Intro to ML - HW4

Daniel Volkov, I.D: 330667494

August 12, 2024

Question 1

Firstly, notice that:

$$z = \begin{pmatrix} z_1 \\ \vdots \\ z_d \end{pmatrix} = X^t y$$

And so we can rewrite the optimization target:

$$\begin{aligned} \frac{1}{2} \cdot \|y - Xa\|_2^2 + \lambda \|a\|_1 &= \frac{1}{2} \cdot (y^t - a^t X^t)(y - Xa) + \lambda \|a\|_1 \\ &= \frac{1}{2} \cdot (y^t y - y^t Xa - a^t X^t y + a^t X^t Xa) + \lambda \|a\|_1 \end{aligned}$$

Now notice that $X^t X = \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_d)$, and thus $a^t X^t Xa = a^t \Sigma a = \sum_{j=1}^d \sigma_j a_j^2$.
Also notice that $y^t y$ is a constant when optimizing with respect to a, and thus we can throw it out of the problem. Meaning that:

$$\begin{aligned} \hat{a}^{\text{lasso}} &= \arg\min_{a \in \mathbb{R}^d} \left\{ \frac{1}{2} \cdot \|y - Xa\|_2^2 + \lambda \|a\|_1 \right\} \\ &= \arg\min_{a \in \mathbb{R}^d} \left\{ \frac{1}{2} \cdot (y^t y - y^t Xa - a^t X^t y + a^t X^t Xa) + \lambda \|a\|_1 \right\} \\ &= \arg\min_{a \in \mathbb{R}^d} \left\{ \frac{1}{2} \cdot (-z^t a - a^t z + a^t \Sigma a) + \lambda \|a\|_1 \right\} \end{aligned}$$

Also notice that $a, z \in \mathbb{R}^d$, and thus $z^t a = \langle a, z \rangle = \langle z, a \rangle = a^t z$. Thus we get:

$$\begin{aligned} \hat{a}^{\text{lasso}} &= \cdots = \arg\min_{a \in \mathbb{R}^d} \left\{ \frac{1}{2} \cdot (-z^t a - a^t z + a^t \Sigma a) + \lambda \|a\|_1 \right\} \\ &= \arg\min_{a \in \mathbb{R}^d} \left\{ \frac{1}{2} \cdot (-2\langle a, z \rangle + a^t \Sigma a) + \lambda \|a\| \right\} \\ &= \arg\min_{a \in \mathbb{R}^d} \left\{ \sum_{j=1}^d -a_j z_j + \frac{1}{2} \cdot \sigma_j a_j^2 + \lambda |a_j| \right\} := \arg\min_{a \in \mathbb{R}^d} \{g(a)\} \end{aligned}$$

Now notice that the target function is convex, since it is a composition of an affine transformation, with a convex function ($\|\cdot\|_2^2$), which is then added to another convex function ($\lambda\|\cdot\|_1$, which is convex for every positive $\lambda \in \mathbb{R}_+$).

But notice that the target function is non-differentiable at points $a \in \mathbb{R}^d$ on the axes ($a_j = 0$ for some j). Because the function is convex, we can choose a sub-gradient which zeroes out at the axes, and check critical points as usual (which will only be critical for the minima, because the function is convex).
The target function's gradient is given by:

$$\frac{\partial g}{\partial a_j} = \begin{cases} a_j \sigma_j - z_j + \lambda & a_j > 0 \\ a_j \sigma_j - z_j - \lambda & a_j < 0 \end{cases}$$

And thus we can choose a sub-gradient at $a_j = 0$:

$$\frac{\partial g}{\partial a_j}\Big|_{a_j=0} := a_j\sigma_j - z_j$$

Setting the gradient (and sub-gradient) to zero yields:

$$\hat{a}_j^{\text{lasso}} = \begin{cases} \frac{z_j-\lambda}{\sigma_j} & \hat{a}_j^{\text{lasso}} > 0 \\ \frac{z_j}{\sigma_j} & \hat{a}_j^{\text{lasso}} = 0 \\ \frac{z_j+\lambda}{\sigma_j} & \hat{a}_j^{\text{lasso}} < 0 \end{cases}$$

Consider the following cases:

