Estimation of Distribution Algorithms in Machine Learning: A Survey

Pedro Larrañaga, *Senior Member, IEEE*, and Concha Bielza, *Senior Member, IEEE*

***Abstract*—The automatic induction of machine learning models capable of addressing supervised learning, feature selection, clustering, and reinforcement learning problems requires sophisticated intelligent search procedures. These searches are usually performed in the possible model structure spaces, leading to combinatorial optimization problems, and in the parameter spaces, where it is necessary to solve continuous optimization problems. This article reviews how the estimation of distribution algorithms, a kind of evolutionary algorithm, can be used to address these problems. Topics include preprocessing, mining association rules, selecting variables, searching for the optimal supervised learning model (both probabilistic and nonprobabilistic models), finding the best hierarchical, partitional, or probabilistic clustering, obtaining the optimal policy in reinforcement learning, and performing inference and structural learning in Bayesian networks for association discovery. Interesting guidelines for future work in this area are also provided.**

***Index Terms*—Bayesian networks, combinatorial optimization, continuous optimization, estimation of distribution algorithms, evolutionary algorithms, machine learning.**

I. Introduction

CURRENTLY, machine learning is the branch of artificial intelligence that is receiving the most investment and development, at the methodological level and in terms of innovation and application. This is due to the increase in accessibility to databases in diverse domains, such as medicine, energy, industry, and smart cities. In all these domains, the modeling process that must be carried out by the appropriate machine learning paradigm—supervised learning, clustering, or reinforcement learning—usually consists of searching for the model (including its structure and parameters) that best fits the data or yields the best performance. This model search is usually carried out in spaces with large cardinalities, i.e., exponential or superexponential in the number of variables.

Examples of machine learning problems where these large spaces arise are as follows.

1) Feature subset selection, where the number of possible feature subsets $f(n)$, n being the number of variables, is given by [1]: $f(n) = 2^n$.
2) Partitional clustering, where the number of possible partitional clustering assignments $S(N, K)$ of N objects into K groups is given by [2]: $S(N, K) = (1/K!) \sum_{i=0}^{K} (-1)^{K-i} \binom{K}{i} i^N$ for $K \in \mathbb{N}$, with initial conditions $S(0, 0) = 1$ and $S(N, 0) = S(0, N) = 0$.
3) Learning a Bayesian network with n nodes from data in the space of directed acyclic graphs, whose cardinality is given by the following recursive formula [3]: $f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i)$, for $n > 2$, which is initialized with $f(0) = f(1) = 1$.
4) Permutation problems, as in the case of the optimal triangulation of the moral graph associated with a Bayesian network [4] with n nodes, whose cardinality space is $n!$.

In all these examples, searches are performed in discrete spaces and correspond with discrete highly complex optimization problems.

Moreover, finding the best parameters of a machine learning model is usually associated with continuous optimization problems. Examples are the solutions of the likelihood equations in a logistic regression model [5], for which there is no closed analytical expression, or the parameters of an artificial neural network that are usually searched by means of a backpropagation algorithm [6], which often becomes stuck in local optima.

The examples described above belong to the category of NP-hard problems, justifying the use of heuristics in the search for optimal solutions. Classical optimization methods cannot solve those problems on any modern computer within reasonable time. The optimization heuristics that have been used in the literature to search for the best machine learning model and its parameters range from deterministic heuristics, such as the sequential forward (backward) feature selection (elimination), greedy hill climbing, best-first, floating search, tabu search, and branch and bound algorithms, to nondeterministic heuristics with single solutions, such as the simulated annealing, greedy randomized adaptive search procedure, and variable neighborhood search algorithms, and with population-based metaheuristics, such as the scatter search, and evolutionary algorithms (genetic algorithms [7], [8], ant colony optimization [9], particle swarm optimization [10], estimation of distribution algorithms [11], differential evolution [12], evolutionary programming [13], genetic programming [14], and evolution strategies [15]).

Estimation of distribution algorithms evolves individuals in the population in a special manner as opposed to other metaheuristics. This is why these algorithms will be the focus of this survey. Their advantages follow [11]. First, the

Manuscript received 12 May 2023; revised 26 July 2023; accepted 3 September 2023. Date of publication 12 September 2023; date of current version 3 October 2024. This work was supported in part by the Spanish Ministry of Science and Innovation under Project AEI/10.13039/501100011033-PID2022-139977NB-I00 and Project TED2021-131310B-I00, and in part by the ELLIS Unit Madrid by the Autonomous Region of Madrid. *(Corresponding author: Pedro Larrañaga.)*

The authors are with the Departamento de Inteligencia Artificial, Universidad Politécnica de Madrid, 28660 Madrid, Spain (e-mail: pedro.larranaga@fi.upm.es).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2023.3314105.

Digital Object Identifier 10.1109/TEVC.2023.3314105

evolutionary process is stochastic and based on (solid) probability theory, allowing to scape from local optima. Second, it is not necessary to design ad hoc operators for each problem, which is usually required in other evolutionary computation methods. Third, the interactions (in terms of conditional independence) between the variables that encode each individual can be seen explicitly in a probabilistic graphical model learned from the selected population at each generation. This enables the search process interpretation. Fourth, it is possible to incorporate expert knowledge of the optimization problem in that graphical model, e.g., by forcing some must-link (or must-not-link) variables as part of the graph. Fifth, the study of the convergence results of different variants of estimation of distribution algorithms is facilitated by the mathematical ductility of some formulations and probability foundations [16]. The last three advantages are only present in estimation of distribution algorithms, unlike any other metaheuristics.

Evolutionary machine learning refers to the use of evolutionary algorithms for solving machine learning problems. A number of surveys and review papers have been published on this topic and focus on evolutionary algorithms designed for particular machine learning tasks, such as feature subset selection [17], association rule mining [18], [19], classification trees [20], deep learning [21], [22], [23], clustering [24], [25], [26], and association discovery with Bayesian networks (including estimation of distribution algorithms) [27]. Other surveys have covered evolutionary algorithms applied to a number of machine learning tasks [28], [29], [30]. As regards [28], it is a brief overview of evolutionary computation in classification, regression, and clustering. Instead, [29] and [30] are devoted to multiobjective evolutionary algorithms for machine learning.

This article surveys the use of estimation of distribution algorithms in machine learning.

1) Preprocessing tasks, such as the optimal rearrangement of rows and columns of tables and multivariate discretization.
2) Association rule mining.
3) Supervised learning tasks, such as feature subset selection, and k-nearest neighbors modeling, classification trees, rule induction, support vector machines, artificial neural networks, logistic regression, Bayesian classifiers, metaclassifiers, and regression.
4) Clustering methods, such as hierarchical clustering, partitional clustering, and probabilistic clustering.
5) Reinforcement learning.
6) Inference and structure learning in Bayesian networks for association discovery.

Real applications solved by machine learning methods based on estimation of distribution algorithms are also shown. In addition, the use of estimation of distribution algorithms is justified by the interpretability of the probabilistic graphical models learned in each generation of the algorithm, which will also be emphasized. To the best of our knowledge, no survey exists on the estimation of distribution algorithms for any machine learning task. This is the main motivation behind our work, which covers all the above-mentioned tasks.

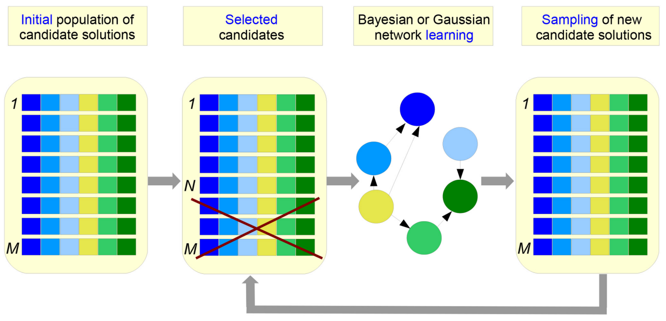
Initial population of candidate solutions
Selected candidates
Bayesian or Gaussian network learning
Sampling of new candidate solutions
1
M
1
N
M
1
M

Fig. 1. Graphical representation of EDAs.

The remainder of this article is organized as follows. Section II provides background information on the estimation of distribution algorithms for discrete and continuous optimization domains with a brief introduction of Bayesian networks. This is necessary for understanding the estimation of distribution algorithms with multivariate dependencies. Estimation of distribution algorithms for multiobjective optimization are also presented. From Sections III–VIII, approaches based on these algorithms in different machine learning tasks are reviewed. In particular, preprocessing, association rule mining, supervised learning, clustering, reinforcement learning, and association discovery with Bayesian networks are discussed in Sections III–VIII, respectively. Estimation of distribution algorithms applied in real machine learning problems are found in Section IX. Finally, Section X presents conclusions and future work.

II. Estimation of Distribution Algorithms

A. Evolutionary Algorithms

Evolutionary algorithms [31] comprise a set of heuristic techniques that aim to solve (combinatorial or continuous) optimization problems by computationally reproducing the principles of natural evolution proposed by Darwin [32]. The search for the optimal solution is carried out by evolving a population of individuals, each of which represents a possible solution to the optimization problem. While genetic algorithms [7], [8] are the best-known examples of evolutionary algorithms, other techniques, such as evolutionary programming [13], evolution strategies [15], genetic programming [14], ant colonies [9], differential evolution [12], and estimation of distribution algorithms [11] have also been developed and used in a large number of real-world applications.

B. General Scheme of Estimation of Distribution Algorithms

In estimation of distribution algorithms (EDAs) [11], [33], [34], [35], [36], [37], a population of candidate solutions is evolved by estimating the probability distribution underlying the individuals selected in each generation according to their fitness (objective function value) and represented as a probabilistic graphical model. Then, this probability distribution is simulated to obtain the new population of candidate individuals (see Fig. 1). For a recent survey, see [38].

Algorithm 1 is a general, unified pseudocode for all variants of EDAs introduced in Section II-D. The initial population of

Algorithm 1: Pseudocode for the EDA

```
$D_0 \leftarrow$ Generate $M$ individuals (initial population) at random
repeat for $l = 1, 2, \ldots$
    1. $D_{l-1}^{Se} \leftarrow$ Select $N \leq M$ individuals from $D_{l-1}$ according to the selection method
    2. $p_l(\mathbf{x}) = p(\mathbf{x}|D_{l-1}^{Se}) \leftarrow$ Estimate the probability distribution of an individual $\mathbf{x}$ being among the selected population
    3. $D_l \leftarrow$ Sample $M$ individuals (new population) from $p_l(\mathbf{x})$
until the stopping criterion is met
```

individuals is randomly drawn by the simulation of a probability distribution defined in the search space. If prior knowledge about the problem is available, it can be used to avoid an uninformative sample distribution, i.e., a uniform distribution.

In the first step, each of the M randomly obtained individuals is evaluated, and a fixed number of them, N, are selected according to a previously established criterion. This criterion can be deterministic (such as selecting the N individuals with the best evaluation functions) or stochastic (incorporating a random selection, where the selection probability for each individual is proportional to its evaluation function).

In the second step, the joint probability distribution (JPD) of the selected individuals is estimated using different assumptions. In the literature on EDAs, three situations are considered: 1) the variables are independent; 2) only the bivariate dependencies are taken into account; and 3) the dependence degree between the variables is not restricted. In the vast majority of optimization scenarios that occur in the real world, assumption 1) is far from reality. On the other hand, assumption 3) can be computationally expensive in problems with a large number of variables.

In the third step, the probability distribution learned in the previous step is simulated to obtain a new population of individuals. In constrained optimization problems, it must be guaranteed that the simulated individuals verify the constraints in the simulation process.

These three steps (evaluation, estimation, and simulation) are repeated until a previously determined stop condition is met. This condition can refer to the number of simulated generations, the convergence of the population of individuals toward the global optimum or even the maximum acceptable execution time (or number of iterations).

In most EDAs that do not restrict the dependence relationships between variables, the JPD is estimated by a Bayesian network (Section II-C) learned from data. EDAs have also been developed with the probability distribution estimated from log-linear probability models [39], probabilistic principal component analysis [40], Kikuchi approximations [41], Markov networks [42], [43], Markov chains [44], copulas and vines [45], a reinforcement learning-based method [46], Gaussian adaptive resonance theory neural networks [47], growing neural gas networks [48], restricted Boltzmann machines [49], [50], [51], and in the deep learning area, from autoencoders [52], variational autoencoders [53], [54], and generative adversarial networks [55]. Model selection in EDAs is a more complex problem. In [56], this problem was addressed based on variable transformations. The authors found a variable transformation technique that implicitly captures higher-order interactions and then uses low-dimensional models in the new transformed space (with easier parameter estimation).

Theoretical issues of EDAs (convergence analysis and runtime analysis) have been primarily addressed in algorithms that assume independence between variables in discrete [57], [58], [59] and continuous domain optimization approaches [60], and limited attempts have been made to study the behavior of EDAs that do not restrict the dependence relationships between variables [61]. See [16] for a survey on this topic. A quantification of the genetic drift effect in EDAs appears in [62].

C. Bayesian Networks

Bayesian Networks for Discrete Variables: For combinatorial optimization problems, EDAs are usually based on Bayesian networks. A Bayesian network [63], [64], [65] is an interpretable compact representation of the JPD $p(X_1, \ldots, X_n)$ over a set of variables $X_1, \ldots, X_n$. Conditional independence between triplets of variables is the central concept in Bayesian networks; it allows the JPD to be represented in a compact manner and with fewer parameters. Two random variables X and Y are conditionally independent given another random variable Z if $p(x|y, z) = p(x|z) \quad \forall x, y, z$ values of X, Y, and Z, that is, whenever $Z = z$, the information $Y = y$ does not influence the probability of x.

Suppose that we find a subset $\mathbf{Pa}(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$ for each X_i such that given $\mathbf{Pa}(X_i)$, X_i is conditionally independent of all variables in $\{X_1, \ldots, X_{i-1}\} \setminus \mathbf{Pa}(X_i)$, i.e.,

$$p(X_i|X_1, \ldots, X_{i-1}) = p(X_i|\mathbf{Pa}(X_i)).$$

Then, the JPD can be factorized as follows:

$$p(X_1, \ldots, X_n) = p(X_1|\mathbf{Pa}(X_1)) \cdots p(X_n|\mathbf{Pa}(X_n))$$

with a (hopefully) substantially smaller number of parameters. This modularity permits easy maintenance and efficient reasoning.

The Bayesian network has two main parts. The qualitative part, by means of a directed acyclic graph (DAG), represents the conditional (in)dependencies between variables. The quantitative part contains the conditional probability tables (CPTs) of each discrete variable X_i given any possible instantiation of its parent variables (variables from which arcs with destination X_i result), $\mathbf{Pa}(X_i)$, in the DAG.

Fig. 2 shows a hypothetical example taken from [66] of a Bayesian network, modeling the risk of dementia. All variables are binary: x denotes "presence" and $\neg x$ denotes "absence," for `Dementia` D, `Neuronal Atrophy` N, `Stroke` S, and `Paralysis` P. For `Age` A, a means "aged 65+"; otherwise the state is $\neg a$. Note that in the Bayesian network structure both `Stroke` and `Neuronal Atrophy` are influenced by `Age` (their parent in the DAG). These two conditions influence

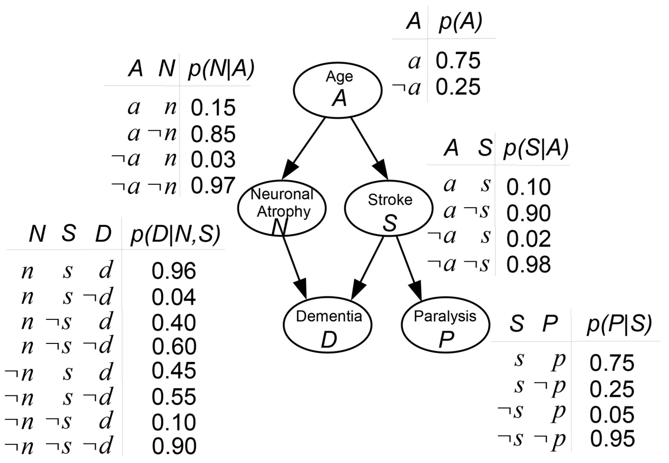

Fig. 2. Example of a Bayesian network model for the hypothetical risk of dementia, taken from [66].

