Evolutionary Bayesian Classifier-Based Optimization in Continuous Domains

Teresa Miquélez, Endika Bengoetxea, and Pedro Larrañaga

Intelligent Systems Group
University of the Basque Country, Computer Engineering Faculty
P.O. Box 649, E-20018 San Sebastian, Spain
`http://www.sc.ehu.es/isg`
`{teresa, endika, ccplamup}@si.ehu.es`

Abstract. In this work, we present a generalisation to continuous domains of an optimization method based on evolutionary computation that applies Bayesian classifiers in the learning process. The main difference between other estimation of distribution algorithms (EDAs) and this new method –known as Evolutionary Bayesian Classifier-based Optimization Algorithms (EBCOAs)– is the way the fitness function is taken into account, as a new variable, to generate the probabilistic graphical model that will be applied for sampling the next population.

We also present experimental results to compare performance of this new method with other methods of the evolutionary computation field like evolution strategies, and EDAs. Results obtained show that this new approach can at least obtain similar performance as these other paradigms[1].

1 Introduction

Evolutionary computation techniques have undergo a great development with the extensive use of paradigms such as Genetic Algorithms (GAs) [5,8], Evolution Strategies (ES) [6], and Estimation of Distribution Algorithms (EDAs) [10,13,17,18,20]. Many other evolutionary computation paradigms have also been recently proposed such as Learnable Evolution Model (LEM) [14], and Evolutionary Bayesian Classifier-based Optimization Algorithms (EBCOAs) [16].

The main difference between them is the way of improving the population of individuals, in order to obtain fitter solutions to a concrete optimization problem. In GAs and ES the evolution is based on using crossover and mutation operators, without expressing explicitly the characteristics of the selected individuals within a population. EDAs

[1] This paper has been partially supported by the Spanish Ministry of Science and Technology and the Basque Government with grants TIN2004-21428-E and SAIOTEK-S-PE04UN25 respectively. The authors would like to thank Alexander Mendiburu, and the rest of the members of the Intelligent Systems Group of the University of the Basque Country for their useful advise and contribution to this work. We would also like to thank Francisco Herrera and the rest of the Soft Computing and Intelligent Information Systems research group of the University of Granada for their support with the use of CMA-ES.

take into account these explicit characteristics by considering the interdependencies between the different variables that form an individual, learning a probabilistic graphical model to represent them. The approach of LEM, EBCOAs, and other similar proposals in this direction [12] is different in the sense of applying classification techniques to build models that represent the main characteristics for which an individual in the population appears in the group of the best (or worst) individuals within the population of a generation. In that sense, these paradigms also take into consideration less fit individuals of the population in order to enhance and estimate the differences between the best and worst cases. This knowledge is later used for instantiating new individuals for a new population.

This paper introduces EBCOAs (Evolutionary Bayesian Classifier-based Optimization Algorithms) for continuous domains, which are motivated as an improvement of EDAs for the need to avoid them to fall into local optima in very complex optimization problems. EBCOAs evolve to a fitter generation by constructing models that take into account more differences than simply a subset of the fittest individuals. Continuous EBCOAs generalize the ideas presented in [16] by supervised classification paradigms in the form of conditional Gaussian networks to improve the generation of individuals every generation.

2 Bayesian Classifiers for Continuous Domains

In EBCOAs for continuous domains, the classifiers are based on the learning of probabilistic graphical models, more concretely Bayesian classifiers based on conditional Gaussian networks [11]. The literature contains several examples of classifiers combined with evolutionary computation techniques. One of the first examples is the LEM algorithm [14] which makes use of rules to build a classifiers that records the main differences between the groups of best and worst individuals of each population.

The *supervised classification* problem with n continuous predictor variables consists in assigning any vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n$ to one of the $|C|$ classes of a class variable C that is known. The class value is denoted by c and therefore we have that $c \in \{1, 2, \ldots, |C|\}$. As a result, a classifier in supervised classification is defined as a function $\gamma : (x_1, \ldots, x_n) \rightarrow \{1, 2, \ldots, |C|\}$ that assigns class labels to observations.

Next, we provide some examples of the classifiers from the ones considering less interdependencies to the ones considering most of them.

