Technische Hochschule Rosenheim

Faculty of Computer Science
Prof. Dr. M. Tilly

Winter Term 2022/2023

Mock Exam - Applied Artificial Intelligence(AAI)
400 - Unsupervised and Reinforcement Learning(URL)

Date: 02.01.2023	Duration: 90 Minutes	Material: handwritten notes A4; calculator

---

Name: ....

Matrix number: ....

Good luck!

Notes:

1. The staples must not be loosened. The exam includes **15 pages incl. cover sheet and worksheets.**.
2. Work on the questions directly in the task. If necessary, use the worksheets at the end.
3. If, in your opinion, there are contradictions in the tasks or information is missing, make reasonable assumptions and document them..
4. The distribution of points is for orientation, but it is not binding.
5. Please do not write in pencil, red or green pens and if possible **legible**.

SOLUTION is available in the new year!

Name: Matrix number:

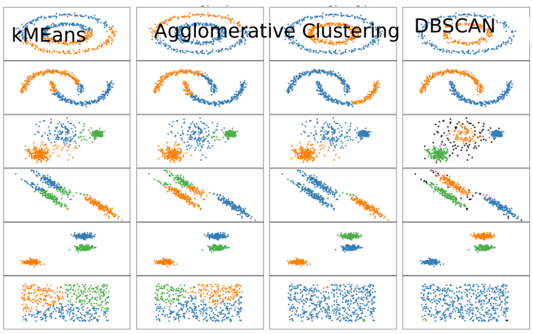
1. Task - General questions 4+6 Points

a)

Mark the correct answer or statement; mark exactly one answer per question.

1. Clustering
   - ☐ The silhouette score calculates the surface of a silhouette for eigenfaces,
   - ✓ The elbow method can be used to determine the number of clusters k in a given data set.
   - ☐ The elbow method is sometimes also called the KneeLocator.
2. k-means vs. DBSCAN
   - ☐ For k-means the number of clusters is calculated by the algorithm.
   - ☐ k-means is faster the DBSCAN.
   - ✓ DBSCAN does not need a-priori knowledge on number of clusters.
3. Clustering
   Given the following figure. Each column shows the result of a cluster algorithm for a different data set. Name 2 out of 4 clustering algorithms. Write on top of each column:

   Solution:



4. Pre-Processing and Normalization
   Name 2 normalization methods:

   Solution:

   a) MinMaxScalar
   b) RobustScalar
   c) StandardScaler

b)

Describe the DBSCAN algorithm.

Solution:
We start with the data points and values of eps and minPts as input:

1. Out of n unvisited sample data points, we'll first move through each point in a loop and mark each one as visited.
2. From each point, we'll look at the distance to every other point in the dataset.
3. All points that fall within the neighborhood radius hyperparameter (eps) should be considered neighbors.
4. The number of neighbors should be at least as many as the minimum points required (MinPts).
5. If the minimum point threshold is reached, the points should be grouped as a cluster, or else marked as noise.
6. This process should be repeated until all data points are categorized in clusters or as noise.

2. Task - k-means Clustering 15 Points

Use the k-means algorithm and Euclidean distance to cluster the following 8 examples into 3 clusters:

A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9).

The distance matrix based on the Euclidean distance is given below

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

Suppose that the initial centers of each cluster are A1, A4 and A7. **Run the k-means algorithm for 1 epoch only**. At the end of this epoch show:

1. The new clusters (i.e. the examples belonging to each cluster)
2. The centers of the new clusters
3. Draw a 10 by 10 space with all the 8 points and show the clusters after the first epoch and the new centroids

