УДК 004.421
ББК 32.811
В52

Вирсански Э.
В52 Генетические алгоритмы на Python / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 286 с.: ил.

ISBN 978-5-97060-857-9

Там, где традиционные алгоритмы бесполезны или не дают результата за обозримое время, на помощь могут прийти генетические алгоритмы. Они позволяют решить целый комплекс сложных задач, в том числе связанных с искусственным интеллектом, упростить оптимизацию непрерывных функций, выполнять реконструкцию изображений и многое другое.

Книга поможет программистам, специалистам по обработке данных и энтузиастам ИИ, интересующимся генетическими алгоритмами, подступиться к стоящим перед ними задачам, связанным с обучением, поиском и оптимизацией, а также повысить качество и точность результатов в уже имеющихся приложениях.

Для изучения материала книги требуются владение языком Python на рабочем уровне и базовые знания математики и информатики.

УДК 004.421
ББК 32.811

ISBN 978-1-83855-774-4 (англ.)
ISBN 978-5-97060-857-9 (рус.)

Содержание

Об авторе 13

О рецензенте 14

Предисловие 15

Часть I. ОСНОВЫ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ 19

Глава 1. Введение в генетические алгоритмы 20

Что такое генетические алгоритмы? 20
Дарвиновская эволюция 21
Аналогия с генетическими алгоритмами 21
Генотип 22
Популяция 22
Функция приспособленности 23
Отбор 23
Скрещивание 23
Мутация 24
Теоретические основы генетических алгоритмов 24
Теорема о схемах 25
Отличия от традиционных алгоритмов 26
Популяция как основа алгоритма 27
Генетическое представление 27
Функция приспособленности 27
Вероятностное поведение 27
Преимущества генетических алгоритмов 28
Глобальная оптимизация 28
Применимость к сложным задачам 29
Применимость к задачам, не имеющим математического представления 30
Устойчивость к шуму 30
Распараллеливание 30
Непрерывное обучение 31
Ограничения генетических алгоритмов 31
Специальные определения 31
Настройка гиперпараметров 31
Большой объем счетных операций 32
Преждевременная сходимость 32
Отсутствие гарантированного решения 32
Сценарии применения генетических алгоритмов 32
Резюме 33
Для дальнейшего чтения 33

Глава 2. Основные компоненты генетических алгоритмов 34
Базовая структура генетического алгоритма 34
Создание начальной популяции 36
Вычисление приспособленности 36
Применение операторов отбора, скрещивания и мутации 36
Проверка условий остановки 37
Методы отбора 37
Правило рулетки 38
Стохастическая универсальная выборка 39
Ранжированный отбор 40
Масштабирование приспособленности 41
Турнирный отбор 42
Методы скрещивания 43
Одноточечное скрещивание 43
Двухточечное и ***k***-точечное скрещивание 44
Равномерное скрещивание 45
Скрещивание для упорядоченных списков 45
Упорядоченное скрещивание 46
Методы мутации 47
Инвертирование бита 48
Мутация обменом 48
Мутация обращением 48
Мутация перетасовкой 49
Генетические алгоритмы с вещественным кодированием 49
Скрещивание смешением 50
Имитация двоичного скрещивания 51
Вещественная мутация 53
Элитизм 53
Образование ниш и разделение 54
Последовательное и параллельное образование ниш 56
Искусство решения задач с помощью генетических алгоритмов 57
Резюме 58
Для дальнейшего чтения 58

Часть II. РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ 59

Глава 3. Каркас DEAP 60
Технические требования 60
Введение в DEAP 61
Использование модуля creator 62
Создание класса Fitness 62
Определение стратегии приспособления 62
Хранение значения приспособленности 63
Создание класса Individual 64
Использование класса Toolbox 64

Создание генетических операторов....65
Создание популяции....66
Вычисление приспособленности....66
Задача OneMax....67
Решение задачи OneMax с помощью DEAP....67
Выбор хромосомы....67
Вычисление приспособленности....68
Выбор генетических операторов....68
Задание условия остановки....68
Реализация средствами DEAP....69
Подготовка....69
Эволюция решения....72
Выполнение программы....75
Использование встроенных алгоритмов....76
Объект Statistics....77
Алгоритм....77
Объект logbook....77
Выполнение программы....78
Зал славы....79
Эксперименты с параметрами алгоритма....81
Размер популяции и количество поколений....81
Оператор скрещивания....83
Оператор мутации....84
Оператор отбора....87
Размер турнира и его связь с вероятностью мутации....88
Отбор по правилу рулетки....92
Резюме....93
Для дальнейшего чтения....94

Глава 4. Комбинаторная оптимизация....95

Технические требования....95
Поисковые задачи и комбинаторная оптимизация....96
Решение задачи о рюкзаке....96
Задача о рюкзаке 0-1 с сайта Rosetta Code....97
Представление решения....98
Представление задачи на Python....98
Решение с помощью генетического алгоритма....99
Решение задачи коммивояжера....102
Файлы эталонных данных TSPLIB....103
Представление решения....104
Представление задачи на Python....104
Решение с помощью генетического алгоритма....106
Улучшение результатов благодаря дополнительному исследованию и элитизму....109
Решение задачи о маршрутизации транспорта....114
Представление решения....115
Представление задачи на Python....116

Решение с помощью генетического алгоритма 118
Резюме 122
Для дальнейшего чтения 123

Глава 5. Задачи с ограничениями 124

Технические требования 124
Соблюдение ограничений в поисковых задачах 125
Решение задачи об *N* ферзях 125
Представление решения 126
Представление задачи на Python 128
Решение с помощью генетического алгоритма 129
Решение задачи о составлении графика дежурств медсестер 133
Представление решения 133
Жесткие и мягкие ограничения 134
Представление задачи на Python 135
Решение на основе генетического алгоритма 137
Решение задачи о раскраске графа 140
Представление решения 142
Жесткие и мягкие ограничения в задаче о раскраске графа 143
Представление задачи на Python 143
Решение с помощью генетического алгоритма 145
Резюме 149
Для дальнейшего чтения 150