`Dementia` (their child). `Paralysis` is directly associated with having a stroke. The CPTs show the Bayesian network parameters and indicate the specific conditional probabilities attached to each node. For instance, if someone has neuronal atrophy and has had a stroke, there is a probability of 0.96 that the person will have dementia: $p(d|n, s) = 0.96$. However, in the absence of neuronal atrophy and stroke, this probability is only 0.10, i.e., $p(d|\neg n, \neg s) = 0.10$. Thus, the JPD $p(A, N, S, D, P)$ requires $2^5 - 1 = 31$ parameters to be fully specified. However, using this Bayesian network which provides the following factorization: $p(A, N, S, D, P) = p(A)p(N|A)p(S|A)p(D|N, S)p(P|S)$, only 11 input probabilities are needed.

Bayesian Networks for Continuous Variables: For continuous optimization problems, EDAs are based on Gaussian Bayesian networks [67], [68]. In a Gaussian Bayesian network, it is assumed that the associated JPD for $\mathbf{X} = (X_1, \dots, X_n) \in \mathbb{R}^n$ is a multivariate (nonsingular) normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ given by

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

where $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)^T$ is the vector of means, $\boldsymbol{\Sigma}$ is the $n \times n$ covariance matrix and $|\boldsymbol{\Sigma}|$ is its determinant.

The JPD in a Gaussian Bayesian network can be equivalently defined by the product of n univariate (linear) Gaussian conditional densities $f(\mathbf{x}) = f_1(x_1)f_2(x_2|x_1)\cdots f_n(x_n|x_1, \dots, x_{n-1})$, each defined as follows:

$$f_i(x_i|x_1, \dots, x_{i-1}) \sim \mathcal{N}\left(\mu_i + \sum_{j=1}^{i-1} \beta_{ij}(x_j - \mu_j), v_i\right)$$

where μ_i is the unconditional mean of X_i (i.e., the ith component of $\boldsymbol{\mu}$), v_i is the conditional variance of X_i given values for $x_1, \dots, x_{i-1}$ and β_{ij} is the linear regression coefficient of X_j in the regression of X_i on $X_1, \dots, X_{i-1}$.

Learning of (Gaussian) Bayesian Networks: The learning and simulation of Bayesian networks and Gaussian Bayesian networks, which correspond to steps 2 and 3 of Algorithm 1, respectively, are performed by similar methods regardless of whether we are working in discrete or continuous optimization scenarios.

Learning Bayesian networks [69], [70], [71] (and Bayesian Gaussian networks) from data can be achieved using two different approaches. On the one hand, constraint-based methods are used to statistically test conditional independencies among triplets of variables from data. A DAG that represents a large percentage (and whenever possible all) of identified conditional independence constraints is provided as the output of this type of algorithm. The most representative method is the PC algorithm [72]. The PC algorithm, which starts with a complete (all nodes are connected) undirected graph, has two stages. In stage 1, the adjacencies in the graph (the skeleton of the learned structure) are output using edge elimination through hypothesis testing (such as the χ^2 test or the G^2 test). For any pair of adjacent nodes X_i and X_j in the graph and for a subset $\mathbf{Z}$ of the adjacent nodes of X_i, edge X_i–X_j is removed if and only if $\mathbf{Z}$ renders X_i and X_j conditionally independent. This elimination process is carried out recursively, and the number of variables in the conditioning part, $\mathbf{Z}$, of the hypothesis test increases in each step. In stage 2, the orientation of the edges and their transformation into arcs are the focus. Constraint-based methods are very general and easily adaptable for learning Gaussian Bayesian networks. However, very few EDAs have been developed based on these methods. Indeed, each time the number of variables in the conditioning part to carry out the hypothesis tests goes up, the cardinality of the dataset from which to learn the structure of the model increases considerably, greatly slowing down the evolutionary search process.

On the other hand, in score and search-based methods, attempts are made to intelligently search the DAG spaces to maximize a given criterion (a number of heuristics have been applied for this purpose). A large number of criteria (Akaike information criterion [73], Bayesian information criterion (BIC) [74]$\cdots$) are based on penalized likelihood. The penalty is defined by taking the complexity of the evaluated structure (its number of parameters) into account. This is necessary because otherwise the search would end with the complete model (all nodes linked with the rest of the nodes). Other criteria, such as the K2 score [75], or the Bayesian Dirichlet equivalence (BDe) score [76] are associated with marginal likelihood, following a Bayesian perspective. Interestingly, the score should be decomposable, that is, it should be expressed as a sum (or product) of values that depend on only one node and its parent nodes.

Simulation of (Gaussian) Bayesian Networks: The simulation of Bayesian networks (or Gaussian Bayesian networks) is carried out in step 3 of Algorithm 1. While most of the EDAs implemented in real applications have used the simulation method called probabilistic logic sampling [77], other methods, such as likelihood weighting [78], or Gibbs sampling [79] can also be considered. In probabilistic logic sampling we use ancestral node ordering, i.e., we sample a node X_i after sampling from all its parent nodes $\mathbf{Pa}(X_i)$ which results in a fixed value $\mathbf{pa}(x_i)$ (forward sampling scheme). Efficient sampling schemes to promote the visit of promising regions and avoid premature convergence have been recently proposed for Gaussian Bayesian networks [80].

In Fig. 1 and Algorithm 1 the fitness value of each of the simulated individuals in the corresponding (Gaussian) Bayesian network is not considered as another node. This is how the vast majority of EDA algorithms work, i.e., without the evaluation or fitness variable appearing explicitly in the model learned in each generation. Exceptions to this general trend are found in [81] where the model learned in each generation is a Bayesian classifier (a special case of a Bayesian network) whose class variable is defined as the variable containing the evaluation of the individuals, and in [82], where in a many-objective optimization problem each objective to be considered is represented as a variable in the Gaussian Bayesian network.

D. Categorization of EDAs

Discrete EDAs are the name given to EDA-based optimization algorithms designed to solve combinatorial optimization problems. Instead, optimization in continuous domains is addressed by continuous EDAs. Discrete [36] and continuous [83] EDAs are categorized according to the probabilistic dependency relationships allowed for $p_l(\mathbf{x})$ in Algorithm 1. In [84], principles for a proper adaptation of discrete EDAs to the continuous case are presented.

Without Dependencies: The joint n-dimensional probability distribution of the selected individuals in each generation is assumed to factorize as the product of n 1-D probability distributions, one per variable. Namely, $p_l(\mathbf{x}) = \prod_{i=1}^{n} p_l(x_i)$, where p_l would be replaced by densities in continuous EDAs. The main representatives are listed below.

1) The univariate marginal distribution algorithm (UMDA) [85] in discrete domains, the UMDA for continuous domains (UMDA$_c$) [83], and UMDA$_c^G$, where a univariate Gaussian density is assumed for each variable.
2) Population-based incremental learning (PBIL) [86], where the selected individuals at each generation are used in updating the components of the probability vector using a Hebbian-inspired rule, and its continuous version [87], where Gaussianity in each marginal univariate density is assumed.
3) The compact genetic algorithm (cGA) [88], where only two individuals from the current probability distribution are simulated in each generation, and the process of adapting probabilities toward the winning individual continues until convergence.

A general formulation of univariate discrete EDAs that incorporates UMDA, PBIL, and cGA is proposed in [89].

Bivariate Dependencies: The three seminal works within this category are: 1) mutual information maximization for input clustering (MIMIC); 2) combining optimizers with mutual information trees (COMITs); and 3) the bivariate marginal distribution algorithm (BMDA). In MIMIC [90], searches for the best permutation between the variables are performed in each generation to find the probability distribution $p_l^{\pi}(\mathbf{x})$ that is closest to the empirical distribution of the selected individuals when using the Kullback–Leibler divergence, where $p_l^{\pi}(\mathbf{x}) = p_l(x_{i_1}|x_{i_2})p_l(x_{i_2}|x_{i_3})\cdots p_l(x_{i_{n-1}}|x_{i_n})p_l(x_{i_n})$ and $\pi = (i_1, i_2, \ldots, i_n)$ denotes a permutation of the indices $1, 2, \ldots, n$. This means that the structure to be learned is a chain. In MIMIC$_c^G$ [83], the MIMIC algorithm is adapted for continuous optimization problems by assuming Gaussianity for marginal and conditional densities. In COMIT [91], a tree structure Bayesian network is learned using the maximum weighted spanning tree algorithm at each generation. In BMDA [92], the JPD is factorized from an acyclic graph formed by a set of trees, that are not necessarily mutually connected. Each tree is created taking into account the dependencies between pairs of variables that exceed a certain dependency threshold.

Multivariate Dependencies: The vast majority of EDAs that belong to this category are based on learning the Bayesian network that best fits the distribution of the selected individuals in each generation and its subsequent simulation. The pioneering EDAs in this area are listed below.

1) The estimation of Bayesian network algorithm (EBNA) [93], [94], where the use of both types of Bayesian network learning algorithms, constraint-based, and score-and-search-based algorithms, is proposed. Among the scores used, the BIC and the K2 scores are the most notable. In each generation, the search procedure for EBNAs starts with the model induced in the previous generation. The EGNA [83] is similar to EBNA although a Gaussian Bayesian network is learned in each generation. Dong et al. [95] adapted the estimation of Gaussian Bayesian network algorithm (EGNA) approach for handling high-dimensional problems by controlling the complexity of the learned models.
2) The Bayesian optimization algorithm (BOA) [96], which uses the BDe metric to measure the goodness of each structure in combination with a greedy search algorithm that starts from scratch in each generation.
3) The learned factorized distribution algorithm (LFDA) [97], which controls the complexity of the learned Bayesian network through the BIC in conjunction with a restriction on the maximum number of parents that each variable can have.
4) The estimation of multivariate normal algorithm (EMNA) [83], which assumes a Gaussian JPD, whose vector of means and covariance matrix are estimated by the maximum-likelihood method. In [98], an EDA based on the eigen analysis of the covariance matrix and its corresponding tuning strategy is proposed. In [99], an archive with a certain number of high-quality solutions from previous generations is preserved, and the evolution direction provided by the individuals in the archive is integrated into the estimation of the covariance matrix of the Gaussian model.
5) Regularization-based EDAs benefit from likelihood regularization during the Bayesian network structure search in each generation. This allows an initial selection of candidate parents for each variable in the graph [100]. In continuous domains, [101] proposes the

use of regularized model learning of Gaussian Bayesian networks, pursuing sparseness in high-dimensional problems.

6) The iterated density evolutionary algorithm (IDEA) [102] (and its multiobjective version MIDEA [103]) use Gaussian kernel probability density functions, in contrast to mixtures of Gaussians to cope with multimodal optimization problems [104].

7) Finally, the extended compact genetic algorithm (EcGA) [105] does not need to learn a Bayesian network in every generation to obtain an EDA with multivariate dependencies. In EcGA, the JPD is factorized as a product of probability distributions of varying size. Each group of variables is assumed to be independent from the others. Therefore, a factorization such as $p_l(\mathbf{x}) = \prod_{c \in C_l} p_l(\mathbf{x}_c)$, where C_l denotes the set of groups of variables in the lth generation, and $p_l(\mathbf{x}_c)$ represents the marginal (discrete) distribution of variables $\mathbf{X}_c$, namely, the variables belonging to the cth group in the lth generation, is obtained.

E. Multiobjective EDAs

Multiobjective optimization problems involve optimizing more than one goal, i.e., there are m ($m > 1$) functions subject to a set of constraints. Optimal solutions are defined based on the Pareto dominance relation. The use of evolutionary computation to multiobjective optimization problems has led to the so-called multiobjective optimization evolutionary algorithms [106]. Most of these algorithms, EDAs included, simplify the problem by reducing the m-dimensional space to a scalar value with fitness functions like the convergence indicator, the Pareto-optimal front coverage indicator, the hypervolume indicator, and the unary additive ϵ-indicator. This is the strategy followed by EDAs based on neural networks [47], [48], [51], [54], on probabilistic models [82], [103], [107] or on a Parzen estimator [108].

III. EDAs in Preprocessing

Optimal Ordering of Tables: The ordering of rows and columns in a table is a very sensitive issue that affects the readability of the table. Rearranging the rows and columns in a table when their orders are irrelevant reveals interesting patterns that make the table easier to read and interpret. Fig. 3 shows an adaptation of the original table introduced by Bertin [110], where the columns represent townships and the rows are characteristics of the townships that are either present (labeled as 1) or absent (0). The cells labeled 0 are shaded red, and the cells labeled 1 are not shaded. The ordering of the rows and columns in Fig. 3(a)–(c) varies, showing that the examples with a reduced stress value result in more intuitive readability of the information. This stress measure f is an overall dissimilarity measure for the whole table and is computed from the distance of each table cell value to its neighboring values. The literature shows that this problem has been a cause of concern in statistics for a long time [111].

From a technical point of view, the optimal ordering of tables (minimum stress) is equivalent to solving two traveling salesman problems: one for the R rows and the other for the C columns, resulting in a search space with a cardinality given by $R! \cdot C!$. Mathematically, the optimization problem is $\min_{\pi^r(1),\ldots,\pi^r(R),\pi^c(1),\ldots,\pi^c(C)} f(\pi^r, \pi^c)$, where the optimization variables $\pi^r(i)$ and $\pi^c(j)$ denote the position of row i and column j in a given π^r ordering of rows and π^c of columns, respectively.

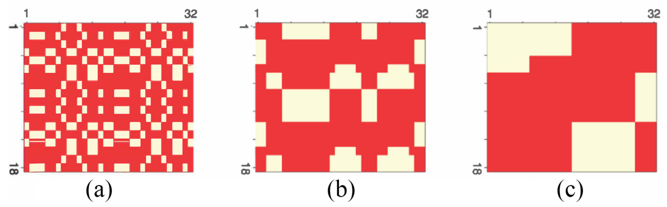

Fig. 3. Different row and column orderings for the same table of dimension 18 × 32, illustrating different levels of readability. High readability is associated with small stress values. Taken from [109]. (a) Stress = 1936. (b) Stress = 848. (e) Stress = 330.

In the EDA approach developed in [109], an individual was defined as $\mathbf{x} = (x_1, \ldots, x_R, x_{R+1}, \ldots, x_{R+C})$, where $x_i = k$ means that the position in the ordering of the original ith row is k, and $x_{R+j} = l$ means that the position for the jth column is l. This double-path individual representation is very intuitive, but it is redundant, because there may be different individuals representing the same solution with the same stress value, which can confuse the search process. An alternative representation was designed in the same paper using continuous EDAs, whose simulated real vectors of values were transformed into permutations as the respective order for the continuous individual. Univariate (UMDA and UMDA$_c^G$), bivariate (MIMIC and MIMIC$_c^G$), and multivariate (EBNA and EGNA) EDAs were used in the experiments.

Multivariate Discretization: A supervised approach to multivariate discretization based on EDAs (UMDA$_c^G$) was presented in [112]. In contrast to many classical approaches, the discretization process is multivariate, that is, all predictor variables are discretized simultaneously, and each discretization is evaluated with a supervised classification method (i.e., it is a wrapper approach). This approach depends on the discretization sequence that contains the cut points for each of the bins (i.e., these cut points are the optimization variables). An individual in the EDA represents a discretization policy that transforms the original dataset into a discretized dataset. The discretized dataset is evaluated by the estimated accuracy (objective function to be maximized) provided by the classifier. k-nearest neighbors, classification trees, and naive Bayes were evaluated for this purpose in the original paper.

