Evolutionary Machine Learning: A Survey

AKBAR TELIKANI, University Of Wollongong, Australia
AMIRHESSAM TAHMASSEBI, Florida State University, USA
WOLFGANG BANZHAF, Michigan State University, USA
AMIR H. GANDOMI**, University of Technology Sydney, Australia

Evolutionary Computation (EC) approaches are inspired by nature and solve optimization problems in a stochastic manner. They can offer a reliable and effective approach to address complex problems in real-world applications. EC algorithms have recently been used to improve the performance of Machine Learning (ML) models and the quality of their results. Evolutionary approaches can be used in all three parts of ML: preprocessing (e.g., feature selection and resampling), learning (e.g., parameter setting, membership functions, and neural network topology), and postprocessing (e.g., rule optimization, decision tree/support vectors pruning, and ensemble learning). This paper investigates the role of EC algorithms in solving different ML challenges. We do not provide a comprehensive review of evolutionary ML approaches here; instead, we discuss how EC algorithms can contribute to ML by addressing conventional challenges of the artificial intelligence and ML communities. We look at the contributions of EC to ML in nine sub-fields: feature selection, resampling, classifiers, neural networks, reinforcement learning, clustering, association rule mining, and ensemble methods. For each category we discuss evolutionary machine learning in terms of three aspects: problem formulation, search mechanisms, and fitness value computation. We also consider open issues and challenges that should be addressed in future work.

CCS Concepts: • **General and reference → Surveys and Reviews**; • **Computing Methodologies → Machine Learning**; **Artificial Intelligence**.

Additional Key Words and Phrases: Evolutionary Computation, Learning Optimization, Swarm Intelligence

ACM Reference Format:
Akbar Telikani, Amirhessam Tahmassebi, Wolfgang Banzhaf, and Amir H. Gandomi. 2021. Evolutionary Machine Learning: A Survey. *ACM Comput. Surv.* 1, 1, Article 1 (January 2021), 35 pages. https://doi.org/10.1145/3467477

1 INTRODUCTION

Finding patterns in data is the core and most important step in Machine Learning (ML). Doubtlessly, one of the most successful early applications of its principles was conducted by Turing when he used it to help crack the Nazi military's vexing Enigma machine by building a machine that could quickly sort through millions of possibilities to divine the code. Then in 1950, an approach called "learning machine" was proposed by Alan Turing to implement the principles of evolution [175]. Today, the most recent and powerful ML techniques are inspired by nature and are known as the field of natural computation. The concept and terminology of natural computation has two essential sources: 1) taking inspiration from nature and 2) employing computers. This terminology can be used to simulate a natural phenomenon,

*Corresponding author

Authors' addresses: Akbar Telikani, akbar.telikani@gmail.com, University Of Wollongong, Australia, ; Amirhessam Tahmassebi, Florida State University, , Tallahassee, FL, 32306, USA, atahmassebi@fsu.edu; Wolfgang Banzhaf, Michigan State University, East Lansing, MI, 48824, USA, banzhafw@msu.edu; Amir H. Gandomi, University of Technology Sydney, Ultimo, Sydney, Australia, 78229, gandomi@uts.edu.au.

employ natural materials, or develop novel techniques to solve problems. As part of Artificial Intelligence, Evolutionary Computation (EC) approaches are considered one category of this field, with their power stemming from the processes nature used to produce intelligent organisms. The processes applied in EC are inspired by natural evolution and the best solutions nature has evolved over millions of years. As a result, EC techniques can be expected to be efficient and effective. EC algorithms generally work with populations of individuals that are associated with a specific problem to be solved.

While evolution and learning are two aspects of adaptation in both natural and artificial systems, one can discern them on the basis of the lifetime of an individual. We speak of learning adaptation if an individual, *during its lifetime* adapts to a certain problem domain. We speak of evolutionary adaptation, if an individual is part of a hereditary sequence of individuals whose features are changing *over the course of generations*. Similar to natural systems, where evolution and learning complement each other, there is a bidirectional relationship in computing between EC techniques and learning algorithms so that they can be combined to attack complex optimization problems in different domains together, e.g., in the energy, machinery, medical, engineering, and pharmaceutical industry. On the one hand, learning algorithms are integrated into evolutionary techniques to address problems with EC approaches, such as being trapped in local optima and premature convergence. Some works presented in this regard are, for example, a Cuckoo Search algorithm that has been improved using Q-learning [96][97], adaptive learning [94], the Taguchi method [95], and balanced-learning strategies [98]. Learning algorithms have also been used in Particle Swarm Optimization (PSO)[148] and elephant herding optimization [99][100].

On the other hand, evolutionary algorithms can be used to improve ML algorithms, the main topic of this paper. Most problems in real-world applications contain inaccurate, noisy, discrete and complex data, for which evolutionary computing algorithms, by virtue of being general-purpose and stochastic search methods, provide great optimization opportunities [183]. In recent years, many researchers have integrated EC approaches into different phases of the ML processes (i.e., preprocessing, learning, and postprocessing) in order to address the limitations of traditional approaches. These new and hybrid methods are known as Evolutionary Machine Learning (EML). EC in the learning phase of ML also refers to evolutionary AutoML concepts, in which different expert-designed components of ML models, such as architecture and hyperparameters, are automatically determined using EC approaches. Also, optimization algorithms, such as gradient-based training algorithms, are replaced by EC algorithms or even invented by an EC approach [103, 144].

A number of surveys and review papers have been published that cover specific aspects of EML. For example, Al-Sahaf et al. [3] published a review paper that addresses major EML tasks such as classification, regression, and clustering. Badhon et al. [15] published a review paper that addresses Multi-Objective Evolutionary Algorithms (MOEAs) for Association Rule Mining (ARM). Also, Telikani et al. [169] published a review paper on the application of EC techniques for ARM. In addition to the recently published paper regarding evolutionary feature selection [183] Barros et al. [17] published a survey of evolutionary algorithms that were designed for Decision Tree (DT) induction. Four survey papers were published to address evolutionary clustering with [61, 64, 129, 133]. Mukhopadhyay et al. [129] published a survey of multi-objective evolutionary clustering techniques looking at different aspects including representation techniques, objective functions, evolutionary operations, strategies for maintaining non-dominated individuals, and final individual selection. Darwish et al. [38] reviewed the application of swarm intelligence and EC approaches to deep learning. Mukhopadhyay et al. [130, 131] published a two-part survey discussing recent developments in multi-objective evolutionary algorithms for data mining problems such as feature selection, classification, clustering, and ARM.

Focus and content of this paper are somewhat different from those surveys. Whereas other surveys focused on EC algorithms designed for a particular task/aspect such as feature selection [183], DT [17], ARM [15, 169], clustering

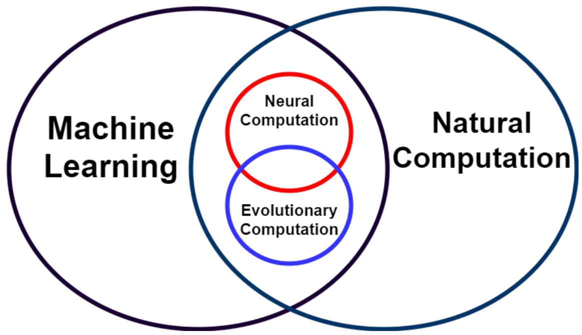


Fig. 1. The intersection of natural and evolutionary computation in the context of machine learning and natural computation

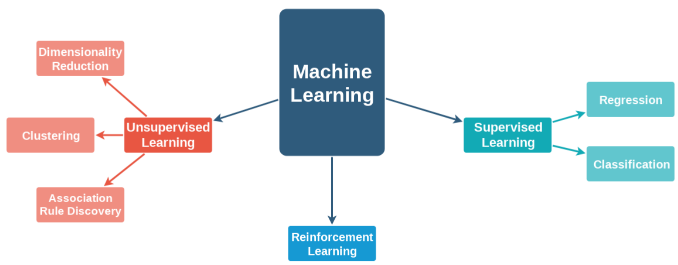


Fig. 2. Taxonomy of machine learning

[61, 64, 129, 133], and deep learning [38], this paper investigates the more general question of how EC algorithms are applied to different aspects of ML and how ML problems can be formulated for optimization using evolutionary search mechanisms. We discuss EML in light of the following three aspects: problem formulation/individual representation, search mechanisms, and fitness function.

The rest of the paper is organized as follows: Section 2 provides background information on ML and EC. Section 3 considers the application of EC to different parts of ML and presents an overview of EML approaches. Section 4 reviews applications of EML approaches. Section 5 discusses current issues and challenges. Finally, Section 6 holds a critical summary of the current state of the art in light of present issues and challenges.

2 FUNDAMENTAL CONCEPTS

The next subsections provide a categorization and the characteristics of machine learning and evolutionary computation. Fig. 1 shows how neural and evolutionary computation concepts intersect and EC as they relate to ML and natural computation.

2.1 Machine Learning

Machine learning, a subset of artificial intelligence techniques, applies algorithms to extract patterns by using mathematics, statistics, optimization, and knowledge discovery methods. Fig. 2 illustrates the basic taxonomy of ML, consisting of three main categories: (1) Supervised Learning, (2) Unsupervised Learning, and (3) Reinforcement Learning (RL) [21].

Supervised learning, the most well-known ML data processing task, attempts to find relationships between a set of inputs and outputs that are provided for training the system [177]. A mapping function from an input x with the best estimation of output y ($f:x \rightarrow y$) is applied at the end of the training process. Supervised learning algorithms *build a model* representing the relationships among the input features used to forecast the target outputs [84]. These algorithms include two main categories: (1) classification (discrete modelling) and (2) regression (continuous modelling). Both categories are predictive modeling techniques; the only difference is their target (response) variables. In classification, the target variable is in the form of categories (class labels), as in binary-class or multi-class problems. In regression, however, the target variable is continuous. Fig. 3 shows a schematic of model building using ML algorithms.

Raw data is the only input to *unsupervised learning*, which - unlike supervised learning - does not have target variables available to supervise the learning process. Unsupervised learning can be categorized into three main categories: (1) clustering, (2) association rule discovery, and (3) dimensionality reduction. This is discussed in more detail in Section 3.

The third category is *reinforcement learning*, widely used to address Markov decision processes. In RL, an agent learns to act in its environment with its own optimal *policy* through interaction with said environment. RL focuses on maximizing the reward for an agent by actions in the environment [177]. The essence of RL involves an autonomous agent, as illustrated in Fig. 4, such as a person, animal, robot or software agent, that navigates an uncertain environment with the goal of maximizing a numerical reward. That reward, however, is not immediate after an action, but only after a sequence of actions that have gradually changed the environment for the agent. Sports are a good example of RL; our autonomous agent would have to deal with the strategy and continual actions that occur in a sporting event such as a tennis match. In a tennis match, the agent would have to consider actions like serving, returns, and volleys. These immediate actions change the state of the game described by the current set; the player currently ahead; and similar state variables which are part of the tennis rule book. Every action is performed to receive a future reward, such as winning a point that leads to winning the game, set, or match. The agent is required to follow a policy, or a set of criteria, rules, and strategies, to maximize the final score achieved at the end of the game. One important question to be addressed is how agents can model the game when the agent's actions change the state of the environment. At the outset the initial inputs to the model are a state and the corresponding action that generates the maximum expected

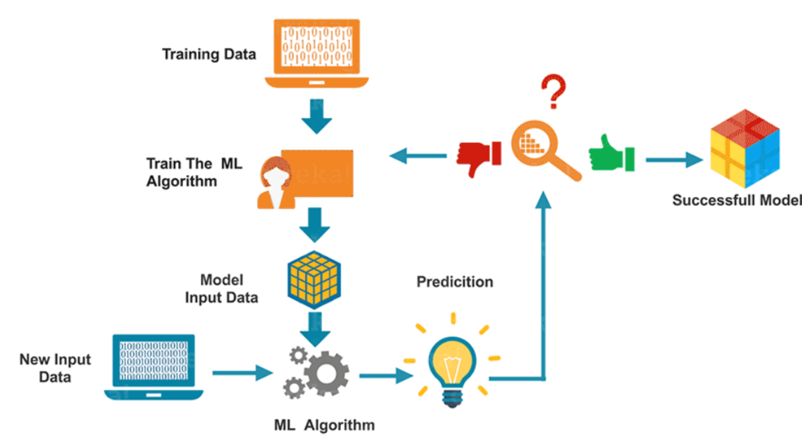


Fig. 3. Schematic of model building using ML

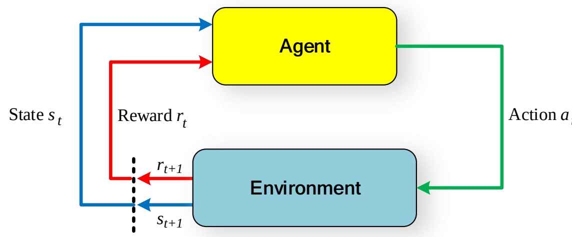


Fig. 4. Schematic process of reinforcement learning

reward [128]. Over multiple attempts, RL refines those reactions based on the response of the environment and the rewards it receives.

2.2 Evolutionary Computation

EC approaches are inspired by the principles of natural evolution. An EC approach encodes a problem in terms of individual(s) to be evolved with the aim of improving the quality of problem solutions. Genetic operators, including crossover, mutation, and selection, are applied to produce new individuals. Based on a *differential fitness survival* mechanism, only the best individuals remain as source of further variation. EC algorithms explore the search space using an iterative heuristic procedure to obtain gradually better solutions [143]. Before we go into details, we need to clarify one point: There are two types of these meta-heuristic algorithms: population-based and single solution-based. The former approaches start an evolutionary process using a set of initial random (or otherwise created) solutions. Examples include the Genetic Algorithm (GA) [63], Ant Colony Optimization (ACO) [119], and Particle Swarm Optimization (PSO) [74]. These are the ones we are discussing here in more detail. Then there are single solution-based approaches, called trajectory optimization, which start from one initial random individual. Tabu search [51] is an example of a single solution-based algorithm, as is simulated annealing [79]. Fig. 5 shows a categorization of the population-based approaches, divided into four categories: bio-inspired, physics-inspired, geography-inspired, and cultural-inspired.

