

Projet Foot 2I013

Nicolas Baskiotis

`nicolas.baskiotis@lip6.fr`

`http://webia.lip6.fr/~baskiotisn`

`http://github.com/baskiotisn/SoccerSimulator`

Université Pierre et Marie Curie (UPMC)
Laboratoire d'Informatique de Paris 6 (LIP6)

S2 (2016-2017)

Plan

Résultats de la semaine

IA et optimisation

Les derniers secrets du simulateur

Classification supervisée : Arbres de décision







Tournoi 1v1

Equipe (login)	Points	(G,N,P)	(GA,GP)
leoni1 (leoniethiriat)	88	(29,1,2)	(297,26)
CurryPoulet (JimTitor)	77	(25,2,5)	(271,58)
TEAM 1 FABIEN (fabien25)	72	(23,3,6)	(291,42)
Toho (MoroMane)	60	(19,3,10)	(220,104)
UPMC (mouloudETjeffrey)	60	(19,3,10)	(197,120)
EDF (SoccerJess)	58	(19,1,12)	(293,161)
Liban (hassanharb)	55	(18,1,13)	(231,204)
TeamFUT (SinghKevin)	51	(16,3,13)	(237,152)
EGYALG (EgyAlg)	48	(16,0,16)	(134,155)
MFC (jeandeb)	44	(13,5,14)	(139,149)
Ariouati (Ariouati)	41	(10,11,11)	(95,129)
DZ (manelfil)	37	(11,4,17)	(97,163)
FUT team (KimmengLy)	37	(11,4,17)	(115,239)
team (yxd117)	30	(10,0,22)	(132,213)
Gryffondor (GeekyCeline)	19	(5,4,23)	(28,261)
DTeam (exrivalis)	10	(3,1,28)	(42,313)
CityHunter (YasmineAitMimoun)	6	(2,0,30)	(40,370)

Tournoi 2v2

Equipe (login)	Points	(G,N,P)	(GA,GP)
MFC (jeandeb)	74	(23,5,4)	(178,35)
Ariouati (Ariouati)	73	(23,4,5)	(149,48)
CurryPoulet (JimTitor)	69	(21,6,5)	(112,30)
France (hassanharb)	67	(21,4,7)	(142,33)
leoni1 (leoniethiriat)	64	(19,7,6)	(105,29)
TEAM 2 FABIEN (fabien25)	62	(20,2,10)	(168,52)
team2 (MoroMane)	59	(18,5,9)	(141,51)
UPMC (mouloudETjeffrey)	59	(18,5,9)	(122,52)
TeamFUT (SinghKevin)	55	(18,1,13)	(131,47)
DZZ (manelfil)	46	(15,1,16)	(77,101)
EGYALG (EgyAlg)	38	(11,5,16)	(79,75)
team (yxd117)	37	(12,1,19)	(175,168)
EDF (SoccerJess)	30	(10,0,22)	(82,220)
FUT team (KimmengLy)	21	(6,3,23)	(38,153)
CityHunter (YasmineAitMimoun)	19	(4,7,21)	(35,175)
None (exrivalis)	7	(2,1,29)	(19,306)
Gryffondor (GeekyCeline)	6	(1,3,28)	(9,187)

Résumé hebdomadaire

Semaine # joueurs	4		5			6			7		8		
	1	2		1	2	1	2	1	2	1	2	1	2
Ariouati	3	1		9	6		11	2					
EgyAlg				14	16		9	11					
exrivalis	5	6		13	14		16	16					
fabien25	8	5		7	3		3	6					
GeekyCeline	10	7		17	17		15	17					
hassanharb	4	8		2	7		7	4					
jeandeb	2	4		8	2		10	1					
JimTitor				1	1		2	3					
KimmengLy	6	2		12	10		13	14					
leoniethiriat				16	15		1	5					
manelfil				10	13		12	10					
MoroMane	7	3		6	4		4	7					
mouloudETjeffrey				3	5		5	8					
SinghKevin	9	9		5	8		8	9					
SoccerJess				4	12		6	13					
YasmineAitMimoun				15	11		17	15					
yxd117	1	10		11	9		14	12					

