Articial Neural Networks Miniproject Report: Nim

Bernhard Schiehl (349305), Noora Torpo (350433)

June 6, 2022

Contents

1 Q-Learning

1.1 Learning from experts

Question 1. See figure 1.

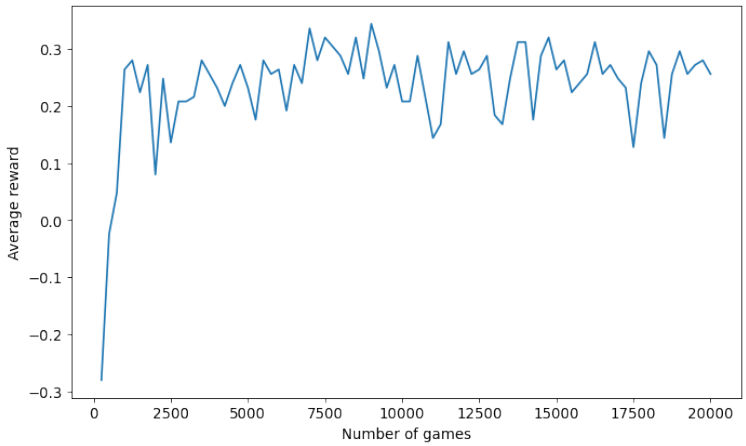


Figure 1: Average reward for every 250 training games of the Q-learning agent with $\epsilon = 0.2$. In the beginning the agent loses more often than it wins but after about 1250 games the agent learned to play better than its opponent. The average reward then fluctuates between 0.1 and 0.3.

1.1.1 Decreasing exploration

Question 2. See figure 2.

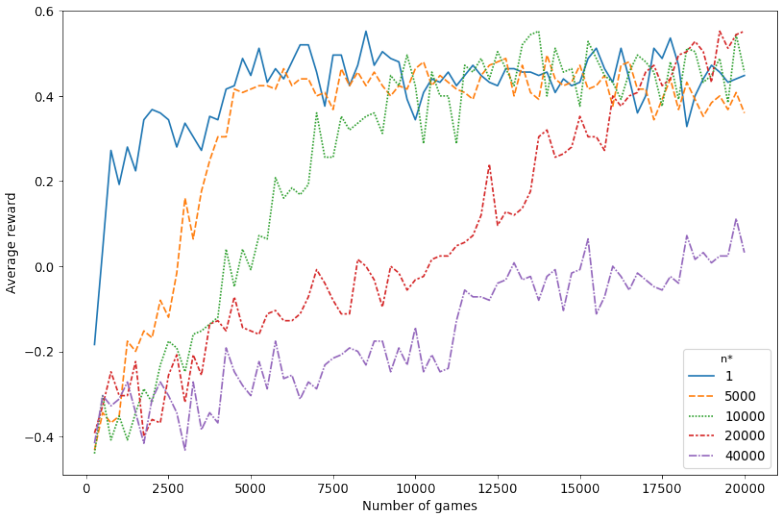


Figure 2: Average reward for every 250 training games of the Q-learning agent with a decreasing value of ϵ over time, for different values of n^*. Every agent reaches a higher average reward than the agent with fixed $\epsilon = 0.2$ at some point, except the agent with $n^* = 40000$. After some amount of games, the average reward for those agents oscillates around 0.4. The biggest influence of n^* is the number of games needed to reach the point where average reward does not significantly improve anymore. This point roughly corresponds to the number of exploratory games. Of course, the agent with $n^* = 1$ is an exception to this, as it reaches its plateau after about 5000 games. Furthermore, it starts off with a higher average reward than all other agents and very quickly reaches positive average reward. After 20000 games, the agent with $n^* = 40000$ has only reached an average reward of about 0. However, it is expected that if one were to continue training for another 20000 games then this agent would reach an average reward similar to that of the other agents.

Question 3. See figure 3.

