Обучение с подкреплением

Обучение с подкреплением используют там, где задачей стоит не анализ данных, а выживание в реальной среде.

Средой может быть видеоигра, смоделированный или реальный мир. Как пример — автопилот Теслы, который учится не сбивать пешеходов.

В процессе обучения модель штрафуют за ошибки и награждают за правильные поступки.

Знания об окружающем мире такой модели могут быть полезны, но чисто для справки. Не важно сколько данных она соберёт, у нее всё равно не получится предусмотреть все ситуации. Потому её цель — минимизировать ошибки, а не рассчитать все ходы. Модель учится выживать в пространстве с максимальной выгодой: собранными монетками в Марио, временем поездки в Тесле.

Запоминать сам город машине не нужно — такой подход называется ***Model-Free***. Конечно, существует и классический подход ***Model-Based***, но в нём нашей машине пришлось бы запоминать модель всей планеты, всех возможных ситуаций на всех перекрёстках мира. Такое просто не работает.

В обучении с подкреплением машина не запоминает каждое движение, а пытается обобщить ситуации, чтобы выходить из них с максимальной выгодой.

«Брось робота в лабиринт и пусть ищет выход»

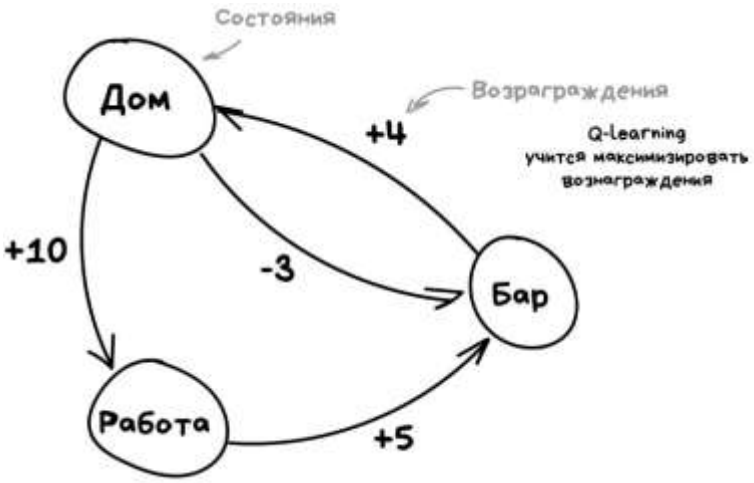
Сегодня используют для:

- Самоуправляемых автомобилей
- Роботов пылесосов
- Игр
- Автоматической торговли
- Управления ресурсами предприятий

Популярные алгоритмы: **Q-Learning,** SARSA, DQN, A3C, **Генетический Алгоритм**

Алгоритм Q-learning

При обучении с подкреплением модель учится выбирать наилучший выход из каждой ситуации. Эта идея лежит в основе алгоритма Q-learning и его производных (SARSA и DQN).



Буква Q в названии означает слово Quality, то есть модель учится поступать наиболее качественно в любой ситуации, а все ситуации он запоминает как простой марковский процесс.

Генетические алгоритмы

Как говорит Википедия: «Отец-основатель генетических алгоритмов Джон Холланд, который придумал использовать генетику в своих целях в 1975 году»

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

Генетические алгоритмы реализуют упрощенный вариант дарвиновской эволюции. Основные принципы **эволюционной теории Дарвина** следующие:

- **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.

Те особи, которые лучше приспособлены 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.

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.

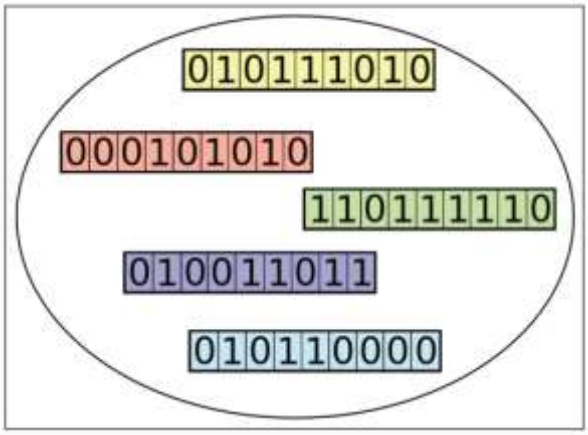
Цель генетических алгоритмов – найти оптимальное решение некоторой задачи. Если дарвиновская эволюция развивает популяцию отдельных особей, то генетические алгоритмы развивают популяцию потенциальных решений данной задачи, называемых индивидуумами. Эти решения итеративно оцениваются и используются для создания нового поколения решений.

