CODING CHALLENGE

Candidate: Castellucci Davide

The project I developed deals with an interpolation of a cluster of 2-D points with some parallel lines.

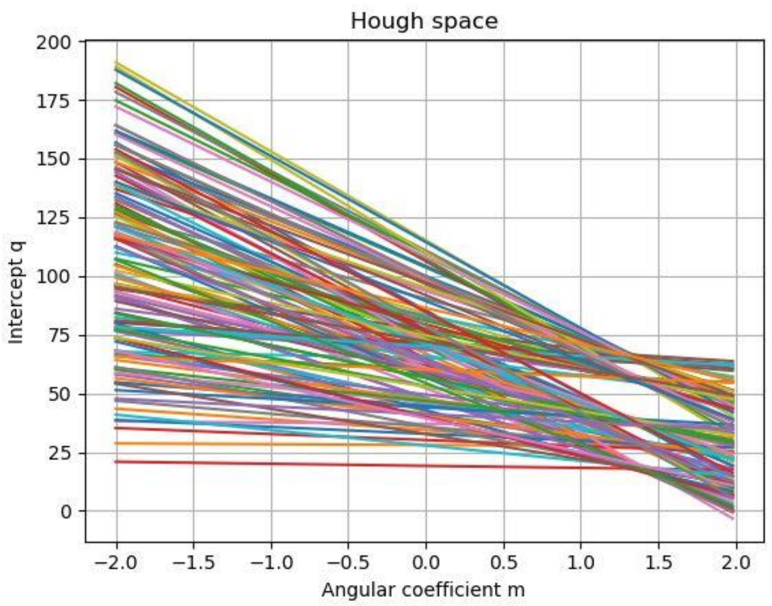
I divided the project in two steps: an initial guess for the position of the n lines and a final refinement of the latter.

1- The algorithm i purpose for this problem make use of the notion of the Hough space (or parameter space). In the normal euclidean space a line is defined by 2 parameters (say m and q) and the equation of the line is y = mx + q.

   If I consider a point in the euclidean space I have 2 parameters that are X0 and Y0 and I will have an infinite set of lines passing through this point with different m and q.

   The trick is to exploit this as a function of m and q and since I have 2 parameters (X0,Y0) I will obtain a line in the m-q plane for every point in the euclidean space.

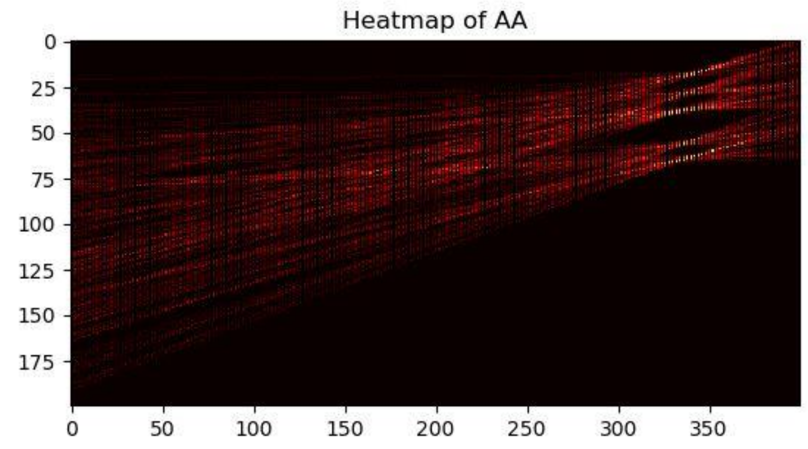
   For every pair of collinear points the two lines in the Hough space, corresponding to the points in the euclidean space, will intersect in a point of the m-q plane that gives me the m and the q of the line that pass for the two points in the euclidean space and if I have more than 2 collinear points I will obtain more lines that intersect in the same point.



   In this way I have traslated all the points in the euclidean space and then I have discretized this space in an accumulator array (AA), incrementing the cells of the AA of 1 for every lines tha pass on this cell.

   At the end I obtained a "voting scheme" where the cells of the AA with higher value were the coefficients of the lines that pass through more point.

In this case since I wasn't interested in just one line I selected the max of this array and set a treshold (depending on the maximum value) to accept as correct all cell of the matrix that had a value above the treshold.



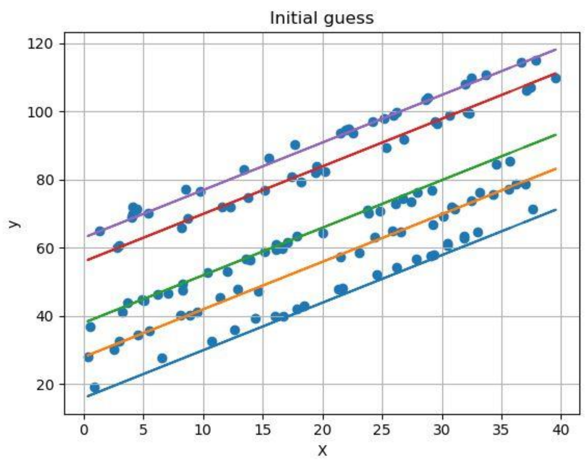
Then I made use of project data that say that the lines have a minimum distance of 8 and so i divided the matrix into stripes of height 5 in the q coordinate and searched for the maximum in every stripes, tresholding the values that I found in order to know if was spurious votes or the votes due to the collinearity of points.