IV. EDAs in Association Rule Mining

Association rule mining is used to identify the rules discovered in datasets of transactions with a variety of items by using some measures of interestingness. There are two stages in the popular Apriori algorithm [113]. In the first stage, frequent item sets are found, and in the second stage, association rules based on the frequent item sets found are generated. The first stage is the most computationally demanding and is where EDAs (specifically PBIL) have been used [114]. The EDA adopts a binary code (1 if the item is selected; 0 otherwise) for

TABLE I
SUPERVISED LEARNING METHODS APPROACHED WITH EDAS

k-nearest neighbors	Weight of each variable [117]	EBNA, EGNA
Classification trees	Tree structure [119]	Ardennes
Rule induction	Pittsburgh approach [122]	UMDA, COMIT, EBNA
	Pittsburgh approach [123], [124]	BOA
	Fuzzy rule-based systems [125]	UMDA, MIMIC, UMDA$_c^G$
Support vector machines	Hyperparameters [127]	UMDA$_c^G$, BUMDA
Artificial neural networks	Weights of a multilayer perceptron [129]	PBIL
	Weights of a multilayer perceptron [130]	PBIL
	Weights of a multilayer perceptron [131]	UMDA$_c^G$, MIMIC$_c^G$
	Weights of a multilayer perceptron [132]	UMDA$_c^G$
	Weights and hidden biases of a feedforward neural network [133]	PBIL
	Weights and structure of a multilayer perceptron [134]	PBIL
	Weights and structure of a multilayer perceptron [135]	UMDA$_c^G$
	Pruning the structure of a multilayer perceptron [136]	cGA, EcGA, BOA
	Design of a fully connected multilayer perceptron [137]	UMDA
	Hyperparameters of a convolutional network [139]	UMDA+UMDA$_c^G$
	Weights of a convolutional network [140]	EMNA, UMDA$_c^G$
	Weights in autoencoders [141]	UMDA$_c^G$
Logistic regression	Parameter estimation [143]	UMDA$_c^G$
	Parameter estimation in a regularized setting [144]	UMDA$_c^G$
Bayesian classifiers	Parameter estimation in naive Bayes [148]	UMDA$_c^G$
	Structure of a seminaive Bayes [150]	UMDA
Metaclassifiers	Selection of base classifiers in stacked generalization [152]	EBNA
	Boosting voting weights of each classifier [154]	UMDA$_c^G$
Regression	Symbolic regression [156]	UMDA$_c^G$
	Symbolic regression [157]	UMDA
	Symbolic regression [158]	Denoising autoencoder genetic programming
	Support vector regression [159]	UMDA$_c^G$
Feature subset selection	Selective naive Bayes [161]	EBNA
	Selective naive Bayes [162]	EBNA
	Selective naive Bayes [163]	cGA, EcGA, BOA
	Support vector machines [164]	TEDA
	Logistic regression [165]	UMDA$_c^G$
	Classification trees [166]	UMDA
	Naive Bayes [167]	FEDA

optimization variables, where each individual length is equal to the total number of items. The fitness function finds frequent item sets as those with support level greater than the minimum support level and discards the rest. Its value is used to update the probability vector in PBIL.

If the aim is to predict a special variable, the task is called class association rule mining. In [115], an EDA replaces the genetic operators (crossover and mutation) of conventional genetic network programming (a graph-based evolutionary algorithm) to improve the efficiency for generating valid offspring and deal with dynamic environments. EDAs are of UMDA- and PBIL-type.

V. EDAS IN SUPERVISED LEARNING

We are given a labeled data set of n variables forming vector $\mathbf{X} = (X_1, \ldots, X_n)$, including features from N observations. Let $\mathcal{D} = \{(\mathbf{x}^1, c^1), \ldots, (\mathbf{x}^N, c^N)\}$ denote the data set, where $\mathbf{x}^i = (x_1^i, \ldots, x_n^i)$, $i = 1, \ldots, N$, while c^i indicates its label from a class variable C. For regression, C will be denoted Y, the real-valued response variable.

Table I shows a summary of the papers reviewed in this section.

k-Nearest Neighbors: To classify a query instance $\mathbf{x} = (x_1, \ldots, x_n)$, the k-nearest neighbors method [116] predicts the unknown class label from the classes associated with the k instances of the training set that are closer to $\mathbf{x}$ in the instance space using a simple majority decision rule. The accuracy of this classifier depends heavily on the weight of each variable to compute distances between instances; i.e., the problem is $\max_{w_1,\ldots,w_n}\text{Acc}(\phi)$, where ϕ is the k-nearest neighbors classifier, Acc is the objective function, and the neighbors are determined with a weighted distance: $d(\mathbf{x}, \mathbf{x}^i) = \sum_{j=1}^n w_j\delta(x_j, x_j^i)$, where w_j is the weight assigned to variable X_j, and $\delta(x_j, x_j^i)$ measures the distance between the jth components of $\mathbf{x}$ and $\mathbf{x}^i$. In [117], a search for these weights the optimization variables was performed with an EDA based on the EBNA approach in a discrete domain (with three different weight values) and based on the EGNA approach in a continuous domain.

Classification Trees: Classification trees [118] are expressed as a recursive partition of the instance space. The tree has three kinds of nodes. First, a root node with no incoming edges and several outgoing edges. Second, internal nodes or test nodes, with one incoming edge and several outgoing edges. Third, leaf nodes with one incoming edge and no outgoing edges. In standard algorithms for inducing classification trees, a tree is built in a greedy manner, and a search for the optimal inner leaf is performed at each step. The use of EDAs to this problem allows a more global approach to find the optimal classification tree; i.e., the tree ϕ^* which solves $\max_{\phi\in\mathcal{CT}(n)}\text{Acc}(\phi)$, where $\mathcal{CT}(n)$ is the set of all classification trees built with the n predictor variables in $\mathcal{D}$.

To the best of our knowledge [119] is the only work in which EDAs use the previous fitness function. The authors defined a novel EDA named Ardennes. The individuals are binary trees whose depth is upper bounded by h, a predefined user parameter. The predictor variables (the optimization variables) are selected to fill root and internal nodes according to independent probability distributions associated with these nodes. In the first generation, all the initial probability distributions are uniform. However, in the first generation, the probability of selecting the class attribute C is zero for all variables between the root and depth $h-1$. At depth h, the probability of selecting the class is 1. Note that the probability of selecting the class at intermediate levels may increase during evolution since a homogeneous node (all objects belonging to the same class label) is automatically transformed into class nodes. The sampling of nodes is performed on-demand following a depth-first search: the root node is sampled first, then its left child, the left child of that child, and so on. If the class attribute is sampled, the current node is turned into a leaf, and its label is set according to the most frequent class found in the subset of instances reaching that node. Otherwise, if a predictive variable is sampled, we perform a binary split on the instances reaching that node.

Rule Induction: The induction of "IF-THEN" rule-based classifiers with evolutionary computation has been achieved using two different approaches. In the Michigan approach to classifier systems [120], an individual of the population with a fixed length string is considered, whereas the classifier system itself is represented by the whole set of individuals in the population. In contrast, in the Pittsburgh approach [121], the use of a variable length string is proposed, and each individual in the population is interpreted as a classifier system. The optimization problem is $\max_{\phi \in \mathcal{PCS}} \text{Acc}(\phi)$, where $\mathcal{PCS}$ is the set of all classifier systems built following a Pittsburgh approach.

In [122], an EDA for rule induction that can be seen as an instantiation of the Pittsburgh approach was proposed. The individual representation of the IF part of the rule (the antecedent) consists of the disjunction of simple antecedents (the optimization variables), each with a dimension given by n, allowing each variable to take values that are "equal to," "different from," and "any possible value." Univariate, bivariate, and multivariate EDAs (UMDA, COMIT, and EBNA) were used in the experiments. A 2-phase adaptation for continuous variables in [123] first uses BOA to model and generate single rules that are then assembled together to form the rule sets. In the second phase, rule sets (classifiers based on rules) are recombined with a procedure similar to a classic genetic algorithm. This improves the effectiveness and efficiency of the rule structure exploration. The same authors add an embedded feature reduction approach to remove irrelevant variables gradually following the evolution of rule population [124]. Features are first discarded according to their low frequency in the rule population. Then, more features are removed if they do not belong to the Markov blanket of any other variable, this being found in the Bayesian network learned in the BOA phase. The evolutionary search turns to be more effective.

In fuzzy rule-based systems, rule weights are used to improve their predictive capability. In [125], EDAs (UMDA, MIMIC, and UMDA$_c^G$) simultaneously evolve the rules and their weights (both are the optimization variables) and incorporate domain knowledge.

Support Vector Machines: Support vector machines [126] apply a simple linear method to data, albeit in a high-dimensional feature space that is nonlinearly related to the original input space. Data, represented as points in space, are mapped, such as to leave a gap or margin as wide as possible between separate categories. New instances are then mapped into the new space and predicted to belong to a category depending on which side of the gap they fall on. Mathematically, we aim at finding a hyperplane $\mathcal{H} : \mathbf{w}^T\mathbf{x} + b = 0$ that separates the positive from the negative instances, where vector $\mathbf{w}$ is normal (perpendicular) to $\mathcal{H}$. The best hyperplane is that maximizing the margin around $\mathcal{H}$. If perfect separation is relaxed to allow for misclassified points, non-negative slack variables ξ_i are introduced and these points are penalized. Thus, the optimization problem results in $\min_{\mathbf{w},b,\boldsymbol{\xi}} (1/2)||\mathbf{w}||^2 + M\sum_{i=1}^{N} \xi_i$, subject to $1 - c^i(\mathbf{w}^T\mathbf{x}^i + b) - \xi_i \leq 0$ and $\xi_i \geq 0 \quad \forall i = 1, \ldots, N$, where M controls the tradeoff between the size of the margin and the slack variable penalty or errors. This (primal) problem is solved by allocating a Lagrange multiplier $\lambda_i \geq 0$ to each constraint and solving the dual problem. Mapping data—via a mathematical projection known as a "kernel trick"—to a much higher dimensional space where there is a linear decision rule is the most used alternative approach. The kernelized dual problem is $\max_{\boldsymbol{\lambda}} -(1/2)\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i\lambda_j c^i c^j K(\mathbf{x}^i, \mathbf{x}^j) + \sum_{i=1}^{N} \lambda_i$, subject to $0 \leq \lambda_i \leq M \quad \forall i = 1, \ldots, N$ and $\sum_{i=1}^{N} \lambda_i c^i = 0$, where $K(\cdot,\cdot)$ is the kernel function. The expressions for $\mathbf{w}$ and b are derived from the λ_i solutions.

In [127], the kernel is fixed as a radial basis function (RBF), whose defining hyperparameters and the penalty factor M are the optimization variables that the EDA uses. Two EDAs are used: 1) UMDA$_c^G$ and 2) BUMDA, based on a non-Gaussian approximation to each univariate Boltzmann density function. The objective function is the classification accuracy.

Artificial Neural Networks: Artificial neural networks are computational models for information processing that attempt to mimic the learning of biological neural networks [128]. They are inspired by an animal's central nervous system (in particular, the brain) and are used to estimate or approximate functions that can depend on a large number of inputs. Some layers of interconnected nodes, each building upon the previous layer, try to refine and optimize the prediction (forward propagation). The input values for each training instance are weighted and summed at each hidden layer neuron and the transfer function converts the weighted sum into the input of the output node layer. This is repeated through the network H times, where H is the number of hidden layers. Both the network architecture and the weights of each connection between neurons are found in view of maximizing the classification accuracy. Mathematically, the problem is $\max_{H,n_1,\ldots,n_H,\mathbf{w}^0,\mathbf{w}^1,\ldots,\mathbf{w}^H,f_1,\ldots,f_H} \text{Acc}(\phi)$, where ϕ is the neural network, n_i and f_i are the number of nodes and the transfer function, respectively, in the ith hidden layer, and $\mathbf{w}^i$ is the vector of weights connecting nodes between the $(i-1)$th and ith hidden layers.

Early work on EDAs in artificial neural networks focused on evolving the weights of multilayer perceptrons with a fixed topology [129], [130], [131], [132], and the use of this evolutionary computation method was considered as an alternative to the backpropagation method given its local optimality characteristic, stemming as a gradient descent method. In all these works, the multilayer perceptron topology only allowed one hidden variable layer. The individuals of the EDAs were real vectors of dimensions equal to the number of weights of the multilayer perceptron. The EDAs used for the search of the optimal weights were PBIL [129], [130], UMDA$_c^G$ [131], [132], and MIMIC$_c^G$ [131]. The input weights and hidden biases are the variables to be optimized in [133] in single-layer feedforward neural networks, coupling an EDA (PBIL) and the extreme learning machine model.

The simultaneous search for the optimal structure and weights of a multilayer perceptron is a more complex problem, that has also been addressed with EDAs. While PBIL was used in combination with a quasi-Newton method in [134] to optimize the discrete architecture and its corresponding real value weights at the same time, in [135], a variant of UMDA$_c^G$ within the probabilistic incremental program evolution framework was applied.

A simpler problem is that of the optimal pruning of synaptic connections from a complete artificial neural network topology, such as that developed in [136] for the case of a multilayer perceptron with a single hidden layer. In this case, the representation of individuals is direct and consists of assigning a bit for each connection between nodes in the multilayer perceptron. The bit denotes whether the corresponding connection is to be used. Three EDAs were used: 1) cGA; 2) EcGA; and 3) BOA.

The automatic design of a fully connected multilayer perceptron with only a hidden layer was approached with a UMDA in [137] for time series forecasting. The parameters to be optimized were the number of input nodes, the number of hidden neurons, how the nodes are connected (weights), and the seed used to initialize the connection weights in the backpropagation method.

Deep neural networks [138] contain multiple hidden layers of units between the input and output layers rendering the search for their optimal hyperparameters much more complex than in a multilayer perceptron with a single hidden layer of units. Evolutionary computation algorithms applied to optimize deep learning is called evolutionary deep learning. According to [22], evolutionary deep learning has been used in all deep learning models, roughly divided into convolutional neural networks, deep belief networks, stacked autoencoders, recurrent neural networks, and generative adversarial networks. The formulation above would contain additional parameters in the case of deep neural networks, e.g., the number of kernels in the convolutional layer, the kernel size, and the kind of pooling layer. In [139], univariate EDAs (with continuous and discrete variables treated separately) were used in convolutional networks for the simultaneous optimal configuration of the number of kernels in the convolutional layer, the kernel size of the convolutional layer, the number of neurons in the fully connected layers, the kind of activation function, and the kind of pooling layer. In [140], a hybrid method based on training gradient-based methods together with EMNA and UMDA$_c^G$ for weight optimization in convolutional neural networks was proposed. A UMDA$_c^G$ was applied in [141] to obtain the optimal weights in stacked autoencoders, i.e., several layers of autoencoders, where the output of each hidden layer is connected to the input of the successive hidden layer.

Logistic Regression: Logistic regression [142] is a flexible probabilistic supervised classification method that allows the coexistence of discrete and continuous predictor variables, and no assumptions are made about their density functions. The model is formulated as $p(C = 1|\mathbf{x}, \boldsymbol{\beta}) = (1/[1 + e^{-(\beta_0+\beta_1 x_1+\cdots+\beta_n x_n)}])$, where $\beta_0, \ldots, \beta_n$ are the coefficients and C denotes the binary (0/1) class variable, although extensions to the multiclass case do exist. The coefficients are estimated by maximizing the conditional log-likelihood function given by $\ln \mathcal{L}(\boldsymbol{\beta}|\mathbf{x}^1, \ldots, \mathbf{x}^N) = \sum_{i=1}^{N} c^i(\beta_0+\beta_1 x_1^i+\cdots+\beta_n x_n^i) - \sum_{i=1}^{N} \ln(1+e^{\beta_0+\beta_1 x_1^i+\cdots+\beta_n x_n^i})$. The equation system, with $n+1$ equations and $n+1$ unknowns to be solved to estimate $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)$ that maximize $\ln \mathcal{L}(\boldsymbol{\beta}|\mathbf{x}^1, \ldots, \mathbf{x}^N)$, does not have an analytical solution. Iterative techniques such as the Newton–Raphson method are used to provide solutions that often become stuck in local maxima.

In [143], EDAs were used to approximate the Pareto front for two objectives (a biobjective fitness function), calibration, as an alternative to the Newton–Raphson method for the maximization of the conditional log-likelihood, and discrimination, to maximize the area under the ROC curve. For each of these objectives, a UMDA$_c^G$ was designed. The best individuals obtained with each of these UMDA$_c^G$s were evaluated in the other objective, i.e., the objective that had not served to guide the search; thus, an approximation to the Pareto front was obtained for the biobjective problem. Each individual in the EDA contained the $n + 1$ optimization variables, the real numbers representing the β coefficients.

Regularized logistic regression is useful for problems with few samples and a large number of variables. Here, the regularization term needs to be determined. This involves searching for the optimal penalty parameter that represents a tradeoff between likelihood and coefficient shrinkage. In [144], a new regularized logistic regression method based on the evolution of regression coefficients using EDAs was presented. The key issue was the modification of the simulation step in a UMDA$_c^G$, whose individuals, as above, represent the β_i coefficients, to guarantee their shrinkage via truncated Gaussians as an intrinsic regularization approach.