Solution:

a)
d(a,b) denotes the Eucledian distance between a and b.
It is obtained directly from the distance matrix or calculated as follows: d(a,b)=sqrt((xb -xa) 2 +(y b -y a)2))
seed1=A1=(2,10), seed2=A4=(5,8), seed3=A7=(1,2)
epoch1 – start:
A1:
d(A1, seed1)=0 as A1 is seed1; d(A1, seed2)= sqrt(13) >0; d(A1, seed3)= sqrt(65) >0
=> A1 cluster1
A2:
d(A2,seed1)= sqrt(25) = 5; d(A2, seed2)= sqrt(18) = 4.24; d(A2, seed3)= sqrt(10) = 3.16 <= smaller
=> A2 cluster3
A3:
d(A3, seed1)= sqrt(36) = 6; d(A3, seed2)= sqrt(25) = 5 <= smaller; d(A3, seed3)= sqrt(53) = 7.28
=> A3 cluster2
A4:
d(A4, seed1)= sqrt(13) >0; d(A4, seed2)=0 as A4 is seed2; d(A4, seed3)= sqrt(52) >0
=> A4 cluster2
A5:
d(A5, seed1)= sqrt(50) = 7.07; d(A5, seed2)= sqrt(13) = 3.60 <= smaller; d(A5, seed3)= sqrt(45) = 6.70
=> A5 cluster2
A6:
d(A6, seed1)= sqrt(52) = 7.21; d(A6, seed2)= sqrt(17) = 4.12 <= smaller; d(A6, seed3)= sqrt(29) = 5.38
=> A6 cluster2
A7:
d(A7, seed1)= sqrt(65) >0; d(A7, seed2)= sqrt(52) >0; d(A7, seed3)=0 as A7 is seed3
=> A7 cluster3
A8:
d(A8, seed1)= sqrt(5); d(A8, seed2)= sqrt(2) <= smaller; d(A8, seed3)= sqrt(58)
=> A8 cluster2
end of epoch1

new clusters: 1: [A1], 2: [A3, A4, A5, A6, A8], 3: [A2, A7]

b) centers of the new clusters:

C1= (2, 10), C2= ((8+5+7+6+4)/5, (4+8+5+4+9)/5) = (6, 6), C3= ((2+1)/2, (5+2)/2) = (1.5, 3.5)

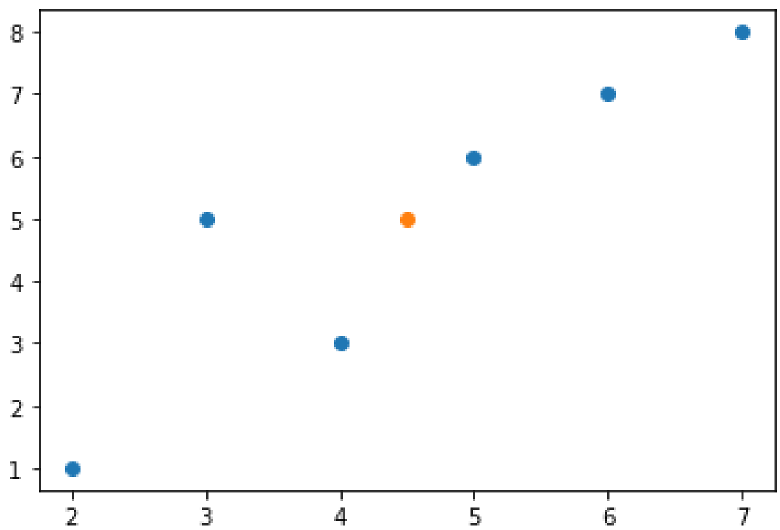
3. Task - PCA **2+6+2+5 Points**

Consider the following data points (2, 1), (3, 5), (4, 3), (5, 6), (6, 7), (7, 8).
Compute the principal component using PCA Algorithm by the following steps:

1. Calculate the mean vector.
2. Calculate the covariance matrix.
3. Proof that $\lambda 1 = 8.22$ and $\lambda 2 = 0.38$ are eigenvalues of the covariance matrix.
4. Calculate the PCA (or eigenvector).

Solution:

Mean: [4.5, 5.0]
Covariance v = (np.sum([(x)*(y) for x,y in zip(cX2,cX2)]))/(len(cX1)-1)
[[3.5 4.4] [4.4 6.8]]
np.linalg.det(cov$_{m}atrix - np.array([[8.22, 0], [0, 8.22]])) = -12.657600000000004$
np.linalg.det(cov$_{m}atrix - np.array([[0.38, 0], [0, 0.38]])) = 0.6704000000000013$
Eigenvalues
[0.45079794 9.84920206]
Eigenvectors as columns
[[-0.82192562 -0.56959484] [0.56959484 -0.82192562]]



4. Task - DBSCAN 10+5 Points

If Epsilon is 2 and minpoint is 2 (minpts), what are the clusters that DBScan would discover with the following 8 examples: **A1=(2,10), A2=(2,5), A3=(8,4), A4=(5,8), A5=(7,5), A6=(6,4), A7=(1,2), A8=(4,9)**.

