Final Exam
Machine Learning B – 2022

Exam ID 141

Deadline January 20, 2022 at 12:00

1 Gaussian Kernel (20 Points) [Christian]

We want to prove that the Gaussian kernel on $\mathbb{R}^d$ for a positive integer d

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma||\mathbf{x} - \mathbf{x}'||^2), \quad \gamma > 0$$

can be expressed as the inner product in an infinite-dimensional feature space. We first prove an intermediate result. Let k' be a kernel on $\mathcal{X}$, and let $\phi : \mathcal{X} \longrightarrow \mathbb{R}$. Then

$$k''(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})k'(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x}')$$

is a kernel on $\mathcal{X}$. To see this, note that per 5., page 29 in Christian Igels notes on Kernel-based methods, we have that

$$k_\phi(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})\phi(\mathbf{x}')$$

Is a positive definite kernel. We can now write

$$k''(\mathbf{x}, \mathbf{x}') = k'(\mathbf{x}, \mathbf{x}')k_\phi(\mathbf{x}, \mathbf{x}')$$

Because k' is a kernel, it now follows from 3. page 29 in Christian Igels notes on Kernel-based methods that k'' is indeed a positive definite kernel. We are now ready to prove our main result. Notice that from hint 3 we can split the Gaussian kernel as

$$k(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\langle\mathbf{x}, \mathbf{x}\rangle)\exp(2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle)\exp(-\gamma\langle\mathbf{x}', \mathbf{x}'\rangle)$$

Notice that

$$k_{\exp}(\mathbf{x}, \mathbf{x}') = \exp(2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle)$$

Is actually a positive definite kernel, since we now per hint 1 that,

$$k_1(\mathbf{x}, \mathbf{x}') = \langle\mathbf{x}, \mathbf{x}'\rangle^1$$

is a kernel, and per 1. page 29 in Christian Igels notes on Kernel-based methods, since $2\gamma \in \mathbb{R}^+$ we have that

$$k_{1_{\text{scaled}}}(\mathbf{x}, \mathbf{x}') = 2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle$$

is a kernel. It then follows from 4. page 29 in Christian Igels notes on Kernel-based methods that

$$k_{\exp}(\mathbf{x}, \mathbf{x}') = \exp(2\gamma\langle\mathbf{x}, \mathbf{x}'\rangle) = e^{k_{1_{\text{scaled}}}(\mathbf{x}, \mathbf{x}')}$$

is indeed also a kernel. Take $\phi(\mathbf{x}) = \exp(-\gamma\langle\mathbf{x}, \mathbf{x}\rangle)$. We can then write the Gaussian kernel as

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})k_{\exp}(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x}')$$

And we can apply our intermediate result, to obtain that the Gaussian kernel is indeed a positive definite kernel. Now, per the last paragraph on page 26 in Christian Igels

notes on Kernel-based methods, we can then construct a canonical feature space, where the Gaussian kernel can be expressed as an inner product. To prove our main result, it now suffices to show that this feature space must necessarily be infinite dimensional.

We focus on $k_{\exp}(\mathbf{x}, \mathbf{x}')$, as the $\phi(x), \phi(x')$ factors simply corresponds to scaling $k_{\exp}(\mathbf{x}, \mathbf{x}')$. Per hint 2, we can write the exponential function as a power series, such that

$$k_{\exp}(\mathbf{x}, \mathbf{x}') = \exp(2\gamma\langle \mathbf{x}, \mathbf{x}'\rangle) = \sum_{i=0}^{\infty} \frac{(2\gamma\langle \mathbf{x}, \mathbf{x}'\rangle)^i}{i!} = \sum_{i=0}^{\infty} \frac{(2\gamma)^i}{i!} \cdot \langle \mathbf{x}, \mathbf{x}'\rangle^i$$

We recognise the polynomial kernel inside the sum, such that we can write

$$k_{\exp}(\mathbf{x}, \mathbf{x}') = \sum_{i=0}^{\infty} \frac{(2\gamma)^i}{i!} \cdot k_i(\mathbf{x}, \mathbf{x}')$$

Where k_i denotes the i'th polynomial kernel on $\mathbb{R}^d$, as in the notation of hint 2. We thus see that the Gaussian kernel is a weighted infinite sum of polynomial kernels. Per hint 2, for each positive integer D there exists feature maps Φ_D such that,

$$k_D(\mathbf{x}, \mathbf{x}') = \langle \Phi_D(\mathbf{x}), \Phi_D(\mathbf{x}')\rangle$$

Consider adding two polynomial kernels k_D and k_C, for positive integers D and C. Using the above we can write the sum of these kernels as

$$k_\sum(\mathbf{x}, \mathbf{x}') = k_D(\mathbf{x}, \mathbf{x}') + k_C(\mathbf{x}, \mathbf{x}') = \langle \Phi_D(\mathbf{x}), \Phi_D(\mathbf{x}')\rangle + \langle \Phi_C(\mathbf{x}), \Phi_C(\mathbf{x}')\rangle$$

And from this this we see that the feature map corresponding to $k_\sum$ is given by $\Phi_\sum(\mathbf{x}) = (\Phi_C(\mathbf{x}), \Phi_D(\mathbf{x}))$ since we have that,

$$\begin{aligned}
\langle \Phi_\sum(\mathbf{x}), \Phi_\sum(\mathbf{x}')\rangle &= \langle (\Phi_C(\mathbf{x}), \Phi_D(\mathbf{x})), (\Phi_C(\mathbf{x}'), \Phi_D(\mathbf{x}'))\rangle \\
&= \langle \Phi_D(\mathbf{x}), \Phi_D(\mathbf{x}')\rangle + \langle \Phi_C(\mathbf{x}), \Phi_C(\mathbf{x}')\rangle \\
&= k_\sum(\mathbf{x}, \mathbf{x}')
\end{aligned}$$

This shows that adding two polynominal kernels corresponds to the feature map that combines the individual feature maps into a single vector. This means that if, say $\Phi_D(\mathbf{x}) \in \mathbb{R}^n$ and $\Phi_C(\mathbf{x}) \in \mathbb{R}^m$, then $\Phi_\sum(\mathbf{x}) \in \mathbb{R}^{n+m}$.

Since we have expressed our Gaussian kernel as an infinite weighted sum of polynomial kernels, it now follows that the corresponding feature map lives in an infinite dimensional feature space. As argued above, since we have shown that the Gaussian kernel is a positive definite kernel, we know that there exists a feature space where the Gaussian kernel can be expressed as an inner product. Combining these two facts, we have thus shown that the Gaussian kernel can be expressed as an inner product in an infinite-dimensional feature space.

2 Landcover Classification with XGBoost (20 Points) [Fabian]

1.

The full notebok is provided in the code folder in the notebook "LandcoverClassification.ipynb".

We first flatten the training and test data. We transform the data into a matrix where each row corresponds to the features of an instance. This matrix will have $13 \cdot 13 \cdot 12 \cdot 6 = 12168$ columns and as many rows as there a training and test instances, respectively. We flatten the data with the `numpy.reshape` function applied to the training and test data, respectively, with the `order` parameter set to 'C', and the `shape` parameter set to $(x, 12168)$ where x is the number of test and training instances, respectively. We then split the training data into a training (90% of training data) and validation set (10% of training data) using the `train_test_split` function from the `sklearn.model_selection` module. We set the `random_state` parameter equal to 120 for reproducability. A code snippet is provided below

```
from sklearn.model_selection import train_test_split
X_train_flat, X_eval_flat, y_train, y_eval = train_test_split(X_train_flat,
y_train, test_size=0.1, random_state=120)
```

2.

