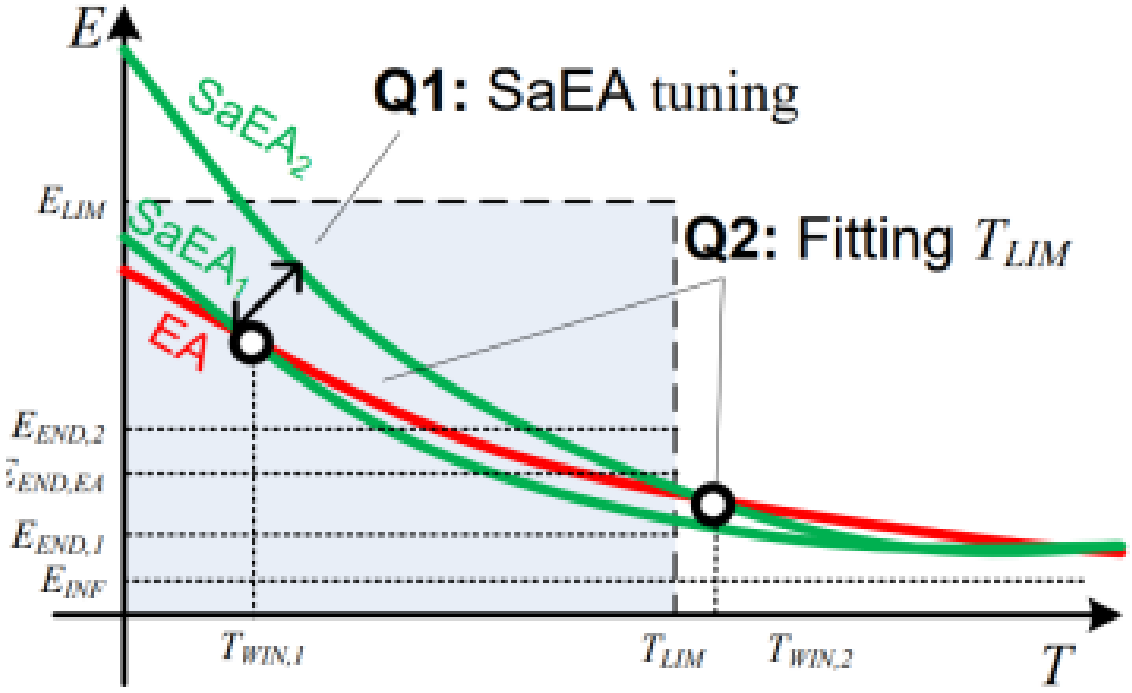
Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49

ИТМО

А.А. Хватов, Н.О. Никитин, А.В. Калюжная

СОВРЕМЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ С ПРИМЕРАМИ НА PYTHON



Санкт-Петербург
2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРЦИИ

УНИВЕРСИТЕТ ИТМО

А.А. Хватов, Н.О. Никитин, А.В. Калюжная

СОВРЕМЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ С ПРИМЕРАМИ НА PYTHON

Учебно-методическое пособие

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 01.04.02 Прикладная математика и информатика в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2023

А.А. Хватов, Н.О. Никитин, А.В. Калюжная. Современные методы оптимизации с примерами на python. – СПб: Университет ИТМО, 2023. – 48 с.

Рецензент: Насонов Денис Александрович, к.т.н., доцент

Учебно-методическое пособие к курсу "Современные методы оптимизации" включает в себя краткое изложение теоретического материала по следующим темам: базовые определения методов оптимизации, линейные методы оптимизации, градиентные методы оптимизации, метаэвристические методы оптимизации, эволюционные методы оптимизации, роевые методы оптимизации, многокритериальные задачи оптимизации, оптимизация с вычислительно емкими целевыми функциями. Помимо этого, методическое пособие содержит задания к лабораторным работам и примеры их выполнения на языке Python.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Методы оптимизации, как и любой математический аппарат, применяемый на практике, прошли долгий путь от математической абстракции до основы многих современных методов машинного обучения и калибровки математических моделей в целом.

В настоящее время выделяется несколько устойчивых направлений: нелинейная оптимизация, вариационные методы, методы калибровки (обучения) математических моделей и другие. При этом одновременно существуют теоретические методы, такие как нелинейная оптимизация и вариационные методы, так и более практические, как калибровка моделей.

Калибровка моделей отличается от остальных методов тем, что целевая функция (то есть, то, что мы оптимизируем) либо содержит много параметров, либо трудно вычисляется (например, суперкомпьютерные модели океана или атмосферы) и поэтому требует особого подхода к оптимизации.

В данном учебном пособии мы постарались агрегировать свой опыт, приобретённый в результате многочисленных исследований и выполненных проектов так или иначе связанных с калибровкой математических моделей. В рамках сравнительно небольшого текста довольно трудно построить непротиворечивую теорию. Мы постарались выбрать те аспекты, которые были бы интересны с точки зрения моделирования.

Пособие построено следующим образом: Глава 2 содержит небольшое введение в то, что мы называем современными методами оптимизации, Глава 3 содержит некоторые факты из анализа и численных методов, а также математические формулировки классических методов оптимизации. Главы 4 и 5 посвящены формулировкам более современных градиентных и классификации неградиентных методов соответственно. Глава 6 содержит конкретные примеры реализаций задач калибровки математических моделей.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ ОПТИМИЗАЦИИ. ОСНОВНЫЕ СВЕДЕНИЯ

Задача оптимизации *без ограничений* может быть формализована в виде:

$$\min_{X \in D \subset R^N} F(X) = F(X^*) = F^*,\ F \in R^M,$$

где X – варьируемые параметры, $|X| = N$ – размерность вектора параметров, F – целевая функция (вектор-функция), $|F| = M$ – размерность целевого пространства.

При необходимости задания ограничений они могут быть записаны в виде:

$$D = \{X | H(X) = 0,\ G(X) \geq 0\}$$

где D – множество допустимых значений параметров X, $H(X), G(X)$ – ограничивающие функции (вектор-функции). Задачи оптимизации с ограничениями называются задачами *условной оптимизации*.

При этом, в случаях, когда целевая функция $F(X)$ и функции ограничений $H(X)$ и $G(X)$ являются детерминированными (т.е. не содержат случайных параметров), задача оптимизации также называется *детерминированной*. И наоборот, задача оптимизации называется *стохастической*, если целевой функции и / или функциям ограничений можно поставить в соответствие функцию распределения $\mathcal{F}$ их параметров или ее моменты (в первую очередь, математическое ожидание $E[X]$). В частности, для целевой функции:

$$X \sim \mathcal{F},\ F(X) \rightarrow E[X]$$

Задачи оптимизации можно классифицировать многими способами. Помимо деления задач оптимизации на *безусловные и условные*, *детерминированные и стохастические*, можно выделить несколько способов классификации. В первую очередь, можно рассмотреть особенности целевых функций оптимизации.

Целевые функции могут быть *однопараметрическими и многопараметрическими* ($|X| > 1$).

Целевая функция, непрерывная и ограниченная на интервале D:

$$F(x^*), x \in [p, q], [p, q] = D,$$

называется *унимодальной* (одноэкстремальной), если $F(x)$ – возрастает на $(x^*, q]$, и $F(x)$ – убывает на $[p, x^*)$.

Функция называется *выпуклой*, если выполняется неравенство:

$$F(ax_1, (1-a)x_2) \leq aF(x_1) + (1-a)F(x_2), x_1 \neq x_2,\ a \in [0,1],$$

где $F(x)$ – однопараметрическая функция. Определение *строго выпуклой* функции дается аналогично, с точностью до знака строгого неравенства ($<$). В случае многопараметрической функции $F(X)$, также может быть дано определение выпуклой и строго выпуклой функций аналогичным образом.

В значительной степени сложность задач оптимизации определяется свойствами многопараметрических целевых функций. Кроме перечисленных основных свойств целевых функций, можно также выделить особые свойства овражности, сепарабельности, алгебраичности и др.

Значительное влияние на свойства задачи оптимизации и, как следствие, алгоритмов ее решения оказывают особенности множества допустимых значений D и самой целевой функции. Так, если целевая функция $F(X)$ является выпуклой, то задача оптимизации называется задачей *выпуклого программирования*. Если целевая функция $F(X)$ является линейной, а множество D формирует выпуклый многогранник, то задача оптимизации называется *линейной*. Если множество D конечно, то это задача *дискретного программирования*.

Если целевая функция $F(X)$ нелинейна, т.е. имеет общий вид то задача оптимизации называется задачей *нелинейного программирования*.

В случаях, когда требуется найти минимум сразу нескольких целевых функций $|F| > 1$, задача оптимизации называется *многокритериальной*.

Еще одним важным фактором при решении задачи оптимизации является характер искомого решения для многоэкстремальных функций. Если достаточно найти любой локальный экстремум, то задача оптимизации является *локальной*. Если необходимо найти глобальный оптимум, задача является *задачей глобальной оптимизации*.

Свойства задач оптимизации определяют основные свойства методов, которые могут быть применены для их решения. Изучению современных методов и алгоритмов оптимизации, применимых для различных задач и будет посвящен данный курс.

Вопросы для самоконтроля:

1. Что подразумевается под задачей численной оптимизации, и какие основные компоненты входят в ее формулировку?
2. Объясните разницу между условной и безусловной оптимизацией, и приведите примеры ограничений, которые могут быть использованы в задаче условной оптимизации.
3. Объясните разницу между задачами выпуклой и невыпуклой оптимизации.

ГЛАВА 2. ОБЗОР СОВРЕМЕННЫХ МЕТОДОВ ОПТИМИЗАЦИИ

Общую постановку многокритериальной задачи оптимизации без ограничений можно записать как:

$$\boldsymbol{\theta_{opt}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \Phi(\boldsymbol{\theta})$$

$$\Phi(\boldsymbol{\theta}) = \mathcal{G}(f_1(\boldsymbol{\theta}), \dots, f_n(\bullet)),$$

где $\mathcal{G}(\bullet)$ - оператор многокритериальной трансформации, $\boldsymbol{Y}$, f_i – критерий оптимизации, где $i = \overline{1,n}$, n – число критериев, $\boldsymbol{\theta}_{opt}$ - Парето-оптимальный набор оптимизируемых переменных. Отметим, что в дальнейшем мы будем пользоваться однокритериальной формулировкой ($n = 1$). В современных методах оптимизации многокритериальные задачи так же используются довольно часто, но вместе с тем, требуют знаний аппарата теории игр. Для более детального знакомства с многокритериальными методами можно, например, использовать

Для решения практических задач общая постановка, как это обычно бывает, требует значительной конкретизации, и каждая практическая задача требует, зачастую, своего подхода. Основные этапы решения практических задач показаны в таблице 2.1.

Таблица 2.1 – Основные этапы решения оптимизационной задачи

Этап	**Подходы**
Постановка задачи	Выбор настраиваемых переменных
	Анализ ограничений
	Выбор целевой функции для оптимизации
Предобработка данных	Нормализация значений, корректировка выбросов и заполнение пропусков
Оценка неопределенности и выбор подхода к её устранению	Выявление видов неопределенности и их источников
Выбор метода оптимизации	Анализ литературы, связанной с решением схожих проблем
	Проверка на тестовых функциях

Выполнение оптимизации	Параллелизация процесса оптимизации Выбор критерия останова
Анализ результатов	Анализ сходимости, робастности, качества решений
Актуализация результата	Оценка динамической изменчивости поверхности целевой функции
	Критерий повторного запуска оптимизации

Так, например, необходимо осуществить выбора метода оптимизации. К базовым методам оптимизации относится поиск по сетке значений (grid search). Он основан на полном переборе заданного подмножества пространства параметров с фиксированным шагом. Благодаря простоте метод часто используется для решения оптимизационных задач небольшой размерности. Например, поиск по сетке часто применяется для настройки гиперпараметров методов машинного обучения. Преимуществом этого метода является хорошая распараллеливаемость, так как генерация следующего набора параметров не зависит от расчета значений целевой функции на предыдущих. Однако даже при не очень большом количестве подлежащих настройке параметров время оптимизации может возрасти до неприемлемого [1].

Из переборных методов многомерной оптимизации также можно выделить методы на основе кривых Пеано [2]. При их использовании решение многомерных задач сводится к решению одномерных задач с помощью непрерывных однозначных отображений на основе типа кривой Пеано (кривая, заполняющая пространство).

«Классическим» подходом к решению задач оптимизации является использование методов из семейства градиентных (методы первого порядка). Они основаны на вычислении функции градиента конкретной точки в пространстве параметров для заданного размера шага. Градиентные методы (градиентный спуск, квазиньютоновские методы) применяются для решения множества задач. Так, они используются в решении задач метаобучения и идентификации гиперпараметров [3].

Схожи с градиентными и методы второго порядка (например, метод Ньютона), в которых производится вычисление не только первых, но и вторых производных (гессиана функции). Вычислительная сложность таких методов выше, чем у методов первого порядка. Методы второго порядка более чувствительны к неопределенности, чем первого (особенно к систематическим ошибкам). Также они более требовательны к детализации численного вычисления градиента [4].

При решении задач оптимизации на недифференцируемых функциях применяются различные методы сведения таких функций к дифференцируемому виду: аппроксимация дифференцируемой функцией, построение субградиента [4].

Однако эта группа методов обладает рядом недостатков. Прежде всего, градиентные методы позволяют хорошо решать задачу локального поиска (уточнения найденного решения), но не обеспечивают нахождения глобального минимума при наличии у минимизируемой функции локальных минимумов. При наличии в задаче неопределенности неизбежно порождение большого числа локальных минимумов, и стагнация процесса оптимизации в одном из них весьма вероятна.

Другой, ещё более критичной, проблемой в применении градиентных методов является необходимость аналитического или численного вычисления градиента. Целевая функция может быть недифференцируема в принципе из-за отсутствия непрерывности. Кроме того, как уже упоминалось ранее, часть оптимизируемых переменных может носить дискретный характер.

Эти особенности затрудняют практическое применение градиентных методов. В работе [5], посвященной сравнению градиентного метода и генетического алгоритма на примере калибровки модели морского льда, показано, что стохастический метод является более практичным выбором для такой задачи именно из-за обилия локальных минимумов и большой неопределенности, что приводит к большому СКО параметров, найденных при многократном запуске градиентного алгоритма.

Применение прямых методов оптимизации (также называемых методами нулевого порядка) требует вычисления лишь значений целевой функции, а необходимость находить значения частных производных отсутствует. К классу прямых недетерминированных методов относятся несколько достаточно распространённых [4]: на основе линейной аппроксимации (например, метод покоординатного спуска, иногда классифицируемый как градиентный), нелокальной аппроксимации (метод Нелдера–Мида, также известный как симплекс-метод), квадратичной аппроксимации.

В целом идеи этих методов достаточно схожи – в пространстве параметров так или иначе выполняются линейные или нелинейные геометрические построения, позволяющие выделить регион пространства поиска, содержащий потенциальный минимум, а затем выполнить уточняющий поиск.

Однако, данные методы склонны к «скатыванию» в локальные минимумы. При наличии неопределенности в решаемой задаче обычно требуется значительное снижение размера шага поиска [4]. Также среди недостатков можно отметить неработоспособность таких методов для дискретных переменных и сильную зависимость от начального приближения, вынуждающую производить процесс оптимизации многократно для разных начальных условий. Поэтому их применение ограничено достаточно простыми моделями (например, в работе [6] метод Нелдера–Мида применен для калибровки модели речных наводнений).

