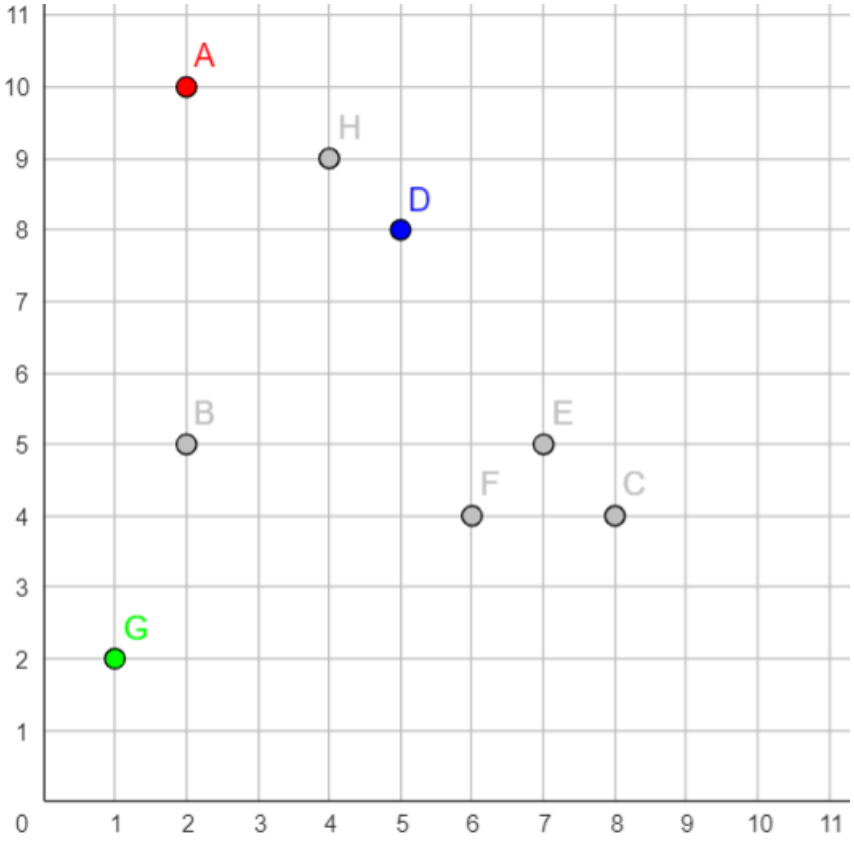
Tarea 2

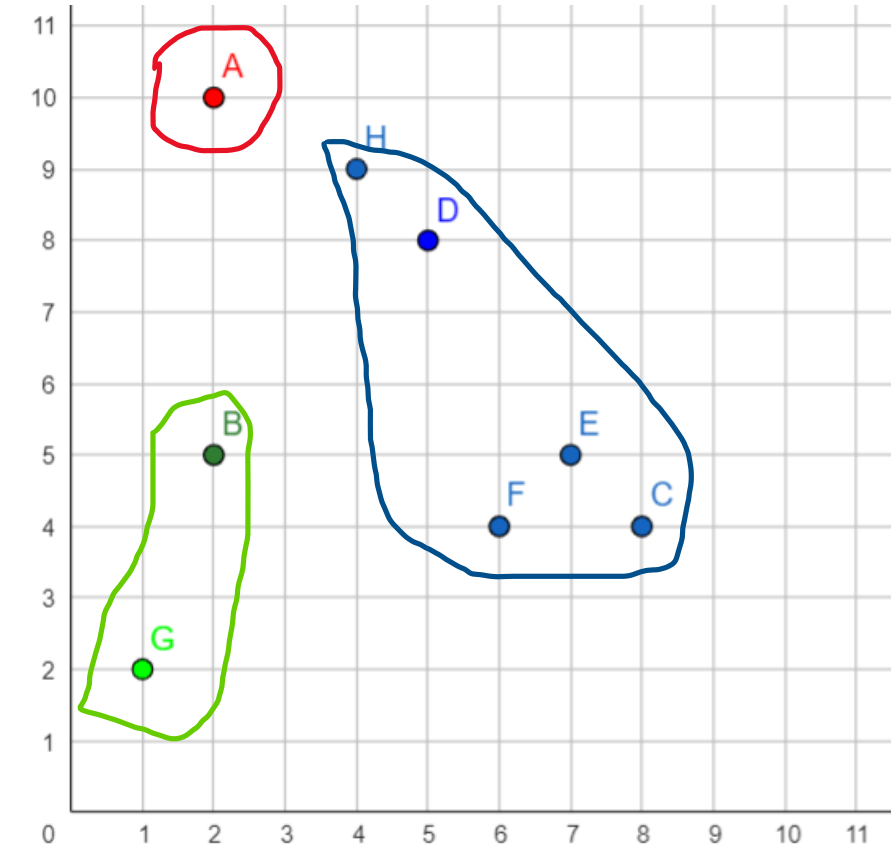
Inteligencia Artificial

Nombre: Miguel Villa Rios

1.



