Information Sciences 485 (2019) 42–54

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Curious Feature Selection

Michal Moran, Goren Gordon*

Curiosity Lab, Department of Industrial Engineering, Tel Aviv University, Israel

ARTICLE INFO

Article history:
Received 24 December 2017
Revised 2 February 2019
Accepted 4 February 2019
Available online 5 February 2019

Keywords:
Intrinsic motivation learning
Curiosity loop
Reinforcement learning
Big data
Data science
Feature selection

ABSTRACT

In state-of-the-art big-data applications, the process of building machine learning models can be very challenging due to continuous changes in data structures and the need for human interaction to tune the variables and models over time. Hence, expedited learning in rapidly changing environments is required. In this work, we address this challenge by implementing concepts from the field of intrinsically motivated computational learning, also known as artificial curiosity (AC). In AC, an autonomous agent acts to optimize its learning about itself and its environment by receiving internal rewards based on prediction errors. We present a novel method of intrinsically motivated learning, based on the curiosity loop, to learn the data structures in large and varied datasets. An autonomous agent learns to select a subset of relevant features in the data, i.e., feature selection, to be used later for model construction. The agent optimizes its learning about the data structure over time without requiring external supervision. We show that our method, called the Curious Feature Selection (CFS) algorithm, positively impacts the accuracy of learning models on three public datasets.

© 2019 Elsevier Inc. All rights reserved.

1. Introduction

In the data science field, researchers apply scientific methods, processes, and systems to extract knowledge or insights from data in various forms, both structured or unstructured [10]. It is desirable to reduce high data dimensionality for many machine learning tasks due to the "curse of dimensionality" [24]. Feature selection methods [18] provide a way to build simpler and more comprehensive models and to improve their learning performances. However, in many scenarios, data scientists are faced with a significant amount of data that must be processed in real time (or near-real time) to gain insights [44]. In the worst cases, the size of the data is unknown and the data distribution changes over time; thus, it is not practical to either wait until all the data is available before performing feature selection or to run the feature selection process manually and repeatedly over time [28].

A similar challenge is encountered by exploring autonomous agents, namely, they are required to learn changing and unknown environments. One class of such agents is known as intrinsically motivated agents: agents that engage in exploration for its own sake rather than as a step toward solving a specific problem [1,35]. Computational approaches to intrinsic motivation, also known as artificial curiosity, are typically based on architectures in which an agent learns to anticipate the consequences of its actions and actively chooses its actions based on internal measures related to the novelty or predictability of the anticipated situation [35]. One example of computationally intrinsic motivation architecture is autonomous Reinforcement Active Learning (ReAL). In conventional reinforcement learning schemes, the reward function is set exter-

* Corresponding author.
E-mail addresses: michal.moran2806@gmail.com (M. Moran), goren@gorengordon.com (G. Gordon).

https://doi.org/10.1016/j.ins.2019.02.009
0020-0255/© 2019 Elsevier Inc. All rights reserved.

nally: the reward depends on the current state and action, and the function does not change over time. In contrast, in an autonomous reinforcement active learning setup, the goal is to optimize on-line supervised learning [14,41].

The curiosity loop algorithm, which is based on ReAL, sets the reward as the prediction error of another learner. All the parameters in the model are autonomously learned during development: they do not rely on external teachers or pre-designed behaviors, but rather on the curiosity loop architecture [14–16]. This type of architecture, which can be applied to any autonomously learning task, is also highly relevant for data science processes.

In this contribution, we apply the concepts of intrinsically motivated autonomous agents to data structures. We implement the curiosity loop architecture for the task of learning the data structure and performing Curious Feature Selection (CFS). First, the training data is divided into small episodes. In each episode, a new feature (action) is selected in an inner-loop based on the already selected previous feature (state), and the internal reward consists of a reduction in the learning model error [35]. The model runs in a loop on multiple episodes until it reaches convergence. The loop attempts to find a feature-selection policy that optimizes the model accuracy. The entire process is completely autonomous in the sense that all actions and rewards are determined autonomously. The resulting policy is used to construct a new learning model on the entire training set and on an unseen testing set. We show that this architecture improves model accuracy compared to running the learning model on the entire dataset or using common feature selection algorithms.

This paper is organized as follows. First, we present related work regarding feature selection, reinforcement learning and the basic curiosity loop in Section 2. Section 3 describes the suggested Curious Feature Selection algorithm. The three public datasets used for experiment are described in Section 4, and Section 5 presents the experimental results. Finally, Section 6 provides a discussion, conclusions, and future work.

2. Related work

2.1. Feature selection

In the machine learning subfield, feature selection is the process of selecting a subset of relevant variables to be used in model construction. Feature selection techniques are primarily intended to improve model prediction performances and runtimes, to reduce model overfitting, and to increase generalization [18]. Several feature selection algorithms exist, most of which use search techniques along with an evaluation index to find an appropriate feature subset. The most basic technique searches all possible subsets and selects the subset that minimizes the model error. However, this exhaustive search is not computationally effective, particularly in situations where a large number of features exist [6].

Feature selection algorithms can be divided into three main categories: Wrappers, filters [26] and embedded [3] methods.

Wrapper methods search the variable space for possible feature subsets and evaluate each by running a model. The model is tested on a testing set to determine the model error rate, which is translated into a score for that subset. Finally, the subset with the best score is selected. Search algorithms such as Sequential methods [21,25,36], the Branch and Bound method [26,33] or the Genetic Algorithm [13] can be used to find the best subset. These methods are computationally intensive and have a risk of overfitting, but usually provide the best-performing feature set for that particular type of model [3].

Sequential methods are the most commonly used wrappers methods for performing feature selection. They begin with a single feature subset and iteratively add or remove features until some termination criterion is met. Sequential Forward Selection (SFS) start with the empty set and add feature while Sequential Backward Selection start with the full set and delete features. These methods do not examine all possible subsets and therefore do not guaranteed to produce the optimal result [21,25,36].

Filter methods select variables without considering the model and are based on general measures such as the Pearson correlation coefficient [2], mutual information [26], chi-square test scores [20], and other measures. Filters provide a feature subset that is not tuned to a specific type of predictive model, but they are usually less computationally intensive than wrapper methods.

The Chi square test is a statistical test of independence which determine the dependence of two variables. For feature selection tasks, the Chi square test is used to calculate the statistics between each feature and the target variable to determine the existence of a relationship between them. If the target variable does not depend on specific feature, this feature can be excluded. If they are dependent, the feature should be included in the feature subset [20].

Embedded methods are embedded in a specific learning algorithm that performs feature selection and classification simultaneously. These methods combine the advantages of both previous methods and tend to fall somewhere between them in terms of computational complexity [3,18].