Для более сложных задач целесообразно применять методы, также формально относящиеся к классу прямых (т. к. они не используют вычисления производных), однако применяющие различные эвристики для ускорения поиска решения и избегания локальных минимумов – стохастические методы. К ним, например, принадлежит метод случайного поиска [7], позволяющий перейти от

полного перебора по сетке значений к случайном выбору значений-кандидатов в соответствии с заданным распределением. Эффективность случайного поиска возрастает, если только часть параметров в выбранном пространстве поиска значима и от перебора значений части из них в действительности можно отказаться. В основу случайного поиска могут быть положены различные алгоритмы генерации квазислучайных последовательностей – например, последовательность Соболя. Также существуют различные модификации случайного поиска, увеличивающие его эффективность – в качестве примера можно привести случайный поиск с возвратом при неудачном шаге.

В задаче сложных задачах оптимизации случайный поиск напрямую применяется редко, поскольку, несмотря на большую, чем у полного перебора, эффективность, требуемое для нахождения минимума целевой функции число её вычислений пропорционально размеру пространства поиска, в то время как для вероятностных методов зависимость носит логарифмический характер [8].

Другой стохастический метод, который находит широкое применение в решении задач оптимизации – метод имитации отжига, основанный на аналогии с физическим процессом постепенного охлаждения металла. Одна из причин сложности применения метода – необходимость подбора функции температуры под конкретную задачу. Однако метод имитации отжига лучше подходит для решения задач локальной оптимизации, чем глобальной, из-за его свойства скорее развивать удачные решения, а не находить новые (хотя баланс между этими задачами также задается функцией температуры). Также задачи многокритериальной оптимизации плохо решаются с помощью имитации отжига из-за необходимости перехода к однокритериальной целевой функции.

Байесовская оптимизация основана на построении вероятностной модели для функции, связывающей параметры модели и значения целевой функции. Решение этой задачи сводится к нахождению распределения $P(y|x)$, где $x -$ калибруемые параметры, y – значение целевой функции.

Применением байесовских методов достигается возможность построения вспомогательной байесовской суррогатной модели, воспроизводящей целевую функцию с известной неопределенностью (широко используемым вариантом такой модели служит кригинг – регрессия на гауссовых процессах [9]). Зная эту неопределенность, можно выделять районы пространства параметров, добавление новых точек в которые позволит повысить точность такого суррогатного приближения. Кроме того, байесовский подход позволяет оценивать не точные значения параметров (не всегда несущие практический смысл в условиях неопределённости), а описывающие их распределения [10].

Зачастую байесовскую оптимизацию применяют совместно с методом Монте-Карло для марковских цепей (Markov Chain Monte Carlo). MCMC позволяет оценить функцию плотности распределения для каждого из параметров без априорного предположения о форме этого распределения (однако возможно задать начальное приближение исходя из предметных знаний).

Преимуществом байесовской оптимизации является хорошая эффективность при наличии шумовой составляющей в целевой функции (из-за сглаживающего эффекта суррогатной функции), а также отсутствие необходимости вычислений первой и второй производных. Числом запусков реальной модели в ходе оптимизации можно управлять с помощью адаптивных методов добавления точек и методов анализа чувствительности параметров.

Однако байесовский подход имеет несколько недостатков. Во-первых, требует построения модели ошибки, специфичной для каждой решаемой задачи. Во-вторых, не вполне подходит для задач многокритериальной оптимизации, в которых необходимо сохранение сразу нескольких решений, не доминирующих друг над другом. В-третьих, байесовский подход может быть достаточно чувствителен к начальным значениям для оптимизации (или предположениям о законе распределения значений параметров).

Для оценки эффективности различных методов оптимизации часто используются стандартные тестовые функции. Тестовые функции позволяют оценить сходимость, точность, робастность, производительность алгоритмов, т.к. в данном случае глобальный оптимум заранее известен (как правило, расположен в начале координат).

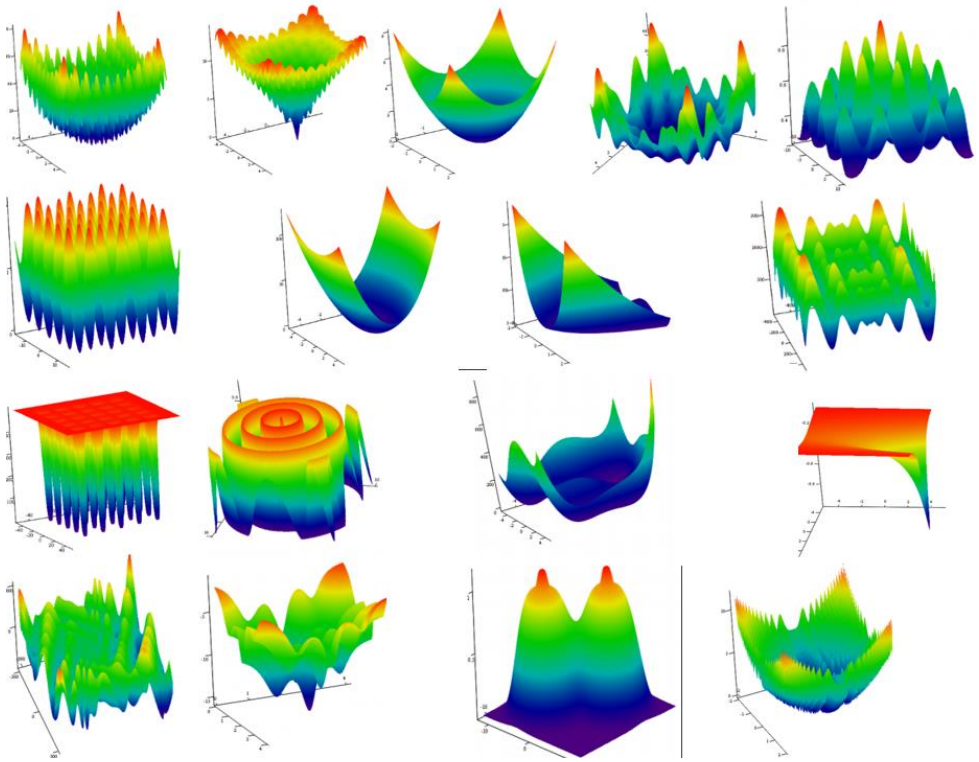


Рисунок 2.1 – Тестовые функции для задач оптимизации (https://raw.githubusercontent.com/Harrix/HarrixTestFunctions/master/_HarrixTestFunctions.pdf)

Разнообразие современных методов оптимизации в том числе обосновано математически с помощью так называемой теоремы “no free lunch”. Упрощённо, не существует универсального метода, который эффективно определяет экстремум для любого типа задач. Отметим так же, что на практике тип задачи (ландшафт функции) определить довольно трудно и поэтому используют различные

методы от суррогатных моделей до применения различных методов для глобального и локального поиска.

Вопросы для самоконтроля:

1. Какие основные этапы решения практической задачи с помощью численной оптимизации можно выделить?
2. Какие методы оптимизации называются прямыми методами? Как они работают?
3. Какие тестовые функции для задач оптимизации существуют и зачем они используются?
4. В чем смысл теоремы об отсутствии бесплатного завтрака?
5. Как выбрать метод оптимизации для решения конкретной прикладной задачи?

ГЛАВА 3. ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ О КЛАССИЧЕСКИХ МЕТОДАХ ОПТИМИЗАЦИИ.

Прежде чем переходить к современным методам оптимизации необходимо определить понятия, которые будут использоваться в дальнейшем. Данная глава содержит некоторые определения и алгоритмы, с которыми человек, интересующийся современными методами оптимизации должен быть знаком.

3.1 Функции одной переменной

В качестве функций мы будем рассматривать отображения

$$f: R^n \to R\ , n \in Z$$

Рассмотрим самый простой случай $n = 1$, функцию одной переменной. Тогда, следуя за классическими книгами, мы вводим бесконечно малое приращение переменной dx. В таком случае можно рассмотреть поведение функции при бесконечно малом приращении аргумента.

Рассмотрим выражение приращение функции при бесконечно малом приращении аргумента:

$$\mathrm{df} = \mathrm{f(x + dx)} - \mathrm{f(x)}$$

Функция называется непрерывной если при любом порядке dx можно подобрать верхнюю оценку для приращения df. Такая оценка может иметь вид

$$\mathrm{f(x + dx)} \sim \mathrm{f(x)} + \mathrm{f'(x)dx}$$

То есть, для любой точки x мы можем построить такую функцию $f'(x)$ которая может служить для оценки приращения функции. Такая линейная часть поведения функции при бесконечно малом приращения аргумента называется производной. Введём для неё классическое обозначение $f'(x)$. Заметим, что если $f'(x)$ в свою очередь непрерывна, то сама функция $f(x)$ называется гладкой.

Назовём экстремумом функции такую точку x_0 в окрестности которой приращение функции равно нулю. Для более детального анализа вводят более точные оценки приращения функции

$$f(x + dx) = f(x) + f'(x)dx + f''(x)dx^2 + \cdots$$

Что ведёт к необходимому условию существования экстремума в знакомом виде

$$f'(x_0) = 0$$

Достаточные же условия имеют довольно разнообразный вид. На практике это означает, что в окрестности точки x_0 нет значений функции больше (если $f''(x)$ существует и не равна нулю, то это эквивалентно $f''(x)<0$) такая точка экстремума называется максимумом) или меньше (минимум) данного. Такой экстремум называется локальным, потому что таким образом мы описываем поведение функции только в окрестности одной точки.

Отметим отдельно случай, когда все существующие производные равны нулю в окрестности данной точки. Такая точка называется седловой. Более интересно рассматривать такие точки для функций нескольких переменных.

Так же определяют глобальный экстремум для этого как правило задают конечную область (интервал в случае функции одной переменной) и ищут значения функции во всех локальных экстремумах и на границе области. Максимальное (минимальное) значение функции на таком множестве точек для непрерывной функции называют глобальным экстремумом.

Отметим, что функции, терпящие разрыв, в том числе определённые только на дискретной сетке требуют других методов исследования. Для отыскания экстремума такой функции зачастую требуется полный перебор всех значений. Некоторые эвристики, позволяющие уменьшить пространство для перебора, обсуждаются в главе 4 «Популяционные алгоритмы, неградиентные методы».

3.2 Функции нескольких переменных

Для более общего случая функции, когда $n > 1$, линейная часть приращения описывается как

$$\mathrm{f}(\mathrm{x}_1 + \mathrm{dx}_1, x_2 + dx_2, \dots) = \mathrm{f}(\mathrm{x}_1, x_2, \dots) + \frac{\partial \mathrm{f}}{\partial x_1}\mathrm{dx}_1 + \frac{\partial \mathrm{f}}{\partial x_2}\mathrm{dx}_2 + \cdots$$

Компонент линейной части приращения по отношению к бесконечно малому увеличению каждой из координат называется частной производной $\frac{\partial f}{\partial x_i}$. Локальным экстремумом функции так же называется точка, в окрестности которой приращение равно нулю.

Вектор из линейных частей приращения называется градиентом функции, будем обозначать его как

$$\nabla f = \{\frac{\partial \mathrm{f}}{\partial x_1}, \frac{\partial \mathrm{f}}{\partial x_2}, \dots\}$$

С введённым обозначением вектора градиента можно так же ввести необходимое условие существования экстремума:

$$\nabla f = 0$$

Квадратичная форма из вторых производных образует матрицу, которая называется матрицей Гессе:

$$H = \begin{pmatrix} \frac{\partial^2 \mathrm{f}}{\partial x_1^2} & \frac{\partial^2 \mathrm{f}}{\partial x_1 \partial x_2} & \cdots \\ \frac{\partial^2 \mathrm{f}}{\partial x_2 \partial x_1} & \ddots & \\ \vdots & & \end{pmatrix}$$

В теории квадратичных форм вводится понятие знакопостоянных форм, одним из признаков такой формы является знакопостоянство всех её собственных чисел. При этом если все собственные числа строго больше или меньше нуля, то матрица называется (строго) положительно или отрицательно определённой. Если гессиан строго положительно определён в точке, где градиент равен нулю, то такая точка является точкой минимума.

Всё вышесказанное так же, как и в случае функции одной переменной справедливо только локально. Для нахождения глобального экстремума непрерывной функции необходимо определить область и добавить к нулям градиента значения функции на границе области.

Существуют так же задачи, например выпуклой оптимизации, для которых заранее известно о количестве и глобальности экстремумов. Подобные задачи как правило применяются в более теоретизированных приложениях, когда известен тип функции, области оптимизации и вид ограничений.

3.3 Линейное программирование

Оптимизация линейных функции – особый случай задач оптимизации, они имеют постоянный градиент, то есть внутри любой замкнутой области нет точек, для которых выполнено необходимое условие существования экстремума. Дополнительно, такие функции имеют вырожденный гессиан, а следовательно, монотонны. Таким образом, задача оптимизации сводится к обходу границы области. Чем оптимальнее выполнен обход, тем быстрее решается задача.

Существуют методы, позволяющие решать задачу линейного программирования за полиномиальное время. Однако, как правило, в литературе рассматривается менее эффективный симплекс-метод, который лишь полиноминален в среднем.

Одной из классических формулировок является транспортная задача, которую как правило формулируют в виде таблицы.

Таблица 3.1 – Формулировка транспортной задачи в виде таблицы

Сырьё	**Затраты сырья на единицу продукции**			**Запас сырья**
	A1	A2	A3	
I	3,5	7	4,2	1400
II	4	5	8	2000
Прибыль	1	3	3	

Математическая форма записи имеет вид

$$f = x_1 + 3x_2 + 3x_3 \to max$$
$$s.t.\begin{cases} 3.5\,x_1 + 7x_2 + 4.2x_3 = 1400 \\ 4\,x_1 + 5x_2 + 8x_3 = 2000 \\ x_1, x_2, x_3 > 0 \end{cases}$$

Заметим, что ограничения могут иметь вид неравенств, однако симплекс метод требует равенств (что и отражено в его названии) в таком случае вводят дополнительные переменные с нулевой прибылью.

