TP2: Semi Supervised Learning (SSL)
Graphs in ML

AL HOUCEINE KILANI

November 16, 2019

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.

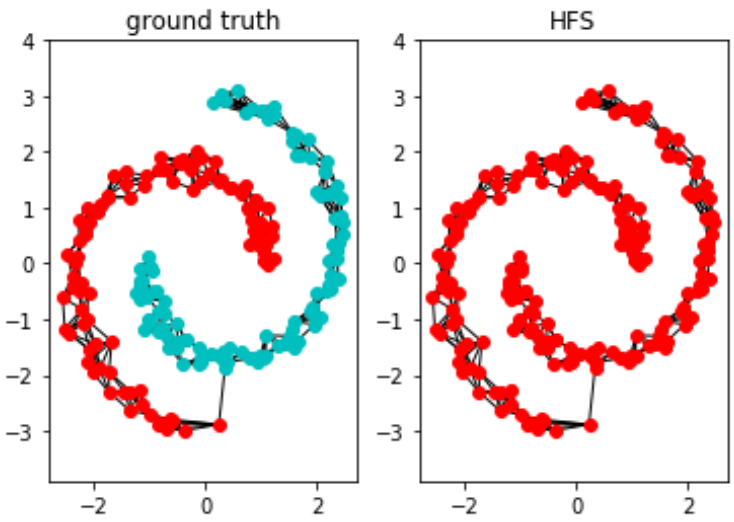

Figure 1: Worst case result

Solution: I used a k-nn graph with $k = 5$.
We see that, if we rerun the code multiple times, the result depends on the initial unmasked labels given to the HFS algorithm. Indeed in the worst case scenario, the labels provided are of the same class, thus the propagation will not give good results.
And in the best case scenario, we get an accuracy of 1 (happened only once, could not reproduce it again to paste the figure).
But in general the accuracy ranges between 60-80%.

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?

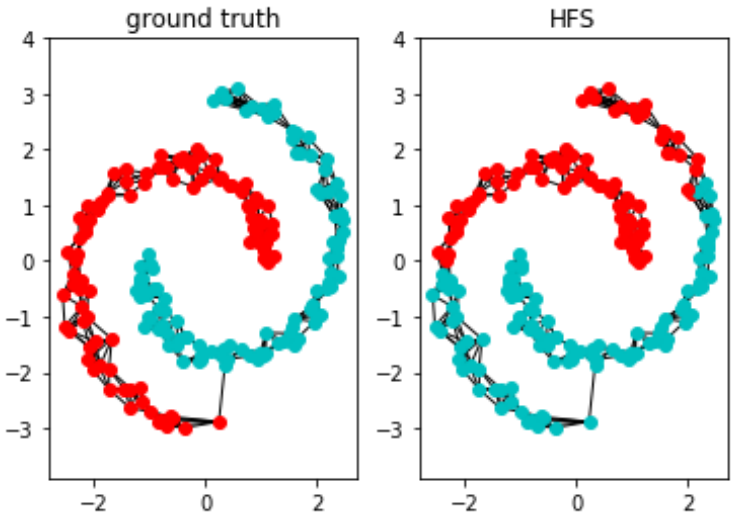

Figure 2: Typical result

Solution: Using HFS on the bigger dataset leads to the same results in the best case scenarios but the problem here is that there is a high probability that the unmasked labels will be of the same class. Therefore that accuracy we get will be 50%. We can solve this issue by increasing the number of labels unmasked provided for the algorithm but this is a solution we can do because we have a toy data set.
Furthermore, running the FHS code took longer to yield a result leading us to think that this is not a vary scalable solution.

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 hardHFS.

