Towards Automated Configuration of Stream Clustering Algorithms

Matthias Carnein[1(✉)], Heike Trautmann[1], Albert Bifet[2], and Bernhard Pfahringer[2]

[1] University of Münster, Münster, Germany
{carnein,trautmann}@wi.uni-muenster.de
[2] University of Waikato, Hamilton, New Zealand
{abifet,bernhard}@waikato.ac.nz

Abstract. Clustering is an important technique in data analysis which can reveal hidden patterns and unknown relationships in the data. A common problem in clustering is the proper choice of parameter settings. To tackle this, automated algorithm configuration is available which can automatically find the best parameter settings. In practice, however, many of our today's data sources are data streams due to the widespread deployment of sensors, the internet-of-things or (social) media. Stream clustering aims to tackle this challenge by identifying, tracking and updating clusters over time. Unfortunately, none of the existing approaches for automated algorithm configuration are directly applicable to the streaming scenario. In this paper, we explore the possibility of automated algorithm configuration for stream clustering algorithms using an ensemble of different configurations. In first experiments, we demonstrate that our approach is able to automatically find superior configurations and refine them over time.

Keywords: Stream clustering · Automated algorithm configuration · Algorithm selection · Ensemble techniques

1 Introduction

One of the hardest challenges for data scientists is to find a suitable algorithm as well as appropriate parameter settings to solve a given problem. This is even more challenging when working with data streams which do not allow re-evaluations and a posteriori optimisation. In addition, data streams can change over time and parameters need to be adapted accordingly. These problems considerably prevent the widespread adoption of stream mining algorithms in the real-world. A popular tool in stream mining are stream clustering algorithms which aim to identify and track clusters, i.e. groups of similar objects in a stream [5]. In this paper we propose an innovative, ensemble-based approach that allows to automatically find and adapt optimal parameters for data stream clustering algorithms. In each iteration, promising configurations are used to sample new

P. Cellier and K. Driessens (Eds.): ECML PKDD 2019 Workshops, CCIS 1167, pp. 137–143, 2020.
https://doi.org/10.1007/978-3-030-43823-4_12

ones that can replace inferior configurations. In first experiments, we demonstrate that our approach can considerably improve clustering results. To the best of our knowledge, this is the first attempt to apply automated algorithm configuration to data streams as well as stream clustering.

2 Automated Algorithm Configuration

Automated algorithm configuration aims at automatically determining the best parameter settings for a given scenario [8,9]. Popular approaches for this are `SMAC` [7] or `irace` [10]. Unfortunately, none of these approaches is directly applicable to the streaming scenario. These algorithms are mostly set-based and do not focus on single instances. In addition, they require multiple evaluations of the data and usually require static and stationary data without concept drift. This would require to apply the parameter configuration a posteriori [4] or on an initial sample of the stream which is both undesirable.

In this paper, we transfer the idea of automated algorithm configuration to stream clustering. Similar challenges and prior work can be found in the algorithm *selection* and stream *classification* literature. In [12], for example, the authors create an ensemble of different stream classification algorithms. All algorithms are trained simultaneously on the same data stream. The stream is divided into windows of specified size and for every window, meta-features such as standard deviation or entropy are computed. Based on these features and the performance of the classifiers, a meta-classifier is trained to predict which classifier is most suited to classify the next window. In [11,13], the `BLAST` algorithm is introduced which uses the same ensemble strategy and inspired this work conceptually. However, instead of using a meta-classifier it always selects the classifier which performed best on the last window to predict the next window.

3 Automated Configuration of Stream Clustering Algorithms

In this section we propose `confStream`, an ensemble-based approach for automated algorithm configuration in stream clustering, focusing on the online phase of the algorithm, i.e. optimising the micro-cluster representation. In particular, our aim is to maintain, adapt and improve an ensemble of different configurations over time. For this, our algorithm requires a given starting configuration as well as predefined parameter ranges. The main idea of `confStream` is summarised in Fig. 1. In order to apply the ensemble strategy, we process the stream in windows of fixed size h. Observations within a window are processed one by one and used to train all algorithms in the ensemble simultaneously. At the end of the window, the clustering performance of every configuration is evaluated (Step 1). For example, the Silhouette Width measures for an observation i, the average similarity to observations in its own cluster $a(i)$ and compares it to the average similarity

