Методы трансформации моделей в задачах нелинейной регрессии*

Р. А. Сологуб
roman.sologub@yahoo.com
Вычислительный центр РАН им. А. А. Дородницына, Россия, г. Москва, ул. Вавилова, 40

Решается проблема автоматического построения и упрощения нелинейных регрессионных моделей. Модели предназначены для описания результатов измерений и прогнозирования экспериментов, составляющих неотъемлемую часть естественно-научных исследований. Порождаемые модели предназначены для аппроксимации, анализа и прогнозирования результатов измерений. При порождении учитываются требования, предъявляемые экспертами-специалистами в предметной области к порождаемым моделям. Это дает возможность получения экспертно-интерпретируемых моделей, адекватно описывающих результат измерения.

Ключевые слова: *анализ данных; регрессионная модель; нелинейная регрессия; порождение моделей; построение суперпозиций*

DOI: 10.21469/22233792.1.14.06

1 Введение

Для создания адекватной модели измеряемых данных используются экспертно-заданные порождающие функции и набор правил порождения. Модель задается в виде суперпозиции порождающих функций. Правила порождения определяют допустимость суперпозиции и исключают порождение изоморфных моделей.

В работе предлагается развить существующие методы автоматического порождения моделей [1, 2]. Исследуются методы и алгоритмы упрощения моделей и их свойства. Анализируется проблема возникновения различных топологически, но при этом равных функционально моделей. Предлагаются новые методы поиска изоморфных суперпозиций, основанные на поиске изоморфных подграфов и подстановке подграфов по правилам.

Использование нелинейной регрессии для решения прикладных задач описывается в работах Дж. Себера [3, 4]. В них описывается построение и оценка параметров нелинейных моделей. Для оценки параметров моделей используется алгоритм Левенберга–Марквадта [5]. Критерием качества при этом, как и в случае обычной линейной регрессии, остается среднеквадратичная ошибка. В работах [6, 7] индуктивное порождение моделей строится с помощью метода группового учета аргументов. В линейной модели предлагается порождать новые признаки с помощью операции произведения. С помощью полиномов Колмогорова–Габора алгоритм целенаправленно порождает и перебирает модели-претенденты различной сложности согласно ряду критериев. В результате находится модель оптимальной структуры в виде одного уравнения или системы уравнений [7]. Для индуктивного порождения моделей в работах Дж. Козы [8, 9], связанных с генетическим программированием [10, 11], осуществляется переход от строковой записи моделей к префиксной записи, таким образом вводится построение модели в виде графа-дерева.

*Работа выполнена при финансовой поддержке РФФИ, проект № 14-07-31326.

Работа [12], продолжающая работы Дж. Козы, связана с аналитическим программированием — дальнейшим алгебраическим развитием методов генетического программирования. Авторы используют строковое представление и цепочки логических предикатов в качестве элементов модели. В процессе построения моделей отсекаются циклические, а также имеющие комплексные или бесконечные значения.

Построение прогностической модели в виде суперпозиции заданных функций, предложенное в работе [13] позволяет получать интерпретируемые модели, а предложенный метод штрафования суперпозиций за сложность порождает менее точные, но более простые суперпозиции. Метод преобразования и упрощения суперпозиций по правилам, рассмотренный в работе [13], позволяет разделить построенные суперпозиции на классы эквивалентности и выбрать из каждого класса наиболее простую (т. е. имеющую наименьшее число структурных элементов) суперпозицию, что также позволяет обосновать возможность экспертной интерпретации. Методы построения комбинаций прогностических моделей описаны в работах [9, 12].

Для упрощения структуры моделей используются методы теории трансформации графов, предложенные в работе [14]. Для трансформации деревьев выделяются некоторые элементарные графы-шаблоны, для которых строятся оболочки изоморфных им графов более сложной структуры. Для упрощения модели производится рекурсивный поиск подграфов, изоморфных графам-шаблонам, с их заменой на более простые подграфы. Задача упрощения моделей, представленных в виде графов, рассматривается в работе [15]. Авторы рассматривают два различных метода упрощения моделей. В первом анализируется структура моделей и выделяются элементы-подграфы, которые подходят под шаблоны упрощения (например, двойное отрицание). Альтернативным методом является вычисление значений элемента модели на исходной выборке. Если значения функции совпадают со значениями более простого шаблона, осуществляется замена элемента модели шаблоном.

Цель работы — исследование проблемы построения и упрощения нелинейных регрессионных моделей как суперпозиций заданных параметрических функций. Предлагается метод трансформации суперпозиций, представленных в виде категории на множестве направленных ациклических графов без самопересечений, соответствующих суперпозициям.

2 Постановка задачи

Пусть задана выборка $D = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$, $\mathbf{x} \in \mathbb{R}^m$. Требуется построить функцию регрессии $\varphi(\mathbf{x}, \mathbf{w}) \mapsto \mathbf{y}$. Из множества функций F требуется выбрать модель f — отображение из декартова произведения множества свободных переменных $\mathbf{x} \in \mathbb{R}^n$ и множества параметров $\mathbf{w} \in \mathbb{R}^m$ в $\mathbb{R}^1$. Сужение модели есть функция регрессии φ с заданными значениями $\mathbf{w} = \mathbf{w}_0$. Требуется оценить набор параметров $\mathbf{w}_0$, доставляющих минимум внешнему критерию качества модели — квадратичной ошибке:

$$S(\mathbf{w}|D, f) = ||f(\mathbf{x}, \mathbf{w}) - y|| \,.$$

Выражение $S(\mathbf{w}|D, f)$ означает значение функции ошибки S, которое зависит от набора параметров $\mathbf{w}$ при заданной выборке D и модели f. Такая модель называется оптимальной при условии, что ее сложность $C(f)$ не превышает заданную. Сложность определяется как количество элементов во всех поддеревьях, которые можно выделить из дерева, представляющего модель. Искомую модель f будем искать среди множества суперпозиций функций $g \in G$. При этом накладываются ограничения на структуру суперпозиции.

Определение 1. Допустимой называется суперпозиция, удовлетворяющая следующим требованиям.

1. Элементами суперпозици f могут являться только порождающие функции g_j и свободные перменные $\mathbf{x}$.
2. Количество аргументов элемента суперпозиции равно арности соответствующей ему функции g_j.
3. Порядок аргументов элемента суперпозиции соотвествует порядку аргументов соответствующей функции g_j.
4. Для элемента s_i, аргументом которого является элемент s_j, область определения соответствующей порождающей функции g_i содержит область значений порождающей функции аргумента g_j: $\mathrm{dom}(g_i) \supseteq \mathrm{cod}(g_j)$.

Порождается множество моделей $f \in F$ — допустимых суперпозиций, состоящих из функций $g_i \in G$. Требуется выбрать модель, доставляющую минимум $S(f|\mathbf{w}^*_{\mathrm{ML}}, \mathfrak{D})$ при условии, накладываемом на сложность $C(f) < C^*$. Различные методы определения сложности модели будут рассмотрены в следующем разделе.