The distance matrix is the following:

	A1	A2	A3	A4	A5	A6	A7	A8
A1	0	$\sqrt{25}$	$\sqrt{36}$	$\sqrt{13}$	$\sqrt{50}$	$\sqrt{52}$	$\sqrt{65}$	$\sqrt{5}$
A2		0	$\sqrt{37}$	$\sqrt{18}$	$\sqrt{25}$	$\sqrt{17}$	$\sqrt{10}$	$\sqrt{20}$
A3			0	$\sqrt{25}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{53}$	$\sqrt{41}$
A4				0	$\sqrt{13}$	$\sqrt{17}$	$\sqrt{52}$	$\sqrt{2}$
A5					0	$\sqrt{2}$	$\sqrt{45}$	$\sqrt{25}$
A6						0	$\sqrt{29}$	$\sqrt{29}$
A7							0	$\sqrt{58}$
A8								0

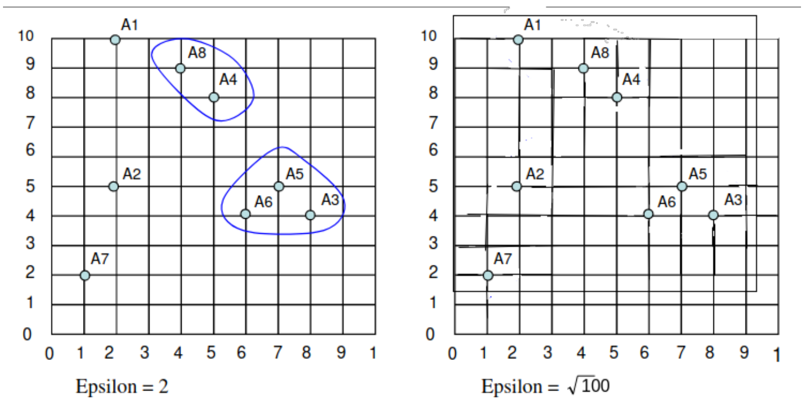
1. Draw the 10 by 10 space and illustrate the discovered clusters.
2. What if Epsilon is increased to 10?

Solution:

What is the Epsilon neighborhood of each point?

N(A1)=();
N(A2)=();
N(A3)=(A5, A6);
N(A4)=(A8);
N(A5)=(A3, A6);
N(A6)=(A3, A5);
N(A7)=();
N(A8)=(A4)

So A1, A2, and A7 are outliers, while we have two clusters C1=(A4, A8) and C2=(A3, A5, A6)
If Epsilon is sqrt(100) then all points are within the neighborhood.



5. Task - Autoencoder 4+6 Points

a)

Given the following code.

```
class Denoise(Model):
   def __init__(self):
      super(Denoise, self).__init__()
      self.encoder = tf.keras.Sequential([
      layers.InputLayer(input_shape=(28, 28, 1)),
      layers.Conv2D(16, (3, 3), activation='relu', padding='same', strides
         =2),
      layers.Conv2D(8, (3, 3), activation='relu', padding='same', strides
         =2)])

      self.decoder = tf.keras.Sequential([
      layers.Conv2DTranspose(8, kernel_size=3, strides=2, activation='relu'
         , padding='same'),
      layers.Conv2DTranspose(16, kernel_size=3, strides=2, activation='relu
         ', padding='same'),
      layers.Conv2D(1, kernel_size=(3, 3), activation='sigmoid', padding='
         same')])

   def call(self, x):
      encoded = self.encoder(x)
      decoded = self.decoder(encoded)
      return decoded
}
```

Provide the shape of each layer:

Solution:
Layer 1: 28x28x1 <= Input
Layer 2: 14x14x16
Layer 3: 7x7x8 <= Compression
Layer 4: 14x14x8
Layer 5: 28x28x16
Layer 6: 28x28x1 <= Output

b)

Explain how to use **Autoencoder** for anomaly detection!