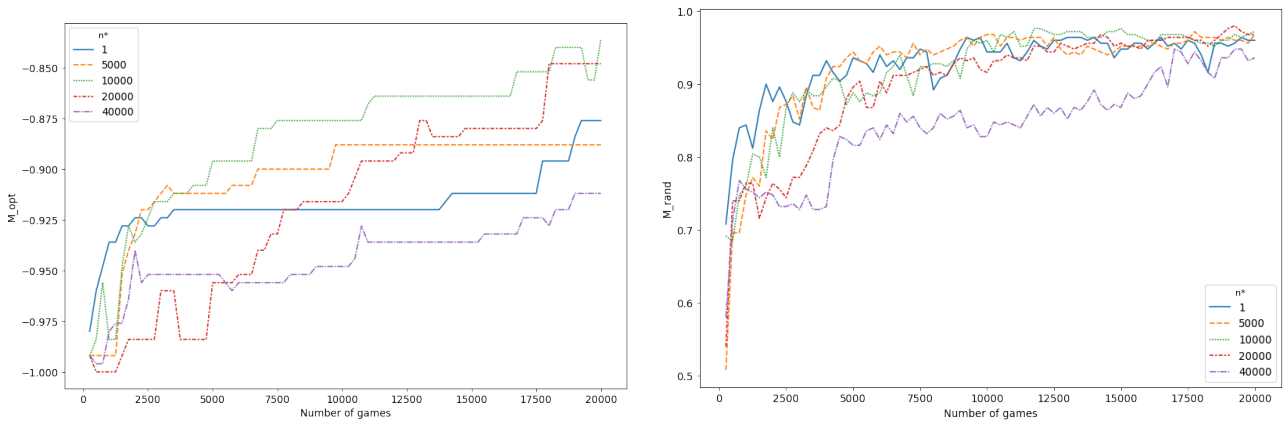


Figure 3: M_{opt} and M_{rand} plotted for every 250 training games of the Q-learning agent with deceasing exploration against Opt(0.5), for different values of n^*. The agents do not perform well against the optimal player, with the peak M_{opt} being around -0.85. However, the value of n^* influences performance, with $n^* = 10000$ resulting in the best and $n^* = 40000$ resulting in the worst performance. The graphs for M_{rand} look more comparable to the ones for average reward, as the agents all perform similarly well after the 20000 games. However, the number of games needed to reach the best performance is not as clearly influenced by the number of exploratory games.

1.1.2 Good experts and bad experts

Question 4. See figure 4.

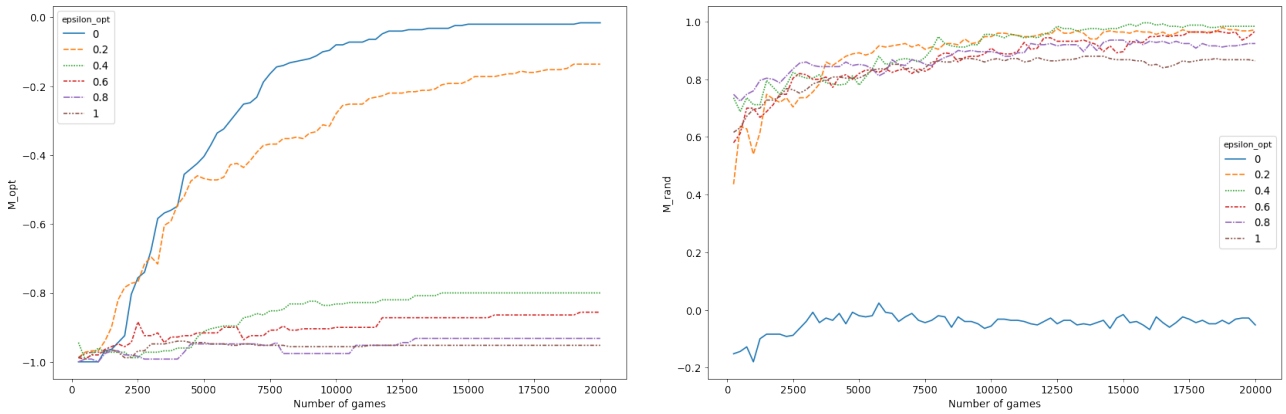


Figure 4: M_{opt} and M_{rand} plotted for every 250 training games of the Q-learning agent with decreasing exploration and $n^* = 20000$ against the optimal player with different values of ϵ. The M_{opt} plot shows that the performance against the optimal player is best when training against the optimal player. In this case the Q-learning agent almost reaches a M_{opt} value of 0, which is close to the best possible. It is expected that the performance against a certain opponent is best when first training against that exact opponent. Decent results are also obtained while training against Opt(0.2), in which case the agent reaches a peak M_{opt} value of about -0.2. When training against opponents with higher degrees of randomness, the performance against the optimal player dramatically drops. Agents that train against an ϵ-greedy optimal opponent with $\epsilon = 0.4$ or higher only achieve very poor results. An explanation could be that during training strategies get reinforced that only work against opponents making suboptimal moves. Regarding the M_{rand} score all agents perform similarly, getting scores from about 0.8 to 1, showing that they learned the game decently well. An exception is the agent that trained against Opt(0). The difference is staggering, as this agent even seems to lose slightly more often than win. This could be explained if the agent often encountered states during its games against the random opponent that it was never confronted with during training. In those game states it cannot do much better than random guessing.

Question 5. The highest M_{opt} value of -0.016 was achieved after 19250 games while training against Opt(0). The highest M_{rand} value of 0.996 was achieved after 15750 games while training against Opt(0.4).

Question 6. Q-values depend only on the state and the action taken in that state. Moreover, the winning strategy of finishing every move with a nim-sum of 0 does not depend on the opponent. So if $Q_1(s, a)$ and $Q_2(s, a)$ are both optimal then they both represent the winning strategy. However, there might be multiple winning moves in a given state, so $Q_1(s, a)$ and $Q_2(s, a)$ might not have the exact same values. In any case, $Q_1(s, a)$ should be found more quickly than $Q_2(s, a)$.

