

SARSA and K-means



Course: Machine Learning
Instructor: Dr. Mirela Popa

Student name: Lena Feiler Student ID: i6246119

Handing in

Upload a single report in form of a PDF. E.g., make a scan. Hand in code in form of a single zip file. Submissions by email or other types of archives are not accepted. Thank you for your understanding.

For the first part (a) include in the report a short description of your result, the best policy and your interpretation of the role of the two parameters alpha and gamma. For the second part (b) include the required explanations.

Filling in

You can use this Word file to answer your questions in a digital form. Alternatively, you can print the document, fill it in, and upload a scan. Make sure that we can read your hand-writing.

Graded: Code and Paper assignment: SARSA

Your task is to implement the SARSA algorithm for a simple single player game, in which an agent explores the environment, collects rewards and eventually arrives in the destination state, finishing the game (e.g., snake game, PacMan). Your goal is to maximize the final score (which is obtained by arriving in the shortest time to the destination state), while also exploring the environment. The grid is 4x4 and the set of valid actions are move up, down, right, left, except for the boundary walls, where only specific actions are possible. All the other values are currently initialized, but you can adjust them as you consider. A part of the code is provided for you in Canvas (tutorial6.ipynb); your task is to **complete the missing steps**, including the **update of the value function**.

The algorithm is the following:

For each s, a initialize the state $Q(s, a)$ to zero

Start from a random state s

Do forever:

- Select an action a randomly and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $Q(s, a)$ as follows

$$Q(s, a) = (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma Q(s', a'))$$

- Make the transition $s \leftarrow s'$
- If s' is the destination state, then stop

Include in this report your observations about the process, the obtained Q matrix and your interpretation about the role of the two parameters alpha and gamma and how do they affect the final policy.

For the Q matrix, we first initialize all its values to 0. Note that, based on the given code, this matrix only contains states and not action, state pairs. Then, using constant values for alpha and gamma (here we applied 0.1 for both by default), we update the values using the SARSA approach. Therefore, we randomly choose the next action the agent will take, which can also result in moving further away from the goal. This way, the values in the matrix are iteratively improved, however, the randomness factor can also result in suboptimal values for some of the values in the final matrix. The goal is initialized is given the value of 0, and, since the stopping condition is defined with giving the goal state a value of 0, the goal states will remain 0 also in the resulting Q matrix.

We can see the values in the matrix as the cost for the given state.

Our final matrix: (the values change, each time the program is run, due to the probabilistic factors the method contains).

```
Q: [[ 0.      -1.08938553 -1.11050311 -1.11108094]
     [-1.0675988 -1.10971526 -1.11101916 -1.11066288]
     [-1.11009984 -1.11101814 -1.10955141 -1.09199067]
     [-1.1110829 -1.11096884 -1.08566385  0.      ]]
```

- The following values were obtained, by changing one of the given parameters and keeping the other one constant.
- Gamma = 0.1

```
1 print(Q) #uses alpha = 0.5
[[ 0.          -1.0551919  -1.11085306 -1.11106195]
 [-1.04146877  -1.11062058 -1.11109868 -1.11106359]
 [-1.10887536  -1.11102279 -1.10967331 -1.1106216 ]
 [-1.11104044  -1.11049268 -1.05127515  0.          ]]
```

```
1 print(Q) #uses alpha = 0.9
[[ 0.          -1.01110839 -1.11103786 -1.11111109]
 [-1.00010995  -1.11109035 -1.11110127 -1.11017628]
 [-1.11103818  -1.11100982 -1.11094622 -1.01000862]
 [-1.11111111  -1.1102109  -1.01093  0.          ]]
```

