Balancing Familiarity and Curiosity in Data Exploration with Deep Reinforcement Learning

Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, Srividya Subramanian

▶ **To cite this version:**

Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, Srividya Subramanian. Balancing Familiarity and Curiosity in Data Exploration with Deep Reinforcement Learning. SIGMOD/PODS '21: International Conference on Management of Data Fourth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM), Jun 2021, Virtual Event China, France. pp.16-23, 10.1145/3464509.3464884 . hal-04600277

HAL Id: hal-04600277
https://hal.science/hal-04600277

Submitted on 4 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Balancing Familiarity and Curiosity in Data Exploration with Deep Reinforcement Learning
Authors' Copy

Aurélien Personnaz
Sihem Amer-Yahia
aurelien.personnaz@cnrs.fr
sihem.amer-yahia@cnrs.fr
CNRS, Univ. Grenoble Alpes
Saint Martin D'Hères, France

Laure Berti-Equille
IRD, ESPACE-DEV
Montpellier, France
laure.berti@ird.fr

Maximilian Fabricius
Srividya Subramanian
mxhf@mpe.mpg.de
sri@mpe.mpg.de
Max Planck Institute for
Extraterrestrial Physics
Garching, Germany

Abstract

The ability to find a set of records in Exploratory Data Analysis (EDA) hinges on the scattering of objects in the data set and the on users' knowledge of data and their ability to express their needs. This yields a wide range of EDA scenarios and solutions that differ in the guidance they provide to users. In this paper, we investigate the interplay between modeling curiosity and familiarity in Deep Reinforcement Learning (DRL) and expressive data exploration operators. We formalize curiosity as intrinsic reward and familiarity as extrinsic reward. We examine the behavior of several policies learned for different weights for those rewards. Our experiments on SDSS, a very large sky survey data set[1] provide several insights and justify the need for a deeper examination of combining DRL and data exploration operators that go beyond drill-downs and roll-ups.

ACM Reference Format:
Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. 2021. Balancing Familiarity and Curiosity in Data Exploration with Deep Reinforcement Learning Authors' Copy. In *Fourth Workshop in Exploiting AI Techniques for Data Management (aiDM'21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3464509.3464884

1 Introduction

Research in Exploratory Data Analysis (EDA) has been around for several decades and is seeing a renewal thanks to the increasing application of machine learning-based techniques. In particular, reinforcement learning applied to data exploration [3, 19, 26] has shown very good results in the cases where EDA logs are data- and scenario-specific and generally not exploitable as training data.

The ability to find a set of records in EDA depends on two key aspects: (1) *data and scenario complexity*: searching for individual records ranges from looking for "a needle in the hay" to "scattered deposits" of clustered records, and (2) *human ability*: some users may be very familiar with the data and/or the application domain and know how and where to search, whereas others may not be able to express their needs.

An Illustrative Example. Consider the case of exploring the SDSS SkyServer database to find a set of galaxies. Today's astrophysicists spend considerable time running series of SQL queries against the SkyServer database[2]. Many discoveries happen "accidentally". Most of the scientists' time is spent in reformulating queries. That was the case for the discovery Green Pea galaxies that recently gained attention in Astronomy as one of the potential sources that drove cosmic reionization. What astronomers need is the ability to search for subsets or supersets of objects (akin to drill-down and roll-up operations), search for objects (i.e., galaxies) with similar properties or similar property distributions, and also search for objects in the vicinity (in terms of attribute values) of a given set objects.

Let us consider a concrete scenario. Sri is an astronomer who wants to engage in finding as many Green Peas as possible, and possibly other kinds of galaxies that are similar. She needs to explore the data and refine her needs on-the-go. Figure 1 shows a sequence of three steps that form an automated exploration pipeline. The pipeline starts looking for *familiar yet different objects*: it first **finds neighboring objects, with similar colors as the Green Pea galaxies** and then it **breaks down those objects into subsets of similar spectral properties**. These two steps result in more Green Peas. At this stage, exploration becomes adventurous and **returns different objects with comparable distributions of relative ratios and strength of emission lines at higher redshifts**. As a result, Sri discovers that Green pea galaxies are analogous to high redshift galaxies. She hence decides to stop the EDA process to have a discussion with her team.

While seemingly simple, the exploration steps of Sri went from looking for needles in a haystack to exploring large sets of clustered objects. In addition to operators that drill-down and roll-up in the data, Sri also needs to look in the vicinity of some objects of interest, or to jump in the space in search for other objects that have some common or similar values with the ones she found in the course of her exploration. To capture that, we define a set of operators that

[1] https://www.sdss.org/

[2] See the query interface and logs produced here http://skyserver.sdss.org/log/en/traffic/sql.asp

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
aiDM'21, June 20–25, 2021, Virtual Event, China
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8535-0/21/06…$15.00
https://doi.org/10.1145/3464509.3464884

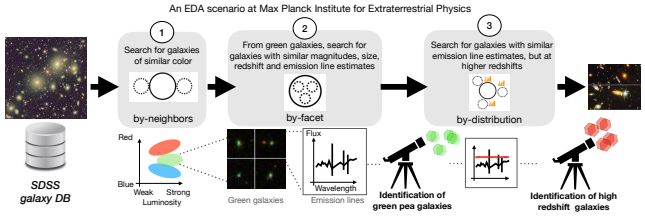


Figure 1: Exploring SDSS Data

can be composed to form an exploration pipeline. Each operator takes a set of records and returns k sets of records. We formalize the problem of learning a policy, i.e., an exploration pipeline. Similarly to previous work on generating automated EDA sessions [3, 26], we choose to use Deep Reinforcement Learning (DRL) in the absence of training data or previous exploration logs. However, we do not limit ourselves to a notion of reward that is based on finding familiar target items and we propose to *investigate a virtually infinite number of scenarios that oscillate between looking for needles in a haystack and more compact sets of clustered records.* To achieve that, we borrow the idea of combining extrinsic reward, i.e., looking for familiar objects, with intrinsic reward, i.e., venturing in the space like a curious explorer [7]. This idea was shown to be effective in learning to play a game where high curiosity, i.e., exploring all the states of a system, yields high learning performance when no or low extrinsic reward is available [20, 24]. This allows us to investigate the interplay between expressive data exploration operators and a DRL approach that combines: (1) intrinsic reward to capture curiosity and (2) extrinsic reward to capture data familiarity, in order to train exploration pipelines.