2.1 Naive Bayes

The Bayesian classifier that considers all the variables $X_1, \ldots, X_n$ to be conditionally independent given the class value C is known as naive Bayes [15]. In this case, the probabilistic graphical model can be considered to be a fixed structure as illustrated in Figure 1(a). In continuous domains it is usual to assume that the joint density function follows a n–dimensional normal distribution, and since independence between the variables –given the class variable C– is assumed, this is factorized by a product of unidimensional and conditionally independent normal densities. Therefore, when classifying a new individual using the naive Bayes classifier we have that:

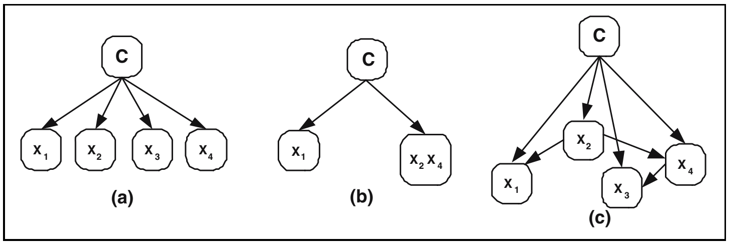

Fig. 1. Example of structures of Bayesian classifiers that can be obtained as a result of the different classification model building algorithms in a problem with four variables $X_1, \ldots, X_4$ and the class-variable C: (a) naive Bayes (b) seminaive Bayes (c) tree augmented naive Bayes.

$$p(C = c|X_1 = x_1, \ldots X_n = x_n) \propto p(c) \cdot f(x_1|c) \cdot f(x_2|c) \cdot \ldots \cdot f(x_n|c)$$

where

$$f(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{ic}} e^{-\frac{1}{2}(\frac{x_{ic}-\mu_{ic}}{\sigma_{ic}})^2}$$

for all $i = 1, \ldots, n$ and $c = 1, \ldots, |C|$ with μ_{ic} and σ_{ic} representing the mean and the standard deviation of $X_i|C = c$ respectively.

In order to apply a naive Bayes classifier in EBCOAs, the estimation of the a priori probability of the class, $p(c)$, as well as the parameters μ_{ic} and σ_{ic} of the conditional density functions, $f(x_i|c)$, are carried out from the database of selected individuals at each generation.

2.2 Seminaive Bayes

The seminaive Bayes classifier [9] provides more complexity than the former since it is able to take into account dependencies between groups of variables. This paradigm represents the variables found to be related as a *fused* node in the conditional Gaussian network, that is the seminaive Bayesian classifier proposed to group some variables in a single node of the structure. Figure 1(b) illustrates the structure of a seminaive Bayesian classifier for a problem with four variables, treating each of these grouped variables as a single super-variable regarding the factorization of the probability distribution. When grouping variables all the inter-dependencies between them are taken into account implicitly in the Bayesian classifier. In a seminaive Bayesian classifier it is also possible to ignore some variables and therefore not to include them in the final probabilistic graphical model, which has the effect of considering these variables not to be relevant for labeling vectors to a particular c class. [19] introduced a greedy algorithm to detect irrelevant as well as dependent variables (susceptible to be grouped) and to propose variables that are likely to be ignored in a Bayesian classifier, although this is described only for discrete domains. We propose to adapt it to the case of continuous domains by grouping dependent continuous variables as a single multidimensional variable in the form of one node in the conditional Gaussian network.

Considering the example in Figure 1(b) to be the seminaive Bayes model structure learned from one supervised classification problem, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the following class:

$$c^* = \arg\max_c p(c) f(x_1|c) f(x_2, x_4|c) \tag{1}$$

Following this approach, in cases in which a variable X_i is estimated to be conditionally independent of the rest given the class variable (such as variable X_1 in the latter example), $f(x_i|c)$ will be computed as

$$f(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{ic}} e^{-\frac{1}{2}(\frac{x_{ic}-\mu_{ic}}{\sigma_{ic}})^2}$$

Analogously, for the case of the dependencies with a number p of grouped variables over a single node in the structure, similarly as variables X_2 and X_4 in our example, the corresponding factor will be assumed to follow a p-dimensional normal distribution for each value of variable C. Considering that $\mathbf{z}_j$ represents the j^{th} group of p variables, we would have that

$$f(\mathbf{z}_j|c) = \frac{1}{\sqrt{(2\pi)^p |\Sigma_{jc}|}} e^{-\frac{1}{2}(\mathbf{z}_j - \boldsymbol{\mu}_{jc})^T \Sigma_{jc}^{-1} (\mathbf{z}_j - \boldsymbol{\mu}_{jc})}.$$

where Σ_{jc} is a $p \times p$ matrix representing the variance-covariance matrix of the j^{th} group of p variables when $C = c$ and $\boldsymbol{\mu}_{jc}$ denotes its corresponding expectation.

