Modèles d'Apprentissage Artificiel

Master 2 : Intelligence Artificielle

Pr. Mustapha Bourahla

Exercice n°1

a) Quelles sont les sources du problème d'apprentissage?

b) Donner une définition du modèle d'apprentissage

c) Expliquer comment les observations, les actions et le feedback peuvent influer sur la difficulté de l'apprentissage

Solutions

a) Les sources du problème d'apprentissage sont:

   1) Le nombre de cycles à effectuer pour l'apprentissage.
   2) Les ressources de calcul nécessaires durant chaque cycle à l'agent pour réviser sa stratégie et choisir une action.

b) Un modèle d'apprentissage est un cadre formel donnant une mesure de ces deux sources de complexité.

c) Les observations, les actions et le feedback peuvent influer sur la difficulté de l'apprentissage: L'espace des observations est immense, un agent doit être capable d'extrapoler (inférer à partir d'un nombre restreint des observations), les valeurs de certains attributs peuvent être imprécises, erronées, ou encore absentes, l'agent doit être aussi capable d'interpoler (inférer à partir d'une situation incertaine). Les actions ont le problème que l'espace des décisions (actions) possède une structure combinatoire et chaque action peut influer le cours des observations. Le problme du feedback est que la représentation peut être utilisée comme stratégie pour étiqueter de nouvelles observations selon la structure induite.

Exercice n° 2

a) Donner avec explication les composants du problème de l'apprentissage supervisé de porte logique XOR

b) TP: Ecrire un programme Python pour montrer ces composants

Solutions

Les composants du problème de l'apprentissage supervisé de porte logique XOR:

1) Espace d'instances X = {{0,0},{0,1},{1,0},{1,1}}

2) Espace de décisions Y = {{0},{1},{1},{0}}

3) Un schéma de représentation (Ω, f) engendrant l'espace H_Ω mesuré par f:

Ω = Réseaux de neurones (exemple: FC(2,3), Activation(tanh), FC(3,1), Activation(tanh))

H_Ω = toutes les hypothèses associées => Y = X * W + B

f : Ω -> R (le nombre de neurones par exemple)

4) Un espace de fonctions cibles H_Ω: toutes les fonctions cibles (les valeurs des poids donnant les fonctions désirées)

Solutions

5) La fonction de perte : l = mse =

```
import numpy as np
# loss function and its derivative
def mse(y_true, y_pred):
  return np.mean(np.power(y_true-y_pred, 2))
def mse_prime(y_true, y_pred):
   return 2*(y_pred-y_true)/y_true.size
```

Solutions

L'exécution du programme nous donne les hypothèses intermédiaires et le critère d'arrêt nous donne l'hypothèse cible

Si nous changeons le schéma on aura autres hypothèses (donc c'est un ensemble)

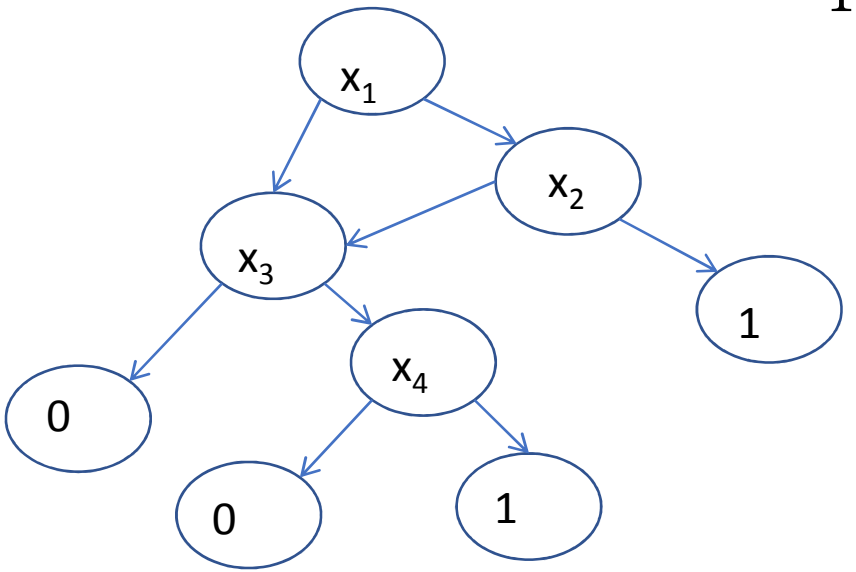
Exercice n° 3

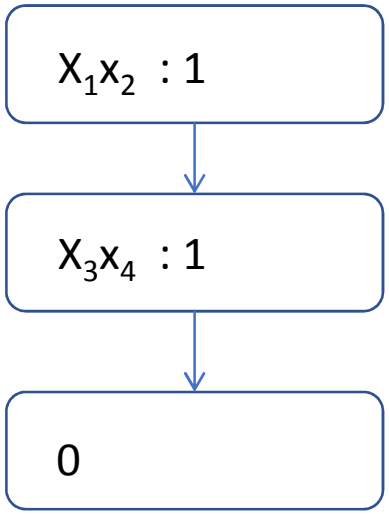
- Donner l’arbre de décision et son correspondante liste de décision de la formule:

$$x_1 \wedge x_2 \vee x_3 \wedge x_4$$

Solutions

- Donner l'arbre de décision et son correspondante liste de décision de la formule: $x_1 \wedge x_2 \vee x_3 \wedge x_4$





