Восстановление матрицы суперпозиции в задаче символьной регрессии*

Р. Г. Нейчев[1], И. А. Шибаев[2], В. В. Стрижов[3]

Аннотация: Исследуется проблема порождения структуры регрессионной модели. Модель представляет собой суперпозицию базовых функций. Структура модели описывается взвешенным цветным графом. Каждая вершина графа соответствует некоторой базовой функции. Ребро задает суперпозицию двух функций. Вес ребра равен вероятности суперпозиции. Для создания оптимальной модели необходимо восстановить ее структуру по матрице смежности графа. Предлагаемый алгоритм восстанавливает минимальное остовное дерево из взвешенного цветного графа. В работе представлено новое решение, основанное на алгоритме дерева Штейнера. Алгоритм сравнивается с альтернативами.

Ключевые слова: символьная регрессия; линейное программирование; суперпозиция; минимальное остовное дерево; матрица смежности

1 Введение

Символьная регрессия — это метод построения нелинейной модели, аппроксимирующей выборку. Структура модели определяется суперпозицией базовых функций. Набор базовых функций фиксируется для конкретной прикладной задачи. Структуры альтернативных моделей генерируются алгоритмом оптимизации для выбора оптимальной модели. В данной статье предлагается определять структуру модели с помощью вероятностного графа. Остовное дерево в графе определяет некоторую суперпозицию. Для выбора оптимальной модели необходимо реконструировать минимальное остовное дерево по графу.

*Работа выполнена при поддержке РФФИ (проект 20-37-90050).

[1]Московский физико-технический институт, neychevr@gmail.com

[2]Московский физико-технический институт, shibaev.kesha@gmail.com

[3]Федеральный исследовательский центр «Информатика и управление» Российской академии наук, strijov@phystech.edu

Методы генетического программирования [1] находят оптимальное подмножество в наборе суперпозиций базовых функций, но имеют высокую вычислительную сложность. В [2] описаны методы, понижающие сложность. Они используют дополнительные ограничения на суперпозиции, например, используют линейные комбинации базовых функций. Символьная регрессия, описанная в [3], используется для оптимизации структуры суперпозиции. Методы решения задачи символьной регрессии основаны на матричном представлении структуры модели [4]. Однако эти методы не содержат ограничений на число аргументов базовых функций и на структуру графа, обеспечивающую допустимую суперпозицию. В этой работе решается задача построения модели с помощью символьной регрессии.

Требуется восстановить допустимую суперпозицию из предсказанной матрицы смежности с вероятностями ребер. Решается задача восстановления k-минимального остовного дерева k-MST (англ. minimum-cost spanning tree). Эта задача NP-сложная, поэтому применимы только приближенные решения [5]. Алгоритм k-MST эквивалентен проблеме дерева Штейнера PCST (англ. Prize-collecting Steiner tree) из-за его эквивалентности ослабленной формулировке постановки задачи линейного программирования [6]. В работах [5, 7, 8] представлены приближенные решения задачи k-MST.

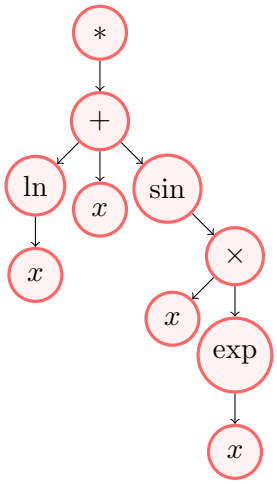

Рис. 1: Структура регрессионной модели представляет собой ориентированный граф

Предлагаемое решение основано на упрощенной версии задачи k-MST, которая трансформируется в задачу PCST с постоянными призами, одинаковыми для всех вершин. Быстрый алгоритм PSCT описан в [9]. Альтернативное решение основано на алгоритме $(2 - \varepsilon)$-аппроксимации для задачи PSCT. Она сравнивается с другими алгоритмами, включая алгоритмы обхода дерева в глубину, обхода дерева в ширину, алгоритмы Прима.

