Chapter 10
Clustering for Binary Featured Datasets

Peter Taraba

Abstract Clustering is one of the most important concepts for unsupervised learning in machine learning. While there are numerous clustering algorithms already, many, including the popular one—k-means algorithm, require the number of clusters to be specified in advance, a huge drawback. Some studies use the silhouette coefficient to determine the optimal number of clusters. In this study, we introduce a novel algorithm called Powered Outer Probabilistic Clustering, show how it works through back-propagation (starting with many clusters and ending with an optimal number of clusters), and show that the algorithm converges to the expected (optimal) number of clusters on theoretical examples.

Keywords Binary valued features · Clustering · Emails · k-Means · Optimal number of clusters · Probabilities

10.1 Introduction

For over half a century, clustering algorithms have generated massive research interest due to the number of problems they can cover and solve. As an example [1], the authors use clustering to group planets, animals, etc. The main reason for the popularity of clustering algorithms is that they belong to unsupervised learning and hence do not require manual data labeling, in contrast to supervised learning, which can require the cooperation of many people who often disagree on labels. As an example one can even mention Pluto, the celestial body no longer considered a planet as of 2006. The advantage of using unsupervised over supervised learning is that rather than relying on human understanding and labeling, clustering algorithms rely purely on the objective properties of the entities in question. A good clustering algorithm survey can be found in [2].

The largest obstacle for clustering algorithms is finding the optimal number of clusters. Some results on this topic can be found in [3], where the authors com-

P. Taraba (✉)
Berkeley, CA 94710, USA
e-mail: taraba.peter@mail.com

S.-I. Ao et al. (eds.), *Transactions on Engineering Technologies*,
https://doi.org/10.1007/978-981-13-2191-7_10

pare several algorithms on two to five distinct, non-overlapping clusters. As another example, one can mention [4], where the authors use a silhouette coefficient (c.f. [5]) to determine the optimal number of clusters.

In this study, we construct three theoretical datasets: one with clusters that are clearly defined, a second with clusters that have randomly generated mistakes and last but not least clusters which have majority of features noisy. We then show that the algorithm introduced in this chapter converges to the expected number of clusters for all three examples.

This chapter is organized as follows. In Sect. 10.2, we describe the Powered Outer Probabilistic Clustering (POPC) algorithm. In Sect. 10.3, we confirm on three theoretical examples that the algorithm converges to the expected number of clusters. In Sect. 10.4, we present an additional example of real email dataset clustering. In Sect. 10.5, we dive deeper into the algorithm properties. And finally in Sect. 10.6, we draw conclusions.

10.2 Algorithm Description

The POPC algorithm, originally introduced in [6], relies on computing discounted probabilities of different features belonging to different clusters. In this section, we use only features that have binary values 0 and 1, which means the feature is either active or not. One can write the probability of feature f_i belonging to cluster k as

$$p(cl(f_i) = k) = \frac{c(s_j(f_i) = 1, cl(s_j) = k)C_m + 1}{c(f_i = 1)C_m + N},$$

where c is the count function, cl is the clustering classification, N is the the number of clusters, $k \in \{1, \ldots, N\}$ is the cluster number, s_j are samples in the dataset, $s_j(f_i) \in \{0, 1\}$ is the value of feature f_i for sample s_j, and C_m is a multiplying constant (we use $C_m = 1000$ in this study). If we sum over all clusters for one feature, we get:

$$\sum_{k=1}^{N} p(cl(f_i) = k) = 1,$$

and subsequently further over all features f_i we get:

$$\sum_{i=1}^{F} \sum_{k=1}^{N} p(cl(f_i) = k) = F. \tag{10.1}$$

If we used (10.1) as the evaluation function, we would not be able to optimize anything, because the function's value is constant no matter how we cluster our samples s_j. Hence, instead of summing over probabilities, our evaluation function J uses higher powers of feature probabilities as follows:

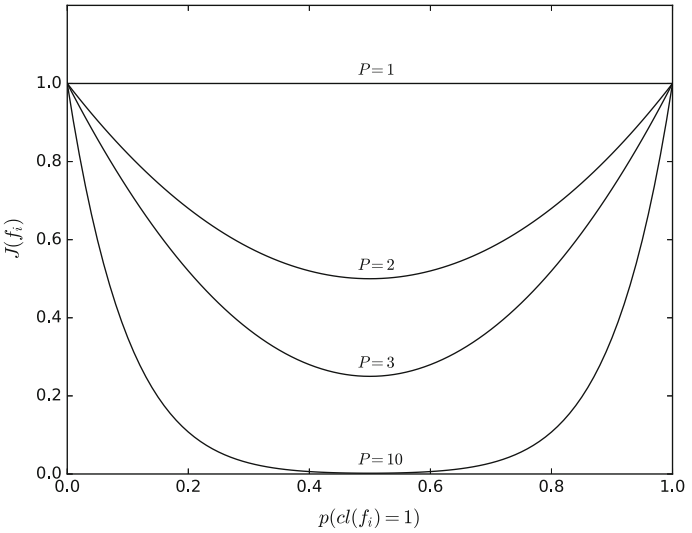

Fig. 10.1 Values of the evaluation function for a hypothetical case where there is one feature and two clusters. If the feature is distributed between two different clusters, the evaluation function J has a lower value (near 0.5 on the x-axis). In case the feature belongs to only one cluster, or mostly to one cluster, J has a higher value for evaluation function (left near 0.0 or right near 1.0 on the x-axis—values close to optimum $J(f_i) = 1$)

