Evolutionary Latent Class Clustering of Qualitative Data

Damien Tessier, Marc Schoenauer, Christophe Biernacki, Gilles Celeux, Gérard Govaert

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Evolutionary Latent Class Clustering of Qualitative Data

Damien Tessier, TAO — Marc Schoenauer, TAO — Christophe Biernacki, Lab. Mathématiques Paul Painlevé, USTL, Lille — Gilles Celeux, Select — Gérard Govaert, UTC, Compiègne

N° 6082

December 2006

Thème COG

ISSN 0249-6399

Evolutionary Latent Class Clustering of Qualitative Data

Damien Tessier, TAO , Marc Schoenauer, TAO , Christophe Biernacki, Lab. Mathématiques Paul Painlevé, USTL, Lille , Gilles Celeux, Select , Gérard Govaert, UTC, Compiègne

Thème COG — Systèmes cognitifs
Projets SELECT et TAO

Rapport de recherche n° 6082 — December 2006 — 21 pages

Abstract: The latent class model or multivariate multinomial mixture is a powerful model for clustering discrete data. This model is expected to be useful to represent non-homogeneous populations. It uses a conditional independence assumption given the latent class to which a statistical unit is belonging.

However, whereas a predictive approach of cluster analysis from qualitative data can be easily derived from a fully Bayesian analysis with Jeffreys non informative prior distributions, it leads to a criterion (the integrated completed likelihood derived from the latent class model) that proves difficult to optimize by the standard approach based on the EM algorithm.

An Evolutionary Algorithms is designed to tackle this discrete optimization problem, and an extensive parameter study on a large artificial dataset allows to derive stable parameters. A Monte Carlo approach is used to validate those parameters on other artificial datasets, as well as on some well-known real data: the Evolutionary Algorithm seems to repeatedly perform better than other standard clustering techniques on the same data.

Key-words: Clustering, Evolutionary Computation, Qualitative features

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 — Télécopie : +33 1 72 92 59 ??

Evolutionary Latent Class Clustering of Qualitative Data

Résumé : Le modèle des classes latentes ou le modèle de mélange multinomial multivarié est un modèle puissant pour la classification de données qualitatives. Il est en effet très utile pour analyser des populations hétérogènes à l'aide d'une hypothèse d'indépendance conditionnelle par rapport aux classes à découvrir.

Cependant, si ce modèle conduit à une formulation explicite de la classification prédictive pour des variables qualitatives d'un pur point de vue bayésien non informatif, il donne naissance à un critère de vraisemblance complétée intégrée difficile à optimiser par les approches classiques associées à l'algorithme EM.

Nous proposons de traiter ce problème d'optimisation sur données discrètes par un algorithme évolutionnaire, pour lequel une étude extensive du comportenent sur une base de données simulées a permis de sélectionner des valeurs stables de ses paramètres. Des expérimentations de Monte-Carlo ont permis de valider ces choix sur d'autres jeux de données simulées, ainsi que sur un jeu classique de données réelles. En particulier, il s'avère que, dans la plupart des cas, l'algorithme évolutionnaire ainsi réglé se comporte mieux que des algorithmes classiques de classification.

Mots-clés : Clustering, Algorithms évolutionaires, Variables qualitatives

1 Introduction

When modeling an optimization problem, all practitioners face similar dilemmas: the most accurate models result in very difficult, if not intractable, optimization problems; And simplifying the model in order to obtain an optimization problem that is tractable by standard optimization methods, with proved convergence, might result in a poor fulfillment the original requirements for the problem at hand, because of the weaknesses of the model itself.

Evolutionary Algorithms, on the other hand, can handle complex optimization problems because of their flexibility that allow them to work well on non-standard search spaces, non-regular objective functions, with many local optima – at the cost of a high computational cost, and, sometimes, a poor fine-tuning of the solution.

The issue is then whether it is better to obtain a very accurate answer to the wrong question, or a possibly approximate answer to the correct question.

There exist many works describing situations where the second branch of the alternative (using Evolutionary Computation to solve the exact model) does give better solutions than working on some simplified problem, at least for some instances of the problem at hand. Examples include many situations where there is a choice between parametric and non-parametric models (e.g. in Structural Mechanics, in Geophysical Inverse problems [16]).

But similar situation arise in Machine Learning, too, where different choices of the criterion to minimize can lead to very different results. A well-known example is that of supervised learning of binary classes, where the 'kernel trick' allows one to turn the minimization of the empirical risk into a well-posed quadratic optimization problem, as proposed by Vapnik [20] as the basis of the SVM theory. But SVMs are very sensitive to noisy data for instance, and ordered-based criteria are better suited in situation of uncertainty, even though the resulting criterion is not differentiable, and hence not amenable to easy numerical optimization. However, evolutionary algorithms have helped to build efficient ROC-based learners and to obtain breakthrough results for instance in medical domains [17].

This paper is concerned with a similar situation in the context of model based cluster analysis for qualitative data. In this context the latent class model is a reference model (see for instance [11]). Usually the parameters of this model are estimated with the maximum likelihood methodology. But embedding the latent class model in a non informative Bayesian framework, it is possible to get a predictive clustering by integrating over the latent class model parameters. Such an approach is expected to be more stable, but it involves a difficult optimization problem that is considered in this paper.

The paper is organized the following way: Section 2 introduces the latent class model, derives the resulting log-likelihood function to be maximized, it presents the predictive clustering approach derived from a Bayesian perspective and the Hill-Climbing method that had been used up to now to optimize the resulting criterion. Section 3 gives the details of the Evolutionary Algorithm. Section 4 presents the parametric study and the first results on artificial data for a large number of examples. Those results allow us to come up with a robust set of parameters. Using such robust setting, the EA then is compared to the Hill-Climber algorithm using a Monte Carlo validation on smaller sets of examples drawn using the same artificial data generator. It is finally tested in Section 5 on a real dataset,

the so called *Toby* dataset, that has been extensively studied to illustrate the latent class model. Finally, Section 6 discusses those results.

