Unsupervised Learning - MVA 2017/2018
Homework 2

Victor Busa, Steeve Vu, Rémi Lespinet

1 Algorithms implementations

1.1 Spectral Clustering

To compute the affinity, we use the symmetrized K-nearest neighbors gaussian affinity

$$w_{ij} = \begin{cases} \exp\left(-\|x_i - x_j\|\right) & \text{if } x_i \text{ and } x_j \text{ are K-neighbors} \\ 0 & \text{otherwise} \end{cases}$$

Where K-neighbors means that one of x_i is in the K nearest neighborhood of x_j or vice versa.

We l_2 normalize the datas before computing the affinity matrix, as it is discussed p.158 in [Vid16]. This allows us to choose σ around the unit value.

1.2 Sparse Subspace Clustering

For SSC We choose $\tau = 10 \times \tau_{min}$, where τ_{min} is defined in [E E09] as:

$$\tau_{min} \triangleq \frac{1}{\min\limits_{i} \max\limits_{j \neq i} |y_i^T y_j|}$$

We also had to normalize the matrix returned by the ADMM_LASSO algorithm using the infinity norm:

$$c_i \leftarrow \frac{c_i}{\|c_i\|_\infty}$$

before computing the Affinity matrix: $W = |C| + |C^T|$.

1.3 K-Subspaces

For the initialization of the subspace basis U, we compute a Singular Value Decomposition on a random matrix (where each coefficient is generated with a standard normal distribution). (We have tried other methods such as QR decomposition of random matrices and the dedicated

Scikit-learn method `rvs`, but this is slower since these methods generate full square matrices, and we only need a small number of orthonormal vectors of high dimension)

For the initialization of mu, we've tried several methods

- We first initialize mu with n points in the training data chosen at random (`init_naive`). We then tried to improve this using the $Kmeans++$ initialization (taking points iteratively with a probability depending on the distances to the set of points already chosen) (`init_kmeanspp`).
- We then tried to sample mu at random using a mutivariate gaussian random variable with mean and covariance matching the covariance of the data (`init_gaussian`).

2 Face clustering

2.1 Tuning the parameters

2.1.1 Tuning σ and k for Spectral Clustering

Working on individuals 1-2, we have tuned the parameters σ and k.

We first fixed $\sigma = 1$ and tune for $k \in [1, 128]$, the results are shown on Figure 1

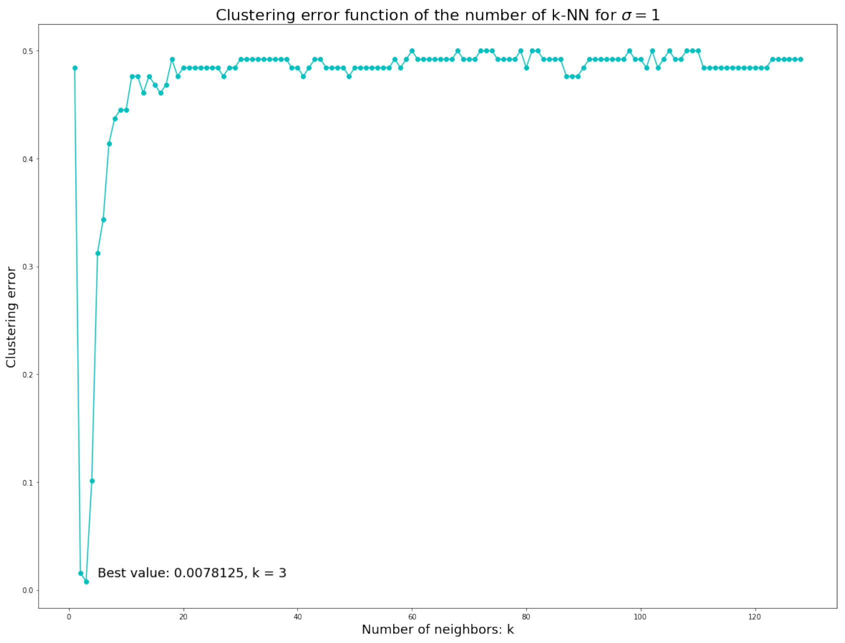

Figure 1: Clustering error in function of the number of k nearest-neighbors considered

As shown on Figure 1, the clustering error is minimal for $k = 3$. Letting $k = 3$ we then tune the parameter σ for σ in the range $[0.001, 10]$. The Figure 2 shows the clustering error in function of the parameter σ.

