

Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

Высшая школа прикладной математики и вычислительной физики

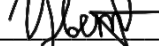
**Отчет по научно-исследовательской работе**

**на тему**

**«Transfer learning»**

Место выполнения:

Научно-исследовательская лаборатория математической биологии  
и биоинформатики

Студент группы 3630102/70401  Цветков А.Д.

Оценка научного руководителя: \_\_\_\_\_ доц. Козлов К.Н.

<внс, к.б.н., Лаборатория математической биологии и биоинформатики>

Санкт-Петербург

2021 г.

## Содержание

|  |    |
|--|----|
| Список схем.....   | 1  |
| Список таблиц.....   | 1  |
| Список иллюстраций.....                                      | 1  |
| Введение .....   | 2  |
| Актуальность темы .....                                      | 2  |
| Цели и задачи работы.....                                    | 2  |
| Описание исходных данных.....                                | 3  |
| Описание изученных методов .....                             | 4  |
| Kernel mean matching (KMM).....                              | 5  |
| Genetic programming (GP) и Differential evolution (DE) ..... | 7  |
| Численные эксперименты и полученные результаты .....         | 9  |
| Оригинальная задача с 10 переменными.....                    | 9  |
| Подготовка входных данных .....                              | 9  |
| Kernel mean matching .....                                   | 10 |
| Результаты численных экспериментов алгоритма ITGP .....      | 11 |
| Эксперимент с задачей меньшей размерности .....              | 14 |
| Выводы.....  | 16 |
| Список литературы.....                                       | 17 |

## Список схем

|  |   |
|--|---|
| Схема 1. Общий вид алгоритма TLGP .....                    | 4 |
| Схема 2. Основные компоненты данных в алгоритме ITGP ..... | 5 |
| Схема 3. Детальный план алгоритма ITGP .....               | 7 |

## Список таблиц

|   |    |
|---|----|
| Таблица 1. Исходные данные.....                 | 9  |
| Таблица 2. Целевые данные.....                  | 9  |
| Таблица 3. Результаты работы алгоритма KMM..... | 11 |

## Список иллюстраций

|  |    |
|--|----|
| Рисунок 1. График зависимости коэффициентов регрессионной модели от значения параметра регуляризации ..... | 10 |
| Рисунок 2. Зависимость точности от количества итераций.....  | 12 |
| Рисунок 3. Зависимость времени от количества итераций .....  | 12 |
| Рисунок 4. Значения параметра $\beta$ .....  | 13 |
| Рисунок 5. Гистограмма распределения значений весов $\beta$ .....  | 13 |
| Рисунок 6. Гистограмма зависимости времени от объема входных данных .....                                  | 15 |
| Рисунок 7. Гистограмма зависимости точности от объема входных данных .....                                 | 15 |

## Введение

Transfer Learning — это подход в машинном обучении, который пытается использовать информацию о хорошо изученном наборе данных для описания нового набора сходных в каких-то параметрах данных. Например, знания, полученные в результате обучения по распознаванию автомобилей, могут быть использованы при обучении распознаванию грузовых автомобилей. [1]

### Актуальность темы

Подход Transfer learning по своей сущности эффективен при обучении на похожих задачах, которые могут встретиться нам даже в повседневной жизни. Конечно, существуют и задачи в биоинформатике, которые могут быть решены таким способом.

Сам по себе transfer learning является относительно новым сценарием обучения в ML и эффективен в двух случаях:

- a. Имеется достаточно исходных данных, но недоступны целевые данные;
- b. Распределение данных меняется со временем.

В последнее время техники transfer learning были внедрены во многих областях, например, для обучения с подкреплением (reinforcement learning) или обучения с учителем (supervised learning). По сравнению с количеством исследований в вышеупомянутых сферах, transfer learning в GPSR (genetic programming with symbolic regression – генетическое программирование с символьной регрессией) встречается редко и до сих пор не рассматривался как важный подход. Поскольку transfer learning является ключом к решению ряда задач биологии и биоинформатики, был предложен новый метод генетического программирования – transfer learning in genetic programming (TLGP).[2]

### Цели и задачи работы

Поставленная задача звучит следующим образом:

Имеется два набора данных: source domain и target domain. Необходимо построить модель для целевых данных, используя знания о исходных данных и их модели.

В нашем случае в качестве исходных и целевых данных выступают dataset'ы с информацией о растениях – датой посадки, датах засухи и дождей, другими агроклиматическими данными. Необходимо построить некоторую аналитическую функцию по данным исходного домена для данных целевого домена, позволяющую определить дату цветения целевого растения.

Для решения поставленной задачи необходимо реализовать частный случай алгоритма TLGP – instance transferring GP (ITGP).

## Описание исходных данных

Поскольку реальные данные о растениях местами могут быть иррациональны или неточны, для проверки работоспособности алгоритма будем использовать данные, сгенерированные по правилу “Friedman”. [3]

В оригинальной задаче Фридмана каждый экземпляр  $X = \{x_1, x_2, \dots, x_{10}\}$  имеет 10 переменных  $x_i$ , где каждая компонента соответствует равномерному распределению  $N(0, 1)$ . Но только первые 5 входных переменных относятся к выходной переменной  $y$ .

Реальное значение данных вычисляется по следующей формуле:

$$y = a_1 \times 10 \sin(\pi(b_1x_1 + c_1) \times (b_2x_2 + c_2)) + a_2 \times 20(b_3x_3 + c_3 - 0.5)^2 + a_3 \times 10(b_4x_4 + c_4) + a_4 \times 5(b_5x_5 + c_5) + N(0, 1)$$

Здесь  $a_i, b_i$  и  $c_i, i \in [1, 5]$  параметры, а  $N$  это нормальное распределение. Для этих параметров устанавливаются разные значения при генерации данных в исходной и целевой областях. Для исходных данных  $a_i = b_i = 1$  и  $c_i = 0$ , в то время как  $a_i, b_i$  взяты из  $N(1, 0.1)$ , а  $c_i$  выбирается из  $N(0, 0.05)$  для целевых (таргетных) данных. [4]

Генерация входных данных реализована с использованием средств языка R в среде разработки RStudio. Исходный код приведен на Github. [5]

## Описание изученных методов

TLGP является сложным составным алгоритмом, сочетающим в себе несколько важных подходов к решению промежуточных задач для поиска оптимального ответа основного задания.

Рассмотрим вначале схему модулей, на которые разбит основной изучаемый алгоритм (см. схему 1).

