Half Field Offense

Half Field offense offers a broad range of complexity, which allows to create very simple and very complex agents.

I decided to t**arget the simplest approach first**, and then increase complexity. So I **only use High Level features and Actions**.

First I will describe the environment's features and actions. Then I present my attempts to create learning agents.

High Level Environment

1. Environment Features

The High level features presented by the environment, are based on Barrett's work. They even pointed out the argument used by Barrett in his work: "There are many ways to represent the state of a game of half field offense. Ideally, we want a compact representation that allows the agent to learn quickly by generalizing its knowledge about a state to similar states without over-constraining the policy."

All features are encoded a floating point values and normalized to the range of [-1,1]. Invalid features are given a value of -2. There are a total of 10 + 6T + 3O + 2 high-level features. The features are as follows(Let T denote the number of teammates in the HFO game and O the number of opponents):

0 X position - The agents x-position on the field. See Figure 2.

1 Y position - The agents y-position on the field. See Figure 2.

2 Orientation - The global direction that the agent is facing.

3 Ball X - The ball's x-position on the field.

4 Ball Y - The ball's y-position on the field.

5 Able to Kick - Boolean indicating if the agent can kick the ball.

6 Goal Center Proximity - Agent's proximity to the center of the goal.

7 Goal Center Angle - Angle from the agent to the center of the goal.

8 Goal Opening Angle - The size of the largest open angle of the agent to the goal;

9 Proximity to Opponent - If an opponent is present, proximity to the closest opponent. Invalid if there are no opponents.

T Teammate's Goal Opening Angle - For each teammate i: is goal opening angle. Invalid if

agent is not playing offense.

T	Proximity from Teammate i to Opponent - For each teammate i: the proximity from the teammate to the closest opponent. This feature is invalid if there are no opponents or if teammates are present but not detected.
T	Pass Opening Angle - For each teammate i: the open angle available to pass to teammate i. If teammates are present but not detected, this feature is considered invalid and given the value of -2.
3T	X, Y, and Uniform Number of Teammates - For each teammate: the x-position, y-position and uniform number of that teammate.
3O	X, Y, and Uniform Number of Opponents - For each opponent: the x-position, y-position and uniform number of that opponent.
+1	Last Action Success Possible - Whether there is any chance the last action taken was successful, either in accomplishing the usual intent of the action or (primarily for the offense) in some other way such as getting out of a goal-collision state. 1 for yes, -1 for no.
+1	Stamina Agent's Stamina: Low stamina slows movement.

2. Environment Actions

The HFO domain provides support for both low-level primitive actions, mid-level, and high-level strategic actions. High-level discrete, strategic actions are available for moving, shooting, passing and dribbling. The actions are as follow:

- **Move**(): Re-positions the agent according to the strategy given by Agent2D. The Move command works only when the agent does not have the ball. If the agent has the ball, another command such as Dribble, Shoot, or Pass should be used.
- **Shoot**(): Executes the best available shot. This command only works when the agent has the ball.
- **Pass**(teammate uniform number): Passes to the teammate with the provided uniform number. Does nothing if the player does not have control of the ball or the requested teammate is not detected.
- **Dribble**(): Advances the ball towards the goal using a combination of short kicks and moves.
- **Catch**(): This goalie-specific action may be used to catch the ball.

- **Reduce Angle To Goal**(): Moves the agent to a point on the field, such that the kicker has the least open angle to the goal.
- **Defend Goal**(): Moves the agent to a point on a fixed line on the field, such that the kicker has the least open angle to the goal.
- **Go To Ball**(): Makes the agent go towards the ball.
- **Mark Player**(uniform number): Moves the agent so as to mark the player with the specified uniform number.
- **Reorient**(): Deal with loss of self or ball localization information and pay increased attention to surroundings.
- **NO-OP**: Indicates that the agent should take no action.
- **Quit**: Indicates to the agent server that you wish to terminate the HFO environment.

First attempt

Simplicity is the rule number 1! So I decided to use only five discrete features, only three actions and a simple reward function created by me, as described bellow:

1. Environment Features

1. **Position** – int [0, 6], subdivided the field in 6 regions, each integer signalizes one of these regions;
2. **Direction –** int [0,3]. 0=East, 1=North, 2=West, 3 == South.
3. **Has Ball** – int (0,1). 1 if agent can kick, else 0.
4. **Should Shoot** – int (0,1). Agent's goal opening angle > 20%;
5. **Opponent is close** – int (0,1). Opponent is close;

2. Actions

1. **Move -** Re-positions the agent according to the strategy given by Agent2D. The Move command works only when the agent does not have the ball. If the agent has the ball, another command such as Dribble, Shoot, or Pass should be used.
2. **Dribble -** Advances the ball towards the goal using a combination of short kicks and moves.
3. **Shoot - Shoot():** Executes the best available shot. This command only works when the agent has the ball.

