Projet Pacman qui apprend (un peu...)

L. Denoyer

16 avril 2013

1 Introduction

Nous allons nous intéresser au développement d'un Pacman qui apprend. Nous nous plaçons dans le cas où il n'y a **qu'un unique pacman**, et autant de fantômes que l'on veut. Nous allons mettre en place des méthodes d'apprentissage basées sur l'utilisation d'un **perceptron** :

- Un algorithme basé sur de la recherche exhaustive
- L'algorithme **Fitted-Q Learning**

2 Préliminaire

2.1 Fonction de récompense

Nous allons tout d'abord définir une fonction récompense. La récompense est obtenue par Pacman à chaque coup effectué, et le but de l'agent consiste à maximiser sa récompense obtenue sur une partie. On définit l'interface **Reward** ainsi :

```
1 public interface Reward
2 {
3     public double getReward(GameState from,GameState to);
4 }
```

Définir une récompense **SimpleReward** telle que :

- Pacman obtient -100 si il perd
- Pacman obtient +100 si il gagne
- Pacman obtient 0 si il mange une cacahuète
- Pacman obtient -1 si il ne mange rien

2.1.1 Calcul de la récompense

La récompense totale d'un agent calculée sur une **trajectoire** est la saomme des récompenses obtenues après chaque action. La méthode **RewardTools.getReward** permet de calculer cette valeur pour une trajectoire (par simulation), pour un agent donné, considérant une taille de tra-

jectoire maximale (la simulation s'arrête au bout de *size_max_trajectory* actions).

Implémentez la méthode **getAverageReward** qui permet le calcul de la récompense moyenne pour N trajectoires simulées. Cette valeur sera plus fiable et plus robuste. Testez cette méthode pour un agent aléatoire **RandomAgent**, considérant que les fantômes sont tous de type **new IntelligentGhost_Agent2(0.1)**.

2.2 Senseur

Le senseur vise à représenter sous forme de vecteur la vision de Pacman à l'instant t. On définit l'interface **StateSensor** ainsi :

```
1 public interface StateSensor
2 {
3     public int size();
4     public SparseVector getVector(GameState s);
5 }
```

où :

- **size()** est la taille du vecteur retourné
- **getVector** renvoie le vecteur qui représente la vision de Pacman dans l'état s.

Le senseur sera noté $\Phi(s)$ par la suite.

Définissez une senseur **SimpleStateSenseur** qui permette à pacman de savoir ce qui se trouve dans un environnement de taille $N \times N$ centré autour de lui. Notez que ce senseur pourrait être calculé à l'aide de JESS (pas obligatoire).

Ce qui va pour la suite nous intéresser est la représentation conjointe d'un état et d'une action (parmi les 4 actions possibles). Cette représentation notée $\Phi(s, a)$ sera calculée à l'aide de la classe **Sensor** de la manière suivante :

```
1 StateSensor s=new .......;
2 Sensor sensor=new Sensor(s);
```

3 Perceptron scalaire

Nous allons maintenant définir un perceptron qui, au lieu de prédir $+1/-1$ ou bien $true/false$, sera capable de prédire n'importe quelle valeur réelle (Perceptron de régression). Pour cela, nous allons modifier les classes **LabeledSet** et **Perceptron** implémentées lors d'un précédent TP :

- Les étiquettes des données ne sont plus des *booléens*, mais des *double*
- La fonction *getScore* du perceptron ne change pas

– La performance du perceptron est calculée commme la somme des erreurs entre la prédiction et la valeur réelle :

$$\frac{1}{n}\sum_{i=1}^{n}(reel - predit) \times (reel - predit) \tag{1}$$

– La mise à jour dépend de l'erreur de prédiction :

```
1 double reel=training_set.getLabel(index);
2 double predicted=getScore(v);
3 double error=reel-predicted;
4 parameters.addVector(v,epsilon*error);
```

3.1 Agent et Perceptron

Un **PerceptronAgent** est un agent qui choisit l'action à effectuer à l'aide d'une perceptron et d'un senseur. Soit θ un perceptron, l'action a^* choisie dans l'état s est :

$$a^* = \arg\max_a < \theta; \Phi(s,a) > \tag{2}$$

Les actions à tester sont uniquement les actions possibles telles que :

```
1 state.isLegalMove(new AgentAction(a),as)==true
```

Implémentez l'agent perceptron. Tester la performance dc'un perceptron choisi au hasard.

3.2 Un premier algorithme de recherche aléatoire

Nous allons implémenter l'algorithme suivant :

– Nous allons générer N perceptron aléatoirement - qui correspondent à N agents
– Nous allons évaluer ces N perceptrons grâce à la fonction **RewardToools.getAverageReward**
– Nous allons conserver les M meilleurs perceptron (voir classe **Elt** permettant de faire le tri d'un tableau)
– Nous allons compléter les M meilleurs en générant $N - M$ nouveau perceptrons de la facon suivante :
  – On prend l'un des M perceptrons
  – On bruite légèrement ses paramètres d'un facteur *variance*

  Implémentez et testez l'algorithme. Qu'en pensez vous ? Adaptez le reward pour que Pacamn survive le plus longtems possible. Vous pouvez utiliser la méthode **RewardTools.vizualize** pour visualiser un agent. Que pensez vous de cet algorithme ?

3.3 Fitted Q Learning

Afin d'implémentez l'algorithme Fitted Q (FQL), il faut commencer par échantilloner des quadruplets (s, a, s', r) (état, action, état atteind, et reward obtenu). Il faudra échantilloner plusieurs dizaines de milliers de ces quadruplets.

– Implémentez une classe permettant d'échantilloner ces quadruplets pour un agent aléatoire

L'étape suivante consiste à calculer les vecteurs $\Phi(s, a)$ pour chaque quadruplet collecté. Implémentez cette méthode.

3.3.1 Perceptron 0

Nous allons maintenant apprendre un premier perceptron de la façon suivante :

– Pour chaque quadruplet collecté (s_i, a_i, s'_i, r_i) nous allons générer un exemple d'apprentissage $^(\Phi(s_i, a_i), r_i)$

– Nous allons apprendre le perceptron sur cet ensemble d'apprentissage

Implémentez cette algorithme, et testez le résultat obtenu. Qu'en pensez vous ? Comprenez vous comment tout cela fonctionne ?

3.3.2 Perceptron t

Considérons que nous connaissons le perceptron numéro t noté θ_t. Nous allons calculer un perceptron $t+1$ censé améliorer le perceptron à t. Pour cela :

– Pour chaque quadruplet collecté (s_i, a_i, s'_i, r_i) nous allons générer un exemple d'apprentissage $^(\Phi(s_i, a_i), m_i)$ tel que

$$m_i = r_i + \arg\max_{a'} < \theta_t; \Phi(s', a') > \tag{3}$$

– Apprendre le perceptron sur l'ensemble obtenu

Implémentez, testez, comprenez, et expliquez à votre professeur.....
Est-ce que la méthod fonctionne ? Que peut on faire pour l'amléiorer ?
Faites des expériences, testez des choses, et dites ous ce que vous en pensez