APReL: A Library for Active Preference-based Reward Learning Algorithms

Erdem Bıyık
Electrical Engineering
Stanford University
ebiyik@stanford.edu

Aditi Talati
Computer Science
Stanford University
atalati@stanford.edu

Dorsa Sadigh
Computer Science & Electrical Engineering
Stanford University
dorsa@cs.stanford.edu

***Abstract*—Reward learning is a fundamental problem in human-robot interaction to have robots that operate in alignment with what their human user wants. Many preference-based learning algorithms and active querying techniques have been proposed as a solution to this problem. In this paper, we present APReL, a library for active preference-based reward learning algorithms, which enable researchers and practitioners to experiment with the existing techniques and easily develop their own algorithms for various modules of the problem. APReL is available at https://github.com/Stanford-ILIAD/APReL.**

***Index Terms*—reward learning, active learning, software library, preference-based learning**

I. Introduction

As robots enter our daily lives, we want them to act in ways that are aligned with our preferences and goals. Learning a reward function that captures human preferences about how a robot should operate is a fundamental robot learning problem that is the core of the algorithms discussed in this work.

There are a number of different information modalities that can be avenues for humans to convey their preferences to a robot. These include demonstrations [1, 2], physical corrections [3, 4], observations [5], language instructions and narrations [6], ratings [7, 8], comparisons and rankings [9, 10, 11], each of which has its own advantages and drawbacks. For example, demonstrations are difficult to collect when the robot has high degrees of freedom, and physically providing corrections on a robot can pose safety challenges. Observations are difficult to learn from, as they are provided by agents with different dynamics and/or objectives. Language instructions and narrations do not provide detailed information about how the task should be done. While ratings do not suffer from these issues, they introduce higher cognitive loads on the users compared to comparisons and rankings. Learning human preferences using comparisons and rankings is well-studied outside of robotics [12], and the paradigm of learning human preferences based on comparisons and rankings shows promise in robotics applications as well [13, 10, 11].

However, preference-based learning poses another important challenge: each comparison or ranking gives a very small amount of information. For example, a pairwise comparison between a trajectory of a car that speeds up at an intersection with another trajectory that slows down gives at most one bit of information. Hence, it becomes critical to optimize for what the user should compare or rank. To this end, prior works have developed several active learning techniques to improve data-efficiency of preference-based learning by maximizing the information acquired from each query to the user. In this paper, we present a *novel* and *unified* Python library, APReL, that enables researchers and practitioners to easily use and experiment with many existing active preference-based reward learning techniques that effectively query humans and are used in robotics applications. Therefore, it is very *relevant* to both human-robot interaction and robot learning fields. For the *ease of use*, APReL enables these various techniques to be applied on any simulation environment that is compatible with the standard OpenAI Gym structure [14].

We first go over the techniques included in APReL in Section II. We also discuss more recent studies that we are actively working to include in later versions of APReL. Next, Section III presents a unifying notation, and Section IV briefly reviews the techniques. Section V presents the modular structure of APReL to guide the researchers for implementing new techniques. Finally, Section VI concludes the paper.

II. Related Work

In this section, we go over prior work that APReL supports, and discuss more recent studies which we plan to include in APReL's later versions.

Active Preference-Based Learning. Active preference-based reward learning is a well-studied problem in machine learning and robotics. Sadigh et al. [9] modeled the reward as a linear function of some trajectory features and proposed using a *volume removal* based acquisition function to select pairwise comparison queries, which are in the form of "do you prefer trajectory A or B?". Katz, Le Bihan, and Kochenderfer [15] formulated the query selection as a multi-objective optimization problem to maximize both the likelihood and the *disagreement* between trajectories.

While the disagreement-based method was only for pairwise comparisons, Palan et al. [16] and Biyik et al. [17] extended the volume removal method to best-of-K queries: "which of the K trajectories is the best?". Later, Biyik et al. [18] found the volume removal optimization is ill-posed and identified failure cases. They proposed maximizing *mutual information*

This work is funded by NSF grants #1849952 and #1941722, FLI grant RFP2-000, and DARPA.

between the response to the query and the reward function. Finally, Wilde, Kulić, and Smith [19] and Tucker et al. [20], again focusing on pairwise comparisons only, developed alternative acquisition functions based on a regret formulation and Thompson sampling, respectively. These further improve the data-efficiency when the goal is to find the optimal trajectory rather than the underlying reward function. Although APReL focuses on reward learning, we include these acquisition functions in the library as additional benchmarks.