Следует заметить, что выборка вместе с суперпозициями составляет категорию $\mathfrak{F}$, так как для данной конструкции выполняются все аксиомы теории категорий:

1. $\mathfrak{F}$-объектами данной категории являются множества независимых перменных x и зависимых переменных y.
2. $\mathfrak{F}$-стрелками в данной категории являются суперпозиции f_i.
3. Функции $\mathrm{dom}(f)$ и $\mathrm{cod}(f)$ для суперпозиции f определяются естественным образом как область определения и область значений соответствующей суперпозиции.
4. Если для пары суперпозиций $\langle f_1, f_2\rangle$ выполняется условие $\mathrm{cod}(f_1) = \mathrm{dom}(f_2)$, то суперпозиция f_2 имеет область определения $\mathrm{dom}(f_2) \in \mathbb{R}^1$. Суперпозиция, в которой вместо независимых переменных из f_2 будет использоваться суперпозиция f_1, будет допустимой, т. е. композиция существует и входит в множество $\mathfrak{F}$-стрелок. Ассоциативность следует из того факта, что замена в суперпозиции одного аргумента на другой является ассоциативной операцией. Вообще все множество $\mathfrak{F}$-стрелок состоит из элементов G и их композиций.
5. Наличие единицы обеспечивается обязательным существованием в G функции $\mathrm{id}(\mathbf{x})$. Для этой функции выполняется закон тождества по определению.

2.1 Описание структуры модели

Условимся считать, что каждой суперпозиции f сопоставлено дерево Γ_f, эквивалентное этой суперпозиции и строящееся следующим образом:

– в вершинах v_i дерева Γ_f находятся соответствующие порождающие функции g_j;
– число дочерних вершин у некоторой вершины v_i равно арности соответствующей ей функции g_j;
– порядок дочерних вершин вершины v_i соотвествует порядку аргументов соответствующей функции g_j;
– листьями дерева Γ_f являются свободные переменные x_i либо числовые параметры w_i.

Таким образом, вычисление значения выражения f в некоторой точке с данным вектором параметров $\mathbf{w} = \{w_1, w_2, \ldots, w_k\}$ эквивалентно подстановке соответствующих значений свободных переменных x_i и параметров w_i в дерево Γ_f, где x_i — элементы вектора свободных переменных $\mathbf{x}$.

Заметим важное свойство таких деревьев: каждое поддерево Γ'_f дерева Γ_f, корнем которого является вершина v_i, также соответствует некоторой суперпозиции, являющейся составляющей исходной суперпозиции f.

Предложим определение сложности суперпозиции, позволяющее штрафовать суперпозиции с большим числом вложенных функций. Введем понятие сложности вершины.

Определение 2. Сложность C суперпозиции f равна сложности дерева Г, соответствующего ей, и определяется как сумма количества элементов во всех поддеревьях дерева Г.

Таким образом штрафуется суперпозиция, содержащая большое число вложенных функций. Определение позволяет вычислять сложность, производя обход дерева снизу вверх обратно обходу дерева «в глубину» — сложность родительской вершины равна удвоенной сложности вершин потомков плюс единица. Сложность корня и будет сложностью всей суперпозиции, $C(1,1) = C(f)$.

3 Трансформация моделей

При порождении моделей в общем случае одному и тому же отображению соответствуют суперпозиции различной сложности, например одно и то же отображение соответствует моделям x и $\sqrt[3]{x^3}$. Также возможны случаи порождения деревьев, некоторые ветви которых не оказывают влияния на значение функции (например, умножаются на 0). Данная проблема оказывается важной для многих классов задач, например для построения логических функций или для задачи угадывания функции [16]. Для понимания, как упрощать подобные суперпозиции, следует ввести понятие эквивалентности моделей.

Определение 3. Модель f_2 с вектором параметров $\mathbf{w}_2$ называется обобщающей для модели f_1 с вектором параметров $\mathbf{w}_1$, если для любого вектора $\mathbf{w}_1$ найдется такой вектор $\mathbf{w}_2$, что для любого $\mathbf{x} \in D$ значения функций $f_1(\mathbf{w}_1, \mathbf{x})$ и $f_2(\mathbf{w}_2, \mathbf{x})$ равны:

$$\mathbf{x} \in D \Rightarrow f_1(\mathbf{w}_1, \mathbf{x}) = f_2(\mathbf{w}_2, \mathbf{x})\,.$$

Определение 4. Модели f_1 и f_2 с векторами параметров $\mathbf{w}_1$ и $\mathbf{w}_2$ называются эквивалентными, если каждая из них является обобщающей для другой.

Для построения оптимальной модели f ограниченной сложности $C(f) < C_0$ необходимо найти способ трансформации модели f большей структурной сложности в модель меньшей сложности f' с помощью специального алгоритма упрощения. Алгоритм упрощения модели $f(\mathbf{w}, \mathbf{x})$ минимизирует сложность суперпозиции, соответствующей ее дереву, при условии, что результирующая модель $f'(\mathbf{w}', \mathbf{x})$ является обобщающей моделью для исходной модели $f(\mathbf{w}, \mathbf{x})$. При проведении данной операции какие-либо вершины и ребра из дерева, соответствующего трансформируемой модели f, будут удалены и будут построены другие вершины и ребра вместо них. Обобщим алгоритм упрощения на орграфы любого вида, а не только на деревья. Далее для каждого графа подразумевается, что это орграф.

Определение 5. Подграф L, удаляемый из графа G в алгоритме упрощения, будет называться заменяемым подграфом.

Определение 6. Создаваемый подграф R, помещаемый в граф G в алгоритме упрощения, называется замещающим подграфом.

Существуют по меньшей мере два широко используемых метода упрощения моделей: «алгебраическое упрощение», являющееся частным случаем алгебраической трансформации графов, и «упрощение эквивалентным решением» [15].

3.1 Алгебраический подход к трансформации графов

Определение трансформации графа как замены одного подграфа на другой является интуитивно понятным, однако нестрогим. Для использования математического аппарата теории категория следует строго определить трансформацию графа.

Определение 7. Трансформацией $\mathfrak{f}$ на множестве графов Γ является пара гиперсхем H_1 и H_2, функция поиска $\mathfrak{m}$, ставящая в соответствие гиперсхеме H_1 подграф Γ, соответствующий этой гиперсхеме, и взаимно однозначное отображение f, ставящее в соответствие корню и листьям H_1 корень и листья H_2. При этом порождающие функции, соответствующие этим вершинам, должны совпадать.

Каждой трансформации, таким образом, может быть поставлена в соответствие обратная трансформация $\mathfrak{f}^{-1}$.

В рамках алгебраического подхода к трансформации графов следует ввести категорию трансформаций графов $\mathfrak{G}$, объектами которой являются графы Γ, а стрелками — трансформации графов $\mathfrak{f}$. Рассмотрим аксиомы категории.

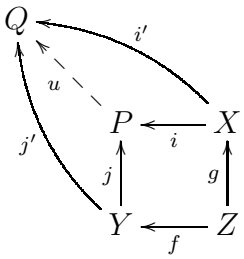
1. $\mathfrak{G}$-объектами в данной категории являются множества графов Γ.
2. $\mathfrak{G}$-стрелками в данной категории являются трансформации графов $\mathfrak{f}_i$.
3. Функции $\mathrm{dom}(\mathfrak{f})$ и $\mathrm{cod}(\mathfrak{f})$ для трансформаций графов определяются с помощью функции поиска $\mathfrak{m}$. $\mathrm{cod}(\mathfrak{f})$ может быть найден как $\mathrm{dom}(\mathfrak{f}^{-1})$.
4. Ассоциативность следует из наличия обратной функции.
5. Единицей является тривиальная трансформация с гиперсхемами $H_1 = H_2 = \#$.

