\*\*Discussion:\*\*

In this part of the assignment, we performed some experiments with three algorithms (Sarsa, Q learning and expected Sarsa) on the Taxi-v2 OpenAI gym. For each experiment we picked a temperature value and a learning rate and then ran the three algorithms in 100 segments, each composed of 10 episodes for training followed by an episode for testing where we ran the greedy policy we learnt so far.

We have chosen three temperature values with a factor of 10 to transition from one temperature to the next one. We picked 0.1, 1 and 10 for these experiments.

We wanted to cover the whole range of possible values for the learning rate, but since we only had to pick 3 values we chose the midpoint of the range and two other learning rates with the same distance from the bounds of the range. The values we considered are: \[0.1, 0.5, 0.9\]

From the graphs, we notice that the experiments with the highest temperature value of 10 and a learning rate of 0.1 tends to be the worst at training with a large gap compared to the two other temperature values. However, the gap is comparable at testing. It is the highest for the \*Q-Learning\* algorithm. Once we have a sufficiently high learning rate, the experiments with the highest temperature value slightly outperform all the other temperature values at testing but are still slightly worst then the other 2 temperature values at training. This pattern can be observed in all the algorithms.

This impact of the temperature can be explained with the following analysis:

Suppose we have \*d\* actions. Let $q\in \mathbb R^d$ be the vector denoting the action-value function taken at a state $s$ and $S\_i(q) = \exp(q\_i) / \sum\_j\exp(q\_j)$ be the probability of picking the $ith$ action while exploring given by the boltzman distribution. Let $A\_t(q) = S(\frac{q}{t}) = \exp(\frac{q\_i}{t}) / \sum\_j\exp(\frac{q\_j}{t})$ where t is a positive real number. We notice that when t is pushed towards infinity we get $\lim\_{t \rightarrow \infty} S\_i(\frac{q}{t}) = 1 / d$. That is we have equal probabilities for all actions, thus the higher the constant t is, the more our algorithm is favoring exploration against the greedy behavior leading to a poor behaviour in training. However when t is heading toward zero we can prove that $\lim\_{t \rightarrow 0} S\_i(\frac{q}{t}) = \mathbb 1\_{q\_i = \max\_{j}(q\_j)} / \ell$ where $\ell$ is the number of actions with the same highest q-value. Therefore, if we suppose that there is only one best action $i$ (with respect to the correct q-value) than the probability of taking that action is $\lim\_{t \rightarrow 0} S\_i(\frac{q}{t}) = 1$ and it's a zero probability for the non-greedy actions. In this case, with low values of t we tend to be favoring the greedy behavior and thus exploiting what we have already learned instead of exploring new actions.

The advantage of the boltzman distribution is that it mimics the $\epsilon$-gready behaviour but with two other interesting advantages:

- Through the parameter t, it gives us the control of how much exploring we want our agent to do, the higher t is the more we get closer to a pure $\epsilon$-greedy algorithm.

- It gives weighted chances for actions to be considered: sub-optimal actions have higher chances of being selected instead of selecting amongst all the actions with an equal probability.

What about learning rate???

In terms of learning curves, when comparing all three algorithms we notice that expected sarsa seems to be the most stable of 3. There is much less noise in the average reward graph at testing. Sarsa seems the least stable at testing (on policy????).

When we look at the training average reward curve we notice that expected Sarsa converges the slowest and Q learning the fastest (off policy???).

Finally when looking at the variance graphs at training we notice: Q learning drops steadily and smoothly, while Sarsa and expected Sarsa drops but very noisily. We also notice that once the average reward value has converged the variance has converged also to a low value.

The test variance is higher than the training as expected since we are not optimizing but rather picking we think is to be the best. When the training has not converged yet, we expect to see big fluctuations in the value, thus the big variance.