2 Predictive Clustering with the Latent Class Model

2.1 The latent class model

Observations to be classified are described with d discrete variables. Each variable j has m_j response levels. Data are $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where $\mathbf{x}_i = (x_i^{jh}; j = 1, \ldots, d; h = 1, \ldots, m_j)$ with

$$\begin{cases} x_i^{jh} = 1 & \text{if } i \text{ has response level } h \text{ for variable } j \\ x_i^{jh} = 0 & \text{otherwise.} \end{cases}$$

In the standard latent class model, data are supposed to arise from a mixture of g multivariate multinomial distributions with probability distribution function (pdf)

$$f(\mathbf{x}_i; \boldsymbol{\theta}) = \sum_{k=1}^{g} p_k f_k(\mathbf{x}_i; \boldsymbol{\alpha}_k) = \sum_{k=1}^{g} p_k \prod_{j=1}^{d} \prod_{h=1}^{m_j} (\alpha_k^{jh})^{x_i^{jh}}$$

where α_k^{jh} is denoting the probability that variable $\mathbf{x}^j$ has level h if object i in cluster k, and $\boldsymbol{\alpha}_k = (\alpha_k^{jh}; j = 1, \ldots, p; h = 1, \ldots, m_j)$, $\mathbf{p} = (p_1, \ldots, p_g)$ is denoting the vector of mixing proportions of the g latent clusters, $\boldsymbol{\theta} = (p_k, \boldsymbol{\alpha}_k, k = 1, \ldots, g)$ denoting the vector parameter of the latent class model to be estimated. Latent class model is assuming that the variables are *conditionally independent* knowing the latent clusters.

Analyzing multivariate categorical data is made difficult because of the curse of dimensionality. The standard latent class model which require $(g-1)+g*\sum_j(m_j-1)$ parameters to be estimated is an answer to the dimensionality problem. It is much more parsimonious than the saturated log-linear model which requires $\prod_j m_j$ parameters. For instance, with $g = 5$, $d = 10$, $m_j = 4$ for all variables, the latent class models is characterized with 154 parameters whereas the saturated log-linear model requires about 10^6 parameters... Moreover, the latent class model can appear to produce a better fit than unsaturated log-linear models while demanding less parameters.

2.1.1 Maximum likelihood inference

Since the latent class model is a mixture model, the EM algorithm is a privileged tool to derive the ml estimates of the latent class model parameters (see [14]). The observed log-likelihood of the model is

$$L(\boldsymbol{\theta}; \mathbf{x}) = \sum_{i=1}^{n} \log \left(\sum_{k=1}^{g} p_k \prod_{j=1}^{d} \prod_{h=1}^{m_j} (\alpha_k^{jh})^{x_i^{jh}} \right).$$

Denoting $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_g)$ with $\mathbf{z}_k = (z_{1k}, \ldots, z_{nk})$ and $z_{ik} = 1$ if $\mathbf{x}_i$ arose from cluster k, $z_{ik} = 0$ otherwise, the unknown indicator vectors of the g clusters, the completed log-likelihood is

$$L(\boldsymbol{\theta}; \mathbf{x}, \mathbf{z}) = \sum_{i=1}^{n} \sum_{k=1}^{g} z_{ik} \log \left(p_k \prod_{j=1}^{d} \prod_{h=1}^{m_j} (\alpha_k^{jh})^{x_i^{jh}} \right).$$

From this completed log-likelihood, the equations of the EM algorithm are easily derived and this algorithm is as follows from an initial position $\boldsymbol{\theta}^{(0)} = (\mathbf{p}^{(0)}, \boldsymbol{\alpha}^{(0)})$.

- E step: calculation of $\mathbf{t}^{(r)} = (t_{ik}^{(r)}, i = 1, \ldots, n, k = 1, \ldots, g)$ where $t_{ik}^{(r)}$ is the conditional probability that $\mathbf{x}_i$ arose from cluster k

$$t_{ik}^{(r)} = \frac{p_k^{(r)} f_k(\mathbf{x}_i; \boldsymbol{\alpha}_k^{(r)})}{\sum_{\ell=1}^{g} p_\ell^{(r)} f_\ell(\mathbf{x}_i; \boldsymbol{\alpha}_\ell^{(r)})}.$$

- M step: Updating of the mixture parameter estimates,

$$p_k^{(r+1)} = \frac{\sum_i t_{ik}^{(r)}}{n} \quad \text{and} \quad (\alpha_k^{jh})^{(r+1)} = \frac{\sum_{i=1}^{n} t_{ik}^{(r)} x_i^{jh}}{\sum_{i=1}^{n} t_{ik}^{(r)}}.$$

2.1.2 Bayesian inference

Since the Jeffreys non informative prior distribution for a multinomial distribution $\mathcal{M}_r(q_1, \ldots, q_r)$ is a conjugate Dirichlet distribution $\mathcal{D}(1/2, \ldots, 1/2)$ a fully non-informative Bayesian analysis is possible for latent class models contrary to Gaussian mixture models (see [15]). The prior distribution of the mixing weights is a Dirichlet $\mathcal{D}(1/2, \ldots, 1/2)$ distribution. Then, denoting $n_k = \#\{i : z_{ik} = 1\}$ and $n_k^{jh} = \#\{i : z_{ik} = 1, x^{jh} = 1\}$, the full conditional distribution of $(p_k, k = 1, \ldots, g)$ is a Dirichlet distribution $\mathcal{D}(1/2+n_1, \ldots, 1/2+n_g)$. The conditional probabilities of the allocation variables are given, for $k = 1, \ldots, g$ and $i = 1, \ldots, n$, by

$$t_{ik} = \frac{p_k f_k(\mathbf{x}_i; \boldsymbol{\alpha}_k)}{\sum_{\ell=1}^{g} p_\ell f_\ell(\mathbf{x}_i; \boldsymbol{\alpha}_\ell)}.$$

In a similar way, the prior distribution of $(\alpha_k^{j1}, \ldots, \alpha_k^{jm_j})$ is a $\mathcal{D}(1/2, \ldots, 1/2)$ for $k = 1, \ldots, g$ and $j = 1, \ldots, d$ and the full conditional distribution for $\{\alpha_k^{jh}\}$, $(j = 1, \ldots, d; k = 1, \ldots, g)$ is

$$\alpha_k^{jh} | \ldots \sim \mathcal{D}(1/2 + n_k^{j1}, \ldots, 1/2 + n_g^{jm_j}).$$

The Gibbs sampling implementation of the fully non informative Bayesian inference is straightforwardly deduced from those formulas and is not detailed here.