Глава 6. Оптимизация непрерывных функций 151

Технические требования 151
Хромосомы и генетические операторы для задач с вещественными числами 152
Использование DEAP совместно с непрерывными функциями 153
Оптимизация функции Eggholder 154
Оптимизация функции Eggholder с помощью генетического алгоритма 155
Повышение скорости сходимости посредством увеличения частоты мутаций 158
Оптимизация функции Химмельблау 159
Оптимизация функции Химмельблау с помощью генетического алгоритма 161
Использование ниш и разделения для отыскания нескольких решений 165
Функция Симионеску и условная оптимизация 168
Условная оптимизация с помощью генетического алгоритма 170
Оптимизация функции Симионеску с помощью генетического алгоритма 171
Использование ограничений для нахождения нескольких решений 172
Резюме 173
Для дальнейшего чтения 173

Часть III. ПРИЛОЖЕНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ В ИСКУССТВЕННОМ ИНТЕЛЛЕКТЕ 174

Глава 7. Дополнение моделей машинного обучения методами выделения признаков 175

Технические требования 176
Машинное обучение с учителем 176
Классификация 177
Регрессия 179
Алгоритмы обучения с учителем 180
Выделение признаков в обучении с учителем 180
Выделение признаков для задачи регрессии Фридмана-1 181
Представление решения 182
Представление решения на Python 182
Решение с помощью генетического алгоритма 184
Выделение признаков для классификации набора данных Zoo 186
Представление задачи на Python 187
Решение с помощью генетического алгоритма 189
Резюме 191
Для дальнейшего чтения 191

Глава 8. Настройка гиперпараметров моделей машинного обучения 192

Технические требования 193
Гиперпараметры в машинном обучении 193
Настройка гиперпараметров 194
Набор данных Wine 195
Классификатор на основе адаптивного усиления 195
Настройка гиперпараметров с помощью генетического поиска на сетке 196
Тестирование качества классификатора с параметрами по умолчанию 198
Результаты традиционного поиска на сетке 198
Результаты генетического поиска на сетке 198
Прямой генетический подход к настройке гиперпараметров 199
Представление гиперпараметров 200
Оценка верности классификатора 200
Настройка гиперпараметров с помощью генетического алгоритма 201
Резюме 204
Для дальнейшего чтения 204

Глава 9. Оптимизация архитектуры сетей глубокого обучения 205

Технические требования 205
Искусственные нейронные сети и глубокое обучение 206
Многослойный перцептрон 207
Глубокое обучение и сверточные нейронные сети 208

Оптимизация архитектуры классификатора на основе глубокой сети 209
Набор данных Iris 209
Представление конфигурации скрытого слоя 210
Оценка верности классификатора 211
Оптимизация архитектуры МСП с помощью генетического алгоритма 212
Объединение оптимизации архитектуры с настройкой гиперпараметров 215
Представление решения 215
Вычисление верности классификатора 216
Оптимизация объединенной конфигурации МСП с помощью генетического алгоритма 217
Резюме 218
Для дальнейшего чтения 218

Глава 10. Генетические алгоритмы и обучение с подкреплением 219

Технические требования 219
Обучение с подкреплением 220
Генетические алгоритмы и обучение с подкреплением 221
OpenAI Gym 221
Интерфейс env 222
Решение окружающей среды MountainCar 223
Представление решения 225
Оценивание решения 225
Представление задачи на Python 226
Решение с помощью генетического алгоритма 226
Решение окружающей среды CartPole 229
Управление средой CartPole с помощью нейронной сети 230
Представление и оценивание решения 231
Представление задачи на Python 232
Решение с помощью генетического алгоритма 233
Резюме 236
Для дальнейшего чтения 237

Часть IV. РОДСТВЕННЫЕ ТЕХНОЛОГИИ 238

Глава 11. Генетическая реконструкция изображений 239

Технические требования 239
Реконструкция изображений из многоугольников 240
Обработка изображений на Python 240
Библиотеки обработки изображений на Python 240
Библиотека Pillow 241
Библиотека scikit-image 241
Библиотека opencv-python 241
Рисование с помощью многоугольников 242

Измерение степени различия двух изображений ... 243
Попиксельная среднеквадратическая ошибка ... 243
Структурное сходство (SSIM) ... 244
Применение генетических алгоритмов для реконструкции изображений ... 244
Представление и оценивание решения ... 245
Представление задачи на Python ... 246
Реализация генетического алгоритма ... 246
Добавление функции обратного вызова в код генетического алгоритма ... 249
Результаты реконструкции изображения ... 250
Применение попиксельной среднеквадратической ошибки ... 251
Применение индекса структурного сходства ... 253
Другие эксперименты ... 256
Резюме ... 257
Для дальнейшего чтения ... 258

Глава 12. Другие эволюционные и бионические методы вычислений ... 259

Технические требования ... 259
Эволюционные и бионические вычисления ... 260
Генетическое программирование ... 260
Пример генетического программирования – контроль по четности ... 262
Реализация с помощью генетического программирования ... 264
Упрощение решения ... 269
Оптимизация методом роя частиц ... 271
Пример применения PSO – оптимизация функции ... 272
Реализация оптимизации методом роя частиц ... 273
Другие родственные методы ... 277
Эволюционные стратегии ... 277
Дифференциальная эволюция ... 278
Муравьиный алгоритм оптимизации ... 278
Искусственные иммунные системы ... 278
Искусственная жизнь ... 279
Резюме ... 279
Для дальнейшего чтения ... 280

Предметный указатель ... 281

Об авторе

Эйял Вирсански – старший инженер-программист, лидер технического сообщества, исследователь и энтузиаст искусственного интеллекта. Начал карьеру как один из первопроходцев в области передачи голоса по IP-сетям и теперь может похвастаться более чем 20-летним опытом создания самых разных высокопроизводительных корпоративных решений. Еще будучи студентом, проявил интерес к генетическим алгоритмам и нейронным сетям. Одним из результатов его исследований стал новый алгоритм обучения с учителем, объединяющий достоинства обоих подходов.

