Пакеты полигармонических сплайнов, их объединение, эффективные процедуры вычисления и дифференцирования

Бахвалов Ю.Н., к.т.н., независимый исследователь, г. Череповец

Аннотация.

В работе [2] было показано, что регрессионная задача машинного обучения может быть решена на основе теории случайных функций [11], причём оптимальное ядро выводится аналитически из принципов симметрии и индифферентности и соответствует полигармоническому сплайну. Однако прямое применение этого решения ограничено вычислительной сложностью $O(N^3)$ и нарушением исходных теоретических предпосылок в случае избыточной размерности входного пространства. В данной статье предлагается каскадная архитектура из «пакетов» полигармонических сплайнов, которая одновременно решает проблему масштабируемости и теоретически обоснована для задач с неизвестной внутренней низкой размерностью. Представлены эффективные матричные процедуры прямого вычисления и сквозного дифференцирования каскада.

Ключевые слова: машинное обучение, регрессия, случайная функция, симметрия, корреляционная функция, полигармонический сплайн, полигармонический каскад, пакет полигармонических сплайнов, дифференцирование.

Packages of polyharmonic splines, their unification, efficient procedures for calculation and differentiation

Bakhvalov Y.N., Ph.D., Independent Researcher, Cherepovets

Abstract.

In the work [2] it was shown that a machine learning regression problem can be solved based on random function theory [11], with the optimal kernel being analytically derived from the principles of symmetry and indifference and corresponding to a polyharmonic spline. However, the direct application of this solution is limited by the computational complexity of $O(N^3)$ and the violation of the original theoretical assumptions in the case of excessive input space dimensionality. This paper proposes a cascade architecture of "packets" of polyharmonic splines that simultaneously addresses the scalability issue and is theoretically justified for problems with unknown internal low dimensionality. Efficient matrix procedures for direct computation and end-to-end differentiation of the cascade are presented.

Keywords: machine learning, regression, random function, symmetry, correlation function, polyharmonic spline, package of polyharmonic splines, polyharmonic cascade, differentiation.

В работе [2] была рассмотрена регрессионная задача машинного обучения, как задача аппроксимации с использованием математического аппарата теории случайных функций [11]. Было показано, что если допустить (исходя из принципов индифферентности), что вероятностная мера в бесконечномерном функциональном пространстве обладает естественными симметриями (инвариантность к сдвигу, повороту, масштабированию и гауссовость), то вся решающая схема — включая вид ядра, форму регуляризации и параметризацию шума — выводится аналитически из этих постулатов.

Опишем ключевые моменты решения из [2].

Пусть обучающая выборка представлена в виде $x_1, x_2, \dots, x_k (x_i \in R^n)$ набора из k векторов размерностью n на входе и $y_1, y_2, \dots, y_k$ $(y_i \in R)$ значений на выходе.

Решением регрессионной задачи будет функция $f(x)$, которая связана с обучающей выборкой следующим образом:

$$y_i = f(x_i) + u_i \,, \tag{1}$$

где $u_1, u_2, \dots, u_k \in R$ – независимые нормальные случайные величины с нулевым математическим ожиданием и дисперсией σ^2 (которую можно задать).

Если рассматривать $f(x)$ как некоторую неизвестную реализацию случайной функции и допустить, что вероятностная мера в бесконечномерном функциональном пространстве обладает некоторыми симметриями (рассмотренными в [2]), то можно получить решение, где $f(x)$ будет определяться как:

$$f(x) = \sum_{i=1}^{k} \lambda_i k_f(x_i - x) \,, \tag{2}$$

где $k_f(\tau)$ было получено в виде:

$$k_f(\tau) = \|\tau\|^2 (\ln(\|\tau\|) - b) + c \,, \tau \in R^n \tag{3}$$

где b и c – можно взять константами, значения которых можно оценить как приближенное значение функций:

$$b(\tau) = \frac{\sin(\omega_0 \tau)}{\omega_0 \tau} - \ln(\omega_0) - \gamma - \int_0^{\omega_0 \tau} \frac{\cos(\omega) - 1}{\omega} d\omega \quad (4)$$

где γ – постоянная Эйлера-Маскерони

$$c(\tau) = \frac{\cos(\omega_0 \tau)}{\omega_0^2} \quad (5)$$

По сравнению со значениями τ (расстояния между векторами в обучающей выборке) частота ω_0 (ниже которой гармоники в случайной функции будут отсутствовать) будет являться очень малой величиной, для которой период T_0, определяемый выражением

$$T_0 = \frac{2\pi}{\omega_0} \quad (6)$$

будет наоборот, значительно больше, чем расстояние между векторами во входной части обучающей выборки. Например, для $\|\tau\|$ в пределах от 0 до 10 и $\omega_0 = 0.001$, в [2] были определены коэффициенты $\mathrm{b} = 7.33054$ и $c = 1000000$.

Коэффициенты λ_i из (2) будут определяться решением системы уравнений:

$$\lambda = (K + \sigma^2 E)^{-1} Y \quad (7)$$

где K – квадратная матрица элементов $k_{ij} = k_f(x_i - x_j)$, $i, j = \overline{1, k}$

E – единичная матрица

λ – вектор столбец $(\lambda_1, \lambda_2, \dots, \lambda_k)$

Y – вектор столбец $(y_1, y_2, \dots, y_k)$ из обучающей выборки

σ^2 – дисперсия случайных величин u_i из (1)

Но значения b и c могут быть оценены однократно и основными рабочими выражениями являются (1), (2), (3) и (7).

Параметр $\sigma^2 > 0$ в (7) гарантирует обратимость матрицы. Для задач точной интерполяции ($\sigma^2 = 0$) даже при самом неудачном расположении точек можно использовать вместо констант b и c в (3) их точные определения через (4) и (5).

Функция $k_f(\tau)$ является обобщенной ковариацией для случайной функции (класс собственных случайных функций порядка 1 Matheron [9]), для которой $f(x)$ будет являться одной из реализаций.

В то же время функция $\tau^2 ln(\tau)$ и ее линейные комбинации известны как полигармонический сплайн, так называемый сплайн тонкой пластины (thin plate spline) [4] и [8].

Значение $k_f(0)$ является также значением дисперсии случайной функции. Одновременное умножение $k_f(\tau)$ в (3) и σ^2 в (7) на одно и то же любое ненулевое значение приведет к тому же самому решению и той же функции в $f(x)$ в (2).

Однако, рассмотренная в [2] задача аппроксимации касается лишь случая, когда $y_i \in R$, т.е. для определения одной единственной функции $f(x)$. Рассмотрим теперь случай, когда на выходе нужно получить вектора значений $y_i \in R^m$. Т.е., чтобы решить задачу аппроксимации нужно вычислить m функций $f_1(x), f_2(x), \dots, f_m(x)$.

Вычислить эти функции можно независимо, используя выражения (1) – (7). Однако, несмотря на то что значения этих функций могут быть никак не связаны между собой, с вычислительной точки зрения между этими функциями будет определенная взаимосвязь. Все они будут описываться выражением (2), отличаясь между собой лишь коэффициентами λ_i. В выражении (7), в котором вычисляются сами λ_i, во всех случаях будет одинаковая матрица K, а если допустить одинаковую дисперсию σ^2, то будет одинаковой обратная матрица $(K + \sigma^2 E)^{-1}$.

