Artificial Intelligence: Assignment 3

May 25, 2016

Prof. Andrea Torsello

Francesco Pelosin

Feature transformation

Preprocessing a dataset might be useful to extract and understand patterns and knowledge that cannot be seen otherwise. Feature transformation does so and in particular, during this assignment, we had to apply Principal Component Analysis and Independent Component Analysis to a set of images. After applying features transformation we were asked to test a K-Nearest Neighbors classifier to study the performances in different settings.

In the feature transformation problem, we have a dataset composed by some observations that are characterized by a set of attributes, we then try to find a transformation that create a new set of features:

$$X \sim F^n \to F^m$$

Where usually $m < n$, because we want the linear transformation to reduce the dimensionality[1] of the feature space and we want to do so by preserving the maximum amount of information.

Principal Component Analysis (PCA)

One feature transformation technique is Principal Component Analysis, and it is strictly related to the Eigenproblem since it exploits the properties of eigenvalues and eigenvectors. In particular the principal components are the normalized eigenvectors of the covariance matrix of data. There are two famous definitions for PCA, one say that what PCA in practice does, is to find directions where to project data, such that these new directions **maximize variance**. Usually the projection space that we use is smaller than the original feature space, this is because we aim to get rid of redundant and not-so-descriptive dimensions and it also provides a mechanism for compressing data and **reducing dimensionality**.

PCA, as the name suggests, returns principal components ordered by the value of the eigenvalues. The first principal component returned is the direction which has maximum variance overall possible directions, then the other components that it retrieves, give less and less variance until we arrive at a point where there are components which does not provide additional information and they might just introduce noise. So we can say that principal components are **ranked** according to how much descriptive they are.

Another property of PCA is that it returns directions which are **mutually orthogonal**, so if we think of a two-dimensional feature space, after finding the first direction, we only have one possible choice for the second component, and PCA simply returns a linear rotation of the plane. Since this method maximizes variance it also provides best reconstruction without any information loss in case we use all the dimensions, because it will be simply perform a change of base. When we project onto a sub-space we also get a good reconstruction because PCA maximizes variance, therefore the error of reconstruction should be minimized since data is spread as much as possible.

Another property that characterize PCA, is that it best works when data respects some **Gaussianity**. This is because PCA mainly works with zero-mean data and exploits the statistic of variance and the only zero-mean probability distribution that is fully described by the variance is the Gaussian distribution.

Independent Component Analysis (ICA)

ICA is a different technique to perform feature transformation, it was developed to solve problems related to the famous *cocktail-party problem* which basically aims to reconstruct and separate different sources from a linear combination of them.

[1]We don't want to do it always, for example Kernels do the opposite : increase dimensionality through a non-linear transformation

We can say that one of the things that ICA tries to do is to find a linear transformation of the feature space, to a new space, such that each of the individual new features are **mutually independent**, that is:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = Y \qquad\qquad y_i \perp y_j$$

ICA simply finds a transformation that separates features as much as possible, it does **not reduce dimensionality** of data as PCA does.

Another important thing is that ICA tries to maximize the **mutual information** between X and Y. That is to say, that ICA transformation has to preserve as much information as possible to be able to reconstruct X when we have Y. ICA differs from PCA also because it does not provide an ordering of the components based on variance, so it does **not rank** the dominants components.

A second major difference is that ICA assumes the independence of the original sources, instead PCA does not have this assumption. As we said for PCA, the optimal conditions where it works best, is in case of Gaussianity in data, instead ICA best works in the opposite situation, that is when data respect **non-Gaussianity**.

Classification

K-nn algorithm

One of the most famous classification algorithms is the K-Nearest Neighbor algorithm, it belongs to the family of **supervised learning** algorithms. K-nn is relatively simple and intuitive, it basically needs a training set composed by a set of points and their classification. When a new point from the test set is tested toward the model, K-nn simply looks for the K nearest already-classified examples and classifies the new example by using a **majority rule**.

Obviously K-nn needs some **metric** to compute the distances between the new point and the training examples. The most used one is the Euclidean distance, but we don't have a best candidate, it depends from the structure of the problem. There might be situations where the Euclidean distance is not useful at all, in fact other famous distances used with K-nn are the Minkowski distance, the Hamming distance and also the Cosine similarity is a well known and used measure, but nothing stops us to use other types of metrics maybe also defined by us specifically for the problem.

K-nn is quite effective and understood also because of its simplicity, but it has some drawbacks. In fact it does **not perform very well on high dimensional data**, because of the high computational effort to compute distances between points and because with high dimension spaces the nearest neighbor of a point might be very far and so the very notion of neighbor becomes a little weak. The latter problem is referred as "*course of dimensionality*".

