Machine Learning A 2024
Exam 2024

214

Contents

1 Conceptual questions

1.1 Cross-validation and overfitting (5 points) [Yevgeny]

1.1.1

Given the algorithms used in this course, my favorite is the k-fold cross-validation (my go-to is usually $k = 5$), as it is intuitive and gives fairly good results in the cases in which I have been using it.

1.1.2

No matter the cross-validation technique, there is almost always a possibility of overfitting. Thus in the k-fold cross-validation, we also may experience overfitting, but it comes down to the data and the model of choice. E.g. if we have an overly specific model relative to the data, we will probably end up overfitting even using k-fold.

1.2 Clustering and visualization (5 points) [Sadegh]

1.2.1

Given in the lecture about clustering[1], the following theorem is presented:

$$\mathbb{E}[\phi] \leq 8(\log k + 2) \cdot \phi^{\star} \tag{1}$$

Thus we see that on average, the cost that the cluster produced is bound by that with a factor of $8(\log k + 2)$ from the optimal cluster, but as it is the average, there must also be some that are worse. And as the statement presented says that *every* cluster produced the statement presented is **false**, since there can be some that are worse than this average.

1.2.2

In general all algorithms in this course use the assumption that the data is I.I.D[2], but more specifically for the visualization algorithms like PCA and t-sne, we build on the manifold assumption[3]. The manifold assumption says, that the data in $\mathbb{R}^d$ is (or is close to) a $\mathbb{R}^d$ manifold. The example often used in $\mathbb{R}^3$ is the spring, which can be "straightened" into a simple steel rod, and thus be represented in $\mathbb{R}^2$

[1] Week 6 slides: "k-means Clustering", slide 27.
[2] Week 7 slides: Limitations & Pitfalls, slide 4.
[3] Week 7 slides: "Non-linear Dimensionality Reduction via Stochastic Neighbor Embedding", slide 6.

2 Optimal decisions (16 points points) [Christian]

2.1

Given the probabilities, we see that it is always beneficial to predict $y = 0$, since when $X = 0$ we know $Y = 0$ with $\mathbb{P}(Y = 0|X = 0) = 1$, and when $X = 1$ then $\mathbb{P}(Y = 0|X = 1) = 0,8 > \mathbb{P}(Y = 1|X = 1) = 0,2$. And since $\mathbb{P}(X = 1) = \mathbb{P}(X = 0) = \frac{1}{2}$ the risk/$\mathbb{E}[\mathcal{L}]$ can be computed as

$$\mathbb{E}[\mathcal{L}] = \frac{1}{2}\underbrace{\mathbb{P}(y \neq Y|x = 0)}_{=0} + \frac{1}{2}\underbrace{\mathbb{P}(y \neq Y|x = 1)}_{=0,2} = \frac{1}{2} \cdot 0,2 = 0,1 \tag{2}$$

2.2

Given the probabilistic classifier with outputs produced by $h(x)$, and we want to calculate the risk/$\mathbb{E}[\mathcal{L}]$, we can do it the same way as before with the equation

$$\mathbb{E}[\mathcal{L}] = \mathbb{P}(x = 0) \cdot \mathbb{P}(h(x) \neq Y|x = 0) + \mathbb{P}(x = 1) \cdot \mathbb{P}(h(x) \neq Y|x = 1) \tag{3}$$

We know that $\mathbb{P}(x = 0) = \mathbb{P}(x = 1) = \frac{1}{2}$, and that $\mathbb{P}(h(x) \neq Y|x = 0) = 0$ since $\mathbb{P}(Y = 0|X = 0) = 1$, thus whenever $x = 0$ then $y = 0$ which is always correctly predicted. Thus we need $\mathbb{P}(h(x) \neq Y|x = 1)$. First we look at the probability that $h(x)$ produces the correct output given $x = 1$:
When $h(x) = 0$ and $y = 0$:

$$\mathbb{P}(h(x) = 0, y = 0|x = 1) = \mathbb{P}(h(x) = 0|x = 1) \cdot \mathbb{P}(y = 0|x = 1) = 0,8 \cdot 0,8 = 0,64$$

When $h(x) = 1$ and $y = 1$:

$$\mathbb{P}(h(x) = 1, y = 1|x = 1) = \mathbb{P}(h(x) = 1|x = 1) \cdot \mathbb{P}(y = 1|x = 1) = 0,2 \cdot 0,2 = 0,04$$

Thus the probability of giving the correct prediction when $x = 1$ is $0,68$, and thus the probability of predicting wrong must be

$$\mathbb{P}(h(x) \neq Y|x = 1) = 1 - \mathbb{P}(h(x) = 0, y = 0|x = 1) - \mathbb{P}(h(x) = 1, y = 1|x = 1) = 1 - 0,68 = 0,32$$

Now calculating equation (3):

$$\mathbb{E}[\mathcal{L}] = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 0,32 = 0,16 \tag{4}$$

Thus the risk is $0,16$, and thus worse than the deterministic classification rule in the previous task.

2.3

If a better probabilistic classifier $h(x)$ would exist, then it would have an expected loss/risk lower than $\mathbb{E}[L] < 0,1$. Thus we can find a probability in which the expected loss is lower than 0,1. Thus we would have to

$$\mathbb{E}[\mathcal{L}] = \frac{1}{2} \cdot \mathbb{P}(h(x) \neq Y | x = 0) + \frac{1}{2} \cdot \mathbb{P}(h(x) \neq Y | x = 1) < 0,1 \tag{5}$$

First, we see that when $x = 0$, any deviation from just predicting $y = 0$ would increase the risk, as this will always happen, and thus $\mathbb{P}(h(x) \neq y | x = 0) = 0$. But for $x = 1$, we have some uncertainty and thus we want $\mathbb{P}(h(x) \neq y | x = 1)$ in terms of p, where p is the probability of predicting $y = 1$, and thus $\mathbb{P}(y = 1 | x = 1) = p$ and likewise is $\mathbb{P}(y = 0 | x = 1) = 1 - p$. Thus to get the probability that we predict wrong in terms of p, we look at

$$\mathbb{P}(Y \neq h(x) | x = 1) = \mathbb{P}(Y = 1, h(x) = 0 | x = 1) + \mathbb{P}(Y = 0, h(x) = 1 | x = 1) \tag{6}$$

