2I013 Groupe 1

Projet Foot

Nicolas Baskiotis - Maxime Sangnier
nicolas.baskiotis@lip6.fr
maxime.sangnier@upmc.fr

S2 (2017-2018)

Laboratoire d'Informatique de Paris 6 (LIP6)
Sorbonne Université

Outline

Optimisation d'une stratégie

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Dépendent de degrés de liberté :

- puissance de tir ;

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Dépendent de degrés de liberté :

- puissance de tir ;
- direction ;

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Dépendent de degrés de liberté :

- puissance de tir ;
- direction ;
- déclenchement de l'action ;

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Dépendent de degrés de liberté :

- puissance de tir ;
- direction ;
- déclenchement de l'action ;
- position de repos ;

Optimisation d'une stratégie

Quelle stratégie ?

- Shoot.
- Défense.
- Goal.
- Rejoindre la balle.
- Etc.

Dépendent de degrés de liberté :

- puissance de tir ;
- direction ;
- déclenchement de l'action ;
- position de repos ;
- vitesse ;
- etc.

Les degrés de liberté

Comment fixer ces degrés de liberté ?

- Connaître le fonctionnement exact du simulateur.

Les degrés de liberté

Comment fixer ces degrés de liberté ?

- Connaître le fonctionnement exact du simulateur.
- Savoir par expérience.
- Etc. . .

Les degrés de liberté

Comment fixer ces degrés de liberté ?

- Connaître le fonctionnement exact du simulateur.
- Savoir par expérience.
- Etc. . .

Ce n'est pas suffisant ! Cela ne prend pas en compte l'aléa.

Les degrés de liberté

Comment fixer ces degrés de liberté ?

- Connaître le fonctionnement exact du simulateur.
- Savoir par expérience.
- Etc. . .

Ce n'est pas suffisant ! Cela ne prend pas en compte l'aléa.

Nécessité d'un choix heuristique, valable en moyenne.

Plan d'expérience

Définir :

1. une action à optimiser ;

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;
4. un critère ;

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;
4. un critère ;
5. les conditions environnementales.

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;
4. un critère ;
5. les conditions environnementales.

Procédure :

1. Étant donné une valeur de paramètre, évaluer le critère dans différentes conditions environnementales.

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;
4. un critère ;
5. les conditions environnementales.

Procédure :

1. Étant donné une valeur de paramètre, évaluer le critère dans différentes conditions environnementales.
2. Moyenner ces évaluations.

Plan d'expérience

Définir :

1. une action à optimiser ;
2. un modèle ;
3. les paramètres du modèle à optimiser ;
4. un critère ;
5. les conditions environnementales.

Procédure :

1. Étant donné une valeur de paramètre, évaluer le critère dans différentes conditions environnementales.
2. Moyenner ces évaluations.
3. Maximiser l'évaluation moyenne du critère par rapport aux paramètres à optimiser.

Paramètres maximisant

Si l'on dispose d'un paramètre discret :

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

- discrétisation ;

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

- discrétisation ;
- nécessite des bornes ;

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

- discrétisation ;
- nécessite des bornes ;
- nécessite un pas de discrétisation.

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

- discrétisation ;
- nécessite des bornes ;
- nécessite un pas de discrétisation.

Si l'on dispose de plusieurs paramètres :

Paramètres maximisant

Si l'on dispose d'un paramètre discret : recherche exhaustive.

Si l'on dispose d'un paramètre continu :

- discrétisation ;
- nécessite des bornes ;
- nécessite un pas de discrétisation.

Si l'on dispose de plusieurs paramètres : recherche en grille.

Paramètres maximisant

Procédure :

1. Discrétiser les paramètres continus.

Paramètres maximisant

Procédure :

1. Discrétiser les paramètres continus.
2. Pour chaque possibilité de valeurs des paramètres, . . .

Paramètres maximisant

Procédure :

1. Discrétiser les paramètres continus.
2. Pour chaque possibilité de valeurs des paramètres, . . .
3. Générer des conditions environnementales aléatoirement, et évaluer le critère.

