Affinity Propagation, and other Data Clustering Techniques

Patrick Redmond*, Prof. John A. Trono, Dave Kronenberg,
(* denotes primary author)

Abstract

In our research we sought to create implementations of several common clustering algorithms and a relatively new approach called Affinity Propagation. Our objective was to compare the techniques by running tests on one and two dimensional datasets provided by Professor Trono. Dave Kronenberg implemented a standard randomly seeded K-Means clustering program and many related support functions. We also implemented a program to run three variations of Graph Linkage clustering: Single-Link, Complete-Link, and Centroid-Link, and we incorporated the Silhouette Index method of cluster verification into this program to rate the linkage strategies and to choose an optimal clustering (*k*-value) from the generated dendrogram. Finally, we completed an Affinity Propagation program which closely follows the details laid out by Frey and Dueck [1]. Using these three programs we clustered a variety of one and two dimensional datasets, and we analyzed the results to confirm that Affinity Propagation produces clustering errors competitive with that of K-Means in an order of magnitude less time.

Introduction

Clustering, the task of making datasets expose their secrets by finding the groupings already inherent in the data, is an established problem in computer science. The groups which clustering uncovers can represent various types of relationships depending upon the kind of data being clustered. Demographics can be revealed by clustering data from social networking websites, features in photographs or video can be found and tracked by clustering the image data, and changing trends can be discovered by comparing data clustering results over time. Clustering is therefore widely applicable in corporate environments, especially with regard to data mining. A robust and scalable general solution to find the true clusters inherent within a dataset efficiently, with respect to time and computing resources, is desirable. However, conventional clustering algorithms haven't quite lived up to these goals.

The well studied K-Means algorithm [3] is a popular strategy to solve many types of clustering problems, but this algorithm must be instructed as to precisely how many clusters should be produced. This is counterintuitive to the objective of finding the groups already present in the data. If there are 4 groups present, and K-Means is instructed to produce 5, then it will produce 5 clusters by somehow separating one or more of the 4 existent groups. Additionally, the K-Means algorithm relies heavily on an initial choice of cluster centers. This choice may be random or influenced by previous clusterings of similar data, but either way, it is not directly based on the true groupings inherent within the data.

The Graph Linkage algorithms sidestep the problem from which the K-Means method suffers by producing every possible number of clusters in the form of a dendrogram. The desired clustering can be chosen from this data structure with additional analysis after it is produced. These algorithms assume all points start in their own cluster and iteratively merge the two clusters which are currently least distinct. The success of this greedy strategy depends as much on the characteristics of the data as on the merit of the algorithm itself.

A great variety of hybrid clustering strategies are used in practice, but these are typically tailored to work in specific problem domains. A general solution to the clustering problem is still desirable, and Brendan Frey and Delbert Dueck published a new idea in the journal Science [1]. This article, entitled *Clustering by Passing Messages Between Data Points*, describes a fresh approach to the clustering problem. This new method "simultaneously considers all data points as potential exemplars" (Science; Feb. 2007; p. 972), and by employing a message passing procedure, the exemplars are "elected" by the other data points. In addition, this method doesn't take a predetermined desired number of clusters as input, but actually determines an appropriate number from the data.

The metric by which we will assess each clustering result is called the *error against cluster means*, or more concisely, *clustering error*. This dependent variable is the sum of all Euclidean errors for each single cluster in the clustering. To find the Euclidean error of a single cluster, the average of all data points in the cluster is taken, and the error for that cluster is the sum of the Euclidean distances between this average and each of the data points in the cluster. Simply put, *clustering error* is the total Euclidean distance of each data point to the center of its cluster, across all clusters.

Clustering Algorithms

1. K-Means Clustering: Input includes the data points, and an integer *k*.

A set of *k* center points are randomly chosen from the same range as the input set of data points, where *k* is the number of desired output clusters. These center points are considered the initial center points of the *k* clusters we will

produce. We assign each data point to the center point it is closest to according to Euclidean distance, but other measures may also be appropriate. This step produces k clusters because each data point is assigned to a center point and there are k center points. We refine the clusters by repeating the following steps until the clusters remain stable: In each cluster, average the values of all members and keep the result as the new center point of the cluster. Reassign all data points to the closest center point. Repeat.

