Feature Extraction from Optimization Data via DataModeler's Ensemble Symbolic Regression

Kalyan Veeramachaneni[1], Katya Vladislavleva[2], and Una-May O'Reilly[1]

[1] CSAIL, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[2] CentER, University of Antwerp, Belgium
katya@vanillamodeling.com kalyan@csail.mit.edu unamay@csail.mit.edu

Abstract. We demonstrate a means of knowledge discovery through feature extraction that exploits the search history of an optimization run. We regress a symbolic model ensemble from optimization run search points and their objective scores. The frequency of a variable in the models of the ensemble indicates to what the extent it is an influential feature. Our demonstration uses a genetic programming symbolic regression software package that is designed to be "off-the-shelf". By default, the only parameter needed in order to evolve a suite of models is how long the user is willing to wait. Then the user can easily specify which models should go forward in terms of sufficient accuracy and complexity. For illustration purposes, we consider a common design heuristic in serial sensor sequencing: "place the *most reliable sensor* last". The heuristic is derived based on the mathematical form of the objective function that lays emphasis on the decision variable pertaining to the last sensor's decision variable. Feature extraction on optimized sensor sequences indicates that the heuristic is usually effective though it is not always trustworthy. This is consistent with knowledge in sensor processing.

1 Introduction

We are interested in *"knowledge mining from optimization data"*. Our strategy is to use genetic programming symbolic regression to model optimization search points and their objective scores. The models are alternate explanations of the relationship between the optimization's decision variables and the objective function. In this paper, through modeling and model analysis, we propose a means of learning about the key players among the decision variables involved in an objective function. This mined information can be exploited in systems design.

The capability of symbolic regression via genetic programming has been continuously strengthened since genetic programming's inception. Much progress has been achieved in making it more accurate and reliable [1,2]. The technology has matured to the point that, for our knowledge mining, we use an "off-the shelf" and virtually parameterless software package called *DataModeler*.

We illustrate our approach via a problem in distributed sensor networks: sensors must be chained serially to minimize the error in network prediction. The sensor network must detect the presence (or absence) of a binary phenomenon

("smoke", "no smoke"). The solution requires two coordinated determinations: the best sequence to chain the sensors, and, the internal decision thresholds of each sensor in the chain. Because sensor optimization is costly in terms of the objective function, and because the coordinated determination requires bi-level search, practice sidesteps optimizing. Instead, mathematical analysis, made with assumptions of very simplified sensor behavior (i.e. independent sensor observations), yields a heuristic: "place the *most reliable sensor* last" [3].

We examine this heuristic first by learning from the optimization data. We ask: *Is it the case that the last sensor's threshold is the key driver in the objective function and hence the best sensor is placed there?*, *Can we identify the position in the sensor sequence whose corresponding decision variable is the key driver of the error?*, and *What happens when the sensors' observations are correlated?*. Next, we match up what we have learned from the data with sensor network design practice. When the situation is simple: the sensors are independently distributed, a simple analytic expression of the error can be mathematically derived. It is straight forward to identify the key variable driving this expression. Our methodology should indicate the importance of this same variable. Once we demonstrate the efficacy of our approach on simple sensor network, we set up the cases of correlated sensors. Theoretically, the heuristic is not applicable when sensor observations are correlated. However, it may frequently do quite well depending on the precise sensor error characteristics and correlations.

The paper is organized as follows: The details of sensor fusion, sensor performance modeling and sensor network decision error are fairly complicated. Thus we first frame how we proceed with our illustrative problem in Section **??**. Our intent is that this framing provides the reader with our motivation in describing the essential details of the serial sensor network problem as provided in Section 2. Section 3 gives the details of the ParetoGP based ensemble design methodology followed by description of the design framework of the learning from optimization data methodology. Section 4 presents the results on specifically designed test problems in serial sensor networks. Conclusions and future work are presented in Section 5.

2 The Serial Sensor Network Problem for Binary Phenomena

Consider the example of smoke detectors placed in a room. In such an environment with a binary phenomena, sensors collect measurements from their spatial observation region. The measurement is information to the network decision of whether the phenomena (smoke or no smoke) is present or absent in the room. The decision must be reached collectively because each sensor is noisy: some measurement values will be sensed in both the absence or presence of the phenomena.

