A Genetic Algorithm Approach to Predictive Modeling of Medicare Payments to Physical Therapists

Annie S. Wu
Computer Science
University of Central Florida
Orlando, FL 32816-2362
aswu@cs.ucf.edu

Xinliang Liu
Health Management and Informatics
University of Central Florida
Orlando, FL 32816-2205
Xinliang.Liu@ucf.edu

Reamonn Norat
Computer Science
University of Central Florida
Orlando, FL 32816-2362
ReaNorat@Knights.ucf.edu

Abstract

We examine the ability of a genetic algorithm to learn a predictive model that can estimate the likelihood that a physical therapist will receive annual Medicare payments above or below the industry median based on the physical therapist's practice parameters. We compare the performance of a canonical genetic algorithm and a self adaptive genetic algorithm with the performance of traditional logistic regression. Results show that both genetic algorithm approaches are competitive with logistic regression with the canonical genetic algorithm consistently outperforming logistic regression.

Introduction

In this work, we investigate the ability of a genetic algorithm (GA) (Holland 1975; Goldberg 1989) to learn a predictive model on when the total Medicare standardized payment amount received by a physical therapist (PT) will be above or below the industry median. Given a set of example data points with the correct classification, we use a GA to learn a model that can be used to predict the classification of future data points.

The ability to collect and analyze data at a large scale has transformed many industries including the health care industry. In the health care industry, this ability has made possible system-wide analyses across populations of providers that were previously impossible or intractable. The Centers for Medicare and Medicaid Services (CMS) has released information on services and procedures provided to Medicare beneficiaries by physicians and other health care professionals including PTs, with the goal of making the US healthcare system more transparent, affordable, and accountable (Brennan, Conway, and Tavenner 2014; mis). With the current widespread health care workforce shortage, it is of great interest to examine what factors may affect the service volume and payments of various health care providers.

Physical therapy services that strive to improve strength and mobility, restore and maintain physical function, and enhance health, well-being, and quality of life are covered by the Medicare Fee-for-Service (FFS) program under Part B. Few studies have examined the PT workforce participating in Medicare Part B. Although there is work that explores the factors associated with Medicare service volume and standardized Medicare payment (Liu et al. 2018), it remains unclear what factors affect a PT's likelihood of receiving an annual Medicare payment that is above the industry median.

We are interested in identifying and understanding correlations between PT practice parameters and expected costs billed to Medicare. Specifically, we would like to know if practice parameters may be used to classify whether the annual Medicare payments received by a PT will be above or below the industry median. The traditional analytical approach to such classification problems is logistic regression (LR), but evidence suggests that machine learning (ML) algorithms may also be effective approaches for these problems (Chaurasia and Pal 2014; Ciresan et al. 2012; de Vasconcelos et al. 2001; Ning et al. 2005; Thornblade, Flum, and Flaxman 2018; Waibel et al. 1989), potentially offering more precise solutions than traditional regression-based methods (Thornblade, Flum, and Flaxman 2018). GAs are a type of ML algorithm that have been successfully applied to classification problems, both alone and in conjunction with other ML approaches.

Multiple examples of GAs for classification exist (Fernández et al. 2010). These methods commonly revolve around the learning of a set of rules with which to make the classification. This rule-set commonly follows an IF-THEN or IF-THEN-ELSE paradigm. GA classification methods have been used on classification problems including the detection of breast cancer (Fidelis, Lopes, and Freitas 2000), multicriteria inventory classification (Guvenir and Erel 1998), as well as multiple common classification test sets (Dehuri et al. 2008; Fernández et al. 2010). Genetic programming, an evolutionary algorithm related to the GA, has been successfully used for the prediction of natural gas consumption of a chemical processing company (Kovačič and Dolenc 2016).