2 Задача выбора регрессионной модели

Требуется выбрать регрессионную модель φ из набора альтернативных моделей. Модель описывает выборку $D = \{(x_i, y_i)\}$ и минимизирует ошибку

$$\hat{\varphi}(D) = \arg\min_{\varphi} \sum_{i=1}^{m} \big(\varphi(x_i) - y_i\big)^2. \tag{1}$$

Модель представляет собой суперпозицию базовых функций из некоторого заданного набора. На рис. 1 показан ее пример. Структура модели φ, суперпозиция, соответствует графу $G = (V, E)$, где базовые функции находятся в вершинах V. Корневая вершина обозначается через $*$. Модель $\varphi(D) = \ln(x) + x + \sin\big(x \times \exp(x)\big)$. Ее структура в виде матрицы смежности графа представлена в табл. 1. Базовые функции перечислены в первой строке. Элементами матрицы являются вероятности ребер E дерева. Жирным шрифтом выделены ребра восстановленного дерева M, образующие суперпозицию φ. Для восстановления структуры модели φ как суперпозиции, заданной деревом M, необходимы только графовое представление G и базовые функции.

Таблица 1: Вероятности суперпозиций в матрице смежности порождают ориентированный граф

Арность	Функция	$*$	$+$	ln	sin	$\times$	exp	x
1	$*$	0,2	**0,7**	0,5	0,4	0,5	0,3	0,2
3	$+$	0,3	0,2	**1.0**	**0,8**	0,6	0,3	**0,7**
1	ln	0,3	0,2	0,0	0,0	0,1	0,5	**0,5**
1	sin	0,1	0,4	0,0	0,5	**0,9**	0,2	0,5
2	$\times$	0,3	0,0	0,3	0,5	0,0	**0,8**	**0,6**
1	exp	0,3	0,3	0,4	0,1	0,5	0,4	**0,4**

Поставим задачу восстановления структуры модели. Задано множество выборок $\{D_k\}$. Каждой выборке D_k соответствует своя модель. Эта модель имеет структуру M_k. Таким образом, имеется набор пар $\{(D_k, M_k)\}$, выборка и структура. Обозначим через P отображение, которое предсказывает вероятности узлов в графе G по выборке D. Для выбора модели $\varphi(D)$ необходимо восстановить структуру модели M по графу G. Обозначим алгоритм восстановления дерева через R. Регрессионная модель $\hat{\varphi}(D)$, которая решает задачу (1), определяется формулой $\hat{M} = R\big(P(D)\big)$. Поскольку дерево M играет центральную роль в этой работе, критерий качества алгоритма восстановления дерева имеет вид:

$$\min_{M_k \in G} \frac{1}{K} \sum_{k=1}^{K} [\hat{M}_k = M_k)].$$

Восстановленное дерево должно быть эквивалентно заданному дереву, следовательно, выбранная модель регрессии приближает выборку.

3 Задача восстановления дерева суперпозиции

Требуется восстановить дерево M_k, задающее суперпозицию и решающее задачу (1). Задан ориентированный взвешенный граф $G = (V, E)$ с раскрашенными вершинами v_i и корневой вершиной r. Каждая вершина $v_i \in V$ имеет свой цвет $t(v_i) = t_i$. Каждое ребро $e_i \in E$ имеет свой вес $w(e_i) = c_i \in [0, 1]$.

Требуется восстановить ориентированное дерево минимального веса с корнем r. Оно должно покрывать не менее k вершин в заданном графе G. А число ребер, выходящих из вершины v_i дерева, должно быть меньше или равно t_i. Корень r имеет одно ребро, $t_r = 1$.