Decision trees are iteratively constructed by splitting the data according to the value of a specific feature. The splitting feature is selected by its importance for the classification task. Different algorithms use different metrics to select “the best” feature. These algorithms typically measure the homogeneity of the target variable within the subset. In many cases, only a subset of features is included in a decision tree. Therefore, the selection of features is implicitly embedded into the algorithm and the decision trees can be understood as an embedded method [27].

Feature selection has been shown to be effective in numerous applications. However, over time, both the data sample size and the feature numbers tend to grow continuously, and the characteristics of big data applications, such as data velocity and variety, have brought new challenges to the existing feature selection methods [28]. In response, new feature selec-

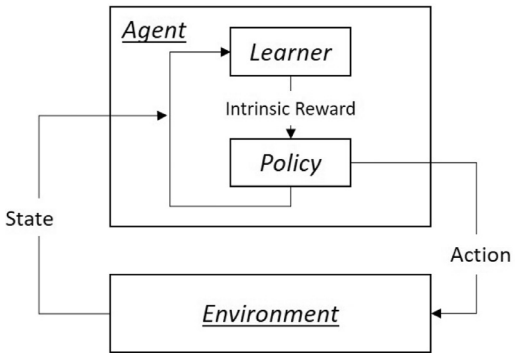

Fig. 1. Basic curiosity loop.

tion approaches such as structured sparsity-inducing feature selection (SSFS) methods [17], online feature selection (OFS) [48], optimization framework for multi-label feature selection with streaming labels [32] and maximum relevance minimum common redundancy method [7] have been proposed. While most of the feature selection methods are restricted to batch learning setting, many of these new methods are online feature selection methods. They are dealing with large-scale high-dimensional data [17,48] and with challenging scenarios such as streaming features, where not all features are available for each training instance [48] or streaming labels [32] in which the number of labels is unknown in advance. These methods are assuming linear classifier for the learning model.

The proposed Curious Feature Selection (CFS) method which is based on the curiosity loop and reinforcement learning architectures, can be applied for both batch and online learning, it is not restricted to a specific predictor or specific feature type and there is no need to know the number of features in advanced. In addition, it has other advantages such as better accuracy, shorter running time and inherent responsiveness to changes in data structure over time.

2.2. Reinforcement learning

Reinforcement learning (RL) deals with learning a sequential decision making problem [43]. RL is a general class of algorithms in the field of machine learning that aims at allowing an agent to learn how to behave in an environment by taking actions and obtaining feedback that consists of a scalar reward signal [34]. In many applications, the RL environment is formulated as a Markov decision process (MDP) [37]. In this model, an environment is modeled as a set of states and a set of actions that can be performed to control those states. The goal is to control the system in such a way that maximizes the future accumulated reward signal over the long run. In model-free RL, no prior knowledge about the MDP exists, and the agent must interact, or experiment with the environment to gain knowledge about how to optimize its behavior. During this process, the agent is guided by the reward feedback [5]. The agent interacts with its environment in discrete time steps. At each step, the agent chooses an action from the set of available actions. Then, the environment changes to a new state and the agent acquires the reward associated with that transition. The goal of the agent is to maximize its reward. At the end of this process, the agent has learned a policy—a computable function that outputs an action for each state. Most learning algorithms for MDPs compute the optimal policies by learning a value function that represents an estimate of how advantageous it is for the agent to be in a certain state or to perform a certain action in a given state [34].

One of the most basic and popular methods to estimate value is the Q-learning algorithm [45]. The basic idea in Q-learning is to incrementally estimate the Q-values for actions, based on rewards and the learned Q-value function. The update rule uses the reward for an action and a max-operator over the Q-values of the next state to update Q_t into $Q_{(t+1)}$:

$$Q_{k+1}(s_t, a_t) = Q_k(s_t, a_t) + \alpha[r_t + \gamma \max_a Q_k(s_{t+1}, a) - Q_k(s_t, a_t)] \quad (1)$$

The agent takes a step in the environment from state s_t to s_{t+1} using action a_t while receiving reward r_t. The update occurs on the Q-value of action a_t in the state s_t from which action a_t was executed, and the Q-value is the expected discounted reward for executing this action [46].

2.3. Intrinsically motivated learning and the curiosity loop

The basic intrinsically motivated learning [35], also known as artificial curiosity [41] or the curiosity loop [14–16], is based on the reinforcement learning paradigm, which is composed of a learner (predictor) and an intrinsic reward. In each loop, the agent selects an action and obtains an internal reward, which is based on prediction errors. The agent's goal is to select an optimal action that maximizes its learning and to learn an action selection policy that maps states to actions. Each loop's convergent dynamics leads to a specific behavior tightly related to the objective learnable correlation [14–16]. A visual depiction of a curiosity loop is shown in Fig. 1.

Algorithm: Curious Feature Selection (CFS)

input: set of possible features: F, $|F| = N_{features}$, dataset: $\vec{x_i}$ with N_{data} samples
output: set of selected features F_{sel}

1: **initialize:**
$Q \leftarrow \mathbb{1}_{n\times n}$,$n = |F| + 1$
$N_{epi} = 100$
$itr = 10 \times N_{data}/N_{epi}$
$\gamma = 0.01$
$learner = DT$
2: **for** i = 1 to itr **do**
3: $x_{epi} \leftarrow x_{j,k}$, $j = sample(\vec{x_i}, n = N_{epi})$, $k \in F$
4: $x^{epi\_train}, x^{epi\_validation} = split(x_{epi}, validation\_size = 0.2)$
5: $e_0 \leftarrow 0.5$
6: $s_t \leftarrow s_0$
7: $F_{sel} \leftarrow \{\}$
8: $F_{available} \leftarrow F$
9: $\epsilon = \begin{cases} 0.09 & i < itr/4 \\ 0.05 & itr/4 \le i < itr/2 \\ 0.01 & itr/2 \le i < itr * 3/4 \\ 0.005 & i \ge itr * 3/4 \end{cases}$
10: $\alpha = \begin{cases} 0.9 & i < itr/4 \\ 0.5 & itr/4 \le i < itr/2 \\ 0.3 & itr/2 \le i < itr * 3/4 \\ 0.1 & i \ge itr * 3/4 \end{cases}$
11: **while** $|F_{available}| > 0$ **do**
12: **if** $p \sim U(0,1) < epislon$ **then**
13: $a_t = random(a \in F_{available})$
14: **else**
15: **if** $max(Q(s_t, a_t)) < threshold$ **then** end episode
16: **else** $a_t = max(Q(s_t, a_t))$
17: $F_{sel} \leftarrow F_{sel} \cup a_t$
18: $x^{epi\_train_{s_t}} = x^{epi\_train}_{k\in F_{sel}}$
19: $x^{epi\_validation_{s_t}} = x^{epi\_validation}_{k\in F_{sel}}$
20: $O_{learner} = learner(x^{epi\_train_{s_t}})$
21: $e_t = \sum_i \left(O_{learner}\left(x_i^{epi\_validation_{s_t}}\right) - O_{real}\left(x_i^{epi\_validation_{s_t}}\right)\right)$
22: $r_t \leftarrow (e_0 - e_t)$
23: $Q(s_t, a) = Q(s_t, a) + \alpha\left[r + \gamma * \max_a Q(s_{t+1}, a) - Q(s_t, a)\right]$
24: $s_{t+1} \leftarrow a_t$
25: $e_0 \leftarrow e_t$
26: $F_{available} \leftarrow F_{available} \setminus a_t$
27: $t \leftarrow t + 1$

Fig. 2. CFS algorithm.