Where

$$\mathbb{P}(Y = 1, h(x) = 0 | x = 1) = (1 - p) \cdot 0,2$$

And

$$\mathbb{P}(Y = 0, h(x) = 1 | x = 1) = p \cdot 0,8$$

Inserting into (6)

$$\mathbb{P}(Y \neq h(x) | x = 1) = (1 - p) \cdot 0,2 + p \cdot 0,8 = 0,2 + 0,6p \tag{7}$$

Inserting back into (5), and solving for p:

$$\frac{1}{2} \cdot (0,2 + 0,6p) < 0,1 \tag{8}$$

$$0,1 + 0,3p < 0,1 \tag{9}$$

$$p < 0 \tag{10}$$

Thus we see that we can not assign p positive value that will decrease the risk. Thus we can not find any probabilistic classifier better than the deterministic classifier from the first task.

3 Function basis `(20 points points) [Christian]`

3.1

Plotting the mean of $f^{(k)}$ at the points $\mathcal{X}$ as seen below in figure 1

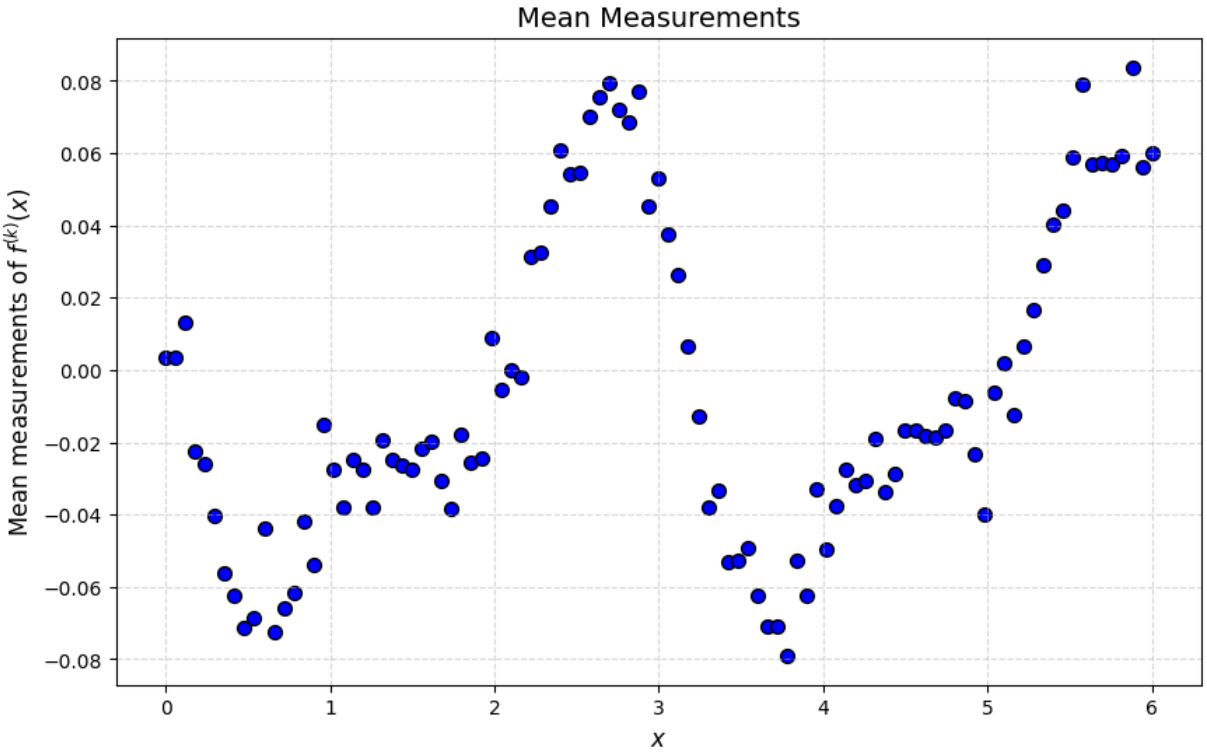

Figure 1: Mean measurements of $f^{(k)}(x)$

3.2

3.2.1 a)

Using the library `sklearn` and the class `pca` to estimate the basis functions from the data. The `pca.fit()` method calculates the principal components from the data, which are equivalent to the basis functions. The function also calculates the variance explained by each basis function stored in the `explained_variance_ratio_` property. Thus we can take the cumulative sum of that property and see how many of the basis functions it takes to explain 95% of the variance, which best can be seen in the plot below in figure 2.