Эйял возглавляет группу пользователей Java в Джексонвилле, а также виртуальную группу пользователей по теме «Искусственный интеллект в корпоративных приложениях» и ведет блог по искусственному интеллекту ai4java, ориентированный на разработчиков.

Я хочу поблагодарить свою семью и близких друзей за терпение, поддержку и ободрение на протяжении длительного процесса написания книги. Отдельное спасибо группе пользователей Python в Джексонвилле (PyJax) за отзывы и поддержку.

О рецензенте

Лайза Бэнг окончила бакалавриат по биологии моря в Калифорнийском университете в Санта-Крус и магистратуру по биоинформатике в Сеульском университете Сонсиль под руководством д-ра Кван Хви Чо. Магистерская диссертация была посвящена методу создания воспроизводимых моделей QSAR (поиск количественных соотношений структура–свойство) с применением блокнота Jupyter, одним из компонентов которого был генетический алгоритм, позволяющий уменьшить пространство поиска. В настоящее время ведется работа по его переводу на каркас DEAP-VS для совместимости с Python 3. Она также работала в системе здравоохранения Гейсингера, входящей в состав Института биомедицинской и трансляционной информатики, где применяла данные следующего поколения о секвенировании и электронные медицинские карты пациентов для анализа исходов раковых и других заболеваний. Сейчас работает в компании Ultragenyx Pharmaceutical, где занимается доклиническими испытаниями и использует методы биоинформатики и хемоинформатики для анализа редких наследственных болезней.

Спасибо моей семье, учителям и наставникам!

Предисловие

Берущие начало в эволюционной теории Дарвина, *генетические алгоритмы* являются одним из самых удивительных методов решения задач поиска, оптимизации и обучения. Они могут привести к успеху там, где традиционные алгоритмы не способны дать адекватные результаты за приемлемое время.

Данная книга покажет вам путь к овладению этим чрезвычайно мощным и вместе с тем простым подходом к разнообразным задачам, венцом которых станут приложения ИИ.

Вы узнаете, как работают генетические алгоритмы и когда имеет смысл их использовать. А заодно получите практический опыт их применения в разных областях с помощью популярного языка программирования Python.

Предполагаемая аудитория

Эта книга призвана помочь программистам, специалистам по обработке данных и энтузиастам ИИ, интересующимся генетическими алгоритмами, подступиться к стоящим перед ними задачам, связанным с обучением, поиском и оптимизацией, а также повысить качество и точность результатов в уже имеющихся приложениях.

Книга адресована также всем, кто сталкивался в своей практике с трудными задачами, в которых традиционные алгоритмы вообще бесполезны или не дают результата за обозримое время. Показано, как использовать генетические алгоритмы в качестве эффективного и в то же время простого подхода к решению сложных задач.

Структура книги

В главе 1 «Введение в генетические алгоритмы» приводится краткое введение в теорию и принципы работы генетических алгоритмов. Рассматриваются также различия между генетическими алгоритмами и традиционными методами, и описываются сценарии, в которых имеет смысл применять генетические алгоритмы.

В главе 2 «Основные компоненты генетических алгоритмов» более глубоко описываются основные компоненты и детали реализации генетических алгоритмов. Познакомившись с потоком управления, вы затем узнаете о различных компонентах и их реализациях.

В главе 3 «Каркас DEAP» дается введение в DEAP – мощный и гибкий каркас эволюционных вычислений, позволяющий решать практические задачи с помощью генетических алгоритмов. Мы продемонстрируем его использо-

вание на примере программы на Python, которая решает задачу OneMax – что-то вроде «Hello World» в мире генетических алгоритмов.

В главе 4 «Комбинаторная оптимизация» рассматриваются такие проблемы, как упаковка рюкзака, задача коммивояжера и задача маршрутизации транспорта, а также программы на Python для их решения с помощью генетических алгоритмов и каркаса DEAP.

Глава 5 «Задачи с ограничениями» представляет собой введение в задачи с ограничениями, например: задача об *N* ферзях, составление графика дежурства медсестер, задача о раскраске графа. Объясняется, как решить их на Python с помощью генетических алгоритмов и каркаса DEAP.

В главе 6 «Оптимизация непрерывных функций» обсуждаются задачи непрерывной оптимизации и их решения с помощью генетических алгоритмов. В качестве примеров приводятся функция Eggholder, функции Химмельблау и Симионеску. Попутно мы изучим концепции образования ниш, разделения и обработки ограничений.

В главе 7 «Дополнение моделей машинного обучения методами выделения признаков» речь пойдет о моделях машинного обучения с учителем. Объясняется, как генетические алгоритмы позволяют повысить качество этих моделей за счет выделения наилучшего подмножества признаков из представленных входных данных.

В главе 8 «Настройка гиперпараметров моделей машинного обучения» показано, как использовать генетические алгоритмы для улучшения качества моделей машинного обучения с учителем путем применения поиска на сетке, управляемого генетическим алгоритмом, или прямого генетического поиска.

В главе 9 «Оптимизация архитектуры сетей глубокого обучения» рассматриваются искусственные нейронные сети и описывается, как генетические алгоритмы позволяют улучшить модель, основанную на нейронной сети, путем оптимизации архитектуры последней. Вы узнаете, как сочетать оптимизацию архитектуры сети с настройкой гиперпараметров.

Глава 10 «Генетические алгоритмы и обучение с подкреплением» посвящена обучению с подкреплением. Применение генетических алгоритмов иллюстрируется на примере двух тестовых окружающих сред – MountainCar (машина на горе) и CartPole (балансировка стержня) – из библиотеки OpenAI Gym.

В главе 11 «Генетическая реконструкция изображений» описываются эксперименты по реконструкции одного хорошо известного изображения с помощью множества полупрозрачных многоугольников на основе генетических алгоритмов. Попутно вы приобретете полезный опыт обработки изображений и научитесь работать с соответствующими библиотеками на Python.

