Министерство образования и науки Российской Федерации
Московский физико-технический институт (государственный университет)
Факультет управления и прикладной математики
Вычислительный центр им. А. А. Дородницына РАН
Кафедра «Интеллектуальные системы»

Рудой Георгий Игоревич

Алгоритмы индуктивного порождения и упрощения и критерии выбора оптимальной существенно нелинейной регрессионной модели для аппроксимации измеряемых данных

010656 — Математические и информационные технологии

Выпускная квалификационная работа магистра

Научный руководитель:
к. ф.-м. н. Стрижов Вадим Викторович

Москва
2014

Содержание

Аннотация

В работе исследуются алгоритмы индуктивного порождения и упрощения допустимых существенно нелинейных моделей.

Предлагается алгоритм, порождающий все возможные суперпозиции заданной сложности за конечное число шагов, и приводится его теоретическое обоснование. Сформулирован практически применимый стохастический алгоритм индуктивного порождения существенно нелинейных суперпозиций. Приводятся результаты вычислительного эксперимента по выбору оптимальной модели для синтетического набора данных, а также на примере моделирования волатильности опционов.

Далее, предлагается алгоритм упрощения суперпозиций, являющийся усовершенствованием ранее предложенных методов упрощения выражений по правилам. Суперпозиции представляются в виде направленного ациклического графа с объединением общих поддеревьев. Такое представление позволяет существенно расширить класс допустимых преобразований суперпозиций. Доказывается корректность и полнота предложенного алгоритма.

Предлагается критерий определения погрешности коэффициентов порожденных суперпозиций, называемый устойчивостью, а также метод оценки устойчивости полученного решения

Приводятся результаты вычислительного эксперимента по порождению и упрощению суперпозиций для набора синтетических данных, а для метода оценки устойчивости — на данных, полученных в ходе эксперимента по определению состава смеси по суммарной дисперсии.

Ключевые слова: *символьная регрессия, нелинейные модели, индуктивное порождение, упрощение выражений, сложность моделей, преобразование графов по правилам, устойчивость решений.*

1 Введение

Для анализа результатов физического эксперимента, как правило, требуется восстановить функциональную зависимость, описывающую соотношение измеряемых величин. При этом необходимо, чтобы эксперт имел возможность интерпретировать полученную зависимость, исходя из соответствующих теоретических моделей [1]. Во многих случаях вид функциональной зависимости заранее известен, либо необходимо сделать выбор между несколькими (также заранее известными) вариантами моделей.

Одним из методов, позволяющих строить интерпретируемые модели, является символьная регрессия [2–6], порождающая, в том числе, и структурно сложные нелинейные модели. Различные приближения сравниваются согласно ошибке на измеряемых данных, при этом оптимизация параметров модели проводится, например, с помощью алгоритма Левенберга-Марквардта [7, 8].

Генетическое программирование, предложенное Джоном Коза и заключающееся в применении эволюционных методов для построения программ (и, в том числе, суперпозиций функций), является одной из возможных реализаций этого метода. Джон Коза существенно расширил идею представления программ в генетическом программировании в виде деревьев, впервые рассмотренную Крамером [9].

В общем виде алгоритм построения требуемой математической модели в генетическом программировании выглядит следующим образом: дан набор примитивных функций, из которых можно строить различные формулы (например, степенная функция, +, sin, tan). Начальный набор формул строится либо произвольным образом, либо на базе некоторых предположений эксперта. Затем на каждом шаге производится оценка каждой из формул согласно функции ошибки либо другого функционала качества [10]. На базе этой оценки у некоторой части формул случайным образом заменяется одна элементарная функция на другую (например, sin на cos или + на ×), а у некоторой другой части происходит взаимный попарный обмен подвыражениями.

Иван Зелинка предложил дальнейшее развитие идеи применения эволюционных алгоритмов для восстановления функциональной зависимости, получившее название аналитического программирования, которое, однако, отошло от представления программ в виде деревьев. В работе [11] предлагается перенумеровать используемые функции и переменные, и по полученным номерам специальным обратимым алго-

ритмом строить строки из нулей и единиц, к которым уже применимы любые эволюционные алгоритмы, работающие с бинарными строками. Для оценки качества получающихся суперпозиций на каждой итерации алгоритма происходит обратное декодирование имеющихся строк и сравнение их результатов с требуемыми.

Алгоритм индуктивного порождения моделей, сформулированный в настоящей работе, свободен от некоторых типичных проблем предложенных ранее методов, в том числе [11]:

- Порождение рекурсивных суперпозиций, а также суперпозиций, содержащих несоответствующее используемым функциям число аргументов, и т. д. (в предложенном алгоритме эти проблемы не возникают по построению).
- Несовпадение области определения некоторой примитивной функции и области значений ее аргументов (возможно, тоже некоторых суперпозиций).
- Порождение слишком сложных суперпозиций.

Действительно, для любой выборки можно построить такой полином, который пройдет через все точки выборки, но при этом число параметров такого полинома линейно растет с объемом выборки. Кроме того, такой многочлен неинтерпретируем экспертами из-за своей высокой степени, соответствующей числу объектов в обучающей выборке. Предложенный в настоящей работе алгоритм решает проблему порождения слишком сложных суперпозиций введением дополнительного штрафа за сложность. Кроме того, так как используемые признаки объектов выборки учитываются при расчете сложности, применение подобного штрафа обеспечивает выбор суперпозиций, использующих меньшее число признаков, то есть, обеспечивает отбор признаков.

Однако при использовании методов генетического программирования возникает проблема порождения избыточных формул [12–14], то есть таких формул, в которых некоторые подвыражения могут быть удалены или заменены на более простые без ухудшения качества аппроксимации. Например, если некоторое подвыражение в суперпозиции умножается на ноль, то все это подвыражение можно заменить нулем.

Таким образом, для построения интерпретируемых суперпозиций целесообразна дополнительная процедура упрощения. Кроме того, процедура упрощения математических выражений представляет самостоятельный интерес и может быть применима, например, в системах компьютерной алгебры [15].

Ранее были предложены такие методы упрощения суперпозиций, как упрощение выражений по правилам (*rule-based simplification, RBS*) [16,17] и замена поддеревьев на эквивалентные меньшей сложности (*equivalent decision simplification, EDS*) [18].

В работах [16–18] слова «суперпозиция», «формула» и «математическое выражение» (или просто «выражение») взаимозаменяемы и обозначают некоторую композицию свободных переменных, констант и элементарных функций. В данной работе далее используется термин «суперпозиция».

В настоящей работе предлагается алгоритм, основанный на упрощении суперпозиций по правилам, описывающим, как функции связаны между собой. В качестве примера таких правил можно указать $\log \circ \exp \equiv \mathrm{id}$ или $t^n \times t^m \equiv t^{n+m}$.

Ранее предлагалось [13, 15, 19–21] представлять суперпозиции в виде соответствующего им дерева, над которым и оперировали предлагавшиеся алгоритмы. В настоящей работе суперпозиция представляется не в виде дерева, а в виде направленного ациклического графа, где различные функции могут принимать в качестве аргумента одно и то же подвыражение. Примером может являться суперпозиция $\cos^2 t + \sin^2 t$, где t — некое сложное подвыражение. Таким образом, искомая задача сводится к задаче нахождения общих подвыражений в исходном дереве суперпозиции и задания правил упрощения на множестве подобных ациклических графов.

Однако при анализе физического эксперимента важна не только интерпретируемость искомой функциональной зависимости и значения ее параметров, но и погрешности их определения, обусловленные погрешностями измеряемых в эксперименте величин. Для задачи линейной регрессии получено точное аналитическое решение в частном случае, когда погрешность определения регрессора пренебрежимо мала, а погрешность определения зависимой переменной во всех экспериментальных точках одинакова [22]. Для более сложного случая нелинейной регрессии и ситуации, когда необходимо учитывать погрешности как регрессора, так и зависимой переменной (которые при этом могут быть разными в различных экспериментальных точках), подобная задача, насколько нам известно, не ставилась.

В настоящей работе метод нелинейной регрессии применяется для восстановления зависимости показателя преломления n от длины волны λ в полосе прозрачности полимера, включающей видимую и ближнюю инфракрасную области спектра. Цель экспериментаторов состояла в том, чтобы по известной дисперсии для каждого полимера с учетом того, что показатель преломления смеси химически инертных полимеров равен взвешенной сумме (с соответствующими весами) показателей пре-

ломления компонентов, определить состав смеси по экспериментально определенной зависимости $n(\lambda)$. Другими словами, для случая двух полимеров, заранее измерив и вычислив зависимости $n_1(\lambda)$ и $n_2(\lambda)$, необходимо экспериментально определить суммарную зависимость $n(\lambda) = \alpha n_1(\lambda) + (1 - \alpha)n_2(\lambda)$ и по ней вычислить коэффициент α, имеющий смысл концентрации первого полимера в смеси.

Поскольку показатели преломления для прозрачных полимеров близкого химического состава различаются незначительно, учет погрешности определения коэффициентов функциональной зависимости $n(\lambda)$ и их связи с погрешностями экспериментального определения длины волны λ и показателя преломления n имеет принципиальное значение. Указанная связь важна еще и потому, что именно она определяет требования к точности и чувствительности измерительной аппаратуры и, следовательно, влияет на стоимость и продолжительность эксперимента.

Обычно в рефрактометрах используются источники широкополосного (непрерывного) спектра, а погрешность выделения конкретной длины волны определяется аппаратной функцией используемого монохроматора (прибора, выделяющего узкий спектральный диапазон) и подробно рассматривается, например, в [23, 24]. В большинстве случаев погрешность λ может быть рассчитана, а также определена экспериментально с использованием узкополосных источников света (лазеров, известных атомных переходов вроде триплета ртути или дублета натрия, и т. д.). Характерная относительная погрешность определения длины волны в рассматриваемой задаче обычно составляет $0.03 \div 0.5\%$, а абсолютная погрешность определения длины волны, как правило, меняется с изменением самой длины волны. Экспериментальная погрешность показателя преломления n зависит от выбранного способа его измерения и, например, при определении n по углу полного внутреннего отражения обусловлена непараллельностью используемых световых пучков, погрешностями в измерении углов и т. д, и составляет от $(1 \div 2) \cdot 10^{-5}$ для приборов высокого класса точности до $(1 \div 10) \cdot 10^{-4}$. Существенно, что величины погрешностей могут считаться известными и, возможно, различными для каждой экспериментальной точки.

В настоящей работе на примере дисперсионной зависимости показателя преломления исследуется возможность применения предложенного в [6] алгоритма восстановления нелинейной регрессии. Результаты его работы сравниваются с результатами применения SVM-регрессии. Кроме того, рассмотрено влияние штрафа за сложность на качество и структурную сложность порождаемых суперпозиций. Формально поставлена задача определения устойчивости регрессионной зависимости от произ-

вольного набора независимых переменных в общем виде, предложен метод оценки устойчивости решения к погрешностям измерений, и изучена зависимость этих характеристик от параметров модели для конкретного случая определения дисперсии полимеров.

В первой части работы поставлена задача восстановления дисперсии полимера и предложено понятие устойчивости суперпозиции, позволяющее учитывать и анализировать зависимость погрешностей различных коэффициентов порожденной суперпозиции от погрешности измеряемых данных. Во второй части вкратце описывается алгоритм [6], используемый для порождения аналитической функции-суперпозиции, аппроксимирующей данные. В третьей части предложен численный метод нахождения устойчивости суперпозиции. В четвертой части приводятся результаты вычислительного эксперимента на реальных данных. Рассматривается два полимера в области прозрачности, для каждого из которых имеется 17 экспериментальных точек, соответствующих величине коэффициента преломления при различных значениях длины волны.

2 Постановка задач

2.1 Порождение суперпозиций

Пусть дана выборка:

$$D = \{(\mathbf{x}_i, y_i) \mid i \in \{1, \ldots, N\}, \mathbf{x}_i \in \mathbb{X} \subset \mathbb{R}^n, y_i \in \mathbb{Y} \subset \mathbb{R}\},$$

где N — число элементов выборки, $\mathbf{x}_i$ — вектор значений свободных переменных для i-го элемента выборки, y_i — значение зависимой переменной для i-го элемента выборки, $\mathbb{X}$ — множество значений независимых переменных, лежащее в $\mathbb{R}^n$, $\mathbb{Y}$ — множество значений зависимой переменной, лежащее в $\mathbb{R}$.

Требуется выбрать параметрическую функцию $f : \Omega \times \mathbb{X} \to \mathbb{R}$ из порождаемого множества $\mathcal{F} = \{f_r\}$, где Ω — пространство параметров, доставляющую минимум некоторому заданному функционалу качества Q, зависящему от функционала ошибки S_f на данной выборке D и сложности суперпозиции $C(f)$.

То есть, для множества всех суперпозиций

$$\mathcal{F} = \{f_r \mid f_r : (\boldsymbol{\omega}, \mathbf{x}) \mapsto y \in \mathbb{Y}, r \in \mathbb{N}\},$$

требуется найти такой индекс $\hat{r}$, что функция f_r среди всех $f \in \mathcal{F}$ доставляет минимум функционалу качества Q при данной выборке D:

$$\hat{r} = \arg\min_{r \in \mathbb{N}} Q(f_r \mid \hat{\boldsymbol{\omega}}_{\boldsymbol{r}}, D), \tag{2.1}$$

где $\hat{\boldsymbol{\omega}}_r$ — оптимальный вектор параметров функции f_r, минимизирующий некоторый функционал ошибки S, для каждой $f \in \mathcal{F}$ при данной выборке D:

$$\hat{\boldsymbol{\omega}}_{\boldsymbol{r}} = \arg\min_{\boldsymbol{\omega} \in \Omega} S(\boldsymbol{\omega} \mid f_r, D). \tag{2.2}$$

Сформулируем также постановку теоретической задачи порождения суперпозиций конечной длины. Для этого сначала введем понятие суперпозиции функций.

Если множество значений $\mathbb{Y}_i$ функции f_i содержится в области определения $\mathbb{X}_{i+1}$ функции f_{i+1}, то есть

$$f_i : \mathbb{X}_i \to \mathbb{Y}_i \subset \mathbb{X}_{i+1}, \quad i = 1, 2, \ldots, \theta - 1,$$

то функция

$$f_\theta \circ f_{\theta-1} \circ \cdots \circ f_1, \quad \theta \geqslant 2,$$

определяемая равенством

$$(f_\theta \circ f_{\theta-1} \circ \cdots \circ f_1)(\mathbf{x}) = f_\theta(f_{\theta-1}(\ldots(f_1(\mathbf{x})))), \quad x \in \mathbb{X}_1,$$

называется *сложной функцией* [25] или *суперпозицией функций* $f_1, f_2, \ldots, f_\theta$.

Таким образом, получаем

Определение 1. Суперпозиция функций — функция, представленная как композиция нескольких функций.

Для упрощения и обобщения этой формулировки на случай функций нескольких переменных примем, что все функции $g_1, \ldots, g_\theta$, входящие в суперпозицию, являются вектор-функциями $\mathbf{g}_1, \ldots, \mathbf{g}_\theta$ от векторной величины $\mathbf{x}$. При этом и области определения $\mathbb{X}_i$, и области значений $\mathbb{Y}_i$ этих вектор-функций являются подмножествами декартова произведения пространств соответствующих переменных.

