Numéro d'anonymat :

Contrôle de Connaissance
Licence 2 Informatique - Vie artificielle - Université Paris-Sud

Mardi 9 Janvier 2018
Durée : 2h00
Documents autorisés : supports et notes de cours

Partie I : Apprentissage supervisé

1 Perceptron [10 pts]

On cherche à écrire un algorithme permettant de distinguer entre les courriels importants des autres. On considérera donc une tâche de classification binaire. Pour entraîner notre algorithme, nous disposons de N données d'apprentissage : $(\boldsymbol{x}_i, \tilde{y}_i)_{i=1,..,N}$ où $\boldsymbol{x}_i$ représente la i-ème données et $\tilde{y}_i$ son étiquette ($+1$ si le courriel est important, -1 sinon).

1.1 Questions de cours [2 pts]

1. En quoi consiste le modèle du perceptron (vous définierez les variables d'entrées, de sorties et les paramètres).
2. Dans quel cas est-on certain que l'algorithme du perceptron binaire converge (arrive à classifier toutes les données de l'ensemble d'apprentissage) ?
3. Lorsqu'on dispose d'un ensemble d'apprentissage et d'un ensemble de validation, comment fait-on pour décider à quelle époque l'algorithme a convergé (à quelle époque correspondent les meilleurs paramètres) ?
4. Citer deux raisons pour lesquelles la version probabiliste du perceptron est mieux que la version "classique".

1.2 Le perceptron binaire "classique" [4 pts]

On s'attaque ici au problème sous l'angle du perceptron binaire non probabiliste. Nous supposerons que nous disposons des données suivantes :

$$\tilde{y} = +1 : \left\{ \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\} \text{ et } \tilde{y} = -1 : \left\{ \begin{pmatrix} 1 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}$$

et que le vecteur de poids initial est donné par $\mathbf{w'}^t = (-2, 1, 1)$

1. Indiquer quelles sont les données bien classées et les données mal classées et les représenter sur un graphe ainsi que la frontière de décision.
2. Effectuer une époque sur ces données (en les prenant dans l'ordre que vous voulez) et indiquer la nouvelle valeur des paramètres du percetron.
3. Sur le graphe précédent, représenter la nouvelle frontière de décision.
4. L'algorithme peut-il converger sur ces données ?

1.3 Perceptron probabiliste [2 pts]

On souhaite implémenter un perceptron binaire probabiliste mais on veut s'assurer que les poids ne puissent pas devenir trop grand. On va donc considérer la fonction de coût suivante composée de l'entropie croisée (EC) à laquelle on ajoute la somme de tous les poids au carré :

$$E_{\text{cout}}(\boldsymbol{w}) = E_{\text{EC}}(\boldsymbol{w}) + \lambda \sum_{i,j} w_{ij}^2 \tag{1}$$

(l'entropie croisée dépend évidement de la donnée que l'on considère et pour la suite, la fonction de coût sera prise séparement pour chaque donnée).

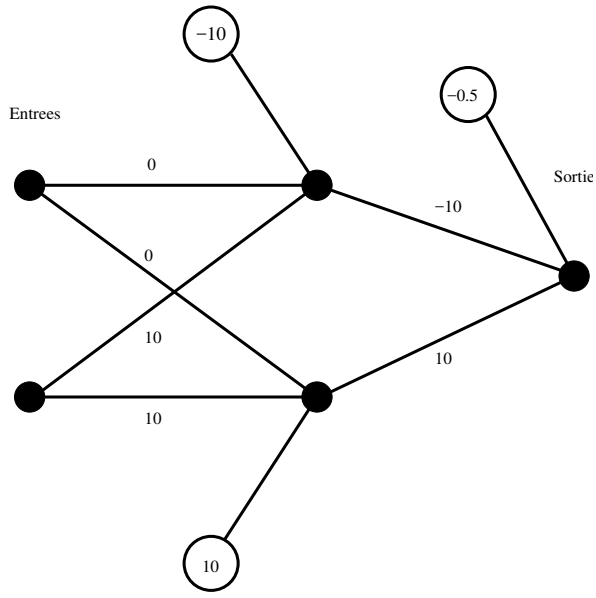
1. Calculer le gradient de cette fonction de coût (vous expliciterez également le gradient de l'entropie croisée).
2. Expliquer comment, à l'aide d'un ensemble de validation comment il est possible de fixer la valeur de λ.