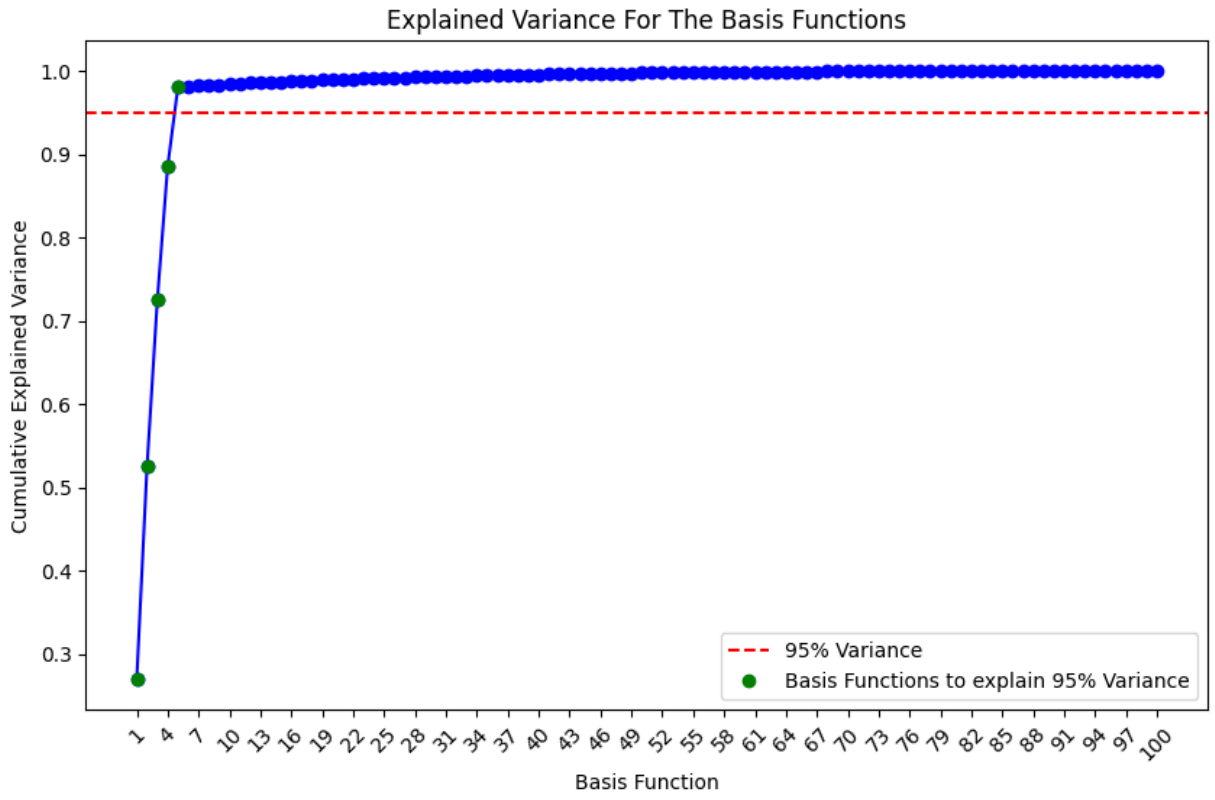

Figure 2: Explained variance from the basis function.

Thus we get that the top 5 basis functions are enough to explain over 95% of the variance.

3.2.2 b)

Plotting the 5 basis functions that explained 95% of the variance below in figure 3.

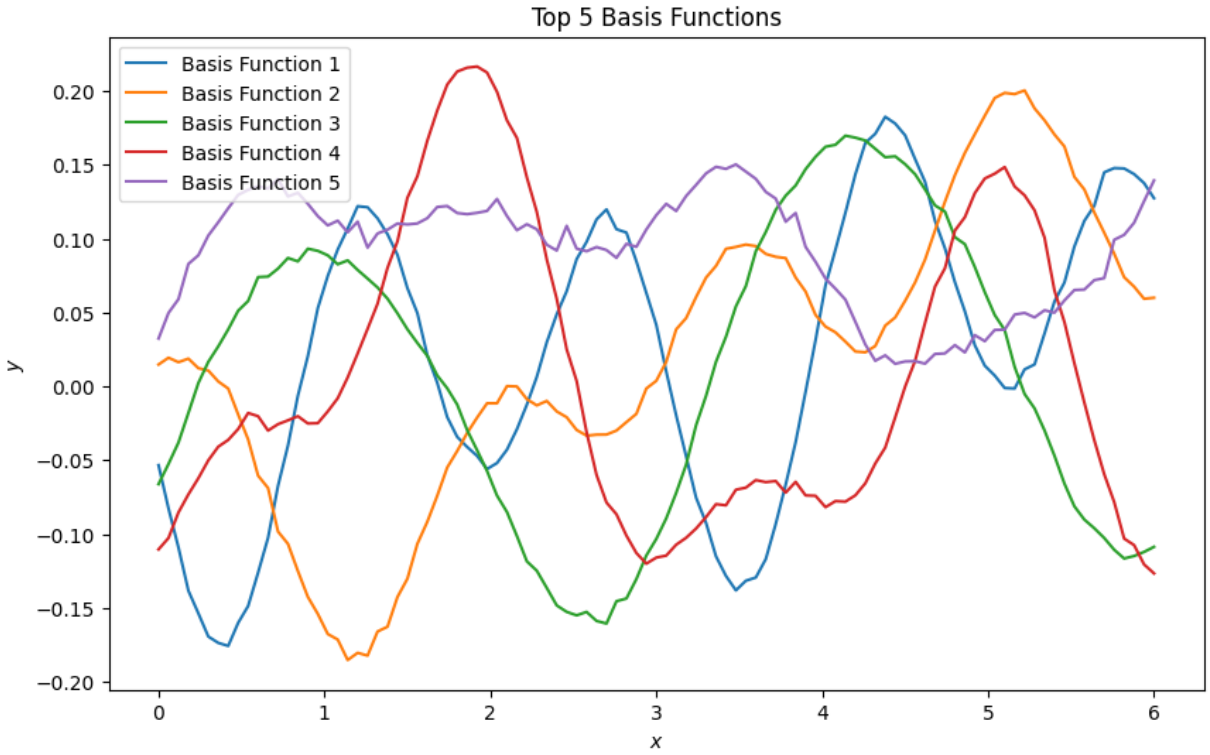

Figure 3: The $F = 5$ basis functions.

3.3

To reconstruct the first function $f^{(1)}$ have the the basis functions g_i, but we are still missing the weights $\alpha_i^{(1)}$. To get the weights, we can use the `transform` method from the `pca` class to project $f^{(1)}$ onto the principal components, which gives us the weights. Thus we can reconstruct using the $F = 5$ basis functions and their weights, which is plotted below in figure 4 along with the original function.

