|  |  |
| --- | --- |
|  | **УТВЕРЖДАЮ**  Профессор кафедры  ИАНИ ННГУ, д.т.н.  \_\_\_\_\_\_\_\_\_\_\_\_\_ Старостин Н.В.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г. |

**Отчёт по НИР**

**«Разработка и реализация программного обеспечения для решения задачи многомерной аппроксимации функции»**

**(Шифр ПО «APPROX»)**

Ответственный исполнитель

\_\_\_\_\_\_\_\_\_\_\_ Лобанкина К. В.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

**Н. Новгород 2019**

Оглавление

[**1.** **Постановка задачи** 3](#_Toc26830911)

[**2.** **Математическая модель** 3](#_Toc26830912)

[**3.** **Метод решения** 3](#_Toc26830913)

[**4.** **Описание разработки** 4](#_Toc26830914)

[**5.** **Использованные методы решения** 4](#_Toc26830915)

[**6.** **Результаты** 6](#_Toc26830916)

**Постановка задачи**

Имеется функция и рассчитанные некоторые её значения. Необходимо вычислить приближение этой функции таким образом, чтобы количество рассчитываемых для этого точек было минимальным.

1. **Математическая модель**

**Входные данные:**

,

– размерности пространств;

 – задают область определения , где ;

 – множество точек, для которых рассчитаны ранее значения функции ;

– максимально возможное число параллельно рассчитываемых точек;

– задает точность аппроксимации функции.

**Выходные параметры:**

 – текущее приближение («дешевая» с точки зрения вычисления значений в точках) исследуемой функции ;

– оценка точности текущего приближения ;

 – множество очередных точек, для которых необходимо рассчитать значение функции .

**Цель:** построение стратегии выбора рассчитываемых точек, обеспечивающих вычисление приближения функции , минимизируя при этом количество рассчитываемых точек.

1. **Метод решения**

Пусть задан – шаг сетки в области определения функции . Пусть .

1. Введем отображение (диаграмма Вороного) , которое каждой точке  ставит в соответствие ближайшую точку . Обратное отображение  определяет область пространства из точек, для которых ближайшей является . Эти множества  будем назвать доменами.

2. Строим неориентированный граф  соседства точек, в котором вершины соответствуют точкам , а ребра связывают только те вершины, которым соответствуют граничащие друг с другом их домены.

3. Строим функцию (аппроксимация Шепарда) на всем множестве точек.

4. Для каждой точки  строим функцию (аппроксимация Шепарда) на базе множества соседних точек . Будем считать, что функция  определена на множестве точек домена .

5. Определим функцию

6. Построим функцию оценки точности текущего приближения .

7. Обозначим через – множество узлов сетки с шагом  в области . В каждом узле вычислим значение оценки . Выберем  узлов в наибольшим значением оценки и удалённых друг от друга на расстояние не меньше , где – параметр, который определят равномерность распределения множества очередных точек.

1. **Описание разработки**

В рамках поставленной задачи реализована библиотека, средства которой позволяют получать **на вход даные о размерности функции, известных точках, требуемом значении точности, находить прогнозные точки и формировать отчёт о работе: координаты точек, в которых необходимо произвести дополнительные вычисления.**

**Для нахождения прогнозных точек реализован ряд алгоритмов.**

1. **Использованные методы решения**

За основу для разработанного метода решения был взят уже реализованный исходный алгоритм. В нём используется диаграмма Вороного: отображение, которое ставит каждой точке x в соответствие ближайшую к ней. Обратное отображение определяет область пространства из точек, для которых ближайшей является некоторая точка x. Такие множества называются доменами.