We train a XGBoost classification model. We use the `XGBClassifier` class from the `xgboost` library. We set `colsample_bytree=0.02`, `learning_rate=0.1`, `max_depth=8`, `reg_lambda=1`, and `n_estimators=100` as specified in the assignment. Further we specify `objective = multi:softmax`, to use the softmax objective function for fitting the model. To monitor the training process using the multiclass logloss, we also set `eval_metric = mlogloss`. The code snippet below shows the initialization of the `XGBClassifer`

```
import xgboost as xgb
classifier = xgb.XGBClassifier(objective = "multi:softmax",colsample_bytree=0.02,
learning_rate=0.1, max_depth=8, reg_lambda=1,
n_estimators=100, eval_metric = "mlogloss")
```

We then use the `fit` method to train the model. We monitor the training on the training and validation set by setting `eval_set` to a list containing tuples of the training and validation instances. A code snippet is provided below.

```
classifier.fit(X_train_flat, y_train,
eval_set=[(X_eval_flat, y_eval), (X_train_flat, y_train)])
```

3.

We can now plot the multiclass loglosses for the training and validation sets as a function of boosting rounds. These can be extracted from the `evals_results` attribute. We plot below

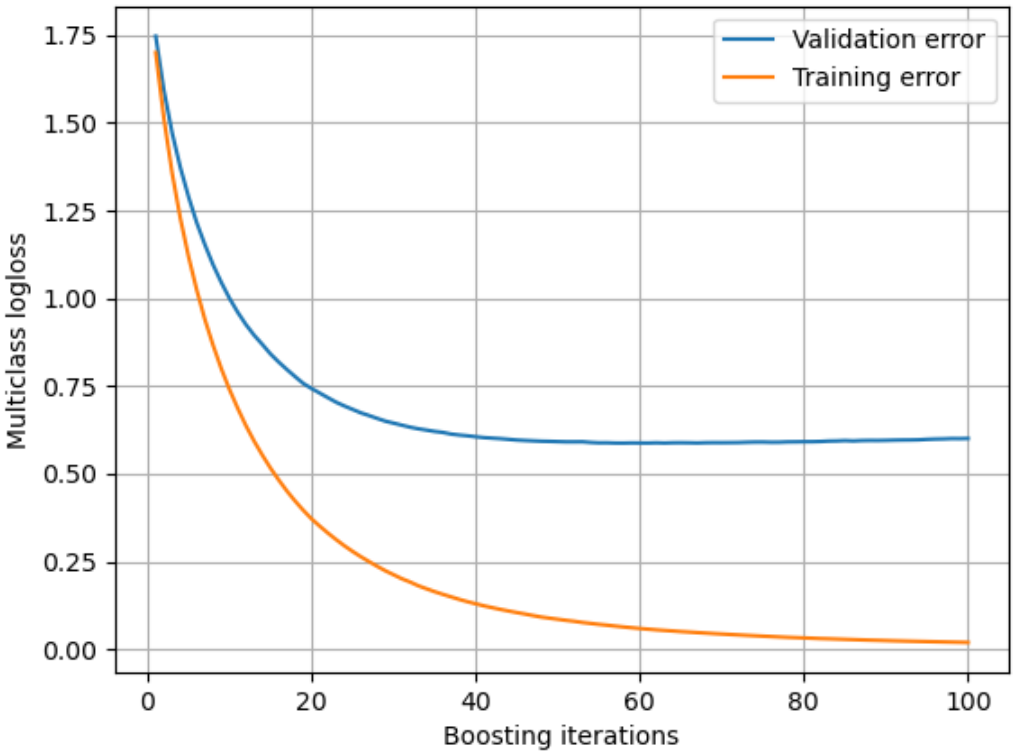


Figure 1: Multiclass log-losses as a function of boosting rounds

4.

We compute the induced accuracy by comparing predicted labels of the test data obtained by the `predict` method, with the test labels. We obtain a accuracy of 0.8217366. Finally, we plot a confusion matrix on the test data to visualize where the model has made errors. For this, we utilize the `from_predictions` method from

the `ConfusionMatrixDisplay` class from `sklearn.metrics`. We set `normalize=True` such that the rows corresponding to the true labels sum to 1.

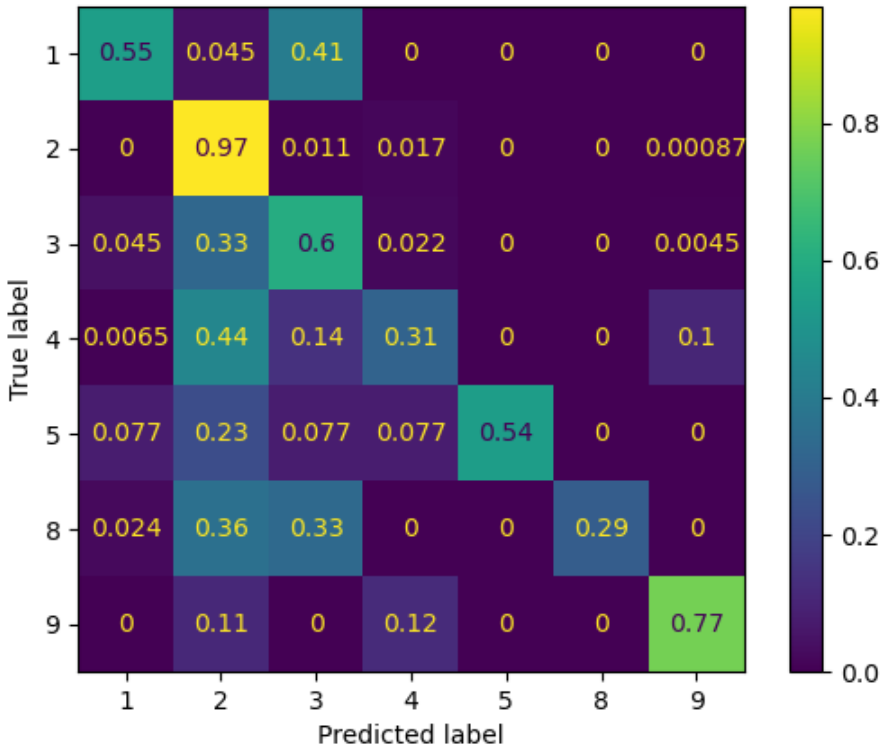


Figure 2: Confusion matrix for the model applied to test data

We notice that the model is most accurate on the 2-label (the forest-label), and the 9-label (the bareland-label) and least accurate on the 4-label (shrubland) and 8-label (artifical surface). We also notice that the model often predicts 2, instead of the correct label - this might be influenced by the strong overweight of 2-labels in the training data.

3 PAC-Bayes-Unexpected-Bernstein (40 Points) [Yevgeny]

3.1

Let $Z \leq 1$ ($Z \leq 1$ almost surely to be completely precise. We will omit this for the rest of the question) be a random variable, and let $\lambda \in \left[0, \frac{1}{2}\right]$. We then have,

$$\lambda Z \leq \frac{1}{2} \Leftrightarrow -\lambda Z \geq -\frac{1}{2}$$

We can thus use the given inequality $z - z^2 \leq \ln(1+z)$ with $z = -\lambda Z$, giving us that,

$$-\lambda Z - (-Z\lambda)^2 = -\lambda Z - Z^2\lambda^2 \leq \ln(1 + (-\lambda Z))$$

Since e^x is monotonously increasing, this gives us that,

$$\mathbb{E}\left[e^{-\lambda Z - Z^2\lambda^2}\right] \leq \mathbb{E}\left[e^{\ln(1+(-\lambda Z))}\right] = \mathbb{E}[1 + (-\lambda Z)] = 1 + (-\lambda \mathbb{E}[Z])$$

Where we have used linearity of expectation. We can now apply the inequality $1+z \leq e^z$ that holds for all z, on $z = -\lambda \mathbb{E}[Z]$. This gives us,

$$1 + (-\lambda \mathbb{E}(Z)) \leq e^{-\lambda \mathbb{E}[Z]}$$

Reading from the start, we have obtained that for any $\lambda \in \left[0, \frac{1}{2}\right]$, we have that,

$$\mathbb{E}\left[e^{-\lambda Z - Z^2\lambda^2}\right] \leq e^{-\lambda \mathbb{E}[Z]}$$

