**AI Project 2 – Siegfried Lein, Max Hallemeesch**

**Question 1:** An MDP is a model in which an agent takes actions in an environment to maximize a reward.A Markov Decision Process becomes a reinforcement learning problem when the agent learns to take actions based on the rewards it receives for those actions. The agents tries to learn a policy that maximizes the reward it receives over time.

**Question 2:**

TD – learning:

Sample = Reward + Discounted Value Estimate of the Next State - Current Value Estimate

Q-Learning:

Sample = Reward + Discounted Maximum Q-Value of the Next State - Current Q-Value

TD-learning updates the value estimate of a state, while Q-learning updates the action-value estimate (Q-value) of a state-action pair. This means that in TD learning, the value estimate represents the expected future return of a state, while in Q-learning, the Q-value represents the expected future return of taking a specific action in a specific state.

**Question 3:**

Q-learning is able to learn the optimal action-value function, which allows the agent to select the optimal action in each state. In contrast, TD learning of values only learns the value of a state, which may not necessarily correspond to the optimal action in that state.

**Question 4:**

The discount factor determines how important future rewards are in the learning process. If the γ is larger, the agent will find future rewards more important and vice versa. If the γ = 0, the agent will only consider immediate rewards and give no importance to future rewards. This greedy approach might not be the best. A γ results in the agent giving an equal amount of importance to immediate rewards and future rewards. This will theoretically give the correct decision to the agent but will result in an impossible amount of time to calculate a choice as it needs to take an infinite number of future rewards into account.

**Question 5:**

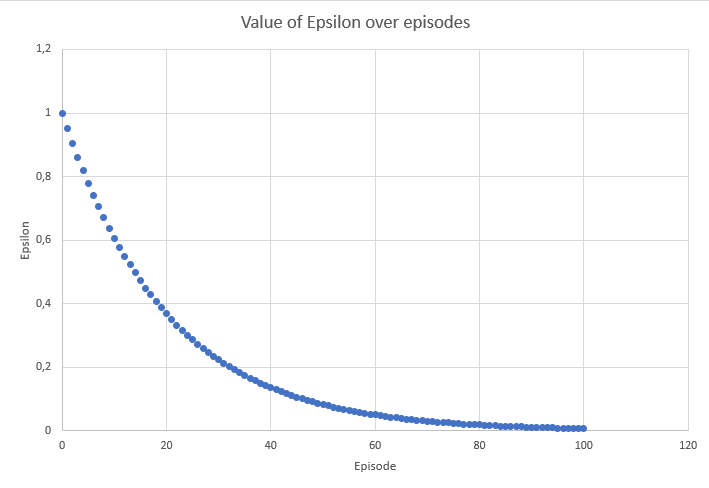
When the epsilon value is equal to 0.1 the agent starts discovering randomly but once a path is found of slightly positive values it will keep taking and reinforcing this path without much extra discovery. Not much of the final map is discovered or filled in with Q-values.

An epsilon value of 0.9 results in more discovery even when the agent has created a path of positive values to the goal. This results in a fully discovered map where every state-action pair has a close to correct Q-value. Because the agent discovers more, ignoring the established Q-value, it will often take the wrong path. This is why this agent will have a worse overall score than the other one with an epsilon value of 0.1.

The epsilon value determines the chance that the agent either takes a random move or a move based on the Q-values. The higher the epsilon value, the higher the chance of a random move, the more discovery is done. The lower the epsilon value, the lower the chance of a random move, the higher the overall score.

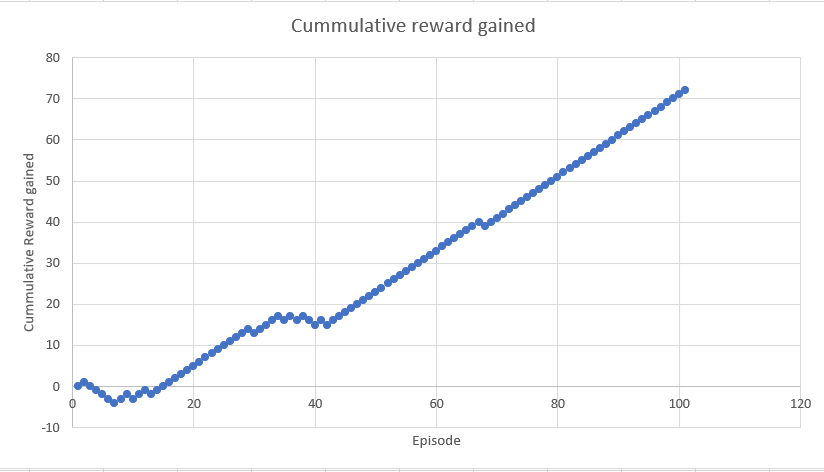
**Question 6:**

Decay = 0.05.



We expect the agent to behave quite random in the first 15 episodes as it’s epsilon is still decently high then. This is good because in the beginning it needs a discovery fase. After this it will behave less and less randomly and start taking the optimal rout based on the calculated Q-values.

**Question 7:**



As we expect we see a random reward gained in the early stage of the training. Here the agent is still discovering and can accidentally reach the +1 or -1 terminal stage. After some training and when the epsilon value is lowered a bit, we see a consistent amount of reward gained. The agent has now found the way to the +1 terminal stage and takes it often because it isn’t discovering as much anymore.

**Question 8:**

Naive Q-learning struggles with increasing amounts of state-action pairs since it just stores the values in tables. This means that the tables become increasingly bigger and the chance that the agent is in a state that it has seen before decreases. Approximate Q-learning uses a function bases on heuristics. This way the agent always has a decent guess to what the best choice is given a certain state-action pair no matter how big the state-action space.

**Question 9:**

One feature you could add is a variant of the ("#-of-ghosts-1-step-away") . This variant would be called ("closest-scared-ghost") and would be -1 when there are no scared ghosts and else the distance to the closest scared ghost. When a power pellet is eaten this should make sure that the ("#-of-ghosts-1-step-away") feature gets negated and the agent goes to the ghosts.

The expression for the linear value function becomes:

Q(s, a) = w1 ▪#-of-ghosts-1-step-away + w2 ▪ eats-food + w3 ▪ closest-food + w4 ▪ closest-scared-ghost + w5 ▪ bias