Proceedings of the World Congress on Engineering and Computer Science 2017 Vol I
WCECS 2017, October 25-27, 2017, San Francisco, USA

Powered Outer Probabilistic Clustering

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.**

***Index Terms*—clustering, probabilities, optimal number of clusters, binary valued features, emails.**

I. 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 one can mention [1], where 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 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 compare 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 two theoretical datasets: one with clusters that are clearly defined and a second with clusters that have randomly generated mistakes. We then show that the algorithm introduced in this paper converges to the expected number of clusters for both the perfect and the slightly imperfect cluster examples.

This paper is organized as follows. In section 2, we describe the POPC algorithm. In section 3, we confirm on three theoretical examples that the algorithm converges to the expected number of clusters. In section 4, we present an additional example of real email dataset clustering. In section 5, we draw conclusions.

Manuscript received May 18, 2017; revised June 07, 2017.

P. Taraba is currently not affiliated with any institution, residing in San Jose, CA 95129. (phone: +1-650-919-3827; e-mail: taraba.peter@mail.com).

II. Algorithm Description

The Powered Outer Probabilistic Clustering (POPC) algorithm relies on computing discounted probabilities of different features belonging to different clusters. In this paper, 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:

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

If we used (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:

$$J=\sum_{i=1}^{F} J(f_i)=\sum_{i=1}^{F}\sum_{k=1}^{N} p^P(cl(f_i)=k)\leq F, \qquad (2)$$

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 Figure 1 for different powers $P\in\{1,2,3,10\}$ and two clusters. For results reported in this paper 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

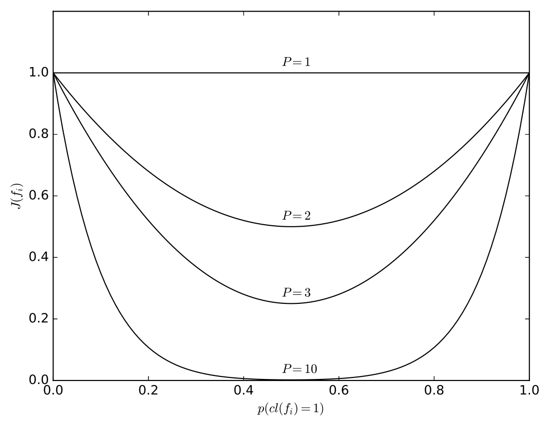


Fig. 1. Values of the evaluation function for a hypothetical case where there is one feature and two clusters. If the feature is distributed between two different clusters, the evaluation function J has a lower value (near 0.5 on the x-axis). In case the feature belongs to only one cluster, or mostly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

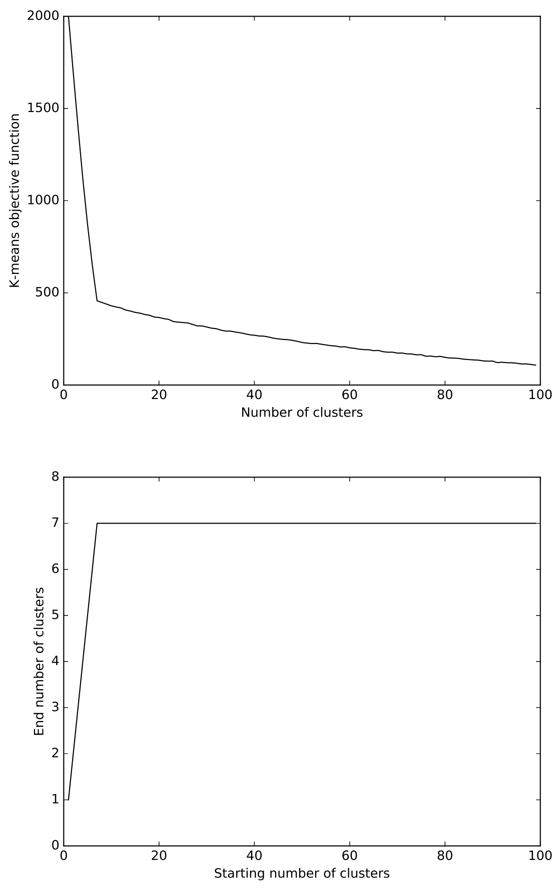


Fig. 2. (Top) K-means evaluation function depending on the number of clusters for example 1. (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters

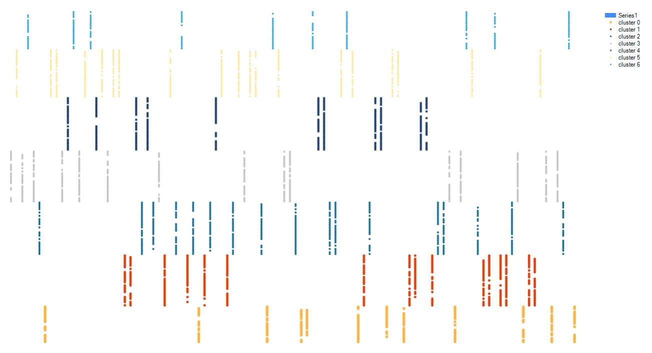


Fig. 3. Theoretical example 1 - seven perfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (colors) and on horizontal axis we have different features.