Пусть $G = \{g_1, \ldots, g_l\}$ — множество данных порождающих функций, а именно, для каждой $g_i \in G$ заданы:

- сама функция g_i (например, sin, cos, $\times$),

- арность функции и порядок следования аргументов,
- домен ($\mathrm{dom} g_i$) и кодомен ($\mathrm{cod} g_i$) функции,
- область определения $\mathcal{D}g_i \subset \mathrm{dom} g_i$ и область значений $\mathcal{E}g_i \subset \mathrm{cod} g_i$.

Требуется построить функцию f как суперпозицию порождающих функций из заданного множества G.

Поясним различие между последними двумя пунктами. Например, $\mathrm{dom} f$ показывает, значения из какого множества принимает функция f (целые числа, действительные числа, декартово произведение целых чисел и $\{0, 1\}$, и т. п.). Область определения же показывает, на каких значениях из $\mathrm{dom} f$ функция f определена и имеет смысл. Так, для функции $f(x_1, x_2) = \log_{x_1} x_2$:

$$\mathrm{dom} f = \mathbb{R} \times \mathbb{R},$$

$$\mathrm{cod} f = \mathbb{R},$$

$$\mathcal{D}f = \{(x_1, x_2) \mid x_1 \in (0; 1) \cup (1; +\infty), x_2 \in (0; +\infty)\},$$

$$\mathcal{E}f = (-\infty; +\infty).$$

Требуется также построить алгоритм $\mathfrak{A}$, за конечное число итераций порождающий любую конечную суперпозицию данных примитивных функций.

Заметим, что мы не требуем для примитивных функций свойства их непорождаемости в наиболее общей формулировке типа принципиальной невозможности породить в ходе работы искомого алгоритма суперпозицию, изоморфную некоторой функции из G. Такое требование является слишком ограничивающим. В частности, невозможно было бы иметь в G одновременно, например, функции id, exp и log, так как $\mathrm{id} \equiv \log \circ \exp$.

Для большей общности будем считать, что константы являются нуль-арными (нуль-местными) функциями — функциями, не имеющими аргументов, а суперпозиция, соответствующая единственной свободной переменной ($f(\mathbf{x}) = x_i$), эквивалентна функции вида $\mathrm{id} x_i$.

2.2 Упрощение суперпозиций

Пусть дана суперпозиция f, состоящая из произвольного набора функций, свободных переменных и констант. Пусть также дан набор правил-аксиом, указывающих на существующие соотношения между элементарными функциями. Требуется

согласно этим правилам построить суперпозицию, изоморфную исходной и обладающую наименьшей сложностью.

Здесь под изоморфизмом двух суперпозиций понимается такое эквивалентное преобразование, что обе суперпозиции дают одинаковые результаты при одних и тех же значениях свободных переменных.

Определим используемые понятия.

Пусть задано множество $G = \{g_i\}$ элементарных функций. Для каждой функции g_i задана область определения $\mathbb{X}_i$ и область значения $\mathbb{Y}_i$.

Определим понятие сложности суперпозиции $C(f)$:

Определение 2. Сложность суперпозиции f, обозначаемая $C(f)$ — число элементарных функций, констант и свободных переменных, каждые из которых считаются столько раз, сколько встречаются в суперпозиции.

Например, сложность суперпозиции $x + y + y$ равна 5, а сложность $x + y + 2y^3$ равна 9.

Введем также множество $\mathcal{F}$ всех возможных суперпозиций, составленных из элементарных функций $g \in G$.

Исходная задача формулируется следующим образом. Для данной суперпозиции f требуется найти суперпозицию $\hat{f}$, имеющую минимальную сложность среди всех суперпозиций, изоморфных f:

$$\hat{f} = \arg \min_{\varphi \in \mathcal{F}_f \subset \mathcal{F}} C(\varphi),$$

где $\mathcal{F}_f \subset \mathcal{F}$ — множество всех возможных суперпозиций, изоморфных f. Множество $\mathcal{F}_f$ строится путем последовательного применения заданных правил, описывающих возможные соотношения между элементарными функциями, и являющихся правилами преобразования суперпозиций.

2.3 Оценка устойчивости суперпозиций

Введем в общем виде понятие устойчивости суперпозиции f, характеризующей поведение коэффициентов суперпозиции $\hat{f}$ при небольшом случайном изменении исходной обучающей выборки $D = \{\mathbf{x}_i, y_i\}$, где $\mathbf{x}_i$ — исходное (полученное в ходе эксперимента) признаковое описание i-го объекта, а y_i — соответствующее экспериментально измеренное значение функции, которую требуется восстановить.

Функционал ошибки S в этом случае выглядит следующим образом:

$$S(f, D) = \sum_{i=1}^{\ell} (f(\mathbf{x}_i) - y_i)^2 \to \min_{f \in \mathcal{F}}. \tag{2.3}$$

Условимся также обозначать матрицу плана $X = \|x_{ij}\|$, строками которой являются признаковые описания объектов выборки D. Иными словами, x_{ij} обозначает j-ую компоненту признакового описания i-го объекта.

Рассмотрим вектор параметров $\boldsymbol{\omega}_f = \{\omega_i^f \mid i \in \{1, \ldots, l_f\}\}$ некоторой суперпозиции f: $f(\mathbf{x}) = f(\mathbf{x}, \boldsymbol{\omega}_f)$. Пусть для некоторой выборки $D = \{\mathbf{x}_i, y_i\}$ и функции f вектор параметров $\hat{\boldsymbol{\omega}}_f(D)$ минимизирует функционал (2.3) с суперпозицией f, имеющей фиксированную структуру:

$$\hat{\boldsymbol{\omega}}_f(D) = \arg\min_{\boldsymbol{\omega}_f} S(f, D).$$

Пусть также дана матрица стандартных отклонений независимых переменных $\Sigma^{\mathbf{x}} = \|\sigma_{ij}^{\mathbf{x}}\|$, где $\sigma_{ij}^{\mathbf{x}}$ характеризует стандартное отклонение j-ой компоненты признакового описания $\mathbf{x}_i$ i-го объекта обучающей выборки, и вектор стандартных отклонений $\boldsymbol{\sigma}^y$, где σ_i^y характеризует стандартное отклонение зависимой переменной, соответствующей i-му объекту. Рассмотрим выборку $\acute{D}$, полученную из исходной выборки D добавлением к каждой компоненте реализаций нормально распределенных случайных величин с нулевым матожиданием и соответствующей $\Sigma^{\mathbf{x}}$ и $\boldsymbol{\sigma}^y$ дисперсией:

$$\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}^y) = \{\mathbf{x}_i + \boldsymbol{\xi}_i^{\mathbf{x}}, y_i + \xi_i^y \mid i \in 1, \ldots, \ell; \boldsymbol{\xi}_i^{\mathbf{x}} \sim \mathcal{N}(0; \boldsymbol{\sigma}_{i\cdot}^{\mathbf{x}}); \xi_i^y \sim \mathcal{N}(0; \sigma_i^y)\}. \tag{2.4}$$

Для этой выборки $\acute{D}$ найдем оптимальный вектор $\hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y))$ параметров суперпозиции f, минимизирующий функционал (2.3):

$$\hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)) = \arg\min_{\boldsymbol{\omega}_{f_D} \in R^{|\bar{\omega}_f|}} S(f_D(\cdot, \boldsymbol{\omega}_{f_D}), \acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)). \tag{2.5}$$

Поскольку $\hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y))$ — векторная случайная величина,

$$\Delta\hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)) = \hat{\boldsymbol{\omega}}_f(D) - \hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y))$$

также векторная случайная величина.

Пусть дано множество $\acute{\mathcal{D}}_N$ из N таких выборок, где каждая выборка соответствует отдельным реализациям случайных величин из (2.4):

$$\acute{\mathcal{D}}_N(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y) = \{\acute{D}_1(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y), \ldots, \acute{D}_N(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)\},$$

и пусть $\overline{\sigma}_i$ — эмпирическое стандартное отклонение i-ой компоненты векторной случайной величины $\Delta\hat{\boldsymbol{\omega}}_f(\acute{D}(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y))$ на множестве $\acute{\mathcal{D}}_N(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)$.

Определение 3. *Относительной устойчивостью* (или просто *устойчивостью*) параметра ω_i относительно $\acute{\mathcal{D}}_N(\Sigma^{\mathbf{x}}, \boldsymbol{\sigma}_y)$ при исходной обучающей выборке D будем называть следующий вектор из $|\mathbf{x}| + 1$ компонент:

$$\mathbf{T}_f^N(i) = \left\{ \frac{\frac{\overline{\sigma}_i}{\hat{\omega}_i}}{r(\boldsymbol{\sigma}^{\mathbf{x}}_{\cdot 1}, \mathbf{x}_{\cdot 1})}, \ldots, \frac{\frac{\overline{\sigma}_i}{\hat{\omega}_i}}{r(\boldsymbol{\sigma}^{\mathbf{x}}_{\cdot |\mathbf{x}|}, \mathbf{x}_{\cdot |\mathbf{x}|})}, \frac{\frac{\overline{\sigma}_i}{\hat{\omega}_i}}{r(\boldsymbol{\sigma}^{y}, \mathbf{y})} \right\}, \tag{2.6}$$

где $r(\boldsymbol{\alpha}, \mathbf{a}) = r(\frac{\alpha_1}{a_1}, \ldots, \frac{\alpha_{|\boldsymbol{\alpha}|}}{a_{|\mathbf{a}|}})$ — функция, переводящая вектор, составленный из отношений соответствующих компонент векторов $\boldsymbol{\alpha}$ и $\mathbf{a}$, в скаляр.

Функция r позволяет сопоставить, вообще говоря, различным отношениям стандартного отклонения зависимой переменной y_i i-го объекта обучающей выборки и самого значения y_i единственное число (аналогично и для x_i-й независимой переменной и ее стандартного отклонения). Примерами такой функции могут являться среднее арифметическое всех отношений или максимальное значение отношения, а конкретная функция выбирается экспертом в зависимости от физического смысла задачи. В настоящей работе предлагается выбирать значения стандартных отклонений так, чтобы все значения соответствующих переменных были равны, поэтому функция r может выбирать любой из своих аргументов.

Каждый компонент вектора $\mathbf{T}_f^N(i)$ показывает, как относится стандартное отклонение параметра $\hat{\omega}_i$, нормированное на значение этого параметра, к характерному стандартному отклонению соответствующего элемента признакового описания, нормированного на значение этого элемента. Например, если это отношение больше единицы, то погрешности определения коэффициента растут быстрее погрешностей измерения параметра.

В частности, в искомой задаче восстановления дисперсионной зависимости для неизменной относительной ошибки эксперимента:

$$\mathbf{T}_f(i) = \left\{ \frac{\frac{\overline{\sigma}_i}{\hat{\omega}_i}}{\frac{\sigma_n}{n}}, \frac{\frac{\overline{\sigma}_i}{\hat{\omega}_i}}{\frac{\sigma_\lambda}{\lambda}} \right\}.$$

Матрицу, столбцами которой являются векторы $\mathbf{T}_f(i) \mid i \in \{1, \ldots, l_f\}$, будем называть устойчивостью функции f и обозначать $\mathbb{T}_f$.

Требуется исследовать зависимость устойчивости $\mathbb{T}_{\hat{f}}$ относительно σ_n и σ_λ.

3 Алгоритм индуктивного порождения допустимых суперпозиций

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 $g_s, s = s(i)$.
- Число дочерних вершин у некоторой вершины V_i равно арности соответствующей функции g_s.
- Порядок смежных некоторой вершине V_i вершин соотвествует порядку аргументов соответствующей функции $g_{s(i)}$.
- В листьях дерева Γ_f находятся свободные переменные x_i либо числовые параметры ω_i.
- Дерево вычисляется снизу вверх. То есть, сначала подставляются конкретные значения свободных переменных x_i, затем вычисляются значения в вершинах, все дочерние вершины которых — свободные переменные, и так далее до тех пор, пока не останется единственная вершина, бывшая корнем дерева. Эта вершина содержит число, являющееся результатом вычисления суперпозиции в точке, соответствующей $\mathbf{x} = \{x_i\}$.

Таким образом, вычисление значения выражения f в некоторой точке с данным вектором параметров $\boldsymbol{\omega} = \{\omega_1, \omega_2, \ldots, \omega_\eta\}$ эквивалентно подстановке соответствующих значений свободных переменных x_i и параметров ω_i в дерево Γ_f, где x_i — компоненты вектора признакового описания объекта $\mathbf{x}$.

Из определения 2 сложности суперпозиции следует, что сложность $C(f)$ суперпозиции f равна числу узлов в дереве Γ_f.

Отметим важное свойство таких деревьев: каждое поддерево Γ_f^i дерева Γ_f, соответствующее вершине V_i, также соответствует некоторой суперпозиции, являющейся составляющей исходной суперпозиции f. Будем обозначать такую суперпозицию, соответствующую вершине V_i, как f_{V_i}.

Для примера рассмотрим дерево, соответствующиее суперпозиции $f = \sin(\ln x_1) + \frac{x_2^3}{2}$ (см. рис 1).

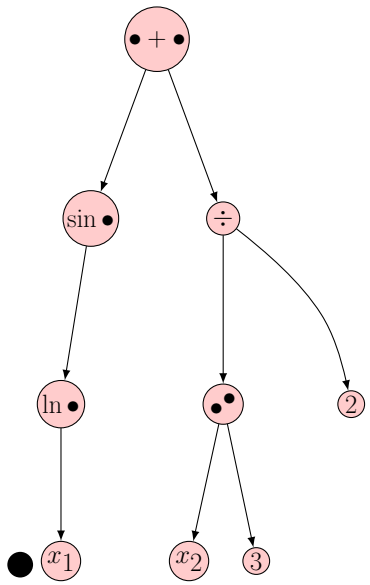


Рис. 1: Дерево выражения $\sin(\ln x_1) + \frac{x_2^3}{2}$

Здесь точками обозначены аргументы функций. Как видно, корнем дерева является вершина, соответствующая операции сложения, которая должна быть выполнена в последнюю очередь. Операция сложения имеет два различных поддерева, соответствующих двум аргументам этой операции. Заметим также, что здесь не использованы операции типа «разделить на два» или «возвести в куб». Вместо этого используются операции деления и возведения в степень в общем виде, а в данном конкретном дереве соответствующие аргументы зафиксированы соответствующими константами.

Алгоритм порождения суперпозиций. Сначала определим понятие *глубины суперпозиции*:

Определение 4. Глубина суперпозиции f — максимальная глубина дерева Γ_f.

Теперь опишем итеративный алгоритм $\mathfrak{A}^*$, порождающий суперпозиции, не содержащие параметров. Предложенный алгоритм породит любую суперпозицию конечной глубины за конечное число шагов.