Our system, DORA THE EXPLORER, pre-trains several models with different weights for intrinsic and extrinsic rewards.[3] We use DeepMind's A3C (Asynchronous Advantage Actor Critic), a well-known DRL agent architecture. Actor-critic methods combine policy gradient methods with a learned value function using asynchronous gradient descent to optimize deep neural network controllers. A3C completes this actor-critic architecture with parallel workers learning from diverse experience and sharing their learning, which was shown to have a stabilizing effect on training [20].

We run extensive experiments and find that rewarding the model every time it ventures into new states (i.e., curiosity) outperforms traditional familiarity-only exploration [3, 19, 26]. However, while some curiosity is good, too much of it is not appropriate for data exploration. Additionally, we observe that the addition of new operators does not degrade extrinsic reward (capturing data familiarity), as the agents learn to choose the most efficient set of operators to produce the type of reward they are looking for.

In summary, our contributions are:

(1) We cast the problem of producing exploration pipelines as a DRL problem and formalize the **BCF Pipeline Generation Problem** (Balancing between Curiosity and data Familiarity) that learns a policy maximizing a combination of extrinsic and intrinsic rewards (Section 3).
(2) We build DORA THE EXPLORER, a data exploration system that leverages state-of-art A3C curiosity-based learning and expressive data exploration operators (Section 4).
(3) Our experiments on real-world data corroborate our claim that curiosity-based DRL combined with expressive data exploration operators that go beyond traditional drill-down and roll-up operations, outperforms existing approaches leveraging RL and DRL for data exploration (Section 5).

[3] DORA THE EXPLORER is available at https://bit.ly/dora-application

2 Related Work

EDA. Guiding users in EDA is a well-studied area. Our work is related to research on query learning. For instance, AIDE [9] focuses on building a query based on generating a decision tree that classifies tuples as relevant/irrelevant, a task requiring considerable user effort to annotate samples. Another work, REQUEST [12] proposes a framework for query-from-examples. It relies on a « User-Driven Pruning » technique to rule out subsets of the space that are irrelevant to the user. This approach reduces the exploration space, but it also requires important user effort.

Numerous other works proposed next-step recommendations by using logs of previous operations (e.g., [11]) or by relying on real-time feedback [9]. Similarly to existing work on automated generation of EDA sessions [3, 26], our work generates end-to-end exploration pipelines and studies the interplay between the data or EDA scenario complexity and the human ability to express needs. Most importantly, it examines the utility of combining expressive data exploration operations with curiosity-based RL.

RL for Data Exploration. ATENA [3] leverages DRL for EDA. Its reward function encourages the agent to perform a sequence of operations to maximize: (i) operator interestingness: for example, GROUP operators are ranked using the Compaction-Gain method [6] to favor actions yielding a small number of groups that cover a large number of tuples; (ii) diversity of actions computed as a distance between actions' results; and (iii) human understandability computed by a weak supervision-based classifier that employs a set of hand-crafted rules and a small set of EDA operators defined by human experts as examples. Our approach differs noticeably in offering a large degree of freedom for the exploration, without requiring any user-defined training sample or complex metrics to compare the actions' result sets. Most importantly, it enables to control "adventurous" explorations via curiosity, which is not possible with previous approaches.

Curiosity driven RL. Intrinsically motivated RL was initially defined in early 2000 [7]. It recently gained interest with successful applications to video games [24] but also other tasks like visual paragraph generation [18].

Exploration Operators. Several expressive data exploration operations have been proposed in recent work. The difference with our work is that they have not been considered in an iterative exploration session and combined with curiosity-based DRL. Our by-facet is the same as other work on faceted search [16, 31, 32]. In [21], `by-example` is used to find sets that are similar to/different from an input set. In [22], `by-example-around` returns k diverse sets that overlap with an input set, and `by-example-within` returns k subsets that maximize the coverage of an input set. Our `by-distribution` is similar to `by-analytics` in [15] as it looks for sets that are similar/different from to an input set in terms of data distributions, or that admits a set of distributions and finds sets with similar distributions [2]. There exist other operators that we did not consider. For

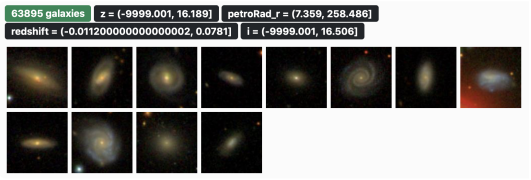


Figure 2: Sample from a set of galaxies

Operator	RCC8 Formalism [25]	Output description
by-facet(D, A)	NTPPi	returns as many subsets of D as there are combinations of values of attributes in A
by-superset(D, k)	NTPP	returns the k smallest supersets of input set D (k is application-dependent)
by-distribution(D)	DC	returns all sets that are distinct from the input set D and whose attribute value distribution is similar to D
by-neighbors(D, a)	EC	returns 2 sets that are distinct from the input set D and that have the previous (smaller) and next (larger) values for attribute a

Table 1: Exploration operators. The initial data is represented with a bold line and the destination results are represented with dashed lines.

instance, by-text [8] leverages textual information such as tags and reviews to find sets that exhibit similar/dissimilar tags or reviews.

