Assignment 3

Romain Bazin

Approach to generalization

In this project, the goal is to train an agent to play the game Ms Pac-man from the Atari 2600 published in 1981. To play this game, we use the ALE (Arcade Learning Environment), which renders the game as a 210 by 160 pixels of values between 0 and 127 (7 bit). The actions in this game are to move Ms Pac-man UP, DOWN, LEFT or RIGHT. To earn points, you have to eat pellets, food or ghosts when possible. The higher the score, the better.

With the basic Q-learning algorithm approach, where the Q values for all state-action pairs are stored in a table, the table would have $4 * 128^{210*160}$ entries, which is way too much for our application. This is why we have to use generalization techniques such as function approximation.

Function approximation alleviates the memory and computation load of the Q table by introducing a number of features, participating by the Q value of a state-action pair thanks to a weighted linear function :

$$\hat{Q}(s,a) = \theta_1 f_1(s,a) + \theta_2 f_2(s,a) + ... + \theta_n f_n(s,a)$$

The parameters, the theta values, represent the weights of each feature. These are the parameters that need to be learned by the Q-learning algorithm. These parameters are learned using the TD (Temporal Difference) algorithm :

$$\theta_i \leftarrow \theta_i + \alpha\,[R(s) + \gamma \max_{a'} \hat{Q}_\theta(s',a') - \hat{Q}_\theta(s,a)] \frac{\partial \hat{Q}_\theta(s,a)}{\partial \theta_i}$$

This puts all the importance on selecting the right features for the game. These features are extracted from the frames of the game.

The features selected are the following :

- distance from nearest food
- distance from nearest scared ghost
- distance from nearest active ghost
- number of foods remaining

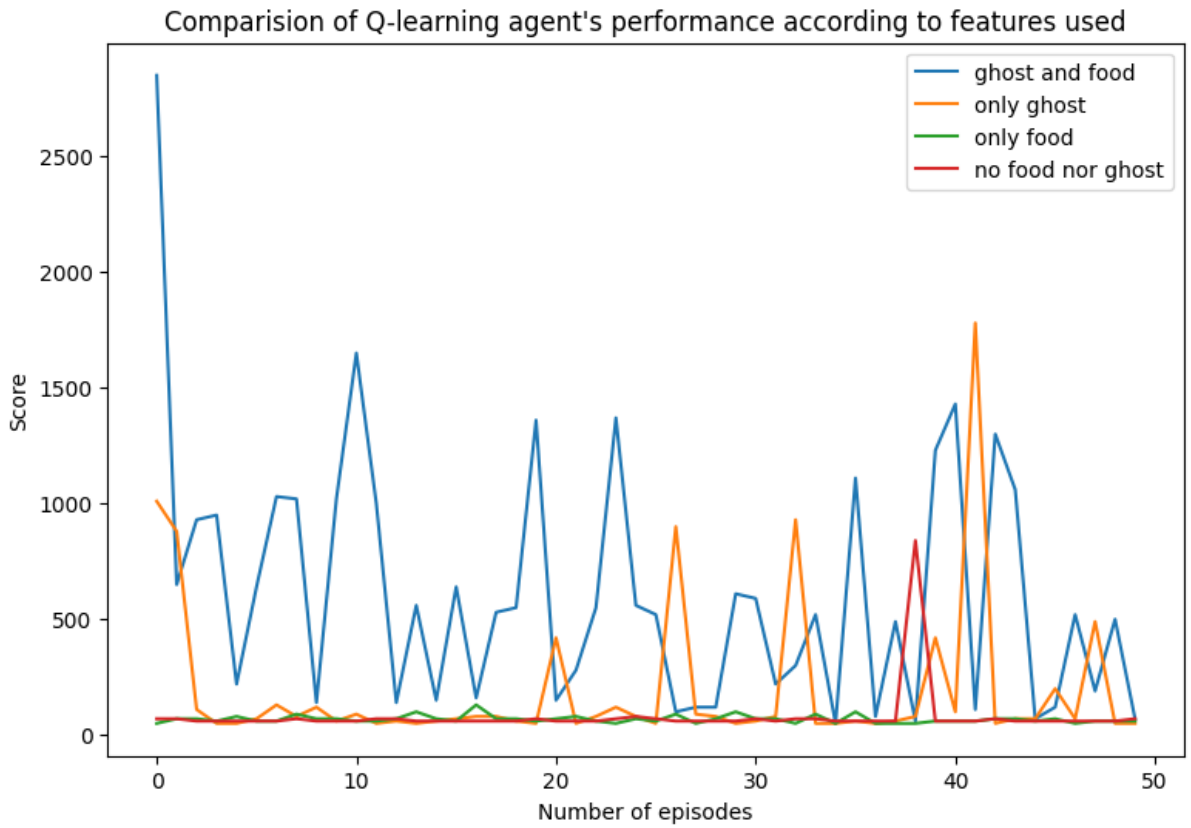
The Q value equation becomes :