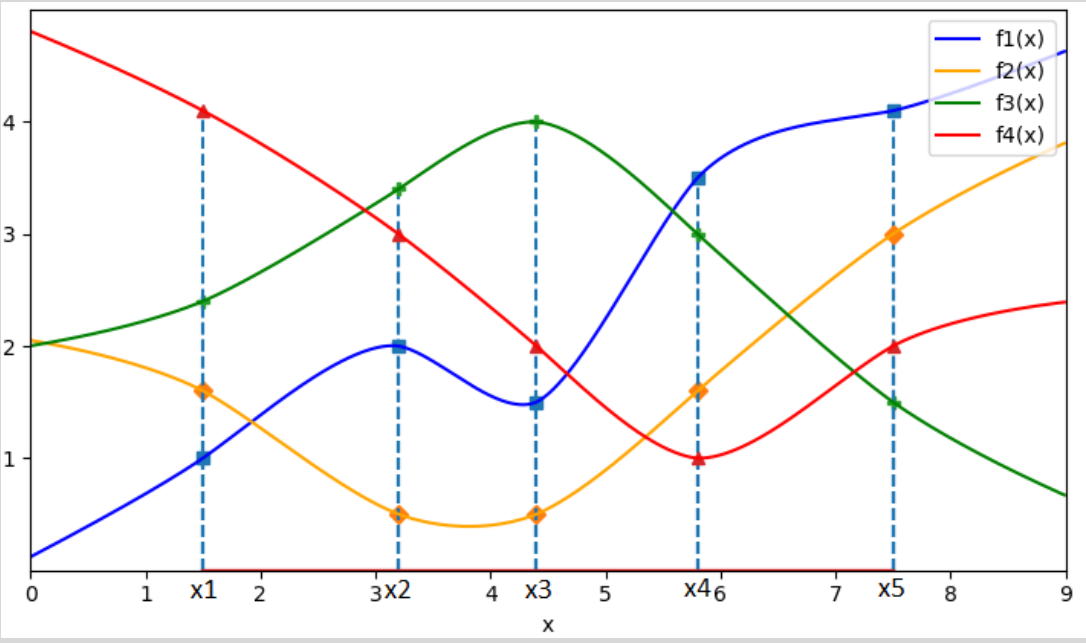

Рисунок 1.

На рисунке 1 проиллюстрирован пример одномерных функций.

Здесь представлен пример построения четырех различных функций (вариант интерполяции при $\sigma^2 = 0$). Они сильно отличаются друг от друга, но все были построены по координатам в одних и тех же точках $x_1, x_2, \dots, x_5$.

Все четыре функции описываются одним и тем же выражением (2), отличаясь только значениями λ_i , которые в свою очередь вычисляются в (7) с использованием одной и той же обратной матрицы.

Таким образом, если мы зададим некоторый набор ключевых точек $c_1, c_2, \dots, c_k$ (назовем их “созвездием”) и однократно вычислим для них матрицу:

$$U = \left(K^{(c)} + \sigma^2 E\right)^{-1}, \tag{8}$$

где матрица $K^{(c)}$ состоит из элементов $k_{ij}^{(c)} = k_f(c_i - c_j)$, значений функции (3).

тогда теперь в рамках этого созвездия мы можем описать любое произвольное количество функций, каждая из которых будет описываться как (2), но заданная через точки созвездия.

Пусть количество функций равно m. Тогда их можно записать:

$$f_t(x) = \sum_{p=1}^{k} \lambda_{pt} k_f(c_p - x) \ , t = \overline{1, m} \tag{9}$$

Задание любой из этих функций сведется к заданию ее значений (в виде вектора столбца) в ключевых точках созвездия (или приближенных её значений, если $\sigma^2 > 0$) и умножению этого вектора на матрицу U из (8) для вычисления вектора столбца $\lambda_t(\lambda_{1t}, \lambda_{2t}, \dots, \lambda_{kt})$.

В общем виде для m функций это будет матрица Λ размерностью $(k \times m)$

$$\Lambda = UY^*, \tag{10}$$

где Y^* - матрица $(k \times m)$ $[y_1^*, y_2^*, \dots, y_m^*]$ в которой y_t^* это вектора столбцы с заданием значений для $f_t(x), t = \overline{1, m}$,

Λ – матрица элементов λ_{pt} из (9)

Выражение (9) можно записать в матричном виде:

$$F_x = K_x \Lambda \ , \tag{11}$$

где K_x – вектор строка $\left[k_f(c_1 - x), k_f(c_2 - x), \dots, k_f(c_k - x)\right]$,

F_x – вектор строка значений $[f_1(x), f_2(x), \dots, f_m(x)]$

Набор описанных таким образом функций на некотором созвездии точек в контексте данной статьи будем называть пакетом. Т.е. под пакетом полигармонических сплайнов будем понимать совокупность функций вида (2), построенных на общем созвездии центров, коэффициенты в которых определяются уравнением (10) через общую обратную матрицу (8).

Рассмотрим более подробно, как удобнее работать с таким математическим объектом. Предположим, мы решаем задачу аппроксимации в R^n с помощью пакета из m функций $f_1(x), f_2(x), \dots, f_m(x)$, $x \in R^n$. Представим созвездие точек (о котором уже говорили выше) в виде матрицы C, каждая строка которой $c_1, c_2, \dots, c_k \in R^n$ соответствует одной из точек созвездия.

Преобразуем (3) в более удобный для вычисления вид:

$$k_f(\tau) = \tau^2(\ln(\|\tau\|) - b) + c = \tau^2\left(\frac{1}{2}\ln(\tau^2) - b\right) + c =$$

$$= \frac{1}{2}\|\tau\|^2(\ln(\|\tau\|^2) - 2b) + c \qquad (12)$$

Замечание. Выражение $\|\tau\|^2 \ln(\|\tau\|)$ не определено при $\|\tau\| = 0$. Однако, поскольку $\lim_{\|\tau\|\to 0} \|\tau\|^2 \ln(\|\tau\|) = 0$, следует явно задавать значение: $k_f(0) = c$. Для очень малых расстояний $\|\tau\|$ (например, $\ll 10^{-10}$) также рекомендуется использовать аппроксимацию $k_f(\tau) \approx c$ во избежание численных нестабильностей.

Выражение (12) удобно тем, что τ в нем везде входит как τ^2(квадрат длины вектора τ). Тогда чтобы найти матрицу $K^{(c)}$ из (8) первым шагом найдем матрицу квадратов расстояний между точками созвездия (обозначим эту матрицу как M_c размерностью $(k \times k)$). Запишем теорему косинусов в матричном виде:

$$M_c = N_c J_{1,k} + J_{k,1} N_c^T - 2CC^T \,, \qquad (13)$$

где N_c – вектор столбец квадратов расстояний до точек созвездия от начала координат

$$N_c = (C \circ C) J_{n,1}, \qquad (14)$$

$\circ$ - произведение Адамара,

$J_{1,k}$ – вектор строка единиц размера k,

$J_{k,1}$ – вектор столбец единиц размера k,

$J_{n,1}$ - вектор столбец единиц размера n.