The quality of the clustering produced by this algorithm is heavily dependent on the initial set of centers chosen from the input data. Given that fact, we opted to run the algorithm with 1000 random initializations, taking the best clustering based on total error. When we refer to a clustering produced by K-Means, we are referring to the best of these 1000 runs.

2. Graph Linkage Clustering: Input includes the data points, and a number k.

A set of data points are each individually considered distinct clusters. We reduce the number of clusters by repeating the following steps until there are k clusters, where k is the number of desired output clusters: Compare each cluster to each other cluster. Choose the two clusters which are closest and merge them into one new cluster containing all of the data points which the previous two old clusters contained. In the new cluster, average the values of all members and keep the result as the new center point of the cluster. Repeat.

In this context, the measure of closeness between two clusters is dependent on which kind of linkage clustering [3] we are performing: In Single-Link clustering, clusters are as close as the closest pair of points which bridge the two clusters. With Complete-Link clustering, clusters are as close as the farthest pair of points which bridge the two clusters. Finally, in Centroid-Link clustering, clusters are as close as their center points. Again, for this measure we used Euclidean distance, but other measures may also be appropriate.

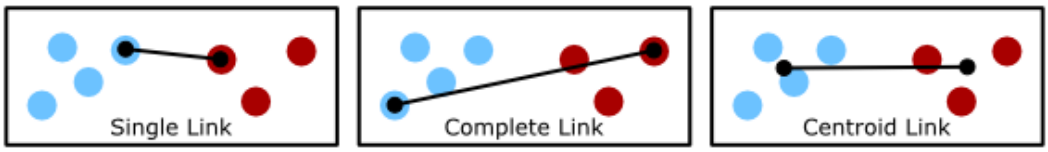

Figure 1. The distance between the red (left) and blue (right) clusters is calculated differently by the three linkage strategies.

3. Affinity Propagation Clustering: Input includes the data points, an integer n, and a value λ.

Consider a set of data points and three matrices, each of which represents a set of relationships between every ordered pair of data points. The matrices hold values called similarity, availability, and responsibility. We populate the similarity matrix using negative squared error (Euclidean distance, squared and negated). This similarity matrix can be sparse or asymmetrical, but in our research we used similarities which were not sparse and exhibited symmetric relationships across the diagonal of the matrix.

The similarity of a point to itself, found on the diagonal of the matrix, influences the likelihood of that data point becoming an exemplar during the message passing process. It is desirable to reduce this influence so that exemplars are chosen based on the characteristics of the data. Uniform self similarities will give each data point an equal chance at becoming an exemplar, but the value which is uniformly applied will affect the number of exemplars in the resultant clustering. If the value is too high every point will be its own cluster, but if it is too low all points will be included in one large cluster. To obtain a clustering which best reflects the data, the choice of what number to apply as the uniform self similarity should be guided by the data. According to Dueck and Frey the useful values lie in the range defined by the maximum and minimum values in the similarity matrix. Therefore, we chose to use the average of the highest and lowest similarities for the uniform self similarity. It would also be reasonable to use the average of all similarities calculated, the center of mass of similarities.

We next initialize the matrix of availabilities to contain all zeros, and enter a loop which performs the following steps: Update the responsibility matrix; assign exemplar points; exit if the stopping conditions are satisfied; update the availability matrix; repeat.

In generating values to update either the responsibility matrix or the availability matrix, we follow a specific procedure for each *ordered pair* of data points. This means that while we kept the similarity matrix symmetrical, the nature of Affinity Propagation requires that the responsibility and availability matrices will not be. The responsibility[i,k] will most likely not equal the responsibility[k,i]. The same can be said about availability.

The procedure for determining a single responsibility value can be written in pseudo-code as follows:

```
given data_point i and data_point k:
      result = -∞
      for_each data_point z such_that (z is_not k):
            temporary = availability[i,z] + similarity[i,z]
            if (temporary greater_than result):
```

```
                  result = temporary
      final_result = similarity[i,k] - result
```

This procedure essentially searches for the maximum availability+similarity sum, where the first data point in the pair is fixed at the input *i* and the second data point ranges over all data points except *k*. Once found, this value is subtracted from the similarity of datapoint *i* to data point *k*. Note that this code is entirely self contained (references no other procedures) and that the availability and similarity matrices are accessed only for reading; therefore this procedure can simply be executed for each ordered pair of data points until the entire responsibility matrix is updated. The value produced represents the "accumulated evidence for how well-suited point *k* is to serve as the exemplar for point *i*" (Science; Feb. 2007; p. 972), where larger values indicate *k* is well suited to *i*.

