université PARIS-SACLAY
FACULTÉ DES SCIENCES D'ORSAY

Introduction du modèle du réseau neuronal

Introduction au perceptron

Ces travaux pratiques visent à consolider les concepts abordés lors de la séance de cours portant sur l'introduction du modèle neuronal. Ils s'étendent jusqu'à l'application du modèle à un robot différentiel. Cette série d'exercices se conclut par la rédaction d'**un compte rendu accompagné de scripts** permettant d'obtenir les résultats mentionnés. Ce rendu devra être déposé sur le site Moodle du cours.

1 Perceptron : implantation d'une porte logique

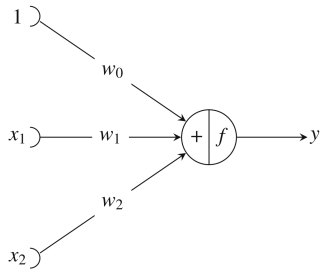


Fig. 1 : Perceptron à deux entrées

Le modèle du perceptron de la Fig. 1 est composé par deux entrées dénommées x_1 et x_2. Les valeurs de x_1 et x_2 peuvent être 0 ou 1. Assignez les poids de w_0,w_1 et w_2 de sorte à assurer une sortie y comme l'indiquent les tables de vérités ci-dessous :

Opérateur Logique OU (OR)		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Opérateur Logique ET (AND)		
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Op. Logique OU-Excl. (XOR)		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Tab. 1 : Tables de vérité des opérateurs logiques OU/ET et OU-Exclusif (XOR)

Afin de valider les poids proposés, implantez le modèle dans un script python en suivant la démarche proposée ci-dessous :

1. Définissez les variables x_1 et x_2 avec des valeurs 0 et/ou 1.
2. Dans une fonction retournant la valeur s, faites le calcul de l'expression :

$$s = w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 \tag{1}$$

3. Dans une deuxième fonction prenant en entrée s et retournant y, vérifiez les conditions suivants :

$$\begin{aligned} &\text{Si } s \geqslant 0\,, \quad y = 1 \\ &\text{Si } s < 0\,, \quad y = 0 \end{aligned}$$

4. **Effectuez dans le script tous les cas indiqués dans le Tab. 1 sous un tracé et rapportez les poids w_0,w_1 et w_2 pour l'opérateur OU et ceux pour l'opérateur ET.**

Questions

- Est-ce qu'on peut obtenir des poids satisfaisant les valeurs des opérateurs logiques ET/OU ?
- Est-ce qu'on peut représenter à l'aide d'un perceptron l'opérateur OU-Exclusif ? Oui, Non. Pourquoi ?

2 Exercice d'application

Implantez les perceptrons dans votre robot Thymio II. Utilisez deux capteurs de proximité et redirigez leur mesure vers les entrées du perception x_1 et x_2. Utilisez la sortie du perceptron, y comme il est défini par le tableau ci-dessous :

- **Comportement du robot**

  Le robot se déplace vers l'avant si les deux capteurs de distance situés à l'arrière détectent la présence d'un obstacle.

3 Perceptron analogique

Les perceptrons peuvent fonctionner dans un domaine dit "analogique". Cela implique que les entrées et la sortie du perceptron pourront évoluer dans une gamme autre que celle du tout-ou-rien (TOR). **Afin de comprendre et de constater le fonctionnement de ce type de perceptron, implantez le modèle de la Fig. 2 dans le robot Thymio II.**

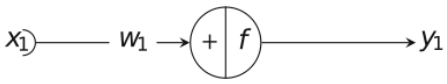


Fig. 2 : Modèle de perceptron à une entrée

- L'entrée du perceptron sera la valeur issue du capteur de proximité situé devant le robot.
- La sortie du perceptron sera dirigée vers :
  - la vitesse de moteurs afin de faire reculer le robot si un objet est détecté par le capteur.

Attention : La sortie du perceptron devra être proportionnelle,c-à-d, la vitesse de moteurs sera plus rapide si l'obstacle est plus proche.

- Implantez le modèle de perceptron à deux entrées (voir Fig. 3) en assignant des poids w_1 et w_2 différents. Illustrer que le capteur associé au poids plus fort a une plus ample influence sur la sortie.