2.3 Tree Augmented Naive Bayes

Another example of a Bayesian classifier that is able to take into account different dependencies between variables than the previous seminaive approach is the tree augmented naive Bayes classifier [4]. Its name comes from the fact that the structures obtained as a result of its learning approach have the form of a tree. This algorithm constitutes an adaptation of the Chow-Liu algorithm [3] for predictor continuous variables by estimating the mutual information between two univariate normal distributions.

Figure 1(c) shows the type of structures that could be obtained when applying the tree augmented naive Bayes algorithm for a problem similarly as for the two previous Bayesian classifiers. Following this particular example, an individual $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ will be assigned to the class

$$c^* = \arg\max_c p(c) f(x_1|c, x_2) f(x_2|c) f(x_3|c, x_4) f(x_4|c, x_2) \tag{2}$$

Note that in this case the calculation of $f(x_i|c, x_{k(i)})$ –where $X_{k(i)}$ represents the predictor parent variable of variable X_i in case that this parent exists– can be computed as

$$f(x_i|c, x_{k(i)}) = \frac{p(c) \cdot f(x_i, x_{k(i)}|c)}{p(c) \cdot f(x_{k(i)}|c)} = \frac{f(x_i, x_{k(i)}|c)}{f(x_{k(i)}|c)}$$

3 The Evolutionary Bayesian Classifier-Based Optimization Algorithm Approach

This approach combines Bayesian classifiers such as the ones presented in the previous section and evolutionary computation to solve optimization problems. The main idea is that having a population of solutions for the optimization problem, we will evolve to a next population of fitter individuals by constructing a Bayesian classifier that will represent the main characteristics between the fittest and the least fit individuals. The EBCOA approach contains the following steps:

1. Firstly, the initial population D_0 of R individuals is generated. This initial population is generated similarly as EDAs, usually by assuming an uniform distribution on each variable. Each of the created individuals is evaluated.
2. Secondly, each of the individuals in D_l are given a label $|K| < R$ to classify them following their respective fitness value. This is the supervised classification step, as each of the R individuals is assigned a k label, and as a result the class variable K is created in the database, forming D_l^K.
3. Thirdly, D_l^C is created by selecting from D_l^K only the $|C| \leq |K|$ classes that will be used for building the Bayesian classifier, usually taking into account at least the best and worst classes of individuals uniquely. A similar scheme of selecting individuals with extreme fitness know as Stabilizing Selection is presented in [2]. The rest of the classes in D_l^K could be discarded to facilitate the learning by enhancing the differences between the most distant classes. The individuals which are in $D_l^K \backslash D_l^C$ are simply ignored.
4. A Bayesian classifier is build based on D_l^C by applying techniques such as the ones described in the previous section. This classifier estimates the probability distribution $p_l(c|\boldsymbol{x}) \propto p_l(c)f_l(\boldsymbol{x}|c)$ which represents the probability of any individual $\boldsymbol{x}$ to be classified in the any of the different possible $|C|$ classes.
5. Finally, the new population D_{l+1} constituted by the R new individuals is obtained by carrying out the simulation of the probability distribution $p_l(c)f_l(\boldsymbol{x}|c)$. This step can be performed very similarly as in EDAs[2].

Steps 2, 3, 4 and 5 are repeated until a stopping criterion is satisfied. Examples of stopping conditions are: achieving a fixed number of populations or a fixed number of different evaluated individuals, uniformity in the generated population, and the fact of not obtaining an individual with a better fitness value after a certain number of generations.

The step of learning the Bayesian classifier is the most critical one regarding the performance of EBCOAs in terms of convergence speed and computation time.

4 Experiments on Standard Continuous Optimization Problems

Experiments were carried out in order to test the performance of continuous EBCOAs compared to such of some continuous EDAs and ES. For this comparison, we have chosen continuous EDAs that take into account different number of dependencies between

[2] The reader can find a detailed description on this topic as well as a review of some of the possible techniques that can be applied in this step in [10].

Table 1. Mean results after 10 runs with each algorithm and objective function. The V and E columns represent the best fitness value obtained and the evaluations number respectively.