Теперь можем вычислить элементы матрицы $K^{(c)}$:

$$k_{ij}^{(c)} = \frac{1}{2} m_{ij}^{(c)} \left(\ln \left(m_{ij}^{(c)} \right) - 2b \right) + c \,, \tag{15}$$

где $k_{ij}^{(c)}$ – элемент матрицы $K^{(c)}$ из (8),

$m_{ij}^{(c)}$ – элемент матрицы M_c из (13).

Зная матрицу $K^{(c)}$, определяем в (8) матрицу U, по которой затем в (10) можем разом вычислить коэффициенты (матрицу Λ) для любого произвольного количества m функций в пакете.

Рассмотрим теперь ситуацию, когда нужно вычислить пакет функций не от одного некоторого значения x (что описано в (2) или (9), (11)), а сразу от множества значений. Это может потребоваться для работы с порциями данных (батчами).

Пусть на вход пакета поступает r векторов $x_1, x_2, \dots, x_r (x_i \in R^n)$, которые обозначим в виде матрицы X размерностью $(r \times n)$, каждая строка которой это будет один из этих векторов x_i. Для каждого из них нужно вычислить вектора на выходе пакета $y_1, y_2, \dots, y_r$ $(y_i \in R^m)$, которые будут размерностью m, в соответствии с количеством функций в пакете. Совокупность всех этих векторов y_i обозначим в виде матрицы Y $(r \times m)$.

Аналогично (13) сначала вычислим матрицу квадратов расстояний между всеми векторами в X и всем точками созвездия в C.

$$M_{xc} = N_x J_{1,k} + J_{r,1} N_c^T - 2XC^T \,, \tag{16}$$

где

$$N_x = (X \circ X) J_{n,1} \,, \tag{17}$$

$N_c = (C \circ C) J_{n,1}$ (аналогично (14)).

В отличие от (13) матрица M_{xc} будет не квадратной, а прямоугольной, размерностью $(r \times k)$. Далее аналогично (15) найдем, но уже прямоугольную, матрицу $K^{(xc)}$.

$$k_{ip}^{(xc)} = \frac{1}{2} m_{ip}^{(xc)} \left(\ln \left(m_{ip}^{(xc)} \right) - 2b \right) + c \,, \tag{18}$$

где $k_{ip}^{(xc)}$ – элемент матрицы $K^{(xc)}$,

$m_{ip}^{(xc)}$ – элемент матрицы M_{xc} из (16),

$$i = \overline{1,r}\ ;\ p = \overline{1,k}\ .$$

Теперь можем вычислить Y:

$$Y = K^{(xc)}\Lambda \tag{19}$$

Таким образом, получаем пакет полигармонических сплайнов специального вида, как универсальную конструкцию вычисления произвольного количества многомерных функций, для описания которых в рамках пакета достаточно задать их значения в ключевых точках (в точках заданного созвездия).

Очевидно, что если взять несколько пакетов полигармонических сплайнов рассматриваемого типа, то их можно соединить последовательно в виде каскада, аналогично, как слои нейронов в алгоритмах многослойных искусственных нейронных сетей (Рисунок 2).

В отличие от стандартных многослойных перцептронов, где нелинейность вводится через активационные функции, предлагаемая каскадная архитектура сохраняет глобальную гладкость на каждом уровне за счёт использования полигармонических ядер. Это делает её ближе к подходам на основе глубоких ядерных машин (Cho & Saul, 2009 [5]) или иерархических RBF-сетей (Fasshauer, 2007 [6]), однако с ключевым отличием: базисные функции здесь не выбираются эмпирически, а выводятся из принципов симметрии, как показано в [2].

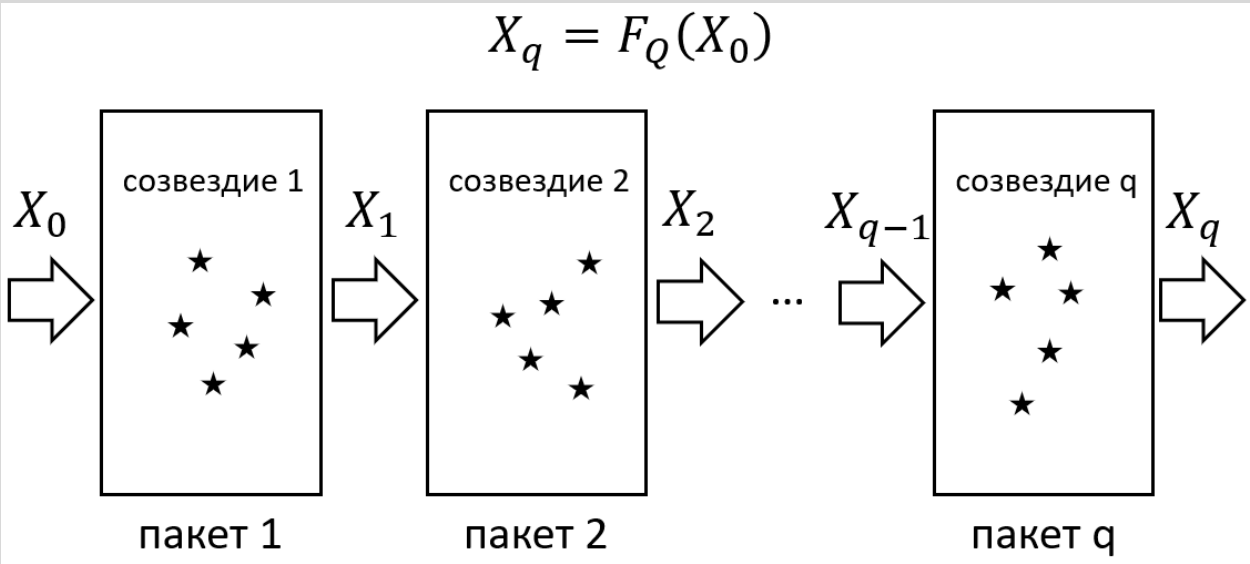

Рисунок 2.

Однако, возникает вопрос. В [2] было показано, что выражения (1) – (7) являются решением на множестве всех возможных непрерывных функций.

Следовательно, даже наличие только одного пакета не накладывает никаких ограничений на аппроксимацию многомерных нелинейных функций сколь угодно большой сложности. К тому же, уже гарантируется, что решение будет в определенном смысле оптимальным (при условиях, описанных в [2]).

Зачем тогда создавать конструкцию из последовательности пакетов? На это есть несколько веских причин. Перечислим наиболее важные.

Первая причина заключается в том, что в (2) (или в (9)) количество слагаемых равно размеру обучающей выборки. Ту же роль играет количество вершин в созвездии, если говорить о пакете. Точно также растет и количество строк и столбцов в вычисляемой обратной матрице (7) - (8), количество элементов которой (а значит и место, занимаемое в памяти) растет уже как вторая степень, а сложность вычисления как третья степень (или около неё) от этого количества.