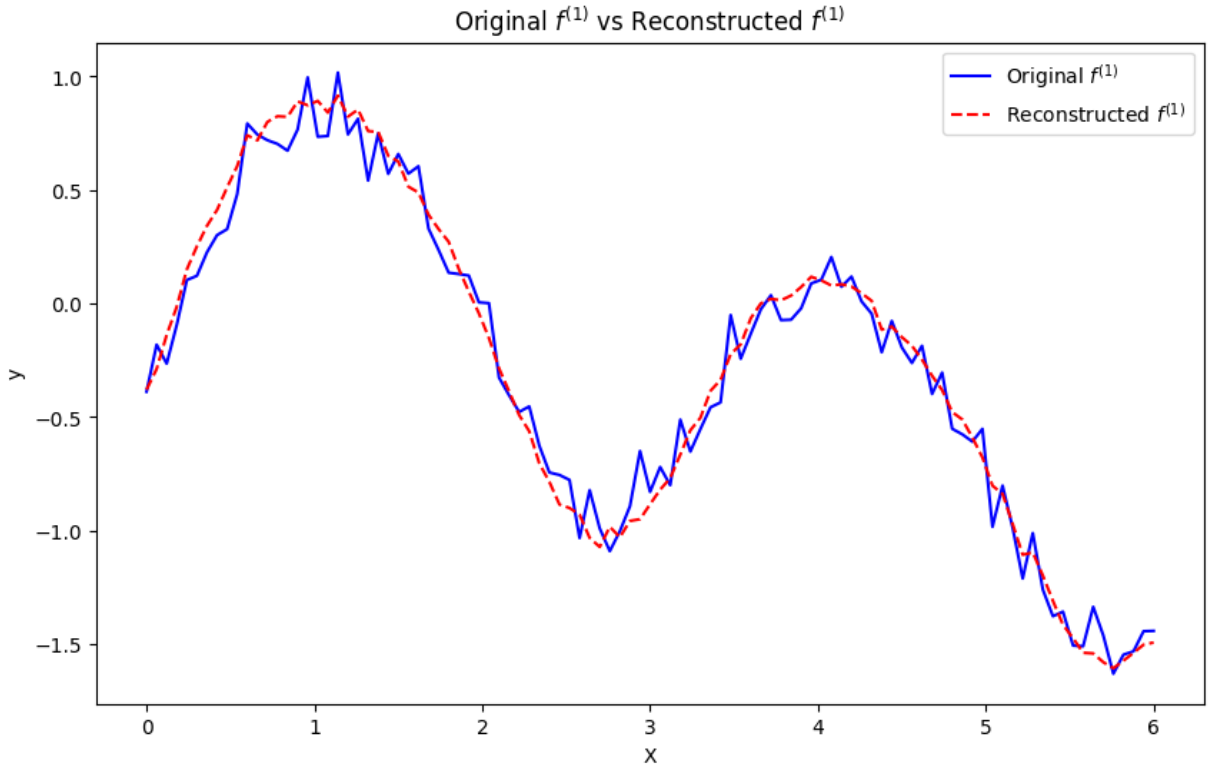

Figure 4: $f^{(1)}$ reconstructed using $F = 5$ basis functions and the original $f^{(1)}$ plotted.

3.4

To give an estimation of the variance of ϵ using the reconstructed and original $f^{(1)}$ from last question, we can simply take the difference of those two, which must an estimation to the error term ϵ, and then take the variance of that

$$\text{var}(\epsilon) = \text{var}\left(f^{(1)}_{\text{original}} - f^{(1)}_{\text{reconstructed}}\right) \tag{11}$$

Which was calculated to be $\text{var}(\epsilon) = 0.0077$.

4 Generalization bound for cross-validation (10 points) [Yevgeny]

4.1

To derive the bound on $L(h_{i,j})$, we can use theorem 3.2[4], and the same approach as that proof for this problem. First, we see that the total amount of models trained is $K \cdot M$, and

[4]Yevgeny Seldin, "Machine Learning: The Science of Selection under Uncertainty.", p. 25

for each model, we use S_i to calculate the empirical loss, and thus the size of S_i must be n/K. We assume the loss is in $[0, 1]$ and the data to be I.I.D. To begin with we need to first find ϵ in

$$\mathbb{P}\left(L\left(\hat{h}^*_{i,j}\right) \geq \hat{L}\left(\hat{h}^*_{i,j}, S_i\right) + \epsilon\right) \tag{12}$$

We need to break the dependency since $\hat{L}$ is a biased estimation of L, and thus, and we therefore can not directly use Hoeffdings inequality. Thus we get the probability of there existing one $h_{i,j}$ that the empirical loss underestimates the loss by no more than ϵ:

$$(12) \leq \mathbb{P}\left(\exists i \in \{1, \ldots, K\}, j \in \{1, \ldots, M\} : L\left(h_{i,j}\right) \geq \hat{L}\left(h_{i,j}, S_i\right) + \epsilon\right) \tag{13}$$

Using the union bound

$$(13) \leq \sum_{i \in \{1,\ldots,K\}, j \in \{1,\ldots,M\}} \mathbb{P}\left(L\left(h_{i,j}\right) \geq \hat{L}\left(h_{i,j}, S_i\right) + \epsilon\right) \tag{14}$$

And now that $\mathbb{E}\left[\hat{L}\left(h_{i,j}, S_i\right)\right] = L\left(h_{i,j}\right)$, we can use Hoeffdings inequality

$$(14) \leq \sum_{i \in \{1,\ldots,K\}, j \in \{1,\ldots,M\}} \exp\left(-2n\epsilon^2\right) \tag{15}$$

And and as mentioned earlier we have the M in the theorem to be substituted with $M \to K \cdot M$, and the n to be substituted with $n \to \frac{n}{K}$, and thus

$$(15) = M \cdot K \cdot \exp\left(-2\frac{n}{k}\epsilon^2\right) \tag{16}$$

Setting (16) equal to δ and solving for ϵ

$$M \cdot K \cdot \exp\left(-2\frac{n}{k}\epsilon^2\right) = \delta \iff \ln\left(M \cdot K \cdot \exp\left(-2\frac{n}{k}\epsilon^2\right)\right) = \ln\left(\delta\right)$$

$$\iff \epsilon = \sqrt{\frac{\ln\left(\frac{M \cdot K}{\delta}\right) \cdot K}{2n}} \tag{17}$$

Inserting (17) into 13

$$\mathbb{P}\left(\exists i \in \{1, \ldots, K\}, j \in \{1, \ldots, M\} : L\left(h_{i,j}\right) \geq \hat{L}\left(h_{i,j}, S_i\right) + \sqrt{\frac{\ln\left(\frac{M \cdot K}{\delta}\right) \cdot K}{2n}}\right) \leq \delta \tag{18}$$

Taking the compliment to get it in terms of $1 - \delta$

$$\mathbb{P}\left(\forall i \in \{1, \ldots, K\}, j \in \{1, \ldots, M\} : L\left(h_{i,j}\right) \leq \hat{L}\left(h_{i,j}, S_i\right) + \sqrt{\frac{\ln\left(\frac{M \cdot K}{\delta}\right) \cdot K}{2n}}\right) \geq 1 - \delta \tag{19}$$

And thus we now have a high probability generalization bound.