1. Алгоритм на вход получает набор точек

2. С помощью метода Шепарда создаётся аппроксимация

3. Аппроксимация анализируется

3.1 Строится сетка

3.2 Строятся домены. Центры доменов - это известные точки, все узлы сетки распределяются по доменам, какая известная точка ближе к узлу, в тот домен узел и будет определен

3.3 Определяются границы доменов

3.4 Определяются точки, пересчитав которые аппроксимация будет улучшена

В нашей работе мы старались улучшить пункт 3.4. Нами было разработано и реализовано два подхода – детерминированный алгоритм и случайный лес.

1. Детерминированный алгоритм

Суть этого подхода заключается в том, что на каждом шаге мы выбираем максимально далекие от известных точек точки, применяя для их выбора некоторые методики, позволяющие не локализироваться.

1. Сортируем точки на границе доменов по степени отдалённости от известных точек

2. Из сортированного списка выбираем первую точку

3. Проверяем следующие условия:

3.1 Среди выбранных точек нет точек из того же домена, что и рассматриваемая точка

3.2 Среди выбранных точек нет точек – соседей по сетке

3.3 Точка не принадлежит отключенному домену (только для одномерных функций)

Если точка удовлетворяет пунктам 3.1 и 3.2 выбираем ее

Для одномерных функций реализовано отключение доменов в том случае, если на предыдущем шаге мы выбрали точку на их границе, а ее значение на аппроксимации оказалось близко к реальному значению. На следующей итерации мы не смотрим на точки на границе отключенных доменов.

1. Случайный лес

Алгоритм случайного леса является алгоритмом машинного обучения, использующий набор деревьев решений.

Пусть обучающее множество имеет размер *N*, а число независимых переменных равно *M*. Введем дополнительно три параметра: коэффициент *r* (*0 ≤ r ≤ 1)*, число признаков *m ≤ M*, число деревьев *NTrees ≥ 1*.

На основе исходного обучающего множества сгенерируем случайную выборку размером *r·N* (без повторений). Элементы, не попавшие в выборку, используем в дальнейшем для оценки ошибки обобщения.

На основе сгенерированной выборки построим дерево решений. В ходе построения очередного узла дерева из *M* имеющихся переменных, на основе которых можно разделить дерево, выберем *m* случайных. Решение о разбиении принимается на основе лучшего возможного выбора из *m* случайно выбранных переменных. Дерево строится до исчерпания обучающено множества и не подвергается прунингу.

Процедура повторяется *NTrees* раз. Полученные деревья объединяются в комитет, принимающий решение путем голосования.

Для обучения использовалась та же функция, что и для анализа.

Обучение:

1. Запускаем детерминированный алгоритм, чтобы получить достаточное количество хороших точек. Под хорошими точками понимаются точки, разница между аппроксимацией и реальными значениями функции на которых меньше заданной точности.

2. Собираем фичи для каждой точки на сетке и обучаем случайный лес.

Анализ точек:

1. Анализируем все точки на сетке, собираем для них фичи и получаем у случайного леса заключение.

2. Сортируем точки по заключениям.

3. Из сортированного списка выбираем первую точку.

4. Проверяем условия:

3.1 Среди выбранных точек нет точек из того же домена, что и рассматриваемая точка.

3.2 Среди выбранных точек нет точек – соседей по сетке.

Если точка удовлетворяет пунктам 3.1 и 3.2 выбираем ее.

1. **Результаты**

Применение описанных алгоритмов позволило добиться хороших результатов. Это можно увидеть на примере функций, изображённых на рисунке 1. Результаты работы программы (с требуемой точностью 0.09) изображены на рисунках 2-4.