Paramètres maximisant

Procédure :

1. Discrétiser les paramètres continus.
2. Pour chaque possibilité de valeurs des paramètres, . . .
3. Générer des conditions environnementales aléatoirement, et évaluer le critère.
4. Calculer le critère moyen.

Paramètres maximisant

Procédure :

1. Discrétiser les paramètres continus.
2. Pour chaque possibilité de valeurs des paramètres, . . .
3. Générer des conditions environnementales aléatoirement, et évaluer le critère.
4. Calculer le critère moyen.
5. Retourner l'ensemble des valeurs des paramètres associé à la valeur maximale du critère.

Exemple : le shoot

Exemple du shoot (simple)

1. Action : shoot.

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.
3. Paramètre : $\alpha \in [0, 2]$, continu.

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.
3. Paramètre : $\alpha \in [0, 2]$, continu.
4. Critère : nombre de buts marqués.

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.
3. Paramètre : $\alpha \in [0, 2]$, continu.
4. Critère : nombre de buts marqués.
5. Conditions environnementales :
   - position de la balle sur le terrain ;

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.
3. Paramètre : $\alpha \in [0, 2]$, continu.
4. Critère : nombre de buts marqués.
5. Conditions environnementales :
   - position de la balle sur le terrain ;
   - position du joueur par rapport à la balle (angle) ;

Exemple du shoot (simple)

1. Action : shoot.
2. Modèle :
   - direction : $\vec{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - shoot = `Vector2D(shoot=`$\alpha \vec{u}$`)`.
3. Paramètre : $\alpha \in [0, 2]$, continu.
4. Critère : nombre de buts marqués.
5. Conditions environnementales :
   - position de la balle sur le terrain ;
   - position du joueur par rapport à la balle (angle) ;
   - vitesse du joueur ;
   - etc.

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
    - placer le ballon au hasard sur le terrain ;

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
   - placer le ballon au hasard sur le terrain ;
   - placer le joueur sur le ballon ;

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
   - placer le ballon au hasard sur le terrain ;
   - placer le joueur sur le ballon ;
   - appliquer la stratégie avec ladite valeur de α ;

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
   - placer le ballon au hasard sur le terrain ;
   - placer le joueur sur le ballon ;
   - appliquer la stratégie avec ladite valeur de α ;
   - enregistrer s'il y a eu but ou non.

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
   - placer le ballon au hasard sur le terrain ;
   - placer le joueur sur le ballon ;
   - appliquer la stratégie avec ladite valeur de α ;
   - enregistrer s'il y a eu but ou non.
4. Calculer le nombre moyen de but (nombre de buts / nombre d'essais).

Procédure

Procédure :

1. Discrétiser le paramètre $\alpha \in \{0, 0.1, 0.2, \ldots, 2\}$.
2. Pour chaque valeur de α, ...
3. Pour `essai` allant de 1 à ...
   - placer le ballon au hasard sur le terrain ;
   - placer le joueur sur le ballon ;
   - appliquer la stratégie avec ladite valeur de α ;
   - enregistrer s'il y a eu but ou non.
4. Calculer le nombre moyen de but (nombre de buts / nombre d'essais).
5. Retourner la valeur de α avec le meilleur ratio nombre de buts / nombre d'essais.

Exemple de code

```python
class FonceurTestStrategy(Strategy):
    def __init__(self, strength=None):
        Strategy.__init__(self, "Fonceur Test")
        self.strength = strength

    def compute_strategy(self, state, id_team, id_player):
        s = SuperState(state, id_team, id_player)
        move = Move(s)
        shoot = Shoot(s)
        return move.to_ball() + shoot.to_goal(self.strength)
```

Exemple de code

```python
class FonceurTestStrategy(Strategy):
    def __init__(self, strength=None):
        Strategy.__init__(self, "Fonceur Test")
        self.strength = strength

    def compute_strategy(self, state, id_team, id_player):
        s = SuperState(state, id_team, id_player)
        move = Move(s)
        shoot = Shoot(s)
        return move.to_ball() + shoot.to_goal(self.strength)
```

```python
from projet.optimization import ParamSearch
from projet.strategies import FonceurTestStrategy


expe = ParamSearch(strategy=FonceurTestStrategy(),
                   params={'strength': [0.1, 1]})
expe.start()
print(expe.get_res())
```