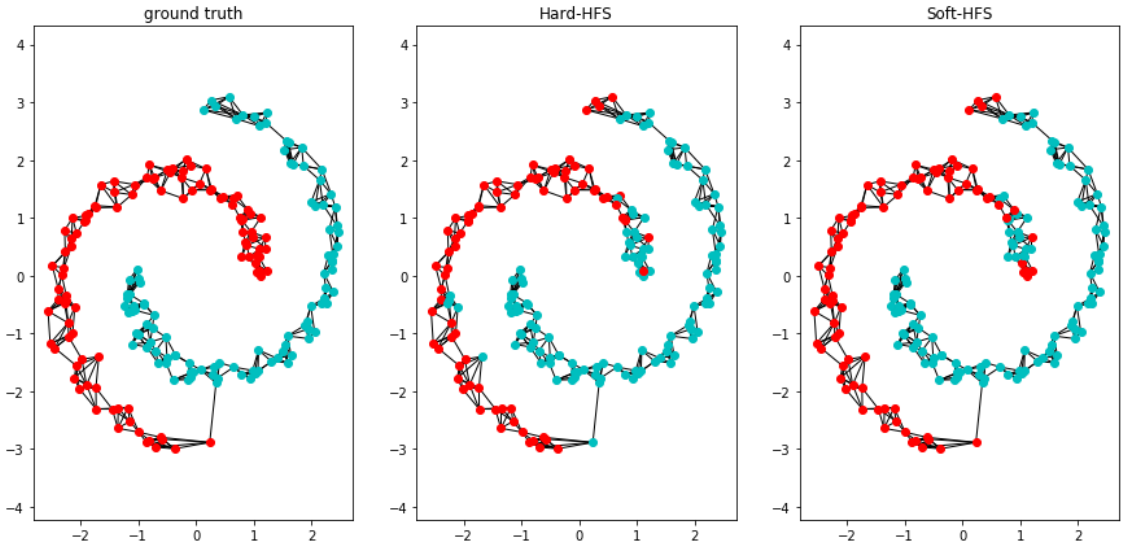

Figure 3: Comparison

Solution: After multiple reruns of the code (with 20 provided labels this time), we see that the SoftHFS always performs better than the hard one (The hard solution considers the noisy labels as true and then propagates the labels accordingly, thus leading to a big misclassification). But sometimes both gives good accuracies (nearly 89%) and we can explain this by the fact that in those cases, the unmasked labels provided wre good labels and also located on each of the moons (Example Figure 10 where we reach 77% accuracy for HardHFS and 91.5% for SoftHFS).
I tried multiple values for the c_l and c_u constants, but the ones that worked were a relatively big value for c_l compared to c_u (corresponding the the "degree of confidence" of the labels provided for the HFS algorithm). Therefore, I set $c_l = 90$ and $c_u = 5$

2 Face recognition with HFS

When completing the code, we tried running the HardHFS on the images but the L_{uu} was not inversible (I could have used some pseudo inversion method to use it anyway but I decided to carry on using SoftHFS for the rest).
For the graph, I chose $k = 5$ and for the HFS I kept the same value for c_l and c_u, $\gamma = 0.5$
On the left, we have the exact labels (each column is a person and each color corresponds to the class assigned to it). On the right, we represent the same way the assigned labels. We see that the result is not that great, even when changing the graph construction parameter k it does not yield to a better result. I think the the similarity metric is not quite adequate for this task (euclidean distance between the pixel values) or that we do not have enough data to perform a good classification.

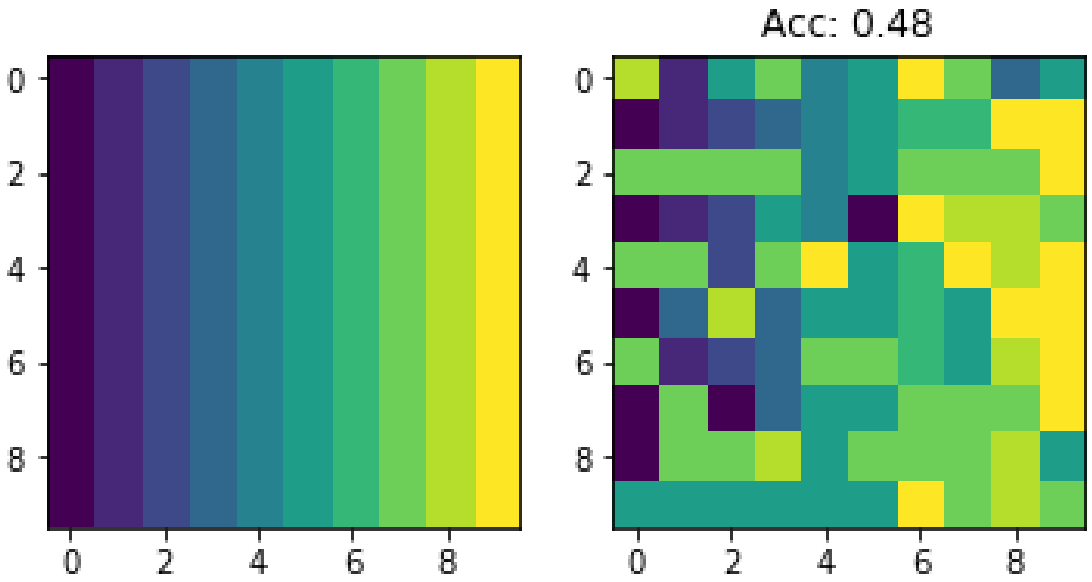

Figure 4: Result : accuracy = 48% total - 9-10% on unlabeled data

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

Solution: I used OneHot encoding for building tha Y vector used in the HFS algorithm (for the masked/unlabelled rows is was 0 everywhere on the row)

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?