A close up of a map

Description automatically generated

Рисунок 1. Исследуемые функции

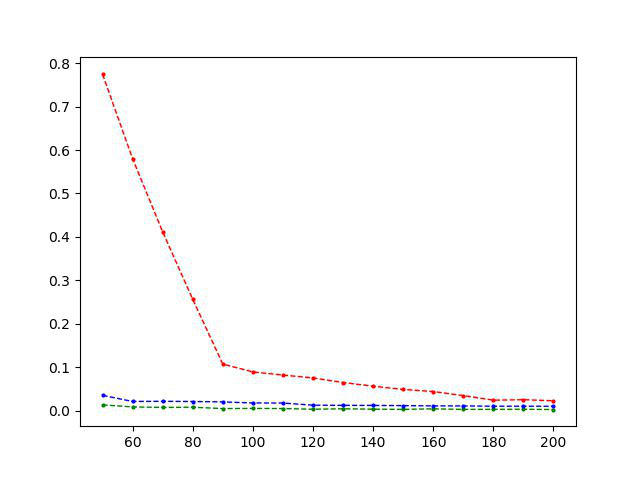


Рисунок 2. Результат работы алгоритмов. По вертикальной оси отложена величина ошибки, по горизонтальной оси – количество точек. Зелёная линия – равномерно взятые точки на сетке, синяя – детерминированный алгоритм, красная –случайный лес.

Ошибка – максимум среди запрошенных точек

|  |  |  |  |
| --- | --- | --- | --- |
|  | Default algorithm | Deterministic algorithm | Random forest |
| Accuracy | 0.158188719024688 | 0.093169468408714 | 0.0961158886858544 |
| Iteration amount | 19 | 16 | 18 |

Считаем до интегральной точности 0.09

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grid | Deterministic algorithm | Random forest |
| Accuracy |  | 0.0453467192241989 | 0.0825628882420977/ 0.0357571704479697 |
| Iteration amount | 100 | 26 | 18/28 |

Как видно из приведённых в таблицах выше результатов, детерминированный алгоритм получает решение с заданной точностью за меньшее число итераций.

A close up of a map

Description automatically generated

Рисунок 3. График функции x2y2

Второй эксперимент был приведён для функции с точностью 0.9.

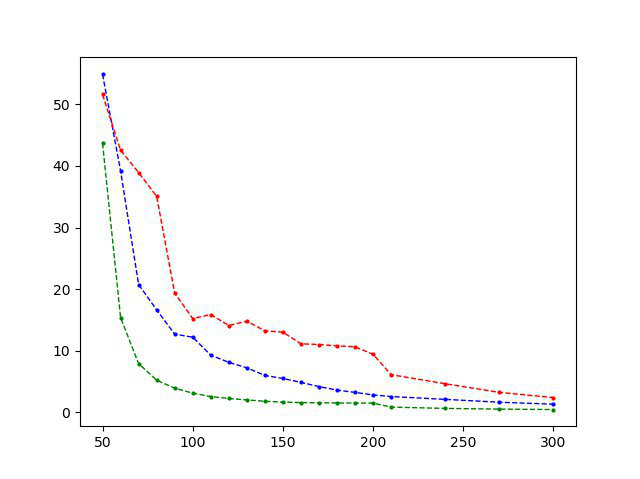


Рисунок 4. Результат работы алгоритмов. По вертикальной оси отложена величина ошибки, по горизонтальной оси – количество точек. Зелёная линия – равномерно взятые точки на сетке, синяя – детерминированный алгоритм, красная –случайный лес.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Default algorithm | Deterministic algorithm | Random forest |
| Accuracy | 323.714825149596 | 3.14725344325278 | 6.70319507416022 |
| Iteration amount | 109 | 196 |  |

Считаем до интегральной точности 0.009

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grid | Deterministic algorithm | Random forest |
| Accuracy |  | 0.882357228852225 | 0.893887074883104 |
| Iteration amount | 10000 | 376 | 571 |

|  |  |  |
| --- | --- | --- |
| Points amount | Deterministic algorithm | Random forest |
| 1976 | 1.02932873360944 | 0.0857362962721949 |
| 400 | 0.720659466712428 | 1.00597003159754 |

Для размера сетки 1000 узлов алгоритм случайного леса потребовал для пересчёта только 1656 точек, из чего можно сделать вывод, что благодаря алгоритму случайного леса можно отказаться от пересчёта всей сетки.