All sensors can transmit a measurement to a central decision point or they can pass a measurement onward in a chain that allows all measurements to eventually reach the decision point. This latter case is termed "serial" and is the

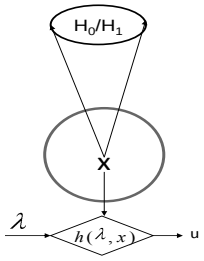


Fig. 1. Decision making process at the sensor

case we consider. However, the bandwidth required to transmit measurements usually prohibits sending them. Alternatively, serial sensors transmit their *decision* about the absence or presence of the phenomena. This requires only one transmission bit. Each sensor uses a threshold λ to arrive at a decision u_i using

$$h(\lambda, x) = \begin{cases} 1 & \text{if } x \geq \lambda \\ 0 & \text{if } x < \lambda \end{cases} \tag{1}$$

Figure 1 shows the decision making process at the sensor. When arranged in serial network, each sensor reaches chooses a threshold for the $h(\lambda, x)$ function based on the decision from its neighbor, and arrives at its decision by consulting its measurement. This process is illustrated in Figure 2.

While this saves bandwidth, it can degrade the accuracy of the system due to loss of information at each sensor. In the serial case, the accuracy can be regained by careful design in two ways: 1) setting up the sequence of the sensors in terms of "who reports to whom" and 2) designating the thresholds each sensor uses to make a decision.

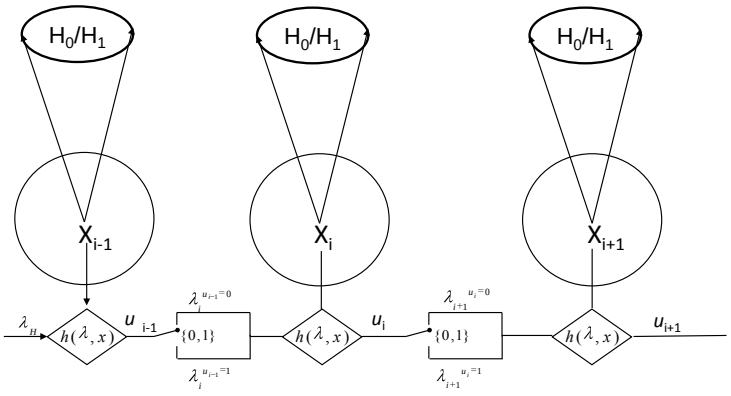


Fig. 2. Illustration of a decentralized serial sensor network

Figure 2 shows a decentralized sensor network model. All sensors observe a phenomena H and arrive at a measurement regarding the phenomena denoted by x_i. The decision module D_i applies a threshold to its observation and arrives at a decision u_i. Under these conditions sensors are said to be performing a binary hypothesis testing problem. The sensor makes a decision u_i based on its own observation x_i and the decision transmitted to it from the previous sensor denoted as $u_i - 1$.

Each sensor, except the first, has two thresholds $\lambda_i = [\lambda^{u_{i-1}=0}, \lambda^{u_{i-1}=1}]$ that it uses to apply to its measurement and arrive at a decision. $\lambda^{u_{i-1}=0}$ is applied if the incoming decision from the previous sensor is '0', i.e., $u_{i-1} = 0$, otherwise the threshold $\lambda^{u_{i-1}=1}$ is applied. Hence inherently each sensor fuses its information with the information from the previous sensor.

The sensor network can make a (system-wide) erroneous decision in two ways: It can issue a *false alarm* (smoke when no smoke present) or it can *miss* raising the alarm (no smoke when smoke present). These errors can be quantified probabilistically as P_{FA} and P_M respectively:

$$P_{FA} = P(H_0)P(u_n = 1|H_0), \tag{2}$$

and

$$P_M = P(H_1)P(u_n = 0|H_1) \tag{3}$$

Above $P(H_h)$ is *a priori* of occurrence of H_h, and $P(u_0 = d|H_h)$ is the probability of final decision being d when hypothesis H_h is present. Notice the mismatch between the h and d in the two equations. These define the errors. The goal of sensor network design is to minimize these two errors. Both the objectives are conflicting and are usually tied into a single objective function called Bayesian risk given by

$$R = C_{FA}P_{FA} + C_M P_M \tag{4}$$

where C_{FA} is the cost of false alarms and C_M is the cost of misses. Their sum, $C_{FA} + C_M$ is set to a constant value c. Each sensor's thresholds play a role in this objective function. We will derive the error expressions for 3 sensor problems with different sensor error models in Section 2.2. [4] provides additional details and information.

The next section proceeds to describe different sensor models in terms of sensor error. We use this quantity when we refer to the sensor's reliability. It allows us to state that one sensor is more reliable than the others. We measure reliability by the area under a sensor's ROC curve. Another ROC curve property - concavity is a way of further differentiating kinds of sensor reliability and captures sensor error interactions. Our test problems are further differentiated by this property.