Алгебраический подход к трансформации графов основывается на конструкции кодекартова квадрата морфизмов.

Определение 8. Кодекартов квадрат морфизмов $f : Z \to Y$ и $g : Z \to X$ — это объект P и два морфизма $i : X \to P$ и $j : Y \to P$, для которых следующая диаграмма коммутативна:

$$\begin{array}{ccc} P & \xleftarrow[i]{} & X \\ {\scriptstyle j}\uparrow & & \uparrow{\scriptstyle g} \\ Y & \xleftarrow[f]{} & Z \end{array} \qquad (1)$$

Кодекартов квадрат (P, i, j) является универсальным среди объектов, для которых диаграмма (1) коммутативна, т. е. для любой (Q, i', j'), такого что предыдущая диаграмма коммутирует, существует единственный морфизм $u : P \to Q$, делающий следующую диаграмму коммутативной:



(2)

Как и любой универсальный оператор, кодекартов квадрат определен с точностью до изоморфизма.

Определение 9. Кодекартов квадрат морфизмов $f : Z \to X$ и $g : Z \to Y$ — это копредел диаграммы $X \leftarrow Z \to Y$.

В контексте категории графов, используемой в данной работе, кодекартов квадрат является дизъюнктивной суммой множеств графов X и Y, при этом элементы с общим прообразом в множестве Z склеиваются, т. е. для каждого графа — элемента множества Z — образы его вершин и ребер относительно преобразований $i \cdot g$ и $j \cdot f$ будут совпадать. В рамках данной работы вместо термина «кодекартов квадрат» также будет использоваться

термин-синоним «склейка». Трансформация графа может строиться сразу как два кодекартовых квадрата. Данный подход называется двойной склейкой в противоположность к однократной склейке. Оба подхода описаны ниже. В процессе трансформации графов каждый граф Γ_1 — элемент множества X — является заменяемым и заменяющим подграфом, каждый граф Γ_2 — элемент множества Y — неизменной частью этого графа, а элементы Z — общей частью заменяемого и заменяющего подграфов. Естественным образом вводится операция соединения графов Γ_1 и Γ_2, результатом которой является объединение множеств, при этом соответствующие вершины и ребра накладываются друг на друга.

3.2 Трансформация двойной склейкой

Для рассмотрения трансформации графов необходимо ввести понятие правила, построенного в виде кодекартова квадрата морфизмов. Множества графов Λ и Φ являются в схеме кодекартова квадрата множеством X, множество граф Ψ — множеством Z, а множеству Y соответствует Δ. Множеству P для двух квадратов соответствуют начальный и конечный графы Γ и Ω.

Определение 10. Правило — это тройка $p = (\Lambda, \Psi, \Phi)$, где Λ и Φ являются заменяемым и замещающим подграфами и граф Ψ является общей частью подграфов Λ и Φ, т. е. их пересечением. Заменяемый, или начальный, подграф Λ называется условием применения правила; замещающий, или конечный, подграф Φ — итогом его применения. Подграф Ψ описывает часть графа, необходимую для применения правила, но неизменную в процессе применения. Множество $\Lambda \setminus \Psi$ является удаляемой частью графа, вместо нее создается множество $\Phi \setminus \Psi$.

Определение 11. Процедура поиска $\mathfrak{m}$ — отображение из Λ в Γ, ставящая в соответствие заменяемому графу эквивалентный ему подграф. При этом процедура $\mathfrak{m}$ сохраняет структуру графа Γ.

Определение 12. Трансформация графа — это пара, элементами которой являются правило p и процедура поиска $\mathfrak{m}$. Процедура трансформации графа Γ в граф Ω с помощью правила p и процедуры поиска $\mathfrak{m}$ будет также обозначаться как $\Gamma \overset{p,\mathfrak{m}}{\to} \Omega$.

Процедура трансформации графа правилом p и процедурой поиска $\mathfrak{m}$ состоит из двух шагов. На первом шаге все ребра и вершины, соответствующие множеству $\Lambda \setminus \Psi$, удаляются из графа Γ. Удаляемая часть может не являться графом, но оставшаяся структура $\Delta = \{\Gamma \setminus \mathfrak{m}(\Lambda)\} \bigcup \mathfrak{m}(\Psi)$ должна оставаться графом, т. е. в ней не должно быть подвешенных ребер. Таким образом, процедура поиска $\mathfrak{m}$ должна удовлетворять условию соединения графов, т. е. результатом соединения $\Lambda \setminus \Psi$ и Δ является граф Γ (см. диаграмму (3)). На втором шаге трансформации граф Δ соединяется с графом $\Phi \setminus \Psi$ для образования производного графа Ω (см. диаграмму (3)). Так как подграфы Λ и Φ могут иметь пересечение Ψ, подграф Ψ существует и в начальном графе Γ и не удаляется на первом шаге, т. е. существует и в промежуточном графе Δ. Для присоединения новых ребер и вершин к графу Δ используется граф Ψ. Таким образом определяются присоединенные вершины, с помощью которых граф Φ присоединяется к графу Δ. Для получения графа оптимальной структуры процедура одиночной трансформации графа может быть выполнена несколько раз.

Формально трансформация графа задается следующим образом. Пусть даны правило

$$p = (\Lambda \leftarrow \Psi \rightarrow \Phi)$$

и промежуточный граф Δ, который включает в себя Ψ, тогда исходный граф Γ трансформации $\Gamma \to \Omega$ с помощью правила p — это соединение Λ и Δ с помощью Ψ:

$$\Gamma = \Lambda +_{\Psi} \Delta\,,$$

а результирующий граф Ω определяется как соединение Φ и Δ с помощью Ψ:

$$\Omega = \Phi +_{\Psi} \Delta\,.$$

Более точно используются морфизмы

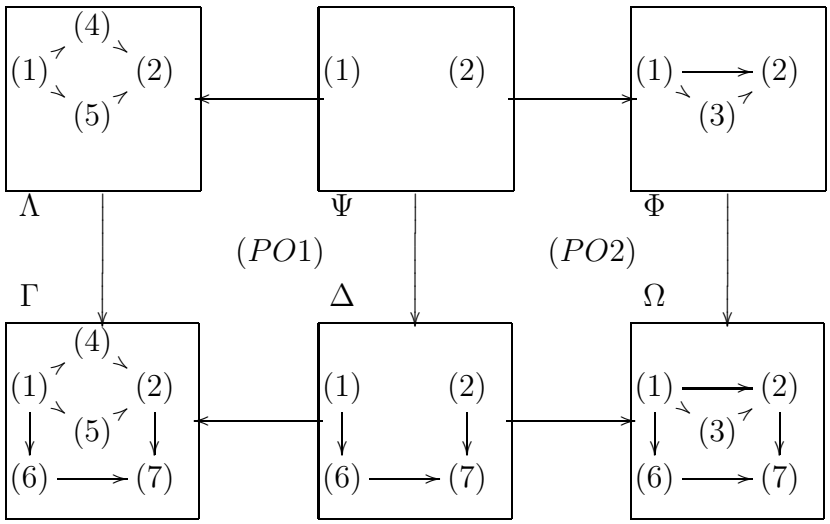
$$r : \Psi \to \Lambda\,; \qquad l : \Psi \to \Phi\,; \qquad k : \Psi \to \Delta$$

