



HAL
open science

Density estimation with Genetic Programming for Inverse Problem solving

Michael Defoin Platel, Sébastien Verel, Manuel Clergue, Malik Chami

► **To cite this version:**

Michael Defoin Platel, Sébastien Verel, Manuel Clergue, Malik Chami. Density estimation with Genetic Programming for Inverse Problem solving. EuroGP'07, the 10th European Conference on Genetic Programming, Apr 2007, Valencia, Spain. pp.45–54, 10.1007/978-3-540-71605-1_5. hal-00164762

HAL Id: hal-00164762

<https://hal.science/hal-00164762>

Submitted on 4 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Density Estimation with Genetic Programming for Inverse Problem Solving

Michael Defoin Platel^{1,2}, Sébastien Vérel², Manuel Clergue², and Malik Cham¹

¹ Laboratoire I3S

CNRS-Université de Nice Sophia Antipolis, France

² Laboratoire d’Océanographie de Villefranche sur Mer
CNRS-Université Pierre et Marie Curie-Paris6, France


Abstract. This paper addresses the resolution, by Genetic Programming (GP) methods, of ambiguous inverse problems, where for a single input, many outputs can be expected. We propose two approaches to tackle this kind of many-to-one inversion problems, each of them based on the estimation, by a team of predictors, of a probability density of the expected outputs. In the first one, Stochastic Realisation GP, the predictors outputs are considered as the realisations of an unknown random variable which distribution should approach the expected one. The second one, Mixture Density GP, directly models the expected distribution by the mean of a Gaussian mixture model, for which genetic programming has to find the parameters. Encouraging results are obtained on four test problems of different difficulty, exhibiting the interests of such methods.

1 Introduction

One of the main application of Genetic Programming (GP) is for the approximation of unknown functions, a task known as Symbolic Regression [7]. To construct a such approximator, a GP system is trained on a given dataset that consists of pairs of inputs and desired outputs, representative of an unknown function. This is a direct problem.

While a direct problem describes a Cause-Effect relationship, an Inverse Problem (IP) consists in retrieving the causes responsible of some observed effects. For example, inferring gene regulatory networks from gene activity data or deriving some water constituents from the ocean colour are actually IP. GP has already been introduced as a method for solving IP, such as in [4,3,11,5]. However, IP are often far more difficult to solve than direct problems, since different causes may produce the same effect, *i.e.* the solution of an IP may be not unique. In that case, the IP is said to be redundant or ambiguous or, in a more formal way, ill-posed [1]. In the context of learning from datasets, a redundant IP corresponds to a Many-To-One mapping inversion, *i.e.* to a given input y_i , a set of outputs \mathcal{X}_i is expected. The purpose of this study is to enhance the inversion

¹ In fact, there are three sufficient conditions for ill-posedness which are the existence, the continuity and the redundancy of the solutions.



of Many-To-One mappings with GP. Here, the set of expected outputs \mathcal{X}_i is seen as a set of realisations of a random variable with an unknown probability density which has to be estimated. We propose two different ways to estimate the probability density of these answers and both are based on the possibility of producing multiple outputs with GP.

The section 2 highlights the limits of the classical Symbolic Regression approach for ambiguous IP and reviews the possibilities of producing multiple outputs with GP. Then, two original methods to tackle the redundancy problem are proposed in section 3 and tested on four benchmark problems in section 4. Finally, the application of this work and the possible further developments are discussed in the conclusion.

2 Inverse Problem and GP

2.1 Learning with Redundancy

With few hypothesis on the instructions set used to build programs, GP is an universal approximator [13] that can learn an arbitrary target function $t : X \mapsto Y$ from a dataset $D_t = \{(x_1, y_1) \dots (x_N, y_N)\}$. Here, we consider the pairs (x_i, y_i) as the realisations of two random variables \mathbf{x} and \mathbf{y} and we note f the function implemented by a GP program. When training a GP system to approximate an unknown mapping defined by D_t , an error function, such as for example the classical Root-Mean-Square Error, is minimized. It is known that in the theoretical case, this process leads to find the optimal answer $f^*(\mathbf{x})$ which is the conditional mean $E(\mathbf{y}|\mathbf{x})$, see [1] for details.