(1) Bio-inspired: This category includes swarm intelligence (SI)-based approaches and evolution-inspired algorithms, which originate from the natural behavior of organisms. SI simulates how swarms (e.g., birds, fish, and insects) behave in

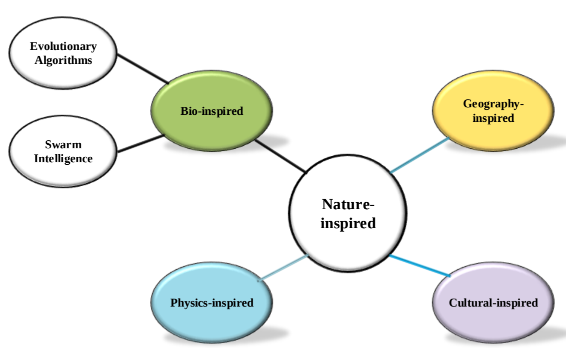


Fig. 5. Taxonomy of nature-inspired algorithms, with evolutionary algorithms as one branch

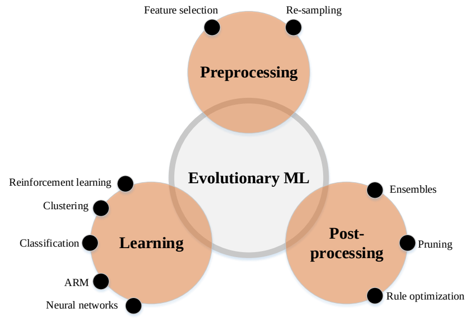


Fig. 6. A classification of evolutionary machine learning approaches

their group life in a colony. Swarm entities can collaboratively perform many complex tasks required for their survival. *Self-organization* and *decentralized control* are two main features of swarm-based systems that lead to emergent behavior due to the local interactions between swarm agents [138]. ACO and PSO are the two first mainstream algorithms of SI. The origins of evolutionary algorithms lie in the Darwinian principles of natural evolution that cause living organisms to become well-adapted to their environment. *Self-organization* and *strong adaptability* are the two main features of these approaches. In these algorithms, entire populations can be replaced from one generation to the next by operators like Selection, crossover and mutation. Genetic algorithms (GAs), evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP) are the four main kinds of evolution-inspired mechanisms.

(2) Physics-inspired: The origin of physics-inspired algorithms resides in physical/chemical rules. For instance, the gravitational search algorithm is an algorithm of this category.

(3) Geography-inspired: These algorithms generate random solutions in the geographical search space; Tabu search falls into this category.

(4) Cultural-inspired: These algorithms are inspired by human behavior seen during cultural interactions with others. Observing natural and inherent behaviors of other people helps individuals to learn new knowledge and improve their own behavior. The Memetic algorithm can be considered one of these approaches that imitates the mutation process through a local heuristic.

3 EVOLUTIONARY MACHINE LEARNING

Evolutionary computation has a wide range of applications in ML. The most important research contributions of EC across different ML areas are summarized in what follows. The three main aspects are: (i) how to formulate an ML problem into an optimization problem in the form of individual representation, (ii) which search mechanism to use for solving a specific ML problem, and (iii) how to compute the quality of solutions for generating a new generation. We organize the EML works into nine sub-fields, focusing on specific ML tasks in which EC has made contributions. Fig. 6 gives a diagrammatic overview of the topics dealt with by the considerations presented in this section.

3.1 Evolutionary Feature Selection/Construction

Some datasets in real-life applications, such as gene selection, comprise thousands, if not tens or hundreds of thousands of dimensions. This is a challenge not only for ML in general, but also for statistics and biology. This problem can be

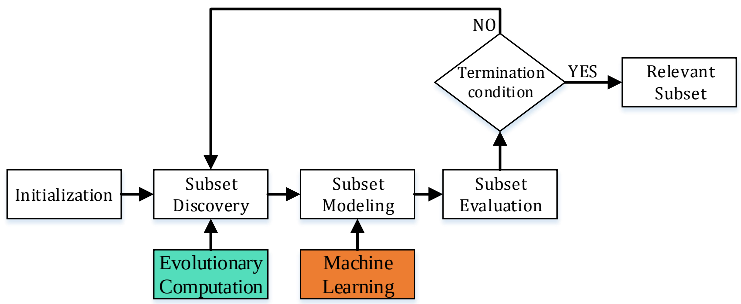
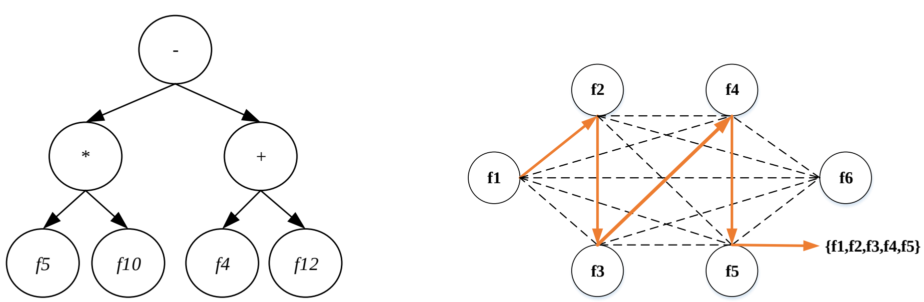


Fig. 7. General evolutionary feature selection process

handled by feature selection and feature construction methods that enhance the quality of the feature space. The former choose only informative features from the original feature set, while the latter create new high-level features [172]. Feature construction can achieve better performance than feature selection if the original features are not informative enough [183]. The application of EC methods in feature selection is known as wrapper approaches. A greedy search strategy is implemented to find an appropriate set of features by utilizing ML as a fitness function [23]. Fig. 7 presents the general framework of evolutionary feature selection.

Step 1 – Encoding: Binary encoding has been commonly used for the feature selection problem. Each solution is a bit-string representation comprising N bits standing for the number of features in a dataset. "1" indicates that the corresponding feature was selected, while "0" indicates that the corresponding feature was deselected. In contrast, most GP work uses a representation in which features that are used appear (e.g., in a tree representation, as leaf nodes, in a linear representation, as registers) and are subsequently considered the final feature set. GP is capable of handling large-scale feature selection since the individual representation does not require information about the selection of all features (e.g. their index). Additionally, there is no need in GP for predefined structures for solutions in order to produce the optimum solution [160]. In ACO, the feature selection problem is represented by a graph in which each feature is considered a node of the graphical model. A node is selected as one of the selected features if an ant visits that node. Fig.8 shows an example of tree-based (Fig. 8a) and graph-based encodings (Fig. 8b).



(a) Tree-based feature selection of $f4, f5, f10, f12$

(b) Graph-based feature selection of $f1, f2, f4, f5$

Fig. 8. Examples of individual representations of feature selection/construction

Step 2 – Initialization: Determination of a starting feature subset is important because initialization directly influences the performance of a search strategy. Forward selection and backward selection are two typical initialization strategies. The process of evolution starts with an empty set in the former; however, missing some of the features in a large search space is the main drawback of forward selection. The latter strategy starts with a full set of features and removes some iteratively. Lengthy computational time is the main disadvantage of backward selection [182]. A Bernoulli process is another well-known technique for generating an initial population. This technique selects corresponding features using a function that produces a random number from $[0, D]$, where D is the size of individuals.

Step 3 – Search Strategy: The use of many heuristic approaches is not practical because of the large search spaces of most of these problems. However, EC algorithms such as a GA, can be used to perform the search process via evolution successive populations [58]. GP is considered a useful search mechanism in filter approaches, in which it is mainly used as a search algorithm, and in wrapper approaches, in which it can be employed as both a search strategy and a classification technique.

Step 4 – Subset Modeling: ML algorithms are employed to build a classification/prediction model using the subset of features that were selected by the EC algorithm.

Step 5 – Model Evaluation: Evaluation methods for models are categorized into three groups: *Wrapper, filter, and embedded methods*. Wrapper methods use the performance of the ML algorithm as its evaluation criterion, while filter methods use the intrinsic characteristics of the data. Longer computation times result for wrapper methods, although the target features usually perform better than features selected/constructed by filter methods. Embedded approaches simultaneously select/construct features *and* learn a classifier. Only GP and learning classifier systems (LCSs) can perform embedded feature selection/construction [183].

3.2 Evolutionary Resampling

A dataset is known as ”imbalanced” or ”skewed” if the number of the instances of a class is much higher compared to that of another class. Skewed distributions influence the effectiveness of ML models, which are biased toward majority classes. Resampling is the most common approach for balancing data distributions and is performed in the preprocessing stage. There are two kinds of resampling methods: undersampling and oversampling. The former removes instances belonging to the majority class, while the latter generates additional samples for the minority class. On the one hand, undersampling may potentially remove useful information regarding the majority classes. On the other hand, oversampling increases the size of the training set, which makes training more complex and burdensome. In addition, random duplication of minority instances makes the oversampling strategy prone to overfitting [168]. Fig. 9 depicts the evolutionary resampling process.

Traditional evolutionary resampling approaches [50] use a binary representation, in which a value of "1" indicates that a record was selected and a value of "0" indicates the absence of an instance in the training set. However, these methods perform poorly when faced with large datasets, because the length of individuals and the search space increase proportionally with the size of the dataset [105]. Modern approaches, in contrast, attempt to circumvent large search space by introducing sparse representations that only contain the indices of those majority class samples that were selected [47]. Fig. 10 shows the difference between binary and sparse representations.

Regarding the fitness function, performance measures such as the accuracy rate are inappropriate for assessing the quality of acquired models, since the performance of both classes is not equally weighted. The F1-score is more suitable for a problem with class imbalance, because it takes into consideration both precision and recall, which generates a single metric that can be used to gauge performance [17].

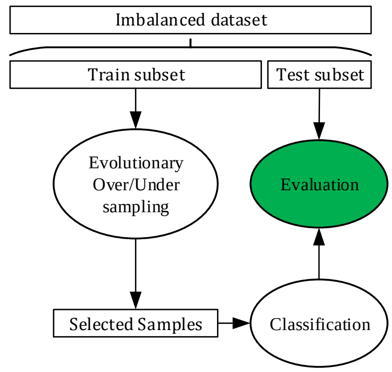


Fig. 9. Evolutionary resampling process [49]

3.3 Evolutionary Classifiers

Data preprocessing methods such as data balancing, feature selection/construction, and data cleansing can provide appropriate data as input to data classifiers. However, these methods are classifier-independent and have their own challenges, such as overfitting and poor generalization caused by resampling methods [19]. Therefore, a modification of classifiers by EC methods is considered next. This section discusses the application of EC algorithms in well-known classifiers such as decision trees, the Support Vector Machine (SVM), and k-Nearest Neighbor (k-NN) algorithms.

3.3.1 Evolutionary Decision Trees. Decision trees (DTs) are one of the most widely used ML representations due to their simple interpretation and their fast construction without the need for domain knowledge. Classic heuristic approaches use a greedy method to select a node for subtree construction. Hence, these approaches apply a locally optimal "test and fail" to converge to globally optimal solutions. EC approaches can be used in DT induction in two ways (Fig. 11): Evolutionary induction of DTs and evolutionary design of DT components. Each individual is a DT in the former, while individuals are components of DT classifiers in the latter.

The training data is split using either a single attribute per node or a (non-)linear combination of attributes in an evolutionary classification tree. Single attribute-based DTs are more common compared to multi-attribute-based DTs, due to their easy interpretation. However, multi-attribute-based DTs are more accurate and smaller, though they require more computation time and loose comprehensibility. A regression tree can be considered a particular type of DT, in which the target value at each leaf node of the tree is a continuous value as opposed to a discrete or nominal value [137].

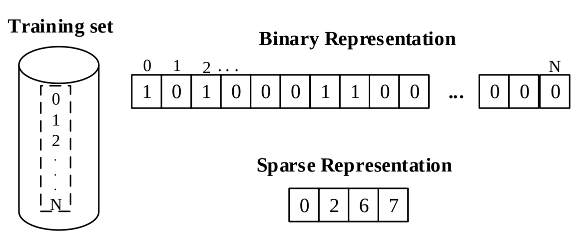


Fig. 10. Difference between a binary and a sparse representation for resampling methods [173]

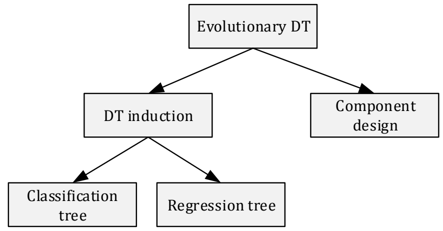


Fig. 11. Evolutionary decision tree types

With regard to problem encoding, tree-based encoding and linear individuals (i.e., fixed-length string representations) are two common approaches used to code individuals in evolutionary DT induction. It is tricky to implement linear individuals for non-binary DTs, so in most studies this type of DT is converted into a binary tree before applying EC algorithms. DTs encoded as linear individuals are easier to handle than those encoded by a tree-encoding technique. However, the need of fitness evaluation for constant mapping between genotype and phenotype and the difficulty with handling non-binary DTs, and of defining a maximum number of bits are some drawbacks of fixed-length string encoding. Some previous studies used dynamic-length string; this generated unnecessary complexity because evolutionary operations, similar to crossover, may have to be modified. As for the fitness function, single-objective optimization and multi-objective optimization are used to evaluate the quality of a DT. Classification accuracy is the most common measure of a single-objective optimization. Some other criteria, such as accuracy, tree size, the number of nodes, sensitivity, and specificity, can be formulated for a multi-objective fitness function.

Escaping from local optima and performing a robust global search are the main advantages of evolutionary DT algorithms, which are able to better cope with attribute interactions compared to greedy DT methods. Another benefit of evolutionary DTs is their ability to apply different measures in multi-objective optimization. However, the evolutionary DTs introduce some negative features as well. For one, EC algorithms for DTs are computationally expensive for large-scale data, because they generally evaluate all candidate solutions in a population for every generation [69]. Fortunately, EC approaches can be parallelized easily, and both the search mechanism and fitness evaluation can be performed on different parallel and distributed platforms such as GPU, and MapReduce.

3.3.2 Evolutionary Support Vector Machine. The idea of support vector machines is based on an optimally separating hyper-plane. The original pattern space in SVMs is first mapped into a high dimensional feature space by using nonlinear functions; then, an optimally separating hyper-plane of the feature space is generated [65]. First, SVMs were successfully applied to binary classification problems. For multi-class classification problems, the problem is divided into multiple binary sub-problems through a decomposition approach. Each sub-problem is then solved by a SVM and the outputs of all predictors are combined [108]. SVMs were subsequently used for regression prediction and time series forecasting.

Higher risk can be expected for a classifier with a smaller margin. Some slack variables are generated if the data cannot be separated linearly. Therefore, a convex quadratic programming problem should be solved to construct a maximal margin [14]. The input space in SVM is mapped into a high-dimensional dot product space when the problem of obtaining an optimal separation plane is not solved in linear space. In this case, a kernel function ("kernel trick") is employed to find the hyper-plane in high-dimensional space without significantly increasing computational cost. Radial Basis Functions (RBFs) (Eq. 1) are a commonly used kernel function technique in SVMs.