Генотип

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



Функция приспособленности (целевая функция)

На каждой итерации алгоритма индивидуумы оцениваются с помощью функции приспособленности (или целевой функции) ***fitness(individ)***. 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.

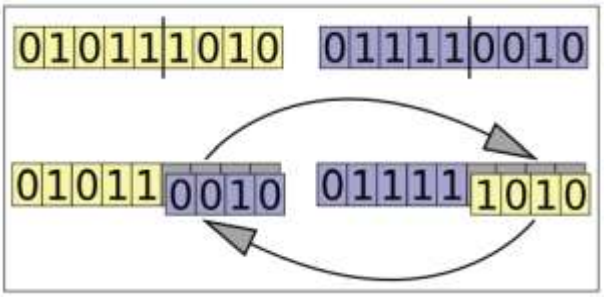
Индивидуумам, которые содержат некоторые из желательных структурных элементов, назначается более высокая оценка. Повторные операции отбора и скрещивания приводят к появлению все лучших индивидуумов, передающих эти структурные элементы следующему поколению, возможно, в сочетании с другими успешными структурными элементами. В результате каждое поколение оказывается лучше предыдущего и содержит больше индивидуумов, близких к оптимальному решению.

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

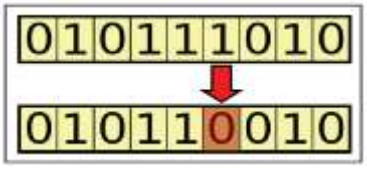
Для создания пары новых индивидуумов родители обычно выбираются из текущего поколения, а части их хромосом меняются местами (скрещиваются), в результате чего создаются две новые хромосомы, представляющие потомков. Эта операция называется скрещиванием, или рекомбинацией.



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



Проверка условий остановки

Может существовать несколько условий, при выполнении которых процесс останавливается. Сначала отметим два самых распространенных:

- достигнуто максимальное количество поколений. Это условие заодно позволяет ограничить время работы алгоритма и потребление им ресурсов системы;
- на протяжении нескольких последних поколений не наблюдается заметных улучшений. Это можно реализовать путем запоминания наилучшей приспособленности, достигнутой в каждом поколении, и сравнения наилучшего текущего значения со значениями в нескольких
- предыдущих поколениях. Если разница меньше заранее заданного порога, то алгоритм можно останавливать;
- с момента начала прошло заранее определенное время;
- превышен некоторый лимит затрат, например процессорного времени
- или памяти;

- наилучшее решение заняло часть популяции, большую заранее заданного порога.

Резюмируем. Генетический алгоритм начинается с популяции случайно выбранных потенциальных решений (индивидуумов), для которых вычисляется функция приспособленности. Алгоритм выполняет цикл, в котором последовательно применяются операторы отбора, скрещивания и мутации, после чего приспособленность индивидуумов пересчитывается. Цикл продолжается, пока не выполнено условие остановки, после чего лучший индивидуум в текущей популяции считается решением. Этот процесс можно представить в виде блок-схемы базовой структуры генетического алгоритма:

Базовая структура генетического алгоритма

Рассмотрим основные подходы к реализации операторов отбора, скрещивания и мутации, которые составляют основу любого ГА:

Методы отбора

Начнем с обзора методов отбора лучших хромосом в популяции. Напомню, что на этом этапе отбираются наиболее приспособленные особи, которые становятся «родителями» для формирования следующего поколения. Но, при этом, необходимо сохранить возможность и менее приспособленным особям давать незначительное количество своих потомков для сохранения разнообразия генофонда.