The procedure for determining a single availability value can be written in pseudocode as follows:

```
given data_point i and data_point k:
      if (i is_not k):
            sum = 0
            for_each data_point z such_that (z is_not i) and (z is_not k):
                  temporary = responsibility[z,k]
                  if (temporary greater_than 0):
                        sum = sum + temporary
            result = sum + responsibility[k,k]
            if (result greater_than 0):
                  result = 0
      else:
            sum = 0
            for_each data_point z such_that (z is_not k):
                  result = responsibility[z,k]
                  if (result greater_than 0):
                        sum = sum + result
            result = sum
      final_result = result
```

Note: This procedure does two types of calculations which are separated by the if-else structure because the availability of distinct points is calculated differently than the availability of a point to itself.

In finding the availability of distinct points, this process calculates the sum of all positive responsibilities, where the first data point in the pair ranges over all data points except *i* and *k*, and the second data point is fixed at *k*. This sum is then added to the self responsibility for the input data point *k*. The value produced represents the "accumulated evidence for how appropriate it would be for point *i* to choose point *k* as its exemplar" (Science; Feb. 2007; p. 972), where higher values indicate that it is appropriate for *i* to choose *k*.

When determining the availability of a point to itself, this process calculates the sum of all positive responsibilities, where the first data point in the pair ranges over all data points except *k*, and the second data point is fixed at the input *k*. Note the restriction, that the first data point in the pair is not *i*, has been lifted here. The value produced represents the "accumulated evidence that point *k* is an exemplar," where a higher sum indicates that it is.

The process by which points are assigned to exemplars is simpler than the availability and responsibility methods. We consider a data point *i*, and find the data point *k* which results in the maximum availability+responsibility sum, where the first data point in the pair is fixed at the input *i* and the second data point ranges over all data points. The chosen data point *k* is the exemplar for data point *i*. If the chosen data point *k* happens to be *i*, then *i* itself is an exemplar. During this process we produce a clustering because each data point is assigned to another. However, sometimes the result is problematic. If a data point, which doesn't identify itself as its own exemplar, is chosen by one or more other points to be their exemplar, then we have a "false clustering." This situation indicates that the message passing procedure needs to continue to iterate in order to propagate the true exemplars out to the farther points.

The final unexplained process in the main loop described above is checking the stopping conditions. The conditions to be met are twofold; the exemplar assignments must not be a "false clustering" and exemplar assignments must not have changed for the last *n* iterations of the main loop, where *n* was one of the original inputs of this algorithm. When these conditions are met, the algorithm loop breaks and the clustering is said to have converged.

As yet, we have still not made use of the input λ. This is called the dampening value, and it is optional because its presence improves the algorithm but the essential properties of Affinity Propagation work without it. If presented with a one dimensional dataset consisting of 1, 5, 6, and 10, the Affinity Propagation would shunt availability and responsibility

magnitudes back and forth forever because 1 and 10 are no better assigned to exemplar 5 than they are if assigned to exemplar 6. Therefore, dampening is needed to escape from the oscillation of exemplar decisions. With dampening, instead of simply replacing responsibility and availability values when a new one is calculated, the new value is averaged against the previous value causing the oscillation to slow. Finally, because exemplar assignments take place between availability and responsibility updates, the oscillation will eventually become "off balance" and either 5 or 6 will get all three other points assigned to it. However, since dampening slows the rate of change in the availability and responsibility matrices, it may cause affinity propagation to take longer to converge. *It is therefore desirable to use the lowest value of λ which leads to convergence when algorithmic efficiency is important; however the minimum value of λ that promotes convergence is not known a priori.*

Another optional addition to the algorithm is noise, which we did not implement in our research. Noise is applied in the same step where dampening is applied. It is simply a random change of very small magnitude in the value which is put into the availability or responsibility matrix. Over many iterations, this small change may result in a cumulative error which is enough to overcome oscillations but is otherwise negligible. In combination with a low dampening value, noise ensures that Affinity Propagation always converges.