Solution:

How will you detect anomalies using an autoencoder?
Recall that an autoencoder is trained to minimize reconstruction error. You will train an autoencoder on the normal rhythms only, then use it to reconstruct all the data. Our hypothesis is that the abnormal rhythms will have higher reconstruction error. You will then classify a rhythm as an anomaly if the reconstruction error surpasses a fixed threshold.
Detect anomalies by calculating whether the reconstruction loss is greater than a fixed threshold.

reconstructions = autoencoder.predict(normalData)
trainLoss = tf.keras.losses.mae(reconstructions, normalData)

reconstructions = autoencoder.predict(anomalousData)
testLoss = tf.keras.losses.mae(reconstructions, anomalousData)

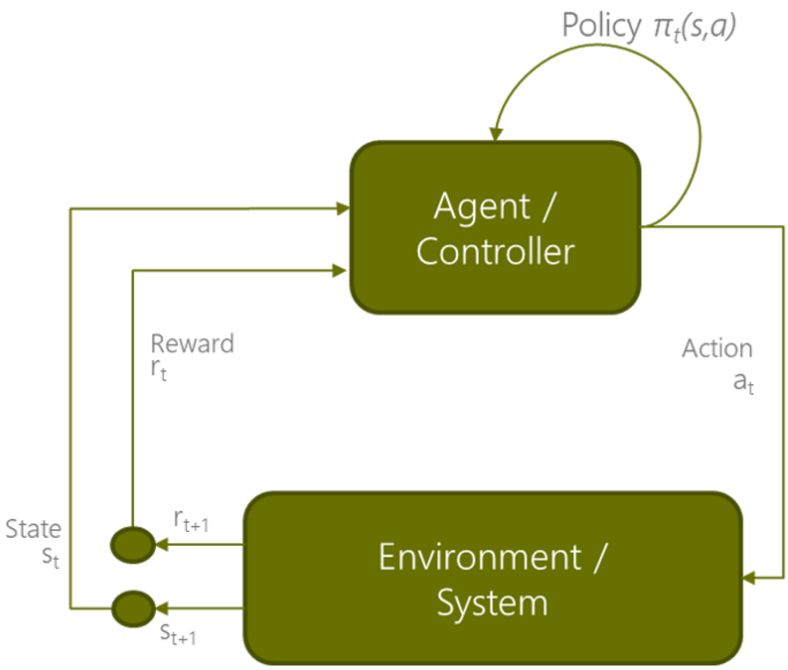
6. Task - Reinforcement Learning 3+5+10 Points

a)

Explain the concept to *Reinforcement Learning* using a sketch.

Solution:

In an RL system, an agent performs actions that change an environment. The agent observes the change of the environment and determines the state in which it believes it is within the environment. In addition, the agent receives rewards. The agent tries to maximize these rewards over time. The agent selects the actions from a set of possible actions. When choosing, it follows a policy that it learns over time based on the rewards it receives.



b)

List and describe the characteristics of a Markov Decision Process (MDP).

Solution:

MDP is represented by five important elements $(S, A, P_{ss'}, R_{ss'}, \gamma)$:

- A set of states S the agent can be in
- A set of actions A that can be performed by an agent, for moving from one state to another
- A transition probability $P_{ss'}$, which is the probability of moving from one state S to another S' by performing an action a
- A reward probability $R_{ss'}$, which is the probability of a reward acquired by the agent for moving from one state S to another S' by performing some action a
- A discount factor γ, which controls the importance of immediate and future rewards

c)

Q-learning is a variant of making an agent experience an environment without knowing the model behind it (learning without knowledge of the MDP). With Q-learning the explicit learning of the *policy* is omitted, instead the *policy* is learned directly.
The following formula applies:

