ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Classification by means of evolutionary response surfaces

Rafael del Castillo-Gomariz and Nicolás García-Pedrajas

Department of Computing and Numerical Analysis
University of Córdoba (SPAIN)
e-mail: rcastillo@uco.es, npedrajas@uco.es

Abstract. Response surfaces are a powerful tool for both classification and regression as they are able to model many different phenomena and construct complex boundaries between classes. Nevertheless, the absence of efficient methods for obtaining manageable response surfaces for real-world problems due to the large number of terms needed, greatly undermines their applicability.

In this paper we propose the use of real-coded genetic algorithms for overcoming these limitations. We apply the evolved response surfaces to classification in two classes. The proposed algorithm selects a model of minimum dimensionality improving the robustness and generalisation abilities of the obtained classifier. The algorithm uses a dual codification (real and binary) and specific operators adapted from the standard operators for real-coded algorithms. The fitness function considers the classification error and a regularisation term that takes into account the number of terms of the model.

The results obtained in 10 real-world classification problems from the UCI Machine Learning Repository are comparable with well-known classification algorithms with a more interpretable polynomial function.

1 Introduction

Models based on response surfaces (RSs) are able to explain a wide variety of phenomena. The expression that defines a RS is a polynomial of degree G on each one of the input variables [1] [2]. So, the functions are of the form:

$$f(x_1, x_2, \ldots, x_n) = c_0 + \sum_{i=1}^{n} c_i x_i + \ldots + \sum_{\substack{i_1, i_2, \ldots, i_G = 1 \\ i_k \leq i_{k+1}}}^{n} c_{i_1 i_2 \ldots i_g} x_{i_1} x_{i_2} \ldots x_{i_g}, \quad (1)$$

where G is the degree of the model, x_i the input variables, n the number of inputs, and c_i the coefficients. For instance, for a quadratic RS (degree 2) the expression becomes:

$$f(x_1, x_2, \ldots, x_n) = c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{i,j=1}^{n} c_{ij} x_i x_j. \quad (2)$$

The main problem we face when using RSs in real-world problems is the number of terms. Even for a RS of degree 2 or 3, and a small number of inputs, the

number of terms is unmanageable large to deal with for most optimisation algorithms. In this way, the search space is too large and classical search algorithms, such as Levenberg-Marquardt [3], are frequently trapped in local minima, need long training times, and achieve poor results.

In such complex, noisy, non-differentiable, multi-modal and deceptive search spaces, evolutionary computation is a very good alternative [4] to classical search methods. In this paper we develop a real-coded genetic algorithm to evolve RSs that overcomes many of these problems.

If we want to use a RS and real-coded genetic algorithms to model any phenomenon we need a chromosome with as many genes as coefficients the model has. The number of coefficients depends on the number of variables and the degree of the polynomial. For example, in the model of eq. 2 the individuals have $3n + 1$ genes. Figure 1 shows an individual that represents a RS of degree 2 and 3 variables.

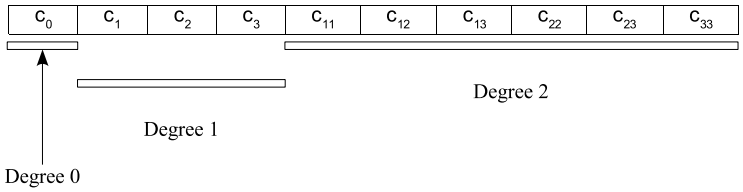

Fig. 1: Response surface of degree 2 and 3 input variables.

The robustness proved by RSs when applied to several problems together with their capacity to establish complex classification boundaries is the rationale behind our application of RSs to classification tasks.

2 Evolutionary response surfaces

Interpretability is a highly desirable property of any model. Simplicity is also a very important property. A simpler model is more interpretable, needs fewer patterns to adjust its coefficients, and its generalisation ability is usually better. In our algorithm we enforce the selection of simpler (with fewer terms) and accurate models. Each individual is codified with a gene for each coefficient of the model. This gene is formed by two different parts. On the one hand, there is one bit that represents the presence/absence of the term in the model. On the other hand, there is real value that represents the value of the coefficient for the corresponding term in case the term is present. Figure 2 shows an individual that represents a RS of degree 2 with 3 variables codified as explained.