Because the goal of the basic curiosity loop is to autonomously and actively learn a correlation in the best way possible, the core of each loop is the learner, which attempts to map presented input-output pairs via an internally supervised learning algorithm. For each presented example, the learner acquires a prediction error (i.e., the difference between the expected output and the correct output) and the reward value is set to this error [14–16,35]. Thus, the agent in the curiosity loop actively learns via the reinforcement of the prediction errors. The agent determines which new example is presented to the learner by selecting an appropriate action. Then, the learner produces the prediction error, which determines the intrinsic reward. The reward is translated into a value function used to update the agent policy, which is then used in the next loop to determine the next appropriate action to select [14–16].

3. Curious Feature Selection (CFS)

The proposed Curious Feature Selection (CFS) algorithm (Fig. 2- number in parenthesis () indicates line number in the algorithm) is based on the curiosity loop architecture. The agent's task is to learn a policy that selects a subset of features from the dataset. The set includes all the features that should be used for constructing a machine learning model for this dataset. Our CFS method is based on the RL Q-learning algorithm. In this section, we describe the architecture of the CFS method and compare its complexity to that of a brute-force exhaustive search.

3.1. Episode

The dataset is divided into small chunks of constant size (i.e., episodes (3)). A new curiosity loop is executed in each episode (2). The learned policy (23) is shared between episodes and converges over time and over episodes. The motivation for episodic learning is to reduce the variance of the policy updates and achieve a more stable convergence. This approach also imitates a stream of data; thus, not all data instances and features are available at any given time.

3.2. States and actions

The basic reinforcement learning model consists of a set of environment and agent states $s \in S$ and a set of actions $a \in A$ that the agent can select. In the CFS algorithm, both the state space and the action space are equal to the data feature space (1), that is, each state represents the previously selected feature (24) and each action is the next feature to select (13,16). The state space contains one additional initial state s_0 that represents the state before any feature is selected (6).

$$s \in S : \{s_0, feature_1, feature_2, \cdots \ feature_n\} \tag{2}$$

$$a \in A : \{feature_1, feature_2, \cdots \ feature_n\} \tag{3}$$

Each state has a set of available actions $A_{available} \in A$ (7,8) that represents the features not yet selected in the current loop.

3.3. Learner and internal reward

The learner can be any supervised machine learning model. The model attempts to learn an input (i)-output (o) transformation: $L(O|I: \chi)$ where χ represents the parameters of the machine learning model. The learner finds the parameters χ that best minimize the generalization error:

$$L = \arg\max_{\chi} \left(\sum_{i} (O_{model}(i) - O_{real}(i))^2 \right) \tag{4}$$

At each time t, the learner is executed on the episode data, using the selected feature (action) and all the other features that were selected in the current episode (17,18,19). After each execution, the model output O_{model} (20) is compared to the real output O_{real}, resulting in a prediction error e_t. The prediction error e_t is compared to the previous error e_{t-1} (21), and the reward r_t is defined as the change in the error (22): larger changes in the error result in larger rewards.

$$r_t = e_{t-1} - e_t \tag{5}$$

To calculate the reward for the first selected feature, we define an initial error, which is a prior error assumption (5). The change in the error represents the importance of the last selected feature and the information it adds to the model. Thus, when the agent selects a feature that adds valuable information to the model and, thus, reduces the total error, the reward is high.

3.4. State-action value function and policy

The received internal reward is used to update a state-action value function that yields the expected utility of adding a specific feature (action) to a given feature (state). The Q-learning method described in Section 2.2, is used to update the action-value function (23).

The policy is the rule to follow when adding a specific feature (action) to a given feature (state). Starting with the initial state, the optimal policy can be constructed by simply selecting the feature (action) with the highest value above some threshold, setting this feature as the next state and repeating this operation until no more actions are available.

$$s_{t+1} = \arg\max_{a}(Q(s_t, a_t)) \tag{6}$$

3.5. Exploration vs. exploitation

To balance between leveraging the model's knowledge in each state by selecting the action with the highest estimate action-value function (exploitation) and exploring the uncharted features space, the epsilon-greedy algorithm is used, and an epsilon parameter ϵ is defined [23]. The action with the highest estimate action-value function is selected with a probability of $1 - \epsilon$ (15,16), while a random action (from $A_{available}$) is selected with a probability of ϵ (12,13).

To improve exploration during the first few episodes, the parameters ϵ and α (learning rate) are set to high values and then automatically reduced based on how many iterations have been completed [8] (9,10). In addition, an optimistic Q-learning [11] approach is used, where the Q values are initialized to 1 (1). This creates a greedy policy with respect to the Q-values for the first few episodes. The Q-value is updated with the actual value of the reward the first time each state-action is selected (23).

Table 1
Selectable algorithm parameters (number in parenthesis () indicates line number in the algorithm).

Parameter		Description	Value
Episode size	N_{epi} (1)	Number of records in each episode	100
Iterations	*itr* (1)	Total number of episodes	10 · data size ÷ episode size
Learner	*learner* (1)	The supervised machine learning model used to define the internal reward	Decision tree, Naive Bayes
Initial error	e_0 (5)	The error associated with the initial state (i.e., prior error)	0.5
Discount factor	γ (1)	The discount factor determines the importance of future rewards in the Q-Learning algorithm. A factor of 0 makes the agent "myopic" so that it considers only the current reward	0–0.01
Epsilon	ϵ (9)	The probability of selecting a random action (from the available actions) in exploration mode	0.9 to 0.1 using a step decay policy
Learning Rate	α (10)	Update step	0.09 to 0.01 using a step decay policy

3.6. Algorithm parameters

Most of the algorithm parameters are learned internally without external supervision; however a few selectable parameters are determined in advance. See Table 1 for the selectable algorithm parameters.

3.7. Complexity analysis

The proposed CFS algorithm can be classified as a wrapper method for feature selection [26]. Since most wrapper methods are computationally intensive [3], we perform a complexity analysis. We compare the computational complexity of the CFS algorithm with 2 methods: A brute-force exhaustive search which provide the optimal solution and Sequential Forward Selection (SFS) method which is a very commonly used wrapper method.

