Mini-Project (ML for Time Series) - MVA 2021/2022

Antonio LOISON antonio.loison@student-cs.fr
Florian LE BRONNEC f.le-bronnec@student-cs.fr

June 26, 2022

1 Introduction

1.1 Introduction

K-SVD is an algorithm for dictionary learning with sparsity constraints. There are usually two kinds of approaches for sparse coding: either encode a signal with a user-defined dictionary or learn a new one on a specified dataset. K-SVD adopts the second approach. K-SVD's goal is to iteratively build the dictionary and the associated coefficients. Unlike classical dictionary learning algorithms, its specificity is that the coefficients and dictionary update steps are not decoupled, we will describe this with more details later.

Despite that K-SVD has been proposed 16 years ago, this algorithm is still a common baseline in different applications of signal processing. A vast majority of K-SVD's applications are in image processing. Therefore in this project, our goal will be to apply this method to time series analysis, with a special focus on denoising.

1.2 Contributions

The Python implementation was done by Antonio and optimized by Florian. The theoretical description of the algorithm was done by Florian. We selected the datasets together. Antonio investigated the respiratory sound datasets, with application to denoising and Florian the stars' lightcurves one, with particular focus on the extracted features.

Coding policy K-SVD algorithm has no implementations in classical libraries such as Scikit-Learn. We coded the algorithm framework ourselves but relied on existing implementation for the sparse coding problem.

Experiments We will present application to denoising and features extraction, with an evaluation of the extraction using a downstream classification task, with a particular attention on the kind of dataset we use.

2 Method

2.1 Problem statement

We quickly recall the dictionary learning problem. We will try to stick as much as possible with the notations adopted in the MVA Time Series course. Let $N \in \mathbb{N}$ be the size of the signal and $K \in \mathbb{N}$ the number of atoms in the dictionary.

In the whole project, we will note the matrix with capital letters and their associated columns or row with small letters. Columns of a matrix will be indexed with subscripts and rows with superscripts (if $\mathbf{A}$ is a matrix, $\mathbf{a}_i$ will designate the i^{th} column of $\mathbf{A}$ and $\mathbf{a}^j$ its j^{th} row). We will note the Frobenius norm on matrices and the euclidean norm on vectors $\|\cdot\|_2^2$.

K-SVD aims at finding the best approximation of a signal $\mathbf{x} \in \mathbb{R}^N$ such that $\mathbf{x} \approx \mathbf{D}\mathbf{z}$, where $\mathbf{D} = \left[\mathbf{d}_1, \dots \mathbf{d}_K\right]$ is the matrix with the atoms $\mathbf{d}_k$ as columns and $\mathbf{z}$ is a sparse vector. In practice, we will often be working either with the trajectory matrix of a signal or with several time series at once. We can hence write this problem in matrix notations, with p the number of time series:

$$\underset{\mathbf{Z}\in\mathbb{R}^{K\times p},\, \mathbf{D}\in\mathbb{R}^{N\times K}}{\arg\min} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_2^2, \quad \text{subject to} \quad \forall i \in [\![1, p]\!],\ \|\mathbf{z}_i\|_0 \leq K_0. \tag{1}$$

2.2 K-SVD algorithm

As this problem is computationally intractable, the K-SVD algorithm proposes, as always, an approximate resolution of the problem. As in the majority of dictionary learning algorithms, it is solved iteratively by successively solving the sparse coding problem with $\mathbf{D}$ fixed and then updating $\mathbf{D}$ and refining $\mathbf{Z}$.

2.2.1 Sparse coding

In this step we suppose that $\mathbf{D}$ is fixed. We can use any sparse coding algorithm to find the best sparse encoding $\mathbf{Z}$ of $\mathbf{X}$, using any common algorithm. For efficiency reasons, in this project we will exclusively use the *Orthogonal Matching Pursuit* that we described during the class (it is also the one recommended by the authors of K-SVD), and is fully implemented and optimized in Scikit-Learn.

2.2.2 Dictionary update

Here readers familiar with dictionary learning methods can expect that for this step we suppose that $\mathbf{Z}$ is fixed. K-SVD algorithm does not. And in fact, relying on this assumption makes the update of the dictionary more accurate, as it has more expressive power to fit to the data.