Одним из методов решения подобных задач с малым количеством переменных является графический метод, когда все ограничения изображаются на одном графике, а функция стоимости движется по поверхностям уровня

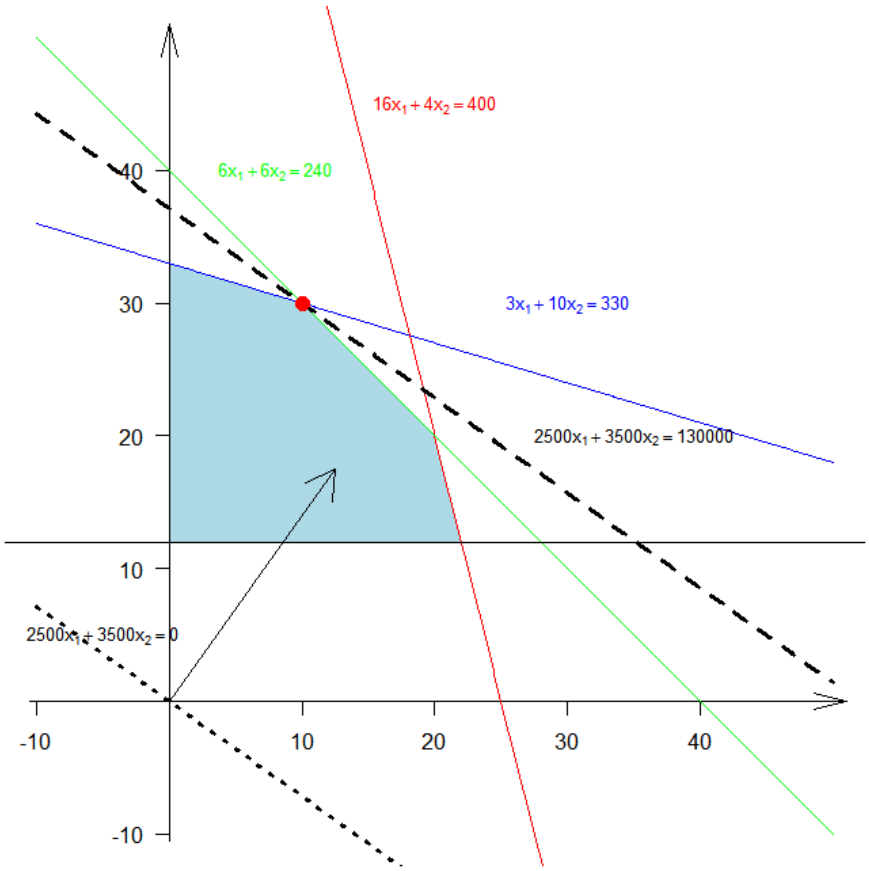


Рисунок 3.1 – Пример графического решения задачи линейного программирования (https://tushavin.ru/lp-graph/)

Для задач с большим числом переменных (в реальных задачах их число достигает десятков и даже сотен тысяч) применяют методы оценки замкнутости области ограничений, а затем методы обхода границ, вроде симплекс метода.

3.4 Численные методы: поиск корней, конечные разности и автоматическое дифференцирование как методы взятия производной

Внутри многих численных методов лежат в том числе и численные методы поиска корней уравнения (например, для решения уравнения $f'(x) = 0$). Метод Ньютона один из классических методов, позволяющих численно найти корень уравнения. Итерация метода Ньютона представляет собой:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Метод призван найти приближённое решение x_* уравнения $f(x_*) = 0$, ход итераций показан на рисунке:

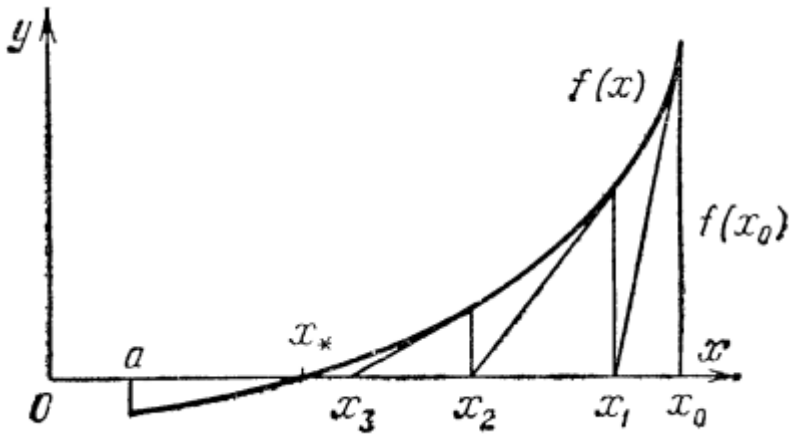


Рисунок 3.2 – Графическая иллюстрация итераций метода Ньютона нахождения корней уравнения (https://math.by)

Для более сложных уравнений и систем существуют и другие методы, например метод пристрелки, метод симуляций (Монте-Карло).

Вычисление производной на компьютере зависит от формы представления функции – это может быть сеточная функция (когда известны значения лишь на дискретном наборе точек) или граф вычислений, например, нейронные сети зачастую имеют именно такую форму записи.

Для сеточной функции существует несколько методов вычисления производной из наиболее прямых в применении метод конечных разностей и метод конечных элементов. Можно так же аппроксимировать или интерполировать функцию и вычислить производную интерполирующей или аппроксимирующей функции, в том числе и нейронной сети.

Рассмотрим метод конечных разностей. В нём первая производная заменяется на выражение такого вида

$$f'(x_i) \sim \frac{f(x_{i+1}) - f(x_{i-1})}{|x_{i+1} - x_i| + |x_i - x_{i-1}|}$$

Подобная замена называется переходом к центральной конечной разности. Для границы рассматриваемой области есть схемы, которые учитывают отсутствие узлов сети слева или справа. Схема представленная выше имеет порядок аппроксимации $O(h^2)$. Опуская детали, можно сказать, что при уменьшении среднего шага сетки, значения, вычисленные с помощью приближения стремятся к «истинным» как квадрат шага сетки. Существуют схемы и более высокого порядка. В большинстве практических применений схемы более высокого порядка уместно применять лишь на границах области.

Автоматическое дифференцирование применяется, когда известен граф вычислений, приводящий к данной функции. Обычно пользуются цепным пра-

вилом дифференцирования проходя граф вперёд от корня к листьям или в обратном от листьев к корню. При этом для отображения $f: R^n \to R^m$,где $m \ll n$, обратный проход является более эффективным, что привело к методу обратного распространения ошибки и позволило вычислять градиенты для глубоких нейронных сетей.

Прямой ход имеет более математизированную форму записи с использованием дуальных чисел [11]. Дуальные числа – расширение поля вещественных чисел, приводящее к другому пространству, нежели чем вещественные. Вводится элемент ε, такой что $\varepsilon^2 = 0$. Можно доказать, что:

$$f(x + y\varepsilon) = f(x) + y\varepsilon f'(x)$$

Таким образом «мнимая» часть функции от дуального аргумента пропорциональна производной, что позволяет автоматически получать значение производной в точке.

Исторически классические методы оптимизации рассматривались как группа аналитических методов, позволяющих найти экстремум выражения, записанного в аналитической форме для оценки данного выражения сверху или снизу. Изобретение симплекс-метода привело к расцвету численных методов оптимизации, до сих пор большая часть задач в промышленности либо формулируется в виде задач линейного программирования, либо линеаризуется. Нейронные сети так же стимулируют развитие численных нелинейных методов.

Вопросы для самоконтроля:

1. Какая функция называется непрерывной? Как оценивается ее приращение?
2. Что такое градиент в математике и как он используется в градиентных методах оптимизации?
3. В чем смысл транспортной задачи? Как она формулируется и решается?
4. В чем смысл симплекс метода?
5. Что такое автоматическое дифференцирование?

ГЛАВА 4. ГРАДИЕНТНЫЕ МЕТОДЫ. КЛАССИЧЕСКИЕ И НЕ ОЧЕНЬ

4.1. Классические градиентные методы. Метод Ньютона, градиентный спуск

В предположении, что функция дифференцируема дважды, можно воспользоваться методом парабол, итерации которого так же служат для поиска корня уравнения $f'(x) = 0$ с помощью метода Ньютона, описанного выше. Формальная запись итерации метода парабол имеет вид [12]:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$

Как правило, используют численное дифференцирование для нахождения первой и второй производных. Так же для подобных методов требуется хорошее начальное приближение в случае функции одной переменной его ищут графически.

Для функций нескольких переменных существует расширение метода Ньютона, которое называется методом градиентного или наискорейшего спуска.

Итерация метода градиентного спуска имеет вид:

$$\vec{x}^{t+1} = \vec{x}^{t} - \eta \nabla F(\vec{x}^{t})$$

Как видно, вместо оптимального шага для функции одной переменной $\frac{1}{f''(x)}$ в классическом градиентном спуске вводится параметр η. По-прежнему, оптимальным является выбор $\eta = H\big(f(x)\big)^{-1}$, где H – матрица Гессе. Однако на практике вычислять матрицу Гессе вычислительно трудно. Для определения параметра η существуют различные модификации метода градиентного спуска, которые так или иначе используют приближение матрицы Гессе для нахождения этого параметра. В классическом же варианте метода наискорейшего спуска параметр η определяется простым перебором значений для каждой конкретной задачи.

Остаётся вопрос начального приближения. Проиллюстрируем локальность градиентного спуска на следующем примере:

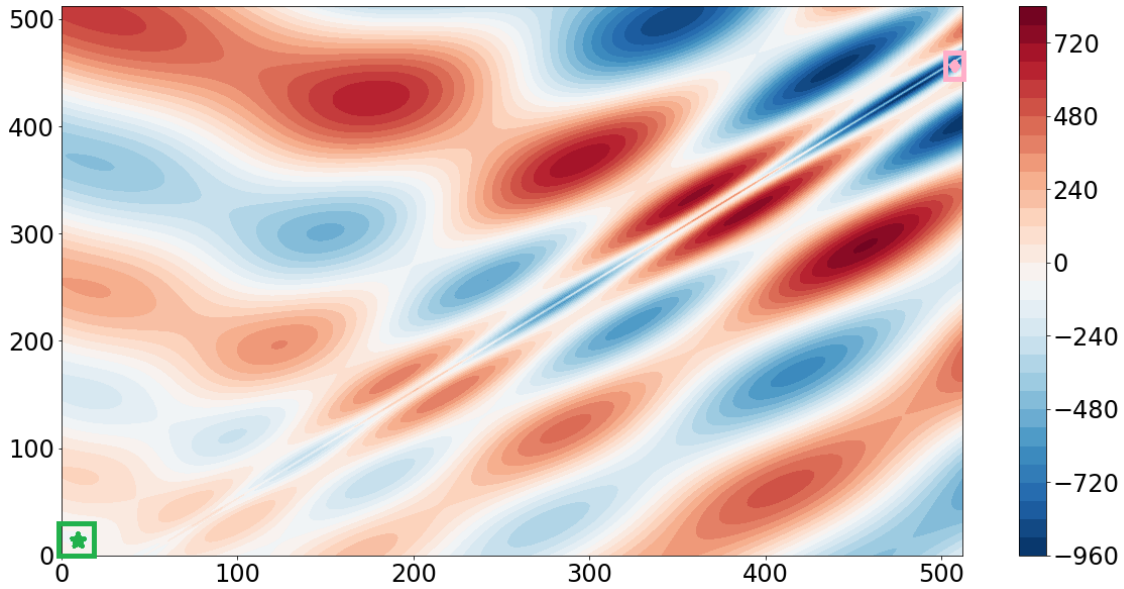


Рисунок 4.1 – Ландшафт тестовой функции для минимизации звезда и ромб – две области рассмотренные ниже

Из начальной точки (0,0) простой градиентный спуск проходит следующий путь в процессе итераций (обратите внимание на шкалу исходной функции и увеличенного рисунка):

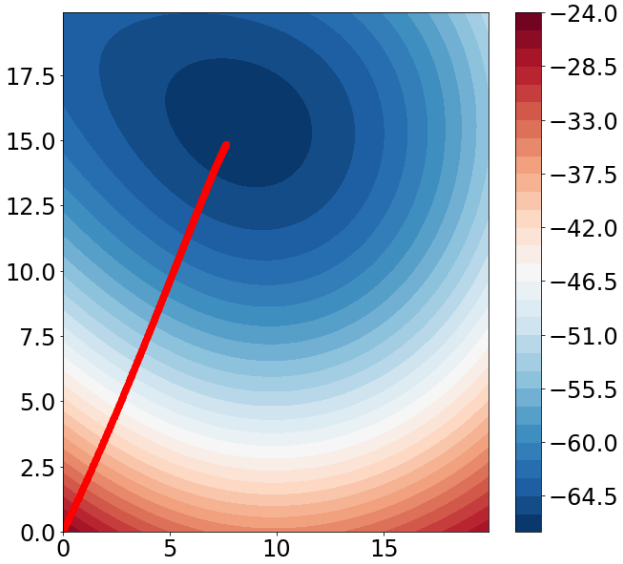


Рисунок 4.2 – Часть ландшафта (звезда на рисунке 4.1), линия – итерации градиентного спуска

Очевидно, произвольная точка не подходит для нахождения глобального минимума функции, обладающей несколькими локальными минимумами.

Простым перебором значений функции при целочисленных значениях аргумента из отрезков в условиях ограничений 0<x<512, 0<y<512 можно определить лучшее начальное приближение. Из начальной точки (511,403) простой градиентный спуск проходит следующий путь в процессе итераций в условиях ограничений x<512, y<512 (обратите внимание на шкалу исходной функции и увеличенного рисунка):

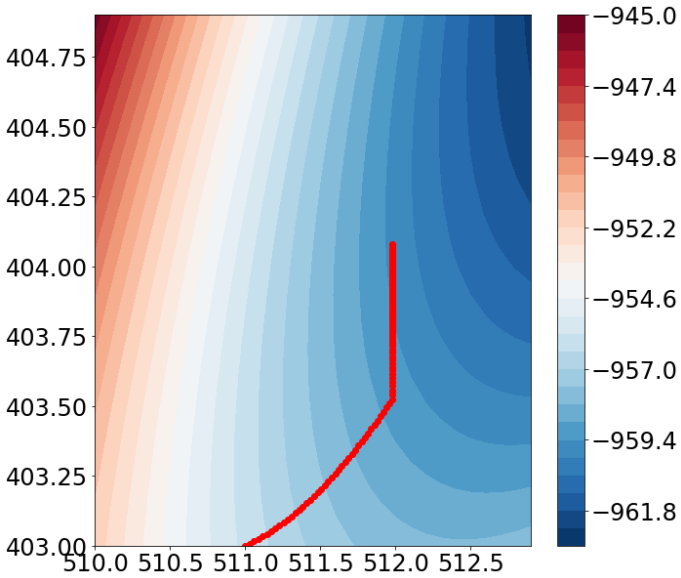


Рисунок 4.3 – Часть ландшафта (ромб на рисунке 4.1), линия – итерации градиентного спуска

Таким образом, даже простой переборный глобальный поиск, совмещённый с градиентным спуском, помогает определить лучшее начальное приближение.

В современных решениях очень часто используют комбинации популяционных алгоритмов (см. Главу 5) для глобального поиска и градиентные методы для локальной до-оптимизации. Для параметров нейронный сетей существуют различные эвристики, позволяющие инициализировать веса нейронной сети в среднем чуть более близкие к оптимальному решению, чем инициализированные случайно для заданной функции активации. Например, инициализация весов Xe для ReLU [13].

4.2. Модификации градиентного спуска

Для задач калибровки математических моделей и моделей в нейронных сетях требуются модификации метода градиентного спуска для обеспечения устойчивости сходимости и в том числе увеличения скорости сходимости.

Группа алгоритмов, учитывающих гессиан являются аналогом метода парабол для функции одной переменной. Классическим представителем данных методов является ADAM. При отсутствии седловых точек, метод обеспечивает скорость сходимости выше, чем обычный метод градиентного спуска.

Итерация ADAM [14] имеет вид:

$$\vec{x}^{t+1} = \vec{x}^{t} - \frac{\eta}{\sqrt{v_t + \varepsilon}} \mathrm{m_t} \nabla F(\vec{x}^{t})$$

,где $m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla F(\vec{x}^t)$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\nabla F(\vec{x}^t)^2$, η, β_1, β_2 – параметры, ε – малая константа (тоже параметр). В качестве начальных параметров (параметров по умолчанию) обычно берут $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 =$

0.999, $\varepsilon = 10^{-8}$. Отметим, что подбор параметров под конкретную задачу может увеличить скорость сходимости.

