ARF Examen

Exercice 1 (8 points) – **QCM**

Pour chaque item, une seule réponse est correcte. Donner vos réponses, **sans justifier**, sous la forme 1-b par exemple si vous souhaitez choisir la réponse b à la question 1. Chaque bonne réponse apporte 0.25 points, chaque mauvaise retire 0.25 points à votre note.

1. Le problème du XOR peut être traité avec un réseau de neurones à trois couches avec activations linéaires.
   a : Vrai, b : Faux
2. Un mauvais score en apprentissage peut être dû à un sur-apprentissage du modèle.
   a : Vrai, b : Faux
3. Augmenter l'hyper-paramètre C du SVM revient à être moins tolérant avec les erreurs de classification en apprentissage.
   a : Vrai, b : Faux
4. Diriez-vous qu'un classifieur prédisant toujours la classe majoritaire sur un ensemble d'entraînement est plutôt en situation de :
   a : sous-apprentissage, b : sur-apprentissage
5. La back-propagation est une méthode permettant la mise à jour des valeurs des états d'un MDP.
   a : Vrai, b : Faux
6. Un coefficient de discount < 1 est indispensable pour garantir la convergence de l'algorithme policy iteration.
   a : Vrai, b : Faux
7. Les paramètres w d'une régression moindres carré $||Xw^T - Y||^2 + \lambda||w||^2$ peuvent toujours être obtenus en prenant $w = X^T Y(X^T X + \lambda I)^{-1}$, avec X la matrice des données (une donnée par ligne), Y celle des labels correspondants et λ un terme de régularisation > 0.
   a : Vrai, b : Faux
8. Dans le cadre d'un problème de maximisation, vaut-il mieux que la fonction à optimiser soit :
   a : convexe, b : concave, c : ni l'un, ni l'autre
9. Un problème d'optimisation convexe se caractérise par le fait que sa dérivée est toujours positive.
   a : Vrai, b : Faux
10. Le boosting est une technique permettant de combiner intelligemment les sorties de classifieurs appris sur des sous-ensembles de données d'entraînement différents.
    a : Vrai, b : Faux
11. Un score de 60% de bonne classification en apprentissage pour un score de 55% en test vaut mieux qu'un score de 100% en apprentissage et 50% en test.
    a : Vrai, b : Faux
12. Le classifieur bayésien naïf est optimal.
    a : Vrai, b : Faux
13. Soit le problème d'opmisation sous-contraintes suivant :

$$min_w \sum_{(x,y)\in(X,Y)} (xw^T - y)^2 \qquad s.t. \sum_{x\in X} (xw^T)^2 < 10$$

    Selon la méthode des multiplicateurs de Langrange, ce problème peut être traité en considérant la formulation suivante :

$$min_w max_\gamma \sum_{(x,y)\in(X,Y)} (xw^T - y)^2 + \rho\gamma((\sum_{x\in X}(xw^T)^2) - 10) \qquad s.t.\gamma \geq 0$$

avec ρ valant :
a : 1, b :-1

14. Le gradient ∇_w de $||wX^T - Y^T||^2$ vaut :
a : $2X(wX^T - Y^T)$, b : $2X^T(wX^T - Y^T)$, c : $2X(wX^T - Y^T)^T$, d : $2X^T(wX^T - Y^T)^T$
15. La matrice de Gram est la généralisation des n-grams aux ensembles vectoriels
a : Vrai, b : Faux
16. Le “Kernel-trick” se résume au remplacement d'un produit scalaire dans l'espace des données par une fonction noyau.
a : Vrai, b : Faux
17. Une fonction noyau peut retourner un score nul.
a : Vrai, b : Faux
18. Soient les points $X = \{(1,2),(1,4),(3,5),(4,5)\}$ associés aux labels $Y = \{-1,-1,1,1\}$. Est-il possible de trouver des paramètres w permettant à la machine $f(x) = sign(< w, x >)$ de classer correctement tous les point de X ?
a : oui, b : non
19. L'optimisation de
$$min_w \sum_{(x,y)\in(X,Y)} (wx^T - y)^2$$
peut se faire par algorithme itératif de mise à jour des poids selon la règle :
$$w_i \leftarrow w_i - 2\epsilon \sum_{(x,y)\in(X,Y)} x_i(wx^T - y)$$
pour toute dimension i du problème, avec ϵ le pas d'apprentissage et x_i la i-ième composante du vecteur x.
a : Vrai, b : Faux
20. Un noyau polynomial peut permettre de délimiter des classes ellipsoïdales.
a : Vrai, b : Faux
21. Dans un réseau de neurones, les poids optimaux de la couche i ne dépendent que des poids de la couche $i+1$ (i.e., pas des poids des autres couches du réseau, ce qui permet de mettre en place l'algorithme de retro-propagation du gradient).
a : Vrai, b : Faux
22. Un problème de bandit est modélisé par un MDP à 1 état.
a : Vrai, b : Faux
23. La régression logistique est bien adaptée pour traiter des problèmes de classification linéaire.
a : Vrai, b : Faux
24. La régression moindres carrés est bien adaptée pour traiter des problèmes de classification polynomiaux.
a : Vrai, b : Faux
25. Soit la projection $\phi(x) = (x_1^2, x_2^2, x_1x_2, x_1, x_2, 1)$ pour des données en deux dimensions. Est-ce possible de traiter le problème du XOR avec une telle projection et une machine de classification de type $f(x) = sign(< w, \phi(x) >)$?
a : Oui, b : Non