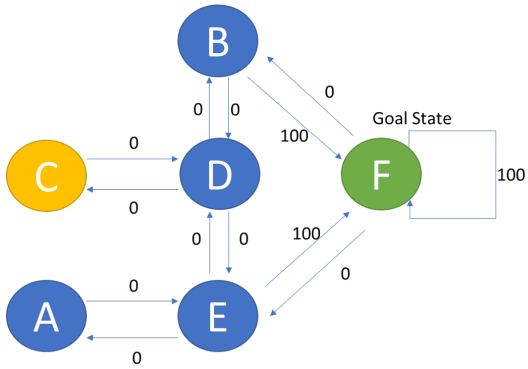
$$Q_{k+1}(s_t, a_t) < -Q_k(s_t, a_t) + \alpha[R(s, a) + \gamma \max(Q_k(s_{t+1}, a)) - Q_k(s_t, a_t))]$$

where α is the *learning rate* and γ is the *discount factor*.
In this example, let the reward matrix R be given as:

state/action	A	B	C	D	E	F
A	-	-	-	-	0	-
B	-	-	-	0	-	100
C	-	-	-	0	-	-
D	-	0	0	-	0	-
E	0	-	-	0	-	100
F	-	0	-	-	0	100

As can be easily seen, there are 6 states $S = A, B, C, D, E, F$ and the actions A that allow an agent to move from a state S_1 to a state S_2 (e.g. from A to E or from D to B,C or E). The example could be a building with rooms, and doors that allow an agent to transition from one room to another.

Name: Matrix number:



Apply the Q-learning algorithm step by step. Calculate the following values (k denotes the respective episodes) with $\alpha = 1$ and $\gamma = 0.8$. An episode ends when the goal (=goal state) is reached:

Solution:

1. Episode:

$Q_{k=1}(B,F) = Q(B,F) + 1 * [R(B,F) + 0.8 * (max[Q(F,B), Q(F,E), Q(F,F)]) - Q(B,F)] = 0 + 100 + 0.8 * (max[0,0,0]) - 0 = 100$

2. Episode:

$Q_{k=2}(D,B) = Q(D,B) + R(D,B) + 0.8 * (max[Q(B,D), Q(B,F)]) - Q(D,B) = 0 + 0 + 0.8 * (max[0,100] - 0) = 80$

next action: $B- > F$

$Q_{k=2}(B,F) = Q(B,F) + R(B,F) + 0.8 * (max[Q(F,B), Q(F,E), Q(F,F)]) - Q(B,F) = 100 + 100 + 0.8 * (max[0,0,0]) - 100 = 100 + 0 = 100$

3. Episode:

$Q_{k=3}(C,D) = Q(C,D) + R(C,D) + 0.8 * (max[Q(D,B), Q(D,E), Q(D,C)]) - Q(C,D) = 0 + 0 + 0.8 * (max[80,0,0]) - 0 = 64$

next action: (max): $D- > B$
$Q_{k=3}(D,B) = Q(D,B) + R(D,B) + 0.8 * (max[Q(B,D), Q(B,F)]) - Q(D,B) = 80 + 0 + 0.8 * (max[0,100]) - 80) = 0.8 * [100] = 80$

next action: (max): $B- > F$
$Q_{k=3}(B,F) = Q(B,F) + R(B,F) + 0.8 * (max[Q(F,B), Q(F,E), Q(F,F)]) - Q(B,F) = 100 + 100 + 0.8 * (max[0,0,0]) - 100) = 100 + 100 - 0.8 * (0) - 100] = 100$

4. Episode:

$Q_{k=4}(E,D) = Q(E,D) + R(E,D) + 0.8 * (max[Q(D,B), Q(D,C), Q(D,E)]) - Q(E,D) = 0 + 0 + 0.8 * (max[80,0,0]) - 0 = 64$

next action: (max): $D- > B$
$Q_{k=4}(D,B) = Q(D,B) + R(D,B) + 0.8 * (max[Q(B,D), Q(B,F)]) - Q(D,B) = 80 + 0 + 0.8 * (max[0,100]) - 80 = 80 + 0.8 * [100] - 80 = 80$

next action: (max): $B- > F$
$Q_{k=4}(B,F) = Q(B,F) + R(B,F) + 0.8 * (max[Q(F,B), Q(F,E), Q(F,F)]) - Q(B,F) = 100 + 100 + 0.8 * (max[0,0,0]) - 100 = 100$

What does the $Q - matrix$ look like after the 4 episodes?

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$