Bayesian Classifiers: Minsky [145] showed that the simplest Bayesian classifier, the naive Bayes classifier, in which the predictor variables are conditionally independent given the class variable, has decision boundaries that are hyperplanes. As a result of this very negative theoretical result for naive Bayes, research on Bayesian classifiers did not resume until 1997 when Friedman et al. [146] introduced tree-augmented naive Bayes classifiers. Bayesian classifiers are used to compute the

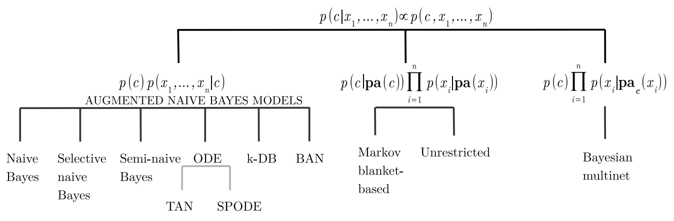
$p(c|x_1,\ldots,x_n) \propto p(c,x_1,\ldots,x_n)$
$p(c)\,p(x_1,\ldots,x_n|c)$
AUGMENTED NAIVE BAYES MODELS
$p(c|\mathbf{pa}(c))\prod_{i=1}^{n} p(x_i|\mathbf{pa}(x_i))$
$p(c)\prod_{i=1}^{n} p(x_i|\mathbf{pa}_c(x_i))$
Naive Bayes
Selective naive Bayes
Semi-naive Bayes
ODE
k-DB
BAN
TAN
SPODE
Markov blanket-based
Unrestricted
Bayesian multinet

Fig. 4. Classification of discrete Bayesian classifiers [147]. ODE: one-dependence estimator; TAN: tree-augmented naive Bayes; SPODE: superparent-one-dependence estimator; k-DB: k-dependence Bayesian classifier; and BAN: Bayesian network augmented naive Bayes.

posterior probability of the class variable $p(c|x_1, \ldots, x_n)$ from the JPD $p(c, x_1, \ldots, x_n)$, factorized according to the graph structure, i.e., $p(\mathbf{x}, c) = p(c|\mathbf{pa}(c)) \prod_{i=1}^{n} p(x_i|\mathbf{pa}(x_i))$, where $\mathbf{pa}(c)$ and $\mathbf{pa}(x_i)$ denote values of the parents of C and X_i, respectively. In the survey by Bielza and Larrañaga [147], Bayesian classifiers were grouped based on the different factorizations of the JPD (Fig. 4).

The optimization problem associated to Bayesian classifiers consists of searching the best parent variables $\mathbf{Pa}(C)$ and $\mathbf{Pa}(X_i)$ (the structure of the classifier) and estimating the parameters $\boldsymbol{\theta}_i = p(x_i|\mathbf{pa}(x_i))$ and $\boldsymbol{\theta}_c = p(c|\mathbf{pa}(c))$ (conditional probabilities) that maximize the classifier performance, i.e., $\max_{\mathbf{Pa}(C),\mathbf{Pa}(X_1),\ldots,\mathbf{Pa}(X_n),\boldsymbol{\theta}_c,\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_n} \mathrm{Acc}(\phi)$, where ϕ is the classifier.

In [148], the interval estimation naive Bayes algorithm was introduced. This algorithm starts with an estimation of the conditional probabilities of each predictor variable given the class variable, using confidence intervals. Subsequently, a UMDA_c^G is used to search within each confidence interval for the optimal point estimate (the optimization variables) in terms of maximizing the accuracy of the naive Bayes classifier.

In the seminaive Bayes model [149], the naive Bayes conditional independence assumption is relaxed by introducing new variables obtained as the Cartesian product (supernode) of two or more original predictor variables. Thus, the model can be used to represent dependencies between the original predictor variables. The new predictor variables are still conditionally independent given the class variable. The seminaive Bayes modeling algorithm proposed in [149] is based on a greedy search. To prevent the search from getting stuck in local maxima, in [150], the search was carried out with a UMDA. The individuals of this UMDA have n bits (the optimization variables), each one with an integer value in $\{0, 1, 2, \ldots, n\}$, representing the supernode that each variable belongs to (0 means that the variable is not part of the model).

Metaclassifiers: In metaclassifiers, which are also referred to as multiple classifier systems, a set (or ensemble) of classifiers are combined to solve the same supervised classification problem. The stacked generalization, bagging, boosting, cascading, and random forest methods are among the most developed metaclassifiers. There are several optimization problems to be solved to obtain the optimal design of the abovementioned metaclassifiers, all with the aim of maximizing the classification accuracy.

Stacked generalization [151] is a generic methodology where the outputs of a set of base classifiers are combined through another classifier. An interesting (combinatorial) optimization problem that arises is the selection of classifiers to be combined from the set of base classifiers. This is called classifier subset selection and is approached with EDAs (EBNA) in [152]. An individual in the EDA algorithm has a binary encoding, where each position (optimization variable) refers to a concrete base classifier from the set of candidates, and its values are 1 if the base classifier is used and 0 otherwise.

Boosting [153] builds the ensemble of classifiers incrementally, adding one classifier at a time. In (the standard algorithm) AdaBoost, the original dataset is first sampled from a uniform distribution over the instances, whereas the classifier added at step t is selectively trained on a dataset sampled from a distribution that is adapted at each step. The instances where the preceding classifiers fail increase their likelihood of being in the next sample. AdaBoost uses the same base classifier in all steps. In [154], a UMDA_c^G was initialized with means equal to the weights of each classifier built by the AdaBoost algorithm. Then, the EDA tries to improve the whole ensemble prediction by evolving the voting weights of each classifier (the optimization variables) for each class label.

Regression: A standard regression task provides a prespecified model structure (mathematical function relating the response variable Y and the predictors $X_1, \ldots, X_n$) and finds the parameters that best fit a given dataset. Fitting means a minimum (squared) distance of data points from the function (least squares). Parameters are usually estimated with closed formulas (linear regression) or iterative methods (nonlinear regression, as empirical growth curves, exponential models, and Coob–Douglas models [155]). Although the latter require choosing starting guesses, incremental change, and step size, this is usually tackled by trial and error strategies. EDAs seem not to have any contribution in this field of standard regression.

A second approach is symbolic regression that simultaneously searches for a model and its parameters. Model search entails moving in the space of mathematical expressions (mathematical operators, analytic functions, constants, and state variables) aiming at best fitting a given dataset, both in terms of accuracy and simplicity. Since symbolic regression is an NP-hard problem, evolutionary algorithms are appropriate tools. Two new operators, called α and β, are proposed to represent a mathematical model such that the resulting equations are simplified. EDAs (UMDA_c^G) are used to select the appropriate operators and parameters (the optimization variables) [156]. The mean square error is used as a fitness function to be minimized. Due to the few parameters used by EDAs compared with other metaheuristics as differential evolution, genetic algorithms, or particle swarm optimization, in [157] a discrete UMDA is proposed to solve the symbolic regression problem. Denoising autoencoder genetic programming is an EDA for genetic programming, where the probabilistic model is denoising autoencoder long short-term memory networks. In [158] this algorithm is used for symbolic regression.

A third approach to regression is based on supervised machine learning techniques. In [159] the parameters of a support vector regression, an extension of support vector machines for regression, are optimized with an UMDA_c^G. The parameters

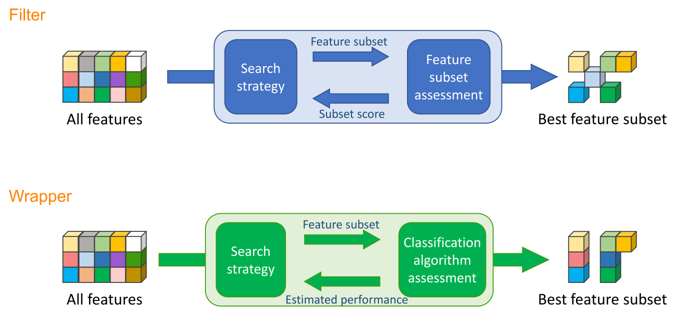

Fig. 5. Schema of filter (top) and wrapper (bottom) feature selection methods.

are the weight vector **w** and the threshold value b (optimization variables).

Feature Subset Selection: Feature subset selection [160] is the process of identifying and removing as many irrelevant and redundant variables as possible. This reduces the dimensionality of the data and may help learning algorithms operate faster and more effectively. The resulting model is a more compact and easily interpreted representation of the target concept, and in some cases, the accuracy of future classification may the improved. Two main approaches for feature subset selection have been developed [1]: 1) filter and 2) wrapper. In filter feature subset selection methods, the relevance of a feature, or a subset of features, is assessed from only the intrinsic properties of the data. In univariate filtering, a feature relevance score is calculated according to the class, and low-scoring features are removed. In multivariate filter methods, the subset of features is chosen according to its relevance (with respect to the class) and interfeature redundancy. Then, the subset of selected features obtained with the univariate or multivariate filter method is used as input variables for the classification algorithm. In contrast, wrapper methods evaluate each possible subset of features with a criterion consisting of the estimated performance of the classifier built with this subset of features. Thus, unlike wrapper methods, filter methods screen variables without taking into account the subsequent classifier to be used (see Fig. 5). The cardinality space for the multivariate filter and wrapper approaches is 2^n, where n denotes the number of features. The univariate filter is simply a feature ranking.

Evolutionary algorithms have also been used to address the feature subset selection problem, mainly because the standard representation of individuals for this problem only requires a binary vector $\mathbf{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ if variable X_i is selected and 0 otherwise. All the papers we have reviewed on the use of EDAs for feature subset selection refer to wrapper approaches, where the optimization problem is formulated as $\max_{\mathcal{S}\subseteq\{X_1,\ldots,X_n\}} \text{Acc}(\phi_{\mathcal{S}})$, where $\phi_{\mathcal{S}}$ is the classifier built with the variables in $\mathcal{S}$.

In the seminal paper [161], EBNA was used in combination with a classification tree induction method and a naive Bayes classifier. The naive Bayes classifier was also used in [162] for an empirical comparison among EBNA, two types of genetic algorithms, and two greedy algorithms as sequential feature selection and sequential feature elimination; in [163], three EDAs, namely, cGA, EcGA, and BOA, were experimentally compared. In [164], a hybrid method consisting of a genetic algorithm and a UMDA, named TEDA, was applied to problems with tens of thousands of predictor variables. In [165], the EDA process (UMDA$_c^G$) was embedded in an adapted recursive feature elimination procedure of a logistic regression model. Recently a biobjective EDA (a variant of UMDA) was proposed for the feature subset selection problem in intrusion detection, and the accuracy of the classifier (a classification tree in this case) and the number of selected features were taken into account [166]. Fast feature selection is a concern in the fast EDA (FEDA) of [167]. FEDA does not evaluate all new individuals by the actual fitness function, but with an approximate model based on a special Bayesian network to assign fitness values. A strategy allows to filter subsets of variables that are informative with high fitness values or in an unexplored region. The wrapper approach is then applied (in this case a naive Bayes model) on this reduced set.

TABLE II
CLUSTERING METHODS APPROACHED WITH EDAS

Hierarchical	Merging clusters [169]	UMDA
Partitional	K-means [171]	MIMIC, COMIT, EBNA
	K-medoids [173]	UMDA
	Affinity propagation [175]	UMDA, EBNA
	Graph-based [176]	UMDA
	Density-based [177]	PBIL
Probabilistic	Bayesian network-based [180]	UMDA

VI. EDAs IN CLUSTERING

Table II provides a summary of the papers reviewed in this section.

In cluster analysis, which is also referred to as unsupervised classification, the aim is to group a collection of N objects into subsets, or clusters, so that the objects within each cluster are more closely related to one another than objects assigned to different clusters. Three main types of cluster analysis methods have been developed: 1) hierarchical clustering; 2) partitional clustering; and 3) probabilistic clustering. In the three clustering methods, the object grouping process can be seen as an optimization problem, hence the use of metaheuristics such as EDAs can help in the search for optimal clustering.

Hierarchical Clustering: Hierarchical clustering algorithms [168] organize the objects in a hierarchical structure depicted by a binary tree or dendrogram (see Fig. 6). Agglomerative hierarchical clustering algorithms start with N clusters, each of which includes exactly one object. A series of merging operations, designed to group all objects within the same cluster is then performed. Merging operations are applied to pairs of subsets of objects. Iterative searches are performed at each step to obtain the best grouping according to a certain dissimilarity criterion (single linkage, complete linkage, average linkage, centroid linkage, Ward's method, etc.); thus, the number of clusters is reduced by one unit until all objects belong to the same cluster at the end of this process. This dendrogram construction process is only locally optimal and does not guarantee that the resulting binary tree is globally optimal, since the actual formulation should be $\min_{D_{HC}} d(D_{HC}, D_{\text{dataset}})$, where d is an appropriate

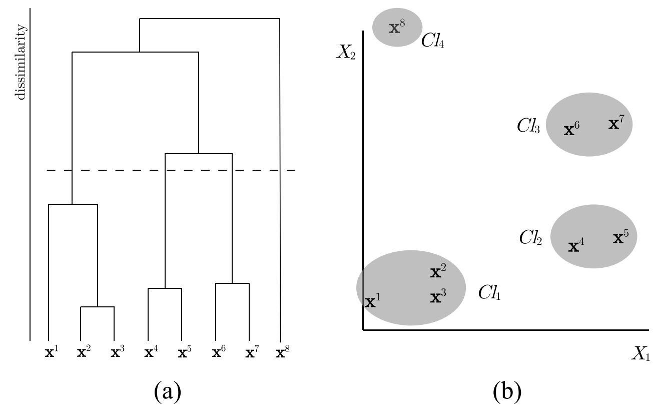
dissimilarity
$\mathbf{x}^1$ $\mathbf{x}^2$ $\mathbf{x}^3$ $\mathbf{x}^4$ $\mathbf{x}^5$ $\mathbf{x}^6$ $\mathbf{x}^7$ $\mathbf{x}^8$
(a)
X_2
X_1
Cl_1
Cl_2
Cl_3
Cl_4
(b)

Fig. 6. Output of hierarchical clustering. (a) Dendrogram representing the hierarchical cluster has been cut at the dotted line, yielding the four clusters in (b). (b) Eight objects represented in the dendrogram have been grouped into four clusters: $Cl_1 = \{\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3\}$, $Cl_2 = \{\mathbf{x}^4, \mathbf{x}^5\}$, $Cl_3 = \{\mathbf{x}^6, \mathbf{x}^7\}$, and $Cl_4 = \{\mathbf{x}^8\}$.

distance measure, D_{HC} is a dissimilarity matrix of dimension $N \times N$ between pairs of objects obtained from the hierarchical clustering output, and D_{dataset} is the dissimilarity matrix obtained from the dataset.

In [169], UMDA was applied to provide stochasticity to the subset merging process, promoting the most numerous subsets to have a higher probability of being joined as long as the value of the centroid linkage did not exceed a certain threshold. This restriction means that the constructed dendrogram does not necessarily have to be complete.

Partitional Clustering: Partitional clustering methods, such as the well-known K-means algorithm [170] (where K is the number of clusters), usually optimize a given cohesion criterion, for example, the sum of the distances of the objects to the centroids of the cluster to which they belong, via an iterative optimization procedure. The general problem is formulated as: $\min_{Cl_1,\ldots,Cl_K} \sum_{k=1}^{K} \sum_{\mathbf{x}^i \in Cl_k} ||\mathbf{x}^i - \mathbf{c}_k||^2$, where Cl_k refers to the kth cluster, and $\mathbf{c}_k = (c_{k1}, \ldots, c_{kn})$ is its corresponding centroid.

EDAs were used in Forgy's version with the object membership representation of individuals [171]. In this representation, the EDA individuals are strings of length N, where each position can take integer values from 1 to the number of clusters K. The value in the ith position (optimization variable) represents the cluster to which the object that occupies that position belongs. Different EDAs, such as MIMIC, COMIT, and EBNA were compared empirically. While the centroid is representative of each cluster in the K-means algorithm, in the K-medoids algorithm [172], the representatives of each cluster must be objects that are present in the dataset (an issue that does not occur in K-means). This facilitates the representation of the individuals of an evolutionary algorithm that attempts to approach the problem of the search for the optimal medoids. In this way, in [173], a UMDA approach with an encoding of individuals was proposed based on binary strings (the optimization variables), in which 1 (0) indicated that the corresponding object in the input dataset was (not) considered as a medoid. The K value or number of clusters was also found.