Classifier1		Classifier2		Classifier3	
C1	*σ1*	*C2*	*σ2*	*C3*	*σ3*
1024.0	0.001	10.0	1.25	96	0.5

Fig. 12. Different parameter values for each binary classifier in a three-class classification problem [108]

$$K(x_i, x_j) = exp(-\frac{\|(x_i, y_j)\|^2}{2\sigma^2}) \quad (1)$$

The kernel parameter σ influences the data mapping process and alters data distribution of the higher dimensional feature space [65]. Overall, the high performance of SVMs stem from three factors, the choice of a kernel function, the choice of kernel parameters, and parameter C. An optimization problem is formulated in a SVM to construct a maximal margin classifier, as Eq. 2:

$$\begin{cases} \text{minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{k} \xi_i \\ \text{subject to} & y_i(w_i x_i + b) > 1 - \xi_i; \end{cases} \quad (2)$$

C is a penalty parameter and imposes a trade-off between training error and generalization. The generalization capability of SVMs may be reduced if the selected value of C is too large or too small. The choice of this parameter becomes even more difficult if a decomposition approach is used in multi-class problems, because the number of parameters increases with each binary classifier. The terms k, ξ, w, b are the number of data points, slack factor, a normal vector, and a scalar quantity, respectively.

Several approaches can be used to adjust hyper-parameters: a grid-search algorithm, trial and error, cross-validation, generalization error estimation and gradient descent, and evolutionary algorithms. Using a grid-search algorithm is complex and time-consuming. Trial and error procedures are time-consuming and the results are also unreliable. The cross-validation method requires long and complicated calculations [56]. The gradient descent algorithm in SVM is sensitive to initial parameters. Parameter optimization using EC algorithms to address the aforementioned challenges has received more attention [14].

The actual encoding representation is employed when encoding hyper-parameters in the SVM problem, which avoids the postcrossover overload problem. In this case, an individual X is represented as $X = \{C, \sigma\}$, where C and σ denote the aforementioned penalty and kernel function parameters. Fig. 12 shows an example of a chromosome representation for a classification problem with three classes [108].

The SVM parameter selection task is often performed by retaining the best combination of parameters. Using an exhaustive procedure to explore the parameter space may lead to good results, although this strategy should be avoided for obvious practical reasons. Therefore, optimization techniques are good choices for preventing exhaustive or random exploration of parameters because they explore the search space using good values for the selected objective function. A drawback of these techniques is, however, that they have to start with random settings that are uniformly sampled from the search space. This can make convergence slow, and the algorithm might get stuck in local minima. Meta-learning is a useful strategy for addressing SVM parameter selection and considers this process a supervised learning task [52]. SVM parameter values are recommended by this strategy according to parameter settings that were successfully determined in previous, similar problems. Fig. 13 presents the general framework of evolutionary SVM based on meta-learning.

The SVM algorithm often generates many support vectors which increases the computational time for calculating decision functions. Postpruning is a strategy that can be used to eliminate inappropriate support vectors generated by the standard algorithm. The length of each individual is equal to the number of support vectors in binary representation (which is widely used). The i-*th* support vector is included in the decision function if a bit is equal to "1" and is excluded if a bit is equal to "0".

3.3.3 *Evolutionary k-nearest Neighbors.*

The nearest neighbor technique [33] and its derivatives are a subset of the lazy learning methods. The k-NN algorithm is an extended version of the nearest neighbor algorithm [174]. The k-NN algorithm is a non-parametric classifier, which means that it does not depend on any prior assumptions regarding the data distribution. k-NN algorithms classify an object by a majority vote of its k neighbors, where k is a user-defined parameter. The output classes are obtained through a voting metric that is applied to all distance vectors between the test pattern and the training patterns. To define the number of neighbors, k, is challenging because a certain value of k may result in good performance for one classification problem and fail for another, depending on the distribution of classes in feature space. It has been shown that when $k = 1$ and the number of training samples $n \longrightarrow \infty$, the probability of inaccurate classification by k-NN can be at most twice the risk of the Bayes classifier [33]. However, this is not applicable if the number of training instances available is finite.

In addition to k and the distance function, the importance of neighbor, class, and feature affect the performance of the k-NN algorithm. Similar to neural networks and SVMs, a k-NN algorithm's accuracy benefits from weight optimization in the training phase. These weights can be assigned to neighbor, class, or feature and each type of weight has a special impact on the performance of the algorithm. Class-specific weighting provides a k-NN algorithm with additional knowledge regarding class properties; attribute-specific weighting can be used to remove the effect of noisy and redundant features [22]. The aim of a weighting scheme is to use a good metric that will lead to high classification accuracies with a given set of raw prototypes. An investigation of differential evolution in a weighting system in terms of different aspects of data was previously published [10].

Two commonly used techniques to perform data reduction in neural networks are prototype selection and prototype generation. Prototype selection selects a subset of instances from the original training set by removing redundant and noisy examples [27]. Prototype generation methods are able not only to select data but also to generate and replace original data with new artificial data [174]. Both prototype selection and prototype generation are combinatorial optimization problems; therefore, EC approaches can be used to solve these types of problems and generate excellent results. Prototype selection and prototype generation can be encoded as binary or as continuous space search problems, respectively. EC techniques for prototype generation are based on the positioning adjustment of prototypes, which optimizes the position of prototypes. A drawback of EC techniques, however, is that they are often dependent upon an initial subset of the prototypes extracted from the training set. Also, scaling-up to large datasets is a challenge in prototype selection, since it results in excessive storage requirements, higher time complexity, and lower generalization

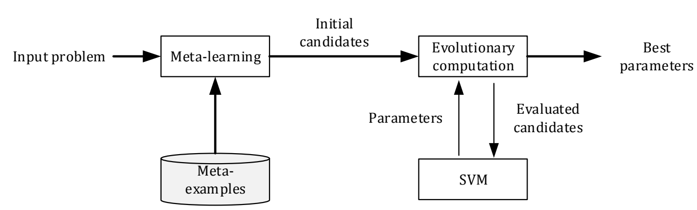


Fig. 13. General framework of meta-learning for evolutionary SVM

accuracy. A prototype-selection algorithm needs to search through all available instances to classify a new input vector and is therefore slow during classification [27].

3.4 Evolutionary Neural Networks and Deep Learning

A standard neural network consists of many connected processors called neurons. Input neurons receive values from the environment while other neurons receive activated values via weighted connections from previously active neurons. The main focus of learning is to find an optimal or sufficiently close to optimal set of connection weights. The success of neural networks largely depends on the architecture, the training algorithm, and the choice of features used in training. Back-propagation learning requires tuning parameters such as learning rate, momentum, and a predetermined structure. Due to its gradient nature, error back-propagation encounters challenges such as slow convergence speed and getting trapped in local minima [70]. It has been proven that gradient descent with manually defined parameters performs poorly in deeper networks, resulting in underfitting or overfitting of the training data [91]. As a result, it is challenging to adjust the parameters and structure of a near-optimal neural network for applications [186].

Evolutionary computing can be applied to neural networks by learning their building blocks (e.g., activation functions), hyper-parameters (e.g., learning rates), architectures (e.g., the number of layers and neurons in each layer), and even the rules for learning. In the early 1980s researchers focused on evolving only the weights of the networks with constrained architectures, including a fixed number of layers and neurons [155]. But in 1994, [54] developed an artificial developmental system for the automatic generation of complex neural networks.

In evolutionary neural networks, the weight matrices are encoded as individuals and are optimized by means of evolutionary operations, such as crossover and mutation. The error produced by a neural network is used as fitness measure. Evolutionary neural networks have two encoding schemes: direct and indirect. The former expresses the existing connections between nodes. This approach requires background knowledge to define a topology (e.g., the number of layers and the number of hidden units). A number is assigned to each neuron and a binary 2D structure $N \times N$ is generated once the topology comprising the N nodes is set up. A value of "1" indicates that a connection exists between two neurons. Feed-forward connections can be guaranteed by only enabling connections between units in layer i and layer $i+1$. The necessity of assumptions about the topology of the network is the main shortcoming of a direct encoding schema, imposing $O(N^2)$ complexity [16]. Indirect encodings, on the other hand, only consider certain important features of the neural network topology rather than the full connectivity pattern, leading to a more compact encoding compared to the direct one. Indirect encodings can be categorized into three main approaches: (1) connectivity parameters that specify the parameters and describe the topology and architecture of a neural network; (2) developmental rules (e.g., recursive equations or production rules) that are used to build a topology; (3) fractal representations of connectivity inspired by some of the processes of biological development [185]. NeuroEvolution of Augmenting Topologies (NEAT) [157] is a well-known algorithm that uses a GA to evolve both structure and connection parameters of a neural network. NEAT used direct encoding with two vectors, one for nodes and one for connections. Each gene defines the connection weight between two nodes; as a result, NEAT is suitable only for small networks. HyperNEAT [156] used an indirect encoding to optimize NEAT for more complex networks.

In deep learning, the use of evolutionary computing has a long history that started quickly after deep learning began to receive significant attention. Cheung and Sable proposed an early approach to neuro-evolution for deep neural networks in 2011 [32] in which EC was used to find optimal values of the architectural parameters of a Convolutional Neural Network (CNN). CoDeepNEAT [127] is an enhancement of NEAT [157] for optimizing topology, components, and hyper-parameters of Long-Short-Term Memory (LSTM). Significant progress in hardware has made the use of deeper architectures increasingly popular, leading to more complex neural networks models with many layers and hyper-parameters.

Despite the successful application of evolutionary learning to address the automatic design of neural networks, one of the major shortcomings of evolutionary neural networks is that they consume a huge amount of resources during the optimization process. Often, thousands of different individuals are evolved, each of which representing a complete training phase of a deep learning model with a complex architecture and evaluation. It was shown early on that evolutionary training is usually computationally intensive and is slower than back-propagation [80]. These algorithms were not practical until 2012 due to the lack of computational resources such as GPUs [53]. Manufacturing specific chipsets and product lines for deep learning is a current technology trend used to address this challenge. Examples of these types of technologies include Google Cloud Tensor Processing Units [41], Amazon EC2 P3 instances [11], and large AI supercomputers such as NVIDIA's DGX SATURNV consisting of 125 servers with a total of 1000 powerful GPUs optimized for deep learning [134].

Today, the field of Neural Architecture Search (NAS) is thriving, with an explosion of research in this area since about 2016 [43]. In NAS, very often hybrid methods are used, in which only architectural hyper-parameters are optimized using evolution, while learning is left to gradient methods. Multi-objective evolutionary algorithms show substantial success recently, as exemplified by [112]. In many of these applications, architectures with minimal complexity are searched, that perform as accurate as possible. This allows more generalization performance with less computational time for training. A further acceleration can be gained by using surrogate fitness functions [111].

3.5 Evolutionary Reinforcement Learning

The three approaches addressing reinforcement learning problems are *value functions, policy search*, and *actor-critic*. The value function approach aims to estimate the expected value of being in each state. In contrast, policy search approaches do not require a value estimation model but instead search directly for an optimal policy. The actor-critic approach employs both of the aforementioned methods [12]. Despite the success of these approaches in RL, there are three main problems: Temporal credit assignment with sparse rewards, a lack of effective exploration, and brittle convergence properties, extremely sensitive to the choice of hyper-parameters. EC algorithms are well-suited to handle each of the problems [75]. Consolidating returns across an entire episode using a fitness function makes EC algorithms invariant to sparse rewards with long time horizons. A population-based approach can lead to diverse exploration. Finally, the inherent redundancy of a population also strengthens resilience and sustainable convergence properties, especially when combined with elitism [35].

In an evolutionary RL algorithm, the fitness value of an individual is the accumulated reward received after an individual operates in its environment. Fig. 14 shows the actor-critic-based evolutionary RL approach in which the reinforcement learner uses the data experiences that the population generates. The policy gradient method is often used to maximize returns in the form of the minimum value of a loss function. This method uses the actor-critic architecture to maintain a deterministic policy and an action-value function critic. In conventional RL, a single reward is achieved once an action is performed by the individual; however, in evolutionary RL, a fitness value (return) is considered for an individual at the end of the lifetime of a population solution or after a sequence of actions (an episode). This characteristic of EC approaches enables them to be directly applicable to episodic RL tasks, such as game playing, where EC algorithms search for optimal function values or optimal policies [126].

One of the major problems with RL is high-dimensional input spaces, such as visual input. This dimensionality problem can be mitigated in two ways: (1) a preprocessor (compressor) can be applied to transform the high-dimensional

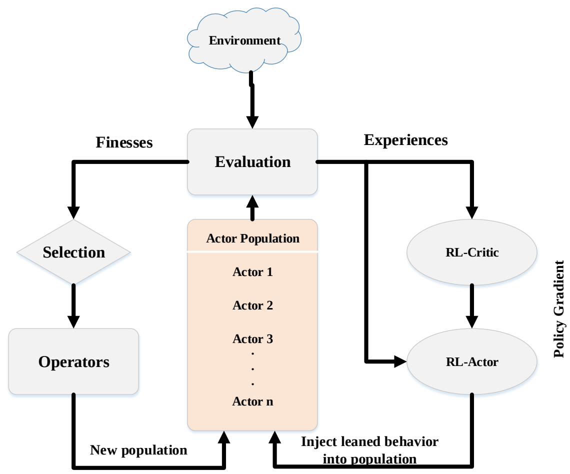


Fig. 14. A schematic of evolutionary reinforcement learning that emphasizes the incorporation of EC population-based learning into gradient-based optimization (inspired by [75])

input spaces into feature spaces with lower dimension, and (2) the representation of neural network controllers can be compressed [82]. The main approach is indirect encoding for transforming small neural networks into networks of arbitrary size through a complex mapping. Another alternative is the combination of action learning with an unsupervised learning compressor to provide a lower-dimensional feature vector as the input of the agent [34]. The combination of unsupervised learning and evolutionary RL was presented in [34, 82]. In contrast, it is not required to perform a compressor phase when a compressed representation of neural network weights is used by evolutionary approaches to train large networks. The use of this technique introduced the first deep neural networks to learn an RL task, directly from the high-dimensional visual inputs [12].

A completely new method for applying EC methods to RL tasks, also introduced in 2017, is the Tangled Program Graph (TPG) method [72], in which a set of linear genetic programs is used to work as a team for solving the RL task. TPGs can work directly on the high-dimensional video input, and have been examined in a variety of game environments. The efficiency gained over deep network reinforcement learning has been used to allow the method approach multi-task learning [73]. Such et al. [158] investigated the performance of GA on deep reinforcement learning for numerous Atari games that are difficult to solve by RL (e.g., Q-learning or policy gradients). The authors found that the combination of DNNs with GA can address sparse reward functions and high-dimensional problem.

3.6 Evolutionary Clustering

Clustering is an unsupervised learning method that partitions unlabeled data objects into several groups according to the similarities among them [64]. The main characteristic of clustering is that there is no prior knowledge required of the data distribution [67]. Partitional clustering and hierarchical clustering are the two main categories of clustering algorithms. Partitional clustering methods divide a dataset into certain groups based on fitness measures over a predefined number of iterations [7, 133]. Simplicity and low computational cost are two main advantages of partitional clustering algorithms, such as *k*-means [107]. However, there are two main problems with these algorithms. First, they are very sensitive to the initialization and the probability of being trapped in local optima. Second, before running the clustering algorithm, the number of clusters must be determined. A small number of clusters can result in a loss of key hidden information. In contrast, a large number of clusters can lead to a high homogeneity of clusters.