Procedure

The data we used as input to the various clustering methods included: a small one-dimensional dataset, a midsized two-dimensional dataset, the two-dimensional Ruspini dataset, and four large one-dimensional datasets consisting of Sagarin Rating values for all Division I men's basketball teams for four specific years. The datasets range from 15 to 319 data points.

Since the number of output clusters (*k*-value) produced by Affinity Propagation is not directly controllable, for each dataset we used the *k*-value produced by Affinity Propagation for input to each of the other algorithms. The dampening factor used with Affinity Propagation was consistently set to λ=0.65, and the average of the highest and lowest similarity value for a given dataset was used as the self-similarity for all points. Additionally, when running the Graph Linkage algorithms we used the Silhouette Index metric [2] to choose three "optimal" *k*-values for comparison with those produced by Affinity Propagation.

Results

Our trials produced several interesting datasets which can be analyzed to tell us more about the differences between the clustering techniques. As stated in the introduction, the primary metric we will be using to assess the clusterings produced by these techniques is the *error against cluster mean* (which is the total Euclidean distance of each data point to the center of its cluster, across all clusters).

Error against cluster mean represents the incorrectness of a clustering with the assumption that a correct clustering is one where every data point is close to its cluster average. A smaller value indicates a better clustering, but the value is not normalized; therefore this metric only has meaning when comparing clusterings where only one independent variable has been altered. Consider, if a very high *k*-value is used to produce a clustering, such that *k* equals the number of data points in the dataset, then clustering error will be zero. If one were to compare the clusterings produced by two different algorithms without using the same *k*-value, then the discrepancy between the clustering errors cannot be attributed solely to the algorithms; the different *k*-values also contributed in some way to the value of the clustering errors. Therefore one also cannot meaningfully compare errors between clusterings where two different types of data are used.

Dataset	Data dimensions	Number of data points	**optimal *k*-value chosen with Affinity Propagation and used in all subsequent trials**
Ruspini	2d	75	**4**
cluster4	2d	200	**3**
cluster	1d	15	**1**
1984	1d	282	**4**
1991	1d	296	**4**
1997	1d	307	**5**
2000	1d	319	**5**

Table 1. This table relates the dataset names with some attributes and, most importantly, the *k*-value which was used by all five clustering methods as chosen by Affinity Propagation. Note that only the Ruspini and cluster4 datasets are two-dimensional.

Dataset	***k*-value used in all trials**	optimal *k*-value chosen with the Silhouette Index in each of the Graph Linkage method trials		
		Centroid	Complete	Single
Ruspini	**4**	4	4	4
cluster4	**3**	3	3	3
cluster	**1**	4	4	4
1984	**4**	44	79	2
1991	**4**	69	62	2
1997	**5**	49	77	72
2000	**5**	2	2	51

Table 2. This table shows the k-values chosen by the Silhouette Index metric from the dendrograms produced as the Graph Linkage trials executed.

