Supplemental Digital Content

A new severity of illness scale using a subset of APACHE data elements shows comparable predictive accuracy

Alistair E. W. Johnson, BS

Centre for Doctoral Training in Healthcare Innovation, Department of Engineering Science, University of Oxford, Oxford, UK and the Institute of Biomedical Engineering, Department of Engineering Science, University of Oxford, Oxford, UK

Andrew A. Kramer, PhD

Cerner Corporation, Vienna, VA

Gari D. Clifford, PhD

Department of Engineering Science, University of Oxford, Oxford, United Kingdom

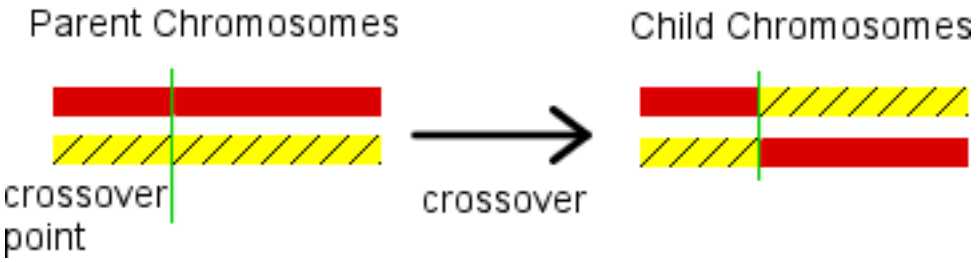
Appendix I. Description of the genetic algorithm methodology used in this study

In order to select variables yielding the most information for a severity score, we employed a machine learning technique known as genetic algorithms (GA) (1). GAs belong to a class of optimizers known as evolutionary algorithms (2), as they mimic real-life processes found in nature. The theoretical groundwork of the technique was originally developed in the late 1950s, and then used to solve a variety of practical engineering problems in the early 1970s (3, 4). The GA implementation in this work is based upon that of Siedlecki and Sklansky (5).

A brief description of the GA used in this work follows. Each one of the 37 potential measurements is represented as a binary (0, 1) variable: a so-called "gene". Initially genes are randomly assigned to be either 0 (off, or not included in the model) or 1 (on / included). The collection (vector) of genes is called a chromosome. The collection of genes is called a chromosome. There are a fixed number of chromosomes, each with 37 genes that make up what is known as the population. The number and location of the genes that are turned on varies across chromosomes.

Each chromosome receives a fitness value. Fitness is assessed by utilizing the genes that are turned on in a predictive model, e.g. logistic regression. In our study fitness was determined by a severity score developed using the Particle Swarm Optimization (PSO) technique described below. The GA operates by evaluating all of the chromosomes in a population. The population is then subjected to a "natural selection cycle" whereby a proportion of the chromosomes with the lowest fitness are removed. Conversely, a fixed proportion of the best performing chromosomes are left unchanged to ensure that the best combinations in the new generation perform at least as well as the previous generation. The remaining members of the population are then constructed via a process that mirrors genetic crossover. Chromosomes are matched via a fitness-weighted Monte Carlo method, and fragments from the distinct chromosomes are combined. Figure E1 shows an example of this process.