Сформулируем это условие в виде задачи линейного программирования с целочисленными ограничениями:

$$
\begin{aligned}
\underset{\substack{x_e, z_S \\ e\in E, S\subseteq V\setminus\{r\}}}{\text{minimize}} \quad & \sum_{e\in E} c_e x_e \\
\text{s.t.} \quad & \sum_{\substack{e\in\delta(S):\\ e=(*,v_i),\ v_i\in\delta(S)}} x_e + \sum_{T:T\supseteq S} z_T \geqslant 1, && S \subseteq V\setminus\{r\}, \\
& \sum_{e\in E:\ e=(*,v)} x_e \leqslant 1, && v \in V, \qquad (2) \\
& \sum_{e\in E:\ e=(v,*)} x_e \leqslant t_i, && v \in V, \\
& \sum_{S\subseteq V\setminus\{r\}} |S| z_S \leqslant n - k, \\
& x_e \in \{0, 1\}, && e \in E, \\
& z_S \in \{0, 1\}, && S \subseteq V\setminus\{r\},
\end{aligned}
$$

где $x_e = 1$, если ребро e входит в финальную суперпозицию, и $x_e = 0$ в противном случае, $z_S = 1$ для всех вершин, исключенных из финальной суперпозиции. Обозначим через $e = (*, v)$ ориентированное ребро с листом v. Обозначим через $e = (v, *)$ ориентированное ребро с вершиной v.

Первое ограничение (2) определяет структуру графа решения в виде дерева с корнем r. Второе ограничение определяет ориентацию дерева: каждая вершина имеет не более одного входящего ребра. Третье ограничение определяет арность используемых базовых функций, поэтому число ребер, имеющих определенную вершину в качестве источника, фиксировано. Четвертое ограничение говорит, что итоговое дерево имеет не менее k вершин. Если все веса неотрицательны, то четвертое ограничение на минимальное число вершин принимает более строгий вид: число вершин должно быть равно k. Однако более слабое ограничение позволяет найти возможные связи с другими оптимизационными задачами. Ограничения в (2) преследуют ту же цель.

4 Алгоритмы восстановления дерева k-MST и PCST

Определение 1 (*k*-минимальное остовное дерево *k*-MST). *Задан взвешенный граф $G = (V, E)$ с корнем r и весами ребер $w(e_i) = c_i \geqslant 0$, $e_i \in E$. Требуется построить ориентированное дерево минимального веса с корнем r, покрывающее не менее k вершин в G.*

Если та же задача ставится для ориентированных графов, то конечное дерево с корнем r должно быть ориентированным. Задача линейного программирования для направленного k-MST исключает третье условие в (2). В таком виде задача k-MST отличается от исходной задачи восстановления дерева суперпозиций (2) отсутствием третьего ограничения на арность базовых функций. Это эквивалентно ограничению на число ребер, выходящих из вершины.

Определение 2 (призовое дерево Штейнера PCST). *Задан взвешенный граф $G = (V, E)$ с корнем r и весами ребер $w(e_i) = c_i \geqslant 0$, $e_i \in E$, где каждой вершине $v_i \in V$ присвоен приз $\pi(v_i) = \pi_i \geqslant 0$. Требуется построить дерево T с корнем r, которое минимизирует функционал*

$$\sum_{e \in E} c_e x_e + \sum_{S \subseteq V \setminus \{r\}} \pi(S) z_S,$$

где $x_e \in \{0, 1\}$, $x_e = 1$, если $e \in E$ входит в тройку T, $z_S \in \{0, 1\}$, $z_S = 1$ для всех вершин, исключенных из дерева T, $S = V \setminus V(T)$ и $\pi(S) = \sum_{v \in S} \pi(v)$.