Batch Active Learning. While these works focus on the data-efficiency of learning, another line of work built upon them to improve time-efficiency by generating batches of queries at a time. This reduces the time taken to generate each query at the expense of increased number of queries needed to learn the reward function. When generating a batch, however, simply selecting the top few queries in terms of the acquisition functions will create a batch of nearly-identical queries. Most of these queries will be redundant after the user responds to one of them, and the data-efficiency will significantly decrease. To handle this problem, Biyik and Sadigh [21] adopted various heuristic methods to jointly maximize informativeness and diversity. They later developed a method based on Determinantal Point Processes (DPP) [22] which are used for tractably sampling informative and diverse batches in a more formal way [23]. APReL enables both the heuristic methods and the DPP method in addition to the non-batch setting.

Integrating Comparisons and Demonstrations. Finally, recent works attempted to combine other sources of information with comparisons and rankings. Palan et al. [16] and Biyik et al. [13] integrated expert demonstrations into the prior to further improve data-efficiency. Using their methods based on Bayesian inverse reinforcement learning [24], APReL enables researchers to optionally include demonstrations (or other forms of offline data) into the learning pipeline and still use all other functionalities.

Although APReL covers many of the techniques in the literature, preference-based reward learning is still an active research field. Hence, there exist recent methods that we are actively working towards adding into the library. These include scale feedback questions [25]; reward functions that are non-stationary [26], multimodal [27], modeled as Gaussian processes [28] (which was later combined with ordinal data by Li et al. [29]) or neural networks [30]; and a new acquisition function that enables the robot to reveal what it has already learned, while asking questions [31].

III. Problem Formulation

APReL synthesizes various techniques discussed in Section II. To this end, we start with formulating the problem with a unifying notation.

MDP. We describe the evolution of the agent as a discrete-time Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, f, r \rangle$. At each time step t, the agent is at state $s_t \in \mathcal{S}$ and takes action $a_t \in \mathcal{A}$. Then, f represents the agent's dynamics distribution such that $s_{t+1} \sim f(s_t, a_t)$. Finally, r is the reward function, so that at every time step t, the agent receives a reward $r(s_t, a_t)$. A trajectory $\xi \in \Xi$ in this MDP is a sequence $\big((s_t, a_t)\big)_t$ of state-action pairs that correspond to a roll-out in the MDP. The goal of the MDP agent is to maximize the expected cumulative reward over its trajectories.

Reward. The reward function is what we are trying to learn, as it encodes how the human wants the agent to behave. We assume it is linear in some state-action features that are known: $r(s_t, a_t) = \omega^\top \phi(s_t, a_t)$.[1] The reward of a trajectory ξ is then based on the cumulative features:

$$R(\xi) = \sum_{(s,a)\in\xi} r(s,a) = \omega^\top \sum_{(s,a)\in\xi} \phi(s,a) = \omega^\top \Phi(\xi)\,. \quad (1)$$

In fact, as we elicit user preferences by asking them to compare or rank trajectories, rather than states or actions, this formulation allows us to define the features more generally. As an example, one can directly design Φ, which might treat states at different time steps differently, rather than designing ϕ. Since Φ is given, we only have to learn the weights ω.

Preferences. The human gives information by selecting their preference among a query $Q = \{\xi_1, \xi_2, \ldots, \xi_K\}$ of K trajectories. The human noisily picks their favorite $q \in Q$, which optimizes their reward function. In addition, differently from the existing work, APReL also allows users to give full rankings of the trajectories in Q. We use these preferences and rankings to learn the human's reward function.

Demonstrations. Expert demonstrations of the optimal behavior may also be available to initialize the learning process. Each demonstration is a trajectory ξ^D, and the human may input a set of demonstrations as $\mathcal{D}$.