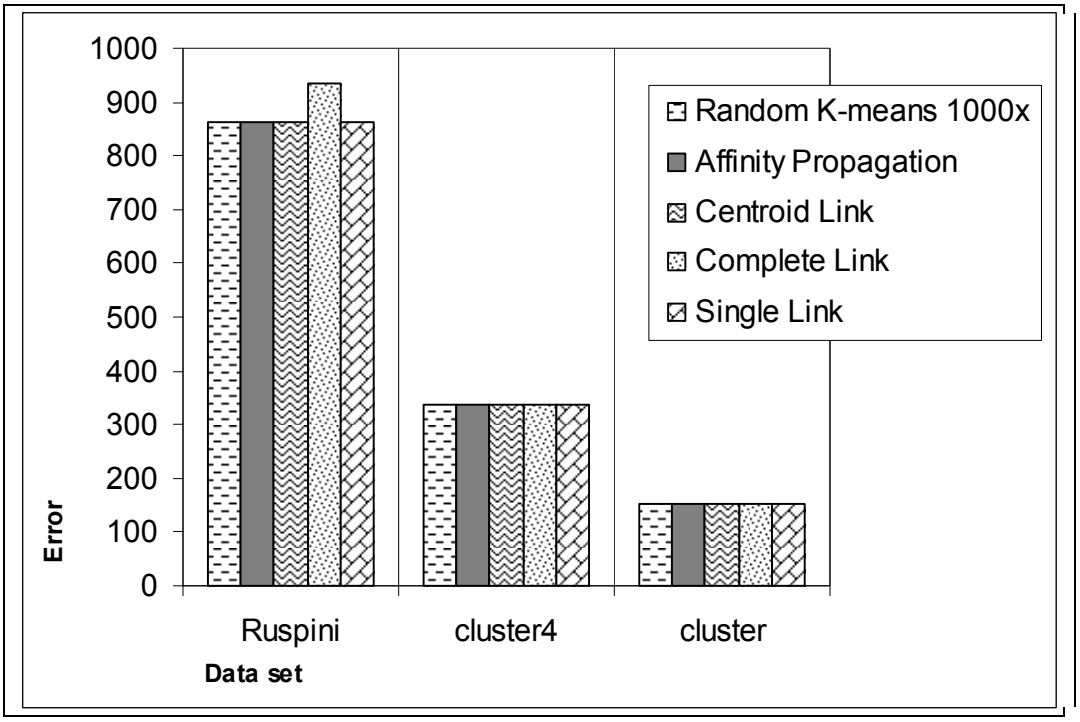

Figure 2. These two graphs illustrate how the clustering error produced by each algorithm for a given dataset differs.

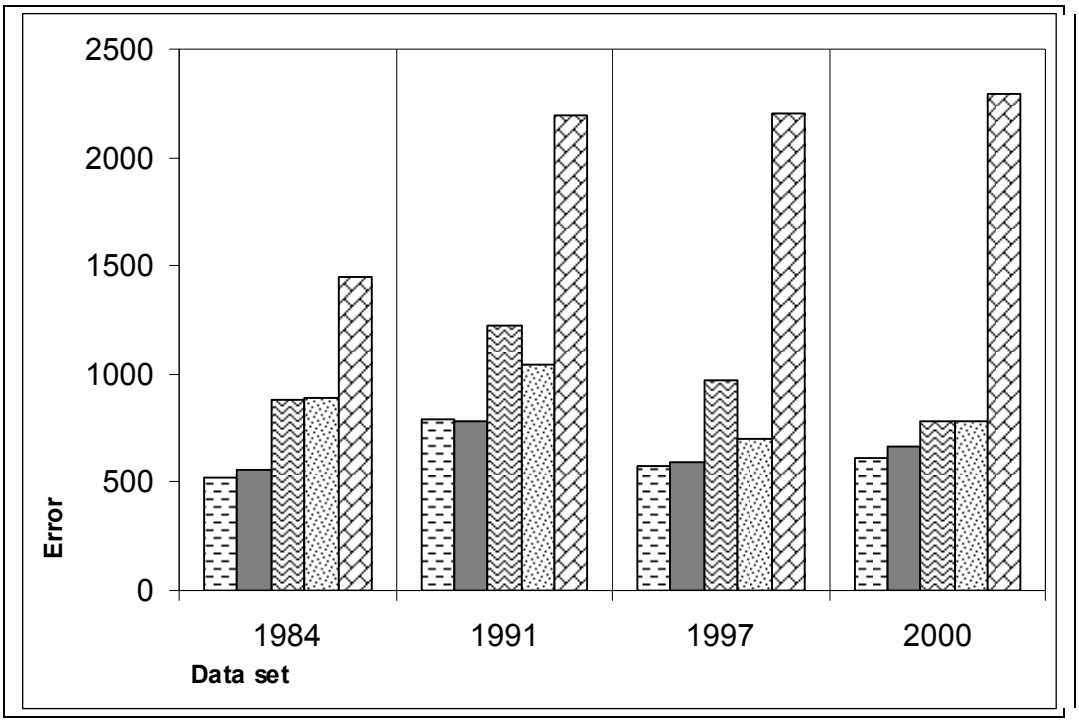

Figure 3. The four Sagarin Rating datasets resulted in more variation between clustering algorithms.