Глава 12 «Другие эволюционные и бионические методы вычислений» расширяет горизонты и знакомит еще с несколькими пришедшими из биологии приемами решения задач. Два из них – генетическое программирование и метод роя частиц – будут реализованы с помощью Python и каркаса DEAP.

Что необходимо для чтения этой книги

Чтобы получить максимум пользы от чтения книги, необходимо владеть языком Python на рабочем уровне и иметь базовые знания по математике и информатике. Знакомство с фундаментальными понятиями машинного обучения желательно, но не обязательно, поскольку в книге приводится краткое изложение всего необходимого.

Для выполнения приведенных в книге примеров понадобится версия 3.7 или более поздняя, а также несколько Python-пакетов (все они здесь же и описываются). Рекомендуется, хотя это и необязательно, установить какую-нибудь интегрированную среду разработки на Python (IDE), например PyCharm или Visual Studio Code.

Скачивание исходного кода

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com на странице с описанием соответствующей книги.

Обозначения и графические выделения

В этой книге применяется ряд соглашений о наборе текста.

`CodeInText`: код в тексте, имена таблиц базы данных, папок и файлов, расширения имен файлов, пути к файлам, URL-адреса, данные, вводимые пользователем, и адреса в Твиттере. Например: «Метод класса `__init__()` создает набор данных».

Отдельно стоящие фрагменты кода набраны так:

```
self.X, self.y = datasets.make_friedman1(n_samples=self.numSamples,
                                          n_features=self.numFeatures,
                                          noise=self.NOISE,
                                          random_state=self.randomSeed)
```

Желая привлечь внимание к какой-то части кода, мы выделяем ее полужирным шрифтом, например:

```
self.regressor = GradientBoostingRegressor(random_state=self.randomSeed)
```

Команды операционной системы и результаты их работы выделяются так:

```
pip install deap
```

Полужирный: новые термины и важные слова, а также части пользовательского интерфейса. Так выделяются команды меню и текст в диалоговых окнах, например: «Выберите команду **System info** на панели **Administration**».

Предупреждения и важные замечания выглядят так.


Советы и рекомендации выглядят так.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге, – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте http://dmkpress.com/authors/publish_book/ или напишите в издательство: dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt Publishing очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

ЧАСТЬ I

ОСНОВЫ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

В этой части мы познакомимся с основными концепциями генетических алгоритмов и способами их применения

Глава 1

Введение в генетические алгоритмы

Позаимствовавший идею у теории эволюции Чарльза Дарвина, один из самых удивительных методов решения задач заслуженно получил название «**эволюционные вычисления**». Самыми известными и широко распространенными представителями этого семейства являются **генетические алгоритмы**. В этой главе мы начнем путешествие в мир этих чрезвычайно мощных и вместе с тем очень простых методов.

Мы познакомимся с **генетическими алгоритмами**, проведем аналогию между ними и дарвиновской эволюцией и опишем как теорию, так и базовые принципы их работы. Затем поговорим о различиях между генетическими и традиционными алгоритмами, раскроем их преимущества и ограничения, опишем области применения. И завершим примерами задач, в которых генетические алгоритмы могут оказаться полезными.

В этой вводной главе рассматриваются следующие вопросы:

- что такое генетические алгоритмы;
- теоретические основы генетических алгоритмов;
- различия между генетическими и традиционными алгоритмами;
- преимущества и ограничения генетических алгоритмов;
- когда имеет смысл использовать генетические алгоритмы.

Что такое генетические алгоритмы?

Генетические алгоритмы – это семейство поисковых алгоритмов, идеи которых подсказаны принципами эволюции в природе. Имитируя процессы естественного отбора и воспроизводства, генетические алгоритмы могут находить высококачественные решения задач, включающих поиск, оптимизацию и обучение. В то же время аналогия с естественным отбором позволяет этим алгоритмам преодолевать некоторые препятствия, встающие на пути тради-

ционных алгоритмов поиска и оптимизации, особенно в задачах с большим числом параметров и сложными математическими представлениями.

В этой части книги мы рассмотрим основные идеи генетических алгоритмов и их аналогию с эволюционными процессами в природе.

Дарвиновская эволюция

Генетические алгоритмы реализуют упрощенный вариант дарвиновской эволюции. Ниже перечислены принципы эволюционной теории Дарвина.

- **Изменчивость.** Признаки (атрибуты) отдельных особей, входящих в состав популяции, могут изменяться. Поэтому особи отличаются друг от друга, например по внешнему виду или поведению.
- **Наследственность.** Некоторые свойства устойчиво передаются от особи к ее потомкам. Поэтому потомки похожи на своих родителей больше, чем на других особей, не связанных с ними родством.
- **Естественный отбор.** Обычно популяции борются за ресурсы, имеющиеся в окружающей их среде. Особи, обладающие свойствами, лучше приспособленными к окружающей среде, более успешны в борьбе за выживание и привносят больше потомков в следующее поколение.

Иными словами, эволюция сохраняет популяцию особей, отличающихся друг от друга. Те, кто лучше приспособлен к окружающей среде, имеют больше шансов на выживание, размножение и передачу своих признаков следующему поколению. Так популяция от поколения к поколению становится все более приспособленной к окружающей среде и встающим на ее пути трудностям.

Важным механизмом эволюции является **скрещивание**, или **рекомбинация**, – когда потомок приобретает комбинацию признаков своих родителей. Скрещивание помогает поддерживать разнообразие популяции и со временем закреплять лучшие признаки. Кроме того, важную роль в эволюции играют **мутации** – случайные вариации признаков, – поскольку они вносят изменения, благодаря которым популяция время от времени совершает скачок в развитии.