Для увеличения устойчивости получаемого результата в задачах калибровки моделей в том числе и моделей машинного обучения, используют стохастический градиентный спуск. Данная группа методов использует поднаборы данных для наблюдений в процессе градиентного спуска для обновления части или всего набора параметров модели.

На практике это означает, что градиент функции ошибки считается лишь для случайно выбранного поднабора данных заданного размера и иногда для случайно выбранного поднабора параметров.

Градиентный спуск получил развитие в современных методах обучения нейронных сетей. Метод обратного распространения ошибки позволил, с одной стороны, обучать глубокие нейронные сети, а с другой стимулировал развитие численных и аналитических алгоритмов градиентного спуска с точки зрения параллелизации и ускорения сходимости.

Вопросы для самоконтроля:

1. Какой метод можно считать расширением метода Ньютона для функций нескольких переменных?
2. Каким образом можно определять параметр η в методе градиентного спуска, если вычисление матрицы Гессе затруднительно?
3. Каким образом можно определить лучшее начальное приближение для градиентного спуска при ограничениях на значения переменных?
4. Какие методы часто используются в современных решениях для глобального и локального поиска оптимальных решений?
5. Какие преимущества имеет метод ADAM по сравнению с обычным методом градиентного спуска?
6. Как стохастический градиентный спуск использует поднаборы данных и параметров в процессе обновления весов модели?

ГЛАВА 5. ПОПУЛЯЦИОННЫЕ АЛГОРИТМЫ, НЕГРАДИЕНТНЫЕ МЕТОДЫ

Неградиентные методы применяются к задачам, в которых пространство поиска слишком широко, либо известно, что число локальных экстремумов слишком велико, а разница между оптимальными значениями в окрестности экстремума, наоборот, небольшая. Как правило популяционные алгоритмы используют для «глобального» поиска, то есть для поиска точек, которые доставляют близкие к экстремуму значения, а затем, если необходимо, дополняют «глобальный» поиск, «локальными» градиентными методами.

5.1. Интеллектуальные методы оптимизации

В данном разделе рассмотрены применяемые методы оптимизации, которые можно отнести к *интеллектуальным* – таким, в рамках которых применяются дополнительные эвристические подходы, зачастую проводящие аналогии с поведением живых существ и их сообществ (bio-inspired), или природными процессами [15]. Также интеллектуальными называют методы, позволяющие решать задачи оптимизации в условиях неопределенности, имитируя при этом действия эксперта.

Рассмотренные ранее методы оптимизации можно отнести к *траекторным* (таким, в которых на каждой итерации на основе предыдущих создается одно новое решение). Однако в задачах многокритериальной оптимизации широкое применение находит *популяционный* (population-based) подход [16]. Популяционные методы основываются на принципе анализа и модификации нескольких кандидатов в возможные решения одновременно (их совокупность и называется популяцией). Популяционные алгоритмы относят к классу «вдохновленных природой» – проводится аналогия с популяциями (роями) живых существ, их поведением и, для генетических алгоритмов, – эволюцией. Как показано на рисунке 5.1.1, популяция обновляется на каждой итерации алгоритма.

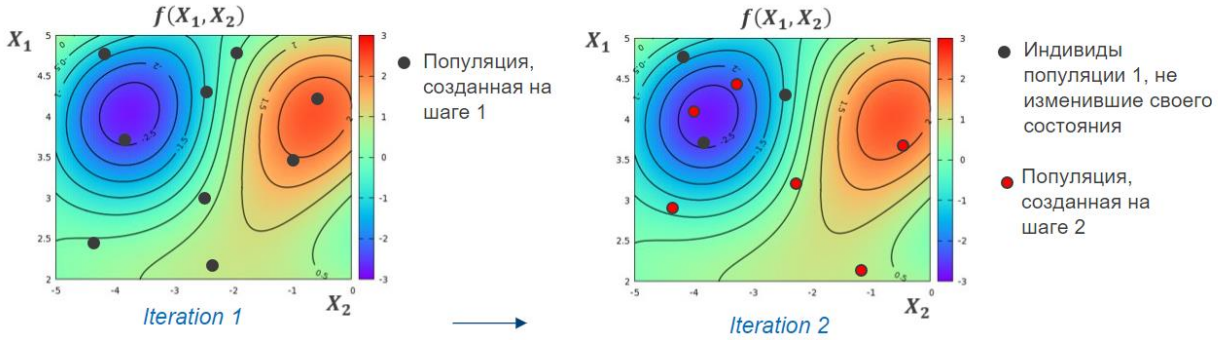


Рисунок 5.1.1 – Изменчивость популяции в ходе оптимизации.

Общим недостатком не-популяционных алгоритмов стохастической оптимизации является сложность их распараллеливания, в то время как популяцион-

ные методы позволяют реализовать «естественный» параллелизм, поскольку поведение агентов в рамках одной популяции чаще всего независимо и поэтому может быть смоделировано параллельно (классический обзор подходов к распараллеливанию алгоритмов оптимизации дан в [17]).

Популяционные алгоритмы широко применяются для решения практических задач. Так, в работе [18] *алгоритм муравьиной колонии* (ant colony algorithm) применяется для настройки гидрологической модели. Муравьиный алгоритм позволяет преобразовать пространство поиска в граф, в котором на последующих итерациях выделяются оптимальные пути. Однако для его применения необходимо дискредитировать калибруемые числовые переменные. Существуют и многочисленные «природные» модификации данного метода – «пчелиные», «голубиные» и др. алгоритмы, отличающиеся в некоторых деталях реализации.

Применяются для задач настройки параметров и *роевые методы* (swarm optimization). Они основаны на аналогии с динамикой роя материальных частиц (соответствующих векторам оптимизируемых переменных) с заданными координатами и скоростью, перемещающихся в соответствии со значениями целевых функций как для каждой отдельной частицы, так и для её «соседей». Благодаря этому весь рой постепенно движется к «центру притяжения» и, в конечном счете, глобальному минимуму [16]. Эффективность метода роя частиц сильно зависит от выбора гиперпараметров. Кроме того, из-за наличия «центральной» точки роя он может остановиться в локальном минимуме при изначальном «неудачном» выборе направления.

Популяционные методы хорошо подходят для решения многокритериальных задач. В рамках решения задачи многокритериальной оптимизации критерии, которые участвуют в оптимизационной задаче, могут быть согласованными (когда оптимизация одного критерия приводит к улучшению других), нейтральными (не влияющими друг на друга) и, в худшем случае конкурирующими, что очень часто встречается при решении реальных задач. В последнем случае решение может быть достигнуто только путем нахождения компромисса. Математическую модель компромисса в оптимизации принято строить на основе понятия множества Парето.

Решение x^* называется эффективным (недоминируемым), если во множестве допустимых альтернатив D не существует решения, которое по целевым функциям было бы не хуже, чем x^*, и, по крайней мере по одной целевой функции было бы строго лучше, чем x^*. Исходя из этого, решение многокритериальной задачи оптимизации следует выбирать из множества Парето, т. к. любое другое, очевидно может быть улучшено некоторой точкой Парето как минимум по одному критерию без ухудшения других критериев. Поскольку решения из множества Парето не могут превосходить друг друга, после формирования данного множества задача многокритериальной оптимизации может считаться решенной.

5.2. Эволюционные методы оптимизации

Несмотря на существование множества подходов к оптимизации (далеко не все рассмотрены в данной тексте), которые можно применять для решения задач оптимизации, при высокой неопределенности идентифицировать устойчивые наборы решений позволяет применение *эволюционных алгоритмов*. Они основаны на итеративном применении к популяции решений-кандидатов эволюционных операторов. Среди часто применяемых эволюционных методов можно выделить генетические и дифференциальные подходы.

Базовая постановка задачи эволюционной оптимизации такова:

Для функции приспособленности $W(x)$ в пространстве поиска X требуется найти

$x^* = \underset{x \in X}{\operatorname{argmax}} W(x)$ для каждого индивида (особи) в k-м поколении $F^k = \{f_1^k \dots f_n^k\}$.

Хромосома x состоит из генов (числовых, категориальных или иных параметров). Создание нового поколения выполняется с помощью операторов селекции S, кроссовера C и мутации M:

$$P^{k+1} = M\ C\ S(P^k, F^k)$$

Назначение данных операторов таково:

– селекция – отбор индивидов по определенному правилу (используются селекция по принципу «рулетки», турнирная селекция, селекция на основе включения в новое поколение лучших особей из предыдущего – «элитизма»). Операции производятся со значением функции пригодности соответствующих индивидов;

– скрещивание (рекомбинация, кроссовер) – перемешивание информации, хранящейся в хромосомах индивидов. Операции производятся с хромосомами (генотипами) соответствующих индивидов;

– мутация – случайное изменение информации, хранящейся в хромосомах. Операции производятся с хромосомами (генотипами) соответствующих индивидов;

Создание новых популяций прекращается при достижении критерия останова. Среди возможных реализаций критерия можно привести:

- выходом на «плато» функции приспособленности;
- отсутствие приращения значения целевой функции выше заданного порога за определенное число поколений
- исчерпание числа максимального поколений или лимита времени

Набор оптимизируемых переменных в терминологии эволюционных алгоритмов называют генотипом (хромосомой), а значений целевой функции – фенотипом [16].

Генетические эволюционные алгоритмы эффективны при надежном поиске решения в многомерном пространстве параметров с нехваткой исторических данных для оценки качества [19] и высокой степенью неопределенности.

Задачи многокритериальной оптимизации хорошо решаются с помощью популяционных алгоритмов из-за возможности оперирования фронтами Парето (например, в семействе алгоритмов SPEA [20]) или метриками сравнения для многомерных целевых функций (гиперобъем и т.п.). Для некоторых задач, связанных с наличием ансамбля модельных реализаций, целесообразно применение коэволюционных алгоритмов [21], характеризующихся наличием нескольких популяций (см. рисунок 5.2.2). Эти и иные алгоритмы показаны на рисунке 5.2.1.

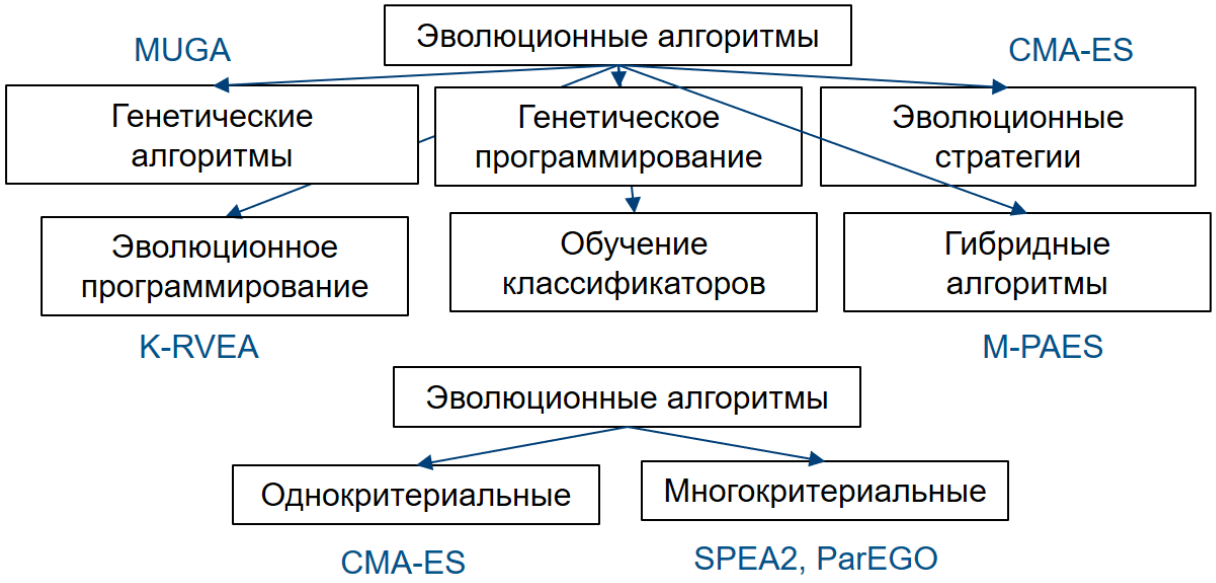


Рисунок 5.2.1 – Возможная классификация современных алгоритмов эволюционной оптимизации

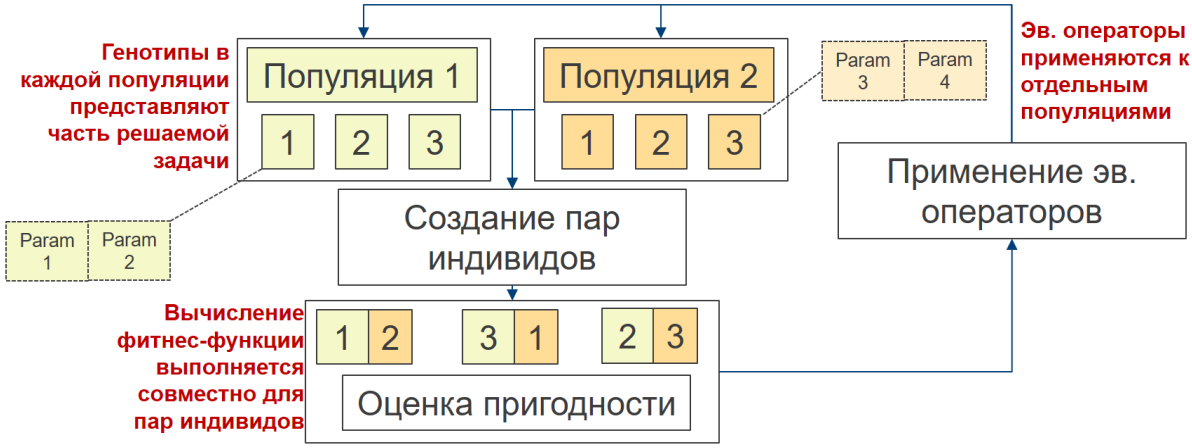


Рисунок 5.2.2 – Схема алгоритма ко-эволюционной оптимизации

К достоинствам эволюционных методов оптимизации также относятся [22]:

– способность эффективно выявлять окрестность глобального минимума;

– баланс между расширением пространства поиска (exploration) и уточнением найденных минимумов (exploitation);

– возможность гибкой адаптации к решаемой задаче с помощью реализации новых эволюционных операторов и настройки гиперпараметров;

– удобство интеграции с оптимизируемым «черным ящиком» (см. рис 5.2.3);

– эффективность в условиях стохастического характера поверхности целевой функции;

– возможность наглядной и интерпретируемой визуализации процесса эволюции;

– применимость как для дискретных, так и непрерывных переменных;

– легкость распараллеливания расчета целевой функции для индивидов в популяции;

– возможность сохранения в популяции несколько Парето-оптимальных решений с оставлением окончательного выбора эксперту.