В случае ориентированных графов эта задача обобщается до асимметричной задачи A-PCST. Задача линейного программирования для A-PCST принимает вид:

$$\begin{aligned}
\underset{\substack{x_e, z_S \\ e \in E, S \subseteq V \setminus \{r\}}}{\text{minimize}} \quad & \sum_{e \in E} c_e x_e + \sum_{S \subseteq V \setminus \{r\}} \pi(S) z_S & & \\
\text{s.t.} \quad & \sum_{\substack{e \in \delta(S): \\ e = (*, v_i),\ v_i \in \delta(S)}} x_e + \sum_{T : T \supseteq S} z_T \geqslant 1, & & S \subseteq V \setminus \{r\}, \\
& \sum_{e \in E:\ e = (*, v)} x_e \leqslant 1, & & v \in V, \qquad (3) \\
& x_e \in \{0, 1\}, & & e \in E, \\
& z_S \in \{0, 1\}, & & S \subseteq V \setminus \{r\}.
\end{aligned}$$

Если последнее ограничение из (2) входит в оптимизируемый функционал, задачи k-MST и A-PCST имеют эквивалентные ограничения и отличаются только оптимизируемым функционалом. Такое преобразование возможно согласно условиям Каруша–Куна–Таккера [10]. Если значения призов эквивалентны $\pi(v) = \lambda$, единственное отличие состоит в постоянном члене $\lambda(n - k)$. Таким образом, задачи оптимизации k-MST

и A-PCST принимают вид:

$$\underset{\substack{x_e, z_S \\ e\in E, S\subseteq V\setminus\{r\}}}{\text{minimize}} \quad \sum_{e\in E} c_e x_e + \lambda \left(\sum_{S\subseteq V\setminus\{r\}} |S| z_S - (n-k) \right),$$

$$\underset{\substack{x_e, z_S \\ e\in E, S\subseteq V\setminus\{r\}}}{\text{minimize}} \quad \sum_{e\in E} c_e x_e + \lambda \sum_{S\subseteq V\setminus\{r\}} |S| z_S.$$

Константа λ обозначает неотрицательный множитель Лагранжа в задаче k-MST и приз за вершину в задаче A-PCST. Существует несколько алгоритмов для решения проблемы PCST, но не для решения проблемы A-PCST. Возможное решение — снять ограничения на ориентацию графа, чтобы алгоритм PCST мог позже восстановить ориентацию дерева.

5 Решение задачи восстановления ограниченного леса с помощью алгоритма $(2-\varepsilon)$-приближения

Обзор методов решения задачи восстановления ограниченного леса представлен в [11]. Задан взвешенный неориентированный граф $G = (V, E)$. Все его веса $w(e_i) = c_i \geqslant 0,\ e_i \in E$. Задана некоторая функция $f : 2^V \to \{0, 1\}$. Требуется решить задачу линейного программирования с целочисленными ограничениями:

$$\begin{aligned} \underset{x_e:\ e\in E}{\text{minimize}} \quad & \sum_{e\in E} c_e x_e & & \\ \text{s.t.} \quad & x\big(\delta(S)\big) \geqslant f(S), & S \subset V,\ S \neq \emptyset, & \qquad (4) \\ & x_e \in \{0,1\}, & e \in E, & \end{aligned}$$

где $x\big(\delta(S)\big) = \sum\limits_{e\in\delta(S)} x_e$; $x_e = 1$, если ребро e входит в финальное решение. Функция $\delta(S)$ обозначает все ребра из E такие, что только одна из смежных вершин входит в S.

Предположим, что отображение f удовлетворяет условиям

$$f(V) = 0, \underbrace{f(S) = f(V\setminus S)}_{\text{симметричность}}, \underbrace{A, B \subset V :\ A \cap B = \emptyset,\ f(A) = f(B) = 0 \to f(A \cup B) = 0}_{\text{дизъюнктивность}}.$$

При выполнении этих условий f задает число ребер, начинающихся в множестве вершин S.

Лемма 1. *Пусть $B \subseteq S \subset V$. Тогда $f(S) = 0$ и $f(B) = 0$ приводит к $f(S\setminus B) = 0$.*

Задача с таким описанием относится к *задачам поиска оптимального леса с ограничениями*. Такая постановка задачи (4) с соответствующим отображением f подходит для многих известных задач взвешенных графов, например: минимальный магистральный поиск, st-путь, задача Штейнера на минимальном дереве. Последняя задача является NP-полной, поэтому применим приближенный алгоритм.