$$Q(s,a) = \frac{\theta_1}{distance\ nearest\ food} + \frac{\theta_2}{distance\ nearest\ scared\ ghost} + \frac{\theta_3}{distance\ nearest\ active\ ghost} + \frac{\theta_4}{number\ of\ foods\ remaining}$$

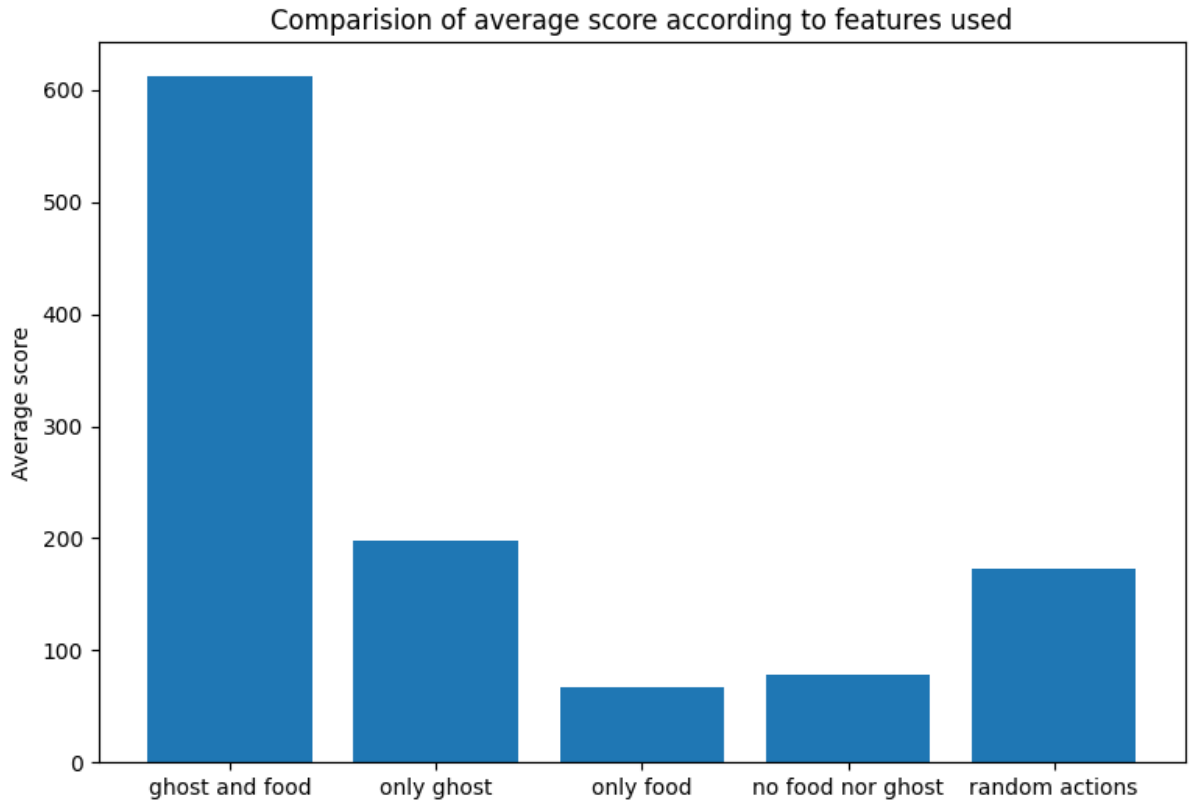
These features were selected with the idea of keeping Ms Pac-man alive as long as possible.

Results of the generalization

In order to assess the performance of these features, I set the seed of the game to 1 and performed the training of the agent over 50 episodes using 3 different combinations of the features. A typical training attempt over 50 episodes looks like the following.



I repeated this attempt over multiple seeds and computed the average score of the agent on multiple games according to the features used.



The results are quite clear, the agent is way better when using ghost and food features at the same time.