- ***Правило рулетки***

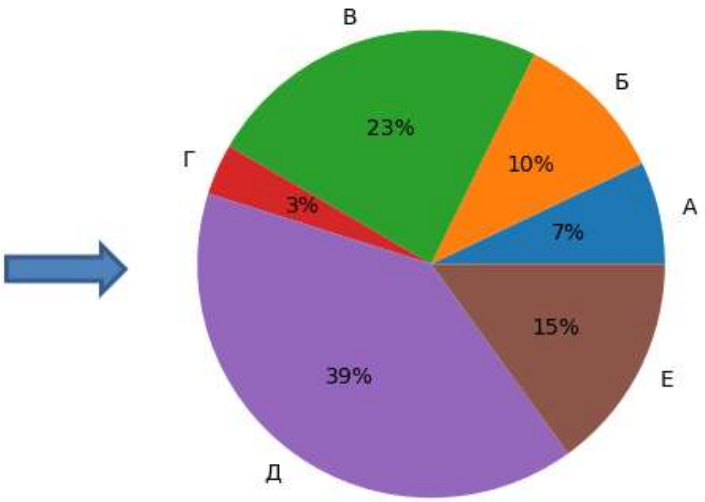
Метод отбора по правилу рулетки, или отбор пропорциональной приспособленности (fitness proportionate selection – FPS), заключается в случайном выборе индивидуума из популяции пропорционально его

приспособленности. Предположим, у нас имеется несколько особей в популяции с разной степенью адаптации:

Индивидуум	Приспособленность	Доля
А	8	7 %
Б	12	10 %
В	27	23 %
Г	4	3 %
Д	45	39 %
Е	17	15 %

Тогда их можно условно представить на круговой диаграмме с секторами, размером соответствующих долей:

После каждого запуска рулетки отбор индивидуума из популяции производится в точке отбора (стрелка на рисунке). Затем рулетка запускается еще раз для выбора следующего индивидуума, и так до тех пор, пока не наберется достаточно индивидуумов для образования следующего поколения. В результате один и тот же индивидуум может быть выбран несколько раз. Очевидно, что чаще (с наибольшей вероятностью) будут отбираться индивидуумы с большей приспособленностью, так как у них сектор занимает большую долю.

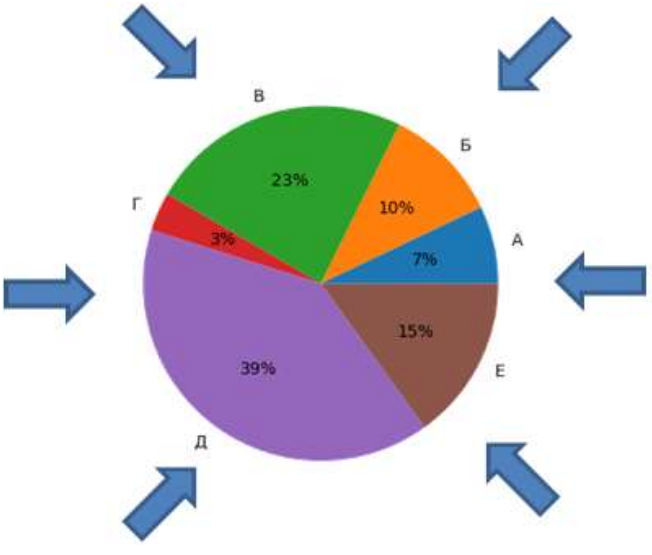


- ***Стохастическая универсальная выборка***

Стохастическая универсальная выборка (stochastic universal sampling – SUS) – модифицированный вариант рулетки, когда точки отбора (стрелки) располагаются равномерно вокруг диаграммы:

Вращение выполняется один раз и сразу производится отбор всех особей для дальнейшей операции скрещивания.

Преимущество этого подхода в том, что он гарантирует отбор не только индивида с большой долей сектора, но, скорее всего, будут отобраны и другие особи с меньшими секторами. То есть, выбор более равномерен и сохраняет, в некоторой степени, разнообразие популяции.