A tree topology is used to represent relationships among cluster sets in hierarchical cluster methods. Hierarchical methods can cluster data using either a divisive approach or an agglomerative approach. The former method merges smaller clusters into larger ones, while the latter method splits large clusters into smaller ones [7]. Hierarchical clustering offers an advantage over partitional clustering in that the number of clusters does not need to be specified in advance. However, the disadvantage of hierarchical clustering is that each element can be assigned to only one cluster [4] and performance suffers if a separation of overlapping clusters is done.

Seen from a optimization perspective, clustering is a NP-hard problem. Evolutionary data clustering approaches either use optimization techniques for data clustering or add an optimization technique to existing clustering algorithms. EC approaches attempt to either minimize or maximize an objective function. In the clustering context, intra-cluster distance should be minimized while inter-cluster distance should be maximized [4]. In clustering, EC algorithms have two main goals: determining the number of clusters and specifying the cluster centers. There are two types of individual representations in evolutionary clustering: prototype-based and point-based. The size of individuals is usually smaller and less-redundant when applying prototype-based representations than when applying point-based representations. However, prototype-based encoding tends to prefer round-shaped clusters, where each cluster is represented by a single prototype. In contrast, point-based representations allow capturing clusters with an arbitrary shape.

EC can use many clustering validity measures as fitness function to evaluate individuals. Some studies focus on minimizing the sum of distances between N objects in the dataset and the medoids encoded into the individuals (Eq. 3):

$$F = \sum_{i=0}^{N} d(x_i + m) \tag{3}$$

where m represents the closest medoid to object x_i. This measure is suitable for medoid-based representations.

Minimizing the sum of squared Euclidean distances of the objects from their respective cluster means is another measure of fitness that can be used, (Eq. 4):

$$f(C_1, ..., C_k) = \sum_{j=1}^{k} \sum_{x_i c_j} \|(x_j, z_j)\|^2 \tag{4}$$

where x_i is an object in the dataset and z_j is the mean vector of cluster C_j. This criterion is appropriate for a centroid-based encoding.

3.6.1 Fixed Clusters.

Some evolutionary clustering algorithms work with a predefined number of clusters (k). This technique can be appropriate especially for applications in which there is information about the number of clusters. Evolutionary clustering algorithms focus on addressing the challenges associated with prototype-based clustering, meaning that centroids, medoids, or other vectors that represent a cluster are optimized. Evolutionary algorithms include operators that use probabilistic rules to explore the search space and select better fit partitions with higher probabilities. The parallel nature of EC also allows a straightforward handling of multiple individuals with different distance criteria and fitness functions.

There are three encoding schemes in evolutionary clustering: binary, integer, and real. In a *binary encoding scheme*, each solution has a length equal to the number of instances in the dataset. Each bit corresponds to an instance, i.e., the i-th bit represents the i-th instance. If the i-th bit is "1", then the i-th instance is a prototype. Fig. 15a shows an example of a binary representation with four clusters (k) with objects 1, 5, 7, and 10 being cluster prototypes. These four objects

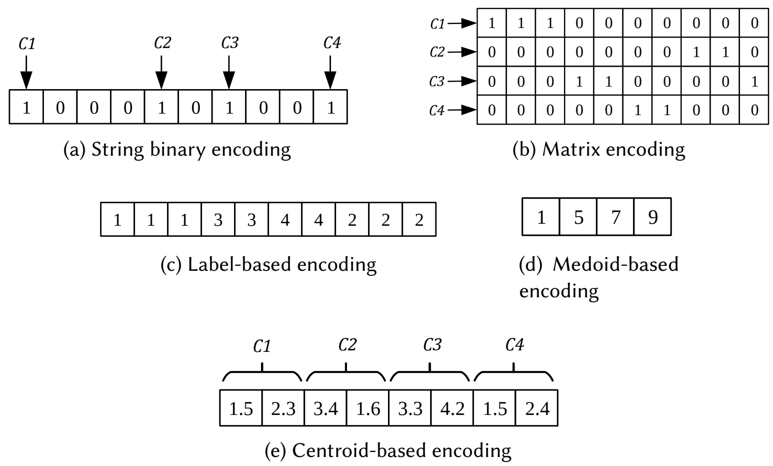


Fig. 15. Different types of encoding in evolutionary clustering

are selected and the similarities of other objects to these instances are calculated. Matrix encoding is another type of individual representation, in which the size of a matrix is $k \times N$ (Fig. 15b). This type of representation requires $O(kN)$ memory space against $O(N)$ space for binary string encoding. However, the computational cost of calculating distance similarity for the matrix encoding is lower than for the string encoding as only the selected objects are considered when computing fitness value.

There are two types of integer encoding: The *label-based representation* (Fig. 15c) and the *medoid-based representation* (Fig. 15d). The former is a vector of N positions, where N is the number of objects. Each bit has a value between 1 and k. The latter provides a medoid-based representation of the dataset using an array of k elements. The length of individuals is equal to k and each element represents the index of an object between 1 and N. The complexity of label-based representations is $O(N)$ whereas it is $O(k)$ for medoid-based representations. However, unlike medoid-based encoding, a label-based representation does not require additional processing to recover clusters encoded in the individual.

In real encoding, the centroid of each feature of the partitions is represented in a *centroid-based representation*. This encoding involves a vector with length nk, where n is the number of attributes and k the number of clusters. Fig. 15e shows an example of a real representation for a dataset with two variables and four clusters.

3.6.2 Variable Clusters. A major benefit of evolutionary clustering algorithms is that they can automatically pratition the data without a prespecified number of clusters and cluster centers [89]. Automatic clustering is helpful, because there is no need to have a priori information regarding the number of clusters. Evolutionary algorithms aim to optimize the number of clusters (k). Certain encoding schemes, such as binary encoding (Fig. 15a and label-based encoding (Fig. 15c) employed in fixed clustering algorithms can be used to encode the variable clustering problem. Also, a different kind of encoding was proposed in which there is a set of axis-aligned hyper-rectangular rules (Fig. 16). Each rule consists of n positions, where n is the number of attributes. The boundaries of the corresponding variables are encoded in each position: l_i and u_i are the lower and upper bounds. Based on Fig. 16, a sample rule could read: if ($1 \leq A_1 \leq 6$) AND ($2 \leq A_2 \leq 5$) THEN (instance belongs to Cluster 1).

GA-based evolutionary clustering was proposed by Bezdek et al. [18] and is one of the earliest successful applications of EC algorithms in clustering. Authors employed the exploratory and exploitative traits of a GA to discover the best centroids. Sarkar and Yegnarayana [149] proposed a clustering algorithm that uses evolutionary programming to determine the number of clusters and cluster centers. EC approaches for partitional clustering were reviewed in [133].

3.6.3 Evolutionary Fuzzy Clusters. Each data object in a fuzzy clustering approach belongs to more than one cluster with a fuzzy membership grade. In order to convert fuzzy clustering into crisp clustering, this approach assigns each data point to the cluster with the highest membership value [133]. Most of fuzzy clustering methods suffer from several inherent drawbacks, such as (1) the user requires a prior knowledge to use a clustering method; (2) different clustering solutions can be generated using random initial choices; and (3) a gradient method is used by an objective, function-based algorithm to search the optimum which can lead to becoming trapped at a local minimum [42]. One other application of EC approaches is to optimize the objective function of a fuzzy clustering algorithm.

3.7 Evolutionary Association Rule Mining

Association rule mining (ARM) aims at deriving the relationship between items in transaction data [1]. ARM has been successfully applied in different domains, such as, e.g., market analysis, recommender systems, or medicine. For example, the patterns extracted by ARM can provide insights into which items are frequently purchased together by customers, which help retailers develop marketing strategies. Classical ARM methods can be divided into two main categories: Level-wise and pattern-growth. Two examples of Level-wise algorithms that use Breadth-First Search (BFS) and Depth-First Search (DFS) to calculate the support value of the item set are Eclat [187] and Apriori [2], respectively. Apriori can generate association rules with high accuracy; however, it needs substantial computation time for large datasets [9]. The FP-growth algorithm [57], a pattern-growth-based algorithm, uses a "divide and conquer" strategy to extract association rules without the candidate generation step [170].

It has been proven that extracting frequent patterns from a transaction dataset is an NP-Hard problem. Traditional ARM methods are dependent on the data preprocessing for discretization, either by means of a user or an automatic process, before applying the algorithm. ARM may be a lossy information discovery process because of the sharp boundary between intervals due to predefined parameters and partitions [123]. Fuzzy ARM deals with this problem by using fuzzy sets to create a smooth transition between a member and a non-member of a set. However, finding a set of suitable Membership Functions (MFs) in fuzzy ARM is one of the main challenges. Overall, the sharp boundary between intervals in quantitative values and distinguishing membership degree for intervals in fuzzy sets are among the main shortcomings of heuristic ARM algorithms.

Mining for frequent patterns with evolutionary means has been introduced to address the drawbacks of conventional rule discovery methods. Rule mining is performed without discretizing continuous attributes so that intervals are obtained in the evolutionary phase to mitigate the impact of the sharp boundary in evolutionary ARM [121]. One application of EC approaches in ARM is rule optimization, in which EC is applied in the postprocessing phase of a conventional ARM algorithm, so that meaningful rules can be extracted by an ARM algorithm such as Apriori.

Rule 1 (Cluster 1)				Rule 2 (Cluster 2)			
1	6	2	5	3	8	2	7
l1	*u1*	*l2*	*u2*	*l1*	*u1*	*l2*	*u2*

Fig. 16. Rule-based encoding

The representation of ARM depends on what type of ARM is performed, with Pittsburgh and Michigan approaches commonly used to encode binary datasets [63]. The Pittsburgh approach encodes different patterns in an individual, whereas the Michigan approach represents only one pattern within an individual. The Pittsburgh technique is more useful for class ARM, where identifying a good set of patterns is the objective. The Michigan strategy, in contrast, works well for mining a set of good patterns and is therefore better at finding high-quality predictions of frequent patterns or rare events. Compared with the Pittsburgh approach, the Michigan technique is simple, straightforward, and syntactically short due to the encoding of fixed-length association rules. Another type of encoding is the binary vector representation, in which each bit represents the presence or absence of an attribute value. Although such a binary string needs to be converted into "IF-Then" rules, it reduces processing speed [139]. This type of encoding is suitable for MFs representation. Each solution encodes the center and the span of a membership function based on the range of an item.

Items and their values correspond to functions of judgment nodes when applying genetic network programming for ARM. The connections of judgment nodes represent the association rules. If a judgment node is satisfied, an attribute is moved to it; otherwise, the attribute is moved to another processing node. Grammar-Guided Genetic Programming (G3P) is an improvement over genetic network programming in which a grammar is used to apply constraints on GP trees [6]. In the case of G3P-based ARM, grammar constraints are created by applying a set of productions rules.

The performance measures for evaluating individuals in evolutionary ARM may conflict with each other and no single individual simultaneously optimizes all functions. But the quality of the solutions can be estimated by both their support and their confidence while they are conflicting. A set of non-dominated solutions is used to provide a trade-off between conflicting objectives [130]. Unlike classification and clustering tasks, in which a single individual is selected from a set depending on user priorities, all non-dominated solutions are considered in a multi-objective ARM as the final set.

3.8 Evolutionary Ensemble Learning

Ensemble learning methods are powerful techniques that generate a final prediction by combining the outputs of multiple models [146]. Ensemble learning has been used successfully to address imbalanced datasets in many different applications. Resampling ensemble techniques are widely employed in such cases. Ensemble performance can be improved when applying a set of accurate and diverse ensemble members [55].

An ensemble methodology comprises several classification tasks; each one is composed of a dataset, an inducer, and a classifier [145]. Three steps are required to construct the ensembles of predictive models: member generation, member selection, and member combination. The first aims to build diverse base models. The second, member selection, is an optional step that uses a heuristic method to prune the pool of models. The third step, member combination, is responsible for generating an ensemble's final output by combining its predictions. These three steps can be formulated as an optimization problem and solved by EC mechanisms. Evolutionary member generation aims to form an ensemble by creating a pool of candidate models. The prediction scores and the complexity values of cadnidates are considered the most important criteria. In evolutionary member selection, candidate models are pruned by EC to build the best possible models for an ensemble. Optimal weights of each candidate model for a weighted average ensemble can be obtained through evolutionary member combination.

Evolutionary ensemble member generation has been used for time series forecasting [25], imbalanced data classification, image classification [5], and fault diagnosis [115]. Because EC algorithms use a population of individuals, they are a natural choice for building potential individual models into an ensemble model. An ensemble learning strategy should then always provide a trade-off between accurate and diverse models, which is summarized by error-ambiguity

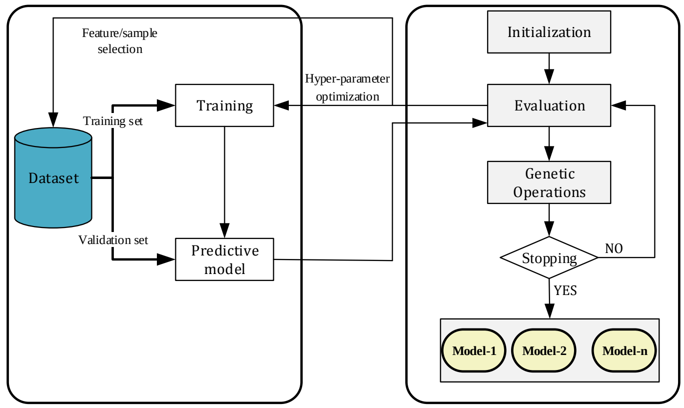


Fig. 17. Evolutionary ensemble member generation

decomposition. This implies that the generalization error of an ensemble is generated by a weighted average of all individual errors and ambiguities. One can then attempt to reduce the overall generalization error by decreasing the generalization error and increasing the ambiguity of each individual which increases an individual's prediction error [151]. Methods for creating diverse ensembles in the member generation step can be categorized into three groups: using different training data, using different learning algorithms, and using different weights or parameters for learning models. Bagging [24] and boosting [125] are two techniques used to prepare different training sets; the former uses random sampling while the latter manipulates the probability of selecting training data from the original training set. An EC method can employ a binary representation to select a subset of the training data in both techniques. Fig. 17 shows the process of evolutionary ensemble member generation in which the performance of the predictive model generated by the training dataset is evaluated on a validation dataset. The predictive model is then optimized by an EC algorithm with regard to the hyper-parameters of the learning algorithm. Alternatively it selects features and instances from the original dataset.

When encoding ensemble member selection, the decision variable is often a binary vector in which each bit represents the selection or not of a base model. This technique has been appplied in sentiment analysis [136] and in the prediction of power transformers' dissolved gas contents [140]. Fig. 18 shows a general view of the process of evolutionary ensemble member selection. A pool of base learners predicts the outputs from a validation dataset, and pruning of the pool optimizes the prediction score generated by the EC algorithm. This is followed by another step that prioritizes the selection to choose the preferable set of models.

