelle est la dimension de l'entrée ? Tracer la frontière de décision.

Q 1.2.2 Le(s)quel(s) de ces vecteurs de poids représente(nt) le même hyper plan séparateur que ce perceptron ? La même classification ? : 1) $(1, 0.5, 0.5)$, 2) $(200, 100, 100)$, 3) $(\sqrt{2}, \sqrt{1}, \sqrt{1})$, 4) $(-2, -1, -1)$

Q 1.3 Proposer un réseau de neurones séparant le couple de points : $\{(0, 0), (1, 1)\}$ et $\{(1, 0), (0, 1)\}$.

Exercice 2 (5) – GentleBoost

Nous allons étudier dans la suite un algo de boosting utilisant un *stump* comme classifieur faible : $f(\mathbf{x}) = a\mathbf{1}_{<\mathbf{w},\mathbf{x}>+b>0} + c$, où a, b et c sont des réels, $\mathbf{w}$ un vecteur de la dimension de $\mathbf{x}$ et $< \mathbf{w}, \mathbf{x} >$ le produit scalaire. On considère la fonction de coût suivant : $L(f) = \sum_{i=1}^{n} \gamma^i (y^i - f(\mathbf{x}^i))^2$, où γ^i est le poids associé à l'exemple $\mathbf{x}^i$: $\gamma^i > 0$ et $\sum_{i=1}^{n} \gamma^i = 1$.

Q 2.1 Rappeler ce qu'est un algorithme de boosting et pourquoi on utilise un classifieur faible.
Supposons dans la suite le vecteur $\mathbf{w}$ fixé (en pratique il est tiré au hasard à chaque itération).

Q 2.2 Calculer les dérivées partielles de f par rapport à a et c.

Q 2.3 Calculer les dérivées partielles de $L(f)$ par rapport à a et c. Vous pouvez faire intervenir l'ensemble $X^{\mathbf{w}^+} = \{\mathbf{x}^i | < \mathbf{w}, \mathbf{x}^i > +b > 0\}$.

Q 2.4 Déduire la valeur optimale pour c. Donner la valeur optimale de a. Commenter.

Q 2.5 Comment calculer b ?

Exercice 3 (8 points) – Classification multiclasse

Les labels ne sont plus binaire mais au nombre de C, $\mathcal{Y} = \{1, \cdots, C\}$. On suppose une matrice des données $X \in \mathbb{R}^{n \times d}$ et les labels associés $Y \in \mathcal{Y}^n$. Nous utiliserons dans la suite des classifieurs linéaires de paramètre $\mathbf{w} \in \mathbb{R}^d$.

Q 3.1 Stratégie un contre tous pour le perceptron
Cette stratégie consiste à construire C perceptrons $\{f_j\}_{j=1,\cdots,C}$ où le j-ème classifieur est chargé de distinguer les exemples de la classe j des exemples de toutes les autres classes. Ainsi pendant

l'apprentissage chaque perceptron f_j est un modèle binaire appris sur la base de données ré-étiquetée comme suit : les points de la classe j prennent temporairement la classe $+1$ et tous les autres points la classe -1.
L'attribution d'un exemple à une classe se fait selon la règle suivante : $\hat{y} = \arg\max_j f_j(x)$: on attribue la classe correspondant au classifieur qui donne le meilleur score. Dans la suite on stockera tous les poids de tous les classifieurs dans une matrice W de taille $d \times C$: chaque colonne j représente le vecteur de poids du j-ème classifieur, chargé de reconnaître la classe j.

Q 3.1.1 Donner le code de la classe $PMulti$ qui permet de stocker la matrice de poids et qui a une fonction $predict(X)$ qui permet de prédire la classe des exemples de la matrice X.

Q 3.1.2 Donner le code de la fonction $fit(X, Y)$ qui permet d'apprendre la matrice de poids W, en apprenant pour chaque classe le perceptron qui la distingue des autres. Vous utiliserez pour cela l'algorithme du perceptron usuel stochastique : à chaque itération, x^i est tiré au hasard et tous les classifieurs sont mis à jour selon l'erreur qu'ils font.

Q 3.2 Stratégie un contre un
En suivant une logique proche, il est possible de créer un classifieur multiclasse basé sur l'ensemble des classifieurs élémentaires biclasses, c'est à dire sur des classifieurs qui séparent les classes deux à deux : f_{ij} sépare l'ensemble d'exemples de la classe i de l'ensemble d'exemples de la classes j. Pour chaque exemple à classifier, il faut alors faire voter tous les classifieurs et trouver la classe majoritaire.

Q 3.2.1 Combien de classifieurs sont nécessaires pour construire un modèle pour C classes ? Quelle est la dimension de la matrice de tous les poids W ?

Q 3.2.2 Donner le code de la fonction fit et $predict$ pour ce modèle dérivé.

Q 3.3 Soft-max
On se place dans le cas d la stratégie un contre tous, avec C classifieurs $\{f_i\}_{i=1,\cdots,C}$ linéaires. La prédiction est donc $\hat{y} = \arg\max_j f_j(x) = \arg\max_j < \mathbf{w}^j, \mathbf{x} >$. Le coût soft-max est le coût

$$s_j(\mathbf{x}) = \frac{e^{f_j(\mathbf{x})}}{\sum_{i=1}^{C} e^{f_i(\mathbf{x})}}$$

.
Le modèle est appris par minimisation du coût logarithmique : $\sum_{i=1}^{n} log(s_{y^i}(x^i))$ où $s_{y^i}(x^i)$ représente la sortie softmax de la classe de l'exemple x^i.

Q 3.3.1 Montrer que $\sum_{j=1}^{C} s_j(\mathbf{x}) = 1$.

Q 3.3.2 Expliquez en quelques mots pourquoi ce critère d'apprentissage vous paraît raisonnable.

Q 3.3.3 Pourquoi les C classifieurs ne peuvent pas être appris séparément les uns des autres ?

Q 3.3.4 On note $C^i(W) = log(s_{y^i}(\mathbf{x}^i))$ le critère local, c'est-à-dire restreint à un exemple. Exprimer le gradient $\frac{\partial C^i(W)}{\partial s_j}$ de $C^i(W)$ par rapport à la sortie soft-max du j-ème classifieur s_j (où on oublie la dépendance à $\mathbf{x}^i$ pour alléger les notations). Vous distinguerez le cas où l'exemple est de la classe j de celui où il n'est pas.

Q 3.3.5 Exprimer le critère $C^i(W)$ en fonction des sorties des classifieurs f_i.

Q 3.3.6 Exprimer le gradient de $C^i(W)$ par rapport à la sortie f_j d'un classifieur donné.

Q 3.3.7 Exploiter ces résultats pour exprimer la dérivée $\frac{\partial C^i(W)}{\partial w_k^j}$ du critère local par rapport à un poids w_k^j du j-ème classifieur, puis la dérivée $\frac{\partial C(W)}{\partial w_k^j}$ du critère global par rapport à un poids w_k^j du j-ème classifieur.

Q 3.3.8 Donner le code correspondant à la minimisation du critère global par un algorithme de gradient.