Пусть дано множество примитивных функций $G = \{g_1, \ldots, g_l\}$ и множество свободных переменных $X = \{x_1, \ldots, x_n\}$. Для удобства будем исходить из предположения, что множество G состоит только из унарных и бинарных функций, и разделим его соответствующим образом на два подмножества: $G = G_b \cup G_u \mid G_b = \{g_{b_1}, \ldots, g_{b_k}\}, G_u = \{g_{u_1}, \ldots, g_{u_l}\}$, где G_b — множество всех бинарных функций, а

G_u — множество всех унарных функций из G. Потребуем также наличия id в G_b.

Алгоритм 1. Алгоритм $\mathfrak{A}^*$ итеративного порождения суперпозиций.

1. Перед первым шагом зададим начальные значения множества $\mathcal{F}_0$ и вспомогательного индексного множества $\mathcal{I}$, служащего для запоминания, на какой итерации впервые встречена каждая суперпозиция:

$$\mathcal{F}_0 = X,$$

$$\mathcal{I} = \{(x, 0) \mid x \in X\}.$$

2. Для множества $\mathcal{F}_i$ построим вспомогательное множество U_i, состоящее из суперпозиций, полученных в результате применения функций $g_u \in G_u$ к элементам $\mathcal{F}_i$:

$$U_i = \{g_u \circ f \mid g_u \in G_u, f \in \mathcal{F}_i\}.$$

3. Аналогичным образом построим вспомогательное множество B_i для бинарных функций $g_b \in G_b$:

$$B_i = \{g_b \circ (f, h) \mid g_b \in G_b, f, h \in \mathcal{F}_i\}.$$

4. Обозначим $\mathcal{F}_{i+1} = \mathcal{F}_i \cup U_i \cup B_i$.

5. Для каждой суперпозиции f из $\mathcal{F}_{i+1}$ добавим пару $(f, i+1)$ в множество $\mathcal{I}$, если суперпозиция f еще там не присутствует.

6. Перейдем к следующей итерации, п. 2.

Тогда $\mathcal{F} = \cup_{i=0}^{\infty} \mathcal{F}_i$ — множество всех возможных суперпозиций конечной длины, которые можно построить из данного множества примитивных функций.

Вспомогательное множество $\mathcal{I}$ позволяет запоминать, на какой итерации была впервые встречена каждая суперпозиция. Это необходимо, так как каждая суперпозиция, впервые порожденная на i-ой итерации, будет порождена также и на любой итерации после i. Одной из возможностей избежать необходимости в этом множестве является построение $\mathcal{F}_{i+1}$ как $\mathcal{F}_{i+1} = U_i \cup B_i$ (без $\mathcal{F}_i$), а множества U_i и B_i строить следующим образом:

$$U_i = \{g_u \circ f \mid g_u \in G_u, f \in \cup_{j=0}^{i} \mathcal{F}_j\},$$

$$B_i = \{g_b \circ (f, h) \mid g_b \in G_b, f, h \in \cup_{j=0}^{i} \mathcal{F}_j\}.$$

Алгоритм $\mathfrak{A}^*$ очевидным образом обобщается на случай, когда множество G содержит функции произвольной (но конечной) арности. Действительно, для такого обобщения достаточно строить аналогичным образом вспомогательные множества для этих функций, а именно: для множества функций G_n арности n построим вспомогательное множество H_i^n вида:

$$H_i^n = \{g \circ (f_1, f_2, \ldots, f_n) \mid g \in G_n, f_j \in \mathcal{F}_i\}.$$

В этих обозначениях $U_i \equiv H_i^1$, а $B_i \equiv H_i^2$.

Тогда множество $\mathcal{F}_{i+1} = \mathcal{F}_i \cup_{n=0}^{n_{max}} H_i^n$, где n_{max} — максимальное значение арности функций из G.

Теорема 1. Алгоритм $\mathfrak{A}^*$ действительно породит любую конечную суперпозицию за конечное число шагов.

Доказательство.

Чтобы убедиться в этом, найдем номер итерации, на котором будет порождена некоторая произвольная конечная суперпозиция f. Чтобы найти этот номер, пронумеруем вершины графа Γ_f по следующим правилам:

- Если это вершина со свободной переменной, то она имеет номер 0.
- Если вершина V соответствует унарной функции, то она имеет номер $i+1$, где i — номер дочерней для этой функции вершины.
- Если вершина V соответствует бинарной функции, то она имеет номер $i+1$, где $i = \max(l, r)$, а l и r — номера, соответственно, первой и второй дочерней вершины.

Нумеруя вершины графа Γ_f таким образом, мы получим номер вершины, соответствующей корню графа. Это и будет номером итерации, на которой получена суперпозиция f.

Иными словами, для любой суперпозиции мы можем указать конкретный номер итерации, на котором она будет получена, что и требовалось. ■

В рассмотренных ранее методах построения суперпозиций [11] необходимо было самостоятельно следить за тем, чтобы в ходе работы алгоритма не возникало «зацикленных» суперпозиций типа $f(x, y) = g(f(x, y), x, y)$. Заметим, что в предложенном алгоритме $\mathfrak{A}^*$ такие суперпозиции не могут возникнуть по построению.

3.1 Порождение моделей с параметрами.

Алгоритм $\mathfrak{A}^*$ в таком виде не позволяет получать выражения, содержащие численные параметры $\boldsymbol{\omega}$ суперпозиции $f(\boldsymbol{\omega}, \mathbf{x})$. Покажем, однако, на примере конструирования множеств U_i и B_i, как исходный алгоритм $\mathfrak{A}^*$ может быть расширен путем введения параметров:

$$U_i = g_u \circ (\alpha f + \beta),$$

$$B_i = g_b \circ (\alpha f + \beta, \psi h + \varphi).$$

Будем обозначать этот расширенный алгоритм как $\mathfrak{A}$. Здесь параметры α, β зависят только от комбинации g_u, f (или g_b, f, h для $\alpha, \beta, \psi, \varphi$). Соответственно, для упрощения их индексы опущены. Иными словами, мы предполагаем, что каждая суперпозиция из предыдущих итераций входит в следующую, будучи умноженной на некоторой коэффициент и с константной поправкой.

Очевидно, при таком добавлении параметров $\alpha, \beta, \psi, \varphi$ мы не изменяем мощности получившегося множества суперпозиций, поэтому алгоритм и выводы из него остаются корректными. В частности, исходный алгоритм является частным случаем данного при $\alpha \equiv \psi \equiv 1, \beta \equiv \varphi \equiv 0$.

Переменные $\alpha, \beta, \psi, \varphi$ являются параметрами модели. В практических приложениях можно оптимизировать значения этих параметров у получившихся суперпозиций, например, алгоритмом Левенберга-Марквардта [7, 8].

Заметим также, что такая модификация алгоритма позволяет нам получить единицу, например, для построения суперпозиций типа $\frac{1}{x}$: $1 = \alpha\ id\ x + \beta \mid \alpha = 0, \beta = 1$.

Отдельно подчеркнем, что параметры $\boldsymbol{\omega}$ у различных суперпозиций различны. Однако, так как каждый из параметров зависит только от соответствующей комбинации функций, к которым он относится, конкретные значения параметров не учитываются при поиске одинаковых суперпозиций. Иными словами, при тестировании суперпозиций на равенство сравниваются лишь структуры соответствующих им деревьев и значения в узлах, соответствующих функциям и свободным переменным.

Заметим, что и этот алгоритм очевидным образом обобщается на случай множества G, содержащего функции произвольной арности.

3.2 Множество допустимых суперпозиций

Предложенный выше алгоритм позволяет получить действительно все возможные суперпозиции, однако, не все они будут пригодны в практических приложениях: например, $\ln x$ имеет смысл только при $x > 0$, а $\frac{x}{0}$ не имеет смысла вообще никогда. Выражения типа $\frac{x}{\sin x}$ имеют смысл только при $x \neq \pi k$.

Таким образом, необходимо введение понятия множества *допустимых* суперпозиций — то есть, таких суперпозиций, которые в условиях данной задачи корректны.

Определение 5. Допустимая суперпозиция f — такая суперпозиция, значение которой определено для любой комбинации значений свободных переменных, область значений $\mathbb{X}$ которых определяется конкретной задачей, $\mathbb{X} \subset \mathbb{R}^n$ где n — число свободных переменных.

Одним из способов построения только допустимых суперпозиций является модификация предложенного алгоритма таким образом, чтобы отслеживать совместность областей определения и областей значения соответствующих функций в ходе построения суперпозиций. Для свободных переменных это, в свою очередь, означает необходимость задания областей значений $\mathbb{X}$ пользователем при решении конкретных задач.

Таким образом, можно сформулировать очевидное *достаточное условие недопустимости* суперпозиции:

Определение 6. Достаточное условие недопустимости суперпозиции f: в соответствующем дереве Γ_f хотя бы одна вершина V_i имеет хотя бы одну дочернюю вершину V_j такую, что область значений функции $g_{s(j)}$ шире, чем область определения функции $g_{s(i)}$:

$$\exists i, j : V_i \in \Gamma_f, V_j \in \Gamma_f \wedge \exists \kappa : \kappa \in \mathcal{E}g_{s(j)} \wedge \kappa \notin \mathcal{D}g_{s(i)}.$$

Говоря, что область значений функции f шире области определения функции g, мы имеем ввиду, что существует по крайней мере одно значение функции f, не входящее в область определения функции g.

Подчеркнем, что, хотя свободные переменные могут принимать, например, все значения из $\mathbb{R}$, выбором множества $\mathbb{X}$ можно обеспечить возможность использования их в качестве аргументов функций с более узкой, чем $\mathbb{R}$, но не менее узкой, чем $\mathbb{X}$, областью определения, если это не противоречит данной выборке.

Для построения множества допустимых суперпозиций достаточно построить множество всех возможных суперпозиций при помощи алгоритма $\mathfrak{A}$, а затем удалить из этого множества все суперпозиции, не удовлетворяющие сформулированному признаку.

3.3 Количество возможных суперпозиций

Оценим количество суперпозиций, получаемых после каждой итерации алгоритма $\mathfrak{A}$. Очевидно, с учетом вышеупомянутых оговорок касательно сравнения параметризованных суперпозиций, это количество равно количеству для алгоритма $\mathfrak{A}^*$.

Итак, пусть дано n независимых переменных: $|X| = n$, а мощность множества G распишем через мощности его подмножеств функций соответствующей арности: $|G_1| = l_1, |G_2| = l_2, \ldots, |G_p| = l_p$. На нулевой итерации имеем $P_0 = n$ суперпозиций.

На первой итерации дополнительно порождается:

$$P_1 = l_1 n + l_2 n^2 + \cdots + l_n n^p = \sum_{i=1}^{p} l_i P_0^i,$$

и суммарное число суперпозиций после первой итерации:

$$\hat{P}_1 = P_1 + P_0 = \sum_{i=1}^{p} l_i P_0^i + P_0.$$

Как было замечено ранее, суперпозиции, порожденные на k-ой итерации, будут также порождены и на любой следующей после k итерации, поэтому суммарное число суперпозиций после второй итерации будет равно:

$$\hat{P}_2 = \sum_{i=1}^{p} l_i \hat{P}_1^i.$$

И вообще, после k-ой итерации будет порождено:

$$\hat{P}_k = \sum_{j=1}^{p} l_i \hat{P}_{k-1}^i.$$

Оценим порядок роста количества функций, порожденных после k-ой итерации.

Теорема 2. Пусть в множестве примитивных функций G содержится l_p функций арности $p > 1$ и ни одной функции арности $p + k \mid k > 0$, и имеется $n > 1$ независимых переменных. Тогда справедлива следующая оценка количества суперпозиций, порожденных алгоритмом $\mathfrak{A}$ после k-ой итерации:

$$|\mathcal{F}_k| = \mathcal{O}(l_p^{\sum_{i=0}^{k-1} p^i} n^{p^k}).$$

Доказательство.

Оценим сначала порядок роста для случая, когда есть лишь одна m-арная функция и n свободных переменных.

После первой итерации алгоритма будет порождено $n^m + n$ суперпозиций. После второй — $(n^m + n)^m + n^m + n$, что можно оценить как $(n^m)^m = n^{m^2}$. И вообще, после k-ой итерации количество суперпозиций можно оценить как n^{m^k}.

Видно, что для оценки скорости роста количества порожденных суперпозиций можно учитывать только функции с наибольшей арностью.

Рассмотрим теперь случай, когда имеется не одна функция арности m, а l_m таких функций. Тогда на первой итерации порождается $l_m n^m + n$ суперпозиций, на второй:

$$l_m(l_m n^m + n)^m + l_m n^m + n \approx l_m^{m+1} n^{m^2},$$

на третьей, с учетом этого приближения:

$$l_m(l_m^{m+1} n^{m^2})^m = l_m l_m^{m(m+1)} n^{m^3} = l_m^{m^2+m+1} n^{m^3}.$$

И вообще, скорость роста количества порожденных суперпозиций можно оценить как:

$$|\mathcal{F}_k| = \mathcal{O}(l_m^{\sum_{i=0}^{k-1} m^i} n^{m^k}).$$

Таким образом, получаем оценку в общем случае, когда в множестве G содержится l_p функций арности p и ни одной функции арности $p + k \mid k > 0$:

$$|\mathcal{F}_k| = \mathcal{O}(l_p^{\sum_{i=0}^{k-1} p^i} n^{p^k}).$$

■

3.4 Алгоритм итеративного стохастического порождения суперпозиций

Несмотря на то, что построенный ранее итеративный алгоритм $\mathfrak{A}$ порождения суперпозиций позволяет получить за конечное число шагов произвольную суперпозицию, для практических применений он непригоден в связи с чрезмерной вычислительной сложностью, как и любой алгоритм, реализующий полный перебор. Вместо него предлагается использовать стохастические алгоритмы и ряд эвристик, позволяющих на практике получать за приемлемое время результаты, удовлетворяющие

заранее заданным условиям. Ниже представлен практически реализуемый вариант алгоритма $\mathfrak{A}$, который и был использован в вычислительном эксперименте.

Сначала опишем вспомогательный алгоритм случайного порождения суперпозиции:

Алгоритм 2. Алгоритм случайного порождения суперпозиции $\mathcal{RF}$.

Вход:

- Набор пороговых значений $0 < \xi_1 < \xi_2 < \xi_3 < 1$.
- Максимальная глубина порождаемой суперпозиции Td.

Алгоритм работает следующим образом. Генерируется случайное число ξ на интервале $(0; 1)$, и рассматриваются следующие случаи:

- $\xi \leqslant \xi_1$: результатом алгоритма является некоторая случайно выбранная свободная переменная.
- $\xi_1 < \xi \leqslant \xi_2$: результатом алгоритма является числовой параметр.
- $\xi_2 < \xi \leqslant \xi_3$: результатом алгоритма является некоторая случайно выбранная унарная функция, для определения аргумента которой данный алгоритм рекурсивно запускается еще раз.
- $\xi_3 < \xi$: результатом алгоритма является некоторая случайно выбранная бинарная функция, аргументы которой порождаются аналогичным образом.

