



Суперкомпьютерные дни в России 2023
международная научная конференция
25 - 26 сентября



УНИВЕРСИТЕТ ЛОБАЧЕВСКОГО
ИНСТИТУТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

Фреймворк методов интеллектуальной эвристической оптимизации iOpt

<https://github.com/aimclub/iOpt/>

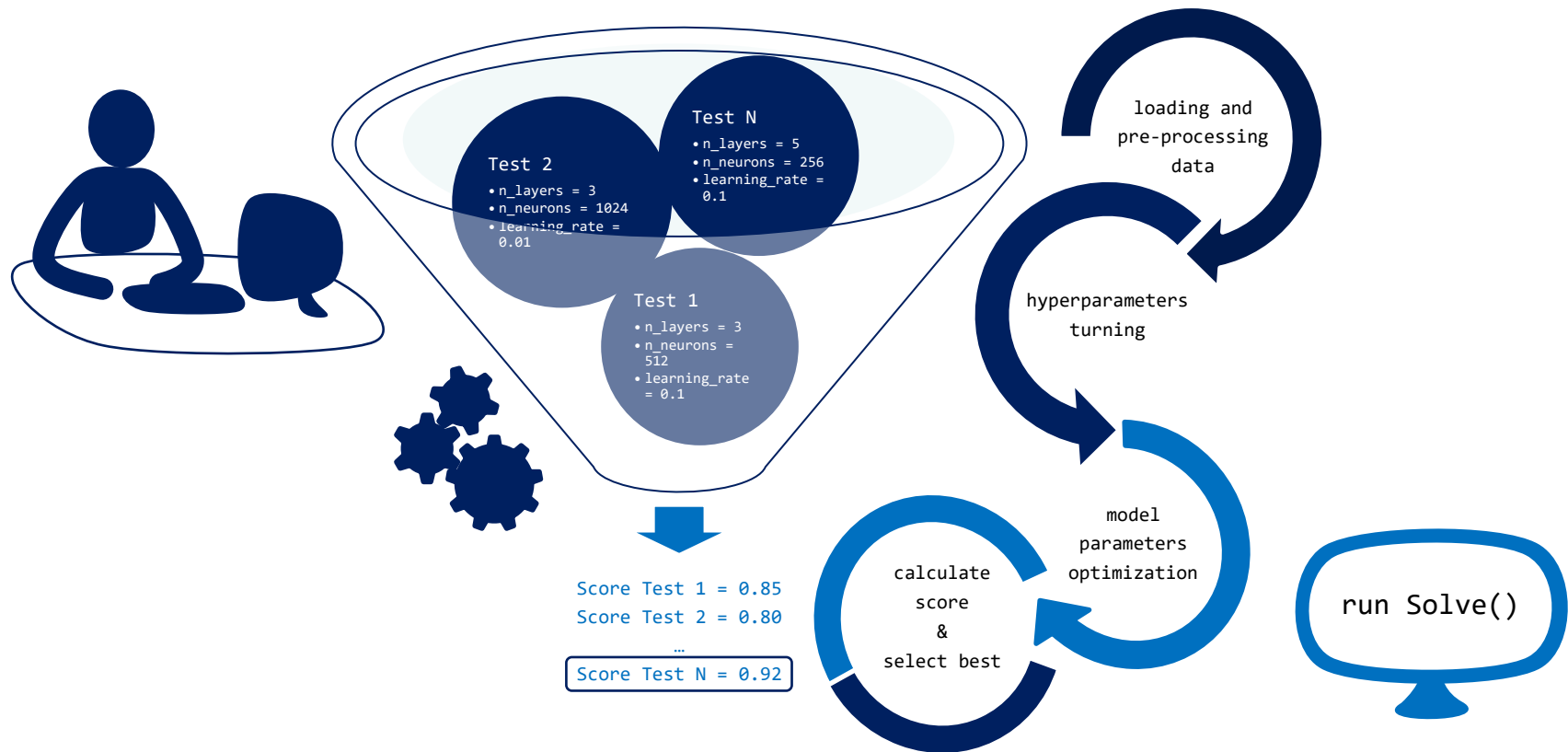
А.В. Сысоев, Е.А. Козинов, К.А. Баркалов, И.Г. Лебедев, Д.А. Карчков, Д.М. Родионов
ННГУ им. Н.И. Лобачевского
Университет ИТМО

Содержание

- Настройка гиперпараметров (мета-оптимизация)
- Задачи глобальной оптимизации
- Методы глобальной оптимизации
- Фреймворк iOpt – структура и возможности
- Сравнение iOpt с другими фреймворками