Another weak point is that we don **not know a priori which** K performs best during classification, the only way to figure out the best value, is trying various instances with different K values. Is important to set K to be a odd number since the majority rule meets natural discrimination and the algorithm will never be in a 50% uncertain case.

Task 1 The first task was to show a scatter plot of the first two principal components of the dataset, Figure 1 shows the projection of data. As we can see there is not a good separation of classes, in fact classification with only two principal components performs very bad as we will see. If we examine a three dimensional scatter plot of the first three principal components, then we can see a little bit more separation between classes and we can easily spot some pattern in data.

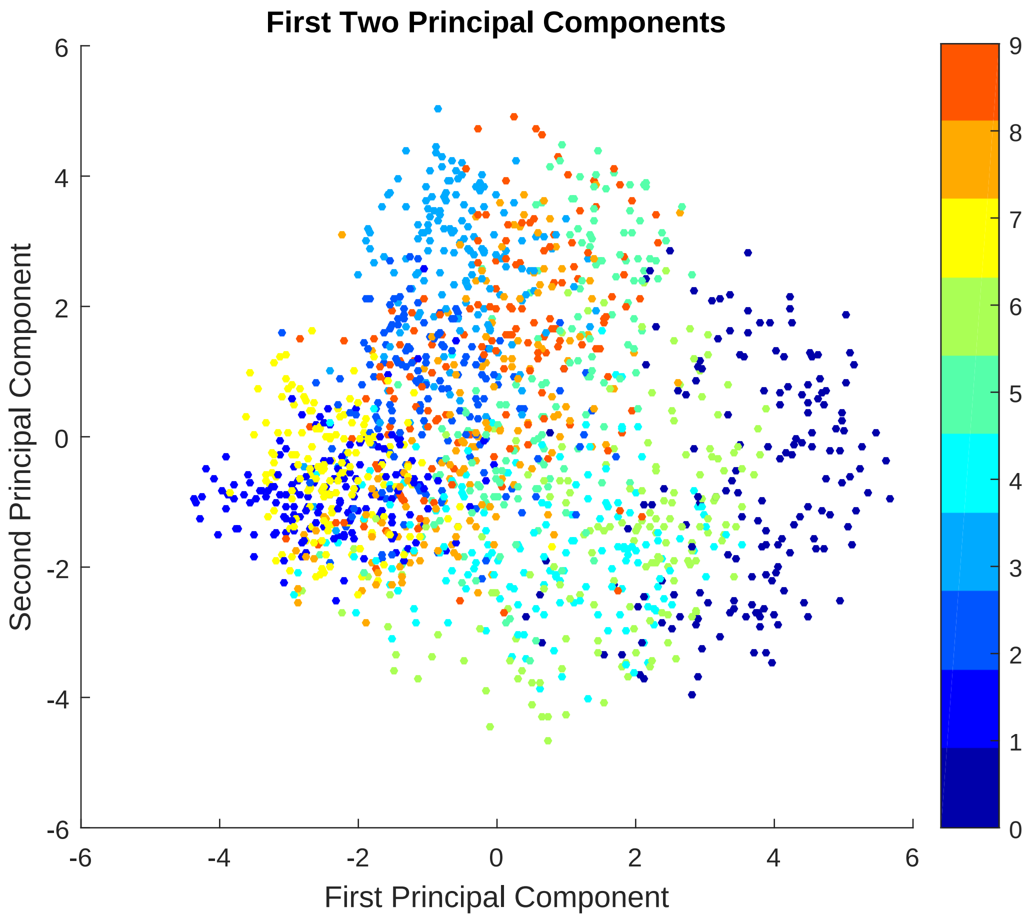

Figure 1: First two Principal Components after centering, each color represent a different class.

Task 2 The second task was to study the performance of a Nearest Neighbor classifier as the number of the principal components increase to the full set. With Figure 2 we have in the x axis the number of the principal components increasing and at the y axis we have the average accuracy of 10 K-nn instances each of them trained over 10 different examples per class taken each time randomly from the full examples set.

As we said we could not know a priori the best K, so I tried with five different K odd values. I also introduced $K = 2$ because I wanted to show that using an even number results in bad performances, in fact Figure 2 prove it.
Another result that could be predicted is that K-nn provides best results when the data is projected between the first 15-60 Principal Components (with its peak between 25-30). This means that only 60 Principal Components are truly relevant to the description of the phenomenon, in fact we can see that as the number of Principal Components increments, less performance we have. That is because after a certain point Principal Components does not provide any relevant information, instead we are actually injecting noise. And the second reason of this behavior comes from the "course of dimensionality" problem of K-nn that does not perform very well in high dimensions.