The affinity propagation algorithm [174] is based on the concept of message passing between objects. Its aim is to find a subset of representative objects called exemplars. As in the K-medoids algorithm, exemplars are members of the input dataset. Unlike K-medoids, the affinity propagation algorithm simultaneously considers all objects as potential exemplars, avoiding the selection of initial exemplars. In [175], UMDA and EBNA were used to find the optimal preference of each object. Preferences are a measure of how likely each object is to be chosen as an exemplar, which is considered a highly influential parameter on the result of the affinity propagation algorithm. The optimization variables are the preference values, with three possible standard assignments.

In [176], an attempt was made to address the clustering problem with EDAs (UMDA) by searching for the optimal graph (a minimum weighted spanning tree based on the distance between objects), where the number of nodes equals the number of objects to cluster, and each edge in the graph links two objects that may belong to the same cluster. An individual is encoded with $N - 1$ variables corresponding to the tree edges. Each (optimization) variable of the EDA follows a Bernoulli distribution with a value of 1 if the two objects at the edge must link within the same cluster and 0 otherwise. The parameter of this distribution is inversely proportional to the distance between those two objects.

The use of EDAs to the automatic generation of density-based clustering algorithms was proposed in [177]. PBIL combined eight density-based cluster algorithms as if they were building blocks, thus creating new clustering algorithms.

Probabilistic Clustering: Probabilistic clustering is a type of model-based clustering based on fitting the density of all the sample data with finite mixture models. Fitting these finite mixture models requires the estimation of some parameters characterizing the component densities and some mixing proportions. Estimations are based on maximizing the log-likelihood of the data. However, these estimations have nonclosed solutions, and *ad hoc* procedures, such as the expectation-maximization (EM) algorithm [178], are widely used. In the E-step, the missing data (in probabilistic clustering, this refers to the cluster membership of each object) are estimated given the observed data and the current estimate of the model parameters. The estimates of the missing data from the E-step are used to output a version of the complete data. In the M-step, the complete-data log-likelihood function is maximized under the assumption that the missing data are known.

Bayesian networks provide an intuitive and natural way of performing model-based clustering. It is sufficient to introduce a hidden node representing the cluster variable, H, to yield models with a latent structure where the data are systematically missing. These Bayesian networks can express the probability distribution of the observed data $\boldsymbol{X}$ as a parametric finite mixture model (Fig. 7). This has two main advantages: 1) the factorization of the distribution according to the structure of the probabilistic graphical model and 2) the use of efficient methods for exact inference to provide a probability distribution over the values of the cluster variable H. Probabilistic clustering via Bayesian networks aims to find the structure

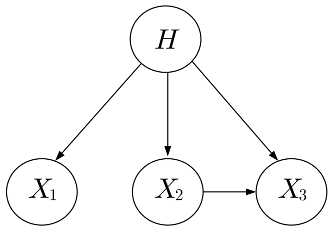

Fig. 7. Structure of a Bayesian network with a hidden variable represented by H. The probability distribution of the observed data can be written as a parametric finite mixture model with K components: $p(x_1, x_2, x_3; \boldsymbol{\theta}) = \sum_{h=1}^{K} p(h; \boldsymbol{\theta}) p(x_1|h; \boldsymbol{\theta}) p(x_2|h; \boldsymbol{\theta}) p(x_3|x_2, h; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the parameter vector and K is the number of clusters.

and parameters of a DAG with similar structure to augmented naive Bayes models of Fig. 4, but replacing the class variable C with the cluster variable H. These structures are denoted as $\mathcal{G}_{BN}^{\text{clus}}$. Thus, the formulation is $\max_{\mathcal{G}_{BN}^{\text{clus}}} \ln \mathcal{L}(\mathcal{D}|\mathcal{G}_{BN}^{\text{clus}})$, where $\mathcal{L}(\cdot)$ is the likelihood of the dataset $\mathcal{D}$ given the Bayesian network-based clustering model. In [179], the use of the EM algorithm was proposed to exploit the message passing procedure for inference to efficiently perform the E-step in Bayesian networks.

In [180], the use of EDAs based on UMDA was proposed to search for the optimal structure of dependency between the variables. Each individual in the UMDA represents an upper triangular connectivity matrix with $(n^2 - n/2)$ elements m_{ij}, such that $m_{ij} = 1$ if $X_j \in \mathbf{Pa}(X_i)$ and 0 otherwise. m_{ij} are the optimization variables.

VII. EDAs in Reinforcement Learning

The aim of reinforcement learning [181] is to develop intelligent agents capable of adopting actions that maximize the expected reward (fitness function) for each state of the system. Many algorithms that search for the policy of actions (optimization variables) to take for each possible state of the system have been developed. In some of these algorithms, the conditional probability distribution of the actions given the agents's state is explicitly determined. In [182], a conditional random field [183] was learned from the selected episodes (individuals providing the best rewards) at each generation of an EDA based on conditional random fields. Conditional random fields are models that naturally adapt to the sequential decision process inherent in reinforcement learning, as they are discriminative models for the classification of sequential data.

Deep reinforcement learning uses deep neural networks to learn policies in high-dimensional input spaces. Deep Q-network [184], a type of deep reinforcement learning, is combined with EDAs (PBIL) to strengthen the exploitation and exploration capabilities in a job shop scheduling problem [185]. An individual in the EDA has three parts: 1) the scheduling sequence of all operations; 2) the machine assignment of all operations; and 3) the processing speed level of each operation. Both maximum completion time and total electricity price are maximized simultaneously (biobjective problem).

TABLE III
Optimization Problems Solved With EDAs in Bayesian Networks

Inference	Triangulation [189]	REDA
	MAP [190]	UMDA, MIMIC, EBNA
Learning	DAGs [191]	UMDA, PBIL
	DAGs [192]	UMDA, PBIL
	DAGs [193]	UMDA, PBIL, MIMIC, BOA
	DAGs [194]	PBIL
	Total ordering [195]	UMDA, MIMIC, UMDA$_c^G$, MIMIC$_c^G$
	HMMs [197]	PBIL

VIII. EDAs in Association Discovery With Bayesian Networks

The machine learning task of discovering associations between a set of random variables has its major representative in Bayesian network models. EDAs have been used within the field of Bayesian networks for different problems that arise in exact methods of inference or, alternatively, for parameter and structure learning algorithms based on the score and search approach. Table III provides a summary of the papers reviewed in this section.

A. Inference

In [186], one of the most popular algorithms for exact inference, a task that is NP-hard [187], was proposed for multiply connected Bayesian networks. The first step in this algorithm is to moralize the Bayesian network structure, i.e., all variables with a common child are linked and then all edge directions are removed. The resulting graph is called a moral graph. The second step of the algorithm (considered the toughest step in terms of computational complexity) is the so-called triangulation of the moral graph. A graph is triangulated if any cycle of length greater than three has a chord. The resulting structure is then used for evidence propagation and probability computation. The basic technique for triangulating a moral graph (Fig. 8) is through the successive elimination of graph nodes. Before eliminating a node and its edges, we check that all its adjacent nodes are directly connected to each other by adding the required edges to the graph (complete subgraph). The nodes are chosen for elimination according to a given order of the variables. The quality of the triangulation is measured by the weight of the triangulated graph, $w(S^t) = \log_2(\sum_{Cl_i} \prod_{X_i \in Cl_i} r_i)$, where Cl_i denotes a clique of the triangulated graph S^t, composed of vertices X_i, each with r_i different states. A clique is a subgraph that is complete (all nodes are pairwise linked) and maximal (it is not a subset of another complete set). This weight (objective function) is determined by the order in which the nodes are eliminated and becomes worse if more edges are added. Hence, the search for an optimal triangulation is equivalent to the search for an optimal node elimination sequence (individuals), i.e., the search for an optimal permutation of nodes. In [189], an approach based on recursive EDAs (REDAs) was proposed for both discrete and continuous representation of the variables. REDAs partition the set of vertices (that are to be ordered) into two subsets. In each REDA call, the vertices in the first subset are fixed, whereas the other subset of variables is evolved with a standard EDA. In the second call, the subsets switch roles.

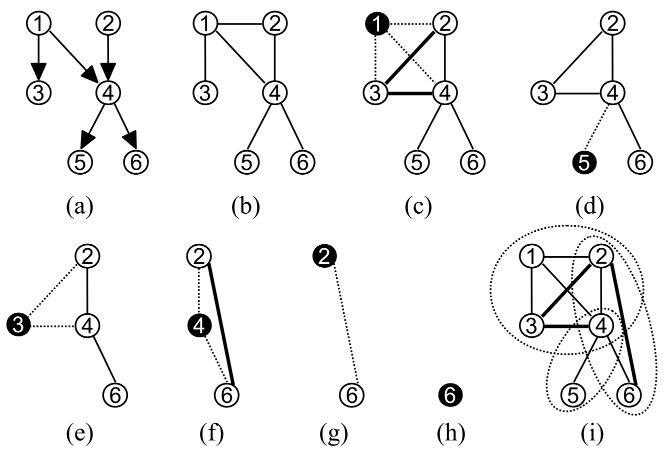

Fig. 8. Example of the triangulation algorithm. Nodes are eliminated in order: $X_1, X_5, X_3, X_4, X_2, X_6$ and it is assumed that $r_i = i + 1$ with $i = 1, \ldots, 6$. (a) Initial DAG. (b) Related moral graph. (c) Eliminate X_1: $Cl_1 = \{X_1, X_2, X_3, X_4\}$, added edges: $\{X_2, X_3\}, \{X_3, X_4\}$. (d) Eliminate X_5: $Cl_2 = \{X_4, X_5\}$. (e) Eliminate X_3: $Cl_3 = \emptyset$. (f) Eliminate X_4: $Cl_4 = \{X_2, X_4, X_6\}$, added edge $\{X_2, X_6\}$. (g) Eliminate X_2: $Cl_5 = \emptyset$. (h) Eliminate X_6: $Cl_6 = \emptyset$. (i) Total weight of the triangulated graph: $\log_2(2 \cdot 3 \cdot 4 \cdot 5 + 5 \cdot 6 + 3 \cdot 5 \cdot 7) = \log_2 255$. Adapted from [188].

The aim of partial abductive inference, also known as the maximum a posteriori (MAP) problem, is to find the most likely configuration for a subset $\boldsymbol{X_E}$ of variables in the Bayesian network, which is known as the explanation set. Given a set of observed variables $\boldsymbol{X_O} = \boldsymbol{x_O}$, and denoting $\boldsymbol{X_U} = \boldsymbol{X} \setminus \boldsymbol{X_O}$ as the set of unobserved variables, we have $\boldsymbol{X_E} \subset \boldsymbol{X_U}$. In a MAP problem, the aim is to obtain the configuration $\boldsymbol{x_E}^*$ for $\boldsymbol{X_E}$ such that $\boldsymbol{x_E}^* = \arg\max_{\boldsymbol{x_E}} p(\boldsymbol{x_E}|\boldsymbol{x_O})$. In [190], discrete EDAs with different degrees of model complexity (UMDA, MIMIC, and EBNA) were proposed to solve the MAP problem. The individuals in the EDA population represent a possible configuration only for the variables in the explanation set.

B. Learning

For parameter and structure learning algorithms based on the score and search approach, the goal is to $\max_{\mathcal{G}_{BN}} \text{Score}(\mathcal{D}, \mathcal{G}_{BN})$, where Score refers to any criterion based on penalized likelihood or a Bayesian score, see Section II-C.

In the space of DAGs, [191] used two univariate EDAs (UMDA and PBIL) in combination with three different scoring metrics (BIC, K2, and entropy). The individuals represent a connectivity matrix. To prevent the new individuals obtained by simulating the Bayesian network from being cyclic graphs, a repair operator was introduced. Once a cycle is detected in the individual, the repair operator randomly deletes one arc of the cycle. This random deletion is repeated until a DAG is obtained. Thibault et al. [192] used a representation of individuals similar to [191] with UMDA and PBIL as algorithms and BIC as the score to be maximized. Kim et al. [193] incorporated a mutation based on matrix transposition in the EDA algorithm flow. This matrix transposition increases the possibility of inferring the correct arc direction by considering the direction of edges in candidate solutions as bidirectional. Four standard EDAs, UMDA, PBIL, MIMIC, and BOA, were used together with BDe and BIC as scores. Fukuda et al. [194] introduced a new mutation operator named the probability mutation. This operator changes the probability distribution learned by PBIL and takes into account that the acyclicity restriction of the graph is fulfilled at the same time.

In the space of possible orderings on the nodes, [195] applied two types of discrete-encoded (UMDA and MIMIC) and continuous-encoded (UMDA$_c^G$ and MIMIC$_c^G$) EDAs to obtain the best ordering for the K2 algorithm. The K2 algorithm needs a fixed total order between the nodes, as its result is dependent on that order. Therefore, EDAs try to obtain the best among the $n!$ possible orders. For discrete encoding, they used a bijective mapping to represent possible orderings of n variables with $n-1$ random variables. The simulation step was adapted to output a valid permutation of the variables. This adaptation is also necessary for continuous encoding, where each n-dimensional real vector can be transformed into a valid permutation of the n variables.

Hidden Markov models (HMMs) [196] are dynamic Bayesian networks used to model Markov processes that cannot be directly observed but can be indirectly estimated by state-dependent outputs; in other words, the state is not directly visible but the state-dependent output is. The parameters of these models (transition probabilities between hidden states, emission probabilities of the observations given the hidden state, and initial probability distribution of the hidden states) are commonly learned using algorithms derived from gradient-based methods, such as the Baum–Welch procedure. Maxwell and Anderson [197] applied PBIL as an alternative to the Baum–Welch procedure, considering an individual as the concatenation of the three types of parameters.

IX. Real-World Applications of EDAs in Machine Learning

This section shows a number of EDAs selected applications for inducing machine learning models. First, EDAs for association rule mining in traffic flow prediction and in blood index analysis are found in [114] and [115], respectively. Second, in supervised learning, EDAs contribute to finding: the support vector regression parameters applied to software reliability prediction [159]; the parameters of a full connected multilayer perceptron with a hidden layer in time series forecasting problems [137]; the weights in convolutional neural networks for text categorization [140]; the input weights and hidden biases in single layer feedforward neural networks for drought prediction in China [133]; the regularized parameters of logistic regression models in microarray data classification [165]; and in different feature selection problems: of clinical findings in cirrhotic patients treated with transjugular intrahepatic portosystemic shunt [198], of peakbins in mass spectrometry data [199], of nucleotides for splice site prediction in *Arabidopsis thaliana* [200], of traffic and connection features in intrusion detection [166], and to predict the likely method (negotiation, won at trial, etc.) of settlement for a claim in a legal business [201]. Third, EDAs help in gene expression data clustering and biclustering [202] and probabilistic clustering [180]. Finally, in [182], EDAs were used in the reinforcement learning problems of perceptual aliasing maze [203], and of flexible job shop scheduling [185].

X. Conclusion and Further Topics

A. Conclusion

We have reviewed work in the literature on the use of discrete and continuous estimation of distribution algorithms to different machine learning problems, ranging from data preprocessing to association rule mining, variable selection, supervised learning (classification and regression), clustering, reinforcement learning, and association discovery with Bayesian networks. According to Tables I–III, univariate EDAs are used three times more than complex probabilistic models (bivariate or multivariate models). Some commonalities drawn from the survey are that many taks in machine learning can be posed as combinatorial optimization problems where discrete EDAs are useful, like searching for the Bayesian network structure, the classification tree, the architecture of an artificial neural network, the feature subset, the operators of symbolic regression, the cluster assignment, or the policy of actions in reinforcement learning. Continuous EDAs may help in continuous optimization problems, like parameter estimation in logistic regression, optimal weights in artificial neural networks, or hyperparameters in support vector machines.

In our view, most papers on recent EDAs focus on solving optimization problems without a machine learning-oriented goal. Thus, the main message of this survey is that the list of machine learning tasks posed as an optimization problem is really long and there is much room for the EDAs to leverage these challenging problems.