1.4 Perceptron multicouches [2 pts]

On se rend compte que l'on va devoir utiliser un réseaux de neurones à plus d'une couche car on a rencontré les données suivantes :

$$\tilde{y} = +1 : \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\} \text{ et } \tilde{y} = -1 : \left\{ \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ -3 \end{pmatrix} \right\}$$

1. Expliquer pourquoi on ne peut pas utiliser un percetron binaire classique.
2. Montrer, à l'aide d'une représentation graphique, que le modèle suivant permet de classifier correctement les données :



où les fonction d'activations sont des sigmoïdes à chaque étape (également en sortie) et la valeur des poids est indiquée sur les liens (les biais correpondent aux cercles vides). On considérera que la sigmoïde $\sigma(x)$ vaut 1 pour des x supérieurs à 5 et 0 pour des x inférieurs à -5.
Pour la représentation graphique, on pourra d'abord regarder les frontières données pour chacun des neurones de la couche intermédiaire pour ensuite représenter ces données dans un nouveau plan en deux dimensions correspondant à la couche intermédiaire.

Partie II : Apprentissage par renforcement

2 Les bandits [3 pts]

Rappeler comment fonctionne le modèle des bandits (on considérera un ensemble de n bandits) :

1. Comment définit-on les récompenses.
2. Comment s'effectue l'apprentissage des paramètres des bandits.
3. Comment choisit-on une action.
4. Que se passe-t-il lorsqu'on choisit une valeur trop grande pour ϵ dans stratégie ϵ-glouton (ou greedy) ?
5. Qu'est-ce que le dilemme exploitation/exploration ?

3 Equation de Bellman [3 pts]

1. Rappeler l'équation de Bellman et expliquer ses différents termes.
2. Montrer comment il est possible d'écrire cette équation sous forme matricielle (vous définierez clairement les différents vecteurs/matrices).
3. Quelle est la solution des équations de Bellmann ?

4 Application [4 pts]

On considère ici un environnement de type GridWorld. Sur la figure (1), l'image de gauche montre les récompenses obtenues en fonction de la case sur laquelle l'agent **arrive**, et la droite, une numérotation des états (dans notre cas, une case correspond à un état).

0	0	0	10
0	1	−2	4
0	0	−2	−2
0	0	−2	−2

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

FIGURE 1 – Sur la gauche, les récompenses, sur la droite une énumération des états

L'agent évoluant dans ce GridWorld a la possibilité de se déplacer selon quatre directions : haut, bas, gauche et droite (on supposera que aller vers un mur nous fait rester sur la même case). La case 3 (dont la récompense est 10) est terminale : une fois arrivé sur cette case, le "jeu" est fini. On prendra comme taux d'apprentissage $\alpha = 1/2$ et le paramètre pondérant les gain futurs $\gamma = 1$.

1. On considère que l'agent part de la case 13 et effectue les actions suivantes : droite, haut, haut, haut, droite. En prenant comme point de départ que toutes les fonctions $Q(s, a)$ sont initialisées à zéro, donner les valeurs de $Q(s, a)$ après avoir effectué ces actions selon l'algorithme Q-learning **ET** SARSA.
2. L'agent procède de nouveaux aux mêmes actions (il part de la case 13 également), mais cette fois-ci les $Q(s, a)$ ont la valeur qui a été obtenu à la question précédente. Calculer les nouvelles valeurs de $Q(s, a)$.

Rappel :

— Q-learning : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma \mathrm{max_a Q(s_{t+1}, a)} - \mathrm{Q(s_t, a_t)}\right)$

— SARSA : $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$