Problem. Our overall objective is to learn the reward function with as few data points as possible. For this, we should:

- Learn from user preferences after optionally initializing our model based on expert demonstrations,
- Actively generate preference/ranking queries that are optimized to be informative for the learning model, and
- Actively generate *batches* of queries to alleviate the computational burden of active query generation.

IV. Methods

In this section, we briefly review the methods included in APReL to solve each of the three parts of the problem.

A. Learning from Demonstrations and Preferences

Biyik et al. [13] proved that it is optimal to initialize the reward prior with demonstrations and then shift to actively collected preference data to update the posterior. Hence, we start with the demonstrations $\mathcal{D}$, which are collected offline, to generate a prior belief over the true reward weights ω:

$$\rho^0(\omega) \propto P(\mathcal{D} \mid \omega) P(\omega) = P(\omega) \prod_{\xi^D \in \mathcal{D}} P(\xi^D \mid \omega)\,, \quad (2)$$

[1] This is a common assumption to make the Bayesian learning approaches tractable. We are actively working on extending these to more expressive reward models in APReL. See, for example, [10, 28, 30].

where the assumption is that these demonstrations are conditionally independent. The probability of a demonstration, i.e., $P(\xi^D \mid \omega)$, is modeled with a computational user model, selected by the system designer.

Then, the belief distribution is updated with proactively generated preference/ranking queries. We first look at how we update the model given these query responses. We will consider how to generate the queries in the next subsection.

The robot asks a new query at each iteration i, starting from $i = 0$. We denote the i^{th} query to be Q_i, and the human response to that query q_i. Then, using Bayes' theorem and again with the conditional independence assumption,

$$\rho^{i+1}(\omega) \propto P(q_i \mid Q_i, \omega) \cdot \rho^i(\omega)\,, \tag{3}$$

which again requires a computational human response model for $P(q_i \mid Q_i, \omega)$. APReL allows implementing and experimenting with different human models. After learning from the query responses, the posterior belief keeps a distribution for the reward function (or more generally, the human model), which can be used to optimize the robot's behavior.

APReL allows users the option to provide demonstrations to initialize our belief, or to learn solely from preferences, without any provided demonstrations, in which case $\rho^0(\omega) = P(\omega)$. Having presented the way the initial belief is generated from the demonstrations and updated with the human responses to the queries, we now proceed to the second problem to consider how to actively generate the queries.

B. Actively Generating Queries

The goal of the robot is to generate queries that accurately and efficiently update its estimate of ω, so it should minimize the number of questions the user must answer.

In addition to allowing the implementation of new acquisition functions, APReL readily provides multiple different options for this active learning problem: volume removal, disagreement, information gain, regret and Thompson sampling. We would ideally find the best *adaptive sequence* of queries for the robot to get accurate, fine-grained information about ω. However, since adaptively reasoning about a sequence of queries is an NP-hard problem [32], the techniques in the literature proceed in a greedy fashion: at each iteration i, we choose Q_i while considering only the next posterior ρ^{i+1}.

Volume Removal. Volume removal is a strategy for selecting queries that maximize the expected difference between the prior and the *unnormalized* posterior [9, 17]. Intuitively, it searches queries Q_i where each possible answer q_i is equally likely given our current belief over ω, which corresponds to the queries where we are most uncertain about which behavior the human will prefer.

Disagreement. By trying to generate uncertain queries for the human, volume removal formulation often presents queries with very similar trajectories. To overcome this, Katz, Le Bihan, and Kochenderfer [15] proposed a multi-objective approach based on a disagreement metric, particularly for pairwise comparison queries. In that approach, the goal is to select two ω's to maximize both their likelihood and disagreement. The optimal ω's are then given to a planner, e.g., a reinforcement learning algorithm, which gives the optimal trajectories with respect to those ω's. These trajectories form the optimal disagreement query. While APReL handles the intermediate planning problem by selecting the trajectories from a predefined trajectory set, its modular structure allows users to implement their own planners.

