Practical Work - Part 1 Option B
Data Processing - Clustering

Advanced Machine Learning Techniques

Universitat Politècnica de Catalunya
Faculty of Informatics

06/01/2017

Erol Kazancli

1. Cluster Validation - KNN Approach

1.1 - Algorithm pseudocode

```
Try for n_clusters in 2..10:
    For each n_clusters repeat 100 times:
        Choose two equal-sized subsamples from the data-set
        Apply k-means to the whole dataset and two sub-samples separately
        Break the symmetry problem in the two sub-samples using the whole sample k-means
          results.
        Form the clusters using the cluster results from the two sub-samples merging the same
          clusters.
        For each cluster do:
            Find average P1 and P2 index using all the examples in the cluster by calculating the
              ratio of membership into the same sub-sample of 10 nearest samples and feeding
              Them into P1 and P2 formulas.
        Select the worst(smallest) index among the clusters and add it to the list of indices for k
    Using the 100 found worst indices for k find the skewness for k and add it to the list of
      skewnesses.
Use the list of skewnesses to find the best k for the dataset. The k with the most negative
  skewness value should be the ideal value.
```

1.2 - Experiments

I used the Flea Beetles data in “http://lib.stat.cmu.edu/DASL/Datafiles/FleaBeetles.html” used by the authors in the paper. In my experiments, sometimes I used only the P1 index, sometimes only P2 and in some cases both to see the different behaviours by also shuffling the data in each case. According to the paper, the ideal number of clusters is when the 100-indices-set has the shortest right tail. I used skewness results for this purpose. Therefore the most negative value must indicate the correct number of clusters. Unfortunately the result was not as expected all the time in my many experiments. One of the expected results I obtained is shown in the figure 1 below. Since the data has 3 clusters in reality, it is logical to expect the most negative value for skewness at this level. Similar results were obtained quite often but not always.

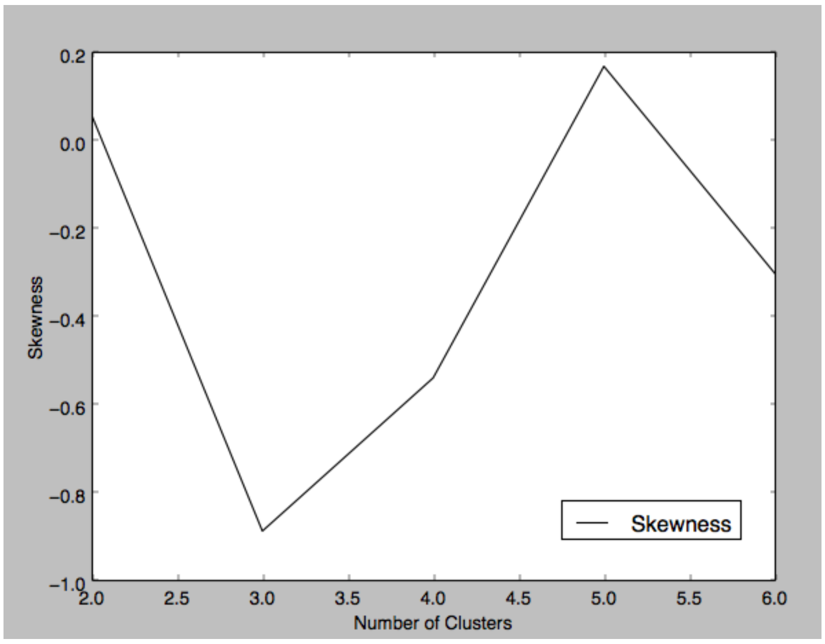

Figure 1 - Skewness Results with Flea Beetles

Authors use another method to measure the dissymmetry, which is error plots for 75th or 90th values in the ascending order of the 100-index values for each number of clusters. The values I obtained for these measures were totally different than theirs and did not give any clue as to the ideal number of clusters. Therefore I am not including them, however, the values are nevertheless included in the code if observation is needed.

Another experiment with a randomly generated 100 data with 3 features and 3 clusters produced the result as shown in figure 2 in one case.