$$J = \sum_{i=1}^{F} J(f_i) = \sum_{i=1}^{F} \sum_{k=1}^{N} p^P(cl(f_i) = k) \leq F, \tag{10.2}$$

where P is the chosen power and we choose $P > 1$ in order to have a non-constant evaluation function. The main reason why this is desired can be explained with reference to a hypothetical case where there is only one feature and two clusters. We want samples with $s_j(f_1) = 1$ to belong only to one of two clusters. In the case that samples are perfectly separated into the two clusters on the basis of the one feature, we maximize $\sum_{k=1}^{N} p^P(cl(f_i) = k)$ (value very close to 1 as we use discounting). In the case that some samples belong to one cluster and some to the other, so the separation is imperfect, we get a lower evaluation score (value significantly lower than 1) for the feature. The higher the power P, the lower the score we obtain when one feature is active in multiple clusters. This is displayed in Fig. 10.1 for different powers $P \in \{1, 2, 3, 10\}$ and two clusters. For results reported in this section we use $P = 10$.

The algorithm starts by using the k-means algorithm to assign each sample s_j a cluster number. The number of clusters is set to half the number of samples in the dataset. Then the algorithm proceeds to reshuffle samples s_j into different clusters $\{1, \ldots, N\}$. If the evaluation function J increases as a result of reshuffling, the new cluster assignment is preserved. The algorithm ends when reshuffling no longer increases the evaluation function J.

The algorithm can be summarized in the following steps:

1. Using k-means clustering, assign each sample s_j cluster $cl(s_j) = k$, where $k \in \{1, \ldots, N\}$ and N—the number of clusters—is chosen to be half the number of data samples.
2. Compute $J_{r=0}$ for the initial clustering, where r denotes the iteration of the algorithm.
3. Increase r to $r := r + 1$, start with $J_r = J_{r-1}$.
4. For each sample s_j, try to assign it into all the clusters to which it does not belong to $(cl(s_j) \neq k)$
   a. If the temporary evaluation score J_T with the temporarily assigned cluster improves over the previous value, i.e. $J_T > J_r$, then assign $J_r := J_T$ and move sample s_j to the new cluster.
5. If J_r is equal to J_{r-1}, then stop the algorithm. Otherwise go back to step 3.

Remark 1 (Implementation detail) The algorithm can be made faster if one saves the counts of different features for different clusters and updates these counts only if and when the evaluation score improves. This is just an implementation detail, which speeds up computations significantly and does not influence the result.

10.3 Examples

In this section, we create three theoretical examples, one perfect, one slightly imperfect and one with a majority of noisy features.

For the perfect example, we create a dataset containing 200 samples with 100 features and assign each sample to a random cluster ($N = 7$) with a uniform discrete distribution, so that every cluster has approximately the same number of samples. In the same way, we assign every feature to a cluster. A feature is active (its value is 1) only if the feature belongs to the same cluster as the sample and only 80% of the time.

In Fig. 10.2 top, we show the k-means evaluation function depending on the number of clusters. There is a break-point at exactly $N = 7$, the expected number of clusters. Figure 10.2 bottom shows the number of clusters created by the POPC algorithm. We can see that if we start with a number of clusters larger or equal to 7, we always end with the expected number of clusters 7.

The first example clustered with POPC algorithm is displayed in Fig. 10.3. As shown, clusters are found as expected.

For theoretical example 2, we create also noisy features, which do not belong to any cluster. These features can be active for samples belonging to any cluster. They are active 20% of time. The results for example 2 are displayed in Fig. 10.4. The POPC algorithm introduced in this chapter once again yields the expected number of clusters.