Определение 3 (алгоритм α-аппроксимации). *Эвристический полиномиальный алгоритм, дающий решение некоторой задачи оптимизации, называется α-аппроксимацией, если он гарантирует удовлетворяющее ограничениям решение этой задачи оптимизации с коэффициентом, меньшим или равным α, так что решение отличается от оптимального не более чем в α раз по оптимизируемому функционалу.*

Чтобы предложить приближенный алгоритм, целочисленные ограничения в (4) должны быть ослаблены:

$$\begin{aligned} \underset{x_e:\ e\in E}{\text{minimize}} \quad & \sum_{e\in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e\in\delta(S)} x_e \geqslant f(S), && S\subset V,\ S\neq\emptyset, \\ & x_e > 0, && e\in E, \end{aligned} \tag{5}$$

Двойственная задача принимает вид:

$$\begin{aligned} \underset{y_S:\ S\subset V,\ S\neq\emptyset}{\text{maximize}} \quad & \sum_{S\subset V} f(S) y_S \\ \text{s.t.} \quad & \sum_{S:\ e\in\delta(S)} y_S \leqslant c_e, && e\in E, \\ & y_S > 0, && S\subset V,\ S\neq\emptyset, \end{aligned} \tag{6}$$

относительно дополнительного условия $y_S \cdot \left(\sum\limits_{e\in\delta(S)} x_e - f(S) \right) = 0, \quad S\subset V.$

Обозначим множество вершин $A = \{v \in V : f(\{v\}) = 1\}$. Предлагается адаптивный жадный алгоритм $\left(2 - \frac{2}{|A|}\right)$-аппроксимации для задач вида (4). Алгоритм состоит из двух этапов. На первом этапе он жадно объединяет кластеры вершин, увеличивая двойственные переменные y_S. Изначально каждая вершина принадлежит своему кластеру. Если следующее ребро e достигает равенства в ограничениях в (6), это ребро добавляется к множеству S и связанные кластеры объединяются. Этот этап аналогичен алгоритму минимального остовного дерева Крускала. На втором этапе из конечного множества S удаляются некоторые ребра. Если обрезка ребра не нарушает ограничений, то это ребро должно быть удалено.

Индекс Z_{DRLP} в алгоритме 1 обозначает линейное программирование с двойной релаксацией. Начальное значение $F := \emptyset$ в алгоритме 1 эквивалентно предположению $x_e = 0, \quad e\in E$. По условиям нежесткости $y_S = 0,\ S\subset V,\ S\neq\emptyset$.

На каждом шаге алгоритма кластер $\mathcal{C}$ содержит две компоненты $\mathcal{C} = \mathcal{C}_i \cup \mathcal{C}_a$, где $C \in \mathcal{C}_a$, если $f(C) = 1$, и $C \in \mathcal{C}_i$ в противном случае. Назовем $\mathcal{C}_a$ активным компонентом. Переменные $d(v)$ в этом алгоритме связаны с переменными y_S из (6) соотношением $d(i) = \sum\limits_{S:i\in S} y_S$.

Рассмотрим две различные компоненты $C_q,\ C_p,\ C_q \cap C_p = \emptyset$, на некоторой итерации первого этапа алгоритма. Все y_S должны быть равномерно распределены по некоторому ε без нарушения ограничений

$$\sum_{S:\ e \in \delta(S)} y_S \leqslant c_e.$$

В терминах $d(v)$ это условие принимает вид $\sum\limits_{S:\ e \in \delta(S)} y_S = d(v_1) + d(v_2),\ e = (v_1, v_2)$, поэтому $y_S = 0$ для любого S, такого что $v_1, v_2 \in S$, потому что компоненты растут только на первом этапе. Увеличение некоторых компонент на ε приводит к уравнению

$$d(v_1) + d(v_2) + \varepsilon \cdot \big(f(C_q) + f(C_p)\big) \leqslant c_e,\ e = (v_1, v_2),$$

что приводит к формуле, используемой в строке 10 алгоритма 1. В случае, когда в состав входит следующее ребро, сумма $\sum\limits_{S:\ e \in \delta(S)} y_S$ не будет увеличиваться, поэтому ограничения выполняются.

