DATA.STAT.770 Dimensionality Reduction and Visualization, Spring 2021, Exercise set 1

General information about exercises:

- All exercises of the course are graded in total based on total points: completing every exercise is not necessary to pass.
- Return deadlines and return areas for each exercise set are given in the course Moodle. Each exercise must be returned in the correct Moodle area.
- The exercises must be completed separately by each student (i.e. not in groups).
- The solutions should consist of a ZIP archive containing: 1) a PDF document detailing your answers including all numerical results and figures, 2) any programming code you created to solve the exercises; code of pre-existing libraries or toolboxes does not need to be included.
- The exercise solutions must document both the answers and the steps taken to reach them.
- Some of the exercises contain computer work, mostly relatively simple application of existing implementations and/or simple implementation tasks. In the case of programming exercises, you can use any programming language of your choice, such as R, SPSS, SAS, Matlab, Octave, Python, Java, or C. In some exercises, the methods we discuss have ready-made implementations typically in R, Python, and Matlab/Octave, making it easier to use those languages for those exercises. Note: If you have very little experience with software like Matlab, Python, or R and have trouble completing the exercises because of that, contact the lecturer - it might be possible to arrange alternative exercises.

Exercises begin on the next pages.

Part Pre: Preliminaries

Problem Pre1: Properties of High-dimensional Spaces

The curse of dimensionality is a term that denotes special properties of high-dimensional spaces, and the problems that high dimensionality causes for statistical estimation.

a) Lecture 1 said that in high enough dimensions, a hypersphere contains much much less volume than the hypercube that surrounds it; let's try this out. Generate **ten million data items** (data points) uniformly distributed into a d-dimensional space, so that for each data item each of the d coordinates is uniformly distributed between -1 and 1. For each data item, test whether it is inside the hypersphere: to do that, compute the total Euclidean distance from the origin, if it is 1 or less the point is inside the hypersphere, otherwise it is outside. What proportion of points is inside the hypersphere? Report the resulting proportion for dimensionalities $d = 2$, $d = 3$, $d = 5$, $d = 7$, $d = 10$, $d = 13$, and $d = 17$.

b) Consider a (hyper)spherical shell between radiuses 0.95 and 1: that is, consider all points whose distance from the origin is between 0.95 and 1. As before, generate ten million data items uniformly distributed into a d-dimensional space, and test how many are within the hypersphere of radius 1. Out of those points that are within the hypersphere, what proportion are inside the (hyper)spherical shell? Report the resulting proportion for dimensionalities $d = 2$, $d = 3$, $d = 5$, $d = 7$, $d = 10$, $d = 13$, and $d = 17$.

Problem Pre2: Curse of Dimensionality

Introduction: the K-nearest neighbor predictor. In this exercise and several later exercises, we use the *K-nearest neighbor predictor*, a simple non-parametric statistical predictor where the values of one or more output variables are predicted for data points based on their input features.

- In a 1-nearest neighbor predictor (often simply called a nearest neighbor predictor), for any data point, the output values are predicted to be the same as in the nearest neighbor whose output values are known. For a data point represented as a d-dimensional feature vector $\mathbf{x}_{\text{test}}$, the "nearest neighbor" is another data point $\mathbf{x}'$ that has the smallest Euclidean distance $||\mathbf{x}_{\text{test}} - \mathbf{x}'||$. The nearest neighbor prediction is the target value at the data point $\mathbf{x}'$.

- In the more general K-nearest neighbor predictor (K could be for example 5 or 10) the prediction is an average of several neighbors: to predict the target variable for a test point $\mathbf{x}_{\text{test}}$, find its K nearest neighbors from the training set: the K points $\mathbf{x}$ that have the smallest Euclidean distances from the test point, $||\mathbf{x} - \mathbf{x}_{\text{test}}||$. Take the average of their target values, and use that as the prediction for the test point.

Curse of Dimensionality. High dimensional data suffers from the curse of dimensionality where it is hard to distinguish actual statistical trends from artifacts of the finite set of samples. Let's test this. For each of the following dimensionalities, $d = 1$, $d = 2$, $d = 4$, $d = 7$, $d = 10$, $d = 15$, repeat the following.

- Generate two thousand data points where each data point $\mathbf{x} = [x_1, \ldots, x_d]$ is normally distributed around the origin with variance 1 in each dimension.

- Compute a target variable $y = x_1 + \sin(5x_1)$ for each data point. We know that the target variable only depends on the first coordinate, and it is a simple function with no noise, but a learning function learning from a data set would not know that, it would have to learn it. The more dimensions we have, the harder it is to tell which part of the data variation is related to the target variable.

- Split this data set into two parts: 1000 points for training and 1000 points for testing.

- Let's learn a 5-nearest-neighbor predictor. The *5-nearest neighbor predictor* is a simple nonparametric statistical predictor, where the values of one or more output variables are predicted for data points based on their input features.

- Plot the value of the first coordinate x_1 (horizontal) versus the target variable y (vertical) in the test set, and plot your predicted values (x_1 versus your predicted target value) on top of the same plot.

- Compute the mean-squared prediction error in the test set: the squared difference between your prediction and the actual target value, averaged over the 1000 test points.

Report for each dimensionality the plot of x_1 versus the target values and predictions and the resulting mean-squared prediction error for each dimensionality.