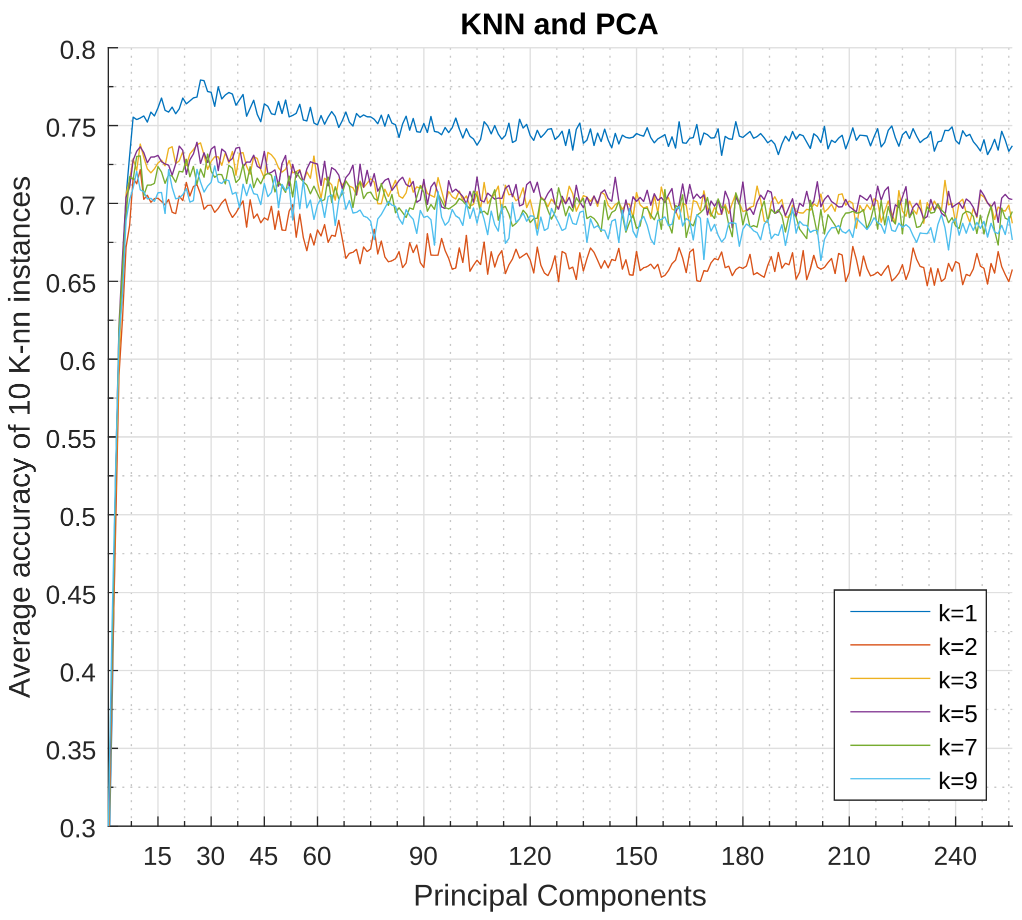

Figure 2: Performance of K-nn with PCA reduction on data.

Task 3 The third task was to test the K-nn algorithm using all the dimensions available, but computing the distances on the whitened data. Whitening provides a mechanism to decorrelate data, the basic idea behind this operation is that we want that all features are uncorrelated and usually when this happens the plot usually assumes a spherical shape, that's why it is also referred as sphering. To give the idea Figure 3 shows the first two principal components computed with whitened data and Figure 4 shows the covariance matrix of the whitened data, which should be an identity matrix since data is uncorrelated.