We may reflect on the very low scores of using only the ghost features or only the food features by saying that either the agent would only run away from the ghosts without earning much points or it would run at the food without caring about the ghosts and dying very fast.

On the contrary, when we use both features, the agent is capable of running away from the ghost while still earning points thanks to the food.

Approach to exploration

In order to explore different states of the game, I implemented two types of exploration techniques.

First, a simple ϵ-greedy method that would select an action randomly with probability ϵ. The rest of the time, the agent would select the action based on the Q-learning algorithm.

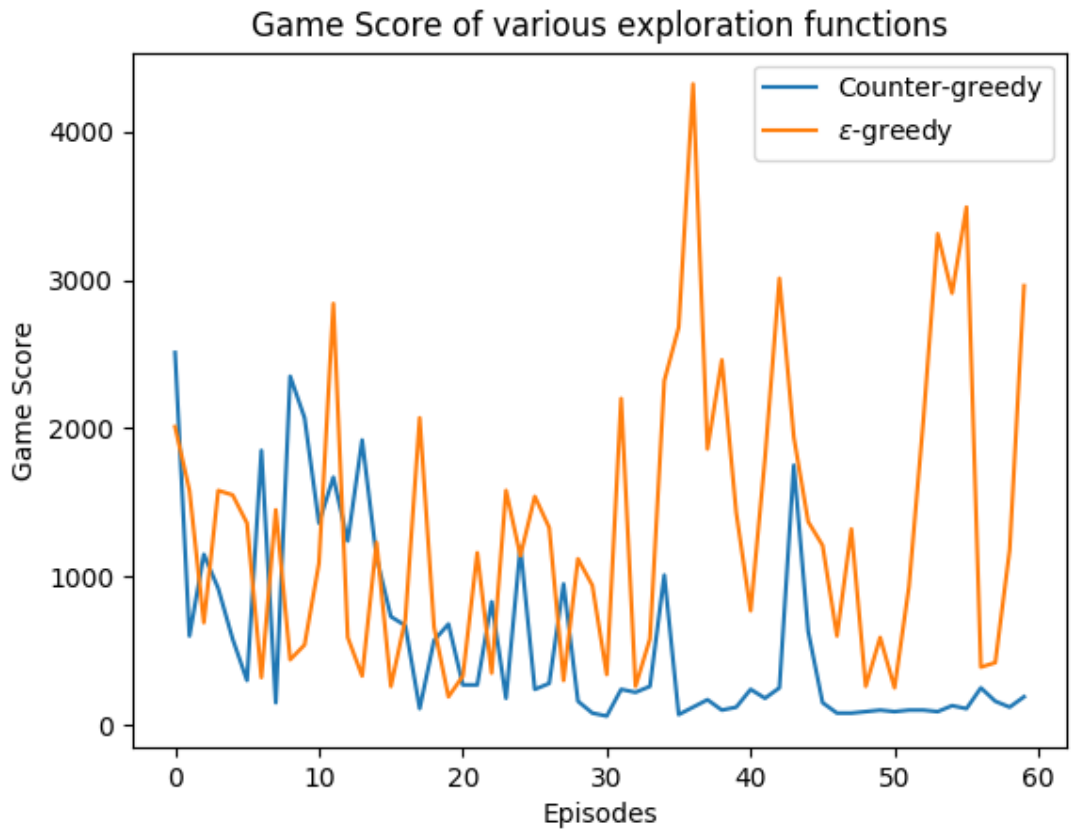
The second method tried is to use the counter-greedy approach, which is a slight modification of the previous method as it also relies on randomly selecting an action

with a certain probability but it takes into account how long the agent has been in a certain area and increases the probability of selecting a random move accordingly.

Both methods were selected because they are very easy to implement and can still yield great results by adding a ton of exploration to the agent. Still, it might be slower to train an agent this way because it wouldn't take into account if the state-action pair select has already been explored or not.