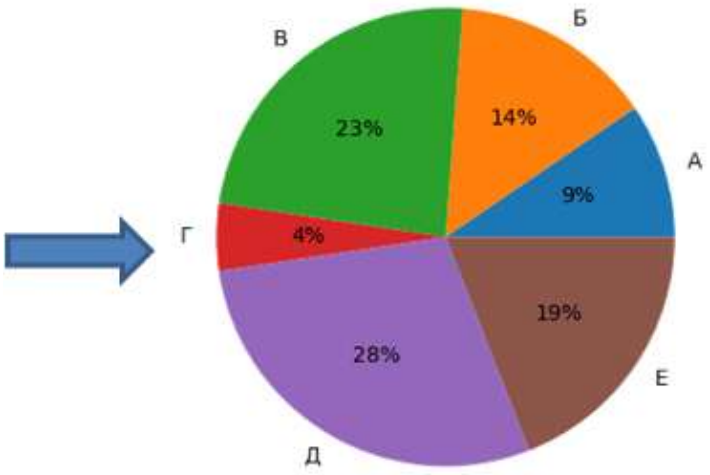
- ***Ранжированный отбор***

Следующий метод ранжированного отбора работает похожим образом на правило рулетки, но доли для индивидуумов вычисляются несколько иначе. Вначале все индивидуумы ранжируются (упорядочиваются) по возрастанию их приспособленности (см. столбец «Ранг»):

Индивидуум	Приспособленность	Ранг	Доля ранга
А	8	2	9 %
Б	12	3	14 %
В	27	5	23 %
Г	4	1	4 %
Д	45	6	28 %
Е	17	4	19 %

Затем, вычисляются доли относительно ранга (а не приспособленности, как в правиле рулетки) и, тем самым, доли секторов становятся более равномерными, а значит, более равномерно будут отбираться родители из популяции.

Этот подход дает хорошие результаты если в популяции имеются отдельные особи с высокой приспособленностью. Тогда, чтобы они не «захватили» всю популяцию, отбор лучше делать по рангу, а не по приспособленности. Также этот способ позволяет отбирать лучших претендентов, когда приспособленности особей в популяции примерно одинаковы. Тогда ранг позволит с заметно большей вероятностью отбирать самых лучших, сохраняя отбор менее приспособленных.



- ***Масштабирование приспособленности***

Если при ранжированном отборе приспособленности заменяются рангами, то в случае масштабирования приспособленности к исходным значениям приспособленности применяется линейное преобразование, которое переводит их в желательный диапазон:

масштабированная приспособленность = a × (исходная приспособленность) + b,

где a и b – постоянные, выбираемые по нашему усмотрению

Например, значения приспособленности от 4 до 45 переводим в новый диапазон [30; 80].

Индивидуум	Приспособленность f	Новое значение f*
А	8	34,9
Б	12	40
В	27	58
Г	4	30
Д	45	80
Е	17	45,8

Тогда получим следующее распределение долей секторов после операции масштабирования:

Благодаря этому, особи отбираются из популяции более равномерно, и у худших появляется реальный шанс оказаться в новой выборке и тем самым обеспечить разнообразие хромосом. С другой стороны, хорошо приспособленные особи не заменяют собой новую популяцию, а всего лишь с большей вероятностью отбираются наряду с остальными претендентами. Все это благотворно влияет на сходимость ГА к оптимальному или близкому к оптимальному решению.

- ***Турнирный отбор***

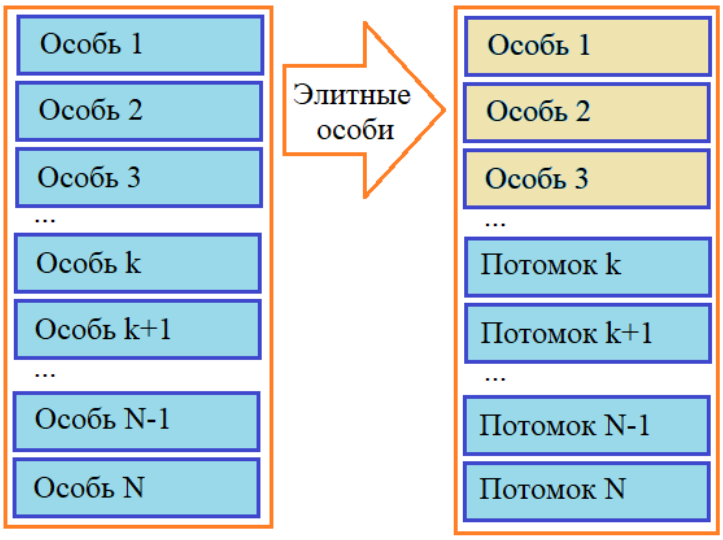
Каждый раз из популяции случайным образом отбирается несколько претендентов (от двух и более). Затем, среди отобранных участников выбирается наиболее приспособленный (с наибольшим значением функции принадлежности). Он и переходит в новую выборку.

