hugomcp@di.ubi.pt, 2024/25

Machine Learning

Practical Sheet 5: Dimensionality Reduction

PCA Exercise

Consider the “breast Cancer” dataset, available at the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29. (It can also be found at the course web page: https://di.ubi.pt/~hugomcp/ml/breast_cancer.csv)

This set contains features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. There are 30 features in this set, plus the first two columns in each row that provide the row IOD and the class information (M=Malign, B=Benign).

1) ID number
2) Diagnosis (M = malignant, B = benign)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness (perimeter^2 / area - 1.0)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

The “mean”, “standard error” (SE) and "worst" or largest (mean of the three largest values) of these features were extracted from each image, resulting in 30 features. For instance, column 3 is Mean Radius (i.e., after “ID” and “Diagnosis”), column 13 is Radius SE and column 23 is Worst Radius.

There are a total of 580 instances, with the following class distribution: 357 benign, 212 malignant

1) Create a “Google Colab” Python script that loads the dataset, and divides the normalized data into “Learn”, “Validation” and “Test” subsets.

2) Normalizes the available data according to min-max criterium.

3) Create a neural network that learns to distinguish between “Malign” and “benign” cases.

4) Use PCA to reduce the dimension of the dataset.

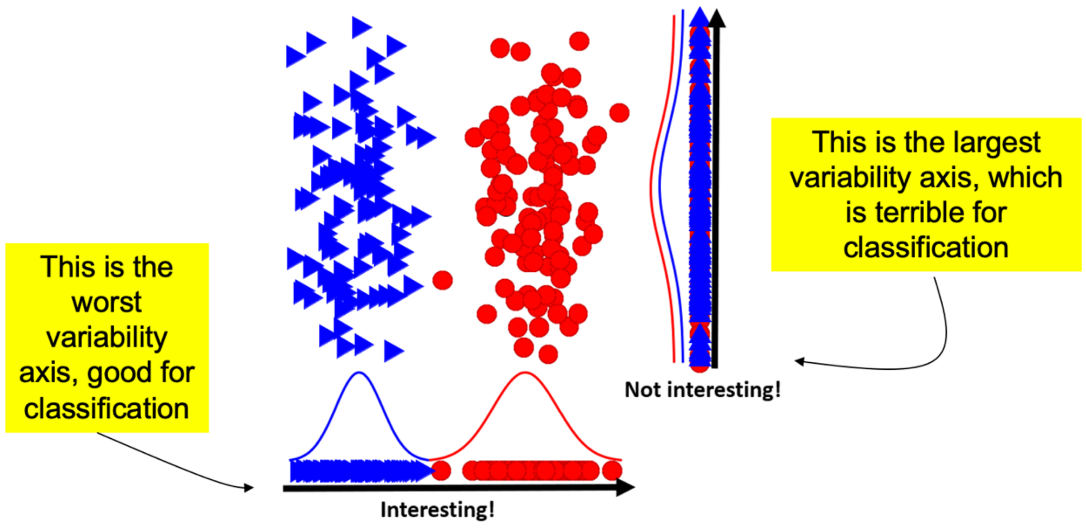
   a. Create different data versions that keep 99%, 95%, 90% of the variability.

5) Create neural networks that distinguish between “Malign” and “benign” cases in the compacted spaces.

6) Compare the effectiveness attained by the neural networks that analyze the raw and the compacted spaces.

LDA Exercise

Recall that there are certain “PCA” projections that are not good for classification purposes. In practice all cases where the axes corresponding to the largest variability are not those that enable to better discriminate between classes.


This is the worst variability axis, good for classification
This is the largest variability axis, which is terrible for classification
Not interesting!
Interesting!

This time we will use the “Digits” dataset to illustrate the differences between **PCA/LDA** projections.

“**sklearn**” library provides an easy way to access the dataset:

```
from sklearn.datasets import load_digits
digits = load_digits(return_X_y=True)
```

There are a total of 1797 instances in the set, with 64 features:

```
print(digits.data.shape)
```

Then, we can see any digit of the dataset (using its index "idx"):

```
import matplotlib.pyplot as plt
plt.gray()
plt.matshow(digits.images[idx])
plt.show()
```

We start by transforming the dataset into a (n_samples, n_features) matrix:

```
digits = load_digits()
n_samples = len(digits.images)
X = digits.images.reshape((n_samples, -1))
y = digits.target
```

Next, we will divide the data into two disjoint sets: training + tests

```
# repeatability
seed = 3
np.random.seed(seed)
# learning/test split
train_indices = np.random.choice(len(X), round(len(X)*0.8), replace=False)
test_indices = np.array(list(set(range(len(X))) - set(train_indices)))
X_train = X[train_indices]
X_test = X[test_indices]
y_train = y[train_indices]
y_test = y[test_indices]
```

Next, we can perform feature normalization:

```
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train=scaler.transform(X_train)
X_test=scaler.transform(X_test)
```

Note that the .fit() method finds the minimum and maximum values per column (**only in the learning data!**), and then we convert all values into the unit interval, both the learning and test. By doing this, note that it is not assured that all elements in the test data are in the unit interval (why?)

Next, we obtain the PCA and LDA representations:

```
pca = PCA(n_components=2)
X_train_pca = pca.fit(X_train).transform(X_train)
X_test_pca = pca.transform(X_test)
lda = LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit(X_train, y_train).transform(X_train)
X_test_lda = lda.transform(X_test)
```

Compare the effectiveness of the original representation to PCA and LDA representations, depending of the number of components used in their projections:

- Using a very simple classifier, such as a NN with a single Layer

- Using a more sophisticated classifier, multi-layered NN.