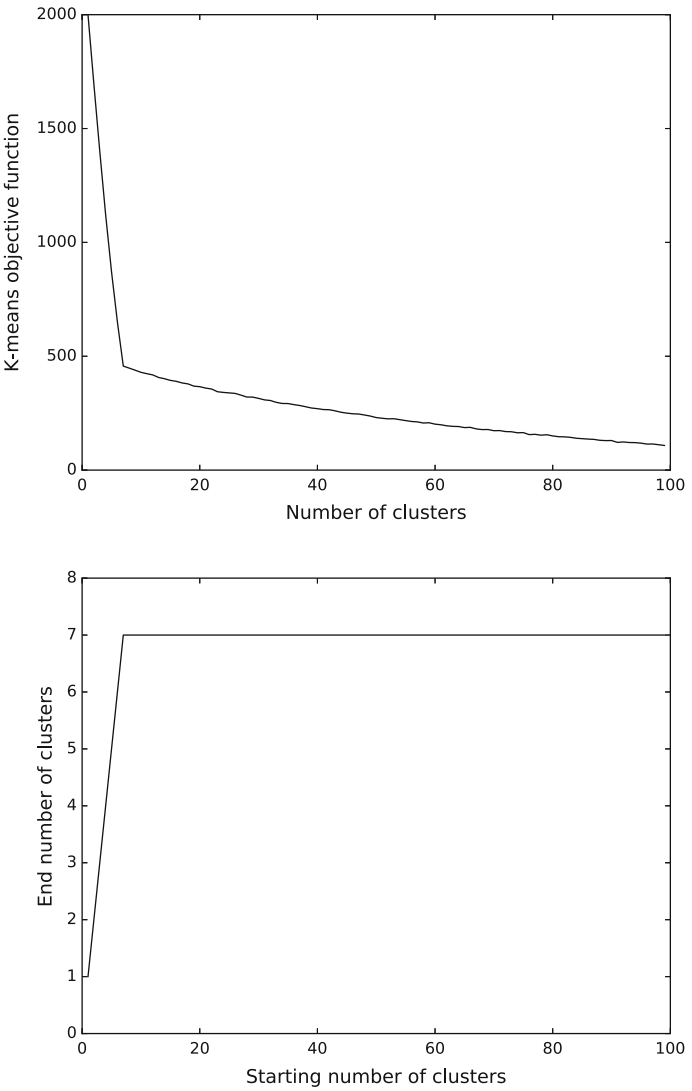

Fig. 10.2 (Top) K-means evaluation function depending on the number of clusters for example 1. (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters

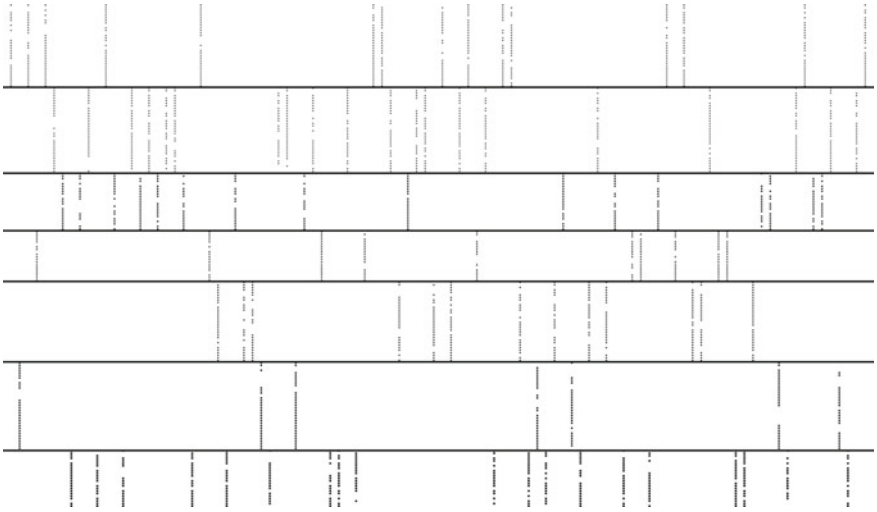

Fig. 10.3 Theoretical example 1—seven perfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (separated by lines) and on horizontal axis we have different features

The second example clustered with POPC algorithm is displayed in Fig. 10.5. As shown, clusters are found as expected despite having a small amount of imperfect features.

For theoretical example 3, we create 7 clusters with every cluster having 30 samples with 20 features. Every cluster has exactly one feature, which is active only for the cluster it belongs to. The remaining 13 features are random features, which do not belong to any cluster and are activated 50% of time. This is the main example, which shows why the POPC algorithm is so useful. K-means algorithm, due to a majority of features being randomly active, cannot find features which are significant features for clusters and finds clusters which are not the way we would expect them. Evaluation function $J_{k-means}$ is very close to 0 even if we knew we are looking for 7 clusters. This is displayed in Fig. 10.6.

On the other hand, when we use POPC algorithm we find exactly 7 clusters we expected and more importantly J_{POPC} is close to 7, which is the amount of features that are significant to respective clusters. This is displayed in Fig. 10.7. This last theoretical example explains why in real life situations, POPC is not only able to find the correct number of clusters, but also to find significantly better clusters than k-means for binary feature datasets.

All the theoretical examples introduced in this section can be generated with C# code which can be downloaded from [7]. Same code contains implementation of popc algorithm.

