Machine learning lab course (Praktikum)

Summer term 2013

Fachgebiet Maschinelles Lernen
Fakultät IV, Technische Universität Berlin
Prof. Dr. Klaus-Robert Müller
Email: klaus-robert.mueller@tu-berlin.de

Problem set 1: PCA, LLE, outlier detection

On the ISIS page you can download `testsNstubs.zip` containing the files `ps1_implementation.py`, `ps1_application.py` and `ps1_tests.py`. Do not modify the names of the files or the names of the functions within these files. You are free to define additional functions within the given files. Make sure that your functions have the correct signatures.

The file `ps1_tests.py` is designed to help you debug your code. It contains test functions for each of the implementation assignments in Part 1. Be aware that (a) a passed test does not guarantee correctness for all possible inputs (b) if the test module produces a plot, you have to check if the plot looks correct. Make sure that your code passes all tests.

You have to submit `ps1_implementation.py`, `ps1_application.py` and a pdf with the analysis digitally AND in print.

Part 1: Implementation

Function stubs for these assignments have been provided in `ps1_implementation.py`.

Assignment 1 (15 pts)

Write the function `pca` with signature

```
Z, U, D = pca(X, m)
```

which receives a $d \times n$ matrix X and the number of components m to be used, and which returns the principal components, as well as the projected data points in a $m \times n$ matrix Z.

U and D should contain the principal components: U is a $d \times d$ matrix, which contains the principal directions, and D is a $1 \times d$ vector, which contains the principal values, sorted in descending order (i.e. $D_1 \geq D_2 \ldots$).

Assignment 2 (15 pts)

Implement the γ-index (see paper on the ISIS page),

```
y = gammaidx(X, k)
```

which receives a $d \times n$ matrix X containing the data points and the number of neighbours k, and returns the γ-index for each datapoint in the $1 \times n$ vector y.

Assignment 3 (20 pts)

Write a function `LLE` with signature

```
Y = lle(X, m, n_rule, param, tol)
```

which receives a $d \times n$ matrix X containing the data points, and which calculates an m-dimensional embedding $Y \in R^{m \times n}$ using the LLE algorithm. The parameter `n_rule` determines the method (`'knn'` or `'eps-ball'`) which is used to build the neighbourhood graph where `param` is the corresponding parameter (k or ϵ, respectively). `tol` is the size of the regularization parameter.

Your function should be robust against malicious parameters. Use `raise Exception` for error reporting. In particular, you should check whether the resulting graph is connected.

Part 2: Application

Function stubs for these assignments have been provided in `ps1_application.py`.

Assignment 4 (10 pts)

Write the function `usps` which applies PCA to the `usps` data set (available on ISIS) in the following manner:

1. Load the `usps` data set.
2. For each digit:
   (a) Extract all the data for this digit.
   (b) Calculate the PCA of this data.

data set	file name	data	true embedding
fishbowl	`fishbowl_data.npz`	`x_noisefree` (3D)	`z` (2D)
swissroll	`swissrole_data.npz`	`x_noisefree` (3D)	`z` (2D)
flatroll	`flatrole_data.npz`	`Xflat` (2D)	`true_embedding` (1D)

Table 1: the data sets (available on ISIS)

(c) Visualize (a) all principal values, (b) the largest 25 principal values (both as a bar plot, see `bar`) and (c) the first 5 principal directions (as images, see `imageshow`).

3. Now add Gaussian noise to the images (see `numpy.random.randn`). Select an appropriate variance on your own, such that the resulting images are very noisy.
4. For each digit:
   (a) Extract all the noisy data for this digit.
   (b) Calculate the PCA of this data and redo the plots of prinicipal values. Explain the differences.
   (c) Denoise the images by reconstruction from projections on the `m` largest principal components: the reconstruction y of a data point x by the m largest eigenvectors $v_1, \dots, v_m$ of the covariance matrix is given by
   $$y = \mu + \sum_{i=1}^{k} v_i(x^\top v_i),$$
   where μ is the mean of the original data set. The reconstruction error of x is $\|x - y\|$.
   The number of components `m` should be tuned by hand such that the reconstructed (denoised) images are fairly good.
   (d) For 5 examples of your choice, plot the original image, the noisy image and its denoised reconstruction (using `imageshow`).

Remark: use `subplot`, `gridspec` or `axes` to arrange multiple plots/images in a single figure.

Assignment 5 (10 pts)

In this exercise, we use the positive class of the banana dataset as "inliers" to which we add outliers from the negative class. We will investigate the performance of outlier detection algorithms for outlier contamination rates (i.e. percentage of outliers in the data set) of 1%, 5%, 10% and 25% relative to the positive class. Write a function `outliers_calc` which –for each of these rates– repeats the following procedure 100 times:

1. Choose a random set of outliers from the negative class of the respective size (depending on the outlier rate).
2. Add the outliers to the positive class, and compute (a) the γ-index with $k = 3$, (b) the γ-index with $k = 10$ and (c) the distance to the mean for each data point.
3. Compute the AUC (area under the ROC) for each method.

For each contamination rate and method we thus obtain 100 AUC values. Store these (use `savez`)!

Write a function `outliers_disp` which creates a plot that allows to compare the performance of the methods, using a `boxplot` to visualize the distribution of the AUC values.

Assignment 6 (20 pts)

Write `lle_visualize` which applies LLE to the data sets fishbowl, swissroll and flatroll (see table 1), using appropriate values for the parameters. It plots the data and the resulting embedding, visualizing the "true" embedding using a colour coding (e.g. using `scatter`).
Remark: LLE is sensitive with respect to the parameters. You have to fine-tune `param` and `tol`.

Assignment 7 (10 pts)

In this exercise, you study the influence of noise on LLE, using the example of the `flatroll` data set. Write the function `lle_noise` which does the following:

1. Loads the data set.
2. Adds Gaussian noise with variance 0.2 and 1.8 to the data set (this results in 2 noisy data sets).
3. Applies LLE on both data sets, where the neighborhood graph should be constructed using k-nn. For both noise levels, try to find (a) a good value for k which unrolls the flat roll and (b) a value which is obviously too large.
4. For each of the four combinations of low/high noise level good/too large k, plots the neighbourhood graph and the resulting embedding.