MDM-2025 Homework 1

Mert Dagli (100830277)
Tuomas Mäkinen (103382573)
Maja Sellmer (103396682)

September 29, 2025

1 Methods

The task was to experimentally study how clustering tendency can be observed both visually and with entropy-based measures. To this end, we plotted the data using histograms and scatter plots, then calculated the probability-based entropy in different ways, and finally simulated two greedy search strategies. We completed the assignment in Python, using the following libraries:

- `matlplotlib` for plotting
- `pandas` to handle the given data as a dataframe
- `numpy` for histograms
- `math` for calculations

2 Histograms

By analysing the distribution of the data points across all three components (see Figure 1), we can see that in dimension 3 there are clearly two peaks, while for dimensions 1 and 2 the distribution appears more uniform.

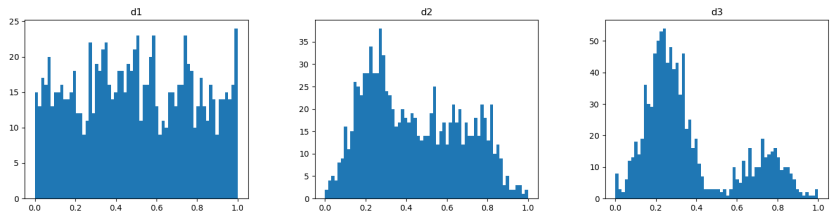

Figure 1: Histograms with 64 bins for the three data components d1, d2, d3

3 Scatter plots

By inspecting the correlations of the three components using scatter plots (see Figure 2), we can see that while there is no visible correlation between d1 and d2, three clusters are clearly visible in the d2-d3 plot, which means the data points could be separated into three clusters based on these two dimensions. In the last plot correlating d3 with d1, we see a separation into two components, which follows from the multimodal distribution of d3 we remarked on in the preceding section.

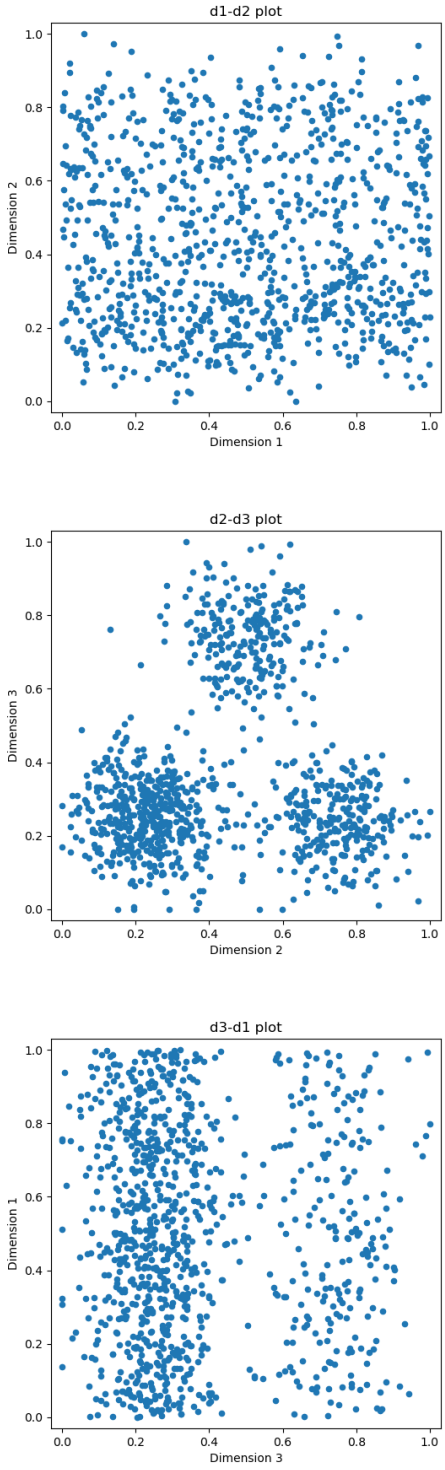

Figure 2: Scatter plots for pairs of components

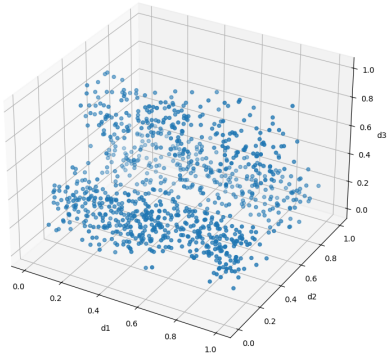

Figure 3: Three-dimensional scatter plot of the data

4 Entropies

To calculate the entropy, we discretize the data with $m = 64$ grid regions. In [1], the probability-based entropy is defined in equation (6.2) as

$$E = -\sum_{i=1}^{m}[p_i \log(p_i) + (1 - p_i) \log(1 - p_i)]]$$

where p_i is the fraction of data points in grid region i. We first calculate the entropy resulting from three-dimensional grid regions, then the three entropies arising from two-dimensional grid regions for each pair of components, and lastly the three entropies resulting from one-dimensional grid regions for each component individually. The results are presented in Table 4.

3-dimensional	d1-d2-d3		
entropy value	4.809		
2-dimensional	d1-d2	d2-d3	d3-d1
entropy values	4.993	4.398	4.775
1-dimensional	d1	d2	d3
entropy values	5.123	4.983	4.723

Table 1: Probability-based entropies

Lower entropy suggests well-defined clusters, whereas higher entropy suggests the data is more uniformly distributed. This is because the contribution towards the total entropy for a particular grid region i is small if p_i is close to 0 (very few points lie in that region) or close to 1 (very many points lie in that region). If there are well-defined clusters, more regions should fall into one of these categories. This explains why we see the lowest entropy value for the d2-d3 combination, where we already observed clusters in the scatter plot in Figure 2. Among the one-dimensional entropy values, the one for d3 is the lowest, which also aligns with our observation in the histogram in Figure 1.

