Hierachical Bayesian Symbolic Regression

Joshua Hewson[1] , **Sida Li**[2] , **Other Authors**
[1]Carney Brain Institute, Brown University
[2]Statistics Department, The University of Chicago
joshua_hewson@brown.edu, listar2000@uchicago.edu

Abstract

These are some initial write-ups for a hierarchical bayesian symbolic regression algorithm. Please ignore the rest of this abstract.

1 Notations

1.1 Dataset

Symbolic Regression (SR) is a task that aims to find the best closed-form mathematical expression that fits in a given dataset $D = \{\mathbf{X}, \mathbf{y}\}$, where the covariates are packed into $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n]^T$ as a $n \times k$ matrix with the i^{th} row $\mathbf{x}_i$ containing the k features of the i^{th} datapoint and the response $\mathbf{y} = [y_1, ..., y_n]^T \in \mathbb{R}^n$. We further denote $x_j^{(i)}$ as the j^{th} covariate of the i^{th} data-point.

1.2 Token Library

Since SR is concerned with finding closed-form, analytical solutions, there need to be some limitations on the search space of valid expressions. A token library, denoted by $\mathbb{L}$, is a set that includes all the "tokens", i.e. features and operators, that are allowed to show up in an expression. By default, a library $\mathbb{L}$ should contain features from the dataset $\{x_1^{(i)}, ..., x_k^{(i)}\}$ and a special token c for constants (note that there might be multiple constants in an expression with the same unique token in $\mathbb{L}$). The default library $\mathbb{L}_0$ we use in the following sections is specified in the Appendix.

1.3 Expression Tree

Similar to other SR methods, Hierarchical Bayesian Symbolic Regression (HBSR) represents an expression as a binary tree with intermediate nodes bearing tokens of unary operators (e.g. $\sin(\cdot), (\cdot)^2$) or binary operators (e.g.$\times, +$), while leaf (terminal) nodes carry input features (covariates) or constants. To better capture individual differences among data points, HBSR does not use a single expression tree (and its corresponding expression) to fit the entire dataset.

Instead, HBSR assumes a **template expression tree** $T = \{S, \Theta, \Omega\}$ at the top-level. S denotes the **tree structure**, which essentially describes the shape of the binary tree with empty nodes. Θ is a collection of **global parameters** that would be shared across all data points. Each $\theta_i \in \Theta$ can either be a constant value or an operator – the key is that elements in Θ are concrete tokens that would show up in the final expressions built from the template (see below). On the other hand, Ω encapsulates **priors** for **local constants** that are not decided yet at the top level. Concretely, each $\omega_i \in \Omega = \{\mu_i, \sigma_i^2\}$ represents the mean and variance of a Gaussian distribution, which serves as the prior density for a constant in the expression tree. Finally, even though the tokens are not inserted yet, S also contains information about where to place tokens from Θ or Ω.

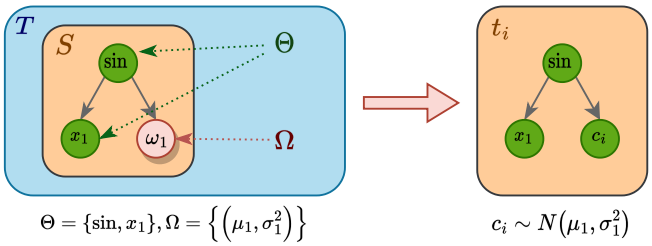

Figure 1: Template expression tree and an instantiation of it. Caption of this image is TBD but this will be referred by other sections of this paper.

On the local level, the concrete expression tree t_i for each data point $d_i = \{\mathbf{x}_i, y_i\}$ is simply a copy of the template T with all **local constants** instantiated with priors specified in Ω. As a concrete example, let $\Theta = \{\sin(\cdot), x_1^{(i)}\}$, $\Omega = \{\omega_1\}$ with $\omega_1 = \{0, 1\}$, and S being a complete binary tree of level-2 with $\sin$ on the root, the feature on the left child and the (uninstantiated) constant on the right. Figure 1 provides a visualization of this example.