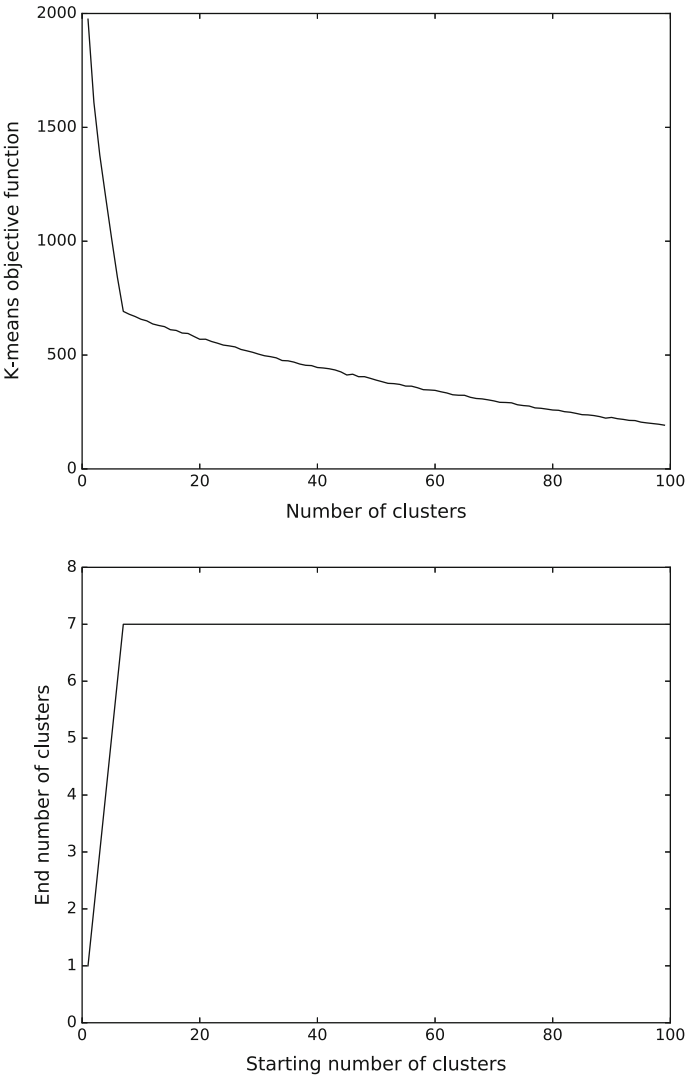

Fig. 10.4 (Top) K-means evaluation function depending on the number of clusters for example 2. (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters

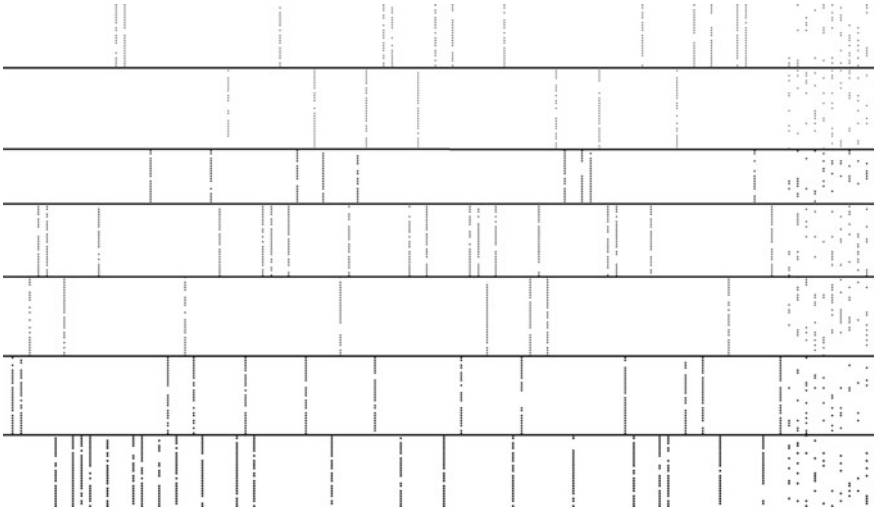

Fig. 10.5 Theoretical example 2—seven imperfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (separated by lines) and on horizontal axis we have different features. Last 10% features (on right) are the features, which do not belong to any cluster and are active 20% of time

10.4 Real Life Example—Email Dataset

As a last example, we will use an example from real life: email data. Each email is one sample in the dataset. The features are the people included on the email. Results are displayed in Fig. 10.8. As shown in the graph on the top, there is no clear break-point as in the previous two examples, showing why it is so hard in real life situations to find the optimal number of clusters. Despite this fact, when we start with the number of clusters larger than or equal to 93, the algorithm introduced in this chapter settles on a final clustering with 26 clusters.

The email dataset contains 270 emails (samples) and 66 people (features). The final score for the evaluation function with 26 clusters is $J = 52.19$ out of the maximum possible value 66. Even if we knew the correct number of clusters (which we do not) and used it with k-means, the evaluation function introduced in this chapter would be $J_{k-means} = 32.91$, which is significantly lower than 52.19. This represents an improvement in cluster quality of 29.21%. The score reflects approximately the number of people who belong only to a single cluster. Email clusters for the email dataset are displayed in Fig. 10.9. Improvement over k-means was confirmed on emails of two other email datasets belonging to other people. Due to privacy issues, the email dataset is not shared as it belongs to a private company.

