MTTTS17 Dimensionality Reduction and Visualization, Spring 2019, Exercise set 1A

The exercises must be completed separately by each student (i.e. not in groups). The exercise solutions must document both the answers and the steps taken to reach them. Completing all exercises is not necessary to pass. **The deadline to return this exercise set is April 21, 2018.**

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.

Exercise solutions must be returned by email to the lecturer. 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.

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 nonparametric 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.

Part B: Linear Dimensionality Reduction

Problem B1: Mathematics of Principal Component Analysis

Derive the solution for Principal Component Analysis (PCA) using maximization of variance in the projected space as an optimization criterion. That is, given data $X_{d\times N}$ with N samples in d dimensions, find projection matrix $W_{d\times k}$, $1 \leq k \leq d$, such that $\mathrm{Var}(Z)$ is maximized for projected data $Z = W^{\mathrm{T}}X$.

You may assume that X has zero mean. Derive the solution for the first principal component w_1 by maximizing $Var(w_1^{\mathrm{T}}X)$, and using the constraint that $w_1^{\mathrm{T}}w_1 = 1$.

Problem B2: Principal Component Analysis of Wines

The Wine Quality data set "winequality-red.txt" and "winequality-white.txt" provided in this archive is a data set which can be used to predict quality of white and red wine varieties based on their chemical characteristics. However, in this exercise we do not try to predict the wine quality but simply analyze the rest of the input variables. The data set comes from the UCI Machine Learning Repository and is available there at `http://archive.ics.uci.edu/ml/datasets/Wine+Quality`. Each wine is described by 12 feature values, in order: 1-fixed acidity, 2-volatile acidity, 3-citric acid, 4-residual sugar, 5-chlorides, 6-free sulfur dioxide, 7-total sulfur dioxide, 8-density, 9-pH, 10-sulphates, 11-alcohol, 12-quality.

- For the red wines, find the two first principal components, project the data onto them, and plot the data along them as a scatter plot. Compute the amount of variance explained by the two first components.
- Then repeat the same analysis for the white wines.
- Principal component analysis finds orthogonal projections (orthogonal axes) that contain the maximal variance. The original variables are also

orthogonal axes; if PCA was restricted to use only one variable per projection, it would reduce to variable ranking of the original variables based on the variance. Perform such variable ranking for the white wines, take the top-two variables, and compare the resulting picture to the "real" (unrestricted) PCA result.

Hint: in Matlab, consider the Matlab functions "cov" and "eig". R has corresponding functions. Try to plot the wines with low quality (quality value 5 or below), medium quality (quality value 6) and high quality (quality value 7 or above) with different colors.

Problem B3: Principal Component Analysis of an Image

The file boats.png contains a grayscale picture of Petersburg small boat harbor in Wrangell Narrows, Alaska. The image is 300 pixels wide and 230 pixels tall. Let's try using PCA to compress this image, just like was shown on the lecture.

A template code for this exercise, containing the parts unrelated to statistical analysis, is provided in this package in two programming languages, R (file image_compression_and_components_template.R) and Matlab (file image_compression_and_components_template.m).

1. Divide the image into 10 by 10 pixel blocks, so that the first block contains pixels in rows 1-10, columns 1-10, the second block contains pixels in rows 1-10, columns 11-20, and so on. Each block contains 100 pixel values and can be thought of as a 100-dimensional input vector, in total you have $30 \times 23 = 690$ such 100-dimensional input vectors.

2. Compute the first PCA component of this data set (690 items, 100 features) and project each input vectors onto that component - the result is one scalar value per input vector. Then project the scalar values back as a reconstruction of the origal features, as described on the lecture - the result is one 100-dimensional vector per input vector. Draw the resulting picture: for each pixel block, instead of the original pixel values, draw the approximated values.

3. Do the same for 2 components, 5, 10, 20, and 30 PCA components.

4. The PCA projection directions themselves are 100-dimensional vectors, and each element of the vectors corresponds to one pixel in the 10 by 10 image blocks. Therefore, each projection direction itself can be represented as a 10 by 10 block: it is like a filter for the image, showing which pixel values contribute positively to that principal component (high projection coefficients) and which pixel values contribute negatively (low projection coefficients). The provided template code also draws these filter images. Analyze the filters: have they identified important features of the image? Where do the features occur most strongly in the image?