- Alpha = 0.1

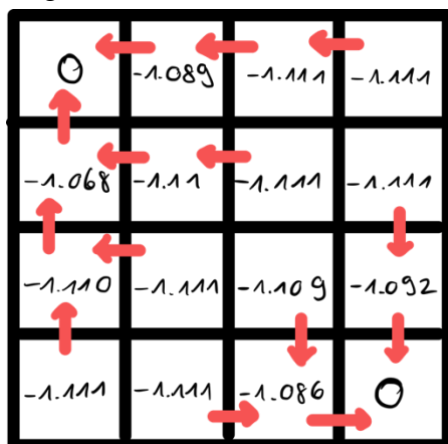
```
1 print(Q) #uses gamma = 0.5
[[ 0.          -1.69587333 -1.90106311 -1.96607861]
 [-1.67283915  -1.89784577 -1.95390574 -1.92441898]
 [-1.94394095  -1.95084544 -1.88597476 -1.49935117]
 [-1.96495077  -1.91805395 -1.63919525  0.          ]]
```

```
1 print(Q) #uses gamma = 0.9
[[ 0.          -3.25575945 -5.22321742 -5.79440382]
 [-3.44088644  -4.76099166 -5.67110895 -5.65563276]
 [-5.49846136  -5.63801162 -5.37111816 -4.69357094]
 [-5.95536521  -5.624643  -3.91916205  0.          ]]
```

Higher values of γ cause an increase in the cost. This shows us, that increasing gamma, increases the rate at which the states are explored, and how much importance they are given. On the other side, the changes in the values are smaller for a change in alpha.

Best policy for maximizing the score (include it as a matrix/drawing)

To determine what the best policy will look like for the given problem, we have determined it using the values of the final Q matrix given above.



To do this, the paths to the 2 goals were selected, such that the given cost is minimized (this is the same as maximizing the result).

Due to the probabilistic factors of the applied algorithm, this policy might change slightly for different outputs, However, these changes will not be very big.

Explanation of the role of the parameters:

α : Alpha is the learning rate. It determines, to what extent new information is allowed to change the previously obtained info. If α is 0, the agent will not continue to learn, and if α is 1, the agent will only consider the most recently acquired info, and the information will experience big changes.

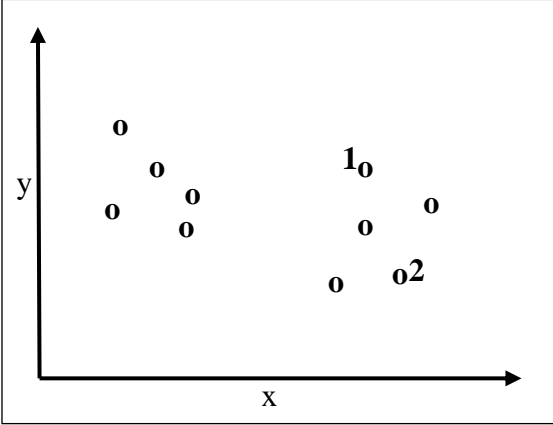
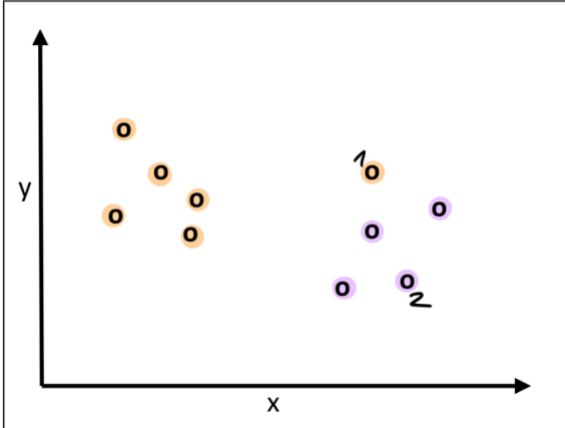
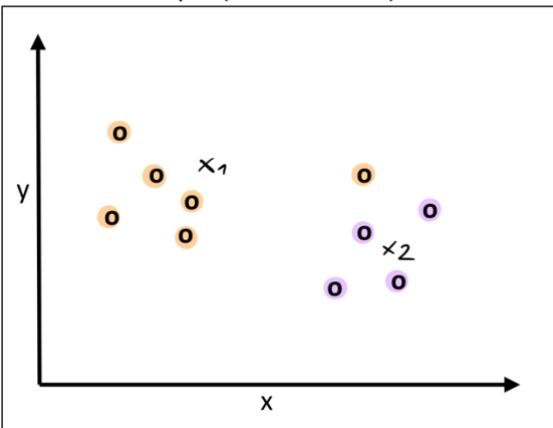
γ : Gamma is the discount rate. It influences how much a reward is considered and gives them importance. The greater γ , the more we prefer future rewards, so, if gamma is close to 1, we prefer long term higher rewards. On the other hand, if γ is small, we favor current rewards.