Exemple de code

```python
class FonceurTestStrategy(Strategy):
    def __init__(self, strength=None):
        Strategy.__init__(self, "Fonceur Test")
        self.strength = strength

    def compute_strategy(self, state, id_team, id_player):
        s = SuperState(state, id_team, id_player)
        move = Move(s)
        shoot = Shoot(s)
        return move.to_ball() + shoot.to_goal(self.strength)
```

```python
from projet.optimization import ParamSearch
from projet.strategies import FonceurTestStrategy


expe = ParamSearch(strategy=FonceurTestStrategy(),
                   params={'strength': [0.1, 1]})
expe.start()
print(expe.get_res())
```

```
/home/maxime/soccer/bin/python2.7 /h
strength: 0.1   Crit: 0   Cpt: 1
strength: 0.1   Crit: 1   Cpt: 2
strength: 0.1   Crit: 2   Cpt: 3
strength: 0.1   Crit: 3   Cpt: 4
strength: 0.1   Crit: 3   Cpt: 5
strength: 0.1   Crit: 3   Cpt: 6
strength: 0.1   Crit: 3   Cpt: 7
strength: 0.1   Crit: 3   Cpt: 8
strength: 0.1   Crit: 3   Cpt: 9
strength: 0.1   Crit: 3   Cpt: 10
strength: 0.1   Crit: 4   Cpt: 11
strength: 0.1   Crit: 4   Cpt: 12
strength: 0.1   Crit: 5   Cpt: 13
strength: 0.1   Crit: 6   Cpt: 14
strength: 0.1   Crit: 7   Cpt: 15
strength: 0.1   Crit: 8   Cpt: 16
strength: 0.1   Crit: 8   Cpt: 17
strength: 0.1   Crit: 9   Cpt: 18
strength: 0.1   Crit: 9   Cpt: 19
(0.1,) 9
strength: 1   Crit: 0   Cpt: 0
strength: 1   Crit: 0   Cpt: 1
strength: 1   Crit: 1   Cpt: 2
strength: 1   Crit: 1   Cpt: 3
strength: 1   Crit: 1   Cpt: 4
strength: 1   Crit: 1   Cpt: 5
strength: 1   Crit: 1   Cpt: 6
strength: 1   Crit: 1   Cpt: 7
strength: 1   Crit: 2   Cpt: 8
strength: 1   Crit: 2   Cpt: 9
strength: 1   Crit: 2   Cpt: 10
strength: 1   Crit: 2   Cpt: 11
strength: 1   Crit: 3   Cpt: 12
strength: 1   Crit: 3   Cpt: 13
strength: 1   Crit: 3   Cpt: 14
strength: 1   Crit: 3   Cpt: 15
strength: 1   Crit: 3   Cpt: 16
strength: 1   Crit: 3   Cpt: 17
strength: 1   Crit: 3   Cpt: 18
strength: 1   Crit: 3   Cpt: 19
(1,) 3
strength: 1   Crit: 0   Cpt: 0
{(0.1,): 0.45, (1,): 0.15}
```

Deuxième exemple de shoot

1. Action : shoot.

Deuxième exemple de shoot

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - $p \leftarrow$ position dans la grille ;
   - shoot = `Vector2D(shoot=`$\alpha_p \overrightarrow{u}$`)`.

Deuxième exemple de shoot

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - $p \leftarrow$ position dans la grille ;
   - shoot = `Vector2D(shoot=`$\alpha_p \overrightarrow{u}$`)`.
3. Paramètres : $\alpha_1, \alpha_2, \cdots \in [0, 100]$, continu.