Our design relies on an expressive set-based data model that is modular and our set of operators can be extended with new operators, assuming they are closed under a set semantics.

3 Casting EDA as Curiosity-based DRL

We consider a set of records D. In our example, each record represents an object in the sky that is described with a set of attributes. We use 7 attributes to describe galaxies: attributes `(u,g,r,i,z)` describe the magnitude in each SDSS color filter[4], `petroRad_r` describes the size of an object, and `redshift` records how far an object is from the Earth. We use D to create an instance of our set-based data model where a set is described with a conjunction of attribute values. Figure 2 shows a sample from a set of 63,895 galaxies described by a conjunction of 4 attribute values.

An exploration pipeline is a sequence of operators. In its general form, an operator takes a set of objects $D' \subseteq \mathcal{D}$ and returns sets of objects that are related to objects in D'. Table 1 summarizes our operators. Since they are applied to sets, we also give in the second column their equivalent definition in the *Region Connection Calculus 8* (RCC8) formalism [17, 25].

3.1 Modeling Pipelines as Policies

We model an exploration pipeline as a policy and propose to train a Markov Decision Process to learn a policy of an agent with the goal to maximize its expected reward such as:

$$\pi^* = argmax_\pi E[R^\pi] \quad (1)$$

[4]http://skyserver.sdss.org/dr16/en/proj/advanced/color/definition.aspx

where π is a policy and R^π is the reward obtained by π.

Our abstraction is a graph $\mathcal{G} = \langle \mathcal{S}, E \rangle$ where each node is a state $s_i \in \mathcal{S}$ in $\mathcal{S}$ and each edge $e_i \in E$ is an interaction between two consecutive states s_i and s_{i+1}. Each state s_i contains several sets of objects referred to as $sets(s_i)$. The exact number of sets associated to a state s_i depends on the operator that was applied to generate s_i (see Table 1). Our model $(\mathcal{S}, E, R)$ is a deterministic discrete MDP [1, 13] with the following properties: (1) States are the set of nodes $\mathcal{S}$ in $\mathcal{G}$; (2) E is a set of actions, where each action $e_i \in E$ is applied to one set of objects in state s_i and returns several sets in state s_{i+1}. An action induces a transition between two nodes in $\mathcal{G}$ and is represented by an edge. The description of an action is deterministic, that is, $S \times E \rightarrow S$; (3) the function $R(s_i, e_i, s_{i+1})$ is the reward obtained from transitioning from s_i to s_{i+1} with taking action e_i.

We designed the reward $R(s_i, e_i, s_{i+1})$ function such that it can simultaneously capture the notion of data familiarity as an extrinsic reward (i.e., as an objective score coming from the environment) and the notion of curiosity as an intrinsic reward determined by the agent's past experience and its knowledge of $\mathcal{G}$.

3.2 Data Familiarity and Curiosity as Rewards

To compute rewards, we assume we are given a target set of familiar objects T. Previous work on data exploration [3, 19, 26] defined familiarity as a function of the number of target objects found. Objects were "found" when they were contained in a set seen by the agent, i.e., returned by an operator. This definition is insufficient when target objects are "drowned" in very big sets, which is the case when exploring very large data sets such as galaxies.

Extrinsic reward. To manage the exploration of big sets, we revisit the notion of data familiarity and define the extrinsic reward as a function of the *concentration ratio* of target objects in a set. The data familiarity of a state s_i is then defined for a target set T:

$$Familiarity(s_i, T) = \Sigma_{O \in sets(s_i)} \frac{|O \cap T|^2}{|O| \times |T|} \quad (2)$$

The above formula is a variant of the Jaccard index. It computes the fraction of objects found in the sets that belong to a state s_i. In practice, to prevent the agent from "over-exploiting" a set of objects it is very familiar with (and going back and forth to reach the reward), the agent stores a familiarity score for each of the found target objects. Retrieving a new object increases the reward only if the agent's updated familiarity score is higher than before.

Using data familiarity for a scattered target set has proven to be a good mean to create an exploration tour of the data [26]. Although a well-defined target set will allow a large exploration of the data, data familiarity-driven exploration is intrinsically limited by the prior knowledge of the person designing the target set for training.

Intrinsic reward. In opposition to extrinsic reward, the RL community has developed the notion of intrinsic reward, also called curiosity [7, 24]. Curiosity consists of rewarding the agent when it reaches states it does not recognize. This type of reward pushes the agent to always go further in its exploration, creating policies looking for far away and unknown states.

Previous work on RL applied to games [24] used a complex intrinsic curiosity module with a double objective: (1) filtering the visual features that are independent and uninteresting to the agent

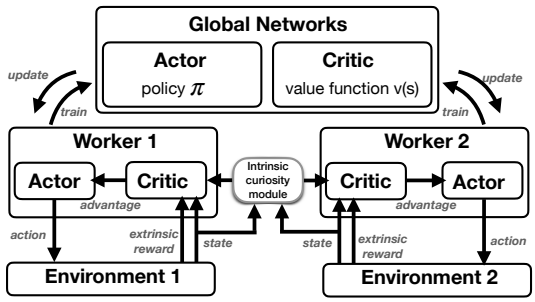


Figure 3: Architecture Details of Offline Training

and (2) recognizing the states unknown to the agent to reward it. As our work does not depend on visual items and takes place in a controlled environment, the feature selection part is not necessary. Although our problem has a very high number of possible states, this number is finite which allows us to keep a counter for each seen state. The curiosity reward for a state s_i is hence:

$$Curiosity(s_i) = \frac{1}{Counter_{s_i}} \tag{3}$$

Combining rewards. Applying an action e_i to a state s_i causes a transition to s_{i+1} and its reward is calculated with $\delta + \beta = 1$ as:

$$R(s_i, e_i, s_{i+1}) = \delta.Familiarity(s_{i+1}, T) + \beta.Curiosity(s_{i+1}) \tag{4}$$

3.3 Problem Statement

Now, we formally define the problem we address to generate EDA pipelines with **b**alancing **c**uriosity and **f**amiliarity as follows.

