Substructural Surrogates for Learning Decomposable Classification Problems: Implementation and First Results

Albert Orriols-Puig[1,2] **Kumara Sastry[2]**
David E. Goldberg[2] **Ester Bernadó-Mansilla[1]**

[1]Research Group in Intelligent Systems
Enginyeria i Arquitectura La Salle, Ramon Llull University

[2]Illinois Genetic Algorithms Laboratory
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana Champaign

Motivation

- Nearly decomposable functions in engineering systems
  - (Simon,69; Gibson,79; Goldberg,02)
- Design decomposition principle in:
  - **Genetic Algorithms (GAs)**
    - (Goldberg,02; Pelikan,05; Pelikan, Sastry & Cantú-Paz, 06)
  - **Genetic Programming (GPs)**
    - (Sastry & Goldberg, 03)
  - **Learning Classifier Systems (LCS)**
    - (Butz et al., 06; Llorà et al., 06)

Aim

- Our approach: *build surrogates for classification*.
  - **Probabilistic models** → Induce the form of a surrogate
  - **Regression** → Estimate the coefficients of the surrogate
  - Use the surrogate to classify new examples

- Surrogates used for efficiency enhancement of:
  - **GAs:**
    - (Sastry, Pelikan & Goldberg, 2004)
    - (Pelikan & Sastry, 2004)
    - (Sastry, Lima & Goldberg, 2006)
  - **LCSs:**
    - (Llorà, Sastry, Yu & Goldberg, 2007)

Outline

1. Methodology
2. Implementation
3. Test Problems
4. Results
5. Discussion
6. Conclusions

Overview of the Methodology

Structural Model Layer → **Surrogate Model Layer** → **Classification Model Layer**

Structural Model Layer

- Extract dependencies between attributes
- Expressed as:
  - Linkage Groups
  - Matrices
  - Bayesian Networks
- Example: x-or

Surrogate Model Layer

- Infer structural form and the coefficients based on the structural model

(Sastry, Lima & Goldberg, 2006)

Classification Model Layer

- Use the surrogate function to predict the class of new examples

Surrogate Model Layer

- **Input:** **Database of labeled examples with nominal attributes**
- **Substructure identified by the structural model layer**
  - **Example: [0 2] [1]**
- **Surrogate form induced from the structural model:**

$$\boxed{f(x_0, x_1, x_2) = c_I + c_0x_0 + c_1x_1 + c_2x_2 + c_{0,2}x_0x_2}$$

- **Coefficients of surrogates:** linear regression methods

Surrogate Model Layer

Map input examples to schema-index matrix

– Consider all the schemas:

$$\{\underbrace{0{*}0,\ 0{*}1,\ 1{*}0,\ 1{*}1}_{[0\ \ 2]},\ \underbrace{{*}0{*},\ {*}1{*}}_{[1]}\}$$

– In general, the number of schemas to be considered is:

Number of linkage groups

$$m_{sch} = \sum_{i=1}^{m} \left[\Pi_{j=1}^{k_i} \chi_{i,j} \right],$$

Cardinality of the variable jth of the ith linkage group

Surrogate Model Layer

- **Map each example to a binary vector of size m_{sch}**
  - For each *building block,* the entry corresponding to the schemata contained in the example is one. All the other are zero.

	[0 2]				[1]	
	0*0	**0*1**	**1*0**	**1*1**	***0***	***1***
010 →	**1**	0	0	0	0	**1**
100 →	0	0	**1**	0	**1**	0
111 →	0	0	0	**1**	0	**1**

Surrogate Model Layer

- This results in the following matrix

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m_{sch}} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m_{sch}} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m_{sch}} \end{pmatrix}$$

n input examples

- Each class is mapped to an integer

$$\mathbf{C} = \begin{pmatrix} c_1 & c_2 & \cdots & c_n \end{pmatrix}^T$$

Surrogate Model Layer

- Solve the following linear system to estimate the coefficients of the surrogate model

Input instances mapped to the schemas matrix → $\mathbf{A}\mathbf{x} = \mathbf{C}$ ← Class or label of each input instance

↑ Coefficients of the surrogate model

- To solve it, we use multi-dimensional least squares fitting approach:

  – **Minimize:** $\chi^2 = (Ax - C)^T \cdot (Ax - C)$

Overview of the Methodology

Structural Model Layer → **Surrogate Model Layer** → **Classification Model Layer**