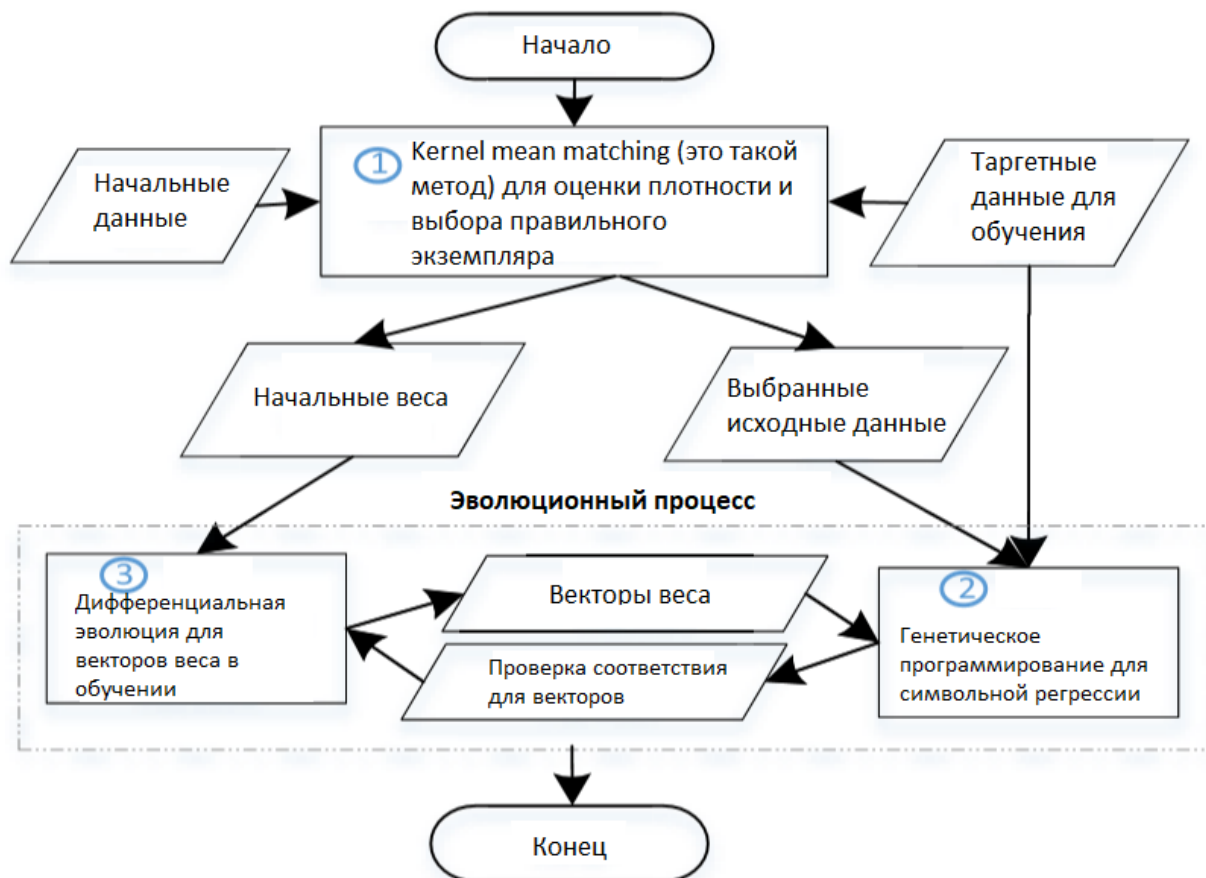


Схема 1. Общий вид алгоритма TLGP

1. На начальном этапе алгоритма с помощью подхода kernel mean matching ищется набор векторов веса для исходных данных, а также вычисляются особенно важные и совсем не важные для обучения экземпляры (для первых веса больше, для вторых – стремятся к нулю). Весовые векторы определяются как  $V_i = \{[v_{1,d}, v_{1,w}], [v_{2,d}, v_{2,w}], \dots, [v_{n,d}, v_{n,w}]\}$ .  $V_i$  – вектор из  $n$  элементов. У каждого элемента два измерения – индекс экземпляра в исходных данных и вес этого экземпляра. [6]  
Мы считаем, что имеем  $n$  входных экземпляров данных (source) и  $m$  разных векторов весов к ним.
2. На основном этапе алгоритма происходит параллельная работа двух методов: разностной эволюции (DE) и генетического программирования (GP). Получая обратную связь друг от друга, происходит эволюционный сходящийся процесс. Разностная эволюция на основе обратной связи от генетического программирования формирует новый вектор весов, опираясь на значения ошибки («векторов соответствия») для каждой модели, сгенерированной алгоритмом GP. В свою очередь, GP генерирует  $p$  моделей на основе входных данных и весов к ним, в процессе алгоритма выбирая  $m$  лучших моделей (с наименьшей ошибкой).
3. Условием выхода из алгоритма может служить либо достижение заданного числа итераций, либо неизменность весов/модели в течение заданного числа эволюций.

Основными компонентами данных в алгоритме ITGP выступают весовые векторы и векторы соответствия (векторы ошибки, fitness vectors). В основном эти данные используются для тестирования моделей, построенных алгоритмом генетического программирования. Векторы ошибки позволяют оценить модели с наименьшей погрешностью, а весовые векторы позволяют построить модели на основе взвешенных данных исходного домена (см. схему 2). [6]

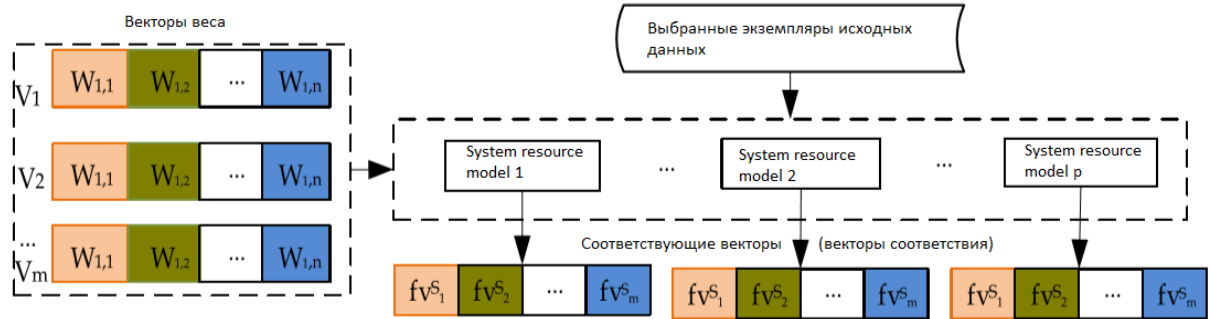


Схема 2. Основные компоненты данных в алгоритме ITGP

Рассмотрим каждый из блоков алгоритма подробнее.

### Kernel mean matching (KMM)

Пристальное внимание необходимо уделить методу поиска начального опорного вектора для начала работы основного алгоритма. Идея KMM заключается в том, чтобы, используя функцию ядра (1), (2) (чаще всего используется радиально-базисная функция), оценить коэффициент отношения плотности путем минимизации максимального среднего расхождения (MMD). (3)

$$K_{i,j} = k(x_i^s, x_j^s) \quad (1)$$

$$k_i = \frac{n_s}{n_t} \sum_{j=1}^{n_t} k(x_i^s, x_j^t) \quad (2)$$

Где  $x_i^s$  – экземпляр исходного домена,  $x_j^t$  – экземпляр целевого домена,  $k$  – ядро,  $n_s, n_t$  – количество исходных и целевых экземпляров соответственно.

MMD можно получить с помощью следующей формулы:

$$MMD = \left\| \frac{1}{n_s} \sum_{i=1}^{n_s} \beta_i \Phi(x_i^s) - \frac{1}{n_t} \sum_{i=1}^{n_t} \Phi(x_i^t) \right\|^2 = \frac{1}{n_s^2} \beta^T K \beta - \frac{2}{n_s^2} k^T \beta + constant \quad (3)$$

Подходящий набор весов  $\beta$  можно найти, решив задачу минимизации:

$$minimize \left( \frac{1}{2} \beta^T K \beta - k^T \beta \right) \quad (4)$$

Где  $\beta \in (0, B)$  и  $|\sum_{i=1}^{n_s} \beta_i - n_s| \leq n_s \epsilon, \epsilon = O\left(\frac{B}{\sqrt{n_s}}\right)$

Экспериментально выяснено, что вектор  $B = \{B_1, B_2, \dots, B_m\}$  должен представлять собой набор  $m$  различных значений ( $B \leq 1000$ ). [7]

В процессе машинного обучения, в частности, в подходе TL, имеет место быть вероятность доминирования данных исходной области. Поэтому в данном случае выбирается лишь фиксированное число экземпляров исходной области, имеющих наибольшие веса. Число выбранных экземпляров исходного домена  $n$  выбирается с помощью  $n_t$  и параметра масштаба  $t$ ,

который определяется из  $n = n_t \times t$ . Ряд экспериментов показывает, что значение  $t \leq 5$  является хорошим выбором.

