1 Task definition and steps

In this homework you will dirty your hands on **PCA** applied on images. You have to show what happens if different **principal components** (**PC**) are chosen as basis for images representation and classification. Then, you have to choose and apply a classifier in order to..classify the images ;-) under different PC re-projection and you should comment the obtained results. In particular, here is the brief list of this homework sub-tasks:

1. Download and load the provided subset of PACS dataset; setup your programming environment accordingly your needings.
2. Choose one image and shows what happens to the image when you re-project it with only first 60 PC, first 6 PC, first 2 PC, last 6 PC. Comment the results.
3. Using scatter-plot, visualize the dataset projected on first 2 PC. Repeat the exercise with only 3&4 PC, and with 10&11. What do you notice? Justify your answer from theoretical perspective behind PCA.
4. Classify the dataset (divided into training and test set) using a Naive Bayes Classifier in those cases: unmodified images, images projected into first 2PC, and on 3&4 PC. Show accuracy and compare results: what are your conclusions?
5. (Optional) Visualize decision boundaries of the classier in the first 2 PC case. Any consideration about those boundaries?

Data preparation. In this homework you will work with a subset of PACS dataset made of only 4 visual object categories. The dataset contains 1087 samples with 3 x 227 x 227 sample size.

1. Download and unpack dataset file "homework1.zip" from:

   https://drive.google.com/open?id=1isX_H74AGk3iy_YjBSfvKO1bU_rr6BqK (available on the course google drive folder)

2. Read raw pixels of all images for four classes of your choice. In python you can do so by:

   from PIL **import** Image *# or import Image*

   import numpy as np

   img data = np.asarray(Image.open($< pathtoimage >$))

   This will give you a 3-D array, where the last dimension species color of a pixel in RGB format.

3. Convert every image into a 154587-dimensional vector and prepare $Nx154587$ matrix X, where n is the number of images you have read. We refer to the rows of X as examples and to columns as features. Next, prepare an n-dimensional vector y holding ordinal labels of your image. Note that in python you can get vectorial representation of your 3-D array image array by:

   x = img_data.ravel()

1.2 Principal Component Visualization

1. Standardize **X** (make each feature zero-mean and unit-variance).

2. Use Principal Component Analysis (PCA) to extract rst two principal components from sample covariance matrix of X. Project X onto those two components. You can do it in python by running:

   from sklearn.decomposition **import** PCA

   X_t = PCA(2).fit_transform(X)

3. Choose one single image from the dataset and plot it after re-projection using only the first 60PC / 6 PC / 2 PC and the last 6 PC. Compare the results with respect to the original image. **UPDATE**: you have to calculate the principal components of the WHOLE dataset, and THEN you can transform the single image to the new chosen base.

4. Visualize X_t using scatter-plot with different colors standing for different classes:

   import matplotlib.pyplot as plt plt.scatter(X_t[:,0],X_t[:,1],c=y)

   Repeat this exercise when considering third and fourth principal component, and then tenth and eleventh. What do you notice? Justify your answer from theoretical perspective behind PCA.

5. How would you decide on the number of principal components needed to preserve data without much distortion?

1.3 Classification

1. Write down formulation of Naive Bayes classier:

$$\widehat{y} = \underset{y \in \{1,\ldots,k\}}{\arg\max}\, p(y \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_d) \ ,$$

   where $\hat{y}$ is a predicted label, k is the number of classes, x_i, i=1..d are examples, $p(x|y)$ is a Gaussian, and distribution of labels is uniform.

2. Split examples in X and y into training and testing set. You can use train_test_split from sklearn.cross_validation package.

3. Train and test Nave Bayes classier with Gaussian class-conditional distribution. You can use *GaussianNB* from package from *sklearn.naive_bayes* for this purpose.

4. Repeat the splitting, training, and testing for the data projected onto rst two principal components, then third and fourth principal components. Compare results: what are your conclusions?

5. (Optional) Visualize decision boundaries of the classier. To learn how, look at this example: http://scikit-learn.org/stable/auto_examples/ensemble/plot_voting_decision_regions.html