При этом порождение тривиальных суперпозиций (свободных переменных и параметров) запрещено: на самом первом шаге пороговые значения масштабируются таким образом, чтобы всегда порождалась унарная или бинарная функция. Аналогично при превышении значения Td пороговые значения масштабируются таким образом, чтобы был порожден узел, соответствующий свободной переменной или параметру, и алгоритм завершился.

Каждой порожденной суперпозиции f ставится в соответствие ее *качество* Q_f (будем также говорить, что суперпозиция *оценивается*), рассчитываемое исходя из функционала ошибки S_f этой суперпозиции на выборке D и ее сложности $C(f)$ — числа узлов в соответствующем графе Γ_f, по следующей формуле:

$$Q_f = \frac{1}{1 + S_f} \left(\alpha + \frac{1 - \alpha}{1 + \exp(\frac{C(f)}{\beta} - \tau)} \right), \tag{3.1}$$

где α — некоторый коэффициент, управляющий влиянием штрафа за сложность, $0 \ll \alpha < 1$, $\beta > 0$ — коэффициент строгости штрафа за сложность, τ — коэффициент, характеризующий желаемую сложность модели.

Второй множитель в (3.1) выполняет роль штрафа за слишком большую сложность суперпозиции, что позволяет выбирать более простые модели, избегая эффекта переобучения и экстремальных случаев вроде порождения интерполяционных полиномов Лагранжа. На рис. 2 приведены поверхности Q_f для различных значений β при фиксированном $\tau = 5$.

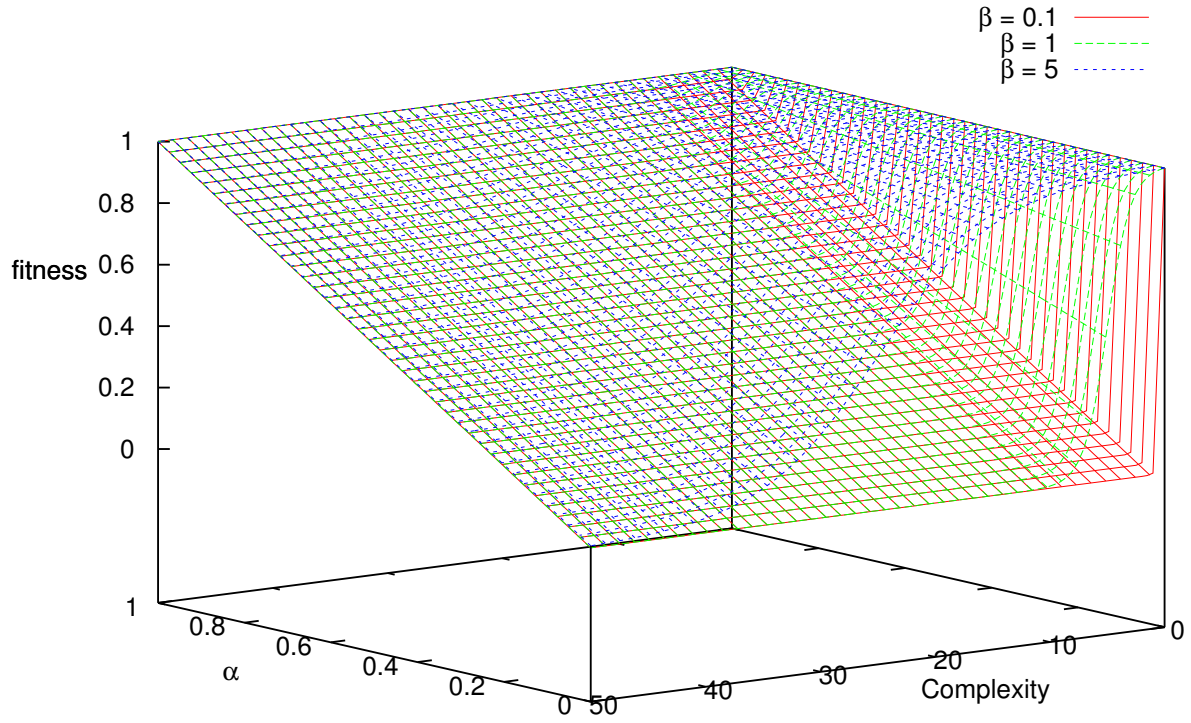


Рис. 2: Поверхности функции Q_f для некоторых β и фиксированного $\tau = 5$.

Таким образом, чем лучше результаты суперпозиции и чем она проще, тем ближе значение ее приспособленности к 1.

Алгоритм 3. Итеративный алгоритм стохастического порождения суперпозиций.

Вход:

- Множество порождающих функций G, состоящее только из унарных и бинарных функций.

- Регрессионная выборка D.
- N_{max} — максимальное число одновременно рассматриваемых суперпозиций.
- I_{max} — максимальное число итераций алгоритма.
- γ_{mut} — доля суперпозиций, подверженных случайной замене узлов их деревьев.
- γ_{cross} — доля суперпозиций, для которых выполняется случайный обмен поддеревьями.
- Прочие параметры, используемые в (3.1) и алгоритме 2.

1. Инициализируется упорядоченный набор $\mathcal{X}_f$ суперпозиций. А именно, порождается N_{max} суперпозиций алгоритмом 2.
2. Оптимизируются параметры $\boldsymbol{\omega}$ суперпозиций из $\mathcal{X}_f$ алгоритмом Левенберга-Марквардта.
3. Выполняются простейшие преобразования, упрощающие суперпозицию: например, выражения вида $0 \cdot x$ заменяются на 0.
4. Вычисляется значение Q_f для каждой еще не оцененной суперпозиции f из $\mathcal{X}_f$: для нее рассчитывается значение функционала качества S_f на выборке D, и ставится в соответствие значение Q_f в соответствии с (3.1). Для суперпозиций, при вычислении Q_f которых была хотя бы раз получена ошибка вычислений из-за несовпадения областей определений и значений, принимается $Q_f = -\infty$.
5. Набор суперпозиций $\mathcal{X}_f$ сортируется согласно их приспособленности.
6. Наименее приспособленные суперпозиции удаляются из массива $\mathcal{X}_f$ до тех пор, пока его размер не станет равен N_{max}.
7. Отбирается некоторая часть γ_{mut} наименее приспособленных суперпозиций из $\mathcal{X}_f$. У этой части происходит случайная замена одной функции или свободной переменной на другую: генерируются две случайные величины, одна из которых служит для выбора вершины дерева Γ_f, которую предстоит изменить, а другая — для выбора нового элемента для этой вершины. Замена такова, чтобы сохранилась структура суперпозиции, а именно: в случае замены функции сохраняется арность, а свободная переменная заменяется только на другую свободную переменную. Исходные суперпозиции сохраняются в массиве $\mathcal{X}_f$.

8. Повторяются шаги $4-5$.

9. Производится случайный обмен поддеревьями у γ_{cross} наиболее приспособленных суперпозиций. Вершины, соответствующие этим поддеревьям, выбираются случайным образом. При этом исходные суперпозиции сохраняются в $\mathcal{X}_f$.

10. Повторяются шаги $2-5$.

11. Проверяются условия останова: если либо число итераций больше I_{max}, либо в массиве $\mathcal{X}_f$ есть хотя бы одна суперпозиция с приспособленностью больше, чем $\hat{Q}$, то алгоритм останавливается, и результатом является наиболее приспособленная суперпозиция, иначе осуществляется переход к шагу 2.

4 Алгоритм преобразования суперпозиций по правилам

Введем несколько определений.

Определение 7. Два дерева Γ_1 и Γ_2 равны тогда и только тогда, когда они имеют одинаковую структуру и при любой упорядоченной нумерации вершин содержимое вершин с одинаковым номером совпадает.

Определение 8. Дерево Γ является поддеревом дерева Γ', если Γ' содержит поддерево, равное Γ в смысле определения 7.

Такое определение позволяет считать, что дерево является поддеревом другого дерева, даже если множества вершин этих деревьев не пересекаются.

Напомним, что f_{V_i} обозначает суперпозицию, эквивалентную поддереву вершины V_i.

Определение 9. Общая компонента дерева — поддерево $\Delta_{f'}$ дерева Γ_f суперпозиции f, встречающееся более чем один раз.

Здесь f' — подвыражение в суперпозици f, соответствующее дереву $\Delta_{f'}$.

Так как суперпозиция f и ее дерево Γ_f эквивалентны, общей компонентой дерева также можно называть суперпозицию, эквивалентную поддереву, являющемуся общей компонентой в смысле определения 9.

Например, для суперпозиции $(x+2)+(x+2)+x$ общие компоненты: x, 2, $x+2$.

Определение 10. Наибольшая общая компонента дерева — такая общая компонента $\hat{\Delta}_{f'}$, что не существует другой общей компоненты, включающей данную.

Для вышеупомянутой суперпозиции $(x+2)+(x+2)+x$ наибольшей общей компонентой является поддерево, соответствующее суперпозиции $x+2$.

Наибольших общих компонент может быть несколько. Например, для суперпозиции $(x+2)+(x+2)+2\sin(x+y)\cos(x+y)$ наибольшими общими компонентами будут подвыражения $x+2$ и $x+y$. Обозначим множество всех наибольших общих компонент графа Γ_f суперпозиции f как $\tilde{\Delta}_f$.

Введем также понятие унифицированного графа суперпозиции f:

Определение 11. Унифицированный граф $\hat{\Gamma}_f$ суперпозиции f — направленный ациклический граф, полученный из дерева Γ_f следующим итеративным методом:

1. Граф $\tilde{\Gamma}_f$ на первом шаге равен Γ_f.

2. Для графа $\tilde{\Gamma}_f$ находятся все наибольшие общие компоненты $\hat{\Delta}_{f'}$.

3. Если таких компонент не найдено, то полученный граф $\tilde{\Gamma}_f$ объявляется искомым графом $\hat{\Gamma}_f$, и алгоритм завершается.

4. Иначе выбирается компонента $\hat{\Delta}_{f'}$ с наибольшей сложностью $C(f')$. Если несколько наибольших общих компонент имеют максимальную сложность, то выбирается первая из них согласно некоторому фиксированному порядку на множестве суперпозиций.

5. Выбранная компонента остается в единственном экземпляре: удаляются все вершины V_i такие, что $f_{V_i} = f'$, кроме одной вершины V_i', и ребра, входящие в удаленные вершины, изменяются таким образом, чтобы они входили в V_i'.

То есть, в графе $\hat{\Gamma}_f$ каждая наибольшая общая компонента $\hat{\Delta}_{f'}$ присутствует не более одного раза.

Таким образом, $\hat{\Gamma}_f$ отличается от Γ_f тем, что в одну вершину могут входить несколько ребер, причем в том и только в том случае, если эта вершина соответствует поддереву, являющемуся некоторой наибольшей общей компонентой дерева Γ_f.

Теорема 3. Граф $\hat{\Gamma}_f$ по данному дереву Γ_f строится единственным образом.

Доказательство.

Утверждение напрямую следует из метода построения $\hat{\Gamma}_f$, если учесть, что не важно, какая именно вершина не была удалена на шаге 5, так как после указанного преобразования графа все такие вершины равнозначны. ■

Теорема 4. Дерево Γ_f по данному графу $\hat{\Gamma}_f$ восстанавливается единственным образом.

Доказательство.

Для того, чтобы построить дерево Γ_f по данному графу $\hat{\Gamma}_f$, достаточно найти в $\hat{\Gamma}_f$ все вершины V_i, имеющие более одной родительской вершины V_j. Каждой такой вершине V_i соответствует некоторая суперпозиция f_{V_i} и ее подграф Γ_f^i.

Добавим в граф Γ_f такое минимальное число подграфов, равных Γ_f^i, чтобы каждая вершина из V_j указывала в свой собственный подграф, равный Γ_f^i, то есть, чтобы множества вершин дочерних подграфов не пересекались.

Так как $\hat{\Gamma}_f$ построен путем выделения и объединения общих компонент дерева Γ_f, то полученный в результате указанной процедуры граф будет являться деревом, причем, эквивалентным Γ_f. ■

Из этих двух утверждений непоследственно следует следующая теорема:

Теорема 5. Между $\hat{\Gamma}_f$ и Γ_f существует взаимно однозначное преобразование.

Эта теорема позволяет говорить об эквивалентности Γ_f и $\hat{\Gamma}_f$ и о том, что взаимно однозначные преобразования для $\hat{\Gamma}_f$ являются также и взаимно однозначными преобразованиями для суперпозиции f.

Работа алгоритма преобразования суперпозиций по правилам состоит из двух этапов.

1. Находятся наибольшие равные в смысле определения 7 поддеревья, и дерево Γ_f суперпозиции f преобразуется в соответствующий унифицированный граф $\hat{\Gamma}_f$.

2. К полученному графу $\hat{\Gamma}_f$ применяются правила преобразования, уменьшающие его сложность, до тех пор, пока правила возможно применять.

Опишем эти этапы в следующих двух разделах.

4.1 Алгоритм выделения наибольших общих компонент дерева

Введем понятие пути PathTo до вершины V_i. Пусть дана некоторая лексикографическая нумерация вершин дерева Γ_f суперпозиции f. Тогда путь из корня дерева V_0 до вершины V_i — последовательность номеров вершин в соответствующем пути из корня в V_i. Будем обозначать эту последовательность как $\mathrm{PathTo}(V_i)$.

Рассмотрим некоторую суперпозицию f', являющуюся подвыражением суперпозиции f, и пусть суперпозиции f' соответствуют вершины $V_{i_1}, V_{i_2}, \ldots, V_{i_k}$. Будем обозначать множество путей в соответствующие f' вершины как $\mathbf{Paths}(f')$. То есть,

$$\mathbf{Paths}(f') = \{\mathrm{PathTo}(V_i) \mid f_{V_i} = f'\}.$$

Опишем алгоритм выделения наибольших общих компонент $\hat{\Delta}_{f'}$.

Алгоритм 4. Алгоритм выделения наибольших общих компонент графа Γ_f.

1. Пронумеруем вершины $V_i \mid i \in \{1, \ldots, C(f)\}$ в лексикографическом порядке.

2. Построим функцию NTS (анг. Node To Superposition), сопоставив каждой вершине V_i суперпозицию, имеющую корнем V_i:

   $$\mathrm{NTS} : V_i \mapsto f_{V_i} \in \mathcal{F}_{\Gamma_f} \subset \mathcal{F},$$

   где $\mathcal{F}_{\Gamma_f}$ — множество всех суперпозиций, являющихся подвыражениями суперпозиции f.

   Заметим, что функция NTS, сюръективна по построению, но она не является инъективной: могут существовать такие $i \neq j$, что $\mathrm{NTS}(V_i) = \mathrm{NTS}(V_j)$.

3. Для каждой суперпозиции f_{V_i} из множества значений NTS добавим путь к каждой из вершин, соответствующей f_{V_i}, в список путей для суперпозиции f_{V_i}.

