Symbolic Regression via Tree MCMC

1 Model

We use a tree model to represent a symbolic expression $g(\cdot)$ from input data $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, a vector whose elements are features like close prices, volumes, etc. of assets to the output data $y \in \mathbb{R}$, which can be returns or prices of assets. To illustrate, the expression may be $g(\mathbf{x}) = x_1 + 2\cos(x_2) + e^{x_3} + 0.1$.

Traditionally, symbolic regression is addressed by combinatorial optimization methods like genetic programming. Instead we express the symbolic expression with a tree structure, pose a probability structure on different solutions to symbolic regression, model the problem with Bayesian method and sample the posterior distribution of tree structures (equivalent to the expressions) using MCMC algorithms. We expect that regardless of the starting point, after sufficiently many sampling, the posterior distribution resembles the ground truth.

1.1 Equivalent Tree Structure

We assume that the output is obtained by some arithmetic expression of input elements, i.e. the combination of basic operators like $+, \times, \exp()$, etc. The expression can be equivalently represented by a tree denoted by T, with non-terminal nodes indicating operations and terminal nodes indicating selected features.

The tree structure we adopt is binary tree, but not necessarily complete. Specifically, a non-terminal node has one child node if it is a unary operator, and two if it is a binary operator. For example, a non-terminal node with operator $+$ represents the operation that the values of its two child nodes are added up. For a non-terminal unary operator, for example $\exp()$, it means taking exponential of the value of its child node.

In the simple case, we only consider four basic unary operators $f(x) = \exp(x)$, $f(x) = ax + b$, $f(x) = 1/x$, $f(x) = -x$ and two binary operators $f(x, y) = x + y$, $f(x, y) = xy$ as building blocks of the tree.

On the other hand, each terminal node η specified by $i_k \in M$ represents a particular feature x_{i_k} of the data vector, which is not necessarily distinct. Terminal nodes function as the starting points of calculation. In calculation of a tree of depth d, we start from the terminal nodes, look at the parents of terminal nodes and conduct the operations indicated by them. For example, we add some two terminal nodes and assign the added value to their common parent node. Then we operate on the nodes of depth $d-1$ accordingly. Finally we arrive at the root node and get the output.

To sum, the tree structure T consists of the set of nodes, denoted by η's, corresponding to operators and having zero to two child nodes. Some operators require parameters, which is summarized in Θ. There are also terminal nodes that selects particular features, which is specified by $M = (i_1, \ldots, i_p)$, here i_k indicates using x_{i_k} of vector $\mathbf{x}$ as the input of the node. The specification of T, Θ and M gives an expression, or function $g(\cdot; T, M, \Theta)$, which obtains $g(\mathbf{x}; T, M, \Theta)$ from input $\mathbf{x}$.

The final output y is a blurred version of $g(\mathbf{x}; T, M, \Theta)$. We assume that the output is obtained by calculating from terminal to root and adding some noise, i.e.

$$y = g(\mathbf{x}; T, M, \Theta) + \epsilon \quad \epsilon \sim N(0, \sigma^2)$$

where $g(\cdot; T, M, \Theta)$ is the arithmetic expression corresponding to the tree structure T with parameter Θ which uses features indicated by M as input.

2 Prior distributions

The crucial part of Bayesian model is to assign prior distributions on the random structures. In our model, we are interested in inferring the tree structure T with parameter Θ and the features adopted, indicated by M. Also we assume that the parameter Θ follows a Bayesian structure, whose hyper-parameter is σ_Θ.

The joint distribution is given by

$$\begin{aligned} p(y, T, M, \Theta, \sigma, \sigma_\Theta \mid x) &= p(y \mid x, T, M, \Theta, \sigma) p(M, T) p(\Theta \mid T, \sigma_\Theta) p(\sigma_\Theta) p(\sigma) \\ &= p(y \mid x, T, M, \Theta, \sigma) p(M \mid T) p(T) p(\Theta \mid T, \sigma_\Theta) p(\sigma_\Theta) p(\sigma) \end{aligned}$$

There are several types of variables to be specified.

2.1 Prior of Tree Structure

We assign a prior $p(T)$ for tree structure T by describing the process a specific tree structure is generated. Unlike the calculation with a given tree which starts from terminal nodes, the construction of a tree starts from the root node. The topology structure of T depends on the operator assignment procedure. The node is randomly assigned a particular operator, which indicates whether it extends to one child node, or split into two child nodes, or function as a terminal node.