для того, чтобы показать, каким образом Ψ входит в Λ, Φ и Δ соответственно. Данный способ построения начального графа Γ и конечного графа Ω позволяет определить конструкции соединения $\Gamma = \Lambda +_{\Psi} \Delta$ и $\Omega = \Phi +_{\Psi} \Delta$ как конструкции склейки (см. диаграмму (3)). Таким образом, диаграмма (3) является двойным кодекартовым квадратом. Результирующий морфизм $\mathfrak{n} : \Phi \to \Omega$ называется ко-поиском трансформации $\Gamma \to \Omega$. Данная функция является функцией поиска в графе Ω подграфа, изоморфного заменяющему подграфу Φ. Коммутативная диаграмма для трансфомации графа строится следующим образом:

$$\begin{array}{ccccc} \Lambda & \xleftarrow{\;r\;} & \Psi & \xrightarrow{\;l\;} & \Phi \\ \downarrow{\scriptstyle \mathfrak{m}} & (1) & \downarrow{\scriptstyle k} & (2) & \downarrow{\scriptstyle \mathfrak{n}} \\ \Gamma & \xleftarrow{\;g\;} & \Delta & \xrightarrow{\;h\;} & \Omega \end{array} \tag{3}$$

Для применения правила p с процедурой поиска $\mathfrak{m}$ подграфа Λ в графе Γ, при заданном морфизме $\mathfrak{m} : \Lambda \to \Gamma$, как показано на коммутативной диаграмме (3), в первую очередь необходимо построить промежуточный граф Δ такой, что соединение $\Lambda +_{\Psi} \Delta$ даст результатом граф Γ. На следующем шаге строим соединение $\Phi +_{\Psi} \Delta$ графов Φ и Δ с помощью графа Ψ, получая граф Ω, и, таким образом, получаем процедуру двойной склейки $\Gamma \to \Omega$ с помощью правила p и процедуры поиска $\mathfrak{m}$. Для первого шага необходимо выполнение условия соединения графов, что позволяет нам построить Δ из условия $\Gamma = \Lambda +_{\Psi} \Delta$. Для процедуры $\mathfrak{m}$ условие соединения означает, что все подвешенные вершины Λ, т. е. вершины $v \in \Lambda$, такие, что $\mathfrak{m}(v)$ является начальной или конечной вершиной некоторого ребра e, принадлежащего $\Gamma \setminus \Lambda$, должны быть в Ψ.

Рассмотрим пример двойной склейки:



(4)

Данная диаграмма соответствует общей схеме (3). Следует заметить, что в диаграмме (3) граф Γ является соединением графов Λ и Δ с помощью Ψ, причем обозначения вершин показывают, как вершины размечаются при применении морфизмов.

Рассмотрим условие корректности построения структуры графа Δ. Разметка ребер может быть единственным образом выведена из разметки вершин. Условие соединения вершин выполнено на диаграмме (4), потому что подвешенные вершины (1) и (2), принадлежащие Λ, также являются соединительными вершинами. Таким образом, не остается подвешенных ребер, выходящих из вершин (1) и (2). При этом граф Ω является соединением графов Φ и Δ вместе с Ψ, что приводит к трансформации $\Gamma \to \Omega$ с помощью правила p. Фактически диаграммы (3) и (4) являются кодекартовыми квадратами в категории графов, состоящей из графов и морфизмов на них.

Сформулируем точное условие соединения графов при трансформации графа. Для этого вводим следующие определения.

Определение 13. Точки соединения — вершины и ребра в Λ, которые не удаляются при применении правила p.

Определение 14. Точки обнаружения — вершины и ребра в Λ, образы которых относительно $\mathfrak{m}$ имеют более одного прообраза.

Определение 15. Подвешенные вершины — вершины в Λ, образы которых относительно $\mathfrak{m}$ в Γ имеют входящие или выходящие ребра, не содержащиеся в Λ.

В данных определениях условие соединения графа выглядит следующим образом.

Теорема 1. **[14]** Пусть даны правило $p = (\Lambda \leftarrow \Psi \rightarrow \Phi)$, граф Γ и процедура поиска $\mathfrak{m} : \Lambda \to \Gamma$. Вершины графов обозначаются буквой V, ребра — E. Тогда правило p с процедурой поиска $\mathfrak{m}$ удовлетворяет условию соединения, если все точки обнаружения и подвешенные вершины также являются точками соединения.

Докажем данную теорему от противного. Пусть существует подвешенная вершина v_0, не являющаяся точкой соединения. Данная вершина удаляется из Γ при применении правила p. Однако в Γ существуют ребра, не содержащиеся в Λ и присоединенные к v_0. Таким образом, полученный граф будет недопустимым, потому что у некоторых ребер не будет

начала или конца. Точки обнаружения являются точками соединения, так как иначе правило будет внутренне противоречивым.□

Ограничения, накладываемые естестенным образом на трансформации двойной склейкой, не позволяют удобно производить многие операции с графами, используемые на практике. Так, операция замены вершины поддерева v_i не может быть описана в виде заменяемого и замещающего графов, состоящих из одной вершины, так как если в заменяемом графе всего одна вершина v_i, эта вершина не будет являться подвешенной, только если весь граф состоит из одной вершины. Таким образом, для применения трансформаций предлагается метод, сопоставляющий неудовлетворяющей условиям трансформации набор допустимых трансформаций.

Теорема 2. Любой трансформации $t = (\Lambda_t, \Psi_t, \Phi_t)$ графа соответствует набор правил $p_t = (\Lambda_{p_t}, \Psi_{p_t}, \Phi_{p_t})$, удовлетворяющих условию соединения, такой, что любое применение трансформации t аналогично применению одного из правил p_t.

Данная теорема доказывается конструктивно — рассматриваются все возможные наборы количеств ребер, которые могут иметь точки соединения v_c, и для каждого набора создаются заменяемый и замещающий подграфы, в который добавляются вершины типа # на концах всех ребер, выходящих из v_c и не содержавшихся ранее в заменяемом подграфе Λ.□

Морфизмы $\Psi \to \Lambda$ и $\Psi \to \Phi$ в произведениях могут быть ограничены как инъективные морфизмы — каждому образу в Λ и Φ соответствует только один проообраз из Ψ. Тем не менее возможны неинъективные варианты процедур поиска $\mathfrak{m} : \Lambda \to \Gamma$ и ко-поиска $\mathfrak{n} : \Phi \to$ $\to \Omega$. Это может быть особенно важным, когда рассматривается параллельное применение правил:

$$p_1 \bigoplus p_2 : \Lambda_1 \bigoplus \Lambda_2 \leftarrow \Psi_1 \bigoplus \Psi_2 \to \Phi_1 \bigoplus \Phi_2 \,,$$

где $\bigoplus$ означает дизъюнктивное объединение. Даже для инъективных вариантов $\mathfrak{m}_1 : \Lambda_1 \to$ $\to \Gamma$ с помощью p_1 и $\mathfrak{m}_2 : \Lambda_2 \to \Gamma$ с помощью p_2, итоговая операция $\mathfrak{m} : \Lambda_1 + \Lambda_2 \to \Gamma$ не является инъективной, если образы процедур поиска $\mathfrak{m}_1(\Lambda_1)$ и $\mathfrak{m}_2(\Lambda_2)$ имеют непустое пересечение в Γ.

