Proposal

Domain Background

For computer scientist, physicists and mathematics Integer Sequences are a very important part of their daily work but historically, sequences have been an integral part of the scientific endeavor, as human beings have been always fascinated by patterns, and our ability to detect such patterns in sequences of numbers seems to be unlimited(1).

Even before Pythagoras mathematicians have been developing formulas and theorems that provide the theoretical basis for understand numerical patterns, modern computational technologies have developed different algebraic systems that perform symbolic and high-precision computations, but even though they're very advanced in terms both of computing power and graphical capabilities they are still no more than aides for the scientists(1).

The On-Line Encyclopedia of Integer Sequences (OEIS), also known simply as Sloane's, and the Journal of Integer Sequences are the main investigative effort in the area of Integer Sequences, both founded by Neil Sloane, are now collective efforts with the main objective of record new sequences as they are discovered and provide the theoretical support for those sequences. So, Machine Learning seems like the next logical step, where a deep level of pattern recognition can be used to find patterns without the inflexibility of mathematical rigor(2).

Problem Statement

For many years' scientists have been trying to discover new patterns in number sequences and developing formulas and theorems that describe the fundamentals of such patterns. Those sequences are not mere mathematical curiosities but powerful tools for many fields like number theory, combinatorics, and discrete mathematics(1).

Currently there are many software applications that provide computational algebraic analysis and graphical capabilities that help scientist if their quests of finding and defining new sequences, at first glance it may seem like this symbiotic relation between man and machine will continue to be the main source of new sequences, unless a deeper level of analysis of how patterns arise can be achieved, so new sequences can be discovered based on a fundamental understanding of their nature instead of a one-to-one basis(1).

Datasets and Inputs

For this problem the main datasets come from On-Line Encyclopedia of Integer Sequences (OEIS) since it's recognized as the biggest collection of integer sequences and it's a concerted effort that have been being developed since 1964, and there doesn't seem to be any other datasets that compares it to it in its completeness.

For this specific solution we have used a subset of the whole OEIS dataset obtained from the Kaggle platform, it contains the majority of the integer sequences from the OEIS. It is split into a

training set, where the full sequences are given, and a test set, where the last number from the sequence has been removed. The datasets were downloaded from the following link: https://www.kaggle.com/c/integer-sequence-learning/data.

Solution Statement

The solution for this problem is utilize machine learning techniques to predict the last number of an integer sequence. I created classifier to predict if the final element is a specific number, for this I defined features like, does the sequence has negatives numbers; the maximum value of a sequence; are there zeroes in the sequence; does the final element in the sequence divisibility by two matches that of most of the elements in the sequence. Finally, a logistic regression is applied to those features and the Area Under the Curve (AUC) is used to make a prediction.

Benchmark Model

Since the domain of the problem and the available datasets are limited to known sequences of integers it's hard to define a benchmark model besides the application of another machine-learning algorithm different for the one I used –i.e. logistic regression- a good example would be random decision forests, but due the uniqueness of the solution, this approach would be a poor benchmark model.

Therefore, one benchmarking could be the utilization of Computer Algebra Systems* (CAS) that can "guess" or try to guess the generating function based on a subset of the sequence, and then check the prediction against the function the software deduced from the sequence: if there's an x such that $f(x) = a$, and there's a x_i such as $f(x_i) = b$ where a is the predicted final element of the sequence and b is the number immediately previous to a. The problem with this approach is that there are many sequences that may not be generated by a function that can be found by any software.

** Maple's gfun package, Wolfram Mathematica and FriCAS among many others.*

Evaluation Metrics

I propose the use of a classification metric as an evaluation metric for this problem, more specifically Area Under Receiving Operating Characteristic (ROC) Curve (or AUC for short) which is a performance metric for binary classification patters. The AUC represents the model's ability to predict the last number of a sequence. An area of 1.0 represents a model that made all predictions perfectly, and an area of 0.5 represents a model as good as random(3). Since we are using normalized units the area under the curve is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one (assuming 'positive' ranks higher than 'negative'). This can be seen as follows: the area under the curve is given by (the integral boundaries are reversed as large T has a lower value on the x-axis)

$$A = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

where X_1 is the score for a positive instance and X_0 is the score for a negative instance(4). The calculation for our problem was done using this trapezoidal rule(5).