GAs have also been used to select the relevant features with which other ML methods build a predictive model. This approach has been used on a variety of classification problems. ML methods with which GAs have been combined include: random forests for outcome prediction on patients with oesophageal cancer (Paul et al. 2017), support vector machines (SVM) for bankruptcy prediction (Min, Lee, and Han 2006), neural networks for surgery outcome prediction

on patients with non-small cell lung carcinoma (Jefferson et al. 1997), linear discriminant analysis for prediction of firms that default on tax payments (Höglund 2017), and logistic regression (Zamuda et al. 2017; Zhang et al. 2018) for prediction of the progress of Alzheimer's disease (Vandewater et al. 2015). In these systems, GAs are sometimes used for other uses beyond feature selection; in the SVM example (Min, Lee, and Han 2006), the GA also optimizes the SVM parameters in addition to feature selection. In each of these examples, GAs were found to be an effective tool for feature selection.

We begin by describing the problem and how it is represented in the GA. We give a description of the two GA methods that we use, and an overview of the LR analysis. Finally, we compare the performance of all three methods on both the training and validation data sets.

Problem description

The problem that we study is a predictive modeling problem. Given a set of data points and a classification for each data point, we apply a GA to learn how the characteristics (or independent variables) of each data point are correlated with the classification (or dependent variable) of that data point. We represent this problem in the GA as a linear weighted sum. The goal of the GA is to evolve appropriate weights or coefficients that produce a correct classification via the sum.

Our data sources are the 2014 Medicare Provider Utilization and Payment Data: Physician and Other Supplier Public Use File (PUF) and the 2015-2016 Area Health Resources File (AHRF). Specifically, we identify 40,662 PTs from the 50 states and the District of Columbia who submitted Medicare Part B non-institutional claims in 2014 using the PUF and link the provider-level variables with county-level variables that reflect the characteristics of the health care market in which each of the PTs practice. Each of the 40,662 data points specifies the data for one PT provider, $P_r : r \in \{1, 2, ..., 40662\}$, which consists of:

- 25 independent variables, $v_{r,i} : i \in \{0, 1, ..., 24\}$, each of which describes a practice parameter for P_r. Table 1 lists the 25 independent variables.
- one dependent variable, B_r, which gives the full Medicare standardized payment amount received by P_r in 2014.

Let B_{median} be the median Medicare standardized payment amounts over all PT in 2014. Given $v_{r,i} : i \in \{0, 1, ..., 24\}$ for any P_r, we would like to be able to predict if B_r will be above or below B_{median}.

The GA addresses this problem as a linear weighted sum

$$Q_r = \sum_{i=0}^{24} c_i v_{r,i} \tag{1}$$

where $v_{r,i}$ are the independent variables for one data point (P_r) and c_i are a set of coefficients evolved by the GA. The goal of the GA is to evolve a set of coefficients $c_i : i \in \{0, 1, ..., 24\}$, such that the following is true for all data points, P_r.

$$Q_r = \begin{cases} > 0 & \text{if } B_r > B_{median} \\ \leq 0 & \text{if } B_r \leq B_{median} \end{cases} \tag{2}$$

i	**Independent Variable**
0	Female gender
1	Doctor of Physical Therapy degree
2	Number of HCPCS/CPT codes billed
3	Number of Medicare beneficiaries served
4	Charge to Medicare allowed amount ratio
5	Avg Medicare standardized payment amt per benef
6	Proportion of physical agent
7	Percent of therapeutic procedures
8	Proxy for number of new patients
9	Average age of beneficiaries
10	Average Hierarchical Condition Category (HCC) risk score of beneficiaries
11	Practice location: Small metro area
12	Practice location: Mid-sized metro area
13	Practice location: Non-metro area or missing
14	Standardized Medicare payment per beneficiary
15	Primary care physicians per 10,000 pop, county level
16	Number of PTs per 10,000 pop (2009), county level
17	Beneficiaries as a share of total pop, county level
18	Average age of beneficiaries, county level
19	Percent of female beneficiaries, county level
20	Avg HCC risk score of beneficiaries, county level
21	Percent of Medicare beneficiaries eligible for Medicaid, county level
22	Median household income, county level
23	Pct persons 65 or older in deep poverty, county level
24	Number of PTs serving Medicare per 10,000 beneficiaries, county level

Table 1: Input parameters of problem.