2.1 Sensor Error Models

Using available data sets, it is possible to model a sensor's measurements under the phenomena's absence or presence. These models are generated and used by

the system designer. A measurement x, is a random variable in the model and variations in x are transformed into a probability density function. For example, one can assume the random variable follows a Gaussian distribution under both H_0 and H_1. Figure 3 shows the distribution of x given H_0, i.e. $p(x|H_0)$ and x given H_1, i.e., $p(x|H_1)$. The threshold is shown by a vertical line and can be varied affecting the two errors known as probability of false alarm and probability of miss. The two errors for given threshold λ of a sensor i are given by

$$p_{fa}{}^{i} = \int_{\lambda}^{\infty} p(x|H_0), \tag{5}$$

and

$$p_{m}{}^{i} = \int_{-\infty}^{\lambda} p(x|H_1). \tag{6}$$

A sensor's decision performance is characterized using a Receiver Operating Characteristic (ROC) curve which is a plot with p^i_{fa} on the *x-axis* and $1 - p^i_m$ (labeled as probability of detect) on the *y-axis*. The higher the curve, the lower the errors the sensor makes. A sensor's reliability hence can be quantified using the area under this curve.

For our illustration, we further differentiate sensor reliability based on the concavity of an ROC curve. We form sensor suites for our test problems that either contain strictly concave ROC curves or have non-concave ROC curves. We assume sensors observations to be normally distributed (Gaussian). This implies that a sensor can be characterized by four parameters μ_0, μ_1,σ_0, σ_1, where μ_h is the mean for the normal distribution under hypothesis h and σ_h is the standard deviation under hypothesis h.

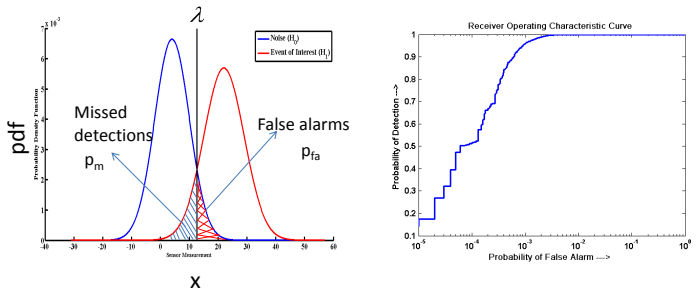


Fig. 3. A Sensor Model (left) and its Receiver Operating Characteristic (ROC) performance curve (right).

Using ROC curve area to characterize the magnitude of sensor reliability and ROC curve concavity to further characterize relative sensor reliability, it is

possible to identify a "most reliable" sensor in each of our test problems and place the most reliable sensor first or last in the sequence. The thresholds of this sequence are now ready for optimization where the optimization objective is minimum decision error. In the next section we provide the decision error (which is a function of sensor reliability models) calculation.

2.2 Threshold Optimization

There are $2N-1$ thresholds for an N sensor problem. For the 3 sensor problem there are 5 thresholds. Let p_i represent the false alarm probability corresponding to a threshold. Table 1 shows false alarm probabilities and their corresponding order in the 3 sensor sequence.

Table 1. Notation for Thresholds and False Alarm Probabilities for 3 Sequenced Sensors

Threshold	P_{FA}	Sequence Order
λ_1	p_1	1
λ_2	p_2	2
λ_3	p_3	2
λ_4	p_4	3
λ_5	p_5	3

We derive these errors for a simple three sensor problem under different combinations of assumptions with respect to independence and correlation. When we optimize a sequence of sensors, we use the appropriate error calculation.
Sensors with Independent Observations When sensors are assumed to report independent observations, the estimate for system wide probability of false alarm is

$$P_{FA} = p_5(p_3(1-p_1)+p_2p_1)+p_4((1-p_3)(1-p_1)+(1-p_2)p_1) \qquad (7)$$

Note that all the values p_i are probabilities and hence are less than 1. We can see from the above equation that p_5 and p_4 are multiplied on the outermost bracket. Hence, the lower the values of p_5 and p_4, the better is the performance of the system. Also, p_4 and p_5 are associated with the last sensor in the sequence, hence it is fair to assume that the last sensor plays an important role in the objective function. One should note that the threshold corresponding to p_4 is typically lower than the one corresponding to p_5. This implies that $p_5 << p_4$.

The expression uses false alarm probabilities that depend on the sensor error properties and where the thresholds are placed. Thus, when we need to analyze its behavior in terms of variable influence, we must do so under the different sensor error properties we described in Section 2.1:

– **Independent: Strictly Concave ROC Curves** This case implies that for any pair of sensors, their ROC curves do not intersect. For the simple Gaussian case this is true when $\sigma_0 = \sigma_1$ and this condition holds true for all the sensors. Under this case, the better sensor of the three will have pairs of (p_{FA}^i, p_M^i) that are lower than the other two and placing it as the final sensor in the sequence achieves lower error.
– **Independent: Non-Concave ROC Curves** For a simple Gaussian case, non-Concave ROC curves exist when the standard deviation under H_0 and H_1 is not the same. Under such conditions the decision as to which sensor should be placed at the end is not straight forward and depends on several factors including the degree of concavity in the ROC curve.