A weighted majority voting scheme is performed using an EC algorithm to weight the base models in the process of evolutionary ensemble member combination. This approach improves the predictive performance of the entire ensemble by adjusting the weights of the base models. Fig. 19 presents a general overview of evolutionary ensemble member combination. EC algorithms are used to prioritize and select a preferable set of models built on the validation dataset.

Using the combination of ensemble techniques and resampling approaches (e.g., undersampling and oversampling) to address the class imbalance problem has been shown to enhance correct classification of the minority class. Ensembles based on random sampling would not perform adequately. In fact, potentially useful samples of the majority class can be denied, which may be important for the learning process. This is more evident when the imbalance ratio increases. Evolutionary sampling is a strategy in which the diversity between classifiers that favor the most diverse individuals

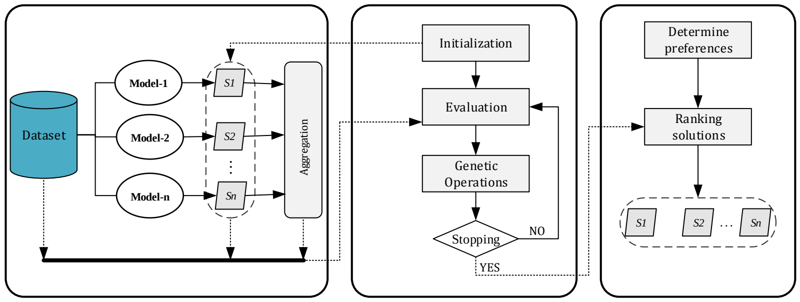


Fig. 18. Evolutionary ensemble member selection

is emphasized [46]. Evolutionary sampling and ensemble methods allow the fitness function to promote diversity of oversampled or undersampled datasets, which leads to more accurate results when dealing with highly imbalanced datasets.

3.9 Evolutionary Model Optimization

EC mechanisms can be used in the post-processing phase of ML when models built by the traditional ML algorithms are optimized. In DT classification, EC algorithms can be recruited as an evolutionary component for pruning the resulting trees to remove all parts potentially affected by noisy or imprecise data, which will prevent both over-fitting by the DT model and reduce the complexity of the final DT. However, it is not easy to find the right trade-off between pruning level and prediction accuracy. Over-pruning can significantly distort the DT, so that only a small portion of training data is represented. In contrast, under-pruning might cause the DT to over-fit the training data. The two major strategies for tree optimization are pre-pruning and post-pruning. Implementing a threshold for each sample is a common solution in the pre-pruning strategy that restricts each expansion if model performance is below a predetermined threshold. Unlike pre-pruning, post-pruning needs to grow a full tree. A full DT is first built by overfitting the training set. The tree is then pruned to both improve its performance and to minimize its size. In practice, post-pruning performs better than pre-pruning [117]. Again, binary encoding is a well-known technique for representation: The length of a solution is equal to the number of branch nodes in the DT. A value of "1" indicates that the branch node was selected for the resulting tree; otherwise, it will not be selected.

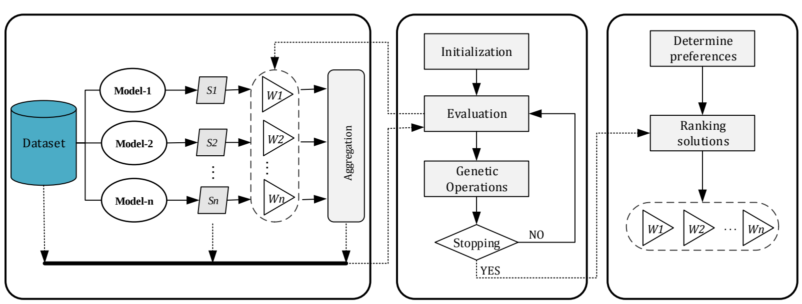


Fig. 19. Evolutionary ensemble member combination

Table 1. A summary of evolutionary machine learning in real-world applications

Category	Applications
Computer networks	Network security [40, 106, 110, 116, 150, 179], Email spam detection [153], Wireless sensor networks[142], Web mining [59], Phishing detection [167]
Business	Marketing [44], Market basket analysis [62], Recommendation systems [190], E-commerce [20], Workflow analysis [104], Collaborative filtering [176]
Robotics	Autonomous vehicle navigation [154], Robotics [192]
Medicine	Disease diagnose [37], Medicine [90, 171], Cancer classification [29], Gene expression data analysis [114], Cardiovascular disease detection [181], Healthcare [88], Thoracic surgery [118], Gastrointestinal infection prediction [152]
Computer vision	Face recognition [178], Handwriting recognition [135], Speaker recognition [188], Personnel identification [30], Character recognition [60], Pedestrian detection [159], Handwritten digit classification [76], Image segmentation [184], Image clustering [39], Document clustering [87]
Industry	Finance [86, 132], Software engineering [78], Construction industry [31], Garment industry [92], Product design [45, 66], Product service system [191]
Environment	Analyzing ozone content [121], Traffic congestion prediction [180], Road traffic prediction [101], Atmospheric pollution [122], Forecasting ozone [120]
Others	Astronomy [28], Education system [113, 147], Car park occupancy prediction [26], Energy price [141], Energy consumption prediction [8], Smart cities [85]

4 APPLICATIONS OF EVOLUTIONARY ML

AI and ML have the potential to usher in another "industrial revolution" able to build intelligent systems automatically. This will not only support many industrial and professional processes but also has the potential to improve everyday living. Different circumstances reduce ML performance of traditional ML in real-life applications. The lack of expert knowledge for running traditional ML effectively is a major challenge for industry and businesses because the quality of ML results critically depends on expert experience to determine hyper-parameters and other adjustments regarding the model design. An EML approach can be a useful substitution if domain knowledge is not readily available. For example, two main problems in traditional neural networks are the definition of the network topology and the adjustment of hyper-parameters; these both require substantial background knowledge of the use case. For instance, high accuracy is necessary for patient diagnoses when applying neural networks to areas such as cancer detection. Inappropriate choices will affect the performance detection system and potentially imperil patient outcomes.

In practice, optimization is critical for minimizing or maximizing objectives due to limitations of resources, like time or budgets, which is relevant in all industries and other business activities. Almost all ML problems can be cast as explicit optimization problems. Training ML models with evolutionary optimization approaches should improve objectives associated with an application. Evolutionary algorithms can update the parameters of ML models in cooperation with a loss function. Table 1 shows some of the applications of EML to solve different real-world problems. These applications cover different fields, such as computer networks, business, computer vision, and robotics.

Computer Networks: EML can be used in computer networks to improve the performance of ML models in different areas, such as network security, sensor networks, or web mining. According to the literature, securing the networks is the primary focus of applying EML approaches to computer networks due to malicious activities that could threaten privacy, integrity, and network resource availability. Evolutionary approaches aim at improving the performance of ML algorithms, such as ARM, clustering and classification (e.g., deep learning classifiers), mainly to secure computer networks against attacks such as intrusion, email spam, and phishing.

Business: Business was the initial target of EML methods such as ARM and prediction. Market basket analysis finds relationships between purchased items that support decision making about store layouts and marketing policy. E-commerce websites (e.g., Amazon and eBay) analyze customer activity to extract personalized preferences and interests as well as to recognize user trends. Recommender systems in businesses, which can be constructed based on knowledge generated by EML techniques use information discovery techniques to offer items to potential customers. The evolutionary ARM is one of the techniques used in collaborative filtering in which user preferences for items of interest are expressed as ratings.

Computer vision: Computer vision is one of the most challenging applications of ML techniques. A large search space in multimedia tasks makes traditional ML methods difficult (such as getting stuck in local optima and/or high computational costs), so evolutionary feature selection/construction/extraction approaches have been a mainstay in this area. Evolutionary applications in deep learning have been successfully employed in computer vision and speech recognition. The application of EML to computer vision can be grouped into two classes: application domain, such as medical or robotics, and target task, such as face recognition or image segmentation. For instance, the definition and measurement of threshold values are two challenging tasks in image segmentation that can be addressed by evolutionary algorithms.

Robotics: RL for robotics can help to autonomously discover optimal behavior through trial-and-error interactions with the environment. When applying RL for robotics in environments with very high dimensions and sparse reward, however, traditional RL techniques cannot improve behavioral learning. The application of evolutionary RL in robotics allows autonomous robots (e.g., vehicles or production lines) to learn behavioral skills with minimum human interaction. Indeed, the integration of ML and evolutionary optimization dramatically increases the decision-making quality and learning ability of decision systems. The full potential of evolutionary optimization has not been reached in Robotics yet, as traditional ML approaches have shown the ability to compete provided that reliable dataset is available. But complex environments and inefficient heuristic optimization functions provide an opening for EML techniques in Robotics.

Recent progress in hardware, such as cloud computing and GPU devices, have allowed previously impossible EML tasks to become addressable. Large corporations such as Google, Microsoft, Uber, or IBM have invested in EML methods and actively pursue solutions for real-life situations.

5 DISCUSSION AND CHALLENGES

5.1 Discussion

EC algorithms have been applied to ML techniques to mitigate problems associated with conventional and heuristic ML techniques. Fig. 20 shows a classification of EML tasks and the challenges associated with the task that EC approaches try to address. The tasks associated with feature selection/construction and resampling methods are the main contributions of EC algorithms to the preprocessing phase. The former contribution is focused on generating a new feature space,

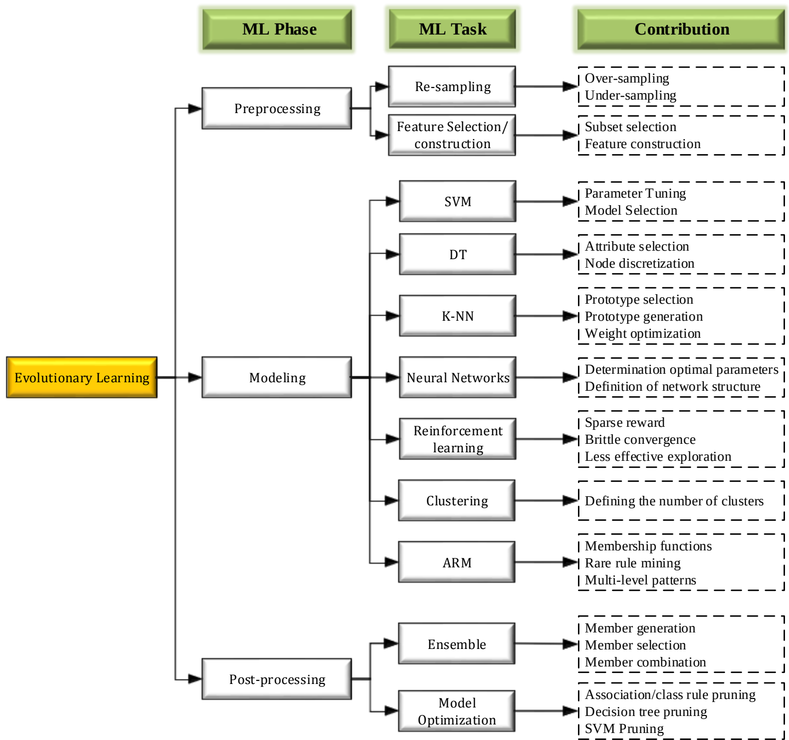


Fig. 20. Classification of evolutionary machine learning

either by selecting a subset feature or by constructing a new set of features from the original features. The latter contribution is achieved by evolutionary undersampling and oversampling techniques.

But the main focus of EC algorithms is to enhance the performance of the learning algorithms. EML algorithms can be categorized into three main classes: supervised evolutionary learning, unsupervised evolutionary learning, and reinforcement evolutionary learning. Evolutionary classification/prediction and evolutionary ensembles are the main contributions of supervised learning. In the realm of unsupervised learning, EC can be used to do feature selection, clustering, dimensionality reduction, anomaly detection among other tasks. Different challenges of RL, such as a long time horizons, the sparse reward, the need for complementary correction mechanisms, and high dimensional action and state spaces can be addressed by integrating EC approaches into reinforcement learning. In particular, EC techniques are often employed as policy search mechanism in RL.

Weighting in ML is a common technique used to emphasize certain characteristics of the data that improves the resulting models. A weighting system can be used, for example, to outline the importance of certain particular instances or features or to rank a set of techniques in the context of ensembles [124]. Neural networks, SVM, and k-NN are the most common techniques that benefit from weights. The main goal of a weighting system is to optimize a set of the model weights in the training phase. Weighting can also be applied to the voting system of the k-NN and EC approaches can be utilized as a weight optimizer in ML.

Overfitting cannot be neglected in classifiers in which the performance of a model on a training dataset is high but is low on unseen data, which results in poor generalization. The large complexity resulting from high depth of deep learning models with their network topology and neural architecture, and from imbalanced or high-dimensional datasets are some of the reasons behind the overfitting issue. While evolutionary algorithms can contribute to addressing each of these factors, their two main contributions are to provide appropriately balanced and feature-reduced input sets for classifiers. Also, learning mechanisms of neural networks often converge to local minima since the loss functions are almost always non-convex [38]. Evolutionarily coded cost-sensitivity is a strategy that improves loss functions to add robustness to classifiers against imbalanced datasets. The use of multi-objective approaches in the optimization of neural networks and deep learning can further balance accuracy with generalization. The penalty parameter C in an SVM provides a trade-off between the generalization and training error [108]. One of the main objectives of the evolutionary approaches in SVM is to optimize parameter C to improve generalization. The optimization and pruning techniques used in decision trees and SVM models lead to be more generalized models [117].

In fuzzy ML, membership functions are used to transform numeric values into linguistic terms. The choice of membership function affects the discovery of patterns in fuzzy ML; thus, learning or tuning membership functions is beneficial. In the traditional fuzzy ML, it was assumed that membership functions were known in advance. However, having prior information of the most effective fuzzy sets covering all domains of numerical variables is not possible. Extracting membership functions using an EC algorithms is a main trend in EML, as regards evolutionary ARM tasks.

Pruning strategies for model optimization have been successfully applied to DT, SVM, and ARM. Here, a model is built using a traditional ML algorithm and EC is then used for model optimization. Most of the traditional ARM algorithms can extract an overwhelming number of rules that often contain redundant and irrelevant information. For example, tree pruning is an ML technique that is used to minimize a DT's size to reduce the complexity of the classifier and improves its predictive accuracy. Some of the DT's subtrees are replaced with leaves in the tree pruning process. SVM algorithms often generate enormous support vectors, which cause a reduction in the speed of decision function convergence. Besides, due to the overfitting effect, the resulting SVM model may adapt itself to noise in the training set rather than to the true underlying data distribution, failing to correctly classify unseen examples. Pruning support vectors in trained SVMs can obtain faster and more accurate SVMs. EC algorithms are the most important techniques for pruning models and patterns extracted by ML algorithms.