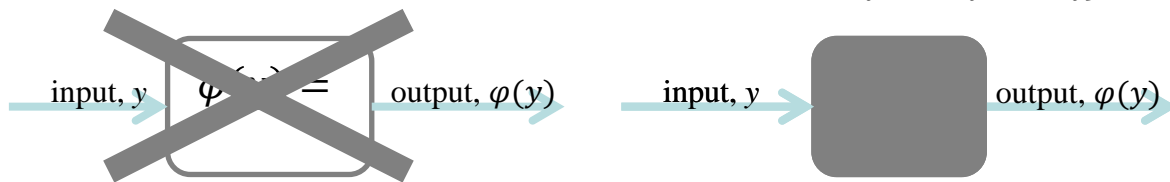
НАСТРОЙКА ГИПЕРПАРАМЕТРОВ (ГЛОБАЛЬНАЯ ОПТИМИЗАЦИЯ)

Настройка гиперпараметров



Задачи глобальной оптимизации

- Найти точку глобального минимума y^* функции $\varphi(y)$
 $\varphi(y^*) = \min\{\varphi(y): y \in D\}, D = \{y \in R^N, a_i \leq y_i \leq b_i\}$



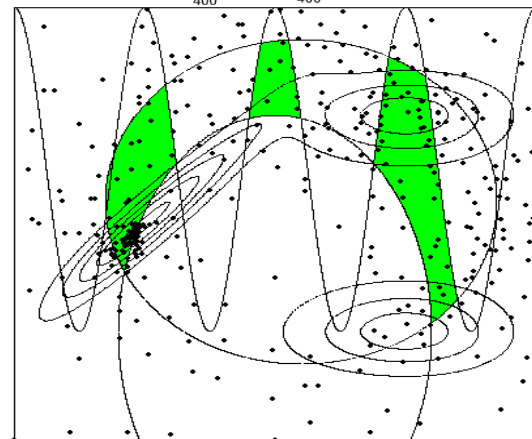
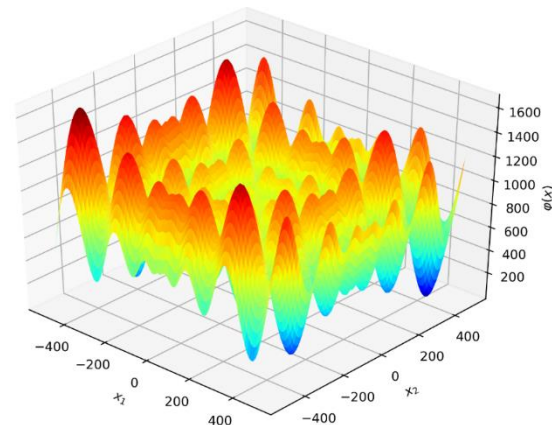
- Оптимизация при наличии ограничений
 $\varphi(y^*) = \min\{\varphi(y): y \in D, g_j(y) \leq 0, 1 \leq j \leq m\}$

учет функциональных ограничений

- Многокритериальная оптимизация
 $w(y) = (w_1(y), w_2(y), \dots, w_s(y)) \rightarrow \min$

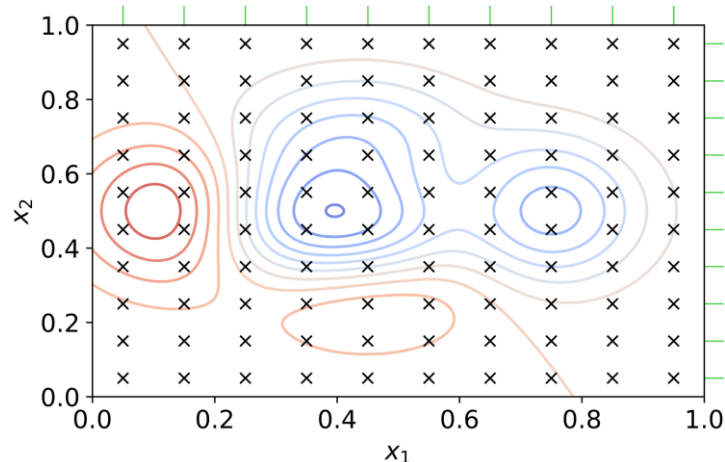
Скаляризация, множество Парето

$$F(x, \lambda) = \max_{1 \leq i \leq k} (\lambda_i w_i(x)) + \gamma \sum_{i=1}^k \lambda_i w_i(x)$$



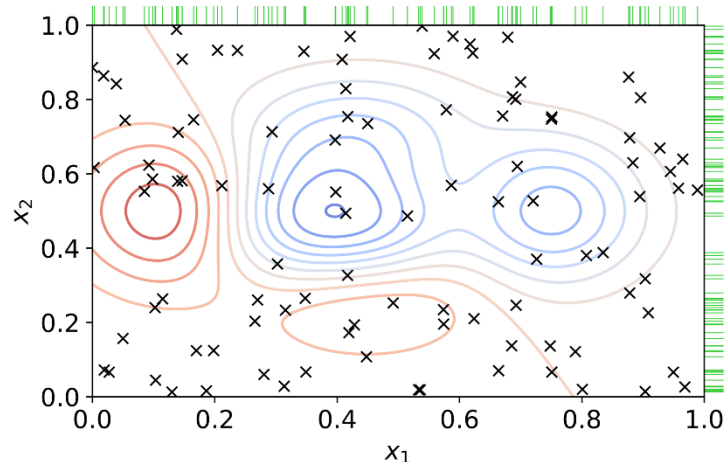
Методы глобальной оптимизации

- Минимизация функции потерь (на некотором датасете) в зависимости от набора гиперпараметров метода
- Используемые алгоритмы:
 - Поиск по равномерной сетке
 - Случайный поиск
 - Байесовская оптимизация
 - Генетические (эволюционные) алгоритмы
 - Липшицева оптимизация
- Доступные реализации (фреймворки)
 - Hyperopt
 - Optuna
 - Sherpa
 - Scikit-Optimize
 - и т.д.



