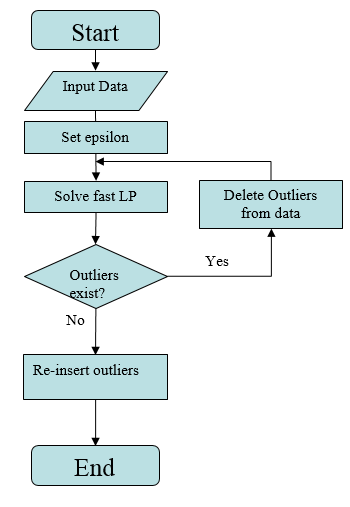
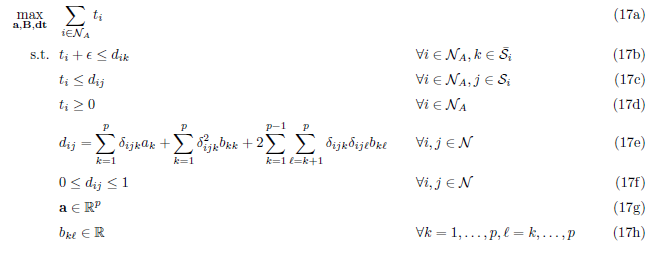
Report 10/19/2017

Our method:



What is Fast LP?

Let’s see the original problem LP:

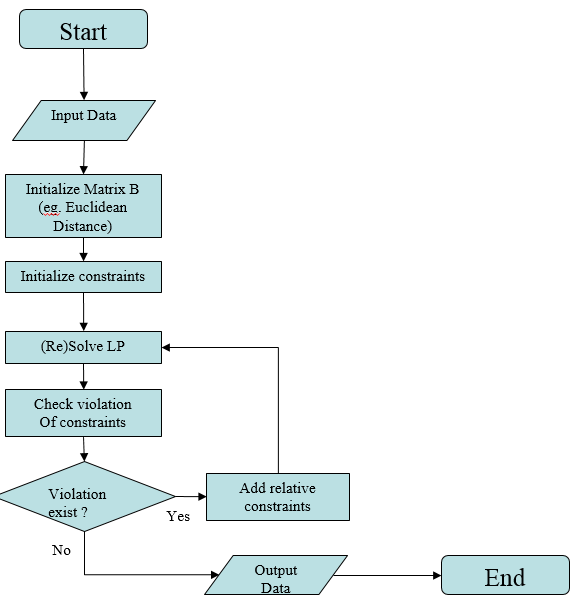


Our goal is to train matrix B as a metric and apply it with NN method to classify data.

Fast LP is based on original LP ,they have same solution, but Fast LP runs faster.

But how? By running iterations, adding limited number of constraints (17a,17b,17c) in every iteration rather than using all constraint in 17a,17b,17c. Number of constraints can be very large if we have large dataset.

Fast LP Algorithm with Row and Column Generation:



My questions:

1. Why NN, not 2-NN, 3-NN? OK, it reduces the calculating time.

2. What if 3-NN or 6-NN performs better in accuracy? Do we need experiments on this?

3. Value of epsilon matters only when it is nearly reaching “max epsilon”, so why do we care about it ? I mean it will not change the solution if we keep epsilon less equal than “max epsilon”/2.

4. About “re-insert outliers”, we got this:

|  |  |
| --- | --- |
| **Algorithm**  Improved Method of Outlier Re-insertion with Ranking of ; | |
| 1: | Compute for each outlier ; |
| 2: | Identify all non-outliers for which is now the nearest co-class/non-class neighbor, i.e. all j whose values are affected by insertion of ; |
| 3: | Define ; |
| 4: | Sort outliers in the order of decreasing ; |
| 5: | Insert the outliers according to decreasing ranking of ; |
| 6: | Repeat steps 1-5 until no further re-insertion is possible, i.e. remaining outliers all have <1 |

If Metric does not change after we put points back, so why do we do that? It’s meaningless.

5. About what to do next?

(1) See how initialization of Matrix B influence the solving time in Fast LP.

(2) Comparison of our method and others( k-nn ,svm…) with different dataset.

(3) Improving of the method (hardest)