3.2

Notice that we can write,

$$e^{\lambda(\mathbb{E}[Z]-Z)-\lambda^2 Z^2} = e^{\lambda \mathbb{E}[Z] - \lambda Z - Z^2\lambda^2} = e^{\lambda \mathbb{E}[Z]} e^{-\lambda Z - Z^2\lambda^2}$$

We can thus apply the bound from 1. and lineraity of expectation (because $e^{\lambda \mathbb{E}[Z]}$ is constant), to obtain that for any $\lambda \in \left[0, \frac{1}{2}\right]$ we have that

$$\mathbb{E}\left[e^{\lambda(\mathbb{E}[Z]-Z)-\lambda^2 Z^2}\right] = e^{\lambda \mathbb{E}[Z]} \mathbb{E}\left[e^{-\lambda Z - Z^2\lambda^2}\right] \leq e^{\lambda \mathbb{E}[Z]} e^{-\lambda \mathbb{E}[Z]} = 1$$

3.3

Let $Z_1, \cdots, Z_n$ be independent and upper bounded by 1. Notice that we can write,

$$e^{\lambda \sum_{i=1}^n (\mathbb{E}[Z_i]-Z_i) - \lambda^2 \sum_{i=1}^n Z_i^2} = e^{\sum_{i=1}^n \left(\lambda(\mathbb{E}[Z_i]-Z_i) - \lambda^2 Z_i^2\right)} = \prod_{i=1}^n e^{\lambda(\mathbb{E}[Z_i]-Z_i)-\lambda^2 Z_i^2}$$

Since the Z_i's are independent, we also have that the random variables

$$Z_i' = e^{\lambda(\mathbb{E}[Z_i]-Z_i)-\lambda^2 Z_i^2}$$

are independent. We can then pull the expectation inside the product such that,

$$\mathbb{E}\left[e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2}\right] = \mathbb{E}\left[\prod_{i=1}^n e^{\lambda(\mathbb{E}[Z_i]-Z_i)-\lambda^2 Z_i^2}\right] = \prod_{i=1}^n \mathbb{E}\left[e^{\lambda(\mathbb{E}[Z_i]-Z_i)-\lambda^2 Z_i^2}\right]$$

Since each $Z_i \leq 1$, we can apply the inequality from 2., to obtain that for any $\lambda \in \left[0, \frac{1}{2}\right]$ we have that,

$$\mathbb{E}\left[e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2}\right] = \prod_{i=1}^n \mathbb{E}\left[e^{\lambda(\mathbb{E}[Z_i]-Z_i)-\lambda^2 Z_i^2}\right] \leq \prod_{i=1}^n 1 = 1$$

3.4

By multiplying with λn and collecting all terms including Z_i on the LHS in the probability, by linearity of expectation we have that,

$$\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Z_i\right] \geq \frac{1}{n}\sum_{i=1}^n Z_i + \frac{\lambda}{n}\sum_{i=1}^n Z_i^2 + \frac{\ln\left(\frac{1}{\delta}\right)}{\lambda n}\right) = \mathbb{P}\left(\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i) - \lambda^2\sum_{i=1}^n Z_i^2 \geq \ln\left(\frac{1}{\delta}\right)\right)$$

Since the exponential function is monotonously increasing we can apply it to both sides of the above inequality, to obtain that

$$\mathbb{P}\left(\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i) - \lambda^2\sum_{i=1}^n Z_i^2 \geq \ln\left(\frac{1}{\delta}\right)\right) = \mathbb{P}\left(e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2} \geq \frac{1}{\delta}\right)$$

We have that,

$$e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2} \geq 0$$

And we can therefore apply Markov's inequality and the bound from 4., to obtain that for any $\lambda \in \left(0, \frac{1}{2}\right]$ and $\delta > 0$ we have that

$$\begin{aligned}
\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Z_i\right] \geq \frac{1}{n}\sum_{i=1}^n Z_i + \frac{\lambda}{n}\sum_{i=1}^n Z_i^2 + \frac{\ln\left(\frac{1}{\delta}\right)}{\lambda n}\right) &= \mathbb{P}\left(e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2} \geq \frac{1}{\delta}\right) \\
&\leq \frac{\mathbb{E}\left[e^{\lambda\sum_{i=1}^n(\mathbb{E}[Z_i]-Z_i)-\lambda^2\sum_{i=1}^n Z_i^2}\right]}{\frac{1}{\delta}} \\
&\leq \frac{1}{\frac{1}{\delta}} \\
&= \delta
\end{aligned}$$

3.5

Let $\Lambda = \{\lambda_1, \cdots, \lambda_k\}$ be a grid of k values of λ, such that $\lambda_i \in \left(0, \frac{1}{2}\right]$ for all $i \in \{1, \cdots, k\}$. We want to prove

$$\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \min_{\lambda\in\Lambda}\left(\frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)\right) \leq \delta$$

If we can prove that

$$\mathbb{P}\left(\exists\lambda \in \Lambda : \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right) \leq \delta$$

We are done, because the event that the minimizing λ makes the inequality inside the above probability true, is contained in the event that there exists a λ such that the above inequality inside the probability holds. In other words, we have that,

$$\begin{aligned}
&\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \min_{\lambda\in\Lambda}\left(\frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)\right) \\
\leq &\mathbb{P}\left(\exists\lambda \in \Lambda : \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)
\end{aligned}$$

By taking a union bound over all λ in the grid, we have that,

$$\begin{aligned}
&\mathbb{P}\left(\exists\lambda \in \Lambda : \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right) \\
\leq &\sum_{\lambda\in\Lambda}\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)
\end{aligned}$$

We can modify the inequality obtained in 4. by replacing the 1 inside the logarithm with k and reapplying Markov's inequality to get that

$$\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right) \leq \frac{\delta}{k}$$

holds for any individual $\lambda \in \Lambda$. Since $|\Lambda| = k$ we then have that

$$\sum_{\lambda\in\Lambda} \mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right) \leq \sum_{\lambda\in\Lambda} \frac{\delta}{k} = \sum_{i=1}^{k} \frac{\delta}{k} = \delta$$

As argued in the start, we thus have that,

$$\mathbb{P}\left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Z_i\right] \geq \frac{1}{n}\sum_{i=1}^{n} Z_i + \min_{\lambda\in\Lambda}\left(\frac{\lambda}{n}\sum_{i=1}^{n} Z_i^2 + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)\right) \leq \delta$$

3.6

We compare the kl and unexpected Bernstein inequality. We compare them using the random variable Z with probability distribution $p_0 = \mathbb{P}(Z = 0)$, $p_{\frac{1}{2}} = \mathbb{P}\left(Z = \frac{1}{2}\right)$ and $p_1 = \mathbb{P}(Z = 1)$ such that $p_0 = p_1 = (1 - p_{\frac{1}{2}})/2$. This way $p_{\frac{1}{2}}$ determines the entire probability distributions. We consider the values in $P_{\frac{1}{2}} = \{0, 0.01, 0.02, \cdots 0.98, 0.99, 1\}$ for $p_{\frac{1}{2}}$. For each of these we draw $n = 100$ samples $Z_1, \cdots, Z_n$ from the distribution of Z and compute $\hat{p}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i$ and $\hat{v}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i^2$. We then compute the kl-bound on $p - \hat{p}_n$ given by $\mathrm{kl}^{-1^+}\left(\hat{p}_n, \frac{\ln\left(\frac{n+1}{\delta}\right)}{n}\right) - \hat{p}_n$ and the unexpected Bernstein Bound on $p - \hat{p}_n$ given by $\min_{\lambda\in\Lambda}\left(\lambda\hat{v}_n + \frac{\ln\left(\frac{k}{\delta}\right)}{\lambda n}\right)$. We take $\delta = 0.05$, $k = |\Lambda| = \lceil \log_2(\sqrt{n/\ln(1/\delta)}/2)\rceil = 2$ and $\Lambda = \{\frac{1}{2}, \frac{1}{2^2}, \cdots, \frac{1}{2^k}\} = \{\frac{1}{2}, \frac{1}{4}\}$. We plot the bounds below

