Assignment 4

Submission by – Hsien Ming Lee (hlee99) and Souradeep Sinha (ssinha04)

Results of Random Clustering

k	Conditional Entropy	Variance of CE	Normalized MI	Variance of Normalized MI
2	1.789652898	0.001939611	0.004403131	0.002431196
3	1.388099965	0.002498981	0.006071621	0.002916565
4	1.144099324	0.00408954	0.008511976	0.003311119
5	0.981346542	0.00271836	0.008974615	0.002628037
6	0.861170809	0.002489265	0.010853501	0.002984549
7	0.769475005	0.002508647	0.013378639	0.003014596
8	0.697201362	0.004429469	0.016130027	0.003809447
9	0.639070809	0.0029159	0.01729104	0.003569944
10	0.591032105	0.002454263	0.017373986	0.003140853
11	0.549531771	0.002490337	0.019835038	0.003622652
12	0.514298761	0.002948807	0.022416984	0.003721683
13	0.484388643	0.002634493	0.024228039	0.004240816
14	0.457047942	0.002246988	0.024806419	0.004529556
15	0.434126503	0.002388892	0.027754214	0.004795042
16	0.412459369	0.003517978	0.030084466	0.004617803
17	0.394633772	0.002728033	0.030857808	0.005099929
18	0.376304994	0.002890814	0.031801301	0.00583237
19	0.360949081	0.002403677	0.033495355	0.005080315
20	0.347181673	0.003136417	0.036136063	0.004061899
Mean	0.694319544	0.00281213	0.020231801	0.003863598
Std Dev	0.379505234	0.000602043	0.009610351	0.000915086

Graph of Random Clustering with increasing values of k.

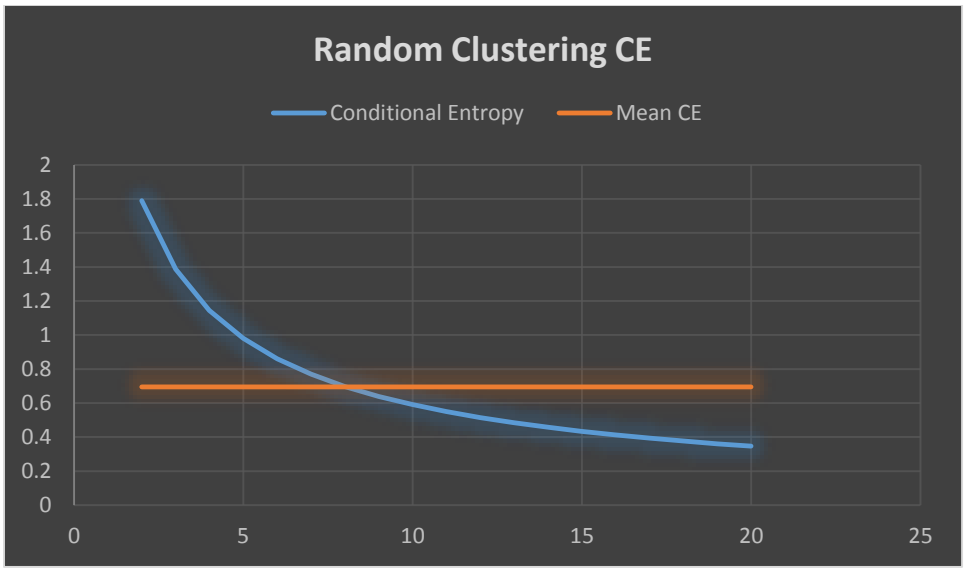

Figure 1 Dropping values of CE with increasing k, Elbow at 11

Graph of Normalized Mutual Information with increasing values of k.