Iteración: 0



Iteración: 1

Cluster 1 = A

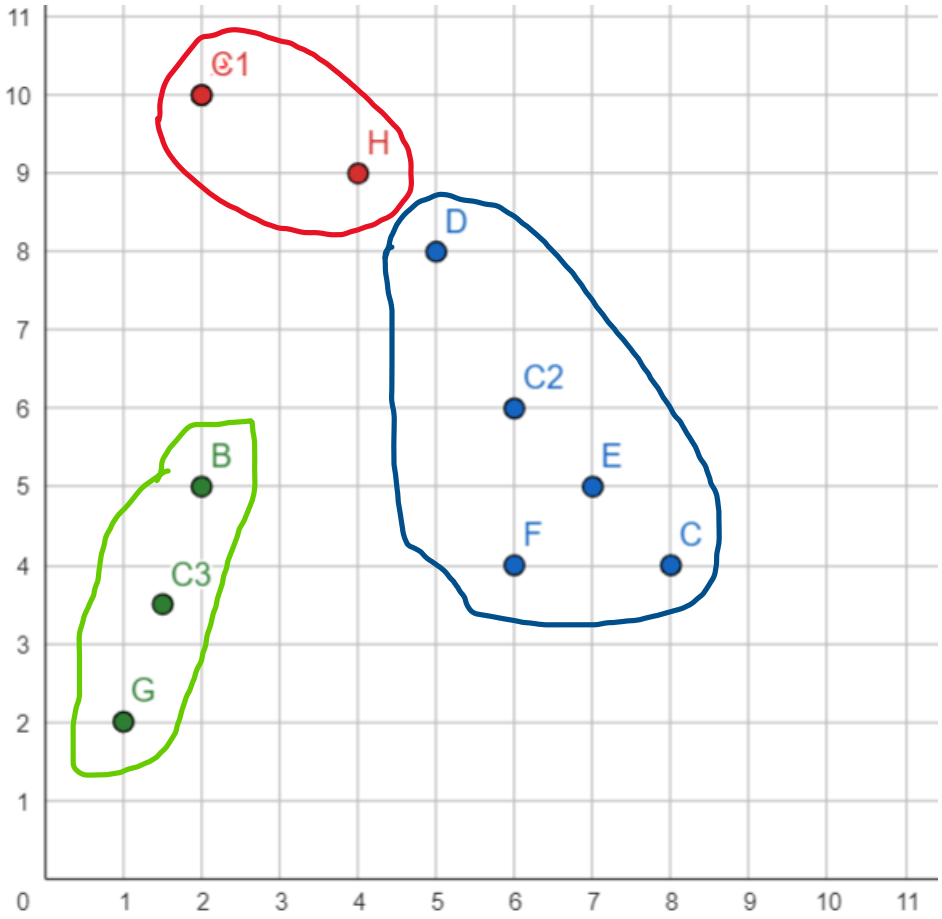
Cluster 2 = C, D, E, F, H

Cluster 3 = B, G

Centroide 1 = A

Centroide 2 = D

Centroide 3 = G



Iteración 2

Cluster 1 = A, H

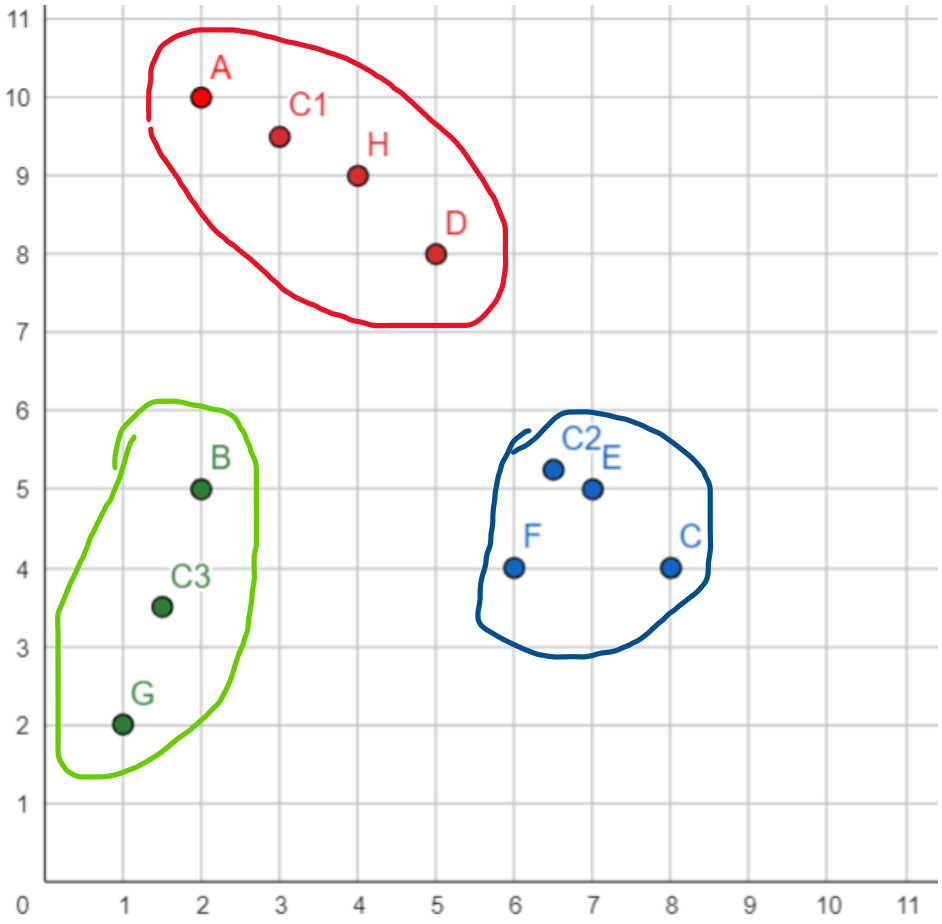
Cluster 2 = C, D, E, F

Cluster 3 = B, G

Centroide 1 = C1

Centroide 2 = C2

Centroide 3 = C3



Iteración 3

Cluster 1 = A, D, H

Cluster 2 = C, E, F

Cluster 3 = B, G

Centroide 1 = C1

Centroide 2 = C2

Centroide 3 = C3

2.1 Imprimir en un archivo los pares [episodio, reward acumulado]

```
reward_output.open("Rewards.txt", ios_base::app);
Initialize_environment();//Initialize the features
```

```
    finalrw[i]=cum_reward;
    cout << " Total reward obtained: " <<finalrw[i] <<"\n";
    // 1.-(5 puntos) Imprimir en un archivo los pares [episodio, reward acumulado]
    reward_output << i << " " << finalrw[i] << "\n";



}
reward_output.close();
```

2.2 Implemente la estrategia (policy) epsilon-greedy para la selección de acciones (con un valor de epsilon = 0.05)

```
if (action_sel == 2) // epsilon-greedy
{
    float epsilon = 0.05; // Valor de epsilon

    // Genera un número aleatorio entre 0 y 1
    float rand_num = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);

    if (rand_num < epsilon) // Exploración: elige una acción aleatoria
    {
        action_taken = rand() % 4;
        return action_taken;  // Devuelve un número aleatorio entre 0 y 3 (que corresponde a las 4 acciones posibles)
    }
    else // Explotación: elige la acción con el mayor valor de Q
    {
        float max_q = Qvalues[x_pos][y_pos][0];
        int max_action = 0;

        for (int action = 0; action < 4; ++action)
        {
            if (Qvalues[x_pos][y_pos][action] > max_q)
            {
                max_q = Qvalues[x_pos][y_pos][action];
                max_action = action;
            }
            else if (Qvalues[x_pos][y_pos][action] == max_q)
            {
                // Si hay múltiples acciones con el mismo valor máximo de Q, elige aleatoriamente entre ellas
                if (rand() % 2 == 0)
                {
                    max_action = action;
                }
            }
        }
        action_taken = max_action;
        return action_taken; // Devuelve la acción con el mayor valor de Q
    }
```