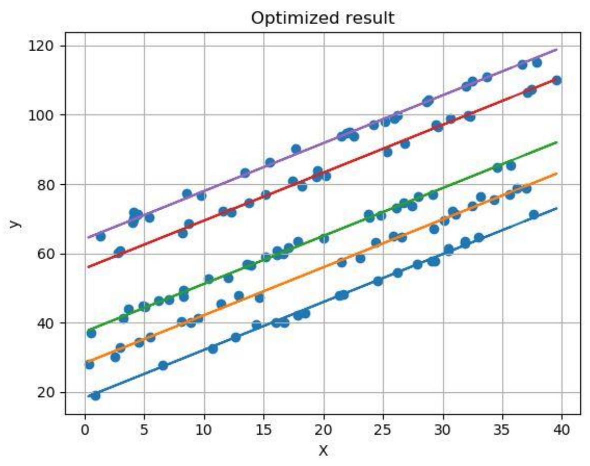
In this way i obtained many lines and I filtered out the ones which had a value of the angular coefficient far away of the mean of the angular coefficients of all the lines (another project data was that the lines were parallel) since it may happen that some spurious votes accumulate and goes above the treshold causing error in finding the right m and q.

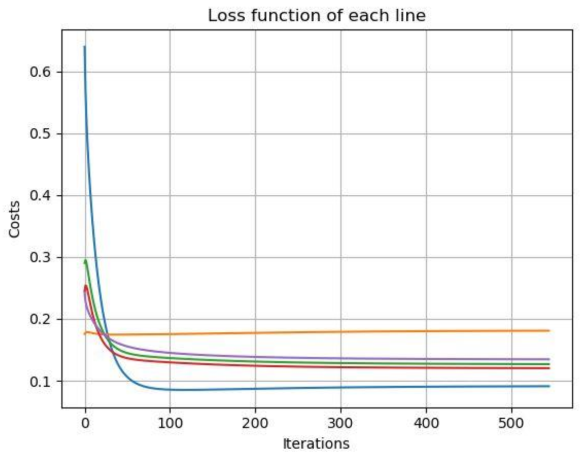
I also filtered out the lines that had a small difference in the q coordinate sice, by the way I divided the matrix, it may happen that the local maxima is between two consecutive stripes and it would be taken as two different maximum that identify two very similar lines, so in this case I kept only the ones with higher value in the correspondig cell of the AA.

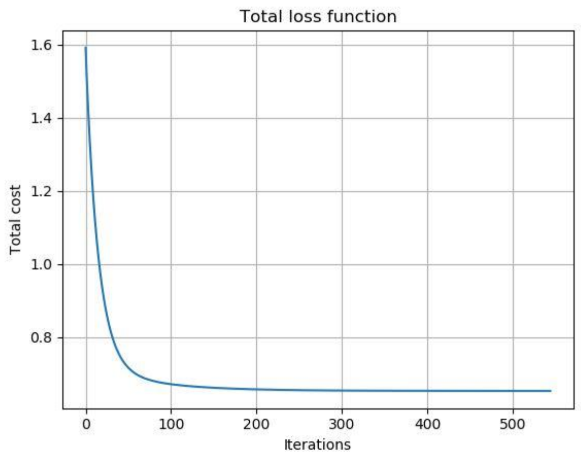
At this point I came up with the lines that passes throgugh the points with some approximation, but looking at the solution it would be however a good result.

2- Finally I applied some refinements to q and m of these lines with a gradient descendt where the loss function was the sum of square distances with the neighbour points of the lines (I found the neighbour point of a line "assigning" every point to the line which have the minimum distance with respect to it).









The strong point of this algorithm is that it’s very fast since finding the maximum in a matrix is way faster than computing *n* linear interplation where also the number *n* of lines is an unknown and in the last part where i have an optimization algorithm it’s still very fast because it’s initialized around an optimal solution.

The drawback is that on one hand that if I set a too low treshold also spurious votes would be taken as evidence and not just noise, on the other hand if I set a too high treshold some lines may not be identified, infact it may happen that due to wrong choises of the treshold or of the discratization of the Hough space, some ideal line is not identified (false negative) or some fake line is identified (false positive)

Finally basing on the AA I propose a measure for the level of noise that is 100-(max(AA)-mean(AA) since if I have random points the max and the mean will be the similar since the lines doen’t intersect exactly in one point and so I don’t have a high value for the max of the AA, instead if I have some set with a lot of collinear point I have in the AA very few points with very high values. Another choise could be to take the standard deviation of the points around the lines but this is less universal since it is related to the correctness of the algorithm.

In the end the results obtained are good since the error in the estimation of the angle is < 0.01 deg in the case with low level of noise and < 0.5 deg in the cases with high level of noise and the estimation of the intercept is < 0.02 in the case with low level of noise and < 0.6 in the cases with high level of noise.

PS: I made use of numpy just for calucating the max and the mean of the matrix, the same result can be obtained (in less efficient way) using the python lists.