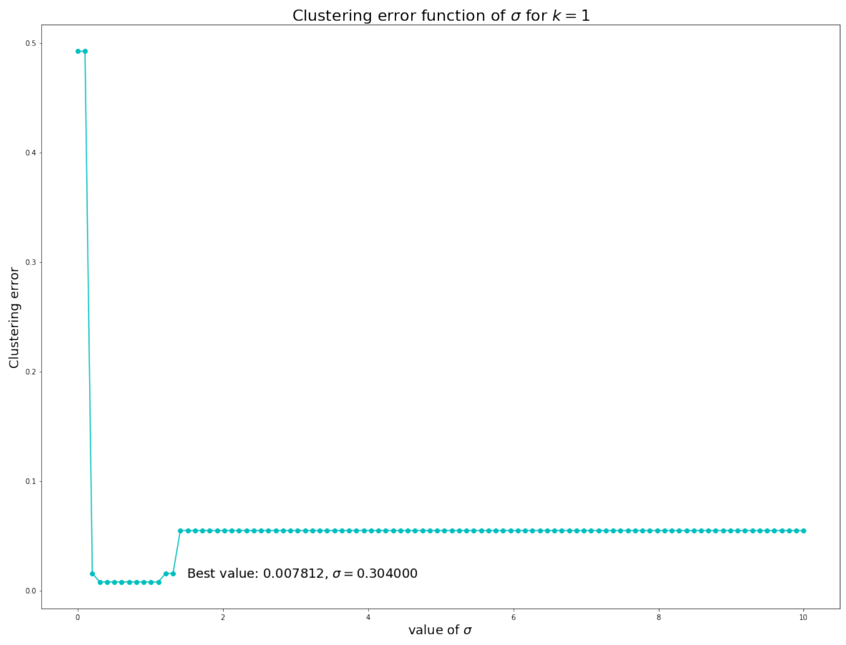

Figure 2: Clustering error function of σ for $k = 3$

2.1.2 Tuning number of restart and dimensions for K-subspaces

2.1.3 Varying the dimension of the subspaces

The figure 3 presents the clustering error as a function of the dimension of the subspaces (each subspace dimension argument is the same). The number of restart is fixed to 10.

It appears that choosing 9 for the dimension of each subspace gives best results. This agrees with the proposition that images of a face under different lighting conditions lye in a subspace of dimension 9. Taking a dimension of 3 also seems to give good results in most cases

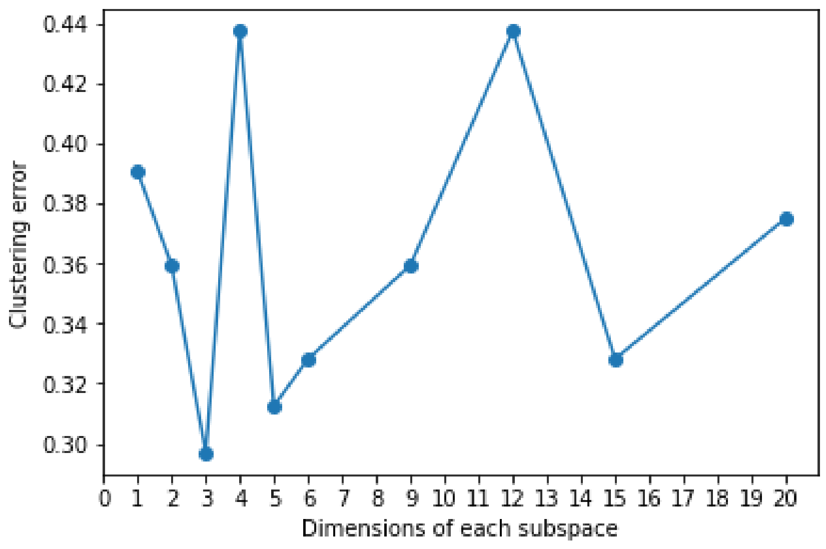

Figure 3: Clustering error for K-subspaces as a function of the dimensions on individual 1 and 2 of the dataset (fixed number of restart=10)

Varying the number of the replicates

In the figure 4, we present the results of the clustering error obtained on individual 1 and 2, when we vary the number of replicates (restarts). The dimension for each subspace is fixed to 9.