Problem B4: Principal Component Analysis of an Audio File

Principal Component Analysis can be used to compress many kinds of data, not just images. The file the_entertainer.wav contains a part of "The Entertainer", a well-known 1902 piano rag by Scott Joplin (original audio available at `http://freemusicarchive.org/music/Scott_Joplin/Frog_Legs_Ragtime_Era_Favorites/04_-_scott_joplin_-_the_entertainer`). Just like an image can be divided into spatial blocks, audio waveforms can be divided into blocks over time; each block is a high-dimensional vector of numbers, and the resulting set of high-dimensional blocks can be reduced to a lower dimensionality by PCA.

The template code musiccompression_template.R (in R; Matlab equivalent will be provided soon) contains code to load the audio file and to transform it into a feature data set of blocks. Your task is to reduce the dimensionality of the feature data set to 15 principal components, and then reconstruct the original data from the principal components, just like was done for the image in problem B3. The template code will the a reconstructed audio waveform which can be plotted as a time series, and can also be played just like the original music. You do not need to play the file, but plot it, and save it to a file; provide it as part of your answer.

Note: if you choose to play any reconstructed audio, please do so at low volumes only! In case the audio waveform is not reconstructed well, it can contain clicks or scratches or other irritating or loud sounds which could harm your hearing if played overly loudly.

Problem B5: Independent Component Analysis for Separating Audio Mixtures

Consider a cocktail party situation where different microphones in the room record different mixtures of the ongoing independent audio sources in the room; the mixture that a microphone records is affected for example by how close the microphone is to each audio source.

a) Download and get to know a software package for independent component analysis. For example, use the FastICA software package; it is available at `http://research.ics.aalto.fi/ica/fastica/` for several programming languages such as Matlab, R, Python and C++.[1] You can also use any other ICA software package.

b) The exercise pack includes four WAV audio files 'musicmix2019_01.wav', 'musicmix2019_02.wav', 'musicmix2019_03.wav' and 'musicmix2019_04.wav'. They each are a different mixture of four independent audio sources (same

[1] If you use Octave, you should use the Matlab version of the FastICA package, not the older Octave version. You may need to replace the file 'fpica.m' in the FastICA package with the modified version provided in this exercise pack.

sources in each file, but mixed in a different way).[2]

Suppose that we are in a noisy environment where several different music pieces are being played at the same time. We want to separate the different pieces of music, for example to identify them. We will use independent component analysis (ICA) for this purpose.

Load the audio files as a time series data set (4 dimensions = sources recorded at each time instant). Plot these four mixed time series (waveforms). Use the ICA software package to recover the four original sources, and plot the source time series (waveforms).

Each of the four sources are public domain music recordings from `http://freemusicarchive.org/music/Masters_Remastered/Masters_Remastered/`. That webpage contains many more music tracks than what are used here. Compare the original sources recovered by ICA to the music tracks on the webpage (you can simply listen to the available music tracks on the webpage). Which four music pieces on the webpage correspond to the original sources?

Hint: the sources are all from near the beginning of the music tracks on the webpages, no need to listen all the way through each track.

Hint: in Matlab the command 'wavread' can be used to read in each audio file as a vector of amplitudes; this vector is suitable as a source time series. The command 'wavwrite' can be used to write the recovered sources as audio files.

c) Now consider another situation where four people are speaking at the same time, and each microphone records a different hard-to-comprehend mixture of the speakers. This is simulated in the audio files 'speechmix2019_01.wav', 'speechmix2019_02.wav', 'speechmix2019_03.wav', and 'speechmix2019_04.wav'. Apply ICA to separate the different speakers. What are they saying?