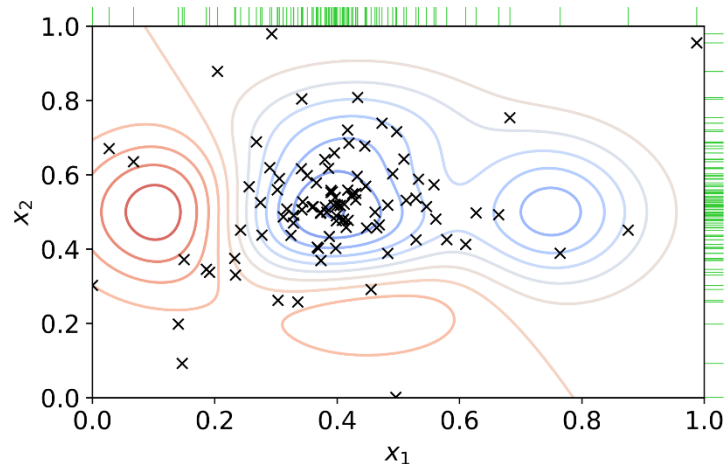
Методы глобальной оптимизации

- Минимизация функции потерь (на некотором датасете) в зависимости от набора гиперпараметров метода
- Используемые алгоритмы:
 - Поиск по равномерной сетке
 - **Случайный поиск**
 - Байесовская оптимизация
 - Генетические (эволюционные) алгоритмы
 - Липшицева оптимизация
- Доступные реализации (фреймворки)
 - Hyperopt
 - Optuna
 - Sherpa
 - Scikit-Optimize
 - и т.д.



Методы глобальной оптимизации

- Минимизация функции потерь (на некотором датасете) в зависимости от набора гиперпараметров метода
- Используемые алгоритмы:
 - Поиск по равномерной сетке
 - Случайный поиск
 - **Байесовская оптимизация**
 - **Генетические (эволюционные) алгоритмы**
 - Липшицева оптимизация
- Доступные реализации (фреймворки)
 - Hyperopt
 - Optuna
 - Sherpa
 - Scikit-Optimize
 - и т.д.



Методы глобальной оптимизации

□ Детерминированные алгоритмы:

- одинаковый результат при повторном запуске
- гарантированный результат в задачах малой ($N < 10$) размерности
- практически не применимы к задачам размерности $N \gg 10$
- ...

□ Метаэвристические алгоритмы:

- Неявно основаны на идеях случайного поиска
- Требуют нескольких запусков для нахождения наилучшего решения
- Хороший результат в задачах большой ($N \gg 10$) размерности
- Содержат параметры, которые сильно влияют на получаемое решение

Липшицева оптимизация

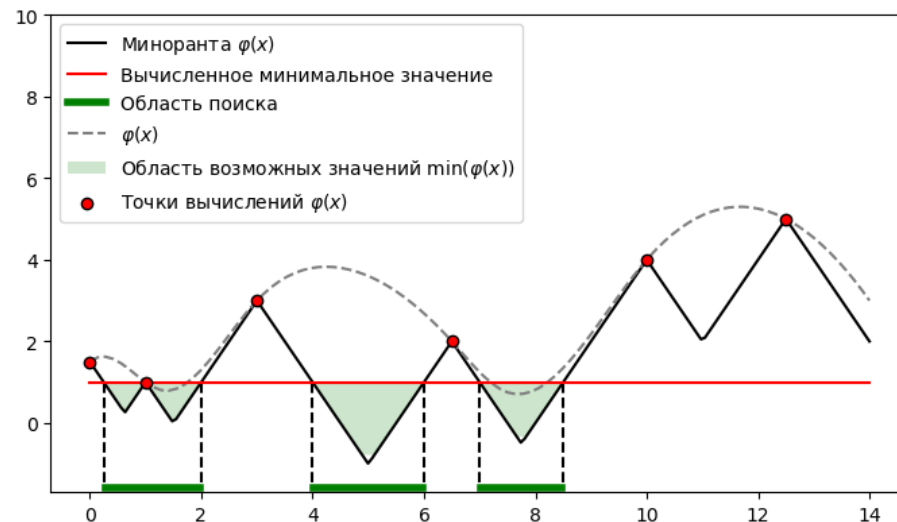
□ Предположение: ограниченное изменение аргумента Δy порождает ограниченное изменение значений функции $\Delta \varphi$

