Rapport de bureaux d'étude

Apprentissage automatique embarqué et mobile
S9 INFO mso 3.4

Thomas Pucci
Mohamed Amine Mejri

Édité le 28/03/2017

Sommaire

1 Introduction

BE.[1]

Ce Bureau d'étude est réalisé sous le logiciel `Matlab`.

---

2 Partie 1

2.1 Sous-titre

2.1.1 Image

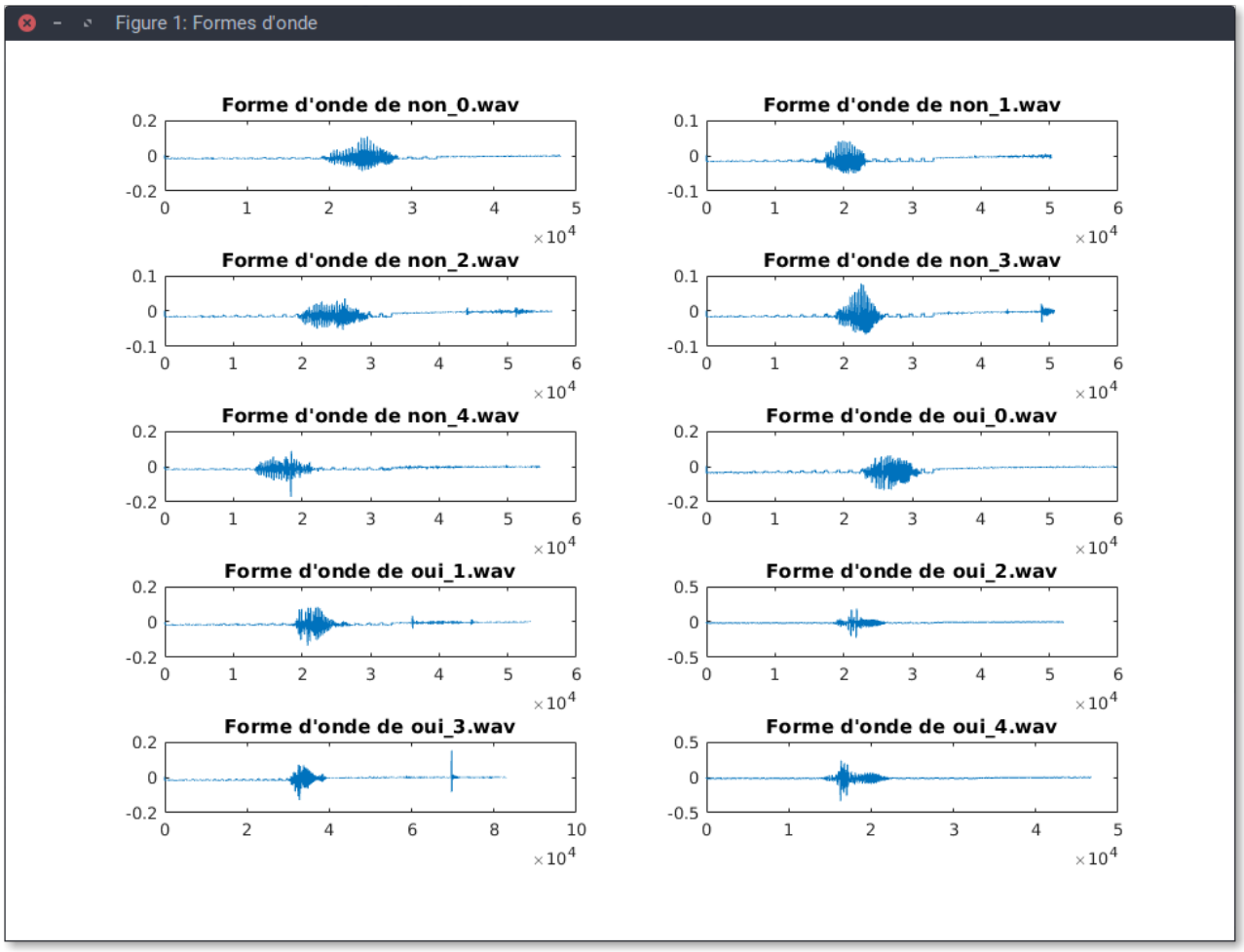

Figure 2.A: Fonctionnement de l'agorithme de récupération des descripteurs

2.1.2 Code

```
function [ output_args ] = get_datasets( input_args )
%Download Cifar-10 dataset
    disp('Downloading Cifar-10 dataset.................');
    url = 'http://www.cs.toronto.edu/~kriz/cifar-10-matlab.tar.gz';
    untar(url);
end
```

2.1.3 Tableau

	non_0	non_1
non_0	0.000	0.109
non_1	0.109	0.000

3 SVM

The gradient could not be strictly differentiable, as in our hinge loss case. In a 1D case, a point right before the hinge should have an analytical gradient of 0, while the numerical gradient would be greater than 0.