Пример турнирного отбора на турнире с тремя участниками

Процесс повторяется до тех пор, пока число «родителей» не станет равно размеру популяции.

- ***Элитизм***

Еще один механизм, применяемый совместно с оператором отбора в ГА – элитизм. Его смысл очень прост. В следующее поколение мы гарантированно отбираем небольшое число наиболее приспособленных (элитных) особей. Причем, эти элитные индивидуумы также участвуют в скрещивании на правах родителей. Такой подход позволяет сохранять в популяции лучшие решения (не теряя их и не переоткрывая заново). В ряде задач это существенно увеличивает скорость сходимости ГА к некоторому приемлемому или оптимальному решению.

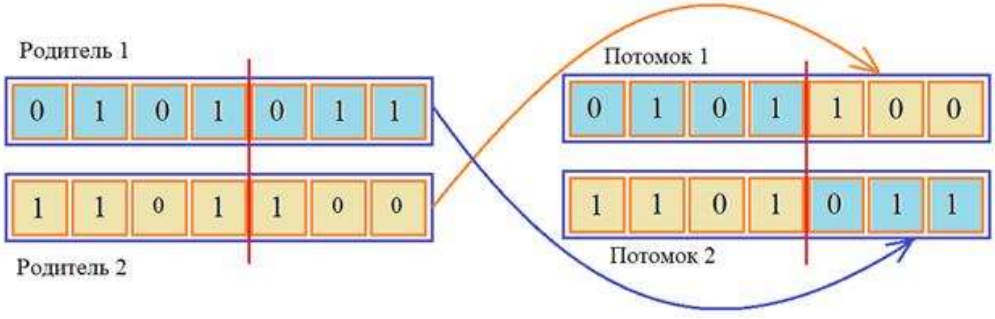


Методы скрещивания

Скрещивание (также называется кроссинговер и кроссовер) следующая базовая операция в генетическом алгоритме. Здесь перебираются пары родителей (как правило, без повторения) из отобранной популяции и с некоторой высокой вероятностью выполняется обмен фрагментами генетической информации для формирования хромосом двух потомков. Если родители не участвовали в скрещивании, то они переносятся (копируются) в следующее поколение.

- ***Одноточечное скрещивание***

В самом простом варианте операция кроссинговера выполняет обмен между двумя половинками хромосом родителей для формирования хромосом потомков.

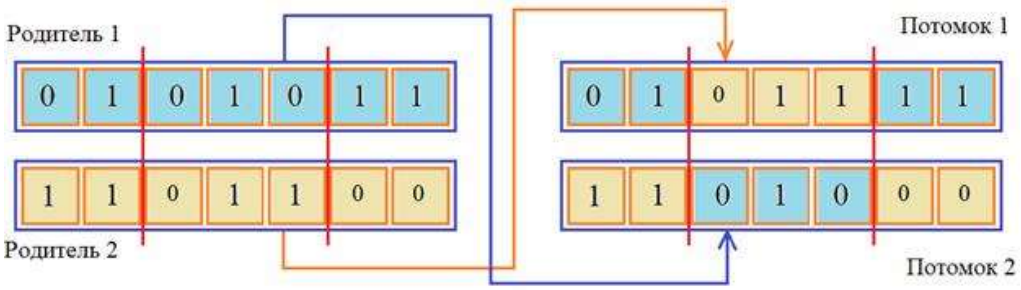


Вначале случайным образом определяется точка разреза хромосомы, а затем, соответствующие части меняются местами. Получаются две новые хромосомы для двух потомков.

- ***Двухточечное и k-точечное скрещивание***

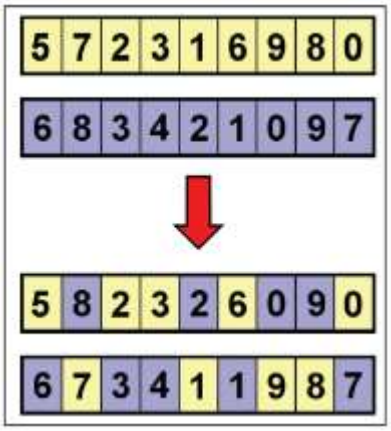
Как показывает практика, двухточечное скрещивание дает лучшие результаты, чем одноточечное. При двухточечном кроссинговере вместо одной точки разреза выбираются две случайным образом. А реализовать его можно с помощью двух одноточечных операторов скрещивания. Сначала делается обмен частями хромосом по первой точке разреза, а затем, - по второй.

