# Lab 9 – PCA and KPCA

MACHINE LEARNING

SAUMAY AGRAWAL
16BCE1151

---

## EXPERIMENT

- Choose a good dataset, that has many features. Maybe two datasets, one that is linearly classifiable and other that is not.
- Perform dimensionality reduction via
    - PCA
    - KPCA
- The reduced dimensionality (EXTRACTED/TRANSFORMED FEATURES) are then to be used in any classifier to report pre and post dimensionality reduction performance.

## ALGORITHMS

PCA (Principle Component Analysis)

As the dimensions of data increases, the difficulty to visualize it and perform computations on it also increases. PCA finds a new set of dimensions that all the dimensions are orthogonal and therefore linearly independent too and ranked according to the variance of data along them. It means the principal axis with more variance or more spread out data comes first than others. So this reduced set of dimensions, is able to represent the original dataset, almost losslesly.

How does PCA work:

1. Calculate the covariance matrix X of data points.
2. Calculate eigen vectors and corresponding eigen values.
3. Sort the eigen vectors according to their eigen values in decreasing order.
4. Choose first k eigen vectors and that will be the new k dimensions.
5. Transform the original n dimensional data points into k dimensions.

KPCA (Kernel PCA)

The PCA technique works very well only if the data is linearly separable. However, in the real world, the data is sheldom linearly separable. Most of the data that we get is the ‘dirty

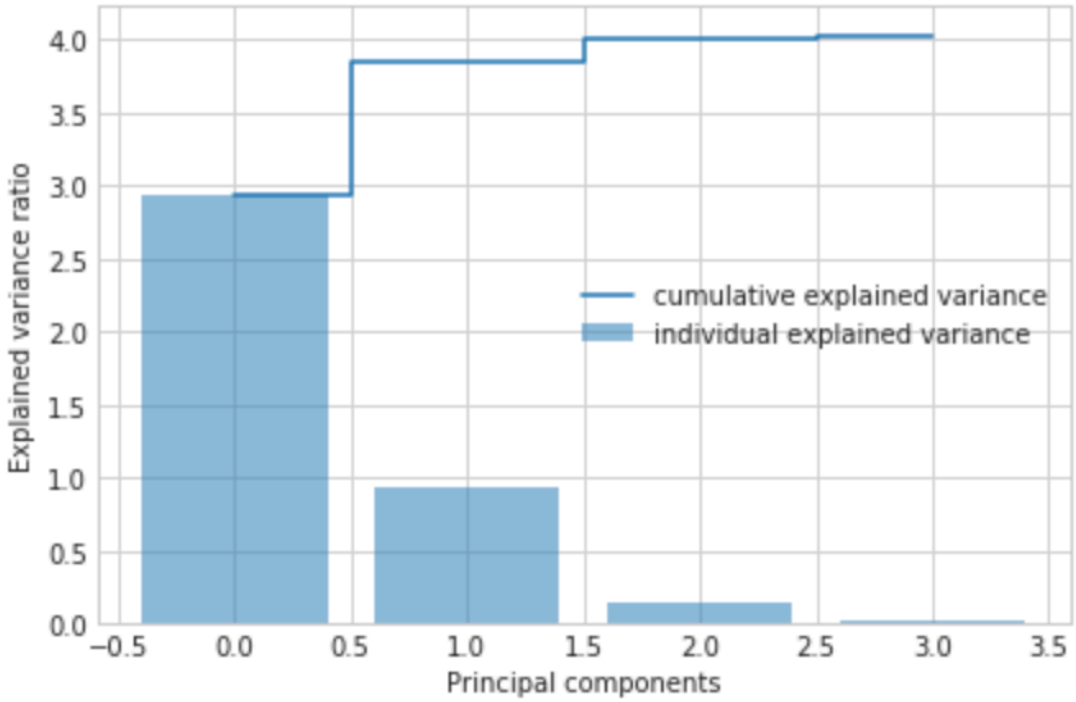
data'.  Since it not linearly separable, the PCA algorithm doesn't help much with such data. Hence the 'kernel trick' is used with PCA, in which the original data is projected onto a higher dimension using a non-linear mapping function called the 'kernel function', to achieve better classification of data. This is also backed by the Vapnik–Chervonenkis theory, which tells us that if we project our data into a higher dimensional space, it provides us with better classification power.

## OBSERVATIONS

Classifier used - XGBoost

IRIS DATASET

- The graph for the importance of components is given below.