1.2 Learning by self-practice

Question 7. See figure 5.

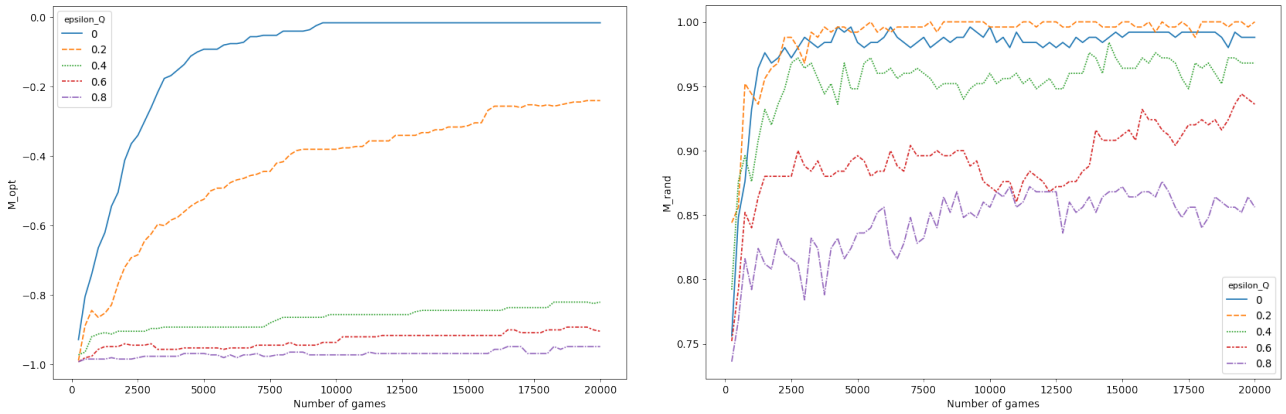


Figure 5: M_{opt} and M_{rand} plotted for every 250 training games of the Q-learning agent playing against itself, for different values of ϵ. The M_{opt} values look very similar to the ones obtained after training against experts with different ϵ values. Only Q-learning agents with low ϵ values perform well against the optimal player. The agent with $\epsilon = 0$ comes close to achieving M_{opt} values of 0 and $\epsilon = 0.2$ leads to a peak M_{opt} of about -0.3. Against the random player all agents perform reasonably well, but low ϵ values clearly play the best.

Question 8. See figure 6.

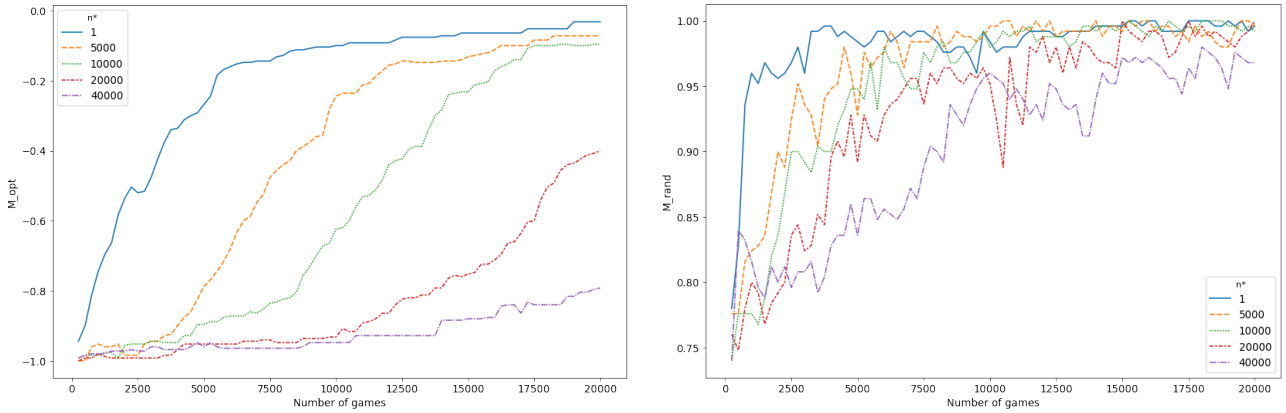
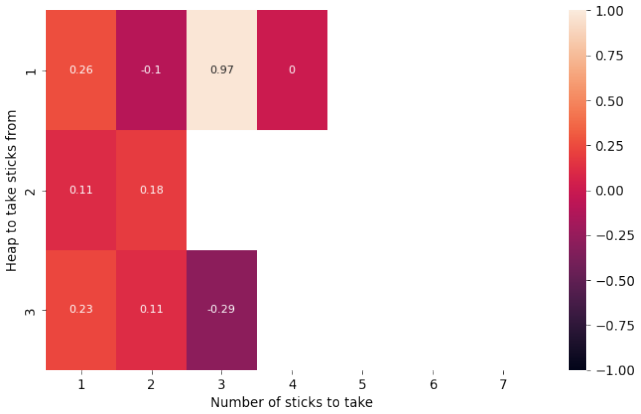