Mutual Information. Motivated by the same issue of trajectories being too similar with volume removal, Biyik et al. [18] identified failure cases and proposed maximizing the mutual information between the user response and the reward function instead, which they showed to yield both informative and easy-to-answer questions.

In addition, they showed the learning efficiency can be improved if the users are allowed to respond "About Equal" to the pairwise comparison queries (called *weak* comparison queries) and these responses are also utilized for learning. APReL also allows having weak comparison queries.

Regret. For a slightly different formulation, where the goal is to find the optimal behavior rather than learning the reward function, Wilde, Kulić, and Smith [19] proposed using pairwise comparison queries that maximize a measure of regret between the ω's that lead to the trajectories in the query. Similar to the disagreement formulation, this acquisition function first optimizes ω's and then uses a planner to get the trajectories in the query.

Thompson Sampling. Finally for a similar modified problem, Tucker et al. [20] proposed using a Thompson sampling approach so that the regret will be minimized during the querying process, which implies the optimal trajectory will be quickly learned.

C. Generating Batches of Queries

When we are fine-tuning our estimate of ω, we want to do so in a way that minimizes the time spent by the human expert. When we generate a new query in between each user response, we spend a significant amount of time to generate the next query. A more time-efficient option is to generate batches of queries at once, so that the time cost will reduce.

APReL has a variety of options for which batch method to use if we are generating batches of queries. We define b to be the number of queries in our batch, and briefly go over each method. First four methods were proposed by Biyik and Sadigh [21], and the DPP-based method by their follow-up work [22].

Greedy. The top b queries that individually optimize the acquisition function form batch. Though computationally efficient, this causes redundancy as we previously discussed.

Therefore, the other batch methods try to increase the diversity in the queries. For this, they first take the top B queries with respect to the acquisition function (so that non-informative queries will not be selected). Then, they try to select b queries out of this reduced set that maximize the total

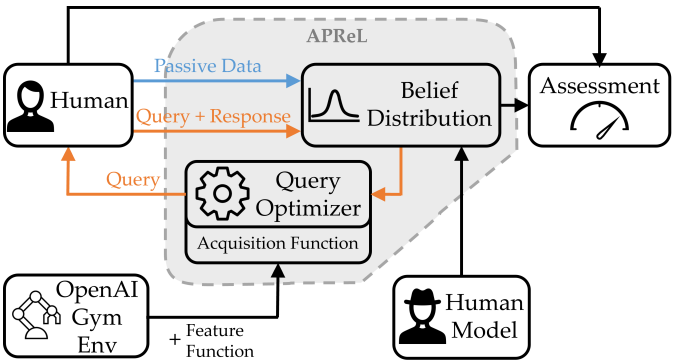

Fig. 1. APReL assumes a human model for how the human provides demonstrations and responds to the queries. Optionally, some passive data from the actual human, e.g., demonstrations, are used to initialize a belief distribution. Based on this belief, query optimizer then outputs a query that will give the most information about actual human. This query is asked to the human, and their response is used to update the belief, which completes the active learning loop (orange arrows). Finally, the quality of the learned model is assessed in comparison with the actual human.

dissimilarity between them with respect to a given distance measure between the queries.

Medoids. This method partitions the reduced query set into b clusters based on the given distance measure using k-medoids algorithm [33, 34]. Then the medoid of each cluster (similar to centroids in k-means algorithm) is taken into the batch.

Boundary Medoids. This method further improves the diversity of the medoids algorithm by applying k-medoids only on the boundary of the convex hull of the reduced set. We note that efficiently finding the boundary of the convex hull may not always be possible. Hence, this method works only when the queries can be mapped into an Euclidean space, as in [21].

Successive Elimination. The successive elimination algorithm starts with the reduced set and iteratively removes the queries until b of them are left, which are taken as the batch. The removal rule is as follows. It first finds the pair of queries with the shortest distance between them. It then removes the one with the worse acquisition function value out of this pair. Among the heuristic batch selection methods, this often gives the highest data-efficiency.

DPP. Finally, the DPP method builds a k-DPP [23] given the distance between the queries and their acquisition function values. The (approximate) mode of this k-DPP distribution, which optimally maximizes the diversity and the informativeness, is then taken as the batch.