Solution: I tried the 4 following configurations multiple times : Without preprocessing, only GaussianBlur, only equalizeHist and both. It seemed that on average, the best result was when we only used GaussianBlur. The images look like this : (Figure 5)

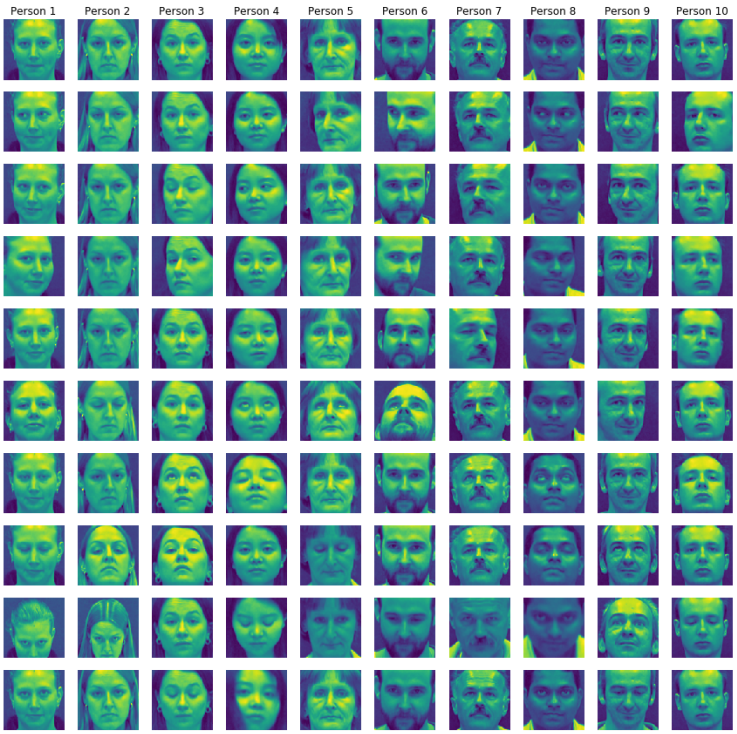

Figure 5: Applying GaussianBlur only

2.3. Does HFS reach good performances on this task?

Solution: Nearly 50% accuracy on a 10 classed classification task is not that great in my opinion...(In reality it is 10% accuracy approximately on the unlabeled / masked data). I do not know if it is indeed the performance intended to be reached here but I could not do better with the multiple configurations of the hyper parameters I tried.

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

Solution: When adding more data, I saw an increase of performance on the unlabeled/-masked images (it is around 15% accuracy). But We see an increase in the accuracy if we decide to increase the number of labeled examples given to the HFS algorithm.
The performance seems to be related to the 50 pictures sampled in each class. If those pictures were of the face of the person and not of the profile or when looking up or down, we get pretty good results (Especially when the labels given for the HFS are of the person when she is looking away)