Remark 2 (Interesting detail) While the maximum evaluation function value can be achieved by assigning all samples to the same single cluster, when starting with a large number of clusters, the algorithm does not converge even for real life datasets

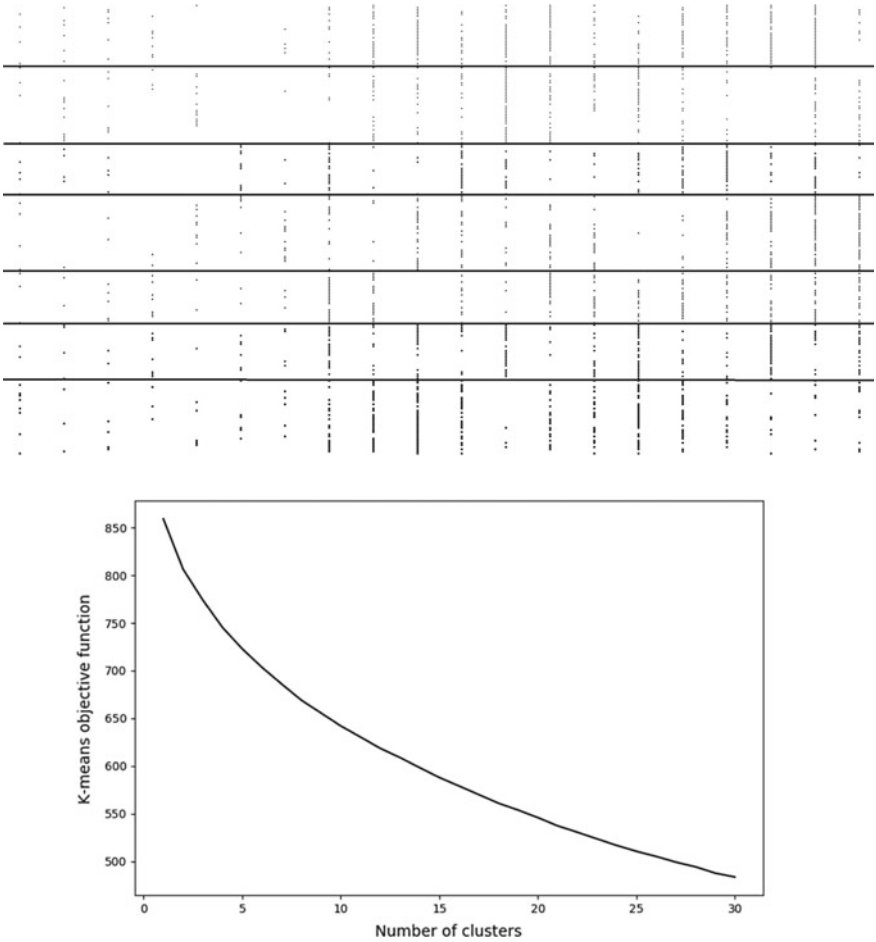

Fig. 10.6 Theoretical example 3—seven imperfect clusters clustered by k-means. (Top) On vertical axis we have different samples belonging to different clusters (separated by lines) and on horizontal axis we have different features. (Bottom) K-means evaluation function based on number of clusters—no clear break of evaluation function as for two previous theoretical examples

to a single cluster due to the use of back-propagation and the presence of local maximums along the way which result from reshuffling only a single sample at a time. This provides unexpected, but desired functionality. For the same reason, we are not able to start the algorithm with one cluster and subsequently subdivide the clusters, because this would only lower the evaluation function's value. Hence, the algorithm works only backwards.

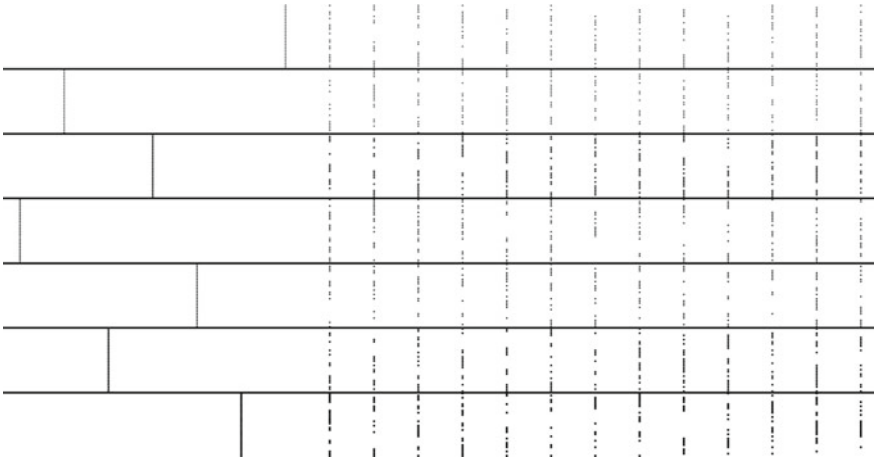

Fig. 10.7 Theoretical example 3—seven imperfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (separated by lines) and on horizontal axis we have different features. First 7 features (from left) are the significant features of respective clusters. Remaining 13 features are non-significant features, which do not belong to any cluster and are active 50% of time

10.5 Deeper Dive into the Algorithm

In this section we dive deeper into properties of the algorithm introduced in this paper (but for simplicity, we omit multiplying constant C_m and $\frac{\cdots+1}{\cdots+N}$ technique which is used to achieve non-zero probabilities).