V. APReL Structure and Usage

APReL implements multiple modules to provide the researchers with *easy experimentation*. These modules include: query types, user models, belief distributions, query optimizers and different acquisition functions. An overview of APReL's general workflow is shown in Figure 1. We next briefly go over the modules.

Basics. APReL implements *Environment* and *Trajectory* classes. An APReL environment requires an OpenAI Gym environment and a features function that maps a given sequence of state-action pairs to a vector of trajectory features as in Eq. (1). *Trajectory* instances then keep trajectories of the *Environment* along with their features.

Query Types. As discussed earlier, researchers developed and used several comparison and ranking query types. Among those, APReL readily implements preference queries, weak comparison queries, and full ranking queries. More importantly, the module for query types is customizable, allowing researchers to implement other query types and information sources. As an example, demonstrations are already included in APReL.

User Models. Preference-based reward learning techniques rely on a human response model, e.g. the softmax model, which gives the probabilities for possible responses conditioned on the query and the reward function. APReL allows to adopt any parametric human model and specify which parameters will be fixed or learned.

Belief Distributions. After receiving feedback from the human (Human in Fig. 1), Bayesian learning is performed based on an assumed human model (Human-hat in Fig. 1) by a belief distribution module. APReL implements the sampling-based posterior distribution model that has been widely employed by the researchers. However, its modular structure also allows to implement other belief distributions, e.g., Gaussian processes.

Query Optimizers. After updating the belief distribution with the user feedback, a query optimizer completes the active learning loop by optimizing an acquisition function to find the best query to the human. APReL implements the widely-used "optimize-over-a-trajectory-set" idea for this optimization, and allows the acquisition functions that we discussed earlier. Besides, the optimizer module also implements the batch optimization methods that output a batch of queries using different techniques. All of these three components (optimizer, acquisition functions, batch generator) can be extended to other techniques.

Assessing. After (or during) learning, it is often desired to assess the quality of the learned reward function or user model. The final module does this by comparing the learned model with the information from the human.

A. Example

An example runner script is included in APReL that explains how to use the library and the available options. We refer the readers to https://github.com/Stanford-ILIAD/APReL for installation and running instructions.

VI. Conclusion

In this paper, we introduced our *novel* Python library APReL, which provides a modular framework for experimenting with and developing active preference-based reward learning algorithms. In short term, we are planning to implement more techniques and features, as we discussed in Section II. By accepting contributions from the community, our goal is to keep APReL up-to-date with state-of-the-art methods, and possibly extend it to non-Bayesian learning methods.

References

[1] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning". In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, p. 1.

[2] Brian D Ziebart et al. "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[3] Mengxi Li et al. "Learning Human Objectives from Sequences of Physical Corrections". In: *arXiv preprint arXiv:2104.00078* (2021).

[4] Andrea Bajcsy et al. "Learning robot objectives from physical human interaction". In: *Conference on Robot Learning*. PMLR. 2017, pp. 217–226.

[5] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. "Third-person imitation learning". In: *International Conference on Learning Representations (ICLR)*. 2017.

[6] Dilip Arumugam et al. "Grounding natural language instructions to semantic goal representations for abstraction and generalization". In: *Autonomous Robots* 43.2 (2019), pp. 449–468.

[7] Wei Chu, Zoubin Ghahramani, and Christopher KI Williams. "Gaussian processes for ordinal regression." In: *Journal of machine learning research* 6.7 (2005).

[8] Ankit Shah, Samir Wadhwania, and Julie Shah. "Interactive robot training for non-markov tasks". In: *arXiv preprint arXiv:2003.02232* (2020).

[9] Dorsa Sadigh et al. "Active Preference-Based Learning of Reward Functions". In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2017. DOI: 10.15607/RSS.2017.XIII.053.

[10] Paul F Christiano et al. "Deep reinforcement learning from human preferences". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 4302–4310.

[11] Daniel Brown et al. "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations". In: *International conference on machine learning*. PMLR. 2019, pp. 783–792.

[12] Marco De Gemmis et al. "Preference learning in recommender systems". In: *Preference Learning* 41 (2009), pp. 41–55.