Deuxième exemple de shoot

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - $p \leftarrow$ position dans la grille ;
   - shoot = `Vector2D(shoot=`$\alpha_p \overrightarrow{u}$`)`.
3. Paramètres : $\alpha_1, \alpha_2, \cdots \in [0, 100]$, continu.
4. Critère : nombre de buts marqués.

Deuxième exemple de shoot

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - $p \leftarrow$ position dans la grille ;
   - shoot = `Vector2D(shoot=`$\alpha_p \overrightarrow{u}$`)`.
3. Paramètres : $\alpha_1, \alpha_2, \cdots \in [0, 100]$, continu.
4. Critère : nombre de buts marqués.
5. Conditions environnementales :
   - position de la balle sur le terrain ;
   - position du joueur par rapport à la balle (angle) ;
   - vitesse du joueur ;
   - etc.

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
    - Pour chaque valeur de α_p, ...

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;
     - placer le joueur sur le ballon ;

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;
     - placer le joueur sur le ballon ;
     - appliquer la stratégie avec ladite valeur de α_p ;

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;
     - placer le joueur sur le ballon ;
     - appliquer la stratégie avec ladite valeur de α_p ;
     - enregistrer s'il y a eu but ou non.

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;
     - placer le joueur sur le ballon ;
     - appliquer la stratégie avec ladite valeur de α_p ;
     - enregistrer s'il y a eu but ou non.
   - Calculer le nombre moyen de but (nombre de buts / nombre d'essais).

Procédure

Procédure :

1. Discrétiser les paramètres $\alpha_1, \alpha_2, \cdots \in \{0, 10, 20, \ldots, 100\}$.
2. Pour chaque position p sur la grille, ...
   - Pour chaque valeur de α_p, ...
   - Pour `essai` allant de 1 à ...
     - placer le ballon au hasard dans la cellule de la grille ;
     - placer le joueur sur le ballon ;
     - appliquer la stratégie avec ladite valeur de α_p ;
     - enregistrer s'il y a eu but ou non.
   - Calculer le nombre moyen de but (nombre de buts / nombre d'essais).
   - Retourner la valeur de α_p avec le meilleur ratio nombre de buts / nombre d'essais.

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ?

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.

De quoi dépend la force ?

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.

De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \| \overrightarrow{u} \|$ (distance aux cages) ;

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \| \overrightarrow{u} \|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \|\overrightarrow{u}\|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\|\overrightarrow{u}\|}$;
   - shoot = `Vector2D(shoot=`$f_\alpha(d)\overrightarrow{u}$`)` ;

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \| \overrightarrow{u} \|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - shoot = `Vector2D(shoot=`$f_\alpha(d)\overrightarrow{u}$`)` ;
     - $f_\alpha(x) = 1 - e^{-\alpha x}$

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \| \overrightarrow{u} \|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\| \overrightarrow{u} \|}$;
   - shoot = `Vector2D(shoot=`$f_\alpha(d)\overrightarrow{u}$`)` ;
     - $f_\alpha(x) = 1 - e^{-\alpha x}$
     - $f_\alpha(x) = \frac{\mathbf{1}}{\mathbf{1}+e^{-\alpha(x-\mathbf{30})}}$

Troisième exemple de shoot (réfléchi)

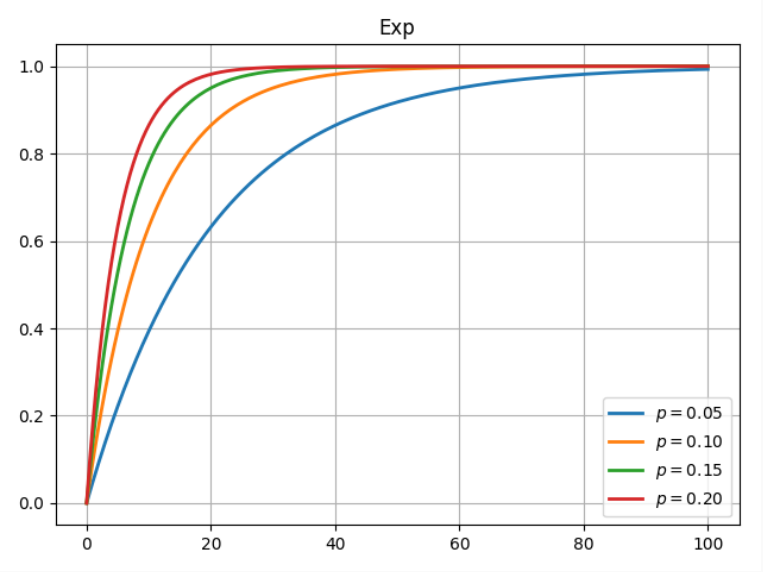
De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