to its closest clusters $b(i)$. It is defined as: $s(i) = (b(i) - a(i))/(\max\{a(i), b(i)\})$. While the Silhouette Width is state-of-the-art, there are also other evaluation measures which are equally applicable here. In order to evaluate our ensemble, we compute the average Silhouette Width for all observations of the last window for the different configurations. The clustering algorithm that performed best becomes the active clusterer or *incumbent* for the next iteration. The incumbent represents the current clustering result of the ensemble and will be used throughout the next window.

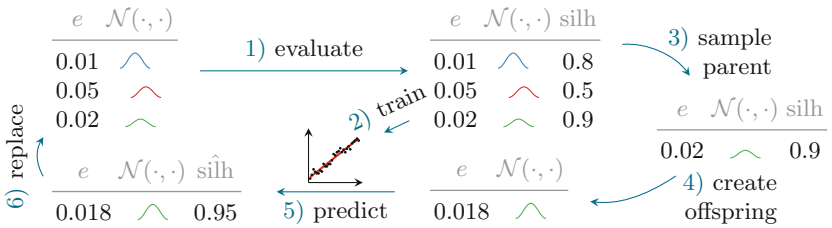

Fig. 1. The performance of algorithms in the ensemble is evaluated and used to train a regression model. Afterwards, one algorithm is sampled to create an offspring. If its predicted performance is high enough, it replaces one of the algorithms in the ensemble.

In a next step, the configurations of the algorithms and their performances are used to train a regression model (Step 2). The regression model is supposed to learn how well certain configurations perform. This is later used in order to determine whether a new configuration is promising and should be incorporated into the ensemble. In our case, we use an Adaptive Random Forest (ARF) regression as proposed in [6]. The ARF is a natural choice, since it is a streaming algorithm which can be trained over time. In order to generate new configurations, one configuration is sampled from the ensemble as a parent (Step 3). The sampling is performed proportionally to the performance of the algorithms such that better performing configurations are more likely to be selected.

The selected configuration is then used as a parent in order to derive a new configuration from it (Step 4). For this, we use a similar strategy as `irace` [10]. In particular, every parameter i of every configuration has an associated truncated normal distribution $\mathcal{N}(\mu_i, \sigma_i)$ with expectation μ_i and standard deviation σ_i. In order to sample a new parameter value, we place the expectation of the distribution at the position of the parent. The distribution has an upper bound U and a lower bound L which are set to the boundaries of the parameter range. The standard deviation σ_i is initialised with $(U - L)/2$ for every parameter and slowly reduced over time. For this, we use a fading strategy which exponentially decreases the standard deviation over time: $\sigma_{t+1} = \sigma_t \cdot 2^{-\lambda}$. The underlying idea is that the configuration will converge to the optimum over time and the smaller standard deviation allows to explore this area better. To account for concept drift, we occasionally explore the full parameter range by resetting the standard

deviation to its initial value with a probability p. While we only consider continuous parameters here, the approach could be extended to categorical parameters, e.g. by drawing a new value from a list of probabilities where the probability of the winning category is increased [10].

Next, the performance of the new configuration is predicted based on the regression model (Step 5). If the size of the ensemble is smaller than e_{size} the new configuration is added directly to the ensemble. If the ensemble is full but the predicted performance is better than a performance in the ensemble, the new configuration can be incorporated into the ensemble (Step 6). For this, we use proportional sampling again, where bad solutions are more likely to be replaced. This is supposed to maintain a higher diversity in the ensemble than removing the worst solutions first. As two special cases, we never remove the incumbent and always remove solutions first which did not yield a valid clustering solution in the last window. We consider solutions invalid when the solution contains only a single cluster or the algorithm failed. The generation of new configurations can be repeated until a user-chosen number of configurations e_{new} has been generated. Afterwards, the next window of the stream is processed. In summary, our approach has 5 main parameters itself: the window size h, the fading parameter λ, the ensemble size e_{size}, the number of new configurations e_{new} and the exploration probability p. We note that our ensemble approach is slower than running individual algorithms. Nevertheless, in our experiments the algorithm was fast enough to work in real-time since the algorithms can be trained in parallel.