- If $\hat{a}_j^{\text{lasso}} > 0$:

$$\hat{a}_j^{\text{lasso}} = \frac{z_j - \lambda}{\sigma_j} \implies z_j = a_j\sigma_j + \lambda > 0$$
$$\implies \hat{a}_j^{\text{lasso}} = \frac{\text{sign}(z_j)}{\sigma_j} \cdot (|z_j| - \lambda)$$

- If $\hat{a}_j^{\text{lasso}} < 0$:

$$\hat{a}_j^{\text{lasso}} = \frac{z_j + \lambda}{\sigma_j} \implies z_j = a_j\sigma_j - \lambda < 0$$
$$\implies \hat{a}_j^{\text{lasso}} = \frac{\text{sign}(z_j)}{\sigma_j} \cdot (|z_j| - \lambda)$$

- If $\hat{a}_j^{\text{lasso}} = 0$:

$$\hat{a}_j^{\text{lasso}} = \frac{z_j}{\sigma_j} = 0 \implies z_j = 0$$
$$\implies \hat{a}_j^{\text{lasso}} = \frac{\text{sign}(z_j)}{\sigma_j} \cdot \max\{|z_j| - \lambda, 0\}$$

Notice that in the first two cases, we can change the $(|z_j| - \lambda)$ term to $\max\{|z_j| - \lambda, 0\}$, without changing the outcome, since $|z_j| - \lambda \geq 0$ in the first two cases.
This of course implies that in any case we can represent $\hat{a}_j^{\text{lasso}}$ with:

$$\hat{a}_j^{\text{lasso}} = \frac{\text{sign}(z_j)}{\sigma_j} \cdot \max\{|z_j| - \lambda, 0\}$$

■

Question 2

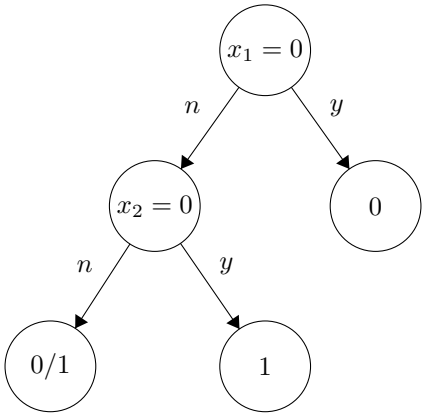
1. Using ID3, the first split (root node of the resulting tree) will be split with $x_1 = 0$, since the gain functions at the first split:

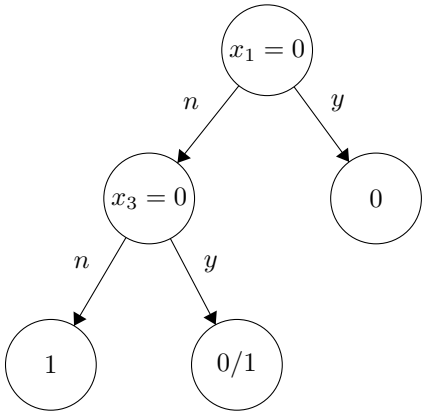
$$Gain(x_1) = C\left(\frac{1}{2}\right) - 0.34436$$
$$Gain(x_2) = C\left(\frac{1}{2}\right) - \frac{1}{2}$$
$$Gain(x_3) = C\left(\frac{1}{2}\right) - \frac{1}{2}$$

Then, the right node will result in a leaf, because only one sample has $x_1 = 0$.
At the second split, we get:

$$Gain(x_2) = C\left(\frac{2}{3}\right) - \frac{2}{3} \cdot \frac{1}{2}$$
$$Gain(x_3) = C\left(\frac{2}{3}\right) - \frac{2}{3} \cdot \frac{1}{2}$$

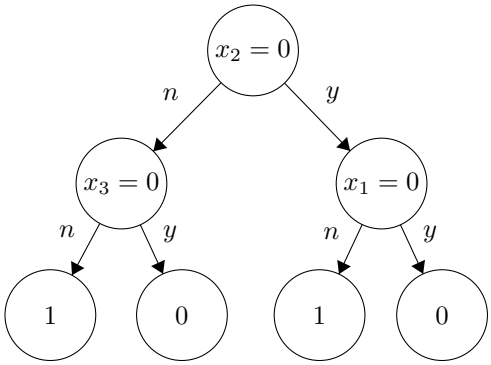
And since both options result in the same gain, ID3 will pick one arbitrarily.
In each case, one of the leaves at depth 2 will represent a subset of data with different labels, meaning that ID3 will again (in each option) pick an arbitrary label at that leaf.
To sum up, the four optional outputs of ID3 in this case will be:





