Building and Evaluating Interpretable Models using Symbolic Regression and Generalized Additive Models

Khaled Sharif [*1]

Abstract

In this paper we investigate new methods to build and evaluate interpretable predictive models for time series data using symbolic regression and generalized additive models. We propose a novel framework to iteratively build a model while maintaining model interprebility as accuracy and complexity increase. We also propose multiple methods that ease interpretation of the built model, the model building process, and the model output. The proposed methods study the contributions of the model constituents, partial derivatives, and behavior in the frequency domain. Finally, we empirically demonstrate the framework methods by modeling weather phenomena using a weather station observational dataset, and show how the resulting model finds underlying meteorological principles automatically.

1. Introduction

The interprebility of a model should stem from how simplistic a model is, but this can cause problems with weakness of an overly simplistic model. Usually, a simple model is an interpretable model but also a weak one, and this is due to the difficulty of interpreting and understanding the large number of constituents that make up an accurate but complex model. Researchers instead must look for simplistic models (such as decision trees) to get an understanding of how the machine learns from the data it is given. Simplistic models, such as the decision tree, are too inaccurate on their own and are usually grouped together in an ensemble to produce accurate results; moreover, accurate decision trees usually span to very large depths and breadths, creating very complex and difficult to understand models (when attempting to interpret them) that are also hard to model mathematically.

Classification problems are generally easier to interpret than regression problems, and this is because they have a finite discrete amount of outcomes. Moreover, a classification model can usually be traced from start to finish (from input to output) to try and understand how this particular outcome came about. Regression problems may have an infinite amount of outcomes, and it is not always so clear as to how a particular input parameter is transforming the model outcome. To interpret a model, one may ask: what is the contribution of a particular parameter to the model outcome on average? Harder questions may follow the form of: if I were to change a particular parameter by a certain amount, how much of that change would then be reflected onto the outcome?

In this paper, we propose a framework that iteratively produces a generalized additive model, in a search form that mimics symbolic regression, while maintaining a reasonable balance between model interprebility and accuracy. We also propose methods to ease the interpretation of the models generated by the framework. Finally, we present an empirical study of the framework when applied to a weather station observational dataset.

2. Related Work

The idea of Generalized Additive Models (GAM) first stemmed from the KolmogorovArnold representation theorem (Braun & Griebel, 2009), which states that every multivariable continuous function can be represented as a superposition of continuous functions of two variables (Hastie & Tibshirani, 1990). Various work has been produced since then that uses GAM, with all work citing the main advantage of using GAM was producing accurate models of the underlying processes (Guisan et al., 2002; Dominici et al., 2002; Yee & Mitchell, 1991).

There has been some amount of work done specifically on using Symbolic Regression (SR) to produce interpretable models. (Giustolisi & Savic, 2006) described a hybrid regression method that combined the best features of conventional numerical regression techniques with the genetic

[*]Equal contribution [1]ArabiaWeather Inc., Amman, Jordan. Correspondence to: Khaled Sharif <khaled.sharif@arabiaweather.com>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

programming SR technique; in it they described three different types of models: white, black, and gray box models. White-box models are based on first principles (such as physical laws), known variables and known parameters, whereas black-box models are systems for which there is no prior information available. The method they propose tries to form a gray-box model, which is a conceptual model whose mathematical structure can be derived through conceptualization of physical phenomena or simplification of complex existing models. The authors also list some challenges of traditional SR methods: these include the large number of evolutionary parameters to tune, the complexity of the generated symbolic models, and the difficulty of interpreting the tree structure that the traditional SR method produces.

(Affenzeller et al., 2014) discussed the existing scrutiny for traditional models and how SR attempts to solve this through complex non-linear white-box models (as discussed earlier) in an attempt to gain an understanding of the underlying processes. They again raise similar problems with genetic programming based SR, citing code bloat, occurrence of non-functional code segments, and genetic drifts in the search process that lead to vastly different models when the search is rerun.

3. Building an Interpretable Regressive Model

The proposed method to build a regressive predictive model is to start building it with simplistic mathematical blocks, and through symbolic regression techniques increase the complexity of those blocks until the final model as a whole has both sufficient accuracy and is simplistic enough to be easily interpreted. This method allows for reasonable interpretation of the model because it is displayed to the interpreter as a series of successive additions to the model by the symbolic regression framework.