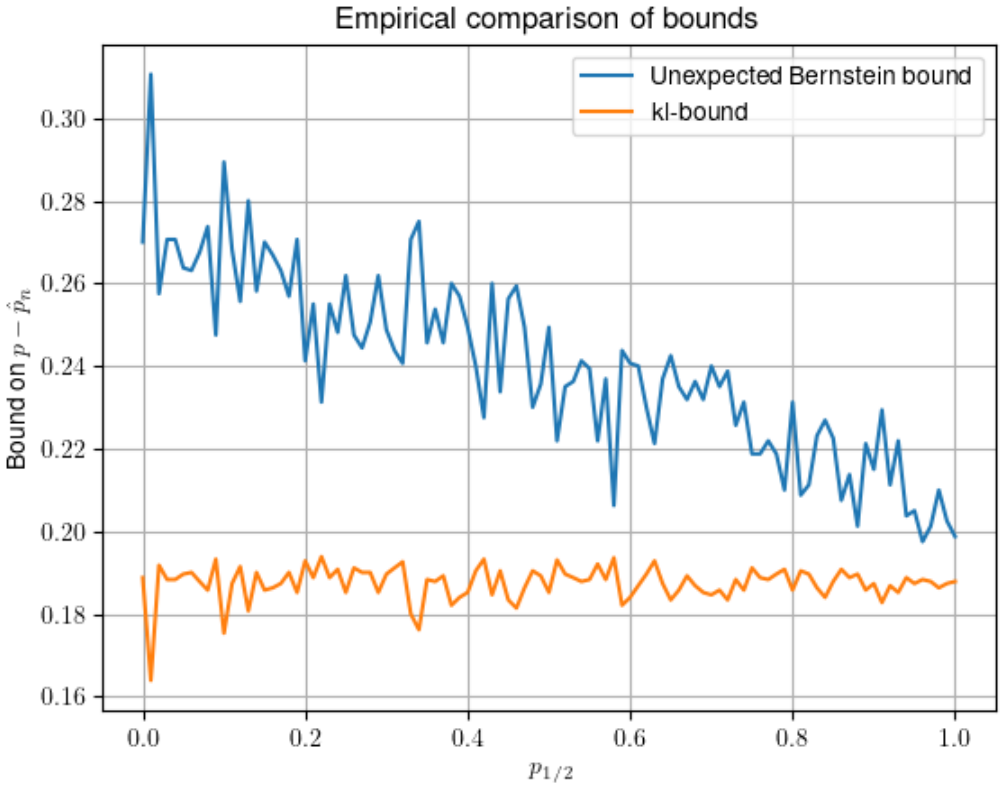


Figure 3: Emperical comparison of kl and Bernstein bounds as a function of $p_{\frac{1}{2}}$

We see a downward trend of the Bernstein bound whereas the kl-bound seems to hover around the same value. Bouth bounds are quite jagged. We resolve this by also plotting a smoothed out version of the plot. That is, we repeat the above experiment $N = 1000$ times, and plot the means of the two bounds as a function of $p_{\frac{1}{2}}$.

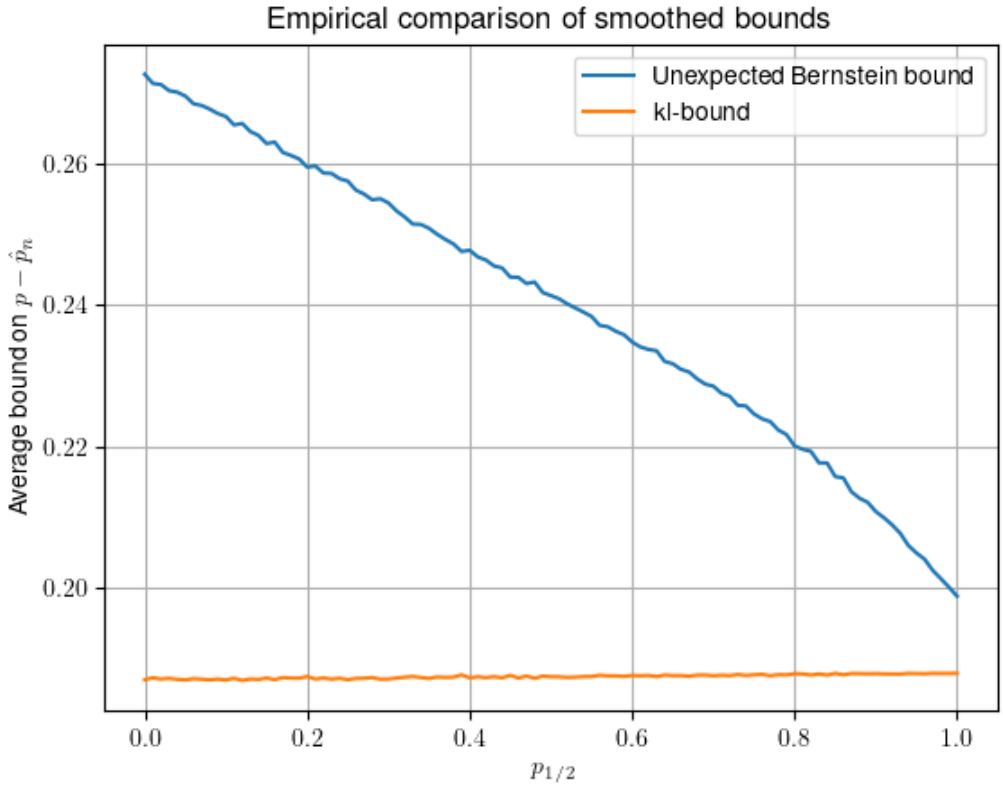


Figure 4: Emperical comparison of means of kl and Bernstein bounds as a function of $p_{\frac{1}{2}}$

Here, we see the clear trend that the unexpected Bernstein bound is decreasing in $p_{\frac{1}{2}}$, whereas the kl-bound is constant in $p_{\frac{1}{2}}$. This is of course clear, since the kl-bound only depends on $\hat{p}_n$, or the mean of the $Z_i's$ and as given by the assigment text, in our construction, we have $\mathbb{E}[Z] = \frac{1}{2}$ for any choice of $p_{\frac{1}{2}}$, such that we always have that $\mathbb{E}[\hat{p}_n] = \frac{1}{2}$. We also see that in this case, the unexpected Bernstein bound never beats the kl-bound (in expectation). To see the reason for this, consider that the (expectation of the) unexpected Bernstein bound is decreasing in $p_{\frac{1}{2}}$. This is due to the fact that the Bernstein bound is decreasing in the second moment of Z, and the second moment of Z is decreasing in $p_{\frac{1}{2}}$ as can be seen by,

$$\mathbb{E}[Z^2] = \mathbb{E}[Z^2] = \frac{1}{4}p_{\frac{1}{2}} + p_1 = \frac{1}{4}p_{\frac{1}{2}} + \frac{1 - p_{\frac{1}{2}}}{2} = \frac{1}{2} - \frac{1}{4}p_{\frac{1}{2}}$$

Which explains why the Bernstein bound becomes tighter as $p_{\frac{1}{2}}$ increases. This also explains why the Bernstein bound in expectation can not outperform the kl-bound in

this example. We know that the Bernstein bound (in expectation) is minimized when $p_{\frac{1}{2}} = 1$. Then $\hat{v}_n = \frac{1}{4}$ almost surely. With our values of Λ, δ and n we then have that the Bernstein bound is almost surely equal to

$$\min\left\{0.25 \cdot 0.25 + \frac{\ln \frac{2}{0.05}}{0.25 \cdot 100}, 0.5 \cdot 0.25 + \frac{\ln \frac{2}{0.05}}{0.5 \cdot 100}\right\} = 0.19877759$$

Which is greater than the average value that the kl-bound is close to in figure 4. That is, the Bernstein bound does not outperform the kl-bound in this example, because the second moment of Z, and therefore the variance of Z, is too large.