Which all result in loss of exactly $\frac{1}{4}$.■

2. An optimal tree (with 0 training loss on the dataset) would be:



Question 3

Denote w^* as the optimal separator.
Firstly, I'll prove two lemmas:

- For all t: $w_{t+1} \cdot w^* \geq \gamma\sqrt{t}$
  Firstly, notice that (I consider the case $y_t = 1$, the case $y = -1$ is very similar):

  $$w_{t+1} \cdot w^* = \left(w_t + x_t \frac{1}{\sqrt{t}}\right) \cdot w^* = w_t \cdot w^* + \frac{1}{\sqrt{t}} x_t \cdot w^* \geq w_t \cdot w^* + \frac{\gamma}{\sqrt{t}}$$

  When the last inequality is from the definition of γ.

  This of course implies that:

  $$w_{t+1} \cdot w^* \geq w_t \cdot w^* + \frac{\gamma}{\sqrt{t}} \geq w_{t-1} \cdot w^* + \frac{\gamma}{\sqrt{t-1}} + \frac{\gamma}{\sqrt{t}} \geq \cdots \geq \gamma \sum_{i=1}^{t} \frac{1}{\sqrt{i}}$$

  Now, we can of course take the trivial bound:

  $$w_{t+1} \cdot w^* \geq \gamma \sum_{i=1}^{t} \frac{1}{\sqrt{i}} \geq \gamma \sum_{i=1}^{t} \frac{1}{\sqrt{t}} = \gamma \frac{t}{\sqrt{t}} = \gamma\sqrt{t}$$

  □

- For all t: $\|w_{t+1}\| \leq \sqrt{\sum_{i=1}^{t} \frac{1}{i}}$
  Firstly, notice that (I consider the case $y_t = 1$, the case $y = -1$ is very similar):

  $$\|w_{t+1}\|^2 = \|w_t + \frac{1}{\sqrt{t}} x_t\|^2 \leq \|w_t\|^2 + \frac{1}{t}\|x_t\|^2 = \|w_t\|^2 + \frac{1}{t}$$

  This implies that:

  $$\|w_{t+1}\|^2 \leq \|w_t\|^2 + \frac{1}{t} \leq \|w_{t-1}\|^2 + \frac{1}{t-1} + \frac{1}{t} \leq \cdots \leq \sum_{i=1}^{t} \frac{1}{i}$$

  And by taking the square root of both sides, we get the desired inequality. □

Now, from the Cauchy-Schwarz inequality: $w_{t+1} \cdot w^* \leq \|w_{t+1}\| \cdot \|w^*\| = \|w_{t+1}\|$. Thus:

$$\gamma\sqrt{M} \leq w_{M+1} \cdot w^* \leq \|w_{M+1}\| \leq \sqrt{\sum_{i=1}^{M} \frac{1}{i}}$$

$$\implies \gamma^2 \cdot M \leq \sum_{i=1}^{M} \frac{1}{i} \leq 1 + \ln M \implies M \leq \frac{1}{\gamma^2 - \frac{1}{M}} \cdot \ln M$$

And from the given hint:

$$M \leq \frac{2}{\gamma^2 - \frac{1}{M}} \cdot \ln\left(\frac{1}{\gamma^2 - \frac{1}{M}}\right) = \frac{-2}{\gamma^2 - \frac{1}{M}} \cdot \ln\left(\gamma^2 - \frac{1}{M}\right) \leq \frac{-2}{\gamma^2} \cdot \ln\left(\gamma^2 - \frac{1}{M}\right)$$

And if we assume $M \geq 1$ (i.e: the model is not initialized with w^*):

$$M \leq \frac{-2}{\gamma^2} \cdot \ln\left(\gamma^2 - \frac{1}{M}\right) \leq \frac{-2}{\gamma^2} \cdot \ln\left(\gamma^2 - 1\right) = \frac{4}{\gamma^2} \cdot \ln\left(\frac{1}{\sqrt{\gamma^2 - 1}}\right) \leq \frac{4}{\gamma^2} \cdot \ln\left(\frac{1}{\gamma - 1}\right) = O\left(\frac{4}{\gamma^2} \cdot \ln\frac{1}{\gamma}\right)$$

I did not find a way to get rid of the -1 term in the denominator inside the logarithm without making more assumptions on M. However, I did reach the desired asymptotic dependence on γ.