Methods

We examine two GA approaches to this problem: a basic GA and a self-adaptive GA (SAGA). The basic GA is a canonical GA in which all parameters must be hand tuned by a user. The SAGA reduces the amount of parameter tuning required of a user by evolving some of the parameter values along with the solution. The traditional approach to this type of problem is LR; we include it as our baseline comparison.

In addition to running the GA and SAGA on the raw data from the PUF and AHRF, we also run both algorithms on the standardized version of the same data. In problems where the ranges of the independent variables vary, standardization is used to equalize the impact of the individual variables in the ML process. The standardized value $v'_{r,i}$ for each independent variable i is calculated as follows: $v'_{r,i} = (v_{r,i} - \vec{v}_i)/\sigma_i$ where

$$\vec{v}_i = \frac{1}{40662} \sum_{r=1}^{40662} v_{r,i} \tag{3}$$

is the average of the i^{th} independent variable and σ_i is the standard deviation of $\vec{v}_i$.

Genetic algorithm

The GA is an ML algorithm that learns by simulating evolution (Holland 1975; Goldberg 1989). Algorithm 1 describes the basic steps of a GA. A GA typically begins with a randomly generated population of candidate solutions. Each solution is evaluated to determine its *fitness* or quality as a

Algorithm 1 Basic steps of a genetic algorithm.

1: Initialize population of candidate solutions randomly
2: **while** $stopping\_condition$ not met **do**
3: Evaluate the fitness of every population member
4: Apply selection method to select parents
5: Apply genetic operators to generate offspring from selected parents
6: **end while**

solution. The more fit individuals are probabilistically selected to be used to create a new population of candidate solutions. Over multiple populations or *generations*, the algorithm evolves better and better solutions. Our GA uses a floating point representation to encode solutions and uses empirically determined experimental parameter settings.

Each individual in the GA population represents one candidate solution (a vector of 25 coeffiecients). Each solution is represented as a floating point array of 50 values, $z_j : j \in \{0, 1, ..., 49\}$ where $-1.0 \leq z_j \leq 1.0$. For each independent variable i listed in Table 1, the coefficient value encoded by the GA is $c_i = z_i^x$ where

$$x = \begin{cases} 1 & \text{if} \quad -1.0 \leq z_i + 25 < -0.5 \\ 2 & \text{if} \quad -0.5 \leq z_i + 25 < 0.0 \\ 3 & \text{if} \quad 0.0 \leq z_i + 25 < 0.5 \\ 4 & \text{if} \quad 1.5 \leq z_i + 25 \leq 1.0 \end{cases} \tag{4}$$

The exponential term is included because the ranges of these variables vary by multiple magnitudes, this term aids the GA reaching all magnitudes. The fitness of an individual indicates how well its set of coefficients classify all of the data points in a training set. Algorithm 2 gives the fitness function used in our GA. Given the 25 coefficients encoded by

Algorithm 2 Fitness function.

1: $sum \leftarrow 0$
2: $r \leftarrow 0$
3: **while** $r < training\_file\_size$ **do**
4: $Q_r = \sum_{i=0}^{24} c_i v_{r,i}$
5: **if** $Q_r > 0$ **and** $B_r > B_{median}$ **then**
6: $sum \leftarrow sum + 1$
7: **else if** $Q_r \leq 0$ **and** $B_r \leq B_{median}$ **then**
8: $sum \leftarrow sum + 1$
9: **end if**
10: $r \leftarrow r + 1$
11: **end while**
12: $fitness \leftarrow sum/training\_file\_size$

one individual, calculate Q_r as shown in Equation 1 for each data point P_r. The fitness of the individual is the percentage of data points for which Equation 2 holds true (percentage of correct classifications).