A brute-force exhaustive search finds the optimal solution by comparing all possible feature subsets. Each subset is used to build a model for the dataset, and the subset that minimizes the model error is selected. Hence, by estimating the number of required arithmetic operations, the complexity of this method is at most

$$O\left(N_{data} \cdot 2^{N_{features}}\right) \tag{7}$$

Sequential Forward Selection (SFS) method start with the empty set and sequentially add the feature that minimize the model error rate when combined with the features that have already been selected. In each step, all features that haven't been selected yet are been evaluated. By estimating the number of required arithmetic operations, the complexity of this method is at most

$$O\left(N_{data} \cdot {N_{features}}^2\right) \tag{8}$$

In comparison, the CFS algorithm runs using iterations (episodes) that are set in proportion to the data size. Each iteration uses only a small portion of the data, and in the worst case, uses all the features inside a specific episode. Hence, using similar logic, the complexity of this method is at most

$$O(N_{data} \cdot N_{features}) \tag{9}$$

For datasets with fewer than 10 features, there is no advantage to using the CSF algorithm: a simple brute-force exhaustive search will provide an optimal solution with less complexity. However, for datasets with more than 10 features the CFS algorithm has a clear computational advantage.

4. Datasets

We analyze the performance of the CFS algorithm in classifying 3 datasets: The Adult and Diabetes 130-US hospitals for the years 1999–2008 (Diabetes) datasets are from the UCI Machine Learning Repository [30], and the Medical Appointment No-shows dataset was sourced from Kaggle [19].

4.1. Dataset selection

The datasets were chosen by the following characteristics conditions:

(1) Number of instances is greater than 10,000
(2) Number of features is greater than 10
(3) Features types: Categorical & Numerical
(4) Balanced Data: the representation of each of the different classification label in data is over 30%.

Table 2
Adults dataset features list.

Feature	Description and values
age	Continuous number
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th–8th, 12th, Masters, 1st–4th, 10th, Doctorate, 5th–6th, Preschool
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, Male
capital-gain	Continuous number
capital-loss	Continuous number
hours-per-week	Continuous number
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US, India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad-Tobago, Peru, Hong, Holland-Netherlands
income	above 50 K, below 50 K

The first condition ensures that the data is large enough for multiple iterations running. The second condition is determined by the complexity analysis which indicates that the algorithm's advantage is for datasets with more than 10 features. The third condition ensures that the experiments represent business application which mostly include variables of different types and the last condition supports the algorithm's episodes approach since the episodes' size is relatively small, and the episode data should include a representation of each of the labels.

4.2. Data cleaning and transformation

First, we applied data cleaning and transformation methods to the datasets. The purpose of data cleaning is to identify incomplete, incorrect, inaccurate or irrelevant data and replace or delete them to improve data quality [39]. The purpose of data transformation is to convert data from one format or structure into another format or structure required for the analysis [38]. The data cleaning steps included removing missing values by removing records with missing values in one or more columns and dropping redundant features, such as features containing both id and description columns, and irrelevant features such as indexes. The data transformation included converting all feature values into integers by encoding the labels with value between 0 and n_{labels} -1.

4.3. Adult dataset

The Adult dataset stems from data collected during the U.S. Census and consists of sampled data of the adult population in the United States from 1994. After data preprocessing, the dataset contained 50,000 instances with 12 demographic features that described the respondents, including age, gender, country of origin, marital status, education, and a variable indicating whether the individual's income exceeded 50 K dollars per year or not. Table 2 provides a full list of the features and their descriptions.

4.4. Diabetes dataset

The Diabetes data was extracted from the Health Facts database (Cerner Corporation, Kansas City, MO), a national data warehouse that collects comprehensive clinical records from hospitals throughout the United States. It was submitted to the UCI Machine Learning Repository on behalf of the Center for Clinical and Translational Research, Virginia Commonwealth University [42]. After data preprocessing, the dataset contained 100,000 instances with 44 features describing encounters with diabetic patients, including their demographics, diagnoses, diabetic medications and number of visits in the year preceding the encounter. Table 3 provides a complete list of the features and their descriptions.

4.5. Medical appointment no-shows

The Medical Appointment No-shows data was donated to Kaggle by Joni Hoppen. This experiment attempts to predict whether or not a patient will attend an appointment using examples of patient attendance for scheduled appointments. After data preprocessing, the dataset contained 300,000 instances with 12 features describing each patient, including personal data, medical data, and data about the appointment. Table 4 provides a full list of the features and their descriptions.

Table 3
Diabetes dataset features list.

Features	Description and values
Race	Values: Caucasian, Asian, African American, Hispanic, and other
Gender	Values: male, female, and unknown/invalid
Age	Grouped in 10-year intervals: 0, 10), 10, 20),, 90, 100)
Admission type	Integer identifier corresponding to 9 distinct values, for example, emergency
Discharge disposition	Integer identifier corresponding to 29 distinct values, for example, discharged to home.
Admission source	Integer identifier corresponding to 21 distinct values, for example, physician referral.
Time in hospital	Integer number of days between admission and discharge
Number of lab procedures	Number of lab tests performed during the encounter
Number of procedures	Number of procedures (other than lab tests) performed during the encounter
Number of medications	Number of distinct generic names administered during the encounter
Number of outpatient visits	Number of outpatient visits of the patient in the year preceding the encounter
Number of emergency visits	Number of emergency visits of the patient in the year preceding the encounter
Number of inpatient visits	Number of inpatient visits of the patient in the year preceding the encounter
Diagnosis 1	The primary diagnosis (coded as first three digits of ICD9); 848 distinct values
Diagnosis 2	Secondary diagnosis (coded as first three digits of ICD9); 923 distinct values
Diagnosis 3	Additional secondary diagnosis (coded as first three digits of ICD9); 954 distinct values
Number of diagnoses	Number of diagnoses entered to the system
Glucose serum test result	Indicates the range of the result or if the test was not taken
A1c test result	Indicates the range of the result or if the test was not taken
Change of medications	Indicates if there a change in diabetic medications occurred
Diabetes medications	Indicates if any diabetic medication was prescribed yes and no
23 features for medications	Indicates whether the drug was prescribed or there was a change in dosage
Readmitted	Days to inpatient readmission. Values: <30, >30 and No for no record of readmission

Table 4
Medical appointment no-shows dataset features list.