Plan

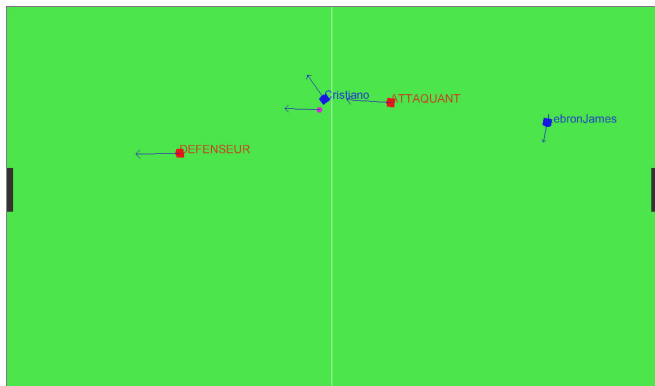
Résultats de la semaine

IA et optimisation

Les derniers secrets du simulateur

Classification supervisée : Arbres de décision

Des problèmes distincts



- Comment effectuer une action ?
- Quand appliquer une stratégie ?

Comment effectuer une action ?

Définir très précisément

- le but à atteindre
- les conditions initiales
- les paramètres sur lesquels joués
- comment modéliser les réactions attendues

Modélisation

Principalement deux possibilités :

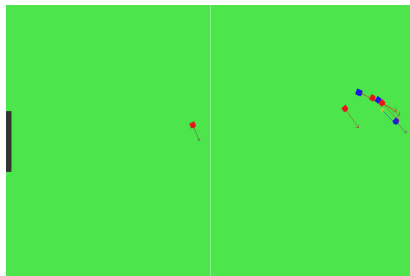
- discrétisation : par un tableau
- continue : par une fonction paramétrée

Optimisation

- par recherche exhaustive (mais intelligente !)
- par algorithme génétique

Quand appliquer une stratégie ?

Choix des stratégies



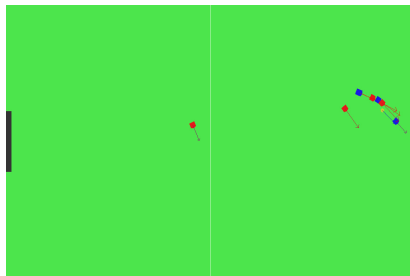
- Besoin de décrire une situation \Rightarrow descripteurs/attributs
- En fonction de leurs valeurs, choisir la situation la plus adaptée

Apprentissage

- Par algorithme génétique
- Par apprentissage supervisé

Quand appliquer une stratégie ?

Choix des stratégies

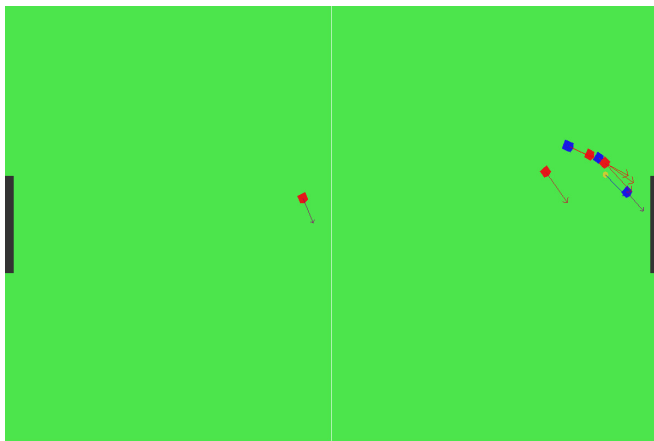


- Besoin de décrire une situation \Rightarrow descripteurs/attributs
- En fonction de leurs valeurs, choisir la situation la plus adaptée

Apprentissage

- Par algorithme génétique
- Par apprentissage supervisé

Description d'une situation

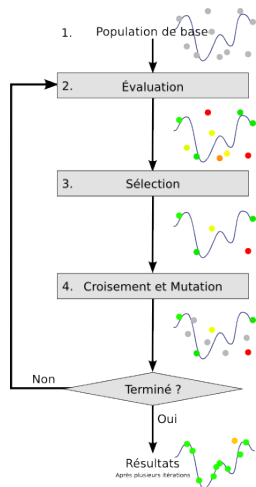


- Attention : un attribut doit pouvoir généraliser !!
- Description : (position joueur, position balle, ...) pas bien → pourquoi ?
- (distance a la balle, au but, distance adversaire) bien ! → pourquoi ?