	Ackley		Griewangk		Rosenbrock		Sphere		SumCan	
$n = 10$	V	E	V	E	V	E	V	E	V	E
EBCOA$_{NB}$	7.7E-6	18116	3.0E-2	33597	4.4E+3	18235	4.4E-6	19632	1.0E+05	34116
EBCOA$_{S_{NB}}$	6.1E-6	11891	4.7E-6	8061	9.0E+0	7821	3.9E-6	7422	1.0E+05	16599
EBCOA$_{TAN}$	6.7E-6	11333	5.3E-6	10495	3.8E+3	6704	5.1E-6	9657	1.0E+05	31163
UMDA$_c$	8.8E-6	23063	5.4E-2	58814	8.7E+0	52589	7.3E-6	15003	1.6E+04	60250
MIMIC$_c$	7.8E-6	23382	8.2E-2	59093	8.7E+0	44968	6.7E-6	15163	1.7E+04	60250
EGNA$_{ee}$	7.9E-6	22983	5.4E-2	58654	8.6E+0	28889	6.7E-6	14884	1.0E+05	37108
EGNA$_{BGe}$	8.5E-6	22904	9.2E-2	54784	8.6E+0	26375	7.0E-6	14884	1.0E+05	37826
CMA-ES	1.9E-7	23962	2.7E-8	14562	3.9E-8	44082	3.7E-8	13802	1.0E+05	45682
$n = 50$										
EBCOA$_{NB}$	5.5E-6	29328	5.8E-6	24619	5.1E+5	2914	3.5E-6	16839	1.0E+05	34036
EBCOA$_{S_{NB}}$	6.9E-6	11851	5.7E-6	9976	4.9E+1	9976	5.7E-6	9976	1.0E+05	27293
EBCOA$_{TAN}$	7.2E-6	10136	5.2E-6	8500	4.9E+1	7343	5.9E-6	10016	1.0E+05	30086
UMDA$_c$	1.6E-5	56819	8.3E-6	35751	4.9E+1	59412	8.6E-6	42175	6.9E-1	60250
MIMIC$_c$	1.7E-5	56819	8.4E-6	35392	4.9E+1	59133	9.2E-6	42135	6.6E-1	60250
EGNA$_{ee}$	5.6E-2	38345	1.4E-4	33517	6.0E+1	50913	7.8E-3	39462	8.9E+0	27572
EGNA$_{BGe}$	2.5E-2	58694	5.0E-5	34116	5.7E+1	60250	1.6E-3	48519	1.8E+1	56340
CMA-ES	2.4E-3	60002	1.5E-6	60002	5.5E+1	60002	1.2E-5	60002	7.0E-1	60002

variables: UMDA$_c$, MIMIC$_c$, EGNA$_{ee}$ and EGNA$_{BGe}$[3]. In addition, the overall performance was compared to such of ES [21]. As an example of the latter we chose CMA-ES (Evolution Strategy with Covariance Matrix Adaptation)[7] which is considered as one of those with better performance[4].

The optimization problems selected are the ones proposed in [1] to compare evolutionary computation algorithms in continuous domains, namely Ackley, Griewangk, Rosenbrock generalized, Sphere Model, and Summation Cancelation.

The size of the population was decided to be $R = 400$ since we consider that this is the smallest reasonable size required to allow EBCOAs discern characteristics of the fittest and less fit individuals. The experiments have been carried out with individuals of 10 and 50 variables ($n = 10$ and $n = 50$ respectively). For EBCOAs, we set the following parameters: $|K| = 3$ and $|C| = 2$ (hence, we consider only the best and worst classes of individuals). The size of the best and worst classes was chosen to be $R/4$, where R is the size of the population as previously mentioned. In the case of continuous EDAs, the learning of the model is done after selecting the $R/2$ best ones of the population, keeping the same size of the population of $R = 400$ and an elitist approach. Finally, for CMA-ES we apply again the same population size and 200 as maximum number of parents, while the rest of parameters have been left as the default values suggested by the original authors.

The stopping criterion for all the algorithms and fitness functions is satisfied when the optimum solution is found (assuming this case when the result obtained was closer

[3] See [10] for a deep review on these algorithms.

[4] For the simulation step we have applied the MATLAB program cmaes.m version 2.34 available at `http://www.icos.ethz.ch/software/evolutionary_computation/cmaes.m`