B. Further Topics

EDAs based on Bayesian networks might adopt modern learning algorithms [204] providing better data fitting structures (such as semiparametric Bayesian networks [205], [206], in which nodes estimated by kernels coexist with nodes that assume Gaussianity) and better efficiency could result in the improved performance of EDAs. This has been done in the new SPEDA [207], based on semiparametric Bayesian networks, which turned out to be one of the best-performing approaches with respect to other state-of-the-art algorithms in continuous optimization.

The most frequent use of simple EDAs as a starting practice is in fact a recent recommendation of the Dagstuhl Report from the EDAs seminar held in May 2022 [208]. In the case of unsatisfactory results, restart strategies should be tried and then ultimately more complex interaction models would be pursued. Furthermore, the same level of dependence among the variables is usually maintained during the EDA run. However, smarter EDAs could change these levels depending on the landscape shape that is visited in each generation. Finally, the interpretability of the probabilistic graphical models learned in each EDA generation is not sufficiently exploited. Mapping the dependencies captured by these probabilistic models to the optimization problem structure might reveal unknown information about the problem [107]. New ideas such as search trajectory networks [209] can be adapted for visualizing and analyzing EDA behavior.

There is scarce literature on the use of evolutionary computation (EDAs included) in machine learning for data stream scenarios, a situation that is becoming increasingly important in real-world applications. Another important issue concerns computationally demanding fitness functions. The use of surrogate models to estimate them is common practice [208]. Within supervised learning, fitness functions are usually estimated with honest methods (repeated holdout, k-fold cross-validation, boostrapping, etc.). However, an interval estimation rather than a point estimation would be fairer to evaluate individuals. This would also influence their subsequent selection.

Some procedures, when preprocessing the dataset, are amenable to the use of EDAs. We can mention the multivariate imputation of missing values, dimensionality reduction (e.g., nonlinear principal component analysis and multidimensional scaling), and visualization issues (e.g., parallel coordinate plots and optimal graph layouts).

The k-nearest neighbors algorithm could take advantage of EDAs in the prototype selection problem, the number of nearest neighbors, the distance function, and the scheme for weighting the nearest neighbors. There are a number of interesting decisions that can be made for classification trees using EDAs, e.g., finding the best combination of variables in the hyperplane that defines each internal split in oblique trees or improving tree pruning. Rule induction based on the Michigan approach has not been found with EDAs. The kernel function in support vector machines could be used as a hyperparameter to be set by an EDA.

A taxonomy of possible optimization tasks in evolutionary deep learning is as follows [22].

1) In data processing, the problems are how to generate better data, how to achieve better data balance, and how to improve the efficiency of data preprocessing to meet the given requirements.
2) In model search, questions consider how to search efficiently for optimal architectures, optimal hyperparameters, and better models satisfying multiple objectives.
3) In model training, key issues are how to efficiently find optimal parameter values, how to efficiently pretrain models to avoid problems caused by the random initialization of weights, and how to reduce the training cost.
4) In model evaluation and utilization, pending tasks include how to better evaluate the robustness of models, how to achieve better model ensembles and how to better prune models.

Although many evolutionary computation algorithms have contributed to 1)–4), EDAs have not, thereby opening opportunities in this popular machine learning area.

From the whole family of discrete Bayesian classifiers (Fig. 4), only selective naive Bayes and semi-naive Bayes models have benefited from the use of EDAs. In Bayesian classifiers with continuous predictor variables and even with both continuous and discrete variables, EDAs have not been used.

This survey has only found two EDA proposals in improving stacking and boosting, but many ensembles, such as cascading, could benefit from EDA-based searches of the number of classifiers and their type in the chain. Furthermore, likewise other evolutionary algorithms, EDAs could be used to improve the diversity of ensembles of classifiers.

In regression, EDAs can be useful for deciding which interaction terms of variables (and their degree) should be considered for a good fitting in terms of some performance measure as the mean squared error. This extends the simple additive effects in standard regression. Also, multioutput regression [210], where many response variables are to be predicted simultaneously, may take advantage of EDAs for searching the parameters of each regression.

Hierarchical clustering with EDAs could benefit from a more global fitness function that can guide the evolution of dendrograms. Lozano and Larrañaga, [211] addressed this topic (although with a genetic algorithm), where an ultrametric distance associated with the dendrogram that fits the dissimilarity matrix of the dataset was sought. Castellanos-Garzón et al. [212] also used a genetic algorithm where individuals represent dendrograms, and the fitness function was recurrently defined according to the concepts of homogeneity and separation. Centroid-based representations of individuals are shorter than label/tree/graph-based representations but are not used in EDAs for partitional clustering. In probabilistic clustering with Gaussian mixture models, the structure of the Bayesian network with a hidden node (Fig. 7) sought by the EDA may be turned into a Bayesian multinet (with the hidden node as the distinguished variable) to yield different Gaussians for each cluster. Moreover, EDAs can be used to find the parameters of each mixture component (weight, mean vector, and covariance matrix) as an alternative to the structural EM algorithm. All EDA approaches found in partitional and probabilistic clustering work with a number of clusters that are known and this could be relaxed. Selecting features (and even weighting them) tailored to the clusters found in high-dimensional spaces is an area where EDA contributions are missing. Clustering based on deep learning could benefit from the use of EDAs. Other more sophisticated variants of clustering, such as spectral clustering, biclustering, multiview clustering, ensemble clustering, or multipartition clustering, could be approached with EDAs.

EDAs may be helpful in inference problems with Bayesian networks for association discovery, which have an extreme computational complexity. Examples are most relevant explanations, most frugal explanations, MAP-independence explanations, same-decision probability, and counterfactual reasoning. There are also opportunities for applying EDAs in Bayesian network structure learning algorithms in dynamic settings.

Fig. 9 includes the main topics for further research.

Most machine learning problems are multiobjective in nature. The choice of a suitable set of objective functions is not trivial. In association rule mining, several rule interestingness measures, such as support, confidence, comprehensibility, and lift, may be optimized. Different performance measures, such as the F1-measure, area under the ROC curve, sensitivity, and specificity, may be maximized simultaneously in a supervised learning method. In multilabel (or multidimensional) classification, the performance measure of each class variable can be considered an objective rather than the common strategy of averaging them. In addition to performance measures, we can take into account model complexity issues, such as the number of hidden units in artificial neural networks, the depth or length in trees and rules, respectively, or the number of parameters in the support vector machine kernel function. Examples of using complexity (number of features) and accuracy [166]; or calibration and discrimination [143] as objectives were mentioned above (Section V). These are the only multiobjective EDAs for solving a machine learning task that we could find. Using multiobjective clustering is convenient to eliminate prior assumptions about the cluster structure that may not hold in the data. Additionally, several cluster validity indices or the number of clusters may act as multiple objectives. When learning Bayesian networks, optimizing many objectives, such as maximizing the likelihood, minimizing the number of parameters, and reducing the inference complexity of the network structure, may be the goal. Unlike other evolutionary algorithms, EDAs are usually guided by only one objective, and there is much room to explore.

Finally, the recent trend of automated machine learning (AutoML) aims at automating machine learning techniques to expand their use to anyone (laypersons or experts). Well-known AutoML methods include Auto-WEKA, Auto-Sklearn, Auto-Keras, and irace, which are based on existing machine learning libraries/packages. These methods can be used to optimize the integration of different techniques and their hyperparameters for data preprocessing, feature engineering, and learning processes. EDAs might play a key role in this integration.

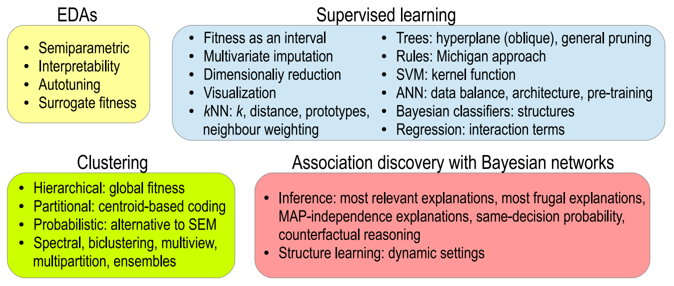
EDAs
Semiparametric
Interpretability
Autotuning
Surrogate fitness
Supervised learning
Fitness as an interval
Multivariate imputation
Dimensionaliy reduction
Visualization
kNN: k, distance, prototypes, neighbour weighting
Trees: hyperplane (oblique), general pruning
Rules: Michigan approach
SVM: kernel function
ANN: data balance, architecture, pre-training
Bayesian classifiers: structures
Regression: interaction terms
Clustering
Hierarchical: global fitness
Partitional: centroid-based coding
Probabilistic: alternative to SEM
Spectral, biclustering, multiview, multipartition, ensembles
Association discovery with Bayesian networks
Inference: most relevant explanations, most frugal explanations, MAP-independence explanations, same-decision probability, counterfactual reasoning
Structure learning: dynamic settings

Fig. 9. Summary of challenges for future work. *k*NN: *k*-nearest neighbors; SVM: support vector machines; ANN: artificial neural networks; and SEM: structural EM.

References

[1] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[2] H. Sharp, "Cardinality of finite topologies," *J. Combinatorial Theory*, vol. 5, pp. 82–86, Jul. 1968.

[3] R. W. Robinson, "Counting unlabeled acyclic digraphs," in *Combinatorial Mathematics* (Lecture Notes in Mathematics), vol. 622. Heidelberg, Germany: Springer, 1997, pp. 28–42.

[4] R. E. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs," *SIAM J. Comput.*, vol. 13, no. 3, pp. 566–578, 1984.

[5] T. Minka, "Algorithms for maximum-likelihood logistic regression," Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. 758, 2001.

[6] D. E. Rumelhart, G. R. Hinton, and R. J. Willians, "Learning representations by back-propagation errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[7] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.

[8] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Boston, MA, USA: Wesley, 1989.

[9] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.

[10] R. C. Eberhart and J. Kennedy, *Swarm Intelligence*. Burlington, MA, USA: Morgan Kaufmann, 2001.

[11] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad., 2002.

[12] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[13] L. J. Fogel, "Autonomous automata," *Ind. Res.*, vol. 4, no. 2, pp. 14–19, 1962.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[15] I. Rechenberg, *Evolutionstrategie: Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*. Stuttgart, Germany: Fromman-Holzboog, 1973.

[16] M. S. Krejca and C. Witt, "Theory of estimation-of-distribution algorithms," in *Theory of Evolutionary Computation*. Cham, Switzerland: Springer, 2020, pp. 405–442.

[17] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.

[18] B. Badhon, M. M. Xabit, S. Xu, and M. Kabir, "A survey on association rule mining based on evolutionary computation," *Int. J. Comput. Appl.*, vol. 41, no. 1, pp. 1–11, 2019.

[19] A. Telikani, A. H. Gandomi, and A. Shahbahrami, "A survey of evolutionary computation for association rule mining," *Inf. Sci.*, vol. 524, pp. 318–352, Jul. 2020.

[20] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. de Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 291–312, May 2012.

[21] A. Darwish, A. E. Hassanien, and S. Das, "A survey of swarm and evolutionary computing approaches for deep learning," *Artif. Intell. Rev.*, vol. 53, no. 3, pp. 1767–1812, 2020.

[22] Z.-H. Zhan, J.-Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, Apr. 2022.

[23] N. Li, L. Ma, G. Yu, B. Xue, M. Zhang, and Y. Jin, "Survey on evolutionary deep learning: Principles, algorithms, applications and open issues," 2022, *arxiv2208.10658v1*.

[24] M. J. A. Hasan and S. Ramakrishnan, "A survey: Hybrid evolutionary algorithms for cluster analysis," *Artif. Intell. Rev.*, vol. 36, no. 3, pp. 179–204, 2011.

[25] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Trans. Syst., Man Cybern. C, Appl. Rev.*, vol. 39, no. 2, pp. 133–155, Mar. 2009.

[26] A. Mukhopadhyay, U. Maulik, and S. Bandyopadhyay, "A survey of multiobjective evolutionary clustering," *ACM Comput. Surveys*, vol. 47, no. 4, pp. 1–46, 2015.

[27] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on evolutionary algorithms in Bayesian network learning and inference tasks," *Inf. Sci.*, vol. 233, pp. 109–125, Jun. 2013.

[28] H. Al-Sahaf et al., "A survey on evolutionary machine learning," *J. Royal New Zealand*, vol. 49, no. 2, pp. 205–228, 2019.

[29] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello-Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 4–19, Feb. 2014.

[30] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello-Coello, "Survey of multiobjective evolutionary algorithms for data mining: Part II," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 20–35, Feb. 2014.

[31] K.A. De Jong, *Evolutionary Computation. A Unified Approach*. Cambridge, MA, USA: MIT Press, 2006.

[32] C. Darwin, *The Origin of Species by Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life*. London, U.K.: John Murray, 1859.

[33] M. Pelikan, D. E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," *Comput. Optim. Appl.*, vol. 21, no. 1, pp. 5–20, 2002.

[34] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Heidelberg, Germany: Springer, 2005.

[35] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm Evol. Comput.*, vol. 1, no. 3, pp. 111–128, 2011.

[36] R. Santana, J. A. Lozano, and P. Larrañaga, "Research topics in discrete estimation of distribution algorithms based on factorizations," *Memet. Comput.*, vol. 1, pp. 35–54, Mar. 2009.

[37] P. Larrañaga, H. Karshenas, C. Bielza, and R. Santana, "A review on probabilistic graphical models in evolutionary computation," *J. Heurist.*, vol. 18, no. 5, pp. 795–819, 2012.

[38] J. Ceberio, A. Mendiburu, and J. A. Lozano, "A roadmap for solving optimization problems with estimation of distribution algorithms," *Nat. Comput.*, 2022, doi: 10.1007/s11047-022-09913-2.

[39] X. Qian, J. Yu, A. Zhao, Q. Liu, and R. Zhang, "Decentralised estimation of distribution algorithm for parallel pumps based on log-linear model," *Int. J. Smart Grid Green Commun.*, vol. 2, no. 1, pp. 70–85, 2020.

[40] W. Dong, Y. Wang, and M. Zhou, "A latent space-based estimation of distribution algorithm for large-scale global optimization," *Soft Comput.*, vol. 23, pp. 4593–4615, Jul. 2019.

[41] R. Santana, "Estimation of distribution algorithms with Kikuchi approximations," *Evol. Comput.*, vol. 13, no. 1, pp. 67–97, 2005.

[42] S. Shakya and J. McCall, "Optimization by estimation of distribution with DEUM framework based on Markov random fields," *Int. J. Autom. Comput.*, vol. 4, no. 3, pp. 262–272, 2007.

[43] S. Shakya and R. Santana, *Markov Networks in Evolutionary Computation*. Heidelberg, Germany: Springer, 2012.

[44] E. Pira, "Using Markov chain based estimation of distribution algorithm for model-based safety analysis of graph transformation," *J. Comput. Sci. Technol.*, vol. 36, pp. 839–855, Jul. 2021.

[45] M. Soto, Y. González-Fernández, and A. Ochoa, "Modelling with copulas and vines in estimation of distribution algorithms," *Investigación Operacional*, vol. 36, no. 1, pp. 1–23, 2015.

[46] T. K. Paul and H. Iba, "Reinforcement learning estimation of distribution algorithm," in *Genetic and Evolutionary Computation* (Lecture Notes in Computer Science 2724). Heidelberg, Germany: Springer, 2003, pp. 1259–1270.

[47] L. Martí, J. García, A. Berlanga, and J. M. Molina, "Multi-objective optimization with an adaptive resonance theory-based estimation of distribution algorithm," *Ann. Math. Artif. Intell.*, vol. 68, no. 4, pp. 247–273, 2013.

[48] L. Martí, J. García, A. Berlanga, and J. M. Molina, "MONEDA: Scalable multi-objective optimization with a neural network-based estimation of distribution algorithm," *J. Global Optim.*, vol. 66, pp. 729–768, Mar. 2016.

[49] L. Bao, X. Sun, D. Gong, and Y. Zhang, "Multisource heterogeneous user-generated contents-driven interactive estimation of distribution algorithms for personalized search," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 844–858, Oct. 2022.