Algorithme génétique

Objectif

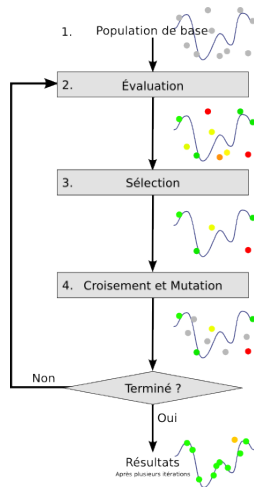
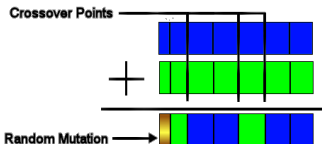
- Optimiser une fonction $f(x_1, x_2, \dots, x_d)$
- ⇒ trouver son minimum
- Mais trop grand nombre de paramètres
- ⇒ impossible de tout explorer
- Inspiration biologique darwinienne :
- ⇒ évolution d'une population (ensemble) de solutions par mutation et croisements aléatoires



Algorithme génétique

Algorithme

- Choisir un codage des paramètres : point crucial !
- Générer au hasard une population initiale
- Itérer :
 - Evaluer chaque individu (chaque solution potentielle)
 - Garder les meilleurs (25% par exemple)
 - Croiser les individus au hasard
 - Muter les individus avec une faible probabilité.



Plan

Résultats de la semaine

IA et optimisation

Les derniers secrets du simulateur

Classification supervisée : Arbres de décision

Observer : un design pattern de plus

SoccerEvents

- Une simulation possède une liste d'observateurs (`listeners`)
- A chaque évènement marquant, tous les observateurs sont avertis par l'appel d'une fonction
- il est possible d'ajouter à la volée ou de supprimer des observateurs de la simulation.

Actions déclenchées lors d'une simulation

- `begin_match(team1, team2, state)` : au début de la simulation
- `end_match(team1, team2, state)` : à la fin
- `begin_round(team1, team2, state)` : au début de chaque engagement
- `end_round(team1, team2, state)` : à chaque but marqué
- `update_round(team1, team2, state)` : à chaque fin de tour

Comment utiliser l'observeur ?

Pour simuler !

```
class Observer(object):
    MAX_STEP=40
    def __init__(self,simu):
        self.simu = simu
        self.simu.listeners+=self #ajout de l'observer
    def begin_match(self,team1,team2,state):
        #initialisation des parametres ...
        self.last, self.cpt, self.cpt_tot = 0, 0, 0
    def begin_round(self,team1,team2,state):
        self.simu.state.states[(1,0)].position = ...
        #ou self.simu.set_state(state)
        self.strat.shoot = ...
        self.last = self.simu.step
    def update_round(self,team1,team2,state):
        if state.step>self.last+self.MAX_STEP: self.simu.end_round()
    def end_round(self,team1,team2,state):
        if state.goal>0: self.cpt+=1
        self.cpt_tot+=1
        self.res[...]= self.cpt*1./self.cpt_tot.
        if ... : #fin de la simu
            self.simu.end_match()
```


Plan

Résultats de la semaine

IA et optimisation

Les derniers secrets du simulateur

Classification supervisée : Arbres de décision

Formalisation de l'apprentissage supervisé

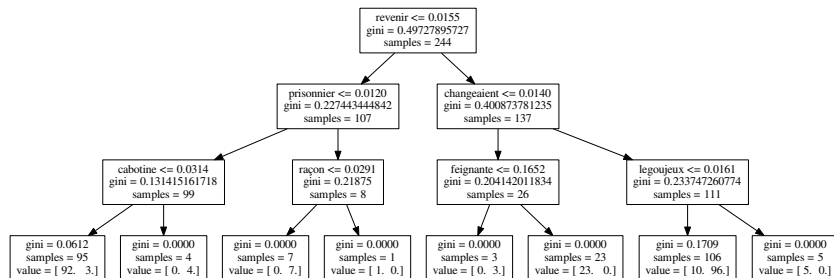
On dispose :

- d'un espace de représentation \mathcal{X} , usuellement \mathbb{R}^n
 n est la dimension de l'espace de représentation,
chaque dimension = un attribut
⇒ une dimension décrit un élément de la situation
- d'un ensemble d'exemples X décrit dans cette espace :
 $x \in X, x = (x_1, x_2, x_3, \dots, x_n)$
⇒ un exemple = une situation
- d'un ensemble d'étiquettes/labels Y décrivant les classes
⇒ Y = ensemble des stratégies possibles
- pour chaque exemple x^i de X , son étiquette y^i
⇒ ensemble d'apprentissage $E = \{(x^i, y^i)\}$

On veut :

Trouver une fonction $f : \mathcal{X} \rightarrow Y$ telle que la prédiction sur de futurs exemples soit la plus précise possible.