Аналогично можно реализовать и k-точечное скрещивание.



- ***Равномерное скрещивание***

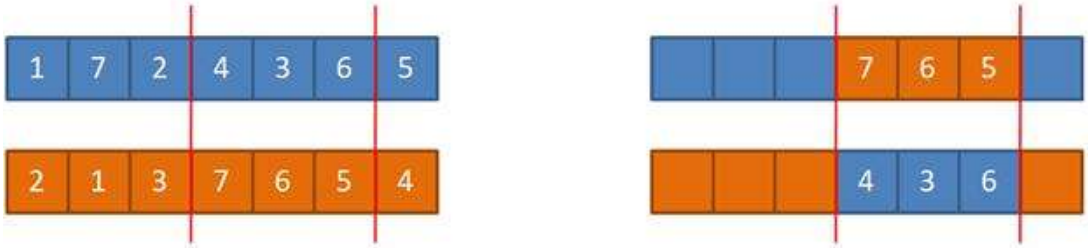
Эта идея скрещивания заключается в формировании потомков из скрещивания отдельных пар ген родителей, которые отбираются из хромосом случайным образом.



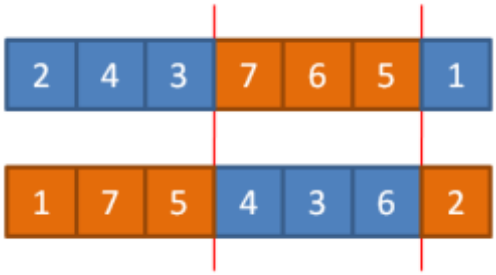
- ***Упорядоченное скрещивание***

В предыдущем примере хромосомы родителей состояли из неповторяющихся чисел от 1 до 9, но после скрещивания мы получили потомков, у которых некоторые числа стали повторяться. В некоторых случаях такой результат является недопустимым. Например, если с помощью ГА решается задача поиска кратчайшего пути обхода всех городов (задача коммивояжера).

Как раз для таких случаев был предложен способ упорядоченного скрещивания хромосом родителей. Идея его очень проста. Вначале делается уже известный вариант скрещивания, который не приводит к дублированию чисел, например, частый случай двухточечного скрещивания:



А оставшиеся значения генов у потомков заполняем следующим образом. Мы проходим все гены первого родителя, начиная от второй точки разреза, и добавляем значения, если они не еще не присутствуют в хромосоме первого потомка. Затем, ту же самую операцию выполняем со вторым родителем и вторым потомком. В результате, получим следующий набор генов у двух потомков:

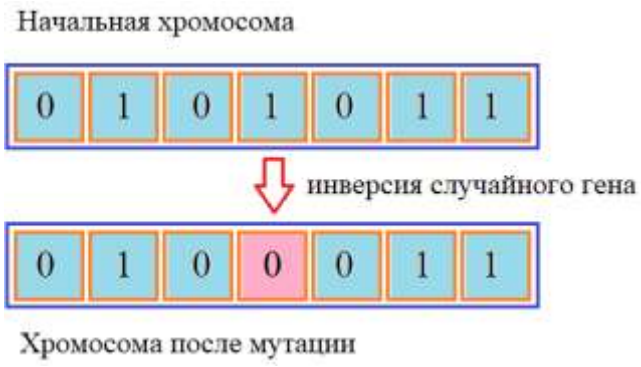


Методы мутации

Последний оператор, который применяется для формирования нового поколения популяции – это мутация. Обычно, она применяется с некоторой малой вероятностью к отдельным генам потомков, меняя их определенным образом. Мутация позволяет поддерживать генетическое разнообразие особей, чтобы популяция не выродилась и хромосомы не стали похожи друг на друга.