26. Pour obtenir une solution parcimonieuse (i.e., de nombreux paramètres à 0) dans un problème de classification, il vaut mieux utiliser (avec $||w||_p$ la norme p de w) :
    a : un terme de régularisation $\lambda||w||_\infty$, b : un terme de régularisation $\lambda||w||_1$, c : un terme de régularisation $\lambda||w||_2$
27. Les classifieurs $sign(1.5x_1 - x_2)$ et $sign(6x_1 - 4x_2)$ ont des performances identiques en apprentissages mais pas en test.
    a : Vrai, b : Faux
28. La cross-validation est une technique pour réduire la complexité des problèmes d'apprentissage.
    a : Vrai, b : Faux
29. Quel que soit le point de départ du problème de minimisation SVM, les paramètres convergent vers les mêmes valeurs.
    a : Vrai, b : Faux
30. L'algorithme Q-Learning permet de converger toujours vers la politique optimale.
    a : Vrai, b : Faux
31. Les problèmes de classification en petite dimension se traitent généralement plus facilement par des machines linéaires que les problèmes en plus grande dimension.
    a : Vrai, b : Faux
32. Si le risque théorique d'un classifieur f s'écrit $\int_{(\mathcal{X},\mathcal{Y})} p(x,y)\delta(f(x),y)dxdy$, comment s'écrit son risque empirique sur un ensemble d'apprentissage (X,Y) de N exemples ?
    a : $\sum_{(x,y)\in(X,Y)} p(x,y)\delta(f(x),y)$, b : $\frac{1}{N}\sum_{(x,y)\in(X,Y)} p(x,y)\delta(f(x),y)$, c : $\frac{1}{N}\sum_{(x,y)\in(X,Y)} \delta(f(x),y)$

Exercice 2 (5 points) – **ReLu**

La fonction d'activation ReLU (rectified linear unit) est définie de la façon suivante : $f(x) = x^+ = max(0,x)$.

Q 2.1 Caractéristiques

Q 2.1.1 Dessiner sur le même schéma la fonction ReLU, la fonction usuelle tangente hyperbolique et la fonction identité.

Q 2.1.2 Donner la dérivée de la fonction ReLU. Son optimisation peut-elle poser problème ? Proposer une solution dans ce cas.

Q 2.1.3 Comparer la ReLU à la tangente hyperbolique ; discuter en particulier quel(s) problème(s) elle peut permettre de résoudre. Pourquoi ne pas employer la fonction identité ?

Q 2.2 Soit le réseau de neurones dont l'architecture est la suivante :

- une entrée $\mathbf{x} \in \mathbb{R}^{256}$
- une 1ère couche linéaire à 64 sorties $z_i^1, i \in \{1,\ldots,64\}$ (paramètre W^1)
- une fonction d'activation ReLU dont les sorties seront notées z_i^2
- une couche linéaire à 2 sorties $z_i^3, \in \{1,2\}$ (paramètre W^3)
- une fonction d'activation tangente hyperbolique dont le vecteur de sortie sera noté $\hat{\mathbf{y}}$.
- un coût aux moindres carrés $L(\mathbf{y},\hat{\mathbf{y}})$ entre la sortie attendue $\mathbf{y}$ et la sortie calculée par le réseau $\hat{\mathbf{y}}$.