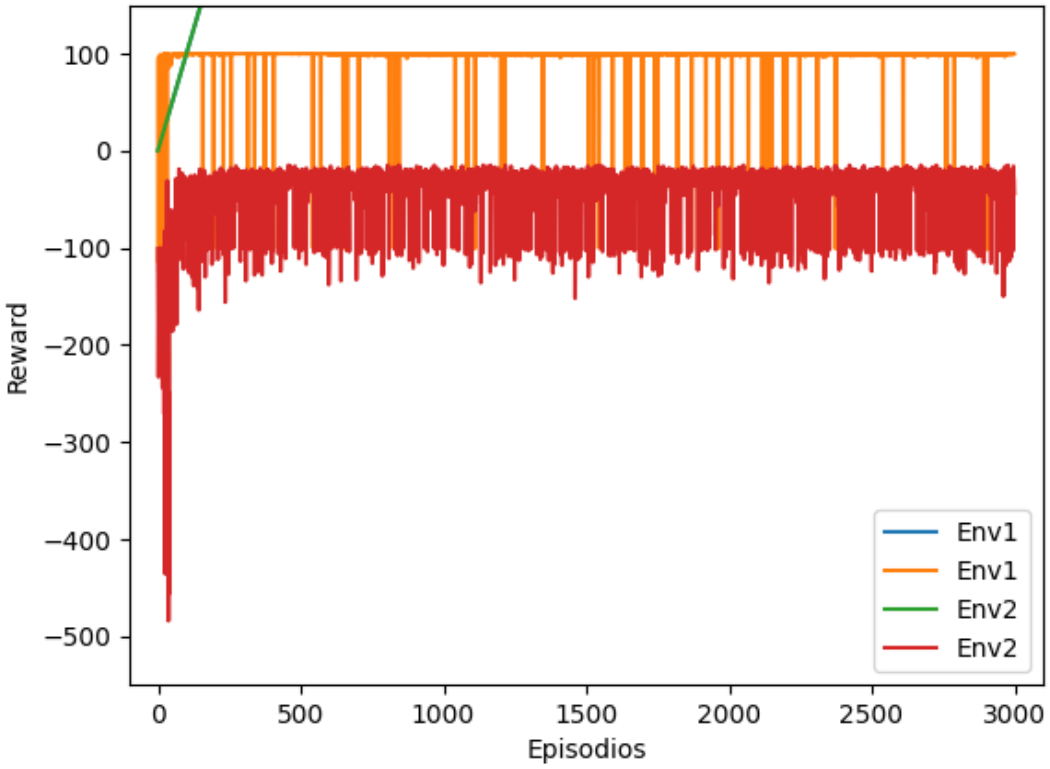
2.3 Implemente acciones estocásticas (que sólo un 80% de las veces el agente se mueva donde se le indica, el 10% de las veces se mueve a la derecha de la dirección deseada y el otro 10% de las veces se mueve a la izquierda de la dirección deseada).

```
void move(int action)
{
    prev_x_pos=x_pos; //Backup of the current position, which will become past position after this method
    prev_y_pos=y_pos;

    //Stochastic transition model (not known by the agent)
    //Assuming a .8 prob that the action will perform as intended, 0.1 prob. of moving instead to the right, 0.1

    if(stochastic_actions)
    {
        //Code here should change the value of variable action, based on the stochasticity of the action outcome
        float rand_num = static_cast<float>(rand()) / static_cast<float>(RAND_MAX);

        if (rand_num < 0.8){

        }
        else if (rand_num < 0.9)
        {
            action = (action + 1) % 4; // Right
        }
        else if (rand_num < 1.0)
        {
            action = (action + 3) % 4; // Left
        }

    }
```

2.4 Implemente y evalúe el algoritmo Q-learning en los ambientes 1 y 2. Incluya un diagrama de la curva de aprendizaje (episodio vs reward acumulado) en ambos casos.

```
void update_q_prev_state() //Updates the Q value of the previous state
{

    //Determine the max_a(Qvalue[x_pos][y_pos])

    //Update the Q value of the previous state and action if the agent has not reached a terminal state
    if(!( ((x_pos==goalx)&&(y_pos==goaly)) ||((environment==1)&&(x_pos==goalx)&&(y_pos==(goaly-1)))||((environment==2)&&(x_pos>0)&&(x_pos<goalx)&&(
    {


        Qvalues[prev_x_pos][prev_y_pos][action_taken]= (1-learn_rate)*Qvalues[prev_x_pos][prev_y_pos][action_taken] +
        (learn_rate*(reward[x_pos][y_pos] + (disc_factor*Qvalues[x_pos][y_pos][action_taken]))); //How should the Q values be updated?

    }
    else//Update the Q value of the previous state and action if the agent has reached a terminal state
    {
        Qvalues[prev_x_pos][prev_y_pos][action_taken]= (1-learn_rate)*Qvalues[prev_x_pos][prev_y_pos][action_taken] +
        (learn_rate*(reward[x_pos][y_pos] + (disc_factor*Qvalues[x_pos][y_pos][action_taken])));

    }
```

```
void Qlearning()
{
    //Follow the  steps in the pseudocode in the slides
    move(action_selection());

    update_q_prev_state();
    cum_reward=cum_reward+reward[x_pos][y_pos]; //Add the r
}
```



El color naranjo son los datos del ambiente 1 y el rojo para el ambiente 2.