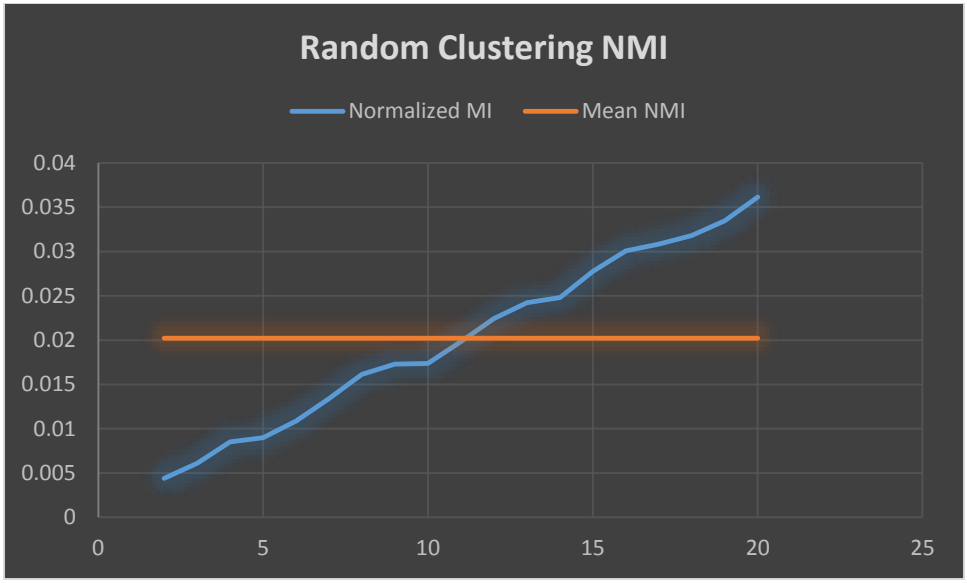

Figure 2 Steadily values of NMI with increasing k

Results of Principal Component Analysis method of data reduction

k	Conditional Entropy	Variance of CE	Normalized MI	Variance of NMI
2	1.815638316	0.142738647	0.494715382	0.0325434
3	1.024249556	0.001700791	0.734321563	0.001469876
4	0.942949024	0.005381312	0.728905447	0.003213487
5	0.964424169	0.010864079	0.685675406	0.005511796
6	0.795580463	0.069544609	0.658565674	0.033539857

7	0.574434964	0.002294711	0.82722533	0.0034271
8	0.464908248	0.002462824	0.841964251	0.003524003
9	0.423397868	0.001928552	0.816544091	0.002730111
10	0.444271343	0.001772045	0.817193104	0.003171026
11	0.404200002	9.26E-04	0.794362798	0.002771833
12	0.357790493	0.001832501	0.780926655	0.003344958
13	0.353729353	0.00181092	0.775454056	0.003281189
14	0.343692106	0.001176732	0.765792246	0.004399757
15	0.324295259	9.14E-04	0.766411627	0.004757541
16	0.30656503	0.001334775	0.741249559	0.002527468
17	0.294595905	0.001340324	0.735131552	0.001529707
18	0.291670548	0.001369301	0.731360267	0.001519304
19	0.29029521	0.001525523	0.729074737	0.00160146
20	0.277996276	0.001744775	0.71962035	0.001537299
Mean	0.562878112	0.013298019	0.744447058	0.006126378
Std Dev	0.381686063	0.034058367	0.07512596	0.009299155