features	Description and values
Age	Continues number
Gender	M=Male, F-Female
DayOfTheWeek	appointment's day of the week
Diabetes	Indicates whether the patient has diabetes, yes and no
Alcoholism	Indicates whether the patient is alcoholic, yes and no
Hypertension	Indicates whether the patient suffers from hypertension, yes and no
Handicap	Indicates whether the patient has a handicap, yes and no
Smokes	Indicate whether the patient smokes, yes and no
Scholarship	Indicates whether the patient receives funds from the Bolsa FamÃlia program, yes and no
Tuberculosis	Indicates whether the patient suffers from tuberculosis, yes and no
Sms_Reminder	Indicates whether the patient received an SMS reminder before the appointment, yes and no
Wait_Time	Number of days from appointment registration to appointment date
Show_Up	Indicates whether the patient kept the appointment, yes and no

5. Results

5.1. Experimental setting

We executed the CFS algorithm on the datasets described in Section 4. As a preliminary step, each of the datasets is split into a training dataset (90%) and testing dataset (10%). The training dataset is used to train the CFS algorithm and select the policy for the feature subset. The testing dataset is used to rebuild the model based on the subset of selected features and to evaluate the model on new data. On each dataset, the CFS algorithm was tested with 2 learners (Decision tree and Naive Bayes). Specifically, these learners were selected since they are applicable for both categorical and numerical features types. Except for the discount factor, all the parameters in Section 3.6 were the same in all the experiments. The discount factor was set to 0.01 for the Adult and No-Shows datasets and to 0 for the Diabetes dataset. Each experiment was repeated 30 times and the average result was taken.

5.2. Comparison algorithms

We compared the CFS algorithm results on the testing dataset with 4 traditional, popular and commonly used feature selection algorithms. The algorithms represent each one of the feature selection categories: including the Sequential Forward Selection (SFS) wrapper method [25], 2 filter methods: the Variance Threshold method, which removes all features whose variance does not meet some threshold and the Chi-Square test [20], and one embedded method based on the GINI importance score for decision tree [4] that measures the average gain of purity by splitting a given variable.

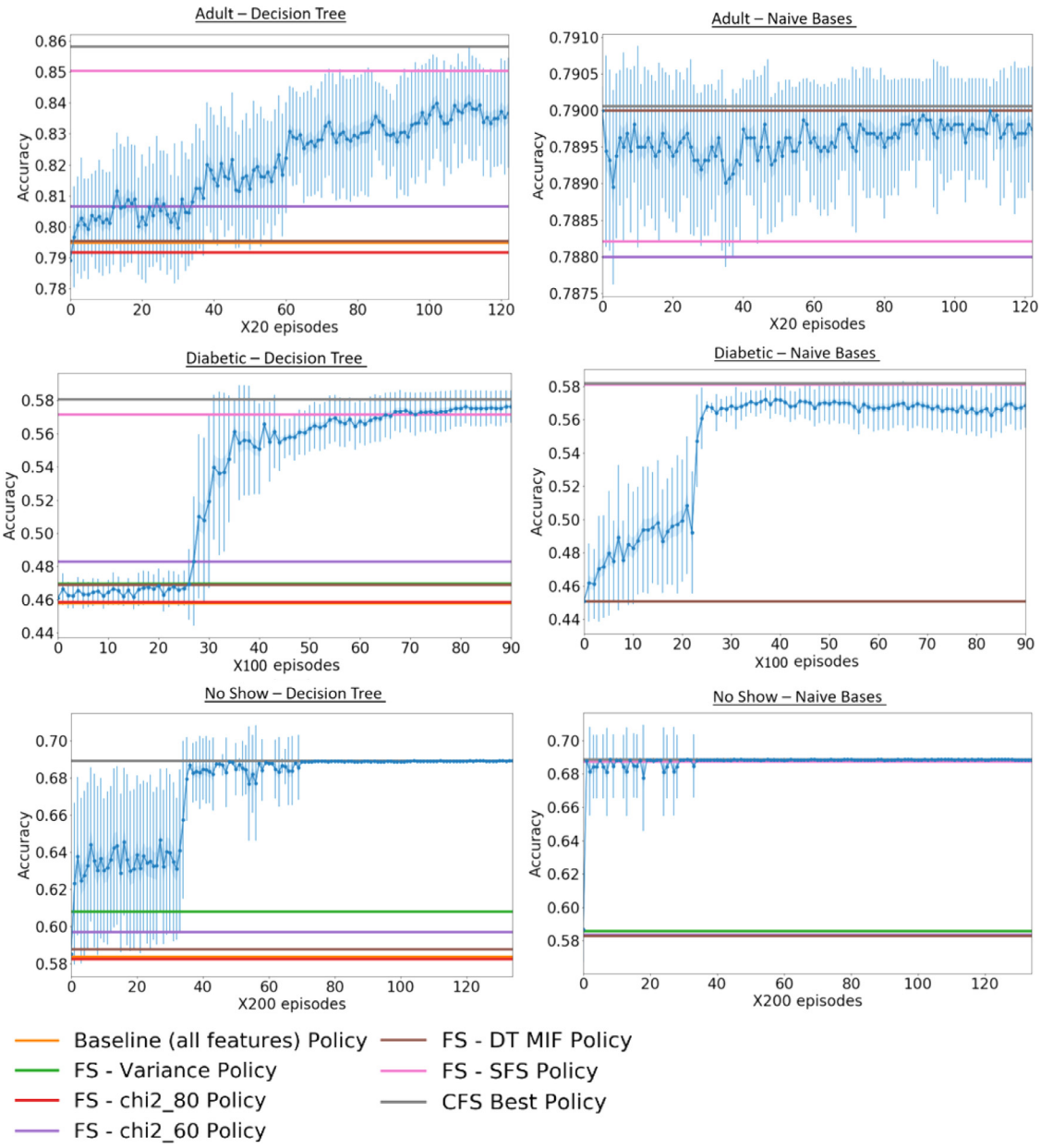

Fig. 3. Test dataset average results.

5.3. Result comparison

Fig. 3 depicts the convergence process with respect to the number of episodes (iterations) for each experiment. Each episode is described by the mean accuracy and standard deviation of the 30 experimental runs.

Tables 5–7 list the experimental results of the algorithms in terms of the selected features policy. Table 8 lists the experimental results of the CFS and the feature selection comparison algorithms in terms of the classification accuracy on the testing dataset. Table 8 shows the following:

(1) In all the experiments, the CSF algorithm's best policy achieved the highest classification accuracy, i.e., 85.5% for Adult: Decision tree, 79% for Adult: Naive Bayes, 58.07% for Diabetes: Decision tree, 78.17% for Diabetes: Naive Bayes, 68.93% for No-shows: Decision tree and 68.86% for No-shows: Naive Bayes. In No-shows: Decision tree, Sequential Forward Selection achieved the same accuracy (68.93%).

Table 5
Selected features policy—Adult dataset.

