OST
Ostschweizer
Fachhochschule

Lab 11

FTP MachLe MSE
HS 2023

Dimensionality Reduction

MSE MachLe
WÜRC

Lernziele/Kompetenzen

- You know the main motivations and applications and drawbacks of *dimensionality reduction* of a high dimensional dataset.
- You can explain by an example what is meant by the term *curse of dimensionality* and the consequences of this fact.
- You can apply *Principal Component Analysis* (**PCA**) to high dimensional datasets and explain how PCA works. You know different versions of PCA like incremental PCA, vanilla PCA, randomized PCA and kernel PCA. You can explain the differences between these methods and know to apply them accordingly using `scikit learn`.
- You know the *kernel trick*, it's advantages and when it can be applied to a given technique. You can list at least four different kernels $k(x, x')$ that are frequently used in Machine Learning.
- You know the four axioms that define a *metrics* and can name four different metrices that are commonly used in Machine Learning.
- You know the manifold hypothesis and different manifold techniques like **MDS** (*Multidimensional scaling*), **LLE** (*local linear embedding*), **t-SNE** (*t-distributed stochastic neighbor embedding*) and **Isomap** (*Isometric mapping*) and can apply them to high dimensional datasets in `scikit learn`.
- You can successfully apply dimensionality reduction techniques for the *visualization* of high dimensional datasets, for *noise reduction* in datasets and for the *elimination of meaningless features used for classification.*

1. Applications of dimensionality reduction techniques [A, I]

a) What are the main *motivations* for reducing a dataset's dimensionality?

b) What are the main *applications* of dimensionality reduction techniques?

c) What are the main *drawbacks* of dimensionality reduction techniques?

2. Curse of Dimensionality [A, I]

What is the *curse of dimensionality*? Can you explain this with an example?

3. General questions [A, II]

a) Once a dataset's dimensionality has been reduced, is it possible to reverse the operation? If so, how? If not, why?

b) Can PCA be used to reduce the dimensionality of a highly nonlinear dataset? Which methods can alternatively be used?

c) How can you evaluate the performance of a dimensionality reduction algorithm?

d) Does it make sense to chain two different dimensionality reduction algorithms?

e) In which cases would you use incremental PCA, randomized PCA or kernel PCA?

4. PCA and scaling importance [A, II]

Feature scaling through standardization (or Z-score normalization) can be an important preprocessing step for many machine learning algorithms. Many algorithms such as SVM, K-nearest neighbors, and logistic regression and PCA require features to be normalized. In PCA we are interested in the components that maximize the variance. If one component (e.g. human height) varies less than another (e.g. weight) because of their respective scales (meters vs. kilos), PCA might determine that the direction of maximal variance more closely corresponds with the weight axis, if those features are not scaled.

The dataset used is the `Wine Dataset` available at UCI `https://archive.ics.uci.edu/ml/datasets/wine`. This dataset has continuous features that are *heterogeneous in scale* due to differing properties that they measure. It is the outcome of a chemical analysis wines grown in the same region in Italy but derived from *three different cultivars*. The analysis determined the quantities of 13 constituents (i.e alcohol content, and malic acid) found in each of the three types of wines.

a) Open the Jupyter notebook `Lab11_A4_PCA_ScalingImportance_Wine.jpynb`. We import the following modules:

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.pipeline import make_pipeline
```

b) Split the data into a training dataset (70%) and a test dat set (30%) using `train_test_split` from `sklearn.model_selection`.

c) Create a *pipeline* called `unscaled_clf` with `make_pipeline` using the following two operations:

1. PCA using only two components: `PCA(n_components=2)`
2. Naive Bayes Classifier (`GaussianNB`) for the different wine types (target).

d) Fit the model using the `.fit` method. Estimate the cultivar (target) using the `.predict` method on the test data and print the accuracy score on the test data using `metrics.accuracy_score`.

e) Change the PCA to 4 components, fit the model using the `.fit` method. Estimate cultivar (target) using the `.predict` method on the test data and print the accuracy score on the test data using
`metrics.accuracy_score`.

f) Create a new pipeline called `scaled_clf` using the following operations:

1. `StandardScaler()`
2. PCA using only two components: `PCA(n_components=2)`
3. Naive Bayes Classifier (`GaussianNB`) for the different wine types (target).

g) Fit the model using the `.fit` method. Estimate the wine type (target) using the `.predict` method on the test data and print the accuracy score on the test data using `metrics.accuracy_score`.

h) Plot the transformed data as a scatter plot in the new coordinate system of the two principal components, once using the scaler and once without the scaler. Use diffent symbols and markers for each target (cultivar).

```
fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=FIG_SIZE)

for l, c, m in
  zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
  ax1.scatter(Xtrain_PCA_unscaled[y_train == l, 0],
        Xtrain_PCA_unscaled[y_train == l, 1],
  color=c, label='class %s' % l, alpha=0.5, marker=m)

for l, c, m in
  zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):
  ax2.scatter(Xtrain_PCA_scaled[y_train == l, 0],
        Xtrain_PCA_scaled[y_train == l, 1],
  color=c, label='class %s' % l, alpha=0.5, marker=m)
```

*Note: **Naive Bayes methods** are a set of supervised learning algorithms based on applying Bayes' theorem with the* naive *assumption of independence between every pair of features. In spite of their apparently over-simplified assumptions,* naive Bayes classifiers *have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the* curse of dimensionality*. However, although naive Bayes is known as a decent* classifier*, it is known to be a bad* estimator*, so the probability outputs from `predict_proba` are not to be taken too seriously.*

5. Stochastic Neighbour Embedding [A, II]

a) Use t-SNE to reduce the MNIST dataset down to two dimensions and plot the result using Matplotlib. You can use a scatterplot using 10 different colors to represent each image's target class.

b) Write colored digits at the location of each instance, or even plot scaled-down versions of the digit images themselves (if you plot all digits, the visualization will be too cluttered, so you should either draw a random sample or plot an instance only if no other instance has already been plotted at a close distance). You should get a nice visualization with well-separated clusters of digits.

c) Try using other dimensionality reduction algorithms such as PCA, LLE, or MDS and compare the resulting visualizations.

6. Feature Engineering using PCA [A, II]

a) Load the MNIST dataset and split it into a training set and a test set (take the first $60'000$ instances for training, and the remaining $10'000$ for testing).

b) Train a *Random Forest classifier* on the dataset and time how long it takes, then evaluate the resulting model on the test set.

c) Next, use PCA to reduce the dataset's dimensionality, with an explained variance ratio of 95%. Train a new Random Forest classifier on the reduced dataset and see how long it takes. How much faster was the training on your machine?

d) Next evaluate the classifier on the test set: how does it compare to the previous classifier?

7. Kernels and the Kernel Trick [A, II]

a) Explain what a kernel is.

b) List four commonly used kernels $k(x, x')$ in Machine Learning.

c) Show, that the RBS-kernel is symmetric and positive semi-definite.