2.2 Predictive Clustering

In a fully Bayesian perspective, it is possible to derive a classification of the data from the joint predictive distribution

$$f(\boldsymbol{x}, \boldsymbol{z}) = \int_{\Theta} f(\boldsymbol{x}, \boldsymbol{z}; \theta)\pi(\theta)d\theta.$$

Such an approach can involve various difficulties for general mixture models. But for the standard latent class model, it leads to a simple formulation. Assuming non informative Dirichlet prior distributions for the mixing proportions and the latent class parameters

$$\pi(\mathbf{p}) = \mathcal{D}(a, \ldots, a) \quad \text{et} \quad \pi(\boldsymbol{\alpha}_k^j) = \mathcal{D}(a, \ldots, a), \tag{1}$$

with $a = 1/2$ for a Jeffreys prior, we get using conjugate property of the Multinomial-Dirichlet distributions (see for instance [15])

$$f(\boldsymbol{x}, \boldsymbol{z}) = \frac{\Gamma(ga)}{\Gamma(a)^g} \frac{\prod_{k=1}^g \Gamma(n_k + a)}{\Gamma(n + ga)} \prod_{k=1}^{g} \prod_{j=1}^{d} \frac{\Gamma(m_j a)}{\Gamma(a)^{m_j}} \frac{\prod_{h=1}^{m_j} \Gamma\left(n_k^{jh} + a\right)}{\Gamma(n_k + m_j a)}.$$

The predictive clustering approach consists of maximizing $f(\mathbf{z}|\mathbf{x})$. Since $f(\mathbf{z}|\mathbf{x}) \propto f(\mathbf{x}, \mathbf{z})$, it leads to maximize the criterion

$$C(\boldsymbol{z}) = \sum_{k=1}^{g} \log\Gamma(n_k + a) - \log\Gamma(n + ga) + \sum_{k=1}^{g} \sum_{j=1}^{d} \left\{ \sum_{h=1}^{m_j} \log\Gamma\left(n_k^{jh} + a\right) - \log\Gamma(n_k + m_j a) \right\}. \tag{2}$$

Before embarking with optimization issues for this criterion, this predictive approach deserves some remarks.

- The criterion $C(\boldsymbol{z})$ is the integrated completed likelihood criterion ICL defined in [2] which is a penalized likelihood criterion favoring mixture models giving rise to a partitioning of the data with the greatest evidence. In [2], this criterion is approximated using a BIC-like approximation (see for instance [14]). But in the context of multivariate multinomial distributions, there is no need to use such an asymptotic approximation. Thus, optimizing $C(\boldsymbol{z})$ with different numbers of mixture components and choosing the number of component maximizing $C(\boldsymbol{z})$ can be regarded as an optimal strategy regardless the size of the data set.
- The classification $\boldsymbol{z}$ derived from the optimization of $C(\boldsymbol{z})$ is not depending on the parameter θ, but is depending on the model at hand.

2.3 A Naive Hill-Climbing Algorithm

In this predictive approach the unique and difficult task is to find the $\boldsymbol{z}$ vector of dimension n optimizing $C(\boldsymbol{z})$. A simple solution consists of using an alternating optimization algorithm for which an iteration ($\boldsymbol{z}^- \rightarrow \boldsymbol{z}^+$) starting from $\boldsymbol{z}^0$ is as follows
For $i = 1, \ldots, n$

$$z_i^+ = \arg\max_{z_i} C(z_1^+, \ldots, z_{i-1}^+, z_i, z_{i+1}^-, \ldots, z_n^-).$$

This algorithm can be expected to be quite sensitive to its initial position. Thus, it is highly recommended to start from a reasonable $\boldsymbol{z}^0$ vector which can be $\boldsymbol{z}^0 = \text{MAP}(\hat{\theta})$ where $\hat{\theta}$ is the ml estimate of θ derived from the EM algorithm and MAP (for Maximum A Posteriori) is the function providing guessed values $\hat{z}_{ik}$ for the missing cluster indicators z_{ik} from $\hat{\theta}$ in the following way

$$\hat{z}_{ik} = \begin{cases} 1 & \text{if } \arg\max_\ell t_{i\ell}(\hat{\theta}) = k \\ 0 & \text{otherwise.} \end{cases}$$

The main advantage of such an algorithm is it simplicity and relative speed. But, since the space where $\boldsymbol{z}$ lies is of large dimension and discrete, this algorithm can be expected to be suboptimal.

3 The Evolutionary Algorithm

Because of the probable sub-optimality of the simple Hill-Climber described in previous section, Evolutionary Algorithms are good candidates to optimize the fitness function given by Equation 2. This section introduces the problem-specific parts of the Evolutionary Algorithm that has been used to tackle this optimization problem, namely the genotype, the variation operators (crossover and mutation) and the initialization procedure. All representation-independent parts will be briefly described in next section, together with the experimental results.