   Таким образом, мы получили функцию STP (анг. Superposition To Paths), сопоставляющую каждой суперпозиции f', являющейся подвыражением суперпозиции f, путь PathTo до вершин V_i, ей соответствующих:

   $$\mathrm{STP} : f_{V_i} \mapsto \mathbf{Paths}(f_{V_i}) = \{\mathrm{PathTo}(V_i) \mid \mathrm{NTS}(V_i) = f_{V_i}\}.$$

4. Расположим суперпозиции f_{V_i} по возрастанию соответствующей длины пути в $\mathrm{STP}(f_{V_i})$, и для каждой суперпозиции, встречаемой по меньшей мере два раза, найдем и удалим все прочие суперпозиции, у которых пути получаются приписыванием одинакового числа номеров вершин к соответствующему пути.

Построенный алгоритм позволяет найти в суперпозиции подвыражения, встречающиеся более одного раза. При этом алгоритм позволяет не рассматривать многократно встречающиеся суперпозиции, являющиеся подвыражениями других суперпозиций.

4.2 Применение правил преобразования унифицированных графов выражения

Определим понятие правила $\mathbf{R}$ преобразования графа следующим образом:

Определение 12. Правило $\mathbf{R}$ преобразования графа — пара (L, R), где L обозначает граф-шаблон, а R — граф-замену.

Преобразование графа Γ по правилу $\mathbf{R}$ сводится к поиску изоморфного L графа в Γ и его замене на граф R.

Обозначим $\mathcal{R} = \{\mathbf{R}\}$ множество правил преобразования графа.

Рассмотримчастные случаи определения 12, возникающие при работе с унифицированными графами $\hat{\Gamma}_f$. Будем считать, что множество элементарных функций G состоит только из функций одного и двух аргументов.

Функции одного аргумента. Ограничимся рассмотрением правил вида

$$\mathbf{R} = (L, R) = (g_i, g_j) = (g_{i_1} \circ g_{i_2} \circ \cdots \circ g_{i_n}, g_{j_1} \circ g_{j_2} \circ \cdots \circ g_{j_m}) \mid n > m. \tag{4.1}$$

Иными словами, мы рассматриваем правила, заменяющие суперпозицию n функций одного аргумента на суперпозицию меньшего числа функций одного аргумента. То есть, рассматриваются правила, где сложность графа-шаблона L больше, чем сложность графа-замены R: $C(L) = C(g_i) > C(R) = C(g_j)$.

Приведем примеры таких правил:

- $(\arcsin \circ \sin, \mathrm{id})$,
- $(\log \circ \exp, \mathrm{id})$.

Применение правил вида (4.1) к графу $\hat{\Gamma}_f$ суперпозиции f сводится к поиску вершин, соответствующих функциям из L, и замене всех найденных вхождений на правую часть правила R. То есть, для каждого правила $\mathbf{R} = (L, R)$ в графе $\hat{\Gamma}_f$ ищется суперпозиция, соответствующая суперпозиции, описываемой графом-шаблоном L, и каждое найденное вхождение заменяется на суперпозицию, описываемую графом-заменой R.

Функции двух аргументов. Заметим, что используемые графы-шаблоны в этом случае представимы в виде тройки $L = (Lst, Rst, Op)$, где Op — бинарная операция, дочерними подграфами которой являются подграфы Lst и Rst. Граф-замена R является функцией от двух аргументов: $R = R(Lst, Rst)$. Эта функция возвращает граф-замену, соответствующий подграфам Lst и Rst, найденным в графе Γ_f.

Введем понятие переменной на множестве графов.

Определение 13. Переменная t_i на множестве графов Γ в графе $\hat{\Gamma}_f$ — вершина $V_j, j = j(t_i)$ в графе $\hat{\Gamma}_f$. Подстановка некоторого значения $t_i = \Gamma_0$ в граф $\hat{\Gamma}_f$ эквивалентна замене вершины t на граф Γ_0.

Говоря о графе суперпозиции f с переменной t_i, будем иметь ввиду любой граф, получаемый заменой t_i на произвольный подграф. Так, $\cos^2 t_i$ является графом суперпозиции $\cos^2(x + 2)$ при $t_i = (x + 2)$.

Lst и Rst являются шаблонами графов — это графы, содержащие некоторый набор переменных t_i. Таким образом, поиск подграфа Lst_{Γ_f} в графе Γ_f, соответствующего Lst, сводится к поиску таких значений переменных t_i, входящих в Lst, при которых граф Lst становится равен графу Lst_{Γ_f}. Аналогично происходит поиск подграфа, соответствующего Rst.

Учитывая, что для каждой тройки (Lst, Rst, Op) задается своя функция R, можно сказать, что функция R на самом деле является функцией от всех входящих в Lst и Rst переменных: $R = R(\{t_i\}, \{t_j\})$, где $\{t_i\}$ — переменные, входящие в состав графа Lst, а $\{t_j\}$ — переменные, входящие в состав графа Rst.

Если Op коммутативна, то Lst и Rst можно поменять местами. Если Op ассоциативна, то правило применимо не только к суперпозициям, соответствующим выражениям вида $Op(Lst, Rst)$, но и выражениям вида $Op(Lst, Op(Rst, t))$, где t — некоторый не участвующий в правиле подграф. В таком случае выражение вида $Op(Lst, Op(Rst, t))$ заменяется на $Op(R(Lst, Rst), t))$.

Приведем процедуру поиска соответствий для случая неассоциативной и некоммутативной Op.

1. В графе суперпозиции находятся все вершины V_i, содержащие Op.

2. Если существует набор значений переменных t_l, используемых в Lst, при подстановке которых в Lst граф Lst становится равен первому (по порядку) дочернему подграфу вершины V_i (то есть, первому аргументу функции в вершине V_i), то считается, что первый аргумент V_i соответствует шаблону, заданному Lst.

3. Аналогично проверяется, соответствует ли шаблону Rst второй аргумент функции в вершине V_i.

4. Если оба аргумента соответствуют шаблонам, и при этом значения переменных t с одинаковыми индексами равны, то считается, что подграф, соответствующий вершине V_i, подходит под правило (Lst, Rst, Op).

Случай коммутативной операции Op сводится к случаю некоммутативной неявным добавлением правил (Rst, Lst, Op) для каждого (Lst, Rst, Op). Случай ассоциативной операции Op аналогичен случаю неассоциативной, но с добавлением перебора по всем возможным комбинациям пар аргументов.

После нахождения соответствия участка графа правилу $((Lst, Rst, Op), R)$ соответствующие значения переменных, полученные в шагах $2-3$, подставляются в функцию R, и исходно найденное подвыражение заменяется на значение этой функции.

Унифицированный граф суперпозиции используется для того, чтобы избежать сравнений подграфов в шаге 4. Вместо этого проверяется, соответствуют ли одинаковые переменные одному и тому же подграфу в унифицированном графе суперпозиции.

Рассмотрим для примера правило, описывающее основное тригонометрическое тождество $\sin^2 x + \cos^2 x = 1$. В этом случае Op соответствует $+$, а Lst и Rst представлены на рис. 3 и 4 соответственно. Функция $R = R(t)$ возвращает граф из единственного узла 1.

При применении правила к суперпозиции $\cos^2(x+2) + \sin^2(x+2)$ переменная t_1 примет значение, соответствующее графу суперпозиции $x+2$, как для Lst, так и для Rst, поэтому правило будет успешно применено.

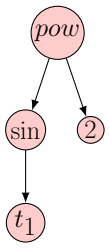


Рис. 3: Представление Lst для правила $\sin^2 x + \cos^2 x = 1$.

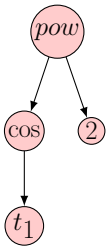


Рис. 4: Представление Rst для правила $\sin^2 x + \cos^2 x = 1$.

Заметим, что, например, правило, описывающее вынесение общего множителя за скобки, заменяющее $ax + bx$ на $(a + b)x$, неприменимо к суперпозиции типа $nx + x$ в описанном выше виде. Это связано с тем, что, например, шаблон Rst у такого правила представляет умножение константы на переменную, в то время как в суперпозиции $nx + x$ второй аргумент представляет собой просто константу.

Чтобы избежать подобной ситуации, в работе предлагается указывать каждое подобное соотношение не в виде правил переписывания графа, а в виде отношений эквивалентности, что позволяет существенно сократить количество указываемых вручную правил.

5 Метод исследования стабильности порожденной модели

Для оценки устойчивости $\mathbb{T}_{\hat{f}}$ решения $\hat{f}$ задачи (2.3), как предложено выше, фиксируется структурный вид суперпозиции $\hat{f}$ и исследуется зависимость стандартного отклонения ее коэффициентов как функция стандартного отклонения нормально распределенной случайной добавки в исходных данных.

Иными словами, выбираются значения σ_λ и σ_n, затем для этих значений генерируется выборка $\acute{D}(\sigma_n, \sigma_\lambda)$ согласно (2.4). Для этой выборки вычисляются значения коэффициентов суперпозиции $\hat{f}$, минимизирующие функционал (2.3) согласно (2.5),

методом Левенберга-Марквардта.

Данная процедура для фиксированной пары σ_λ и σ_n повторяется до достижения некоторого критерия останова (например, по количеству итераций), после которого и рассчитывается $\mathbb{T}_{\hat{f}}$.

Повторяя описанные выше шаги для различных σ_λ и σ_n, можно оценить зависимость стандартного отклонения коэффициентов суперпозиции от стандартного отклонения шума.

Из физических соображений ясно, что гладкий вид такой зависимости означает устойчивое в физическом смысле решение, тогда как отклонения от гладкости означают ту или иную ошибку в суперпозиции и могут являться свидетельством переобучения: чем меньше коэффициенты зависят от случайных шумов в данных, тем больше обобщающая способность.

Кроме того, сравнение различных суперпозиций может также производиться по критерию устойчивости в дополнение к сравнению по сложности и по значению функционала (2.3). В ряде практических приложений критерий устойчивости может иметь приоритетное значение.

6 Вычислительный эксперимент

6.1 Порождение суперпозиций

6.1.1 Синтетические данные

Восстанавливается функциональная зависимость $z = 2\cosh\frac{\sqrt{x^2+y^2}}{2}$, соответствующая образующей фигуры вращения цепной линии. При этом значения зависимой переменной z были искусственно зашумлены аддитивной добавкой из распределения $\mathcal{N}(0, 0.1)$, и соответствующая ей переменная присутствовала во множестве используемых свободных переменных. Также был искусственно добавлен один выброс — точка с координатами $(0, 0, -1)$.

В качестве функционала ошибки S_f используется сумма квадратов регрессионных остатков для данной суперпозиции f с вектором параметров $\boldsymbol{\omega}$ при регрессионной выборке D:

$$S(\boldsymbol{\omega}, f, D) = \sum_{i=1}^{N}(y_i - f(\boldsymbol{\omega}, \mathbf{x}_i))^2. \tag{6.1}$$

Значение функционала ошибки S_f при подстановке исходной незашумленной

функциональной зависимости составляет ≈ 13.28, сложность исходной суперпозиции — 16.

Таблица 1: Результаты вычислительного эксперимента для предложенного алгоритма.

N	i	Суперпозиция	S_f	$C(f)$
1	16	$0.9998\left(2.71^{\frac{\sqrt{x\cdot x+y^2}}{2.02}}+2.57^{\frac{\sqrt{x\cdot x+y^2}}{-1.94}}\right)-0.0059$	≈ 13.25	31
2	7	$1.999\cosh\frac{\sqrt{x\cdot x+y\cdot y\cdot 1.0002}}{2.002}$	≈ 13.89	16

Таблица 2: Результаты вычислительного эксперимента для алгоритма [11].

i	Суперпозиция	S_f	$C(f)$
29	$2.69^{\frac{\sqrt{x\cdot x+y\cdot y}}{2.1}}+\frac{x\cdot x}{6988}+\frac{y}{1896}-\frac{x\cdot y\cdot y}{504}+0.69+0.00079\cdot x\cdot x\cdot y\cdot y$	≈ 25.44	43

Использованные параметры алгоритма 3: $N_{max}=200, I_{max}=50, \hat{Q}=0.95, \tau=20, \alpha=0.05$, $\beta=1$, $\gamma_{mut}=\frac{1}{3}$, $\gamma_{cross}=\frac{1}{3}$. При отсутствии улучшения результатов в течение нескольких итераций подряд алгоритм 3 также завершался.

Результаты вычислительного эксперимента приведены в таблице 1. Указан номер итерации i, на которой суперпозиция была впервые получена, сама суперпозиция, среднеквадратичная ошибка (6.1) и сложность в смысле количества узлов в соответствующем графе выражения. Числовые коэффициенты в приведенных формулах и значения функционала S_f искусственно округлены до нескольких значащих цифр.

Алгоритм запускался для двух разных наборов элементарных функций. В обоих случаях элементарные функции включали в себя стандартные арифметические операции и операцию возведения в степень. Для удобства восприятия возведение в степень $\frac{1}{2}$ (и близкие ей) представлено в таблице как извлечение корня.

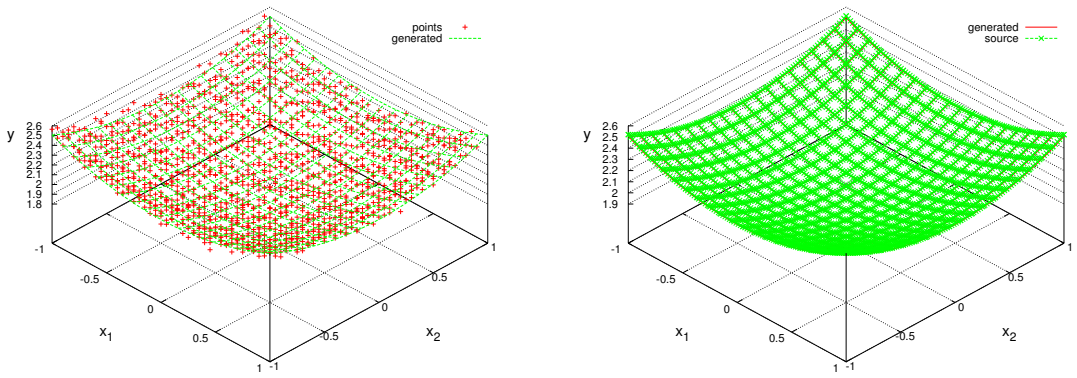
В первом случае в наборе отсутствовала функция cosh. При этом по результатам 10 запусков наилучшей суперпозицией, полученной предложенным алгоритмом, являлась функция за номером 1 из таблицы 3. Видно, что выражение в скобках близко определению $\cosh x=\frac{e^x+e^{-x}}{2}$, однако, разные значения оснований степенных функций могут затруднить экспертный анализ полученного выражения, которое само по себе является достаточно громоздким.

Во втором случае набор элементарных функций также включал в себя функ-

цию cosh, результату этого выражения соответствует суперпозиция за номером 2. Включение cosh в G позволило существенно быстрее подобрать искомую функцию, и сложность получившейся суперпозиции также существенно меньше.

Кроме того, предложенный алгоритм сравнивался с алгоритмом [11], где предложено кодировать суперпозицию бинарной строкой и применять стандартные генетические алгоритмы на бинарных строках. При этом, в множестве используемых функций также отсутствовала функция cosh.