*Использование метода Лассо*

Переформулируем нашу задачу с учетом ограничений:  $|\sum_{i=1}^{n_s} \beta_i - n_s| \leq n_s \epsilon$

Тогда будем решать такую задачу минимизации:  $\text{minimize} \left( \frac{1}{2} \beta^T K \beta - k^T \beta + \lambda |\sum_{i=1}^{n_s} \beta_i - n_s| \right)$

Таким образом, будем использовать  $\lambda$  как коэффициент штрафной функции. Коэффициент  $\lambda$  найдем для решения следующей задачи:

$$\beta = \arg \min \left( \sum_{i=1}^n \left( k_i - \sum_{j=1}^m \beta_j K_{ij} \right)^2 + \lambda |\beta| \right)$$

Построим график зависимости коэффициентов регрессионной модели от значения параметра регуляризации, чтобы узнать оптимальный интервал для  $\lambda$ .

Затем решим  $m$  задач минимизации для разных  $\lambda$  и получим искомые данные.

Ниже приведен алгоритм КММ.

**Алгоритм: КММ для инициализации векторов весов и выбора нужных экземпляров в ITGP**

---

*Вход:*  $n_s$  – какое-то количество экземпляров начальных данных из области  $S(X)$ ,  $n_t$  целевых экземпляров  $T(X)$ , параметр масштаба  $t$ , количество векторов веса  $m$ , набор  $B = \{B_1, B_2, \dots, B_m\}$ ;

*Выход:* набор векторов веса  $V = \{V_1, V_2, \dots, V_m\}$

**for**  $q := 1$  **to**  $m$  **do** получение весов экземпляров из  $n_s$  исходных доменов с помощью КММ и выбор исходных экземпляров

Инициализируем  $V_q$  длинами из  $n = n_t \times t$

Ищем  $K_{i,j}$  из формулы (1), с использованием радиально-базисной функции;

Ищем  $k_i$  с помощью формулы (2)

Вычисляем  $\epsilon = \frac{B_q}{\sqrt{n_s}}$

Ищем  $\beta$  = решение квадратичной задачи  $(K_{i,j}, k_i, B_q, \epsilon)$  – по уравнению (4)

Ранжируем  $\beta = \{\beta_o, \beta_p, \dots, \beta_q\}$ , где  $\beta_o > \beta_p > \beta_q$  – где  $o, p, q$  – индексы исходных экземпляров;

**for**  $r := 1$  **to**  $n_t \times t$  **do**

$V_{q,r0} = \text{index}(\beta_r)$

$V_{q,r1} = \beta_r$

Добавляем  $V_q$  в  $V$ ;

Возвращаем  $V = \{V_1, V_2, \dots, V_m\}$ ;

---

Так, получим, что КММ произвел входные данные для алгоритма разностной эволюции (векторы веса) и генетического программирования (отобрал важные экземпляры).

## Genetic programming (GP) и Differential evolution (DE)

Итак, мы получили входные данные для алгоритма с помощью КММ. Алгоритм разностной эволюции принимает  $m$  векторов весов (где каждый вектор содержит  $n$  измерений, соответствующих весам для  $n$  выбранных экземпляров исходных данных) в качестве начальных данных, а GP сразу запускается и строит  $p$  деревьев (моделей).

Тогда процесс эволюции векторов, показанный на Схема 3, состоит из следующих этапов: [8]

1. GP оценивает  $p$  элементов/моделей на выбранных экземплярах исходных данных, и для каждой модели существует  $m$  взвешенных значений ошибок (соответствующих  $m$  весовым векторам).
2. GP выбирает  $m$  моделей, сравнивая векторы ошибок всех моделей в каждом из  $m$  весовых векторов с соответствующими исходными экземплярами. Для каждого вектора весов и соответствующего ему вектора исходных экземпляров  $k$  мы можем найти модель с наименьшей взвешенной ошибкой  $f v_k^S$ . Эта модель считается лучшей в измерении  $k$ . Иначе говоря, лучшая модель в измерении  $k$  имеет наименьшее  $f v_k^S$  (см. Схема 2). Для вычисления взвешенной ошибки используется следующая формула:

$$f v_i^S = WMSE = \sum_{j=1}^n w_j \cdot (f_j - y_j)^2, (i = \{1, 2, \dots, m\}),$$
 где  $n$  – количество выбранных экземпляров исходного домена,  $w_j$  – вес  $j$ -ого выбранного экземпляра,  $f_j$  – значение, посчитанное моделью,  $y_j$  – реальное значение  $j$ -ого экземпляра.

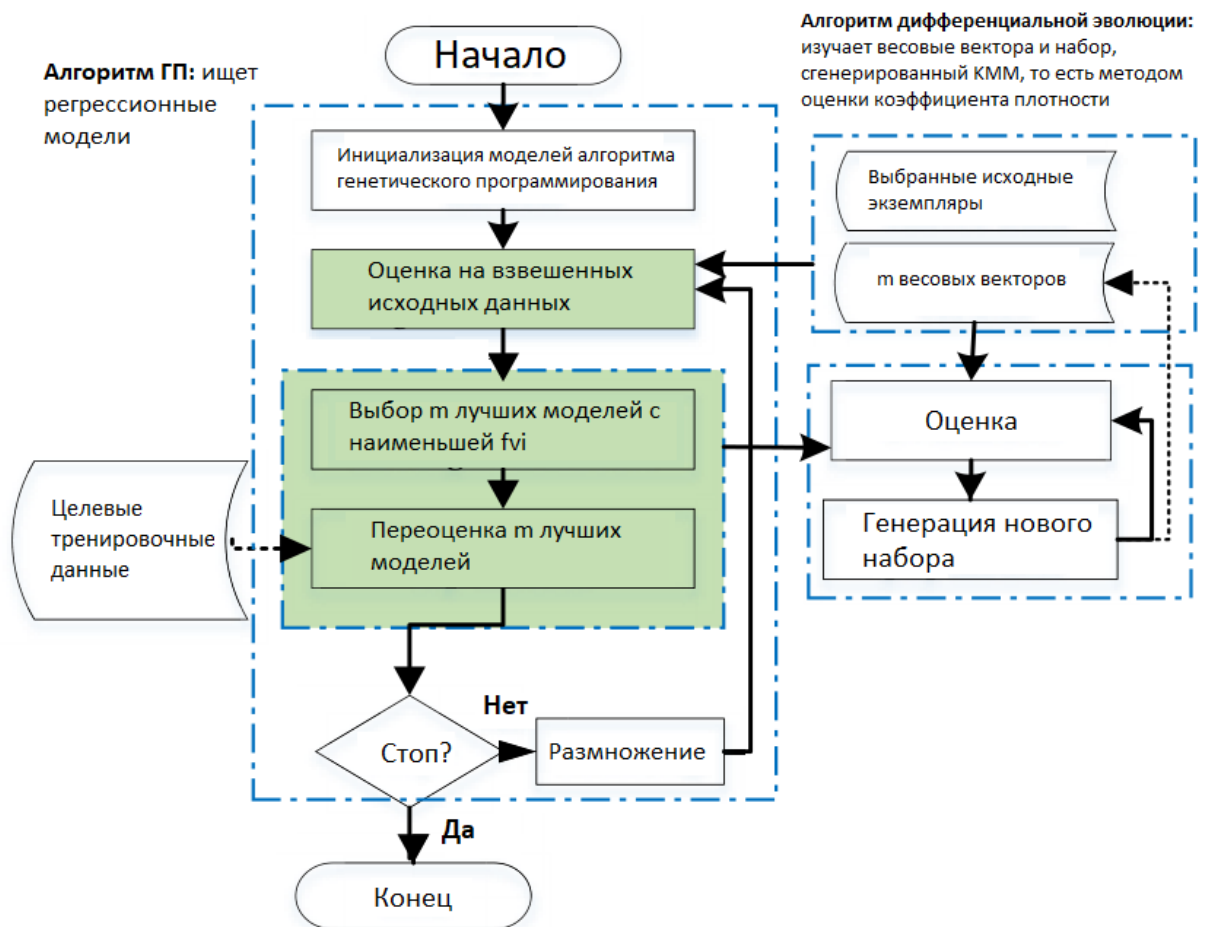


Схема 3. Детальный план алгоритма ITGP



3. GP повторно оценивает  $m$  моделей, измеряя их ошибки уже на экземплярах целевой области. В данном случае ошибка уже не взвешенная и вычисляется по следующей формуле:

$$fv^T = MSE = \frac{1}{n_i} \sum_{i=1}^{n_t} (f_i - y_i)^2, \text{ где } n_t - \text{количество целевых экземпляров для обучения, } f_i$$

– прогнозируемое значение модели в  $i$ -ом экземпляре,  $y_i$  – соответствующее целевое значение (полученное экспериментально).

4. DE собирает обратную связь от GP, то есть получает на вход значения ошибок  $m$  моделей и генерирует новые векторы весов с учетом информации о погрешности моделей. Для генерации новых весовых векторов SaDE (улучшенный алгоритм дифференциальной эволюции) используются четыре общепринятые и эффективные стратегии генерации нового весового вектора, которые вычисляются так:

$$V_i = W_{r1} + F \cdot (W_{r2} - W_{r3})$$

$$V_i = W_{best} + F \cdot (W_{r2} - W_{r3})$$

$$V_i = W_i + K \cdot (W_{r3} - W_i) + F \cdot (W_{r1} - W_{r2})$$

$$V_i = W_i + F \cdot (W_{best} - W_i) + F \cdot (W_{r1} - W_{r2})$$

Здесь индексы  $r1, r2, r3$  являются векторами случайно выбранных элементов.  $K$  и  $F$  являются коэффициентами масштаба, которые обычно имеют положительное значение для масштабирования вектора разности.

Повторяем шаги 1-4, пока не будет достигнуто условие окончания вычислений. В качестве ответа будет выбрана модель, имеющая наименьшую ошибку на целевых данных.

## Численные эксперименты и полученные результаты

### Оригинальная задача с 10 переменными

#### Подготовка входных данных

Рассмотрим результат генерации исходных и целевых данных, согласно алгоритму Фридмана. Для генерации коэффициентов функции  $(a, b, c)$  и вектора  $X = \{x_1, x_2, \dots, x_{10}\}$  были использованы встроенные функции языка R для математической статистики: *rnorm* (создание выборки, подчиняющейся нормальному распределению) и *runif* (создание выборки, подчиняющейся равномерному распределению).

Разработчиком алгоритма было принято решение хранить векторы исходных данных в виде таблицы формата csv (разделитель – запятая).

|    | x_1      | x_2      | x_3      | x_4      | x_5      | x_6      | x_7      | x_8      | x_9      | x_10     | y        |
|----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1  | 0.380837 | 0.75423  | 0.447185 | 0.94852  | 0.85604  | 0.238693 | 0.522789 | 0.462424 | 0.482758 | 0.069515 | 23.71803 |
| 2  | 0.085485 | 0.790889 | 0.512323 | 0.203429 | 0.075902 | 0.958262 | 0.432474 | 0.655098 | 0.305363 | 0.484506 | 5.548117 |
| 3  | 0.94509  | 0.16041  | 0.82559  | 0.881963 | 0.641696 | 0.431505 | 0.702839 | 0.368749 | 0.497654 | 0.311957 | 19.29851 |
| 4  | 0.706434 | 0.989738 | 0.921867 | 0.274793 | 0.779752 | 0.00509  | 0.144677 | 0.113349 | 0.09152  | 0.906402 | 21.0587  |
| 5  | 0.348638 | 0.686279 | 0.748478 | 0.677423 | 0.336529 | 0.415651 | 0.058114 | 0.329642 | 0.973752 | 0.115129 | 13.23912 |
| 6  | 0.014769 | 0.577196 | 0.44931  | 0.947952 | 0.904107 | 0.587485 | 0.607724 | 0.214628 | 0.12854  | 0.057667 | 13.77132 |
| 7  | 0.705334 | 0.08329  | 0.381832 | 0.26374  | 0.845097 | 0.086579 | 0.515771 | 0.124779 | 0.037613 | 0.812063 | 7.56161  |
| 8  | 0.560836 | 0.998816 | 0.691137 | 0.766862 | 0.887151 | 0.947449 | 0.55735  | 0.00635  | 0.326479 | 0.55384  | 20.9555  |
| 9  | 0.647755 | 0.499935 | 0.010644 | 0.056248 | 0.525915 | 0.873604 | 0.030827 | 0.870224 | 0.550656 | 0.45517  | 15.28147 |
| 10 | 0.91869  | 0.828899 | 0.805583 | 0.529191 | 0.443504 | 0.204089 | 0.301116 | 0.696339 | 0.693826 | 0.593874 | 19.43896 |
| 11 | 0.398092 | 0.962579 | 0.01043  | 0.990209 | 0.52436  | 0.650917 | 0.175379 | 0.146344 | 0.658319 | 0.469557 | 21.12075 |
| 12 | 0.150899 | 0.964985 | 0.025419 | 0.289254 | 0.774618 | 0.093869 | 0.408496 | 0.376268 | 0.529377 | 0.994299 | 14.437   |
| 13 | 0.229646 | 0.870364 | 0.526341 | 0.091455 | 0.64615  | 0.766427 | 0.469331 | 0.910421 | 0.840025 | 0.737864 | 7.73429  |
| 14 | 0.343991 | 0.048751 | 0.958892 | 0.128578 | 0.052029 | 0.677557 | 0.711865 | 0.09227  | 0.481394 | 0.854733 | 6.69448  |
| 15 | 0.694213 | 0.848042 | 0.886137 | 0.598886 | 0.913772 | 0.74524  | 0.149246 | 0.367631 | 0.128594 | 0.14606  | 20.63044 |
| 16 | 0.458343 | 0.767742 | 0.539029 | 0.006212 | 0.824111 | 0.090533 | 0.445735 | 0.388447 | 0.903705 | 0.712218 | 9.542767 |
| 17 | 0.433214 | 0.76964  | 0.443448 | 0.499776 | 0.270709 | 0.578164 | 0.500941 | 0.855871 | 0.257751 | 0.886534 | 17.71495 |
| 18 | 0.176089 | 0.973622 | 0.384141 | 0.493973 | 0.409944 | 0.484841 | 0.609349 | 0.550576 | 0.986803 | 0.831912 | 10.32747 |
| 19 | 0.550909 | 0.446313 | 0.252674 | 0.987898 | 0.604186 | 0.857764 | 0.9305   | 0.682095 | 0.544781 | 0.109338 | 19.66449 |
| 20 | 0.724138 | 0.310724 | 0.509072 | 0.668943 | 0.378111 | 0.633603 | 0.451571 | 0.079988 | 0.433144 | 0.702999 | 15.966   |