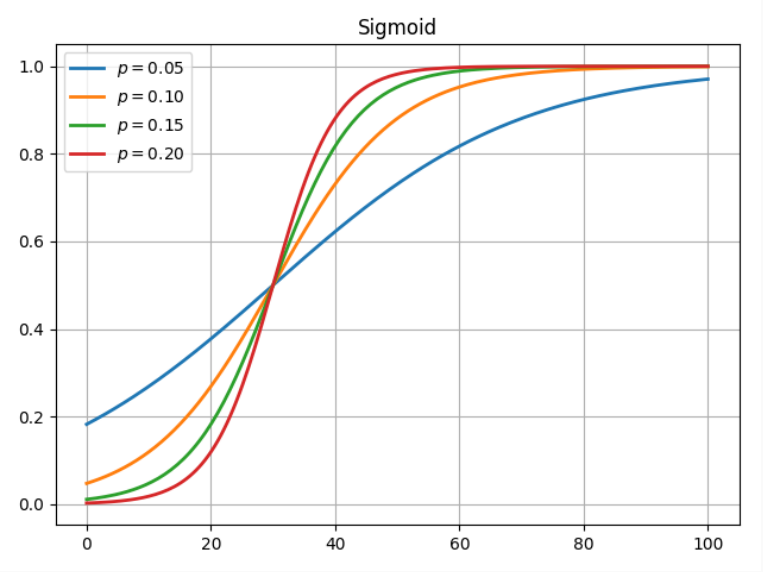
1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \|\overrightarrow{u}\|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\|\overrightarrow{u}\|}$;
   - shoot = `Vector2D(shoot=`$f_\alpha(d)\overrightarrow{u}$`)` ;
     - $f_\alpha(x) = 1 - e^{-\alpha x}$
     - $f_\alpha(x) = \frac{\mathbf{1}}{\mathbf{1}+e^{-\alpha(x-\mathbf{30})}}$
3. Paramètre : $\alpha \in [0.05, 0.2]$, continu.
4. ...

Troisième exemple de shoot (réfléchi)

De quoi dépend le succès ? $\rightarrow$ De la force.
De quoi dépend la force ? $\rightarrow$ De la distance aux cages.

1. Action : shoot.
2. Modèle :
   - direction : $\overrightarrow{u} = \overrightarrow{\text{GOAL}} - \overrightarrow{\text{PLAYER}}$;
   - $d \leftarrow \|\overrightarrow{u}\|$ (distance aux cages) ;
   - $\overrightarrow{u} \leftarrow \frac{\overrightarrow{u}}{\|\overrightarrow{u}\|}$;
   - shoot = `Vector2D(shoot=`$f_\alpha(d)\overrightarrow{u}$`)` ;
     - $f_\alpha(x) = 1 - e^{-\alpha x}$
     - $f_\alpha(x) = \frac{1}{1+e^{-\alpha(x-30)}}$
3. Paramètre : $\alpha \in [0.05, 0.2]$, continu.
4. ...