	Decision tree	Naive Bayes
Best policy	2, 3, 6, 7, 8, 9, 11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Most converged policy	2, 3, 5, 6, 7, 8, 9, 11	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Sequential Forward Selection	2, 3, 4, 8, 9	0, 1, 2, 5, 8, 9
Variance Threshold Method	0, 2, 3, 4, 5, 7, 8, 9, 10	
Chi-square test (highest scoring 80% features)	0, 2, 3, 4, 5, 7, 8, 9, 10	
Chi-square test (highest scoring 60% features)	0, 3, 5, 7, 8, 9, 10	
DT GINI Importance	0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 11	

Features: 0: age, 1: workclass, 2: education, 3: marital.status, 4: occupation, 5: relationship, 6: race, 7: sex, 8: capital.gain, 9: capital.loss, 10: hours.per.week, 11: native.country

Table 6
Selected features policy—Diabetic dataset.

	Decision tree	Naive Bayes
Best & most converged policy	12	25, 12, 38
Sequential Forward Selection	3–4, 12, 24, 27, 31–34, 45, 38–41	0, 11–12, 24, 27, 32–34, 45, 36, 38–39, 41
Variance Threshold Method	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,36,42	
Chi-square test (highest scoring 80% features)	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,24,25,26,27,28,29,32,33,36,38,40,41,42,43	
Chi-square test (highest scoring 60% features)	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,19,24,25,27,33,38,40,42,43	
DT GINI Importance	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,18,19,25,26,36,42	

Features: 0: race, 1: gender, 2: age, 3: admission_type_id, 4: discharge_disposition_id, 5: admission_source_id, 6: time_in_hospital, 7: num_lab_procedures, 8: num_procedures, 9: num_medications, 10: number_outpatient, 11: number_emergency, 12: number_inpatient, 13: diag_1, 14: diag_2, 15: diag_3, 16: number_diagnoses, 17: max_glu_serum, 18: A1Cresult, 19–41: features for medications, 42: change, 43: diabetesMed

Table 7
Selected features policy—No-shows dataset.

	Decision tree	Naive Bayes
Best & most converged policy	3, 4, 6, 7, 9	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
Sequential Forward Selection	1, 2, 7	0, 1, 2, 3, 6, 7, 10
Variance Threshold Method	0, 2, 10, 11	
Chi-square test (highest scoring 80% features)	0, 1, 2, 3, 4, 5, 7, 8, 11	
Chi-square test (highest scoring 60% features)	0, 2, 3, 4, 5, 8, 11	
DT GINI Importance	0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11	

Features: 0: Age, 1: Gender, 2: DayOfTheWeek, 3: Diabetes, 4: Alcoholism, 5: Hypertension, 6: Handicap, 7: Smokes, 8: Scholarship, 9: Tuberculosis, 10: SMS_Reminder, 11: Wait_Time

Table 8
Test dataset accuracy.

	Adults		Diabetes		No-shows	
	Decision tree	Naive Bayes	Decision tree	Naive Bayes	Decision tree	Naive Bayes
Policy Average	83.62%	78.97%	57.52%	56.68%	68.91%	68.85%
Best policy (# of experiments)	**85.5%**(2)	**79%** (26)	**58.07%** (22)	**58.17%** (1)	**68.93%** (5)	**68.86%** (14)
Most frequent policy (# of experiments)	85.1% (9)	**79%** (26)	**58.07%** (22)	N/A	**68.93%** (5)	**68.86%** (14)
Baseline (all columns)	78.9%	**79%**	46.08%	45.06%	58.53%	58.67%
Sequential Forward Selection	85.08%	78.82%	57.17%	58.12%	**68.93%**	68.75%
Variance Threshold Method	79.6%	**79%**	46.95%	45.06%	60.81%	58.59%
Chi-square test (highest scoring 80% features)	79.2%	**79%**	45.86%	45.06%	58.28%	58.36%
Chi-square test (highest scoring 60% features)	80.7%	78.8%	48.3%	45.06%	59.73%	58.36%
DT GINI Importance	79.6%	**79%**	46.9%	45.06%	58.79%	58.31%

(2) In all the experiments except for Adult: Naive Bayes and No-shows: Decision tree, the CSF algorithm achieved the highest average classification accuracy, i.e., 83.62% for Adult: Decision tree, 78.97% for Adult: Naive Bayes, 57.52% for Diabetes: Decision tree, 56.68% for Diabetes: Naive Bayes and 68.85% for No-shows: Naive Bayes.
(3) In most cases, The converged policy is the best feature-set policy in most of the experimental runs (i.e., the variance in convergent policies is extremely small).

On the Adult dataset, the traditional feature selection methods reduced the number of features from 12 features to 5–6 features (Sequential Forward Selection), 7 features (Chi-square test - 60%), 9 features (Variance and Chi-square test - 80%) and 11 features (DT GINI importance), and in most of the cases improved the model accuracy compared to the baseline accuracy when using all 12 features. For the decision tree model, the baseline accuracy is 78.9%. The best traditional

method is the Sequential Forward Selection which achieved an accuracy of 85.08%. The other traditional methods improve the baseline accuracy in 0.3%–1.8%. The CSF algorithm best policy included 7 features and achieved an accuracy of 85.5%, 6.6% higher than the baseline and 0.42% higher than the Sequential Forward Selection. For the Naive Bayes model, feature reduction does not improve model accuracy, and in some cases, it reduces the accuracy. The CSF algorithm's best and most frequent policy (26 out of 30) includes all features for Naive Bayes.

On the Diabetic dataset, the traditional feature selection methods reduce the number of features from 44 to 13–14 features (Sequential Forward Selection),20 features (Variance), 23 features (DT GINI importance), 26 features (Chi-square test - 60%) and 35 features (Chi-square test - 80%), and in some cases improved the model accuracy compared to the baseline. However, in other cases it did not change the accuracy or reduce the accuracy. For the decision tree model, the baseline accuracy was 46.08%. The best traditional method (Sequential Forward Selection) achieved an accuracy of 57.17% and improved the baseline accuracy by 11.09%. The CSF algorithm's best and most frequent policy (22 out of 30) included only one feature and achieved an accuracy 11.27% higher than the baseline. For the Naive Bayes model, Sequential Forward Selection methods achieved accuracy of 58.12% however, the other traditional feature selection methods didn't change the model accuracy (45.06%). The CSF algorithm's best and most frequent policy (5 out of 30) which included only 3 features, achieved an accuracy of 58.17% and improved on the baseline method by 13.11%.