Structural Model Layer

- Extract dependencies between attributes
- Expressed as:
  - Linkage Groups
  - Matrices
  - Bayesian Networks
- Example: x-or

Surrogate Model Layer

- Infer structural form and the coefficients based on the structural model

(Sastry, Lima & Goldberg, 2006)

Classification Model Layer

- Use the surrogate function to predict the class of new examples

Classification Model Layer

- **Use the surrogate to predict the class of a new input instance**

New input example ⇒ **Map the example to a vector *e* of length m_{sch} according to the schemas** ⇒ **Output = round ($x \cdot e$)**

Coefficients of the surrogate model

001 ⇒

0*0	0*1	1*0	1*1	*0*	*1*
0	0	1	0	1	0

⇒ **Output = $x \cdot (001010)'$**

Overview of the Methodology

Structural Model Layer → **Surrogate Model Layer** → **Classification Model Layer**

Structural Model Layer

- Extract dependencies between attributes
- Expressed as:
  - Linkage Groups
  - Matrices
  - Bayesian Networks
- Example: x-or

Surrogate Model Layer

- Infer structural form and the coefficients based on the structural model

(Sastry, Lima & Goldberg, 2006)

Classification Model Layer

- Use the surrogate function to predict the class of new examples

How do we obtain the structural model?

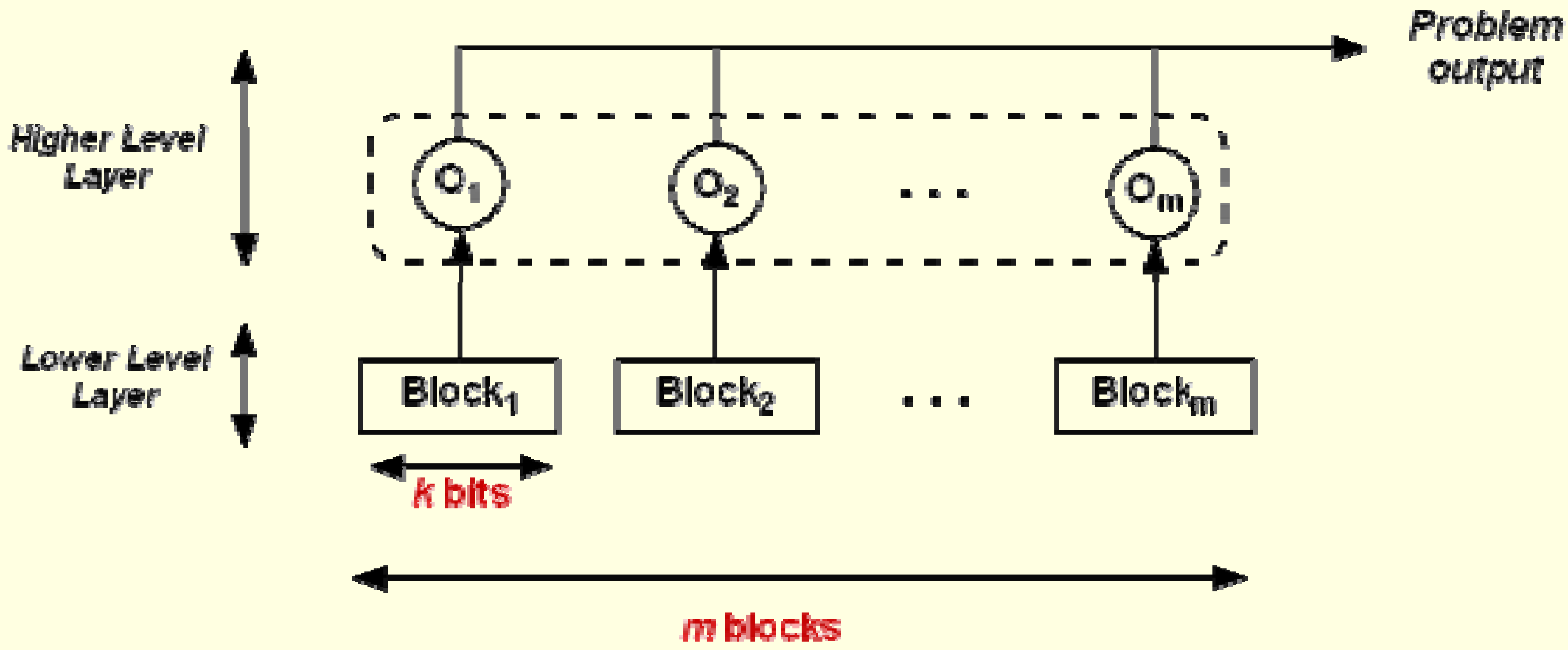
- **Greedy extraction of the structural model for classification (gESMC): Greedy search *à la* eCGA (Harik, 98)**