Следовательно, если обучающая выборка начинает превышать 10-100 тысяч примеров, то решать регрессионную задачу напрямую с помощью разобранных выше выражений (1) – (7), просто наращивая количество слагаемых в (2), становится невозможным. Аналогично, использовать только один пакет полигармонических сплайнов, бесконечно наращивая количество точек в созвездии (вместо каскадного соединения нескольких пакетов) выглядит как тупиковый путь.

Но дело не только в вычислительной сложности. При решении многих практических регрессионных задач машинного обучения конструкция из последовательности пакетов может дать лучшее качество решения, чем один слой. Рассмотрим далее, почему так происходит.

При решении многих реальных задач машинного обучения, количество признаков (размерность векторов x_i во входной части обучающей выборки) на входе модели на самом деле является избыточным. Часть признаков могут быть неинформативными и никак не быть связанными со значениями, которые должны быть вычислены на выходе. Часть признаков могут быть коррелированы между собой и нести одну и ту же информацию.

Т.е. часто существует способ преобразовать входное исходное пространство параметров до другого пространства, гораздо меньшей размерности, на котором без потери качества можно решить задачу аппроксимации. Этот способ преобразования одного пространства в другое может быть неизвестен, но в данном случае, важен сам факт его существования.

Рассмотрим, к чему это может привести, если вернуться к решению регрессионной задачи на основе теории случайных функций, в результате приведшее к выражениям (1) – (7).

Но сначала перечислим три положения о симметриях вероятностной меры μ в функциональном пространстве, которые легли в основу математических выражений, на основе которых было получено решение в [2].

1. Инвариантность вероятностной меры, относительно преобразований сдвига и поворота системы координат.

Для любых M, t и преобразования T: $f_1 \to f_2$ связанных между собой:

$$f_2(x) = f_1(Mx + t),\ x, t \in R^n \tag{20}$$

где M- некоторая матрица поворота

И любого измеримого подмножества функций $A \subset \mathcal{F}$ должно быть выполнено $\mu(A) = \mu\big(T(A)\big)$.

Т.е. для любых подмножеств функций, которые преобразуются друг в друга путем поворота или параллельного переноса вероятностная мера μ должна быть одинакова.

2. Инвариантность вероятностной меры относительно преобразований масштаба.

Для любого k и преобразования T: $f_1 \to f_2$ таких что:

$$f_2(x) = kf_1(x/k), x \in R^n \tag{21}$$

где $k \in R$ – некоторый коэффициент

И любого измеримого подмножества функций $A \subset \mathcal{F}$ должно быть выполнено $\mu(A) = \mu\big(T(A)\big)$.

Т.е. для любых подмножеств функций, которые преобразуются друг в друга изменением масштаба вероятностная мера μ должна быть одинакова.

3. Вероятностная мера μ является бесконечномерной гауссовской мерой с нулевым средним.

Это означает, что любая конечномерная проекция случайной функции имеет многомерное нормальное распределение, а сама мера согласована по Колмогорову (Гихман И. И., Скороход А. В. [7]).

Для лучшего понимания содержания пункта 3, в [2] были сформулированы два его следствия, имеющих наглядную интерпретацию.

Следствие 3.1. Из гауссовости вероятностной меры μ следует, что если какое-то подмножество функций порождается масштабированием одной базовой функции $f_1(x) \to kf_1(x)$ то условная плотность вероятности на этом множестве пропорциональна нормальному закону по параметру k с нулевым средним.

Т.е. если предположить, что решением является одна из функций на подмножестве функций, которые описываются как:

$$f_k(x) = kf_1(x), k \in R \tag{22}$$

где k – некоторое вещественное число (назовем его в данном контексте амплитудой $f_1(x)$)

$f_1(x)$ – некоторая произвольная непрерывная функция, взятая чтобы породить множество в (5)

Тогда условная плотность вероятности на этом подмножестве функций будет определяться нормальным законом от амплитуды k, с математическим ожиданием при $k = 0$.

Следствие 3.2. Если к каждой такой функции из множества сл.3.1 прибавить произвольную фиксированную $f_2(x)$ то распределение по k сохранит гауссов характер (хотя среднее может сместиться).

Прибавим ко всем функциям в множестве, что описано в сл.3.1 одну и ту же функцию $f_2(x)$ и получим новое множество функций, элементы которого можно описать как:

$$f_k^*(x) = kf_1(x) + f_2(x), k \in R \tag{23}$$

где k – некоторое вещественное число (амплитуда $f_1(x)$)

$f_1(x)$ и $f_2(x)$ – любые непрерывные функции

Какую бы мы функцию $f_2(x)$ не взяли, плотность вероятности на этом подмножестве функций (порожденная как конечномерная проекция меры μ) также как и в сл.3.1 сохранит нормальный закон распределения от амплитуды k (но математическое ожидание уже может не равняться нулю).

Итак, теперь рассмотрим ситуацию, когда априорно ожидается, что часть признаков на входе избыточна (или неинформативна).

В качестве примера рассмотрим функцию двух переменных, аппроксимацию которой мы собираемся выполнить алгоритмом, в основе которого лежат пункты (20) – (23) (соответственно можем ее рассматривать как реализацию случайной функции).

$$y = f(x_1, x_2)$$

Предположим известно, что с очень большой вероятностью (но меньшей, чем 1) априорно ожидается, что одна из переменных либо x_1 либо x_2 будет являться неинформативной. Т.е. функция $y = f(x_1, x_2)$ на самом деле является либо функцией $y = f(x_1)$, либо функцией $y = f(x_2)$, хотя какой именно вариант неизвестно. Будем обозначать их как $f(x_1, \_)$ и $f(\_, x_2)$, подразумевая, что функция остается двумерная, но в первом случае ее значения никак не зависят от x_2, а во втором случае никак не зависят от x_1. И только с очень малой вероятностью (но не равной нулю) будем ожидать, что $y = f(x_1, x_2)$ является некоторой нелинейной функцией от двух переменных.

Рассматривая $y = f(x_1, x_2)$ как некоторую неизвестную реализацию случайной функции, в качестве вариантов ответа могут быть любые непрерывные двумерные функции, среди которых в качестве подмножеств входят в том числе и всевозможные непрерывные функции вида $y = f(x_1, \_)$ и $y = f(\_, x_2)$. Но получается, что значения вероятностной меры на этих подмножествах должны быть значительно выше, по сравнению с другими вариантами решений.

Возьмем некоторые две нелинейные функции из этих подмножеств $f_1(x_1, \_)$ и $f_2(\_, x_2)$. Введем в рассмотрение еще одну функцию, являющуюся их разностью:

$$f_3(x_1, x_2) = f_2(\_, x_2) - f_1(x_1, \_) \tag{24}$$

Очевидно, что $f_3(x_1, x_2)$ будет нелинейной функцией от двух переменных и для определения её значения нужно знать как значение x_1 так и x_2.