3.1. Generalized Additive Model Formation

The generalized additive model proposed by the framework is composed of multiple separate constituents, of which the basic form is represented by the equation below.

$$x_0 f(ax_1 + x_2) + x_3$$

This form was chosen because of its simplicity (the function in question can be a sine, cosine, exponential, etc.) and is thought to be a "building block" from which highly complex models can be built from. Assuming the function in the constituent is easily differentiable, finding the coefficients in the equation that best fit the model to the output is done by using the Gradient Decent (GD) method (Mandic, 2004). Put simply, the GD method is the process of updating a set of parameters (in our case, all constituent coefficients) in an iterative manner to minimize an error function (in our case, the difference between the sum of the constituents and the actual data set). To demonstrate this method, consider a model with only one constituent. In order to find the coefficients of the constituent, we must solve the following equation.

$$x_0 f(ax_1 + x_2) + x_3 = b$$

To start, we rearrange the equation to form the function $G(X)$, which is the square difference between the sum of constituents and the desired output.

$$G(X) = (-b + x_0 f(ax_1 + x_2) + x_3)^2$$

In which X is a vector representing all our coefficients, from x_0 to x_n. Throughout our search for the function global minimum, we will create successive iterations of X, and to simplify this, we will denote each iteration of X as X_i. X is therefore defined as follows.

$$X_i = \begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_n \end{bmatrix}$$

$$X_0 = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \end{bmatrix}$$

Our objective function, denoted here as $F(X)$, is related to $G(X)$ (as defined in the GD method) in the following way.

$$F(X) = 0.5\,(-b + x_0 f(ax_1 + x_2) + x_3)^4$$

We will need to define a Jacobian matrix for our function $G(X)$. This is because the GD method Gradient descent is based on the mathematical fact that if the multi-variable function $F(X)$ is defined and differentiable in a neighborhood of a point A, then $F(X)$ decreases fastest if one goes from A in the direction of the negative gradient of F at A. The Jacobian for $G(X)$ is therefore defined as follows.

$$J = \begin{bmatrix} J_0 & J_1 & J_2 & J_3 \end{bmatrix}$$

$$J_0 = 2\,(-b + x_0 f(ax_1 + x_2) + x_3)\, f(ax_1 + x_2)$$

$$J_1 = 2ax_0\,(-b + x_0 f(ax_1 + x_2) + x_3)\ \frac{d}{d\xi_1} f(\xi_1)\Big|_{\xi_1 = ax_1 + x_2}$$

$$J_2 = 2x_0\,(-b + x_0 f(ax_1 + x_2) + x_3)\ \frac{d}{d\xi_1} f(\xi_1)\Big|_{\xi_1 = ax_1 + x_2}$$

$$J_3 = -2b + 2x_0 f(ax_1 + x_2) + 2x_3$$

The gradient of the objective function $F(X)$ can now be produced from the Jacobian of $G(X)$, denoted as J, in the following way.

$$\bigtriangledown F(X_i) = J(X_i)^T G(X_i)$$

Finally, through our derivation of the objective function, we reach our iterative equation for the coefficients. The λ_0 in the equation below is a constant, usually referred to as the learning rate, and is set such that $F(X_1) <= F(X_0)$. Therefore, to iteratively find the best coefficients for our single constituent, we use the following iterative function.

$$X_{i+1} = X_i - \lambda_0 \bigtriangledown F(X_i)$$

This process is valid for a model with only one constituent. Usually, accurate models have a large number of constituents that are all summed together to produce the output. We will demonstrate empirically that increasing the number of constituents in a model produced through our proposed framework increases correlation of the predictions to the actual data.

In the framework implementation, we make use of an automatic symbolic differentiation library, Tensorflow, to automatically derive and iterate over a large and complex equation formed of an arbitrary number of constituents using the GD (Abadi et al., 2015). In practice, there are many downsides to the GD method, and it is possible to use the framework with more advanced optimization algorithms, such as the Adam optimization algorithm (Kingma & Ba, 2014). The use of Tensorflow not only allows the framework to generate arbitrarily large equations using a mix of simplistic functions and compositive functions too; it also allows the framework to run the optimization algorithm on the GPU, yielding extraordinary speedup in time needed for the algorithm to converge to a minimum. The use of this library and hardware acceleration is thought to make this method of symbolic regression the most performant compared to other methods. The framework code is open sourced on Github and can be viewed by clicking here.

3.2. Iteratively Evolving the Model

We presented in the previous section a way of fitting a constituent (or multiple constituents) with known function type using the GD method (or a similar optimization algorithm). However, during model creation we may not know the best combination of functions (or composites of multiple functions), and it is very computer intensive to exhaustively search through all combinations of constituents.