When solving an IP, the dataset \overline{D}_t used can always be seen as the set of the N reversed pairs (y_i, x_i) . If the direct function t is not injective, the learning of \overline{D}_t corresponds to a Many-To-One mapping inversion. Hence, for each y_i a set of of outputs \mathcal{X}_i is expected, and the single answer $f(y_i)$ given by GP can be very poorly adapted. Indeed, during the training phase, f tends to converge towards the theoretical optimal answer $f^*(\mathbf{x}) = E(\mathbf{x}|\mathbf{y})$, which is, in the better case, only one of the expected answer.

To illustrate this, we have created a dataset D_d from the target function $d : \mathfrak{R} \mapsto \mathfrak{R}$ such that $y = \sin(x^2) + \epsilon$, with ϵ a random variable with normal distribution $\mathcal{N}(0, 0.2)$. An (nearly good) approximation of the corresponding theoretical $f^*(\mathbf{x})$ is given by a standard GP system from D_d with $N = 500$ learning examples in the range $[-2, 2]$, see Figure 1. In a same way, in Figure 2, the output of GP corresponding to the inversion of d is plotted². We can see that GP has produced a very unstable function that highly overfits the dataset \overline{D}_d .

2.2 Multiple Outputs

In the GP field, several studies are related to IP solving (see for example [4,3,11,5]) but very few of them have investigated the question of the redundancy. However a

² The evolutionary parameters settings of the direct case were kept here.

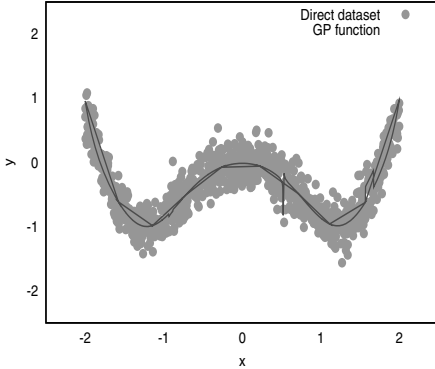


Fig. 1. Example of direct problem solving with Symbolic Regression

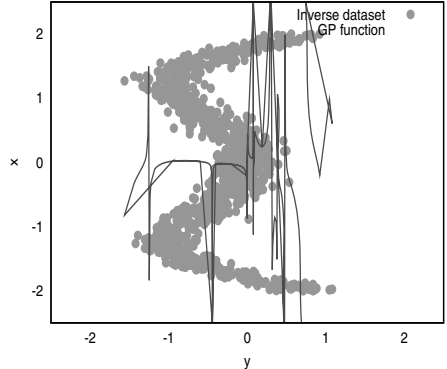


Fig. 2. Example of inverse problem solving with Symbolic Regression

noticeable exception can be found in [9] and will be discussed further. To overcome the non-uniqueness of the solution, instead of predicting the conditional mean, one way is to ensure that the output of an inverse model is at least one of the expected solutions [6]. A second idea is to produce a reduced list of illustrative examples [8] and a third possibility consists in approximating the distribution of the plausible solutions, that is the conditional probability density $p(\mathbf{x}|\mathbf{y})$ as explained in [1]. In this study, we are interested in the latter two possibilities, namely, those that require for GP to produce multiple outputs.