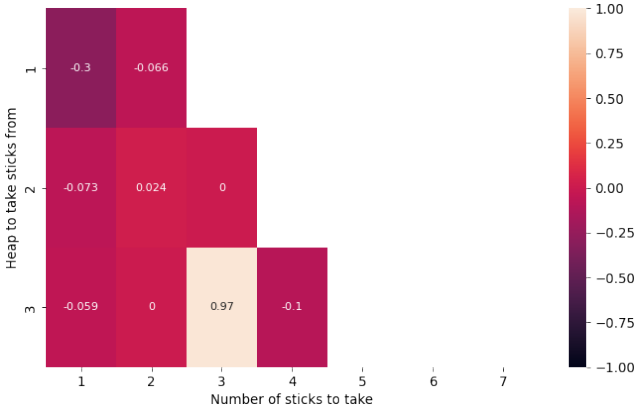
Figure 6: M_{opt} and M_{rand} plotted for every 250 games while training the Q-learning agent against itself with decreasing exploration for different values of n^*. Every agent seems to reach M_{opt} values of about -0.1 at some point, with n^* influencing how much time the agent needs. For most of the training games the M_{opt} scores are higher for agents with a low value of n^*, but after about 17500 games the agents with n^* values of 1, 5000 and 10000 have similar performance levels as they reached their full potential. This also roughly holds true for the M_{rand} scores, but here the agents with $n^* = 20000$ and $n^* = 40000$ seem to plateau sooner.

Question 9. The highest M_{opt} value of -0.016 was achieved after 9500 games while training with fixed $\epsilon = 0$. The highest M_{rand} value of 1.0 was achieved after 5500 games while training with fixed $\epsilon = 0.2$ and after 10500 games while training with fixed $\epsilon = 0.1$.

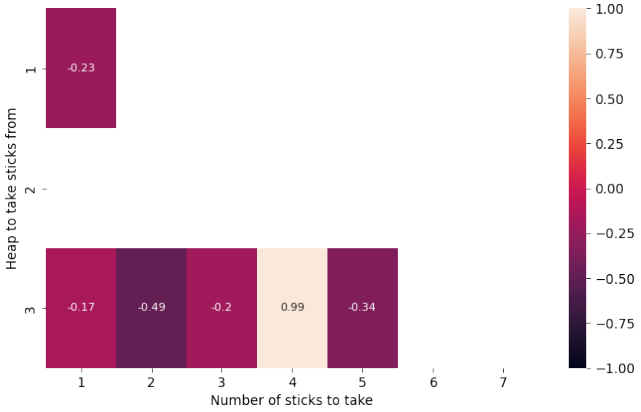
Question 10. See figure 7.



(a) State 4/2/3



(b) State 2/3/4



(c) State 1/0/5

Figure 7: Q-values of available actions in three states visualized as heatmaps. The agent that learned these Q-values trained against itself for 20000 games with decreasing exploration and $n^* = 10000$. Figure 7a shows the Q-values for actions in the state with 4 sticks on the first heap, 2 sticks on the second heap and 3 sticks on the third heap. The winning action of taking three sticks from heap 1 got reinforced way more than the other actions. The other actions also mostly have positive Q-values, but the winning move has the maximum Q-value by a wide margin. The state of figure 7b is a permutation of the heap numbers of the previous state. Again, we can see that taking 3 sticks from the heap containing 4 sticks by far has the largest Q-value. The Q-values of the other actions are on average lower than before, and two actions did not get explored at all. Figure 7c shows a state that is one turn away from victory. Here the agent also learned the correct action of taking 4 sticks from heap 3.

2 Deep Q-Learning

Deep Q-learning was implemented using the specifications and the suggested hyperparameters in the project description.

2.1 Learning from experts

Question 11. See figure 8.