The prior of T, i.e., that of operator assignment resembles that in [1] which is used to construct complete binary tree. The depth of a node η, which is defined by the number of edges between η and the root, is adopted to specify the probability.

Specifically, for a node with depth d_η, with probability

$$p_1(\eta, T) = \alpha_1(1 + d_\eta)^{-\beta}$$

it has one child, then with probability 1/4 it is assigned one of the unary operators $f(x) = \exp(x)$, $f(x) = ax + b$, $f(x) = 1/x$, $f(x) = -x$. With probability

$$p_2(\eta, T) = \alpha_2(1 + d_\eta)^{-\beta}$$

it has two child nodes, then with probability 1/2 it is assigned one of the binary operators $f(x, y) = x + y$, $f(x, y) = xy$. Otherwise it is a terminal node. Here α_1, α_2, β are prefixed parameters.

2.2 Prior of Linear Parameters

Some basic operators needs specific parameters. For example, linear transformation $f(x) = ax + b$ takes parameters (a, b). In our simple case we only consider this kind of parameterized operator, and denote the parameters with Θ, which only depends on T and the set $L(T)$ of nodes indicating linear transformation in T.

If there exists some node η which is assigned a linear transformation operator $f(x) = lt(x) = a_\eta x + b_\eta$, the prior of the parameter Θ is taken to be

$$p(\Theta \mid T) = \prod_{\eta \in L(T)} p(a_\eta, b_\eta)$$

where the prior is that a_η's, b_η's are independent and

$$a_\eta \sim N(1, \sigma_a^2), \quad b_\eta \sim N(0, \sigma_b^2)$$

This indicates that the prior of the linear transformation is random around identity function. The prior of $\sigma_\Theta = (\sigma_a, \sigma_b)$ is conjugate prior of normal distribution, which is

$$\sigma_a^2 \sim IG(\nu_a/2, \nu_a\lambda_a/2), \quad \sigma_b^2 \sim IG(\nu_b/2, \nu_b\lambda_b/2)$$

where ν_a, λ_a, ν_b, λ_b are prefixed hyper-parameters.

2.3 Prior of Terminal Nodes

The number and locations of terminal nodes depend on structure of T. Conditioned on T, the specific feature that one terminal node takes is uniform over all d features of input $\mathbf{x} \in \mathbb{R}^d$.

2.4 Prior of σ

The prior of σ is taken to be the conjugate distribution

$$\sigma^2 \sim IG(\nu/2, \nu\lambda/2)$$

where ν and λ are prefixed parameters.

3 Metropolis-Hastings Algorithm

We use Metropolis-Hastings (MH) algorithm to simulate the Markov sequence of different trees $(T^0, M^0, \Theta^0), (T^1, M^1, \Theta^1), (T^2, M^2, \Theta^2), \ldots$ which converge in distribution to the posterior distribution $p(M, \Theta, T \mid X, Y)$.

Our model involves two parts of sampling. The first part is the structure specified by T and M, which is discrete. Another part is Θ aggregating parameters of all linear transformation nodes. The dimension of Θ may vary according to (T, M) since the number of linear transformation nodes vary among different tree structures. We use Reversible Jump MCMC algorithm to address the trans-dimensional problem. For simplicity, we denote the structure parameters as $S = (T, M)$.

3.1 Structure transition kernel

We first specify how the sampling algorithm jumps from a tree structure to a new one. Four reversible actions are defined as below.

- **Stay:** If the expression involves $n_l \geq 0$ `lt()` operators, with probability

$$p_0 = n_l/4(n_l + 3),$$

the structure $S = (T, M)$ stays unchanged, and ordinary MH step follows to sample new linear parameters.

- **Grow:** Uniformly pick a terminal node to activate. A sub-tree is then generated iteratively, where each time a node is randomly terminated or assigned an operator according to the prior until all nodes are terminated or assigned.

To regularize the complexity of the expression, when the tree depth and amount of nodes are large, the proposal grows with lower probability. Therefore the probability of Grow is set as

$$p_g = \frac{1 - p_0}{2} \cdot \min\left\{1, \frac{8}{N + d + 2}\right\}$$

where N is the total amount of all nodes, and d is the depth of the tree.