Теорема 3. Существует набор трансформаций $(p_1, \mathfrak{m}_1)$ и $(p_2, \mathfrak{m}_2)$, такой, что их параллельное применение имеет неинъективную функцию поиска $\mathfrak{m} = \mathfrak{m}_1 \bigoplus \mathfrak{m}_2$.

Построим пример таких трансформаций. Пусть трансформация преобразует дерево Γ_0, соответствующее функции $f_0 = (x+1) * (x - 1 + x^2 - x + 1)$, и есть два правила

$$p_1 = \{(x+1)(x-1), x^2 - 1\}\,; \quad p_2 = \{(x+1)(x^2 - x + 1), x^3 + 1\}\,.$$

В обоих этих правилах подграф Ψ является пустым. Обе процедуры поиска $\mathfrak{m}_1$ и $\mathfrak{m}_2$ будут находить часть суперпозиции $(x+1)$. Из этого следует, что при объединении Λ_1 и Λ_2 у одного образа из Γ будет более одного прообраза, т. е. объединенное правило p_{12} дважды найдет в графе Γ_0 подграф, соотвествующий суперпозиции $(x+1)\cdot$.□

Для рассмотрения случаев применения нескольких трансформаций необходимо определить условие, при котором трансформации могут применяться последовательно и параллельно. Введем понятия параллельно и последовательно независимых трансформаций.

Определение 16. Две трансформации графов $\Gamma \overset{p_1,m_1}{\to} \Omega_1$ и $\Gamma \overset{p_2,m_2}{\to} \Omega_2$ являются параллельно независимыми, если все вершины и ребра, попадающие в образ обоих морфизмов поиска, являются соединительными:

$$\mathfrak{m}_1(\Lambda_1) \bigcap \mathfrak{m}_2(\Lambda_2) \subseteq \mathfrak{m}_1(l_1(\Psi_1)) \bigcap \mathfrak{m}_2(l_1(\Psi_2)).$$

Две трансформации графов $\Gamma \overset{p_1,m_1}{\to} \Omega_1$ и $\Omega_1 \overset{p_2,m_2}{\to} \Omega_2$ являются последовательно независимыми, если все вершины и ребра, попадающие в пересечение морфизмов $\mathfrak{n}_1$ и m_2, являются соединительными:

$$\mathfrak{n}_1(\Phi_1) \bigcap \mathfrak{m}_2(\Lambda_2) \subseteq \mathfrak{n}_1(r_1(\Psi_1)) \bigcap \mathfrak{m}_2(l_2(\Psi_2)).$$

Следует заметить, что для графов-деревьев возникает простой достаточный критерий пареллельной и последовательной независимости трансформаций, если их замещаемые графы Λ являются односвязными.

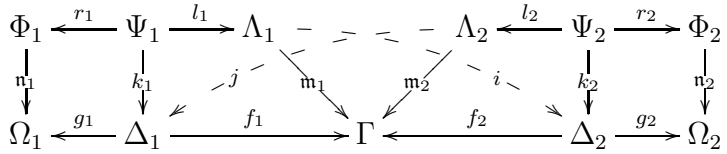
Теорема 4. Две трансформации графов-деревьев $\Gamma \overset{p_1,m_1}{\to} \Omega_1$ и $\Gamma \overset{p_2,m_2}{\to} \Omega_2$ являются параллельно и последовательно независимыми, если образы корней v_1 и v_2 деревьев $\mathfrak{m}_1(\Lambda_1)$ и $\mathfrak{m}_2(\Lambda_1)$ не принадлежат друг другу:

$$v_1 \notin \mathfrak{m}_2(\Lambda_2)\,; \quad v_2 \notin \mathfrak{m}_1(\Lambda_1)\,. \tag{5}$$

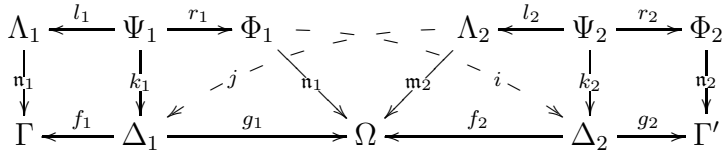
Данная теорема простым образом доказывается от противного. Пусть условие (5) выполняется и существует вершина v_0, принадлежащая пересечению множеств $\mathfrak{m}_1(\Lambda_1)$ и $\mathfrak{m}_2(\Lambda_1)$. Тогда в графе будет цикл, проходящий через вершины v_1, v_0, v_2 и корень дерева. Но в дереве не может быть циклов.□

Определение независимости оказывается неудобным для применения, так как оно слабо формализовано. Определим необходимое и достаточное условие, при котором графы являются параллельно или последовательно независимыми через существование соответствующих морфизмов.

Теорема 5. **[14]** Две трансформации графов $\Gamma \overset{p_1,m_1}{\to} \Omega_1$ и $\Gamma \overset{p_2,m_2}{\to} \Omega_2$ являются параллельно независимыми, если существуют морфизмы $i : \Lambda_1 \to \Delta_2$ и $j : \Lambda_2 \to \Delta_1$, такие, что $f_2 \circ i = \mathfrak{m}_1$ и $f_1 \circ j = \mathfrak{m}_2$:



Теорема 6. Две трансформации графов $\Gamma \overset{p_1,m_1}{\to} \Omega$ и $\Omega \overset{p_2,m_2}{\to} \Gamma'$ являются последовательно независимыми, если существуют морфизмы $i : \Phi_1 \to \Delta_2$ и $j : \Lambda_2 \to \Delta_1$, такие, что $f_2 \circ i = \mathfrak{n}_1$ и $g_1 \circ j = \mathfrak{m}_2$:



Доказательство. Рассмотрим необходимость и достаточность критерия для параллельной независимости. Для последовательной независимости доказательство будет строиться аналогичным образом. Вершина $v \in \Lambda_1$ или принадлежит множеству $\mathfrak{m}_2(\Lambda_2)$, или лежит вне его. Рассмотрим оба случая.

