Sparse Coding and Dictionary Learning have become important tools machine learning and data representation recently. It has a well-developed mathematical foundation and has been applied to solve a wide range of problems, from signal processing to computer vision and language understanding.

In this exercise, you will get hand-on experience with both dictionary learning and sparse coding, with the help from built-in functions from `sklearn`.

1 Theory

Sparse coding is a class of unsupervised methods to represent data efficiently. Sparse coding is data dependant. Unlike PCA which represents the data using a complete set of basis, Sparse coding represents the data using an over-complete set of basis. This representation is often constrained to be sparse, meaning that only a few number of features are non-zero.

Sparse coding often goes with dictionary learning stage. While the former finds the data representations, the latter learns the most suitable dictionary or set of basis for the representations.

2 Python Exercise

In this exercise, you will use Dictionary Learning and Sparse Coding to image compression and reconstruction. You will make full use of existing functions in `sklearn`.

Step 1: Load the image We will use the *face* image, available inside the `scipy` package for this exercise. For your convenience, the code for this step is provided. To reduce computational consumption, we will first resize the image to half its dimensions.

Step 2: Sample image patches As the size of the whole image is 384×512, performing dictionary learning and sparse coding on the whole image is impractically expensive. As a result, we will work with small image patches.

In this step, you need to sample a lot of patches from the image to use in learning the dictionary. You can use the function `extract_patches_2d` from `feature_extraction.image` to do this sampling. We will sample 30000 patches of size 4×4 in this step.

Step 3: Normalize the patches for learning dictionary In this step, you need to:

- First, flatten the patches, from 2D arrays to 1D vectors; save all of them as a Numpy array X
- Second, convert X to `float` type
- Third, normalize X by subtract mean and divide by its standard deviation

Step 4: Learn the dictionary We are now all set to learn the dictionary from the patches. For efficiency reason, we will use the function `MiniBatchDictionaryLearning` from `sklearn`. As the number of features is $4 \times 4 = 16$, we will use 64 components for the dictionary (to be over-complete).

Besides, let us set the number of iterations to 500, number of of non-zero components to 6 and use the *least angle regression method* as the fitting algorithm. After initializing the learner, call the `fit` function to learn the dictionary.

Step 5: Prepare patches for Sparse coding experiment In this step, we use the provided utility function, `sample_patches` to sample non-overlapping patches from the image. We preprocess these patches by subtracting the mean values.

Step 6: Run Sparse coding experiment In this step, you need to loop over different sparsity levels, find the sparse representations of the patches using the learned dictionary, and reconstruct the original image. Follow the steps in the code. You should be able to observe the gradual quality improvement of the reconstructed image when more entries are non-zero.