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 $k \in \{1, \dots, N\}$ 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 (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.*

III. Examples

In this section, we create three theoretical examples, one perfect, one slightly imperfect and one with 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 eighty percent of the time.

In Figure 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 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.

First example clustered with POPC algorithm is displayed in Figure 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

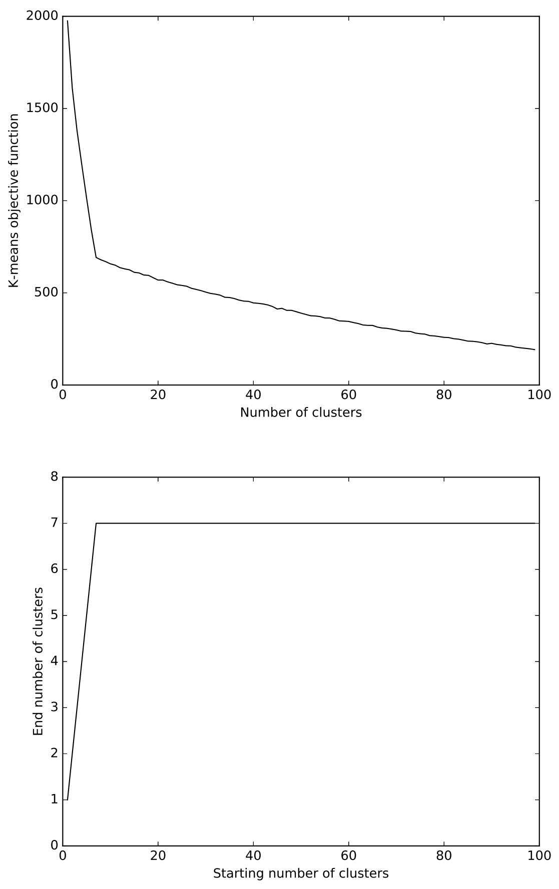


Fig. 4. (Top) K-means evaluation function depending on the number of clusters for example 2. (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters.

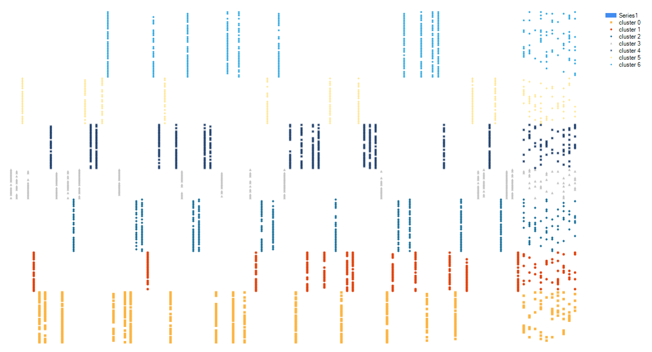


Fig. 5. Theoretical example 2 - seven imperfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (colors) and on horizontal axis we have different features. Last ten percent features (on right) are the features, which do not belong to any cluster and are active twenty percent of time.

active twenty percent of time. The results for example 2 are displayed in Figure 4. The POPC algorithm introduced in this paper once again yields the expected number of clusters.

Second example clustered with POPC algorithm is displayed in Figure 5. As shown, clusters are found as expected despite having 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. Remaining 13 features are random features, which do not belong to any cluster and are activate fifty per-

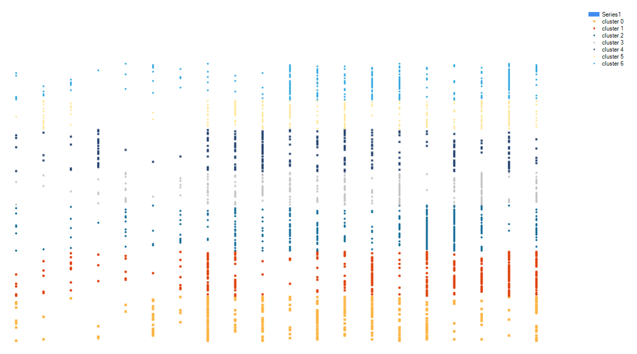


Fig. 6. Theoretical example 3 - seven imperfect clusters clustered by k-means. On vertical axis we have different samples belonging to different clusters (colors) and on horizontal axis we have different features.