However, instead of updating both the entire $\mathbf{D}$ and $\mathbf{Z}$, we will focus on one column of $\mathbf{D}$ at a time. So let's choose a column $\mathbf{d}_k$ of $\mathbf{D}$. $\mathbf{z}^k$ hence designates the associated row of $\mathbf{Z}$ with the coefficients of $\mathbf{d}_k$ for the reconstruction of each time series $\mathbf{x}_k$. And assume that all the others $\mathbf{d}_i$ and $\mathbf{z}^i$ are fixed for $i \neq k$. We can rewrite the squared norm of the error using a sum of rank-1 matrices:

$$\|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_2^2 = \left\|\mathbf{X} - \sum_{i=1}^{K} \mathbf{d}_i \mathbf{z}^i\right\|_2^2 = \left\|\mathbf{X} - \sum_{i \neq k} \mathbf{d}_i \mathbf{z}^i + \mathbf{d}_k \mathbf{z}^k\right\|_2^2 = \left\|\mathbf{E}_k - \mathbf{d}_k \mathbf{z}^k\right\|_2^2. \tag{2}$$

Here, solving (2) is straightforward using the SVD of $\mathbf{E}_k$. But doing that won't provide any guarantee on the sparsity of the solution.

Therefore, we are going to force the solution of the minimization $\widehat{\mathbf{z}}^k$ to have the same support as $\mathbf{z}^k$.

Assume that $\left\|\mathbf{z}^k\right\|_0 = S_0$. We introduce the *truncating matrix* $\mathbf{\Omega}_k$, that is the matrix in $\mathbb{R}^{K \times S_0}$ of the linear form that associates to a row vector its coefficients corresponding to the positions where $\mathbf{z}^k$ is non zero. So it amounts at solving the problem:

$$\underset{\mathbf{z}^k \in \mathbb{R}^p,\, \mathbf{d}_k \in \mathbb{R}^N}{\arg\min} \left\|\mathbf{E}_k \mathbf{\Omega}_k - \mathbf{d}_k \mathbf{z}^k \mathbf{\Omega}_k\right\|_2^2 \iff \underset{\widetilde{\mathbf{z}}^K \in \mathbb{R}^{S_0},\, \mathbf{d}_k \in \mathbb{R}^N}{\arg\min} \left\|\mathbf{E}_k \mathbf{\Omega}_k - \mathbf{d}_k \widetilde{\mathbf{z}}_r^k\right\|_2^2, \tag{3}$$

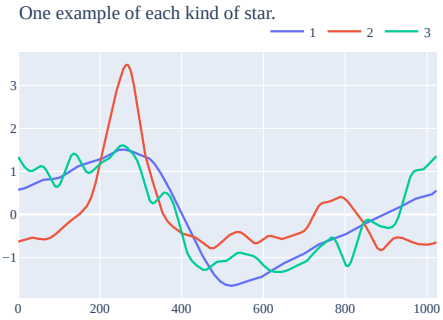
which optimal solution is known thanks to SVD of $\mathbf{E}_k \mathbf{\Omega}_k = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$: $\widehat{\mathbf{d}}_k$ is the first column of $\mathbf{U}$ and $\widehat{\mathbf{z}}_R^k$ is the first column of $\mathbf{V}$ multiplied by that first diagonal coefficient of $\mathbf{\Delta}$ (best rank one approximation of $\mathbf{E}_k \mathbf{\Omega}_k$ for the L_2 norm).

And we repeat the process for each column of $\mathbf{D}$. A big advantage of these updates is that we can implement them with pure efficient Numpy operations.

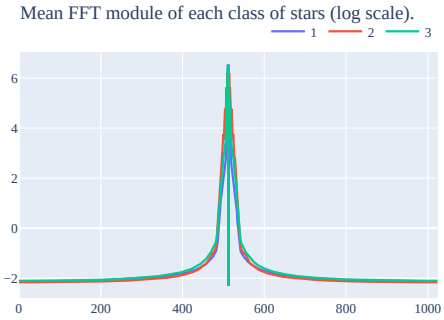
Once all the columns have been updated, return to the sparse coding step and iterate. Note that after updating the columns of $\mathbf{D}$ we could continue to cycle through the columns without the sparse coding step. However, doing the sparse coding step after all the columns have been updated once greatly improves the result.