Аналогия с генетическими алгоритмами

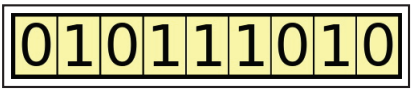
Цель генетических алгоритмов – найти оптимальное решение некоторой задачи. Если дарвиновская эволюция развивает популяцию отдельных особей, то генетические алгоритмы развивают популяцию потенциальных решений данной задачи, называемых **индивидуумами**. Эти решения итеративно оцениваются и используются для создания нового поколения решений. Те, что лучше проявили себя при решении задачи, имеют больше шансов пройти отбор и передать свои качества следующему поколению. Так постепенно потенциальные решения совершенствуются в решении поставленной задачи.

В следующих разделах описываются различные компоненты генетических алгоритмов, поддерживающие эту аналогию с дарвиновской эволюцией.

Генотип

В природе скрещивание, воспроизводство и мутация реализуются посредством **генотипа** – набора генов, сгруппированных в хромосомы. Когда две особи скрещиваются и производят потомство, каждая хромосома потомка несет комбинацию генов родителей.

В случае генетических алгоритмов каждому индивидууму соответствует хромосома, представляющая набор генов. Например, хромосому можно представить двоичной строкой, в которой каждый бит соответствует одному **гену**:

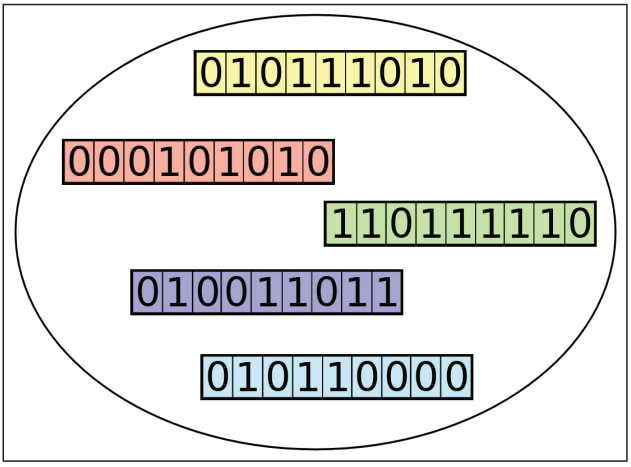


Простое двоичное кодирование хромосомы

На рисунке выше показан пример такой двоично-кодированной хромосомы, представляющей одного индивидуума.

Популяция

В любой момент времени генетический алгоритм хранит популяцию **индивидуумов** – набор потенциальных решений поставленной задачи. Поскольку каждый индивидуум представлен некоторой хромосомой, эту популяцию можно рассматривать как коллекцию хромосом:



Популяция индивидуумов,
представленных двоично-кодированными хромосомами

Популяция всегда представляет текущее поколение и эволюционирует со временем, когда текущее поколение заменяется новым.

Функция приспособленности

На каждой итерации алгоритма индивидуумы оцениваются с помощью **функции приспособленности** (или **целевой функции**). Это функция, которую мы стремимся оптимизировать, или задача, которую пытаемся решить.

Индивидуумы, для которых функция приспособленности дает наилучшую оценку, представляют лучшие решения и с большей вероятностью будут отобраны для воспроизводства и представлены в следующем поколении. Со временем качество решений повышается, значения функции приспособленности растут, а когда будет найдено удовлетворительное значение, процесс можно остановить.

Отбор

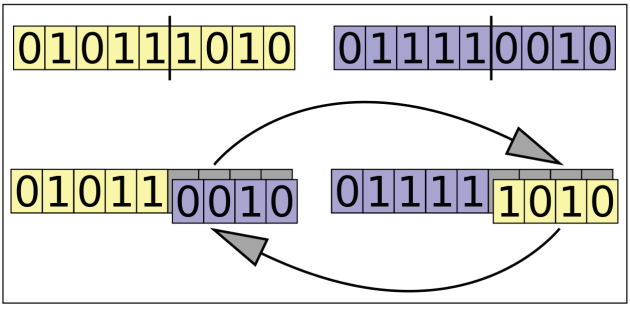
После того как вычислены приспособленности всех индивидуумов в популяции, начинается процесс отбора, который определяет, какие индивидуумы будут оставлены для воспроизводства, т. е. создания потомков, образующих следующее поколение.

Процесс отбора основан на оценке приспособленности индивидуумов. Те, чья оценка выше, имеют больше шансов передать свой генетический материал следующему поколению.

Плохо приспособленные индивидуумы все равно могут быть отобраны, но с меньшей вероятностью. Таким образом, их генетический материал не полностью исключен.

Скрещивание

Для создания пары новых индивидуумов родители обычно выбираются из текущего поколения, а части их хромосом меняются местами (скрещиваются), в результате чего создаются две новые хромосомы, представляющие потомков. Эта операция называется скрещиванием, или рекомбинацией.



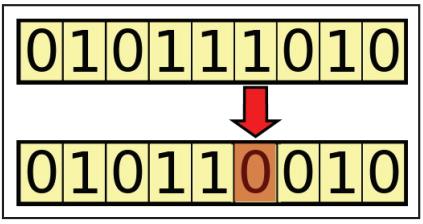
Операция скрещивания двух двоично-кодированных хромосом

На рисунке выше показана операция скрещивания с созданием двух потомков родителей.

Мутация

Цель оператора мутации – периодически случайным образом **обновлять** популяцию, т. е. вносить новые сочетания генов в хромосомы, стимулируя тем самым поиск в неисследованных областях пространства решений.

Мутация может проявляться как случайное изменение гена. Мутации реализуются с помощью внесения случайных изменений в значения хромосом, например инвертирования одного бита в двоичной строке.



Применение оператора мутации
к двоично-кодированной хромосоме

На рисунке выше приведен пример операции мутации.

Далее мы рассмотрим теорию, стоящую за генетическими алгоритмами.

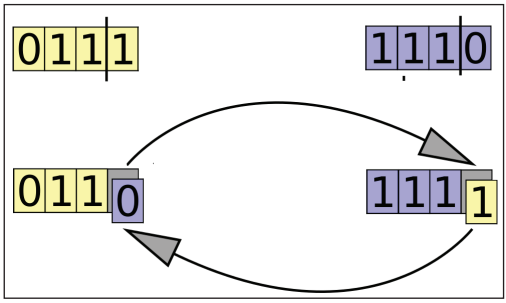
Теоретические основы генетических алгоритмов

Гипотеза структурных элементов, лежащая в основе генетических алгоритмов, заключается в том, что оптимальное решение задачи может быть собрано из небольших структурных элементов, и чем их больше, тем ближе мы подходим к оптимальному решению.