---

4 Q-Learning

Cet exercice est réalisé en suivant l'exemple de cours [2] dont nous reprenons ici l'illustration :

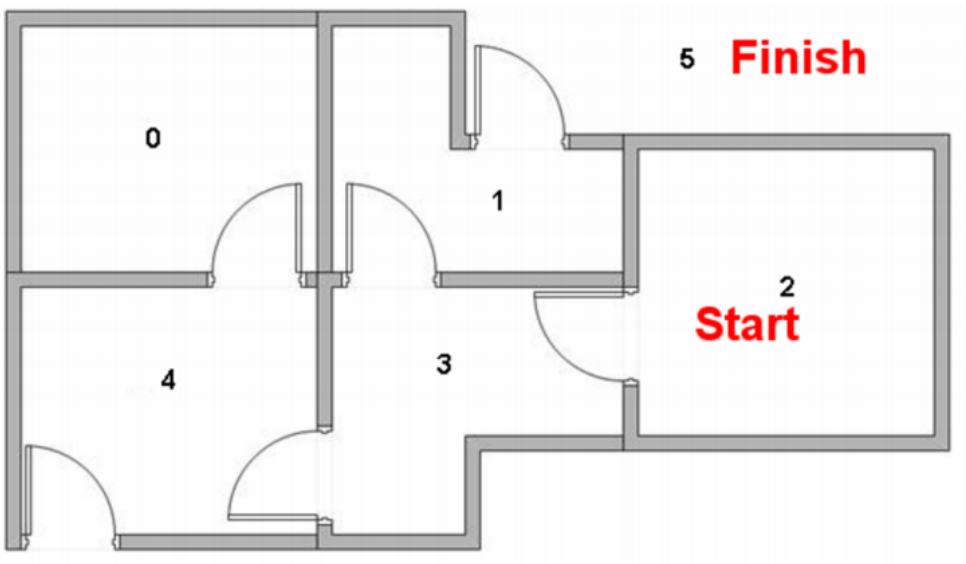

Figure 4.A: L'agent doit apprendre les meilleurs chemins vers la position 5

4.1 Configuration

Nous utilisons les données de l'énoncé pour configurer les variables de l'algorithmes. Dans un premier temps, nous considérons la matrice *récompenses* suivante :

$$R = \begin{pmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{pmatrix}$$

Celle-ci est codée en créant une matrice ne comportant que les valeurs -1 (murs), puis en modifiant certaines valeurs à 0 (représentant les portes) et d'autres à 100 (représentant les changements de pièces gagnants).

```
R = -1*ones(6);
doors = [[0,4]; [4,3]; [4,5]; [2,3]; [1,3]; [1,5]];
wins = [[1,5];[4,5];[5,5]];

for i = 1:size(doors,1) % Création des portes
    R(doors(i,1)+1,doors(i,2)+1) = 0;
    R(doors(i,2)+1,doors(i,1)+1) = 0;
end

for i = 1:size(wins,1) % Chemins gagnants
```

```
    R(wins(i,1)+1,wins(i,2)+1) = 100;
end
```

On règle ensuite les paramètres `alpha`, `gamma` et le nombre d'épisodes à réaliser :

```
alpha = 1;
gamma = .8;
nEpisodes = 100;
```

NB : Ces paramètres entraîne donc la formule d'apprentissage suivantes (annulation du terme $Q_t(s_t, a_t)$):

$$Q_{t+1}(s_t, a_t) = R_{t+1} + 0,8 * max_a Q_t(s_{t+1}, a_t)$$

On initialise la matrice d'apprentissage Q et on choisis les états initiaux pour chacun des épisodes de manière aléatoire.

```
Q = zeros(size(R));
randomStates = randi([1 size(R,2)],1,100);
```

4.2 Fonction récursive qLearn

Dans un nouveau fichier `qLearn.m` nous programmons une fonction récursive définie de la manière suivante :

```
function Q = qLearn(Q,R,alpha,gamma,state,stopState)
```

Nous identifions dans un premier temps les états suivants possibles `possibleNextStates` étant donné l'état courant `state`. Nous choisissons ensuite aléatoirement l'état suivant `nextState` parmi les possibilités `possibleNextStates`. Nous identifions ensuite les états futurs possibles `possibleFutureStates` étant donné l'état suivant `nextState`. Puis nous appliquons la formule d'apprentissage et actualisons la valeur de `Q(state,nextState)` en fonction de `alpha`, `R(state,nextState)`, `gamma` et `max(Q(nextState,possibleFutureStates))`.

Si l'état suivant `nextState` correspond à l'état final, nous arrêtons la récursion, sinon nous rappelons la fonction récursive `qLearn`.

```
function Q = qLearn(Q,R,alpha,gamma,state, stopState)
    possibleNextStates = find(R(state,:)>=0);
    nextState = possibleNextStates(randi(size(possibleNextStates)));
    possibleFutureStates = find(R(nextState,:)>=0);
    Q(state,nextState) = Q(state,nextState) + alpha * (R(state,nextState) +
        gamma*max(Q(nextState,possibleFutureStates)) - Q(state,nextState));
    if nextState == stopState
        return
    else
        Q = qLearn(Q,R,alpha,gamma,nextState,stopState);
    end
end
```

Dans le script principal, nous appelons cette fonction sur chacun des épisodes :

```
for i = 1:nEpisodes % Boucle de nEpisodes
    beginningState = randomStates(i);
    Q = qLearn(Q,R,alpha,gamma,6, 5+1); % Appel de la fonction récursive qLearn
end
```

Enfin nous affichons le résultat :

```
QNormalized = round(Q./max(max(round(Q)))*100) % Affichage du résultat arrondi
```

4.3 Traces d'exécution

Nous exécutons l'ensemble du code Matlab à partir du fichier `Run_qlearning.m`. Voici le résultat :

```
>> Run_qlearning

QNormalized =

     0     0     0     0    80     0
     0     0     0    64     0   100
     0     0     0    64     0     0
     0    80    51     0    80     0
    64     0     0    64     0   100
     0    80     0     0    80   100
```

Nous obtenons effectivement la même matrice résultat que celle se trouvant dans l'énoncé du BE [2].

5 Conclusion

Ce bureau d'étude nous a permis de réaliser...

Références

[1] E.D. L. Cheng, TD – convolutional neural networks for visual recognition, (2016).

[2] E. Dellandréa, Cours – apprentissage par renforcement, (2016).