EC approaches enable us to develop ARM algorithms for the extraction of association rules without the frequent item-set mining step, which leads to a reduction in computational complexity. However, the main focus of evolutionary ARM algorithms is to deal with quantitative data, in which either discretization of values into appropriate intervals or derivation of membership functions by EC approaches for fuzzifying the quantitative transactions are considered.

The combination of the EC approaches with the hierarchical clustering algorithms still remains untouched in the literature. This is probably due to the fact that defining a fitness function that is capable of guiding evolution is not straightforward. Only a few studies have addressed this topic to the best of our knowledge [68, 109].

5.2 Challenges and Future Insights

Over the past several decades, a variety of EC algorithms have been applied to ML tasks, yet some serious issues remain still insufficiently researched. Some of these major research gaps are described next.

Lack of experimental results: Various surveys have attempted to provide a classification of papers, which published on EML methods, using a research methodology. Since the mathematical analysis of run-time, convergence guarantee, and parameter configurations are an essential need, it makes the selection of a proper EML algorithm for real-world applications a challenging undertaking by organizations and practitioners. EC algorithms can be successfully used for parameter tuning of neural networks and SVM; however, for a novice user, it is difficult to judge which algorithm to use for a particular task. Most proposals have focused on comparing an EML algorithm with non-evolutionary or traditional techniques. However, none of these studies systematically compared the performance of different EC algorithms in multiple ML tasks. We were unable to find a comparative study that identifies which EC technique achieves better performance in terms of parameter settings, structure design, computational complexity, and other aspects. The unavailability of such surveys may be due to a variety of reasons, including the lack of publicly available source code for evolutionary ML approaches, variation of encoding techniques, different objective functions and evolutionary operators. But presenting such a systematic comparison should help new ML users to select a suitable method for a particular application.

Lack of surveys on evolutionary machine learning: A variety of survey papers have been published on different aspects of evolutionary ML, such as clustering, DTs, neural networks, and deep learning. However, there is a lack of comprehensive studies regarding the application of EC algorithms in other fields such as RL, resampling, classification, SVM, and ensembles. For example, different review papers on RL have been published focusing on different aspects, such as RL in robotics [81], deep RL [13], and safe RL [48]. However, evolutionary RL has not been reviewed to date. This area of research should be considered and further studied.

Modular EML: Most EML models are especially developed to address particular problems and cannot be applied to different domains. Modular learning is a possibility for applying ML models to different problems, in which various versatile models are built and learning can be carried out by small autonomous modules. Each independently-trained model would aim at solving a particular subtask that is common among a large number of ML problems. A large problem can be addressed when there is a cooperation between different models. Training models autonomously means that they can be reused in other fields. Different issues should be taken into account in modular learning, such as identifying subtasks and defining their specific modules, determining candidates from a set of previously learned modules, and creating a coherent and effective model by connecting the modules. Multi-task learning is one way to approach this problem [71].

Transfer learning: The main idea of transfer learning is to reuse previously learned models for a new problem. This is a relatively new research area in ML community. The idea has recently become more important with the continuous growth of problems. For example, handwritten character recognition models can be used to recognize characters from digitized books. In summary, different ML problems have certain common aspects that require the ability to transform some of the expertise obtained for one problem to others.

Evolutionary CNNs: CNNs contribute to large applications and have been successfully used in numerous fields. However, evolutionary CNNs have remained an unexplored field, which only recently has received attention. This is a promising research line that provides various opportunities for researchers. The automatic evolutionary design of a CNN topology is a very promising area in need of further study.

Multi/many-objective EML: Standard EML algorithms typically optimize only one objective in the model development process, while most of ML problems have different objectives to be optimized. For instance, an ARM problem has objectives such as support, confidence, and comprehensibility that all must be optimized simultaneously. The choice of an objective functions is an important issue in multi-objective EML. Most algorithms optimize two objectives, and only few algorithms can optimize more than three objective functions simultaneously. Multi-objective algorithms such as NSGA-II, PESA-II, and SPEA2 face difficulties when solving problems with more than four objectives. Currently, the use of such approaches in ML has attracted little attention in the literature. Further, evolutionary algorithms use operators such as selection, crossover, and mutation. The selection operator is mainly influenced by the multi-objective evolutionary algorithm that is used as an optimizer for ML. Crossover and mutation operators, in contrast, are often determined by the encoding strategy. The method of selecting a final solution is one of the most important tasks in multi-objective ML. Multi-objective evolutionary algorithms provide a set of non-dominated solutions in the final generation, and it is important to select one solution from this set. Objective-based, knee-based, and ensemble-based methods are three primary selection techniques. Pareto dominance is used to find the relationship and compare solutions in multi-objective problems [162–164]. However, as the number of objectives increases beyond three, Pareto dominance alone is no longer satisfactory [93]. Such problems that necessitate increased algorithmic complexity are called "many-objective optimization problems" [36]. They appear in different real-world areas such as air traffic control or nurse rostering. Integrating multi/many-objective evolutionary approaches into ML models can solve a diverse set of application problems.

EML on big data: Big data offers new opportunities for ML, but it also brings challenges such as computational costs, huge high-dimensional sample sizes, storage impasse, and error extent [161]. Most studies of evolutionary ML have only focused on the quality of ML models, whereas computational efficiency, a critical issue in seriously large-scale ML problems, has attracted less attention. The costs of searching mechanisms and fitness value computations are major challenges in large-scale EML processes because a population of individuals is evaluated in each generation in EML approaches. An EML algorithm should show good scalability when a dataset increases in size. These types of datasets require large memory and long computation times. The scalability issue may limit the applicability of EML algorithms on large-scale problems. Parallel/distributed evolutionary ML using big data processing technologies, such as master/slave, MapReduce, and CPU/GPU architectures, are one of the major solutions to deal with large-scale EML [165].

Evolutionary cost-sensitive learning: The cost difference between mis-classification errors can be quite high in some classification problems. For instance, in a cancer diagnostic system in which each class represents whether a person has cancer or not, wrongly classifying a patient as healthy will result in a much greater cost compared to classifying a healthy person as a patient. Therefore, a wrong diagnosis may cause a treatment delay or the patient's death [166]. Cost-sensitive learning is a strategy for minimizing the overall cost of learning that creates learning models in such a way that the training process is more sensitive to the classes with higher costs. In addition to the mis-classification cost, test cost is another important type of cost in real-world applications, including money, time, or other resources. Some methods have been recently proposed in the cost-sensitive learning field that attempt to integrate class-specific costs into ML algorithms such as deep learning [49, 77] and DTs [83, 102]. However, to date the integration of tEC algorithms and cost-sensitive learning in ML classifiers has received little attention. Due to lack of prior knowledge, misclassification costs are usually unknown and hard to choose in practice. Recently, an evolutionary cost-sensitive DBN has been developed in which an adaptive DE is employed to optimize the mis-classification costs used in the cost function [189].

6 CONCLUSIONS

Evolutionary computation algorithms have focused on addressing particular challenges of traditional ML tasks. In this paper, we surveyed the importance of EC algorithms in ML tasks regarding with respect to various key aspects of their design, such as problem encoding, search mechanism, fitness function, and the different challenges that EC algorithms have tried to address. We studied nine different tasks in which EC algorithms made significant contributions. An ML problem in EML can be formulated in terms of three major representations: graph (which are suitable for ACO), tree (which are suitable for canonical genetic programming), and vector (which are used by most EC algorithms such as GA, PSO, and ABC). Search mechanisms can be used to find optimal solutions, either based on single solutions or on populations of solutions. Each task has specific evaluation measures that are formulated in the form of a fitness function. For example, accuracy, recall, sensitivity, specificity, and precision are main objectives in classification applications. Evaluation measures can be considered single-objective or multi-objective.

We described various fields in which existing evolutionary ML algorithms have been applied, including medicine (thoracic surgery and disease diagnosis), computer networks (intrusion detection, traffic classification, and email spam detection), image and video processing (face recognition and handwritten recognition), and the environment (e.g., atmospheric pollution, analyzing ozone content, and forecasting ozone). It appears that EML techniques can play a significant role in AI and ML in the future and are expected to broaden their application reach further.

EML still suffers from some problems that have not yet been addressed. It appears that major research efforts are necessary for evolutionary cost-sensitive ML, modular EML, transfer learning, EML on big data, and multi-objective EML. It is expected that, in the next few years, the integration of EC algorithms with deep learning will speed up training processes while balancing accuracy.

Also still lacking are comparative studies that would be helpful for assessing the effectiveness of EC approaches in different applications and ML tasks. There are often concerns about the utility of a specific EC algorithm for solving a wide variety of ML problems. Different statistical tests should be conducted. Additionally, some surveys would appear to be useful in the EML field, such as evolutionary RL, evolutionary resampling, evolutionary classification, evolutionary SVM, and evolutionary ensembles.

Given the wide applicability of ML algorithms in real-life applications, the challenges of traditional ML must be consistently and aggressively addressed by the academic research community, industry, and manufacturers. Until now, the optimization of ML using evolutionary algorithms has mostly been investigated in academic publications. In the future, EML will likely be present across many industries in a number of software packages and will further be integrated into our daily lives. The importance of ML in various applications is constantly growing; thus, we are likely to see cutting-edge cloud-based technologies such as Machine Learning-as-a-Service (MLaaS) where evolutionary optimization can also play a significant role.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, Vol. 22. ACM, 207–216.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. 20th Intl. Conf. on Very Large Data Bases, VLDB*, Vol. 1215. 487–499.

[3] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. 2019. A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand* 49, 2 (2019), 205–228.

[4] Shafiq Alam, Gillian Dobbie, Yun Sing Koh, Patricia Riddle, and Saeed Ur Rehman. 2014. Research on particle swarm optimization based clustering: a systematic review of literature and techniques. *Swarm and Evolutionary Computation* 17 (2014), 1–13.

[5] Wissam A Albukhanajer, Yaochu Jin, and Johann A Briffa. 2017. Classifier ensembles for image identification using multi-objective Pareto features. *Neurocomputing* 238 (2017), 316–327.

[6] Hamid Ali, Waseem Shahzad, and Farrukh Aslam Khan. 2012. Energy-efficient clustering in mobile ad-hoc networks using multi-objective particle swarm optimization. *Applied Soft Computing* 12, 7 (2012), 1913–1928.

[7] Ibrahim Aljarah, Majdi Mafarja, Ali Asghar Heidari, Hossam Faris, and Seyedali Mirjalili. 2019. Clustering analysis using a novel locality-informed grey wolf-inspired clustering approach. *Knowledge and Information Systems* (2019), 1–33.

[8] Abdulaziz Almalaq and Jun Jason Zhang. 2018. Evolutionary deep learning-based energy consumption prediction for buildings. *IEEE Access* 7 (2018), 1520–1531.

[9] Mehrdad Almasi and Mohammad Saniee Abadeh. 2015. Rare-PEARs: A new multi objective evolutionary algorithm to mine rare and non-redundant quantitative association rules. *Knowledge-Based Systems* 89 (2015), 366–384.

[10] Akram AlSukker, Rami Khushaba, and Ahmed Al-Ani. 2010. Optimizing the k-nn metric weights using differential evolution. In *2010 International Conference on Multimedia Computing and Information Technology (MCIT)*. IEEE, 89–92.

[11] Amazon. 2017. Amazon EC2 P3 Instances. https://aws.amazon.com/es/ec2/instance-types/p3. Last queried on 14 Nov 2017.

[12] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866* (2017).

[13] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* 34, 6 (2017), 26–38.

[14] Ilhan Aydin, Mehmet Karakose, and Erhan Akin. 2011. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Applied Soft Computing* 11, 1 (2011), 120–129.

[15] Bodrunnessa Badhon, Mir Md Jahangir Kabir, Shuxiang Xu, and Monika Kabir. 2019. A survey on association rule mining based on evolutionary algorithms. *International Journal of Computers and Applications* (2019), 1–11.

[16] Alejandro Baldominos, Yago Saez, and Pedro Isasi. 2020. On the automated, evolutionary design of neural networks: past, present, and future. *Neural Computing and Applications* (2020), 1–27.

[17] Rodrigo Coelho Barros, Márcio Porto Basgalupp, Andre CPLF De Carvalho, and Alex A Freitas. 2011. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 3 (2011), 291–312.

[18] James C Bezdek, Srinivas Boggavarapu, Lawrence O Hall, and Amine Bensaid. 1994. Genetic algorithm guided clustering. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. IEEE, 34–39.

[19] Urvesh Bhowan, Mark Johnston, and Mengjie Zhang. 2011. Developing new fitness functions in genetic programming for classification with unbalanced data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42, 2 (2011), 406–421.

[20] Cosimo Birtolo, Diego De Chiara, Simona Losito, Pierluigi Ritrovato, and Mario Veniero. 2013. Searching optimal product bundles by means of GA-based Engine and Market Basket Analysis. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. IEEE, 448–453.

[21] Christopher M Bishop. 2006. *Pattern recognition and machine learning*. Springer.

[22] Nimagna Biswas, Saurajit Chakraborty, Sankha Subhra Mullick, and Swagatam Das. 2018. A parameter independent fuzzy weighted k-nearest neighbor classifier. *Pattern Recognition Letters* 101 (2018), 80–87.

[23] Veronica Bolon-Canedo, Noelia Sanchez-Marono, and Amparo Alonso-Betanzos. 2011. Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset. *Expert Systems with Applications* 38, 5 (2011), 5947–5957.

[24] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140.

[25] Lam Thu Bui, Thi Thu Huong Dinh, et al. 2018. A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates. *Data & Knowledge Engineering* 114 (2018), 40–66.

[26] Andrés Camero, Jamal Toutouh, Daniel H Stolfi, and Enrique Alba. 2018. Evolutionary deep learning for car park occupancy prediction in smart cities. In *International Conference on Learning and Intelligent Optimization*. Springer, 386–401.

[27] José Ramón Cano, Francisco Herrera, and Manuel Lozano. 2005. Stratification for scaling up evolutionary prototype selection. *Pattern Recognition Letters* 26, 7 (2005), 953–963.

[28] Erick Cantú-Paz and Chandrika Kamath. 2000. Combining evolutionary algorithms with oblique decision trees to detect bent-double galaxies. In *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation III*, Vol. 4120. International Society for Optics and Photonics, 63–71.

[29] PA Castillo, Maribel García Arenas, Juan J Merelo, VM Rivas, and Gustavo Romero. 2006. Multiobjective optimization of ensembles of multilayer perceptrons for pattern classification. In *Parallel Problem Solving from Nature-PPSN IX*. Springer, 453–462.

[30] Kingshuk Chakravarty, Diptesh Das, Aniruddha Sinha, and Amit Konar. 2013. Feature selection by differential evolution algorithm-a case study in personnel identification. In *2013 IEEE Congress on Evolutionary Computation*. IEEE, 892–899.