□ Математическая модель: условие Липшица

$$|\varphi(y') - \varphi(y'')| \leq L \|y' - y''\|, \quad y', y'' \in D$$

□ Липшицева оптимизация:

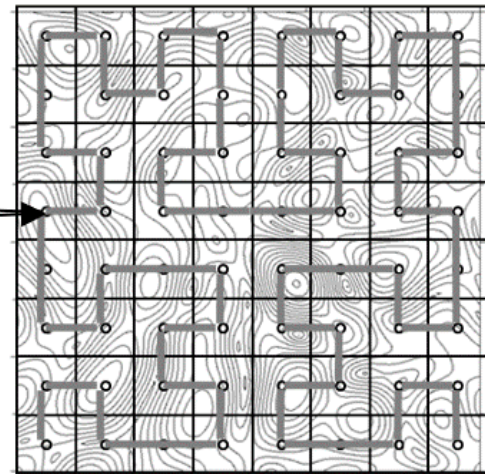
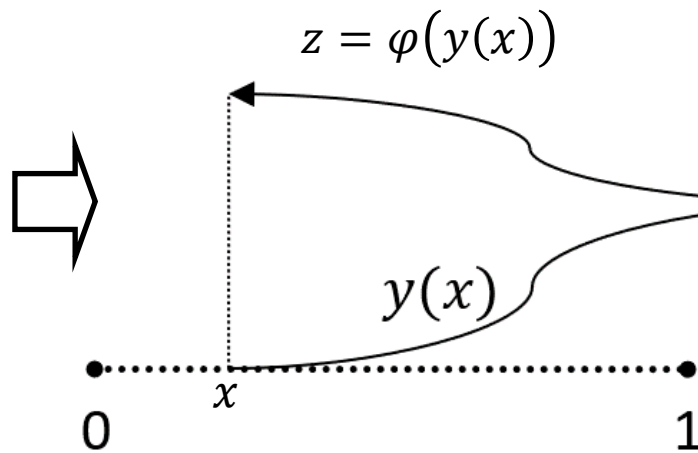
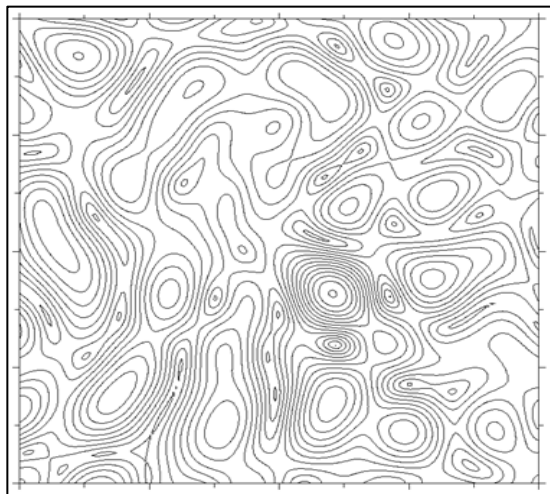
- С.А. Пиявский, Ю.Г. Евтушенко, М.А. Посыпкин, Я.Д.Сергеев, J. Pinter, P. Hansen, D. Jones, J. Žilinskas, ...
- Р.Г. Стронгин – Нижегородская школа глобальной оптимизации



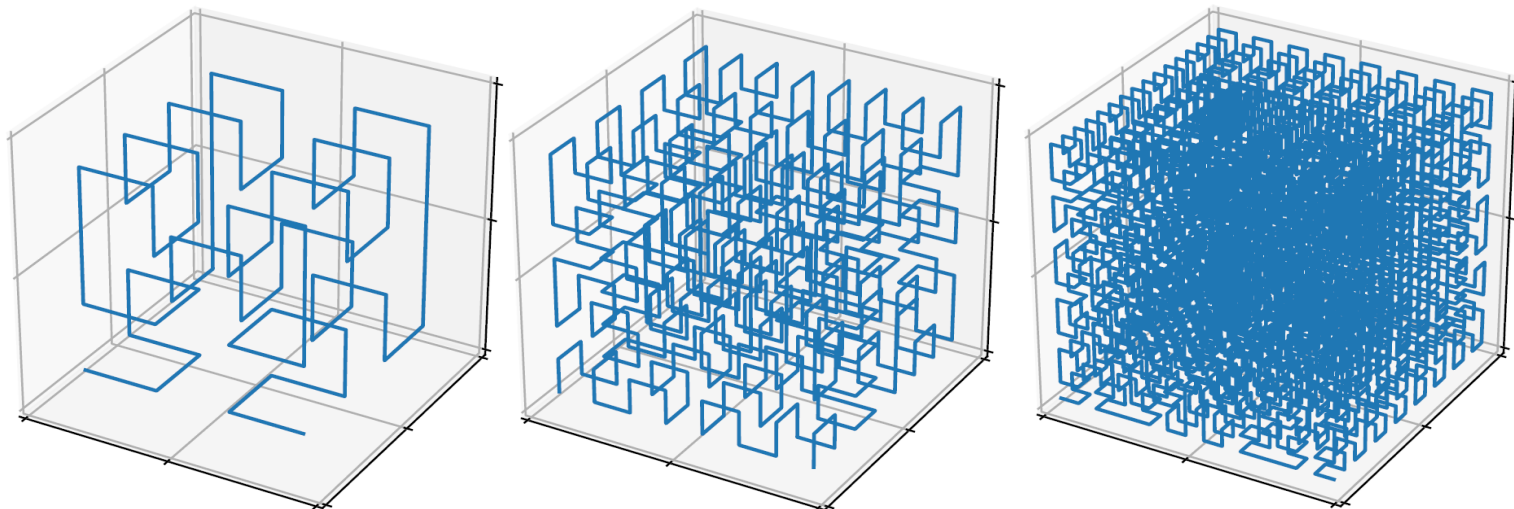
Редукция размерности на основе кривых Пеано