The proposed solution to this is to iteratively evolve a model; this is similar to a genetic algorithm search of symbolic regression methods. The process first randomly generates models with only one constituent composing each model; this is because in each generation of the search, we will add one more constituent to the model. The models are then fit using the GD method, then they are evaluated for accuracy and sorted by their evaluation. We then filter the models so that we are left only with a certain percentage of models that make up the best models of the generation.

To create a new generation in a genetic algorithm search, we need to do two things: we first must cross-over the highest ranking models that passed through our filter, then we mutate these models by randomly adding a new constituent. The cross-over process is done by forming a new model from two highly ranked models; in the process, the new model is generated by adding the constituents of each of the models together, and all products of a cross-over operation are then mutated. This leaves us with more complex models than the previous generation in our search, and it is assumed that these new models have greater accuracy than the models before them.

The use of this iterative evolutionary search for increasingly complex and accurate models is not only for finding accurate models efficiently; this process also greatly helps with the interpretation of the final model. This is because it is very valuable to the interpreter to see a flow chart that shows how the model being interpreted has evolved from a single constituent to a complex multi-constituent model. During each step of the evolution the increase in accuracy is shown along with the additional constituents that were added to the model in that generation. The interpreter is therefore able to see the added value of each part of the model and understand from this "flow chart" the underlying properties of the model.

3.3. Empirical Comparison with the Decision Tree method

The result of this iterative search is an interpretable and easily understood predictive model that is close to what was described as a gray-box model in the literature review. We use a decision tree regressor to serve as a comparison for interpretable regressive models. The test involves modeling the surface temperature T observed from a weather station, using only the time of day t and the relative humidity H recorded by the station. The model generated from the framework is detailed below.

$$T(t,H) = \frac{53.0}{\left((0.012rh+2.0)^2\right)^{0.68}} + 3.4\sin\left(\frac{716\pi}{1947}time - \frac{2359}{2493}\right) + 3.4\sin\left(\frac{5238\pi}{2909}time - \frac{5984}{7753}\right) + 8.7$$

Through empirical testing, the decision tree (even at a depth of 50) does not reach in terms of accuracy what a model produced from our framework reaches at a magnitude smaller complexity. On our cross-validated testing dataset (10-fold), the decision tree achieves around 72% correlation at a tree depth of 50, whereas the frameworks produced equation achieves around 80% correlation using only 4 constituents.

4. Interpretable Model Evaluation

4.1. Constituent Contribution Factor

Given a predictive regressive model that can be divided into multiple individual independent constituents, we can find what we coin as the "contribution factor" of each independent constituent. In our context, the contribution we are referring to is mathematical contribution of the constituent to the overall model output.

To find the contribution of each constituent, we will need to create a statistical confidence interval for the output of that constituent as a percentage of the average output, given a well defined reasonable range of inputs that influence to the constituent in consideration.

There are a number of benefits to generating constituent confidence intervals of a model that relate directly to the interprebility of that model. Because each confidence interval is represented as a percentage of the average output, this gives the interpreter an idea of how important a certain feature is within the model and how it governs the output; similarly, the differing range of intervals given their corresponding confidence levels show how variant the contribution of the constituent is, and may give the interpreter some idea of how volatile this constituent is.

In Figure 1, through empirical analysis and by using our weather station observational dataset, we outline the relationship between average constituent contribution (as a percentage of the output) and the accuracy gained from the model. The graph shows a negative correlation between average constituent contribution and model accuracy, and signifies that an accurate model gains strength through a small average constituent contribution. In Figure 2, through the same empirical analysis we outline the relationship between the number of constituents of the model generated by our framework, and the accuracy of the model. The graph shows a positive correlation between the number of constituents and the model accuracy; this signifies that a strong model contains a large number of constituents.

4.2. Model Partial Derivation

Another test for interprebility of a model is the correlation of the input/output partial derivatives for the model and data set. This can be explained simply as such: given a model with a clear relationship between the partial derivatives (which can be interpreted as the change in output given a change in a particular input of interest) of the model and for the actual data, it is easier for the interpreter to gain confidence in the output of this model, and the model is therefore more interpretable. This causation stems from the fact that partial derivatives for the actual data set may already be known, or at least the interpreter may have some idea as to what the derivative shape may look like for a particular subset of the input parameters (this may be thought of as derivatives that stem from traditional scientific fields, of which the relationships between variables has already been understood to some degree).