Table 2 gives the GA parameter settings used in our experiments. These values were empirically shown through a series of tuning experiments to produce good performance. In step 4 of Algorithm 1, the GA uses Tournament selection with a tournament of size 10 to select a temporary population of parents of the same size as the regular population. In step 5 of Algorithm 1, the GA applies two-point crossover

Parameter	Setting
Population size	100
Stopping condition	200 generations
Selection method	Tournament, size=10
Crossover type	Two point
Crossover rate	0.9
Mutation type	Uniform random
Mutation rate	0.2

Table 2: GA parameter settings.

at a rate of 0.9 to each pair of parents in the temporary population. Two crossover locations are randomly selected and the segment within those two locations are exchanged to create two offspring. Each offspring then undergoes mutation. Mutation rate indicates the probability with which each coefficient on an individual will be mutated. Values that are selected for mutation are reset to a randomly generated floating point value between -1.0 and 1.0. The offspring population becomes the next generation and the cycle repeats until the stopping condition is met. Each GA run executes for 200 generations and returns the best solution found.

Self adaptive genetic algorithm

One of the challenges in using the GA as a learning tool is how to appropriately set the GA parameters. Tuning GA parameters can be highly impactful on result quality but is also a difficult and time consuming task (Eiben and Smit 2011). One method to alleviate this difficulty is through the use of self-adaptation in a SAGA. A SAGA is a GA in which a subset of the parameters are evolved over the course of the execution instead of being manually set at a static value. The SAGA is still fundamentally a GA and so adheres to Algorithms 1 and 2.

In our SAGA configuration (Norat and Wu 2018), four parameters are self-adaptive. These self-adaptive parameters are: crossover type, crossover rate, mutation type, and mutation rate. These parameters are encoded in each individual. Each individual thus has unique values for the self-adaptive parameters. In our SAGA, each individual has two *chromosomes*; the solution chromosome encodes the potential solution of 25 coefficients and the parameter chromosome encodes the self-adaptive parameters. Both chromosomes are modified by the genetic operators. The solution chromosome representation is identical to that of the standard GA.

Self-adaptation of the mutation rate and crossover rate allow the SAGA to vary the rate at which mutation and crossover occur throughout a run. The mutation rate is initialized as a random floating point value between 0.0 and 0.5. The crossover rate is initialized as a random floating point value between 0.5 and 1.0. Both rates can evolve within the range of 0.0 to 1.0. The encoded mutation rate directly represents the mutation rate of that particular individual. Crossover requires two or more individuals, thus the effective crossover rate of any set of potential parents is equal to the mean of the parents' encoded crossover rates.

Self-adaptation of the crossover type and mutation type allow the SAGA to vary which crossover and mutation operators are used during a run. Instead of using a single crosover

	LR	GA		SAGA		GA Standardized		SAGA Standardized	
		Best	Avg (95% CI)†	Best	Avg (95% CI)†	Best	Avg (95% CI)†	Best	Avg (95% CI)†
Training 80%	93.29	93.50	90.26 (±1.006)	**93.64**	78.20 (±3.599)	88.60	88.30 (±0.052)	89.26	89.07 (±0.034)
Validation 20%	93.32	**93.44**	90.19 (±1.035)	93.44	77.43 (±3.575)	88.86	88.61 (±0.073)	89.60	89.31 (±0.047)
Training 65%	93.07	**93.51**	89.52 (±0.828)	92.59	79.98 (±3.100)	88.87	88.20 (±0.057)	90.18	89.07 (±0.080)
Validation 35%	93.45	**93.55**	89.58 (±0.791)	92.56	79.32 (±3.050)	88.85	88.70 (±0.249)	89.65	89.38 (±0.045)
Training 50%	93.03	**93.35**	89.28 (±0.990)	92.57	79.85 (±3.190)	88.73	88.33 (±0.052)	89.27	89.07 (±0.048)
Validation 50%	93.39	**93.59**	89.58 (±1.014)	92.75	79.13 (±3.214)	88.68	88.52 (±0.068)	89.24	88.97 (±0.053)

Table 4: Percent correct classification with highest value in bold. †GA and SAGA average performance is averaged over 50 runs.