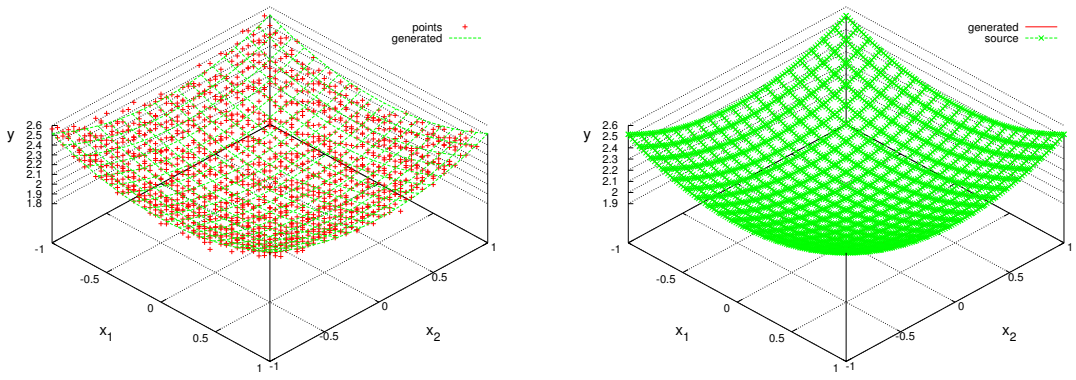
Наилучшая суперпозиция, полученная алгоритмом [11] по результатам 10 запусков, приведена в таблице 2. Полученная суперпозиция имеет существенно более высокую сложность, чем суперпозиции, перечисленные в таблице 1, что может быть связано с отсутствием этапов упрощения суперпозиции и отсутствием множителя, соответствующего сложности суперпозиции, при оценке приспособленности суперпозиции в алгоритме [11].



(а) (б)

Рис. 5: Первая порожденная суперпозиция и зашумленные точки выборки (а) и исходная зависимость (б).

На рис. 5 отображены изометрические проекции первой из приведенных в таблице 1 суперпозиций. На графике слева данная суперпозиция сравнивается с точками синтезированной зашумленной выборки, на графике справа она же приведена вместе с исходной незашумленной зависимостью. Аналогичные проекции приведены для второй суперпозиции на рис. 6.



(а) (б)

Рис. 6: Вторая порожденная суперпозиция и зашумленные точки выборки (а) и исходная зависимость (б).

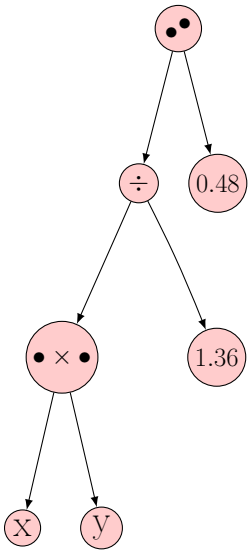
6.1.2 Моделирование волатильности опционов

Восстанавливается регрессионая зависимость волатильности опциона от его стоимости и сроков исполнения [26, 27]. Используются исторические данные о волатильности опционов Brent Crude Oil. Срок действия опциона — полгода, с 02.01.2001 по 26.06.2001, тип — право на продажу базового инструмента. Базовым инструментом в данном случае является нефть. Использовались ежедневные цены закрытия опциона и базового инструмета.

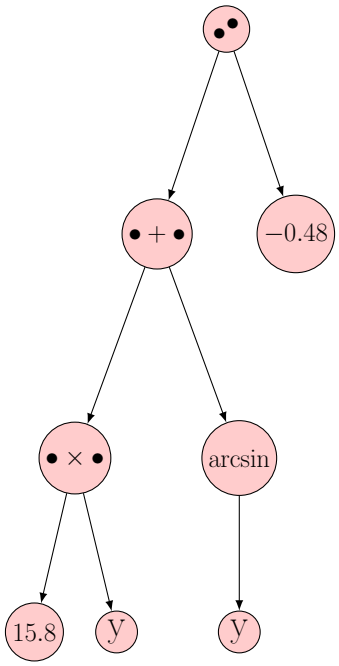
Данный инструмент имеет низкую волатильность, вследствие чего среди данных нет выбросов. В данных имеются пропуски, так как опционы с ценами, далекими от цен базового инструмента, не торговались сразу после выпуска опционов.

i	Суперпозиция	S_f	$C(f)$
9	$\left(\frac{1.36}{xy}\right)^{0.48}$	≈ 0.0182	7
14	$(15.8y + \arcsin y)^{-0.48}$	≈ 0.0208	8
13	$(yx^{0.882} + \arcsin y)^{-0.482}$	≈ 0.0178	10
8	$0.125\frac{y}{(y^2)^{0.8+y}}$	≈ 0.0171	11
14	$\frac{\frac{3.86\cdot 10^{11}+y}{y\frac{1.227\cdot 10^{11}}{x^y}-2.46\cdot 10^8}}{y\cos\left(\frac{\frac{-5.89\cdot 10^{-3}+y}{y-5.47\cdot 10^{-3}}}{\frac{y\cos y}{y}}\right)}}{y^y\cos y + xy}$	≈ 0.0092	42

Таблица 3: Результаты вычислительного эксперимента



Граф суперпозиции $\left(\frac{1.36}{xy}\right)^{0.48}$



Граф суперпозиции $(15.8y + \arcsin y)^{-0.48}$

Таблица 4: Графы результирующих суперпозиций

В ходе предварительной обработки данных выяснено, что для больших значений волатильности зависимость принимает существенно неоднозначный характер, поэтому для облегчения аналитического описания моделировалась зависимость цены от волатильности и времени.

Использованные параметры алгоритма 3: $N_{max} = 200, I_{max} = 50, \hat{Q} = 0.95, \tau = 10, \alpha = 0.05$, $\gamma_{mut} = \frac{1}{3}$, $\gamma_{cross} = \frac{1}{3}$. При отсутствии улучшения результатов в течение нескольких итераций подряд алгоритм 3 также завершался.

В таблице 3 приведены некоторые из суперпозиций, порожденных в результате работы алгоритма 3, в порядке возрастания их сложности. Указан номер итерации i, на которой суперпозиция была впервые получена, сама суперпозиция, среднеквадратичная ошибка (S_f) и сложность в смысле количества узлов в соответствующем графе выражения. Числовые коэффициенты в приведенных формулах и значения функционала S_f искусственно округлены до $2-3$ значащих цифр. Переменная x в суперпозициях обозначает срок исполнения опциона, y — волатильность.

В таблице 4 представлены графы первых двух упомянутых в таблице суперпозиций. На рисунках 7 и 8 отображены изометрическая проекция и проекция на одну из плоскостей для суперпозиции $\left(\frac{1.36}{xy}\right)^{0.48}$.

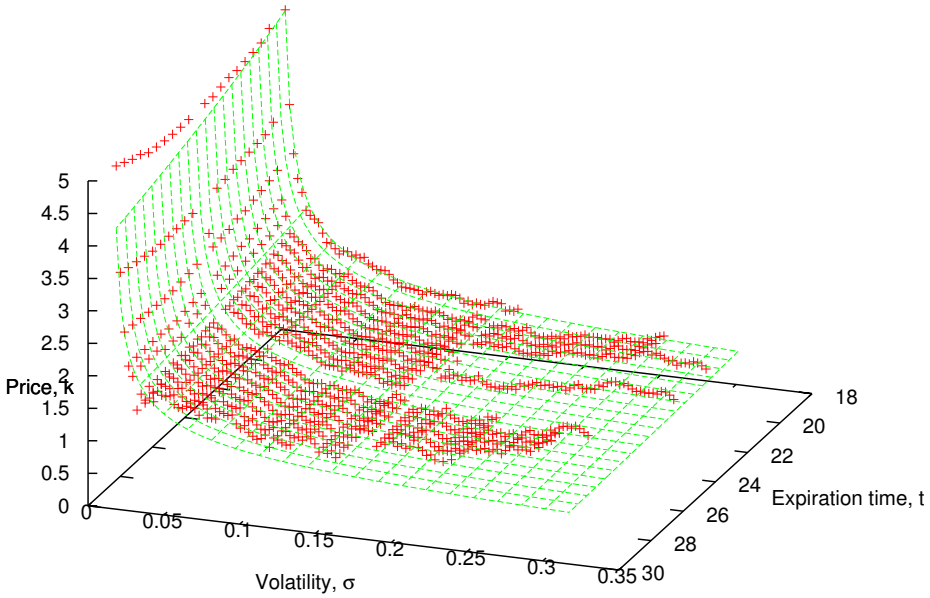


Рис. 7: Изометрическая проекция результирующей суперпозиции

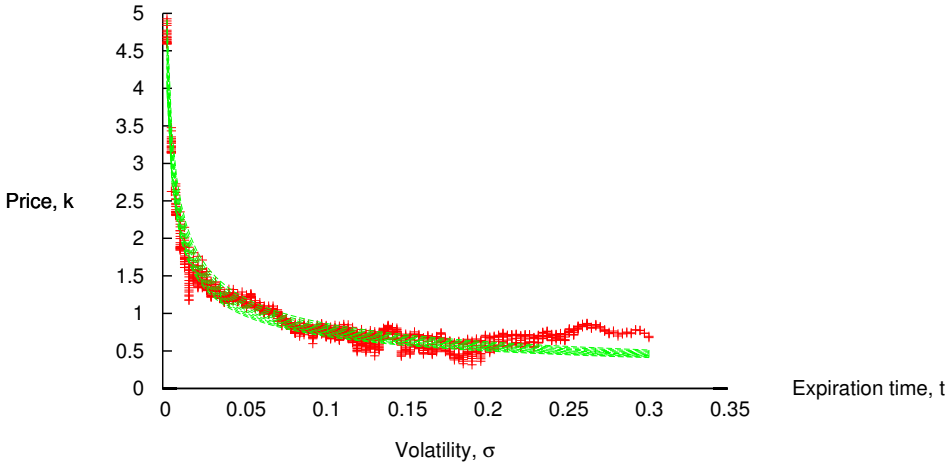


Рис. 8: Проекция результирующей суперпозиции на плоскость (σ, k)

6.2 Упрощение суперпозиций

В вычислительном эксперименте предложенный алгоритм применяется к суперпозициям, сгенерированным вручную.

Результаты применения предложенного алгоритма также сравниваются с результатами, полученными применением программы GNU Octave с пакетом Symbolic [28] к тем же выражениям. Пакет Symbolic основан на системе символьных вычислений GiNaC [29].

Суперпозиции вместе с результатами перечислены в таблице 5. Исходным суперпозициям соответствует колонка f, полученным в результате применения предложенного алгоритма суперпозициям соответствует колонка $\hat{f}$, а суперпозициям, полученным в результате применения GNU Octave — f_{oct}. Если суперпозиция не была упрощена, на соответствующем месте ставится прочерк.

При описании правил будем пользоваться следующими обозначениями для вершин различного типа:

- **NBin** Op $t1$ $t2$ обозначает вершину V_i с функцией Op от двух аргументов — $t1$ и $t2$.
- **NUn** Op t обозначает вершину V_i с функцией Op ОТ одного аргумента t.
- **LVar** x_i обозначает вершину V_i с переменной x_i.
- **LC** c обозначает вершину V_i с константой c.

Перечислим используемые элементарные функции одного аргумента из G:

Sin, Cos, Log, Tan, Asin, Acos, Atan, Id.

Используются следующие обозначения для функций двух переменных из G:

- Plus, Minus, Mul, Div: операции сложения, вычитания, умножения и деления соответственно.
- Pow: операция возведения в степень. Первый ее аргумент возводится в степень, соответствующую второму.

Используемый набор правил для функций двух аргументов включал в себя, в том числе, следующие нетривиальные правила:

1. Функция от двух констант заменяется на соответствующее значение этой функции.

$$(f, \mathbf{LC}\ a, \mathbf{LC}\ b) \mapsto \mathbf{LC}\ f(a, b).$$

2. Дистрибутивность умножения:

$$(\text{Plus}, \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ a)\ t, \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ b)\ t) \mapsto \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ (a+b))\ t.$$

3. Внесение константы под знак деления:

$$(\text{Mul}, \mathbf{LC}\ a, \mathbf{NBin}\ \text{Div}\ t\ (\mathbf{LC}\ b)) \mapsto \mathbf{NBin}\ \text{Div}\ t\ (\mathbf{LC}\ \frac{a}{b}).$$

4. Сворачивание выражений вида $at^2 + bt$ в квадрат суммы:

$$(\text{Plus}, \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ a)\ (\mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ 2)), \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ b)\ t) \mapsto$$
$$\mapsto \mathbf{NBin}\ \text{Plus}$$
$$(\mathbf{NBin}\ \text{Pow}\ (\mathbf{NBin}\ \text{Plus}\ (\mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ \sqrt{a})\ t)\ (\mathbf{LC}\ \frac{b}{2\sqrt{a}}))\ (\mathbf{LC}\ 2))$$
$$(\mathbf{LC}\ (-(\frac{b}{2\sqrt{a}})^2)).$$

5. Произведение и частное степеней:

$$(\text{Mul}, \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ a), \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ b)) \mapsto \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ (a+b)),$$

$$(\text{Div}, \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ a), \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ b)) \mapsto \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ (a-b)).$$

6. Формула двойного угла:

$$(\text{Mul}, \mathbf{NUn}\ \text{Cos}\ t, \mathbf{NUn}\ \text{Sin}\ t) \mapsto \mathbf{NBin}\ \text{Div}\ (\mathbf{NUn}\ \text{Sin}\ (\mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ 2)\ t))\ (\mathbf{LC}\ 2)$$

7. Последовательное возведение в степень t^{ab}:

$$(\text{Pow}, \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ a), \mathbf{LC}\ b) \mapsto \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ ab).$$

Следующие деревья принимаются изоморфными:

1. $\mathbf{NBin}\ \text{Minus}\ t\ (\mathbf{LC}\ b) \equiv \mathbf{NBin}\ \text{Plus}\ t\ (\mathbf{LC}\ (-b))$
2. $t \equiv \mathbf{NBin}\ \text{Mul}\ (\mathbf{LC}\ 1)\ t \equiv \mathbf{NBin}\ \text{Pow}\ t\ (\mathbf{LC}\ 1)$

Также при упрощении подвыражения вида $\mathbf{NUn}\ \text{g}\ (\mathbf{LC}\ c)$ заменяются на $\mathbf{LC}\ (g(c))$, то есть, функции одного аргумента от константы заменяются на значение соответствующих функций при подстановке этой константы. Данная подстановка выполняется рекурсивно: например, выражение $\tan\sin 0$ заменится на константу 0.

N	f	$\hat{f}$	f_{oct}	$C(f)$	$C(\hat{f})$	$C(f_{oct})$
1	$1+0(3x+4\cos^2(x+2)/(x+3))$	1	1	20	1	1
2	x^2+2x+1	$(x+1)^2$	—	9	5	9
3	$(x+2)^2+2(x+2)$	$-1+(3+x)^2$	—	11	7	11
4	$(x+2)^2+2x+4$	—	—	11	11	11
5	$(2\sin x\cos x)^2+2(\cos x\cdot 2\cdot \sin x)$	$-1+(\sin 2x)^2$	—	19	8	19
6	$(x+2)^2+1.9998(x+2)+1$	$(x+2.9999)^2+0.0001999$	—	11	7	11

Таблица 5: Результаты вычислительного эксперимента

Заметим, что предложенный алгоритм не смог упростить суперпозицию $(x+2)^2+2x+4$, вынеся 2 за скобки в подвыражении $2x+4$. Если же это было предварительно сделано, то алгоритм успешно справляется с упрощением этой суперпозиции.