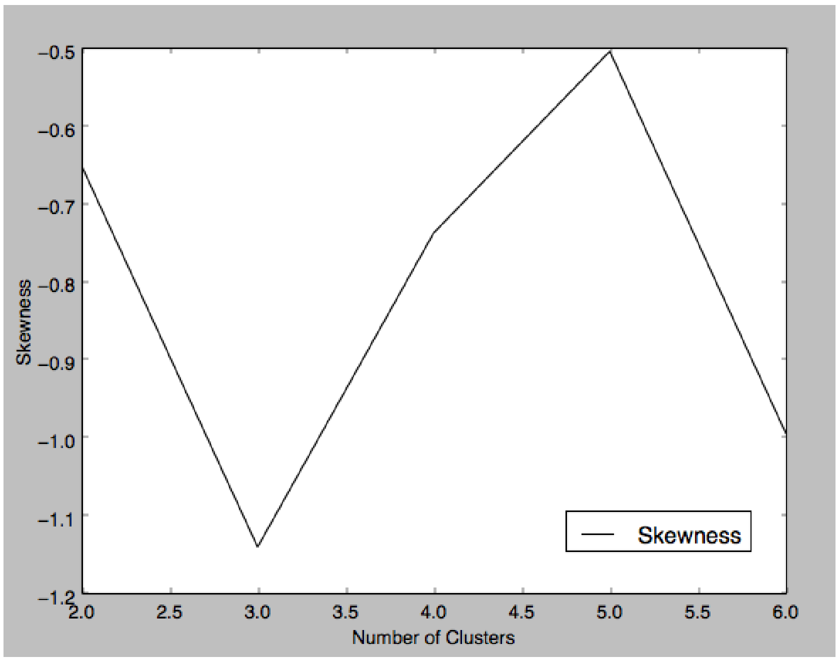

Figure 1 - Skewness Results with generated data

1.3 - Conclusions

I could not obtain very satisfactory results with the implementation I made based on the explanations in the paper. I found some of the information unclear. For example, in another paper [3] from the same authors, which explains the same method with small differences, while explaining their experiments they say they used 1−binocdf(2,10,0.5) = 0.0107 for the threshold value. However this cdf calculation is completely wrong, the value of this should be 0.9453. What they are trying to say must be 1−binocdf(8,10,0.5), because this expression exactly equals the value stated. Therefore, I am not sure if the index formulations were done accurately or if I could understand them correctly.

The difference between the results may also be due to the clustering algorithm used. Although they say Partitioning Around Medoids(PAM) is preferable for clustering , I used k-means for convenience. However I do not think this difference might affect the results to this degree.

The algorithm is quite slow for even a small number of data (a few minutes). This is expected to a certain extent since there are many iterations and in every iteration there is k-means clustering involved. My implementation also needed some improvements in terms of efficiency so I made some last minute changes to make it more efficient, for example I calculated k nearest neighbours for each example in the beginning since this does not change with different iterations. For big data, it would be logical to apply this test only to a subsample randomly chosen, assuming that the subsample is representative enough all the classes.

2. Clustering - Recursive Partition Based K-means

2.1 - Algorithm pseudocode

```
Choose a range mMin..mMax for the partitions to be obtained
For each m in mMin..mMax
  If m == mMin divide each dimension d into mMin equal parts
  Else obtain dimensions by dividing the previous dimension-partitions into two
  Using the intervals obtained obtain partitions and obtain partition membership for each
    Example. Save the partition and the metadata found.
For each partitioning previously obtained and using the metadata of each partition do:
   Apply weighted-lloyd algorithm 1000 times or until the error does not change
   Keep the best error and the partitioning
With the best partitioning found obtain the real k clusters
```

2.2 - Experiments

I used flee beetles data from "http://lib.stat.cmu.edu/DASL/Datafiles/FleaBeetles.html" in my first trials. I used mMin = 2, mMax = 5, k = 3 values. For this dataset, when i observed the clustering results, I saw very similar results with the standard k-means algorithm. I obtain error values in both cases by taking the square of differences to the core of each cluster and adding them up. While the error value for RPKM algorithm was 32, the error for standard k-means was to be found 30. So in this case the algorithm seems to roughly work like k-means. However this was a small dataset and a relatively easy case.

I generated a sample with 10000 examples with 3 features. In one case, the error for RPKM clustering was 18.415 while the error for k-means was 16.631. Here the difference is more conspicuous. In one experiment with 100000 generated examples with 3 features, the error for RPKM was 172991 while the error with k-means was 164936.

One thing I observed in my experiments was in RPKM, the coarsest partition almost always gave the best clustering. This might be due to random selection of the cores because in partitions where there are more and finer partitions, the probability of finding a bad core is quite high. Besides, we are only making one trial for each partition, not several trials to obtain better cores.

2.3 - Conclusions

I see from my experiments that this algorithm has the ability to mimick k-means to a certain extent, however, some improvements can be done in the algorithm like more initializations in the case of finer partitions. Besides, I found my implementation slower than the standard k-means although I made several improvements to increase the efficiency in the algorithm. This performance difference might become less or even reversed - which is what is supposed to happen - with real big data. Otherwise, if this performance problem is not solved, there is no point in using this algorithm instead of k-means.

3. Bibliography

[1] On Application of the k-nearest neighbour approach for cluster validation. Volkovich, Z. and Barzily, Zeev and Avros, Renata and Toledano-Kitai, Dvora. 2009.

[2] An efficient approximation to the k-means clustering for massive data. Capó, Marco and Pérez, Aritz and Lozano, Jose A. 2016.

[3] Cluster Validation: A binomial Noised Model. Kitai, Volkovich, Avros, Yahalom, Weber.