The correlation between the model derivative of a particular constituent and the corresponding derivative generated from the data set is best described by a non-linear correlation coefficient or indicator, such as the Coefficient of Determination. The need for a non-linear coefficient of correlation is mainly due to the non-linear nature of partial derivatives and the need to find a correlation indicator that shows the similarity between shapes.

To a model interpreter, there is large value in seeing high partial derivative correlations for the majority of constituents of a model; it shows to the interpreter that the model inhibits constituents whose derivatives are similarly shaped to the real derivatives approximated from real data. The models sensitivity to change in input, and most importantly the correlation between model change and actual change, can be therefore interpreted with confidence. Consequently, a model that is easier to partially differentiate is a more interpretable model than a model that is harder to partially differentiate.

To demonstrate this method we use the framework to model surface temperature, similar to what we did in the previous section, and we will assess the partial derivative of temperature by humidity. We want to ask: how much does the surface temperature change due to a change in relative humidity by one unit? Because the framework produces a computational symbolic equation, the partial derivation task is trivial, and the resulting equation is shown below.

$$\frac{dT}{dH} = -\frac{0.77\left((0.012H+0.34)^{5.6}\right)}{(0.012H+0.34)^{2}}\left(0.00083H+0.024\right) + 0.068\sin\left(2\pi t-\frac{37712}{8993}\right)-0.023$$

Through interpretation of the partial derivative above, we observe a non-linear negative correlation between the surface temperature and relative humidity; this is to say that an increase in relative humidity yields a non-linear decrease in surface temperature. While this is a gross oversimplification, this is true (on average) from a meteorological point of view. The ability of the framework to find underlying physical relationships makes it, in some sense, more interpretable.

4.3. Frequency Domain Analysis

When modeling a time series, especially one that is periodic or exhibits periodic components (or constituents), an interpreter of the model may want to interpret these periodic components in a proper manner, and for that the interpreter may resort to the frequency domain. It is worth noting that while this method exists for time series models, it is generalized easily to any periodic model, so long as it can be analyzed in the frequency domain. Put simply, the frequency domain here refers to the analysis of time series predictive models with respect to frequency, rather than time. To an interpreter of a model, it is useful to both observe the frequencies present in the predictive model, and to compare those frequencies with the corresponding frequencies present in the actual data, or from a prior knowledge of frequencies that should be inherent in the model.

Perhaps out of these three methods, this method is the hardest to implement on all regressive predictive models due to the fact that not all models are easily transformed into the frequency domain in a process known as the Fourier Transform. The Fourier Transform (FT) decomposes a function of time (a signal) into the frequencies that make it up.

When comparing the frequencies present in the predictive regressive model, and in the actual data, it is made clear to the interpreter how closely the model is predicting the periodic components of the time-series; we can therefore say that the frequency domain check for a predictive model is a way of easily interpreting the periodicity of a model. Moreover, by comparing the frequencies present in the model and in the actual data, we are able to interpret the similarity of a model to the actual data in domains different from the traditional time domain, and this increases confidence in the interpretation of a model.

Similar to the test we did previously, we will test this check on our model generated for surface temperature. For simplicity purposes, we only observe the frequency present in a sine wave that relates the hour of day to the temperature.

$$T(t) = 2.762 \sin\left(2.02813\pi t - 1.43\right)$$

Through a FT of the sine wave above, we can determine that the frequency of the sine wave roughly corresponds to the equivalent of a 24 hour period. Our interpretation of this coincides with our previous meteorological knowledge of the 24 hour period of the solar day, which is the main reason behind the daily temperature fluctuation on Earth.

5. Conclusions

In this paper, we presented a framework and several accompanying methods that together allow the creation of complex non-linear interpretable models that are more accurate and more easily understood than previously available methods that utilize SR and GAM. Using the framework, we were able to produce an accurate gray-box model of several weather phenomena such as surface temperature and relative humidity. The combination of accuracy and interprebility in the produced model was hard to achieve using any other framework. Finally, we were able to utilize the three methods outlined in this paper (constituent contribution, partial derivation, frequency domain analysis) to interpret the underlying model and its relation to our previous understanding of meteorology and the physical laws that govern weather phenomena.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S., Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Goodfellow, Ian, Harp, Andrew, Irving, Geoffrey, Isard, Michael, Jia, Yangqing, Jozefowicz, Rafal, Kaiser, Lukasz, Kudlur, Manjunath, Levenberg, Josh, Mané, Dan, Monga, Rajat, Moore, Sherry, Murray, Derek, Olah, Chris, Schuster, Mike, Shlens, Jonathon, Steiner, Benoit, Sutskever, Ilya, Talwar, Kunal, Tucker, Paul, Vanhoucke, Vincent, Vasudevan, Vijay, Viégas, Fernanda, Vinyals, Oriol, Warden, Pete, Wattenberg, Martin, Wicke, Martin, Yu, Yuan, and Zheng, Xiaoqiang. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Affenzeller, Michael, Winkler, Stephan M, Kronberger, Gabriel, Kommenda, Michael, Burlacu, Bogdan, and Wagner, Stefan. Gaining deeper insights in symbolic regression. In *Genetic Programming Theory and Practice XI*, pp. 175–190. Springer, 2014.