4.2

When M increases we will have more models to choose from, which can potentially lead to finding a smaller $\hat{L}\left(h_{i^*,j^*}, S_{i*}\right)$, but as we see that M appears in $\sqrt{\frac{\ln\left(\frac{M \cdot K}{\delta}\right) \cdot K}{2n}}$, which will increase with M, thus making the uncertainty of the empirical loss larger. This is the classic bias-variance trade-off, if M increases we often get a lower bias but also a larger variance.

5 Generalization bound for early stopping with multiple initializations (15 points) [Yevgeny]

5.1

Since we have a infinite but still countable set of hypothesis's, we can use occams razor to make a generalization bound, which is given by theorem 3.3[5] as

$$\mathbb{P}\left(\exists h \in \mathcal{H} : L(h) \geq \hat{L}(h, S) + \sqrt{\frac{\ln\left(\frac{1}{\pi(h)\delta}\right)}{2n}}\right) \leq \delta \tag{20}$$

First we need to establish a prior $\pi(h_{i,t})$. The prior has to put emphasis on the simpler models to avoid overfitting and sum to 1 or less, thus a good prior would be something that decreases the complexity term as i and t increase. I have chosen the prior

$$\pi\left(h_{i,t}\right) = \frac{1}{2^{(i+t)}} \tag{21}$$

Thus we first need to show that it is a valid prior thus that $\sum^{\infty} \pi\left(h\right) \leq 1$:

$$\sum_{i \in \mathbb{N}} \sum_{t \in \mathbb{N}} \frac{1}{2^{(i+t)}} = \sum_{i \in \mathbb{N}} \frac{1}{2^i} \sum_{t \in \mathbb{N}} \frac{1}{2^t} \tag{22}$$

And a geometric series that keeps halving is equal to one, we have

$$(22) = 1 \cdot 1 = 1 \tag{23}$$

And thus the prior is valid, and we can set up (20):

$$\mathbb{P}\left(\exists i \in \mathbb{N}, t \in \mathbb{N} : L(h_{i,t}) \geq \hat{L}(h_{i,t}, S_{\text{val}}) + \sqrt{\frac{\ln\left(2\right)\left(i+t\right) - \ln\left(\delta\right)}{2n}}\right) \leq \delta \tag{24}$$

[5] Yevgeny Seldin, "Machine Learning: The Science of Selection under Uncertainty.", p. 26

To prove that the bound holds, I follow the same approach as in the course notes[6] and first take the union bound, applying Hoeffding's inequality and the fact that the sum of the prior is 1:

$$\begin{aligned}
&\mathbb{P}\left(\exists i \in \mathbb{N}, t \in \mathbb{N} : L(h_{i,t}) \geq \hat{L}(h_{i,t}, S_{\text{val}}) + \sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}}\right) \\
&\qquad \leq \sum_{i \in \mathbb{N}} \sum_{t \in \mathbb{N}} \mathbb{P}\left(L(h_{i,t}) \geq \hat{L}(h_{i,t}, S_{\text{val}}) + \sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}}\right) \qquad (25) \\
&\qquad \leq \sum_{i \in \mathbb{N}} \sum_{t \in \mathbb{N}} \frac{1}{2^{i+t}} \delta \\
&\qquad \leq \delta
\end{aligned}$$

And to get it in terms of $1 - \delta$ we take the compliment to get the final equation

$$\mathbb{P}\left(\forall i \in \mathbb{N}, t \in \mathbb{N} : L(h_{i,t}) \leq \hat{L}(h_{i,t}, S_{\text{val}}) + \sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}}\right) \geq 1 - \delta \qquad (26)$$

5.2

The bound derived in the last task does no longer provide any meaningful bound when the sum of the empirical loss and the complexity term is above 1, thus we can make the inequality

$$\hat{L}(h_{i,t}, S_{\text{val}}) + \sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}} \leq 1 \qquad (27)$$

And given we want it in terms of i we just set $t = 1$, and in the case of the strictest bound when $\hat{L}(h_{i,t}, S) = 0$, we have the inequality

$$\sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}} \leq 1 \qquad (28)$$

Solving for i:

$$\begin{aligned}
&\sqrt{\frac{\ln(2)(i+t) - \ln(\delta)}{2n}} \leq 1 \\
&\qquad \iff \quad i + 1 \leq \frac{2n + \ln(\delta)}{\ln(2)} \qquad (29) \\
&\qquad \iff \quad i \leq \frac{2n + \ln(\delta)}{\ln(2)} - 1
\end{aligned}$$

Thus when n becomes substantially large, it is the important term, and thus the larger the n, the more initialization we can make before the bound becomes trivial, and thus the magnitude of i should be $O(n)$ to keep the bound non-trivial.

[6] Yevgeny Seldin, "Machine Learning: The Science of Selection under Uncertainty.", p. 26

6 Clustering (17 points) [Sadegh]

6.1 Part 1

First finding the min distance from $c_1 = A(-2, 2)$ and $c_2 = D(7, -5)$ to the rest of the points:
$D(B, \{c_1, c_2\}) = \min(\sqrt{8}, \sqrt{130}) = \sqrt{8}$
$D(C, \{c_1, c_2\}) = \min(\sqrt{65}, \sqrt{65}) = \sqrt{65}$
$D(E, \{c_1, c_2\}) = \min(10, \sqrt{10}) = \sqrt{10}$
$D(F, \{c_1, c_2\}) = \min(\sqrt{40}, \sqrt{50}) = \sqrt{40}$
And thus the probability distribution for choosing c_3 is

$$c_3 = \begin{cases} B(0,4) & \text{w.p.} \quad \frac{8}{123} \\ C(6,3) & \text{w.p.} \quad \frac{65}{123} \\ E(4,-6) & \text{w.p.} \quad \frac{10}{123} \\ F(0,-4) & \text{w.p.} \quad \frac{40}{123} \end{cases} \tag{30}$$

6.2 Part 2

6.2.1 a

Using the library `sklearn` and the class `KMeans` that has the K-means algorithm implemented. Thus running the algorithm with the initial centers as

```
kmeans = KMeans(n_clusters=3, init=np.array([[0, 4], [1, 5], [-1, 5.5]]))
kmeans.fit(X)
```

Where the resulting centers can be visualized as seen below in figure 5.