5 Greedy backward selection

Simulating greedy backward selection as a search strategy for a potentially optimal set of data dimensions, we start with a full set of features with $E = 4.809$. Next, we remove d1 as it has the highest entropy. Now, the entropy for the remaining dimensions 2 and 3 is 4.398. This is lower than all one-dimensional entropies and removing features would only increase entropy. Thus, we arrive at the solution $\{d2, d3\}$ as the optimal set of components.

6 Greedy forward selection

Simulating greedy forward selection as a search strategy for a potentially optimal set of data dimensions, we start with zero features. We add d3 to the set because it has the lowest entropy. Next, we compare the entropy for d1-d3 and d2-d3. Since d2-d3 is lower we add d2 to the set. Now adding d1 would increase the entropy, so we arrive at the solution $\{d2, d3\}$ as the optimal set of components.

7 Analysis

Both strategies arrive at the same solution, namely components 2 and 3, which is the lowest entropy value in Table 4, and the same we expected from the scatter plot in Figure 2, where clusters were clearly visible for this combination.

8 Why 64 bins?

Why was $m = 64$ a good choice? If there are not enough bins, the distribution looks more uniform and entropy values are very high. If there are too many bins, many will be empty and entropy values are very low. In our case, we had 1000 samples and wanted a value m which would work well in all 3 dimensions. The lower bound arises from calculating the three-dimensional entropy, where we had $\sqrt[3]{64} = 4$ bins along each axis, and a lower number might have been problematic. The upper bound comes from one-dimensional entropy, where we have m bins, and a too high number could lead to artificially low entropy. The value $m = 64$ was picked as a good middle ground. Another similar value such as $m = 5^3 = 125$ might have also worked well.

In higher dimensions, we might run into problems, since assuming we don't want less than 3 points per axis, we need $m \geq 3^d$, which grows exponentially with the number of dimensions. Therefore, we may need to use a different method to calculate entropy.

9 Conclusions

To summarize, in this experiment we have studied how clustering tendency can be observed visually and numerically, by calculating the entropy. The clustering, or lack thereof, that we saw in our scatter plots in Section 3 was reflected in low or high entropy values in Section 4. Therefore, the entropy appears to be an accurate reflection of the patterns we see in the data. The greedy search strategies in Section 5 and Section 6 also both arrived at this result, indicating that at least in this example, they are reliable methods to find the (local) minimum entropy.

10 Appendix – Code

```
import pandas as pd
import numpy as np
import math

# Step 1
df = pd.read_csv("clustering-tendency.csv", header=None)
df.columns = ["d1","d2","d3"]
print(df.shape)
df.head(5)

# Step 2
df.hist(bins=64, figsize=(18,4), layout=(1,3), grid = False, range=(0,1))
plt.savefig("plot1.png")
```

```
# Step 3
ax = df.plot.scatter(x="d1", y="d2", title="d1-d2 plot")
ax.set_aspect('equal')
plt.savefig("plot2.png")

ax = df.plot.scatter(x="d2", y="d3", title="d2-d3 plot")
ax.set_aspect('equal')
plt.savefig("plot3.png")

ax = df.plot.scatter(x="d3", y="d1", title="d3-d1 plot")
ax.set_aspect('equal')
plt.savefig("plot4.png")

from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(12, 9))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(df["d1"], df["d2"], df["d3"])
ax.set_xlabel("d1")
ax.set_ylabel("d2")
ax.set_zlabel("d3")
plt.savefig("plot5.png")
plt.show()

# Step 5
hist3D, _ = np.histogramdd(df.values,bins=[4,4,4], range=[(0,1),(0,1),(0,1)])
Entropies = {}
total = hist3D.sum()
print(df.columns)
E= 0
p = hist3D.flatten()/total
for p_i in p:
    if p_i > 0:
        E += p_i * math.log(p_i)
    if p_i < 1:
        E += (1-p_i) * math.log(1-p_i)
E *= -1
print (f"Entropy: {E}")
Entropies3D = {tuple(df.columns): E}
#print(Entropies3D)
Entropies.update({0 : Entropies3D})

# Step 6
Entropies2D = {}
for x,y in [("d1","d2"),("d2", "d3"),("d3","d1")]:
    hist2D, _ = np.histogramdd(df[[x,y]].values,bins=[8,8], range=[(0,1),(0,1)])
    total = hist2D.sum()
    E= 0
    p = hist2D.flatten()/total
    for p_i in p:
        if p_i > 0:
            E += p_i * math.log(p_i)
        if p_i < 1:
            E += (1-p_i) * math.log(1-p_i)
    E *= -1
    Entropies2D.update({tuple([x,y]):E})
    print (f"Entropy for {x}-{y}: {E}")
#print(Entropies2D)
Entropies.update({1 : Entropies2D})

# Step 7
Entropies1D = {}
```

```
for d in ["d1", "d2", "d3"]:
    hist2D, _ = np.histogram(df[d].values,bins=64, range=(0,1))
    total = hist2D.sum()
    E= 0
    p = hist2D.flatten()/total
    for p_i in p:
        if p_i > 0:
            E += p_i * math.log(p_i)
        if p_i < 1:
            E += (1-p_i) * math.log(1-p_i)
    E *= -1
    Entropies1D.update({tuple([d]):E})
    print (f"Entropy for {d}: {E}")
print(Entropies1D)
Entropies.update({2 : Entropies1D})
```

11 Bibliography

References

[1] Charu C. Aggarwal. *Data Mining - The Textbook*. Springer, 2015.