[31] Yusi Cheng, Qiming Li, et al. 2015. GA-based multi-level association rule mining approach for defect analysis in the construction industry. *Automation in Construction* 51 (2015), 78–91.

[32] Brian Cheung and Carl Sable. 2011. Hybrid evolution of convolutional networks. In *2011 10th International Conference on Machine Learning and Applications and Workshops*, Vol. 1. IEEE, 293–297.

[33] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27.

[34] Giuseppe Cuccu, Matthew Luciw, Jürgen Schmidhuber, and Faustino Gomez. 2011. Intrinsically motivated neuroevolution for vision-based reinforcement learning. In *2011 IEEE International Conference on Development and Learning (ICDL)*, Vol. 2. IEEE, 1–7.

[35] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.

[36] David M Curry and Cihan H Dagli. 2014. Computational complexity measures for many-objective optimization problems. *Procedia Computer Science* 36 (2014), 185–191.

[37] SéRgio Francisco Da Silva, Marcela Xavier Ribeiro, João do ES Batista Neto, Caetano Traina-Jr, and Agma JM Traina. 2011. Improving the ranking quality of medical image retrieval using a genetic feature selection method. *Decision Support Systems* 51, 4 (2011), 810–820.

[38] Ashraf Darwish, Aboul Ella Hassanien, and Swagatam Das. 2020. A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review* 53, 3 (2020), 1767–1812.

[39] Swagatam Das and Amit Konar. 2009. Automatic image pixel clustering with an improved differential evolution. *Applied Soft Computing* 9, 1 (2009), 226–236.

[40] Emiro De la Hoz, Eduardo De La Hoz, Andrés Ortiz, Julio Ortega, and Antonio Martínez-Álvarez. 2014. Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps. *Knowledge-Based Systems* 71 (2014), 322–338.

[41] Jeff Dean and U Hölzle. 2017. Build and train machine learning models on our new Google Cloud TPUs. https://www.blog.google/topics/google-cloud/google-cloud-offer-tpus-machine-learning

[42] Hongbin Dong, Yuxin Dong, Cheng Zhou, Guisheng Yin, and Wei Hou. 2009. A fuzzy clustering algorithm based on evolutionary programming. *Expert Systems with Applications* 36, 9 (2009), 11792–11800.

[43] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *J. Mach. Learn. Res.* 20, 55 (2019), 1–21.

[44] Zhiwei Fu, Bruce L Golden, Shreevardhan Lele, S Raghavan, and Edward A Wasil. 2003. A genetic algorithm-based approach for building accurate decision trees. *INFORMS Journal on Computing* 15, 1 (2003), 3–22.

[45] KY Fung, CK Kwong, Kin Wai Michael Siu, and KM Yu. 2012. A multi-objective genetic algorithm approach to rule mining for affective product design. *Expert Systems with Applications* 39, 8 (2012), 7411–7419.

[46] Mikel Galar, Alberto Fernández, Edurne Barrenechea, and Francisco Herrera. 2013. EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition* 46, 12 (2013), 3460–3471.

[47] Salvador Garcı, Isaac Triguero, Cristobal J Carmona, Francisco Herrera, et al. 2012. Evolutionary-based selection of generalized instances for imbalanced classification. *Knowledge-Based Systems* 25, 1 (2012), 3–12.

[48] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.

[49] Salvador García, Alberto Fernández, and Francisco Herrera. 2009. Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. *Applied Soft Computing* 9, 4 (2009), 1304–1314.

[50] Salvador García and Francisco Herrera. 2009. Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation* 17, 3 (2009), 275–306.

[51] Fred Glover. 1986. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research* 13, 5 (1986), 533–549.

[52] Taciana AF Gomes, Ricardo BC Prudêncio, Carlos Soares, André LD Rossi, and André Carvalho. 2012. Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing* 75, 1 (2012), 3–13.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT press.

[54] Frédéric Gruau. 1994. Automatic definition of modular neural networks. *Adaptive Behavior* 3, 2 (1994), 151–183.

[55] Shenkai Gu and Yaochu Jin. 2014. Generating diverse and accurate classifier ensembles using multi-objective optimization. In *2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*. IEEE, 9–15.

[56] XC Guo, JH Yang, CG Wu, CY Wang, and YC Liang. 2008. A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. *Neurocomputing* 71, 16-18 (2008), 3211–3215.

[57] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *ACM sigmod record*, Vol. 29. ACM, 1–12.

[58] Emrah Hancer, Bing Xue, Dervis Karaboga, and Mengjie Zhang. 2015. A binary ABC algorithm based on advanced similarity scheme for feature selection. *Applied Soft Computing* 36 (2015), 334–348.

[59] Julia Handl and Bernd Meyer. 2002. Improved ant-based clustering and sorting in a document retrieval interface. In *International Conference on Parallel Problem Solving from Nature*. Springer, 913–923.

[60] Shigeru Haruyama and Qiangfu Zhao. 2002. Designing smaller decision trees using multiple objective optimization based gps. In *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 6. IEEE, 5–pp.

[61] Mohamed Jafar Abul Hasan and Sivakumar Ramakrishnan. 2011. A survey: hybrid evolutionary algorithms for cluster analysis. *Artificial Intelligence Review* 36, 3 (2011), 179–204.

[62] Majeed Heydari and Amir Yousefli. 2017. A new optimization model for market basket analysis with allocation considerations: A genetic algorithm solution approach. *Management & Marketing. Challenges for the Knowledge Society* 12, 1 (2017), 1–11.

[63] John H. Holland. 1992. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT Press.

[64] Eduardo Raul Hruschka, Ricardo JGB Campello, Alex A Freitas, et al. 2009. A survey of evolutionary algorithms for clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 39, 2 (2009), 133–155.

[65] Jian Huang, Xiaoguang Hu, and Fan Yang. 2011. Support vector machine with genetic algorithm for machinery fault diagnosis of high voltage circuit breaker. *Measurement* 44, 6 (2011), 1018–1027.

[66] Huimin Jiang, CK Kwong, WY Park, and KM Yu. 2018. A multi-objective PSO approach of mining association rules for affective design based on online customer reviews. *Journal of Engineering Design* 29, 7 (2018), 381–403.

[67] Hua Jiang, Shenghe Yi, Jing Li, Fengqin Yang, and Xin Hu. 2010. Ant clustering algorithm with K-harmonic means clustering. *Expert Systems with Applications* 37, 12 (2010), 8679–8684.

[68] Raja Jothi, Elena Zotenko, Asba Tasneem, and Teresa M Przytycka. 2006. COCO-CL: hierarchical clustering of homology relations based on evolutionary correlations. *Bioinformatics* 22, 7 (2006), 779–788.

[69] Krzysztof Jurczuk, Marcin Czajkowski, and Marek Kretowski. 2017. Evolutionary induction of a decision tree for large-scale data: a GPU-based approach. *Soft Computing* 21, 24 (2017), 7363–7379.

[70] Dervis Karaboga, Bahriye Akay, and Celal Ozturk. 2007. Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *International Conference on Modeling Decisions for Artificial Intelligence*. Springer, 318–329.

[71] Stephen Kelly, Wolfgang Banzhaf, and Cedric Gondro. 2021. Evolving Hierarchical Memory-Prediction Machines in Multi-Task Reinforcement Learning. *Genetic Programming and Evolvable Machines* (2021).

[72] Stephen Kelly and Malcolm I Heywood. 2017. Emergent tangled graph representations for Atari game playing agents. In *European Conference on Genetic Programming*. Springer, 64–79.

[73] Stephen Kelly and Malcolm I Heywood. 2018. Emergent solutions to high-dimensional multitask reinforcement learning. *Evolutionary Computation* 26, 3 (2018), 347–380.

[74] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, Vol. 4. IEEE, 1942–1948.

[75] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. In *Advances in Neural Information Processing Systems*. 1188–1200.

[76] Mujahid H Khalifa, Marwa Ammar, Wael Ouarda, and Adel M Alimi. 2017. Particle swarm optimization for deep learning of convolution neural network. In *2017 Sudan Conference on Computer Science and Information Technology (SCCSIT)*. IEEE, 1–5.

[77] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. 2017. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems* 29, 8 (2017), 3573–3587.

[78] Taghi M Khoshgoftaar and Yi Liu. 2007. A multi-objective software quality classification model using genetic programming. *IEEE Transactions on Reliability* 56, 2 (2007), 237–245.

[79] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680.

[80] Hiroaki Kitano. 1990. Empirical Studies on the Speed of Convergence of Neural Network Training Using Genetic Algorithms.. In *AAAI*. 789–795.

[81] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[82] Jan Koutník, Jürgen Schmidhuber, and Faustino Gomez. 2014. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. 541–548.

[83] Bartosz Krawczyk, Michał Woźniak, and Gerald Schaefer. 2014. Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing* 14 (2014), 554–562.

[84] D Praveen Kumar, Tarachand Amgoth, and Chandra Sekhara Rao Annavarapu. 2019. Machine learning algorithms for wireless sensor networks: A survey. *Information Fusion* 49 (2019), 1–25.

[85] Pardeep Kumar and Amit Kumar Singh. 2019. Efficient generation of association rules from numeric data using genetic algorithm for smart cities. In *Security in Smart Cities: Models, Applications, and Challenges*. Springer, 323–343.

[86] Chan-Sheng Kuo, Tzung-Pei Hong, and Chuen-Lung Chen. 2007. Applying genetic programming technique in classification trees. *Soft Computing* 11, 12 (2007), 1165–1172.

[87] RJ Kuo and LM Lin. 2010. Application of a hybrid of genetic algorithm and particle swarm optimization algorithm for order clustering. *Decision Support Systems* 49, 4 (2010), 451–462.

[88] RJ Kuo, SY Lin, and CW Shih. 2007. Mining association rules through integration of clustering analysis and ant colony system for health insurance database in Taiwan. *Expert Systems with Applications* 33, 3 (2007), 794–808.

[89] RJ Kuo, YJ Syu, Zhen-Yao Chen, and Fang-Chih Tien. 2012. Integration of particle swarm optimization and genetic algorithm for dynamic clustering. *Information Sciences* 195 (2012), 124–140.

[90] Halina Kwaśnicka and Kajetan Świtalski. 2006. Discovery of association rules from medical data-classical and evolutionary approaches. *Annales Universitatis Mariae Curie-Sklodowska, Sectio AI–Informatica* 4, 1 (2006), 204–217.

[91] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*. Springer, 9–48.

[92] CKH Lee, King Lun Choy, George TS Ho, and Cathy HY Lam. 2016. A slippery genetic algorithm-based process mining system for achieving better quality assurance in the garment industry. *Expert Systems with Applications* 46 (2016), 236–248.

[93] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. 2015. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys (CSUR)* 48, 1 (2015), 1–35.

[94] Juan Li, Yuan-xiang Li, Sha-sha Tian, and Jie-lin Xia. 2019. An improved cuckoo search algorithm with self-adaptive knowledge learning. *Neural Computing and Applications* (2019), 1–31.

[95] Juan Li, Yuan-xiang Li, Sha-sha Tian, and Jie Zou. 2019. Dynamic cuckoo search algorithm based on Taguchi opposition-based search. *International Journal of Bio-Inspired Computation* 13, 1 (2019), 59–69.

[96] Juan Li, Dan-dan Xiao, Hong Lei, Ting Zhang, and Tian Tian. 2020. Using cuckoo search algorithm with q-learning and genetic operation to solve the problem of logistics distribution center location. *Mathematics* 8, 2 (2020), 149.

[97] Juan Li, Dan-dan Xiao, Ting Zhang, Chun Liu, Yuan-xiang Li, and Gai-ge Wang. 2021. Multi-swarm cuckoo search algorithm with q-learning model. *Comput. J.* 64, 1 (2021), 108–131.

[98] Juan Li, Yuan-Hua Yang, Hong Lei, and Gai-Ge Wang. 2020. Solving Logistics Distribution Center Location with Improved Cuckoo Search Algorithm. *International Journal of Computational Intelligence Systems* 14, 1 (2020), 676–692.

[99] Wei Li and Gai-Ge Wang. 2021. Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization. *Engineering with Computers* (2021), 1–29.

[100] Wei Li, Gai-Ge Wang, and Amir H Alavi. 2020. Learning-based elephant herding optimization algorithm for solving numerical optimization problems. *Knowledge-Based Systems* 195 (2020), 105675.

[101] Xianneng Li, Shingo Mabu, Huiyu Zhou, Kaoru Shimada, and Kotaro Hirasawa. 2010. Genetic network programming with estimation of distribution algorithms for class association rule mining in traffic prediction. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

[102] Xiangju Li, Hong Zhao, and William Zhu. 2015. A cost sensitive decision tree algorithm with two adaptive mechanisms. *Knowledge-Based Systems* 88 (2015), 24–33.

[103] Jason Liang, Elliot Meyerson, Babak Hodjat, Dan Fink, Karl Mutch, and Risto Miikkulainen. 2019. Evolutionary neural automl for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 401–409.

[104] Amy HL Lim, Chien-Sing Lee, and Murali Raman. 2012. Hybrid genetic algorithm and association rules for mining workflow best practices. *Expert Systems with Applications* 39, 12 (2012), 10544–10551.

[105] Pin Lim, Chi Keong Goh, and Kay Chen Tan. 2016. Evolutionary cluster-based synthetic oversampling ensemble (eco-ensemble) for imbalance learning. *IEEE Transactions on Cybernetics* 47, 9 (2016), 2850–2861.

[106] Yongguo Liu, Kefei Chen, Xiaofeng Liao, and Wei Zhang. 2004. A genetic clustering method for intrusion detection. *Pattern Recognition* 37, 5 (2004), 927–942.

[107] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

[108] Ana Carolina Lorena and Andre CPLF De Carvalho. 2008. Evolutionary tuning of SVM parameter values in multiclass problems. *Neurocomputing* 71, 16-18 (2008), 3326–3334.

[109] José Antonio Lozano and Pedro Larranaga. 1999. Applying genetic algorithms to search for the best hierarchical clustering of a dataset. *Pattern Recognition Letters* 20, 9 (1999), 911–918.

[110] Nannan Lu, Shingo Mabu, Tuo Wang, and Kotaro Hirasawa. 2013. An efficient class association rule-pruning method for unified intrusion detection system using genetic algorithm. *IEEJ Transactions on Electrical and Electronic Engineering* 8, 2 (2013), 164–172.

[111] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2020. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*. Springer, 35–51.

[112] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. NSGA-Net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 419–427.

[113] José María Luna, Cristóbal Romero, José Raúl Romero, and Sebastián Ventura. 2015. An evolutionary algorithm for the discovery of rare class association rules in learning management systems. *Applied Intelligence* 42, 3 (2015), 501–513.