Sensors with Correlated Observations For sensors with correlated observations the error expression changes and gets much more complicated. It involves evaluation of multiple multivariate integrals. For the three sensor problem the P_{FA} is summation of six multivariate integrals and is calculated as follows. Let $\{t_1, t_2, t_3, t_4, t_5\}$ be the five thresholds for 3 sensors. The system wide probability of false alarm is given by

$$P_{FA} = \int_{t_5}^{\infty}\int_{t_3}^{\infty}\int_{-\infty}^{t_1} f(x_1, x_2, x_3|H_0) + \int_{t_5}^{\infty}\int_{t_2}^{\infty}\int_{t_1}^{\infty} f(x_1, x_2, x_3|H_0)$$
$$+ \int_{t_4}^{\infty}\int_{-\infty}^{t_3}\int_{-\infty}^{t_1} f(x_1, x_2, x_3|H_0) + \int_{t_4}^{\infty}\int_{-\infty}^{t_2}\int_{t_1}^{\infty} f(x_1, x_2, x_3|H_0), \tag{8}$$

where $f(x_1, x_2, x_3|H_0)$ is the joint density function under hypothesis H_0.

This complexity is precisely why the heuristic is used. It is difficult to separate the multivariate integral into a multiplicative term of three separate terms of probability to determine where to place the most reliable sensor before optimizing the thresholds.

However, when following a heuristic, one should be careful about when it might fail. Our knowledge mining of threshold drivers of error for a sequence should aid the heuristic when the objective function is not transparent. We can model the optimization data archived for a small set of iterations. Using our approach we can identify the key variable, i.e. λ. We can then identify the position in the sequence and place the most reliable sensor there. However, decision as to which sensor to place at this position is dependent on individual sensors reliability, reliability differential between the sensors in the network, and correlation between the sensors.

3 Modeling Optimization Data and Feature Extraction

At this point we have described the serial sensor problem as that of optimally sequencing sensors and setting their thresholds to minimize network decision

error. We have described sensor error models which allow us to set up test problems where we know the most reliable sensor and can place it in the first or last position. We have stated the decision error function for each test problem in terms of independent error or correlated observations. Let us now proceed to consider how post-run optimization data is modeling and analyzed.

Figure 4 shows how we model with optimization data. A particle swarm optimization [5] stores the points it generates into an archive and the modeler accesses the archive.

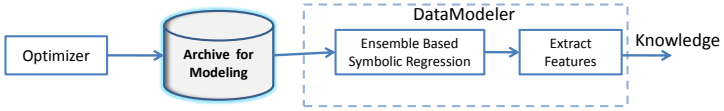


Fig. 4. Knowledge Mining Integrated with Optimization

Symbolic regression by GP learns both model structure and model parameters via an integrative process that adaptively searches for a model of minimum error. In pareto-GP [1, 6–13], the conventional GP algorithm is extended to optimize two objectives simultaneously. The primary objective is high model accuracy and the secondary objective is low model complexity. High complexity is synonymous with lack of generality because highly complex models over-fit the data. The multi-objective algorithm derives a suite of models that trade off predictive accuracy with generality. From the resulting suite of models, an ensemble that meets some combination of accuracy and complexity thresholds is selected. Feature selection proceeds by examining each model and accumulating the frequency of a decision variable appearing within the model set. Figure 5 shows the extraction of the key variables via the variable presence table in *DataModeler*. This form of "ensemble-based" symbolic regression supports the principle that no plausible model structure should be eliminated from explaining the input data.

DataModeler is an "off the shelf" add-on for Mathematica [14, 15] for ensemble based symbolic regression with genetic programming. It has a general purpose design. It generates a model ensemble for a user supplied set of data points presented in table form via a file name. To the user it outputs a suite of models that differ in their accuracy and complexity. To obtain the model suite, the user only has to tell *DataModeler*, first, in seconds, how long she is willing to wait and how many independent runs should be evolved. After that time interval, the user accesses two *DataModeler* built-in functions that allow a subset of models to be 1) down-selected by accuracy and complexity thresholds and 2) examined for variable presence frequency. Variable presence frequency provides the measure of the significance of the variable that we use. Complexity is by default defined as the quantify of subtrees nodes belonging to each node of the expression but can be specialized by the user. The software also embodies a

concept of robustness for modeling - if you can't find accurate models quickly, it is unlikely you will find them with more effort - you can only expect a slight fine tuning of accuracy. Many option defaults are set "under the hood" but they are accessible to the user.