1. Start considering all variables independent. sModel = [1] [2] … [n]
2. Divide the dataset in ten-fold cross-validation sets: train and test
3. Build the surrogate with the train set and evaluate with the test set
4. While there is improvement
    1. Create all combinations of two subset merges

       Eg. { [1,2] .. [n];   [1,n] .. [2]; [1]..[2,n]}
    2. Build the surrogate for all candidates. Evaluate the quality with the test set
    3. Select the candidate with minimum error
    4. If the quality of the candidate is better than the parent, accept the model (update smodel) and go to 4.
    5. Otherwise, stop. sModel is optimal.

Test Problems

- **Two-level hierarchical problems** (Butz, Pelikan, Llorà & Goldberg, 2006)



Test Problems

Problems in the lower level

– **Position Problem:** ***Condition length (l)*** **000110 :3** — *Position of the left-most one-valued bit*

– **Parity Problem:** ***Condition length (l)*** **01001010 :1** — *Number of 1* **mod** *2*

Test Problems

- **Problems in the higher level**

  – **Decoder Problem:** *Condition length (l)* **000110 :5** — *Integer value of the input*

  – **Count-ones Problem:** *Condition length (l)* **000110 :2** — *Number of ones of the input*

- **Combinations of both levels:**

  – **Parity + decoder (hParDec)** - **Parity + count-ones (hParCount)**

  – **Position + decoder (hPosDec)** - **Position + count-ones (hPosCount)**

Results with 2-bit lower level blocks

- First experiment:
  - hParDec, hPosDec, hParCount, hPosCount
  - Binary inputs of length 17.
  - The first 6 bits are relevant. The others are irrelevant
  - 3 blocks of two-bits low order problems
  - Eg: hPosDec

10 01 00 00101010101

2 1 0

irrelevant bits

Output = 21

Interacting variables:
[x0 x1] [x2 x3] [x4 x5]

Results with 2-bit lower level blocks

- Performance of gESMC compared to SMO and C4.5:

	gESMC	**C4.5**	**SMO**
HPosDec	$0.00\% \pm 0.00\%$	$0.00\% \pm 0.00\%$	$0.00\% \pm 0.00\%$
HPosCount	$0.00\% \pm 0.00\%$	$0.00\% \pm 0.00\%$	$21.89\% \pm 0.13\%$
HParDec	$0.00\% \pm 0.00\%$	$3.32\% \pm 2.90\%$	$89.11\% \pm 0.94\%$
HParCount	$0.00\% \pm 0.00\%$	$5.15\% \pm 4.43\%$	$62.72\% \pm 0.27\%$

Results with 2-bit lower level blocks

- Form and coefficients of the surrogate models
  - Interacting variables [x0 x1] [x2 x3] [x4 x5]

HPosDec	*link. groups*	$[x_0x_1][x_2x_3][x_4][x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$16.75+9(1-\overline{x}_0x_1)-6\overline{x}_2\overline{x}_3-3\overline{x}_2x_3-1.5x_4+0.5x_5$
HPosCount	*link. groups*	$[x_0x_1][x_2x_3][x_4][x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$0.75+\overline{x}_0x_1+(1-\overline{x}_2x_3)+0.5\overline{x}_4+0.5x_5$
HParDec	*link. groups*	$[x_0x_1][x_2x_3][x_4x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. func.	$2+4(\overline{x}_0x_1+x_0\overline{x}_1)-2(\overline{x}_2\overline{x}_3+x_2x_3)-(\overline{x}_4x_5+x_4\overline{x}_5)$
HParCount	*link. groups*	$[x_0x_1][x_2x_3][x_4x_5][x_6][x_7][x_8][x_9][x_{10}][x_{11}][x_{12}][x_{13}][x_{14}]$
	surr. function	$\overline{x}_0x_1+x_0\overline{x}_1+\overline{x}_2x_3+x_2\overline{x}_3+\overline{x}_4x_5+x_4\overline{x}_5$