En pratique

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

L'observateur :

- tout changement d'état d'un objet (dit observé ou observable) est notifié aux objets dépendants (dits observateurs) ;

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

L'observateur :

- tout changement d'état d'un objet (dit observé ou observable) est notifié aux objets dépendants (dits observateurs) ;
- les observateurs peuvent répercuter le changement notifié de façon dynamique ;

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

L'observateur :

- tout changement d'état d'un objet (dit observé ou observable) est notifié aux objets dépendants (dits observateurs) ;
- les observateurs peuvent répercuter le changement notifié de façon dynamique ;
- la dépendance entre objet observable et objets observateurs est gérée dynamiquement $\rightarrow$ on peut ajouter et retirer un observateur à tout moment sans bien sûr affecter le code des classes existantes ;

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

L'observateur :

- tout changement d'état d'un objet (dit observé ou observable) est notifié aux objets dépendants (dits observateurs) ;
- les observateurs peuvent répercuter le changement notifié de façon dynamique ;
- la dépendance entre objet observable et objets observateurs est gérée dynamiquement $\rightarrow$ on peut ajouter et retirer un observateur à tout moment sans bien sûr affecter le code des classes existantes ;
- l'objet observé connaît les objets observateurs $\rightarrow$ en pratique l'objet observé contient une liste d'observateurs (`listeners`) ;

Motifs de conception

Les trois groupes de motifs de conception :

1. création ;
2. structure ;
3. comportement $\rightarrow$ observateur.

L'observateur :

- tout changement d'état d'un objet (dit observé ou observable) est notifié aux objets dépendants (dits observateurs) ;
- les observateurs peuvent répercuter le changement notifié de façon dynamique ;
- la dépendance entre objet observable et objets observateurs est gérée dynamiquement $\rightarrow$ on peut ajouter et retirer un observateur à tout moment sans bien sûr affecter le code des classes existantes ;
- l'objet observé connaît les objets observateurs $\rightarrow$ en pratique l'objet observé contient une liste d'observateurs (`listeners`) ;
- les observateurs implémentent des méthodes spécifiques de l'objet observé, déclenchées lors d'évènements marquants.

La classe simulation

Méthodes implémentées par les observateurs :

- `begin_match(team1,team2,state)` : déclenchée au début du match. Peut être utilisé pour initialiser les paramètres de la procédure d'optimisation.

La classe simulation

Méthodes implémentées par les observateurs :

- `begin_match(team1,team2,state)` : déclenchée au début du match. Peut être utilisé pour initialiser les paramètres de la procédure d'optimisation.
- `end_match(team1,team2,state)` : déclenchée à la fin du match.

La classe simulation

Méthodes implémentées par les observateurs :

- `begin_match(team1,team2,state)` : déclenchée au début du match. Peut être utilisé pour initialiser les paramètres de la procédure d'optimisation.
- `end_match(team1,team2,state)` : déclenchée à la fin du match.
- `begin_round(team1,team2,state)` : déclenchée à chaque coup d'envoi. Peut être utilisé pour régler les conditions environnementales du jeu (placement aléatoire du ballon).

La classe simulation

Méthodes implémentées par les observateurs :

- `begin_match(team1,team2,state)` : déclenchée au début du match. Peut être utilisé pour initialiser les paramètres de la procédure d'optimisation.
- `end_match(team1,team2,state)` : déclenchée à la fin du match.
- `begin_round(team1,team2,state)` : déclenchée à chaque coup d'envoi. Peut être utilisé pour régler les conditions environnementales du jeu (placement aléatoire du ballon).
- `end_round(team1,team2,state)` : déclenchée à chaque but. Peut être utilisé pour calculer le critère et passer à la prochaine valeur du paramètre à optimiser.

La classe simulation

Méthodes implémentées par les observateurs :

- `begin_match(team1,team2,state)` : déclenchée au début du match. Peut être utilisé pour initialiser les paramètres de la procédure d'optimisation.
- `end_match(team1,team2,state)` : déclenchée à la fin du match.
- `begin_round(team1,team2,state)` : déclenchée à chaque coup d'envoi. Peut être utilisé pour régler les conditions environnementales du jeu (placement aléatoire du ballon).
- `end_round(team1,team2,state)` : déclenchée à chaque but. Peut être utilisé pour calculer le critère et passer à la prochaine valeur du paramètre à optimiser.
- `update_round(team1,team2,state)` : déclenchée à chaque pas de jeu. Peut être utilisé pour contrôlé le temps alloué à chaque essai.

