Practical 2 : Kernelized PCA and Clustering on 20 Newsgroups

In this practical, we will implement a simple unsupervised learning algorithm using the 20 Newsgroups Data Set. This data set contains approximately 20000 newsgroup documents, partitioned (almost) evenly across a variety of topics, such as:

- comp.
  - graphics
  - windows.x
- rec.
  - autos
  - motorcycles
- ...

Loading the data can be done as in the case of Breast Cancer database. The initial raw data can be automatically split in two sets and loaded:

```
from sklearn.datasets import fetch_20newsgroups
cats = [ 'sci.med', 'misc.forsale', 'soc.religion.christian']
newsgroups_train = fetch_20newsgroups(subset='train', categories=cats)
newsgroups_test = fetch_20newsgroups(subset='test', categories=cats)
```

Make sure to load relatively few categories when testing as this dataset is quite large. As before, `newsgroups_train.data` contains the data as *strings*, with the corresponding labels in `newsgroups_train.target`. However, this representation is not too useful to work on. You can use feature extraction functions to convert the data in the usual matrix form:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
vectors = vectorizer.fit_transform(newsgroups_train.data)
```

This will convert each entry from a string to a vector, where each coordinate corresponds to the frequency of a word within this document. Alternatively, the following representation might be more useful:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(newsgroups_train.data)
```

Instead of just counting each word, this representation assigns a score based on the overall frequency of the word in the whole dataset as well as it's frequency within the document. As such, words like 'this' will have a low score since it is not indicative of specific features, even though it appears frequently. You can check tf-idf for more details. Both these transformations will store the results in *vectors* which is an $N \times D$ matrix where N is the number of documents and D the number of features, matching the organization of `newsgroups_train.target`. However, it is a sparse matrix. If needed, you can convert it to a dense one by:

```
vectors = vectors.toarray()
```

You have to implement the Principal Component Analysis (PCA) with Kernels and apply it to the training. Compute the eigenvectors associated with the $\phi(X)\phi(X)^T$ matrix, where ϕ is a low-degree polynomial or a Radial Basis Function kernel, as shown on slides $23 - 26$. Once you have reduced the dimensionality of your training data set, use the k-means algorithm to cluster it, setting the number of clusters equal to the number of categories. Unlike the first tutorial however, this time you should provide your own implementation of k-means clustering using either this description or the lecture notes.

Finally, use `matplotlib` to plot the classification error, relative to the number of extracted features for PCA and whether linear, polynomial or radian functions were used for basis expansion.

Optional: You can try implementing a hierarchical clustering algorithm. You may want to use the *cosine similarity* between the tf-idf vectors as your notion of similarity to use in hierarchical clustering. However, you should use this opportunity to explore various similarity/dissimilarity measures and different linkage methods.