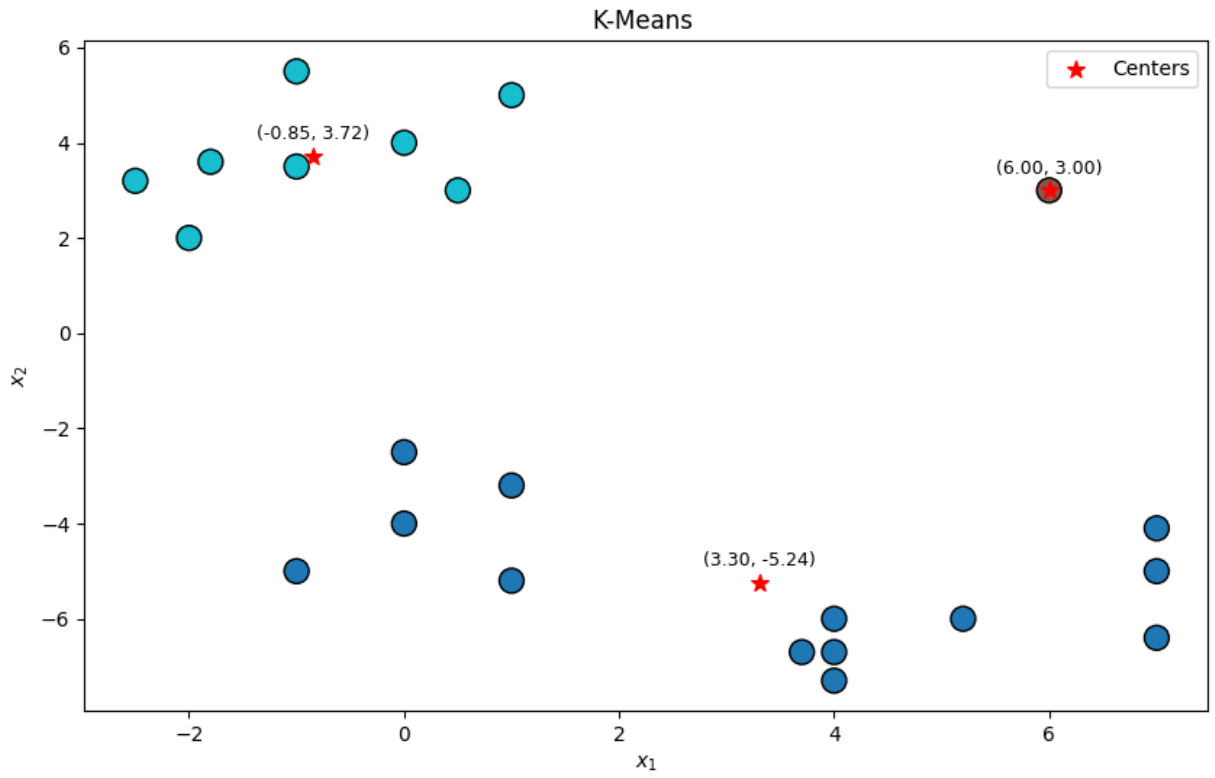

Figure 5: The 3 centers found using K-Means

And the k-means cost is also calculated using the `fit()` method, and can be extracted with the `inertia_` property, which gave a k-means cost at 142.45.

6.2.2 b

To run the k-means 40 times with random initialization seeds each run, I again use the library `sklearn` and the `KMeans` class with the arguments `init='random'` to get the random start centers, and since the method defaults to 10 runs if nothing is specified I set `n_init=1`. I then perform this 40 times and get the best cost, where the cost is based on the `intertia` property, which is the sum of the squared distance from the point to its closest center. The code to perform this can be seen below

```
n = 40
best_cost = np.inf
best_centers = None
best_labels = None
best_times_got = 0
for _ in range(n):
    # n_init=1 to avoid multiple runs
    kmeans = KMeans(n_clusters=3, init='random', n_init=1)
```

```
    kmeans.fit(X)
    cost = kmeans.inertia_
    if cost < best_cost:
        best_cost = cost
        best_centers = kmeans.cluster_centers_
        best_labels = kmeans.labels_
        best_times_got = 1
    elif cost == best_cost: # Count times equal to best cost
        best_times_got += 1
```

The best centers this found can be seen below in figure 6.