3 Data

We used different kinds of data for our applications. The first one we used are light curves emitted from stars. This dataset consists of 1000 time series of fixed length 1024 corresponding to 3 kind of periodic lights emitted by stars (the period were rescaled in order to have same length). The kind of signals we have are represented on fig. 1. We see that visually the stars seems distinguishable but it can be quite hard for a computer to do so.



(a) Three examples of light curves corresponding to each kind of star.



(b) Mean FFT module of each class.

Figure 1: Example of signals in the stars dataset

Indeed, while the red curve on fig. 1 seems to have different "peaks", the green and blue ones are quite similar and a naive estimate based only on the value of each star at each time step might not be sufficient. On the other hand, the green curve seems to have a lot more high frequencies than the other ones. But in fact, on average their frequency content are not so different. Another important point is that these data were heavily processed beforehand with methods specific to astrophysics (outliers removal, smoothing, normalization and interpolation). But the data may still contain noise, due to the difficult process of astrophysics data acquisition.

We also used a dataset with more complex time series of respiratory sounds. With this dataset, we could see the limits of the KSVD algorithm when the dataset is richer in frequencies. For computational reasons, we only worked with "Al"-labeled signals, that are measures taken from the anterior left part of the chest. This gave us 162 signals. Most of them were of length 17640 but some could be smaller with length down to 1000. We cut these signals into subsignals of length 600 (that more or less correspond to 3 respiratory periods).

4 Results

We decided to investigate some properties of K-SVD algorithm. First, we observe the behavior regarding the K-SVD algorithm that one should be aware of when using. Then we observe the performance of K-SVD on denoising. Then, we present a way of doing classification with K-SVD. For all the examples, we initialized $\mathbf{D}$ with a vectors from the Discrete Cosine Transform.

4.1 Convergence

At each step of the algorithm the MSE decrease, hence we have a convergence guarantee of the reconstruction error. However it will only converge to a local minimum. And we do not have any guarantee on the atoms' convergence. The reconstruction error and the MSE between two dictionaries from successive iterations are plotted on fig. 5 for a simple noisy sinus.

4.2 Denoising

4.2.1 Gaussian Denoising

As said above, we noticed interesting denoising capabilities of K-SVD. To illustrate that, we used a signal with a varying frequency, and a gaussian noise. The result is displayed on fig. 6 and is rather satisfying.

4.2.2 Salt and Pepper Denoising

In the original article the authors illustrated the denosing power of K-SVD on images. We wanted here to see how it can perform on audio data. For this experiment, inspired by the reconstruction experiment of the

paper, we take the 4000 sub-signals of size 600 from 150 signals of respiratory sounds and trained a dictionary on them with K-SVD. The optimal parameters where determined through a grid-search. We ended with a dictionary of 300 atoms with 20 coefficients.

Then we added 50% of outlier values to 12 new signals that were not in the training set and we tried to recover the original signals. On this test set, we ended up with an average MSE of **0.445** compared to **0.239** when the signal has no outliers. An example of recovery is given on fig. 2.

It has interesting performances because it a bit more competitive than a median filter, which gives an average MSE of **0.602**.

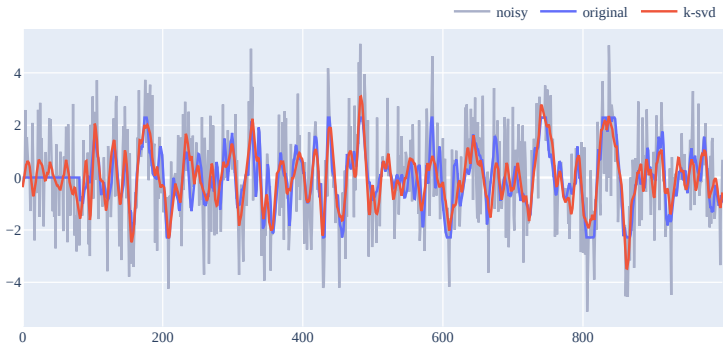


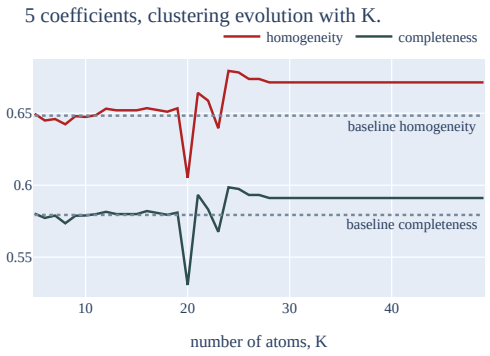
Figure 2: Recovery of signal with outliers. 50% of the points are noisy. MSE $= 0.374$.

