Independent Component Analysis

Sometimes, it's useful to process the data in order to extract components that are uncorrelated and independent. To better understand this scenario, let's suppose that we record two people while they sing different songs. The result is clearly very noisy, but we know that the stochastic signal could be decomposed into the following:

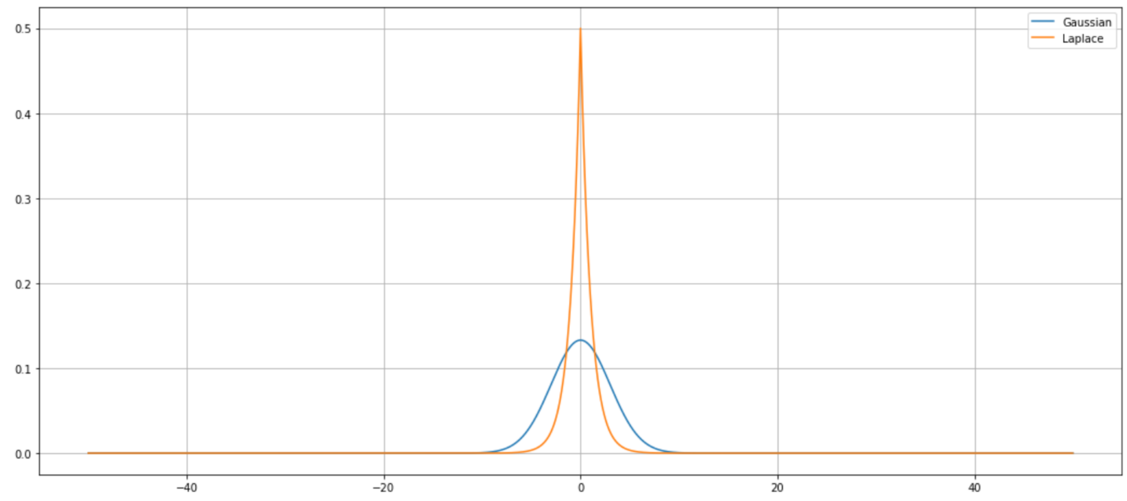
$$s(t) = s_1(t) + s_2(t) + n(t)$$

The first two terms are the single music sources (modeled as stochastic processes), while *$n(t)$* is additive Gaussian noise. Our goal is to find *$s_1(t) + n_1(t)$* and *$s_2(t) + n_1(t)$* in order to remove one of the two sources (with a part of the additive noise that cannot be filtered out). Performing this task using a standard PCA is very difficult because there are no constraints on the independence of the components. This problem has been widely studied by Hyvärinen and Oja (please refer to *Independent Component Analysis: Algorithms and Applications*, *Hyvarinen A.*, *Oja E.*, *Neural Networks 13/2000*), who considered the conditions that must be enforced in order to extract independent components. The most important thing is that we cannot assume Gaussian distributions, but instead, we need to look for more peaked ones (with heavy tails). The reason is based on the statistical property of the Gaussian distribution and, in particular, we know that if some random variables $s_i \sim N(\mu, \sigma)$ are independent, they are also jointly normally distributed (that is, $(s_1, s_2, ..., s_i) \sim N(\mu, \Sigma)$).

Considering the previous example, this result implies that if s_1 and s_2 are assumed to be independent and normal, the additive signal will be represented by a multivariate normal distribution. Unfortunately, this result makes the research of independent components impossible. As this book is more introductory, I omit the proofs (which can be rather complex, but can be found in the aforementioned paper and in *Mastering Machine Learning Algorithms*, *Bonaccorso G.*, *Packt Publishing*, *2018*). However, in order to understand the logic, it's helpful to consider a statistical index called **Kurtosis** (which corresponds to the fourth moment):

$$Kurt(X) = E_{x \sim X}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^4\right]$$

This measure is normally referred to as a normal distribution, whose value is *Kurt(N)* = *3*. All the distributions with a Kurtosis greater than *3* are called super-gaussian and they are very peaked around the mean. This implies that the probability is high, but only by a very small region, and it's close to zero elsewhere. An example of this is shown in the following graph:



Gaussian distribution versus Laplace distribution

The Laplace distribution is an example of super-gaussian distribution with *Kurtosis* = *6*. As it's possible to see, it's very peaked around the mean and the tails are heavy. The probability density function is as follows:

$$p_{Laplace}(x) = \frac{1}{2\beta}e^{-\frac{|x-\mu|}{\beta}}$$

Hence, the exponent now has degree *1* (while Gaussians are squared), and the joint probability of different components can never be represented as a Gaussian distribution. Assuming that the sources are modeled with such kinds of distributions (there are many other alternatives) means that we are implicitly saying that they are separable and independent (this also forces sparsity). Hence, if the final samples are obtained with super-gaussian additive contributions, such an approach can help us in identifying and isolating the sources because they are no more jointly normally distributed and, moreover, the probability distributions of the single component have a potential smaller overlap.

However, even if this measure is very powerful, it's extremely sensitive to outliers (due to the fourth power), and therefore the authors proposed an alternative algorithm that they called **Fast Independent Component Analysis** (**FastICA**). A full description of the mathematical background is beyond the scope of this book, but I want to show how to extract 256 independent components from the original MNIST dataset (which can be downloaded by using the scikit-learn built-in `fetch_mldata('MNIST original')` function).