Braun, Jürgen and Griebel, Michael. On a constructive proof of kolmogorovs superposition theorem. *Constructive approximation*, 30(3):653, 2009.

Dominici, Francesca, McDermott, Aidan, Zeger, Scott L, and Samet, Jonathan M. On the use of generalized additive models in time-series studies of air pollution and health. *American journal of epidemiology*, 156(3):193–203, 2002.

Giustolisi, Orazio and Savic, Dragan A. A symbolic data-driven technique based on evolutionary polynomial regression. *Journal of Hydroinformatics*, 8(3):207–222, 2006.

Guisan, Antoine, Edwards, Thomas C, and Hastie, Trevor. Generalized linear and generalized additive models in studies of species distributions: setting the scene. *Ecological modelling*, 157(2):89–100, 2002.

Hastie, Trevor J and Tibshirani, Robert J. *Generalized additive models*, volume 43. CRC press, 1990.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Mandic, Danilo P. A generalized normalized gradient descent algorithm. *IEEE signal processing letters*, 11(2): 115–118, 2004.

Yee, Thomas W and Mitchell, Neil D. Generalized additive models in plant ecology. *Journal of vegetation science*, 2(5):587–602, 1991.

Figure 1. The relationship between average constituent contribution (as a percentage of the output) and the accuracy gained from the model. The graph shows a negative correlation between average constituent contribution and model accuracy.

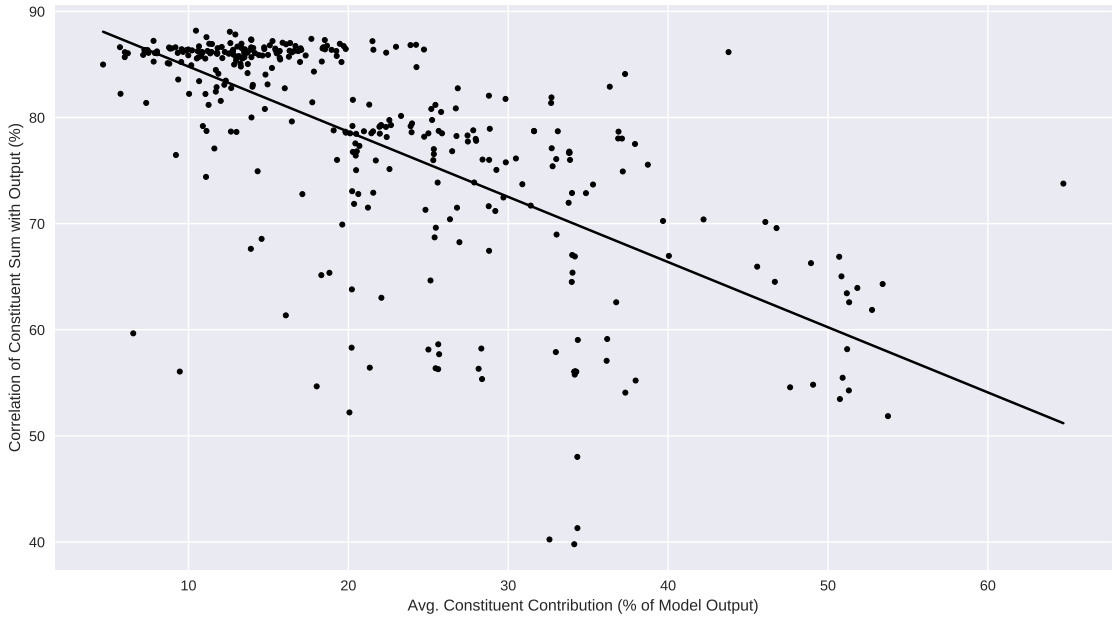

Figure 2. The relationship between the number of constituents of the model generated by our framework, and the accuracy of the model. The graph shows a positive correlation between the number of constituents and the model accuracy.