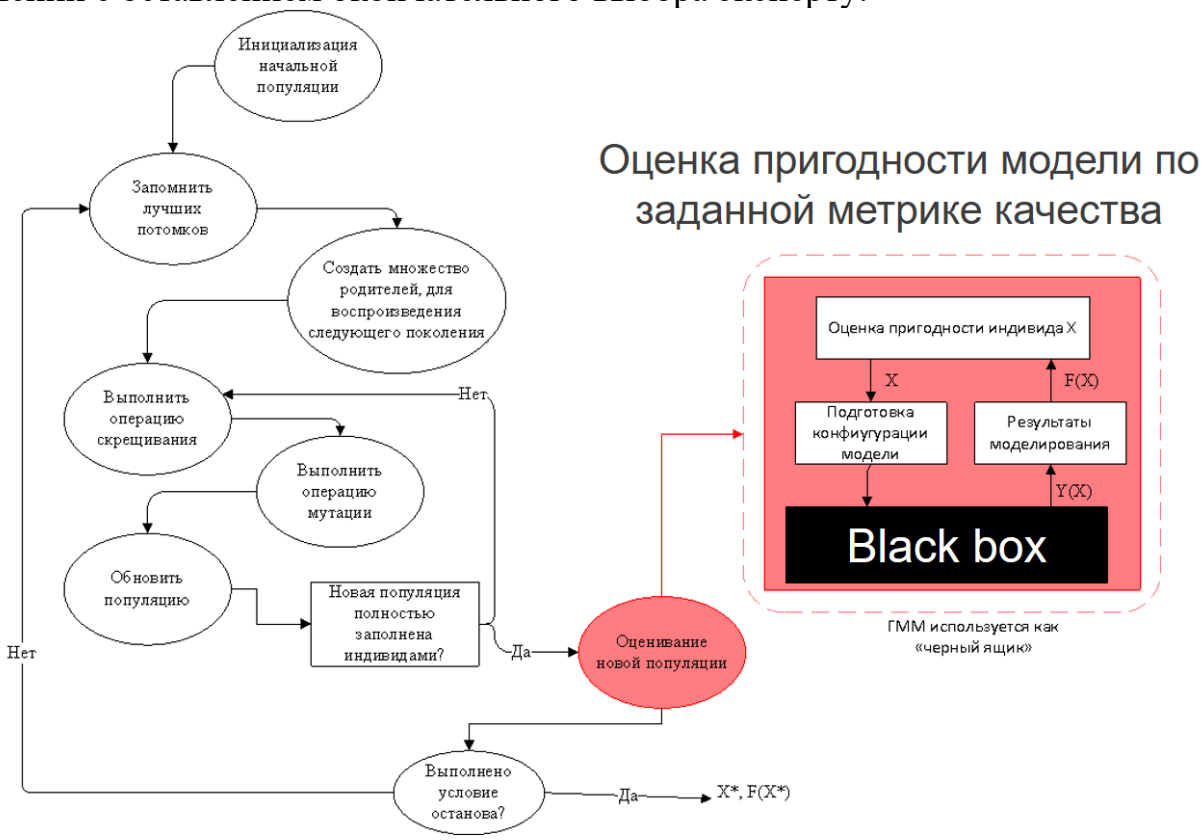


Рисунок 5.2.3 – Применение эволюционного алгоритма для решения задачи оптимизации «черного ящика»

Решение многих практических задач даже наиболее совершенными из методов, рассмотренных ранее, как правило, затруднительно из-за требуемых вычислительных ресурсов или недостатка данных для настройки. Поэтому целесообразно применение дополнительных методов, предназначенных для решения отдельных задач.

Первая группа таких методов – методы суррогатной оптимизации, основанные аппроксимации целевой функции с помощью построения суррогатной модели [23] и её применения в процессе оптимизации.

Суррогатные модели позволяют ускорить сходимость эволюционного алгоритма, избежать попадания в связанные с неопределенностью моделирования локальные минимумы и как следствие, уменьшить число вычислений целевой

функции, требуемое для решения задачи оптимизации. Суррогатные модели в литературе также называет метамоделями [24]. Суррогатная модель для целевой функции может быть основана на нейронной сети, полиноминальных моделях, радиальных базисных функциях, сплайнах и др. подходах [24].

Так, полиноминальные модели хорошо воспроизводят крупномасштабные вариации и тренды в пространстве параметров. Порядок полинома определяется формой поверхности целевой функции. Для более корректной его оценки, как правило, применяется кроссвалидация [24]. Наиболее широко используется упомянутый ранее кригинг. Он обладает несколькими преимуществами перед другими методами аппроксимации целевой функции:

– хорошо интерполирует исходные данные, при этом сохраняя точные значения в известных точках;

– позволяет оценить неопределенность предсказанных моделью кригинга значений;

– не задает требований к равномерности разброса исходных точек в пространстве параметров.

Недостатком кригинга является вычислительная сложность построения модели для пространства параметров большой размерности.

Широко распространено использование суррогатного подхода к моделированию в сочетании с эволюционными алгоритмами. Методы на основе такого подхода называют суррогатными эволюционными алгоритмами (СЭА, surrogate-assisted evolutionary algorithms, SaEA) [25].

Идентификация самой суррогатной модели также может потребовать большого количества вычислений целевой функции в высоком пространственном и временном разрешении (high fidelity, HFM), поэтому применяются методы, основанные на совместном применении полноценной HFM и её упрощенного варианта (low fidelity, LFM) с более грубым разрешением или упрощенными численными схемами [26]. LFM позволяют выполнить лишь достаточно грубую аппроксимацию целевой функции по сравнению с HFM, но их применение дает возможность быстро оценить приблизительное местоположение глобальных минимумов и перейти к уточнению найденных решений с помощью HFM. Такие подходы в общем случае называют оптимизацией с множественной детализацией (multi-fidelity) [27].

Недостатком СЭА является появление большого числа дополнительных гиперпараметров: размера популяции, числа поколений, шагов уточнения разрешения по пространству и времени, характеристик эволюционных операторов и т.д.), что увеличивает сложность задачи метаоптимизации такого алгоритма [28]. Для их подбора помимо простых эвристических подходов применяются и метаэволюционные алгоритмы (применение такого подхода при определении наилучшего набора гиперпараметров для СЭА описано в ряде исследований [29]). Однако этот подход требует большого числа запусков модели в ходе многократного выполнения «низлежащего» эволюционного алгоритма, чтобы найти соответствующие гиперпараметры метаалгоритма, что может сделать затраченное время неприемлемым даже несмотря на СЭА.

Кроме того, оптимизация с помощью СЭА неявно предполагает, что минимумы HFM модели расположены в той же области ландшафта целевых функций, что и минимумы LFM. Для физических моделей допущение о таком поведении оправдано тем, что в их основе лежат конкретные физические процессы, которые так или иначе воспроизводят модели и в том, и в другом разрешении. Однако применение СЭА в задачах оптимизации в каждом случае должно сопровождаться дополнительной валидацией для эмпирической проверки данного допущения.

Вопросы для самоконтроля:

1. Что такое интеллектуальные методы оптимизации и какие подходы они используют? Какие методы оптимизации относятся к траекторным, а какие к популяционным?
2. Какие преимущества имеют популяционные методы оптимизации?
3. Что такое генотип и фенотип в терминологии эволюционных алгоритмов?
4. Какие операторы входят в процесс создания нового поколения в эволюционных алгоритмах?
5. Что такое коэволюционные алгоритмы и в каких случаях они применяются?

ГЛАВА 6. ПРИМЕРЫ

В методах оптимизации довольно трудно ориентироваться без примеров данная глава содержит простой пример, который иллюстрирует принципы машинного обучения и три примера посложнее, которые основаны на реальных задачах, которые мы решали в рамках исследований.

6.1 Стохастический градиентный спуск для калибровки простых моделей

Рассмотрим один из типов задач машинного обучения – регрессию. В этой главе нас будет интересовать символьная регрессия. Рассмотрим модель в виде:

$$\mathrm{M}(x|\theta) = \sum_{i=0}^{N_{exp}-1} w_{2i}\exp(-w_{2i+1}x)\ , \theta = \{w_0, \dots w_{2N}\}$$

Стандартная задача обучения модели машинного обучения подразумевает изменение параметров θ так, чтобы при входных данных $X = (\hat{x}_1, \dots, \hat{x}_k)$ выводом модели были соответствующие им заданные $Y = (\hat{y}_1, \dots, \hat{y}_k)$. Для этого стандартно вводят меру ошибки (так же в машинном обучении она называется функцией потерь). В данной задаче предлагается уменьшать среднеквадратичную ошибку:

$$L = \sum_{j=1}^{k}\left(M(\hat{x}_j) - \hat{y}_j\right)^2 \to \min_{\theta}$$

Для нашей простой модели градиент функции потерь в пространстве параметров модели можно вычислить напрямую, он имеет вид:

$$\left.\frac{\partial L}{\partial w_i}\right|_{(\hat{x}_j,\hat{y}_j)} = \left(M(\hat{x}_j) - \hat{y}_j\right)\cdot \begin{cases} 2\exp(-w_{2i+1}\hat{x}_j), & i\ mod\ 2 = 0 \\ -2w_{2i}\,\hat{x}_j\exp\left(-w_{2i+1}\hat{x}_j\right), & i\ mod\ 2 = 1 \end{cases}$$

Для заданной сеточной функции с помощью похожих моделей можно решать задачу аппроксимации. Рассмотрим задачу аппроксимации функции $y = \sin(x)$, $x \in [0, \pi]$ для тысячи точек из отрезка и $N_{exp} = 24$.

Начальные параметры модели зададим случайно, равномерным распределением на отрезке [0,1].

На каждом шаге выберем случайный поднабор $A = \{a_1, \dots a_s\}$ (то есть $s \leq 1000$ случайных натуральных чисел) и соответствующие этим номерам точки $(\hat{x}_{a_l}, \hat{y}_{a_l})$. Дальше производим шаг градиентного спуска:

$$\theta_{t+1} = \theta_t - \eta\sum_{A}\nabla L(\vec{x}_t)|_{(\hat{x}_{a_l},\hat{y}_{a_l})}$$

Выбор поднаборов и шаги градиентного спуска выполняются до тех пор, пока функция ошибки не станет меньше заданного числа, то есть $L < \varepsilon$. Таким образом мы реализовали численный метод стохастического градиентного

спуска. Отметим, что шаг градиентного спуска может выполняться по усреднённому градиенту, вычисленному по поднабору, вместо последовательного, как показано в примере. Поэтому, следует помнить, используя готовый алгоритм, что он может быть реализован по-разному.

Скорее всего, читатель остановился на десятках тысяч итераций и получил неудовлетворительный результат вроде:

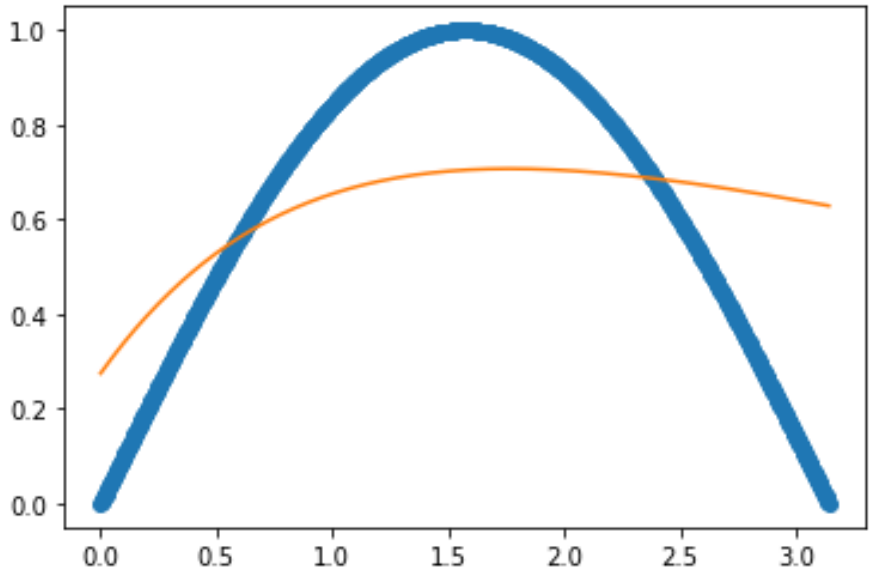


Рисунок 6.1.1 – Попытка аппроксимировать данные (синяя линия) суммой экспонент (оранжевая) при недостаточном числе итераций

При достаточно значительном увеличении числа шагов (в этой задаче нам понадобилось порядка миллиона) можно получить приемлемый уровень ошибки и модель вида:

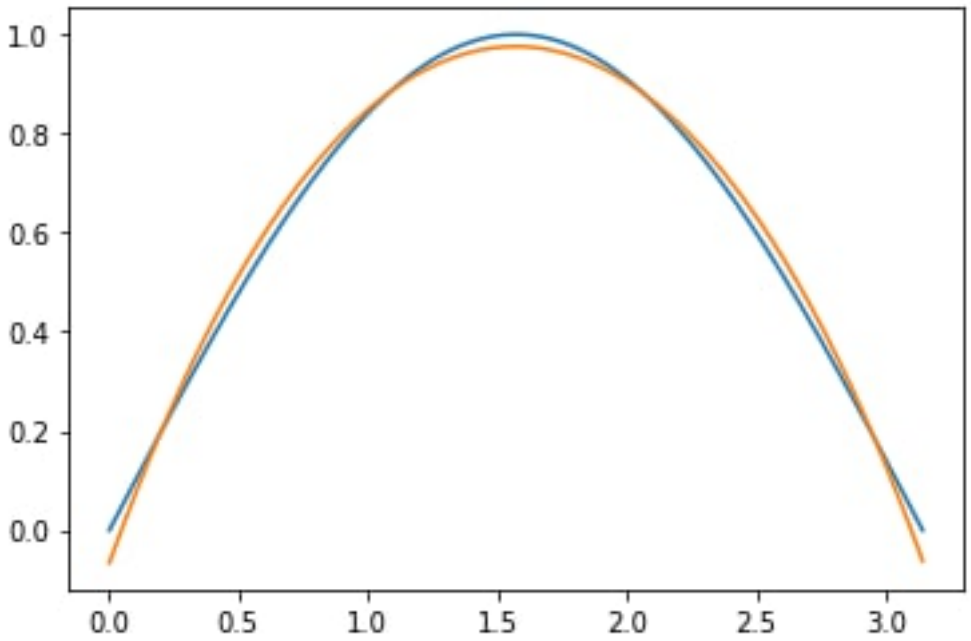


Рисунок 6.1.2 – Попытка аппроксимировать данные (синяя линия) суммой экспонент (оранжевая) при неприлично большом числе итераций

Это связано с неудачным выбором функций для аппроксимации, что отчасти было сделано для того, чтобы показать, что модель нужно выбирать адекватно предложенной задаче. Например, в качестве отправной точки можно взять существующие решения или решения из смежных областей.