Таблица 1. Исходные данные [9]

|    | x_1      | x_2      | x_3      | x_4      | x_5      | x_6      | x_7      | x_8      | x_9      | x_10     | y        |
|----|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1  | 0.985106 | 0.905632 | 0.753788 | 0.58832  | 0.99088  | 0.621225 | 0.825614 | 0.369627 | 0.943716 | 0.518136 | 15.61763 |
| 2  | 0.521438 | 0.633654 | 0.959915 | 0.784232 | 0.909114 | 0.481397 | 0.48241  | 0.128901 | 0.017297 | 0.364995 | 25.23199 |
| 3  | 0.98698  | 0.389296 | 0.505244 | 0.702653 | 0.254002 | 0.79166  | 0.223391 | 0.395437 | 0.103808 | 0.806401 | 18.16488 |
| 4  | 0.535986 | 0.912412 | 0.515321 | 0.312044 | 0.48132  | 0.922515 | 0.662374 | 0.474312 | 0.315256 | 0.466174 | 15.70909 |
| 5  | 0.358245 | 0.3147   | 0.219261 | 0.85515  | 0.396917 | 0.751468 | 0.372913 | 0.029794 | 0.250952 | 0.288607 | 16.29654 |
| 6  | 0.284291 | 0.229106 | 0.208702 | 0.364179 | 0.291152 | 0.114277 | 0.528506 | 0.074527 | 0.405154 | 0.738456 | 7.624411 |
| 7  | 0.498092 | 0.340759 | 0.386031 | 0.945184 | 0.423097 | 0.10914  | 0.372682 | 0.400851 | 0.761905 | 0.763413 | 14.48147 |
| 8  | 0.896497 | 0.014072 | 0.789573 | 0.088116 | 0.088384 | 0.595491 | 0.891841 | 0.971277 | 0.020671 | 0.993446 | 4.100136 |
| 9  | 0.544641 | 0.532373 | 0.793533 | 0.322849 | 0.309523 | 0.431199 | 0.61294  | 0.509801 | 0.789595 | 0.536144 | 14.13796 |
| 10 | 0.886771 | 0.264258 | 0.490774 | 0.89975  | 0.485954 | 0.744396 | 0.85599  | 0.372913 | 0.778592 | 0.191572 | 18.13417 |

Таблица 2. Целевые данные [10]

Соотношение размерностей Таблицы 1 и Таблицы 2 иллюстрирует, что параметр  $t = 2$ . В последнем столбце (в последнем элементе вектора строки) хранятся данные о значении аналитической функции, зависящей от параметров  $X = \{x_1, x_2, \dots, x_{10}\}$ .

Также можно заметить, что данные в таблицах похожи. Соответственно, можно утверждать, что для таких датасетов применим алгоритм transfer learning.

Файлы с полученными данными приведены на Github. [9][10]

### Kernel mean matching

Поскольку задача минимизации была переформулирована, необходимо найти оптимальный интервал для коэффициента штрафной функции  $\lambda$ .

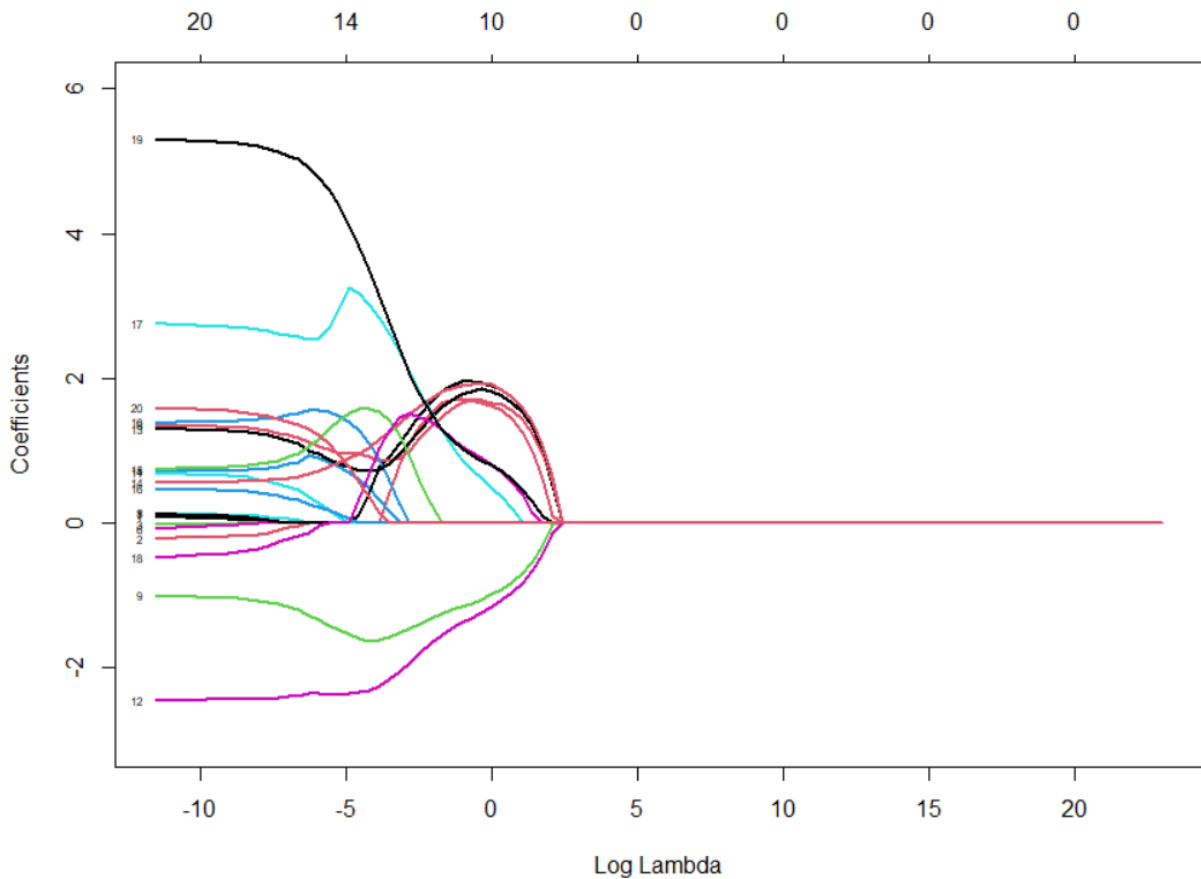


Рисунок 1. График зависимости коэффициентов регрессионной модели от значения параметра регуляризации

Лассо-модель была принята,  $\lambda \in [1.5, 4]$ . Количество различных коэффициентов штрафной функции было выбрано равным размерности каждого экземпляра, то есть  $m = 10$ .

Итак, в результате работы алгоритма для каждого  $\lambda$  было получено  $n_s = 20$  весовых коэффициентов с условием  $\beta \in [0, 1000]$ , которые послужат компонентами опорного вектора для основного алгоритма. Каждое значение  $\beta$  показывает, какой вклад вносит тот или иной экземпляр исходных данных. Для наглядности матрица весовых коэффициентов была подобрана не содержащей нулевые значения.

Выходными данными алгоритма являются две матрицы: первая показывает веса в порядке убывания вклада для каждого  $\lambda$ ; вторая хранит индексы исходных экземпляров, к которым относится тот или иной весовой коэффициент (см. Таблица 3). Так, например, 19й экземпляр при любых значениях коэффициента штрафной функции получал наибольший вес, то есть имел наибольший вклад.