3 Online SSL

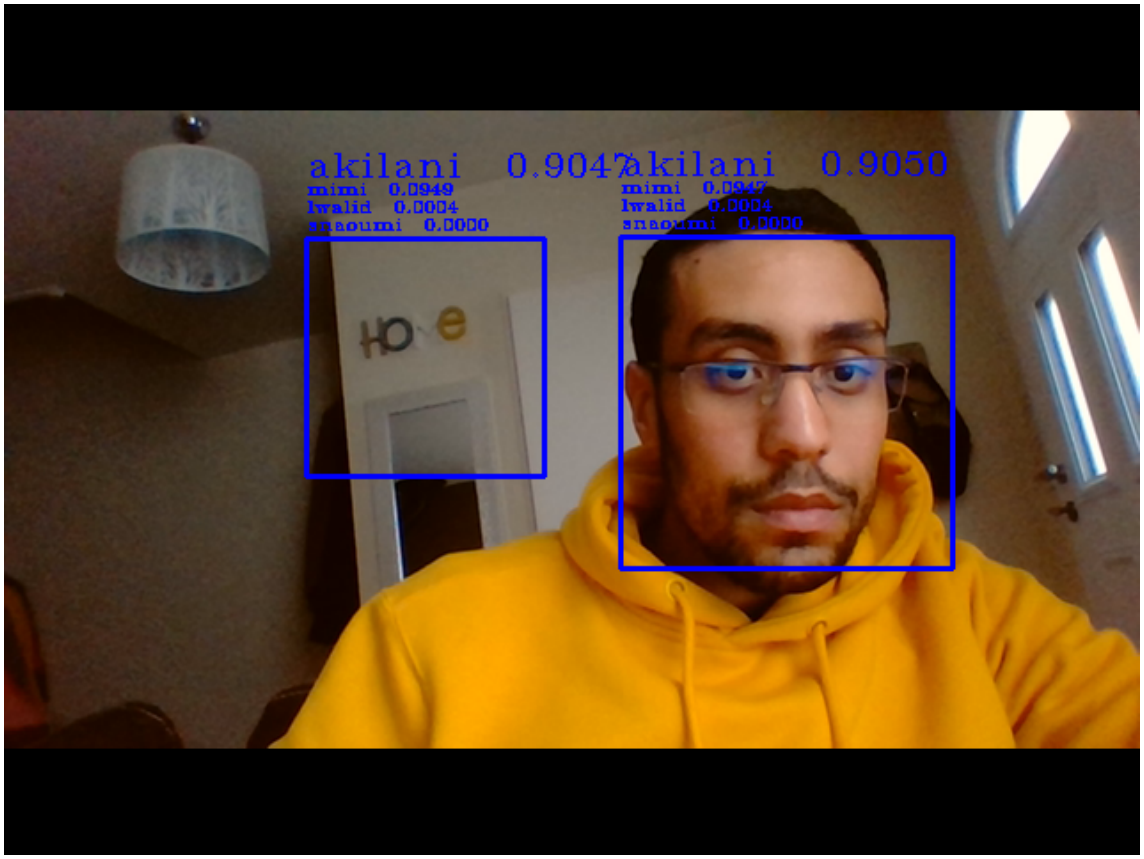

Figure 6: Face correctly labelled (Maybe a ghost behind me ? or more seriously a problem wih the face detector of openCV)

I followed the pseudo-codes and had no problem when implementing them. I encountered a problem of singular matrix inversion, this is why I replaced it by the pseudo inverse.
But when I was getting darker around, I noticed that the "probability" of recognizing my face was getting smaller (or even incorrectly labelled).
Furthermore, when retrying multiple times, I noticed that it took some time before it began to correctly label my face with high probability.
Concerning the new unlabelled faces, the solution I found and implemented was to put a threshold on the absolute value of the components of f_u vector (Greater than 80%) and if the prediction was below it, I do not predict any label (Unsure). This not a great solution because sometimes

Figure 7: Incorrect label (low luminosity)

it wrongly classifies a face with great confidence on the probability. I do not know how to explain this behavior..