Crossover	Mutation
Two point	Uniform random
Uniform (Syswerda 1989)	Gaussian
Arithmetic (Michalewicz 1992)	Polynomial
Linear (Wright 1991)	(Deb and Deb 2014)
Simulated Binary (Deb 1995)	Swap
Blend (Kita et al (2000))	(Norat and Wu 2018)
Simplex (Tsutsui et al (1999))	

Table 3: SAGA crossover and mutation operators.

and mutation operator throughout a run as occurs with a canonical GA, SAGA selects from among a pool of potential operators each time crossover or mutation occurs. The genetic operators from which our SAGA selects are listed in Table 3. Every operator is encoded in each individual as a floating point value. These values are initialized between, and can evolve between, 0.0 and 1.0. These encoded values are treated as the fitness of each operator. Every time mutation occurs, one mutation operator is probabilistically selected using tournament selection of size 3 with a win probability of 0.9. Crossover uses this same tournament selection scheme. Again, since crossover requires two or more individuals, the effective crossover operator fitness values of any set of parents is equal to the mean of the parents' crossover operator fitness values.

Logistic regression

LR is commonly used to model the relationship between a binary response variable (e.g., yes and no) and a set of explanatory or independent variables (Hosmer and Lemeshow 1989). Suppose that Y is a binary response variable that can take on one of two possible values, denoted by 1 and 0 for example, and $\mathbf{x}$ is a set of independent variables which can be categorical or continuous. The probability that the response variable will be 1 given the independent variables, $Pr(Y = 1|\mathbf{x})$, can be modeled as

$$Pr(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \mathbf{x}\beta)}} \quad (5)$$

where β_0 is the intercept and β represents the vector of regression coefficients.

Our logistic regression analysis is performed in the SAS software, Version 9.4 for Windows (SAS Institute, Inc., Cary, North Carolina, USA). In this study, the response or outcome variable equals 1 if a PT receives an above median Medicare payment and equals 0 otherwise. The independent variables (shown in Table 1) include a combination of PT demographic, practice, and market level factors. We estimate a separate LR model for each training data set tested by the GA and SAGA, and measure the percent correct prediction rates of the model on both the training data set and corresponding validation data set.

Results

Using the 40,662 data points aggregated from the PUF and AHRF, we compare the ability of the GA, SAGA, and LR to generate a model that will correctly classify data points as being above or below the industry median. In order to evaluate the effectiveness of the learned model on unseen data, we randomly assign the data points into a training set and a validation set. We perform three experiments, each using a different training:validation ratio: 80:20, 65:35, and 50:50. Because the GA and SAGA are pseudorandom algorithms, we perform 50 runs of each and report the best solution found in all 50 runs as well as the average performance over all 50 runs and its 95% confidence interval.

Table 4 gives the percent correct classification for the three training:validation data sets. The leftmost column gives the results from LR analysis on the original data file. The next four columns give the GA and SAGA results when learning using the original raw data. The last four columns give GA and SAGA results when learning on standardized data. Each pair of rows gives results for the training and validation results for one experiment. The best result in each row is highlighted in bold text.

In all but one instance, the GA running on the raw data generates the best solution, with classification percentages ranging from 93.44% to 93.59%. On the 80:20 training set, the SAGA running on the raw training data generates the best solution, with a classification percentage of 93.64%. LR classification percentages range from 93.03% to 93.39%. There is little difference between the training and validation results for all algorithms. In several experiments, the validation set performance exceed training set performance.

Both the GA and SAGA perform significantly better on the raw data sets than on the standardized data sets. In the raw data experiments, the GA exhibits slightly better performance than SAGA in terms of the best solution found over multiple (50) runs; the GA shows a significant improvement over SAGA in terms of the average solution quality over multiple runs; and SAGA runs produce a significantly wider confidence intervals than GA runs, indicating greater

variability across runs. In the standardized data experiments, SAGA performs better than the GA but the performance difference between the GA and SAGA is much less apparent.