This representation scheme is not enough, by itself, to enforce expressions with a minimum number of terms. In order to obtain the desired effect of preferring smaller polynomials we must include a term in the fitness function that rewards simpler models. In this way, our problem becomes an optimisation problem of two objectives: the accuracy of the model and the number of terms. Due to the fact that we have only two objectives we have opted for an unique

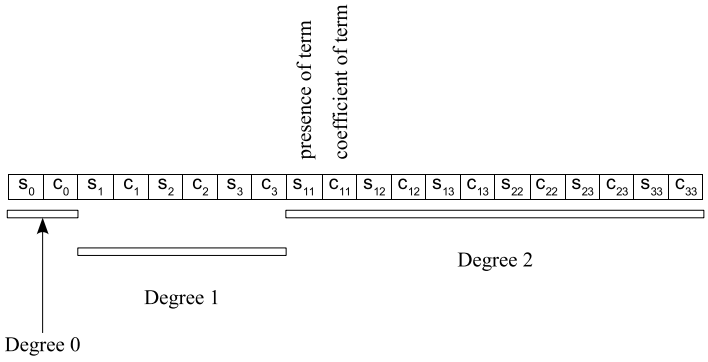

Fig. 2: Modified individual to allow the selection of the simplest model.

fitness function that is a lineal combination of two terms weighted by a coefficient given by the researcher. The fitness function, F, is given by:

$$F = (1 - \beta)F_{error} + \beta * F_{compl}, \tag{3}$$

where F_{error} is the error term, and F_{compl} is the complexity term, and β is a adjustable coefficient, $0 \leq \beta \leq 1$. The exact form of the two terms can be defined in different ways.

To classify a pattern we evaluate the function induced by the polynomial. If the value is greater than 0, the pattern is classified to the class labelled $+1$, otherwise to the class labelled -1. The result of applying the genetic algorithm is a binary classifier with a minimum number of terms within the family of polynomial functions and able to distinguish between two classes.

2.1 Genetic algorithm

The evolution of the population is carried out using a standard genetic algorithm. With a population size of N individuals, each generation P_dN individuals of the population are copied to the new population, P_cN are reproduced by crossover, and P_m are copied and undergo mutation. The values of these four parameters must be fixed by the user.

Each individual is codified by a gene formed by two parts. The first one codifies the real value of the coefficient of the term, and the second one is a selector that shows the presence/absence of the term in the model. Conceptually, we are working with a hybrid model that includes binary and real-valued values. In order to simplify the implementation, we have considered all the values real-valued. The selectors take values in the interval $[0, 1]$, and a term is not considered in the model if its corresponding selector is below 0.5.

As we have explained above, the fitness function consists of two terms: the first one is the classification accuracy of the model, and the second one is the complexity term, that is bigger as the number of terms, n_T, becomes smaller. The expression for the fitness is given by:

$$F = (1-\beta)\frac{\text{Patterns correctly classified}}{\text{Number of patterns}} + \beta\left(1 - \frac{n_T - n_{T_m}}{n_{T_M} - n_{T_m}}\right), \qquad (4)$$

where n_{T_M} and n_{T_m} represent, respectively, the maximum and minimum number of coefficients of the model. This is a monotonous fitness function that only achieves the maximum value, 1, if the classification accuracy is 100%, and the number of terms is n_{T_m}.

The number of genes of each individual depends on the degree of the RS chosen to perform the classification. In the experiments reported we have used RSs of degree 2.

2.2 Crossover operator

The proposed crossover operator is an adaptation of BLX-α [5]. We have used non-uniform mutation. These two operators are designed for real-coded genetic algorithms and have been adapted for our dual scheme. The adapted BLX-α uses two parents, $\beta^1 = \{(s_1^1, c_1^1), \ldots, (s_p^1, c_p^1)\}$ and $\beta^2 = \{(s_1^2, c_1^2), \ldots, (s_p^2, c_p^2)\}$, with p genes each one and representing two RS models with p coefficients. As we have explained each gene represents a term and a selector that tells whether the term is present, s_i^j, and the value of the coefficient, c_i^j. The two parents generate two descendants $\beta^{d_1} = \{(s_1^{d_1}, c_1^{d_1}), \ldots, (s_p^{d_1}, c_p^{d_1})\}$ and $\beta^{d_2} = \{(s_1^{d_2}, c_1^{d_2}), \ldots, (s_p^{d_2}, c_p^{d_2})\}$. The basic idea of the modification is that the genetic material of the best parent will have a higher probability of being inherited by the offspring. In this way, each gene of the two descendants, $(s_i^{d_1}, c_i^{d_1})$, and $(s_i^{d_2}, c_i^{d_2})$, is obtained by the algorithm depicted in Figure 3.

`Data:` Two genes: (s_i^1, c_i^1) and (s_i^2, c_i^2), from two parents β_1 and β_2 with fitness F_1 and F_2 respectively.
`Result:` Two genes: $(s_i^{d_1}, c_i^{d_1})$ and $(s_i^{d_2}, c_i^{d_2})$.