Ребра, которые можно удалить из F без добавления новых активных компонентов, удаляются на втором этапе алгоритма. Следующая лемма определяет свойства компонент связности в F'.

Лемма 2. *Для каждой компоненты связности N из F' выполняется равенство: $f(N) = 0$.*

Следующая теорема утверждает, что решение, полученное с помощью описанного алгоритма, удовлетворяет ограничениям исходной задачи линейного программирования.

Теорема 1. *Набор ребер F', полученный алгоритмом 1, удовлетворяет всем ограничениям исходной задачи* (4).

Лемма 3. *Обозначим граф H, каждая вершина которого соответствует одной из компонент связности $C \in \mathcal{C}$ на фиксированном шаге алгоритма. Ребро (v_1, v_2) присутствует, если существует ребро $\hat{e}$ исходного графа, входящее в F': $\hat{e} \in F'$, поэтому граф H — это лес. Внутри H нет листовых вершин, соответствующих неактивным вершинам исходного графа.*

Теорема 2. *Алгоритм 1 представляет собой α-приближенный алгоритм для задачи* (4) *с $\alpha = 2 - \frac{2}{|A|}$, где $A = \{v\ V : f(\{v\}) = 1\}$.*

Несмотря на эту теоретическую основу, не существует подходящей функции f для постановки задачи PCST, указанной в (4). Чтобы быть применимым в этих условиях, алгоритм 1 нуждается в нескольких модификациях.

6 Модифицированная постановка задачи для PCST

Как и в случае A-PCST, упрощенный вид задачи линейного программирования PCST принимает вид:

$$\begin{aligned}
\underset{\substack{x_e, s_v \\ e\in E, v\in V\setminus\{r\}}}{\text{minimize}} \quad & \sum_{e\in E} c_e x_e + \sum_{v\in V\setminus\{r\}} (1-s_v)\pi_v \\
\text{s.t.} \quad & \sum_{e\in\delta(S)} x_e \geqslant s_v, && S\subseteq V\setminus\{r\},\ v\in S, \qquad (7)\\
& x_e \geqslant 0, && e\in E,\\
& s_v \geqslant 0, && v\in V\setminus\{r\}.
\end{aligned}$$

Эта постановка задачи отличается от исходной (3) тем, что с ней возможно согласовать задачу k-MST. Индикаторы s_v показывают, что вершина v включена в дерево.

Двойственная задача принимает вид:

$$\begin{aligned}
\underset{y_S:\ S\subset V\setminus\{r\}}{\text{maximize}} \quad & \sum_{S\in V\setminus\{r\}} y_S \\
\text{s.t.} \quad & \sum_{S: e\in\delta(S)} y_S \leqslant c_e, && e\in E, \qquad (8)\\
& \sum_{S\subseteq T} y_S \leqslant \sum_{v\in T} \pi_v, && T\subset V\setminus\{r\},\\
& y_S \geqslant 0, && S\subset V\setminus\{r\}.
\end{aligned}$$

Алгоритм 2 решает эту задачу. Он похож на алгоритм 1. Двойные переменные должны обновляться равномерно с дополнительными ограничениями. Тогда ε примет минимальное из двух значений в соответствии с обеими группами ограничений. Более широкий анализ аппроксимационных свойств обновленного алгоритма представлен в [11]. Алгоритм 2 представляет собой α-приближенный алгоритм для задачи PCST с $\alpha = 2 - \frac{2}{n-1}$, где n — число вершин в графе G.