1. Множество $\mathfrak{m}_1(v) \notin \mathfrak{m}_2(\Lambda_2)$. Все вершины графа Γ являются образами при применении отображений $\mathfrak{m}_2$ или f_2. Отсюда $\mathfrak{m}_1(v) \in f_2(\Delta_2)$.
2. Множество $\mathfrak{m}_1(v) \in \mathfrak{m}_2(\Lambda_2)$. Тогда $\mathfrak{m}_1(v) \in \mathfrak{m}_1(\Lambda_1) \cap \mathfrak{m}_2(\Lambda_2) \subseteq \mathfrak{m}_1(l_1(\Psi_1)) \cap \mathfrak{m}_2(l_2(\Psi_2))$. При этом из коммутативной диаграммы следует, что $\mathfrak{m}_2(l_2(\Psi_2)) = f_2(k_2(\Psi_2))$. Отсюда $\mathfrak{m}_1(v) \in f_2(\Delta_2)$.

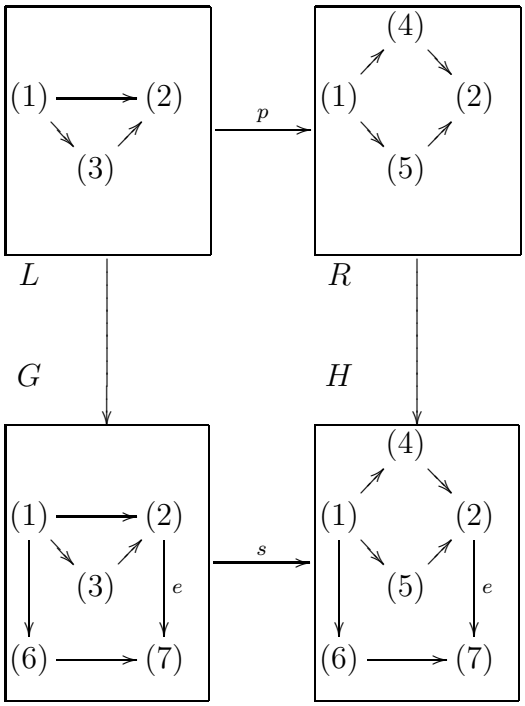
В обоих случаях оказывается, что $\mathfrak{m}_1(x) \in f_2(\Delta_2)$, так что инъективность f_2 позволяет нам определить $i(x) = f_2^{-1} \circ \mathfrak{m}_1(x)$. Аналогично, j определяется из условия $f_1 \circ j = \mathfrak{m}_2$.

При данных i, j с $f_2 \circ i = m_1$ и $f_1 \circ j = m_2$ пусть $y \in \mathfrak{m}_1(\Lambda_1) \bigcap \mathfrak{m}_2(\Lambda_2)$. Тогда $y \in \in \mathfrak{m}_1(L_1) \bigcap f_1(j(\Lambda_2))$. Из условия кодекартова квадрата следует, что существует $z_1 \in \Psi_1$, такое, что $y = \mathfrak{m}_1(l_1(z_1)) = f_1(k_1(z_1))$. Значит, $y \in \mathfrak{m}_1(l_1(\Psi_1))$, аналогично $y \in \mathfrak{m}_2(l_2(\Psi_2))$, откуда следует условие независимости $\mathfrak{m}_1(\Lambda_1) \bigcap \mathfrak{m}_2(\Lambda_2) \subseteq \mathfrak{m}_1(l_1(\Psi_1)) \bigcap \mathfrak{m}_2(l_1(\Psi_2))$.□

С использованием данных критериев можно определить, как связаны друг с другом независимые параллельно и последовательно трансформации. Данная теорема является частным случаем теоремы Черча–Россера.

3.3 Трансформация одиночной склейкой

Как было отмечено, конструкции соединения в алгебраическом подходе являются кодекартовыми квадратами в смысле морфизмов категории графов. С другой стороны, правило $p = (\Lambda \leftarrow \Psi \rightarrow \Phi)$ может быть также рассмотрено как частичный морфизм графов $p : \Lambda \rightarrow \Phi$, доменом которого является множество $\mathrm{dom}(p) = \Psi$. Более того, диаграмма $(\Gamma \leftarrow \Delta \rightarrow \Omega)$ может быть рассмотрена как частичный морфизм графов $s : \Gamma \rightarrow \Omega$ с доменом $\mathrm{dom}(s) = \Delta$. Таким образом, получается следующая диаграмма:



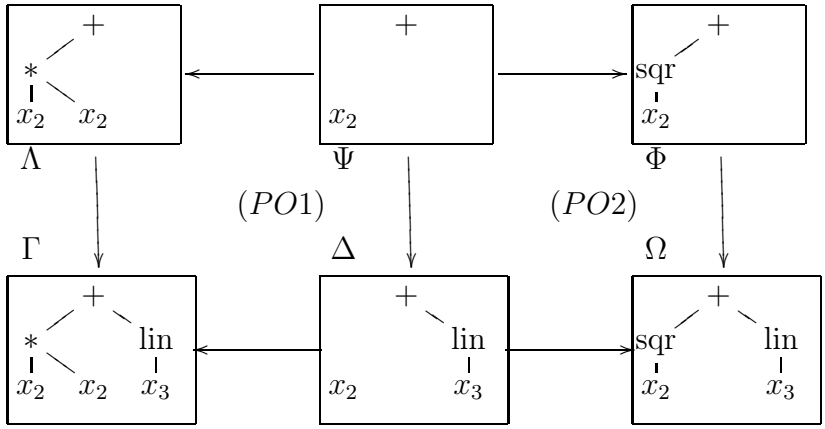
(6)

На данной диаграмме горизонтальные морфизмы являются частичными, а вертикальные — полными морфизмами графов. По сути, диаграмма (6) является кодекартовым квадратом в расширенной категории графов, которая состоит из графов и частичных морфизмах на графах и показывает, что трансформации графов могут быть выражены как одиночные кодекартовы квадраты в расширенной категории графов. Данный подход развивался Раулем [17] и был полностью разработан Лёве [18], итогом их работы является подход однократного вытеснения.

С точки зрения прикладного использования подход с одиночным вытеснением отличается от подхода с двойным вытеснением в одном главном отношении, которое касается удаления вспомогательных элементов графа в процессе трансформации графа. Процедура поиска $m : \Lambda \rightarrow \Gamma$ не удовлетворяет условию соединения по отношению к правилу $p = \\ = (\Lambda \leftarrow \Psi \rightarrow \Phi)$, поэтому данное правило не применимо в подходе с двойным вытеснением.

Но оно может быть применимо в подходе с однократным вытеснением, которое позволяет появляться подвешенным ребрам после удаления подграфа $\Lambda \setminus \Psi$ из Γ. Следует заметить, что подвешенные ребра из Γ также удаляются для создания допустимого графа Ω.

Если на диаграмме (4) вершина (2) была бы удалена из Ψ, то конструкция соединения не удовлетворяла бы подходу с двойным вытеснением. В подходе с однократным вытеснением это значило бы, что вершина (2) не находится в домене p, что ведет к подвешенному ребру в Γ после удаления $\Lambda \setminus \mathrm{dom}(p)$ на диаграмме:



В результате ребро e удялется из Ω.

Более подобное описание и сравнение данных подходов разобрано в [19].