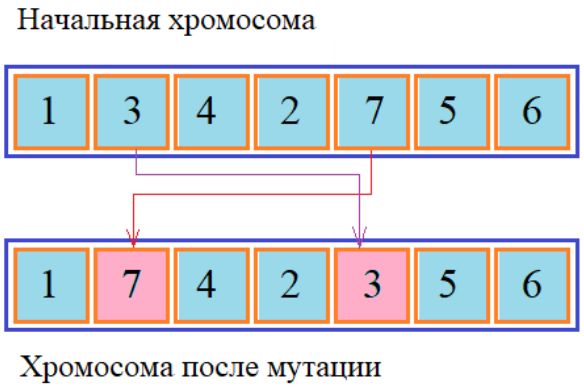
- ***Инвертирование битов***

Самый простой вариант мутации – это инвертирование битов, записанные в генах. Здесь может быть несколько вариантов реализации. Например, с некоторой (обычно низкой) вероятностью хромосома подвергается мутации и далее случайный бит (ген) переводится и инверсное состояние:



- ***Мутация обменом***

Для упорядоченных списков (когда в генах хромосомы хранятся индексы некоторого списка и они не должны повторяться, как в задаче коммивояжера) можно выполнять мутацию путем обмена случайно выбранных генов:

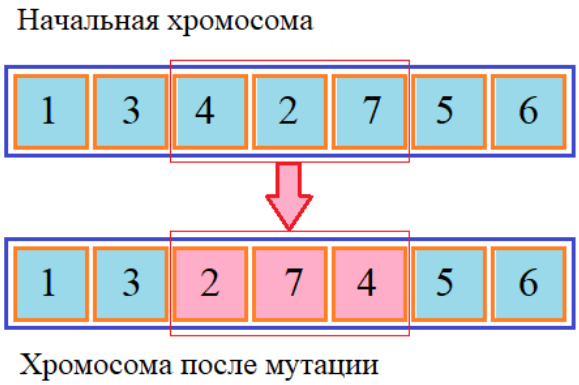


- ***Мутация обращением***

Несколько видоизмененная идея мутации обменом является другой способ – мутация обращением. Здесь мы выбираем также случайным образом непрерывную последовательность генов, которые, затем, записываем в обратном порядке:

- ***Мутация перетасовкой***

Несколько более разнообразные результаты мутации можно получить, если в предыдущем алгоритме значения ген записывать не в обратном, а случайном порядке. Такой подход известен под названием мутация перетасовкой:



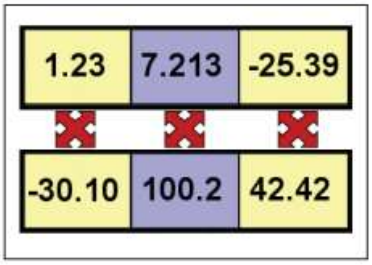
Генетические алгоритмы с вещественным кодированием

Часто встречаются задачи с непрерывным пространством решений, когда *индивидуумы описываются вещественными числами с плавающей точкой*. Исторически в генетических алгоритмах применялись двоичные строки для представления целых чисел, но для вещественных чисел это не годится.

Все методы отбора, описанные выше, будут работать точно так же для вещественных хромосом, поскольку они зависят только от приспособленности индивидуумов, а не от их представления.

Однако рассмотренные выше методы скрещивания и мутации для вещественных хромосом не годятся, поэтому нужны специальные методы. Эти операции скрещивания и мутации применяются по отдельности к каждой позиции массивов, представляющих вещественные хромосомы. Например, если родителями в операции скрещивания являются хромосомы [1.23, 7.213, –25.39] и [–30.10, 100.2, 42.42], то скрещивание производится отдельно для следующих пар:

- 1.23 и –30.10 (первая позиция);
- 7.213 и 100.2 (вторая позиция);
- –25.39 и 42.42 (третья позиция)