Problem 1. **(BCF Pipeline Generation Problem)***: Find a policy π^* that maximizes the expected cumulative reward:*

$$\pi^* = argmax_\pi \, \mathbb{E}\Big[\sum_{i=1}^{|\pi|} \gamma^i R(s_i, e_i, s_{i+1})\Big] \tag{5}$$

where γ is a discount factor in $[0, 1]$ and $|\pi|$ is the length of policy π which corresponds to the number of operators in a pipeline.

4 Our Solution

There are many methods for solving MDPs, including value iteration and policy iteration. This problem could be solved using Dynamic Programming [4] or the Monte Carlo method or a combination thereof using the Temporal Difference based approach [3, 14, 26, 27]. Policy iteration involves two steps: policy evaluation and policy improvement that are repeated until convergence. Value iteration consists in finding the optimal value function, followed by one policy extraction. Once the value function is optimal, the policy is also optimal (i.e., as it converges). While these two methods appear seemingly close, it has been proved theoretically and empirically in [23] that policy iteration is computationally more efficient and requires a smaller number of iterations to converge. So, we adapt model-free RL [3, 14, 26, 27] with inputs $(\mathcal{S}, E, R)$ as a policy iteration method which fits our proposed problem remarkably well in the absence of logs as training samples.

Algorithm 1 Pseudo-code of a worker training loop

Require: *globalCritic* and *globalActor*: Global critic and actor; *workerCritic* and *workerActor*: Worker critic and actor; *maxZ*: Maximal number of episodes; *env*: Pipeline environment; δ: Data familiarity weight; β: Curiosity weight; *maxSteps*: Number of steps per episode; *z*: Global episode counter

1: **repeat**
2: Initialize step counter $t \leftarrow 0$
3: Reset *env* and get initial state $s_t \leftarrow env.reset()$
4: **repeat**
5: $e_t \leftarrow workerActor.selectAction(s_t)$
6: $f_t, s_{t+1} \leftarrow env.exectute(e_t)$ ▷ returns extrinsic reward and next state
7: $c_t \leftarrow computeCuriosityReward(s_t, e_t, s_{t+1})$
8: $r_t \leftarrow \delta * f_t + \beta * c_t$
9: $adv \leftarrow workerCritic.computeAdvantages(s_{t+1}, r_t)$
10: $globalActor.train(s_t, e_t, adv)$
11: $globalCritic.train(s_{t+1}, r_t)$
12: $workerActor.weights \leftarrow globalActor.weights$
13: $workerCritic.weights \leftarrow globalCritic.weights$
14: $t \leftarrow t + 1$
15: **until** $t == maxSteps$
16: $z \leftarrow z + 1$
17: **until** $z == maxZ$

Ensure: A policy π maximizing reward

4.1 Deep Reinforcement Learning Algorithm

Model-free RL allows us to address the problem of finding a policy that maximizes the discounted cumulative reward. Similarly to [24], our curiosity reward model can potentially be used by a range of policy learning methods. In our implementation, we use A3C [20], a state-of-the-art Deep Reinforcement Learning framework that has been shown to outperform other critic-based methods on a wide range of applications [20].

Actor-critic methods combine policy gradient methods with a learned value function. Each learning episode contains *action probabilities and values* that get periodically updated as the agent learns from the environment based on the reward function (defined in Eq. (4)). The policy (the actor) adjusts action probabilities based on the current estimated advantage of taking that action; the value function (the critic) updates this advantage based on the rewards such as: $Advantage(s_i, e_i, s_{i+1}) \approx R(s_i, e_i, s_{i+1}) + \gamma V(s_{i+1}) - V(s_i)$ where $V(.)$ is the expected reward function. Several workers run in parallel (see Figure 3) and update the actor and critic values. A simplified version of the training loop of a worker is presented in Algorithm 1. Lines #1-17 present an episode and lines #5-16 present an operator execution step within which the action is selected (line #5) and the reward is computed (lines #6-8). This serves the advantage computation step (line #9). The value network learns a baseline state value to which the current reward estimate is compared to obtain the "advantage". The policy network adjusts the log probabilities of the actions based on the advantage via the classic RL algorithm. Line #10 trains the policy with the newly computed advantage values and line #11 trains the value function with the obtained reward. Lines #12-13 return the weights of the updated

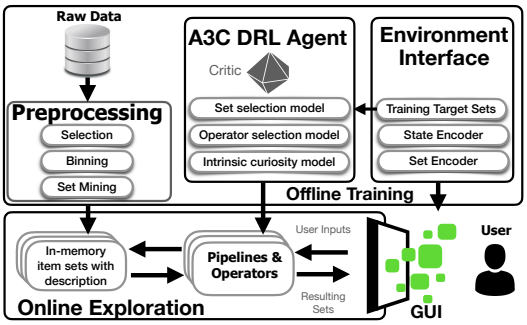


Figure 4: Overall Architecture of DORA THE EXPLORER

policy. This process is completed in parallel by each worker as presented in Figure 3.

4.2 Our System

The overall architecture of DORA THE EXPLORER is shown in Figure 4. The purpose of model training is to generate exploration pipelines. To train our models, we first preprocess data and instantiate our set-based model. Equi-depth binning is applied to numerical attributes and we use the LCM closed frequent pattern mining algorithm [28] to generate sets of objects.

The appeal of A3C comes from its parallelized and asynchronous architecture: multiple actor-learners are dispatched to separate instantiations of the environment; they all interact with the environment and collect experience and asynchronously push their gradient updates to a central target network. We use a Tensorflow-based implementation of A3C[5].