Question 4

Assume an initialization where the centroids: $(\mu_1^{(1)}, \ldots, \mu_k^{(1)}) = (\phi(x_{a_1}), \ldots, \phi(x_{a_k}))$.
Denote $S_j^t := \{i : \ \phi(x_i) \text{ is assigned to cluster j at time t}\}$.

The initial assignment of all datapoins can be easily done, since each centroid is located at sone datapoint, and we can use the kernel matrix to calculate the inner products:

$$\begin{aligned}
\text{Initial-Cluster}(\phi(x_i)) &= \arg\min_{1\le j\le k}\left\{\|\phi(x_i) - \mu_j^{(1)}\|_2^2\right\} = \arg\min_{1\le j\le k}\left\{\|\phi(x_i) - \phi(x_{a_j})\|_2^2\right\} \\
&= \arg\min_{1\le j\le k}\left\{\langle\phi(x_i),\phi(x_i)\rangle - 2\langle\phi(x_i),\phi(x_{a_j})\rangle + \langle\phi(x_{a_j}),\phi(x_{a_j})\rangle\right\} \\
&= \arg\min_{1\le j\le k}\left\{K_{i,i} - 2K_{i,a_j} + K_{a_j,a_j}\right\} = \arg\min_{1\le j\le k}\left\{K_{a_j,a_j} - 2K_{i,a_j}\right\}
\end{aligned}$$

And thus:

$$S_j^1 = \{i : \ \text{Initial-Cluster}(\phi(x_i)) = j\}$$

Since we don't know anything about the transofrmation $\phi(\cdot)$, we cannot really save μ_j^t at any time. Still, similarly to what we've seen in class:

$$\mu_j^{t+1} = \frac{1}{|S_j^t|}\sum_{i\in S_j^t}\phi(x_i)$$

And thus the next assignment step will be:

$$\begin{aligned}
\text{Cluster}(\phi(x_i))^{t+1} &= \arg\min_{1\le j\le k}\left\{\|\phi(x_i) - \mu_j^{t+1}\|_2^2\right\} = \arg\min_{1\le j\le k}\left\{\|\phi(x_i) - \frac{1}{|S_j^t|}\sum_{q\in S_j^t}\phi(x_q)\|_2^2\right\} \\
&= \arg\min_{1\le j\le k}\left\{\langle\phi(x_i),\phi(x_i)\rangle - \frac{2}{|S_j^t|}\sum_{q\in S_j^t}\langle\phi(x_i),\phi(x_q)\rangle + \frac{1}{|S_j^t|^2}\sum_{\ell\in S_j^t}\sum_{p\in S_j^t}\langle\phi(x_\ell),\phi(x_p)\rangle\right\} \\
&= \arg\min_{1\le j\le k}\left\{K_{i,i} - \frac{2}{|S_j^t|}\sum_{q\in S_j^t}K_{i,q} + \frac{1}{|S_j^t|^2}\sum_{\ell\in S_j^t}\sum_{p\in S_j^t}K_{\ell,p}\right\} \\
&= \arg\min_{1\le j\le k}\left\{\frac{1}{|S_j^t|}\cdot\left(\left[\frac{1}{|S_j^t|}\sum_{\ell,p\in S_j^t}K_{\ell,p}\right] - \left[2\sum_{q\in S_j^t}K_{i,q}\right]\right)\right\}
\end{aligned}$$

Note that the above rule decides the next cluster of a given datapoint, as a function of the previous clustering $(S_1^t, \ldots, S_k^t)$. This implies that we actually don't need to maintain the centroids at all (!), we should only maintain the sets $\{S_j\}_{j=1}^k$, and update them in each iteration of the above rule, like so:

$$S_j^{t+1} = \{i : \ \text{Cluster}(\phi(x_i))^{t+1} = j\}$$

Running this clustering rule iteratively until convergence, we get an identical behaviour to k-means, when using the kernel matrix alone, without ever calculating $\phi(x_i)$ and without ever saving the centroids.
The final clustering of the datapoints will be given by the sets $\{S_j^{t_{\text{final}}}\}_{j=1}^k$.