Results of the exploration

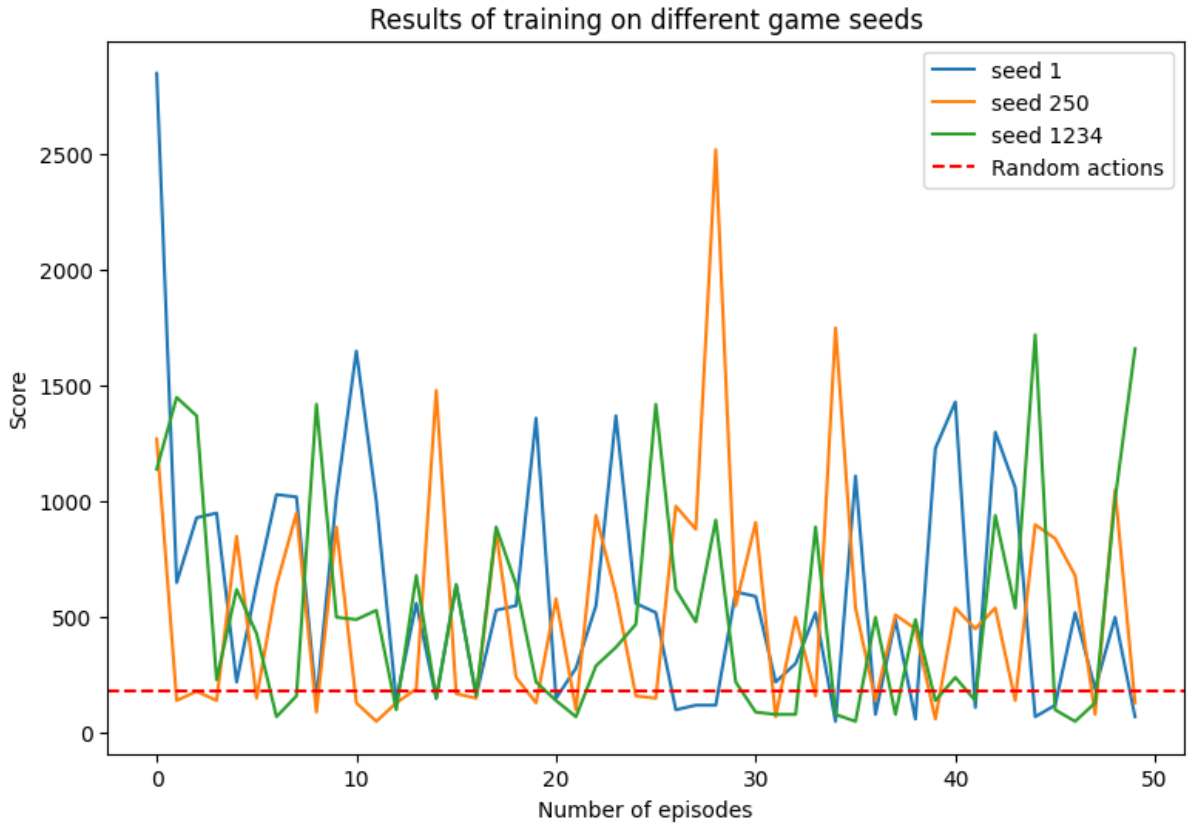
To compare the results of both exploration methods, I trained the agent with both and plotted the evolution of the score over 60 episodes of training.



We can see the ϵ-greedy method yields way better results than the counter greedy one. This may be explained by the fact that the counter greedy method reduces the total amount of random moves taken in the game, reducing the exploration of the agent.

Agent performance

We can see the performance of the agent on different seeds on the following graph.



We can see that the agent achieves similar results on different game seeds, meaning the results are generalizable. As a baseline, I provide the average score value of an agent taking only random actions. We can see the trained agent is far better than the random one *on average*. Indeed, even after the training, the agent still makes deadly mistakes which will lead to bad scores, this probably underlines that the selected features are not appropriate to all kinds of situations.

Let's now reflect on the actions of the agent and how it learns to play the game :

Initially, the agent starts to move in one direction from the starting zone and starts to realize that it is a good thing to eat food. When the agent first encounters the ghosts, the agent doesn't realize the danger and loses a life to the ghost. It takes a few lives lost for the agent to understand that it should avoid colliding with the ghosts. It starts developing the strategy of avoiding the ghosts by running away from the nearest ghost in the direction that makes it get away from the ghost.

When the agent accidently eats a power pellet, it does not understand that the ghosts give lots of points when eaten until a ghost accidently goes into the agent.

The agent then realizes that the scared ghosts are great and, after this game event happens a few more times, starts to hunt down the scared ghosts and chase them.

On a last note, I want to mention I didn't try to optimize any of the hyperparameters such as α, ϵ, γ so the results could probably be improved by doing so as I found a high variance in the results when testing different parameters combinations.

The parameters used were $\alpha = 0.2$, $\epsilon = 0.10$, $\gamma = 0.95$