Теперь рассмотрим подмножество функций, которое описывается следующим образом:

$$f_k(x_1, x_2) = f_1(x_1, \_) + k f_3(x_1, x_2)\,, \tag{25}$$

где k – некоторое вещественное число

Выражение (25) задает подмножество функций с конечномерной проекцией вероятностной меры, которое полностью попадает под действие сл.3.2 (23), однако подставив в него (24) получим:

$$f_k(x_1, x_2) = f_1(x_1, \_) + k\big(f_2(\_, x_2) - f_1(x_1, \_)\big) \tag{26}$$

Тогда при $k = 0$:

$$f_k(x_1, x_2) = f_1(x_1, \_), \tag{27}$$

а при $k = 1$:

$$f_k(x_1, x_2) = f_2(\_, x_2) \tag{28}$$

Таким образом получается, что при $k = 0$ получившаяся функция $f_k(x_1, x_2)$ зависит только от параметра x_1 а при $k = 1$ только от параметра x_2. При остальных значениях k в этом подмножестве будет нелинейная зависимость от двух переменных.

Но тогда, исходя из наших условий, можно ожидать, что плотность вероятности на этом подмножестве функций в зависимости от k будет иметь две локальные точки максимума при $k = 0$ и $k = 1$. Это противоречит сл.3.2 (23).

Значит тогда мы уже не можем рассматривать случайную функцию, которая имеет в функциональном пространстве гауссову меру. В этом случае уже нельзя рассматривать каноническое и спектральное разложения случайной функции и все следующие выводы. Значит решение в виде выражений (1) – (7) уже не будет выглядеть как логичный вариант из принципов индифферентности для аппроксимации такой функции.

Посмотрим, какие качественные рассуждения можно провести про вероятностную меру в этом случае?

В [2] для рассматриваемой случайной функции распределение вероятностей ее реализаций в функциональном пространстве было задано как гауссова мера. В этом случае линии уровня для любой её конечномерной проекции представляют собой гиперэллипсоиды, главные оси которых направлены вдоль функций e^{iwx} (если рассматривать спектральное разложение случайной функции в комплексной форме).

В ситуации же когда мы априорно ожидаем, что размерность исходного пространства признаков (в котором решается задача регрессии) избыточна, и это пространство относительно легко (например, сокращая размерность отбрасыванием неинформативных признаков или с помощью преобразований близких к линейным) может быть преобразовано (пусть и исходно это преобразование неизвестно) в пространство гораздо меньшей размерности, в котором без потери качества можем решить эту же задачу регрессии, тогда рассматривать многомерные эллипсы в качестве линий уровня конечномерных проекций вероятностной меры мы уже не можем.

Функции, которые будут зависеть от меньшего числа признаков (а также функции, которые преобразуются в них путем некоторого многомерного поворота) будут обладать большей плотностью вероятности и образовывать выступы на линиях уровня, по сравнению с функциями большей размерности. В этом случае линии уровня будут скорее похожи на многомерного “ежа” или

“осьминога”, чем на гиперэллипсоиды. Другими словами, плотность вероятности будет иметь мультимодальные пики в подпространствах меньшей размерности.

Строгое описание таких мер выходит за рамки настоящей работы, здесь мы ограничиваемся конструктивным предложением вычислительной архитектуры, мотивированной этими соображениями. Хотя это конечно лишь качественные рассуждения, но они позволяют предположить, что в случае избыточности исходного признакового пространства решение задачи, рассмотренное в [2], нельзя строго рассматривать как предпочтительное в условиях отсутствия априорной информации.

Однако, если мы рассмотрим задачу аппроксимации с помощью каскада пакетов полигармонических сплайнов, то ситуация меняется.

Рассмотрим пространство размерностью n, с заданной на нем случайной функцией с рассматриваемыми симметриями (20) - (23). Далее пусть мы наблюдаем m ($m \ll n$) реализаций этой случайной функции. На множестве значений (размерностью меньшей чем n) этих функций может быть снова задана случайная функция с симметриями (20) - (23). Процедуру можно многократно повторить далее.

Весь этот каскад также можно рассматривать как единую случайную функцию. Но как распределяться вероятности различных реализаций такой случайной функции между собой? Будут ли выполняться для такой составной случайной функции симметрии вероятностной меры в функциональном пространстве (пункты (20) - (23))?

Обозначим такую составную случайную функцию как $F_Q(x)$, которая состоит из q слоёв случайных функций (описанных выше), которые обозначим как $F_1(x), F_2(x), \dots F_q(x)$.

$$F_Q(x) = F_q\left(F_{q-1}\left(\dots\left(F_1(x)\right)\right)\right) \tag{29}$$

Поскольку $F_1(x)$ обладает пространственной инвариантностью, не зависит ни от выбора начала системы координат, ни от ориентации, то пространственной инвариантностью будет обладать и случайная функция (29) целиком. Таким образом, пункт первый, соответствующий (20), для составной случайной функции будет выполняться.

Также будет выполнено и сл.3.1 (22). Т.е. вероятностное распределение на любом подмножестве реализаций, в котором реализации отличаются друг от друга только амплитудой, будет нормальным распределением, зависящим от амплитуды (с математическим ожиданием равным нулю). Для этого достаточно, чтобы сл.3.1 выполнялось для случайных функций в $F_q(x)$.

Но как на счет выполнения второго пункта (21)?

Рассмотрим некоторую конкретную реализацию $f_Q(x)$ случайной функции (29).

$$f_Q(x) = f_q\left(f_{q-1}\left(\dots\left(f_1(x)\right)\right)\right), \tag{30}$$

где $f_1(x), f_2(x), \dots f_q(x)$ – некоторые конкретные реализации, которые приняли случайные функции $F_1(x), F_2(x), \dots F_q(x)$.

На самом деле под $f_i(x)$ будем подразумевать сразу несколько реализаций, количество которых равно количеству выходов $i-$го слоя.

Ожидание события появления $f_Q(x)$ как этой конкретной некоторой комбинации реализаций в (30) будет равно ожиданию одновременного появления соответствующих реализаций $f_1(x), f_2(x), \dots f_q(x)$ из этой комбинации.

Теперь рассмотрим реализацию $f_Q^*(x)$:

$$f_Q^*(x) = f_q^*\left(f_{q-1}^*\left(\dots\left(f_1^*(x)\right)\right)\right), \tag{31}$$

где

$$f_i^*(x) = kf_i\left(\frac{x}{k}\right), \tag{32}$$

k – некоторое число $(k \neq 0)$, одинаковое для всех $f_i^*(x)$ в (32)

Для всех рассмотренных случайных функций $F_1(x), F_2(x), \dots F_q(x)$ на основании свойства в п.2. (21) ожидание появления $f_i(x)$ и $f_i^*(x)$ должно быть одинаково.

Тогда появление $f_Q(x)$ как некоторой конкретной комбинации реализаций $f_1(x), f_2(x), \dots f_q(x)$ в (30) появление $f_Q^*(x)$ как соответствующей комбинации $f_1^*(x), f_2^*(x), \dots, f_q^*(x)$ в (31) будут одинаково ожидаемы.

Но используя (32) можно записать (31) как:

$$f_Q^*(x) = kf_q\left(kf_{q-1}\left(\dots kf_2\left(\frac{kf_1\left(\frac{x}{k}\right)}{k}\right)/k\right)/k\right) =$$

$$= kf_q\left(f_{q-1}\left(\dots\left(f_1(x/k)\right)\right)\right) = kf_Q(x/k) \tag{33}$$

Поскольку (33) будет справедливо для любых реализаций и их комбинаций в (30)-(31), а также для любого значения k, то можно ожидать, что

составная случайная функция (29) будет обладать тем же свойством п.2. (21) что и входящие в ее состав случайные функции.