Although the standardized data produces worse results than the non-standardized raw data, the standardized data appears to generate more consistent results across multiple runs in both the GA and SAGA, as indicated by the narrower confidence intervals. Examination of the individual solutions generated supports this conclusion. Figure 1 shows the coefficients of the best solution found in each of 50 training runs. The top row shows GA results; the bottom row shows SAGA results. The left column shows results using raw data; the right column shows results using standardized data. All data is from training runs that use 80% of the data.

In each plot of Figure 1, each line represents one coefficient value. All values (each from a different run) for each coefficient are grouped together. We see that, in the plots involving raw data, there is often significant variation within each group of lines indicating significant variation from one run to the next as to what is a good value for most coefficients. In the plots involving standardized data, there is significantly more agreement across GA and SAGA runs as to the evolved value for each coefficient. These plots in combination with the data from Table 4 show a correlation between the stability of the GA and SAGA search process and the quality of the solutions found. When learning on standardized data, both the GA and SAGA runs generate more consistent solutions, but solutions of lesser quality. When learning on raw data, the GA and SAGA results are less predictable but, over multiple runs, may achieve significantly higher quality. Such behavior is common when the problem fitness landscape is rugged and multi-modal.

Conclusions and Future Work

We examine the ability of a GA to learn a predictive model that can estimate the likelihood that a PT will receive annual Medicare payments above or below the median based on the PT's practice parameters. We compare the performance of a canonical GA and a self adaptive GA with the performance of traditional LR. Both GA approaches are tested on the raw data and on the same data after it has been standardized.

Results indicate that both GA approaches are competitive with LR when learning on the raw data. The canonical GA consistently outperforms LR while SAGA performance varies slightly from experiment to experiment. When learning on the standardized data, both the GA and SAGA perform worse than LR.

Because the GA and SAGA are pseudorandom algorithms, we also examine the performance of these algorithms over multiple runs. The standardized data results in more consistent but slightly worse performance in both algorithms. The raw data results in less consistent but more competitive results in both algorithms. The SAGA exhibits more variation of results across multiple runs when compared to the canonical GA. Despite this variation, the best SAGA runs are still competitive while requiring less manual tuning than a canonical GA.

References

Brennan, N.; Conway, P. H.; and Tavenner, M. 2014. The Medicare physician-data release – context and rationale. *New England Journal of Medicine* 371(2):99–101.

Chaurasia, V., and Pal, S. 2014. Data mining techniques: to predict and resolve breast cancer survivability. *Int'l Journal of Computer Science and Mobile Computing* 3:10–22.

Ciresan, D.; Meier, U.; Masci, J.; and Schmidhuber, J. 2012. Multi-column deep neural network for traffic sign classification. *Neural Networks* 32:333–338.

de Vasconcelos, M. J. P.; Silva, S.; Tomé, M.; Alvim, M.; and Pereira, J. M. C. 2001. Spatial prediction of fire ignition probabilities: comparing logistic regression and neural networks. *Photogrammetric Engineering & Remote Sensing* 67(1):73–81.

Deb, K., and Deb, D. 2014. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing* 4(1):1–28.

Deb, K. 1995. Simulated binary crossover for continuous search space. *Complex Systems* 9:115–148.

Dehuri, S.; Patnaik, S.; Ghosh, A.; and Mall, R. 2008. Application of elitist multi-objective genetic algorithm for classification rule generation. *Applied Soft Comp.* 8(1):477–487.

Eiben, A. E., and Smit, S. K. 2011. Evolutionary algorithm parameters and methods to tune them. In *Autonomous Search*. Springer. 15–36.

Fernández, A.; García, S.; Luengo, J.; Bernadó-Mansilla, E.; and Herrera, F. 2010. Genetics-based machine learning for rule induction: state of the art, taxonomy, and comparative study. *IEEE Trans. Evol. Computation* 14(6):913–941.

Fidelis, M. V.; Lopes, H. S.; and Freitas, A. A. 2000. Discovering comprehensible classification rules with a genetic algorithm. In *Proc. Congress on Evol. Comp.*, 805–810.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley.