7 Вычислительный эксперимент

Основная цель эксперимента — восстановить дерево суперпозиции. Алгоритмы, используемые для восстановления, перечислены ниже.

DFS, BFS. Алгоритмы жадного дерева обхода в глубину и жадного дерева обхода в ширину. Обход ребер с наибольшим весом эквивалентен выбору наиболее вероятного пути. Алгоритм обхода останавливается, когда число ребер, исходящих из некоторой вершины, становится равным арности соответствующей функции.

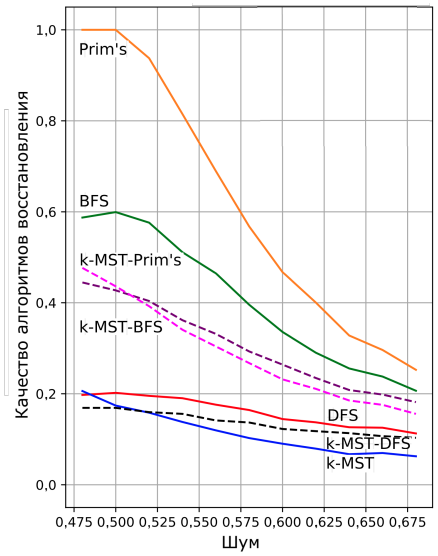

Рис. 2: Качество алгоритмов восстановления с базовыми функциями небольших арностей

Алгоритм Прима. Алгоритм восстанавливает минимальное остовное дерево для графа с дополнительными ограничениями на арность базовых функций. Эти ограничения задают минимальный вес ребра. После добавления вершины все листовые ребра этой вершины исключаются, чтобы сохранить направление дерева. Если число ребер, начинающихся в какой-либо вершине, превышает соответствующую арность, то остальные ребра исключаются из множества возможных ребер в этой вершине. Алгоритм не зависит от процедуры обхода. В случае небольшого шума в матрице смежности этот алгоритм способен восстановить дерево суперпозиции без ошибок.

Таблица 2: Качество алгоритмов реконструкции с равномерным шумом, близким к 0,5

	Шум				
Алгоритм	0,50	0,52	0,54	0,56	0,58
DFS	0,20	0,20	0,19	0,18	0,16
BFS	0,60	0,58	0,51	0,46	0,40
Прима	1,00	0,94	0,81	0,69	0,57
k-MST	0,17	0,16	0,14	0,12	0,10
k-MST-DFS	0,17	0,16	0,16	0,14	0,14
k-MST-BFS	0,43	0,40	0,36	0,33	0,29
k-MST-Прима	0,44	0,39	0,34	0,33	0,27

Алгоритмы на основе PCST. Матрица смежности M должна быть приведена к неориентированному виду. Использована квадратная матрица M' без последнего столбца. PCST принимает матрицу смежности $1 - \frac{1}{2}(M' + M'^{\mathsf{T}})$ с призовым значением

0,5 для каждой вершины. Призовое значение равно 0,5, поскольку при меньших значениях дерево будет обрезано: если шум равен 0,5, некоторые вершины могут быть обрезаны по ошибке. В случае больших призовых значений дерево PCST может содержать ненужные вершины. Дерево восстанавливается по одному из описанных алгоритмов. Результаты PCST можно использовать в качестве априорных для других подходов, $M' := \frac{1}{2}(M'_{\text{PCST}} + M')$, поэтому результаты PCST обновляются M'.

Процедура генерации данных имеет следующие допущения: арности функций генерируются биномиальным распределением, поэтому существует много функций с малой арностью, все базовые функции имеют только один вход. Любой случай с частичной реконструкцией считается ошибкой. Качество алгоритмов реконструкции

$$\frac{1}{K}\sum_{k=1}^{K}[R\big(\bar{N}(M_k)\big) = M_k],$$

где R — алгоритм реконструкции, а $\bar{N} = \big(N - \min(N)\big)/\big(\max(N) - \min(N)\big)$ — нормированная матрица шума. Матрица N генерируется как $N(M) = M + U(-\alpha, \alpha)$. Генератор случайных чисел возвращает матрицу того же вида, что и M, где каждый элемент является независимой переменной из равномерного распределения в сегменте $[-\alpha, \alpha]$.