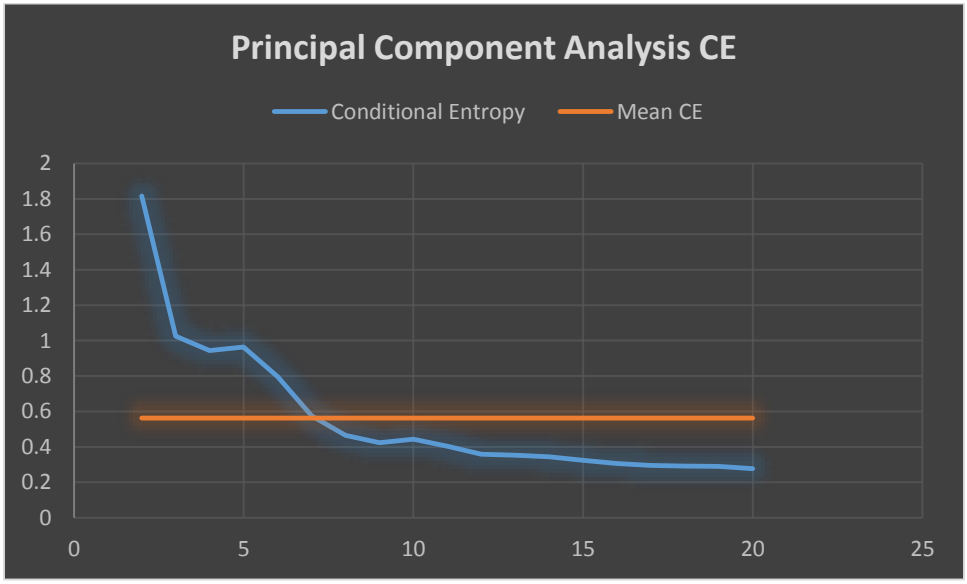

Figure 3 Dropping values of CE, with saddle point at 5 and elbow at 11

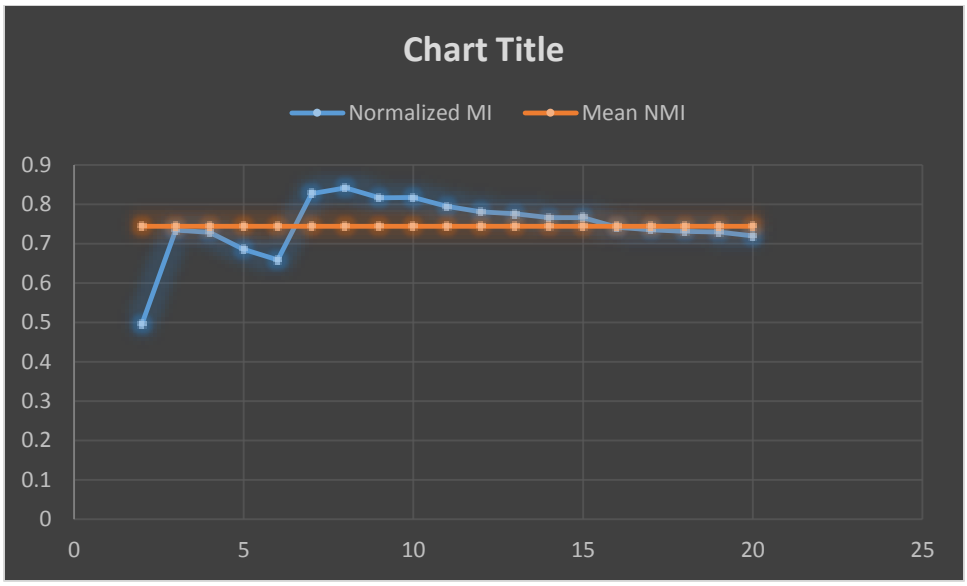

Figure 4 NMI values with increasing k

Results of Random Projection + EM

k	Conditional Entropy	Variance of CE	Normalized MI	Variance of NMI
2	2.173135311	0	0.238026515	0
3	2.173135311	0	0.238026515	0
4	2.173135311	0	0.238026515	0
5	2.173135311	0	0.238026515	0
6	2.173135311	0	0.238026515	0
7	2.173135311	0	0.238026515	0
8	2.173135311	0	0.238026515	0
9	2.173135311	0	0.238026515	0
10	2.173135311	0	0.238026515	0
11	2.173135311	0	0.238026515	0
12	2.173135311	0	0.238026515	0
13	2.173135311	0	0.238026515	0
14	2.173135311	0	0.238026515	0
15	2.173135311	0	0.238026515	0
16	2.173135311	0	0.238026515	0
17	2.173135311	0	0.238026515	0
18	2.173135311	0	0.238026515	0
19	2.173135311	0	0.238026515	0
20	2.173135311	0	0.238026515	0

Inconclusive results.

Analysis:

The running time of RP Ensemble is $O(M*D^2*P*E)$; M is number of attribute. D is number of data. P is projection. E is number of ensemble members. It takes roughly O(D^3) time. One way to improve the performance is to reduce the size of similarity matrix for fast lookup and also come up a faster way to compute the sum square error. These improvement can reduce the running time by one magnitude.

Using a K-mean instead of EM can reduce the running time of the process. It would generate a parse matrix containing 0 or 1 for each data point. 1 means i point is in j cluster, and 0 means i point is not in cluster j. The pairwise storage can be compute by comparing the membership of each point, 1 if both data are in the same cluster, 0 if they are not.

The algorithm we prefer is the PCA. It has relatively fast running time, 82825ms, compare RP +EM with O(n^3) running time. Despite having more consistent result, the running time of RP +EM is a huge disadvantage. Although the random clustering has the fastest run time, 1587ms, but the result is not as consistent.

In our submit code, every classes are working as desired beside ClusterEnsemble.java. Our PCA class was able to produce similar result as describe in the paper. However, there are still some issues in the ClusterEnsemble.java, the result we got does not match the outcome in the paper.

Coding Decisions:

We used computeSSE function to be used instead of computeBIC which finds the optimal number of k, depending on elbow generated by sum of squared errors. We chose the elbow to be an angle of 10° hoping that a saddle point won't be lesser angle.

Bugs:

In the ClusterEmsemble.java file, mergeClusters function, we are still unable to figure out why the maxSims values are sorted even before we call the Collection.sort(maxSims) function. Hence, the noise point detection and remerging is not correctly implemented.

RP+EM does not produce desired different results with varying values of k. Which might be caused by the bug above.