Guvenir, H. A., and Erel, E. 1998. Multicriteria inventory classification using a genetic algorithm. *European Journal of Operational Research* 105(1):29–37.

Höglund, H. 2017. Tax payment default prediction using genetic algorithm-based variable selection. *Expert Systems with Applications* 88:368–375.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Hosmer, D. W., and Lemeshow, S. 1989. *Applied Logistsic Regression*. Wiley.

Jefferson, M. F.; Pendleton, N.; Lucas, S. B.; and Horan, M. A. 1997. Comparison of a genetic algorithm neural network with logistic regression for predicting outcome after surgery for patients with nonsmall cell lung carcinoma. *Cancer: Interdisciplinary Int'l Journal of the American Cancer Society* 79(7):1338–1342.

Kita, H.; Ono, I.; and Kobayashi, S. 2000. Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms. *Transactions of the Society of Instrument and Control Engineers* 36(10):875–883.

Kovačič, M., and Dolenc, F. 2016. Prediction of the natural gas consumption in chemical processing facilities with genetic programming. *Genetic Programming & Evolvable Machines* 17(3):231–249.

Liu, X.; Oetjen, R. M.; Hanney, W. J.; Rovito, M. J.; Masaracchio, M.; Peterson, R. L.; and Dottore, K. 2018. Characteristics of