Exemple de code

```
class ParamSearch(object):
    def __init__(self, strategy, params, simu=None, trials=20, max_steps=1000000,
                 max_round_step=40):
        self.strategy = strategy
        self.params = params.copy()
        self.simu = simu
        self.trials = trials
        self.max_steps = max_steps
        self.max_round_step = max_round_step

    def start(self, show=True):
        if not self.simu:
            team1 = SoccerTeam("Team 1")
            team2 = SoccerTeam("Team 2")
            team1.add(self.strategy.name, self.strategy)
            team2.add(Strategy().name, Strategy())
            self.simu = Simulation(team1, team2, max_steps=self.max_steps)
        self.simu.listeners += self

        if show:
            show_simu(self.simu)
        else:
            self.simu.start()
```

Exemple de code

```python
class ParamSearch(object):
    def __init__(self, strategy, params, simu=None, trials=20, max_steps=1000000,
                 max_round_step=40):
        self.strategy = strategy
        self.params = params.copy()
        self.simu = simu
        self.trials = trials
        self.max_steps = max_steps
        self.max_round_step = max_round_step

    def start(self, show=True):
        if not self.simu:
            team1 = SoccerTeam("Team 1")
            team2 = SoccerTeam("Team 2")
            team1.add(self.strategy.name, self.strategy)
            team2.add(Strategy().name, Strategy())
            self.simu = Simulation(team1, team2, max_steps=self.max_steps)
        self.simu.listeners += self

        if show:
            show_simu(self.simu)
        else:
            self.simu.start()
```

```python
def begin_match(self, team1, team2, state):
    self.last = 0  # Step of the last round
    self.crit = 0  # Criterion to maximize (here, number of goals)
    self.cpt = 0  # Counter for trials
    self.param_keys = list(self.params.keys())  # Name of all parameters
    self.param_id = [0] * len(self.params)  # Index of the parameter values
    self.param_id_id = 0  # Index of the current parameter
    self.res = dict()  # Dictionary of results
```

Exemple de code

```
def begin_round(self, team1, team2, state):
    ball = Vector2D.create_random(low=0, high=1)
    ball.x *= GAME_WIDTH / 2
    ball.x += GAME_WIDTH / 2
    ball.y *= GAME_HEIGHT

    # Player and ball postion (random)
    self.simu.state.states[(1, 0)].position = ball.copy()  # Player position
    self.simu.state.states[(1, 0)].vitesse = Vector2D()  # Player acceleration
    self.simu.state.ball.position = ball.copy()  # Ball position

    # Last step of the game
    self.last = self.simu.step

    # Set the current value for the current parameter
    for i, (key, values) in zip(self.param_id, self.params.items()):
        setattr(self.strategy, key, values[i])

def update_round(self, team1, team2, state):
    # Stop the round if it is too long
    if state.step > self.last + self.max_round_step:
        self.simu.end_round()
```

Exemple de code

```
def end_round(self, team1, team2, state):
    # A round ends when there is a goal
    if state.goal > 0:
        self.crit += 1  # Increment criterion

    self.cpt += 1  # Increment number of trials
    if self.cpt >= self.trials:
        # Save the result
        res_key = tuple()
        for i, values in zip(self.param_id, self.params.values()):
            res_key += values[i],
        self.res[res_key] = self.crit * self.trials
        print(res_key, self.crit)

        # Reset parameters
        self.crit = 0
        self.cpt = 0

        # Go to the next parameter value to try
        key = self.param_keys[self.param_id_id]
        if self.param_id[self.param_id_id] < len(self.params[key]) - 1:
            self.param_id[self.param_id_id] += 1
        elif self.param_id_id < len(self.params) - 1:
            self.param_id_id += 1
        else:
            self.simu.end_match()

    for i, (key, values) in zip(self.param_id, self.params.items()):
        print("{}: {}".format(key, values[i]), end="   ")
    print("Crit: {}   Cpt: {}".format(self.crit, self.cpt))
```