TP2 - report

By : Thibault Lahire (thibault.lahire@student.isae-supaero.fr)

1 Harmonic Function Solution (HSF)

1.1 Complete `hard_hfs` and `two_moons_hfs`. Select uniformly at random 4 labels (S), and compute the labels for the unlabeled nodes (T) using the hard-HFS formula. Plot the resulting labeling and the accuracy.

The constructed graph is a k-nn graph with $k = 6$. If there is at least one of the revealed labels in each class, then we get a perfect labelling with the hard-HFS function (accuracy = 1).

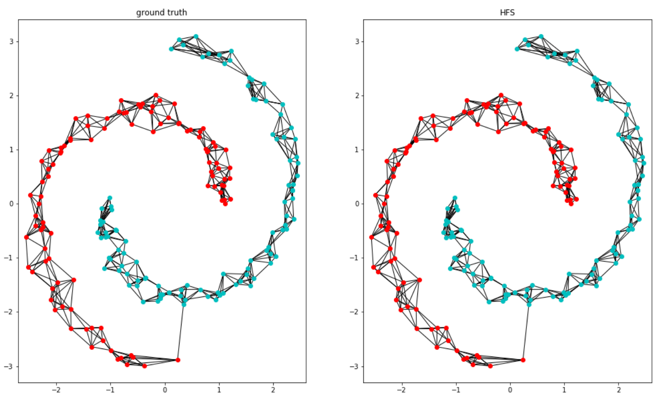

Figure 1: Accuracy of 1 for hard-HFS and this 6-nn construction

1.2 At home, run `two_moons_hfs` using the `data_2moons_large.mat`, a dataset with 1000 samples. Continue to uniformly sample only 4 labels. What can go wrong?

Change the beginning of the function `two_moons_hfs()` so that the function works on the large dataset.
The results on the large dataset are very similar to the previous results. However, there are two main differences. First, since the dataset is larger, the algorithm takes a bit more time to run. Second, for this larger dataset, the probability of revealing 4 labels that belong to the same class is higher. If this is the case, the accuracy drops to 0.5 : the algorithm believes that there exists one class only, so it attributes all the points to the same label.
However, in the case where all the revealed labels are in the same class, if we precise to the algorithm that there are two classes (which means modifying the code at the beginning of `two_moons_hfs()` in such a way it would not work in a general case), since the two classes are not inter-connected, the algorithm will reach an accuracy of 1. This is an interesting

fact, but this is cheating !

1.3 Complete `soft_hfs` and test it with `two_moons_hfs`. Now complete `hard_vs_soft_hfs`. Compare the results you obtain with soft-HFS and hard-HFS.

We wrote the function `soft_hfs` with the help of the formula given during the lesson: $f^* = (C^{-1}Q + I)^{-1}y$. The matrix C was built with the coefficient $c_l = 0.96$ and $c_u = 0.04$. I made this choice to highlight the fact that we are confident in the labeled points (so c_l is close to 1) and unconfident in the unlabeled points (so c_u is close to 0).

To test `soft_hfs` on `two_moons_hfs`, comment and decomment the corresponding lines in the function `two_moons_hfs`. We obtain very similar results, in other words accuracy$= 1$ both for small and large "two moons".

Finally, we compare the results we obtain with soft-HFS and hard-HFS thanks to `hard_vs_soft_hfs`. Run multiple times the program. Since there is some noise in the labeling, as seen in class, the soft-HFS is more robust and then gives (in general, but exceptions exist...) better results.

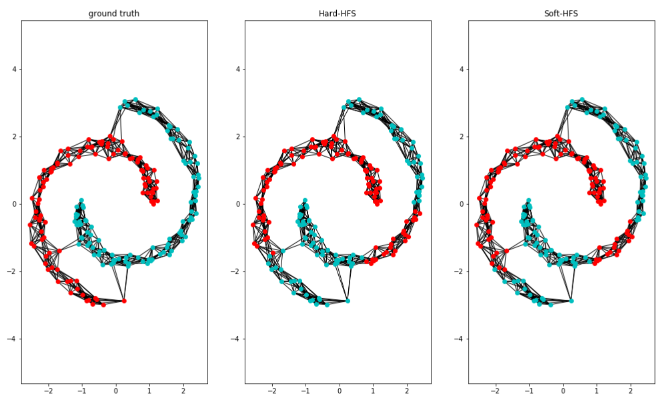

Figure 2: Accuracy $= [0.78; 0.835]$ for this random realization

2 Face recognition with HFS

2.1 How did you manage to label more than two classes?

To label more than two classes, we use one-hot encoding. For instance, if a point belongs to class 3, it will be encoded as the vector $[0, 0, 1]$ and no longer a float.