Рассмотрим теперь соответствие составной функции (29) сл.3.2 (23).

Теоретически, в качестве реализаций составной случайной функции (29) может быть любая непрерывная вещественная функция в виде сложной нелинейной зависимости от всех n входных параметров. Например, получится такая функция может в виде одной из реализаций выпавшей на первом слое случайной функции $F_1(x)$, а все остальные реализации случайных функций $F_2(x), \dots F_q(x)$ могут случайно оказаться даже линейными.

Однако, поскольку характеристики всех случайных функций $F_1(x), F_2(x), \dots F_q(x)$ одни и те же, а появление той или иной реализации носит случайный характер, то более вероятным следует ожидать такие их комбинации, когда сборка и усложнение случайной функции (29) будет носить поэтапный характер от слоя к слою. Но, если, как мы уже решили, количество реализаций m случайной функции $F_1(x)$ (размерность после первого слоя) будет значительно меньше размерности исходного пространства признаков n $(m \ll n)$, то тем самым мы моделируем ситуацию, когда сложные нелинейные зависимости если будут присутствовать, то будут определяться в пространстве внутренних признаков меньшей размерности m, в которое будет осуществлен переход более простым способом с помощью лишь одного слоя реализаций случайной функции $F_1(x)$.

Таким образом мы получаем модель того, что как раз можем ожидать при решении большинства практических регрессионных задач машинного обучения. И как уже было показано выше, сл.3.2 (23) в этом случае может не выполняться.

Подобное нарушение гауссовости при композиции случайных функций не является уникальным для рассматриваемой постановки. В работах по глубоким гауссовым процессам (Deep Gaussian Processes) показано, что композиция гауссовых процессов, вообще говоря, не является гауссовым процессом (Damianou & Lawrence, 2013 [13]). Следовательно, хотя каждый слой каскада по отдельности может быть интерпретирован как реализация гауссовой меры с симметриями (20)–(23), вся композиция в целом способна породить негауссову меру.

В итоге, можно предположить, что, если для случайной функции будут выполняться симметрии вероятностной меры, касающиеся сдвига, поворота и масштабирования п.1 – п.2 (20) – (21), но имеется нарушение в п.3 о гауссовости меры (23), когда сл.3.1 выполнено, а сл.3.2. нарушено в ситуации, когда априорно ожидается, что решение регрессионной задачи лежит в

пространстве меньшей размерности, чем исходное пространство признаков, тогда моделью для решения задачи может служить каскадное последовательное соединение пакетов полигармонических сплайнов (1) – (7).

Если бы были известны последовательно все промежуточные правильные значения преобразования входной части обучающей выборки после каждого пакета, то можно было бы напрямую вычислить все функции в каждом слое, используя (8) – (19).

Однако обучающая выборка в регрессионной задаче состоит из представленных данных на входе и соответствующим им данным на выходе. Промежуточные значения, которые должны быть внутри модели между слоями не известны.

Таким образом, на основе рассуждений выше, можно сделать обоснованное предположение, что объединение пакетов функций в виде каскада может повысить качество и открыть путь к масштабированию решения, предложенного в [2]. Можно ожидать, что получение такой математической конструкции и разработка методов решения с ее помощью задач машинного обучения позволит использовать ее для работы с “большими данными”.

Существуют и другие подходы к масштабированию ядерных методов. Например, метод Нистрома (Williams & Seeger, 2001 [12]) аппроксимирует ядро через подвыборку опорных точек, а быстрые мультипольные методы (Fast Multipole Methods, FMM, Beatson & Greengard, 1997 [3]) позволяют ускорить вычисления за счёт иерархической кластеризации пространства. Однако эти методы, как правило, вносят приближения, которые нарушают глобальную гладкость решения или требуют эвристического выбора подмножеств. В отличие от них, предложенная каскадная архитектура сохраняет точную форму полигармонического ядра на каждом уровне и при этом обеспечивает естественное понижение размерности, что особенно важно при избыточности признакового пространства.

Процедура обработки данных каскадом в прямом направлении выглядит просто. Чтобы вычислить результат обработки каскадом некоторого набора входных данных нужно сначала вычислить результат на выходе первого пакета. Этот результат поступит на вход второго пакета и т.д. Т.е. нужно просто последовательно вычислить набор пакетов функций (в том порядке как они соединены в каскаде в систему) с использованием выражений (16) – (19).

Разберем теперь процедуру дифференцирования. Рассмотрим пакет полигармонических сплайнов, который находится в качестве одного из элементов внутри некоторой более сложной вычислительной системы.

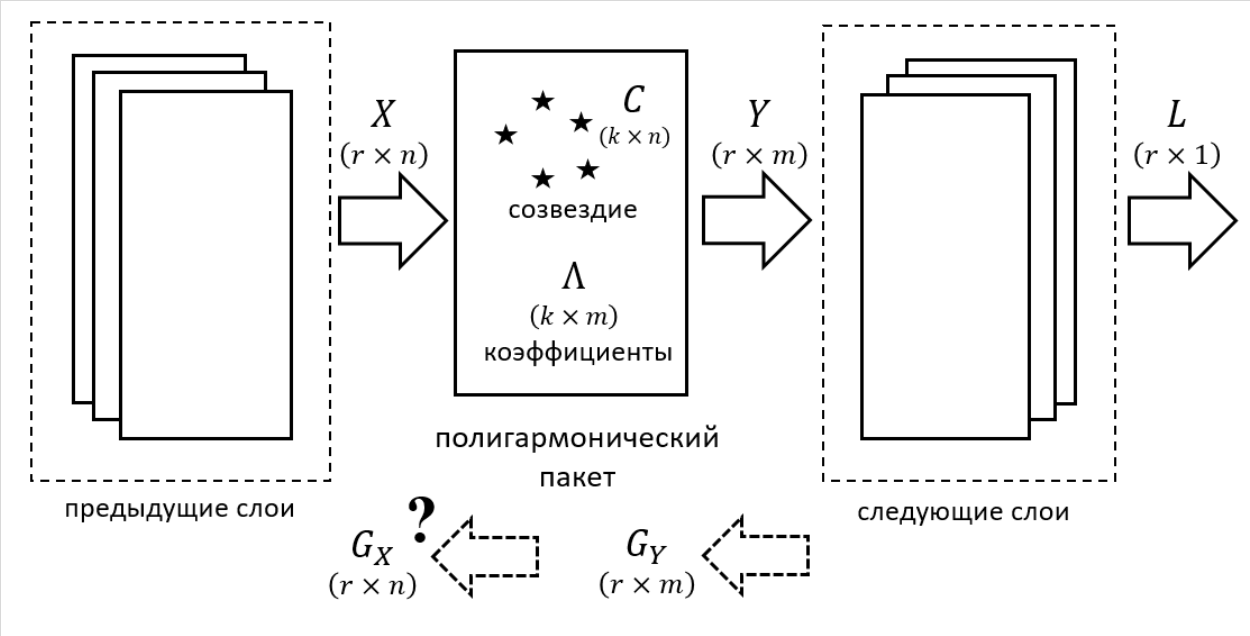

Рисунок 3.

