Machine Learning Assignment 2

Børge Wiik
Matricel Number: 03685405

August 2017

1 Exercise 1

The GMM-parameters after an arbitrary simulation are the following:

The priors of each class are given by the vector in (1):

$$priors = \begin{bmatrix} 0.6707 & 3.2912e-05 & 0.3260 & 0.0033 \end{bmatrix} \tag{1}$$

The means of each class are given by the matrix in (2). Each row contains an x-coordinate and a y-coordinate.

$$mu = \begin{bmatrix} -0.0075 & -0.0139 \\ 0.0638 & 0.0559 \\ -0.0221 & 0.0459 \\ 7.0962e-04 & 0.0759 \end{bmatrix} \tag{2}$$

Four different covariance matrices represent the four different classes, shown in (3), (4), (5) and (6) respectively.:

$$cov1 = \begin{bmatrix} 0.0011 & 0.0003 \\ -0.0003 & 0.0032 \end{bmatrix} \tag{3}$$

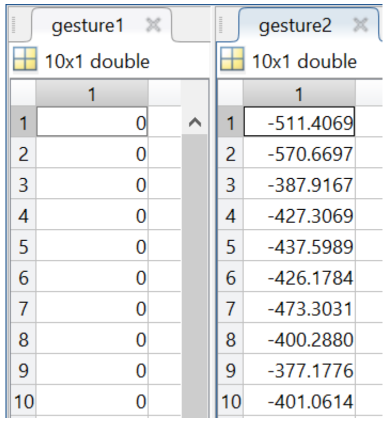
$$cov2 = 1.0e-03 * \begin{bmatrix} 0.2248 & -0.1048 \\ -0.1048 & 0.0585 \end{bmatrix} \tag{4}$$

$$cov3 = \begin{bmatrix} 0.0014 & 0.0004 \\ -0.0004 & 0.0006 \end{bmatrix} \tag{5}$$

$$cov4 = 1.0e-03 * \begin{bmatrix} 0.3950 & 0.0566 \\ -0.0566 & 0.0716 \end{bmatrix} \tag{6}$$

2 Exercise 2

A screenshot of the classification results is seen in Figure (1). All sequences where classified into gesture 2.



gesture1 — 10x1 double | gesture2 — 10x1 double

	gesture1 (1)	gesture2 (1)
1	0	-511.4069
2	0	-570.6697
3	0	-387.9167
4	0	-427.3069
5	0	-437.5989
6	0	-426.1784
7	0	-473.3031
8	0	-400.2880
9	0	-377.1776
10	0	-401.0614

Figure 1: Sequences classified by their loglikelihoods.

3 Exercise 3

3.1 Policy Iteration

My reward matrix is shown in (7). I have weighed unwanted state-actions with a -1 and state-action pairs that make the robot move forward with +1. The rest are set to 0.

$$rew = \begin{bmatrix} 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & -1 & 0 & -1 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 0 & -1 \end{bmatrix} \tag{7}$$

I have used $\gamma = 0.99$.γ makes sure the current reward is greater than future rewards. Increasing γ gives future rewards a higher effect on the expected total payoff. Similarly, decreasing γ gives future rewards less effect on the expected total payoff.

Depending on the initial policy and input state the algorithm requires between 3 and 8 iterations before it converges.

The results of WalkPolicyIteration(s) starting from state 3 and state 10 are shown in figures 2 and 3 respectively.

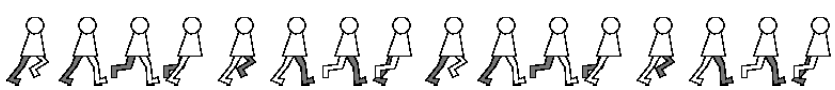


Figure 2: walkPolicyIteration starting at state 3

Figure 3: walkPolicyIteration starting at state 10

3.2 Q-Learning

I have chosen epsilon as $\epsilon = 20$ percent, and alpha as $\alpha = 0.5$.

if a pure greedy policy is used (chance of random action = 0 percent) then the system never reaches a good converged optimal policy. Comparing the Q-Learning algorithm with $\epsilon = 0$ percent with $\epsilon = 20$ percent the first doesn't converge. Thus, the chance of random action (ϵ) does highly matter.

It takes approximately 11000 steps until the Q-Learning algorithm finds an optimal policy.

The results of WalkQLearning(s) starting from state 5 and state 12 are shown in Figure 4 and Figure 5 respectively.

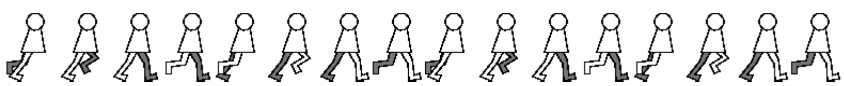


Figure 4: walkQLearning starting at state 5

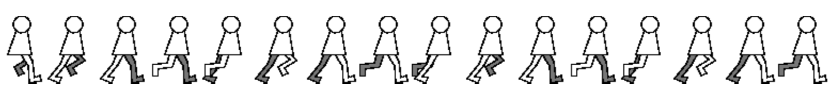


Figure 5: walkQLearning starting at state 12