3.1 Representation

The genotype is the vector (z_i) giving for each example the cluster it is assigned to. It is of size n, the number of examples, and takes values $z_i \in [0, g-1]$, where g is the number of clusters. Because the latent-class predictive clustering technique tries different number of clusters to find the optimal number, the general integer representation has been chosen, even in the case where $g = 2$ (though the integer-based variation operators amounts to the standard bitstring operators as will be seen in Section 3.2).

Note that though the values are represented as integers, they must be considered as purely symbolic, as there is no notion of order or distance among the different classes.

3.2 Variation Operators

Straightforward variation operators for vectors of symbolic values have been used:

- **Uniform crossover** is analogous to the corresponding bitstring operator: the parents exchange the values for each example independently with probability 0.5. Note that 1-point crossover could also be used, randomly choosing a crossover point and swapping the values of all examples on the second half of the vector (similar to standard bitstring 1-point crossover).
- **Uniform mutation** applies some *gene-level mutation* to each position with a given probability $p_{mutGene}$. Here, the gene-level mutation amounts to choose for the class value of the given example a new value (i.e., different from its original value to ensure variation) chosen uniformly within all available values.

3.3 Initialization

Three initialization procedures have been compared:

- The natural evolutionary way is to initialize the whole population uniformly, choosing for each feature a value uniformly in $[0, g]$.
- However, a good solution can be obtained by the algorithm EM+HC described in Section 2.3, and another way to initialize the population for the Evolutionary Algorithm is to start around this local optimum for the EM+HC algorithm [19]: each initial individual is obtained by performing a number of gene mutations on this good solution, in order to introduce some diversity in the initial population. The parameters of this initialization procedure determine the number of mutations performed on the initial good solution: This number can be either fixed (one parameter), or drawn according to a binomial law (2 parameters).
- Finally, it is possible to choose one part of the population uniformly over the search space, and the other part by perturbing the known local optimum, with different numbers of gene-mutation, resulting in a hybrid initialization method, with additional parameters describing the proportion used for each procedure.

3.4 Implementation

The algorithm described above has been implemented using the Evolving Object library [10] using the Graphical User Interface *GUIDE-II*[12, 13]: the user has to (graphically) provide the structure of the genotype, and to write the fitness function. Default representation-dependent operators (i.e., initialization, crossover and mutation) are then proposed to the user, who can choose to use them directly, or to customize then. *GUIDE-II* then writes the full set of EO classes, that is compiled into a fully functional program. Additionally, the Evolution Engine (selection, population and offspring sizes, replacement) can be also

graphically set by the user (as in the original *GUIDE* [4, 1]), or can be set at run-time through user-defined parameters (see section 4.3).

4 Results on Artificial Data

4.1 The Artificial Dataset

Artificial data are generated from a mixture of $g = 2$ six-variate multinomial distributions ($d = 6$), the number of response levels or modalities being four for the first four ones ($m_1 = \ldots = m_4 = 4$) and six for the last two ones ($m_5 = m_6 = 6$). Values of mixing proportions and parameters of the multivariate distributions are the following:

$$p_1 = 0.3, p_2 = 0.7$$

$$\begin{array}{lll} \alpha_1^1 = (0.2\ 0.2\ 0.6) & \alpha_1^2 = (0.2\ 0.2\ 0.6) & \alpha_1^3 = (0.2\ 0.6\ 0.2) \\ \alpha_1^4 = (0.2\ 0.7\ 0.1) & \alpha_1^5 = (0.2\ 0.2\ 0.4\ 0.2) & \alpha_1^6 = (0.2\ 0.2\ 0.4\ 0.2) \\ \alpha_2^1 = (0.6\ 0.2\ 0.2) & \alpha_2^2 = (0.6\ 0.2\ 0.2) & \alpha_2^3 = (0.2\ 0.3\ 0.5) \\ \alpha_2^4 = (0.1\ 0.1\ 0.8) & \alpha_2^5 = (0.2\ 0.2\ 0.2\ 0.4) & \alpha_2^6 = (0.2\ 0.2\ 0.2\ 0.4) \end{array}$$

The overlapping rate of the two components is about 10%.

As a first step, only one sample of size $n = 3200$ is generated, the aim being to determine a reasonable parameter set for the evolutionary algorithm in order to maximize the predictive criterion C. Nevertheless, the number of mixture components used in the unknown partition $\mathbf{z}$ was varied from two to five.

4.2 Preliminary experiments

Parameter tuning can be considered as Achille's heel of Evolutionary Algorithms practitioners. Whereas adaptivity can sometimes help [5], most parameters have to be fixed by the user based on his own experience, and/or on work on similar problems. Hence, since [9], the systematic trial-and-error remains the most widely used method to determine the best set of parameters, a noteworthy exception being the statistical approach proposed in [6].

Some very preliminary experiments were done to limit the number of parameters to explore and the range of exploration. It became rapidly clear, for instance, that inoculation of the local optimum found by the EM+HC algorithm, as recommended in [19], is beneficial in terms of speed, without any improvement on quality. Hence the only initialization procedure used in the following is the uniform initialization, in order to allow the algorithm to better explore the search space. Further work will be devoted to investigate how inoculation should be optimally used to lead to the same results with a lower computational effort.

4.3 Parameter study

4.3.1 Evolution Engine

The Evolution Engine describes the (representation-independent) way used to evolve a population of individuals, i.e., the selection and replacement procedures, as well as the population size and the number of generated offspring at each generation.

Though they can all be put into the same framework [4], some well-known general classes of selection/replacement procedures have been distinguished here:

- Generational Genetic Algorithm (GGA): P selected parents give birth to P offspring, that in turn replace all P parents. The parameters are P, the population size, the selection mechanism, and its selective pressure. Tournament selection has been chosen here for its simplicity and robustness (it is insensitive to bad fitness scaling for instance). Three values for the selection pressure have been tried, namely 1.6 (the so-called "stochastic tournament" with parameter 0.8, that uniformly draws 2 individuals and returns the best one with probability 0.8), 2 and 4 (corresponding to "deterministic tournaments" of sizes 2 and 4 respectively).