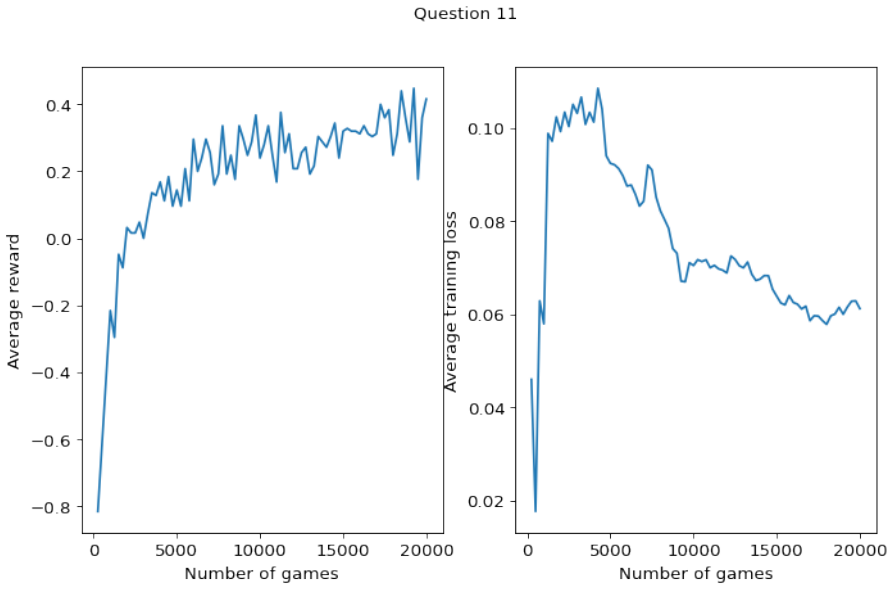


Figure 8: Average reward and average training loss computed every 250 games when DQN is trained against Opt(0.5) with $\epsilon = 0.2$. We can see that the algorithm learns to play Nim as the reward increases and the training loss decreases. The reached average reward fluctuates between 0.2 and 0.4 meaning that DQN wins more games than it loses. At the very beginning the training loss increases as both target network and policy network predict zero for the Q-values. After that, the training loss decreases throughout the training. At the end, the training loss seems to stay more stable that might mean that the agent is not learning as fast any more.

Question 12. See figure 9.

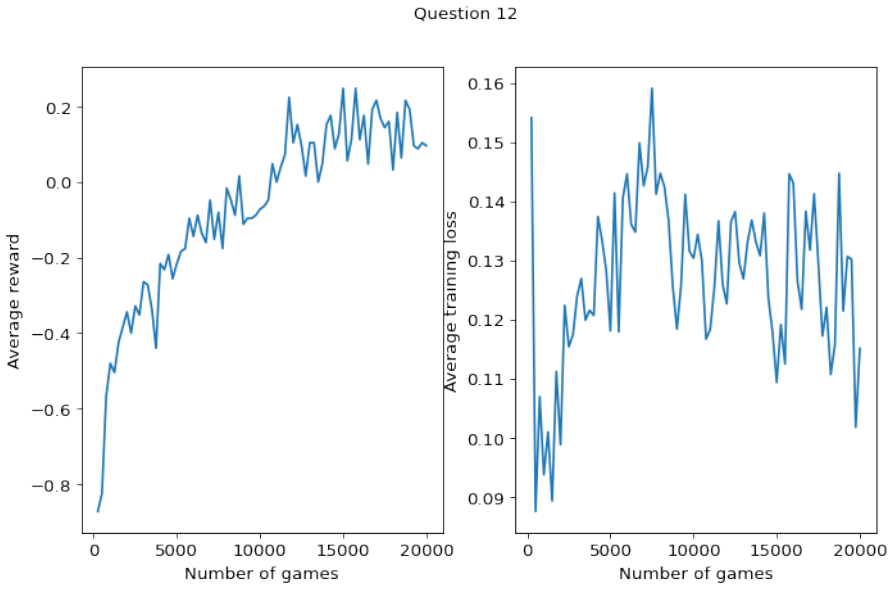


Figure 9: When DQN is trained without the replay buffer, we can see that average reward still increases but not as rapidly nor to as high values as before. The average training loss oscillates more than with replay buffer and generally stays higher.

Question 13. See figure 10.