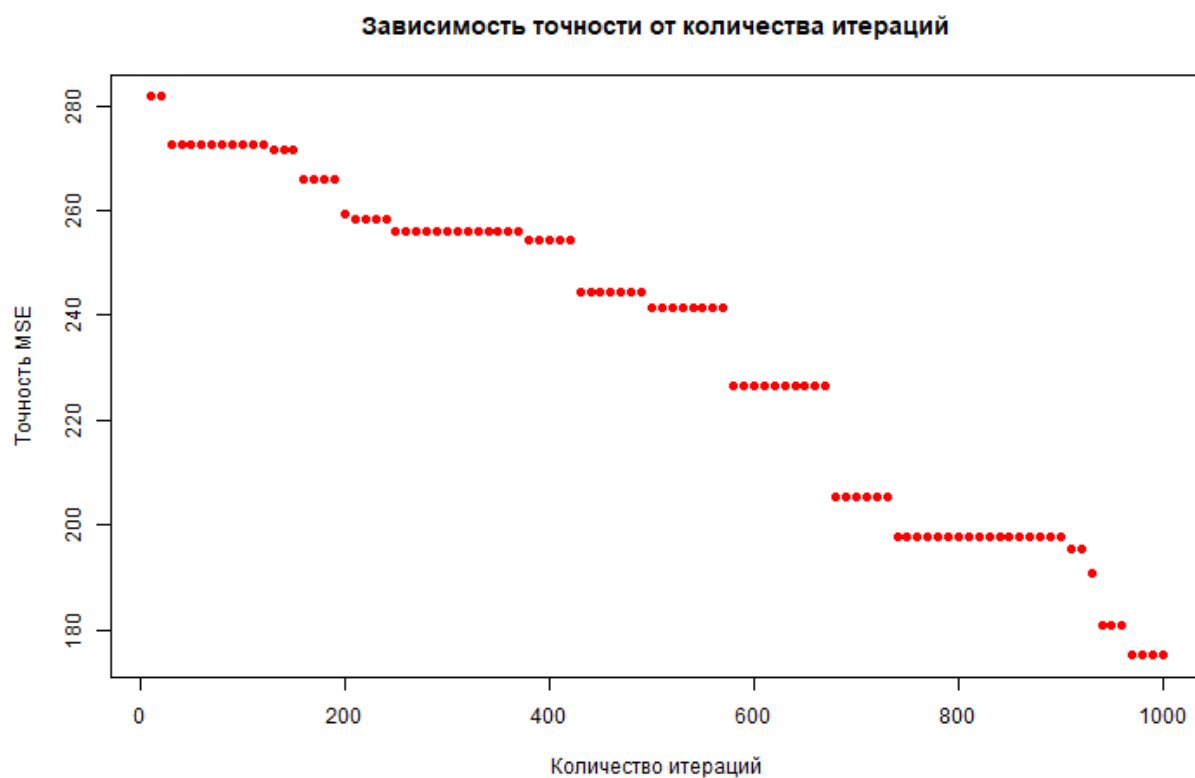


Рисунок 2. Зависимость точности от количества итераций

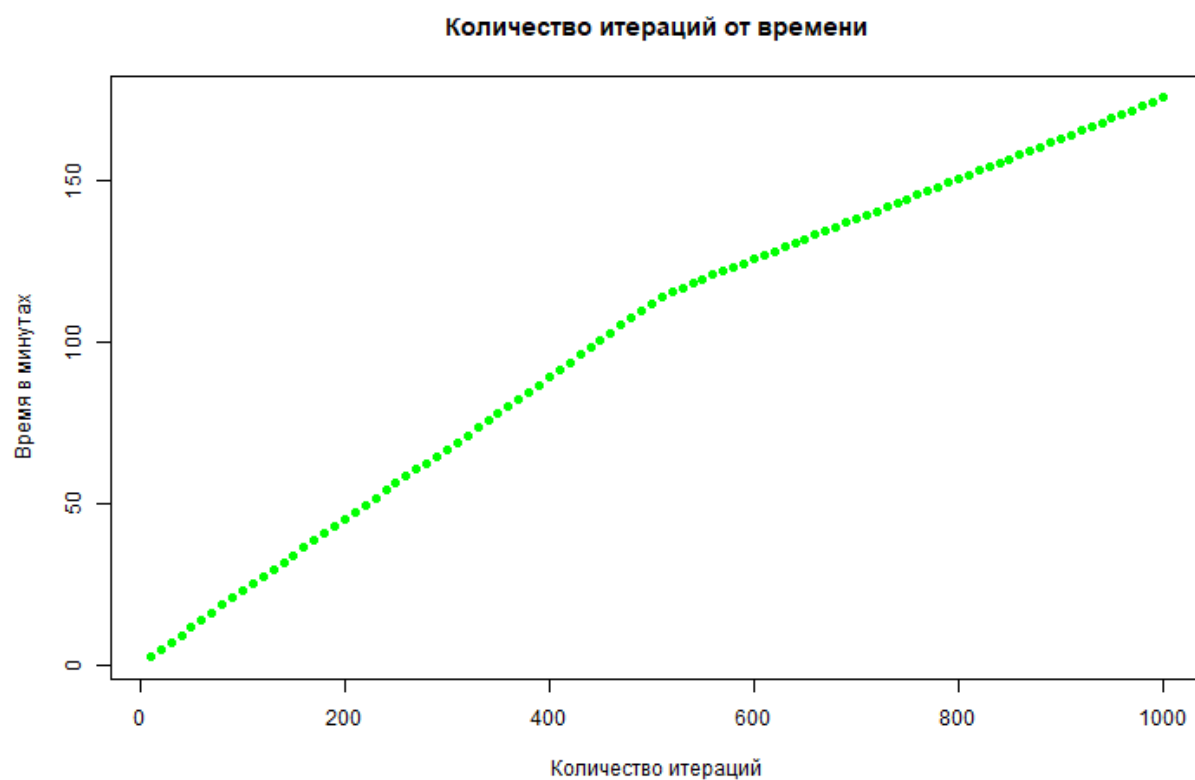


Рисунок 3. Зависимость времени от количества итераций

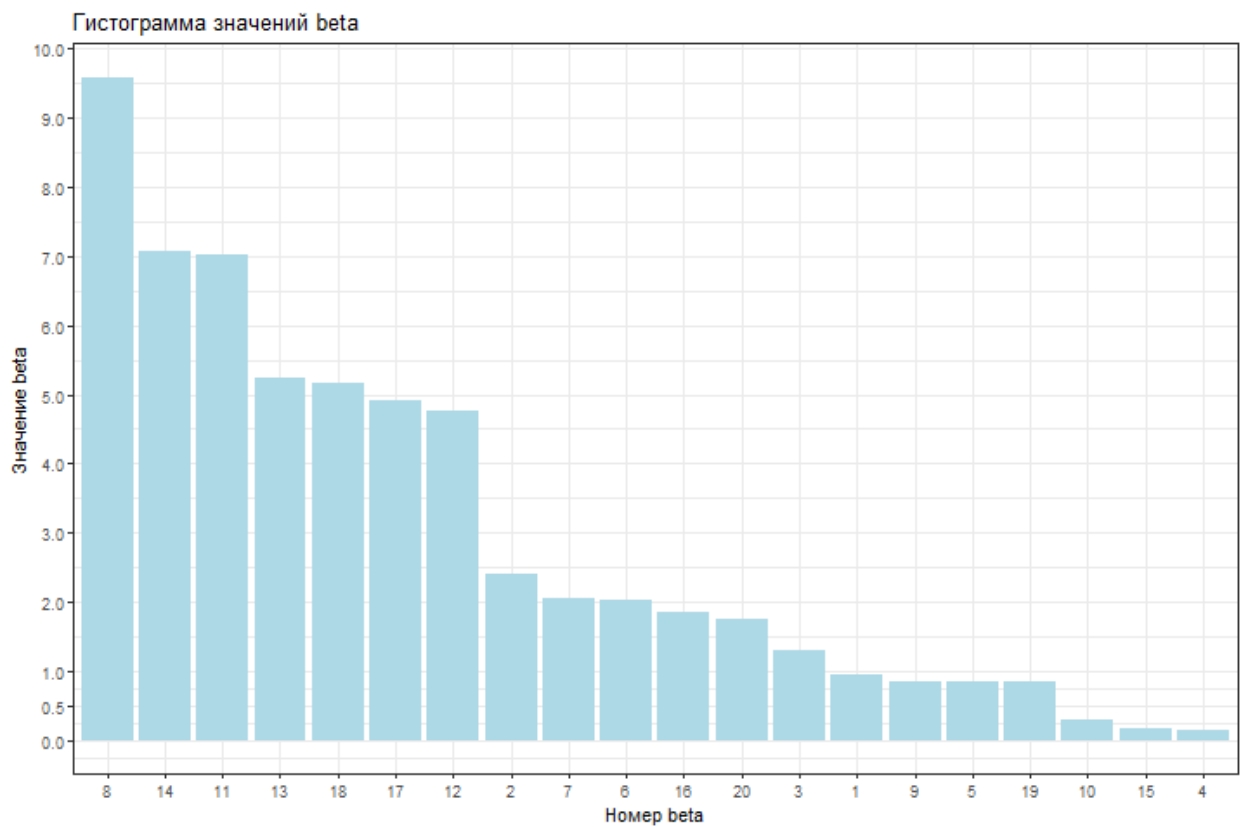


Рисунок 4. Значения параметра  $\beta$

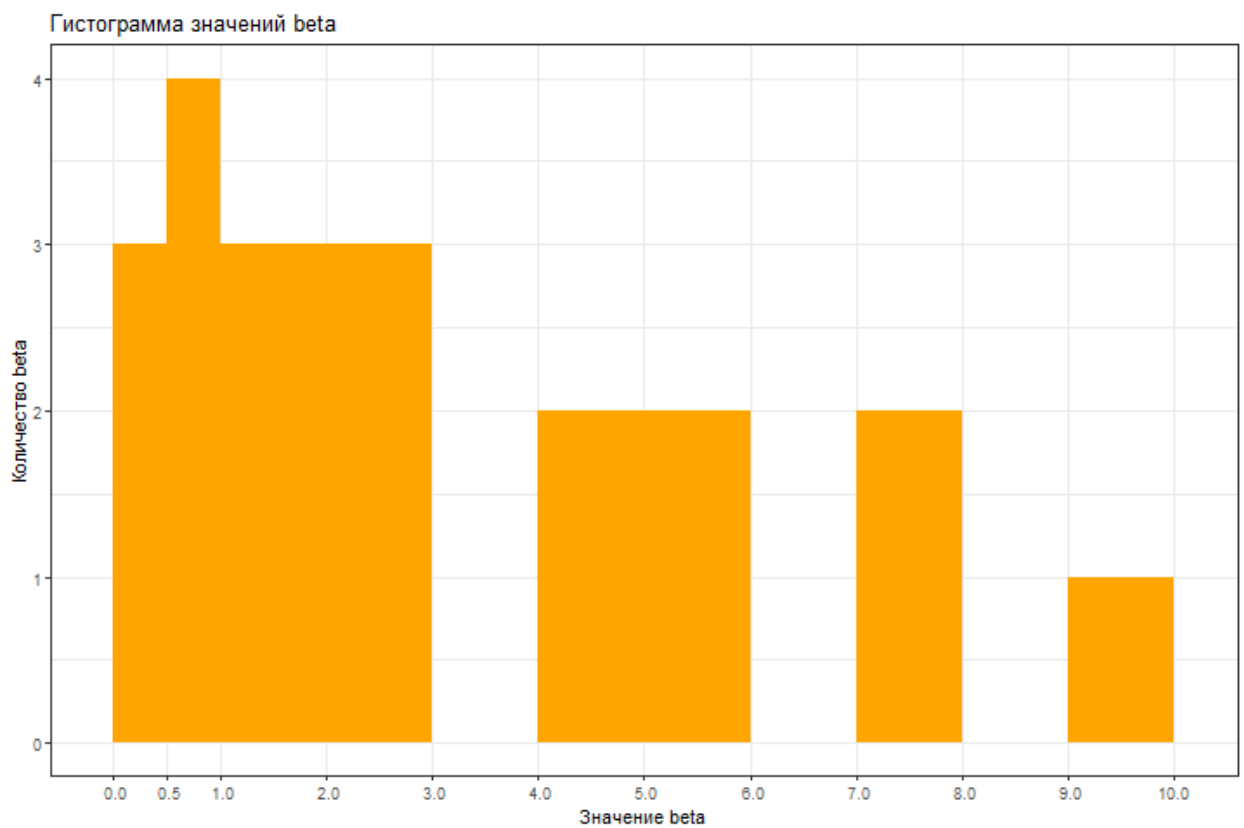


Рисунок 5. Гистограмма распределения значений весов  $\beta$

Для дерева (1) была построена гистограмма значений весов  $\beta$ . По гистограмме (рис. 4) можно заметить, что вклады исходных данных по сравнению с входным опорным вектором значительно

поменялись. Те экземпляры, которые считались значимыми в начале работы алгоритма, при завершении оказываются не несущими большого вклада, и наоборот. Также можно заметить, что во входном векторе параметров  $\beta$  много весовых значений, близких к нулю. В выходном же векторе весовых коэффициентов это не так.

По гистограмме распределения весов (рис. 5) можно видеть, что экземпляры в основном вносят небольшой вклад - в интервале  $[0, 1]$  находятся целых 7 весов. Более-менее значимый вклад ( $\geq 4$ ) вносят также 7 экземпляров.

Наилучший результат алгоритма, который удавалось получить варьированием параметров генетического программирования и разностной эволюции, имел ошибку 0.0607 при дереве