As a first example, we consider the simple example of two clearly separated clusters, which features do not intersect as it is displayed in Table 10.1. Evaluation of the function is the same whether we have one or two clusters:

$$J_{2clusters} = n\left(\left(\frac{k}{k}\right)^P + \left(\frac{0}{k}\right)^P\right) + m\left(\left(\frac{l}{l}\right)^P + \left(\frac{0}{l}\right)^P\right) = n + m = J_{1cluster}$$

But if we start with two clusters, the evaluation function will not allow joining these two clusters, as we move only one sample at a time and the evaluation function would not improve for different rounds r of algorithm ($J_r > J_{r+1}$). Consider moving sample from cluster 2 to cluster 1:

$$J_r = n + m > n + m\left(\left(\frac{l-1}{l}\right)^P + \left(\frac{1}{l}\right)^P\right) = J_{r+1}$$

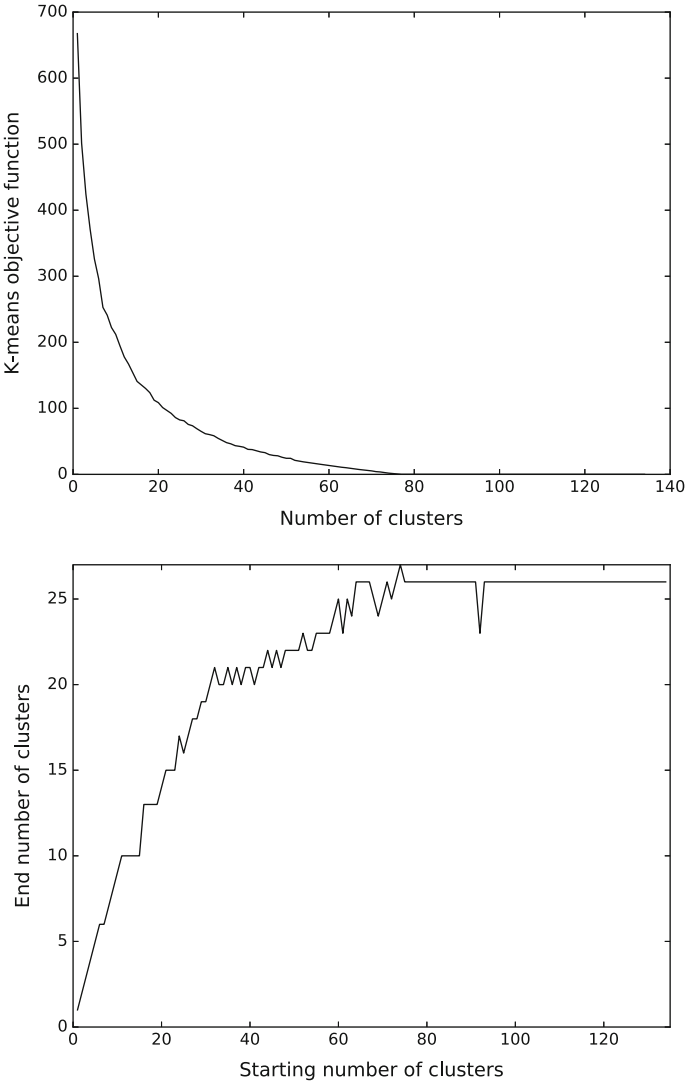

Fig. 10.8 (Top) K-means evaluation function depending on the number of clusters for real life example (emails). (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters

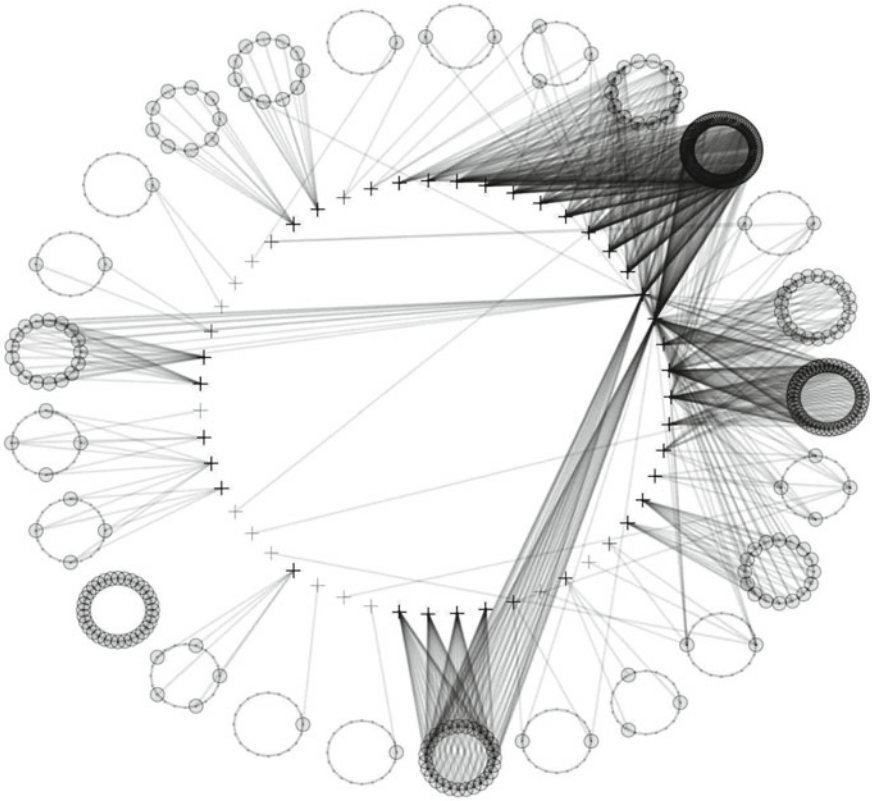

Fig. 10.9 Email clustering example. Emails are displayed as 'o' in different cluster circles. People are displayed as '+' and connections between emails and people included on these emails are displayed as lines between them

for $P > 1$, hence sample from cluster 2 will not move to cluster 1 and it could be shown the same way for sample from cluster 1 to cluster 2. This is the reason these two clusters will not be joined for $k > 1$ and $l > 1$.

Now, we explore a slightly more complicated situation when we have two clusters (Email Group 1 and 2). In one cluster, we have communication with all people (features 1 to $n + m$), and in the other email group we communicate only with a subgroup of people in the first email group (features $n + 1$ to $n + m$). This is shown in Table 10.2.

Now if we ask if these two clusters should be together or not, without optimizing our evaluation function by moving only one sample at a time, the answer is they should be together as

$$J_{2clusters} = n + m \left(\left(\frac{k}{k+l} \right)^P + \left(\frac{l}{k+l} \right)^P \right) < n + m = J_{1cluster}$$

Table 10.1 Two groups of emails with $k + l$ samples and $n + m$ people (features)—completely separated

Sample number	Email group	F_1	...	F_n	F_{n+1}	...	F_{n+m}
1	1	1	...	1	0	...	0
...	1	1	...	1	0	...	0
k	1	1	...	1	0	...	0
k + 1	2	0	...	0	1	...	1
...	2	0	...	0	1	...	1
k + l	2	0	...	0	1	...	1

Table 10.2 Two groups of emails with $k + l$ samples and $n + m$ people (features)—intersected features

Sample number	Email group	F_1	...	F_n	F_{n+1}	...	F_{n+m}
1	1	1	...	1	1	...	1
...	1	1	...	1	1	...	1
k	1	1	...	1	1	...	1
k + 1	2	0	...	0	1	...	1
...	2	0	...	0	1	...	1
k + l	2	0	...	0	1	...	1

for $P > 1$, which we consider.

So the question to answer is why, when using back-propagation, even though $J_{1cluster} > J_{2clusters}$ for the situation in Table 10.2, we do not end up with one cluster. Consider we have two clusters, and we move one sample from group two to group one only if the evaluation function is increased:

$$\begin{aligned} J_r &= n + m\left(\left(\frac{k}{k+l}\right)^P + \left(\frac{l}{k+l}\right)^P\right) \\ &< n + m\left(\left(\frac{k+1}{k+l}\right)^P + \left(\frac{l-1}{k+l}\right)^P\right) = J_{r+1} \end{aligned}$$

which holds if

$$k^P + l^P < (k+1)^P + (l-1)^P$$

and for case when $P = 2$ it is if

$$l < k + 1.$$

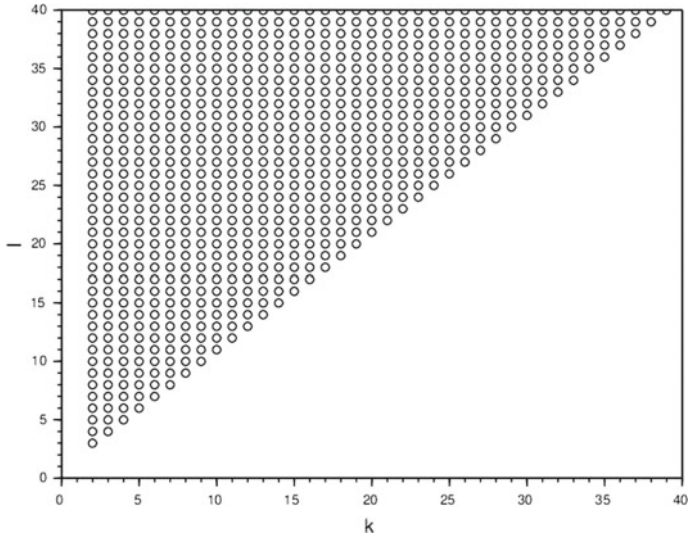

Fig. 10.10 The condition $k^P + l^P < (k+1)^P + (l-1)^P$ is the same for $P = 2$ and $P = 10$ for $k \in \{2, ..., 40\}$ and $l \in \{2, ..., 40\}$. Circle indicates condition is not fulfilled, while empty space opposite.