В то же время рекурсивное применение правил, учет ассоциативности и коммутативности функций в предложенном алгоритме позволяют упрощать даже сложные выражения, такие как выражение 5 в таблице 5.

Заметим, что алгоритм не допускает отклонений констант от заданных правилами значений. Так, выражение 6 с хорошей точностью равно $(x+3)^2$, однако алгоритм не выполнил соответствующих округлений. Соответствующие округления могут быть выполнены отдельной процедурой.

Из результатов эксперимента видно, что предложенный алгоритм показывает существенно лучшие результаты, чем GNU Octave/Symbolic, способный упростить лишь самые простые случаи типа умножения на ноль.

6.3 Восстановление дисперсионной зависимости полимеров и исследование ее стабильности

В вычислительном эксперименте используются данные, полученные в ходе изучения возможности определения состава смеси прозрачных полимеров по суммарной дисперсионной зависимости, если известна экспериментальная зависимость дисперсии для каждого конкретного полимера. Рассматривается два полимера, для каждого из которых имеется 17 экспериментальных точек, соответствующих коэффициенту преломления n при разных значениях длины волны λ. Значения приведены в таблице 6.

Предполагается, что дисперсионные свойства полимеров описываются одной и

λ, нм	Первый полимер	Второй полимер
435.8	1.36852	1.35715
447.1	1.36745	1.35625
471.3	1.36543	1.35449
486.1	1.36446	1.35349
501.6	1.36347	1.35275
546.1	1.36126	1.35083
577.0	1.3599	1.34968
587.6	1.3597	1.34946
589.3	1.35952	1.34938
656.3	1.35767	1.34768
667.8	1.35743	1.34740
706.5	1.35652	1.34664
750	1.35587	1.34607
800	1.35504	1.34544
850	1.3544	1.34487
900	1.35403	1.34437
950	1.35364	1.34407

Таблица 6: Экспериментальные значения коэффициентов преломления.

той же функциональной зависимостью, так как подчиняются одним и тем же физическим закономерностям. Поэтому сначала получена суперпозиция $\hat{f}$, минимизирующая (3.1) с учетом (2.3) для первого полимера, а затем для каждого из полимеров находятся соответствующие векторы параметров $\hat{\boldsymbol{\omega}}_{\hat{f}}$ и оценивается устойчивость полученного решения.

Разделение на обучающую и контрольную выборку не производилось, однако переобучения удается избежать и без такого разделения, опираясь целиком на штраф за сложность.

Из физических соображений следует [30], что зависимость $n(\lambda)$ должна выражаться суммой четных степеней длины волны, поэтому множество элементарных функций состоит из стандартных операций сложения и умножения:

$$g_1(x_1, x_2) = x_1 + x_2,$$

$$g_2(x_1, x_2) = x_1 x_2,$$

а также из функции

$$g_3(\lambda, p) = \frac{1}{\lambda^{2p}}.$$

В ходе вычислительного эксперимента константы, меньшие 10^{-7}, заменялись на 0.

В результате применения описанного выше алгоритма со значениями $\alpha = 0.05$, $\tau = 10$ получена следующая суперпозиция (константы округлены до пятой значащей цифры):

$$f(\lambda) = 1.3495 + \frac{3.5465 \cdot 10^3}{\lambda^2} + \frac{2.023 \cdot 10^3}{\lambda^4}, \tag{6.2}$$

со сложностью 13, среднеквадратичной ошибкой $2.4 \cdot 10^{-8}$ и значением $Q_f \approx 0.095$. Длины волн выражаются в нанометрах.

Отметим, что обычно в приложениях учитывают только квадратичный член, а более высокими степенями пренебрегают. Величина поправки, вносимой в результирующее значение суперпозиции последним слагаемым, указывает на полное согласие полученных результатов с принятой практикой.

Влияние штрафа за сложность. Исследуем, как влияет добавление нечетных степеней на результат решения задачи (3.1), заменив функцию g_3 в порождающем наборе на

$$g_3(\lambda, p) = \frac{1}{\lambda^p}.$$

Следует отметить, что при тех же $\alpha = 0.05$ и $\tau = 10$ результирующей функцией остается (6.2).

Увеличим τ до 30. Получим следующую формулу (константы округлены до третьей значащей цифры):

$$n(\lambda) = 1.34 + \frac{11.6}{\lambda} + \frac{17.37}{\lambda^2} + \frac{0.0866}{\lambda^3} + \frac{2.95 \cdot 10^{-4}}{\lambda^4} + \frac{8.54 \cdot 10^{-7}}{\lambda^5}, \tag{6.3}$$

сложность которой составляет 31, и для которой среднеквадратичная ошибка на выборке составляет $\approx 3.9 \cdot 10^{-9}$, а значение $Q_f \approx 0.31$.

Иными словами, при большей желаемой сложности, регулируемой параметром τ, выигрывает более сложная (а в данном случае и физически некорректная) модель, которая лучше описывает экспериментальные данные.

Как и следовало ожидать, чрезмерное увеличение τ ведет к переобучению.

SVM. В качестве базового алгоритма используется SVM-регрессия с RBF-ядром [31]. Параметр γ ядра подбирался по методу скользящего контроля, наилучшим результатом является комбинация из 15 опорных векторов с $\gamma \approx 2 \cdot 10^{-6}$, при этом

среднеквадратичная ошибка при кросс-валидации с тестовой выборкой, содержащей по 2 объекта, составляет $8.96 \cdot 10^{-8}$. Однако, проинтерпретировать полученную решающую функцию не представляется возможным.

Исследование стабильности решения. Для оценки стабильности полученной суперпозиции фиксировался структурный вид суперпозиции (6.2) и исследовалась зависимость стандартного отклонения ее коэффициентов от стандартного отклонения нормально распределенного случайного шума в исходных данных описанным выше методом. Критерием останова в нем являлось достижение 10000 итераций для каждой пары $(\sigma_\lambda, \sigma_n)$.

В таблице 7 представлены поверхности уровня дисперсии для первого, второго и третьего коэффициентов каждого из полимеров соответственно.

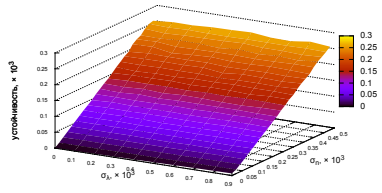
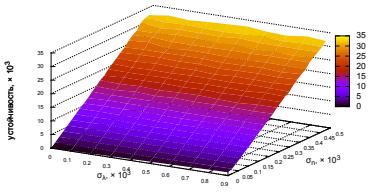
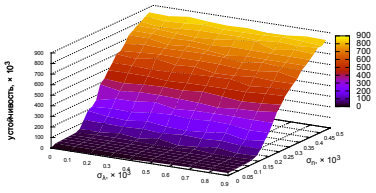
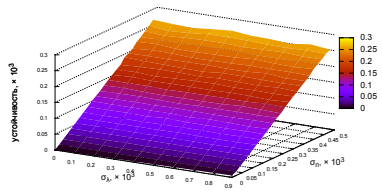
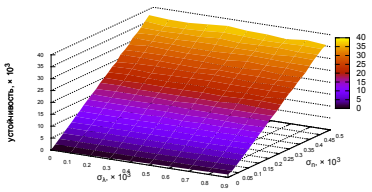
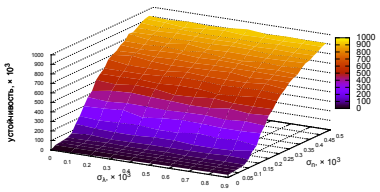
	Коэффициент 1	Коэффициент 2	Коэффициент 3
Полимер 1			
Полимер 2			

Таблица 7: Поверхности дисперсии для формулы (6.2).

Полимер	Коэффициент 1	Коэффициент 2	Коэффициент 3	MSE
1	1.34946	3558.95	1924.33	$2.2 \cdot 10^{-8}$
2	1.34047	3118.84	1578.59	$1.4 \cdot 10^{-8}$
Разность	$6.71 \cdot 10^{-3}$	$1.41 \cdot 10^{-1}$	$2.2 \cdot 10^{-1}$	

Таблица 8: Значения коэффициентов для формулы (6.2) и их относительная разность.

Из графиков видно, что от шума, накладываемого на значения длины волны, дисперсия значений первого и второго коэффициентов практически не зависит в достаточно широком диапазоне точности определения длины волны, представляющем

Коэфф.	$\frac{\sigma_\lambda}{\lambda} = 2 \cdot 10^{-4}; \frac{\sigma_n}{n} = 2 \cdot 10^{-5}$	$\frac{\sigma_\lambda}{\lambda} = 6 \cdot 10^{-4}; \frac{\sigma_n}{n} = 6 \cdot 10^{-5}$	$\frac{\sigma_\lambda}{\lambda} = 9 \cdot 10^{-4}; \frac{\sigma_n}{n} = 2 \cdot 10^{-4}$
1	$1.22 \cdot 10^{-5}$	$3.59 \cdot 10^{-5}$	$1.19 \cdot 10^{-4}$
2	$1.48 \cdot 10^{-3}$	$4.38 \cdot 10^{-3}$	$1.44 \cdot 10^{-2}$

Таблица 9: Значения стандартного отклонения для коэффициентов формулы (6.2) для первого полимера в зависимости от относительных дисперсий.

практический интерес. В то же время дисперсия значений первого коэффициента зависит от дисперсии шума коэффициента преломления практически линейно, тогда как для второго коэффициента после некоторого характерного значения зависимость становится слабой.

Практическая интерпретация этих результатов — при построении прибора для измерения дисперсии такого типа полимеров в их полосе прозрачности значительное внимание следует уделять точности измерения коэффициента преломления, тогда как измерения длины волны могут быть неточны вплоть до нескольких нанометров. Кроме того, предложенный метод прямо указывает, на каких интервалах шума каким будет выигрыш в точности определения параметров регрессионной модели в зависимости от увеличения точности измерений.

Принципиально важно, что значения стандартного отклонения параметров регрессионной модели существенно меньше разности между самими значениями этих параметров (см. таблицы 8 и 9), что означает, в частности, что полимеры могут быть различены даже не очень точным рефрактометром.

Стабильность некорректного решения. Аналогично исследуем стабильность решения (6.3). Приведем только графики зависимости первых трех коэффициентов, см. таблицу 10.

Из графиков видно, что в случае формулы (6.3) стандартное отклонение соответствующих параметров существенно превышает таковое для (6.2). В частности, второй, третий и четвертый коэффициенты имеют стандартное отклонение, на порядки превышающее характерные значения самих коэффициентов.

Данные результаты свидетельствуют о переобучении, и что полученная модель не может быть использована для надежного приближения экспериментальных данных ввиду большой чувствительности к шумам.

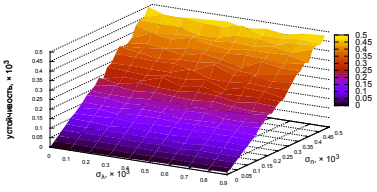
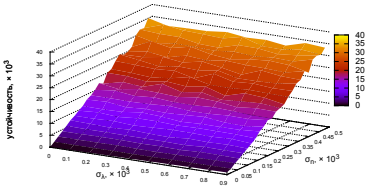
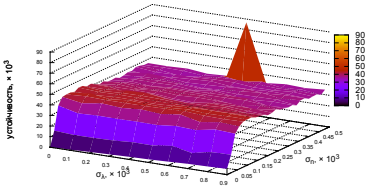
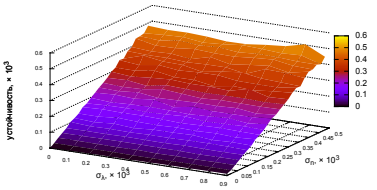
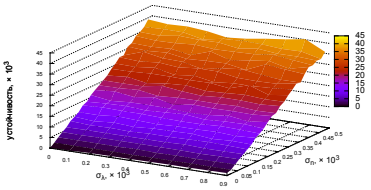
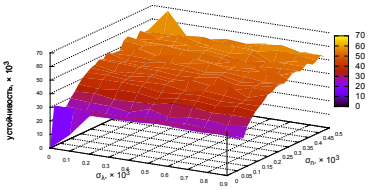
	Коэффициент 1	Коэффициент 2	Коэффициент 3
Полимер 1			
Полимер 2			

Таблица 10: Поверхности дисперсии для формулы (6.3).

6.3.1 Сходимость к классическому случаю

Рассмотрим случай, когда зависимость линейна:

$$y = ax + b,$$

и с учетом ошибок измерений представима в виде

$$y_i = ax_i + b + \xi_i \mid i \in \{1, \dots, n\},$$

где ошибки ξ_i независимы, $E(\xi_i) = 0; D(\xi_i) = \sigma^2$ [22]. То есть, рассматривается случай независимости ошибки измерения от точки измерения, при этом независимая переменная измеряется точно.

Перейдем к представлению

$$y_i = a(x_i - \overline{x}) + b + \xi_i \mid i \in \{1, \dots, n\},$$

для которого, согласно [22], случайные величины a и b независимы и нормально распределены, и, кроме того, их дисперсии выражаются известными соотношениями:

$$D(a) = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2}. \tag{6.4}$$

$$D(b) = \frac{\sigma^2}{n}. \tag{6.5}$$

Рассмотрим, насколько результаты предложенного метода отличаются от значений, полученных согласно (6.4) и (6.5). Для этого исследуем зависимость относительной разности между этими значениями и эмпирическими значениями устойчивости

от числа итераций N:

$$\delta_1 = \frac{|\mathbf{T}_y^N(1) - D(a)|}{D(a)},$$

$$\delta_2 = \frac{|\mathbf{T}_y^N(2) - D(b)|}{D(b)}.$$

Соответствующие графики для функции $y = 2x + 1 + \xi_i$ на интервале $x \in [0; 10]$ при различном количестве порожденных точек (10 и 50) приведены на рис. 9 для значений $D(\xi_i)$, равных 1 и 10. По оси x отложено число итераций в тысячах (до 10^7), по y — значения δ_1 и δ_2.

На рис. 10 и 11 приведены отдельные графики, соответствующие интервалам от 0 до $5 \cdot 10^5$ итераций и от 10^6 до 10^7 итераций соответственно. Рис. 10 демонстрирует сходимость в начале работы алгоритма, в то время как рис. 11 показывает характер сходимости при достаточно больших N.

Из графиков видно, что значения разности стабилизируются в районе $(1.5 \div 3) \cdot 10^6$ итераций и не демонстрируют явной зависимости от числа точек или дисперсии погрешности.