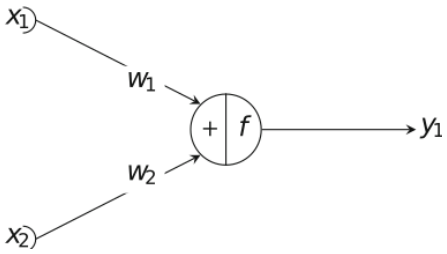


Fig. 3 : Modèle de perceptron à deux entrées

université PARIS-SACLAY

FACULTÉ DES SCIENCES D'ORSAY

Application des réseaux de neurones artificiels

Véhicule de Braitenberg

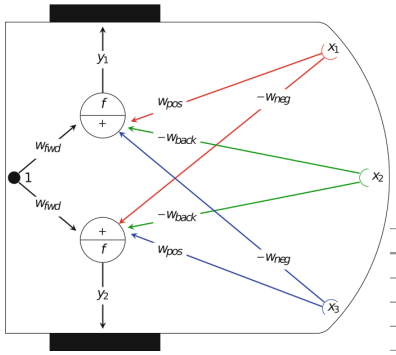
Les réseaux de neurones artificiels (ANN) ont diverses applications dans le domaine de la robotique. Nous nous intéressons particulièrement aux applications de la robotique mobile permettant la navigation d'un robot dans un environnement non connu. Cette application offre un large champs de possibilités et de variantes pour le design et l'implantation des ANN.

Ci-dessous, nous formalisons les objectifs ciblés par l'application :

La formulation d'un ANN faisant utilisation de trois capteurs de distance (obstacle) situés devant le robot Thymio II, L'implantation du comportement de navigation suivant : s'assurer que le robot se déplace vers l'avant dans l'absence totale d'obstacle, la détection d'un obstacle à partir du capteur centrale impose au robot de reculer, la détection d'un obstacle par le capter situé à gauche impose au robot de tourner à droite et la détection d'un obstacle par le capteur situé à droite impose au robot de tourner à gauche.

Description du modèle

Chaque neurone a quatre entrées, trois provenant de capteurs et une entrée unitaire. Cette dernière permet d'assurer que le robot avance en absence d'obstacles. La **fonction d'activation** du neurone doit être non linéaire afin de limiter la vitesse d'avancement et du recul. Lors de l'implantation, il est nécessaire d'identifier le poids nécessaire pour obtenir une entrée similaire à celles associées aux capteurs. Les valeurs prévenantes des capteurs devront être normalisées dans le but d'assurer qu'en absence d'objets la mesure est zéro et une valeur incrémentale positive en s'approchant à un objet. Le capteur central est associé aux deux neurones du réseau. Ses mesures sont pondérées par un poids négatif qu'en présence d'un obstacle indique au robot de reculer.



Symbole	Description
w_{fwd}	Poids associé au mouvement avancer
w_{back}	Poids associé au mouvement reculer
w_{pos}	Poids associé à une rotation positive de la roue
w_{neg}	Poids associé à une rotation negative de la roue

Fig. 4 : Modèle d'ANN proposé

1. Conception d'une stratégie d'évitement d'obstacles par réseau de neurones

Implantez le modèle illustré par la Fig. 4 en respectant une représentation vectoriel du modèle. La structure vectorielle vous permettra d'ajuster facilement les poids du réseau à l'expérimental.

Questions

- Quelle est la relation entre w_{fwd} et w_{back} ?
- Quelle est la relation entre w_{fwd} et w_{pos} ainsi qu'entre w_{fwd} et w_{neg} ?
- Quelle est la relation entre w_{back} et w_{pos} ainsi qu'entre w_{back} et w_{neg} ?
- Quelle est la relation à respecter entre w_{pos} et w_{neg} ?
- Indiquez les poids permettant d'obtenir une stratégie d'évitement d'obstacles :
  - w_{fwd} :
  - w_{back} :
  - w_{pos} :
  - w_{neg} :
- Décrire le comportement du robot lorsque le capteur de gauche et le capteur du centre détectent un obstacle simultanément ?

2. Conception d'une stratégie de suivi d'objet

Sur la base de l'expérience de l'exercice 1, modifiez le poids du réseau afin d'assurer le comportement décrit ci-dessous :

- Le robot avance en absence d'objets
- Si le capteur centrale détecte un objet très proche, il s'arrête.
- Si un objet est détecté par le capteur situé sur la gauche, le robot turne à gauche pour assurer son suivi.
- Si un objet est détecté par le capteur situé sur la droite, le robot turne à droite pour assurer son suivi.

