MTTTS17 Dimensionality Reduction and Visualization, Spring 2017, Exercise set 1

The exercises must be completed separately by each student (i.e. not in groups).

Exercise solutions must be returned by email to the lecturer. The solutions must document both the answers and the steps taken to reach them. In the case of programming exercises, any programming language can be used.

For all exercise problems, **code written to solve an exercise should be included as an appendix to the answer** (code of pre-existing libraries/toolboxes does not need to be included). 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 and Matlab/Octave, making it easier to use those languages for those exercises.

The deadline to return this exercise set is March 31, 2017.

Note: some of the later exercises contain a fair amount of computer work, most of it relatively simple application of existing implementations and/or simple implementation tasks. If you have very little experience with software like Matlab or R and have trouble completing the exercises because of that, contact the lecturer - it might be possible to arrange alternative exercises. Completing all exercises is not necessary to pass.

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 $d = 3$, $d = 6$, $d = 9$, and $d = 12$.

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 $d = 3$, $d = 6$, $d = 9$, and $d = 12$.

Problem Pre2: 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, $d = 1$, $d = 2$, $d = 4$, $d = 8$, $d = 12$, $d = 16$, 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 $y = \sin(3x_1)$ 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. To predict the target variable for a test point $\mathbf{x}_{\text{test}}$, find its 5 "nearest neighbors from the training set: the 5 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.
- 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 A: Forward Selection and Variable Ranking

The *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. 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 $\mathbf{x}$, the "nearest neighbor" is another data point $\mathbf{x}'$ that has the smallest Euclidean distance $||\mathbf{x} - \mathbf{x}'||$.

Given a finite training data set of N data points where the output values of each data point are known, the goodness of the nearest neighbor predictor can be estimated by *leave-one-out validation error*: for each data point, predict its output values using the $N - 1$ other points as the set of potential neighbors, and count the sum of squared errors between the predictions and the true values (sum over all output variables and all data points where predictions were made).

The Sculpt Faces data set "noisy_sculpt_faces.txt" is provided in the same archive as this description. It is a data set of 100 face images of a statue taken from different directions with different lighting directions; each image is a 16×16 grayscale image, where the $16 \cdot 16 = 256$ pixel values are the features. The file is an ASCII file where each row is an image: in each row, the first 256 values are the pixel values of the 16×16 image listed column by column, and the rest of the values describe how the image was taken: the 257th and 258th values are two pose angles (left-right and top-down) describing which direction the face is looking at and the 259th value is the angle of lighting. The images in this data set have been downscaled from original 64×64 sized images to reduce the amount of computation time needed to solve the exercise, and noise has been added to the pixels to simulate a poor-quality grainy camera. The original data set, and a picture of a nonlinear embedding of the original images, can be seen at `http://isomap.stanford.edu/datasets.html`. However, the images in your data set are heavily corrupted and it is hard for a human observer to see the original faces anymore—but it may be possible for a computer to notice useful features (pixels that are not too noisy and contain useful information for classification).

Task. We want to predict the orientation of the face and the direction of lighting based on the content of the noisy low-resolution image (the pixel values): the target values are the two pose angles and lighting angle. We will use nearest neighbor prediction, and use squared error of leave-one-out prediction as a performance measure. We want to choose the best features for the task: that is, we want to find out which of the 16×16 pixels are most helpful for predicting the target values. The amount of noise in the different pixels varies, and also each pixel shows a different part of the faces, so it is reasonable to expect that some pixels will be more useful for the prediction than others.

Problem A1: Forward Selection

In this exercise, we will use *forward selection* to select the best features for nearest neighbor prediction of the angles in the face images, by minimizing the leave-one-out validation error.

a) Implement the nearest neighbor predictor, and compute the leave-one-out error of nearest neighbor prediction for the Sculpt Faces data set when using

all the data features. This number is the *baseline* performance that we must improve, in order to have any benefit from feature selection.

Hint 1: the leave-one-out predictor simply means that for each face i, you find the most similar other face j, and return the pose and lighting angles of that face j as your predictions.

Hint 2: To find the most similar other face, compute the sum of squared differences between pixels of face i and corresponding pixels of face j. The face with smallest sum of squared differences is the nearest face.