- Steady-State Genetic Algorithm (SSGA): 1 offspring is generated from 1 or 2 selected parents (depending on whether crossover should be applied or not), and it is inserted back in the population by removing a poor parent. The same parameters than for GGA apply for selection. An additional parameter comes from the replacement procedure: it was chosen to be either deterministic (the worst parent dies), or stochastic, involving a tournament of size 10 (the worst of 10 uniformly chosen parents dies).

- Evolution Strategies (ES): λ offspring are generated from the μ parents, without selection (i.e. all parents will give birth to the same number of offspring on average); two replacement procedures can be used, namely (μ, λ), where the best of the λ offspring become the new population, and $(\mu + \lambda)$, where the μ best of the μ parents PLUS the λ offspring become the new population. This schema is borrowed from the ES world (see e.g. again [3]), and admittedly good setting is to take $\lambda = 7 * \mu$. Note that the "population size" to be considered when comparing the ES engine to one of the GA engines is λ rather than μ, in connection with the required computing effort.

Population sizes of 50, 100 and 200 have been considered for the xxGA engines, while the values of μ for the ES engines have been chosen among 1, 10 and 30, to approximately obtain the same number of evaluations per generations (for 10 and 30) while testing the $(1 \stackrel{+}{,} 7) - ES$ (the xxGA engines generally require larger populations).

4.3.2 Variation Operators

The parameters related to the variation operators are twofold. At the population level, one has to decide how to apply crossover and mutation (assuming the general case where no fancy variation operator involving more parents is used). It has been decided here to follow

	GGA	SSGA	ES
Pop size	50/100/200	50/100/200	01/10/30
Sel. Pressure	1.6/2/4	1.6/2/4	1.0
No offspring	100.00%	1	700.00%
Parent survive	0.00%	PopSize-1	0%/100%
Repl, "Pressure"	–	$+\infty/10$	–
p_{cross}	0/0.5/1	0/0.5/1	0/0.5/1
p_{mut}	0.5/1	0.5/1	0.5/1
$n * p_{mutGene}$	1/2/3	1/2/3	1/2/3

Table 1: Set of parameters used on the two latent class simulated data

what could be termed the standard-GA-way [7]: apply to all selected individuals first the crossover operator with probability p_{cross} and then the mutation operator with probability p_{mut}. Another alternative could have been to follow the standard-GP-way, i.e., to apply to the selected parents either crossover or mutation or the no-operation cloning – with relative weights. In order to limit the number of values, but to nevertheless test the extreme cases, only the values 0.0, 0.5 and 1.0 were considered. However, because mutation is known to be mandatory, only 0.5 and 1.0 were used for p_{mut}.

At the individual level, some variation operators have specific parameters. Here (see Section 3.2), the only representation-specific parameter is $p_{mutGene}$, the probability that a given example is given a different class. It is well-known [3] that the value $\frac{1}{n}$ (where n is the size of the vector) usually is a robust choice. An exploration of higher values was also done here, namely $\frac{2}{n}$ and $\frac{3}{n}$.

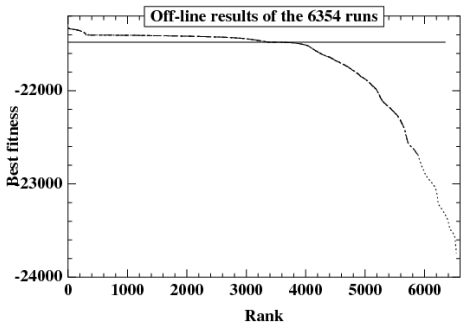
4.3.3 Experimental settings

Table 1 summarizes the different parameter sets that have been used for those experiments. For each set of parameters, 11 runs were performed. The total number of runs are thus 11 times 162, 324, and 108 respectively for the GGA, SSGA and ES evolution engines, leading to a total of 6354 runs altogether.

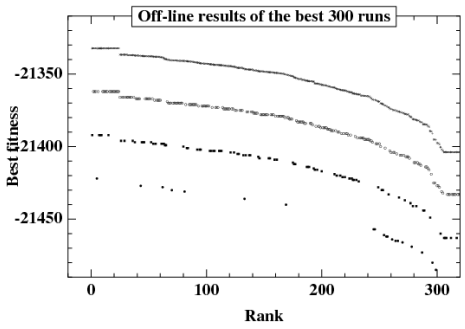
All runs were given the same stopping criterion based on the number of fitness evaluations: a run stopped after a maximum of 500000 evaluations, or after 3000 evaluations without improvement following an initial number of 30000 evaluations, whichever came first. During the curse of this parameter study, the average running time for a run was about 30mn, and the 40-nodes cluster that was used run during two weeks, including single node crashes and general power failures.

4.4 Comparing Evolution Engines

First of all, out of the 6354 runs, 3393 found a solution that is equal or better than the solution found by the EM+HC algorithm. However, out of those, only 304 were using the



All 6354 runs. The horizontal line is the fitness reached by the EM+HC algorithm.



Zoom on the first 300 runs. From top to bottom: all runs, SSGA, ES and GGA – see text for the details.

Figure 1: Off-line results of the runs on the two latent class simulated data. One dot corresponds to one run, the x-axis indicates its relative rank among all 6354 runs, and the y-axis is the fitness reached at the end of the run.

GGA engine, compared to the 2413 and 676 respectively for the SSGA and ES engines – remember that there were 3 times more runs for SSGA than for ES.

Moreover, the best fitness value (-21 332.2) was obtained 24 times, only once by a GGA engine, compared to respectively 18 and 5 times for SSGA and ES.

Figure 1-a shows a snapshot of all results, while Figure 1-b displays a zoom on the best 300 runs: The upper plot corresponds to the actual fitness, and represents all 300 runs. Each one of the curve below is an excerpt of the top one, artificially translated downward to make it readable. From top to bottom: SSGA, ES and GGA, obviously outperformed here as witnessed by the sparseness of the plots.