.

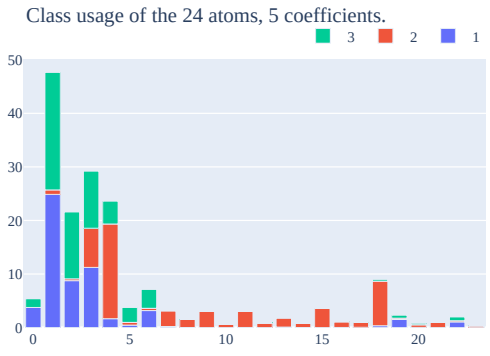
4.3 Classification

Our goal was to see how meaningful are the features retrieved by K-SVD. We conduct this task the light curve dataset. We wanted to make a fully unsupervised experiment. To this end, we defined a baseline which the clustering returned by KMeans algorithm ran on the raw 1000 time series of length 1024 (each time series is seen as a vector in $\mathbb{R}^{1024}$. An interesting experiment is to monitor how the class information are available through the learnt encoding.

We fixed a number of atoms of 5 and we monitored the clustering result ran on the $\mathbf{Z}$ coefficients matrix (evaluated through homogeneity and completeness score with respect to the ground truth). We plotted some information relative to this clustering on fig. 3.



(a) Evolution of the quality of the clustering as the number of atoms grows. 5 coefficients per signal.



(b) Usage of atoms per class (sum of the absolute value of the coefficients of $\mathbf{Z}$ for each class).

Figure 3: It is interesting to see how having a small number of coefficients can make the algorithm learn more meaningful representation for the signals clustering. The usage of each atom seems well separated according classes, at least for classes red and green.

Some atoms are plotted in appendix, fig. 8. We plotted some reconstructed signals, on fig. 4. Here we have a huge compression of the signals. We only need to store 24 atoms and 5 coefficients per signal. But we have to keep in mind that here the signal we took was not extremely complicated (like it could have been with audio for example). Indeed, we conducted other experiments with signal very rich in terms of frequencies and the algorithm was not performing as good as with this dataset.

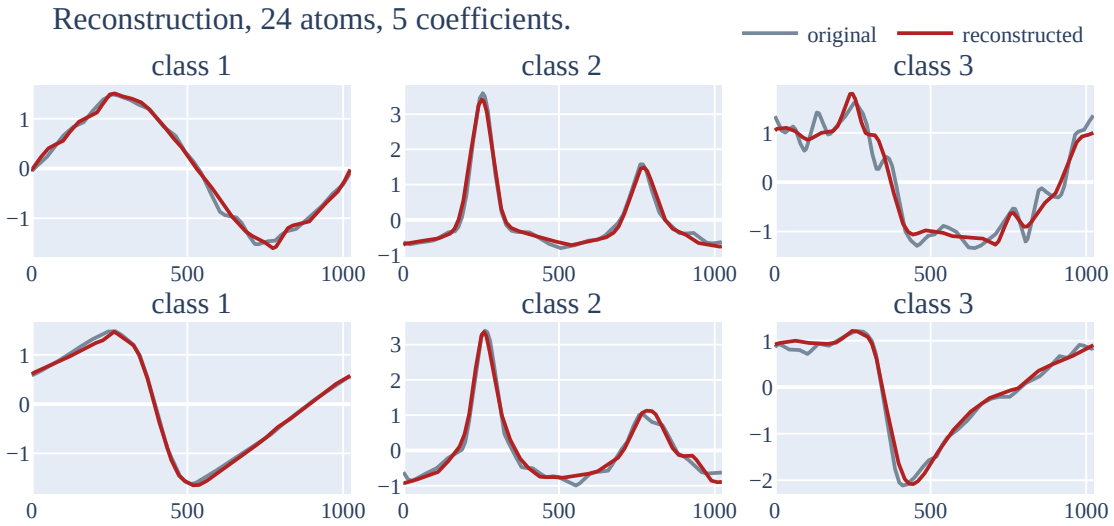


Figure 4: Reconstruction of some signals. MSE=0.0098 for class 1, 0.0331 for class 2, 0.0228 for class 3.