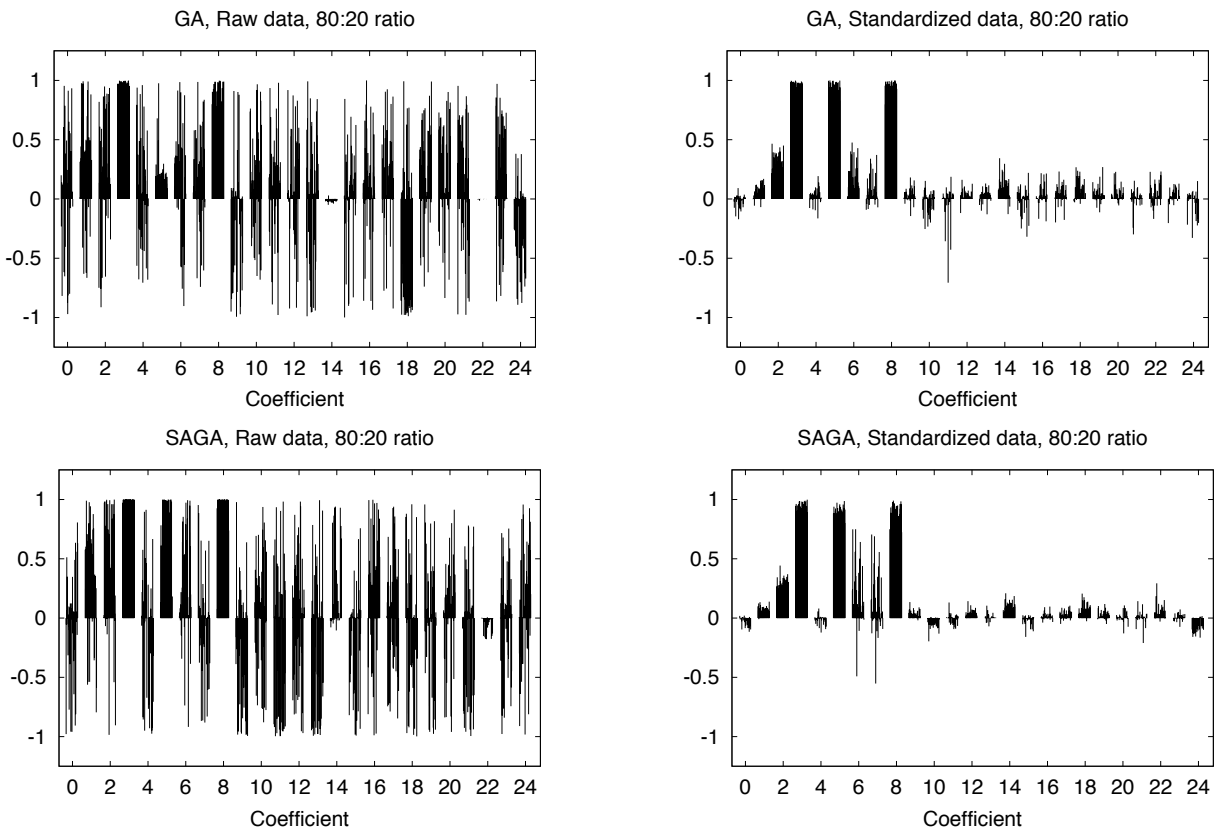

Figure 1: Coefficients of the best solution from each of 50 GA (top row) and SAGA (bottom row) training runs using raw (left column) and standardized (right column) data on the 80:20 training:validation sets.

physical therapists serving Medicare fee-for-service beneficiaries. Unpublished manuscript.

Michalewicz, Z. 1992. *Genetic Algorithms Data Structures = Evolution Programs*. Springer.

Min, S.-H.; Lee, J.; and Han, I. 2006. Hybrid genetic algorithms and support vector machines for bankruptcy prediction. *Expert Systems with Applications* 31(3):652–660.

The Centers for Medicare and Medicaid Services, Office of Enterprise Data & Analytics. Medicare Fee-For-Service Provider Utilization & Payment Data Physician & Other Supplier Public Use File: A Methodological Overview 2016.

Ning, F.; Delhomme, D.; LeCun, Y.; Piano, F.; Bottou, L.; and Barbano, P. E. 2005. Toward automatic phenotyping of developing embryos from videos. *IEEE Trans. Image Processing* 14(9):1360–1371.

Norat, R., and Wu, A. S. 2018. Improving genetic algorithm usability through self adaptation. Technical Report CS-TR-CS-18-02, University of Central Florida.

Paul, D.; Su, R.; Romain, M.; Sébastien, V.; Pierre, V.; and Isabelle, G. 2017. Feature selection for outcome prediction in oesophageal cancer using genetic algorithm and random forest classifier. *Computerized Medical Imaging & Graphics* 60:42–49.

Syswerda, G. 1989. Uniform crossover in genetic algorithms. In *Proc. 3rd Int'l Conference on Genetic Algorithms*.

Thornblade, L. W.; Flum, D. R.; and Flaxman, A. D. 2018. Predicting future elective colon resection for diverticulitis using patterns of health care utilization. *Journal for Electronic Health Data and Methods* 6(1):1–8.

Tsutsui, S.; Yamamura, M.; and Higuchi, T. 1999. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In *Proc. Genetic and Evolutionary Computation Conf.*, volume 1, 657–664.

Vandewater, L.; Brusic, V.; Wilson, W.; Macaulay, L.; and Zhang, P. 2015. An adaptive genetic algorithm for selection of blood-based biomarkers for prediction of Alzheimer's disease progression. *BMC Bioinformatics* 16(18):S1.

Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; and Lang, K. J. 1989. Phoneme recognition using time-delay neural networks. *IEEE Trans. Acoustics, Speech, and Signal Processing* 37(3).

Wright, A. H. 1991. Genetic algorithms for real parameter optimization. In *Foundations of Genetic Algorithms*, volume 1. Elsevier. 205–218.

Zamuda, A.; Zarges, C.; Stiglic, G.; and Hrovat, G. 2017. Stability selection using a genetic algorithm and logistic linear regression on healthcare records. In *Proc. Genetic and Evolutionary Computation Conf.*, 143–144.

Zhang, Z.; Trevino, V.; Hoseini, S. S.; Belciug, S.; Boopathi, A. M.; Zhang, P.; Gorunescu, F.; Subha, V.; and Dai, S. 2018. Variable selection in logistic regression model with genetic algorithm. *Annals of Translational Medicine* 6(3).