Similarly to [3], we were confronted to a relatively large action-space while combining the choice of the input set, operators, and parameters, so we split the problem in two, both parts handled by different actors. The first actor selects on which set the next operator should be executed and the second selects the operator and its parameters. The states the agent goes through are evaluated by a single critic model, producing the advantage evaluations to train both actors. The DRL agent is trained based on parameters of the environment: (1) the **Target Set**; (2) a **Set Encoder** that generates a feature vector for each set of objects. The features are the set size, the set description, the number of distinct values in the set, and the entropy of the values in the set, and (3) a **State Encoder** that concatenates the encoded vectors of every displayed set. For each training, the system chooses different weights for intrinsic and extrinsic rewards. The outcome of the training step is saved in a storage unit that contains all our pre-trained models, i.e., pipelines.

5 Experiments

The purpose of our experiments is twofold: (i) Study familiarity and curiosity rewards with different training variants (i.e., weights); (ii) Examine the interplay between our curiosity-based RL and expressive data exploration operations.

[5]https://github.com/marload/DeepRL-TensorFlow2/

5.1 Experimental Setup

Data set. SDSS[6] is a large sky survey data set containing images and metadata about hundreds of millions of astronomical objects. We selected the data from the 2.6 million galaxies having clean photometry and spectral information. Each galaxy is described with 7 attributes (commonly used in Astronomy) from two join tables named `photoobj` and `specobj`. Each column was binned into 10 equi-depth bins. We used LCM [29] with a support value of 10, and generated 348,857 sets whose size ranges from 10 to 261,793 galaxies. The data was used as an in-memory pandas dataframe[7], to have a sufficiently fast data manipulation to train the agents in a reasonable time. The pipeline operators and item set representations were implemented in python, in a library shared with DORA THE EXPLORER.

Pipeline starting point. The agents were trained and incorporated into DORA THE EXPLORER under the partial/full guidance modes. The input set of an episode is always the complete data set which requires to start with a by-facet (other operators are not meaningful on the full set).

Training setting. The agents were trained on multiple servers and desktop computers. Training took 100 hours for about 1,700 episodes with 250 steps (operator selection and execution) per episode. Each agent used 6 workers in parallel; the update interval (i.e., number of steps before a policy update) was set to 20 steps and we concatenated five successive states for the LSTM[8] layers of the networks. Training data was stored using wandb [5].

Exploration policies. We compare 5 training variants: `FAMO` for familiarity-only (this mimics exiting data exploration work), `CURO` for curiosity-only, `50FAM-50CUR` for 50% familiarity and 50% curiosity, `75FAM-25CUR` for 75% familiarity and 25% curiosity, and `25FAM-75CUR` for 25% familiarity and 75% curiosity.

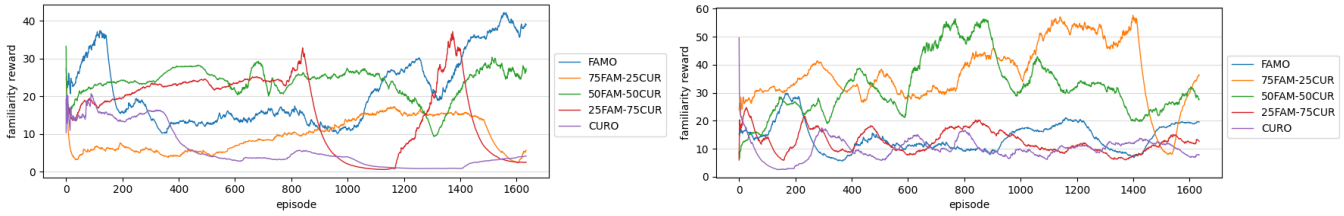
Exploration operator modes. Our models are trained with two operator modes: (1) `traditional` mode that is limited to drill-down operations with by-facet and roll-up operations with by-superset and (2) `all-operator` mode that extends the previous mode with by-neighbors and by-distribution operators (see Table 1).

Familiarity with the target set. To incite agents to visit a maximum number of galaxy types during the exploration, we designed the target set used for training to be "scattered" in the data space. The set was composed by picking 100 samples from 170 classes defined in the Galaxy Zoo classification [30] (a citizen science project with over 16 million morphological classifications of 304,122 galaxies drawn from the Sloan Digital Sky Survey), resulting in a target set containing 17,000 heterogeneous objects (0.65% of the total data).

Metrics. We study the offline training phase of our agents and the online exploration phase using the learned policies. We measure the evolution of extrinsic and intrinsic rewards and the operators chosen for different learning variants.

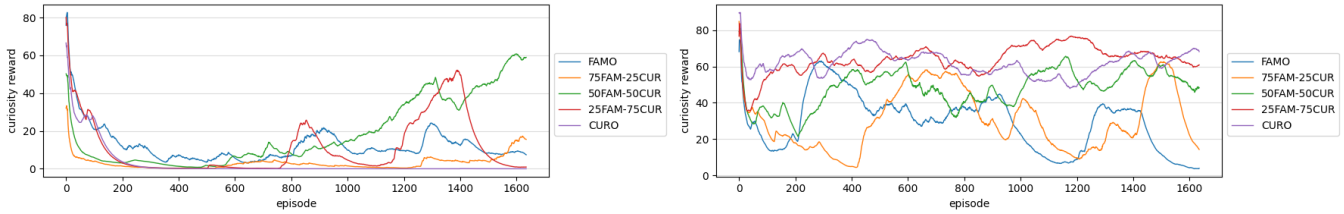
Online evaluation. We run 20 pipelines composed of 250 operations for each of the trained agents.