- Наши методы основаны на использовании кривых Пеано $y(x)$, которые однозначно и непрерывно отображают интервал $[0,1]$ на N -мерную область D

$$\min_{y \in D} \varphi(y) = \min_{x \in [0,1]} \varphi(y(x))$$



Редукция размерности на основе кривых Пеано



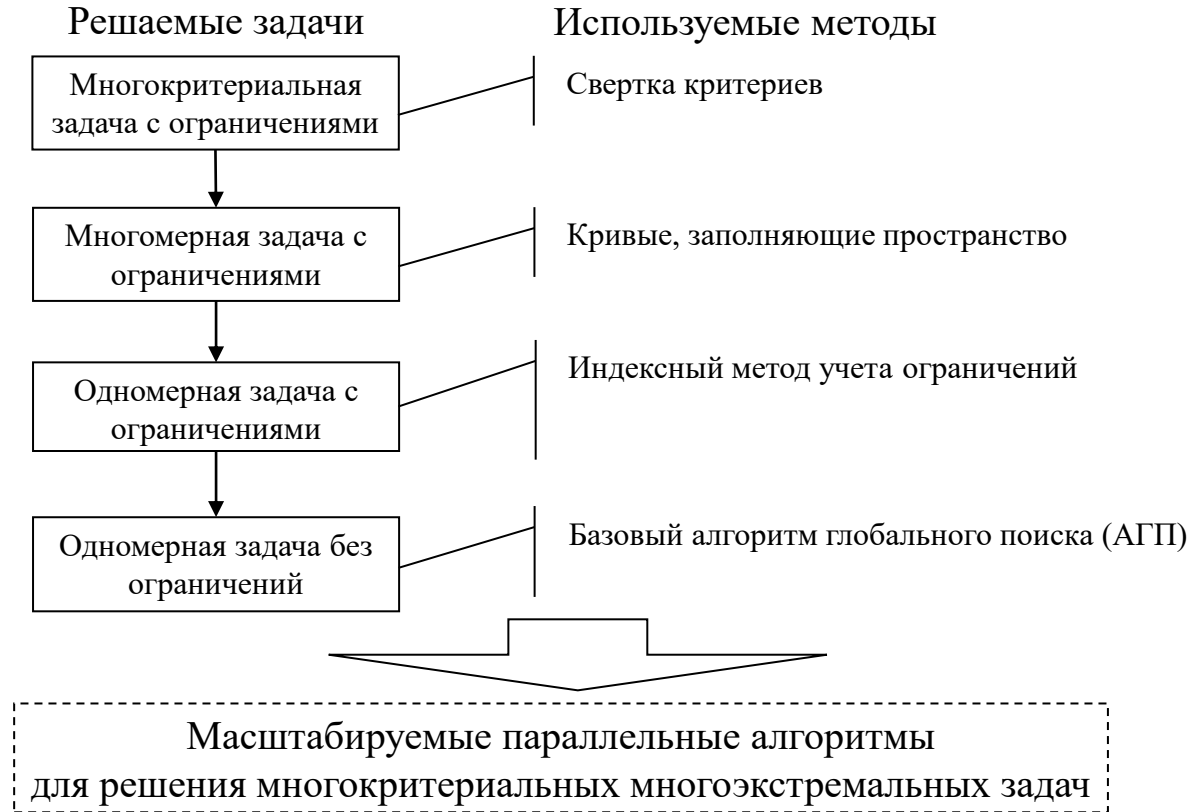
Реализованы быстрые алгоритмы для построения аппроксимаций кривых Пеано (*разверток*) с заданной точностью для заданной размерности.