3.4 Прикладная задача упрощения суперпозиций

При последовательном порождении моделей зачастую оказывается так, что некоторые части модели становятся рудиментарными. Упрощение Соула [20] является вариантом алгебраического упрощения, в котором объектами упрощения являются элементы моделей, параметры которых не влияют на значение функции. Область применения подобных методов ограничена [20], однако они показывают хороший результат на некоторых задачах, например при обнаружении функции. В данном типе задачи восстановления регрессии дисперсия случайной ошибки равна нулю и выборка генерируется в соответствии с какой-либо эталонной функцией f_0, которая должна быть обнаружена алгоритмом.

Упрощение эквивалентным решением заключается в сравнении значений моделей, а не структур. Эквивалентность моделей проверяется не по структуре деревьев, соответствующих им, а численно. В таком случае два выражения, дающие равные значения на области определения независимых переменных модели, считаются равными.

Определение 17. Шаблон θ — гиперсхема, обладающая наименьшей сложностью среди всех гиперсхем, таких, что при их взаимном замещении получаемые модели оказываются эквивалентными. Сложность гиперсхемы определяется как сложность суперпозиции при замещении всех символов $\{=\}$ и $\{\#\}$, означающих соответственно произвольную независимую переменную и произвольное поддерево, на константы.

Экспертно выбирается некоторый набор шаблонов Θ. Процедура упрощения состоит из двух шагов.

1. Все поддеревья Γ_j в выбранном дереве Γ проверяются на эквивалентность шаблонам из Θ согласно заданным правилам.
2. Если какое-либо поддерево Γ_j в дереве эквивалентно дереву из Θ, данное поддерево заменяется соответствующим элементом из Θ.

Процедура повторяется до тех пор, пока после вышеперечисленных итераций дерево Г не останется неизменным. При наличии в множестве порождающих функций коммутативных функций вводится алфавитное упорядочение для ветвей, выходящих из вершины γ_i дерева Г, соответствующей коммутативной порождающей функции g_i.

Эквивалентное упрощение является альтернативой алгебраическому упрощению, позволяя упрощать некоторые модели за меньшее количество операций.

Рассмотрим сложность алгоритма, упрощающего поддерево высоты l с вершинами арности не более m. Количество вершин в таком дереве ограничивается сверху как m^l. Рассмотрим дерево с максимальным количеством вершин — для такого дерева все вершины, кроме листьев, будут иметь арность m. Для сравнения всех возможных поддеревьев с шаблонами из Θ необходимо рассмотреть поддеревья любой высоты с корнем в каждой из вершин дерева. Подсчитаем количество поддеревьев всех возможных высот в таком дереве. Обозначим высоту данного дерева $l = \log_m k + 1$. Тогда для вершины, находящейся на расстоянии x от корня, количество поддеревьев с корнем в этой вершине составляет не менее чем $l - x$. Тогда искомое количество поддеревьев:

$$\sum_{x=0}^{l-1}(l-x)m^x = \frac{m(m^l-1)-lm+l}{(m-1)^2}\,.$$

Данное выражение пропорционально m^l, т. е. количеству элементов в дереве, все вершины которого (кроме листьев) имеют максимальное число потомков. Сложность алгоритма, упрощающего дерево, состоящее из k вершин, оказывается порядка не менее чем k. В случае если алгоритм проверки правил эквивалентности имеет значительную сложность, подсчет значений оценок зависимых переменных $\hat{y}$ на множестве независимых переменных $x \in D$ и сравнение этих значений с получаемыми при использовании шаблонов Θ имеет значительно меньшую сложность. Данный метод может применяться в случае, если независимые и зависимые переменные принимают ограниченное число значений. Для такого поддерева, вне зависимости от количества элементов k в нем, область определения соответствующей функции содержит 2^t точек, где t — количество независимых переменных, являющихся листьями данного поддерева. При небольших t число 2^t не превосходит k, и в таком случае алгоритм сравнения по значениям оказывается менее сложным, чем алгоритм сравнения структур поддеревьев с шаблонами.

Важным частным случаем использования алгоритма упрощения по значениям является случай равенства функций на области определения независимых переменных при необязательном равенстве вне этой области. Для решения прикладной задачи функции, дающие равные значения на области определения, будут равны, и подобная замена будет корректна.

4 Заключение

В работе предложены методы направленного порождения, модификации и упрощения нелинейных регрессионных моделей. Описаны условия существования решений, получаемых в результате порождения, доказаны необходимые теоремы. Разработан метод последовательного направленного порождения суперпозиций, введено понятие изоморфных суперпозиций, исследованы свойства порождаемых суперпозиций. Предлагаемые в работе методы упрощения моделей предназначены непосредственно для применения на практике. Создана базовая библиотека правил порождения экспертно-интерпретируемых моделей.

Литература

[1] *Стрижов В. В., Сологуб Р. А.* Индуктивное порождение поверхности волатильности опционных торгов // Вычислительные технологии, 2009. № 5. С. 102–113.

[2] *Сологуб Р. А.* Алгоритмы порождения нелинейных регрессионных моделей // Информационные технологии, 2013. № 5. С. 8–12.

[3] *Seber G., Wild C. J.* Nonlinear regression. — Wiley ser. in probability and statistics. — John Wiley & Sons, 2005. 2907 p.

[4] *Seber G.* The collected works of George A. F. Seber. — Wiley ser. in probability and statistics. — Wiley, 2009. 792 p.

[5] *Levenberg K.* A method for the solution of certain non-linear problems in least squares // Quart. J. Appl. Math., 1944. Vol. II. No. 2. P. 164–168.

[6] *Ивахненко А. Г.* Индуктивный метод самоорганизации моделей сложных систем. — Киев: Наукова думка, 1982. 296 с.

[7] *Madala H. R., Ivakhnenko A. G.* Inductive learning algorithms for complex systems modeling. — CRC Press, 1994. 384 p.

[8] *Koza J. R.* Genetic programming: On the programming of computers by means of natural selection. — Complex adaptive systems ser. — 1st ed. — Cambridge, MA – London: The MIT Press, 1992. 840 p.

[9] *Koza J. R., Keane M. A., Streeter M. J., et al.* Genetic programming IV: Routine human-competitive machine intelligence. — Norwell, MA, USA: Kluwer Academic Publs., 2003. 590 p.

[10] *Banzhaf W., Francone F. D., Keller R. E., Nordin P.* Genetic programming — an introduction: On the automatic evolution of computer programs and its applications. — San Francisco, CA, USA: Morgan Kaufmann Publs. Inc., 1998. 490 p.

[11] *Michell M.* An introduction to genetic algorithms. — Cambridge, MA – London: The MIT Press, 1998. 164 p.

[12] *Kominkova Oplatkova Z., Senkerik R., Zelinka I., Pluhacek M.* Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic systems // Comput. Math. Appl., 2013. Vol. 66. No. 2. P. 177–189.

[13] *Рудой Г. И., Стрижов В. В.* Алгоритмы индуктивного порождения суперпозиций для аппроксимации измеряемых данных // Информатика и её применения, 2013. Т. 7. Вып. 1. С. 17–26.

[14] *Ehrig H., Ehrig K., Prange U., Taentzer G.* Fundamentals of algebraic graph transformation. — Berlin: Springer, 2006. 390 p.