Project Design

The approach to solve this problem must be the implementation of a supervised learning regression algorithm for different type of variables, including binary, ordinal and nominal with continuous and categorical predictors, the idea is to estimate parameters in the model so that the fit in the model is optimized what is an optimal task for a Generalized Linear Model, more specifically a Logistic Regression(6).

The original dataset as it was available in the OEIS website required the removal of the last integer of the sequence and it wasn't in an Comma Separated Values (.csv) format, for that reason I decided to use a shorter version available in the Kaggle website, it contains the majority of the integer sequences from the OEIS and it is split into a test and training set, while keeping duplicated sequences. It required no additional analysis.

The data was imported from the csv file using Python *csv* libraries and converted to *pandas* dataframes so it could be processed and used.

The main strategy is to predict if the final element be the most frequent final element of each sequence in the training set, so I built a classifier and engineered the following features:

- Does the sequence has negatives numbers (*negatives*). This is a Boolean feature that returns true if there are digits with negative values in the sequence and false if there isn't.
- The maximum value of a sequence (*max*). This is a nominal feature that returns the digit with the digit with the maximum value in the sequence.
- Are there any zeroes in the sequence (*zeroes*). This is a Boolean feature that returns true if there are '0' digits in the sequence and false if there isn't.
- Does the final element in the sequence divisibility by two matches that of most of the elements in the sequence (evenOddMatch). This is a Boolean feature that returns true if the final element divisibility by two matches that of most of the digits in the sequence and false if it isn't.
- How frequent is the final element (previousCount). This is a nominal feature that returns the number of times the final digit appears in the sequence.
- Most Frequent Final Element (featureClass). This is a nominal feature that returns the most frequent final element.

All the features were plotted in order to visualize how predictive they were, using *matplotlib.pyplot* libraries, then the model was fitted using a logistic regression using the *statesmodel* ***glm*** api method, for the fitting the formula has the most Frequent Final Element (featureClass) as the dependent variable and a binomial family, that specifies a binomial error distribution and a logistical regression function.

Finally a prediction was calculated using and Area Under ROC Curve score using sklearn *metrics* library, with an accuracy score of ~0.95.

(1) Staff. "AT&T Researchers — Inventing the Science Behind the Service." *AT&T Labs Research - The Achievement of The Online Encyclopedia of Integer Sequences*. Research ATT, 6 Mar. 2012. Web. 1 Dec. 2016. <http://www.research.att.com/articles/featured_stories/2012_03/201203_OEIS.html?fbid=eb3d4qlYcuI>.
(2) "On-Line Encyclopedia of Integer Sequences." *Wikipedia*. Wikimedia Foundation, 2 Mar. 2004. Web. 12 Dec. 2016. <https://en.wikipedia.org/wiki/On-Line_Encyclopedia_of_Integer_Sequences>.
(3) Hallinan, Jennifer. "Assessing and Comparing Classifier Performance with ROC Curves - Machine Learning Mastery." *Machine Learning Mastery*. Machine Learning Process, 04 Sept. 2016. Web. 12 Dec. 2016. <http://machinelearningmastery.com/assessing-comparing-classifier-performance-roc-curves-2/>.
(4) "Receiver Operating Characteristic." *Wikipedia*. Wikimedia Foundation, 15 Sept. 2003. Web. 12 Dec. 2016. <https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve>.
(5) "Sklearn.metrics.auc." *Sklearn.metrics.auc — Scikit-learn 0.18.1 Documentation*. Scikit Learn, n.d. Web. 12 Dec. 2016. <http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html>.
(6) P. McCullagh and J. A. Nelder (1992). Generalized Linear Models. Chapman & Hall.