[13] Erdem Biyik et al. "Learning Reward Functions from Diverse Sources of Human Feedback: Optimally Integrating Demonstrations and Preferences". In: *The International Journal of Robotics Research (IJRR)* (2021).

[14] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[15] Sydney M Katz, Anne-Claire Le Bihan, and Mykel J Kochenderfer. "Learning an urban air mobility encounter model from expert preferences". In: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. IEEE. 2019, pp. 1–8.

[16] Malayandi Palan et al. "Learning Reward Functions by Integrating Human Demonstrations and Preferences". In: *Proceedings of Robotics: Science and Systems (RSS)*. June 2019. DOI: 10.15607/rss.2019.xv.023.

[17] Erdem Biyik et al. "The Green Choice: Learning and Influencing Human Decisions on Shared Roads". In: *Proceedings of the 58th IEEE Conference on Decision and Control (CDC)*. Dec. 2019. DOI: 10.1109/CDC40024.2019.9030169.

[18] Erdem Biyik et al. "Asking Easy Questions: A User-Friendly Approach to Active Reward Learning". In: *Proceedings of the 3rd Conference on Robot Learning (CoRL)*. Oct. 2019.

[19] Nils Wilde, Dana Kulić, and Stephen L Smith. "Active preference learning using maximum regret". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 10952–10959.

[20] Maegan Tucker et al. "Preference-based learning for exoskeleton gait optimization". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 2351–2357.

[21] Erdem Biyik and Dorsa Sadigh. "Batch Active Preference-Based Learning of Reward Functions". In: *Proceedings of the 2nd Conference on Robot Learning (CoRL)*. Vol. 87. Proceedings of Machine Learning Research. PMLR, Oct. 2018, pp. 519–528.

[22] Erdem Biyik et al. "Batch Active Learning Using Determinantal Point Processes". In: *arXiv preprint arXiv:1906.07975* (2019).

[23] Alex Kulesza and Ben Taskar. "Determinantal Point Processes for Machine Learning". In: *Foundations and Trends in Machine Learning* 5.2–3 (2012), pp. 123–286. ISSN: 1935-8237. DOI: 10.1561/2200000044. URL: http://dx.doi.org/10.1561/2200000044.

[24] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning." In: *IJCAI*. Vol. 7. 2007, pp. 2586–2591.

[25] Nils Wilde et al. "Learning Reward Functions from Scale Feedback". In: *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2021.

[26] Chandrayee Basu et al. "Active learning of reward dynamics from hierarchical queries". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 120–127.

[27] Vivek Myers et al. "Learning Multimodal Rewards from Rankings". In: *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2021.

[28] Erdem Biyik et al. "Active Preference-Based Gaussian Process Regression for Reward Learning". In: *Proceedings of Robotics: Science and Systems (RSS)*. July 2020. DOI: 10.15607/rss.2020.xvi.041.

[29] Kejun Li et al. "ROIAL: Region of Interest Active Learning for Characterizing Exoskeleton Gait Preference Landscapes". In: *International Conference on Robotics and Automation (ICRA)*. May 2021.

[30] Sydney Katz et al. "Preference-based Learning of Reward Function Features". In: *arXiv preprint arXiv:2103.02727* (Mar. 2021).

[31] Soheil Habibian, Ananth Jonnavittula, and Dylan P Losey. "Here's What I've Learned: Asking Questions that Reveal Reward Learning". In: *arXiv preprint arXiv:2107.01995* (2021).

[32] Nir Ailon. "An Active Learning Algorithm for Ranking from Pairwise Preferences with an Almost Optimal Query Complexity." In: *Journal of Machine Learning Research* 13.1 (2012).

[33] Hae-Sang Park and Chi-Hyuck Jun. "A simple and fast algorithm for K-medoids clustering". In: *Expert systems with applications* 36.2 (2009), pp. 3336–3341.

[34] Christian Bauckhage. *NumPy / SciPy Recipes for Data Science: k-Medoids Clustering*. Feb. 2015. DOI: 10.13140/2.1.4453.2009.