Question 5

1. Notice that the log likelihood:

$$\begin{aligned}\ell(\lambda) &= \sum_{i=1}^{n} \log(p(x_i;\lambda)) = \sum_{i=1}^{n} \log(\lambda e^{-\lambda x_i}) \\ &= \sum_{i=1}^{n} \log(\lambda) - \lambda x_i = n\log(\lambda) - \lambda \sum_{i=1}^{n} x_i\end{aligned}$$

And so the maximum likelihood estimator would be:

$$\arg\max_{\lambda\in\mathbb{R}} \left\{ n\log\lambda - \lambda \sum_{i=1}^{n} x_i \right\} := \arg\max_{\lambda\in\mathbb{R}} \{g(\lambda)\}$$

Consider the first and second order derivatives of g:

$$\begin{aligned}\frac{dg}{d\lambda} &= \frac{n}{\lambda} - \sum_{i=1}^{n} x_i \\ \frac{d^2g}{d\lambda^2} &= -\frac{n}{\lambda^2} < 0\end{aligned}$$

And since the second derivative is always negative, g is a concave function, which implies that if it's derivative zeroes out at some point, this is indeed an extrema of g, and it's the maximum of g.
The derivative of g zeroes out at $\lambda = \frac{n}{\sum_{i=1}^{n} x_i}$, and thus:

$$\lambda_{\text{MLE}} = \frac{n}{\sum_{i=1}^{n} x_i}$$

Which is the Maximum Likelihood Estimator.

2. As we've seen in class:

$$\mathbb{P}(\Lambda = \lambda | X_1 = x_1, \ldots, X_n = x_n) \propto \mathbb{P}(\Lambda = \lambda) \cdot \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | \Lambda = \lambda)$$

When the constant factor does not depend on λ.
Thus, using the given prior distribution, and the given $\mathbb{P}(X_i = x_i | \Lambda = \lambda) = \lambda e^{-\lambda x_i}$, we get:

$$\begin{aligned}
\lambda_{\text{MAP}} &= \arg\max_{\lambda \in \mathbb{R}} \{\mathbb{P}(\Lambda = \lambda | X_1 = x_1, \ldots, X_n = x_n)\} \\
&= \arg\max_{\lambda \in \mathbb{R}} \left\{ \mathbb{P}(\Lambda = \lambda) \cdot \prod_{i=1}^{n} \mathbb{P}(X_i = x_i | \Lambda = \lambda) \right\} \\
&= \arg\max_{\lambda \in \mathbb{R}} \left\{ e^{-\lambda} \prod_{i=1}^{n} \lambda e^{-\lambda x_i} \right\}
\end{aligned}$$

And since $\log(\cdot)$ is an increasing function, we can take the log of the target function, without changing the arg max:

$$\begin{aligned}
\lambda_{\text{MAP}} &= \arg\max_{\lambda \in \mathbb{R}} \left\{ e^{-\lambda} \prod_{i=1}^{n} \lambda e^{-\lambda x_i} \right\} = \arg\max_{\lambda \in \mathbb{R}} \left\{ -\lambda + \sum_{i=1}^{n} (\log(\lambda) - \lambda x_i) \right\} \\
&= \arg\max_{\lambda \in \mathbb{R}} \left\{ -\lambda + n \log(\lambda) - \lambda \sum_{i=1}^{n} x_i \right\} := \arg\max_{\lambda \in \mathbb{R}} \{g_2(\lambda)\}
\end{aligned}$$

Consider the first and second order derivatives of g_2:

$$\begin{aligned}
\frac{dg_2}{d\lambda} &= -1 + \frac{n}{\lambda} - \sum_{i=1}^{n} x_i \\
\frac{d^2 g_2}{d\lambda^2} &= -\frac{n}{\lambda^2} < 0
\end{aligned}$$

And again for the same reasons, the only point that zeroes the first derivative will be the maximum of g_2:

$$\lambda_{\text{MAP}} = \frac{n}{1 + \sum_{i=1}^{n} x_i}$$

Which is the Maximum A Posteriori value.

Programming assignment

(a)

Code submitted.
Note that I normalized the dataset in my implementation, meaning I made it so that the data's mean is $\bar{0}$.

