Deadline: Th. January 30th , 14:00 Drop your printed or legible handwritten submissions into the boxes at Samelsonplatz. Alternatively upload a `.pdf` file via LearnWeb. (e.g. exported Jupyter notebook)

1. Singular Value Decomposition (4 points)

Let X be a $N \times M$ matrix of rank K with SVD $X = U\Sigma V^\mathsf{T}$. Then $X^+ = V\Sigma^+ U^\mathsf{T}$ is called it's **Moore-Penrose pseudoinverse**, where

$$\Sigma = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_K & \\ \hline & 0 & & 0 \end{array}\right] \qquad \Sigma^+ = \left[\begin{array}{ccc|c} \sigma_1^{-1} & & & \\ & \ddots & & 0 \\ & & \sigma_K^{-1} & \\ \hline & 0 & & 0 \end{array}\right]$$

Note that Σ is $N \times M$ while Σ^+ is $M \times N$ with appropriate block sizes for the zero matrices.

A. [2p] Verify that the pseudoinverse satisfies $XX^+X = X$ and $X^+XX^+ = X^+$

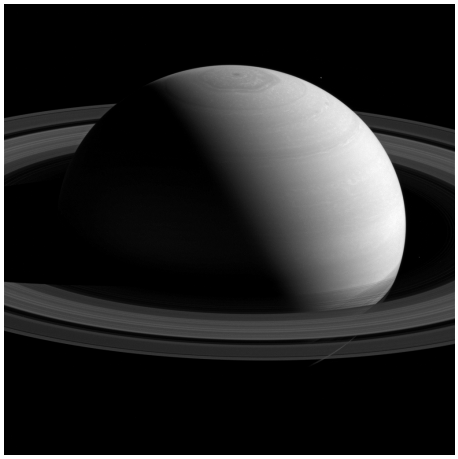
B. [2p] Show that $\hat\beta^{\text{OLS}} = X^+y$, i.e. $\beta = X^+y$ solves the normal equation $X^\mathsf{T}X\beta = X^\mathsf{T}y$ of linear regression, even if $X^\mathsf{T}X$ is not invertible.

2. SVD for Image Compression (8 points)

Recall the **Eckart-Young-Theorem**: The solution to the constrained minimization problem

$$\min \|X - \widetilde{X}\|_2^2 \quad \text{s.t.} \quad \operatorname{rank}(\widetilde{X}) \le k$$

is given by the truncated SVD of X, i.e. $\widetilde{X} = U_{1:k} \operatorname{diag}(\sigma_{1:k}) V_{1:k}^\mathsf{T}$. One possible application of this is **image compression**: Assume we are given a matrix with values between 0 and 255, representing a gray scale image. Then, instead of saving X, which requires $\mathcal{O}(N \cdot M)$ bits, we save $U_{1:k}, \sigma_{1:k}, V_{1:k}$ which requires only $\mathcal{O}(K(N + M + 1))$ bits of memory.



https://photojournal.jpl.nasa.gov/tiff/PIA18314.tif

A. Download the shown image of Saturn from NASA's repository. It's a 8-bit 1024x1024 gray scale image (≈ 1049KB). Load the image as a matrix in python using `matplotlib.pyplot.imread`.

For $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$, compute the truncated SVD, save the resulting matrices in 16-bit floating precision using `numpy.savez`. Finally, reload the saved arrays and restore the image as a 8-bit uint array. Plot the reconstructed images and compute the compression factor in each case. (Either compare file sizes manually or compute it.)

3. Principal Component Analysis (10 points)

As in Tutorial 8, load the IRIS dataset via `sklearn.datasets.load_iris` and construct a 3:1 training-test split via `sklearn.model_selection.train_test_split`; use 2020 as the random seed for part A and B.

A. [2p] Compute the principal components using **only the training data**. What is the transformation from the old features to the new features?

B. [2p] Make a plot of **the whole dataset**, using the first two principal components. (from part 3A)

C. [6p] For $k = 1, 2, 3$ train two linear classifiers (you can use `sklearn`'s `LogisticRegression`): one on the original data, and one on the embedded data. Compare their performance by computing the mean and standard deviation of the test accuracy over 1000 independent runs for each classifier.