Hint: they are readings from four out of the following classic books available at Project Gutenberg, but which ones? The possible books are: "Alice's Adventures in Wonderland" by Lewis Carroll, "A Tale of Two Cities" by Charles Dickens, "Dracula" by Bram Stoker, "Frankenstein; Or, The Modern Prometheus" by Mary Wollstonecraft Shelley, "Grimms' Fairy Tales" by Jacob Grimm and Wilhelm Grimm "Heart of Darkness" by Joseph Conrad, "Moby Dick; Or, The Whale" by Herman Melville, "Pride and Prejudice" by Jane Austen, "The Adventures of Sherlock Holmes" by Arthur Conan Doyle, "The Adventures of Tom Sawyer" by Mark Twain, "War and Peace" by graf Leo Tolstoy.

[2]The sounds used in these mixtures are public domain recordings from `http://www.soundbible.com` and `http://www.pdsounds.org`.

Problem B6: Independent Component Analysis for Compressing Images

In problem B3, a template is provided for compressing 10 by 10 blocks of an image using principal component analysis. That template code can also be used with Independent Component Analysis. In this exercise we try to use independent component analysis for the same image as in B3.

- Download and get to know a software package for independent component analysis. For example, use the FastICA software package; it is available at `http://research.ics.aalto.fi/ica/fastica/` for several programming languages such as Matlab, R, Python and C++.[3] You can also use any other ICA software package.
- Form the 690 by 100 feature data matrix as in B3.
- Compute the mean of the data (a 1 by 100 vector), and substract the mean from the data.
- Give the resulting zero-mean feature data matrix as an input to an independent component analysis software, e.g. the same software as in B3. Use the software to extract the first 20 independent components (sources), which will be a matrix of size 690 by 20; the software should also provide their corresponding mixing matrix $\mathbf{W}$ which is a matrix of size 20 by 100.
- Project the sources with the mixing matrix, to form an reconstruction of the feature data matrix.
- Add the mean back to the reconstructed data.

The above steps can be inserted into the template code of B3, instead of the PCA solution, so that the ICA-reconstructed feature data matrix and the ICA mixing matrix are used in place of the PCA-reconstructed feature data matrix and the PCA principal components matrix, respectively.

Compare the ICA reconstruction to the PCA reconstruction with 20 components that you computed in B3; which one looks "better" to you? What about the resulting filter images, is there a difference?

Problem B7: Linear Discriminant Analysis

Consider the white wines data set used in Problem B2, which has wine varieties and their quality ratings (low, medium, or high). Suppose we want to use Linear Discriminant Analysis to reduce the data dimensionality to 2 dimensions in a way that tries to discriminate the classes.

[3] If you use Octave, you should use the Matlab version of the FastICA package, not the older Octave version. You may need to replace the file 'fpica.m' in the FastICA package with the modified version provided in this exercise pack.

a) Either implement Linear Discriminant Analysis (LDA; as discussed in Lecture 3, slide 21) in a programming language of your choice, or become familiar with a software package that implements it. Note that for the wine data, which has more than two classes, you need the multi-class case of LDA, the two-class case is not suitable here.

   Hint: if you implement LDA yourself, in Matlab the generalized eigenvalue problem $\Sigma_b \Phi = \lambda \Sigma_w \Phi$ can be solved by the function 'eig(Σ_b, Σ_w)'. If you use Octave instead of Matlab, note that the function 'eigs' is buggy in some versions and 'eig' is thus preferable.

b) Apply LDA to reduce the wine data to two dimensions. Plot the resulting two-dimensional data set as a scatter plot (color each data point by the class of the point). Does the result separate classes well? Does it separate them better than PCA in Problem B1? (Note that this is a difficult problem and you should not expect very clear separation.)

c) Analyze the resulting projection matrix: which original features have the greatest influence on the projected coordinates?

Part C: Graphical Excellence

Problem C1: Appealing Visualization

Look for an example of scientific visualization that you find particularly beautiful in a recent issue of a high profile journal (Nature, Science, etc...). Try to explain what makes it appealing or useful.

