Machine Learning 2016/17, Homework 1. Principal Component Analysis and Naïve Bayes Classification

October 10, 2016

General information. Problem solutions should be submitted in PDF format in report style (no source code listings required).
All reports must be submitted by email (`fabiom.carlucci@dis.uniroma1.it`), with subject "`[ML1617] PCA report`". It is advised to use Python as a programming language, but you can use any language of your choice (at your own risk). In case you use Python, free `Anaconda` distribution comes with all needed packages:

`https://www.continuum.io/downloads`

In particular, you might find useful `scikit-learn` general machine learning library and `matplotlib` plotting facilities. When in doubt, read the manual and take a look at the large set of examples:

`http://scikit-learn.org/stable/documentation.html`

`http://scikit-learn.org/stable/auto_examples/`

`http://matplotlib.org/examples/`

Data preparation. In this homework you will work with the `COIL-100` dataset of 100 visual object categories. The dataset contains ≈ 7000, 128×128 colored images with subtracted background. Steps:

1. Download and unpack dataset from `http://www.cs.columbia.edu/CAVE/databases/SLAM_coil-20_coil-100/coil-100/coil-100.zip`.

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('<path-to-image>'))
```

This will give you a 3-d array, where the last dimension specifies color of a pixel in RGB.

3. Convert every image into a 49152-dimensional vector and prepare $n \times 49152$ matrix $\boldsymbol{X}$, where n is the number of images you have read. We refer to the rows of $\boldsymbol{X}$ as *examples* and to columns as *features*. Next, prepare an n-dimensional vector $\boldsymbol{y}$ 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()
   ```

Principal Component Visualization.

1. Standardize $\boldsymbol{X}$ (make each feature zero-mean and unit-variance).

2. Use Principal Component Analysis (PCA) to extract first two principal components from sample covariance matrix of $\boldsymbol{X}$. Project $\boldsymbol{X}$ 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. 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.

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

Classification.

1. Write down formulation of Naïve Bayes classifier

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

   where $\widehat{y}$ is a predicted label, k is the number of classes, $\{\boldsymbol{x}_i\}_{i=1}^{d}$ are examples, $p(\boldsymbol{x} \mid y)$ is a Gaussian, and distribution of labels is uniform.

2. Split examples in $\boldsymbol{X}$ and $\boldsymbol{y}$ into training and testing set. You can use `train_test_split` from `sklearn.cross_validation` package.

3. Train and test Naïve Bayes classifier 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 first two principal components, then third and fourth principal components. Compare results: what are your conclusions?

5. (Optional) Visualize decision boundaries of the classifier.