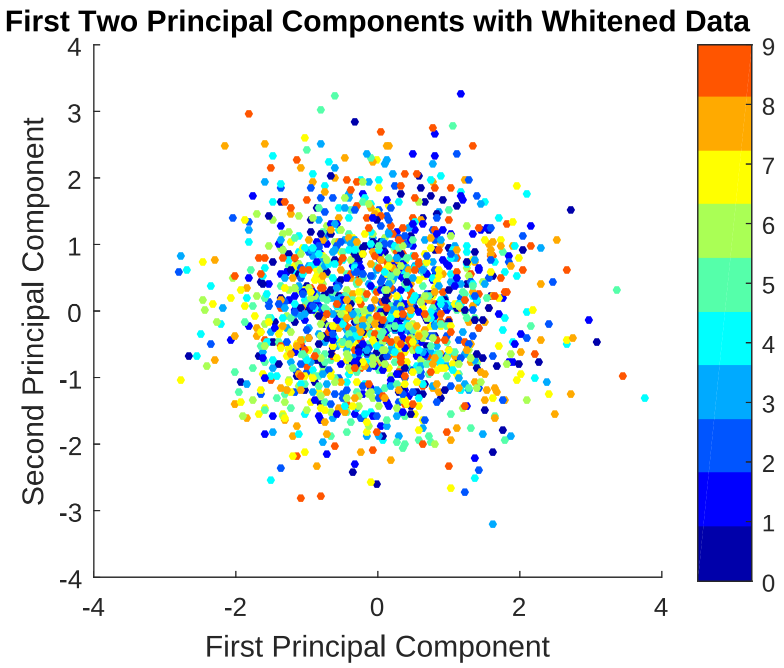

Figure 3: First Two Principal Components computed after whitening of data and centering.

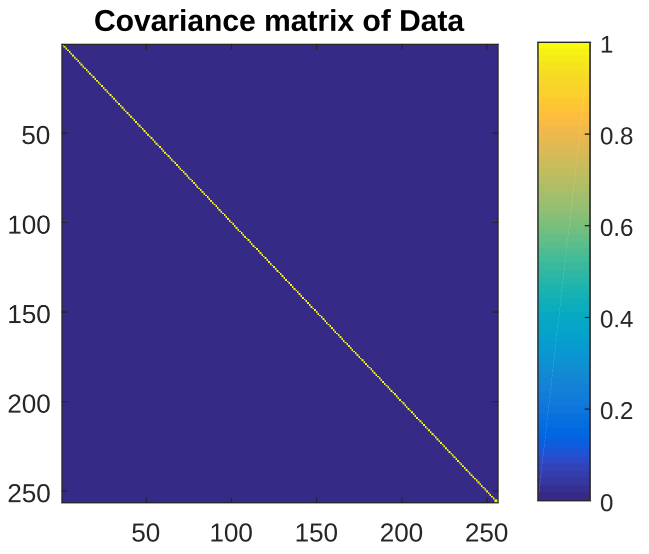

Figure 4: Covariance matrix after the whitening of data.

After this short digression on whitening we now see how the K-nn algorithm works when tested on whitened data using all the dimensions available (256). The number of instances has been incremented to 50 to gather more data. As we can see in Figure 5 performances are very bad.

Whitening is a very used preprocessing step of data to decorrelate features, but in this case it is not effective to the task of classification, even if usually decorrelating features may separate and improve distinction between classes allowing better results during the classification step.