Problem C3: Chartjunk and Data-Ink Ratio

Kenneth and Jeffrey are running a high-tech company. Unfortunately, business has been slow and the next interim report is not going to please the shareholders. Sales are declining and operating costs are going through the roof.

a) Create some artificial data about sales and operating costs that corresponds to the description above.

b) Help Kenneth and Jeffrey create a visualizartion of their sales and expenses from the past four quarters that makes the situation look less dramatic. You can use chartjunk, optical illusions, "creative layout", etc. , to display the data.

c) Kenneth and Jeffrey got busted by the SEC for providing incorrect information about the companys financial situation to the shareholders. The new management asks you to create a truthful visualization of the sales and expenses. Do this by improving a standard graph from MS Excel, OpenOffice, Matlab, etc., according to Tuftes principles. Present both the standard graph and the improved one. Discuss what improvements you made and why they are useful. Use the same data as in the previous part.

Problem C4: Improving Real-world Visualizations

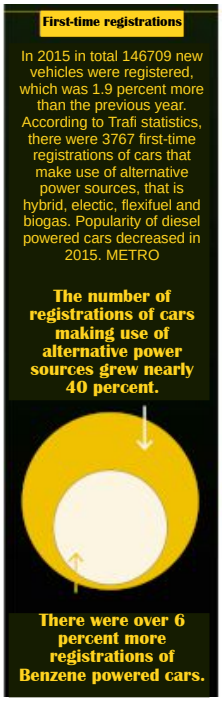
The visualizations in the pictures on the next page are translated into English from real infographics that have appeared in the widely distributed HS Metro magazine (http://www.hs.fi/metro/).

a) Analyze the figures. Do they represent good data visualizations? Do they involve chartjunk? Can the graphics cause misleading inferences about the data? Do they misrepresent the size of some effect - if so, what is the lie factor?

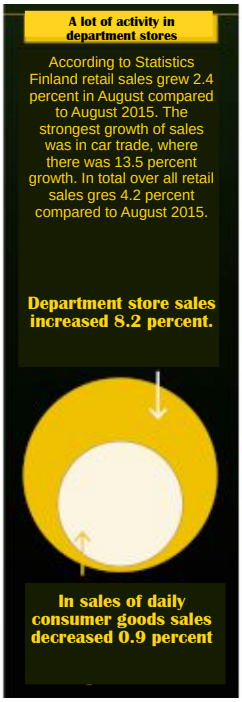
b) For each figure, create an improved visualization.

c) Consider the figure published by Finland's Ministry of Finance on their Twitter account on February 28, 2017: https://pbs.twimg.com/media/C5vOmxkWYAEzBRh.jpg . This figure represents amount of unemployed job seekers (purple curve) and amount of open jobs (yellow curve) from 2007 to 2016. Discuss the figure - can it misrepresent the size of some effect? If so, can you suggest an improved visualization?

From HS Metro, March 2, 2016, translated into English



From HS Metro,January 14, 2016, translated into English



From HS Metro, October 17, 2016, translated into English

Part D: Human Perception

Problem D1: Simultaneous Brightness Contrast

Discuss the simultaneous brightness and contrast effect shown in the figure below and explain it using the Difference of Gaussians model. Your answer does not have to contain any mathematical calculations. Instead,

1. try to give a clear explanation why the rectangles appear to be of a different gray tone while they in fact are all of the same "color".
2. if you enlarge the figure and look at the central rectangle, its left side looks brighter than the right side even though the whole rectangle is of a single color. Try to give a clear explanation why that happens.

Problem D2: Glyphs and Preattentive Processing

Consider the Ames Housing Data set of housing prices in Ames city, Iowa, USA. The data set is available at
`https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data`
but for this exercise you do not need to download the data, you only need to consider the list of variables: the houses are described by a large number of variables described in the "Data fields" section on the above page. Suppose that you have available also the physical location of each sold house, and you want to display the houses as glyphs over a map of Ames, Iowa.

Obviously the house price variable is important, but one may also want to analyze many other variables and how they relate to the price. Design a glyph that enables

the preattentive perception of as many variables (discrete or continuous) as possible; you do not need to include all variables on the above page, only try to include as many as you reasonably can. How many variables can you represent with the glyph and how accurately can the values be perceived? What should be taken into account when designing the glyph?

Problem D3: Gestalt Laws

Analyze the figures below. The picture at the top is of a trolley bus in 1950's Tampere. The picture at the bottom is the subjective Necker cube. Which Gestalt laws help to group the black shapes into something meaningful?