On the No-shows dataset, the traditional feature selection methods reduced the number of features from 12 to 3 features and 7 features (Sequential Forward Selection), 4 features (Variance), 7 features (Chi-square test - 60%), 8 features (Chi-square test - 80%) and 11 features (DT GINI importance). Similar to the diabetic dataset, beside the Sequential Forward Selection method, the other methods do not consistently improve the model accuracy compared to the baseline. For the decision tree model, the baseline accuracy was 58.53%. The Sequential Forward Selection method achieved an accuracy of 68.93%, 10.5% higher than the baseline. The CSF algorithm's best policy included 5 features and achieved the same accuracy. For the Naive Bayes model, the Sequential Forward Selection method achieved an accuracy of 68.75%, 10.08% higher than the baseline. The CSF algorithm's best and most frequent policy (14 out of 30) achieved slightly better results, increasing the accuracy from the baseline's 58.67% to 68.86% using 10 features.

6. Discussion and future work

In this paper, we have presented a new Curious Feature Selection (CFS) algorithm based on the curiosity loop architecture [14]. Unlike other feature selection methods, which are restricted to either batch or online learning setting, the proposed method can be applied for both batch and online learning, it is not restricted to a specific predictor, and has other advantages such as running time and responsiveness to data changes. In this feature selection mechanism, a curious agent, using an internal reward system [35] autonomously learns the set of actions (features) that must be selected to maximize the accuracy of a target learning model. We evaluated the algorithm on 3 datasets and showed that the results obtained by the features chosen by the CFS policy were better than the results of standard feature selection methods and achieved higher accuracy.

In the current experiments, running the algorithm several times on the same dataset did not produce the same convergent feature set (i.e., policy). This situation appears to occur for two main reasons:

(1) The exploration parameter ϵ is set to a high value during the first episodes to foster a wider search in the feature space. This improves the results of the model but leads to high randomization and, in certain cases, to different policy results.
(2) The algorithm is based on a Markov Decision Process [37] with no memory. The current state is defined only by the last selected feature and not by all the features selected so far. This leads to situations in which the decision to select a specific feature based on a specific feature state results in a different reward, because the reward depends on the location from which the features are selected during the episode. In other words, although we used a Markov algorithm, the scenario is highly non-Markov; thus, any convergence proof is not applicable.

On one hand, the disadvantage of a policy that does not converge uniformly is that the algorithm must be run several times to obtain the best policy. On the other hand, its advantage is that each policy gives different insights about how specific features-sets explain the data. By getting several policies, the researcher can achieve more insights and select the policy that best fits the specific problem.

An additional consideration when using the algorithm and selecting a learner is that the CFS is most suitable for cases in which the model accuracy can be improved by choosing fewer features. It is less relevant, for example, when using ensemble models such as Random Forest [29] or Gradient Boosting [12] which work well using the entire feature set.

The main value of the CFS algorithm is that it can be implemented for big data applications, in which the data streams are large, heterogeneous and often unlabeled, and the task involves exploring complex and evolving relationships among the data in real time [47]. The algorithm's advantages for big data applications are as follows:

(1) The algorithm does not depend on feature type; it can be applied to various types of features (numeric, categorical, binary, etc.), assuming that the selected learner is applicable for the available data types.
(2) Similarly, the algorithm can be used for unsupervised learning tasks by simply replacing the learner with an unsupervised learner such as clustering [22] and adjusting the reward function to an unsupervised evaluation measurement [40].

(3) Because the algorithm learns in small episodes, similar to the mini-batch approach [9], it can be applied as an on-line algorithm on a data stream. The algorithm learns the data structure over time and updates the policy accordingly.
(4) Finally, one important property of big data applications is the ability to implement them in a distributed manner to support scalability. Scaling up an algorithm based on a reinforcement learning architecture is not simple and often not practical. Recently, however, novel algorithms that extend reinforcement learning to distributed computing have been introduced [31].

The Sequential Forward Selection (SFS) algorithm which is greedy and simple, presents similar accuracy results comparing to the proposed Curious Feature Selection algorithm. Nevertheless, we believe that the new proposed algorithm has 3 main advantages comparing to the SFS:

(1) **Overfitting** - The SFS algorithm achieves better results on the training dataset while the CFS achieves better results on the Test dataset. This can indicate that the SFS is overfitted to train data and can explain less well unseen data
(2) **Online learning** - In some big data scenarios, the data stream contains a new data structure, and there is a need to perform a new feature selection which will capture the change but with respect to the data history. Unlike the SFS algorithm which need to be run on the entire history data together with the new data, the CFS algorithms which has already captured the entire history structure inside the policy, can be run only on the new data to update the policy.
(3) **Scale** - The complexity analysis indicated that as increasing the number of features and /or the size of the data, the CFS will become more efficient compare to the SFS in terms of run time.

In future work we intend to (i) apply neural network architecture to the policy to better represent a non-Markovian state by remembering all the selected features so far; (ii) modify the algorithm to support unsupervised learning tasks; (iii) use the curiosity loop paradigm to address additional big data tasks such as active learning and data sampling.

In summary, we introduced and analyzed the Curious Feature Selection algorithm, which is based on the curiosity loop paradigm. We showed that CFS improves the classification accuracy of various models compared to standard feature selection algorithms. Finally, we discussed the strengths and weaknesses of the model and its relevance to big data domains.

