Rudoy G.I., Strijov V.V. Simplification of superpositions of primitive functions with graph rule-rewriting // Intellectual Information Processing. Conference proceedings, 2012 : 140-143.

Упрощение суперпозиций элементарных функций при помощи преобразований графов по правилам*

Рудой Г. И.[1]***, Стрижов В. В.***[2]
rudoy@forecsys.ru
[1]Москва, Московский физико-технический институт; [2]Москва, ВЦ РАН

Предлагается алгоритм упрощения суперпозиций существенно нелинейных регрессионных моделей. Данный алгоритм является 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. Приводятся результаты вычислительного эксперимента для набора синтетических данных.

Введение

Предлагается алгоритм упрощения суперпозиций, основанный на преобразовании графов по правилам. Ранее суперпозиции представлялись в виде соответствующего им дерева, над которым и оперировали предлагавшиеся алгоритмы [1, 2]. 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 — сложное подвыражение. Рассмотрена задача нахождения общих подвыражений в исходном дереве суперпозиции и задания правил упрощения на множестве подобных ациклических графов.

Постановка задачи

Пусть задано множество $G = \{g_i\}$ элементарных функций. Для каждой функции $g_i \colon \mathbb{X}_i \to \mathbb{Y}_i$ задана область определения $\mathbb{X}_i$ и область значения $\mathbb{Y}_i$. Для большей общности будем считать, что константы являются нуль-арными (нульместными) функциями — функциями, не имеющими аргументов. Пусть также задана некоторая суперпозиция f элементарных функций $g_i \in G$.

Определение 1. *Сложность $C(f)$ суперпозиции f — число элементарных функций, констант и свободных переменных, учитывающихся столько раз, сколько они встречаются в суперпозиции.*

Например, сложность суперпозиции $x + y + y$ равна 5, как и у суперпозиции $x + 2y$.

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

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

где $\mathcal{F}$ — множество всех суперпозиций, составленных из элементарных функций $g \in G$, а $\mathcal{F}_f \subset \mathcal{F}$ — множество всех изоморфных f суперпозиций.

Работа выполнена при финансовой поддержке РФФИ, проект № 10-07-00422.

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

Алгоритм преобразования суперпозиций по правилам

Каждой суперпозиции f сопоставим дерево Γ_f, строящееся следующим образом:

- в вершинах V_i дерева Γ_f находятся соответствующие элементарные функции $g_s, s = s(i)$;
- число дочерних вершин V_j у некоторой вершины V_i равно арности соответствующей функции g_s;
- порядок смежных некоторой вершине V_i вершин соответствует порядку аргументов соответствующей функции $g_{s(i)}$;
- в листьях дерева Γ_f находятся свободные переменные x_i либо числовые параметры ω_i.

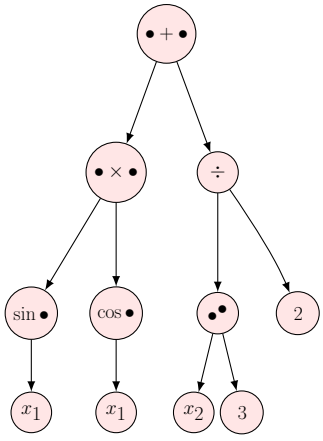

Рис. 1. Дерево выражения $\sin x_1 \cos x_1 + x_2^3/2$.

Вычисление значения суперпозиции f в некоторой точке $\mathbf{x} = \{x_i\}$ эквивалентно подстановке соответствующих значений свободных переменных x_i в дерево Γ_f. Кроме того, по построению дерева Γ_f сложность $C(f)$ суперпозиции f равна числу узлов в дереве Γ_f.

Определение 2. *Если деревья Γ_1 и Γ_2 состоят из единственной вершины со свободной переменной или параметром, то они равны тогда и только тогда, когда вершины соответствуют одинаковым переменным или параметрам соответственно.*

Если вершины V_1 в корне дерева Γ_1 и V_2 в корне дерева V_2 соответствуют одним и тем же N-арным функциям, то деревья Γ_1 и Γ_2 равны тогда и только тогда, когда поддеревья вершин V_1 и V_2 попарно равны друг другу, и порядок этих поддеревьев деревьев также равен друг другу.

Во всех прочих случаях деревья Γ_1 и Γ_2 не равны.

Отметим важное свойство деревьев суперпозиций: 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}.

Определение 3. *Общая компонента дерева — поддерево $\Delta_{f'}$ дерева Γ_f суперпозиции f, встречающееся более чем один раз.*

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

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

Определение 4. *Наибольшая общая компонента дерева — такая общая компонента $\hat{\Delta}_{f'}$, что не существует другой общей компоненты, включающей данную.*

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

Наибольших общих компонент может быть несколько. Обозначим множество всех наибольших общих компонент $\hat{\Delta}_{f'}$ графа Γ_f суперпозиции f как $\tilde{\Delta}_f$.

Определение 5. *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.

Можно показать, что граф $\hat{\Gamma}_f$ по данному дереву Γ_f строится единственным образом, и дерево Γ_f по данному графу $\hat{\Gamma}_f$ восстанавливается единственным образом. Отсюда следует, что между $\hat{\Gamma}_f$ и Γ_f существует взаимно однозначное преобразование.

Таким образом графы Γ_f и $\hat{\Gamma}_f$ эквивалентны, и 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 2 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.