- **Prune:** Uniformly pick a non-terminal node and turn it into a terminal node by discarding its child node(s). Then randomly choose one from the d features of $\mathbf{x}$ to the newly pruned terminal node.

We set the probability of Prune as

$$p_p = \frac{1 - p_0}{2} - p_g$$

such that Grow and Prune share half of the probability that the structure does not Stay.

- **ReassignOperator:** Uniformly pick a non-terminal node, and assign a new operator, which is uniformly sampled from the set of basic functions. If the type of the node is not changed, then nothing else is needed. If the node changes from unary to binary, then the original

child is taken as the left child, and a new sub-tree is generated as the right child. If the node changes from binary to unary, then we preserve the left sub-tree (this is to make the transition reversible).

- **ReassignFeature:** Uniformly pick a terminal node and assign another feature with the aforementioned prior probability to it.

  The probability of ReassignOperator and ReassignFeature is set as

$$p_{ro} = p_{rf} = \frac{1 - p_0}{4}$$

In the above five kinds of actions, Grow may increase the dimension of parameter Θ if new nodes are assigned `lt()` operators. Prune may decrease the dimension of Θ if the sub-trees include `lt()` operators. ReassignOperator may increase or decrease the dimension if it involves linear transformation operator. ReassignFeature will not change the dimension of Θ. We denote the transition probability from $S = (T, M)$ to $S^* = (T^*, M^*)$ as $q(S^* \mid S)$.

3.2 Jump between models

We adopt reversible jump MCMC algorithm to address the dimension expansion or shrinkage. Generally, in reversible jump MCMC, the crucial parts are to sample auxiliary variables which allows the dimensionality to be equal, and transform from one model to another, finally drop the auxiliary ones. Here auxiliary variables are also used to get new values for the staying variables.

After we generate S^* from S, there are three situations.

- **No change.** When the new structure does not change the number of `lt()` nodes, the dimensionality of parameters does not change. In this case, we do not need reversible jump MCMC and instead use ordinary MH step.

  Note that in this case, the set of `lt()` nodes may change, but the sampling of new parameters is i.i.d., so the MH step is sufficient.

- **Expansion.** When the number of `lt()` nodes increases, the dimensionality of Θ, denoted by p_Θ, increases. Note that we may simultaneously lose some original `lt()` nodes and get some new `lt()` nodes. But due to the i.i.d. sampling of parameters we only need to care about the number of all `lt()` nodes.

In this case, we sample auxiliary variables $U = (u_\Theta, u_n)$ where u_Θ is of the same dimension as Θ, and u_n has the same dimension as the increased part.

The hyper-parameters $U_\sigma = (\sigma_a^2, \sigma_b^2)$ are firstly sampled independent from the inverse gamma prior, then each element of u_Θ and newly-added parameters (wrapped in u_n) is sampled independent from $N(1, \sigma_a^2)$ or $N(0, \sigma_b^2)$. Then the new parameter candidate Θ^* along with corresponding auxiliary variable U^* is obtained by

$$(U^*, \Theta^*, \sigma_\Theta^*) = j_e(\Theta, U, U_\sigma) = j_e(\Theta, u_\Theta, u_n, U_\sigma) = \Big(\frac{\Theta - u_\Theta}{2}, \frac{\Theta + u_\Theta}{2}, u_n, U_\sigma\Big) \tag{1}$$

where

$$U^* = \frac{\Theta - u_\Theta}{2}, \quad \Theta^* = (\frac{\Theta + u_\Theta}{2}, u_n), \quad \sigma_\Theta^* = U_\sigma$$

- **Shrinkage.** When the number of `lt()` nodes decreases, p_Θ decreases. Similar to the Expansion case, here some original nodes are lost but could also have new ones (especially in the ReassignOperator transition).

  In this case, the change from Θ to Θ^* needs auxiliary variables to generate a new sample for remaining parameters, meanwhile transformation j_e in equation (1) is reversed. Specifically, assume that the original parameter is $\Theta = (\Theta_0, \Theta_d)$ where Θ_d corresponds to the vector of parameters to be dropped.