This means if there are enough samples l of the second group, at least $k + 1$, the member of group two will not move to group one and the clustering algorithm ends up with two clusters. Even for $P = 10$, the condition seems to be unchanged and is displayed in Fig. 10.10.

Another question for this example is why does the sample from group one not move to group 2. This can be explained by

$$\begin{aligned} J_r &= n + m\left(\left(\frac{k}{k+l}\right)^P + \left(\frac{l}{k+l}\right)^P\right) \\ &> n\left(\left(\frac{k-1}{k}\right)^P + \left(\frac{1}{k}\right)^P\right) + m\left(\left(\frac{k-1}{k+l}\right)^P + \left(\frac{l+1}{k+l}\right)^P\right) = J_{r+1} \end{aligned}$$

Even this simple situation can get complicated, but for $n \gg m$, we can simplify the equation to

$$\frac{J_r}{n} \approx 1 > \left(\left(\frac{k-1}{k}\right)^P + \left(\frac{1}{k}\right)^P\right) \approx \frac{J_{r+1}}{n}$$

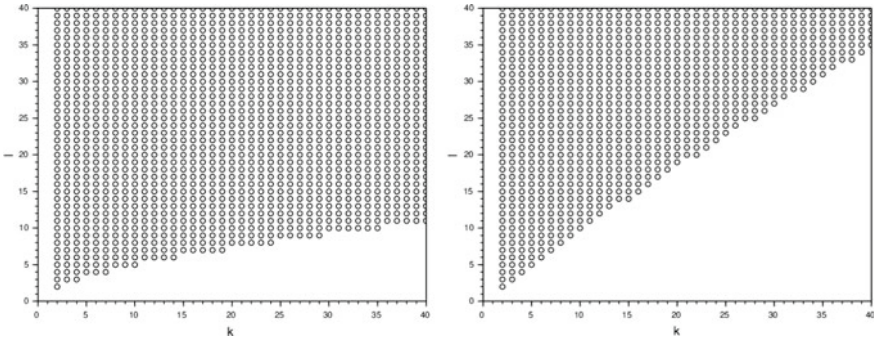

Fig. 10.11 The condition $J_r = n + m\left(\left(\frac{k}{k+l}\right)^P + \left(\frac{l}{k+l}\right)^P\right) > n\left(\left(\frac{k-1}{k}\right)^P + \left(\frac{1}{k}\right)^P\right) + m\left(\left(\frac{k-1}{k+l}\right)^P + \left(\frac{l+1}{k+l}\right)^P\right) = J_{r+1}$ is not the same for $P = 2$ (left) and $P = 10$ (right) for $k \in \{2, ..., 40\}$ and $l \in \{2, ..., 40\}$. For this example, we chose $n = m = 10$. Circle indicates condition is not fulfilled, while empty space opposite

which is true for $P > 1$ and $k > 1$. So if n is significantly larger than m, we have no reason to move the sample from group 1 to group 2. When this condition is fulfilled versus not is displayed in Fig. 10.11 also for case $n = m = 10$.

This is when parameter P starts to play its role unlike before (Fig. 10.10) and whether two clusters are joined or not depends on P.

10.6 Conclusions

We introduced a novel clustering algorithm POPC, which uses powered outer probabilities and works backwards from a large number of clusters to the optimal number of clusters. On three theoretical examples, we show that POPC converges to the expected number of clusters. In a real life example with email data, we show that it would be difficult to determine the optimal number of clusters based on the k-means evaluation score, but when the algorithm introduced in this chapter is used, it settles on the same number of clusters as if we had started with a large enough initial number of clusters. Importantly, the clusters are of higher quality in comparison with those produced by k-means even if we happened to know the correct number of clusters ex ante. Software *Small Bang*, which clusters emails and uses POPC algorithm, can be downloaded from [8].

Acknowledgements The authors would like to thank David James Brunner for many fruitful discussions on knowledge workers information overload as well as proofreading of the first draft. The authors would also like to thank anonymous reviewers for providing feedback which led to significant improvement of this chapter.

References

1. J. Hartigan, *Clustering Algorithms* (Wiley, 1975)
2. R. Xu, D. Wunsch, Survey of clustering algorithms. IEEE Trans. Neural Netw. **16** (2005)
3. G. Milligan, M. Cooper, An examination of procedures for determining the number of clusters in a data set. Psychometrika **50** (1985)
4. G. Frahling, C. Sohler, A fast k-means implementation using coresets. Int. J. Comput. Geom. Appl. **18** (2008)
5. P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Comput. Appl. Math. **20** (1987)
6. P. Taraba, Powered outer probabilistic clustering, in *Proceedings of the World Congress on Engineering and Computer Science 2017, 25–27 October, 2017, San Francisco, USA*. Lecture Notes in Engineering and Computer Science (2017), pp. 394–398
7. P. Taraba, Popc examples [Online] (2017), https://github.com/pepe78/POPC-examples
8. P. Taraba, Small bang [Online] (2017), http://www.frisky.world/p/small-bang.html