At the other extreme, though less significant, the worse 500 results have been obtained by GGA, and out of the worse 1000 off-line results, only 17 and 83 were using the SSGA and ES engines respectively.

Obviously, from those results, the GGA engine is outperformed by both SSGA and ES. It might be that the correct settings for GGA were not among the very few we have tried. However, the relative robustness of the results of the other evolution engines was a clear argument to abandon GGA from thereon.

4.5 Comparing gene-mutation probabilities

Figure 2 shows the off-line results, for the SSGA and ES engines only, and for the best runs out of the 300 overall best runs. Three values were tested for $n * P_{mutGene}$, 1, 2 and 3 (see Section 4.3. Similarly to what was done in Figure 1-b, the top line contains all runs (one point for each run), and the other scatter plots are artificially translated downward to make them readable: the first three plots downwards are the SSGA runs with respective values 1, 2 and 3 for $n * P_{mutGene}$, and the three other plots represents the results of the ES runs for the same values (1, 2, and 3) of $n * P_{mutGene}$. Be careful that the plots with $n * P_{mutGene} = 3$ only appear between ranks 240 and 300.

So a clear conclusion can also be drawn here: the value 1 for $n * P_{mutGene}$ is far more efficient and robust that the value 2 and 3. Note that the same phenomenon is also clearly visible for the GGA engine – and, unsurprisingly, the worst results, at the other end of the plot, were all obtained with $n * P_{mutGene} = 3$.

Hence from now on, only $n * P_{mutGene} = 1$ will be considered, together with SSGA and ES engines.

4.6 Other parameters

The situation however is not so clear when it comes to study the influence of the other parameters from Table 1.

As far as ES runs are concerned, the "plus" strategy seems more efficient to find high values of the criterion than the "comma" strategy, while a value of 1 for μ is worse than both values 10 and 30, both giving equivalent offline results. And, surprisingly, nothing can actually be deduced about the values of P_{cross} and P_{mut}.

As for the SSGA engine, again no very strong conclusion could be drawn. However, one could notice in the off-line results a slight advantage of the deterministic over the stochastic replacement, a slight advantage of the standard selective pressure (value 2) over the values 1.6 and 4, and a clearer disadvantage when no crossover was present ($P_{cross} = 0$) compared to both other trials ($P_{cross} = 0.5$ and $P_{cross} = 1$). No influence of the mutation parameter (with possible values 0.5 or 1.0) could be identified.

4.7 A Robust Parameter Setting

From this parametric study, different sets of parameters for the EA seemed equivalently efficient and robust. One was nevertheless chosen for all following experiments:
Steady State GA with population size 50, selection by tournament of size 2, deterministic replacement, crossover and mutation rate 1, gene-mutation rate $\frac{1}{n}$ (in average, one mutation

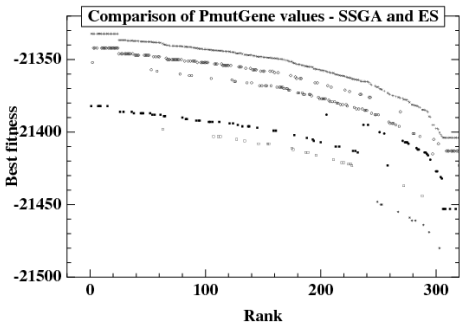


Figure 2: Comparison of different values of $n * P_{mutGene}$, for SSGA and ES engines only. From top to bottom: all values, (SSGA, 1), (SSGA,2), (SSGA,3). (ES.1), (ES.2) and (ES,3) – see text for details.

per genotype). The stopping criterion remains the same, 500000 evaluations, or 3000 without improvement after a minimum number of 30000, whichever comes first.

4.8 Best Results on Artificial Data

The same artificial dataset (3200 examples, see Section 4.1) was then tested, using the hopefully robust setting described above, with more than 2 classes. When considering only the overall best results within 11 independent runs, the results are those presented in Table 2: whatever the number of classes (between 2 and 5), the EA significantly outperforms the EM+HC algorithm. Moreover, the results of the EA suggest that the optimal number of classes is here 2, while EM+HC favors 3 classes.

4.9 Monte Carlo validation

The aim is now to make a more extensive evaluation of both the optimization strategy (ability to obtain the optimal value of C) and the predictive criterion (ability to detect

# cl.	2	3	4	5
EM+HC	-21 479.18	-21 407.19	-21 849.26	-21 858.79
EA	-21 332.2	-21 361.9	-21 373.1	-21 469.9

Table 2: Best results on the artificial data (Section 4.1) for different number of classes (best value reached in 11 independent EA runs).

the right number of components). Extensive Monte Carlo simulations have been performed using the same data generator (see Section 4.1), the sample size being $n = 200$ in each situation. Both the EM+HC algorithm and the EA, with the robust parameters from previous section, were run. For the EA, as usual, 11 independent runs were performed, in order to check the variability of the algorithm. Such a small size of sample was chosen to decrease the computational cost: a single EA run took about 3 mn in this context. The stopping criterion was the same as before, but all runs stopped after the minimal possible number of evaluations, i.e., 33000 here, and had reached their best value sometimes long before that.