Условию Липшица для $\varphi(y)$ будет соответствовать условие Гельдера для $\varphi(y(x))$

$$|\varphi(y(x_1)) - \varphi(y(x_2))| \leq 2L\sqrt{N+3}|x_1 - x_2|^{1/N}$$

где $x_1, x_2 \in [0,1]$

Схема решения задачи глобальной оптимизации в iOpt

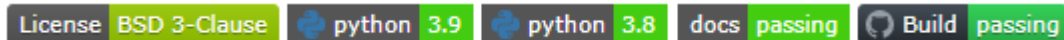


ФРЕЙМВОРК iOPT

Фреймворк iOpt

Классы решаемых задач

- Задачи липшицевой глобальной оптимизации (2022)
- Задачи смешанной дискретной оптимизации (2023)
- Задачи многокритериальной оптимизации (2024)



Назначение

- Выбор значений параметров математических моделей сложных объектов и процессов (методов ИИ и МО, методов дискретной оптимизации, и т.п.)
- Интеграция с внешними библиотеками или фреймворками ИИ и МО, а также предметными моделями
- Автоматизация предварительного анализа исследуемых моделей, например, выделение разных классов зависимостей модели от разных групп параметров
- Визуализация процесса выбора оптимальных параметров

Фреймворк iOpt

- Интерфейс определения задач оптимизации

```
class Problem(ABC):  
    """Базовый класс для задач оптимизации"""  
    def __init__(self):  
        """Описание количественных характеристик задачи поиска"""  
        self.name: str = ''  
        self.numberOfFloatVariables: int = 0  
        self.numberOfDiscreteVariables: int = 0  
        self.numberOfObjectives: int = 0  
        self.numberOfConstraints: int = 0  
        """Имена оптимизируемых параметров"""  
        self.floatVariableNames: np.ndarray(shape=(1), dtype=str) = []  
        self.discreteVariableNames: np.ndarray(shape=(1), dtype=str) = []  
        """Описание области поиска"""  
        self.lowerBoundOfFloatVariables: np.ndarray(shape=(1), dtype=np.double) = []  
        self.upperBoundOfFloatVariables: np.ndarray(shape=(1), dtype=np.double) = []  
        self.discreteVariableValues: np.ndarray(shape=(1, 1), dtype=str) = []  
        """Известное решение (используется для отладки методов)"""  
        self.knownOptimum: np.ndarray(shape=(1), dtype=trial) = []  
  
        """Абстрактный метод проведения испытаний в заданной точке"""  
        @abstractmethod  
        def Calculate(self, point: Point, functionValue: FunctionValue) -> FunctionValue:
```


Фреймворк iOpt

□ Пример описания задачи предсказания рака молочной железы

```
class SVC_2D(Problem):
    def __init__(self, x_dataset: np.ndarray, y_dataset: np.ndarray,
                 regularization_bound: Dict[str, float],
                 kernel_coefficient_bound: Dict[str, float]):
        """
        x_dataset, y_dataset :      входные данные обучающей выборки
        kernel_coefficient_bound: границы изменения значения параметра регуляризации
        regularization_bound:      границы изменения значения коэффициента ядра
                                   (low - нижняя граница, up - верхняя)
        """
        super(SVC_2D, self).__init__()
        self.numberOfFloatVariables = 2
        self.numberOfObjectives = 1
        self.x = x_dataset
        self.y = y_dataset

        self.floatVariableNames = np.array(["Regularization parameter", "Kernel coefficient"], dtype=str)
        self.lowerBoundOfFloatVariables = np.array([regularization_bound['low'],
                                                    kernel_coefficient_bound['low']],
                                                    dtype=np.double)

        self.upperBoundOfFloatVariables = np.array([regularization_bound['up'],
                                                    kernel_coefficient_bound['up']],
                                                    dtype=np.double)

        ...
```

Фреймворк iOpt

□ Пример описания задачи предсказания рака молочной железы

```
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
...
class SVC_2D(Problem):
...
    def Calculate(self, point: Point, functionValue: FunctionValue) -> FunctionValue:
        """
        Метод расчёта значения целевой функции в точке
        point: Точка испытания
        functionValue: объект хранения значения целевой функции в точке
        """
        cs, gammas = point.floatVariables[0], point.floatVariables[1]
        clf = SVC(C=10 ** cs, gamma=10 ** gammas)
        clf.fit(self.x, self.y)

        functionValue.value = -cross_val_score(clf, self.x, self.y, scoring='f1').mean()
        return functionValue
```

Фреймворк iOpt

□ Пример описания задачи предсказания рака молочной железы

Создание объекта задачи

```
x, y = load_breast_cancer_data()
regularization_value_bound = {'low': 1, 'up': 6}
kernel_coefficient_bound = {'low': -7, 'up': -3}
```

```
problem = SVC_2D(x, y, regularization_value_bound, kernel_coefficient_bound)
```

Формируем параметры решателя

```
method_params = SolverParameters(r=3.0, itersLimit=100, numberOfParallelPoints=4)
```

Создаем решатель

```
solver = Solver(problem, parameters=method_params)
```

Добавляем вывод результатов в консоль

```
cfol = ConsoleOutputListener(mode='result')
solver.AddListener(cfol)
```