Exercice n° 4

a) Quelle est la différence entre une requête d'appartenance et une requête d'équivalence?

b) Si pour toute instance (x_1, x_2, x_3), nous avons $h_1((x_1,x_2,x_3)) = x_1$ et $h_2((x_1,x_2,x_3)) = x_1 \wedge x_2$, quelle est l'hypothèse (le concept) le plus spécifique,

c) Si $h(x_1,x_2,x_3) = x_1 \wedge x_2$ et la requête est $h((0,1,1)) = 0$ quel est le type de cette requête? et si $h^*((0,1,1)) = 1$, quelle serai la réponse à cette requête? Si le contre exemple est (0,1,0) mettre à jour h.

Solutions

a) La différence est que avec une requête d'appartenance MQ(x) = oui si $h^*(x) = 1$, et sinon MQ(x) = non et une requête d'équivalence EQ(h) = oui si $h = h^*$, et EQ(h) = non sinon

b) l'hypothèse (le concept) le plus spécifique est $h_2((x_1,x_2,x_3)) = x_1 \wedge x_2$ car $h_2(x) \leq h_1(x)$ pour toute instance (x_1, x_2, x_3)

c) Si $h(x_1,x_2,x_3) = x_1 \wedge x_2$ et la requête est h((0,1,1)) = 0 alors le type de cette requête est d'appartenance. Si $h^*((0,1,1)) = 1$ alors la réponse à cette requête serait oui, Si le contre exemple est (0,1,0) alors la mise à jour de h est $h(x_1,x_2,x_3) = x_2$.

Exercice n° 5 (TP)

- Comme travail pratique, essayer de reprogrammer l'apprentissage de la porte logique XOR en utilisant l'algorithme d'apprentissage avec conseil d'experts "Weighted Majority" (la majorité pondérée). Utiliser trois experts:
  1. $h_1((0,0,1),(0,1,1),(1,0,1),(1,1,0))$
  2. $h_2((0,0,0),(0,1,1),(1,0,1),(1,1,0))$
  3. $h_3((0,0,0),(0,1,1),(1,0,1),(1,1,1))$
- Expliquer les résultats de l'exécution

Solutions

```
x_train_1 = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train_1 = np.array([[[1]], [[1]], [[1]], [[0]]])

x_train_2 = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train_2 = np.array([[[0]], [[1]], [[1]], [[0]]])

x_train_3 = np.array([[[0,0]], [[0,1]], [[1,0]], [[1,1]]])
y_train_3 = np.array([[[0]], [[1]], [[1]], [[1]]])
```

Solutions

```
# network1
net1 = Network()
net1.add(FCLayer(2, 3))
net1.add(ActivationLayer(tanh, tanh_prime))
net1.add(FCLayer(3, 1))
net1.add(ActivationLayer(tanh, tanh_prime))
```

Solutions

```
# network2
net2 = Network()
net2.add(FCLayer(2, 3))
net2.add(ActivationLayer(tanh, tanh_prime))
net2.add(FCLayer(3, 1))
net2.add(ActivationLayer(tanh, tanh_prime))
```

Solutions

```
# network3
net3 = Network()
net3.add(FCLayer(2, 3))
net3.add(ActivationLayer(tanh, tanh_prime))
net3.add(FCLayer(3, 1))
net3.add(ActivationLayer(tanh, tanh_prime))
```

Solutions

```
# train
net1.use(mse, mse_prime)
net1.fit(x_train_1, y_train_1, epochs=100, learning_rate=0.1)
net2.use(mse, mse_prime)
net2.fit(x_train_2, y_train_2, epochs=100, learning_rate=0.1)
net3.use(mse, mse_prime)
net3.fit(x_train_3, y_train_3, epochs=100, learning_rate=0.1)
```

Solutions

```
t = 0
w = [1, 1, 1]
while t < 4:
        y_hat = (np.array([w[0]]) * net1.predict(x_train_1[t]) +
                 np.array([w[1]]) * net2.predict(x_train_2[t]) +
                 np.array([w[2]]) * net3.predict(x_train_3[t])) / np.sum(w)
        w[0] = w[0] * np.exp(-0.7 * mse(y_true=y_hat, y_pred=net1.predict(x_train_1[t])))
        w[1] = w[1] * np.exp(-0.7 * mse(y_true=y_hat, y_pred=net2.predict(x_train_2[t])))
        w[2] = w[2] * np.exp(-0.7 * mse(y_true=y_hat, y_pred=net3.predict(x_train_3[t])))
        t = t+1
```

Solutions

print(w)

[0.5924887308231594, 0.9000798076295213, 0.7933110774899882]

Exercice n° 6

Expliquer la différence entre le modèle agnostique et le modèle PAC

Solutions

La différence entre le modèle agnostique et le modèle PAC est que dans le modèle “agnostique” il n’existe a priori aucune dépendance fonctionnelle entre une instance x et une décision y dans un exemple tiré. Dans le modèle d’apprentissage probablement approximativement correct (PAC), nous supposons qu’il existe une dépendance fonctionnelle gouvernée par une fonction cible $h^* \in H^*$.