References

[1] A. Barto, S. Singh, N. Chentanez, Intrinsically motivated learning of hierarchical collections of skills, 2004. http://www-anw.cs.umass.edu/pubs/2004/barto_sc_ICDL04.pdf.
[2] R. Battiti, Using mutual information for selecting features in supervised neural net learning, IEEE Trans. Neural Netw. 5 (4) (1994) 537–550, doi:10.1109/72.298224.
[3] A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning, Artif. Intell. 97 (1) (1997) 245–271, doi:10.1016/S0004-3702(97)00063-5.
[4] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32. http://www.springerlink.com/index/U0P06167N6173512.pdf.
[5] L. Busoniu, R. Babuska, B.D. Schutter, D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, 1st, CRC Press, Inc., Boca Raton, FL, USA, 2010.
[6] G. Chandrashekar, F. Sahin, A survey on feature selection methods, Comput. Electr. Eng. 40 (1) (2014) 16–28, doi:10.1016/j.compeleceng.2013.11.024.
[7] J. Che, Y. Yang, L. Li, X. Bai, S. Zhang, C. Deng, Maximum relevance minimum common redundancy feature selection for nonlinear data, Inf. Sci. 409 (2017) 68–86, doi:10.1016/j.ins.2017.05.013.
[8] C. Darken, J. Chang, J. Moody, Learning rate schedules for faster stochastic gradient search, in: Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, 1992, pp. 3–12, doi:10.1109/NNSP.1992.253713.
[9] O. Dekel, R. Gilad-Bachrach, O. Shamir, L. Xiao, Optimal distributed online prediction using mini-batches, (2010), [cs, math] http://arxiv.org/abs/1012.1367.
[10] V. Dhar, Data science and prediction, Commun. ACM 56 (12) (2013) 64–73, doi:10.1145/2500499.
[11] E. Even-Dar, Y. Mansour, Convergence of optimistic and incremental Q-learning, in: Advances in Neural Information Processing Systems, 2002, pp. 1499–1506. http://papers.nips.cc/paper/1944-convergence-of-optimistic-and-incremental-q-learning.pdf.
[12] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. (2001) 1189–1232.
[13] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1st, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[14] G. Gordon, E. Ahissar, Hierarchical curiosity loops and active sensing, Neural Netw. 32 (2012) 119–129, doi:10.1016/j.neunet.2012.02.024.
[15] G. Gordon, E. Fonio, E. Ahissar, Emergent exploration via novelty management, J. Neurosci. 34 (38) (2014) 12646–12661, doi:10.1523/JNEUROSCI.1872-14.2014.
[16] G. Gordon, E. Fonio, E. Ahissar, Learning and control of exploration primitives, J. Comput. Neurosci. 37 (2) (2014) 259–280, doi:10.1007/s10827-014-0500-1.
[17] J. Gui, Z. Sun, S. Ji, D. Tao, T. Tan, Feature selection based on structured sparsity: a comprehensive study, IEEE Trans. Neural Netw. Learn. Syst. 28 (7) (2017) 1490–1507, doi:10.1109/TNNLS.2016.2551724.
[18] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, J. Mach. Learn. Res. 3 (2003) 1157–1182. http://dl.acm.org/citation.cfm?id=944919.944968.
[19] J. Hoppen, Medical appointment no shows, 2017, https://www.kaggle.com/joniarroba/noshowappointments.
[20] I.F. Imam, R.S. Michalski, L. Kerschberg, Discovering attribute dependence in databases by integrating symbolic learning and statistical analysis techniques, Proceeding of the AAAI-93 Workshop on Knowledge Discovery in Databases, Washington DC, 1993. http://www.mli.gmu.edu/papers/91-95/93-19.pdf.
[21] A. Jain, D. Zongker, Feature selection: evaluation, application, and small sample performance, IEEE Trans. Pattern Anal. Mach. Intell. 19 (2) (1997) 153–158, doi:10.1109/34.574797.
[22] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. (CSUR) 31 (3) (1999) 264–323.
[23] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, J. Artif. Int. Res. 4 (1) (1996) 237–285. http://dl.acm.org/citation.cfm?id=1622737.1622748.
[24] E. Keogh, A. Mueen, Curse of dimensionality, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 257–258. http://link.springer.com/10.1007/978-0-387-30164-8_192.
[25] J. Kittler, Feature set search algorithms, Pattern Recognit. Signal Process. (1978).
[26] R. Kohavi, G.H. John, Wrappers for feature subset selection, Artif. Intell. 97 (1) (1997) 273–324, doi:10.1016/S0004-3702(97)00043-X.

[27] T.N. Lal, O. Chapelle, J. Weston, A. Elisseeff, Embedded methods, in: Feature Extraction, Springer, 2006, pp. 137–165.
[28] J. Li, H. Liu, Challenges of feature selection for big data analytics, IEEE Intell. Syst. 32 (2) (2017) 9–15, doi:10.1109/MIS.2017.38.
[29] A. Liaw, M. Wiener, Classification and regression by randomForest, R News 2 (3) (2002) 18–22.
[30] M. Lichman, UCI Machine Learning Repository, University of California, School of Information and Computer Science., Irvine, CA, 2013. http://archive.ics.uci.edu/ml.
[31] L.-J. Lin, Scaling up reinforcement learning, in: Machine Learning Proceedings 1993: Proceedings of the Tenth International Conference on Machine Learning, University of Massachusetts, Amherst, June 27–29, 1993, Morgan Kaufmann, 2014, p. 182.
[32] Y. Lin, Q. Hu, J. Zhang, X. Wu, Multi-label feature selection with streaming labels, Inf. Sci. 372 (2016) 256–275, doi:10.1016/j.ins.2016.08.039.
[33] P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection, IEEE Trans. Comput. C-26 (9) (1977) 917–922, doi:10.1109/TC.1977.1674939.
[34] M.v. Otterlo, M. Wiering, Reinforcement learning and Markov decision processes, in: M. Wiering, M.v. Otterlo (Eds.), Reinforcement Learning, Adaptation, Learning, and Optimization, Springer Berlin Heidelberg, 2012, pp. 3–42. http://link.springer.com/chapter/10.1007/978-3-642-27645-3_1.
[35] P.Y. Oudeyer, F. Kaplan, V.V. Hafner, Intrinsic motivation systems for autonomous mental development, IEEE Trans. Evol. Comput. 11 (2) (2007) 265–286, doi:10.1109/TEVC.2006.890271.
[36] P. Pudil, J. Novoviov, J. Kittler, Floating search methods in feature selection, Pattern Recognit. Lett. 15 (11) (1994) 1119–1125, doi:10.1016/0167-8655(94)90127-9.
[37] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2014.
[38] D. Pyle, Data Preparation for Data Mining, 1, Morgan Kaufmann, 1999.
[39] E. Rahm, H.H. Do, Data cleaning: problems and current approaches, IEEE Data Eng. Bull. 23 (4) (2000) 3–13.
[40] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.
[41] J. Schmidhuber, Formal theory of creativity, fun, and intrinsic motivation (1990 #x2013;2010), IEEE Trans. Auton. Mental Dev. 2 (3) (2010) 230–247, doi:10.1109/TAMD.2010.2056368.
[42] B. Strack, J.P. DeShazo, C. Gennings, J.L. Olmo, S. Ventura, K.J. Cios, J.N. Clore, Impact of HbA1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records, 2014. https://www.hindawi.com/journals/bmri/2014/781670/.
[43] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 1, MIT Press Cambridge, 1998. http://www.cell.com/trends/cognitive-sciences/pdf/S1364-6613(99)01331-5.pdf.
[44] H. Wang, Z. Xu, H. Fujita, S. Liu, Towards felicitous decision making: an overview on challenges and trends of big data, Inf. Sci. 367 (2016) 747–765, doi:10.1016/j.ins.2016.07.007.
[45] C.J.C.H. Watkins, Learning from Delayed Rewards, Ph.D. thesis, King's College, Cambridge, 1989. https://www.researchgate.net/profile/Christopher_Watkins2/publication/33784417_Learning_From_Delayed_Rewards/links/53fe12e10cf21edafd142e03.pdf.
[46] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3–4) (1992) 279–292, doi:10.1007/BF00992698.
[47] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2014) 97–107, doi:10.1109/TKDE.2013.109.
[48] Y. Wu, S.C.H. Hoi, T. Mei, N. Yu, Large-scale online feature selection for ultra-high dimensional sparse data, ACM Trans. Knowl. Discov. Data 11 (4) (2017) 48:1–48:22, doi:10.1145/3070646.