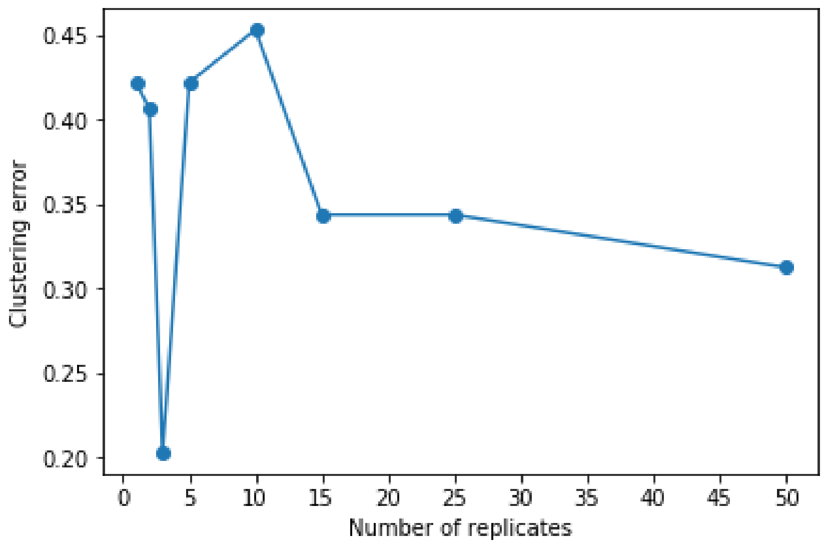

Figure 4: Clustering error as a function of the number of restarts for K-subspaces on individual 1 and 2 of the dataset (fixed number of dimension=9)

2.1.4 Tuning μ_2 for SSC

We fix τ as $\tau = 10 \times \tau_{min}$ as we emphasized in 1.2. We then tune the parameter μ_2 for μ_2 varying in $[1.10^1, 1.10^6]$ in a logarithmic way. The clustering error obtained is displayed on Figure 5.

2.2 Clustering error as a function of the number of groups

2.2.1 SSC

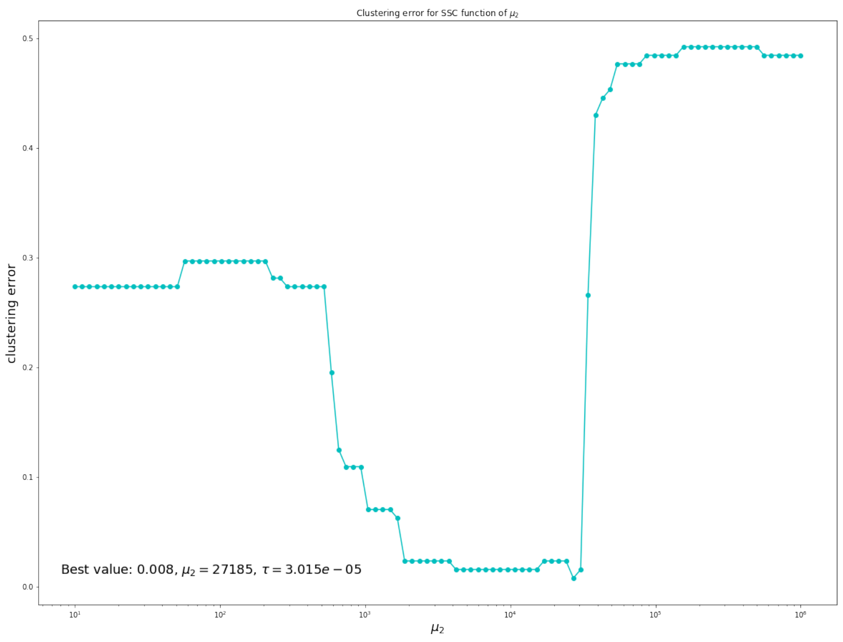

Figure 5: Clustering error function of μ_2

2.3 Results as a function of the number of groups

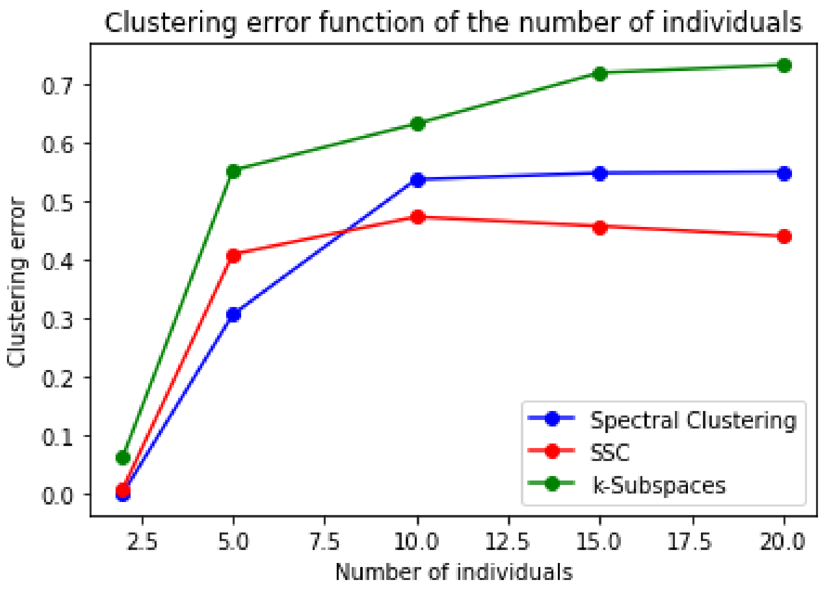

Figure 6: Clustering error as a function of the number of groups

3 Motion segmentation

We tuned the hyperparameters for each algorithm in order to get the minimal mean error rate for the 157 videos.