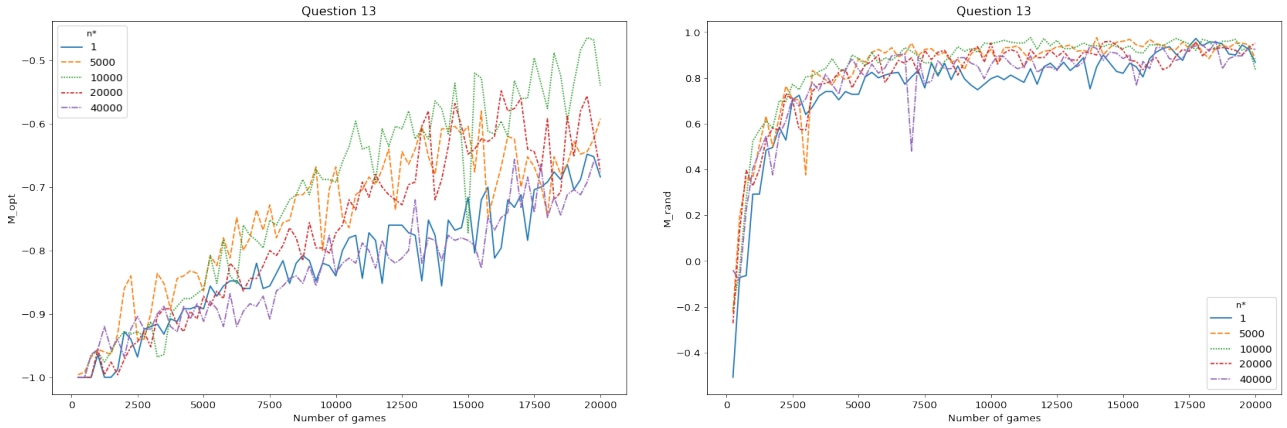


Figure 10: M_{opt} and M_{rand} plotted for every 250 games while training the DQN-agent against Opt(0.5) with decreasing exploration for different values of n^*. We can see that when played against the optimal player, it takes a long time for the agent to perform well and in fact the best value of $M_{\text{opt}} = -0.5$ is reached only at the end of training. Values $n^* = 1$ and $n^* = 40000$ result in the worst performance while the differences between three other values are smaller. It seems that with value $n^* = 10000$ the best values for M_{opt} are obtained. The agent very quickly achieves a positive win rate against the random player and M_{rand} increases rapidly close to 1.0. Different values for n^* do not affect M_{rand} as much as in the case of M_{opt} but $n^* = 1$ and $n^* = 40000$ again have the worst performance. When $n^* = 1$, we are using a fixed ϵ and therefore we can conclude that decreasing exploration helps in learning.

Question 14. See figure 11.

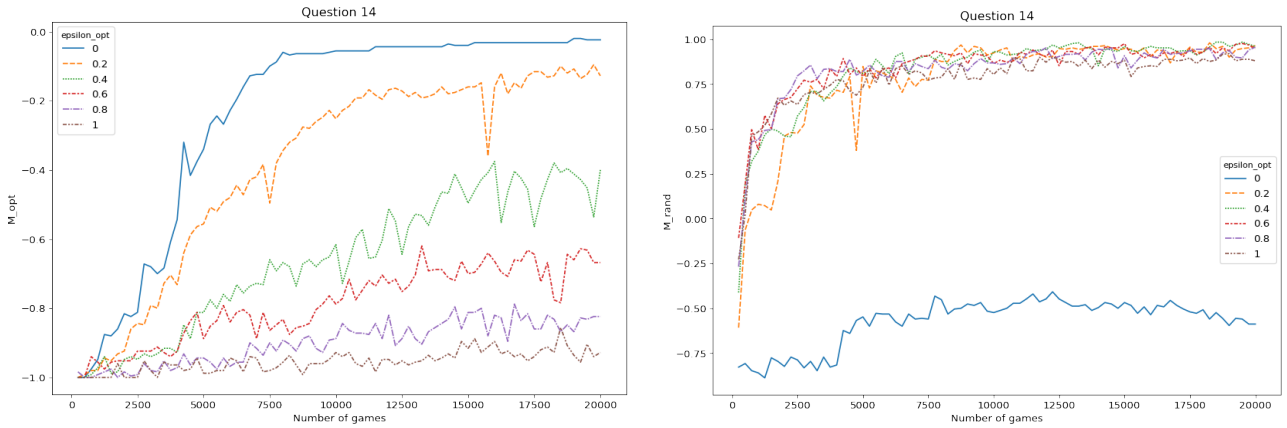


Figure 11: M_{opt} and M_{rand} plotted for every 250 games while training the DQN-agent against Opt(ϵ_{opt}) with decreasing exploration, $n^* = 10000$. We observe that the best results for M_{opt} can be obtained when $\epsilon_{opt} = 0$. The learning is slower with higher values and with $\epsilon_{opt} = 1$ the agent hardly never wins against the optimal player. The explanation is that the agent learns to win against the optimal player if it encounters optimal moves in the training phase. The results are very different for M_{rand}. We can see that with all other values except $\epsilon_{opt} = 0$, the agent learns to win against the random player quite fast. When trained against the optimal player, the agent does not learn how to answer to random moves because they never come up in the training games and the agent ends up losing most of the games.

Question 15. The highest M_{opt} value of -0.02 was achieved after 19000 games while training against Opt(0). The highest M_{rand} value of 0.984 was achieved after 18500 games while training against Opt(0.4). For both, we used decreasing exploration with value $n^* = 10000$.

2.2 Learning by self-practice

Question 16. See figure 12.