[15] *Naoki M., McKay B., Xuan N., et al.* A new method for simplifying algebraic expressions in genetic programming called equivalent decision simplification // Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living / Eds. S. Omatu, M. P. Pocha, J. Bravo, *et al.* — Lecture notes in computer science ser. — Salamanca, Spain: Springer, 2009. Vol. 5518. P. 171–178. doi: `http://dx.doi.org/10.1007/978-3-642-02481-8_24`

[16] *D'haeseleer P.* Context preserving crossover in genetic programming // 1st IEEE Conference on Computational Intelligence Proceedings: Evolutionary Computation, 1994. Vol. 1. P. 256–261.

[17] *Raoult J. C.* On graph rewritings // Theor. Comput. Sci., 1984. Vol. 32. No. 1. P. 1–24. doi: `http://dx.doi.org/10.1016/0304-3975(84)90021-5`

[18] *Löwe M., Ehrig H.* Algebraic approach to graph transformation based on single pushout derivations // Graph-theoretic concepts in computer science / Ed. R. Möhring. — Lecture notes in computer science ser. — Berlin–Heidelberg: Springer, 1991. Vol. 484. P. 338–353.

[19] Handbook of graph grammars and computing by graph transformation. Vol. 3: Concurrency, parallelism, and distribution / Eds. H. Ehrig, H.-J. Kreowski, U. Montanari, G. Rozenberg. — River Edge, NJ, USA: World Scientific Publishing Co., Inc., 1999. 472 p. doi: http://dx.doi.org/10.1142/9789812814951

[20] *Soule T., Heckendorn R. B.* An analysis of the causes of code growth in genetic programming // Genet. Program. Evol. M., 2002. Vol. 3. No. 3. P. 283–309. doi: http://dx.doi.org/10.1023/A:1020115409250

Поступила в редакцию 08.12.15

Methods of the nonlinear regression model transformation*

R. A. Sologub

roman.sologub@yahoo.com

Dorodnicyn Computing Centre of the Russian Academy of Sciences, 40 Vavilova st., Moscow, Russia

The problem of the nonlinear regression models automatic construction and simplification has been addressed. The models describe the results of measurements and forecasting experiments. The generated models are designed for the approximation, analysis, and forecasting of the experimental results. To generate the models, the expert requirements in the subject field have been considered. This approach allows to get the interpretable models, adequately describing the given measurements. The goal of the paper is to investigate the problem of generation and simplification of the nonlinear regression models. The models are supposed to be the superpositions of the given parametric functions. A method of the function superposition transformation has been suggested. The superpositions category defined over the set of directed acyclic graphs corresponding to the superpositions have been considered. The isomorphic superpositions notion have been introduced and a method of their detection has been developed. An algorithm of finding the isomorphic subgraphs corresponding to the generated superpositions has been developed.

Keywords: *data analysis; regression model; nonlinear regression; model generation; superposition construction*

DOI: 10.21469/22233792.1.14.06

References

[1] Strijov, V. V., and R. A. Sologub. 2009. Inductive generation of the regression models for option volatility. *Russ. J. Computational Technologies* 5:102–113. (In Russian.)

[2] Sologub, R. A. 2013. Algorithms for the nonlinear regression models generation. *Russ. J. Information Technologies* 5:8–12. (In Russian.)

[3] Seber, G., and C. J. Wild. 2005. *Nonlinear regression*. Wiley ser. in probability and statistics. John Wiley & Sons. 2907 p.

[4] Seber, G. 2009. *The collected works of George A. F. Seber*. Wiley ser. in probability and statistics. Wiley. 792 p.

[5] Levenberg, K. 1944. A method for the solution of certain non-linear problems in least squares. *Quart. J. Appl. Math.* II(2):164–168.

*This work was done under financial support of the Russian Foundation for Basic Research (grant 14-07-31326)

[6] Ivakhnenko, A. G. 1982. *Inductive method of self-organized complex models system*. Kiev: Naukova Dumka. 296 p. (In Russian.)

[7] Madala, H. R., and A. G. Ivakhnenko. 1994. *Inductive learning algorithms for complex systems modeling*. CRC Press. 384 p.

[8] Koza, J. R. 1992. *Genetic programming: On the programming of computers by means of natural selection*. 1st ed. Complex adaptive systems ser. Cambridge, MA – London: The MIT Press. 840 p.

[9] Koza, J. R., M. A. Keane, M. J. Streeter, *et al.* 2003. *Genetic programming IV: Routine human-competitive machine intelligence*. Norwell, MA: Kluwer Academic Publs. 590 p.

[10] Banzhaf, W., F. D. Francone, R. E. Keller, and P. Nordin. 1998. *Genetic programming — an introduction: On the automatic evolution of computer programs and its applications*. San Francisco, CA: Morgan Kaufmann Publs. Inc. 490 p.

[11] Michell, M. 1998. *An introduction to genetic algorithms*. Cambridge, MA – London: The MIT Press. 164 p.

[12] Kominkova Oplatkova, Z., R. Senkerik, I. Zelinka, and M. Pluhacek. 2013. Analytic programming in the task of evolutionary synthesis of a controller for high order oscillations stabilization of discrete chaotic systems. *Comput. Math. Appl.* 66(2):177–189.

[13] Rudoy, G. I., and V. V. Strijov. 2013. Algorithms for inductive generation of superpositions for approximation of experimental data. *Informatika i ee Primeneniya — Inform. Appl.* 7(1):17–26.

[14] Ehrig, H., K. Ehrig, U. Prange, and G. Taentzer. 2006. *Fundamentals of algebraic graph transformation*. Berlin: Springer. 390 p.

[15] Naoki, M., B. McKay, N. Xuan, *et al.* 2009. A new method for simplifying algebraic expressions in genetic programmingc alled equivalent decision simplification. *Distributed computing, artificial intelligence, bioinformatics, soft computing, and ambient assisted living*. Eds. S. Omatu, M. P. Rocha, J. Bravo, *et al.* Lecture notes in computer science ser. Salamanca, Spain: Springer. 5518:171–178. doi: http://dx.doi.org/10.1007/978-3-642-02481-8_24

[16] D'haeseleer, P. 1994. Context preserving crossover in genetic programming. *1st IEEE Conference on Computational Intelligence Proceedings: Evolutionary Computation*. 1:256–261.

[17] Raoult, J. C. 1984. On graph rewritings. *Theor. Comput. Sci.* 32(1):1–24. doi: http://dx.doi.org/10.1016/0304-3975(84)90021-5

[18] Löwe, M., and H. Ehrig. 1991. *Algebraic approach to graph transformation based on single pushout derivations*. Ed. R. Möhring. Lecture notes in computer science ser. Berlin–Heidelberg: Springer. 484:338–353.

[19] Ehrig, H., H.-J. Kreowski, U. Montanari, and G. Rozenberg, eds. 1999. *Handbook of graph grammars and computing by graph transformation. Vol. 3: Concurrency, parallelism, and distribution*. River Edge, NJ: World Scientific Publishing Co., Inc. 472 p. doi: http://dx.doi.org/10.1142/9789812814951

[20] Soule, T., and R. B. Heckendorn. 2002. An analysis of the causes of code growth in genetic programming. *Genet. Program. Evol. M.* 3(3):283–309. doi: http://dx.doi.org/10.1023/A:1020115409250

Received December 8, 2015