[6]https://www.sdss.org/
[7]https://pandas.pydata.org/
[8]https://en.wikipedia.org/wiki/Long_short-term_memory



(a) traditional **(b) all-operator**

Figure 5: Evolution of extrinsic reward (familiarity-based) during training



(a) traditional **(b) all-operator**

Figure 6: Evolution of intrinsic reward (curiosity-based) during training

5.2 Offline Training Phase

Figures 5 and 6 show the evolution of extrinsic and intrinsic rewards respectively with `traditional` mode for Figures 5(a) and 6(a) and `all-operator` for Figures 5(b) and 6(b).

5.2.1 Familiarity and Curiosity-driven Policies. Our first observation is that both FAMO and CURO policies produce a mix of extrinsic and intrinsic rewards. FAMO produces some intrinsic reward at the beginning of the training (Figures 6), as every state it goes through is unknown. This reward quickly decreases as FAMO focuses on refining its data familiarity strategy to tour data. On the other hand, as CURO explores the data, it finds target objects fortuitously, generating a moderate amount of extrinsic reward.

Secondly, although FAMO gets the best results for familiarity with `traditional` operators, in every other case, both CURO and FAMO under-perform when compared to other learning variants. For both reward types, and both operator modes, the highest rewards are reached by agents with mixed intrinsic and extrinsic rewards. It is particularly noticeable with `traditional` operators, where CURO rapidly runs out of reward and lacks motivation to develop a working policy (Figure 6(a)), while `50FAM-50CUR` and `75FAM-25CUR` end up with relatively successful curiosity-driven strategies. Similarly, for familiarity with `all-operator` mode, we observe that `75FAM-25CUR` and `50FAM-50CUR` largely outperform FAMO.

While looking at the details of the intrinsic and extrinsic rewards for `75FAM-25CUR` with `all-operator`, we notice that it begins by favoring data familiarity, focusing on the first target objects it found. To do so, it keeps returning to the same states; this results in a steadily decreasing intrinsic reward until episode #450. Then, it loses interest and starts exploring new states, resulting in a high intrinsic reward and a slight decrease in extrinsic reward. It switches back to extrinsic reward around episode #900, with a steady increase until a last temporary switch to intrinsic reward around episode #1500 (Figures 5(b) and 6(b)).

In summary, when both reward sources are available, the agents tend to alternate between curiosity- and familiarity-based policies that prioritize one over the other and as the amount of total reward evolves during training, priorities shift. This illustrates the importance of optimizing for familiarity and curiosity in tandem.

5.2.2 Impact of Operators. The first remarkable difference between the two operator modes is the apparent difficulty faced by the agents using only `traditional` operators to produce an intrinsic reward.

Unlike in `all-operator` mode, where the agents instantly manage to produce a constant amount of curiosity-based intrinsic reward (until some of them shift their focus to familiarity-based reward), it takes them a relatively long time in `traditional` mode to learn a strategy that produces curiosity-based reward. With `traditional` operators, the policies can only explore via set generalization and specialization, while with by-neighbors and by-distribution operators, they can explore all sets at the same level, yielding reachable new states more easily. This is substantiated by Figure 8 where the agent favors by-neighbors and by-distribution during the intrinsic reward ones (episodes #500 to 900, #1300 to 1350 and #1400 to 1500) and by-facet and by-superset operators during the extrinsic reward production phases (the rest of the time).

It is quite the opposite for familiarity-driven policies that does not benefit from more expressive operators. Indeed, we can see that FAMO reaches much higher performances with `traditional` operators than with `all-operator`, where it is outperformed by agents with a mixed reward.

These observations suggest that the `traditional` mode is adapted to familiarity-driven policies, which could justify why the database community has mainly focused on FAMO and `traditional` operators. But they do also show that `traditional` mode is not fit for curiosity-driven EDA. That is illustrated by CURO which never manages to learn due to a long absence of reward (which can be interpreted as boredom).

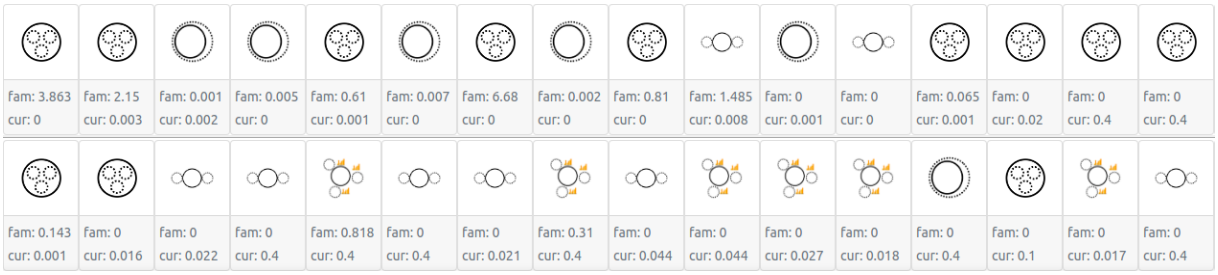


Figure 7: Fragments of pipelines generated by `75FAM-25CUR` (top) and `25FAM-75CUR` (bottom) with `all-operator` mode

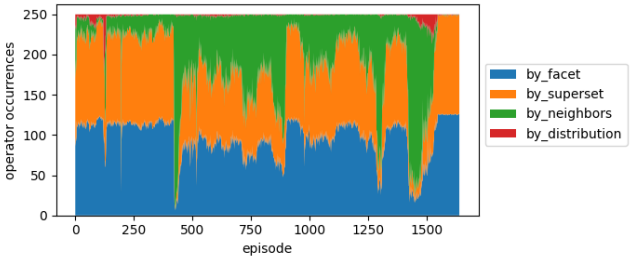


Figure 8: Operator distribution evolution during training for `75FAM-25CUR` with `all-operator` mode

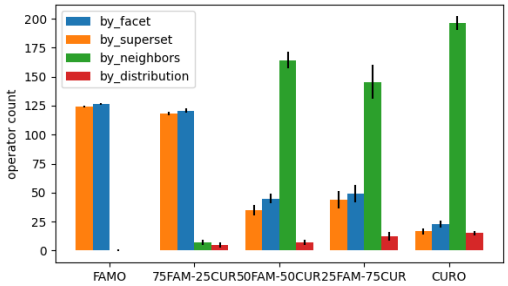


Figure 9: Operator distribution in online pipelines.