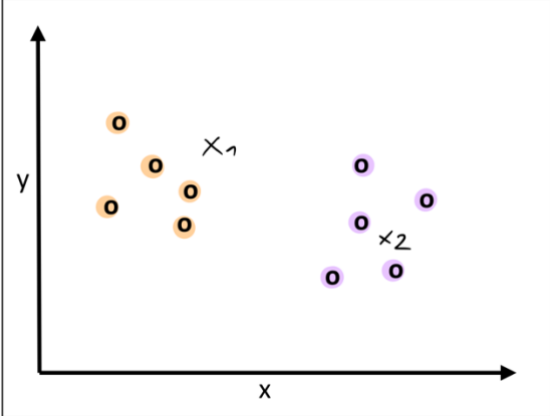
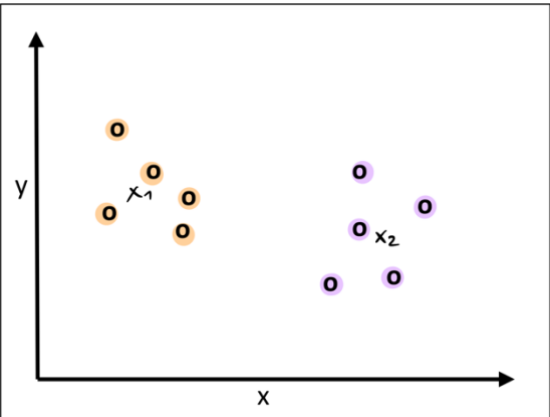
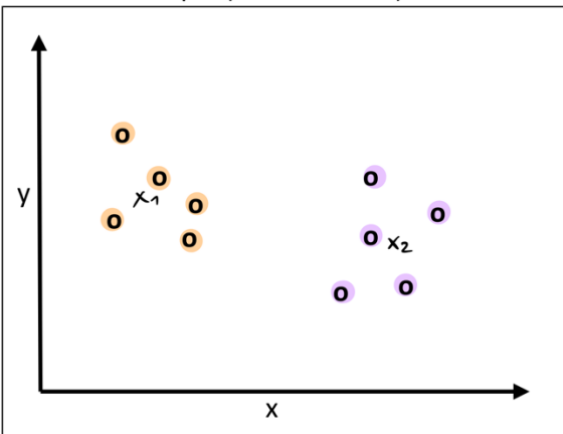
Graded: Paper assignment: K-Means

Given the following data set, show (with drawings) and explain (with your own words) the different steps of a k-means algorithm when $k=2$. Show and explain individual steps of the algorithm – not just full iterations.

(Explanation of symbols: o = data points; 1 = marker for first centroid, 2 = marker second centroid)



<p>Step 1 (not iteration!):</p> 	<p>Explanation:</p> <p><i>Initialization: Centroids get assigned to random locations. Here two random points from the data set are picked as initial seeds.</i></p>
<p>Step 2 (not iteration!):</p> 	<p>Explanation:</p> <p><u>Computation of distances:</u> <u>For every data point, compute its distance from the two labeled points (1, 2). For each data point, assign it to the same class, as the seed (1 or 2), which is closest to it.</u></p>
<p>Step 3 (not iteration!):</p> 	<p>Explanation:</p> <p><u>Now, each data point is assigned a class, and will see all the data points assigned to the same class as a cluster. Now, take the average of each cluster. These, respectively, will be our new centroids (cluster centers) for the next iteration.</u></p>

<p>Step 4 (not iteration!):</p> 	<p>Explanation:</p> <p>Iteration 2: <u>Again, compute the distances from all points to the newly assigned cluster centers (1, 2), and assign the datapoint to the class, for which the center point is closest.</u> <u>This will lead to some points being reassigned to the opposite class.</u></p>
<p>Step 5 (not iteration!):</p> 	<p>Explanation:</p> <p><u>Take the averages of each cluster. These, respectively, will be our new centroids (cluster centers) for the next iteration.</u></p>
<p>Step 6 (not iteration!):</p> 	<p>Explanation:</p> <p>Iteration 3: <u>As before, compute the distances from all points to the newly assigned cluster centers (1, 2), and assign the datapoint to the class, for which the center point is closest. We now see, that no data points get reclassified, and also when recalculating the centroids, their positions do not change. Hence, we know that the algorithm converged.</u></p>