Technische Universität München
Fakultät für Informatik
Prof. N. Navab, Ph.D.

SS 2014
Exercise Sheet 6
May 27th, 2014

Practical Course: Machine Learning in Medical Imaging

PCA and Statistical Shape and Appearance Models

Please download the zipped MATLAB folder from the course website and complete the following exercises. In the end, please zip the entire folder again (including your written functions and scripts) and submit via email to `ahmadi@cs.tum.edu`.

Exercise 0 Implement PCA

In this exercise, you should implement the **PCA algorithm** according to the lecture slides, in a function called `myPCA.m`. Choose either SVD or the covariance method to compute the eigenvectors and eigenvalues of your data. In the MATLAB exercise folder, there is a small toy dataset `toydata.mat` with a matrix D on which you can try your PCA implementation as a sanity check. The data is a point cloud with 100 samples, sampled from a normal distribution which was non-uniformly scaled, rotated and translated in space.

Use your PCA implementation to find the principal components of the data and the variance along each PCA direction. Plot the original data cloud like in Figure 1, with the principal components coordinate-system centered at the cloud mean, and stretched with their respective eigenvalue in order to show the amount of variance along each principal axis. Project and plot the de-meaned (!) data using the principal components (see lecture slide) and observe that the point cloud is now at the Cartesian origin, with principal components re-oriented along the regular $x/y/z-$axes. Use the following figure to compare your algorithm. Once you obtain a comparable result (note that the direction of the axes may flip, depending on your implementation), you can proceed with the following exercises.

For submission, please create a small script `scriptToyData.m`, included in the MATLAB folder. It should load the data, compute the PCA and create the plot as described above.

Caveat: After eigenvector computation, we follow a sign convention for the eigenvectors, which is also used in MATLAB's PCA functions: the largest element (i.e. its absolute value) in each eigenvector column should have a positive sign - i.e., change the sign of the whole eigenvector accordingly.

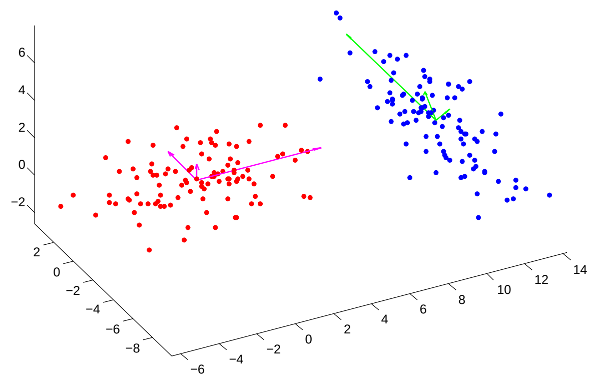

Figure 1: PCA on toy dataset: Original data (blue) and principal axes (green), weighted with the data variance in that direction. Re-oriented data (red) at origin, with re-oriented principal axes (magenta) which now are aligned with the Cartesian axes.

Exercise 1 Build a SSM

Preface. In exercises 1 and 2, we will implement parts of a method for (semi-)automatic segmentation of the midbrain from 2D transcranial ultrasound (TCUS) images. Due to the challenging nature of TCUS imaging, as described in the lecture, we will follow an Active Shape Model (ASM) approach, which employs a Statistical Shape Model (SSM) and an Active Appearance Model (AAM) as a shape prior for regularization of the segmentation boundary.

We have provided you with a function `main.m` that

- loads and prepares the required data (30 TC-US images of the midbrain and manually labeled landmarks).
- splits the data into training and testing set.
- builds a SSM model (shape prior) from the training set (exercises 1 and 2 deal with completing this step).
- builds an AAM model (appearance prior) from the training set.
- tests the model by segmenting a test image (one that does not belong to the training set) by finding the model parameters that describe current image and shape optimally.

Task. The function `trainSSModel` from `main.m` creates a SSM from a training set of shape landmarks. The first step is to align all training shapes and create a Point Distribution Model (PDM). We have provided you with a function that accomplishes that. It centers the first shape from the training set at the origin and aligns all other training shapes to this shape.

Your first task is to implement the function `statisticalShapeModel.m`, which takes the PDM as input, as well as a desired percentage of variance at which further PCA modes are cut off for dimensionality reduction. The necessary steps are:

- Convert all shapes from a x-y-representation into shape vectors (i.e. concatenate dimensions into a column vector $\mathbf{x} = [x_1, x_2, \dots x_n, y_1, y_2, \dots, y_n]^T$ of dimension $2 \cdot n \times 1$). Arrange all training shape vectors into a large matrix by concatenating all training shape vectors horizontally, in order to match the data structure for your function `myPCA.m`.
- Use your PCA implementation `myPCA.m` to find the eigenmodes of shape variation and the variance for each deformation mode (i.e. eigenvectors and eigenvalues).
- Find n_p, the number of SSM deformation modes that cover a desired percentage p of variance (e.g. 0.95) in the training set. More formally, find n_p, such that $\sum_{i=1}^{n_p} \lambda_i >= p \cdot \sum_{i=1}^{n_{train}} \lambda_i$. Discard all following modes. (Hint: in case of 0.95, you should retain $n_p = 11$ shape eigenmodes.)

Exercise 2 Visualize SSM modes

Implement the function `visualizeShapeModesOfVariation.m`. It should visualize the modes of shape variation and play a small animation for each mode by changing the mode weight. Please note that every shape created from the SSM model is a linear combination of the mean shape plus weighted shape modes. We denote the mean shape with S_μ, the eigenvectors (shape variations) from PCA as S_m, the eigenvalues as λ_m and the mode weight as w_m.

- You can calculate a legal shape from the SSM and its m-th shape mode as $S_{\text{new}} = S_{\mu} + w_m \sqrt{\lambda_m} * S_m$.
- Implement the final version of the method `visualizeShapeModesOfVariation(SSM)` in `trainSSMModel.m`, which visualizes all shape mode variations, one after another. Begin with the most significant mode of variation $m = 1$ (i.e. use the first eigenvector and first eigenvalue). Vary the mode weight w_m between $[-2, \dots, 2]$, in steps of 0.2 and plot the resulting shape (always in the same MATLAB figure) to create a small animation for the changing modes. Do the same for modes $m = 2, m = 3 \dots$ etc. and visualize all the higher modes of variation that you retained previously.