Q 2.2.1 Quelles sont les dimensions de W^1 et de W^3 ?

Q 2.2.2 Donner l'expression du coût aux moindres carrés et l'expression de la sortie $\hat{\mathbf{y}}$ en fonction de $\mathbf{x}$ (et des fonctions f et $tanh$ et des paramètres du réseau, sous forme matricielle ou exacte).

Q 2.3 Apprentissage du réseau. Soit $M^h(\mathbf{x})$ la transformation d'une entrée x de la couche h du réseau : $z^h = M^h(z^{h-1})$. On pose $\delta_j^h = \frac{\partial L}{\partial z_j^h}$.

Q 2.3.1 A quoi correspondent les δ_j^h en termes intuitifs ?

Q 2.3.2 Quelle est la formule qui relient δ_j^h à tous les δ_i^{h+1} d'une couche à une autre ?

Q 2.3.3 Quels sont les gradients utiles pour l'apprentissage d'un tel réseau de neurones ? Quel est l'algorithme utilisé ?

Q 2.3.4 (bonus) Donner l'expression mathématique exacte des gradients du réseau définis à la question précédente (rappel : $tanh'(x) = 1 - tanh(x)^2$).

Exercice 3 (7.5 points) – **RL**

On s'intéresse au jeu classique "frog" dans un contexte simplifié : le monde est constitué d'une grille de 5 cases par 5 cases ; une grenouille se trouve sur la dernière ligne du monde (ligne 5), au milieu ; elle peut faire 3 actions : monter (m), descendre (d), rester sur place (r) ; des voitures circulent sur chaque ligne intérieure du monde (ligne 2 à 4), de gauche à droite pour les lignes 2 et 4, de droite à gauche pour la ligne 3. L'objectif de la grenouille est d'atteindre la ligne 1 sans se faire écraser. On supposera que : les voitures ont toutes une vitesse constante d'une case par pas de temps - tout comme la grenouille ; qu'il n'y a qu'une voiture par ligne et que le monde est cyclique (lorsque la voiture atteint la dernière case sur le bord elle apparaît de l'autre côté au prochain coup). A l'état initial, les voitures sont positionnées aléatoirement, une par ligne (de 2 à 4).

Q 3.1 Dans cette question on simplifie le problème : la grille ne comporte plus que 3 lignes et 3 colonnes, la grenouille est située toujours au milieu (ligne 3, colonne 2) et il n'y a plus que une voiture en ligne 2 qui se déplace de gauche à droite. La ligne à atteindre pour la grenouille est la ligne 1.

Q 3.1.1 Décrire tous les états possibles et les transitions entre états par un schéma. Indiquer les états finaux.

Q 3.1.2 Proposer des récompenses pour chaque couple d'état-action (ou triplet état-action-état à votre convenance).

Q 3.1.3 Rappeler l'algorithme de Value-itération et la différence principale avec l'algorithme de Policy-itération.

Q 3.1.4 Détailler pour quelques itérations l'algorithme de value-itération, en précisant pour chaque itération la politique optimale associée.

Q 3.2 Dans cette question, on revient dans le cadre générique du jeu.

Q 3.2.1 Proposer une modélisation du monde (la plus simple possible et avec le moins d'états possibles) en MDP : quels sont les états, les transitions, les états finaux ? Combien d'états au total ? Indications : vous pouvez utiliser en particulier des variables entières pour décrire les états dénotant les positions de chaque objet du jeu.

Q 3.2.2 Proposer un système de récompense pour le jeu $r(s, a)$ avec s un état et a une action.

Q 3.2.3 Quelle serait l'effet sur la politique optimale de poser une récompense strictement positive pour l'action r ? Une récompense nulle ?

Q 3.2.4 Les valeurs précises des récompenses associées aux états finaux sont-elles importantes ?

Q 3.3 Dans le vrai jeu de la grenouille, le monde n'est plus discret : les voitures ont une vitesse non constante, leur position n'est plus une case mais une coordonnée réelle, plusieurs voitures peuvent être sur la même ligne et elles apparaissent de manière aléatoire ; seule la grenouille se déplace de case en case. Proposer une modélisation du problème (une description des états). Est-il encore envisageable de résoudre le problème par value-iteration ? Proposer un autre algorithme pour résoudre le problème.