Индивидуумам, которые содержат некоторые из желательных структурных элементов, назначается более высокая оценка. Повторные операции отбора и скрещивания приводят к появлению все лучших индивидуумов, передающих эти структурные элементы следующему поколению, возможно, в сочетании с другими успешными структурными элементами. Тем самым создается генетическое давление, направляющее популяцию в сторону появления все большего числа индивидуумов, обладающих структурными элементами, образующими оптимальное решение.

В результате каждое поколение оказывается лучше предыдущего и содержит больше индивидуумов, близких к оптимальному решению.

Например, если имеется популяция четырехзначных двоичных строк и требуется найти строки с максимальной суммой цифр, то цифра 1 в любой из четырех позиций является хорошим структурным элементом. В процессе работы алгоритм будет определять решения, содержащие такие структурные элементы, и объединять их. В каждом поколении будет больше индивидуумов, содержащих 1 в различных позициях, а в конечном итоге получится строка 1111, объединяющая все четыре желательных структурных элемента. Это показано на рисунке ниже.



Демонстрация операции скрещивания –
структурные элементы оптимального решения собираются вместе

Здесь мы видим, как два индивидуума, представляющих хорошие решения задачи (в каждом три бита равны 1), порождают потомка, дающего наилучшее решение (четыре единичных бита), после того как операция скрещивания объединяет желательные элементы обоих родителей.

Теорема о схемах

Формальное выражение гипотезы о структурных элементах является содержанием **теоремы Холланда о схемах**, или **основной теоремы генетических алгоритмов**.

Эта теорема говорит о схемах, т. е. паттернах (или закономерностях), обнаруживаемых в хромосомах. Каждая схема представляет подмножество хромосом, обладающих определенным сходством.

Например, если набор хромосом представлен двоичными строками длины 4, то схема *1*01* представляет все хромосомы, у которых в крайней левой позиции находится 1, в двух крайних справа – 01, а во второй слева позиции может находиться как 1, так и 0, поскольку * – это **метасимвол**, сопоставляемый с любым конкретным символом.

Для каждой схемы можно определить две характеристики:

- **порядок**: количество фиксированных цифр (не метасимволов);
- **определяющая длина**: расстояние между крайними фиксированными цифрами.

В таблице ниже приведены примеры четырехзначных двоичных схем и их характеристик:

Схема	Порядок	Определяющая длина
1101	4	3
1*01	3	3
*101	3	2
*1*1	2	2
**01	2	1
1***	1	0
****	0	0

Каждая хромосома в популяции соответствует нескольким схемам – точно так же, как заданная строка соответствует разным регулярным выражениям. Например, хромосома 1101 соответствует любой из перечисленных в таблице схем. Если оценка этой хромосомы высока, то она с большей вероятностью успешно пройдет отбор вместе со всеми схемами, которые представляет. После скрещивания с другой хромосомой или мутации некоторые схемы выживут, а остальные исчезнут. Больше шансов на выживание у схем низкого порядка с малым определяющим расстоянием.

Теорема о схемах утверждает, что частота схем низкого порядка с малым определяющим расстоянием и приспособленностью выше средней экспоненциально возрастает в последующих поколениях. Иными словами, генетический алгоритм увеличивает частоту в популяции небольших и простых структурных элементов, представляющих атрибуты, благодаря которым решение становится лучше.

Отличия от традиционных алгоритмов

Между генетическими и традиционными алгоритмами поиска и оптимизации имеется несколько важных различий.

Перечислим основные характеристики генетических алгоритмов, отличающие их от традиционных:

- поддержание популяции решений:
- использование генетического представления решений;
- использование функции приспособленности;
- вероятностное поведение.

В следующих разделах эти факторы описываются более подробно.

Популяция как основа алгоритма

Целью генетического поиска является популяция потенциальных решений (индивидуумов), а не единственное решение. В любой точке поиска алгоритм сохраняет множество индивидуумов, образующих текущее поколение. На каждой итерации генетического алгоритма создается следующее поколение индивидуумов.

С другой стороны, в большинстве других алгоритмов поиска хранится единственное решение, которое итеративно улучшается. Например, алгоритм **градиентного спуска** итеративно сдвигает текущее решение в направлении наискорейшего спуска, которое определяется антиградиентом заданной функции.

Генетическое представление

Генетические алгоритмы работают не с самими потенциальными решениями, а с их кодированными представлениями, которые часто называют **хромосомами**. Простым примером хромосомы является двоичная строка фиксированной длины.

Хромосомы позволяют определить генетические операции скрещивания и мутации. Скрещивание реализуется обменом частей родительских хромосом, а мутация – изменением частей хромосом.

Побочный эффект генетического представления – отделение поиска от исходной предметной области. Генетические алгоритмы не знают, что именно представляют хромосомы, и не пытаются их интерпретировать.

Функция приспособленности

Функция приспособленности представляет проблему, которую мы пытаемся решить. Цель генетического алгоритма – найти индивидуумов, для которых оценка, вычисляемая функцией приспособленности, максимальна.

В отличие от традиционных алгоритмов поиска, генетические алгоритмы анализируют только значение, возвращенное функцией приспособленности, их не интересует ни производная, ни какая-либо другая информация. Поэтому они могут работать с функциями, которые трудно или невозможно продифференцировать.

Вероятностное поведение

Многие традиционные алгоритмы по природе своей детерминированы, тогда как правила, применяемые генетическими алгоритмами для перехода от предыдущего поколения к следующему, вероятностные.

Например, вероятность отбора индивидуума для создания следующего поколения тем выше, чем больше значение функции приспособленности, но элемент случайности все равно присутствует. Слабо приспособленные индивидуумы могут быть отобраны, хотя вероятность этого ниже.