$$f(x) = \frac{x_3}{0.078} + (x_3 - (x_3 + 0.427) \times (x_9 \times (0.138 - x_5) + (0.909 + 0.451 \times x_4) + (\frac{0.360}{x_1} - \frac{x_7}{x_6}) \\ + (x_7 - 0.385 \times x_3) - (x_3 + x_1)) + (\frac{x_4}{0.081} + x_1 \\ + \left( 0.740 + \left( x_1 + x_6 - \frac{(x_{10} - (x_8 - x_3) \times (x_9 \times (x_7 - x_9)) + 0.749)}{x_2} \right) \right) \times x_3) \\ - 0.768)$$

Исходный код программы приведен на GitHub. [12]

### Эксперимент с задачей меньшей размерности

Для эксперимента над объемом вычислений была взята задача меньшей размерности – функция 5 переменных вида

$$y = a_1 \times 5 \cos(\pi(b_1x_1 + c_1) \times (b_2x_2 + c_2x_4)) + a_2 \times 8(b_3x_3 + c_3x_5 - 0.5)^2 + N(0, 1)$$

где  $\{x_1, \dots, x_5\}$  подчиняются нормальному распределению, для исходных данных  $a_i = b_i = 1$  и  $c_i = 0$ , в то время как  $a_i, b_i$  взяты из  $N(1, 0.1)$ , а  $c_i$  выбирается из  $N(0, 0.05)$  для целевых данных.

Заметим, что в данном случае  $y$  зависит от всех 5 переменных (в исходной задаче  $y$  не зависел от 5/10 переменных).

Для исходных данных был взят параметр  $t = 2$ . Количество исходных данных варьировалось от 20 до 50 экземпляров. Количество итераций алгоритма = 500.

По рис. 6 можно видеть, что при увеличении количества обрабатываемых данных скорость работы алгоритма практически линейно уменьшается.

По рис. 7 видно, что значительное увеличение точности наблюдается при 30 source экземплярах, далее увеличение точности на 500 итерациях замедляется.