A lot of previous work in GP deals with multiple outputs. Probably the most straightforward way of doing this is by implementing sophisticated systems evolving programs able to manipulate a complex data structure such as a vector or a matrix [10]. Thus, one single GP program is responsible of producing multiple outputs. In [9], a technique based on boosting, that usually creates *a posteriori* mixtures of potential solutions, was extended to handle the ambiguity problem. Similarly, with the *Parisian* approach, used also to tackle an IP in [4], a subset of the population is selected to build the final answer. In these two previous examples, some GP programs are optionally associated to form a set of multiple outputs. Conversely, it is also possible to definitely link together several programs as being the co-operative members of a team and then, to make the whole team evolve [25]. In what follows, we will see how the multiple outputs of a team of programs can be used to estimate a probability density.

3 Density Estimation with GP

3.1 From Team to Probability Distribution

Practically, to solve an IP, a dataset \overline{D} consisting of pairs (y_i, x_i) is used and for a given input y_i , the goal is to approximate the distribution of the associated outputs \mathcal{X}_i . In the statistical inversion theory (see [12]), the pairs (y_i, x_i) are

considered as the joint realisations of two random variables \mathbf{y} and \mathbf{x} and the solution of an IP has the form of a conditional probability density $p(\mathbf{x}|\mathbf{y})$. We propose two different ways to approximate the subsequent distribution.

In the first approach, called Stochastic Realisation GP (SR-GP), for each input y_i , we consider the n outputs of a GP team T as the n realisations of an unknown random variable. We note f_j , the function implemented by the j^{th} member of T . With SR-GP, the evolutionary system try to find teams which outputs $f_j(y_i)$ report distributions similar to the distributions $p(\mathbf{x}|\mathbf{y} = y_i)$ for all the y_i of the dataset.

The second approach is called Mixture Density GP (MD-GP). Here, a parametric model, namely the finite Gaussian Mixture Model (GMM), is used as explained in [1]. An unknown density $p(\mathbf{x}|\mathbf{y})$ can always be represented as a finite sum of G Gaussian densities such as :

$$p(\mathbf{x}|\mathbf{y}) = \sum_{g=1}^G w_g(\mathbf{y}) \phi_g(\mathbf{x}|\mathbf{y})$$

with $\phi_g(\mathbf{x}|\mathbf{y})$ a normal density $\mathcal{N}(\mu_g(\mathbf{y}), \sigma_g(\mathbf{y}))$. In MD-GP, each of the $n = 3 \times G$ members of a team T approximates one of the functions ϕ_g , μ_g and σ_g that actually tune the GMM. It is worth noticing that, except for the means μ_g , the parameters of a GMM are constrained for a given y_i , since the deviations $\sigma_g(y_i)$ are positive real numbers, and since the weights $w_g(y_i)$ are also positive numbers but with $\sum w_g(y_i) = 1$. We note W and S , two functions that transform the team outputs into respectively valid $w_g(y_i)$ and $\sigma_g(y_i)$ parameters for GMM³. So, for a given input y_i , the answer of a team T is :

$$T(y_i) = \left\{ \begin{array}{lll} w_1(y_i) = W(f_1(y_i)) & \dots & w_G(y_i) = W(f_{n-2}(y_i)) \\ \mu_1(y_i) = f_2(y_i) & \dots & \mu_G(y_i) = f_{n-1}(y_i) \\ \sigma_1(y_i) = S(f_3(y_i)) & \dots & \sigma_G(y_i) = S(f_n(y_i)) \end{array} \right\}$$

Hence $T(y_i)$ is directly used to tune the GMM from which a set of r realisations can be produced (with usually $r \gg n$). With MD-GP, the evolutionary system tries to find teams tuning GMM, so that the GMM realisations according to y_i report distributions similar to the distributions $p(\mathbf{x}|\mathbf{y} = y_i)$.

Intuitively, we understand that in SR-GP, a huge number of parameters have to be retrieved, since the size of the teams have to be big enough to produce a sufficient number of realisations (probably more than 10^3) so that the subsequent distributions can be significantly tested. However, one can presume that with this representation, the search space is “smooth” since the modification of one team member only affects one realisation and so slightly modify the distribution. At the contrary, with MD-GP, fewer parameters have to be retrieved to properly tune the GMM (less than 30 in this paper) and so to produce significant results but it is clear that the modification of only one team member can induce important consequences on the fitness of the whole team.