2 Methods

The construction process of expressions $t_i, i = 1, ..., n$ from the template T has highlighted the hierarchical nature of HBSR. In order to formulate a complete hierarchical Bayesian model for SR, we need to further define the (hyper-)priors $p(T)$, the model likelihood $p(D \mid t_1, ..., t_n)$, and a proper Markov-Chain Monte-Carlo (MCMC) algorithm to sample from the posterior. We will explain each of these components in the below subsections.

(From this part on, I'm not trying to write the sections formally as these methodologies require further discussion. Instead I will try my best to list out certain concerns and issues we can think about together)

2.1 Priors

In the original BMS paper, they adopt the **exponential random graph model** framework to perform "moment matching" (i.e. align the means and variances of operation frequencies) on the operators. A few concerns are:

- this seems to ignore the effect of tree structure (i.e. S in our formulation). The BSR paper does look into this however.
- when we move from BSR towards HBSR, we need to place priors onto the template T instead of concrete trees, i.e. we need to provide hyper-priors for the $\omega_i = (\mu_i, \sigma_i^2)$. These parameters for Gaussian are not "frequencies" for operators and need to be dealt with independently.
- it remains unclear to me (maybe it's done using some tricks in implementation), through reading the BMS paper, on how it places priors onto constants (i.e. I know that a constant can be viewed as an operator, but in addition to looking at the frequency of this operator, does BMS places any prior on its actual value?).

2.2 Likelihood

I believe that the log-likelihood approximation (using Gaussian noise assumption) mentioned in the BSR supplementary text still applies to HBSR, without much theoretical adjustment needed. With the instantiated trees $t_1, ..., t_n$, our model assumes that

$$y_i = f_i(x_1^{(i)}, ..., x_k^{(i)}) + \epsilon_i \quad \epsilon_i \sim N(0, \sigma^2) \tag{1}$$

where f_i is the expression corresponding to tree t_i. The MLE for σ^2 is computed as

$$\hat{\sigma}^2 = \frac{\sum_i (f_i(x_1^{(i)}, ..., x_k^{(i)}) - y_i)^2}{n} \tag{2}$$

However, one of the confusing part (not to say it would cause any problem in implementation, but more on the formulation) is that the original BMS also assumes additional θ which need to be optimized via MLE (which means that those θs are considered fixed instead of stochastic). We can discuss about this further if you don't get what I'm concerned about here.

Overall, the likelihood is expressed as

$$p(D|f_1, ..., f_n) = \prod_i p(d_i|f_i) \tag{3}$$

$$= (2\pi\hat{\sigma}^2)^{-n/2} \exp\left\{ \frac{\sum_i (f_i(x_1^{(i)}, ..., x_k^{(i)}) - y_i)^2}{2\hat{\sigma}^2} \right\} \tag{4}$$

and the corresponding log-likelihood follows

$$\log p(D|f_1, ..., f_n) = -\frac{n}{2} \log 2\pi\hat{\sigma}^2 \tag{5}$$

$$+ \frac{\sum_i (f_i(x_1^{(i)}, ..., x_k^{(i)}) - y_i)^2}{2\hat{\sigma}^2} \tag{6}$$

2.3 Posterior inference with MCMC

In HBSR, the difference between a template expression T and local (instantiated) expressions t_i makes posterior more complicated. To make things simpler, we can consider the joint posterior

$$p(T, t_1, ..., t_n|D) \tag{7}$$

and if we would love to consider our model as the collection $M = \{T, t_1, ..., t_n\}$, the usual BIC-approximation still applies if we do not consider potential approximation error. A few remaining issues include

- For the joint model M, how do we determine the number of parameters in M (i.e. there are potential redundancies).
- The original BMS paper has an integral form for the joint model & data $p(D, f_i)$; this integration form is crucial for justifying the use of BIC. Whether this would change in HBSR requires further investigation.

Overall, I believe the adjustment on the posterior inference (both theoretically and practically in terms of the parallel MCMC design) is definitely needed, but not a priority. As the posterior is closely related to the prior & likelihood formulation. We can take a step-by-step approach to figuring out formulations for the former ones.

3 Experiments

TBD

References