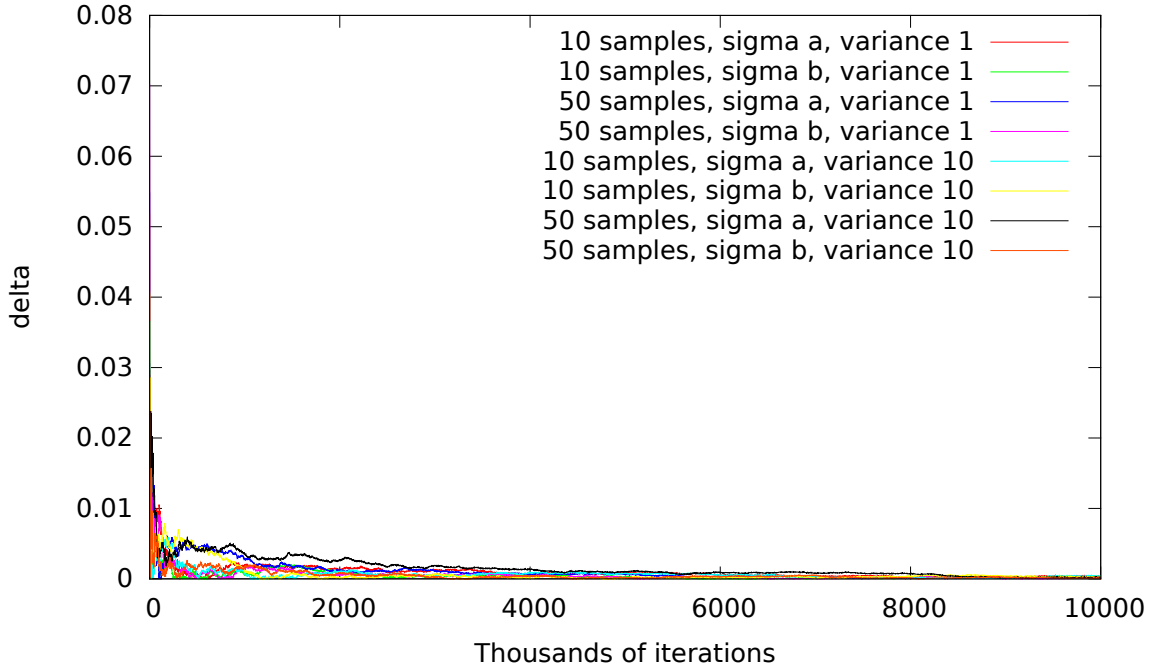


Рис. 9: График зависимости δ от числа итераций (от 0 до 10^7 итераций).

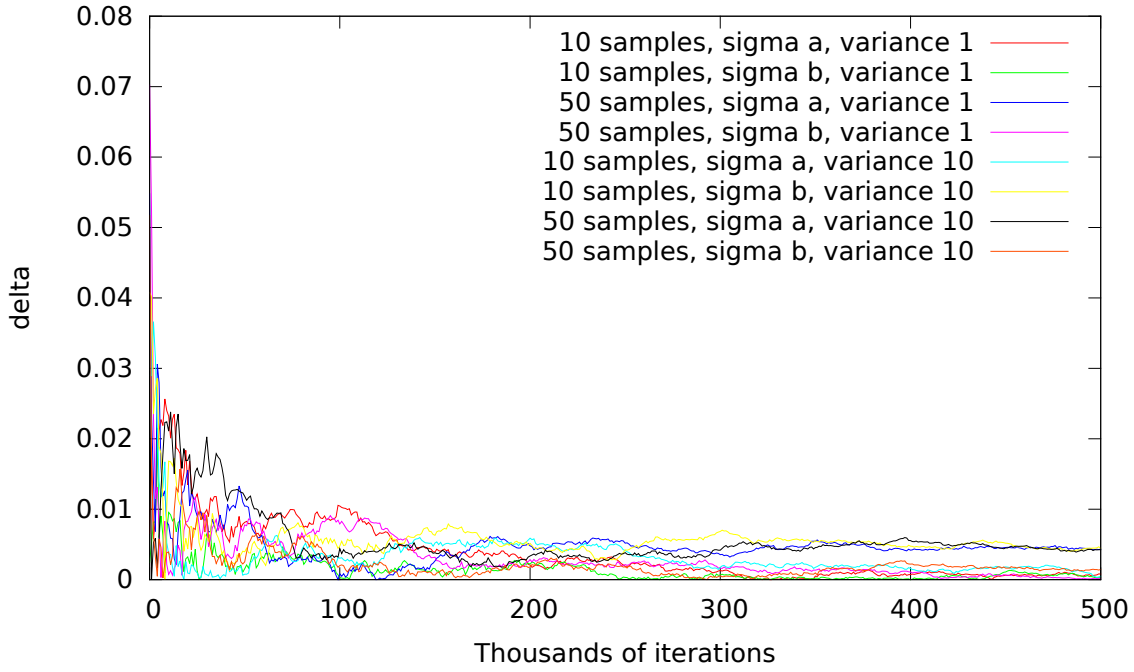


Рис. 10: График зависимости δ от числа итераций (от 0 до $5 \cdot 10^5$ итераций).

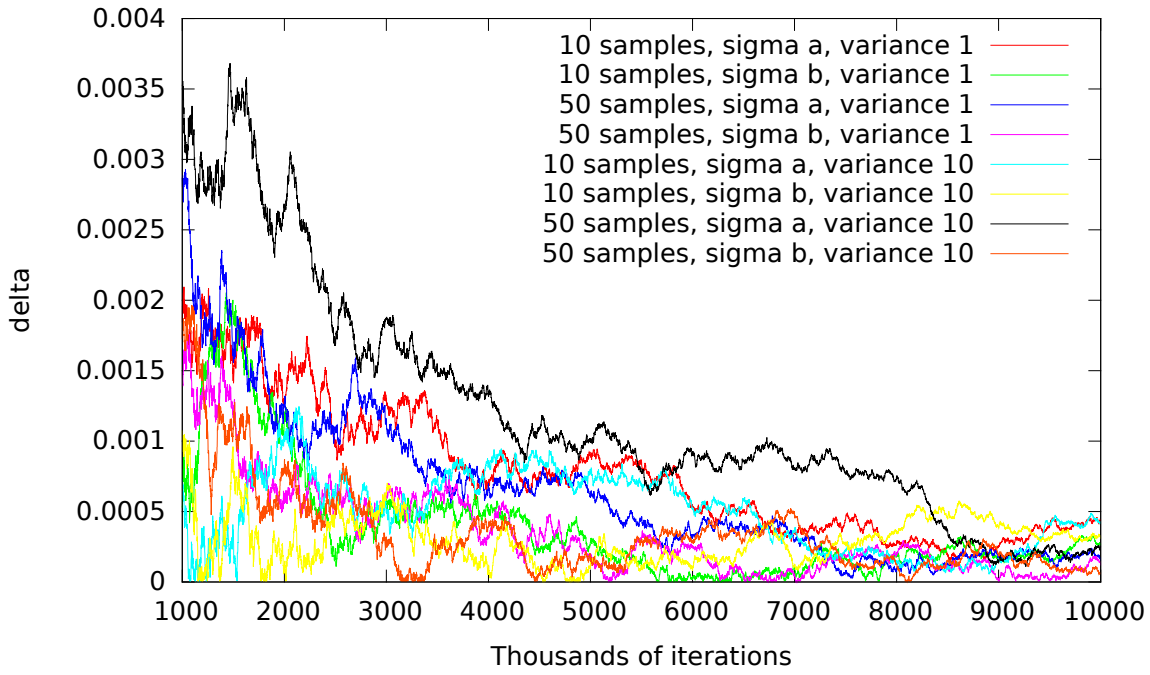


Рис. 11: График зависимости δ от числа итераций (от 10^6 до 10^7 итераций).

7 Заключение

Предложенный в [6] алгоритм позволяет получить интерпретируемую аналитическую формулу, описывающую зависимость коэффициента преломления среды от

длины волны. Введенный штраф за сложность позволяет избежать переобучения без использования методов вроде скользящего контроля, и, таким образом, отпадает необходимость в контрольной выборке.

Хотя другие алгоритмы, такие как SVM-регрессия, могут демонстрировать более высокое качество приближения данных, их результаты неинтерпретируемы и не защищены от переобучения «по построению», поэтому требуют разделения выборки на обучающую и контрольную. Кроме того, их структурные параметры так же требуют оценки по методам вроде кросс-валидации.

Предложенный в настоящей работе метод оценки стабильности решения позволяет исследовать вклад различных членов результирующей суперпозиции и зависимость изменения этих членов от случайных шумов во входных данных. В частности, в прикладных областях данный метод позволяет выявить, какие именно элементы признакового описания объектов в генеральной совокупности наиболее чувствительны к шуму. Кроме того, для корректных с экспертной точки зрения решений оказывается, что они стабильны, в то время как некорректные результаты нестабильны.

Список литературы

[1] Павловский, Ю. Н.: *Имитационные модели и системы.* М.: Фазис, 2000.

[2] Davidson, J. W., Savic, D. A., and Walters, G. A.: *Symbolic and numerical regression: experiments and applications.* In John, Robert and Birkenhead, Ralph (editors): *Developments in Soft Computing*, pages 175–182, De Montfort University, Leicester, UK, 29-30 6 2000. 2001. Physica Verlag, ISBN 3-7908-1361-3.

[3] Sammut, C. and Webb, G. I.: *Symbolic regression.* In Sammut, Claude and Webb, Geoffrey I. (editors): *Encyclopedia of Machine Learning*, page 954. Springer, 2010, ISBN 978-0-387-30768-8. `http://dx.doi.org/10.1007/978-0-387-30164-8`.

[4] Strijov, V. and Weber, G. W.: *Nonlinear regression model generation using hyperparameter optimization.* Computers & Mathematics with Applications, 60(4):981–988, 2010. `http://dx.doi.org/10.1016/j.camwa.2010.03.021`.

[5] Стрижов, В. В.: *Методы индуктивного порождения регрессионных моделей.* Препринт ВЦ РАН им. А. А. Дородницына. — М., 2008.

[6] Рудой, Г. И. и Стрижов, В. В.: *Алгоритмы индуктивного порождения суперпозиций для аппроксимации измеряемых данных.* Информатика и ее применения, 7(1):44–53, 2013.

[7] Marquardt, D. W.: *An algorithm for least-squares estimation of non-linear parameters.* Journal of the Society of Industrial and Applied Mathematics, 11(2):431–441, 1963.

[8] More, J. J.: *The Levenberg-Marquardt algorithm: Implementation and theory.* In *G.A. Watson*, Lecture Notes in Mathematics 630, pages 105–116. Springer-Verlag, Berlin, 1978. Cited in Åke Björck's bibliography on least squares, which is available by anonymous ftp from `math.liu.se` in `pub/references`.

[9] Nichael Lynn Cramer: *A representation for the adaptive generation of simple sequential programs.* In Grefenstette, John J. (editor): *Proceedings of the First International Conference on Genetic Algorithms and Their Applications.* Lawrence Erlbaum Associates, Publishers, 1985.

[10] Тырсин, А.Н.: *Об эквивалентности знакового и наименьших модулей методов построения линейных моделей.* Обозрение прикладной и промышленной математики, 12(4):879–880, 2005.

[11] Zelinka, I., Oplatkova, Z., and Nolle, L.: *Analytic programming — symbolic regression by means of arbitrary evolutionary algorithms*, 2008. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.116.9558;http://ducati.doc.ntu.ac.uk/uksim/journal/vol-6/no.9/paper5.pdf`.

[12] Soule, Terence and Foster, James A.: *Support for multiple causes of code growth in GP.* `http://citeseer.ist.psu.edu/cache/papers/cs/1832/http:zSzzSzwww.ai.mit.eduzSzpeoplezSzunamayzSzicga-ws-paperszSzsoule.pdf/support-for-multiple-causes.pdf`, Position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97, 1997.

[13] Soule, Terence and Foster, James A.: *Removal bias: a new cause of code growth in tree based evolutionary programming.* In *1998 IEEE International Conference on Evolutionary Computation*, pages 781–786, Anchorage, Alaska, USA, 5-9 5 1998. IEEE Press, ISBN 0-7803-4869-9. `http://citeseer.ist.psu.edu/313655.html`.

[14] Streeter, Matthew J.: *The root causes of code growth in genetic programming.* In Ryan, Conor, Soule, Terence, Keijzer, Maarten, Tsang, Edward, Poli, Riccardo, and Costa, Ernesto (editors): *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 443–454, Essex, 14-16 4 2003. Springer-Verlag, ISBN 3-540-00971-X. `http://www.cs.cmu.edu/~matts/Research/mstreeter_eurogp_2003.pdf`.

[15] Carette, Jacques: *Understanding expression simplification*, 2004. `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.8442;http://www.cas.mcmaster.ca/~carette/publications/simplification.pdf`.

[16] Ehrig, H., Ehrig, G., Prange, U., and Taentzer, G.: *Fundamentals of Algebraic Graph Transformation.* Springer, 2006.

[17] Ehrig, H. and Engels, G.: *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1-3. World Scientific Publishing, 1997.

[18] Mori, Naoki, McKay: *Equivalent decision simplification: A new method for simplifying algebraic expressions in genetic programming.* 2007.

[19] Sasaki, T.: *Simplification of algebraic expression by multiterm rewriting rules.* In *Proceedings of the 1986 Symposium on Symbolic and Algebriaic Computation*, pages 115–120, 7 21-23 1986.

[20] Stoutemyer, David R.: *Ten commandments for good default expression simplification.* J. Symb. Comput, 46(7):859–887, 2011. `http://dx.doi.org/10.1016/j.jsc.2010.08.017`.

[21] Stoutemyer, David R.: *Simplifying products of fractional powers of powers.* `http://arxiv.org/abs/1203.1350`, Comment: 34 pages. 17 tables. Includes Mathematica rewrite rules. To appear in Communications in Computer Algebra.

[22] Ватутин, В. А., Ивченко, Г. И., Медведев, Ю. И., и Чистяков, В. П.: *Теория вероятностей и математическая статистика в задачах.* Дрофа, 3 редакция, 2005.

[23] Малышев, В. И.: *Введение в экспериментальную спектроскопию.* Наука, 1979.

[24] Зайдель, И. Н.: *Техника и практика спектроскопии.* Наука, 1972.

[25] Битюцков, В. И., Войцеховский, М. И., и Иванов, А. Б.: *Математическая энциклопедия*, том 4. М.: Советская Энциклопедия, 1984.

[26] Daglish, T., Hull, J., and Suo, W.: *Volatility surfaces: Theory, rules of thumb, and empirical evidence*. Quantitative Finance, 7(5):507–524, 2007.

[27] Стрижов, В. В. и Сологуб, Р. А.: *Индуктивное порождение регрессионных моделей предполагаемой волатильности для опционных торгов*. Вычислительные технологии, 14(5):102–113, 2009.

[28] *Symbolic package for gnu octave.* `http://octave.sourceforge.net/symbolic/index.html`.

[29] *Ginac.* `http://www.ginac.de/`.

[30] Серова, Н. В.: *Полимерные оптические материалы.* Научные основы и технологии, 2011.

[31] Вапник, В. Н.: *Восстановление зависимостей по эмпирическим данным.* М.: Наука, 1979.