2.2 Which preprocessing steps (e.g. cv.GaussianBlur, cv.equalizeHist) did you apply to the faces before constructing the similarity graph? Which gave the best performance?

I have tested several configurations of preprocessing and I did not remark an increase in performance for one configuration compared to the others. When working on part 3 of this homework, I noticed that the preprocessing was done in the following manner : bilateral filter, then histogram equalization, then gaussian blur. I have tested this configuration for the code in this part, and I did not remark a huge increase in performances. However, let me explain why preprocessing steps is a good idea in general.

In the code given, the first function called is an histogram equalization, which means that we "adjust" all the given histograms in order to compare things that are comparable. The idea of histogram equalization is to apply a contrast change for which the target distribution probability is the uniform distribution on the dynamic of the considered image.

The second function called is the gaussian blur. If the cumulative histogram has some "pics", i.e. contains lots of pixels with a certain value, the gaussian blur will "add smoothness" to the distribution by giving to these pixels a different (but very close) value.

I have learnt this during the MVA lesson "Introduction à l'imagerie numérique" taught by Julie Delon and Yann Gousseau.

I remarked that adding a small gaussian blur to our images sometimes gives us a better performance, sometimes degrades it. Histogram equalization seems to give us more stable results, but this is not obvious to see...

2.3 Does HFS reach good performances on this task?

Roughly, on this task, we have an accuracy of 85%. I don't see a difference between hard-HFS and soft-HFS. The results are obtained for a k-nn construction with $k = 3$. When $k \geq 10$, the performance begins to decrease.

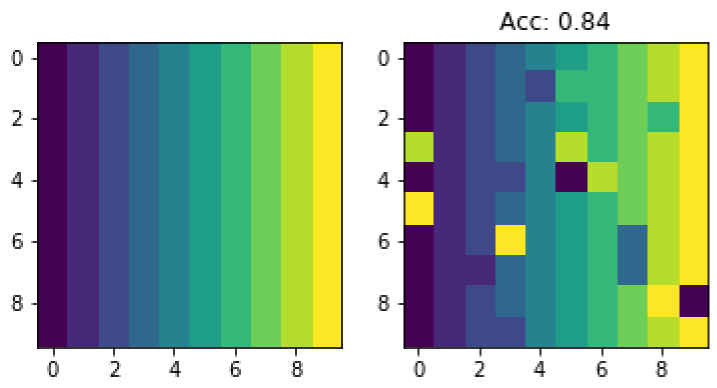

Figure 3: Results of the face recognition task for 10 people and 10 images per person. 4 images are revealed.

2.4 Did adding more data to the task improve performance? If so, which kind of additional data improves performance?

Adding the data given does not improve the performance. On the contrary, the accuracy has decreased and is now about 60%.

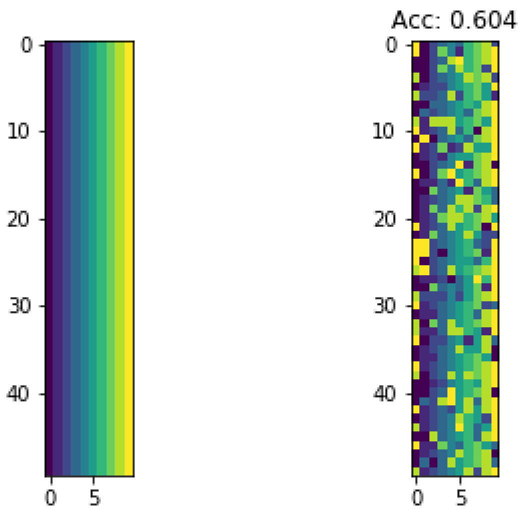

Figure 4: Results of the face recognition task for 10 people and 50 images per person. 4 images are revealed.

2.5 If the performance does not improve when including additional data, try to justify why. Which kind of additional data degrades performance instead of improving it?

The additional data do not improve the performance.

First, when we look at the additional data, we remark that many images are blurred, probably because the people were moving their head. To add a complementary information to the answer of question 2.4, it is possible that if we had added images of comparable quality we would have obtained better results.

Second, we only show 4 labeled images per person in both cases. But in the previous case, there were 10 images per person available, whereas now there are 50 images per person available. As a consequence, in the previous case, we obtain without any effort an accuracy of 0.4 (which corresponds to the 4 images out of 10 that we know). In this case, we obtain without any effort an accuracy of 0.08 (which corresponds to the 4 images out of 50 that we know), which is much lower than 0.4 in the previous case.