4 Evaluation

In order to evaluate our approach we implemented a proof-of-concept in Java[1] as a clustering algorithm for the MOA framework [2]. For our analysis, we consider a simple configuration scenario for the `DenStream` [3] algorithm, one of the most popular stream clustering algorithms [5]. First, we evaluate the performance of `DenStream`'s default configuration $\epsilon = 0.02$, $\beta = 0.2$, $\mu = 1$. We then compare this with our ensemble approach, where we start with the same configuration but optimise the distance threshold ϵ in its full value range $[0, 1]$. We set the ensemble size $e_{\text{max}} = 25$, fading $\lambda = 0.05$, reset probability $p = 0.001$ and evaluate the solutions every $h = 1000$ data points. After each window, we create $e_{\text{new}} = 10$ new configurations. In order to evaluate the quality of the clustering algorithms, we use the Silhouette Width. Since we want to evaluate cluster quality over time, we evaluate the quality for windows of 1000 observations in our experiments. We evaluate both algorithms, i.e. the default parametrisation of `DenStream` vs. the configured version `confStream`, using a Random Radial Basis Function (RBF) stream [1], `sensor` stream[2], and `covertype` data set[3]. All data sets are popular choices in the (stream) clustering literature.

[1] Implementation available at: https://www.matthias-carnein.de/confStream.
[2] Dataset available at: http://db.csail.mit.edu/labdata/labdata.html.
[3] Dataset available at: http://archive.ics.uci.edu/ml/datasets/Covertype.

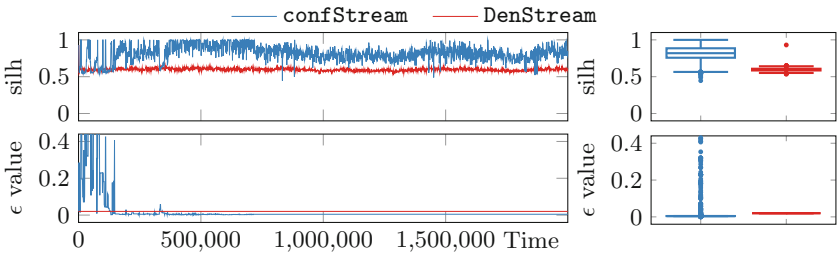

Fig. 2. Development of Silhouette Width and ϵ parameter for the `Random RBF` stream

Figure 2 shows the Silhouette Width for every window of the `Random RBF` stream. The boxplot on the right summarises the range of values. It is obvious, that our ensemble approach quickly improves the default configuration and remains superior for the vast majority of the stream. When observing the development of the ϵ parameter in our ensemble, it becomes obvious how `confStream` first explores a large range of values. Over time, the algorithm reduces the standard deviation of the distributions in order to explore promising regions further before settling on roughly $\epsilon = 0.005$. Note that this is similar to the initial configuration of $\epsilon = 0.02$. Nevertheless, the performance is vastly improved which also highlights how sensitive stream clustering algorithms are to different configurations.

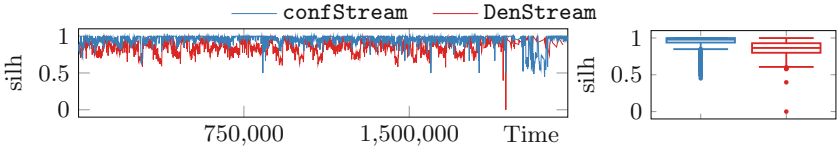

Fig. 3. Silhouette Width for the `sensor` stream

For the other data sets, we observe similar trends. Figure 3 shows the results for the `sensor` data stream. Again, `confStream` quickly improves upon the initial configuration and yields better results with a near-perfect median silhouette width of 0.98. While the default configuration also yields a good results, it is stronger affected by concept drift in the data stream. In particular, the `sensor` data set exhibits a periodic pattern of day and night. `confStream` is less affected by this since it adapts to the changing scenarios. For the `covertype` data set the difference is most obvious. Using the default configuration, `DenStream` is not able to produce a single valid solution with at least two clusters throughout the entire stream. While `confStream` starts with the same initial configuration, it quickly adapts and is able to produce very high quality (Fig. 4). This also shows in the development of the parameter value which quickly changes from the initial value $\epsilon = 0.02$ and explores more suitable values between $\epsilon = 0.1$ and $\epsilon = 0.2$.