[114] Patrick CH Ma, Keith CC Chan, Xin Yao, and David KY Chiu. 2006. An evolutionary clustering algorithm for gene expression microarray data analysis. *IEEE Transactions on Evolutionary Computation* 10, 3 (2006), 296–314.

[115] Sai Ma and Fulei Chu. 2019. Ensemble deep learning-based fault diagnosis of rotor bearing systems. *Computers in Industry* 105 (2019), 143–152.

[116] Shingo Mabu, Ci Chen, Nannan Lu, Kaoru Shimada, and Kotaro Hirasawa. 2010. An intrusion-detection model based on fuzzy class-association-rule mining using genetic network programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41, 1 (2010), 130–139.

[117] Arif Jamal Malik and Farrukh Aslam Khan. 2018. A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection. *Cluster Computing* 21, 1 (2018), 667–680.

[118] Veenu Mangat and Renu Vig. 2014. Novel associative classifier based on dynamic adaptive PSO: Application to determining candidates for thoracic surgery. *Expert Systems with Applications* 41, 18 (2014), 8234–8244.

[119] Vittorio Maniezzo, Luca Maria Gambardella, and Fabio De Luigi. 2004. Ant colony optimization. In *New Optimization Techniques in Engineering*. Springer, 101–121.

[120] María Martínez-Ballesteros, Francisco Martínez-Álvarez, A Troncoso, and José C Riquelme. 2009. Quantitative association rules applied to climatological time series forecasting. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 284–291.

[121] María Martínez-Ballesteros, Francisco Martínez-Álvarez, Alicia Troncoso, and José C Riquelme. 2011. An evolutionary algorithm to discover quantitative association rules in multidimensional time series. *Soft Computing* 15, 10 (2011), 2065.

[122] María Martínez-Ballesteros, A Troncoso, Francisco Martínez-Álvarez, and José C Riquelme. 2010. Mining quantitative association rules based on evolutionary computation and its application to atmospheric pollution. *Integrated Computer-Aided Engineering* 17, 3 (2010), 227–242.

[123] Jacinto Mata, José-Luis Alvarez, and José-Cristobal Riquelme. 2002. Discovering numeric association rules via evolutionary algorithm. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 40–51.

[124] Daniel Mateos-García, Jorge García-Gutiérrez, and José C Riquelme-Santos. 2016. An evolutionary voting for k-nearest neighbours. *Expert Systems with Applications* 43 (2016), 9–14.

[125] Ron Meir and Gunnar Rätsch. 2003. An introduction to boosting and leveraging. In *Advanced Lectures on Machine Learning*. Springer, 118–183.

[126] Jan Hendrik Metzen, Mark Edgington, Yohannes Kassahun, and Frank Kirchner. 2008. Analysis of an evolutionary reinforcement learning method in a multiagent domain. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Citeseer,

291–298.
[127] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Daniel Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 293–312.
[128] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.
[129] Anirban Mukhopadhyay, Ujjwal Maulik, and Sanghamitra Bandyopadhyay. 2015. A survey of multiobjective evolutionary clustering. *ACM Computing Surveys (CSUR)* 47, 4 (2015), 1–46.
[130] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos Artemio Coello Coello. 2013. A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation* 18, 1 (2013), 4–19.
[131] Anirban Mukhopadhyay, Ujjwal Maulik, Sanghamitra Bandyopadhyay, and Carlos A Coello Coello. 2013. Survey of multiobjective evolutionary algorithms for data mining: Part II. *IEEE Transactions on Evolutionary Computation* 18, 1 (2013), 20–35.
[132] SR Nanda, Biswajit Mahanty, and MK Tiwari. 2010. Clustering Indian stock market data for portfolio management. *Expert Systems with Applications* 37, 12 (2010), 8793–8798.
[133] Satyasai Jagannath Nanda and Ganapati Panda. 2014. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation* 16 (2014), 1–18.
[134] NVIDIA. 2017. The world's most efficient supercomputer for AI and deep learning. http://images.nvidia.com/content/pdf/infographic/dgx-saturnv-infographic.pdf. Last visited on 15 July 2017.
[135] Luiz S Oliveira, Robert Sabourin, Flávio Bortolozzi, and Ching Y Suen. 2002. Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In *Object recognition supported by user interaction for service robots*, Vol. 1. IEEE, 568–571.
[136] Aytuğ Onan, Serdar Korukoğlu, and Hasan Bulut. 2017. A hybrid ensemble pruning approach based on consensus clustering and multi-objective evolutionary algorithm for sentiment classification. *Information Processing & Management* 53, 4 (2017), 814–833.
[137] Fernando EB Otero, Alex A Freitas, and Colin G Johnson. 2012. Inducing decision trees with an ant colony optimization algorithm. *Applied Soft Computing* 12, 11 (2012), 3615–3626.
[138] Rafael S Parpinelli and Heitor S Lopes. 2011. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation* 3, 1 (2011), 1–16.
[139] Russel Pears and Yun Sing Koh. 2011. Weighted association rule mining using particle swarm optimization. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 327–338.
[140] Abdolrahman Peimankar, Stephen John Weddell, Thahirah Jalal, and Andrew Craig Lapthorn. 2018. Multi-objective ensemble forecasting with an application to power transformers. *Applied Soft Computing* 68 (2018), 233–248.
[141] Lu Peng, Shan Liu, Rui Liu, and Lin Wang. 2018. Effective long short-term memory with differential evolution algorithm for electricity price prediction. *Energy* 162 (2018), 1301–1314.
[142] Pyari Mohan Pradhan and Ganapati Panda. 2012. Connectivity constrained wireless sensor deployment using multiobjective evolutionary algorithms and fuzzy decision making. *Ad Hoc Networks* 10, 6 (2012), 1134–1145.
[143] Esmat Rashedi, Elaheh Rashedi, and Hossein Nezamabadi-pour. 2018. A comprehensive survey on gravitational search algorithm. *Swarm and Evolutionary Computation* 41 (2018), 141–158.
[144] Esteban Real, Chen Liang, David So, and Quoc Le. 2020. AutoML-zero: evolving machine learning algorithms from scratch. In *International Conference on Machine Learning*. PMLR, 8007–8019.
[145] Victor Henrique Alves Ribeiro and Gilberto Reynoso-Meza. 2019. A holistic multi-objective optimization design procedure for ensemble member generation and selection. *Applied Soft Computing* 83 (2019), 105664.
[146] Victor Henrique Alves Ribeiro and Gilberto Reynoso-Meza. 2020. Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. *Expert Systems with Applications* 147 (2020), 113232.
[147] Cristóbal Romero, Amelia Zafra, Jose María Luna, and Sebastián Ventura. 2013. Association rule mining using genetic programming to provide feedback to instructors from multiple-choice quiz data. *Expert Systems* 30, 2 (2013), 162–172.
[148] Hussein Samma, Chee Peng Lim, and Junita Mohamad Saleh. 2016. A new reinforcement learning-based memetic particle swarm optimizer. *Applied Soft Computing* 43 (2016), 276–297.
[149] Manish Sarkar, B Yegnanarayana, and Deepak Khemani. 1997. A clustering algorithm using an evolutionary programming-based approach. *Pattern Recognition Letters* 18, 10 (1997), 975–986.
[150] Mansour Sheikhan and Maryam Sharifi Rad. 2013. Using particle swarm optimization in fuzzy association rules-based feature selection and fuzzy ARTMAP-based attack recognition. *Security and Communication Networks* 6, 7 (2013), 797–811.
[151] Christopher Smith and Yaochu Jin. 2014. Evolutionary multi-objective generation of recurrent neural network ensembles for time series prediction. *Neurocomputing* 143 (2014), 302–311.
[152] Qin Song, Yu-Jun Zheng, Yu Xue, Wei-Guo Sheng, and Mei-Rong Zhao. 2017. An evolutionary deep neural network for predicting morbidity of gastrointestinal infections by food contamination. *Neurocomputing* 226 (2017), 16–22.
[153] Pedro Sousa, Paulo Cortez, Rui Vaz, Miguel Rocha, and Miguel Rio. 2013. Email spam detection: A symbiotic feature selection approach fostered by evolutionary computation. *International Journal of Information Technology & Decision Making* 12, 04 (2013), 863–884.

[154] Andreas Stafylopatis and Konstantinos Blekas. 1998. Autonomous vehicle navigation using evolutionary reinforcement learning. *European Journal of Operational Research* 108, 2 (1998), 306–318.

[155] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (2019), 24–35.

[156] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial Life* 15, 2 (2009), 185–212.

[157] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.

[158] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2017. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567* (2017).

[159] Thorsten Suttorp and Christian Igel. 2006. Multi-objective optimization of support vector machines. In *Multi-objective machine learning*. Springer, 199–220.

[160] Amirhessam Tahmassebi and Amir H Gandomi. 2018. Building energy consumption forecast using multi-objective genetic programming. *Measurement* 118 (2018), 164–171.

[161] Amirhessam Tahmassebi and Amir H Gandomi. 2018. Genetic programming based on error decomposition: A big data approach. In *Genetic Programming Theory and Practice XV*. Springer, 135–147.

[162] Amirhessam Tahmassebi, Amir H Gandomi, and Anke Meyer-Baese. 2018. An evolutionary online framework for mooc performance using eeg data. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[163] Amirhessam Tahmassebi, Amir H Gandomi, and Anke Meyer-Baese. 2018. A Pareto front based evolutionary model for airfoil self-noise prediction. In *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[164] Amirhessam Tahmassebi, Behshad Mohebali, Anke Meyer-Baese, and Amir H Gandomi. 2020. Multiobjective genetic programming for reinforced concrete beam modeling. *Applied AI Letters* 1, 1 (2020), e9.

[165] Amirhessam Tahmassebi and Trace Smith. 2021. *SlickML: Slick Machine Learning in Python.* https://github.com/slickml/slick-ml

[166] Pınar Tapkan, Lale Özbakır, Sinem Kulluk, and Adil Baykasoğlu. 2016. A cost-sensitive classification algorithm: BEE-Miner. *Knowledge-Based Systems* 95 (2016), 99–113.

[167] Kshitij Tayal and Vadlamani Ravi. 2016. Particle swarm optimization trained class association rule mining: Application to phishing detection. In *Proceedings of the International Conference on Informatics and Analytics*. 1–8.

[168] Akbar Telikani and Amir H Gandomi. 2019. Cost-sensitive stacked auto-encoders for intrusion detection in the Internet of Things. *Internet of Things* (2019), 100122.

[169] Akbar Telikani, Amir H Gandomi, and Asadollah Shahbahrami. 2020. A survey of evolutionary computation for association rule mining. *Information Sciences* (2020).

[170] Akbar Telikani and Asadollah Shahbahrami. 2018. Data sanitization in association rule mining: An analytical review. *Expert Systems with Applications* 96 (2018), 406–426.

[171] Cuong To and Tuan D Pham. 2009. Analysis of cardiac imaging data using decision tree based parallel genetic programming. In *2009 Proceedings of 6th International Symposium on Image and Signal Processing and Analysis*. IEEE, 317–320.

[172] Binh Tran, Bing Xue, and Mengjie Zhang. 2016. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing* 8, 1 (2016), 3–15.

[173] Isaac Triguero, Mikel Galar, Humberto Bustince, and Francisco Herrera. 2017. A first attempt on global evolutionary undersampling for imbalanced big data. In *2017 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2054–2061.

[174] Isaac Triguero, Salvador García, and Francisco Herrera. 2011. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognition* 44, 4 (2011), 901–916.

[175] Alan M Turing. 1950. Computing machinery and intelligence. *Mind* 59, 236 (1950), 433–460.

[176] Shweta Tyagi and Kamal K Bharadwaj. 2013. Enhancing collaborative filtering recommendations by utilizing multi-objective particle swarm optimization embedded association rule mining. *Swarm and Evolutionary Computation* 13 (2013), 1–12.

[177] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. 2020. A Survey on Distributed Machine Learning. *ACM Computing Surveys (CSUR)* 53, 2 (2020), 1–33.

[178] Leandro D Vignolo, Diego H Milone, and Jacob Scharcanski. 2013. Feature selection for face recognition based on multi-objective evolutionary wrappers. *Expert Systems with Applications* 40, 13 (2013), 5077–5084.

[179] Wengdong Wang and Susan M Bridges. 2000. Genetic algorithm optimization of membership functions for mining fuzzy association rules. *Department of Computer Science Mississippi State University* 2 (2000).

[180] Feng Wen, Guo Zhang, Lingfeng Sun, Xingqiao Wang, and Xiaowei Xu. 2019. A hybrid temporal association rules mining method for traffic congestion prediction. *Computers & Industrial Engineering* 130 (2019), 779–787.

[181] Chun-Hui Wu, Ta-Cheng Chen, Yi-Chih Hsieh, and Huei-Ling Tsao. 2019. A hybrid rule mining approach for cardiovascular disease detection in traditional Chinese medicine. *Journal of Intelligent & Fuzzy Systems* 36, 2 (2019), 861–870.

[182] Bing Xue, Mengjie Zhang, and Will N Browne. 2014. Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. *Applied Soft Computing* 18 (2014), 261–276.

[183] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. 2015. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2015), 606–626.

[184] Dongdong Yang, Licheng Jiao, Maoguo Gong, and Fang Liu. 2011. Artificial immune multi-objective SAR image segmentation with fused complementary features. *Information Sciences* 181, 13 (2011), 2797–2812.

[185] Xin Yao. 1993. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems* 8, 4 (1993), 539–567.

[186] Jianbo Yu, Lifeng Xi, and Shijin Wang. 2007. An improved particle swarm optimization for evolving feedforward artificial neural networks. *Neural Processing Letters* 26, 3 (2007), 217–231.

[187] Mohammed Javeed Zaki. 2000. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering* 12, 3 (2000), 372–390.

[188] Maider Zamalloa, Germán Bordel, Luis Javier Rodríguez, and Mikel Peñagarikano. 2006. Feature selection based on genetic algorithms for speaker recognition. In *2006 IEEE Odyssey-The Speaker and Language Recognition Workshop*. IEEE, 1–8.

[189] Chong Zhang, Kay Chen Tan, Haizhou Li, and Geok Soon Hong. 2018. A cost-sensitive deep belief network for imbalanced classification. *IEEE Transactions on Neural Networks and Learning Systems* 30, 1 (2018), 109–122.

[190] Lei Zhang, Guanglong Fu, Fan Cheng, Jianfeng Qiu, and Yansen Su. 2018. A multi-objective evolutionary approach for mining frequent and high utility itemsets. *Applied Soft Computing* 62 (2018), 974–986.

[191] Zaifang Zhang, Nana Chai, Egon Ostrosi, and Yuliang Shang. 2019. Extraction of association rules in the schematic design of product service system based on Pareto-MODGDFA. *Computers & Industrial Engineering* 129 (2019), 392–403.

[192] Changjiu Zhou. 2002. Robot learning with GA-based fuzzy reinforcement learning agents. *Information Sciences* 145, 1-2 (2002), 45–68.