³ In this paper, $S(\sigma_g(y_i))$ is simply the absolute value $|\sigma_g(y_i)|$ and similarly $W(w_g(y_i)) = |w_g(y_i)|/\sum |w_g(y_i)|$.

3.2 Construction of Target Distributions

The conditional probability density $p(\mathbf{x}|\mathbf{y})$ is unknown and only pairs (y_i, x_i) are available in a dataset \overline{D} . However, illustrative training distributions are required to properly educate the GP system. So for each input y_i , we have to build the probability distribution $p(\mathbf{x}|\mathbf{y} = y_i)$. Our idea is that even if a given value y_i is more likely present at most once in the dataset, many comparable values can be found. Thus here, for each value y_i , a set of k -nearest neighbours $\{y_i \dots y_j\}$ is computed. Then, for each value y_i of the dataset, the set $\{x_i \dots x_j\}$ is turned into binned data, by grouping the events into C specific ranges and so binned distributions are computed. The underlying assumption being that the distribution of the $\{x_i \dots x_j\}$ is similar to $p(\mathbf{x}|\mathbf{y} = y_i)$.

The dataset \overline{D}_d presented section 2.1 is extended to hold $N = 10^5$ pairs and in Figure 3, we have plotted three sets of the $k = 1000$ pairs $(y_i, x_i) \dots (y_j, x_j)$ corresponding to $y_i = -0.93, -0.19$ and 0.49 . In Figure 3, three binned distributions are drawn for $C = 50$. Each of them represents an approximation of the theoretical conditional probability $p(\mathbf{x}|\mathbf{y} = y_i)$ for $y_i = -0.93, -0.19$ and 0.49 .

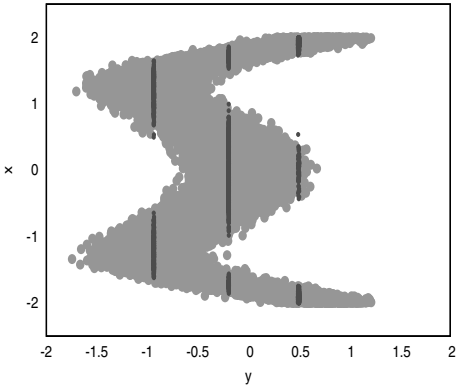


Fig. 3. Examples of pairs (y_i, x_i) corresponding to the k -nearest neighbours of $y_i = -0.93, -0.19$ and 0.49

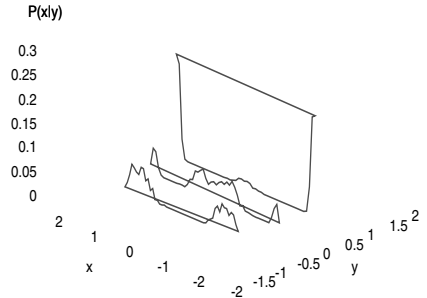


Fig. 4. Examples of binned distributions used to approximate $p(\mathbf{x}|\mathbf{y} = y_i)$, with $y_i = -0.93, -0.19$ and 0.49

3.3 Fitness Function

In this paper, only the fitness of the teams are computed and no effort has been made to estimate the fitness of the team members. Either with SR-GP or MD-GP, a team will be considered as a good team, if the set of the realisations produced for each y_i tested report a binned distribution comparable to the binned distribution obtained from a dataset \overline{D} , as explained above. To measure the distance between distributions, ϕ -divergences can be used, as for example the chi-square distance. Let be $U(y_i)$ the expected binned distribution computed from \overline{D} for a given y_i and $V(y_i)$, the distribution produced by a team of programs for the same y_i .