Обозначим как X матрицу элементов x_{ij} размерностью $(r \times n)$, содержащую некоторый фрагмент данных (батч), который пришел на вход пакету с предыдущих слоёв. Каждая строка $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ матрицы X является независимо обрабатываемым вектором, от которого в пакете вычисляется m функций, образуя строку $y_i = \big(f_1(x_i), f_2(x_i), \dots, f_m(x_i)\big)$ в матрице Y на выходе пакета. Таким образом каждый элемент матрицы Y равен $y_{it} = f_t(x_i)$.

Строки матрицы $C(k \times n)$ представляют собой точки созвездия (k точек). Матрица Λ содержит коэффициенты для вычисления функций $f_t(x_i)$ в (19).

Пусть где-то на выходе вычислительной системы был получен вектор значений $L = (l_1, l_2, \dots, l_r)$ размерностью $(r \times 1)$ так, что его составляющие l_i были получены как результаты вычислений следующими после рассматриваемого пакета слоями над поступившими им на вход векторами y_i. Т.е. каждая строка y_i матрицы Y дала (независимо от остальных строк) на выходе системы значение l_i, а матрица Y целиком дала на выходе вектор L.

Теперь предположим, что каким-то образом была получена матрица G_Y размерностью $(r \times m)$, элементы которой равны:

$$g_{it}^{(Y)} = \frac{dl_i}{dy_{it}}, \quad i = \overline{1,r}\,; t = \overline{1,m} \tag{34}$$

Т.е. если мы представим все последующие слои в виде некоторой единой m – мерной функции, то $g_{it}^{(Y)}$ будут ее частными производными в точках y_i.

Рассмотрим теперь, как на основе представленных матриц вычислить матрицу G_X , элементы которой равны:

$$g_{it}^{(X)} = \frac{dl_i}{dx_{ij}},\ i = \overline{1,r}\ ; j = \overline{1,n} \tag{35}$$

Имея такую процедуру можно рассчитать производные функции, которую образует в совокупности собой каскад, последовательно от конца к началу для всех слоёв.

Формально процедура пересчёта производных от слоя к слою напоминает алгоритм обратного распространения ([1] и [10]), однако в данном контексте она служит не для минимизации ошибки, а для обеспечения дифференцируемости всей вычислительной цепи, что позволяет интегрировать архитектуру в современные фреймворки автоматического дифференцирования.

Элементы x_{ij} матрицы X взятые из i – той строки будут влиять только на элементы y_{it} взятые тоже из i – той строки матрицы Y и соответственно будут влиять только на i – тый элемент l_i вектора L.

Поскольку $g_{it}^{(Y)}$ это частные производные функции, вычисляющей l_i по y_{it} в точке y_i то дифференциал l_i будет равен:

$$dl_i = \sum_{t=1}^{m} g_{it}^{(Y)} dy_{it} \tag{36}$$

Отсюда выразим $g_{ij}^{(X)}$:

$$g_{ij}^{(X)} = \frac{dl_i}{dx_{ij}} = \sum_{t=1}^{m} g_{it}^{(Y)} \frac{dy_{it}}{dx_{ij}} = \sum_{t=1}^{m} g_{it}^{(Y)} \frac{df_t(x_i)}{dx_{ij}} \tag{37}$$

Поскольку используя (9) можно $f_t(x_i)$ можно записать как:

$$f_t(x_i) = \sum_{p=1}^{k} \lambda_{pt} k_f\left(c_p - x_i\right) \tag{38}$$

Тогда

$$\frac{df_t(x_i)}{dx_{ij}} = \sum_{p=1}^{k} \lambda_{pt} \frac{dk_f\left(c_p - x_i\right)}{dx_{ij}} \tag{39}$$

Но $k_f\left(c_p - x_i\right)$ в свою очередь можно выразить используя (12) и (18) как:

$$k_f(c_p - x_i) = \frac{1}{2} m_{ip}^{(xc)} \left(\ln\left(m_{ip}^{(xc)} \right) - 2b \right) + c \,, \tag{40}$$

где $m_{ip}^{(xc)}$ – элемент матрицы квадратов расстояний (16), который можно выразить как:

$$m_{ip}^{(xc)} = \left\| c_p - x_i \right\|^2 = \sum_{u=1}^{n} \left(c_{pu} - x_{iu} \right)^2 \tag{41}$$

Тогда

$$\frac{dk_f(c_p - x_i)}{dx_{ij}} = \frac{dk_f(c_p - x_i)}{dm_{ip}^{(xc)}} \cdot \frac{dm_{ip}^{(xc)}}{dx_{ij}} \tag{42}$$

Рассмотрим отдельно два множителя из (42)

$$\frac{dk_f(c_p - x_i)}{dm_{ip}^{(xc)}} = \frac{1}{2} \left(\ln\left(m_{ip}^{(xc)} \right) - 2b + 1 \right) \tag{43}$$

$$\frac{dm_{ip}^{(xc)}}{dx_{ij}} = \frac{d\left(\sum_{u=1}^{n} \left(c_{pu} - x_{iu} \right)^2 \right)}{dx_{ij}} = 2\left(x_{ij} - c_{pj} \right) \tag{44}$$

Подставив (43) и (44) в (39):

$$\frac{df_t(x_i)}{dx_{ij}} = \sum_{p=1}^{k} \lambda_{pt} \left(x_{ij} - c_{pj} \right) \left(\ln\left(m_{ip}^{(xc)} \right) - 2b + 1 \right) \tag{45}$$

Тогда (37) можем записать как:

$$g_{ij}^{(X)} = \sum_{t=1}^{m} g_{it}^{(Y)} \sum_{p=1}^{k} \lambda_{pt} \left(x_{ij} - c_{pj} \right) \left(\ln\left(m_{ip}^{(xc)} \right) - 2b + 1 \right) \tag{46}$$

В выражении (46) вложенность циклов суммирования можно поменять местами, а также представить выражение в виде двух слагаемых:

$$g_{ij}^{(X)} = x_{ij} \sum_{p=1}^{k} \left(\ln\left(m_{ip}^{(xc)} \right) - 2b + 1 \right) \sum_{t=1}^{m} g_{it}^{(Y)} \lambda_{pt}$$
$$- \sum_{p=1}^{k} c_{pj} \left(\ln\left(m_{ip}^{(xc)} \right) - 2b + 1 \right) \sum_{t=1}^{m} g_{it}^{(Y)} \lambda_{pt} \tag{47}$$

Введем в рассмотрение матрицы Θ и Ψ размерностью $(r \times k)$ (такой же размерностью как и матрица M_{xc} из (16)), что элементы матрицы Θ вычисляются как:

$$\theta_{ip} = \ln\left(m_{ip}^{(xc)}\right) - 2b + 1\,, \tag{48}$$

где $m_{ip}^{(xc)}$ – элементы матрицы M_{xc} (16)

а элементы матрицы Ψ вычисляются как:

$$\psi_{ip} = \theta_{ip} \sum_{t=1}^{m} g_{it}^{(Y)} \lambda_{pt} \tag{49}$$

Замечание. Возможна ситуация, когда в (48) $m_{ip}^{(xc)} = 0$ и выражение оказывается не определено. Это соответствует вычислению непосредственно прямо в одной из точек созвездия. В этом случае на практике, можно вместо $m_{ip}^{(xc)}$ взять какое-нибудь очень маленькое число. Что в этом случае получится в (48) будет не так важно, поскольку из (46) следует, что это значение в итоге будет умножено на ноль ($x_{ij} - c_{pj}$ в этом случае также обратится в ноль).