Arbres de décision

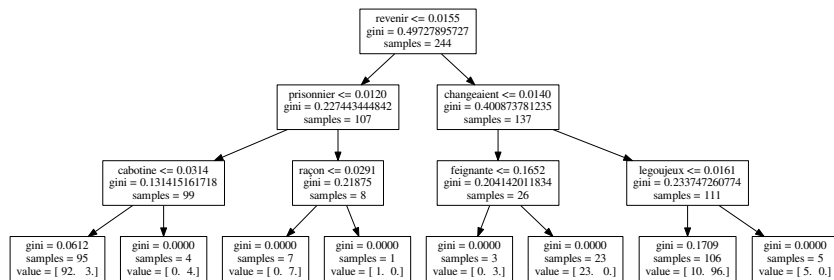


Principe

- Chaque nœud interne : un test sur une des dimensions de \mathcal{X}
- Chaque branche : un résultat du test
- Chaque feuille : un label de \mathcal{Y}

⇒ classification en parcourant un chemin de la racine à une feuille.

Arbres de décision

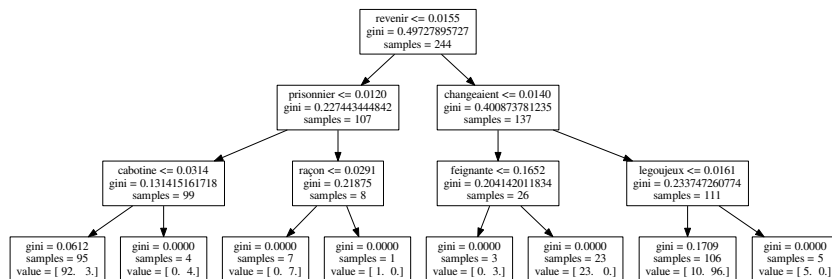


Exercice

Comment représenter :

- $balle > 1 \wedge adversaire < 0.5$?
- $balle > 1 \vee adversaire < 0.5$?
- $balle > 1 \wedge adversaire < 1 \vee balle < 1 \wedge equipier > 1$?

Apprentissage d'un arbre de décision

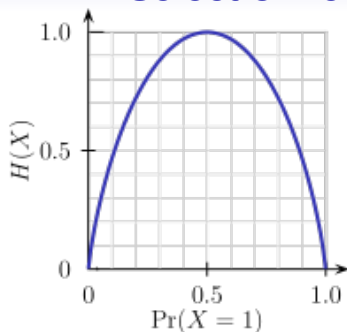


Algorithme glouton, top-down

Initialisation à la racine, considérer tous les exemples

- Si le nœud n'est pas pur, alors
 - Trouver x_i le "meilleur" attribut pour ce nœud et le seuil
 - Pour chaque test, créer un fils au nœud courant
 - Affecter les exemples du nœud courant au fils correspondant
- sinon transformer le nœud en feuille.

Sélectionner le meilleur attribut



Entropie d'une variable aléatoire

Soit X une variable aléatoire pouvant prendre n valeurs :

$$H(X) = - \sum_{i=1}^n P(X = u) \log(P(X = i))$$

Plus l'entropie est grande, plus le désordre est grand.

Entropie nulle \rightarrow pas d'aléa.

Sélectionner le meilleur attribut

Entropie d'un échantillon : cas binaire

- X un ensemble de données
- p_+ la proportion d'exemples positifs
- p_- la proportion d'exemples négatifs
- $H(X) = -p_+ \log(p_+) - p_- \log(p_-)$

Cas général : entropie conditionnelle

- $H(X|Y = y) = -\sum_{i=1}^n P(X = i|Y = y) \log P(X = i|Y = y)$
 - $H(X|Y) = \sum_{y \in Y} P(Y = y) H(X|Y = y)$
- ⇒ Gain d'information : $I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$

Comment collecter les exemples ?

Une stratégie spéciale : KeyboardStrategy

Assigne une touche à une stratégie; pendant le jeu, chaque changement de stratégie provoque la sauvegarde de l'état et de la stratégie choisie.

```
strat = KeyboardStrategy() #ou pour une sauvegarde automatique  
                                #KeyboardStrategy(fn="monfichier.exp")  
strat.add("d",DefenseStrategy())  
strat.add("a",AttaqueStrategy())  
player1 = Player("j1",strat)  
....
```