We note $U_c(y_i)$ and $V_c(y_i)$, the number of events in the bin c of the two distributions. The total number of events $\#U(y_i)$ is actually independent of y_i in this study and corresponds to the value k defined in section 3.2. Here, the partial fitness of a team T for one sample y_i is

$$E_T(y_i) = \frac{1}{\#U(y_i)} \sum_{c=1}^C \frac{(U_c(y_i) - V_c(y_i))^2}{U_c(y_i) + V_c(y_i)}$$

and the total fitness is simply the average of the $E_T(y_i)$ for all the y_i of a dataset \overline{D} .

4 Experiments

In this section, we aim to verify the ability of the SR-GP and MD-GP systems to approximate conditional probabilities and so to solve an IP. The linear stack-based GP implementation described in 5 is used to run the experiments but any other GP implementation suitable for Symbolic Regression can be used instead. The number of members in the teams is static and fixed *a priori*, see 2 for details. Different settings for the fitness function and the genetic operators but also various options for the representation and the conversion of teams are investigated. For each experiment, we perform 30 independent runs and a statistical unpaired, two-tailed t -test with 95% confidence determines if results are significantly different. Populations of teams are randomly created and the maximum creation size of the teams members is 50. The instruction set contains: the four arithmetic instructions ADD, SUB, MUL, DIV, the input variable Y and one stack-based GP specific instruction DUP which duplicates the top of the operand stack. We add also into the instructions set, two Ephemeral Random Constants noted ERC1 and ERC2, as described in 7, respectively in the ranges $[-100, 100]$ and $[-1, 1]$. The evolution is achieved with elitism, 4-tournament selection and steady-state replacement. Recombination is performed by the standard crossover operator with a rate of 0.7 and mutation with a rate of 1.0, meaning that each program involved in reproduction will undergo, on average, either one insertion or one deletion or one substitution.

Four problems P_a, P_b, P_c, P_d are tested. The corresponding datasets consist of $N = 10^5$ samples from which 100 binned distributions are computed as explained in section 3.2 and 80 are dedicated to the training phase. In fact, the two first test problems P_a and P_b do not correspond to the inversion of a direct function, but for P_a , the GP system has to approximate a bimodal distribution⁴ $\frac{1}{2}\mathcal{N}(\frac{1}{10}y_i^3 - y_i^2 + y_i + 9, 0.1) + \frac{1}{2}\mathcal{N}(5(y_i^3 - 3y_i^2) * e^{-y_i} - 3, 0.1)$ and for P_b , for each input y_i we have $\mathcal{N}(y_i, 2.3y_i^2 - 1.7y_i - 5.4)$. With an increasing difficulty, P_c and P_d are true IP that correspond respectively to the inversion of the function $c(x) = x + 0.3\sin(2\pi x) + \epsilon$ with ϵ , a random variable with uniform distribution in the range $[-0.1, 0.1]$ and to the inversion the function d described