Поскольку сумма произведений $g_{it}^{(Y)}$ и λ_{pt} в (49) означает не что иное, как произведение матриц G_Y и Λ^T, то получим матричное представление для Ψ :

$$\Psi = \Theta \circ (G_Y \Lambda^T)\,, \tag{50}$$

$\circ$ - произведение Адамара

Выражение (47) можно записать:

$$g_{ij}^{(X)} = x_{ij} \sum_{p=1}^{k} \psi_{ip} - \sum_{p=1}^{k} c_{pj} \psi_{ip} \tag{51}$$

Но тогда (51) можно выразить в матричном виде:

$$G_X = X \circ \left(\left(\Psi J_{k,1}\right) J_{1,n}\right) - \Psi C\,, \tag{52}$$

где $\circ$ - произведение Адамара,

$J_{k,1}$ – вектор столбец единиц размера k,

$J_{1,n}$ – вектор строка единиц размера n

Подведем итог.

Пусть мы рассматриваем пакет функций (полигармонических сплайнов специального вида), который можно описать с помощью матрицы C, содержащей точки созвездия, а также матрицы Λ, вычисленную в (10) и содержащую коэффициенты для уравнений функций в пакете. На вход поступает набор данных (батч) в виде матрицы X.

Тогда мы можем найти матрицу квадратов расстояний M_{xc} между векторами строками в матрицах X и C через выражения (16) – (17). А также матрицу $K^{(xc)}$ через (18), содержащую значения корреляционной функции. Умножив $K^{(xc)}$ на Λ в (19) получим матрицу Y на выходе пакета.

Кстати, матрицу $K^{(xc)}$ можно выразить и через матрицу Θ, вычисление элементов которой описано в (48):

$$K^{(xc)} = \frac{1}{2} M_{xc} \circ \left(\Theta - J_{r,k}\right) + c \tag{53}$$

где $J_{r,k}$ – матрица единиц размерностью $(r \times k)$

Если же нужно вычислить частные производные в точках, заданных в матрице X, от некоторой более глобальной функции, частью которой является рассматриваемый пакет, по известным частным производным в точках (строках) матрицы Y(на выходе пакета), то такой пересчет можно сделать используя выражения (48) – (52).

Таким образом появляется возможность соединения пакетов в виде каскада в многоуровневую вычислительную систему. Последовательно вычисляя пакеты сплайнов можно вычислить значения на выходе всей системы. А также можно выполнять дифференцирование, последовательно пересчитывая частные производные, но уже двигаясь в обратном порядке от последнего пакета в каскаде к первому. Очевидно, что на последнем слое (где непосредственно вычисляется вектор L) чтобы начать процедуру в качестве G_Y нужно взять вектор размерности $(r \times 1)$, состоящий из единиц.

Также по полученным преобразованиям видно, что все вычисления сводятся к набору операций над матрицами, а значит могут быть эффективно произведены с помощью параллельных вычислений с использованием GPU.

Кроме того, все матрицы двумерны. Нигде в вычислениях не используются трехмерные или более тензоры. Это может дать дополнительную возможность. Предположим, что есть сразу несколько вычислительных систем с одинаковой структурой, с одинаковыми, входящими в их состав пакетами сплайнов (с одним и тем же количеством точек в созвездиях). Вычисляемые функции в этих системах, причем на всех этапах в пакетах внутри них, могут быть совершенно разные (значения матриц

C и Λ различные). Но чтобы смоделировать такую многоуровневую систему достаточно просто перейти от двумерных матриц к 3-х мерным тензорам. Все выражения как для прямого прохода вычислений, так и для дифференцирования останутся те же самые.

Заключение.

Таким образом, в данной работе предложена вычислительно эффективная и теоретически обоснованная архитектура на основе каскада пакетов полигармонических сплайнов. В отличие от эвристических глубоких моделей, каждый слой каскада сохраняет строгую интерпретацию через теорию случайных функций и наследует ядро, выведенное из принципов симметрии в [2]. Представленные процедуры прямого и обратного прохода позволяют интегрировать такую архитектуру в современные фреймворки машинного обучения. Методы обучения параметров каскада и экспериментальная оценка его эффективности на бенчмарках будут рассмотрены в последующих публикациях.

Автор благодарит анонимных рецензентов за ценные отзывы о более ранних версиях этой работы.

Список литературы:

1. *Барцев С. И., Охонин В. А.* Адаптивные сети обработки информации. Красноярск : Ин-т физики СО АН СССР, 1986. Препринт N 59Б. — 20 с.
2. Бахвалов Ю. Н. 2025. Решение регрессионной задачи машинного обучения на основе теории случайных функций. PREPRINTS.RU. https://doi.org/10.24108/preprints-3113020
3. Beatson, R. K., & Greengard, L. (1997). A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W. Light, & M. Marletta (Eds.), Wavelets, Multilevel Methods and Elliptic PDEs (pp. 1–37). Oxford University Press.
4. Bookstein, F. L. (June 1989). "Principal warps: thin plate splines and the decomposition of deformations". IEEE Transactions on Pattern Analysis and Machine Intelligence. 11 (6): 567–585. doi:10.1109/34.24792
5. Cho, Y. & Saul, L. K. (2009). Kernel methods for deep learning. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., & Culotta, A. (Eds.), Advances in Neural Information Processing Systems 22, (pp. 342–350)., Cambridge, MA. MIT Press.
6. Fasshauer G.E., Meshfree Approximation Methods with Matlab, World Scientific Publishing, Singapore, 2007.
7. Гихман И. И., Скороход А. В. Теория случайных процессов. М., 1971. Т.1. 664 с

8. Harder R.L. and R.N. Desmarais: Interpolation using surface splines. Journal of Aircraft, 1972, Issue 2, pp. 189−191
9. Matheron, G. (1973). The Intrinsic Random Functions and Their Applications. Advances in Applied Probability, 5(3), 439–468. [DOI: https://doi.org/10.2307/1425829]
10. Rumelhart D.E., Hinton G.E., Williams R.J., Learning Internal Representations by Error Propagation. In: Parallel Distributed Processing, vol. 1, pp. 318—362. Cambridge, MA, MIT Press. 1986
11. Пугачев В.С., Теория случайных функций и её применение к задачам автоматического управления. Изд. 2-ое, перераб. и допол. — М.: Физматлит, 1960.
12. Williams, C. K. I., & Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), Advances in Neural Information Processing Systems 13 (pp. 682–688). MIT Press.
13. Wilson, A. G., Hu, Z., Salakhutdinov, R., & Xing, E. P. (2016). Deep Kernel Learning. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 370–378.