Добавляем построение 3D визуализации после решения задачи

```
spl = StaticPainterNDListener("svc2d_stat.png", "output", varsIndxs=[0, 1],
                             mode="surface", calc="interpolation")
```

```
solver.AddListener(spl)
```

Фреймворк iOpt

□ Пример поиска оптимального решения

Решение задачи

```
sol = solver.Solve()
```

Пользовательский вывод информации

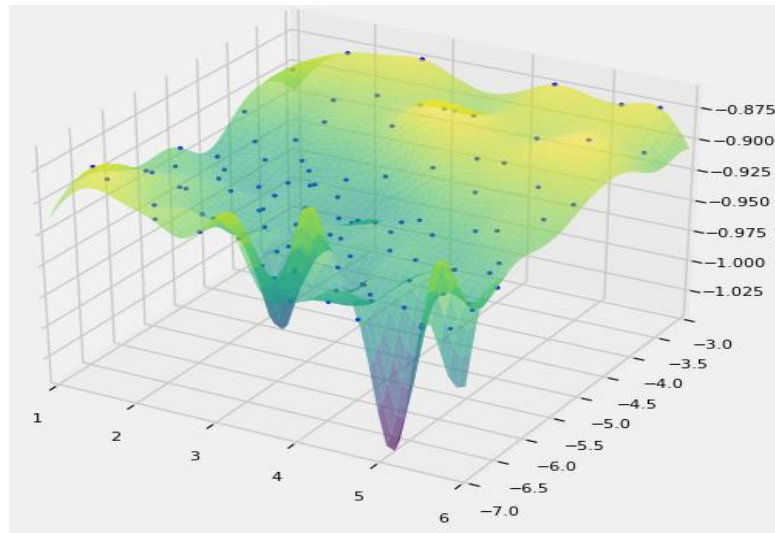
```
print(sol.numberOfGlobalTrials)
```

```
print(sol.numberOfLocalTrials)
```

```
print(sol.solvingTime)
```

```
print(sol.bestTrials[0].
```

```
functionValues[0].value)
```



```
Result
-----
global iteration count: 101
local iteration count: 0
solving time: 1.513753
solution point: [ 3.42919922 -4.99804688]
solution value: -0.97354927
accuracy: 0.05590170
-----
```

СРАВНЕНИЕ С ИЗВЕСТНЫМИ ФРЕЙМВОРКАМИ ПРИ РЕШЕНИИ МОДЕЛЬНЫХ ЗАДАЧ

Тестовая инфраструктура

- Вычислительные эксперименты проводились на суперкомпьютере «Лобачевский» Нижегородского государственного университета.
 - В экспериментах использовался вычислительный узел с двумя 64-ядерными процессорами AMD EPYC 7742 (всего 128 вычислительных ядра).
 - На узле установлено 512 Gb оперативной памяти.
 - Операционная система CentOS 7.

Решаемые задачи оптимизации

- Эффективность фреймворка iOpt проверялась на решении следующих задач:
 - Настройка параметров алгоритма машинного обучения для построения модели предсказания рака молочной железы.
 - Настройка параметров генетического алгоритма при решении задачи коммивояжера.
 - Настройка параметров XGBoost при построении модели на разных наборах данных.

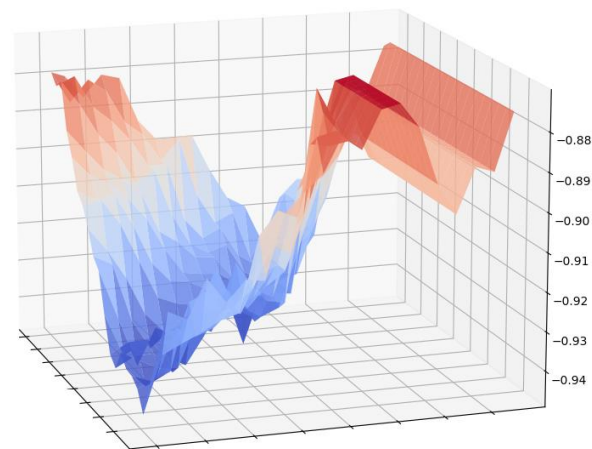
Построения модели предсказания рака молочной железы

Стандартный датасет breast_cancer

Целевая метрика – f1_score, настройка параметров C и gamma (100 испытаний)

График целевой функции

Фреймворк	f1_score
----	0.87
Scikit-Optimize	0.90
iOpt	0.94



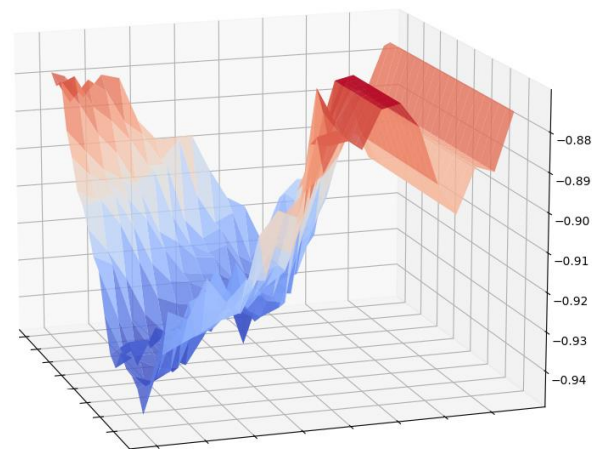
Построения модели предсказания рака молочной железы