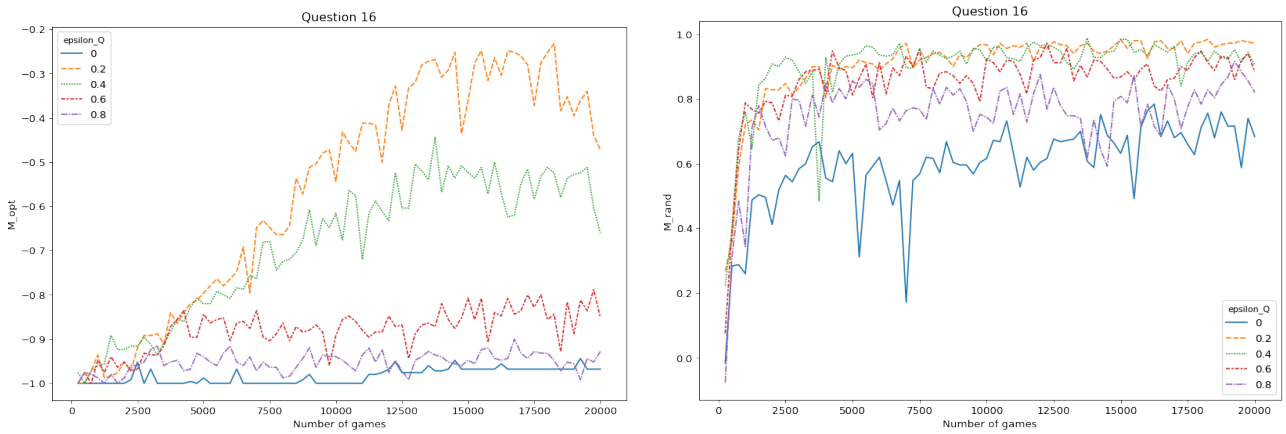


Figure 12: M_{opt} and M_{rand} plotted for every 250 games while training the DQN-agent against itself with different values of ϵ. We can see that the agent learns to play Nim but the effect of ϵ is quite strong. The values of M_{opt} reach -0.2 when $\epsilon = 0.2$ but for larger values the agent performers clearly worse than the optimal player. A similar effect can be seen with M_{rand} but the random player can be beaten sooner. Particularly with $\epsilon = 0$, the agent does not improve against the optimal player at all and even the values of M_{rand} do not increase the same way as for other values.

Question 17. See figure 13.

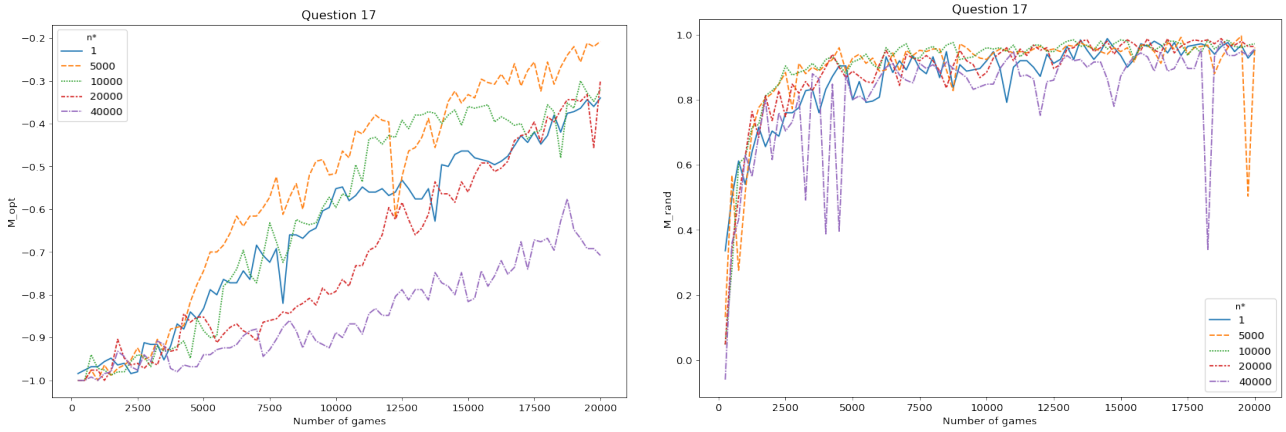


Figure 13: M_{opt} and M_{rand} plotted for every 250 games while training the DQN-agent against itself with decreasing exploration and different values of n^*. We can see that the results for different values are quite similar for M_{rand} but for M_{opt} there is clear variation. Decreasing ϵ with values $n^* = 5000$ and $n^* = 10000$ makes the agent learn better than with a fixed value but larger values for n^* result in worse performance.

Question 18. The highest M_{opt} value of -0.208 was achieved after 20000 games while training with $n^* = 5000$. The highest M_{rand} value of 0.996 was achieved after 19500 games while training with $n^* = 5000$.

Question 19.
See figure 14.