We also observe that adding new operators benefits data familiarity-driven strategies for agents with a mixed reward, as they learn to choose the most efficient operators to produce the reward they seek.

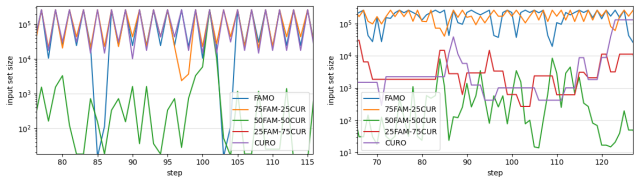
5.2.3 Training Summary. We can conclude that, unlike in games [24], where exploring all possible states inevitably helps moving towards the output of a maze, a full curiosity-based intrinsic reward is not adapted and sufficient for EDA. We see that `CURO` produces a very limited amount of data familiarity, which means that it does not even get close to the objects of interest that we are familiar with. On the other hand, we saw that the best familiarity-driven results were attained by agents with some level of curiosity reward and that the production of curiosity-based intrinsic reward is easier in the `all-operator` mode. This dictates the use of a moderate level of curiosity with additional operators, instead of familiarity only with the usual drill-down and roll-up operations.

5.3 Online Exploration Phase

We now turn to examining the execution of our pipelines in the online exploration phase.

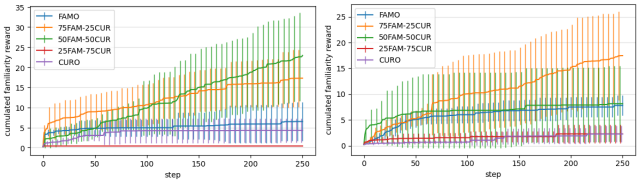
5.3.1 Operator Selection. We observe in Figure 9 that by-facet and by-superset operators are predominantly selected in familiarity-driven policies such as `FAMO` and `75FAM-25CUR`, whereas by-neighbors and by-distribution operators are preferred in pipelines generated by curiosity-driven policies such as `CURO` and `25FAM-75CUR`. This can also be seen in the two pipeline fragments below extracted from the two variants (Figure 7). This confirms that, for different weights, the agents will adopt the operators that best support their strategy. This further motivates studying the interplay between data exploration operators and curiosity-driven DRL of EDA.

Figure 10(a) (resp. 10(b)) shows the order in which operators are executed in a pipeline using `traditional` (resp. `all-operator`) mode. Both figures show that our policies make full use of the operators at their disposal. Figure 10(a) shows that policies oscillate between by-facet and by-superset operators, the only two available operators in `traditional` mode. The flat lines in Figure 10(b) illustrate the exploration of sets at the same level using by-neighbors and by-distribution operators.



(a) traditional **(b) all-operator**

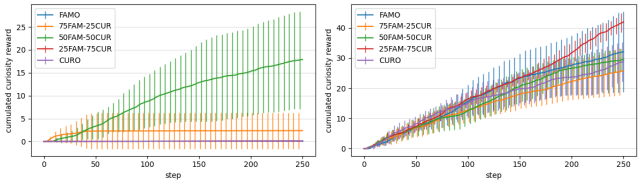
Figure 10: Evolution of input set sizes in their order of execution in online pipelines (log scale)



(a) traditional **(b) all-operator**

Figure 11: Evolution of the cumulative extrinsic reward during online pipelines

5.3.2 Reward Evolution. Figures 11 and 12 show the evolution of extrinsic and intrinsic rewards respectively. The results are largely



(a) traditional **(b) all-operator**

Figure 12: Evolution of the cumulative intrinsic reward during online pipelines

compatible with the offline phase. We can see in Figure 11 that in both operator modes, mixed reward agents clearly outperform `FAMO` and that `CURO` is the worst performer on cumulated familiarity. Figure 12 corroborates that curiosity-based reward is widely produced by every variant with `all-operator`, while only `50FAM-50CUR` manages to produce intrinsic reward with `traditional`.

6 Conclusion and Discussion

We presented a formalism and an empirical study of the interplay between expressive data exploration operators and curiosity-based RL. Our framework learns pipelines in the form of policies with a combination of curiosity and data familiarity as reward. Our findings set the stage for exploiting AI methods for data exploration.

An immediate future investigation is to examine the relation between curiosity/familiarity and the scattering of target objects in the data. Another immediate investigation is to control the overlap between the sets used in the exploration and examine its effect on the utility of each operator. For instance, if the sets constitute a partition of the data, the use of by-facet and by-superset is not necessary and other operators such as by-neighbors are needed.

In this work, we learned an end-to-end policy with no user interventions. In the medium/longer term, we would like to investigate the role of user feedback. For instance, users may only be allowed to modify the parameters of an operator such as the input set and attribute of by-facet or replace an operator altogether. Interestingly, accounting for feedback will necessitate a mix of policy learning and operator recommendation. For policy learning, we would like to learn the weights of intrinsic and extrinsic rewards based on feedback and re-adjust the policy as users are exploring data. For operator recommendation, feedback will help break ties between operators with very similar reward values and favor those for which users have provided positive feedback.

The new research directions outlined above will only be possible if both real-world and synthetic data are available. It has become a wide practice to work with semi-synthetic data and we would like to contribute to that trend toward a benchmark. We have started with an application that we made available at https://bit.ly/dora-application, and our code and data at https://github.com/apersonnaz/rl-guided-galaxy-exploration. By focusing on the effectiveness of applying learning techniques for data exploration, our benchmark will complement existing efforts in our community that mainly focus on performance [10].