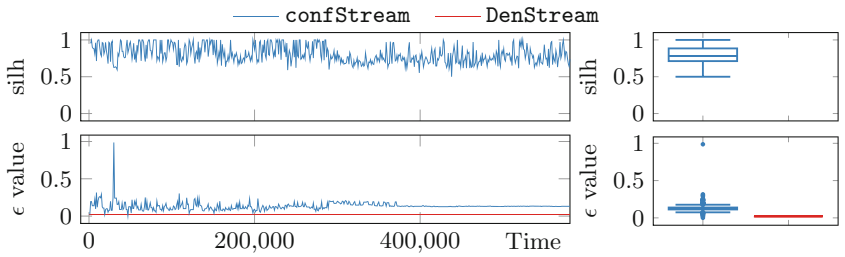

Fig. 4. Development of Silhouette Width and ϵ parameter for the `covertype` data set

Overall, these initial results show that our ensemble strategy produces vastly better clustering solutions than the default configuration. In particular, changes and improvements are made over time which allow for adapting to stream characteristics and/or unsuitable starting configurations.

5 Outlook and Conclusion

In this paper we explored the possibility of automated algorithm configuration for stream clustering. By training an ensemble of algorithms in parallel and deriving new configurations from promising solutions, we are able to efficiently adapt the configuration over time. Results for a configuration problem with one parameter have shown to improve the overall clustering result considerably in comparison to its default configuration. In future work, we will extend our approach and evaluation beyond a single algorithm and parameter. In particular, we will optimise multiple parameters simultaneously, which can be of different types, such as categorical or integer. Ultimately, we also aim to include different kinds of stream clustering algorithms into the ensemble approach resulting in per-instance algorithm selection and configuration on streaming data.

References

1. Bifet, A., Gavalda, R., Holmes, G., Pfahringer, B.: Machine Learning for Data Streams with Practical Examples in MOA. MIT Press, Cambridge (2018)
2. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
3. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Proceedings of the Conference on Data Mining (SIAM 2006), pp. 328–339 (2006)
4. Carnein, M., Assenmacher, D., Trautmann, H.: An empirical comparison of stream clustering algorithms. In: Proceedings of the ACM International Conference on Computing Frontiers (CF 2017), pp. 361–365. ACM (2017)
5. Carnein, M., Trautmann, H.: Optimizing data stream representation: an extensive survey on stream clustering algorithms. Bus. Inf. Syst. Eng. **61**(3), 277–297 (2019). https://doi.org/10.1007/s12599-019-00576-5

6. Gomes, H.M., Barddal, J.P., Ferreira, L.E.B., Bifet, A.: Adaptive random forests for data stream regression. In: Proceedings of the 26th European Symposium on Artificial Neural Networks (ESANN 2018) (2018)
7. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
8. Hutter, F., Kotthoff, L., Vanschoren, J. (eds.): Automated Machine Learning. TSS-CML. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05318-5
9. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: survey and perspectives. Evol. Comput. (ECJ) **27**(1), 3–45 (2019)
10. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016)
11. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Having a blast: meta-learning and heterogeneous ensembles for data streams. In: Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM 2015), pp. 1003–1008 (2015)
12. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: Algorithm selection on data streams. In: Džeroski, S., Panov, P., Kocev, D., Todorovski, L. (eds.) DS 2014. LNCS (LNAI), vol. 8777, pp. 325–336. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11812-3_28
13. van Rijn, J.N., Holmes, G., Pfahringer, B., Vanschoren, J.: The online performance estimation framework: heterogeneous ensemble learning for data streams. Mach. Learn. **107**(1), 149–176 (2017). https://doi.org/10.1007/s10994-017-5686-9