  Firstly, $U_\sigma = (\sigma_a^2, \sigma_b^2)$ are sampled from the independent inverse gamma prior. Then the new parameter candidate is obtained by first sampling U of the same dimensionality as Θ_0, whose elements are independently sampled from $N(0, \sigma_a^2)$ and $N(0, \sigma_b^2)$, respectively. Note that we incorporate σ_a^2 and σ_b^2 into σ_Θ. Then the new candidate Θ^* as well as the corresponding auxiliary variable U^* is obtained by

  $$(\sigma_\Theta^*, \Theta^*, U^*) = j_s(U_\sigma, U, \Theta) = j_s(U_\sigma, U, \Theta_0, \Theta_d) = (U_\sigma, \Theta_0 + U, \Theta_0 - U, \Theta_d) \tag{2}$$

  where

  $$\sigma_\Theta^* = U_\sigma, \quad \Theta^* = \Theta_0 + U, \quad U^* = (\Theta_0 - U, \Theta_d)$$

For simplicity, we denote the two transformation j_e, j_s as j_{S,S^*}, indicating that this is a transformation from parameters of S to those of S^*. The

auxiliary variables are denoted as U and U^* respectively. Note that (Θ, U) and (Θ^*, U^*) are of the same dimensionality.

According to the reversible jump MCMC algorithm, the procedure of one sampling is as follows.

- Start from the state $(S^{(t)}, \Theta^{(t)})$.
- Propose a candidate model S^* by sampling $S^* \mid S^{(t)} \sim q(\cdot \mid S^{(t)})$.
- – If the set of linear nodes changes, sample auxiliary $U^{(t)}$ as described above from proposal density $h(U^{(t)} \mid \Theta^{(t)}, S^{(t)}, S^*)$ where the sampling of σ_a^2. σ_b^2 is included, and obtain $(U^*, \Theta^*) = j_{S^{(t)},S^*}(\Theta^{(t)}, U^{(t)})$ as in Equation (1) or (2). Calculate the ratio

$$R = \frac{f(y \mid S^*, \Theta^*, x) f(\Theta^* \mid S^*) f(S^*) q(S^{(t)} \mid S^*) h(U^* \mid \Theta^*, S^*, S^{(t)})}{f(y \mid S^{(t)}, \Theta^{(t)}, x) f(\Theta^{(t)} \mid S^{(t)}) f(S^{(t)}) q(S^* \mid S^{(t)}) h(U^{(t)} \mid \Theta^{(t)}, S^{(t)}, S^*)} \left| \frac{\partial j_{S^{(t)},S^*}(\Theta^{(t)}, U^{(t)})}{\partial(\Theta^{(t)}, U^{(t)})} \right| \tag{3}$$

– If the set of linear nodes does not change, directly sample new parameters Θ^* from $f(\cdot \mid S^*)$ and calculate the ratio

$$\begin{aligned} R &= \frac{f(y \mid S^*, \Theta^*, x) f(\Theta^* \mid S^*) f(S^*)}{f(y \mid S^{(t)}, \Theta^{(t)}, x) f(\Theta^{(t)} \mid S^{(t)}) f(S^{(t)})} \cdot \frac{q(S^{(t)} \mid S^*) f(\Theta^{(t)} \mid S^{(t)})}{q(S^* \mid S^{(t)}) f(\Theta^* \mid S^*)} \\ &= \frac{f(y \mid S^*, \Theta^*, x) f(S^*) q(S^{(t)} \mid S^*)}{f(y \mid S^{(t)}, \Theta^{(t)}, x) f(S^{(t)}) q(S^* \mid S^{(t)})} \end{aligned} \tag{4}$$

- Accept the proposed move to model S^* and Θ^* with probability $\alpha = \min(1, R)$.

Note that the density $h(U^* \mid \Theta^*, S^*, S^{(t)})$ incorporates the density of sampling σ_a^2, σ_b^2 from the prior.

4 Sequential Monte Carlo Algorithm

Another approach of approximating the posterior distribution is the Sequential Monte Carlo (SMC) algorithm. We use a modification of the algorithm in [2]. Instead of the MH algorithm which employs some transformations to propose new trees, a symbolic tree is sampled from a root node, where in

each step some node is selected to split or stop and the weight is updated. No pruning is carried out. Then the trees are resampled according to the weights. Using the same notations, a symbolic tree is described by (S, Θ) where $S = (T, M)$.