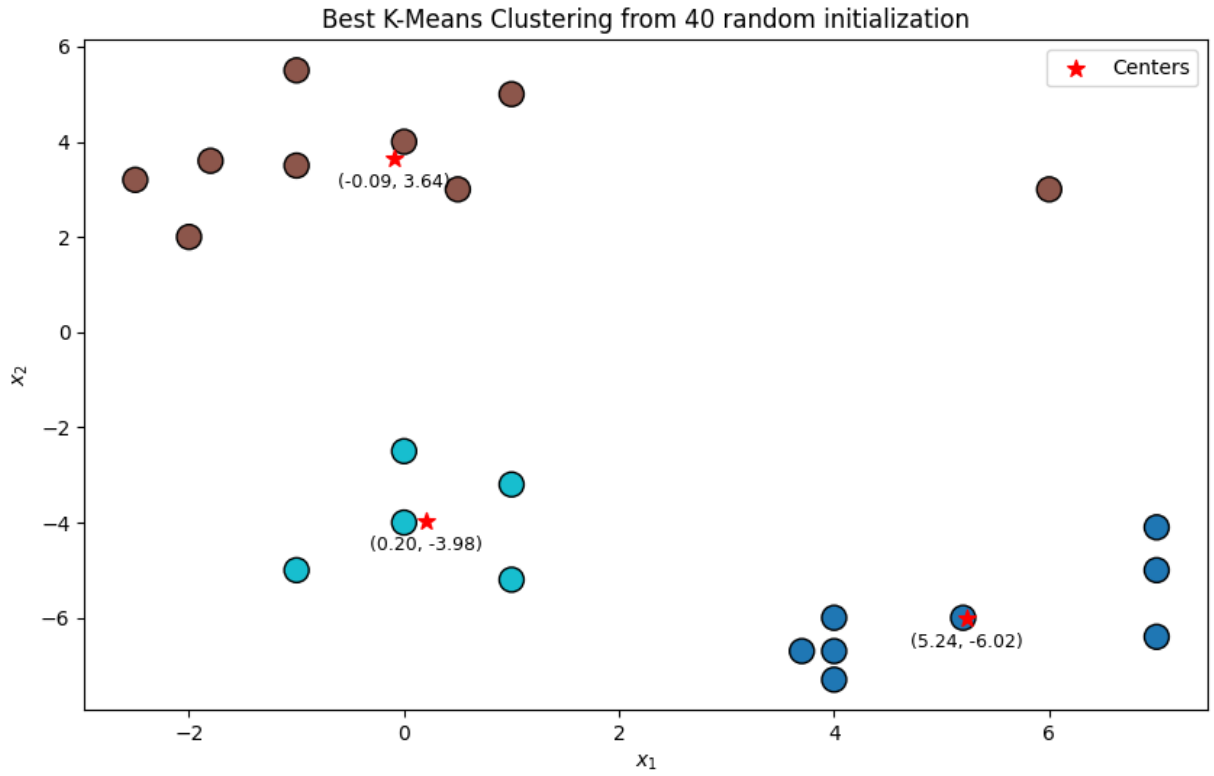

Figure 6: Best centers from K-Means from 40 random initializations.

Which got a K-Means cost of 93.67 and was encountered 15 times during the 40 runs. To get deterministic results, I have set the random seed as `np.random.seed(24)`.

6.2.3 c

Using the same approach as the previous task but with the `init='k-means++'` argument instead of using the random initialization. This gives the same best cluster as seen below in figure 7.

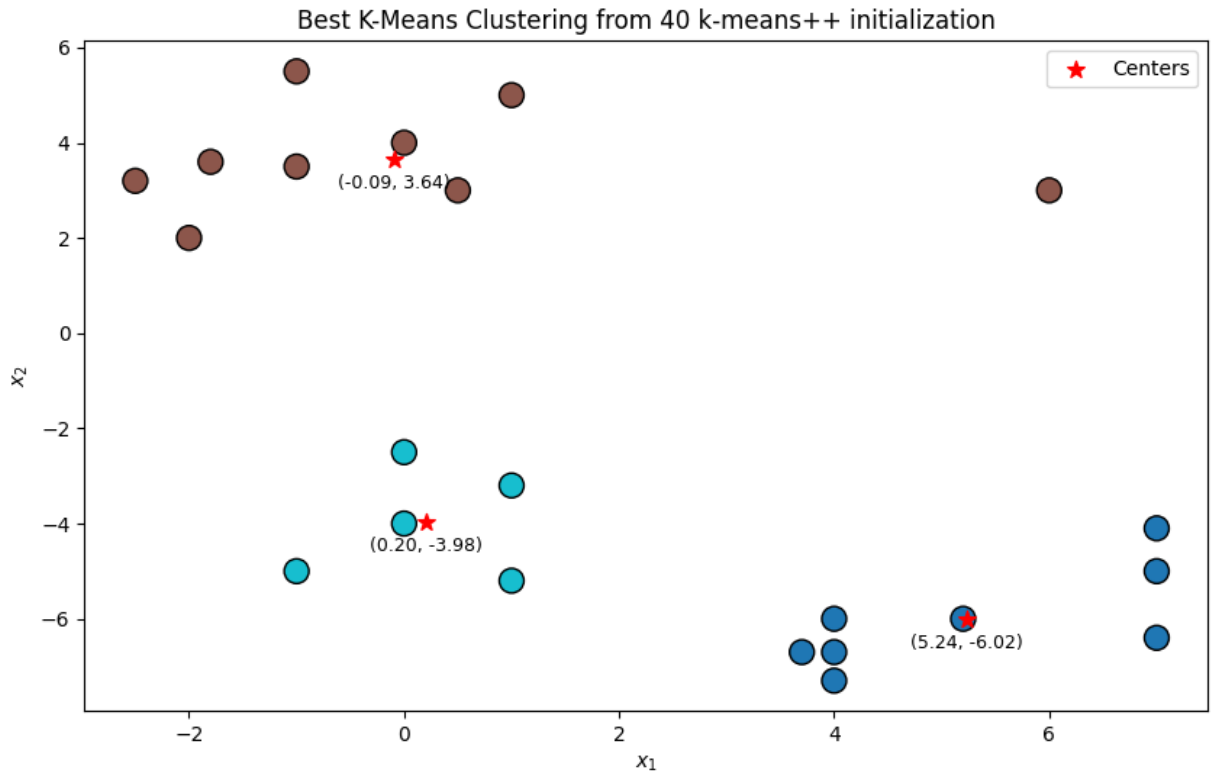

Figure 7: Best centers from K-Means from 40 k-means++ initializations.

The best centers got a K-Means cost of 93.67 and 32 of the 40 runs encountered those centers.

6.2.4 d

Both initialization methods end up finding the same best centers, however, the K-Means++ initialization method is must more consistent in getting those centers at 32 times out of 40 compared to the only 15 times the random initialization methods got it.

7 Visualization `(12 points) [Sadegh]`

7.1

Plotting the data set with an orange color for the first half of the data and a purple color for the second half. I have plotted it from 3 different angles as can be seen below in figure 8.

Figure 8: Data plotted in 3d from 3 different angles

7.2

Using the `PCA` class from the `sklearn` library to produce a 2d map of the dataset

```
pca = PCA(n_components=2) # Initialize PCA with 2 components
data_2d = pca.fit_transform(data) # Transform the data to 2D
```

Where the produced principle components are plotted below in figure 9.