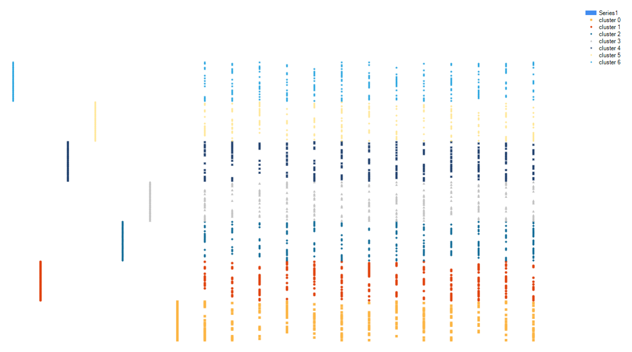


Fig. 7. Theoretical example 3 - seven imperfect clusters clustered by POPC. On vertical axis we have different samples belonging to different clusters (colors) and on horizontal axis we have different features. First 7 features (from left) are the significant features of respective clusters. Remaining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

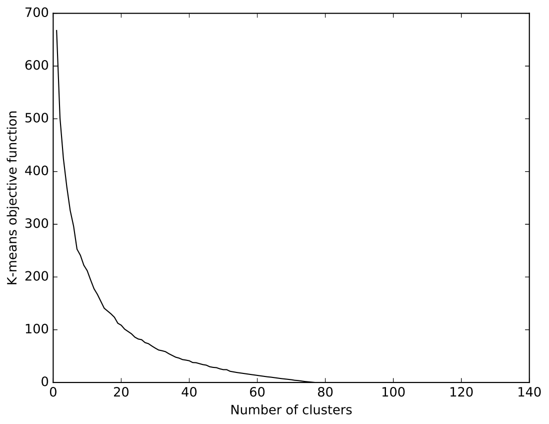
cent of time. This is main example, which shows why POPC algorithm is so useful. K-means algorithm due to majority of features being randomly active cannot find features which are significant features for clusters and finds clusters which are not the way we would expect them. Evaluation function $J_{k-means}$ is very close to 0 even if we knew we are looking for 7 clusters. This is displayed in Figure 6.

On 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 amount of features that are significant to respective clusters. This is displayed in Figure 7. This last theoretical example explains why in real life situations, POPC is not only able to find 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 paper can be generated with C# code which can be downloaded from [6]. Same code contains implementation of popc algorithm.

IV. 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 Figure 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



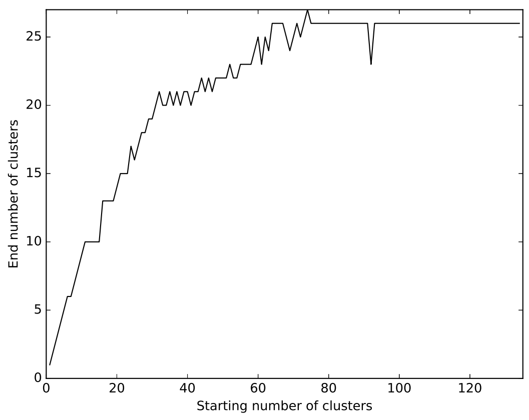


Fig. 8. (Top) K-means evaluation function depending on the number of clusters for real life example (emails). (Bottom) Number of clusters created by the POPC algorithm depending on the number of starting clusters

algorithm introduced in this paper 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 paper would be $J_{k-means} = 32.91$, which is significantly lower than 52.19. This represents an improvement in cluster quality of 29.21 percent. The score reflects approximately the number of people who belong only to a single cluster. Email clusters for the email dataset are displayed in Figure 9. Improvement over k-means was confirmed on emails of two other email datasets belonging to other people. Due to privacy issues, email dataset is not shared as it belongs to a private company.

Remark (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 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.*

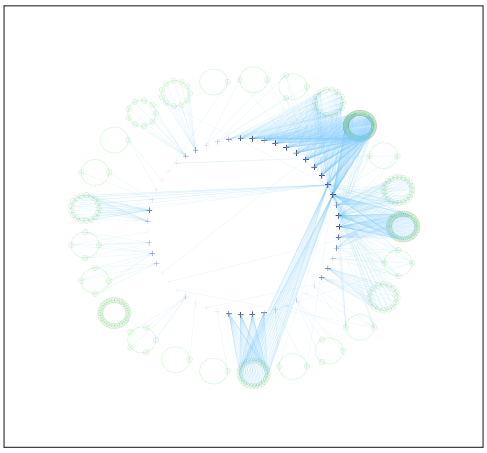


Fig. 9. Email clustering example. Emails are displayed as 'o' in different cluster circles, people are displayed as '+' and connections between emails and people included on these emails are displayed as blue lines in between them.

V. 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 paper is used, it settles on the same number of clusters if we start 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 [7].

Acknowledgment

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 paper.

References

[1] J. Hartigan, *Clustering Algorithms.* John Wiley & Sons, 1975.
[2] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, 2005.
[3] G. Milligan and M. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, 1985.
[4] G. Frahling and C. Sohler, "A fast k-means implementation using coresets," *Int. J. Comput. Geom. Appl.*, vol. 18, 2008.
[5] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Computational and Applied Mathematics*, vol. 20, 1987.

[6] P. Taraba. (2017) Popc examples. [Online]. Available: https://github.com/pepe78/POPC-examples
[7] ——. (2017) Small bang. [Online]. Available: http://www.frisky.world/p/small-bang.html