⁴ This problem is strongly inspired by the work of Paris et al. 9

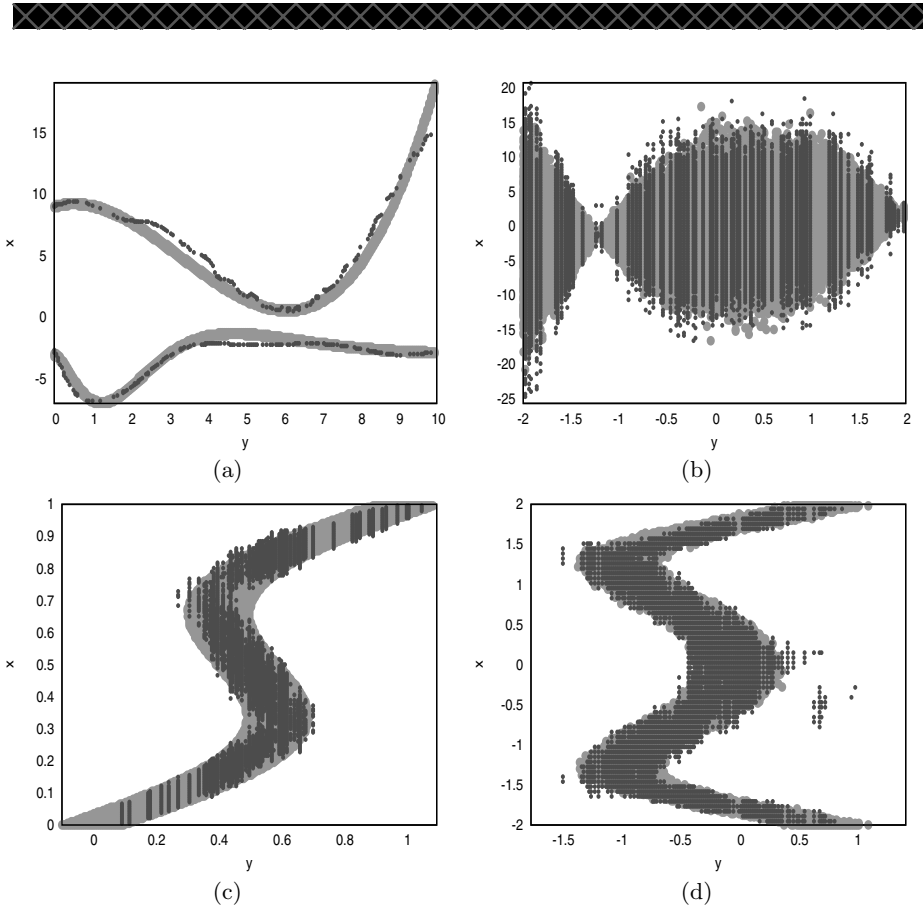


Fig. 5. Density estimation : four test problems

Table 1. Average fitness of the best programs found on the four test problems

GP Type	Pop. Size	Team Size	Max Size	No Gen.	k Real.	Train Fit.	Test Fit.
Problem P_a							
MD-GP	$9 \cdot 10^2$	2×3	$2 \cdot 10^2$	10^2	10	$0.43_{\sigma=0.16}$	$0.49_{\sigma=0.24}$
SR-GP		2				$0.22_{\sigma=0.09}$	$0.25_{\sigma=0.09}$
Problem P_b							
MD-GP	$9 \cdot 10^2$	4×3	10^2	10^2	10^3	$0.21_{\sigma=0.11}$	$0.23_{\sigma=0.14}$
SR-GP	10^4	10^3				$1.24_{\sigma=0.21}$	$1.43_{\sigma=0.34}$
Problem P_c							
MD-GP	$9 \cdot 10^2$	5×3	10^2	$2 \cdot 10^2$	10^3	$0.55_{\sigma=0.11}$	$0.58_{\sigma=0.14}$
SR-GP	10^4	10^3				$1.02_{\sigma=0.18}$	$1.03_{\sigma=0.24}$
Problem P_d							
MD-GP	$9 \cdot 10^2$	6×3	10^2	10^2	10^3	$0.28_{\sigma=0.06}$	$0.35_{\sigma=0.07}$
SR-GP	10^4	10^3		10^4		$0.42_{\sigma=0.15}$	$0.51_{\sigma=0.14}$