Figure E1. Schematic diagram of crossovers in genetic algorithms

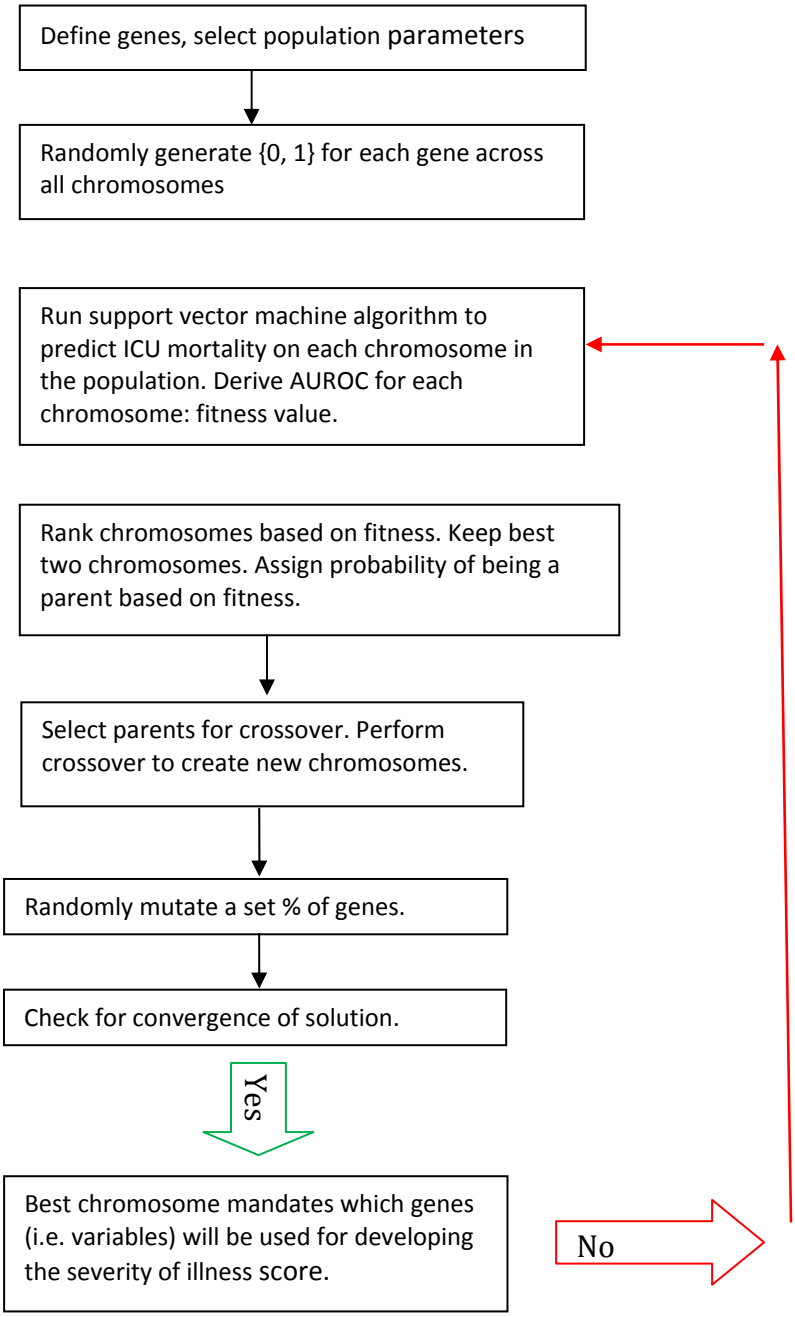


Finally, genes are randomly selected to "mutate". In the GA used here that meant reversing a gene's value, i.e. switched from on to off or vice versa. The resultant collection of chromosomes then

cycles through the evaluation-natural selection-crossover-mutation process. This continues from one iteration to the next until the best chromosome does not improve the AUROC. The genes on the best chromosome which are turned on represent the variables that are selected for creating OASIS.

A schematic diagram of the GA process is given in Figure E2.

Figure E2. The genetic algorithm process



Appendix II. Description of the particle swarm optimization methodology used in this study

To select the weights for the components making up OASIS we used particle swarm optimization (PSO), a non-linear optimization technique widely applied in the fields of bio-informatics and power systems (6, 7). PSO is a population based search in that it randomly initializes a set of candidate solutions ("particles") and iteratively updates them until the best solution found ceases to improve.

The implementation of PSO in this work now follows. First, 10 quantiles of the data were calculated for each variable, resulting in 10 ranges over which scores could be optimized. Initially 24 candidate solutions (particle "positions") were initialized by sampling from a uniform random distribution in the range (0,10). For each candidate solution, an additional update vector (particle "velocity") was initialized to 0. Finally, each particle has an associated prior best position which saves the best performing particle's position across all iterations. Thus, for each particle there is: a position (the values of the score), a velocity (used in the iterative update to the score), and a prior best position (also used in the iterative update to the score).

The algorithm then begins iteratively optimizing the candidate solutions. First, each particle position is assessed by the fitness function and assigned a score, where the fitness function is the AUROC of a score calculated on the development data set using the current particle position as the individual in the severity score. The ranges for the severity score are kept fixed using ten data derived quantiles. If the current particle position scores higher than its prior best position, then the prior best position is set to the current position. The best performing prior best position is then assigned as the globally best position. The velocity is then updated as a weighted sum of the current velocities, the difference between the current position and the prior best position, and the difference between the current position and the globally best position. These weights are randomly selected from the range (0, 2.05). Thus the solutions are encouraged to return to their previous best position (similar to a local search), and toward the global best position (similar to a global search), providing a flexible trade-off between these two paradigms. This process is repeated and the algorithm terminates when the global best position's score no longer improves. This global best position is the severity score output by the algorithm.

Figure E3 shows as an example how a weight for a patient was determined, in this case age. If the patient is aged 54, he/she is assigned an age component score of 7; if he/she is aged 23 then the component score is 1, etc… These scores are summed over all components to give the final OASIS score for the patient. The scores (1,4,7,10, and 8 in this example) were obtained through the PSO algorithm.

Figure E3. Example of Particle Swarm Optimization (PSO) scores developed for the age component.

$$s_{age} = \begin{cases} 1\ , & Age < 24 \\ 4\ , & 24 \leq Age < 53 \\ 7\ , & 53 \leq Age < 77 \\ 10, & 77 \leq Age < 90 \\ 8\ , & Age \geq 90 \end{cases}$$

References

1. Mitchell M : An Introduction to Genetic Algorithms, Third Printing, Cambridge, MA, MIT Press, 1997.

2. Sumathi S, Hamsapriya T, Surekha P : Evolutionary Intelligence, Berlin, Springer-Verlag, 2008.

3. Barricelli NA: Symbiogenetic evolution processes realized by artificial methods, *Methodos* 1957, 143–182.

4. Holland JH: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, Ann Arbor, MI, . University of Michigan Press, 1975.

5. Siedlecki W, Sklansky J: A note on genetic algorithms for large-scale feature selection, *Pattern Recognition Letters* 1989, 10:335–347.

6. Eberhart R, Kennedy J: Particle Swarm Optimization, *IEEE International Conference on Neural Networks* 1995; 4:1942–1948.

7. Kramer AA, Vida M: Evolutionary algorithms in the analysis of physiologic measures and their relationship to critical care outcomes, *J Crit Care* 2006, 21:357.