Lab Course Machine Learning and Data Analysis (Praktikum Maschinelles Lernen und Datenanalyse)

Fachgebiet Maschinelles Lernen
Fakultt IV, Technische Universitt Berlin
Prof. Dr. Klaus-Robert Mller
Email: klaus-robert.mueller@tu-berlin.de

Summer term 2016

Problem Set 1: PCA, LLE, Outlier Detection

On the ISIS page you can download the datasets and `sheet1_tests.py` which is designed to help you debug your code. It contains test functions for each of the implementation assignments in Part 1. Make sure that your code passes all tests. Be aware that

- a passed test does not guarantee correctness for all possible inputs,
- if the test module produces a plot, you have to check if the plot looks correct, and
- your code has to be efficient, e.g. you can not use for-loops when matrix vector multiplications would give you a significant speed-up.

You have to submit

1. a report `sheet1.pdf` with your analysis (use latex template on ISIS, max. 20 pages) and
2. your implementation `sheet1.py`.

Part 1: Implementation

Assignment 1 (10 pts)

Write the class `PCA` with signature

```
class PCA():
    def __init__(self, X):
        # ...
        self.U = ...
        self.D = ...
    def project(self, X, m):
        # ...
        return Z
    def denoise(self, X, m):
        # ...
        return X'
```

which receives a $n \times d$ matrix X and computes the principle components U and D: U is a $d \times d$ matrix, which contains the principal directions, and D is a vector of length d, which contains the principal values sorted in descending order (i.e. $D_1 \geq D_2 \ldots$). The `project` method returns the projected data points in a $n \times m$ matrix Z and the `denoise` method returns the denoised data in a $n \times d$ matrix X'.

Assignment 2 (5 pts)

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

```
def gammaidx(X, k):
    # ...
    return y
```

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

Assignment 3 (10 pts)

Implement the AUC metric

```
def auc(y_true, y_val, plot=False):
    return c
```

that takes as input the true labels `y_true` $\in \{-1, +1\}^d$, and for each point a predicted value in `y_val` where lower values tend to correspond to label -1 and higher values to label $+1$, and returns the AUC (area under the receiver operator curve). If `plot=True`, the method should also plot the ROC curve (`scatter` plot).

Assignment 4 (15 pts)

Write a function `LLE` with signature

```
def lle(X, m, tol, n_rule='knn', k=5, epsilon=1.):
    # ...
    return Y
```

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

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

Part 2: Application

Assignment 5 (15 pts)

In this assignment, you will analyze the `usps` dataset (available on ISIS) with PCA in the following manner:

1. Load the `usps` data set.
2. Analysis of PCA:
   (a) Calculate the PCA of this data.
   (b) 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 `imshow`).
3. Consider three noise scenarios (use `numpy.random.randn`):
   - `low gaussian noise`: add Gaussian noise to the images. Select an appropriate variance such that the resulting images are *moderately* noisy.
   - `high gaussian noise`: add Gaussian noise to the images. Select an appropriate variance such that the resulting images are *very* noisy.
   - `outliers`: add Gaussian noise to *only five* images. Select an appropriate variance such that those five images are *extremely* noisy and that the noise is reflected in the spectrum.

   For each of the scenarios
   (a) Calculate the PCA of this data and redo the plots of prinicipal values. Explain the differences to the original spectrum.
   (b) 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 $v_1, \ldots, v_m$ of the covariance matrix is given by

$$y = \mu + \sum_{i=1}^{m} v_i (x^\top v_i),$$

   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.
   (c) For 10 examples of your choice (including the noisy ones in the `outlier` setting), plot the original image, the noisy image and its denoised reconstruction (using `imshow`).

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

Assignment 6 (15 pts)

In this exercise, we use the positive class of the `banana` dataset (available on ISIS) 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. For each of these rates repeat 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.

Compare the performance of all methods by using a `boxplot` to visualize the distribution of the 100 AUC values for each contamination rate / method combination.

data set	file name	dim	data	reference
fishbowl	`fishbowl_dense.npz`	3d	`x_noisefree.T`	`x_noisefree.T[:, 2]`
swissroll	`swissrole_data.npz`	3d	`x_noisefree.T`	`z.T[:, 1]`
flatroll	`flatrole_data.npz`	1d	`Xflat.T`	`true_embedding`

Table 1: the data sets (available on ISIS)

Assignment 7 (15 pts)

Apply LLE to the data sets fishbowl, swissroll and flatroll (see Table 1), using appropriate values for the parameters. LLE is sensitive with respect to the parameters. You have to fine-tune `n_rule`, `epsilon`, `k`, and `tol`.

For 3d datasets, plot (1) the dataset in 3d and (2) a 2d embedding (found by LLE). Color the datapoints according to the reference value. For 2d datasets, plot (1) the dataset in 2d and (2) the reference values versus a 1d embedding (found by LLE) in a scatter plot.

Assignment 8 (15 pts)

In this exercise, you study the influence of noise on LLE, using the example of the `flatroll` data set. Do the following:

1. Load the data set.
2. Add Gaussian noise with variance 0.2 and 1.8 to the data set (this results in 2 noisy data sets).
3. Apply 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, plot the neighborhood graph and the resulting embedding.