Data Mining

Optimization, Preprocessing and *IBL* applied with scikit-learn

http://www.salle.url.edu

Engineering La Salle – Ramon Llull University

Assignment 2: Learn how to categorize digit images with IBL

Objective

The main objectives of this assignment are:

- Learn how to master the scikit-learn Python package.
- Become fluent with the selection, normalization, preprocessing and search of attributes.
- Understand the different implementations of the learning algorithms based on IBL examples.

Requirements

The assignment can be done on any operating system: Windows / Mac OS X / Linux. You need a Python interpreter. If you are not used to the Python environment, please refer to the guide posted in the e-study:

https://estudy2324.salle.url.edu/mod/resource/view.php?id=34017

The submission:

https://estudy2425.salle.url.edu/mod/assign/view.php?id=57793

If you have any problems, start a discussion at:

https://estudy2425.salle.url.edu/mod/forum/view.php?id=52225

It is also recommended to use an advanced programming editor like:

- VSCode: https://code.visualstudio.com/
- IntelliJ PyCharm: https://www.jetbrains.com/pycharm/
- Notepad++: http://notepad-plus-plus.org/

Estimated time: 2 hours (maximum 3 hours)

Deadline: December 29 at 23:59:59 CET

Description

The assignment provides the following resources:

- **Assignment2.pdf:** Statement.
- Remember that through **pip** or **easy-install** you can install the **numpy**, **scipy** and **scikit-learn** packages.

Submission:

- A document in **PDF** format that answers the questions below, alongside the corresponding **Python code**.

Tasks:

1. **Analyze the Digits Data Set**
   - *The first step is to load the following libraries:*

```
import numpy
import sklearn
import sklearn.datasets
import sklearn.model_selection
import sklearn.decomposition
import sklearn.neighbors
import sklearn.metrics
```

- *Now you can load the digits dataset into the X and Y variables:*

```
digits = sklearn.datasets.load_digits()
X = digits.data
y = digits.target
print(X.shape, y.shape)
```

- *Matrix X has 1797 rows and 64 columns (which correspond to 8x8 matrices) while variable y has 1797 rows and one column. You can run* `print(digits.DESCR)` *for more information.*

- Run the previously indicated functions in a Python script.
- Give your report: a brief description of what the basic statistics of this data are. Averages, standard deviations, number of training elements for each class, etc.
- (OPTIONAL) Import the matplotlib library and make an 8x8 imshow plot of some digits. You can use the information from: http://matplotlib.org/users/image_tutorial.html

2. **Splits: train, test and normalization of data**

- Check the function: `sklearn.model_selection.train_test_split`

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

- Divide the data in train (70%) and test (30%) sets and put them in the variables X_train, X_test, y_train and y_test.
- Check: http://scikit-learn.org/stable/modules/preprocessing.html
- Normalize the X data (train and test) so that they are centered at 0 with standard deviation of 1 (z-score normalization). This normalization should be carried out with respect to training data only. Why?

3. **Projection in different main components**

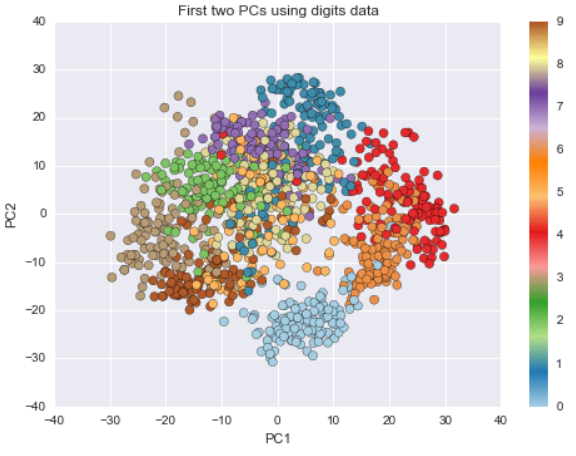
- Display the data broken down into principal components:
    - Principal Component Analysis:

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

    - Decomposition into singular values:

http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

- Break down the data into 2 main components according to the two aforementioned methods.
- (OPTIONAL) Display a scatter plot of the data by classes with the two aforementioned techniques. You should get something like this:



- Can you think of any other method to decompose the data into projections to achieve a better data separation? Try to find its implementation on the scikit-learn website. Can you apply it to the data?

4. **Use cross-validation to estimate the optimal number of K neighbors**

- Using the `sklearn.model_selection` object `KFold`, we will perform a division into *10-fold cross validation* to first estimate the optimal number of neighbors, the optimal number of dimensions.

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html

➢ Define a test function that assesses the performance of the models, use `sklearn.metrics`.

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

```
def compute_test(x_test, y_test, clf, cv):
    kfolds = sklearn.model_selection.Kfold(…)
    scores = []
    for i, j in kfolds:
        …
        scores.append(…)
    return scores
```

➢ **Implementation of the search for the closest K neighbors**

o To make a parametrization with respect to the K closest elements we use:

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

o `knearest = sklearn.neighbors.KNeighborsClassifier()`

➢ **How can we perform a parameter search process with cross-validation?** For this purpose, you can use:

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

o Now that we know the learning method (KNN) to be used, we need to search for the best option regarding:

- Input transformation (normalization) method
- Number of dimensions (columns) of the input data
- Number of neighbors
- Algorithm for weighting the closest neighbors

- Search for the aforementioned parameters that achieve the best possible classifier based on the cross-validation score. Print or graphically visualize the evolution of the search and show the results and how you obtained them.

5. **Conclusions**
   - Report the conclusions of the assignment by discussing, at least, the following sections with examples and proper reasoning:

   a) **Explanation of the effect of dimensionality in KNN.**

   b) **Interpretation of the results.**

Congratulations! Not only have you learned IBL with Scikit-Learn, but you have also learned how to recognize digits, transform the input space, and work on search processes!