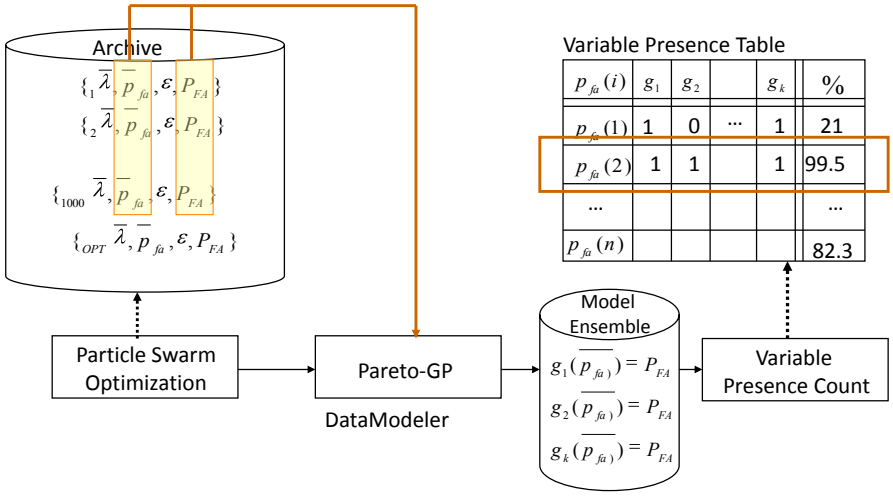


Fig. 5. Knowledge mining from optimization data and feature extraction for the sensor network problem

4 Threshold Drivers in Serial Sensor Network Design

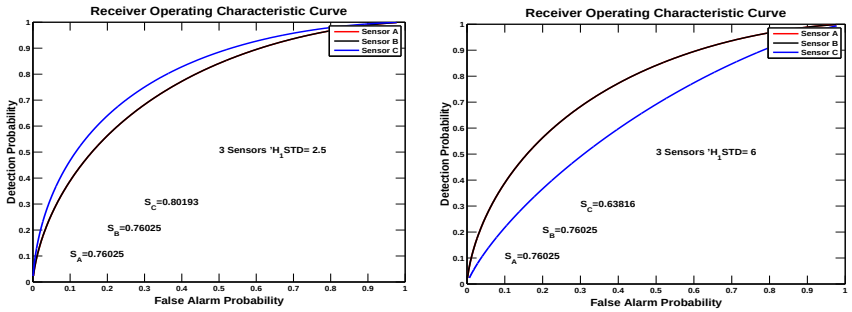
Having (finally) set up our problem domain, described the means by which optimization data can be modeled with ensemble based Pareto-GP and described how features can be extracted from the models to indicate variable drivers, we now experiment with our test problems and discuss our findings. Recall that our goal is to examine the match between the heuristic and the knowledge we learn from the optimization data. The heuristic refers to the "most reliable" sensor. Experimentally we design a number of test problems involving 3 sensors, the minimum quantity sufficient for illustrative purposes. Test problems differ in terms of how the sensor are modeled with respect to sensor-to-sensor correlation and whether individual sensor error performance is independent.

In each test problem, two sensors, named A and B will have equal reliability and one sensor, named C, will be either more or less reliable than both of them. We name our test problems "Independent-Concave", "Independent-Convex" (referring to the ROC curve) and "Correlated". Gaussian models are assumed under both H_0 and H_1. Each sensor under this assumption is characterized by μ, σ_h, where $h = \{0, 1\}$ corresponding to hypothesis '0' or '1' and μ is the mean and σ

is the standard deviation. We optimized all sequences' thresholds and determine for which sequence the decision error is minimal.

For each threshold optimization problem we execute a particle swarm optimization (PSO) for 50 iterations and collect as model decision variables the 5 probabilities of false alarms corresponding to the thresholds and, as the response variable, the corresponding system wide probability of false alarm. We used a swarm size of 20, hence there were 1000 points after 50 iterations. The points were then passed to the *DataModeler*. We specified only a compute budget of 300 seconds and 5 runs to generate the models. The model set generated in each run is then down-sized under an accuracy threshold. We then ask *DataModeler* to compute the variable presence analysis on the resulting models. This methodology is illustrated in Figure 5.