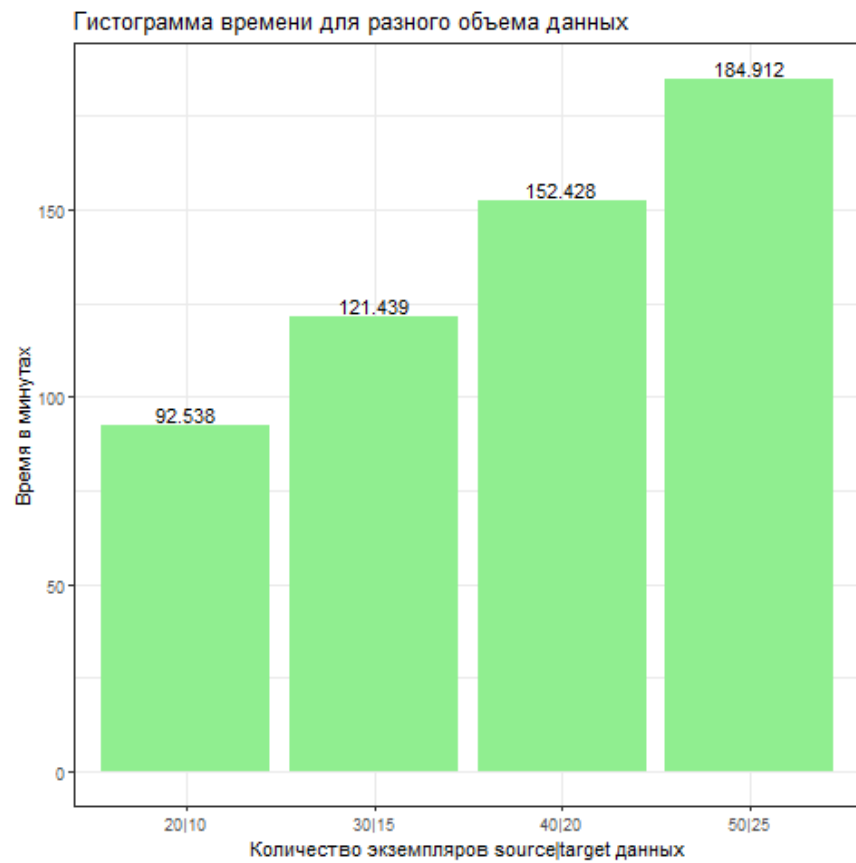


Рисунок 6. Гистограмма зависимости времени от объема входных данных

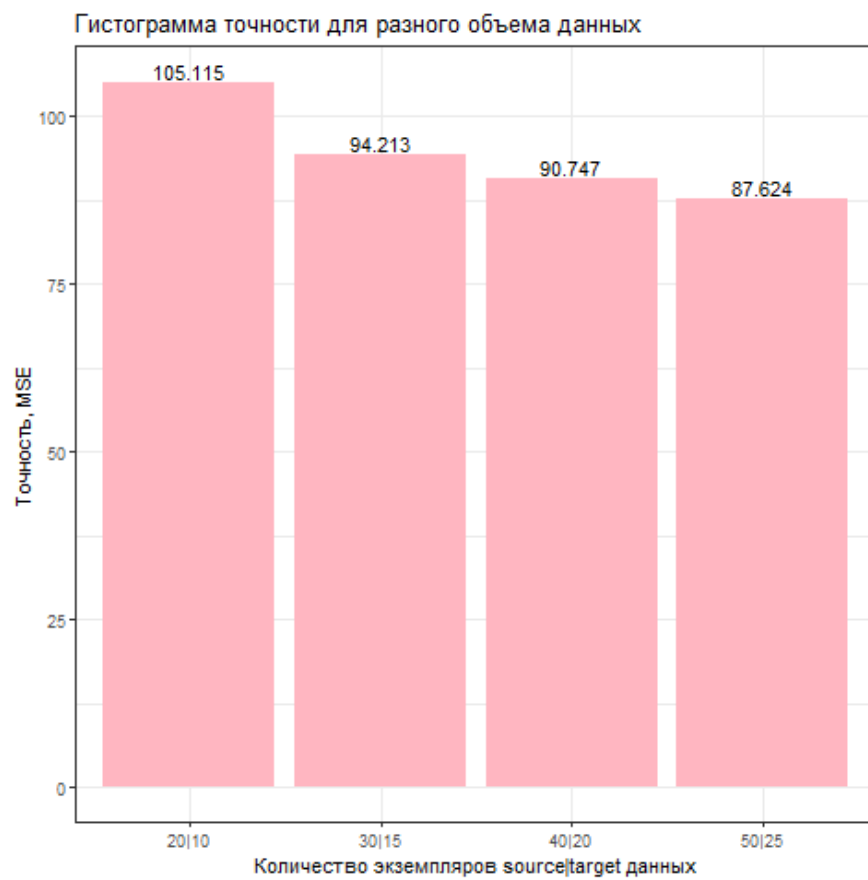


Рисунок 7. Гистограмма зависимости точности от объема входных данных



## Выводы

Реализованный алгоритм ITGP обладает небольшой итерационной сходимостью за счет присутствия в алгоритме элементов случайности. Алгоритм работает довольно медленно, однако с увеличением количества итераций ускоряется, также можно гарантировать, что заданное значение точности будет найдено. Большое влияние на скорость увеличения точности имеет входной вектор деревьев и его изначальная ошибка. Для каждой задачи с учетом ее специфики рекомендуется варьировать значения параметров как для основного алгоритма, так и для вспомогательных (GP и DE) в целях увеличения эффективности.

## Список литературы

- [1] [https://en.wikipedia.org/wiki/Transfer\\_learning](https://en.wikipedia.org/wiki/Transfer_learning)
- [2] Qi Chen, Bing Xue, Mengjie Zhang, «Genetic programming for Instance Transfer Learning in Symbolic Regression», 2020, p. 1
- [3] Qi Chen, Bing Xue, Mengjie Zhang, «Genetic programming for Instance Transfer Learning in Symbolic Regression», 2020, p. 7
- [4] D. Pardoe and P. Stone, “Boosting for regression transfer,” in Proc. 27<sup>th</sup> Int. Conf. Mach. Learn., 2010, pp. 863–870.
- [5] [https://github.com/adtsvetkov/Transfer\\_learning/blob/master/Preparing%20data.R](https://github.com/adtsvetkov/Transfer_learning/blob/master/Preparing%20data.R)
- [6] Qi Chen, Bing Xue, Mengjie Zhang, «Genetic programming for Instance Transfer Learning in Symbolic Regression», 2020, p. 4
- [7] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf, “Correcting sample selection bias by unlabeled data,” in Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 601–608.
- [8] Qi Chen, Bing Xue, Mengjie Zhang, «Genetic programming for Instance Transfer Learning in Symbolic Regression», 2020, p. 5
- [9] [https://github.com/adtsvetkov/Transfer\\_learning/blob/master/source\\_domain.csv](https://github.com/adtsvetkov/Transfer_learning/blob/master/source_domain.csv)
- [10] [https://github.com/adtsvetkov/Transfer\\_learning/blob/master/target\\_domain.csv](https://github.com/adtsvetkov/Transfer_learning/blob/master/target_domain.csv)
- [11] [https://github.com/adtsvetkov/Transfer\\_learning/blob/master/KMM\\_results.xlsx](https://github.com/adtsvetkov/Transfer_learning/blob/master/KMM_results.xlsx)
- [12] [https://github.com/adtsvetkov/Transfer\\_learning/blob/master/main.R](https://github.com/adtsvetkov/Transfer_learning/blob/master/main.R)