Hint 3: To compute the leave-one-out error, create the predictions (the three predicted angles) for each face i as in Hint 1. Then, for each face i, compute the squared error (sum of squared differences) between the predicted angles and the true angles of face i. Lastly, sum the squared error over all the faces.

b) Implement forward selection of features, as described in slide 17 of lecture 2. Run forward selection on the Sculpt Faces data, stopping when the perfomance does not improve anymore. What is the leave-one-out validation error that you reached? Is it better than the corresponding value using all features? How many features were needed?

c) Implement a variant of the forward selection: instead of stopping when the performance measure does not improve, add the best feature in each iteration even if it does not improve the performance, and continue to run the algorithm in this way until all the features have been added to the feature set. Run this variant on the Sculpt Faces data set: record the order in which the features were added, and the performance achieved with each number of features.

d) Analyze the results of the previous step:

- Did you, at any point in the algorithm, reach a better solution than in step b)? Why?
- Plot a curve of the performance measure with respect to the number of features.

Problem A2: Variable Ranking

1. Discuss the use of simple variable ranking methods for the Sculpt Faces data set: could, for example, ranking by Pearson correlation be used, how?

2. As a more complicated ranking function, it was suggested to "make a non-linear fit of the target with simple variables, rank by goodness of fit" (lecture 2, slide 9). Let us try this, using squared error of leave-one-out nearest neighbor prediction as the goodness of fit. In other words, compute the squared error of leave-one-out nearest-neighbor prediction using each variable alone, and rank the variables by those errors. How does this ranking differ from the order that the variables were added in problem 1, step c)?

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 palomino_horse.png contains a grayscale picture of a palomino horse, 200 by 200 pixels in size. 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, will be provided separately soon.

- Divide the image into 400 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 400 such 100-dimensional input vectors.
- Compute the first PCA component of this data set (400 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.
- Do the same for 2 components, 5, 10, 20, and 30 PCA components.

Problem B4: Independent Component Analysis

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 'ica_mixdown01.wav', 'ica_mixdown02.wav', 'ica_mixdown03.wav' and 'ica_mixdown04.wav'. They each are a different mixture of four independent audio sources (same sources in each file, but mixed in a different way).[2]

   Suppose that a train passed by the room during the recording, and we want to remove this undesired audio source and only analyze the remaining three

[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.

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

interesting sources. 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. Identify the unwanted source corresponding to the passing train, and plot the remaining interesting source time series (waveforms). What kinds of sounds are the interesting sources?

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 'ica_mixdown05.wav', 'ica_mixdown06.wav', 'ica_mixdown07.wav', and 'ica_mixdown08.wav'. Apply ICA to separate the different speakers. What are they saying?

Problem B5: 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.

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 OlivettiFaces 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 face images 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?

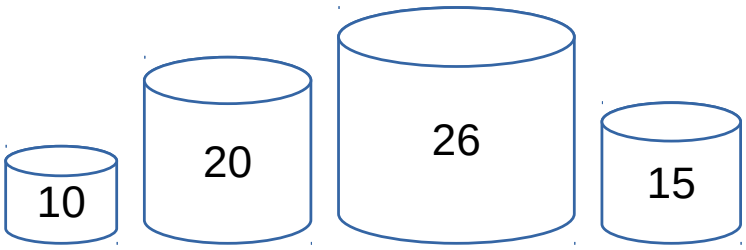
Part C: Graphical Excellence

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 C2: Lie Factor

Calculate the lie factor of the following graph. Do you think the concept of a lie-factor is useful or relevant in practice ?



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

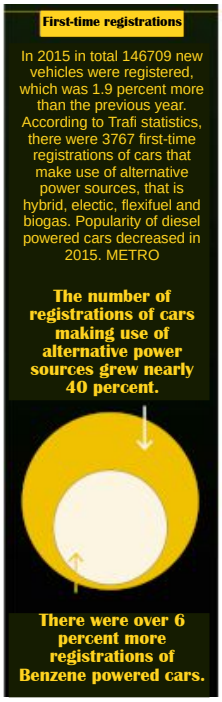
The visualizations in the pictures on the next page are translated into English from real infographics that have appeared in the widely distributed HS Metro magazine (http://www.hs.fi/metro/).

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?

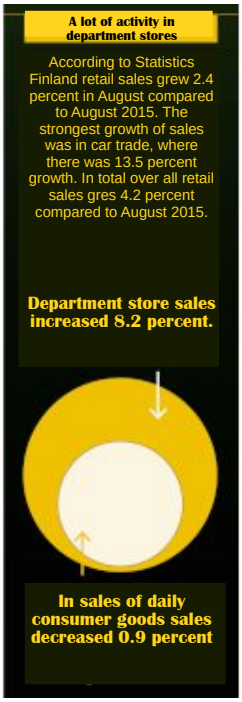
b) For each figure, create an improved visualization.

c) Consider the figure published by Finland's Ministry of Finance on their Twitter account on February 28, 2017: https://twitter.com/VMuutiset/status/836489639896690688 . 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