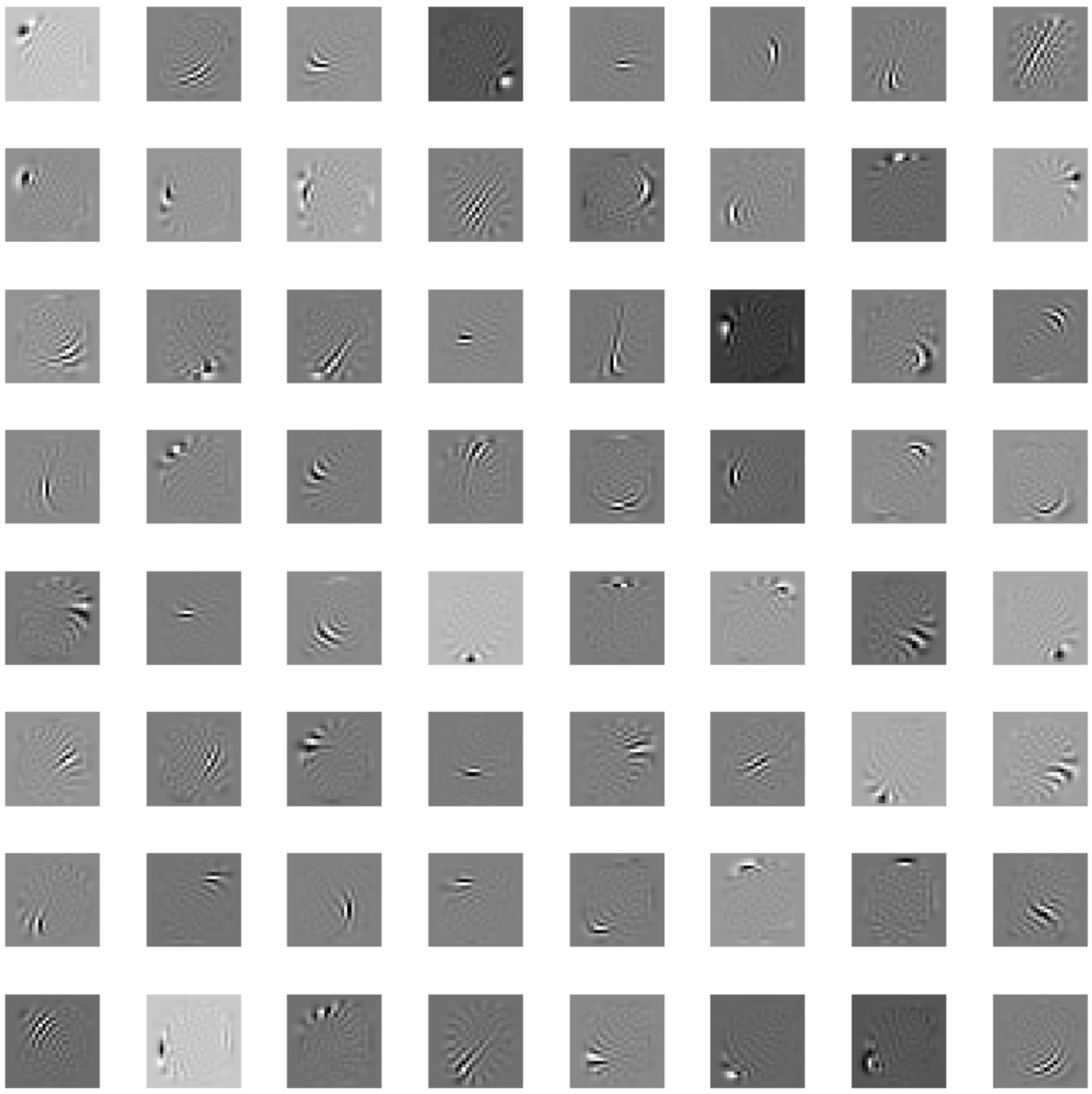
The first step is loading and zero-centering the dataset (this algorithm is very sensitive to symmetric data):

```
import numpy as np

from sklearn.datasets import fetch_mldata

def zero_center(Xd):
    return Xd - np.mean(Xd, axis=0)

digits = fetch_mldata('MNIST original')
X = zero_center(digits['data'].astype(np.float64))
np.random.shuffle(X)
```

At this point, we can instantiate the `FastICA` class, selecting `n_components=256` and `max_iter=5000` (sometimes, it's necessary to raise this value to guarantee the convergence), and train the model:

```
from sklearn.decomposition import FastICA

fastica = FastICA(n_components=256, max_iter=5000, random_state=1000)
fastica.fit(X)
```

At the end of the process, the `components_` instance variable contains all the extracted values. In the following screenshot, we can see the representation of the first 64 independent components:



64 independent components extracted from the MNIST dataset

Contrary to PCA, we can now detect many small *building blocks* that represent portions of the digits that can be shared among many of them. For example, it's possible to detect small horizontal strokes, as well as several rounded angles in different positions. These components are almost independent, and a sample can be rebuilt as a weighted sum of them. In particular, when the number of components is very large, the resultant weight vector will be sparse because many components won't be used in all of the samples. I invite the reader to repeat this exercise with different `n_component` values, trying to understand when the components contain overlaps (so they are made up of a group of independent parts) and when they begin to become *basic elements*. From a neuroscientific viewpoint, the fundamental mechanism employed by the brain to decode images is based on neurons that are only receptive to specific patterns. Therefore, using this algorithm, it is possible to implement a filtering mechanism that splits a complex source (for example, an image) into blocks that can elicit a selective response. For example, it's possible to analyze a dataset and exclude all of those samples where a particular component has a large weight (such as background noises or echoes) or, conversely, it's possible to select only those elements that are helpful for a specific task (for example, containing strong, vertical components).