The results obtained for two classes confirm the tendency observed on the 3200-example dataset: out of 20 sets of 11 runs, 13 found a better value than the EM+HC algorithm, only one found a worse value, and the other 6 runs found exactly the same value. Moreover, those 10 "equivalent" sets of runs correspond to the runs where the EA found its optimal solution the more often out of its 11 runs (3.3 times on average over those 6 datasets vs 2 for the other 13 datasets). Hence those datasets seem to corresponds to easier problems. Note also that out of its 11 independent runs, the EA outperforms the EM+HC algorithm 3.7 times on average. Finally,

When running both algorithms on 3 to 5 classes, there are no more ties in the comparisons: the EA outperforms EM+HC on 17 datasets for 3 classes, on 14 datasets for 4 classes and on 16 datasets for 5 classes, but on different datasets. Moreover, when increasing the number of classes, the EA repeatedly finds its best results by removing one or more classes (i.e., using only part of the possible values for the classification): for instance, 8 runs optimizing the 3-classes fitness ended up with only 3 classes in their best result; when using the 4-classes fitness, 11 of the 20 best runs ended up with 3 classes, and 3 with only 2 classes; and for the 5-classes fitness, 9 results used only 4 classes, 5 only 3, and even 1 ended up using only 2 classes. This ability to use less classes than what is asked gives clear indications about the actual optimal number of classes.

5 Results on Real-World Data

The data from Stouffer and Toby [18] have been used in many works devoted to latent class models (see for instance [8]). The dataset is made of 216 examples involving four binary variables (i.e. possibly taking 2 values each). The number of classes is unknown. For the

# cl.	EM + HC	EA	EM	K-means
2	-559.7498	**-553.4546**	-562.8296	-569.6086
3	-573.6737	**-563.0172**	-592.4112	-579.6012
4	-603.6050	**-576.1582**	-582.7882	-609.6562
5	-609.6562	**-593.2363**	-598.6893	-622.9583

Table 3: Results on the Stouffer and Toby data

anecdote, those data were gathered in a sociological questionnaire where the goal was to find out whether you are more faithful to a friend than to the law ...

Table 5 presents the best results obtained with four different algorithms on those data: the EM+HC algorithm described in section 2.3, the Evolutionary Algorithm described in section 3, using one of the best parameter settings that came out of the parametric study (as described in section 4.3), and two algorithms from the *WEKA* toolbox [21], EM and k-means. Beware that the two latter algorithms do **not** optimize the log-likelihood objective function described by equation 2 – the values given in Table 5 have been computed a posteriori from the optimal clusters given by the algorithms.

It is clear from the results of Table 5 that the Evolutionary Algorithms gives the best results in all cases. Note that the results of the EA that are presented in this Table are the best results out of 11 runs: 11 runs seem sufficient here to outperform the EM+HC algorithm, though of course more runs might always find better results, as the optimal values are not known. Those results also suggest that the optimal number of classes is 2, a solution which makes sense from the statistical viewpoint.

6 Discussion and Further Work

6.1 EA parameters

Because parameter tuning is known to be one of the weaknesses of EAs, it is important to try to obtain a set of parameters that can be considered as robust – this is what we tried to achieve in section 4.3.

Strong evidences appeared that the generational model of evolution was to be abandoned (and other experiments using other settings of crossover and mutation rates did not seem to indicate any improvement), and that the mutation rate per gene should not be larger than $\frac{1}{n}$, the number of examples in the dataset. However, all other parameters (population size, selection pressure, operators probabilities) did not seem to be correlated with the overall result of the EA. This included, and it was a rather big surprise, P_{cross} and P_{mut}, the crossover and mutation probabilities. But we must here remember that, because the goal was here to find the best possible clusters regardless of the CPU cost, only off-line results have been considered, and it could be the case that P_{cross} and P_{mut} indeed modify the dynamics of the runs.

And it is indeed – partially – the case, as a few observations of on-line results demonstrates: Figure 3 shows the evolution of the best fitness, averaged over the 11 runs with the same parameters, for the SSGA engine with population size 50, selection pressure 2 and deterministic replacement, for the 6 possible values of (P_{cross}, P_{mut}). Whereas this confirms the idea that the crossover rate actually has no influence on the results (the two groups of 3 curves, each corresponding to the three values of P_{cross}, are remarkably similar), it also shows that the runs using a smaller mutation rate of 0.5 perform much better (as confirmed by a Wilcoxon test with 99% confidence), and reach their maximum much more rapidly, than those with a mutation rate of 1 .

The interesting case in Figure 3 is that with $P_{cross} = 0$: indeed, it seems a little intriguing at first that applying mutation only on half of the selection individuals can make any difference. The reason is that, even during the generations where no mutation is applied (and hence no additional evaluation is performed), the selected individual is copied and inserted back in the population, replacing the worst individual: decreasing the mutation rate (at the individual level) amounts here to make the algorithm more elitist even though a larger selection pressure didn't seem to help in any way (see Section 4.6). Further investigations are needed to deepen this analysis.

6.2 Links with Hill-Climbing

The method that was used was a standard hill-climbing, based on a simple change of one example from one class to another. Hence it is easily trapped into local optima for this move operator. Going back to the results presented in section 4, the barrier at fitness level -21479.18 is easily seen on figure 1: it is the fitness reached using the EM+HC algorithm, but it is also the best fitness reached by some EA runs. On the same figure, other barriers can be seen: for instance, running the HC algorithm from a solution obtained by a run that gave a slightly better answer than -21479.18, say -21477.8, stops again at next barrier, namely -21404. Again, starting the HC algorithm from a solution obtained in a run that stopped at -21402.7 now gives the best answer we ever obtained at -21332.2, and no further improvement has ever been obtained using the hill-climbing, neither from this stopping point, nor from any of its neighbors.

Those post-experiments strongly suggest that a hybridization between an Evolutionary Algorithm and the Hill-Climbing procedure might give faster if not better results. However, there are many possible hybridization, and on-going investigation are dedicated to finding which one works best.

6.3 Inoculation

Inoculation has been proposed many years ago as a way to introduce domain knowledge into an evolutionary algorithm through a non-uniform initialization [19]. Here, the EM algorithm can be a provider of a good starting point – as it does for the EM+HC algorithm. Indeed, preliminary experiments in that direction suggest that, again, initializing the population using some perturbations of the EM solution does speed-up considerably the convergence.