3.7

Let S be an i.i.d. sample, h a prediction rule independent of S and $\ell(y', y)$ a loss function upper bounded by 1. $\hat{L}(h, S) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(X_i), Y)$, $L(h) = \mathbb{E}[\ell(h(X), Y)]$, where (X, Y) are new samples from the distribution that generates S. and $\hat{V} = \frac{1}{n}\sum_{i=1}^{n} \ell(h(X_i), Y)^2$.We have that,

$$\begin{aligned} n\left(\lambda\left(L(h) - \hat{L}(h,S)\right) - \lambda^2 \hat{V}(h,S)\right) &= \lambda \sum_{i=1}^{n} \left(\mathbb{E}[\ell(h(X), Y)] - \ell(h(X_i), Y_i)\right) - \lambda^2 \sum_{i=1}^{n} \ell(h(X_i), Y_i)^2 \\ &= \lambda \sum_{i=1}^{n} \left(\mathbb{E}[\ell(h(X_i), Y_i)] - \ell(h(X_i), Y_i)\right) - \lambda^2 \sum_{i=1}^{n} \ell(h(X_i), Y_i)^2 \end{aligned}$$

Where we have used that (X, Y) come from the same distribution as (X_i, Y_i) for all i - which is a standard assumption in MLB. Further, as the samples in S are i.i.d. the random variables,

$$G_i = \ell(h(X_i), Y_i)$$

are also independent. As the loss is upper bounded by 1, G_i is upper bounded by 1. We can thus apply the bound from 3. with $Z_i = G_i$ to obtain that for $\lambda \in \left[0, \frac{1}{2}\right]$,

$$\mathbb{E}\left[e^{n\left(\lambda\left(L(h) - \hat{L}(h,S)\right) - \lambda^2 \hat{V}(h,S)\right)}\right] = \mathbb{E}\left[e^{\lambda \sum_{i=1}^{n} (\mathbb{E}[G_i] - G_i) - \lambda^2 \sum_{i=1}^{n} G_i^2}\right] \leq 1$$

3.8

Now, let $\mathcal{H}$ be a set of prediction rules and let π be a probability distribution over $\mathcal{H}$, such that π is independent of S. We want to show that for any $\lambda \in \left(0, \frac{1}{2}\right]$, we have that

$$\mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{1}{\delta}\right)}{n\lambda}\right) \leq \delta$$

Using the linearity of expectation and taking $f(h,S) = n\left(\lambda\left(L(h) - \hat{L}(h,S)\right) - \lambda^2\hat{V}(h,S)\right)$ we can write the above probability as

$$\mathbb{P}\left(\exists\rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda\mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{1}{\delta}\right)}{n\lambda}\right)$$

$$=\mathbb{P}\left(\exists\rho : \mathbb{E}_\rho[f(h,S)] \geq \mathrm{KL}(\rho||\pi) + \ln\left(\frac{1}{\delta}\right)\right)$$

We proceed as in the proof of theorem 3.28. We add the assumption that f is measurable[1]. Now, applying the change of measure inequality, theorem 3.27, we have that,

$$\begin{aligned}\mathbb{P}\left(\exists\rho : \mathbb{E}_\rho[f(h,S)] \geq \mathrm{KL}(\rho||\pi) + \ln\left(\frac{1}{\delta}\right)\right) &\leq \mathbb{P}\left(\exists\rho : \mathrm{KL}(\rho||\pi) + \ln\left(\mathbb{E}_\pi\left[e^{f(h,S)}\right]\right) \geq \mathrm{KL}(\rho||\pi) + \ln\left(\frac{1}{\delta}\right)\right)\\ &= \mathbb{P}\left(\exists\rho : \mathbb{E}_\pi\left[e^{f(h,S)}\right] \geq \frac{1}{\delta}\right)\\ &= \mathbb{P}\left(\mathbb{E}_\pi\left[e^{f(h,S)}\right] \geq \frac{1}{\delta}\right)\end{aligned}$$

Where the first equality comes from the fact that e^x is a monotonously increasing function, so taking it on both sides does not change the inequality, and the second equality holds because the expression $\mathbb{E}_\pi\left[e^{f(h,S)}\right] \geq \frac{1}{\delta}$ does not depend on ρ. We have that $\mathbb{E}_\pi\left[e^{f(h,S)}\right]$ is a positive random variable with respect to the sample S, and we can apply Markovs inequality to obtain,

$$\mathbb{P}\left(\mathbb{E}_\pi\left[e^{f(h,S)}\right] \geq \frac{1}{\delta}\right) \leq \frac{\mathbb{E}_S\left[\mathbb{E}_\pi\left[e^{f(h,S)}\right]\right]}{\frac{1}{\delta}} = \delta\mathbb{E}_S\left[\mathbb{E}_\pi\left[e^{f(h,S)}\right]\right]$$

Because π is independent of S we can swap the expectation order, and apply the bound from 7.(notice that the expectation in 7. is with respect to the randomness of S) to see that for $\lambda \in \left(0, \frac{1}{2}\right]$.

$$\delta\mathbb{E}_S\left[\mathbb{E}_\pi\left[e^{f(h,S)}\right]\right] = \delta\mathbb{E}_\pi\left[\mathbb{E}_S\left[e^{n\left(\lambda\left(L(h)-\hat{L}(h,S)\right)-\lambda^2\hat{V}(h,S)\right)}\right]\right] \leq \delta\mathbb{E}_\pi[1] = \delta$$

Which is what we wanted to show.

[1]This of course requires an argument, but seeing as we haven't discussed σ-algebras and measurability during the course, and considering that the lecture notes use this assumption implicitly without mentioning it through section 3.8, adding this assumption is considered unproblematic.

3.9

Let $\Lambda = \{\lambda_1, \cdots, \lambda_k\}$ be a grid of k values as in 5. We want to show that,

$$\mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \min_{\lambda\in\Lambda}\left(\lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda}\right)\right) \leq \delta$$

We proceed exactly as in 5. We have the bound,

$$\begin{aligned}
&\mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \min_{\lambda\in\Lambda}\left(\lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda}\right)\right) \\
&\leq \mathbb{P}\left(\exists \lambda \in \Lambda \exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda}\right)
\end{aligned}$$

Taking a union bound over all λ's, we have that,

$$\begin{aligned}
&\mathbb{P}\left(\exists \lambda \in \Lambda \exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda}\right) \\
&\leq \sum_{i=1}^{k} \mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda_i \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda_i}\right)
\end{aligned}$$

For each individual $\lambda_i \in (0, \frac{1}{2}]$ we can apply the bound from 8. (with the slight modification that 1 is replaced with k, but all else equal as in 5.) such that,

$$\sum_{i=1}^{k} \mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \lambda_i \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda_i}\right) \leq \sum_{i=1}^{k} \frac{\delta}{k} = \delta$$

Reading from start to finish we have that,

$$\mathbb{P}\left(\exists \rho : \mathbb{E}_\rho\left[L(h)\right] \geq \mathbb{E}_\rho\left[\hat{L}(h,s)\right] + \min_{\lambda\in\Lambda}\left(\lambda \mathbb{E}_\rho\left[\hat{V}(h,s)\right] + \frac{\mathrm{KL}(\rho||\pi) + \ln\left(\frac{k}{\delta}\right)}{n\lambda}\right)\right) \leq \delta$$

4 Dimensionality Reduction (20 points) [Sadegh]

Part (i)

We load in the data and plot it as a three-dimensional scatterplot. We assign different colors to the first half and last half of the dataset (i.e. 2000 points in each). The code for doing this is provided below,

```
import numpy as np
import matplotlib.pyplot as plt
data= np.loadtxt("dataset-vis-exam2022.txt", delimiter=",") # load in data
# Regular 3d Plot

#Split into first and last part of dataset
X1 = data[:2000,0]
Y1 = data[:2000,1]
Z1 = data[:2000,2]

X2 = data[2000:,0]
Y2 = data[2000:,1]
Z2 = data[2000:,2]


fig = plt.figure()
# Plot in 3d-space
ax = fig.add_subplot(projection='3d')
ax.scatter(X1, Y1, Z1, s = 3)
ax.scatter(X2, Y2, Z2, s = 3)

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.showfig()
```

