KERNEL PRINCIPAL COMPONENT ANALYSIS

Authors: G. H. Zerwes, *Universidade Federal do Espirito Santo, Vitória, Brazil*

Research Supervisor: L. Homri, *Arts et Métiers ParisTech, Metz, France*

1 Introduction

The Kernel Principal Component Analysis (KPCA) is an algorithm, similar to PCA, that has applications in denoising and compression. Through its use of kernel functions it allows for a separability to be obtained in non-linear datasets.

The basic working principle consists of using some user-specified kernel that will transform the non-linear data into a much bigger dimensional space where it's hoped to be linearly separable. After the data is transformed, the same approach used in Principle Component Analysis applies, that is, the data is projected in the directions that will retain the most amount of information.

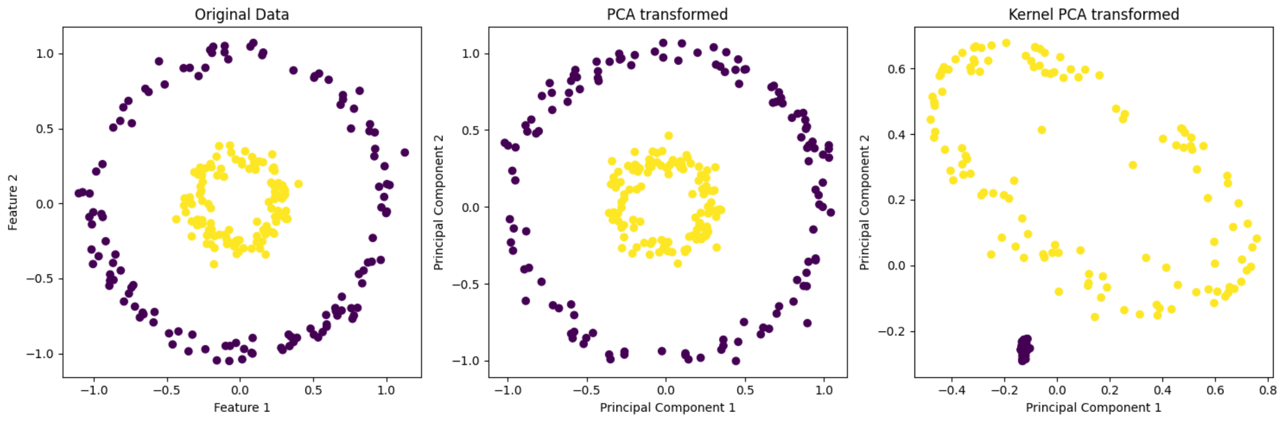

Figure 1: Comparison between PCA and KPCA

An example of this algorithm can be seen in figure 1. In the first figure, it is possible to observe the raw data on which the principal component analysis would like to be performed. Note the fact that the data is not linearly separable, that is to say, there is no straight line that will divide correctly the two classes.

In the middle plot, the data after the PCA transformation has been applied is seen. However, as PCA projects the data into straight lines, it has not managed to separate the two classes. Finally, the last plot shows the application of KPCA on the dataset, which managed to separate the classes into two linearly separable groups.

2 Pseudo-code

To use this algorithm on your data, the pseudo-code below shows a procedure that should be followed.

```
#Step 1: Gather the data.

X = [x_value_1, x_value_2, ...]

#Step 2: Fit and transform the data.
reduced_data = KPCA.fit_transform(X)
```

References

Bishop, C. M. (2006). *Pattern Recognition And Machine Learning*. Number 758. Springer.