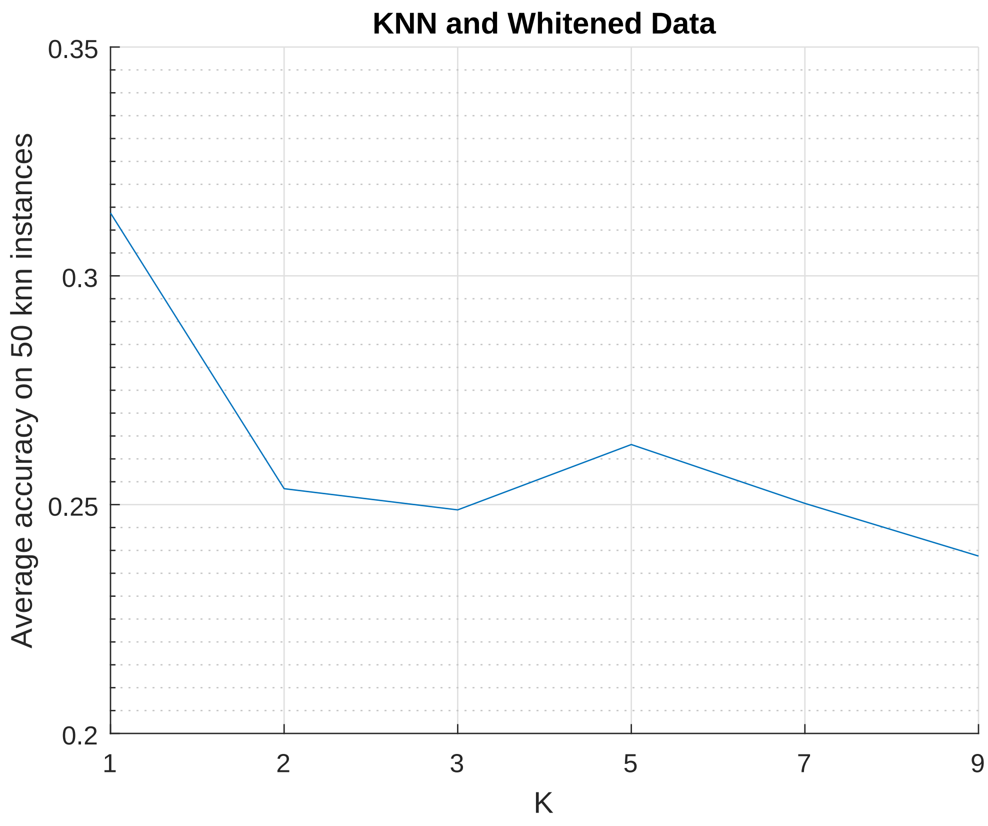

Figure 5: Performance of K-nn algorithm as K changes, with all features.

Task 4 In the last task we were asked to test the performances of K-nn algorithm applying Independent Component Analysis. As we can see in Figure 6 K-nn applied to the full set of Independent Components give very bad results. This somehow respects our expectations since we said that PCA performs well in presence of Gaussianity but ICA doesn't, in fact this might be the proof of the theory if we assume Gaussianity in data.

An interesting thing is that there are some similarities between the results obtained from the Task 3 and the latter one, maybe because whitening decorrelates features and ICA makes them independent. Even if the two concepts are different, there might have some similarities that result in similar performances during the classification phase.

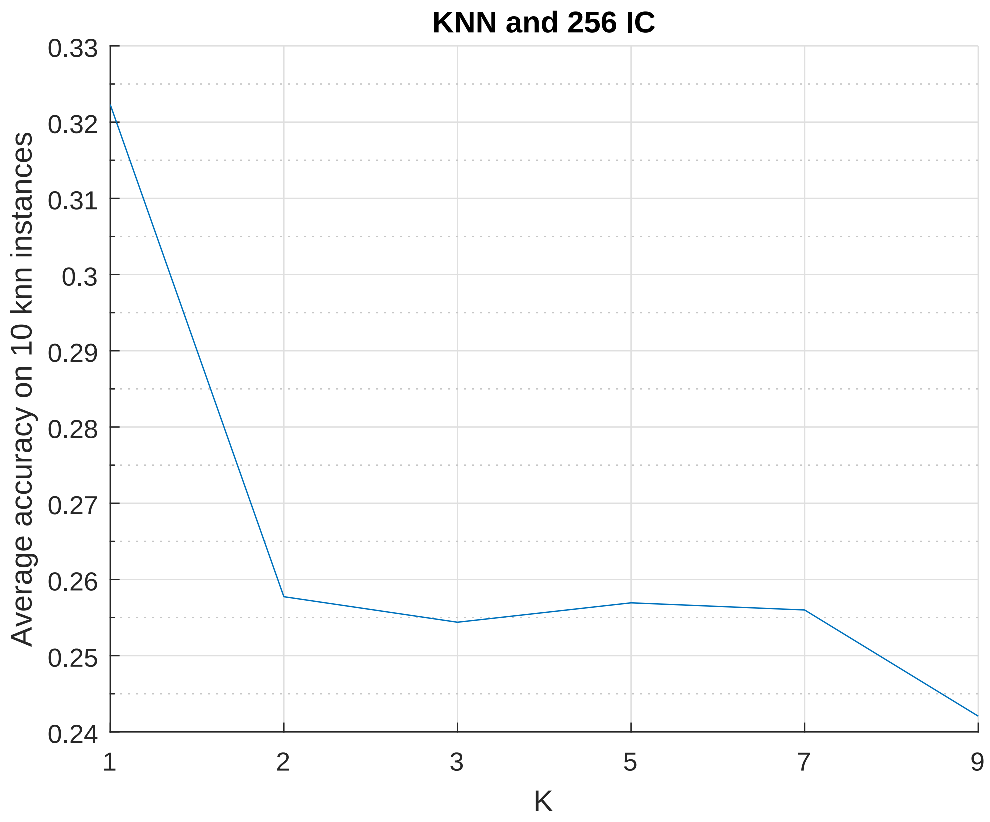

Figure 6: Performance of K-nn algorithm with 256 Indipendent Components as K varies.

Student: Francesco Pelosin
Email: 839220@stud.unive.it *OR* pelosinfrancesco@live.it
Id: 839220