5 Conclusion

In this project we have investigated a lot of properties of K-SVD on various datasets. We have now some experience with this algorithm and we can make a quick recap of its strengths and weaknesses.

Strengths This algorithm provides a solution to sparse dictionary learning with a L_0 norm constraint, which is not a simple problem. It is then rather flexible because it can be used with any sparse coding algorithm. It performs also well on regular datasets. This is probably why it has been illustrated in the original article with images.

Weaknesses

- An obvious problem of the algorithm is it's speed. We have not talked a lot about it but it is really slow, due to the sparse coding which is quick for one iteration but not when we need to repeat it, and due to the SVD computation which can be quite long if $\mathbf{Z}$ is not extremely sparse. We cannot use multiprocessing because columns update depends on all the columns. It is quite funny to notice that in the original article in 2006 they mention that speed of algorithm is going to become less important in the future.
- It is not suited for very rich signals. All the applications we tested with audio or even EEG data were not concluding.
- K-SVD is not well suited for multivariate signals.

To conclude, we would advice to use this algorithm whenever you have rather simple and small datasets without too many noise, but to find alternatives if the tasks or the data to voluminous.

A Convergence

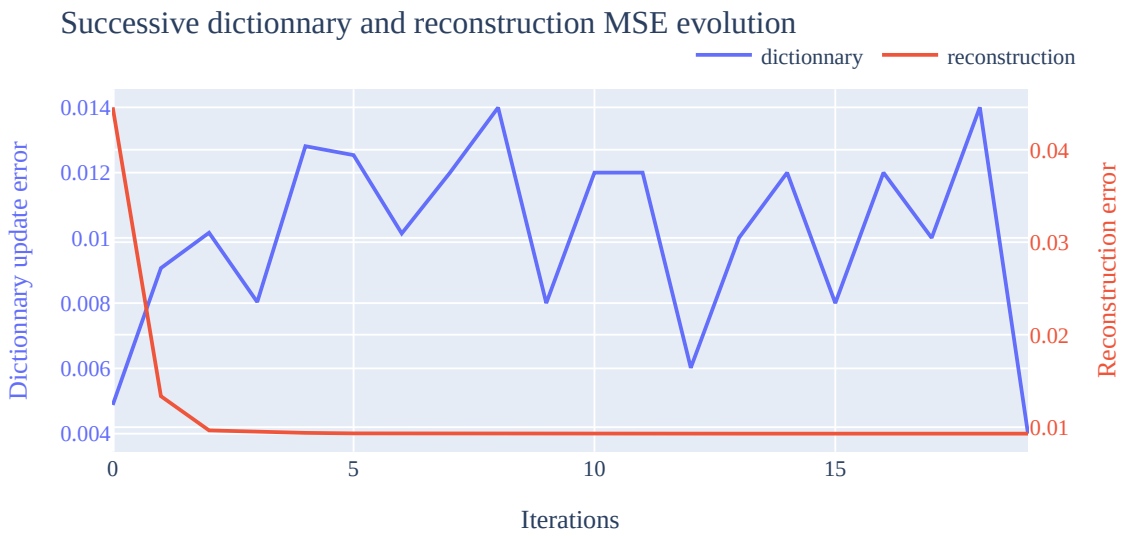


Figure 5: In blue the difference between successive dictionaries MSE($\mathbf{D}^{t+1} - \mathbf{D}^t$)

.