An alternative metric for measuring a clustering result, which we did not use in our research, is the *error against exemplars*. This metric is almost the same as that which we used except that this metric sums distance between data points and the cluster exemplar (as opposed to the cluster mean). Not all clustering methods produce exemplars; therefore this metric shouldn't be confused with the one we did use.

Conclusions

The very competitive results produced by all five clustering methods in the first three datasets (Ruspini, cluster4, cluster) require explanation. It could be postulated that the datasets were very similar in their general properties, and the resultant clusterings for each dataset were similar in their accuracy or inaccuracy. However, this is likely not the correct explanation because these three datasets range greatly in size (75, 200, 15) and don't all share the same dimensionality (2d, 2d, 1d). In addition, the errors range too greatly between datasets to support this idea.

It might be shown with statistical analysis of the clusters produced that their structure is simply very strong (tightly grouped and obvious). This is virtually the case with the Ruspini dataset, as only three datapoints are questionable with all others clearly belonging to one of the 4 groupings. However, this may not be the explanation for all three; the same analysis, using the other datasets (cluster4 and cluster), was not undertaken.

The conclusion to be drawn then is that all five clustering methods will produce similar clusterings when the underlying structure of the data is strong. *This idea supports the integrity of the algorithms against the integrity of each other*, and is supported further by the results of the Silhouette Index metric. In Table 2 the Silhouette Index agrees with Affinity Propagation more clearly for these first three datasets than the final four datasets. Affinity Propagation and the Silhouette Index metrics of the Centroid Link, Complete Link, and Single Link clustering algorithms are four very different methods of producing a *k*-value for a dataset; however, being in close agreement, it is heavily implied that the underlying clustering structure of these datasets must be strong.

The Silhouette Index chosen *k*-values resultant from trials on the final four datasets were consistently in disagreement with both each other and that of Affinity Propagation. Their disagreement implies that the underlying structure of the data may not have a strong clustering. In addition, their large difference from the *k*-value chosen by Affinity Propagation may point out a problem with the way it arrives at a *k*-value in our research.

A potential problem with the way our implementation of Affinity Propagation arrives at a *k*-value is mentioned in our discussion of self similarity. In our research, we chose to use the average of the highest and lowest similarities in the dataset as the uniform self-similarity. This approach may be naive because it allows outliers to influence the chosen self-similarity value, and this in turn influences how many clusters are in the resultant clustering. For example, two duplicate data points make the minimum self-similarity equal to zero, while two opposing outliers could make the maximum self-similarity

much too large. (As already suggested it would be quite reasonable to use the center of mass of all similarities as the uniform self-similarity, but this was not investigated during our research.)

The final four datasets produced wildly different clustering errors. This is perhaps the most exciting result of our research, although it is tempered by the possibility of their being a problem with which *k*-values were chosen for our trials. In these four datasets Affinity Propagation and the K-Means algorithm clearly outperform all three of the Graph Linkage methods. Affinity Propagation tends to have error slightly worse than that of K-Means, but scores better than it on the 1991 dataset. The most important aspect of this result, however, is that while each K-Means trial took more than 30 minutes to complete, all Affinity Propagation trials completed running in less than 1 minute.

Affinity Propagation produced clustering errors at least as low as the Graph Linkage methods in all trials and consistently competitive with the errors produced by the standard K-Means method, but it did so at least one order of magnitude faster than the K-Means method.

(Note: Unfortunately, in the course of our research we neglected to take figures with a profiler on the time it took to run each algorithm on each dataset. An important result of this research is directly related to this oversight, and we have included a conservative estimate - possibly larger than actual - for the running times of both K-Means and Affinity Propagation.)

Bibliography

[1] Frey, Brendan J., and Delbert Dueck. "Clustering by Passing Messages Between Data Points." *Science* 315 (2007): 972-76. Print.

[2] Kaufman, Leonard, and Rousseeuw, Peter, J. *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, 1990/2005. Print.

[3] Jain, A. K., M. N. Murty, and P. J. Flynn. "Data Clustering: A Review." *ACM Computing Surveys* volume 31, issue #3 (1999): 265-96. Print.