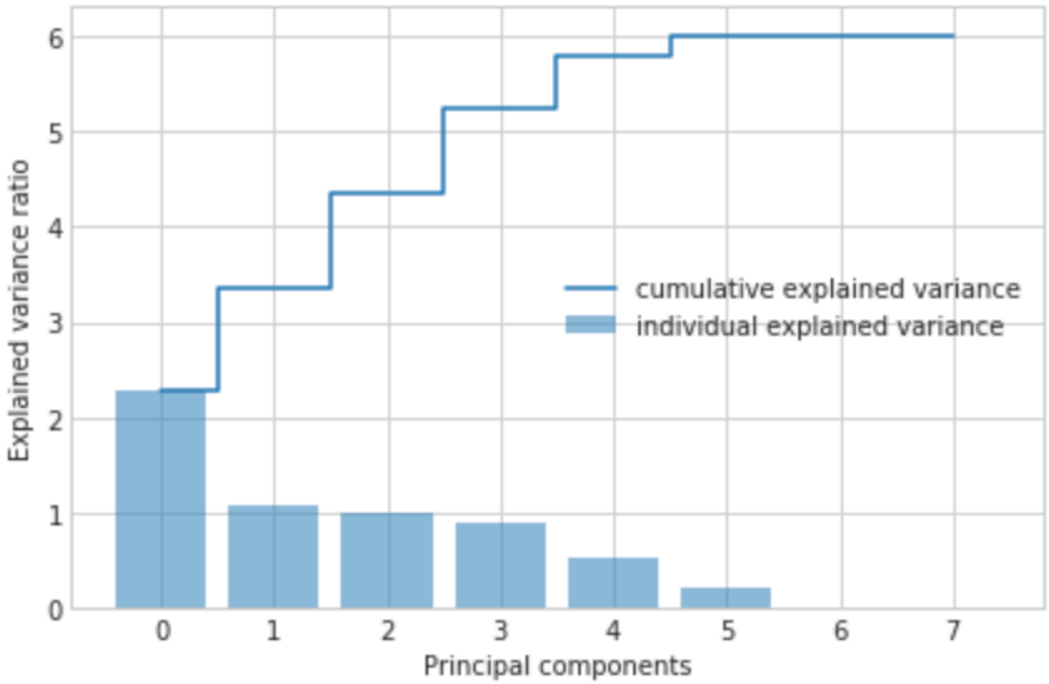



- For PCA, I got the accuracy in following order of no. of components, 3 > 1,4 > 2. Accuracy on using 3 components is the highest, which is 95.6%.
- In KPCA, the results are as follows:
    - For 'rbf' kernel, we get the best results for gamma=2 with accuracy crossing 90% with 2 components, and reaching a peak of 93.3% for 3 and 4 components. Whereas, gamma=1 takes minimum 3 components to cross

90% accuracy, but gives the highest accuracy of 95.6% for all the four components.
    - For 'sigmoid' kernel, we get the best results for gamma=1, with minimum accuracy being 91.1% for 1, 2 and 3 components all, and maximum being 97.8% for 4 components.
    - For 'poly' kernel, gamma=5 gives the best results, with accuracy values of 82.2% for 1 component, 88.9% for 2 and 3 components, and 91.1% for 4 components.
    - The 'cosine' kernel doesn't need the kernel coefficient (gamma). The results for all gamma values are same, as these values are ignored by sklearn for cosine kernel. We get the accuracy of 75.6% for 2 components, 77.8% for 1 component followed by 88.4% for 3 and 4 components.
- Accuracy has increased due to dimensionality reduction as previously it was 95.6%, but now we have peaked to 97.8%.

OLYMPICS DATASET

- I had to subsample the data drastically, as the computation time kernel PCA was way too much for this dataset. The dataset was reduced to roughly 14,000 rows and 9 attributes.
- Below is the graph showing the importance of various components. We can see that unlike iris dataset, the importance of the individual components is very low.




- For PCA, I got the accuracy of 85.7% for number of components between 1 and 4. For higher number of components, accuracy increased to 85.8% ie only by 0.1%.

- For KPCA, these were the results:
    - For ‘rbf’ kernel, the results were unaffected by the number of components, and gamma values, the accuracy being 85.7%.
    - For ‘sigmoid’ kernel, the results lay between 85.7% and 85.8% accuracy. Higher accuracy value for the lowest number of components (5 in this case), was achieved for gamma=5
    - For ‘poly’ kernel, the accuracy values were either 85.6% or 85.7%. The drop in accuracy came for higher gamma values.
    - For ‘cosine’ kernel, similar results were obtained.
- Accuracy didn’t increase from previous results upon using dimensionality reduction.

## INFERENCE

- The PCA and KPCA techniques are good techniques to reduce the dimensions of dataset, only to the most relevant component axes.
- For non-linearly separable data, the kernel transformation will improve the results of the machine learning model used only if appropriate kernel is chosen along with appropriate value of kernel coefficients.
- These techniques reduce the computational time taken by the model. However, for practical applications, it should be noted that the transformation process will also take some or significant amount of computational time of its own, depending on the kernel and number of components used.