than 10^{-6} from the global optimum fitness), when a maximum of 150 generations was reached, or simply when no improvement has been obtained in the generation of the last population. Each algorithm is run 10 times for each of the optimization problems. Table 1 shows mean values of these 10 runs, illustrating the fitness value of the best individual obtained (V) and the number of evaluations (E) required for each of the experiments. The results of Table 1 evidence that EBCOAs perform quite well in most of the optimization problems proposed, and also that the results obtained at least are comparable (in some cases even improved) to the ones of EDAs and CMA-ES. If we focus individually in each of the proposed optimization problems, we appreciate that in those containing no local optima (i.e. Sphere problem) or in the ones having many small local optima (such as Ackley and Griewangk) EBCOAs show a performance comparable to such of EDAs and CMA-ES. On the other hand, in optimization problems containing local optima with much higher size (i.e. Summation Cancellation) EBCOAs manage to reach the optimum for $n = 10$ and $n = 50$, while both EDAs and CMA-ES are also capable of solving them when the size of the problem is of 10 variables but show a poorer performance when the size of the problem increases to 50. The Rosenbrock function is especially difficult to optimize regarding the fact that the global optimum is located in the middle of a quite flat region. This is the reason for EDAs not to manage to find this global optimum and to fall in a local optimum. EBCOAs do not either manage to improve these results and perform similarly as EDAs. CMA-ES is able to obtain the global optimum uniquely in small problem sizes (n=10), although when increasing the size of individuals to n=50 it also shows the same behavior as EDAs and EBCOAs.

At the light of the results we can conclude that continuous EBCOAs is a new paradigm that is able to obtain comparable results as continuous EDAs and ES, although its bigger computation cost makes it more suitable for complex problems since mainly in those the results outperform EDAs and ES.

5 Conclusions and Further Work

The original contribution of this paper is to generalize EBCOAs to continuous domains by applying Bayesian classifiers in these type of domains. EBCOAs combine evolutionary computation techniques and Bayesian classifiers in order to solve optimization problems. Experimental results to compare the performance of this new approach on typical optimization problems in continuous domains have been shown, and they have been compared with such of continuous EDAs and ES.

Future research trends also include the study and experimentation of other Bayesian classifiers, even more complex ones capable to take into account more interdependencies between variables that could be useful for more complex problems.

References

1. E. Bengoetxea, T. Miquélez, P. Larrañaga, and J. A. Lozano. Experimental results in function optimization with EDAs in continuous domain. In P. Larrañaga and J. A. Lozano, editors, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*, pages 181–194. Kluwer Academic Publishers, 2001.

2. G. Cervone. LEM2 Theory and Implementation of the Learnable Evolution. Technical report, Machine Learning and Inference Laboratory, George Mason University, 1999.
3. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.
4. N. Friedman, D. Geiger, and M. Goldsmidt. Bayesian network classifiers. *Machine Learning*, 29(2):131–163, 1997.
5. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, USA, 1989.
6. N. Hansen. The CMA evolution strategy: A comparing review. In I. Inza J.A. Lozano, P. Larrañaga and E. Bengoetxea, editors, *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*, pages 75–102. Springer, 2006.
7. N. Hansen and S. Kern. Evaluating the CMA evolution etrategy on multimodal test functions. In *Eighth International Conference on Parallel Problem Solving from Nature – PPSN VIII*, pages 282–291, 2004.
8. J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Michigan, 1975.
9. I. Kononenko. Semi-naïve Bayesian classifiers. In *Proceedings of the 6th European Working Session on Learning*, pages 206–219, Porto, Portugal, 1991.
10. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.
11. S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *Annals of Statistics*, 17:31–57, 1989.
12. X. Llorà and D.E. Goldberg. Wise breeding GA via machine learning techniques for function optimization. In Cantú-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-03, Part I*, Lecture Notes in Computer Science 2723, pages 1172–1183, Chicago, Illinois, 2003. Springer.
13. J.A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea. *Towards a New Evolutionary Computation. Advances in Estimation of Distribution Algorithms*. Springer, 2006.
14. R.S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, 38:9–40, 2000.
15. M. Minsky. Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers*, 49:8–30, 1961.
16. T. Miquélez, E. Bengoetxea, and P. Larrañaga. Evolutionary computation based on Bayesian classifiers. *International Journal of Applied Mathematics and Computer Science*, 14(3): 335–349, 2004.
17. H. Mühlenbein, T. Mahning, and A. Ochoa. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247, 1999.
18. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions: I. Binary parameters. In M. Voigt et al., editor, *Parallel Problem Solving from Nature - PPSN IV. Lecture Notes in Computer Science 1411*, pages 178–187, 1996.
19. M. Pazzani. Searching for dependencies in Bayesian classifiers. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 239–248, New York, NY, 1997. Springer–Verlag.
20. M. Pelikan. *Hierarchical Bayesian Optimization Algorithm: Toward a New Generation of Evolutionary Algorithms*. Springer, 2005.
21. H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley InterScience, 1995.