This results in the following three-dimensional scatterplot

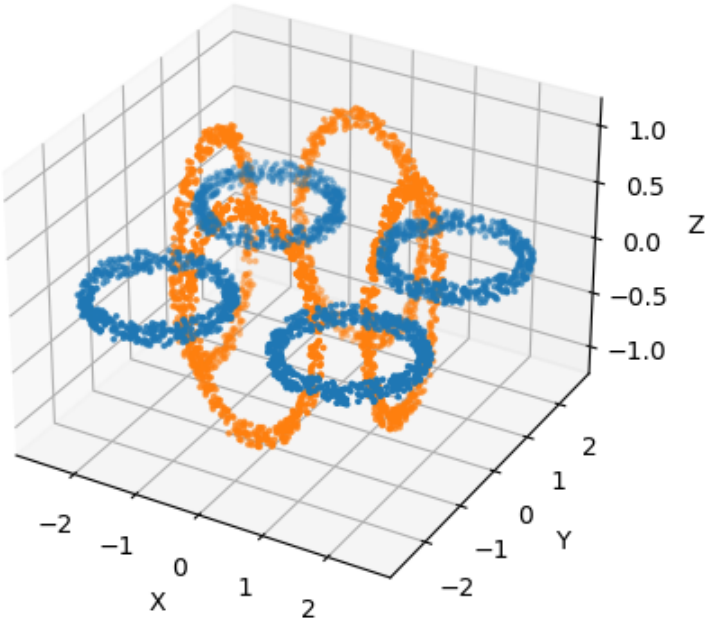


Figure 5: Original data as 3-d scatterplot

Note that there is no inherent meaning in the axis-labels. They are simply added for convinience when discussing the plot.

Part (ii)

We do PCA first on the unnormalized data. We utilize the `PCA` class from `sklearn.decompostion`. We set `whiten=False` and use `svd_solver="full"`. As we are producing two-dimensional maps of the data we set `n_components=2` and keep the rest of the parameters at their default values. We do PCA on the data by applying the `fit` method to data. Afterwards we project our data onto the two principal components by using the `transform` method. A code snippet is provided below

```
from sklearn.decomposition import PCA # Import PCA
#PCA on non-standardized data
pca = PCA(whiten=False, svd_solver = "full",n_components = 2) #Intialize PCA
pca.fit(data) #Fit PCA to data
```

```
pca_projection = pca.transform(data) # Project data
# onto two principal components
```

We plot the projected data below, again using different colors for the first and last half of the dataset,

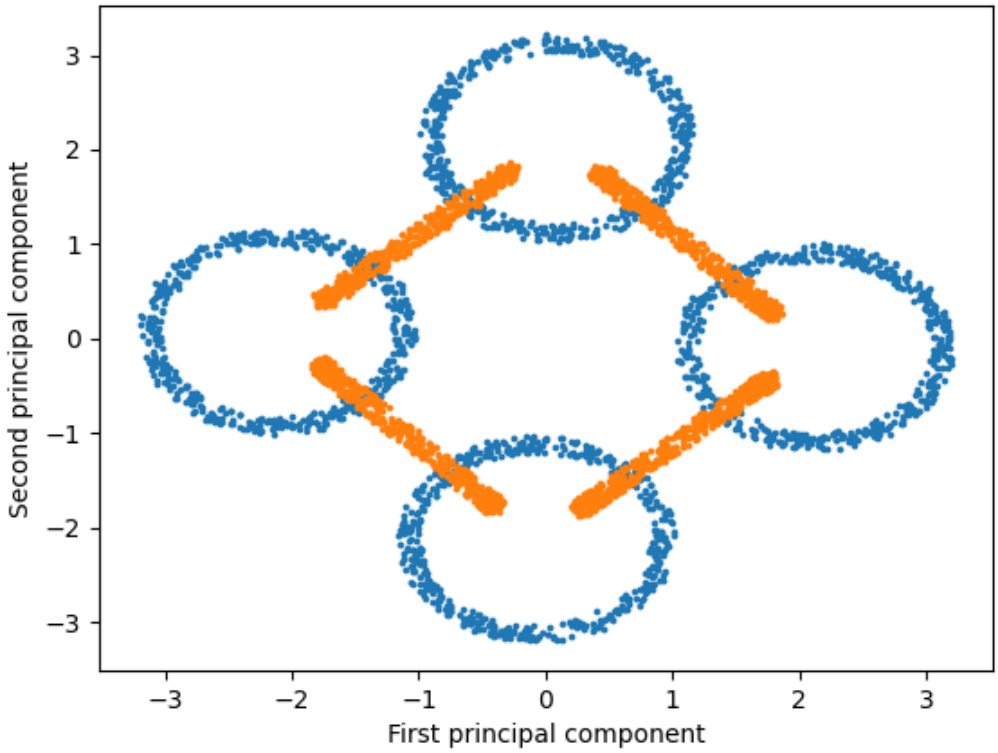


Figure 6: PCA on unnormalized data

We now normalize the data, such that each feature has zero mean and unit variance. The map f that normalizes a matrix X in this way is given by

$$f(X) = XD - \mu D$$

Where D is the diagonal matrix containing the (non Bessel corrected) standard deviation of each feature in X along the diagonal, and μ is the matrix with as many rows as X containing the means of each feature as entries in each row. The code for implementing and applying this to our data is provided below

```
# Compute mean of each feature
mu = np.mean(data, axis=0)
# Compute standard deviation of each feature
sd = np.std(data, axis=0, ddof = 0)
# Function to normalize array given mean and standard deviation
def normalize(X,mean,sd):
     D = np.diag(1/sd) # diagonalize std deviation vector
     return X @ D - mu @ D
data_norm = normalize(data, mu, sd) # Compute normalized data
```

We then do the PCA using our normalized data, exactly as we did with the unnormalized data, this time using our normalized data, `data_norm`, as we did the unnormalized data, `data`, before. We use the same parameters. We plot the projected data below