Стандартный датасет breast_cancer

Целевая метрика – f1_score, настройка параметров C и gamma (1000 испытаний)

Число процессов	f1_score	Время настройки параметров	Ускорение
1	0,975	31,199	1
5	0,975	7,876	4,0
10	0,975	4,775	6,5
20	0,975	3,251	9,6
40	0,975	2,002	15,6
80	0,975	1,616	19,3

График целевой функции



Решение задачи коммивояжёра

- Решалась задача a280 из базы «TSPLIB is a library of sample instances for the TSP»
 - <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/>
- В рамках исследования применялся генетический алгоритм, реализованный в библиотеке scikit-opt
 - <https://github.com/guofei9987/scikit-opt>
- Настраиваемые параметры
 - Вероятность мутации
 - Размер популяции
- Число итераций генетического алгоритма было фиксировано и равно 200



* Картинка взята с сайта
[https://ru.wikipedia.org/wiki/ Задача_коммивояжёра](https://ru.wikipedia.org/wiki/Задача_коммивояжёра)

Решения задачи коммивояжёра

- Длина пути с настройками генетического алгоритма по умолчанию 25835,04
- Эффективность параллельной реализации метода число испытаний при настройке параметров 400

Число процессов	Длина пути (лучше меньше)	Время настройки параметров	Ускорение
1	16040,52	1162,5	1
5	16223,25	316,0	3,7
10	16011,50	164,3	7,1
20	16089,76	90,25	12,9
40	16216,21	54,69	21,3
80	15910,54	34,86	33,3



* Картинка взята с сайта [https://ru.wikipedia.org/wiki/ Задача_коммивояжёра](https://ru.wikipedia.org/wiki/Задача_коммивояжёра)

Настройка XGBoost

□ XGBoost – одна из реализаций алгоритма градиентного бустинга на деревьях решений.

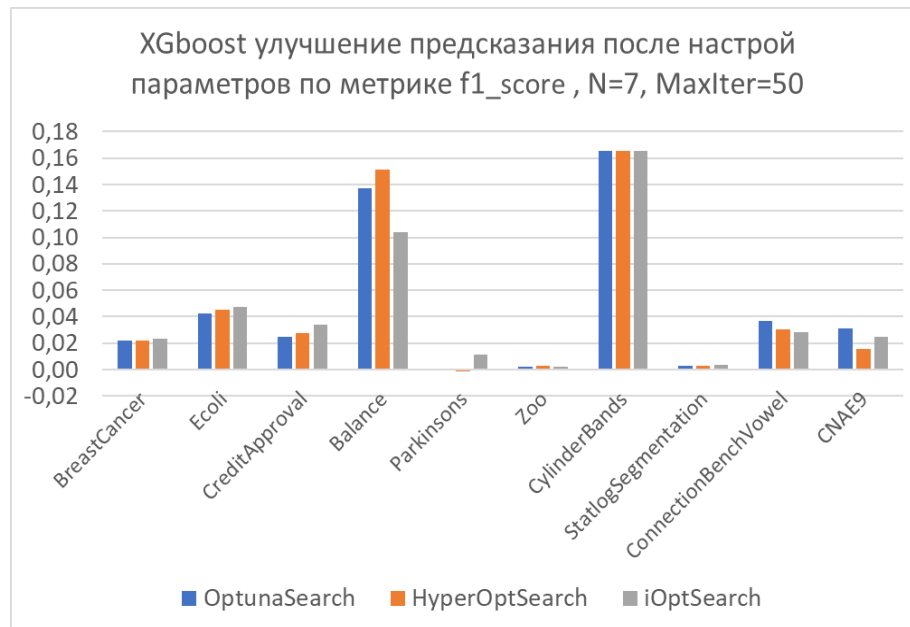
□ Метод построения модели классификации

– На первом шаге модель строилась с параметрами по умолчанию

– Далее параметры XGBoost оптимизировались и модель строилась вновь

– Оптимизируемые параметры

```
params = {  
    'n_estimators': (int, 10, 200),  
    'max_depth': (int, 5, 20),  
    'min_child_weight': (int, 1, 10),  
    'gamma': (float, 0.01, 0.6),  
    'subsample': (float, 0.05, 0.95),  
    'colsample_bytree': (float, 0.05, 0.95),  
    'learning_rate': (float, 0.001, 0.1)  
}
```



ВОПРОСЫ?

konstantin.barkalov@itmm.unn.ru
evgeny.kozinov@itmm.unn.ru