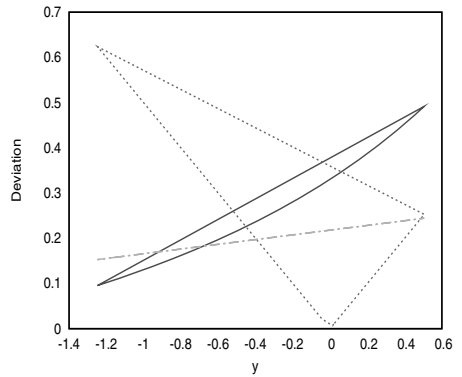
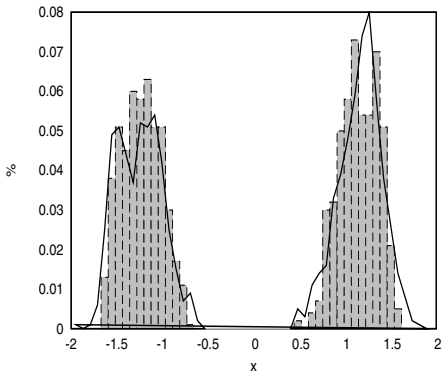
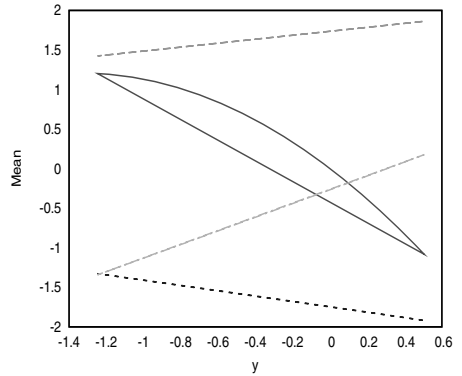
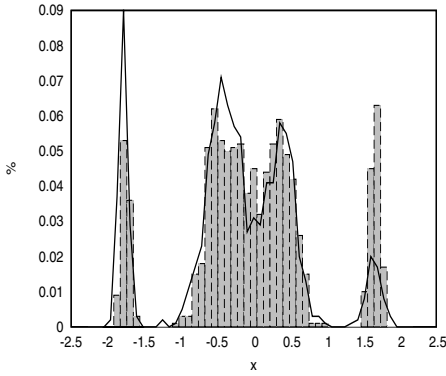
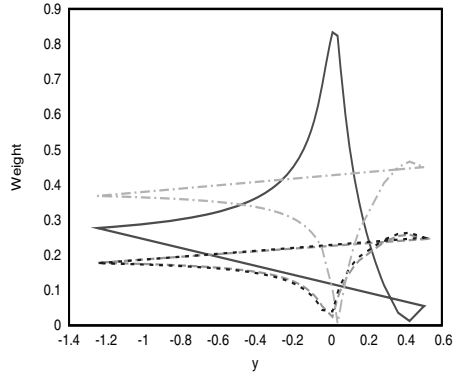
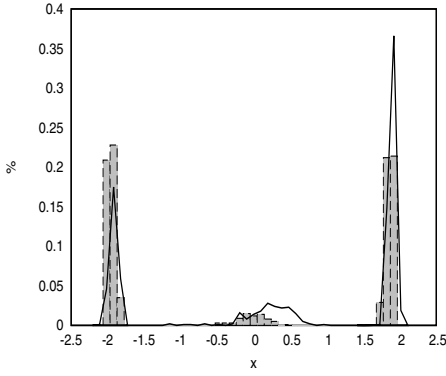



Fig. 6. Examples of density $p(\mathbf{x}|y = y_i)$ with, from top to bottom, $y_i = 0.49$, -0.93 and -0.19

Fig. 7. Example of Gaussian Mixture Model settings as a function of y with $G = 5$ for the problem P_d

section 2.1 The training samples of the four test problems, are plotted (gray points) in Figure 5, respectively part (a), (b), (c) and (d). The outputs (realisations) of representative of the best density estimations performed by our GP systems are also superimposed (in black).




We note from Table 1 that SR-GP is the best approach for tackling P_a even if we allow more generations to SR-GP. Actually, solving P_a consists more in retrieving two different functions than in a density estimation, and we think this is exactly the reason why SR-GP is more adapted. Conversely, MD-GP performs better on the three others problems. To figure out the quality of density estimation given by MD-GP, we have plotted for P_a , three examples of expected and retrieved distributions corresponding to $y_i = 0.49, -0.93$ and -0.19 , (see fig. 6), and we think that a good agreement has been obtained. The retrieved distributions are produced by three Gaussian mixtures with $G = 5$. The outputs of the corresponding GP team, consisting of 15 programs, is plotted in Figure 7. We can see (middle part), the expected 'Σ' shape realized by the 5 functions μ_g controlling the means of the Gaussian. Actually, the unwanted null-function has no influence in the model outputs since the corresponding weight Φ_g in the mixture is also null in the top part of Figure 7. We note that there are two different ways of almost "switching-off" a Gaussian since fixing a deviation σ_g to zero is another possible option for the system. This possibility should also be investigated for the approximation of non-continuous function with teams.

