ML – Assignment 2 – Gustav von Zitzewitz - 03636797

Exercise 1 – GMM Parameters

Priors:

gmm_pi

1x4 double

	1	2	3	4
1	0.2968	0.2400	0.2015	0.2617

Means:

gmm_mean

4x2 double

	1	2
1	-0.0194	-0.0166
2	-0.0432	0.0446
3	-0.0147	-0.0796
4	0.0262	0.0617

Covariance Matrices:

gmm_cov{1, 1} | gmm_cov{1, 2}

gmm_cov{1, 1}

	1	2
1	7.4319e-04	-5.9073e-04
2	-5.9073e-04	6.0885e-04

gmm_cov{1, 2} | gmm_cov{1,

gmm_cov{1, 2}

	1	2
1	1.7483e-04	2.6160e-04
2	2.6160e-04	3.9764e-04

gmm_cov{1, 3} | gmm_cov{

gmm_cov{1, 3}

	1	2
1	3.9524e-04	2.1734e-04
2	2.1734e-04	1.2814e-04

gmm_cov{1, 4}

gmm_cov{1, 4}

	1	2
1	0.0011	-4.2434e-04
2	-4.2434e-04	2.4312e-04

Exercise 2 - HMM Classification Results

All gestures in Test.txt belong to 'gesture2' as all Log-Likelihoods are smaller than -120:

Log Likelihoods for each Repetition (1-10):

loglikelihood

10x1 double

	1
1	-511.4069
2	-570.6697
3	-387.9167
4	-427.3069
5	-437.5989
6	-426.1784
7	-473.3031
8	-400.2880
9	-377.1776
10	-401.0614

Exercise 3 – Reinforcement Learning

WalkPolicyIteration

1. Report your Reward Matrix:

rew

16x4 double

	1	2	3	4
1	0	0	0	0
2	0	1	-1	0
3	0	-1	-1	-1
4	0	0	0	0
5	-1	-1	0	1
6	0	0	0	0
7	0	0	0	0
8	-1	1	0	0
9	-1	-1	0	-1
10	0	0	0	0
11	0	0	0	0
12	-1	0	0	0
13	0	0	0	0
14	0	0	-1	1
15	0	-1	-1	1
16	0	0	0	0

2. What value of γ have you used and what is the result of increasing or decreasing γ?

As Discountfactor Gamma I used γ = 0.94.
Gamma regulates the importance of in the future collected rewards:
Increasing Gamma results in higher importance of future rewards (less discounting).

3. Approximately how many iterations are required for the policy iteration to converge?

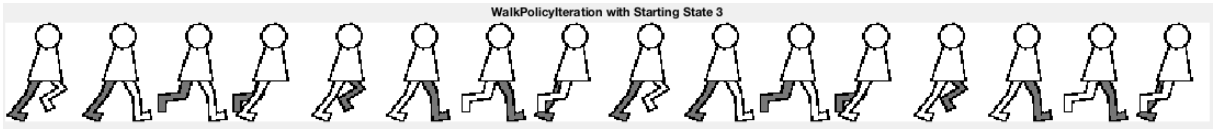
Between 3 and 7 Iterations are required to converge depending on the starting state.
The following figure shows depending on which state the algorithm starts how many iterations are necessary:

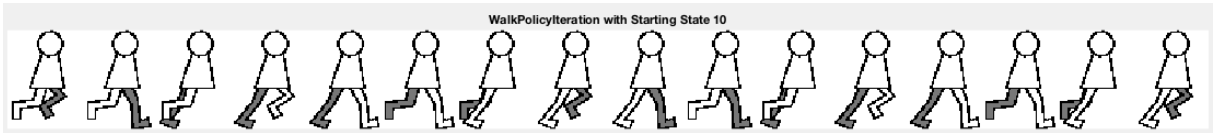
iteration

16x1 double

	1
1	6
2	7
3	5
4	6
5	6
6	6
7	6
8	7
9	6
10	6
11	5
12	6
13	7
14	5
15	3
16	6

4. Attach the result of WalkPolicyIteration(s) when starting from state 10 and 3.





WalkQLearning

1. Report the values of ε and α that you have used.

Epsilon (Percentage of random actions) ε = 0.3,
Alpha (Learning Rate) α = 0.8

2. What happens if a pure greedy policy is used? Implement and compare with the ε-greedy policy. Does it matter what value of ε you use?

Pure greedy takes always the actions with the highest reward.
Therefore it can get stuck in local maxima or minima.
Epsilon is the probability that QLearning will take a random action.
This is used to overcome local to find global extrema.
The greater Epsilon is, the more random the taken actions are.
Pure greedy is the same as QLearning with Epsilon = 0.

3. Approximately how many steps are necessary for the Q-learning algorithm to find an optimal policy?

It takes between 30000 and 100000 steps .
I used 60000 in my final code version.

4. Attach the result of WalkQLearning(s) when starting from states 5 and 12.