  - Linkages between variables are detected
  - Easy visualization of the salient variable interactions
  - All models only consist of the six relevant variables

Results with 2-bit lower level blocks

- C4.5 created trees:
  - **HPosDec and HPosCount**
    - Trees of 53 nodes and 27 leaves on average
    - Only the six relevant variables in the decision nodes
  - **HParDec**
    - Trees of 270 nodes and 136 leaves on average
    - Irrelevant attributes in the decision nodes
  - **HParCount**
    - Trees of 283 nodes and 142 leaves on average
    - Irrelevant attributes in the decision nodes
- SMO built support vector machines:
  - 351, 6, 6, and 28 machines with 16 weights each one were created for HPosDec, HPosCount, HParDec, and HParCount

Results with 3-bit lower level blocks

- Second experiment:
  - hParDec, hPosDec, hParCount, hPosCount
  - Binary inputs of length 17.
  - The first 6 bits are relevant. The others are irrelevant
  - 2 blocks of three-bits low order problems
  - Eg: hPosDec

000 100 00101010101

irrelevant bits

0 3

Output = 3

Interacting variables:
[x0 x1 x2] [x3 x4 x5]

Results with 3-bit lower level blocks

- Second experiment:
  - 2 blocks of three-bits low order problems

	gESMC	C4.5	SMO
HPosDec	0.00% ± 0.00%	0.00% ± 0.00%	0.00% ± 0.00%
HPosCount	0.00% ± 0.00%	0.00% ± 0.00%	14.24% ± 1.03%
HParDec	24.00% ± 25.50%	9.01% ± 5.86%	76.91% ± 2.01%
HParCount	49.99% ± 0.00%	12.25% ± 6.69%	49.94% ± 0.22%

M1: [0][1][2][3][4][5]

M2: [0 1][2][3][4][5]

M3: [0 1 2][3][4][5]

Interesting observations:

•As expected, the method stops the search when no improvement is found

•In HParDec, half of the times the system gets a model that represents the salient interacting variables due to the stochasticity of the 10-fold CV estimation. Eg. [0 1 2 8] [3 4 9 5 6]

Discussion

- Lack of guidance from low-order substructures
  – Increase the order of substructural merges

$$Cost = \binom{\ell}{2} \cdot s + \binom{\ell}{3} \cdot s + \cdots + \binom{\ell}{\ell_{max}} \cdot s$$

  – Select randomly one of the new structural models
    - Follow schemes such as simulated annealing (Korst & Aarts, 1997)

4. While there is improvement
   1. Create all combinations of two subset merges

      Eg. { [1,2] .. [n]; [1,n] .. [2]; [1]..[2,n]}
   2. Build the surrogate for all candidates. Evaluate the quality with the test set
   3. Select the candidate with minimum error
   4. If the quality of the candidate is better than the parent, accept the model and go to 4.
   5. Otherwise, stop. sModel is optimal.

Discussion

- Creating substructural models with overlapping structures
  - Problems with overlapping linkages
  - Eg: multiplexer problems
  - gESMC on the 6-bit mux results in: [0 1 2 3 4 5]
  - What we would like to obtain is: [0 1 2] [0 1 3] [0 1 4] [0 1 5]
  - Enhance the methodology permitting to represent overlapping linkages, using methods such as the *design structure matrix genetic algorithm (DMSGA) (Yu, 2006)*

Conclusions

- **Methodology to learn from decomposable problems**
  - Structural model
  - Surrogate model
  - Classification model
- **First implementation approach**
  - greedy search of the best structural model
- **Main advantage of gESMC over other learners**
  - It provides the structural model jointly with the classification model.
- **Some limitations of gESMC:**
  - It depends on the guidance on the low-order structure
  - Several approaches outlined as further work.

Substructural Surrogates for Learning Decomposable Classification Problems: Implementation and First Results

Albert Orriols-Puig[1,2]
Kumara Sastry[2]
David E. Goldberg[2]
Ester Bernadó-Mansilla[1]

[1]Research Group in Intelligent Systems
Enginyeria i Arquitectura La Salle, Ramon Llull University

[2]Illinois Genetic Algorithms Laboratory
Department of Industrial and Enterprise Systems Engineering
University of Illinois at Urbana Champaign