5 Conclusion and Perspectives

In many scientific domains, solving IP is now a major concern and the question of their redundancy requires particular answers. A wide range of methods can be used. Often based on a Bayesian approach, they actually report very good performances compared the two GP variants presented here, so that no inter-comparisons were made. However, this paper is a promising proof of concept. Indeed, we note that Symbolic Regression has been introduced probably more as a benchmark problem than as a true potential application for GP, but after more than a decade of improvements, the best programs found by GP are now competitive with others methods, which is very encouraging for SR-GP and MD-GP.

One of the difficulty in comparing with published work is that, for the two alternatives presented here, the fitness function used is a ϕ -divergence, very useful for computing distance between two distributions but unusual in GP. As far as we know, the only previous GP work addressing explicitly the redundancy problem [9], is much more appropriated for functions decomposition, as the problem P_a , than for more complex IP. Our SR-GP system can also easily solve P_a , while MD-GP is much more suitable to tackle IP where complex densities have to be estimated. So a broad variety of redundant IP can be addressed with SR-GP and MD-GP and we think that, for a given IP, preliminary statistical analysis of the dataset, will help to decide which method is adapted but also to *a priori* tune the systems parameters. Moreover, a lot of improvements can be made. We think that further work should address the possibility of : assigning a partial fitness to the team members, producing a variable number of realisations according to the inputs and designing genetic operators appropriate to teams, in particular to teams which members have different semantics in the system as the weights, the means and the deviations of the Gaussian Mixture Model.



References

1. C. M. Bishop. Mixture density networks. Technical report, Aston University, 1994.
2. M. Brameier and W. Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–407, 2001.
3. M. Chami and D. Robilliard. Inversion of oceanic constituents in case i and case ii waters with genetic programming algorithms. *Applied Optics*, 40(30):6260–6275, 2002.
4. P. Collet, E. Lutton, F. Raynal, and M. Schoenauer. Polar IFS+parisian genetic programming=efficient IFS inverse problem solving. *Genetic Programming and Evolvable Machines*, 1(4):339–361, 2000.
5. M. Defoin Platel, M. Chami, M. Clergue, and P. Collard. Teams of genetic predictors for inverse problem solving. In *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 341–350, Lausanne, Switzerland, 30 March - 1 April 2005. Springer.
6. M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16:307–354, 1992.
7. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
8. R. Morishita, H. Imade, I. Ono, N. Ono, and M. Okamoto. Finding multiple solutions based on an evolutionary algorithm for inference of genetic networks by s-system. In *Congress on Evolutionary Computation, 2003. CEC '03*, pages 615–622, 2003.
9. G. Paris, D. Robilliard, and C. Fonlupt. Genetic programming with boosting for ambiguities in regression problems. In *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 183–193, Essex, 14-16 April 2003. Springer-Verlag.
10. L. Spector. Autoconstructive evolution: Push, pushGP, and pushpop. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 137–146, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
11. F. Streichert, H. Planatscher, C. Spieth, H. Ulmer, and A. Zell. Comparing genetic programming and evolution strategies on inferring gene regulatory networks. In *Genetic and Evolutionary Computation – GECCO-2004, Part I*, pages 471–480, Seattle, WA, USA, 2004.
12. A. Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*. Elsevier, Amsterdam, Netherlands, 1987.
13. Xin Yao. Universal approximation by genetic programming. In *Foundations of Genetic Programming*, Orlando, Florida, USA, 13 1999.