Мутации тоже имеют вероятностный характер, обычно их вероятность мала, а изменению подвергаются случайные позиции в хромосоме.

Случайность присутствует и в операторе скрещивания. В некоторых генетических алгоритмах скрещивание происходит лишь с некоторой вероятностью. Если скрещивания не было, то оба родителя дублируются в следующем поколении вообще без изменений.

Несмотря на вероятностную природу процесса, поиск, основанный на генетическом алгоритме, нельзя назвать случайным; случайность используется, чтобы направить поиск в сторону тех областей пространства поиска, где выше шансы улучшить результаты. Теперь рассмотрим преимущества генетических алгоритмов.

Преимущества генетических алгоритмов

Особенности генетических алгоритмов, рассмотренные в предыдущих разделах, определяют их преимущества по сравнению с традиционными алгоритмами поиска.

Ниже перечислены основные преимущества генетических алгоритмов:

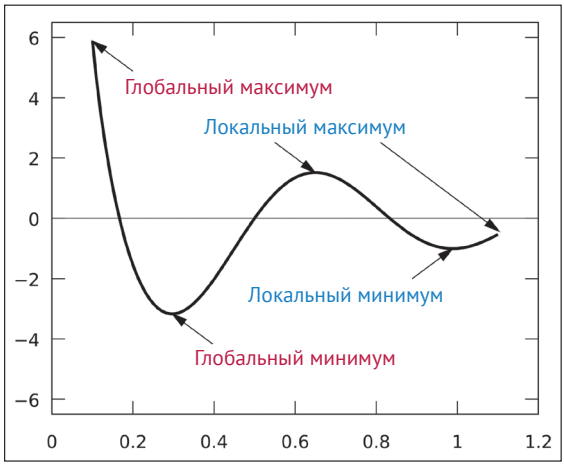
- способность выполнять глобальную оптимизацию;
- применимость к задачам со сложным математическим представлением;
- применимость к задачам, не имеющим математического представления;
- устойчивость к шуму;
- поддержка распараллеливания и распределенной обработки;
- пригодность к непрерывному обучению.

Мы рассмотрим их поочередно в следующих разделах.

Глобальная оптимизация

Во многих задачах оптимизации имеются точки локального максимума и минимума, которые представляют решения, лучшие, чем те, что находятся поблизости, но необязательно лучшие в глобальном смысле.

На рисунке ниже показаны различия между локальными и глобальными минимумами и максимумами.



Локальные и глобальные максимумы и минимумы функции

Источник: https://commons.wikimedia.org/wiki/File:Computational.science.Genetic.algorithm.Crossover.One.Point.svg. Автор KSmrq. Публикуется по лицензии Creative Commons CC BY-SA 3.0: https://creativecommons.org/licenses/by-sa/3.0/

Большинство традиционных алгоритмов поиска и оптимизации, а особенно те, что основаны на вычислении градиента, могут застревать в локальном максимуме, вместо того чтобы найти глобальный. Это связано с тем, что в окрестности локального максимума всякое небольшое изменение решения ухудшает оценку.

Генетические алгоритмы менее подвержены этой напасти и имеют больше шансов отыскать глобальный максимум. Объясняется это тем, что используется популяция потенциальных решений, а не единственное решение, а операции скрещивания и мутации зачастую порождают решения, далеко отстоящие от ранее рассмотренных. Это остается справедливым при условии, что мы поддерживаем разнообразие популяции и избегаем **преждевременной сходимости**, о чем поговорим в следующем разделе.

Применимость к сложным задачам

Поскольку генетическим алгоритмам нужно знать только значение функции приспособленности каждого индивидуума, а все остальные ее свойства, в частности производные, несущественны, их можно применять к задачам со сложным математическим представлением, включающим функции, которые трудно или невозможно продифференцировать.

К сложным случаям, когда достоинства генетических алгоритмов раскрываются во всем блеске, относятся также задачи с большим числом параметров или со смешанными параметрами, например непрерывными и дискретными.

Применимость к задачам, не имеющим математического представления

Генетические алгоритмы применимы и к задачам, вообще не имеющим математического представления. Один из таких случаев, представляющий особый интерес, – когда оценка приспособленности основана на мнении человека. Пусть, например, требуется найти наиболее привлекательную цветовую палитру для веб-сайта. Мы можем попробовать разные комбинации цветов и попросить пользователей оценить привлекательность сайта. А затем применить генетический алгоритм, чтобы найти лучшую комбинацию, используя функцию приспособленности, основанную на оценках пользователей. Алгоритм будет работать, несмотря на то что никакого математического представления нет и невозможно вычислить оценку заданной комбинации непосредственно.

В следующей главе мы увидим, что генетические алгоритмы справляются даже со случаями, когда нельзя получить оценку каждого индивидуума, при условии что имеется возможность сравнить двух индивидуумов и определить, какой из них лучше. Примером может служить алгоритм машинного обучения, который управляет автомобилем в имитации гонок. Поиск на основе генетического алгоритма может оптимизировать и настроить алгоритм машинного обучения, организовав состязание различных вариантов алгоритма между собой, чтобы определить, какой лучше.

Устойчивость к шуму

Для некоторых задач характерно присутствие **шума**. Это означает, что даже при близких истинных значениях входных параметров результаты их измерений могут довольно сильно различаться. Например, так бывает, когда данные считываются с датчиков или когда оценка основана на мнении человека, как было описано в предыдущем разделе.

Подобное поведение может сделать непригодными многие традиционные алгоритмы поиска, но генетические алгоритмы в общем случае устойчивы к нему благодаря повторяющимся операциям сборки и оценивания индивидуумов.

Распараллеливание

Генетические алгоритмы хорошо поддаются распараллеливанию и распределенной обработке. Функция приспособленности независимо вычисляется для каждого индивидуума, а это значит, что все индивидуумы в популяции могут обрабатываться одновременно.