B Gaussian Denoising

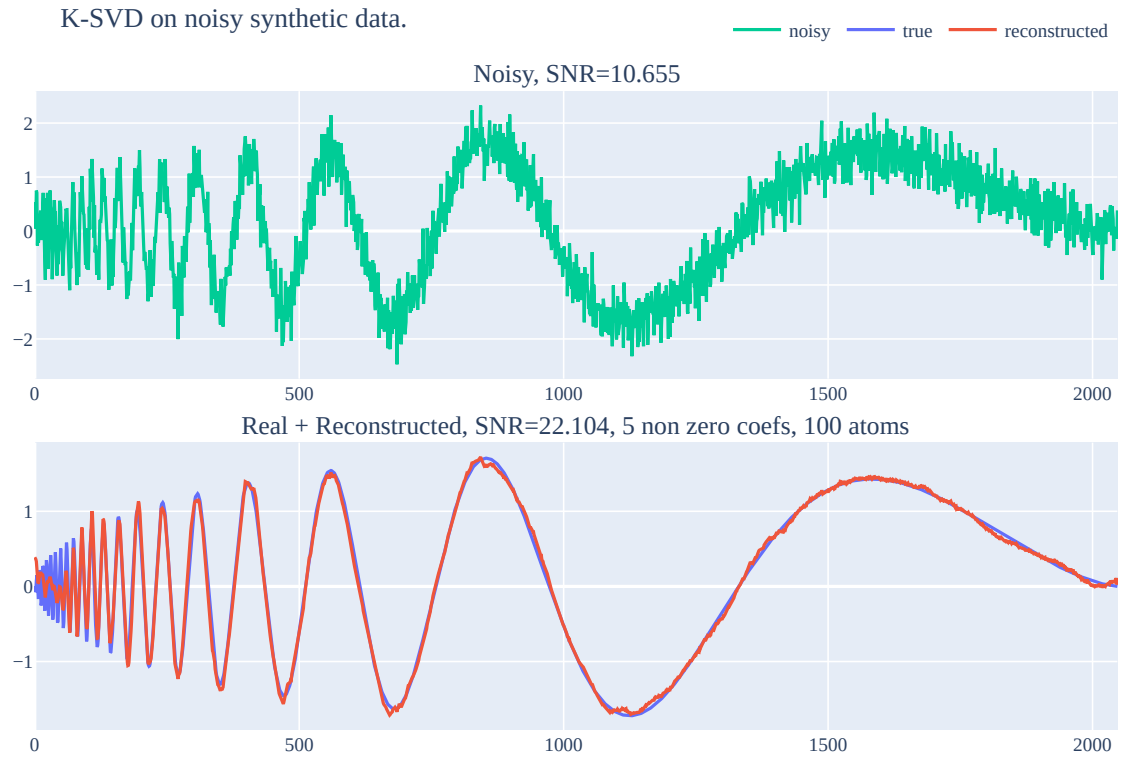


Figure 6: Gaussian noisy signal reconstruction.

C Outliers removal

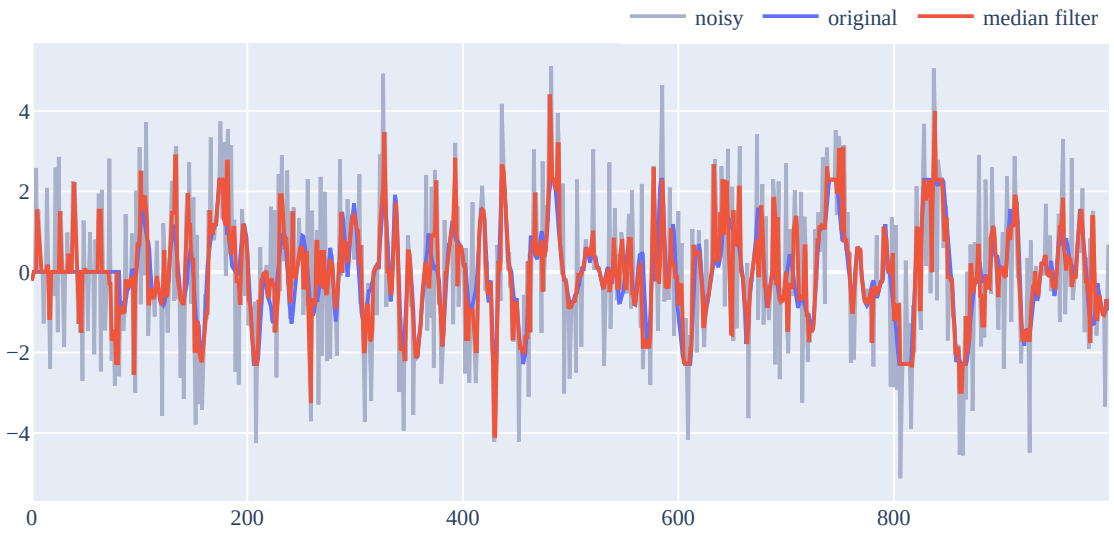


Figure 7: Median filtering, with a window size of 3. MSE = 0.490.