- Quelle est la relation entre w_{fwd} et w_{back} ?
- Quelle est la relation entre w_{fwd} et w_{pos} ainsi qu'entre w_{fwd} et w_{neg} ?
- Quelle est la relation entre w_{back} et w_{pos} ainsi qu'entre w_{back} et w_{neg} ?
- Quelle est la relation à respecter entre w_{pos} et w_{neg} ?
- Indiquez les poids permettant d'obtenir une stratégie d'évitement d'obstacles :
  - w_{fwd} :
  - w_{back} :
  - w_{pos} :
  - w_{neg} :
- Décrire le comportement du robot lorsque le capteur de gauche et le capteur du centre détectent un obstacle simultanément ?

Annexe

Interactions avec le jumeau numérique du Robot Thymio II

Il s'agit d'un système de simulation robotique simple, permettant le **contrôle** à partir d'un script python. Le code d'exemple du List. 1 illustre la structure global à respecter lors du contrôle distant du robot.

```python
import math
import numpy as np
import cv2
from controller import Robot

robot = Robot()
timestep = int(robot.getBasicTimeStep())
keyboard = robot.getKeyboard()
keyboard.enable(timestep)
print(chr(27) + "[2J") # ANSI code for clearing command line
print("Initialization of thymio_variables controller")

camera = robot.getDevice("camera")
camera.enable(timestep)
cam_w = camera.getWidth()
cam_h = camera.getHeight()
lidar = robot.getDevice('lidar')
lidar.enable(timestep)
lidar.enablePointCloud()
distanceSensors = []
for i in list(range(0,7)):
  distanceSensors.append(robot.getDevice('prox.horizontal.'+str(i)))
  distanceSensors[i].enable(timestep)

motor_left = robot.getDevice("motor.left");
motor_right = robot.getDevice("motor.right");
motor_left.setPosition(float('inf'))
motor_right.setPosition(float('inf'))
robot_speed = 0.0
distanceVal = [0.0,0.0,0.0,0.0,0.0,0.0,0.0]
state = 0
time= 0
once = True
# Touch sensors
button_fwd = robot.getDevice("button.forward")
button_fwd.enable(timestep)
# LEDs
led_top = robot.getDevice("leds.top")
led_bottomr = robot.getDevice("leds.bottom.right")
led_bottoml = robot.getDevice("leds.bottom.left")
led_buttons0= robot.getDevice("leds.buttons.led2")
```

```python
while (robot.step(timestep) != -1):
  image = camera.getImage()
  #print(type(image), len(image))
  img = np.frombuffer(image, np.uint8).reshape((camera.getHeight(),
  camera.getWidth(), 4))
  frame = cv2.cvtColor(img, cv2.COLOR_BGRA2BGR)
  # Now you can process or display:
  cv2.imshow("Thymio Camera", frame)
  cv2.waitKey(1)

  points = []
  point_cloud = lidar.getRangeImage()
  angle=0.0

  if(button_fwd.getValue()):
     print("Forward button was pressed")
  led_top.set(0xffffff)
  led_bottomr.set(0xffffff)
  led_bottoml.set(0xffffff)
  led_buttons0.set(0xffffff)

  # Read the LiDAR sensor, like:
  for i in point_cloud:
    time +=1
    points.append([i*math.sin(angle),i*math.cos(angle),0.0])
    angle+= 2*math.pi / lidar.getHorizontalResolution()

  # Read the proximity sensors, like:
  for i in list(range(0,7)):
    distanceVal[i] = distanceSensors[i].getValue()

  #Set motors speed :
  motor_left.setVelocity(robot_speed)
  motor_right.setVelocity(robot_speed)

  # Process sensor data here

  # Enter here functions to send actuator commands, like:
  command = keyboard.getKey()
  #print(command)

  if command==keyboard.LEFT:
    #print('Left')
    motor_left.setVelocity(0.0) #-robot_speed
    motor_right.setVelocity(robot_speed)
  elif command==keyboard.RIGHT:
    #print('right')
    motor_left.setVelocity(robot_speed)
    motor_right.setVelocity(0.0) #-robot_speed
  elif command==keyboard.UP:
    print('up')
    if robot_speed<2:
      robot_speed+=0.2
  elif command==keyboard.DOWN:
    print('down')
    if robot_speed>-2:
      robot_speed-=0.2
  elif command==83: # capture S key
    print('stop')
    robot_speed = 0
```

Listing 1 : Code des premiers pas avec le jumeau numérique du Thymio II