Вот список из семи сравниваемых алгоритмов: DFS, BFS, алгоритм Прима, k-MST через PCST, k-MST + DFS, k-MST + BFS, k-MST + алгоритм Прима. На рис. 2 показана ошибка алгоритмов реконструкции с шумом, близким к порогу 0,5. Наилучшие результаты дает алгоритм Прима. Второе по точности решение основано на BFS. Табл. 2 соответствует рис. 2 и показывает качество реконструкции семи алгоритмов для значений граничного шума 0,50–0,58.

8 Заключение

Предлагаются и сравниваются алгоритмы восстановления суперпозиции для задачи символьной регрессии. Алгоритм Прима дает наиболее точные результаты и устойчив к небольшому шуму в данных. Предлагаемый алгоритм дает точные результаты, но он более подвержен шуму в матрице суперпозиции. Алгоритмы, основанные на BFS и DFS, не могут восстановить исходную суперпозицию с зашумленными матрицами суперпозиции. Алгоритм PCST с BFS, используемый для реконструкции матрицы суперпозиции, показывает приемлемые для практического использования результаты.

Список литературы

[1] *Koza J. R.* Genetic programming as a means for programming computers by natural selection // Statistics and Computing, 1994. Vol. 4. P. 87–112.

[2] *Searson D. P., Leahy D. E., Willis M. J.* GPTIPS: an open source genetic programming toolbox for multigene symbolic regression // Proceedings of the International multiconference of engineers and computer scientists, 2010. Vol. 1. P. 77–80.

[3] *Stanley K. O., Miikkulainen R.* Evolving neural networks through augmenting topologies // Evolutionary computation, 2002. Vol. 10. Iss. 2. P. 99–127.

[4] *Бочкарев А. М., Софронов И. Л., Стрижов В. В.* Порождение экспертно-интерпретируемых моделей для прогноза проницаемости горной породы // Системы и средства информатики, 2017. Том. 27. Вып. 3. С. 74–87.

[5] *Ravi R., Sundaram R., Marathe M. V., Rosenkrantz D. J., Ravi S. S.* Spanning trees — short or small // SIAM Journal on Discrete Mathematics, 1996. Vol. 9. Iss. 2. P. 178–200.

[6] *Chudak F. A., Roughgarden T., Williamson D. P.* Approximate k-MSTS and k-Steiner trees via the primal-dual method and Lagrangean relaxation // Mathematical Programming, 2004. Vol. 100. Iss. 2. P. 411–421.

[7] *Awerbuch B., Azar Y., Blum A., Vempala S.* New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen // SIAM Journal on computing, 1998. Vol. 28. Iss. 1. P. 254–262.

[8] *Aror S., Karakostas G.* A $2 + \varepsilon$ approximation algorithm for the k-MST problem // Mathematical Programming, 2006. Vol. 107. Iss. 3. P. 491–504.

[9] *Hegde C., Indyk P., Schmidt L.* A fast, adaptive variant of the Goemans-Williamson scheme for the prize-collecting steiner tree problem // Workshop of the 11th DIMACS Implementation Challenge, 2014. P. 1–32.

[10] *Ras C., Swanepoel K., Thomas D. A.* Approximate Euclidean Steiner trees // Journal of Optimization Theory and Applications, 2017. Vol. 172. Iss. 3. P. 845–873.

[11] *Goemans M. X., Williamson D. P.* A general approximation technique for constrained forest problems // SIAM Journal on Computing, 1995. Vol. 24. Iss. 2. P. 296–317.