- **Spectral Clustering**: We used $k = 8$ nearest neighbors and $\sigma = 1$
- **K-subspaces**: For each subspace s we used $d_s = 3$ with 5 replicates
- **SSC**: Just like the section 2.1.4, we chose $\tau = 10 \times \tau_{min}$ with a $\mu_2 = 800$. Recall that τ_{min} depends on the data so it is not fixed.

Table 1: Performance for each algorithm on motion segmentation

Algorithm	Mean	Median	Standard deviation
Spectral Clustering	0.204	0.203	0.172
K-subspaces	0.0991	0.0379	0.1343
SSC	0.186	0.162	0.161

We observe that the results are better on this dataset than on the set of images. In particular K-subspaces give very good results (on more than 50% of the videos, the clustering error is less than 3%)

4 Organization of the work

We all wrote all of the algorithms on our side, so that we can easily share and debug our code. We then met to share our results, and write the report. (The 3 python notebook are included in the final archive)

It appears that SSC is not as good as expected on the dataset of images, we may have an error somewhere or we may not have chosen the right parameters.

A Note on the performance of K-Subspaces on the images

The spectral clustering algorithm is based on the assumption that the images in the same clusters form (or almost form) a convex component in the underlying graph. Note that this is wrong in our case, since the distance between images is the sum of square distances of its pixels, which is not necessarily smaller for 2 faces of the same individual than the one for 2 faces of 2 different individuals. To illustrate that, the figure 7 Shows an image (left) with the top 4 closer faces according to this distance in the database (and their label). We see that the closest images does not match the right individual, hence, the graph will most likely be corrupted.

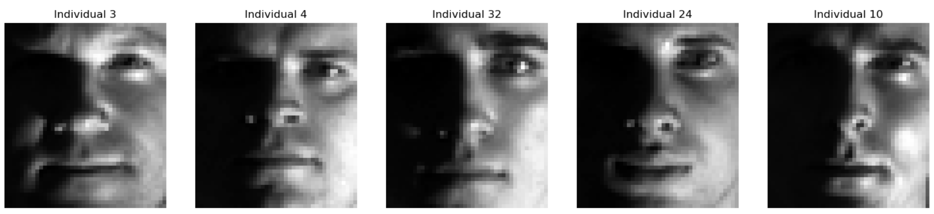

Figure 7: A data face (left) and its closest 4 faces (sorted from left to right) in the database according to the euclidian distance

We can also illustrate that by computing the affinity on the gradient of the images instead on the image directly, and we obtain better results, as shown in the figure 8

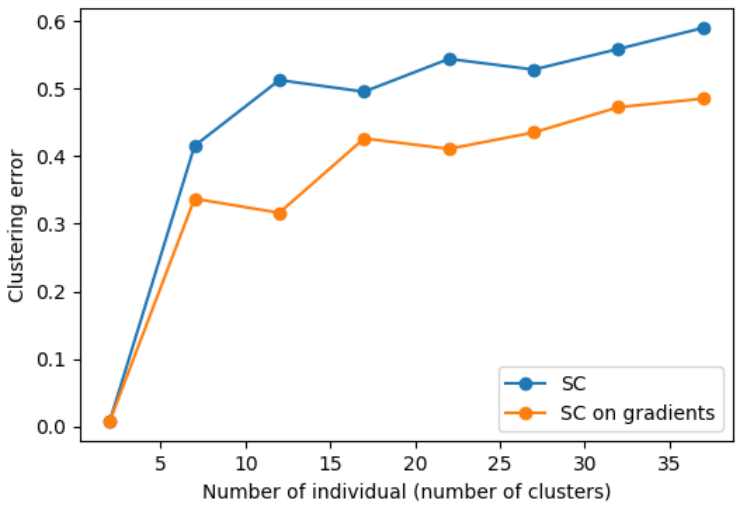

Figure 8: Comparison of the performances of the SC classifier applied on images, and on their gradients

B Note on the performance of SSC

Sparse Spectral Clustering performance doesn't match the performance highlighted p.343 in [Vid16]. This may be due to:

- The fact that we implemented SSC for Noisy Data
- We don't update the Lagrange multipliers at each iteration of the ADMM_LASSO algorithm

References

[E E09] R. Vidal E. Elhamifar. “Sparse Subspace Clustering: Algorithm, Theory, and Applications”. In: (2009).

[Vid16] R. Vidal. “Generalized Principal Component analysis”. In: (2016).