(b)

I ran PCA on the pictures of George W Bush. The Principal components I got using PCA were:

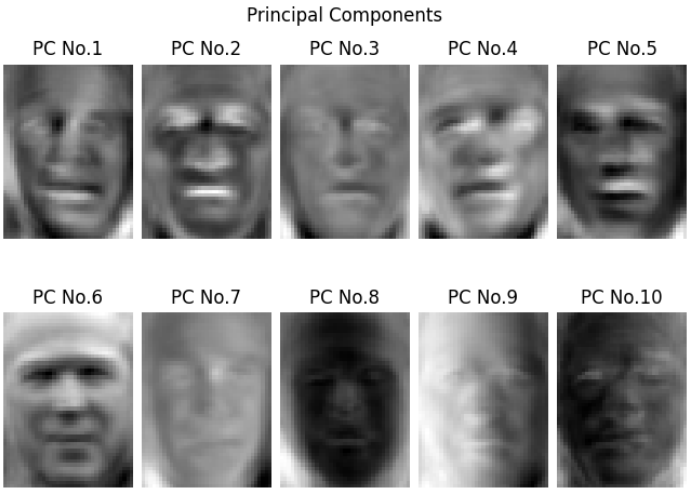


Figure 1: The 10 principal components of George W Bush's face, according to PCA

We can clearly see that the principal components are (somewhat horrifying) faces.
Some of the principal components we got strongly resemble a feature of George W Bush's face. For example, PC No.2 resembles the eyes and mouth, since those are the "lit up" areas (that got high values in the vector).
Some other principal components include more general aspects of the face, such as the forehead and the chin (PC No.6).

We can also observe that vectors like PC No.8 resemble the "background" of the image, which tends to change a lot.

(c)

The reconstructed images and the ℓ_2 reconstruction error I got were:

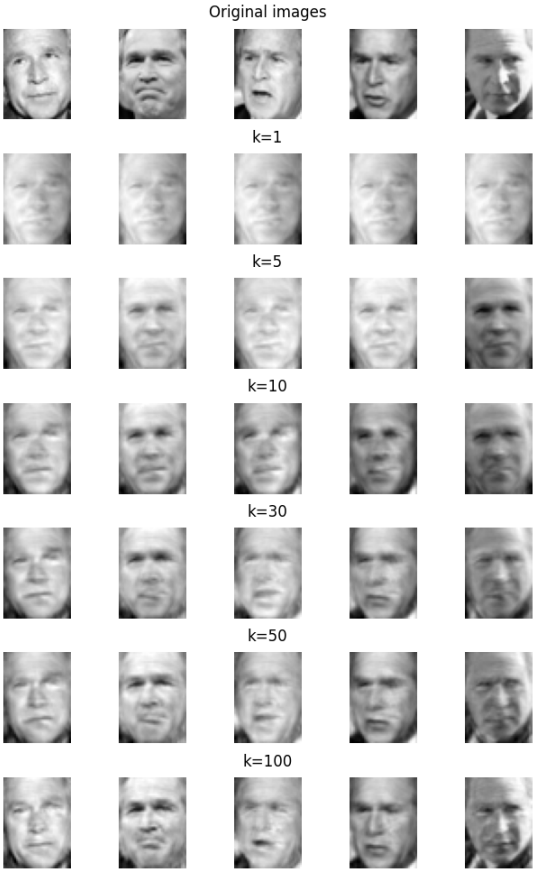


Figure 2: Image reconstruction after dimension reduction with PCA

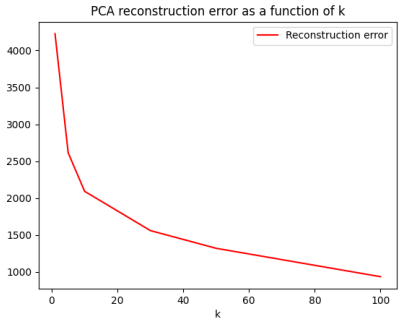


Figure 3: PCA ℓ_2 reconstruction error as a function of k

As we can observe, the reconstruction tends to get better with larger values of k.
This implies, intuitively, that larger values of k yield a more accurate representation of the data (which is taken from $\mathbb{R}^d$) in $\mathbb{R}^k$. This is what we would expect, since a larger dimension is more "expressive".