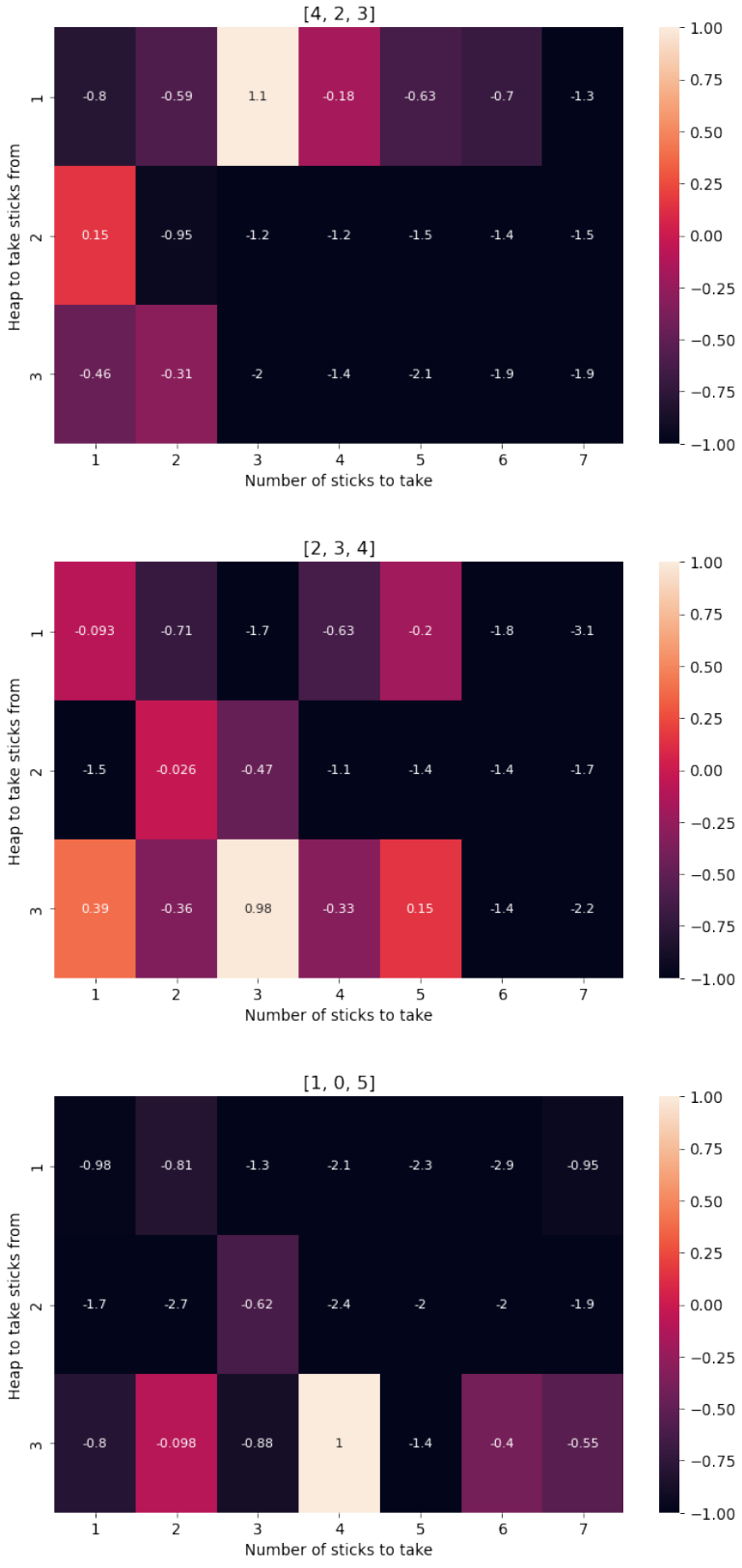


Figure 14: Q-values of available actions in three states visualized as heatmaps. The agent that learned these Q-values was trained against itself for 40000 games with decreasing exploration and $n^* = 10000$. We can see that the highest Q-value is always the same as for normal Q-learning agent presented in figure 7. This indicates that the Q-values make sense and that the agent learns to play Nim. Unavailable actions have very negative value because the agent learns fast that they lead to negative reward. After a while the agent should not make these actions anymore and their values tend to be smaller than what the theoretical limit for the expected reward is. In deep Q-learning we don't restrict the Q-values to be between -1 and 1 but they should converge to that while the learning proceeds. Winning and optimal moves have a Q-value very close to 1.0.

3 Comparing Q-Learning with Deep Q-Learning

Question 20. See Table 1.

Table 1: Highest M_{opt}, M_{rand} and corresponding T_{train} for Q-learning and DQN.

	Q-learning against experts	Q-learning against itself	DQN against experts	DQN against itself
M_{opt}	-0.016	-0.016	-0.02	-0.208
M_{rand}	0.996	1.0	0.984	0.996
T_{train}	8000	5500	6250	14250

Question 21.
After training against experts, Q-learning and DQN perform similarly. DQN manages to achieve better results when training against Opt(0.5). When decreasing exploration it is a good choice for both algorithms to set the number of exploratory games to about half of the training games. Against the optimal player both algorithms only achieve decent results when they are trained against an optimal or nearly optimal expert. Q-learning performs very badly against the optimal player when it was trained against an opponent making a lot of random moves and DQN only does a little better. However, when training the algorithms against the optimal player they both do terribly against the random player. Therefore, the overall best teacher has an ϵ value of about 0.2.
With self-practice we start to see some differences between the algorithms. Q-learning achieves its best M_{opt} value when training with $\epsilon = 0$, but DQN performs abysmally in comparison. For all other ϵ values, M_{opt} and M_{rand} are again similar. The DQN seems to benefit more from using decreasing exploration, as it performs best with about 5000 exploratory games. Q-learning on the other hand plays the best after having trained against itself with a fixed $\epsilon = 0.1$. Too much exploration hurts the performance of both algorithms against the optimal player, as the optimal strategies have not been reinforced enough. Against the random player the number of exploratory games does not make as much of a difference as both algorithms perform quite well. DQN seems to learn a lot slower than regular Q-learning when playing against itself, which could be the reason why Q-learning is able to win more games against the optimal player after both algorithms trained for 20000 games. The reason for this difference is in the fact that DQN needs to learn the rules as well as the best strategy.