Рассмотрим более удачный пример модели в виде полинома:

$$\mathrm{M}(x) = \sum_{i=0}^{N} w_i x^i$$

Градиент среднеквадратичной ошибки в пространстве параметров модели в таком случае имеет вид

$$\left.\frac{\partial L}{\partial w_i}\right|_{(\hat{x}_j, \hat{y}_j)} = \left(M(\hat{x}_j) - \hat{y}_j\right) \cdot 2\hat{x}_j{}^i$$

Рассмотрим полином степени N и N+1 точек, выбранных из данных для предыдущей задачи. Подобная постановка приводит нас к задаче интерполяции. В отличие от задачи аппроксимации нам фактически нужно найти решение системы по отношению к неизвестным параметрам:

$$\begin{cases} M(\hat{x}_1) = \hat{y}_1 \\ \dots \\ M(\hat{x}_{N+1}) = \hat{y}_{N+1} \end{cases}$$

Для полиномов такое решение существует и единственно, и минимизация функционала среднеквадратичной ошибки приводит к этому решению для большинства начальных приближений. Покажем результат для N=7:

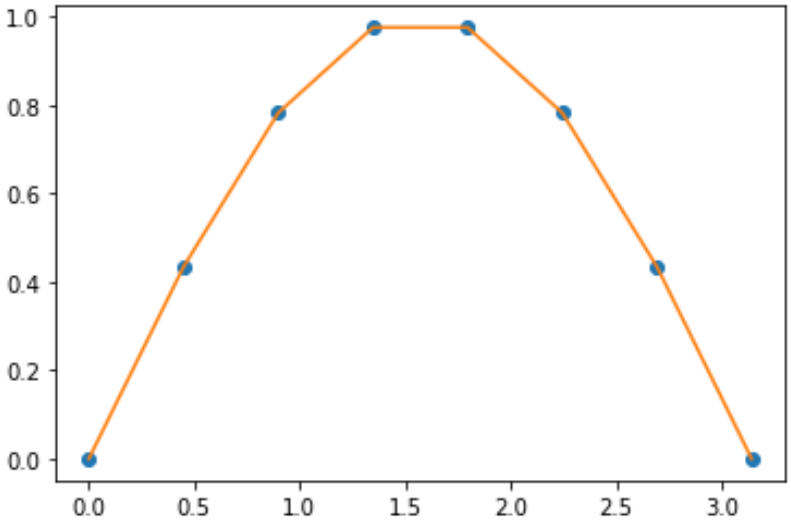


Рисунок 6.1.3 – Попытка аппроксимировать данные (синие точки) полиномом (оранжевая линия) при неприлично большом числе итераций

Очевидно, итоговый полином проходит точно через все точки.

Подводя итог можно сказать, что с помощью таких простых постановок в виде задач оптимизации можно аппроксимировать и интерполировать функции используя почти любой вид модели – от полинома до сложной нейронной сети. Более сложные модели требуют, тем не менее более продвинутых алгоритмов оптимизации, при этом оставляя постановку неизменной. Градиентные методы

известны тем, что сходятся к локальному минимуму и требуется хорошее начальное приближение для быстрого достижения результата.

6.2 Калибровка параметров численных моделей физических процессов

Под настройкой численных моделей в широком смысле понимается совокупность действий, позволяющих определить структуру, параметры и условия использования эмпирической части их численной реализации (т.е. той части, которая не основывается на первых принципах – естественных законах природы, представленных в виде уравнений) и в совокупности обеспечивающих приемлемую адекватность, оцениваемую на основе анализа согласованности результатов моделирования и данных натурных экспериментов.

Параметры, которые учитываются в модели, делятся на входные параметры (управляемые – эндогенные и независимые – экзогенные, описывающие влияние внешней среды), внутренние параметры модели (числовые, функциональные, структурные) и выходные параметры. Также в любой модели присутствует неопределенность. Под неопределенностью здесь понимается оценка расхождения между объективной реальностью и модельной аппроксимацией, характеризующая недостаток знаний о воспроизводимом объекте.

Общая структура численной модели, включающая в себя перечисленные виды параметров, а также связанные с ними задачи настройки приведены на рисунке 6.2.1.

Настройка под задачу формализуется как задача оптимизации, которая, в свою очередь, сводится к различным постановкам в зависимости от типа настраиваемых параметров. Так, для настройки выходных параметров в задачах прогноза применяются методы усвоения данных, настройка числовых параметров решается как задача непрерывной оптимизации, задача выбора структуры и функциональных параметров модели – как задача комбинаторной оптимизации.

Разработка методов настройки моделей позволяет не только обеспечить адекватность результатов моделирования, но и выполнить интерпретацию имеющихся данных натурных экспериментов. Так, например, настроенная «эталонная» модель может быть использована для оценки качества измерений, содержащих значительную неопределенность и выявления заведомо некорректных значений.

В узком смысле настройку моделей интерпретируют как подбор значений параметров численной реализации математической модели. Первичная настройка модели выполняется разработчиками (которые стараются обеспечить "равновесное" решение, одинаково удовлетворительное для различных ситуаций, или специализировать модель под конкретную решаемую задачу). Повторная настройка может выполняться при переносе моделей в новые пространственно-временные условия (другие акватории, даты и пр.) или другие масштабы моделирования.

Рисунок 6.2.1 – Общая структура численной модели и методы её настройки

Интересным с точки зрения настройки классом численных моделей являются модели гидрометеорологических процессов (гидрометеорологические модели, ГММ). Это связано в первую очередь с тем, что в задаче настройки ГММ большую роль играет неопределенность. Неопределенность в гидрометеорологических задачах можно рассматривать с точки зрения вероятностного подхода, в рамках которого она характеризуется распределением вероятности анализируемой переменной. Такое допущение обусловлено тем, что в гидрометеорологических процессах есть стохастические по своей природе составляющие: турбулентность, волновые движения жидкости и диффузионные явления. Подлежащие настройке параметры ГММ (ранее показанные в общем виде на рис. 6.2.1) представлены на рис. 6.2.2.

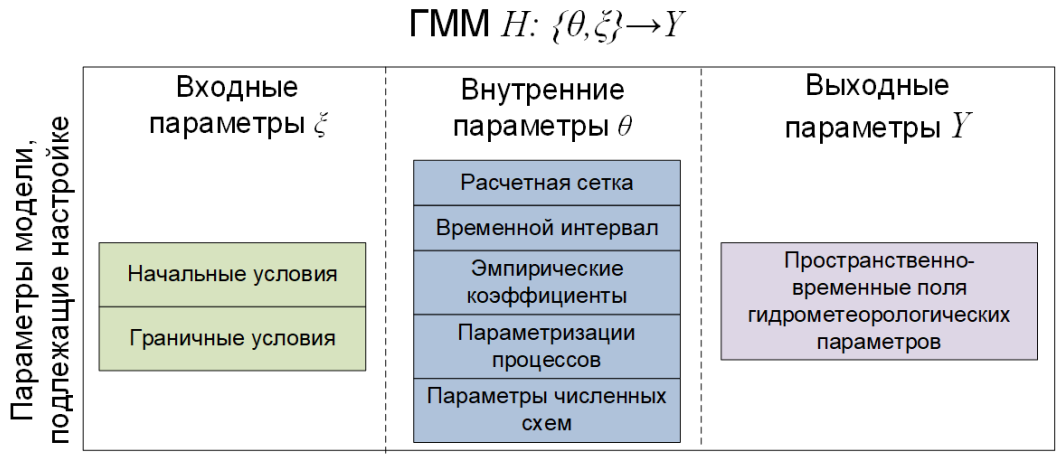


Рисунок 6.2.2 – Структура ГММ с подлежащими настройке параметрами

Результаты оптимизации параметров ГММ с помощью различных методов оптимизации показаны на рисунке 6.2.3.

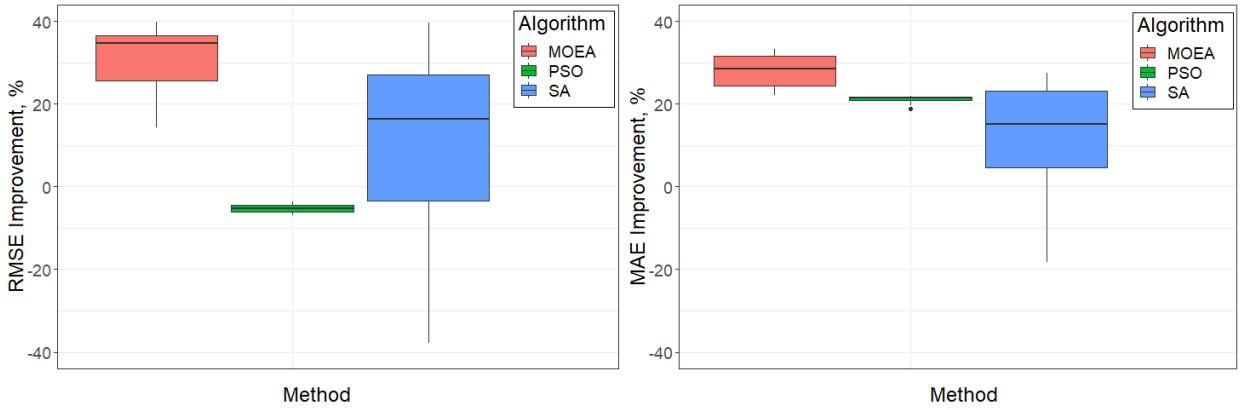


Рисунок 6.2.3 – Эффективность многокритериального эволюционного алгоритма SPEA2 (MOEA), метода роя частиц (PSO, Particle Swarm Optimisation) и метода имитации отжига (SA, Simulated Annealing)

Для повышения качества моделирования ГММ объединяют в ансамбли. Настройка ансамбля ГММ может быть выполнена как с помощью эволюционных, так и с помощью ко-эволюционных алгоритма (как показано на рис 6.2.4).

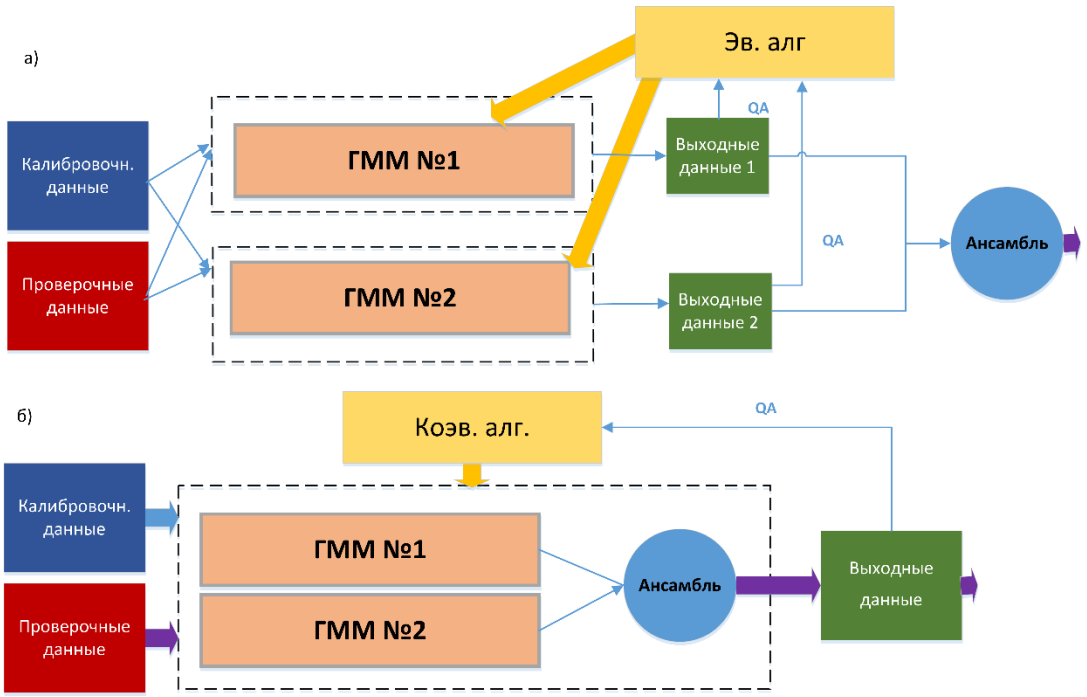


Рисунок 6.2.4 – Ансамбль реализаций модели ГММ в составе комплекса, настраиваемый (а) раздельно с помощью эволюционного алгоритма и (б) совместно с помощью коэволюционного алгоритма

6.3 Оптимизация структуры физических объектов

В качестве примера задачи оптимизации вектора числовых параметров будет использована задача поиска оптимальной конфигурации волнозащитных сооружений. Обычно для решения такой задачи задействуется эксперт в данной области, который должен предложить или выбрать такую конфигурацию, которая лучше всего соответствует набору критериев стоимости и качества. В большинстве случаев лицо принимающее решение подбирает конфигурацию, основываясь на предыдущем опыте и своих знаниях предметной области. Поскольку возникающая в данном случае задача структурной оптимизации является многокритериальной, она сложна для экспертного анализа. Эксперту сложно учесть все критерии оптимизации и ограничения задачи, а также проверить большое количество возможных конфигураций сооружений, необходимых при принятии решения. Поэтому интеллектуальные методы оптимизации (например, эволюционные алгоритмы) широко используются для решения данной задачи.

Описываемый подход к автоматизированному проектированию гидротехнических сооружений основан на интеллектуальных методах, использующих эволюционные алгоритмы для решения задач многокритериальной оптимизации. В рамках решения задачи применяется модифицированный алгоритм SPEA2. При выполнении экспериментальных расчетов, предназначенных для оценки эффективности дополнительных волнозащитных сооружений в рамках решения задачи структурной оптимизации, была использована конфигурация модели ветрового волнения SWAN для акватории порта (схема взаимодействия показана на рисунке 6.3.1) Данный подход позволяет идентифицировать Парето-

оптимальный набор возможных конфигураций, который может быть проанализирован лицами, принимающими решения, и использован для окончательного заключения по проектированию.

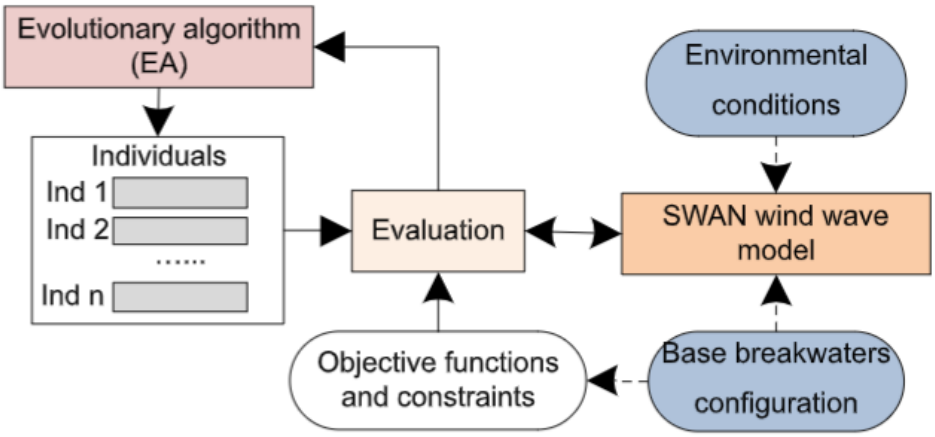


Рисунок 6.3.1 – Схема взаимодействия между эволюционным алгоритмом и моделью ветрового волнения в ходе оптимизации

Каждый индивид задается вектором вершин длины (x_i, y_i) длиной N, где N – это количество сегментов защитных сооружений, а x и y – их координаты. Для обеспечения лучшей сходимости и устойчивости найденных решений, в ходе оптимизации генотип был представлен в качестве набора пар «длина сегмента доп. сооружений – угол поворота сегмента относительно предыдущего» (вариант, изображенный слева на рисунке 6.3.2). Однако, возможно и представление вершин в декартовых координатах.