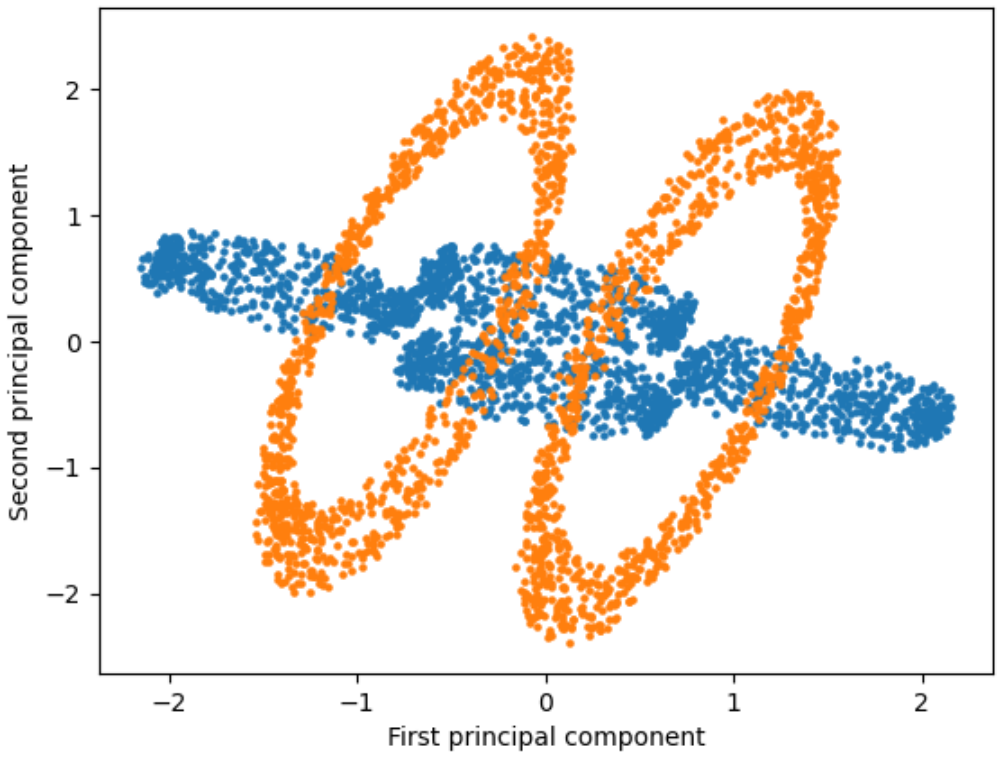


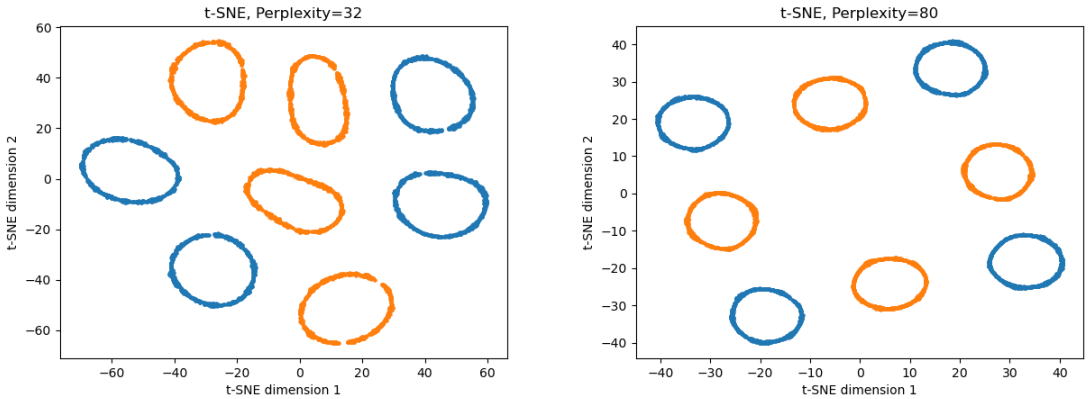
Figure 7: PCA on normalized data

Part (iii)

We do t-SNE with random initialisation using perplexities 32, 80, 180 and 280. For the implementation of t-SNE we utilize the `TSNE` class from `sklearn.manifold`. We set `init="random"` to do random initialisation, and set `perplexity` equal to 32, 80, 180 and 280, respectively. We also set `random_state=120` for reproducibility. We keep the rest of the parameters at their default values. Notably this means that we use the Barnes-Hut approximation of the gradient calculation when doing gradient descent. This is of course not as exact as computing the exact gradient, but a tolerable approximation that makes running the code substantially faster. We map our data transformed using t-SNE by applying the `fit_transform` method to our data. A code snippet for doing t-SNE for a perplexity of 32 is provided below.

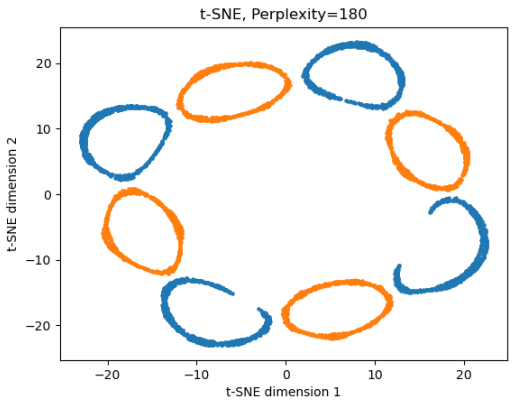
```
from sklearn.manifold import TSNE # import TSNE
perplexity = 32
tsne = TSNE(learning_rate="auto", init = "random",
perplexity=perplexity, random_state=120).fit_transform(data)
```

Below we plot the transformed data for each perplexity

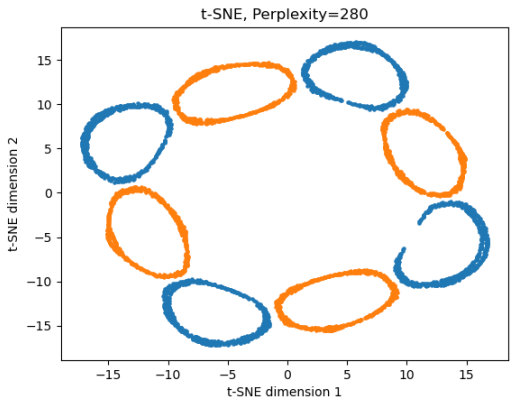


(a) Perplexity=32

(b) Perplexity=80



(c) Perplexity=180



(d) Perplexity=280

Figure 8: t-SNE with random initialisations with different perplexities

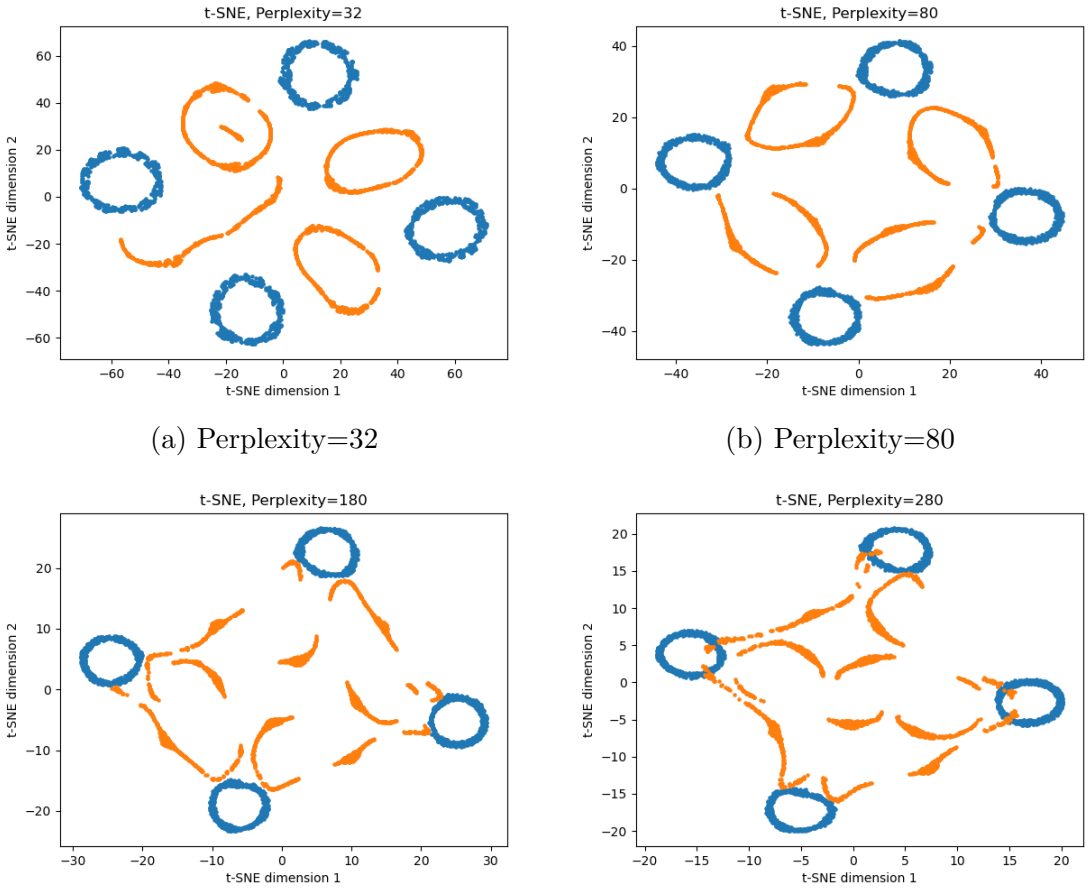
Part (iv)

We repeat what we did in (iii), this time on normalized data, but instead of using a random initialisation we use the transformed data points we obtained when using PCA on normalized data by setting `init` equal to the array containing our PCA-transformed points, i.e. we specify `init` to be the points in figure 7. A code snippet for doing this for a perplexity of 32 is provided below. Recall that `data_norm` is the normalized data obtained in (ii)

```
perplexity=32
#Intialize PCA
pca_norm = PCA(whiten=False, svd_solver = "full", n_components = 2)
pca_norm.fit(data_norm) #Fit PCA to normalized data
```

```
pca_projection_norm = pca_norm.transform(data_norm) # Transform data
#Transform data with t-SNE
tsne = TSNE(learning_rate="auto", init = pca_projection_norm,
perplexity=perplexity, random_state=120).fit_transform(data_norm)
```

Below we plot the results of this experiment for the same perplexities as in (iii).