4.1 Proposal kernel

4.1.1 Prior proposal

A natural choice of generating a tree (S^*, Θ^*) from (S, Θ) is to propose from prior distribution described in Section 2.

Specifically, starting from $S = (T, M)$, the proposal $\mathbb{Q}_i(\cdot \mid S, \Theta)$ chooses a non-terminal node η with depth d_η, assign it one child node with probability $p_1(\eta) = \alpha_1(1+d_\eta)^{-}\beta$ or two child nodes with probability $p_2(\eta) = \alpha_2(1+d_\eta)^{-}\beta$ or otherwise stop it as terminal node. Then if the node is not stopped, it is assigned an operator uniformly as described in Section 2. If it corresponds to linear transformation, its parameters (a_η, b_η) is sampled from prior $a_\eta \sim N(1, \sigma_a^2)$, $b_\eta \sim N(0, \sigma_b^2)$, where, different from previous setting, σ_a^2 and σ_b^2 are prefixed values.

On the other hand, if the picked node η is stopped as a terminal node, its associated feature x_{i_η} is selected uniformly among $x_1, \ldots, x_d$.

In each stage, the growing node is chosen in a deterministic rule. One possible method is to choose all nodes of depth d_i. Another method is to choose the oldest node (with the smallest depth), i.e. in a breadth-first way.

4.1.2 One-step optimal kernel

Another choice is one-step optimal kernel. First we define target distributions P_i^y. At stage i with $(S_i, \Theta_i$, we generate label y' with likelihood $f(y' \mid S_i, \Theta_i, x)$ as if (S_i, Θ_i) is the final tree. Then P_i^y is defined as the conditional distribution of (S_i, Θ_i) given $y' = y$. The distribution is described in [2] as the posterior with a truncated prior.

4.1.3 Algorithm

The algorithm is as follows.

Data: Feature X, label y

Inputs: Number of stages N, Number of samples M;

Initialize: $T_0^{(m)}$, $\Theta_0^{(m)}$, $\sigma_{\Theta 0}^{(m)}$, $\sigma^{(m)}$; $m = 1, \ldots, M$, $w_0^{(m)} = 1$, $W_0 = \sum_m w_0^{(m)}$;

for *i=1:N* **do**

  for *m=1:M* **do**

    Sample $(T_i^{(m)}, \Theta_{i-1}^{(m)})$ from $\mathbb{Q}(\cdot \mid T_i^{(m)}, \Theta^{(m)})$;

    Update weights:

$$
\begin{aligned}
w_i^{(m)} &= \frac{P(S_i^{(m)}, \Theta_i^{(m)}) f(y \mid S_i^{(m)}, \Theta_i^{(m)}, x)}{\mathbb{Q}_i(S_i^{(m)}, \Theta_i^{(m)} \mid S_{i-1}^{(m)}, \Theta_{i-1}^{(m)}) P(S_{i-1}^{(m)}, \Theta_{i-1}^{(m)})} \\
&= w_{i-1}^{(m)} \frac{P(S_i^{(m)}, \Theta_i^{(m)} \mid S_{i-1}^{(m)}, \Theta_{i-1}^{(m)}) f(y \mid S_i^{(m)}, \Theta_i^{(m)}, x)}{\mathbb{Q}_i(S_i^{(m)}, \Theta_i^{(m)} \mid S_{i-1}^{(m)}, \Theta_{i-1}^{(m)}) f(y \mid S_{i-1}^{(m)}, \Theta_{i-1}^{(m)}, x)}
\end{aligned}
$$

  end

  Compute normalization $W_i = \sum_m w_i^{(m)}$;

  Normalize weights $\bar{w}_i^{(m)} = w_i^{(m)} / W_i$;

  if $(\sum_m (\bar{w}_i^{(m)})^2)^{-1} < threshold$ **then**

    Resample indices j_m according to $\bar{w}_i$;

    Assign new samples according to indices;

  end

  if *All nodes are terminal* **then**

    exit for loop

  end

end

Algorithm 1: SMC for Symbolic Regression Tree

References

[1] Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayesian cart model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

[2] Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Top-down particle filtering for bayesian decision trees. *30th International Conference on Machine Learning, ICML 2013*, 03 2013.