4.1 Test Problem 1: Independent-Concave



(a) (b)

Fig. 6. Test Problem 1: Sensor suite comprised of sensors with concave ROC curves. Sensors A and B perform identically well and the area under their ROC curves is 0.76. We generate two cases by controlling C's standard deviation under both H_1 and H_0. Problem 1(a)'s ROC curves are shown on the left hand side. C's area = 0.80. All 3 sensors' H_1 and H_0 have a standard deviation of 2.5. For Problem 1(b), ROC curves are shown on right hand side, C's area = 0.63 and sensor C's standard deviation under both H_1 and H_0 is increased to 6. In 1(b) C is not as reliable as A or B.

We use two different characterizations, 1(a) and 1(b), of sensor C's performance relative to A and B by increasing the standard deviation of all 3 sensors under both H_0 and H_1 while holding the means constant. This keeps the ROC curves concave. Figure 6 shows the ROC curves and areas. In both these characterizations, we observed that placing sensor C last (rather than first or middle) in the sequence results in minimal decision error. In the case where C is less reliable, the error differentials between placing C last, middle or first are almost inconsequential.

After optimization, in both test problems 1(a) and 1(b) we discover $P_{FA} = p_4$ as the of the most important threshold with a frequency in the model set of 99.7% or 98.4%. See Tables 2 and 3 for details. Recall, C is the most reliable sensor in case 1(a), but in case 1(b), C is actually *less* reliable that A and B. So the most important variables belong to the sensor placed last – regardless of whether it is more reliable or not. This is actually consistent with the mathematical analysis but shows how the heuristic be overly general. In this circumstance it is more accurate to design with the knowledge that "the last sensor's first threshold is the most important". However, because the decision error differential is so small between placing the most reliable sensor first or last, "place the best sensor last" practically speaking does no harm.

Variable	Sensor	Frequency (%)
p_5	C	96.9
p_4	C	99.7
p_3	B	99.3
p_2	B	93.6
p_1	A	97.3

Table 2. Test Problem 1(a) Threshold Frequency. Cross-reference to Figure 6(a)

Variable	Sensor	Frequency (%)
p_5	C	97.9
p_4	C	98.4
p_3	B	69.9
p_2	B	95.3
p_1	A	74.6

Table 3. Test Problem 1(b) Threshold Frequency. Cross-reference to Figure 6(b)

4.2 Case 2: Independent-Non-Concave

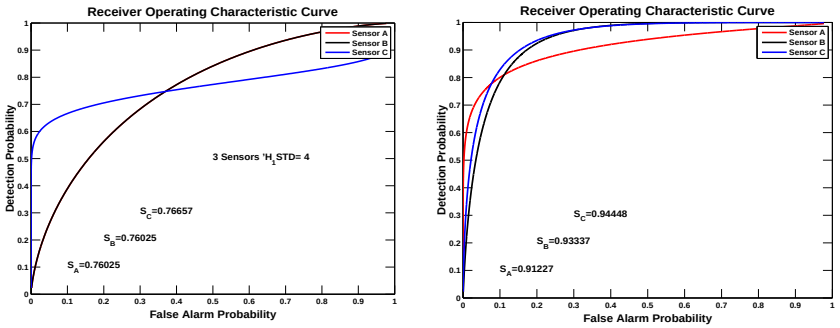
Variable	Sensor	Frequency (%)
p_5	C	93.2
p_4	C	99.7
p_3	B	98.4
p_2	B	90.6
p_1	A	96.5

Table 4. Test Problem 2(a) Threshold Frequency. Cross-reference to Figure 7(a)

Variable	Sensor	Frequency (%)
p_5	A	88.2
p_4	A	98.6
p_3	C	95.1
p_2	C	75.0
p_1	B	88.2

Table 5. Test Problem 2(b) Threshold Frequency. Cross-reference to Figure 7(b)

We now proceed to consider sensor errors which affect the relationship between the two ROC curves to make them intersect. This belies a complicated outcome in terms of network design. See Figure 7 for the two cases we set up. As in Test Problem 2(a), C is more reliable, though only by a slim margin (0.76 vs 0.77). While C is overall more reliable, there is a range: $P_{FA} > 0.4$ where the ROC curve of C is below A and B, and thus C is less reliable. Elsewhere (to the left and down), C is more reliable.



(a) (b)

Fig. 7. Test Problem 2: Sensor suite composed of sensors with non-concave ROC curves. In (a) Sensors A and B perform identically and the area under their ROC curves = 0.76. The standard deviation under H_1 for sensor C = 4. Its ROC area = 0.77. For the range of P_{FA} and $1 - P_M$ where the ROC curve of C is above A and B, C is more reliable. Elsewhere (to the right and up), C is less reliable. (b) presents a more complex relationship between A, B and C. They have ROC areas 0.91, 0.93 and 0.94 respectively and intersect each other in different points of the $P_{FA} and 1 - P_M$ axes.