(a) Perplexity=32

(b) Perplexity=80

(c) Perplexity=180

(d) Perplexity=280

Figure 9: t-SNE on normalized data with PCA initialisations with different perplexities

Part (v)

We compare the plots from (i)-(iv). We see from figure 5 that the data in 3-dimensional space looks like two sets[2] (corresponding to blue and orange points) of 4 two-dimensional circles each. Each circle from each set is intertwined with two circles from

[2]Note that the algorithms are oblivious to the notion of these 'sets'.

the other set.

(a) PCA vs t-SNE

We first notice the difference between doing PCA on the normalized and unnormalized data. PCA on the unnormalized data, looks like what one would see if one looked at figure 5 from above, i.e. with the Z-axis collapsed. The PCA-plot for the normalized data has preserved more of the 'geometric' structure of the original plot where one can still sense that the blue and orange sets form circles, whereas the orange set of points are reduced to lines in figure 6.

Nonetheless, both PCA-transformations are stil linear in nature, so they look like flattened and angled versions of the original data. This means that local similarities are not particularly well-preserved. In contrast, the t-SNE transformed data, has much more local structure. Looking at e.g. Figure 8b we clearly see that there are 4 circles of blue and orange points, respectively. However, the circles are disjoint so we do not get the same impression of intertwinedness, as we get in the PCA-plots. In this sense some global structure has been lost compared to the PCA-plots.

(b) Impact of perplexity

An interpretation of perplexity, is that it measures how many effective neighbors each point in the dataset has. The larger the perplexity, the more likely it is a point far away from a certain other point will be "picked" as a neighbor by the t-SNE algorithm [3]. We see this quite clearly in e.g. figure Figure 8, where for relatively low perplexity (32 and 80) the circles are further apart than they are for higher perplexity (180 and 280).

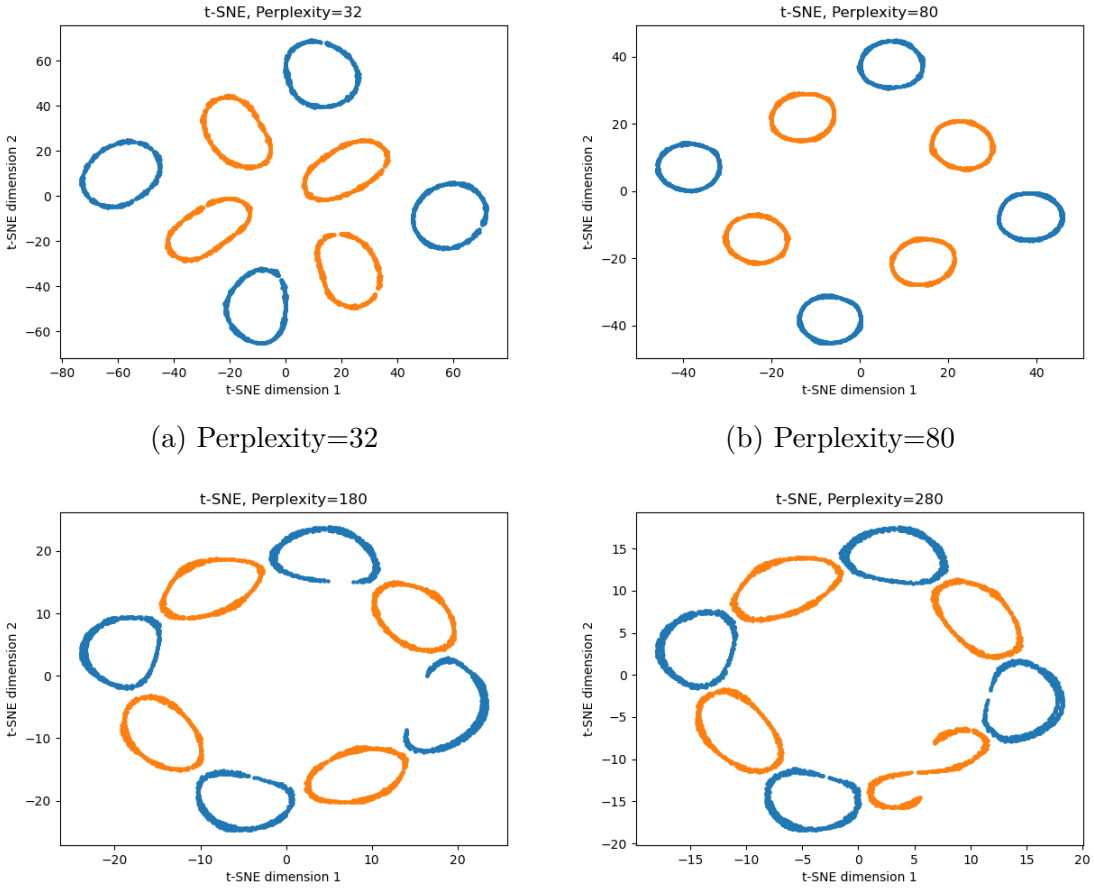
Picking a high perplexity can potentially distort the local structure, as points far away are considered as neighbors. One can see this beginning to happen in figure Figure 8c and Figure 8d where one of the blue circles is no longer "closed". On the other hand, a low perplexity (potentially combined with a bad initialisation) can result in the loss of some global structure. We see this in figure Figure 8a, where the different circles are in seemingly random locations of the plot, in contrast to the plots with higher perplexities, where every other circle is blue and every other circle is orange - which one might argue resembles the intertwinedness of the original data more closely.

(c) Initialisation

As discussed above, t-SNE on this particular dataset seems to do well at preserving the local structure of the dataset, especially for low perplexities when done on the unnormalised dataset. From figure Figure 9 we see that the local structure of especially the orange circles has been lost - for none of the chosen perplexities do we see 4

[3] Acording to Sadegh's slides on clustering, slide 20/35

orange circles. This indicates that either the normalisation or the PCA-initialisation, is causing the t-SNE map to lose local structure compared to the random initialisation on the original dataset. A suspicion one might have, is that when rescaling data to have unit variance in all dimensions some information could be lost in the t-SNE mapping. To explore this, and also discuss the impact of initialisation without simoultaneously varying whether the input data to the t-SNE algorithm is normalised or not, we also plot the output of the t-SNE algorithm when run on unormalised data, but with the same initialisation as in figure Figure 9.



(a) Perplexity=32

(b) Perplexity=80

(c) Perplexity=180

(d) Perplexity=280

Figure 10: t-SNE on unnormalized data with PCA initialisations with different perplexities

We see from figure Figure 8 and figure Figure 10, that the plots for perplexities 80, 180 and 280 are structurally pretty close no matter the initialisation. However, figure Figure 10a and Figure 8a are pretty different with regards to the global structure of the plots. As mentioned above, the circles are seemingly randomly scattered across

the plot in figure Figure 8a. On the other hand we see that the PCA-initialisation has preserved a lot of global structure in figure Figure 10a - here one can clearly see that "every other" circle is orange and every other circle is blue. It can even be seen that the orange circles point in towards each other, and the blue circles sit outside the orange, which could be perceived as a projection of the global intertwinedness structure seen in figure 5.

in part (a) we discussed that the PCA-visualisations had slightly more global structure than the t-SNE-visualiations. What the above comparison of figure Figure 10a and Figure 8a indicates, is that by initialising t-SNE with the global structure recovered by PCA, we get slight improvements in the t-SNE-algorithms ability to keep global structure of the original data for certain perplexities when compared to their randomly initialised counterparts. This is however only an indication - we would need to do the same experiment on different datasets to ultimately draw this conclusion. We also note from the above experiments, that one should be careful with normalising data before given it to the t-SNE