Алгоритм выделения наибольших общих компонент дерева

Пусть дана некоторая 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 $V_{i_1},\ldots,V_{i_k}$. Будем обозначать множество путей в соответствующие f' вершины как $\mathbf{Paths}(f')$. То есть

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

Алгоритм 1. Выделение наибольших общих компонент графа Γ_f.

1. Пронумеруем вершины V_i, $i \in \{1,\ldots,C(f)\}$ в лексикографическом порядке.
2. Построим функцию NTS (node to superposition), сопоставив каждой вершине V_i суперпозицию, имеющую корнем V_i:

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

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

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

3. Для каждой суперпозиции f_{V_i} из $\mathcal{F}_{\Gamma_f}$ 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:

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

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, не рассматривая их подвыржаения.

Алгоритм применения правил к унифицированному графу суперпозиции

Определение 6. *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.

Рассмотрим, частные случаи определения 6, 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

$$\mathbf{R} = (L, R) = (g_i, g_j) = \\ = (g_{i_1} \dots g_{i_n}, g_{j_1} \dots g_{j_m}) \mid n > m. \quad (1)$$

Иными словами, мы рассматриваем правила, заменяющие суперпозицию n функций одного аргумента на суперпозицию меньшего числа функций одного аргумента. Примерами таких правил могут быть $(\arcsin \circ \sin, \mathrm{id})$ и $(\log \circ \exp, \mathrm{id})$.

Применение правил вида (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.

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

Определение 7. *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 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 $\mathrm{Op}(\mathrm{Lst}, \mathrm{Rst})$, но и выражениям вида $\mathrm{Op}(\mathrm{Lst}, \mathrm{Op}(\mathrm{Rst}, t))$, где t — некоторый не участвующий в правиле подграф. В таком случае выражение вида $\mathrm{Op}(\mathrm{Lst}, \mathrm{Op}(\mathrm{Rst}, t))$ заменяется на $\mathrm{Op}(R(\mathrm{Lst}, \mathrm{Rst}), t))$.

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.
4. Если оба аргумента соответствуют шаблонам, и при этом значения переменных t с одинаковыми индексами равны, то считается, что под-

граф, соответствующий вершине V_i, подходит под правило $(\mathrm{Lst}, \mathrm{Rst}, \mathrm{Op})$.

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

После нахождения соответствия участка графа правилу $((\mathrm{Lst}, \mathrm{Rst}, \mathrm{Op}), R)$ 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 соответствие одинаковых переменных одному и тому же подграфу в унифицированном графе.

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 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, предлагается указывать каждое подобное соотношение не в виде правил преобразования графа, а в виде отношений эквивалентности, что позволяет существенно сократить количество указываемых правил.

Вычислительный эксперимент

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 к тем же выражениям.

Суперпозиции перечислены в таблице 1, результы перечислены 2. Исходным суперпозициям соответствует колонка f, суперпозиции, полученные в результате применения предложенного алгоритма обозначены $\hat{f}$, а в результате Octave — f_o (прочерк означает, что исходное выражение не упрощено).

Используемый набор правил для функций двух аргументов включал в себя следующие правила:

1. Бинарная функция от двух констант (или унарная функция от одной константы) заменяется на соответствующее значение этой функции.
2. Дистрибутивность умножения.
3. Внесение константы под знак деления:

$$a \cdot \frac{t}{b} \equiv \frac{t}{\frac{b}{a}}.$$

Таблица 1. Исходные суперпозиции.

N	f	$C(f)$
1	$1 + 0(3x + 4\cos^2(x + 2)/(x + 3))$	20
2	$x^2 + 2x + 1$	9
3	$(x + 2)^2 + 2(x + 2)$	11
4	$(x + 2)^2 + 2x + 4$	11
5	$(2\sin x \cos x)^2 + 2(\cos x \cdot 2 \cdot \sin x)$	19
6	$(x + 2)^2 + 1.9998(x + 2) + 1$	11

Таблица 2. Результирующие суперпозиции.

N	$\hat{f}$	f_o	$C(\hat{f})$	$C(f_o)$
1	1	1	1	1
2	$(x + 1)^2$	—	5	9
3	$-1 + (3 + x)^2$	—	7	11
4	—	—	11	11
5	$-1 + (\sin 2x)^2$	—	8	19
6	$(x + 2.99)^2 + 0.000199$	—	7	11

4. Сворачивание в квадрат суммы:

$$at^2 + bt \equiv \left(\sqrt{a}t + \tfrac{b}{2\sqrt{a}}\right)^2 - \left(\tfrac{b}{2\sqrt{a}}\right)^2.$$

5. Операции со степенями:

$$t^a t^b \equiv t^{a+b}; \quad \tfrac{t^a}{t^b} \equiv t^{a-b}; \quad t^{ab} \equiv t^{ab}.$$

6. Формула двойного угла:

$$\cos t \sin t \equiv \tfrac{1}{2}\sin(2t).$$

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.

Заключение

В работе описан и исследован алгоритм упрощения произвольных суперпозиций элементарных функций. Предложенный алгоритм решает задачу упрощения суперпозиций путем представления их в виде соответствующего дерева, затем преобразовываемого в граф специального вида, к которому применяются различные изоморфизмы согласно заранее определенному набору правил. Приведены результаты применения алгоритма к синтетическим суперпозициям для данного набора правил.

Литература

[1] *Ehrig H. et al* Fundamentals of Algebraic Graph Transformation. Springer, 2006.

[2] *Carette J.* Understanding expression simplification // Int'l Symp. on Symbolic and Algebraic Computation, Cantabria: ACM Press, 2004. — С. 72–79.