Кроме того, операции отбора, скрещивания и мутации могут одновременно выполняться для индивидуумов и пар индивидуумов.

Поэтому подход, основанный на генетических алгоритмах, естественно адаптируется к распределенным и облачным реализациям.

Непрерывное обучение

В природе эволюция никогда не прекращается. Если окружающие условия изменяются, популяция приспосабливается к ним. Так и генетические алгоритмы могут непрерывно работать в постоянно изменяющихся условиях, и мы всегда можем получить и использовать лучшее на данный момент решение.

Но это возможно, только если окружающая среда изменяется медленно по сравнению со скоростью смены поколений в генетическом алгоритме. Итак, преимущества мы обсудили, теперь перейдем к ограничениям генетических алгоритмов.

Ограничения генетических алгоритмов

Чтобы получить максимум пользы от генетических алгоритмов, мы должны знать об их ограничениях и потенциальных подвохах.

Ниже перечислены ограничения генетических алгоритмов:

- необходимы специальные определения;
- необходима настройка гиперпараметров;
- большой объем счетных операций;
- опасность преждевременной сходимости;
- отсутствие гарантированного решения.

Специальные определения

Пытаясь применить генетические алгоритмы к некоторой задаче, мы должны создать подходящее представление – определить функцию приспособленности и структуру хромосом, а также операторы отбора, скрещивания и мутации. Зачастую это совсем не просто и занимает много времени.

По счастью, генетические алгоритмы уже бессчетное число раз применялись к самым разным задачам, так что многие определения стандартизованы. В этой книге рассматриваются многочисленные примеры реальных задач и способы их решения с помощью генетических алгоритмов. Можете использовать ее как руководство, когда столкнетесь с новой задачей.

Настройка гиперпараметров

Поведение генетических алгоритмов контролируется набором гиперпараметров, например размером популяции и скоростью мутации. Точных правил для выбора значений гиперпараметров не существует.

Однако так обстоит дело практически со всеми алгоритмами поиска и оптимизации. Проработав примеры в книге и поэкспериментировав самостоятельно, вы научитесь подбирать разумные значения гиперпараметров.

Большой объем счетных операций

Работа с потенциально большими популяциями и итеративный характер генетических алгоритмов обусловливают большой объем вычислений, поэтому на получение приемлемого результата может уйти много времени.

Проблему можно сгладить за счет хорошего выбора гиперпараметров, распараллеливания и в некоторых случаях кеширования промежуточных результатов.

Преждевременная сходимость

Если приспособленность какого-то индивидуума гораздо больше, чем у всей остальной популяции, то не исключено, что он продублируется так много раз, что в конечном счете, кроме него, в популяции ничего не останется. В результате генетический алгоритм может застрять в локальном максимуме и не найдет глобального.

Чтобы предотвратить такое развитие событий, важно поддерживать разнообразие популяции. В следующей главе мы рассмотрим различные способы достижения этой цели.

Отсутствие гарантированного решения

Использование генетических алгоритмов не гарантирует нахождения глобального максимума.

Однако это типично для всех алгоритмов поиска и оптимизации, если только у задачи не существует аналитического решения.

Но, вообще говоря, при правильном применении генетические алгоритмы находят хорошие решения на разумное время. Далее мы рассмотрим несколько ситуаций, в которых применение генетических алгоритмов оправдано.

Сценарии применения генетических алгоритмов

Резюмируя изложенное в предыдущих разделах, можно сказать, что генетические алгоритмы лучше применять для решения следующих задач.

- **Задачи со сложным математическим представлением.** Поскольку генетическим алгоритмам нужно знать только значение функции приспособленности, их можно использовать для решения задач, в которых целевую функцию трудно или невозможно продифференцировать, задач с большим количеством параметров и задач с параметрами разных типов.
- **Задачи, не имеющие математического представления.** Генетические алгоритмы не требуют математического представления задачи,

коль скоро можно получить значение оценки или существует метод сравнения двух решений.

- **Задачи с зашумленной окружающей средой.** Генетические алгоритмы устойчивы к зашумленным данным, например прочитанным с датчика или основанным на оценках, сделанных человеком.
- **Задачи, в которых окружающая среда изменяется во времени.** Генетические алгоритмы могут адаптироваться к медленным изменениям окружающей среды, поскольку постоянно создают новые поколения, приспосабливающиеся к изменениям.

С другой стороны, если для задачи известен специализированный способ решения традиционным или аналитическим методом, то вполне вероятно, что он окажется эффективнее.

Резюме

Мы начали эту главу со знакомства с генетическими алгоритмами, с аналогии между ними и дарвиновской эволюцией и с основных принципов их работы: использования популяции, генотипа, функции приспособленности и операторов отбора, скрещивания и мутации.

Затем рассмотрели теорию, лежащую в основе генетических алгоритмов, – гипотезу структурных элементов и теорему о схемах – и показали, как генетические алгоритмы работают, собирая наилучшие решения из небольших структурных элементов, обладающих превосходными качествами.

Далее мы разобрали различия между генетическими и традиционными алгоритмами, в т. ч. поддержание популяции решений и использование генетического представления решений.

После этого мы обсудили сильные стороны генетических алгоритмов, включая возможность глобальной оптимизации, применимость к задачам со сложным математическим представлением или вообще без такового и устойчивость к шуму. Мы также поговорили о недостатках: необходимости специальных определений и настройки гиперпараметров, опасности преждевременной сходимости.

В заключение перечислили ситуации, когда применение генетических алгоритмов может дать выигрыш: математически сложные задачи, оптимизация в зашумленной или постоянно изменяющейся среде.

В главе 2 мы более подробно рассмотрим ключевые компоненты и детали реализации генетических алгоритмов. Это станет подготовкой к последующим главам, где мы приведем код решения различных задач.

Для дальнейшего чтения

Дополнительный материал, относящийся к этой главе, можно найти в главе «Введение в генетические алгоритмы» книги Amita Kapoor «Hands-On Artificial Intelligence for IoT» (январь 2019), опубликованной по адресу https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788836067.