References

[1] J. Achiam, D. Held, A. Tamar, and P. Abbeel. 2017. Constrained policy optimization. In *International Conference on Machine Learning*. PMLR, 22–31.
[2] S. Amer-Yahia, S. Kleisarchaki, N. K. Kolloju, L. V. Lakshmanan, and R. H. Zamar. 2017. Exploring Rated Datasets with Rating Maps. In *WWW*.
[3] O. Bar El, T. Milo, and A. Somech. 2020. Automatically generating data exploration sessions using deep reinforcement learning. In *SIGMOD*. 1527–1537.
[4] R. Bellman. 1966. Dynamic programming. *Science* 153, 3731 (1966), 34–37.
[5] L. Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/ Software available from wandb.com.
[6] V. Chandola and V. Kumar. 2007. Summarization - compressing data into an informative representation. *Knowl. Inf. Syst.* 12, 3 (2007), 355–378.
[7] N. Chentanez, A. Barto, and S. Singh. 2005. Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, L. Saul, Y. Weiss, and L. Bottou (Eds.), Vol. 17. MIT Press. https://proceedings.neurips.cc/paper/2004/file/4be5a36cbaca8ab9d2066debfe4e65c1-Paper.pdf
[8] M. Das, S. Thirumuruganathan, S. Amer-Yahia, G. Das, and C. Yu. 2012. Who Tags What? An Analysis Framework. *VLDB Endow.* 5, 11 (2012), 1567–1578.
[9] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. 2016. AIDE: an active learning-based approach for interactive data exploration. *IEEE TKDE* 28, 11 (2016).
[10] P. Eichmann, E. Zgraggen, C. Binnig, and T. Kraska. 2020. IDEBench: A Benchmark for Interactive Data Exploration. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 1555–1569.
[11] M. Eirinaki, S. Abraham, N. Polyzotis, and N. Shaikh. 2014. Querie: Collaborative database exploration. *IEEE TKDE (TKDE)* 26, 7 (2014), 1778–1790.
[12] X. Ge, Y. Xue, Z. Luo, M. A. Sharaf, and P. K. Chrysanthis. 2016. REQUEST: A scalable framework for interactive construction of exploratory queries. In *IEEE Intl. Conf. on Big Data*. 646–655.
[13] P. Geibel. 2006. Reinforcement learning for MDPs with constraints. In *European Conference on Machine Learning*. Springer, 646–653.
[14] L. P. Kaelbling, M. L. Littman, and A. W. Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.
[15] M. Kahng, S. B. Navathe, J. T. Stasko, and D. H. P. Chau. 2016. Interactive Browsing and Navigation in Relational Databases. *pVLDB Endow.* 9, 12 (2016), 1017–1028. http://www.vldb.org/pvldb/vol9/p1017-kahng.pdf
[16] N. Kamat, P. Jayachandran, K. Tunga, and A. Nandi. 2014. Distributed and interactive cube exploration. In *ICDE*. 472–483.
[17] S. Li and M. Ying. 2003. Region connection calculus: Its models and composition table. *Artificial Intelligence* 145, 1-2 (2003), 121–146.
[18] Y. Luo, Z. Huang, Z. Zhang, Z. Wang, J. Li, and Y. Yang. 2019. Curiosity-driven Reinforcement Learning for Diverse Visual Paragraph Generation. arXiv:1908.00169 [cs.CV]
[19] P. Marcel, N. Labroche, and P. Vassiliadis. 2019. Towards a benefit-based optimizer for Interactive Data Analysis. In *EDBT/ICDT*.
[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*. 1928–1937.
[21] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. 2017. New Trends on Exploratory Methods for Data Analytics. *pVLDB Endow.* 10, 12 (2017), 1977–1980.
[22] B. Omidvar-Tehrani, S. Amer-Yahia, and A. Termier. 2015. Interactive user group analysis. In *CIKM*. 403–412.
[23] E. Pashenkova, I. Rish, and R. Dechter. 1996. Value iteration and policy iteration algorithms for Markov decision problem. In *AAAI'96: Workshop on Structural Issues in Planning and Temporal Reasoning*. Citeseer.
[24] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. 2017. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2778–2787.
[25] D. A. Randell, Z. Cui, and A. G. Cohn. 1992. A Spatial Logic Based on Regions and Connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. 165–176.
[26] M. Seleznova, B. Omidvar-Tehrani, S. Amer-Yahia, and E. Simon. 2020. Guided Exploration of User Groups. *pVLDB Endow.* 13, 9 (2020), 1469–1482.
[27] R. S. Sutton and A. G. Barto. 2018. *Reinforcement learning*. MIT press.
[28] T. Uno, M. Kiyomi, and H. Arimura. 2004. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI)*, Vol. 126.
[29] T. Uno, M. Kiyomi, and H. Arimura. 2004. LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK*.
[30] K. W. Willett, C. J. Lintott, S. P. Bamford, K. L. Masters, B. D. Simmons, K. R. V. Casteels, E. M. Edmondson, L. F. Fortson, S. Kaviraj, W. C. Keel, and et al. 2013. Galaxy Zoo 2: detailed morphological classifications for 304 122 galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society* 435, 4 (Sep 2013), 2835–2860. https://doi.org/10.1093/mnras/stt1458
[31] N. Yan, C. Li, S. B. Roy, R. Ramegowda, and G. Das. 2010. Facetedpedia: enabling query-dependent faceted search for Wikipedia. In *CIKM*. 1927–1928.
[32] L. Zhang and Y. Zhang. 2010. Interactive retrieval based on faceted feedback. In *SIGIR*. 363–370.