---

if round(s_i^1) = round(s_i^2) **then**

$s_i^{d_1} = s_i^{d_2} = \text{round}(s_i^1)$
$(c_i^{d_1}, c_i^{d_2}) =$ Apply BLX-α over (c_1^1, c_i^2).

else

Obtain n_1 and n_2 randomly in $\{1, 2\}$ with probability $\frac{F_1}{F_1+F_2}$ of having value 1, and probability $\frac{F_2}{F_1+F_2}$ of having value 2.
$(s_i^{d_1}, c_i^{d_1}) = (s_i^{n_1}, c_i^{n_1})$
$(s_i^{d_2}, c_i^{d_2}) = (s_i^{n_2}, c_i^{n_2})$

end if

Fig. 3: Genes of the two descendants of the adapted BLX-α crossover

In this way, the descendants inherit the terms common to both parents, and the coefficients of these common terms are obtained by means of a BLX-α operator. The terms that are present only in one of the parents are inherited

with a higher probability if the parent where they are present has a higher fitness value.

Standard non-uniform mutation is applied to the individual considering all the values as real numbers.

3 Experiments

For testing the validity of the proposed model we have selected 10 datasets from the UCI Machine Learning Repository. Each set of available data was divided into two subsets: 75% of the patterns were used for learning, and the remaining 25% for testing the generalization of the RSs. For each problem we run the algorithm 10 times using the following parameters:

Population	500 individuals		
F	$\beta = 0.5$		
Operators	Duplication	$P_d = 0.2$	Tournament selection
	Crossover	$P_c = 0.6$	Tournament selection
			Adapted BLX-α ($\alpha = 0.5$)
	Mutation	$P_m = 0.2$	Random selection
			Non-uniform mutation ($b = 5$)
Stop	500 generations		

We have compared our model with three well-known classification methods: a support vector machine (SVM) [6] with a Gaussian kernel, C4.5 classification algorithm [7], and a cascade-correlation neural network [8].

Table 1 shows the results in terms of test error for these three standard methods of classification and the RSs. The table shows the generalisation error of the different algorithms and the number of terms of the RSs. In boldface is shown the best results for each problem. The best result is obtained by means of a t-test at a confidence level of 95%.

Table 1: Summary of test errors for three standard methods of classification and the response surfaces of degree 2. The best result for each problem is highlighted in boldface.

Dataset	Standard methods			Response surfaces	
	SVM	C4.5	Cascade	Error	#Coefs.
breast-cancer	0.4226	0.3380	0.6020	**0.3099**	6.9
cancer	0.0345	0.0340	0.0400	**0.0247**	3.0
german	0.4000	**0.3320**	0.4144	**0.3316**	728.5
heart-c	0.4605	0.2500	**0.1465**	0.2408	43.7
heart	0.4559	0.2210	**0.1598**	0.1691	5.6
hepatitis	0.2369	0.1580	**0.1000**	0.1658	16.3
liver	0.3721	0.3840	0.4725	**0.3488**	3.0
pima	**0.2240**	0.2550	0.2590	0.2573	3.2
sick	0.0891	**0.0100**	0.0669	0.0568	31.6
vote	**0.0556**	0.0650	0.0614	0.0639	5.4

We can see how the RSs perform comparatively well. They are able to achieve significantly better results for 4 of the 10 problems. There is also interesting

to note that the obtained polynomials have few terms, with the exception of german problem, due to the large number of inputs. Furthermore, we must take into account that no local search algorithm is used. We believe that these results would be improved after the addition a local search algorithm that we are currently developing. This local search algorithm will be implemented as a parametric mutation operator.

4 Conclusions

In this paper we have shown how real-coded genetic algorithms offer a very interesting approach for using RSs for classification. RSs are a powerful tool for regression and classification but their applicability is undermined by the lack of efficient algorithms for optimising their parameters. We have shown that the structure of the polynomial and its parameters can be adjusted using a genetic algorithm. The evolution of both, parameters and structure, is achieved by the implementation of a dual codification scheme. Specific genetic operators have been developed for this codification.

We have proposed a fitness function that considers not only the accuracy of the classification, but also the simplicity of the obtained model. This fitness function yields simpler models that improve their interpretability and generalisation error.

The proposed model has been compared with three well-known classification algorithms with comparable performance. We can state that our model performs in this preliminary formulation, at least as well as these standard methods.

References

[1] J. O. Rawlings, S. G. Pantula, and D. Dickey. *Applied Regression Analysis: A Research Tool*. Springer-Verlag, New York, USA, 1998.

[2] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization using Designed Experiments*. John Wiley & Sons, New York, USA, 2nd edition, 2002.

[3] D. W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[4] G. F. Miller, P. M. Todd, and S. U. Hedge. Designing neural networks. *Neural Networks*, 4:53–60, 1991.

[5] L. J. Eshelman and J. D. Schaffer. Real-coded genetic algorithms and interval-schemata. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, 1993.

[6] V. Vapnik. *The nature of Statistical Learning Theory*. Springer Verlag, New York, 1999.

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, 1993.

[8] S. E. Fahlman and C. Lebiere. The cascade-correlation architecture. In D. S. Touretzky, editor, *Advances in Neural Information Systems 2*, pages 524–532, San Mateo, CA, 1990. Morgan Kauffman.