D Stars' lightcurve

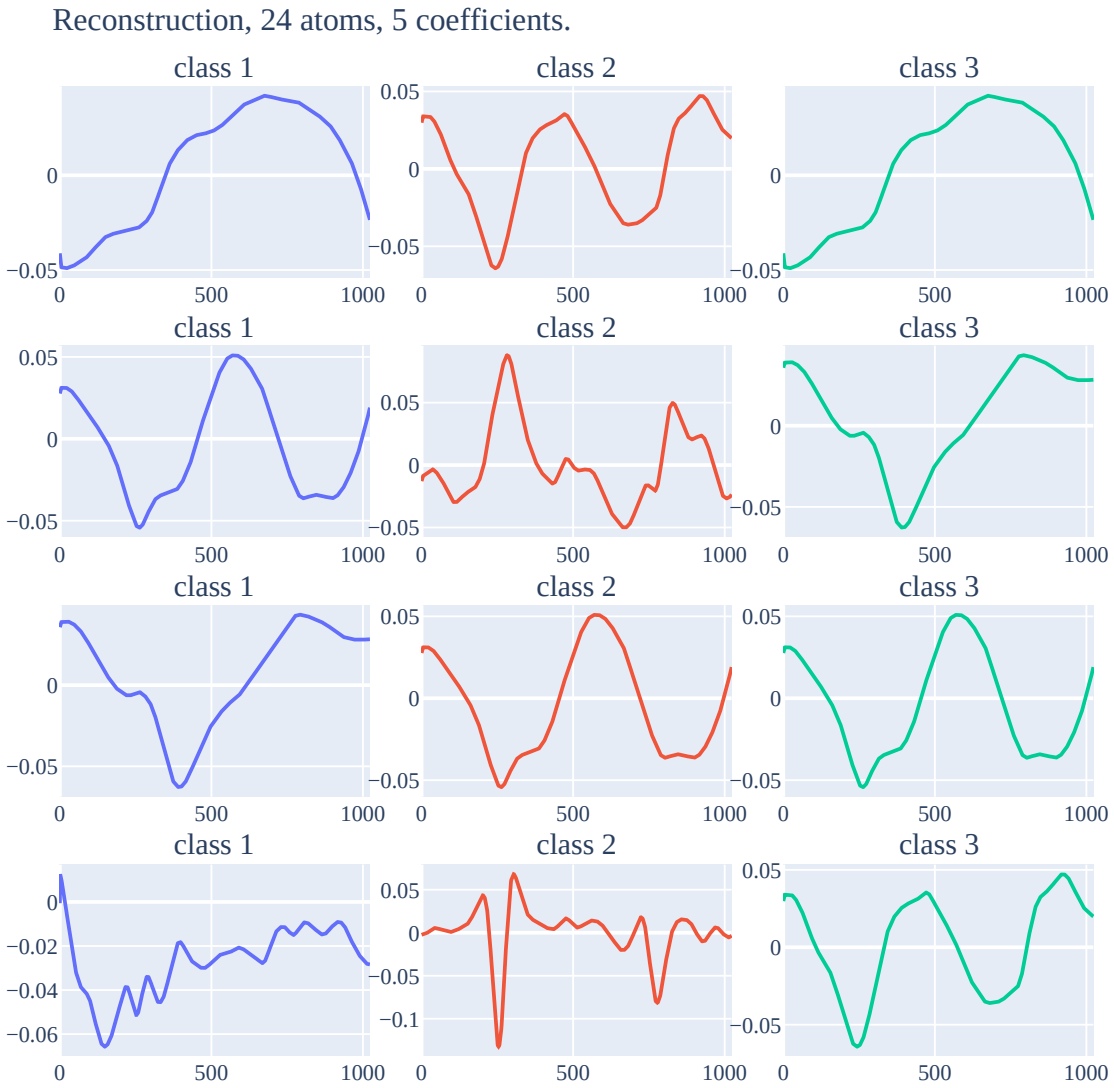


Figure 8: Most used atoms of each class.