We first experimentally observe that C in the last position in both cases (its reliability being better, (a), or worse, (b)) is the optimal sequence of the sensors. Using data from the two optimizations with C in the last position, we then regress models and extract features. See Tables 4 and 5 for details. Here we observe that threshold p_4 of C is again the strongest driver in the models at 99.7% and 99.5% frequency in the respective cases. This is consistent with a mathematical analysis. It also makes sense that C can be placed last, even if is not the most reliable: to the left and downward of where ROC curves intersect is where decision error is minimized and in this interval C actually performs more reliably than A and B.

In Test Problem 2(b), all 3 sensor's ROC curve areas differ: they have ROC areas 0.910, 0.93 and 0.94 respectively (so C is the most reliable). However, they intersect each other in different points of the $P_{FA} and 1 - P_M$ axes implying that C is not always the most reliable. When we optimized the thresholds with C as first, middle and last in the sequences, we discover that the decision error is minimal *when C is in the middle*. p_4 now the first threshold of sensor A is the most influential. This is consistent with the mathematical analysis for the independence of sensors assumption. The only time another variable other than p_4 will be most influential is when one sensor is far and away better than the other two and it was not placed at the end. In addition, notice that when the best sensor was placed in the most important position and the variance in the variable frequency was very low, the sequence indeed had the minimum error. For example. for cases 1(a) and 2(a), the best sensor C was placed the last, and this sequence was in fact the best performing sequence, the variance in the

frequency is 6.1% and 9.1%, whereas for 1(b) and 2(b) the variance was higher at 28.5% and 23.6%.

4.3 Test Problem 3: Correlated

For Test Problem 3(a) we ran the same sensor suite (non-convex, C most reliable) as in Figure 7(a) but this time additionally modeled a symmetric correlation of 0.5 between each pair of sensors. To get sufficiently accurate models, we collected data from 100 iterations which evaluated 10 particles/iteration and sent the data to *DataModeler*.

In this case, like the two previous test problems, C is more reliable and placing it last was observed to minimize decision error in both cases. However, in contrast to the previous test problems, the variable frequency is remarkably different. See Table 6 for details. First, p_1, associated with the threshold of the *first* sensor, is most influential. Its frequency is only 90% - quite lower most frequencies seen in Problem Sets 1 and 2. Secondly, the frequency of the other p_i's spans a large spread and there are clear distinctions among them. The ranking is p_5, p_4, p_2 and p_3 with frequencies of 79.9%, 65.7%, 39.7% and 20.5%. This can partially be explained by the presence of variable p_1 in every integral in the multivariate integral.

So, in this case "place the best sensor last" is valid but it is the threshold of the first sensor that influences the error minimization. This is slightly counter to design intuition when engineers are permuting the sensor sequence with knowledge of the error characteristics.

For Test Problem 3(b), we ran the same sensor suite (non-convex, C most reliable, A and B different) as in Figure 7(b) and additionally modeled a symmetric correlation of 0.5 between each pair of sensors. When we optimized the thresholds with C as first, middle and last in the sequences, we discover that the decision error is minimal *when C is first*. In this case, even knowing the theory ourselves, we could still not predict what the most influential threshold or p_i would be. Variable frequency analysis (see Table 7) reveals it to be the p_4=94.2% again. Next are $p_2 = 82.5\%$, $p_5 = 70.4\%$, $p_1 = 62.1\%$ and $p_3 = 1.0\%$. We were confident in advance that a correlated sensor error configuration would invalidate the heuristic. Our observation confirms this confidence. In fact, correlated sensor design will more frequently not follow the heuristic.

Variable	Sensor	Frequency (%)
p_5	C	79.9
p_4	C	65.7
p_3	B	20.5
p_2	B	39.7
p_1	A	90.8

Table 6. Test Problem 3(a) Threshold Frequency. Cross-reference to Figure 7(a)

Variable	Sensor	Frequency (%)
p_5	A	70.4
p_4	A	94.2
p_3	B	1.0
p_2	B	82.5
p_1	C	62.1

Table 7. Test Problem 3(b) Threshold Frequency. Cross-reference to Figure 7(b)

5 Conclusions and Future Work

We have demonstrated a means of knowledge discovery through genetic programming symbolic regression and feature extraction that exploits the sample history of an optimization run. Our results confirm approximate knowledge in serial sensor sequencing: the most influential thresholds in the sensor sequence are those of the final sensor when sensors are independently distributed. In sensor sequence design, the rule of thumb is to "place the *most reliable* sensor last". Our knowledge discovery shows how this design heuristic is usually effective though it is not trustworthy when the sensor observations are correlated.