[50] M. Probst, F. Rothlauf, and J. Grahl, "Scalability of using restricted Boltzmann machines for combinatorial optimization," *Eur. J. Oper. Res.*, vol. 256, no. 2, pp. 368–383, 2017.

[51] V. A. Shim, K. C. Tan, C. Y. Cheong, and J. Y. Chia, "Enhancing the scalability of multi-objective optimization via restricted Boltzmann machine-based estimation of distribution algorithm," *Inf. Sci.*, vol. 248, pp. 191–213, Nov. 2013.

[52] M. Probst and F. Rothlauf, "Harmless overfitting: Using denoising autoencoders in estimation of distribution algorithms," *J. Mach. Learn. Res.*, vol. 21, no. 78, pp. 1–31, 2020.

[53] S. Bhattacharjee, "Variational autoencoder based estimation of distribution algorithms and applications to individual based ecosystem modelling using EcoSim," Ph.D. dissertation, Dept. Comput. Sci., Univ. Windsor, Windsor, ON, Canada, 2019.

[54] J.-H. Jeong, E. Lee, J.-H. Lee, and C.W. Ahn, "Multi-objective deep network-based estimation of distribution algorithm for music composition," *IEEE Access*, vol. 10, pp. 71973–71985, 2022.

[55] M. Probst, "Generative adversarial networks in estimation of distribution algorithms for combinatorial optimization," 2016, *arXiv:1509.09235v2*.

[56] D. Cucci, L. Malagó, and M. Matteucci, "Variable transformations in estimation of distribution algorithms," in *Parallel Problem Solving from Nature* (Lecture Notes in Computer Science 7491). Heidelberg, Germany: Springer, 2012, pp. 428–437.

[57] C. González, J. A. Lozano, and P. Larrañaga, "Analyzing the PBIL algorithm by means of discrete dynamical systems," *Complex Syst.*, vol. 12, no. 4, pp. 465–479, 2001.

[58] T. Chen, K. Tang, G. Chen, and X. Yao, "Analysis of computational time of simple estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 1–22, Feb. 2010.

[59] B. Doerr and M. S. Krejca, "A simplified run time analysis of the univariate marginal distribution algorithm on LeadingOnes," *Theor. Comput. Sci.* vol. 851, pp. 121–128, Jan. 2021.

[60] C. González, J. A. Lozano, and P. Larrañaga, "Mathematical modelling of UMDAc algorithm with tournament selection. Behaviour on linear and quadratic functions," *Int. J. Approx. Reason.*, vol. 31, no. 4, pp. 313–340, 2002.

[61] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 127–136, Apr. 2004.

[62] B. Doerr and W. Zheng, "Sharp bounds for genetic drift in estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 2, no. 6, pp. 1140–1149, Dec. 2020.

[63] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Burlington, MA, USA: Morgan Kaufmann, 1988.

[64] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[65] M. Maathuis, M. Drton, S. Lauritzen, and M. Wainwright, *Handbook of Graphical Models*. Boca Raton, FL, USA: CRC Press, 2019.

[66] C. Bielza and P. Larrañaga, *Data-Driven Computational Neuroscience. Machine Learning and Statistical Models*. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[67] R. Shachter and C. Kenley, "Gaussian influence diagrams," *Manag. Sci.*, vol. 35, no. 5, pp. 527–550, 1989.

[68] D. Geiger and D. Heckerman, "Learning Gaussian networks," in *Proc. 10th Int. Conf. Uncertainty Artif. Intell.*, 1994, pp. 235–243.

[69] R. Neapolitan, *Learning Bayesian Networks*. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

[70] R. Daly, Q. Shen, and S. Aitken, "Learning Bayesian networks: Approaches and issues," *Knowl. Eng. Rev.*, vol. 26, no. 2, pp. 99–157, 2011.

[71] M. Scanagatta, A. Salmerón, and F. Stella, "A survey on Bayesian network structure learning from data," *Progr. Artif. Intell.*, vol. 8, pp. 425–439, May 2019.

[72] O. Spirtes and C. Glymour, "An algorithm FOS fast recovey of sparse causal graphs," *Social Sci. Comput. Rev.*, vol. 90, no. 1, pp. 62–72, 1991.

[73] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. 19, no. 6, pp. 716–723, Dec. 1974.

[74] G. Schwarz, "Estimating the dimension of a model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, 1978.

[75] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, pp. 309–347, Oct. 1992.

[76] D. Heckerman, D. Geiger, and D. M. Chickering, "Learning Bayesian networks: The combination of knowledge and statistical data," *Mach. Learn.*, vol. 20, pp. 197–243, Sep. 1995.

[77] M. Henrion, "Propagating uncertainty in Bayesian networks by probabilistic logic sampling," in *Uncertainty in Artificial Intelligence*, vol. 2. Amsterdam, The Netherlands: North-Holland, 1988, pp. 149–163.

[78] R. M. Fung and K. C. Chang, "Weighing and integrating evidence for stochastic simulation in Bayesian networks," in *Uncertainty in Artificial Intelligence*, vol. 5. Amsterdam, The Netherlands: North-Holland, 1990, pp. 209–220.

[79] J. Pearl, "Evidential reasoning using stochastic simulation of causal models," *Artif. Intell.*, vol. 32, no. 2, pp. 245–257, 1987.

[80] Q. Dang, W. Gao, and M. Gong, "An efficient mixture sampling model for Gaussian estimation of distribution algorithm," *Inf. Sci.*, vol. 608, pp. 1157–1182, Aug. 2022.

[81] T. Miquélez, E. Bengoetxea, and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers," *Int. J. Appl. Math. Comput. Sci.*, vol. 14, pp. 101–115, Jan. 2004.

[82] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga, "Multiobjective estimation of distribution algorithm based on joint modeling of objectives and variables," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 519–542, Aug. 2014.

[83] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," in *Proc. Genet. Evol. Comput. Conf. Workshop Program*, 2000, pp. 201–204.

[84] P. A. N. Bosman and J. Grahl, "Matching inductive search bias and problem structure in continuous estimation-of-distribution algorithms," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1246–1264, 2008.

[85] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions. I. Binary parameters," in *Parallel Problem Solving From Nature* (Lecture Notes in Computer Science 1411). Heidelberg, Germany: Springer, 1996, pp. 178–187.

[86] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-94-163, 1994.

[87] M. Sebag and A. Ducoulombier, "Extending population-based incremental learning to continuous spaces," in *Parallel Problem Solving from Nature* (Lecture Notes in Computer Science 1498). Heidelberg, Germany: Springer, 1998, pp. 418–427.

[88] G. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," in *Proc. IEEE Conf. Evol. Comput.*, 1998, pp. 523–528.

[89] B. Doerr and M. Dufay, "General univariate estimation-of-distribution algorithms," in *Proc. 17th Int. Conf. Parallel Problem Solving Nat.*, 2022, pp. 470–484.

[90] J. S. De Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 9, 1997, pp. 424–430.

[91] S. Baluja and S. Davies, "Combining multiple optimization runs with optimal dependency trees," Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-97-157, 1997.

[92] M. Pelikan and H. Mühlenbein, "The bivariate marginal distribution algorithm," *Advances in Soft Computing-Engineering Design and Manufacturing*. London, U.K.: Springer, 1999, pp. 521–535.

[93] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Proc. 2nd Int. Symp. Artif. Intell.*, 1999, pp. 332–339.

[94] P. Larrañaga, R. Etxeberria, J. A. Lozano, and J. M. Peña, "Combinatorial optimization by learning and simulation of Bayesian networks," in *Proc. 16th Conf. Uncertainty Artif. Intell.*, 2000, pp. 343–352.

[95] W. Dong, T. Chen, P. Tino, and X. Yao, "Scaling up estimation of distribution algorithms for continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 6, pp. 797–822, Dec. 2013.

[96] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. Genet. Evol. Comput. Conf.*, vol. 1, 1999, pp. 525–532.

[97] H. Mühlenbein and T. Mahning, "FDA—A scalable evolutionary algorithm for the optimization of additively decomposed functions," *Evol. Comput.*, vol. 7, no. 4, pp. 353–376, Dec. 1999.

[98] W. Dong and X. Yao, "Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms," *Inf. Sci.*, vol. 178, no. 15, pp. 3000–3023, 2008.

[99] Y. Liang, Z. Ren, X. Yao, Z. Feng, A. Chen, and W. Guo, "Enhancing Gaussian estimation of distribution algorithm by exploiting evolution direction with archive," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 140–152, Jan. 2020.

[100] H. Xu, J. Yang, P. Jia, and Y. Ding, "Effective structure learning for estimation of distribution algorithms via L1-regularized Bayesian networks," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 1, p. 17, 2013.

[101] H. Karshenas, R. Santana, C. Bielza, and P. Larrañaga, "Regularized continuous estimation of distribution algorithms," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2412–2432, 2013.

[102] P. A. N. Bosman and D. Thierens, "IDEAs based on the normal kernels probability density function," Dept. Comput. Sci., Utrecht Univ., Utrecht, The Netherlands, Rep. UU-CS-2000-11, 2000.

[103] P. A. N. Bosman and D. Thierens, "Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms," *Int. J. Approx. Reason.*, vol. 31, no. 3, pp. 259–289, 2002.

[104] J. M. Peña, J. A. Lozano, and P. Larrañaga, "Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks," *Evol. Comput.*, vol. 13, no. 1, pp. 43–66, 2005.

[105] G. Harik, "Linkage learning via probabilistic modelling in the ECGA," Univ. Illinois Urbana-Champaign, Champaign, IL, USA, Rep. 99010, 1999.

[106] C. A. Coello-Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York, NY, USA: Springer, 2007.

[107] R. Santana, C. Bielza, J. A. Lozano, and P. Larrañaga, "Mining probabilistic models learned by EDAs in the optimization of multi-objective problems," in *Proc. 11th Annu. Conf. Genet. Evol. Comput.*, 2009, pp. 445–452.

[108] M. Costa and E. Minisci, "MOPED: A multi-objective Parzen-based estimation of distribution algorithm for continuous problems," in *Evolutionary Multi-Criterion Optimization* (Lecture Notes in Computer Science 2632). Heidelberg, Germany: Springer, 2003, pp. 282–294.

[109] E. Bengoetxea, P. Larrañaga, C. Bielza, and J. A. Fernández del Pozo, "Optimal row and column ordering to improve table interpretation using estimation of distribution algorithms," *J. Heurist.*, vol. 17, no. 5, pp. 567–588, 2011.

[110] J. Bertin, *Graphics and Graphic Information Processing*. Berlin, Germany: Walter de Gruyter, 1981.

[111] H. Walker and W. Durost, *Statistic Tables: Their Structure and Use*, Bureau of Publications, Columbia Univ., New York, NY, USA, 1936.

[112] J. L. Flores, I. Inza, and P. Larrañaga, "Wrapper discretization by means of estimation of distribution algorithms," *Intell. Data Anal. J.*, vol. 11, no. 5, pp. 525–546, 2007.

[113] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.

[114] X. Zhang, B. Xue, G. Sui, and J. Cui, "Association rule mining based on estimation of distribution algorithm for blood indices," in *Proc. Int. Conf. Comput. Netw., Electron. Autom.*, 2017, pp. 59–65.

[115] X. Li, S. Mabu, H. Zhou, K. Shimada, and K. Hirasawa, "Genetic network programming with estimation of distribution algorithms for class association rule mining in traffic prediction," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[116] E. Fix and J. L. Hodges, "Discriminatory analysis–non parametric discrimination: Consistency properties," USAF School Aviation Med., Wright-Patterson AFB, OH, USA, Rep. 4, 1951.

[117] I. Inza, P. Larrañaga, and B. Sierra, "Feature weighting for nearest neighbour by EDAs," in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad. Publ., 2002, pp. 295–311.

[118] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[119] H. E. L. Cagnini, R. C. Barros, and M. P. Basgalupp, "Estimation of distribution algorithms for decision-tree induction," in *Proc. Congr. Evol. Comput.*, 2017, pp. 2022–2029.

[120] J. Holland, "A mathematical framework for studying learning in classifier systems," *Physica D*, vol. 2, no. 1, pp. 307–317, 1986.

[121] K. De Jong and W. Spears, "Learning concept classification rules using genetic algorithms," in *Proc. 12th Int. Joint Conf. Artif. Intell.*, 1991, pp. 651–656.

[122] B. Sierra, E. A. Jiménez, I. Inza, and P. Larrañaga, "Rule induction by estimation of distribution algorithms," in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad., 2002, pp. 313–322.

[123] J. Yang, H. Xu, and P. Jia, "Effective search for Pittsburgh learning classifier systems via estimation of distribution algorithms," *Inf. Sci.*, vol. 198, pp. 100–117, Sep. 2012.

[124] J. Yang, H. Xu, and P. Jia, "Effective search for genetic-based machine learning systems via estimation of distribution algorithms and embedded feature reduction techniques," *Neurocomputing*, vol. 113, pp. 105–121, Aug. 2013.

[125] L. DelaOssa, J. A. Gámez, and J. M. Puerta, "Learning weighted linguistic fuzzy rules by using specifically-tailored hybrid estimation of distribution algorithms," *Int. J. Approx. Reason.*, vol. 50, no. 3, pp. 541–560, 2009.

[126] B.E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, 1992, pp. 144–152.

[127] L.C. Padierna, M. Carpio, A. Rojas, H. Puga, R. Baltazar, and H. Fraire, "Hyper-parameter tuning for support vector machines by estimation of distribution algorithms," in *Nature-Inspired Design of Hybrid Intelligent Systems*, vol. 667. Cham, Switzerland: Springer, 2017, pp. 787–800.

[128] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, Dec. 1943.

[129] S. Baluja, "An empirical comparison of seven iterative and evolutionary function optimization heuristics," Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-95-193, 1995.

[130] M.R. Gallagher, "Multi-layer perceptron error surfaces: Visualization, structure and modelling," Ph.D. dissertation, Comput. Sci. Electr. Eng., Univ. Queensland, Herston, QLD, Australia, 2000.

[131] C. Cotta, E. Alba, R. Sagarna, and P. Larrañaga, "Adjusting weights in artificial neural networks using evolutionary algorithms," in *Estimation of Distribution Algorihtms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad., 2002, pp. 361–377.

[132] Y. Chen and A. Abraham, "Estimation of distribution algorithms for optimization of neural networks for intrusion detection," *Proc. 8th Int. Conf. Artif. Intell. Soft Comput.*, 2006, pp. 9–18.

[133] Q. Li, Y. Du, Z. Liu, Z. Zhou, G. Lu, and Q. Chen, "Drought prediction in the Yunnan–Guizhou plateau of China by coupling the estimation of distribution algorithm and the extreme learning machine," *Nat. Hazards*, vol. 113, pp. 1635–1661, May 2022.

[134] E. Galić and M. Höhfeld, "Improving the generalization performance of multi-layer-perceptrons with population-based incremental learning," in *Parallel Problem Solving From Nature* (Lecture Notes in Computer Science 1141). Heidelberg, Germany: Springer, 1996, pp. 740–750.

[135] G. Holker and M. V. dos Santos, "Toward an estimation of distribution algorithm for the evolution of artificial neural networks," in *Proc. 3rd Conf. Comput. Sci. Softw. Eng.*, 2010, pp. 17–22.

[136] E. Cantú-Paz, "Pruning neural networks with distribution estimation algorithms," in *Proc. Genet. Evol. Comput. Conf.*, 2003, pp. 790–800.

[137] J. Peralta-Donate, X. Li, G. G. Sánchez, and A. S. de Miguel, "Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm," *Neural Comput. Appl.*, vol. 22, pp. 11–20, Jan. 2013.

[138] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[139] J.-Y. Li, Z.-H. Zhan, J. Xu, S. Kwong, and J. Zhang, "Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2338–2352, May 2023.

[140] O. G. Toledano-López, J. Madera, H. González, and A. Simón-Cuevas, "A hybrid method based on estimation of distribution algorithms to train convolutional neural networks for text categorization," *Pattern Recognit. Lett.*, vol. 160, pp. 105–111, Aug. 2022.

[141] Q. Xu, A. Liu, X. Yuan, Y. Song, C. Zhang, and Y. Li, "Random mask-based estimation of the distribution algorithm for stacked autoencoder one-step pre-training," *Comput. Ind. Eng.*, vol. 158, Aug. 2021, Art. no. 107400.