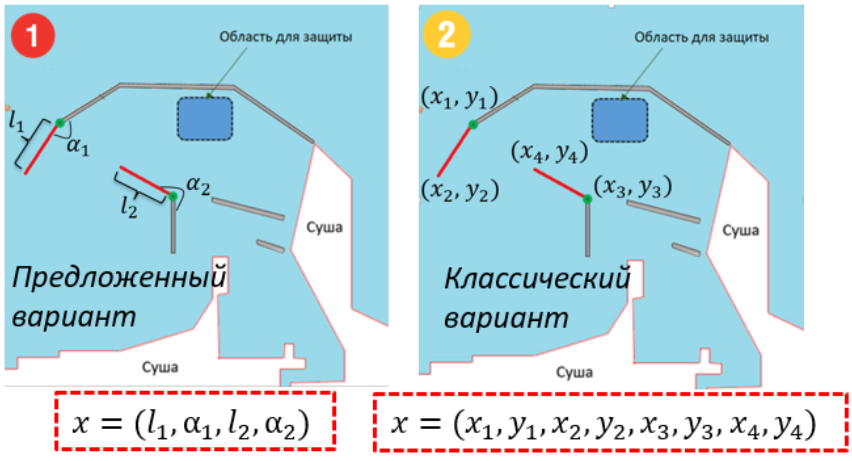


Рисунок 6.3.2 – Способы представления решения в генотипе эволюционного алгоритма

В общем виде многокритериальная задача оптимизации включает набор n переменных, m целевых функций и k ограничений (зависящих от этих переменных). Таким образом, при решении многокритериальной задачи необходимо найти оптимум по m критериям, а сама задача формально записывается следующим образом:

$$y = f(x) = (f_1(x), f_2(x), \dots, f_m(x)) \to opt \quad (1)$$

где $x = (x_1, x_2, \dots, x_n) \in X$ – вектор решений, удовлетворяющий k ограничениям, $g(x) = (g_1(x), g_2(x), \dots, g_k(x)) \geq 0, y = (y_1, y_2, \dots, y_m) \in Y$ – вектор целевых функций.

В процессе решения поставленной задачи оптимизации предполагается использовать многокритериальный эволюционный алгоритм, основанный на концепции доминирования по Парето. Для того, чтобы идентифицировать близкую к оптимальной структуру дополнительных волнозащитных сооружений требуется осуществить оптимизацию по нескольким критериям с учетом ряда ограничений, являющимися необходимыми в данной задаче. Для того, чтобы решить поставленные задачи, было сформировано три критерия и основанных на них целевых функции.

Для решения задачи улучшения волновой обстановки в контрольных точках вводится целевая функция $f_1^{obj}(hs)$, задающая значимую высоту волны в каждой целевой точке. В ходе оптимизации её значение минимизируется. Количество целевых функций такого вида будет равно количеству выбранных точек – в данной их задаче их три. Значение значимой высоты волны в точке рассчитывается с использованием конфигурации спектральной модели ветрового волнения SWAN.

Для того, чтобы перейти от целевых функций различной размерности к безразмерным значениям, было выполнено их нормирование на значения целевых функций для базовой конфигурации волнозащитных сооружений:

$$f_i^{obj\_norm}\left(x_i^{cand}\right) = \frac{f_i^{obj}\left(x_i^{cand}\right) - f_i^{obj}\left(x_i^{def}\right)}{f_i^{obj}\left(x_i^{def}\right)} * 100\ (\%), \quad (2)$$

где i – индекс целевой функции, x_i^{def} – её параметры для базовой конфигурации сооружений, x_i^{def} – её параметры для конфигурации-кандидата.

В работах в области оптимизации гидротехнических сооружений выделяют несколько классов ограничений, которые можно наложить на задачу: экологические, гидродинамические, структурные, навигационные. В рамках данного проекта в задачу оптимизации введено два структурных и навигационных ограничения. Первым из них является ограничение на самопересечения волнозащитных сооружений. Вторым ограничением является пересечение центральных линий фарватеров. Потенциальные конфигурации, нарушающие такие ограничения, отбрасываются и не рассматриваются далее.

Эволюционный оператор скрещивания (кроссовера) был реализован по схеме, согласно которой два выбранных родительских элемента обмениваются генетической информацией между частями хромосомы, созданными путем разделения хромосом в случайно сгенерированной точке. В этой реализации пространственные координаты, принадлежащие одному и тому же сегменту волнолома, не могут быть разделены точкой разрыва. Оператор мутации реализован как добавление или вычитание случайного значения (угла и направления) из параметров сегментов. Для угловых значений дополнительно отбрасывались решения, приводящие к возникновению слишком «резких» углов (>60 градусов)

между сегментами. Оператор мутации продемонстрирован на рисунке 6.3.3 а), а оператор скрещивания на рисунке 6.3.3 б).

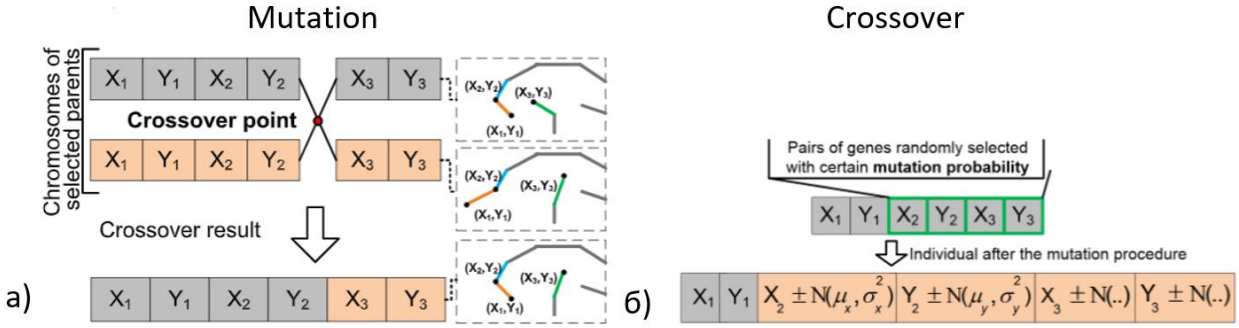


Рисунок 6.3.3 – Схемы генетических операторов: а) скрещивания, б) мутации

Результаты оптимизации и их сходимость показаны на рисунке 6.3.4. Черными линиями показаны различные варианты фронтов Парето, полученные в ходе нескольких независимых запусков.

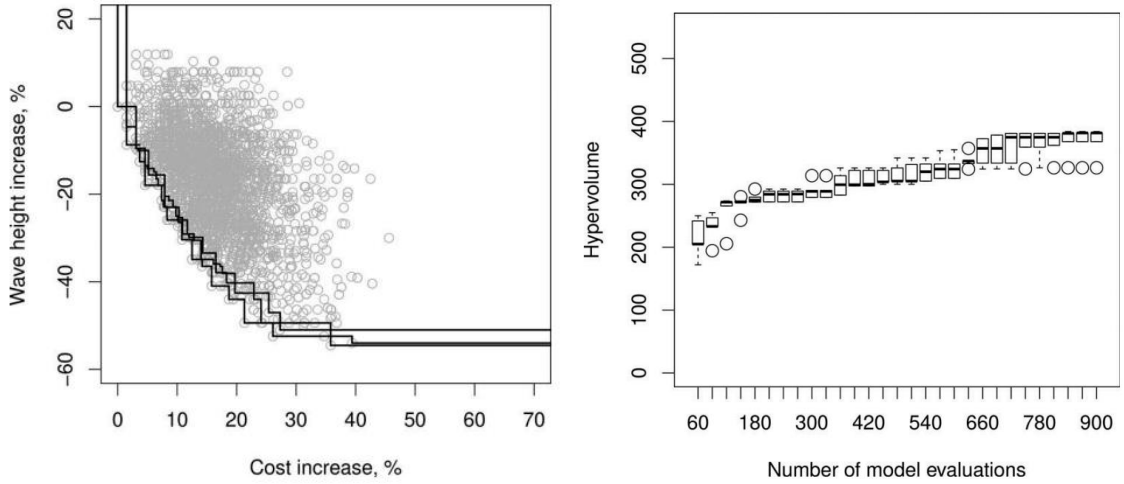


Рисунок 6.3.4 – а) Результаты оптимизации, представленные в виде фронтов Парето; б) Сходимость гиперобъема в ходе оптимизации.

6.4 Оптимизация структуры композитных моделей машинного обучения

В качестве примера решения задачи комбинаторной оптимизации будет использована задача автоматического создания композитных моделей (пайплайнов) машинного обучения. Примеры возможных структур таких пайплайнов показаны на рисунке 6.4.1.

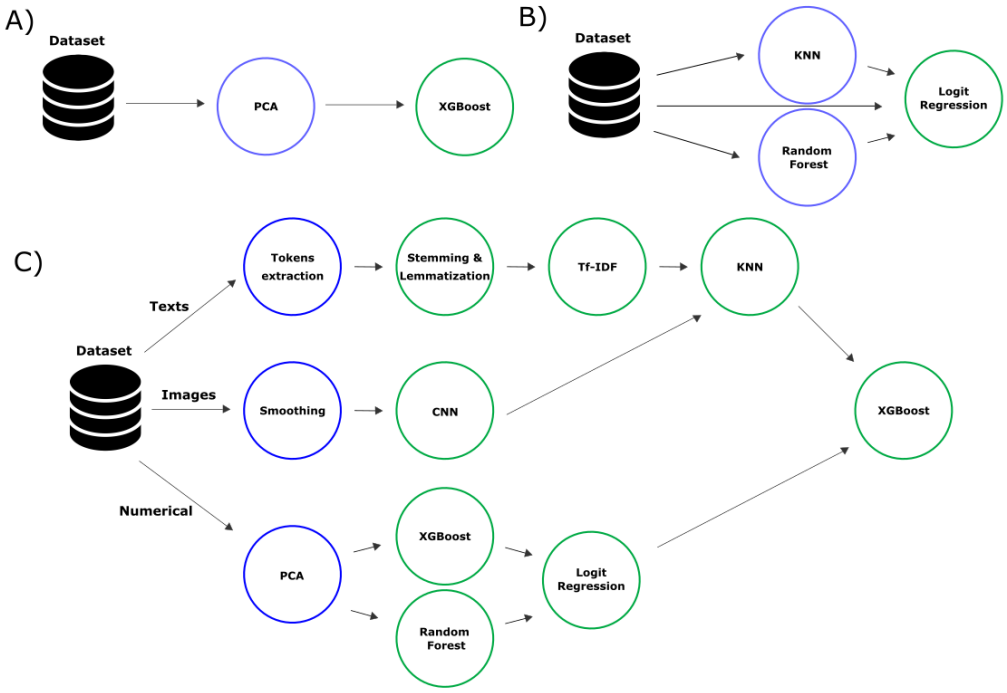


Рисунок 6.4.1 – Возможные структуры пайпалайнов машинного обучения: а) линейная; б) ансамблевая; в) мульти-модальная

Данная задача во многих случаях может быть сведена к решению задачи многокритериальной оптимизации, поскольку при составлении модели необходимо достичь оптимальных значений сразу по нескольким критериям (например, максимизировать качество моделирования при минимальной структурной сложности модели) и удовлетворить ограничениям (например, связанных со временем выполнения). Для решения такой задачи могут быть использованы различные подходы: основанные на случайном поиске [30], байесовской оптимизации [31], обучении с подкреплением [32], поиске по дереву Монте-Карло [33], градиентные подходы [34] и т. д. Однако разрабатываемые в рамках данного примера решения предполагается основывать на эволюционных подходах. Их концептуальным преимуществом является открытость [35], что позволяет «выращивать» модель с наиболее подходящей структурой вместо поиска подходящих решений в некотором ограниченном диапазоне структурных параметров. Эволюционные алгоритмы позволяют избегать преждевременной сходимости в локальном экстремуме в отличии от большинства распространенных оптимизационных алгоритмов. К тому же, не менее важным аспектом является возможность их эффективного распараллеливания.

В качестве ядра решения использован открытый фреймворк для автоматического машинного обучения FEDOT (https://github.com/nccr-itmo/FEDOT). Он позволяет строить пользовательские композитные модели для различных задач реального мира, используя алгоритм генетического программирования. Алгоритм не ограничен конкретным типом данных и может работать с табличными, текстовыми и графическими данными, а также с временными рядами.

Использованные в ходе решения задачи оптимизации эволюционные операторы показаны на рисунке 6.4.2.

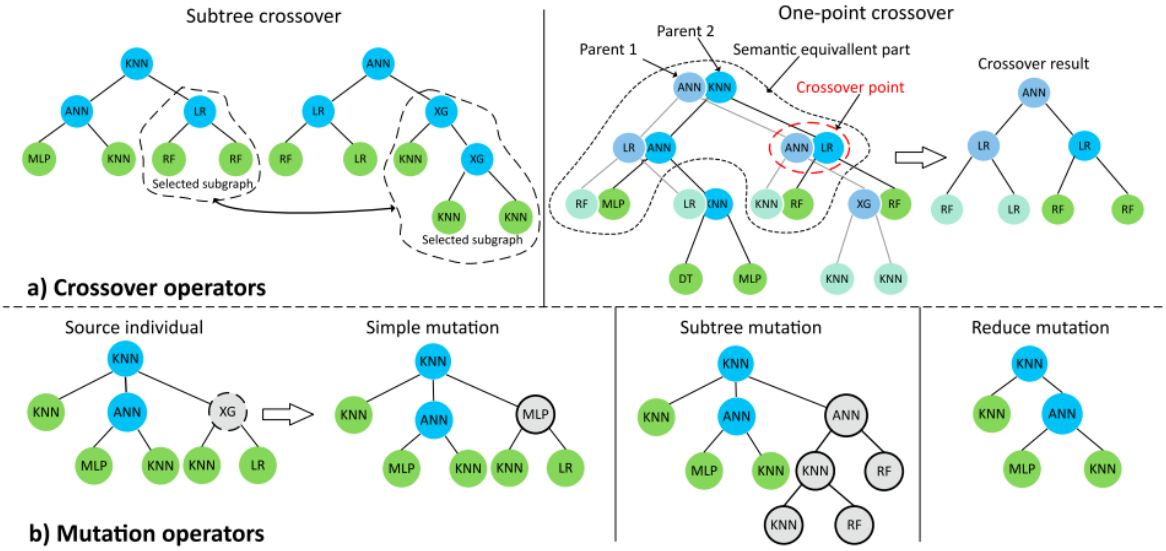


Рисунок 6.4.2 – Реализация операторов мутации и кроссовера для графовой структуры моделей

На рисунке 6.4.3. показан эффект, которые оказывает выбор реализации эволюционных операторов на сходимость целевой функции эволюционного алгоритма.