3. Rewards

- - 10000 – the game ends without scoring goal;
- + 10000 – score goal;
- + 2 – select the right action, which means that the agent will "move" when it does not have the ball, and it will only "dribble" and "shoot" when it has the ball;
- - 3 – for each time-step, the idea was to motivate the agent to goal as fastest as possible;

4. Train Q-agent (V1)

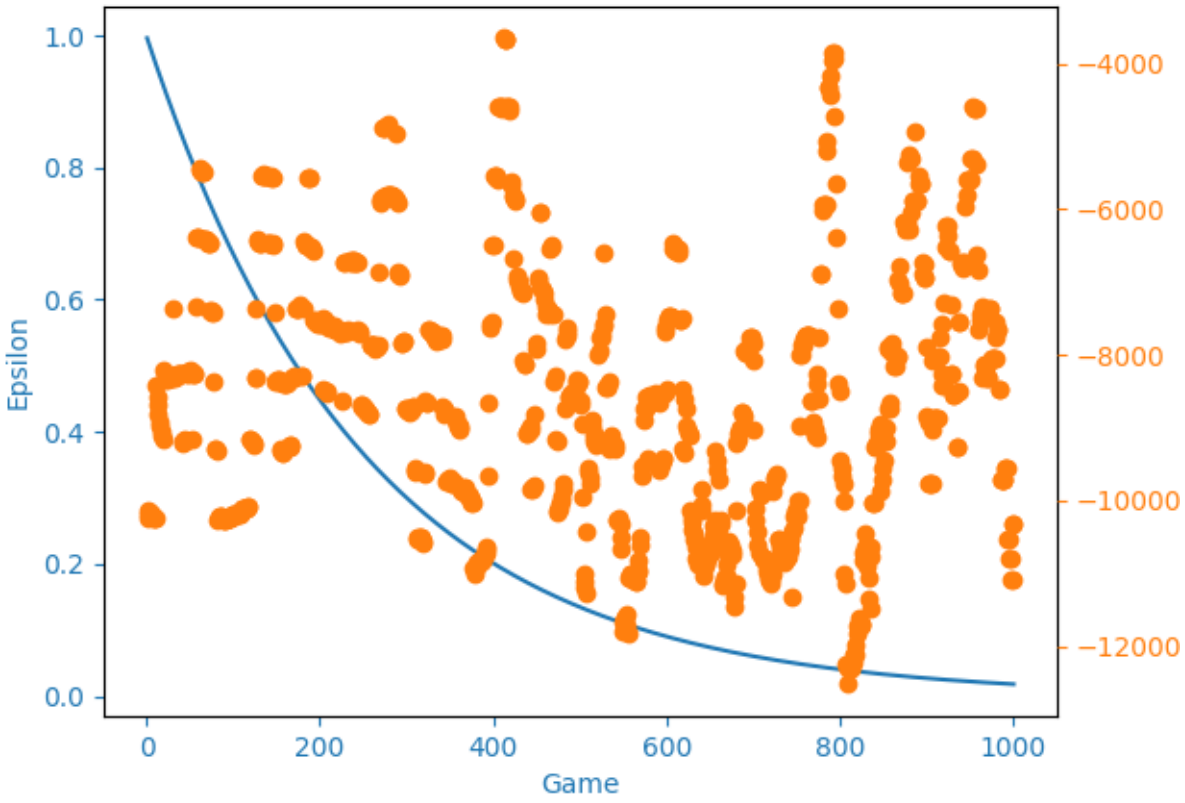
Game parameters:

- Number of **teammates: 0;**
- Number of **opponents: 1;** (Helios Goalie)
- Number of **episodes: 1000;**

Q agent parameters:

- Learning rate: 0;
- Epsilon: 1; Epsilon decrescent: 0.996; Epsilon end: 0.01;
- Discount factor: 0.9;
- **Q learning table dim: 192 states** = environment features * number of actions;

Result:



Discuss Results:

After 1000 games, the agent is not behaving as expected. The agent is choosing actions which are “not legal”. Despite promoting it throw rewards

4. Train Q-agent (V2) - 10/03/2020

Game parameters:

- Number of **teammates: 0;**
- Number of **opponents: 0;**
- Number of **episodes: 1000;**

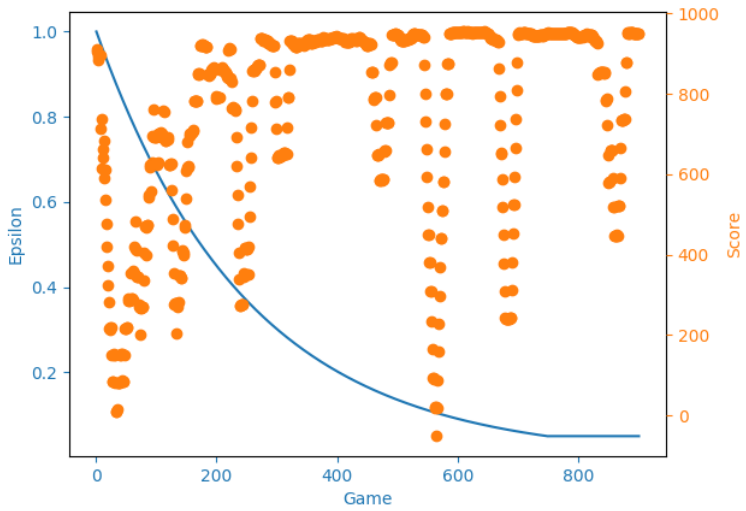
Q agent parameters:

- Learning rate: 0.1;
- Epsilon: 1; Epsilon decrescent: 0.996; Epsilon end: 0.05;
- Discount factor: 0.9;
- **Q learning table dim: 192 states** = environment features * number of actions;

Rewards:

- - 500 – the game ends without scoring goal;
- + 1000 – score goal;
- - -1 – for each time-step, the idea was to motivate the agent to goal as fastest as possible;

Result:



Discuss Results:

After 1000 games, the agent is always scores. He learn to move to the goal and shoot. This simple test was just to test if the agent is able to learn to score by himself.