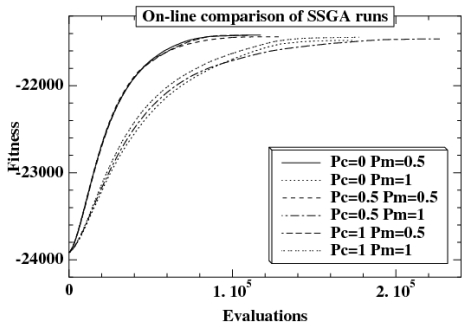


Figure 3: On-line results for different combination of operator rates, for SSGA with popSize 50, selection pressure 2, deterministic replacement and $\frac{1}{n}$ gene-mutation rate. The 3 upper plots correspond to $P_{mut} = 0.5$ and the 3 lower ones to $P_{mut} = 1$

Of course, this biases the search toward a specific region of the search space, while the global optimum might lie somewhere else, and uniform initialization should always be used, at least partially, to ensure a global exploration. However, though further detailed experiments are required, our initial tests never showed that a better solution could be obtained by uniform initialization and not by EM-based initialization.

7 Conclusion

Overall, the results presented in this paper witness that indeed, EAs are a good choice to optimize the latent class criterion for qualitative clustering: the (naive) EM+HC algorithm was outperformed except for very few tests. Moreover, the results on real well-studied data confirmed the efficiency of the evolutionary approach.

Many improvements are still possible, in particular with respect to the convergence speed of the EA, as only off-line results were considered here as a validation criterion. In particular, a clever inoculation of the EM solution seems to be a royal road to seep improvement.

Nevertheless, we claim that those results are yet another success of the evolutionary approach in Machine Learning and Statistics, demonstrating that we should consider not only objective functions that can be solved by standard methods, with guaranteed convergence, but also measures of success that are more difficult to optimize, but yet give more accurate insight on the data at hand.

References

[1] M. Arenas, P. Collet, A. Eiben, M. Jelasity, J. Merelo, B. Paechter, M. Preuß, and M. Schoenauer. Dream: A framework for distributed evolutionary algorithms. In J. M. et al., editor, *Proceedings of the* 7^{th} *Conference on Parallel Problems Solving from Nature*, LNCS 2439, pages 665–675. Springer-Verlag, 2002.

[2] C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. on PAMI*, 22:719–725, 2000.

[3] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. New-York:Oxford University Press, 1995.

[4] P. Collet, E. Lutton, M. Schoenauer, and J. Louchet. Take it easea. In M. S. et al., editor, *Proceedings of the* 6^{th} *Conference on Parallel Problems Solving from Nature*, LNCS 1917, pages 891–901. Springer-Verlag, 2000. http://sourceforge.net/projects/easea.

[5] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124, 1999.

[6] O. François and C. Lavergne. Design of evolutionary algorithms: a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, 2001.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[8] L. A. Goodman. Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61:215–231, 1974.

[9] J. J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-16, 1986.

[10] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving Objects: a general purpose evolutionary computation library. In P. C. et al., editor, *Artificial Evolution'01*, pages 229–241. Springer Verlag, LNCS 2310, 2002. URL: http://eodev.sourceforge.net/.

[11] J. Magidson, J.and Vermunt. Latent class analysis. In D. Kaplan, editor, *The Sage Handbook of Quantitative Methodology for the Social Sciences*, pages 175–198. Thousand Oakes: Sage Publications, 2000.

[12] J. Manley. Guide-ii: une interface graphique pour la conception d'algorithmes évolutionnaires. Master's thesis, Master Intelligence Artificielle, Université Paris 6, 2003. In French.

[13] J. Manley. *GUIDE-II User Manual*. INRIA, 2003.

[14] G. J. McLachlan and D. Peel. *Finite Mixture Models*. New York, Wiley, 2000.

[15] C. P. Robert. *The Bayesian Choice*. Springer Verlag, 2001. Second edition.

[16] M. Schoenauer and M. Sebag. Using Domain Knowledge in Evolutionary System Identification. In K. G. et al., editor, *Evolutionary Algorithms in Engineering and Computer Science*. John Wiley, 2002.

[17] M. Sebag, J. Azé, and N. Lucas. Impact Studies and Sensitivity Analysis in Medical Data Mining with ROC-based Genetic Learning. In *Proceedings of IEEE International Conference on Data Mining, ICDM03*, pages 637–640, 2003.

[18] S. Stouffer and J. Toby. Role conflict and personality. *American Journal of Sociology*, 56:395–406, 1951.

[19] P. Surry and N. Radcliffe. Inoculation to initialize evolutionary search. In *Evolutionary Computing: AISB workshop*, number 1141 in LNCS, pages 269–285. Springer Verlag, 1996.

[20] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

Contents

1 Introduction **3**

2 Predictive Clustering with the Latent Class Model **4**
2.1 The latent class model 4
2.1.1 Maximum likelihood inference 4
2.1.2 Bayesian inference 5
2.2 Predictive Clustering 6
2.3 A Naive Hill-Climbing Algorithm 7

3 The Evolutionary Algorithm **7**
3.1 Representation 7
3.2 Variation Operators 8
3.3 Initialization 8
3.4 Implementation 8

4 Results on Artificial Data **9**
4.1 The Artificial Dataset . . . 9
4.2 Preliminary experiments . . . 9
4.3 Parameter study . . . 10
4.3.1 Evolution Engine . . . 10
4.3.2 Variation Operators . . . 10
4.3.3 Experimental settings . . . 11
4.4 Comparing Evolution Engines . . . 11
4.5 Comparing gene-mutation probabilities . . . 13
4.6 Other parameters . . . 13
4.7 A Robust Parameter Setting . . . 13
4.8 Best Results on Artificial Data . . . 14
4.9 Monte Carlo validation . . . 14

5 Results on Real-World Data **15**

6 Discussion and Further Work **16**
6.1 EA parameters . . . 16
6.2 Links with Hill-Climbing . . . 17
6.3 Inoculation . . . 17

7 Conclusion **18**

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399