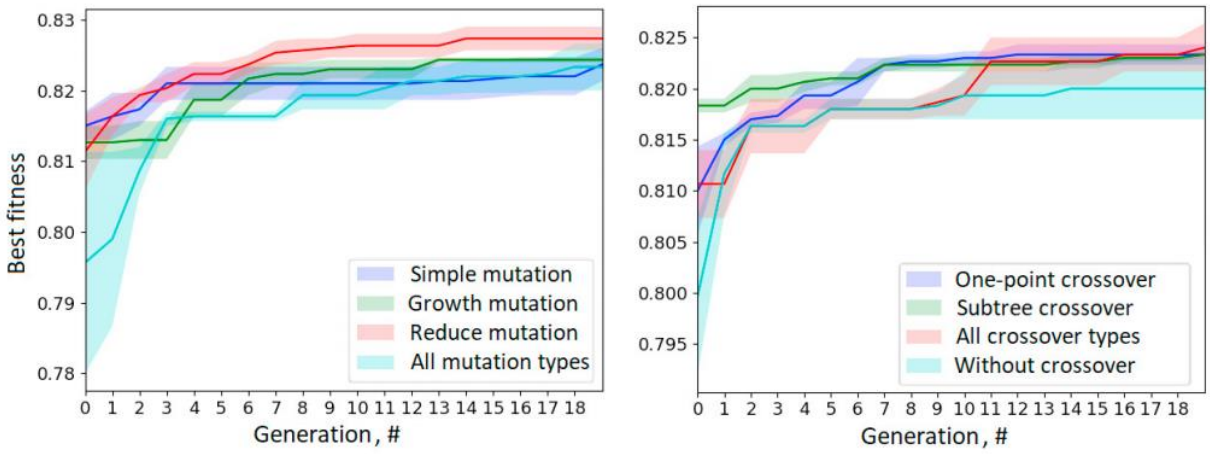


Рисунок 6.4.3 – Влияние типа оператора на сходимость алгоритма

Таким образом, применение эволюционной оптимизации позволяет эффективно решать задачу автоматического машинного обучения для сложных ансамблевых пайплайны, т.к. позволяет работать непосредственно с графовыми представлениями их структуры.

Вопросы для самопроверки:

1. Каким образом задается модель для символьной регрессии?
2. Какие начальные параметры модели задаются в задаче 6.1?

3. Как автоматизировать настройку численных моделей с точки зрения постановки задачи оптимизации?
4. Как автоматизировать проектирование физических сооружений с точки зрения постановки задачи оптимизации?
5. Как и какие применяются методы оптимизации в задаче автоматического машинного обучения?

ЗАКЛЮЧЕНИЕ

Из-за широты заявленной темы, в пособии мы рассмотрели лишь избранные методы однокритериальной оптимизации, которые так или иначе применяются для калибровки моделей, в частности при обучении моделей машинного обучения. Мы верим, что читатель, пользуясь нашим сжатым описанием сможет самостоятельно найти современные работы по необходимым ему методам оптимизации.

Отметим, что несмотря на засилье книг по машинному обучению, намеренно или случайно вводящих читателя в заблуждение о том, что кроме градиентных методов ничего не существует, нам удалось приоткрыть завесу неградиентных методов и показать их применение в решении современных исследовательских задач.

Примеры, приведённые в пособии, идут от простых к сложным, от решения одной небольшой задачи до итогового решения большой научной проблемы. Мы стремились показать, что не только и не столько градиентные методы являются авангардом современной науки, сколько совместное применение комбинаторных и градиентных методов для решения задач сначала глобальной, а потом локальной оптимизации.

Приведённые в пособии теоретические и практические примеры помогают найти ориентиры в разнообразии современных методов оптимизации. Читателю, которому изложение кажется слишком сжатым, мы можем посоветовать для начала обратиться к более фундаментальным учебникам [4,12,16], приведённым в списке литературы. В ином случае, если материал показался слишком лёгким, мы припасли статьи, которые также можно найти в этом списке.

Приложение А - Примерный список тем практических заданий

В рамках практического задания должны быть выполнены следующие шаги:

- Анализ преимуществ и недостатков выбранного алгоритма на основе проведенного сравнительного обзора аналогов (примерно 10 источников).

- Представление формулировки метода и решаемой задачи.

- Пример реализации на выбранных задачах (можно на тестовых функциях). Реализация должна продемонстрировать все или некоторые преимущества и недостатки выбранного алгоритма по сравнению с аналогами (не менее 2 аналогов).

Список алгоритмов оптимизации (не исчерпывающий), которые могут быть использованы в рамках практического задания:

1) Эволюционные алгоритмы: CMA-ES, NSGA-I-II-III (отличия), Ко-эволюционные алгоритмы, Алгоритмы генетического программирования и др.

2) Роевые алгоритмы: Голубиный алгоритм, Поиск косяком рыб, Алгоритм серых волков, Мультироевой роевой алгоритм (Multi swarm PSO) и др.

3) Градиентные алгоритмы: Алгоритм сопряженных градиентов, ADAM, SQN (stochastic quasi-Newton), SARAH (StochAstic Recusive gRadient algoritHm) и др.

4) Прочие алгоритмы: Алгоритм байесовской оптимизации, Гибридные алгоритмы (например, меметический алгоритм), Алгоритм гармонического поиска и др.

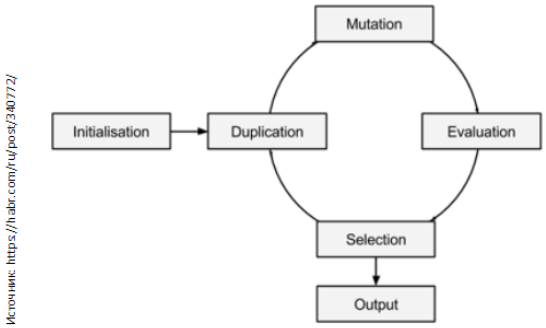
Приложение Б - Пример выполнения практической работы

Тема работы «Исследование свойств алгоритма CMA-ES»

Описание работы алгоритма:

CMA-ES - алгоритм, который может принимать результаты каждого поколения и адаптивно увеличивать или уменьшать пространство поиска для следующего поколения. Он вычисляет всю ковариационную матрицу пространства параметров.

Основные этапы работы алгоритма показаны в таблице:

Графическое представление	Основные этапы
 Источник: https://habr.com/ru/post/340772/	• **Initialization (инициализация).** Эволюция должна стартовать с начального решения. Выбор хорошей стартовой точки важен, потому что он может привести к совершенно различным результатам. • **Duplication (дупликация).** Создаётся множество копий текущего решения. • **Mutation (мутация).** Каждая копия случайным образом мутирует. Величина мутации критична, потому что она управляет скоростью выполнения эволюционного процесса. • **Evaluation (оценка).** Изменяется оценка генотипа, зависящая от показателей, продемонстрированных сгенерированной особью. • **Selection (отбор).** После оценки особей лучшим из них разрешается репликация, позволяющая стать основой следующего поколения. • **Output (вывод).** Эволюция — итеративный процесс, на любом этапе её можно остановить, чтобы получить улучшенную (или ту же самую) версию предыдущего поколения.

Достоинства алгоритма:

- Для изучения выборочного распределения используется только ранжирование возможных решений, и метод не требует ни производных, ни даже самих значений функций;
- Отсутствует тяготение к локальному оптимуму, как у простого генетического алгоритма и простейшей стратегии эволюции за счет регулирования пространства поиска;
- Эмпирически подтверждена успешность, в частности, на невыпуклых, плохо обусловленных, мультимодальных или зашумленных целевых функциях;
- Наиболее эффективная работа со сложными функциями или более крупными пространствами поиска.

Недостатки алгоритма:

- Производительность низка, если количество параметров модели, которые нам нужно решить, велико, поскольку расчет ковариации составляет O (N ^ 2). Алгоритм уместен при количестве параметров <1000;
- Для дифференцируемых функций метод будет уступать в производительности градиентным методам.

Фрагменты листинга исходного кода (на Python):

```
# Импорт зависимостей
      import time
      import warnings

      import matplotlib
      import matplotlib.pyplot as plt
      import numpy as np

      warnings.filterwarnings("ignore")
      matplotlib.rcParams.update({"font.size": 11})

      # Импорт библиотеки с реализацией Covariance Matrix Adaptation Evolution S
trategy

      import cma

      # решение задачи оптимизации на функции розенброка
      es = cma.CMAEvolutionStrategy(2 * [0], 0.5)
      results = []
      while not es.stop():
          solutions = es.ask()
          es.tell(solutions, [cma.ff.rosen(s) for s in solutions])
          es.disp()
          results.append(es.result_pretty()[1])
      values = np.arange(es.result_pretty()[4])

      print("_____________________________________________")
      print(es.result_pretty())
      plt.rcParams["figure.figsize"] = [5, 4]
      plt.plot(values, results)
      plt.title("CMA-ES Rosenbrock function")
      plt.xlabel("Iterations")
      plt.ylabel("Function value")
      plt.show()

      # исследование производительности алгоритма
      es = cma.CMAEvolutionStrategy(2 * [0], 0.5)
      func_values = []
      time_values = []
      for i in range(100):
          start_time = time.time()
          es.optimize(cma.ff.rastrigin)
          current_time = time.time() - start_time
          time_values.append(current_time)
          func_values.append(es.result_pretty()[1])

      # сравнение с базовым генетическим алгоритмом
      import numpy as np
      from geneticalgorithm import geneticalgorithm as ga

      varbound = np.array([[-5, 5]] * 2)
      params = {
          "max_num_iteration": 500,
```

```
    "population_size": 100,
    "mutation_probability": 0.1,
    "elit_ratio": 0.01,
    "crossover_probability": 0.5,
    "parents_portion": 0.3,
    "crossover_type": "uniform",
    "max_iteration_without_improv": 50,
}
model = ga(
    function=cma.ff.rastrigin,
    dimension=2,
    variable_type="real",
    variable_boundaries=varbound,
    algorithm_parameters=params,
)

func_values = []
time_values = []
for i in range(50):
    start_time = time.time()
    model.run()
    current_time = time.time() - start_time
    time_values.append(current_time)
    func_values.append(model.output_dict["function"])
```

СПИСОК ЛИТЕРАТУРЫ

1. Wainwright, J., Mulligan, M.: Environmental modelling: finding simplicity in complexity. John Wiley & Sons (2005).
2. Стронгин Р. Г., Гергель В. П., Баркалов К. А. Параллельные методы решения задач глобальной оптимизации // Изв. вузов. Приборостроение. – 2009. – Т. 52. – № 10.
3. Bengio Y. Gradient-based optimization of hyperparameters //Neural computation. – 2000. – Т. 12. – №. 8. – С. 1889-1900.
4. Поляк Б. Т. Введение в оптимизацию. – Наука. Гл. ред. физ.-мат. лит., 1983.
5. Sumata H. et al. A comparison between gradient descent and stochastic approaches for parameter optimization of a sea ice model //Ocean Science. – 2013. – Т. 9. – №. 4. – С. 609-630.
6. Barati R. Parameter estimation of nonlinear Muskingum models using Nelder-Mead simplex algorithm //Journal of Hydrologic Engineering. – 2011. – Т. 16. – №. 11. – С. 946-954.
7. Bergstra J., Bengio Y. Random search for hyper-parameter optimization //Journal of Machine Learning Research. – 2012. – Т. 13. – №. Feb. – С. 281-305.
8. Klepper O., Hendrix E. M. T. A method for robust calibration of ecological models under different types of uncertainty //Ecological Modelling. – 1994. – Т. 74. – №. 3-4. – С. 161-182.
9. Laurenceau J., Sagaut P. Building efficient response surfaces of aerodynamic functions with kriging and cokriging //AIAA journal. – 2008. – Т. 46. – №. 2. – С. 498-507.
10. Karagiannis G., Konomi B. A., Lin G. On the Bayesian calibration of expensive computer models with input dependent parameters //Spatial Statistics. – 2017.
11. Neidinger R. D. Introduction to automatic differentiation and MATLAB object-oriented programming //SIAM review. – 2010. – Т. 52. – №. 3. – С. 545-563.
12. Калиткин Н. Н. Численные методы. 2 изд. – БХВ-Петербург, 2011.
13. He K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification //Proceedings of the IEEE international conference on computer vision. – 2015. – С. 1026-1034.
14. Kingma D. P., Ba J. Adam: A method for stochastic optimization //arXiv preprint arXiv:1412.6980. – 2014.
15. Pham D., Karaboga D. Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks. – Springer Science & Business Media, 2012.
16. Карпенко А. П. Современные алгоритмы поисковой оптимизации. Алгоритмы, вдохновленные природой //М.: Изд-во МГТУ им. НЭ Баумана. – 2014. – Т. 448. – С. 11.
17. Grefenstette J. J. Parallel adaptive algorithms for function optimization //Vanderbilt University, Nashville, TN, Tech. Rep. CS-81-19. – 1981.

18. Nguyen D. C. H. et al. Framework for computationally efficient optimal crop and water allocation using ant colony optimization //Environmental Modelling & Software. – 2016. – T. 76. – C. 37-53.

19. Schmitt, C., Rey-Coyrehourcq, S., Reuillon, R., Pumain, D.: Half a billion simulations: Evolutionary algorithms and distributed computing for calibrating the simpoplocal geographical model. Environment and Planning B: Planning and Design 42(2), 300–31.

20. Zitzler E., Laumanns M., Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm //TIK-report. – 2001. – T. 103.

21. Kovalchuk S. V. et al. A conceptual approach to complex model management with generalized modelling patterns and evolutionary identification //Complexity. – 2018. – T. 2018.

22. Maier H. R. et al. Introductory overview: Optimization using evolutionary algorithms and other metaheuristics //Environmental modelling & software. – 2018.

23. Yang Liu and Soon-Thiam Khu. 2007. Automatic calibration of numerical models using fast optimisation by fitness approximation. In Neural Networks, 2007. IJCNN 2007. International Joint Conference on. IEEE, IEEE, 1073–1078.

24. Simpson T. et al. Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come-or not //12th AIAA/ISSMO multidisciplinary analysis and optimization conference. – 2008. – C. 5802.

25. Jin Y. Surrogate-assisted evolutionary computation: Recent advances and future challenges //Swarm and Evolutionary Computation. – 2011. – T. 1. – № 2. – C. 61-70. Elsevier, 2011. Vol. 1, № 2. P. 61–70.

26. Bo Liu, Slawomir Koziel, and Qingfu Zhang. 2016. A multi-fidelity surrogatemodel-assisted evolutionary algorithm for computationally expensive optimization problems. Journal of computational science 12 (2016), 28–37.

27. Yiming Zhang, Nam Kim, Chanyoung Park, and Raphael Haftka. 2018. Multifidelity Surrogate Based on Single Linear Regression. AIAA Journal 56, 12 (2018), 4944–4952.

28. Fernando Lobo, Cláudio F Lima, and Zbigniew Michalewicz. 2007. Parameter setting in evolutionary algorithms. Vol. 54. Springer Science & Business Media.

29. Agoston Eiben and Selmar Smit. 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1, 1 (2011), 19–31.

30. LeDell E., Poirier S. H2o automl: Scalable automatic machine learning //7th ICML workshop on automated machine learning. – 2020.

31. Feurer M. et al. Auto-sklearn: efficient and robust automated machine learning //Automated Machine Learning. – Springer, Cham, 2019. – C. 113-134.

32. Heffetz Y. et al. DeepLine: AutoML Tool for Pipelines Generation using Deep Reinforcement Learning and Hierarchical Actions Filtering //Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data

Mining. – 2020. – C. 2103-211.
33. Rakotoarison H., Schoenauer M., Sebag M. Automated machine learning with monte-carlo tree search //arXiv preprint arXiv:1906.00170. – 2019.
34. Liu H., Simonyan K., Yang Y. Darts: Differentiable architecture search //arXiv preprint arXiv:1806.09055. – 2018.
35. Packard N. et al. Open-ended evolution and open-endedness: Editorial introduction to the open-ended evolution i special issue. – 2019.

Александр Александрович Хватов
Николай Олегович Никитин
Анна Владимировна Калюжная

СОВРЕМЕННЫЕ МЕТОДЫ ОПТИМИЗАЦИИ С ПРИМЕРАМИ НА PYTHON

Учебно-методическое пособие

В авторской редакции

Редакционно-издательский отдел Университета ИТМО
Зав. РИО
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49, литер А