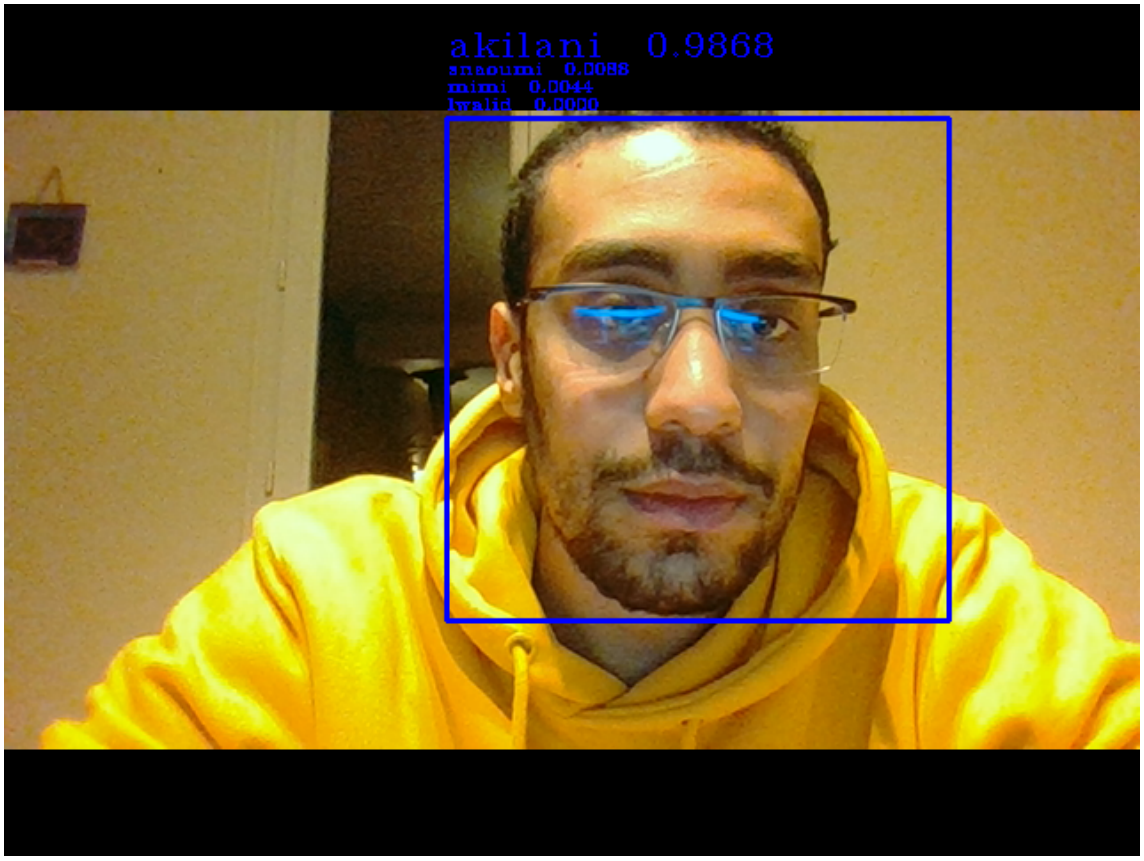

Figure 8: Correct classification

Figure 9: Correct classification

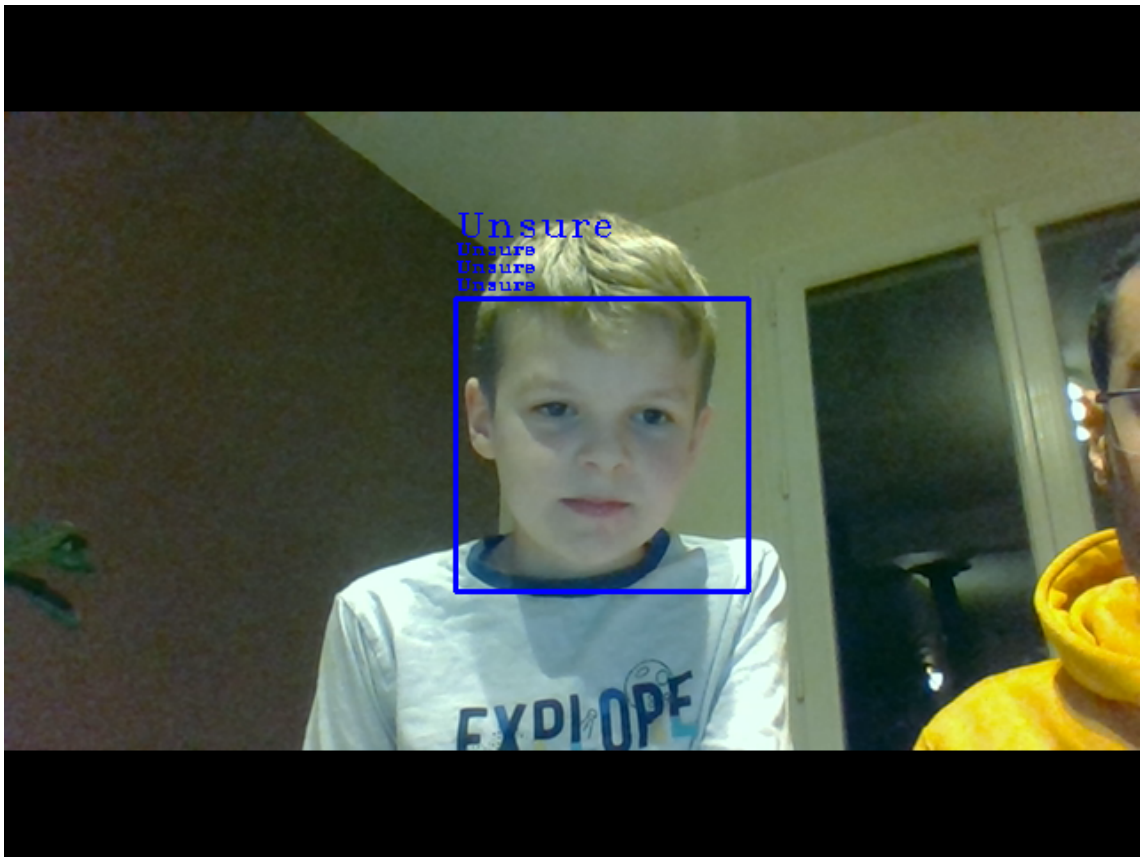

Figure 10: Unsure/ no classification