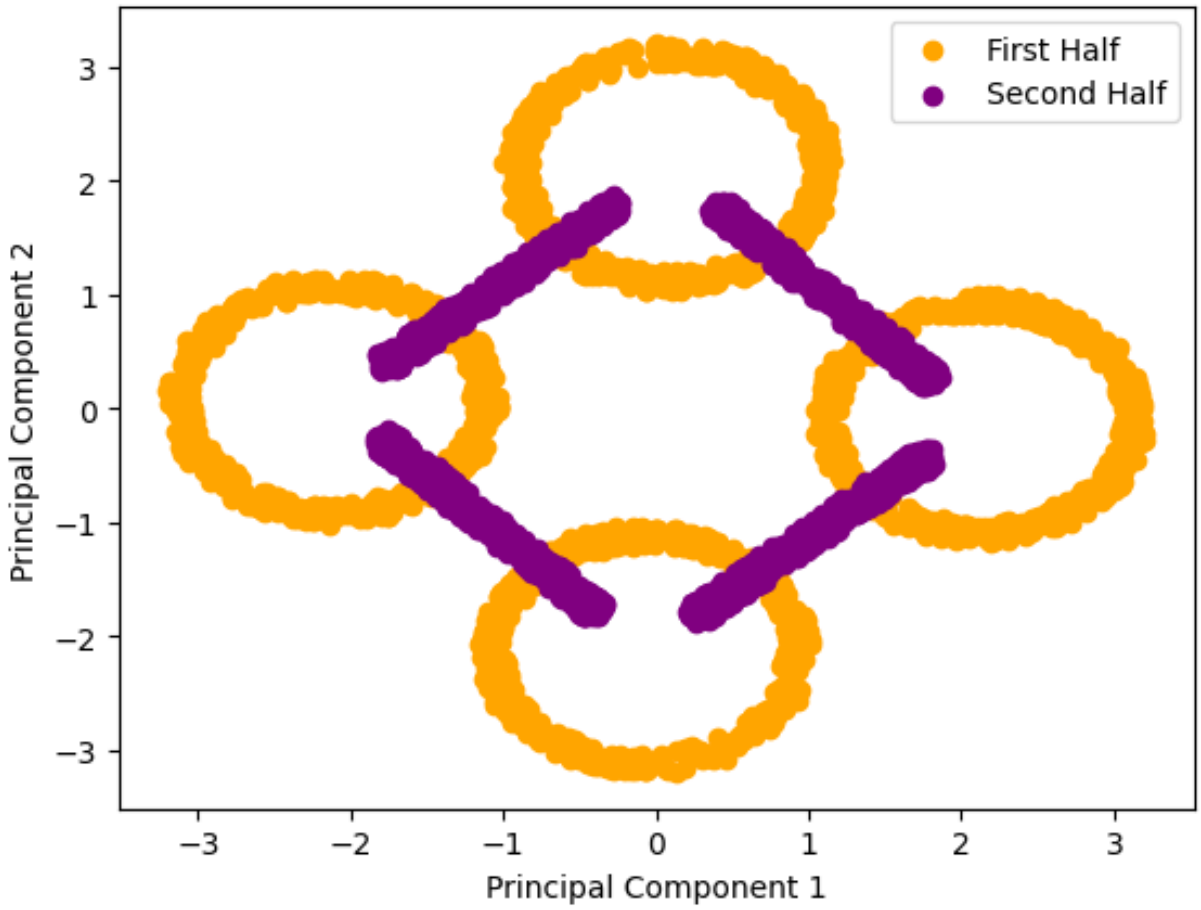

Figure 9: Tha PCA transformation of the data.

7.3

Using the `TSNE` class from the `sklearn` library to perform the t-SNE algorithm on the data to get it mapped into 2d using 3 different perplexities of 40, 120, and 300. The class is further initialized with the arguments `init='random'` to denote a random starting point and `random_state=25` to ensure the 3 runs are initialized the same.

```
perplexities = [40, 120, 300]
for i, perplexity in enumerate(perplexities):
    tsne = TSNE(n_components=2, perplexity=perplexity, init='random', random_state=25)
    data_2d = tsne.fit_transform(data)
```

The 3 plots produced can be seen below in figure 10.

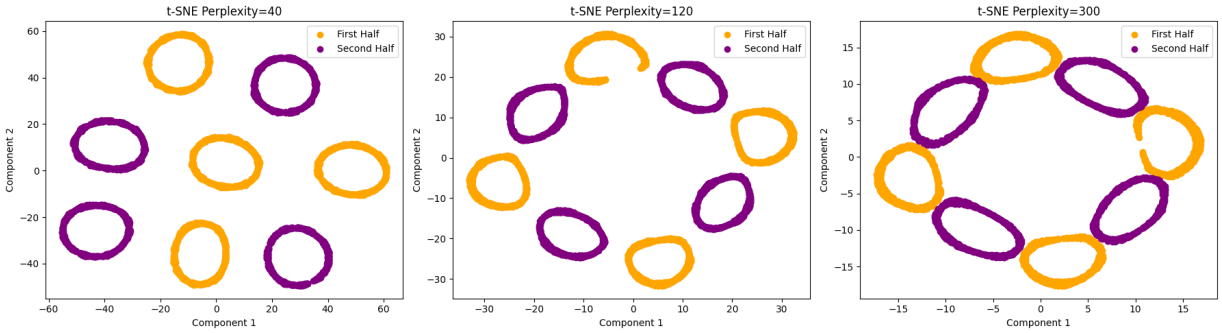

Figure 10: 3 plots produced by the t-SNE transformation with 3 different perplexities.

7.4

For the 3 plots produced with different perplexities in part 3, we see that all the different plots group the circles together, but the bigger the perplexity the closer the circles are together. When looking at the different angles in part 1, we see that the data consists of 8 circles interlocked in 2 different rotations by 90 degrees, thus the data is quite close to each other. Therefore looking at it qualitatively, I believe the third plot with a perplexity of 300 is the best representative of the 3d plot, as it is symmetric and captures that the data is represented in 8 circles.
When comparing the plot produced in part 2 by PCA and the plot produced with t-SNE with a perplexity of 300, we see that the PCA plot does not capture the 8 circles, as it ignores the 3'rd axis, as the PCA plot looks like the 3d plot from above (as we can see in part 1). I will therefore argue that the t-SNE transformation better captures and preserves the data structure from 3d to 2d.