We have worked with known error structure but when the underlying structure of the error is unknown (and the sensors are independent or not) or the network structure is more complicated (e.g. a tree) modeling the optimization data may provide more detailed insights into determining how to sequence the sensors. An infrequently occurring variable can likely be eliminated from the optimization. Or, if two thresholds of the same sensor fall a lot lower than the others, this sensor also is a candidate to be ignored. Sometimes it is helpful to know, at a system design level above network design, that no sensor can be ignored without a loss of information. This information is not apparent from the error objective but is indicated by variable presence table.

The evolutionary software we have used is within an "off-the-shelf" , mostly parameterless toolkit named "DataModeler". It requires a lot of background knowledge to understand the principles of symbolic regression and ensemble modeling but one does not have to refer at all to their implementation but it is flexible enough to let a EA expert control these if desired.

In terms of future work, we plan to purse whether we can integrate the modeling with our optimizer. We are aware this something similar been done in evolutionary algorithm parameter self-tuning. In this case, it is control parameters of the algorithm that are tuned. We would like to consider if the model can provide optimum predictions and queries of search points for which it has low confidence.

References

1. Vladislavleva, E.: Model-based Problem Solving through Symbolic Regression via Pareto Genetic Programming. PhD thesis, Tilburg University, Tilburg, the Netherlands (2008)
2. Keijzer, M.: Scientific Discovery Using Genetic Programming. PhD thesis, Danish Technical University, Danish Technical University (2002)
3. Papastavrou, J., Athans, M.: Distributed detection by a large team of sensors in tandem. Aerospace and Electronic Systems, IEEE Transactions on **28**(3) (1992) 639 –653
4. Veeramachaneni, K., Osadciw, L.: Swarm intelligence based optimization and control of decentralized serial sensor networks. (2008) 1 –8
5. Kennedy, J., Eberhart, R.: Particle swarm optimization. IEEE Conference on Neural Networks **4** (1995) 1942–1948

6. Smits, G., Vladislavleva, E.: Ordinal pareto genetic programming. In Yen, G.G., Lucas, S.M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.T., Coello, C.A.C., Runarsson, T.P., eds.: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, IEEE Press (2006) 3114–3120
7. Kotanchek, M., Smits, G., Vladislavleva, E.: Pursuing the pareto paradigm tournaments, algorithm variations & ordinal optimization. In Riolo, R.L., Soule, T., Worzel, B., eds.: Genetic Programming Theory and Practice IV. Volume 5 of Genetic and Evolutionary Computation. Springer, Ann Arbor (2006) 167–186
8. Kotanchek, M., Smits, G., Vladislavleva, E.: Trustable symoblic regression models. In Riolo, R.L., Soule, T., Worzel, B., eds.: Genetic Programming Theory and Practice V. Genetic and Evolutionary Computation. Springer, Ann Arbor (2007) 203–222
9. Kotanchek, M., Smits, G., Vladislavleva, E.: Exploiting trustable models via pareto GP for targeted data collection. In Riolo, R.L., Soule, T., Worzel, B., eds.: Genetic Programming Theory and Practice VI. Genetic and Evolutionary Computation. Springer, Ann Arbor (2008) 145–163
10. Kotanchek, M.E., Vladislavleva, E.Y., Smits, G.F.: Symbolic regression via GP as a discovery engine: Insights on outliers and prototypes. In Riolo, R.L., O'Reilly, U.M., McConaghy, T., eds.: Genetic Programming Theory and Practice VII. Genetic and Evolutionary Computation. Springer, Ann Arbor (2009) 55–72
11. Vladislavleva, E., Smits, G., Kotanchek, M.: Soft evolution of robust regression models. In Riolo, R.L., Soule, T., Worzel, B., eds.: Genetic Programming Theory and Practice V. Genetic and Evolutionary Computation. Springer, Ann Arbor (2007) 13–32
12. Vladislavleva, E.J., Smits, G.F., den Hertog, D.: Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. IEEE Transactions on Evolutionary Computation **13**(2) (2009) 333–349
13. Smits, G., Kordon, A., Vladislavleva, K., Jordaan, E., Kotanchek, M.: Variable selection in industrial datasets using pareto genetic programming. In Yu, T., Riolo, R.L., Worzel, B., eds.: Genetic Programming Theory and Practice III. Volume 9 of Genetic Programming. Springer, Ann Arbor (2005) 79–92
14. Research, W.: Wolfram mathematica overview: Compute and visualize key capabilities. (2009) www. wolfram.com/products/mathematica/overview/compute.html
15. Wikipedia: Mathematica entry in wikipedia. (2009) http://en.wikipedia.org/wiki/Mathematica

Acknowledgements

We would like to thank Guido Smits and Mark Kotanchek for their assistance and helpful discussions.