[142] J. Berkson, "Application of the logistic function to bio-assay," *J. Amer. Stat. Assoc.*, vol. 39, no. 227, pp. 357–365, 1944.

[143] V. Robles, C. Bielza, P. Larrañaga, S. González, and L. Ohno-Machado, "Optimizing logistic regression coefficients for discrimination and calibration using estimation of distribution algorithms," *TOP*, vol. 16, no. 2, pp. 345–366, 2008.

[144] C. Bielza, V. Robles, and P. Larrañaga, "Regularized logistic regression without a penalty term: An application to cancer classification with microarray data," *Expert Syst. Appl.*, vol. 38, no. 5, pp. 5110–5118, 2011.

[145] M. L. Minsky, "Step toward artificial intelligence," *Proc. IRE*, vol. JRPROC-49, no. 1, pp. 8–30, Jan. 1961.

[146] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 50, nos. 1–2, pp. 95–125, 1997.

[147] C. Bielza, and P. Larrañaga, "Discrete Bayesian network classifiers," *ACM Comput. Surveys*, vol. 47, no. 1, 2014, Art. no. 5.

[148] V. Robles, P. Larrañaga, J. M. Peña, E. Menasalvas, and M. S. Pérez, "Interval estimation Naïve Bayes," in *Advances in Intelligent Data Analysis* (Lecture Notes in Computer Science 2810). Heidelberg, Germany: Springer, 2003, pp. 143–154.

[149] M. Pazzani, "Constructive induction of Cartesian product attributes," in *Proc. Inf., Stat. Induct. Sci. Conf.*, 1996, pp. 66–77.

[150] V. Robles et al., "Bayesian networks multi-classifiers for protein secondary structure prediction," *Artif. Intell. Med.*, vol. 31, no. 2, pp. 117–136, 2004.

[151] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[152] I. Mendialdua, A. Arruti, E. Jauregi, E. Lazkano, and B. Sierra, "Classifier subset selection to construct multi-classifiers by means of estimation of distribution algorithms," *Neurocomputing*, vol. 157, pp. 46–60, Jul. 2015.

[153] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[154] H. E. L. Cagnini, M. P. Basgalupp, and R. C. Barros, “Increasing boosting effectiveness with estimation of distribution algorithms,” in *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1–8.

[155] H. Bunke, “18 Parameter estimation in nonlinear regression models,” in *Handbook of Statistics*, vol. 1. Amsterdam, The Netherlands: Elsevier, 1980, pp. 593–615.

[156] L. M. Torres-Treviño, “Symbolic regression using α-β operators and estimation of distribution algorithms: Preliminary results,” in *Proc. 13th Genet. Evol. Comput. Conf.*, 2011, pp. 647–654.

[157] M. A. Sotelo-Figueroa, A. Hernández-Aguirre, A. Espinal, J. A. Soria-Alcaraz, and J. Ortiz-López, “Symbolic regression by means of grammatical evolution with estimation distribution algorithms as search engine,” in *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications. Studies in Computational Intelligence*, vol. 749. Cham, Switzerland: Springer, 2018, pp. 169–180.

[158] D. Wittenberg and F. Rothlauf, “Denoising autoencoder genetic programming for real-world symbolic regression,” in *Proc. Genet. Evol. Comput. Conf.*, 2022, pp. 612–614.

[159] C. Jin and S.-W. Jin, “Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms,” *Appl. Soft Comput.*, vol. 15, pp. 113–120, Feb. 2014.

[160] P. M. Lewis, “The characteristic selection problem in recognition systems,” *IRE Trans. Inf. Theory*, vol. TIT-8, no. 2, pp. 171–178, Feb. 1962.

[161] I. Inza, P. Larrañaga, R. Etxeberria, and B. Sierra, “Feature subset selection by Bayesian network–based optimization,” *Artif. Intell.*, vol. 123, pp. 157–184, Oct. 2000.

[162] I. Inza, P. Larrañaga, and B. Sierra, “Feature subset selection by Bayesian networks: A comparison with genetic and sequential algorithms,” *Int. J. Approx. Reason.*, vol. 27, pp. 143–164, Aug. 2001.

[163] E. Cantú-Paz, “Feature subset selection by estimation of distribution algorithms,” in *Proc. 4th Annu. Conf. Genet. Evol. Comput.*, 2002, pp. 303–310.

[164] G. Neuman and D. Cairns, “Applying a hybrid targeted estimation of distribution algorithm to feature selection problems,” in *Proc. 5th Int. Joint Conf. Comput. Intell.*, 2013, pp. 136–143.

[165] C. Bielza, V. Robles, and P. Larrañaga, “Estimation of distribution algorithms as logistic regression regularizers of microarray classifiers,” *Methods Inf. Med.*, vol. 48, no. 3, pp. 236–241, 2009.

[166] S. Maza and M. Touahria, “Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms,” *Appl. Intell.*, vol. 49, pp. 4237–4257, May 2019.

[167] H. Chen, S. Yuan, and K. Jiang, “Fitness approximation in estimation of distribution algorithms for feature selection,” in *Advances in Artificial Intelligence* (Lecture Notes in Computer Science 3809), Heidelberg, Germany: Springer, 2005, pp. 904–909.

[168] T. Sorensen, “A method for establishing groups of equal amplitude in plant sociology based on similarity of species contents and its application to analyses of the vegetation on Danish commons,” *Biologiske Skrifter*, vol. 5, pp. 1–34, 1948.

[169] J. Fan, “OPE-HCA: An optimal probabilistic estimation approach for hierarchical clustering algorithm,” *Neural Comput. Appl.*, vol. 31, pp. 2095–2105, Jul. 2019.

[170] E. W. Forgy, “Cluster analysis of multivariate data: Efficiency versus interpretability of classification,” *Biometrics*, vol. 21, no. 3, pp. 768–769, 1965.

[171] J. Roure, P. Larrañaga, and R. Sangüesa, “An empirical comparison between *k*-means, GAs and EDAs in partitional clustering,” in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad., pp. 343–360, 2002.

[172] L. Kaufman, and P. J. Rousseeuw, “Clustering by means of medoids,” in *Statistical Data Analysis Based on the L_1 Norm and Related Methods*. Amsterdam, The Netherlands: North-Holland, 1997, pp. 405–416.

[173] H. E. L. Cagnini, R. C. Barros, C. V. Quevedo, and M. P. Basgalupp, “Medoid-based data clustering with estimation of distribution algorithms,” in *Proc. 31st Annu. ACM Symp. Appl. Comput.*, 2016, pp. 112–115.

[174] B. J. Frey and D. Dueck, “Clustering by passing messages between data points,” *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[175] R. Santana, C. Bielza, and P. Larrañaga, “Affinity propagation enhanced by estimation of distribution algorithms,” in *Proc. Genet. Evol. Conf.*, 2011, pp. 331–338.

[176] H. E. L. Cagnini, and R. C. Barros, “PASCAL: An EDA for parameterless shape-independent clustering,” in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3434–3440.

[177] A. S. G. Meiguins, R. C. Limão, B. S. Meiguins, F. S. Samuel Jr., and A. A. Freitas, “AutoClustering: An estimation of distribution algorithm for the automatic generation of clustering algorithms,” in *Proc. IEEE World Congr. Comput. Intell.*, 2012, pp. 1–7.

[178] A. P. Demspter, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. Royal Stat. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[179] S. Lauritzen, “The EM algorithm for graphical association models with missing data,” *Comput. Stat. Data Anal.*, vol. 19, pp. 191–201, Feb. 1995.

[180] J. M. Peña, J. A. Lozano, and P. Larrañaga, “Unsupervised learning of Bayesian networks via estimation of distribution algorithms: An application to gene expression data clustering,” *Int. J. Uncertainty, Fuzziness Knowl. Based Syst.*, vol. 12, pp. 63–82, Jan. 2004.

[181] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA, USA: MIT Press, 1998.

[182] H. Handa and T. Nishimura, “Solving reinforcement learning problems by using estimation of distribution algorithms,” in *Proc. 2nd Int. Conf. Soft Comput. 9th Intell. Syst. Int. Symp. Adv. Intell. Syst.*, 2008, pp. 676–681.

[183] J. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proc. 19th Int. Conf. Mach. Learn.*, 2001, pp. 282–289.

[184] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[185] Y. Du, J.-Q. Li, X.-L. Chen, P.-Y. Duan, and Q.-K. Pan, “Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 7, no. 4, pp. 1036–1050, Aug. 2023.

[186] S. L. Lauritzen and D. J. Spiegelhalter, “Local computations with probabilities on graphical structures and their application to expert systems,” *J. Royal Stat. Soc. B, Methodol.*, vol. 50, no. 2, pp. 157–224, 1988.

[187] G. F. Cooper, “The computational complexity of probabilistic inference using Bayesian belief networks,” *Artif. Intell.*, vol. 42, nos. 2–3, pp. 393–405, 1990.

[188] P. Larrañaga, C. M. H. Kuijpers, M. Poza, and R. H. Murga, “Decomposing Bayesian networks: Triangulation of the moral graph with genetic algorithms,” *Stat. Comput.*, vol. 7, no. 1, pp. 19–34, 1997.

[189] T. Romero and P. Larrañaga, “Triangulation of Bayesian networks with recursive estimation of distribution algorithms,” *Int. J. Approx. Reason.*, vol. 50, no. 3, pp. 472–484, 2009.

[190] L. M. de Campos, J. A. Gámez, P. Larrañaga, S. Moral, and T. Romero, “Partial abductive inference in Bayesian networks: An empirical comparison between GAs and EDAs,” in *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Amsterdam, The Netherlands: Kluwer Acad., pp. 323–341, 2002.

[191] R. Blanco, I. Inza, and P. Larrañaga, “Learning Bayesian networks in the space of structures by estimation of distribution algorithms,” *Int. J. Intell. Syst.*, vol. 18, no. 2, pp. 205–220, 2003.

[192] G. Thibault, S. Bonnevay, and A. Aussem, “Learning Bayesian network structures by estimation of distribution algorithms: An experimental analysis,” in *Proc. IEEE Int. Conf. Digital Inf. Manag.*, 2007, pp. 127–132.

[193] D. W. Kim, S. Ko, and B. Y. Kang, “Structure learning of Bayesian networks by estimation of distribution algorithms with transpose mutation,” *J. Appl. Res. Technol.*, vol. 11, no. 4, pp. 586–596, 2013.

[194] S. Fukuda, Y. Yamanaka, and T. Yoshihiro, “A probability-based evolutionary algorithm with mutations to learn Bayesian network,” *Int. J. Artif. Intell. Interact. Multimedia*, vol. 3, no. 1, pp. 7–13, 2014.

[195] T. Romero, P. Larrañaga, and B. Sierra, “Learning Bayesian networks in the space of orderings with estimation of distribution algorithms,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 18, no. 4, pp. 607–625, 2004.

[196] L. Rabiner and B. Juang, “An introduction to hidden Markov models,” *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, Jan. 1986.

[197] B. Maxwell and S. Anderson, “Training hidden Markov models using population-based learning,” in *Proc. 1st Annu. Genet. Evol. Comput. Conf.*, vol. 1, 1999, pp. 944–950.

[198] I. Inza, M. Merino, P. Larrañaga, J. Quiroga, B. Sierra, and M. Girala, “Feature subset selection by genetic algorithms and estimation of distribution algorithms: A case study in the survival of cirrhotic patients treated with TIPS,” *Artif. Intell. Med.*, vol. 23, no. 2, pp. 187–205, 2001.

[199] R. Armañanzas et al., "Peakbin selection in mass spectrometry data using a consensus approach with estimation of distribution algorithms," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 8, no. 3, pp. 760–774, May/Jun. 2011.

[200] Y. Saeys, S. Degroeve, D. Aeyels, Y. van de Peer, and P. Rouzé, "Fast feature selection using a simple estimation of distribution algorithm: A case study on splice site prediction," *Bioinformatics*, vol. 19, no. 2, pp. II179–II188, 2003.

[201] M. Ayodele, "Application of estimation of distribution algorithm for feature selection," in *Proc. 19th Genet. Evol. Comput. Conf.*, 2019, pp. 43–44.

[202] C. Cano, F. García, F. J. López, and A. Blanco, "Intelligent system for the analysis of microarray data using principal components and estimation of distribution algorithms," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4654–4663, 2009.

[203] H. Handa, "EDA-RL: Estimation of distribution algorithms for reinforcement learning problems," in *Proc. 11th Annu. Conf. Genet. Evol. Comput.*, 2009, pp. 405–412.

[204] N. K. Kitson, A. C. Constantinou, Z. Guo, Y. Liu, and K. Chobtham, "A survey of Bayesian network structure learning," *Artif. Intell. Rev.*, vol. 56, pp. 8721–8814, Jan. 2023.

[205] D. Atienza, C. Bielza, and P. Larrañaga, "Semiparametric Bayesian networks," *Inf. Sci.*, vol. 584, pp. 564–582, Jan. 2022.

[206] D. Atienza, P. Larrañaga, and C. Bielza, "Hybrid semiparametric Bayesian networks," *TEST*, vol. 31, pp. 299–327, Jun. 2022.

[207] V. P. Soloviev, C. Bielza, and P. Larrañaga, "Semiparametric estimation of distribution algorithms for continuous optimization," *IEEE Trans. Evol. Comput.*, early access, Jun. 29, 2023, doi: 10.1109/TEVC.2023.3290670.

[208] J. Ceberio, B. Doerr, C. Witt, and V. P. Soloviev, "Estimation-of-distribution algorithms: Theory and applications," *Dagstuhl Rep.*, vol. 12, no. 5, pp. 17–36, 2022.

[209] G. Ochoa, K. M. Malan, and C. Blum, "Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics," *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107492.

[210] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," in *Wiley Interdisciplinary Reviews—Data Mining and Knowledge Discovery*, vol. 5. Hoboken, NJ, USA: Wiley, 2015, pp. 216–233.

[211] J. A. Lozano and P. Larrañaga, "Applying genetic algorithms to search for the best hierarchical clustering of a dataset," *Pattern Recognit. Lett.*, vol. 20, no. 9, pp. 911–918, 1999.

[212] J. A. Castellanos-Garzón, C. A. García, and L. A. Miguel-Quintales, "An evolutionary hierarchical clustering method with a visual validation tool," in *Proc. Int. Work Conf. Artif. Neural Netw.*, 2009, pp. 367–374.

Pedro Larrañaga (Senior Member, IEEE) received the M.Sc. degree in mathematics (statistics) from the University of Valladolid, Valladolid, Spain, in 1981, and the Ph.D. degree in computer science from the University of the Basque Country, Bilbo, Spain, in 1995 (Excellence Award).

He is a Professor of Computer Science and Artificial Intelligence with the Universidad Politécnica de Madrid, Madrid, Spain. He has published over 200 papers in high-impact factor journals, and he has supervised over 30 Ph.D. theses. His research interests include the areas of probabilistic graphical models, metaheuristics for optimization, classification models, and real applications, such as biomedicine, bioinformatics, neuroscience, industry 4.0, and sports.

Prof. Larrañaga received the 2013 Spanish National Prize in Computer Science, the Spanish Association for Artificial Intelligence Prize in 2018, and the 2020 Machine Learning Award from Amity University, India. He has been a Fellow of the European Association for Artificial Intelligence since 2012, the Academia Europaea since 2018, and the Asia–Pacific Artificial Intelligence Association since 2021, a member of the Jakiunde, the Academy of Sciences, Arts, and Letters of the Basque Country since 2022, and an ELLIS Fellow since 2023.

Concha Bielza (Senior Member, IEEE) received the M.Sc. degree in mathematics from the Universidad Complutense de Madrid, Madrid, Spain, in 1989, and the Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, in 1996 (Extraordinary Doctorate Award).

She is a Professor of Statistics and Operations Research with the Artificial Intelligence Department, Universidad Politécnica de Madrid. She has published more than 150 papers in impact factor journals and has supervised 21 Ph.D. theses. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, machine learning, anomaly detection, and real applications, such as biomedicine, bioinformatics, neuroscience, and industry.

Prof. Bielza was awarded the 2014 UPM Research Prize and the 2020 Machine Learning Award from Amity University, India. She has been an ELLIS Fellow since 2023.