Пример скрещивания вещественных хромосом

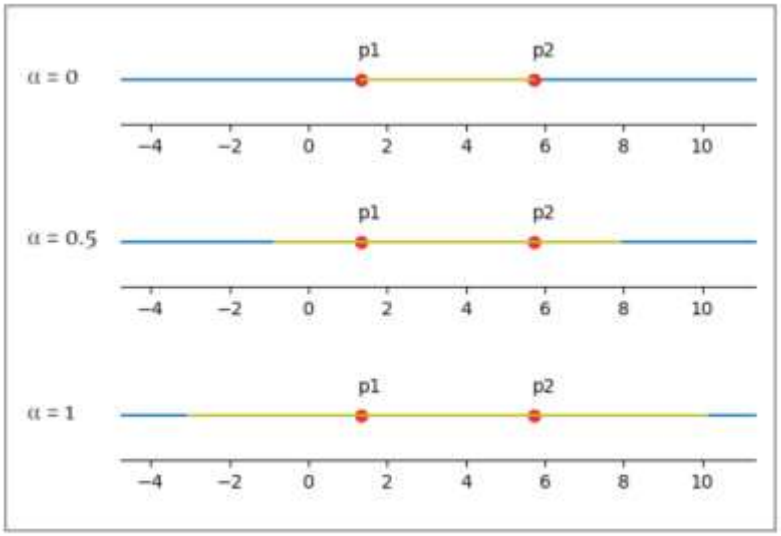
- **Скрещивание смешением**

В случае скрещивания смешением (blend crossover – BLX) каждый потомок случайным образом выбирается из следующего интервала, созданного родителями parent1 и parent2:

$$[parent1 - \alpha(parent2 - parent1), parent2 + \alpha(parent2 - parent1)].$$

Параметр α – постоянная от 0 до 1. Чем больше α, тем шире интервал.

Это показано на рисунке ниже, где родители обозначены p1 и p2, а интервал скрещивания изображен желтым цветом:



Пример скрещивания смешением

- **Имитация двоичного скрещивания**

Идея имитации двоичного скрещивания (simulated binary crossover – SBX) – имитировать свойства одноточечного скрещивания, часто применяемого для двоичных хромосом. Одно из свойств заключается в том, что среднее значений родителей равно среднему значений потомков.

В случае метода SBX потомки создаются из родителей по формуле:

$$offspring1 = ½ [(1 + в]parent1 + (1 – в)parent2]);$$

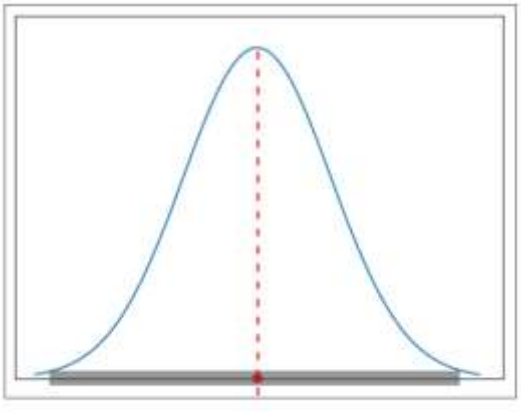
$$offspring2 = ½ [(1 – в]parent1 + (1 + в)parent2]),$$

где β – случайное число, называемое коэффициентом распределения.

- **Вещественная мутация**

Один из способов применения мутации в генетических алгоритмах с вещественным кодированием – заменить любое вещественное значение совершенно новым, случайно сгенерированным. Однако при этом может оказаться, что мутировавший индивидуум не имеет ничего общего с исходным.

Другой подход – сгенерировать случайное вещественное число, находящееся рядом с исходным индивидуумом. Примером может служить нормально распределенная (или гауссова) мутация: случайное число выбирается из нормального распределения со средним 0 и каким-то заранее заданным стандартным отклонением, как показано на рисунке:



Пример гауссовой мутации

Поисковые задачи и комбинаторная оптимизация

Одна из основных областей применений генетических алгоритмов – поисковые задачи, которые находят широкое применение в логистике, исследовании операций, искусственном интеллекте и машинном обучении. Примерами могут служить *нахождение оптимального маршрута доставки посылок, проектирование сетей транспортных авиалиний с хабами, управление портфельными инвестициями и распределение пассажиров между свободными машинами такси*.

Поисковый алгоритм решает задачу путем методичного перебора состояний и переходов состояний с целью найти путь из начального состояния в конечное (или целевое). Обычно с каждым переходом состояний ассоциированы некоторые затраты (стоимость) или доходы (выигрыш), а цель поискового алгоритма – найти путь, минимизирующий затраты или максимизирующий выигрыш. Поскольку оптимальный путь – один из многих возможных, такой поиск неразрывно связан с комбинаторной оптимизацией – нахождением оптимального объекта в конечном, но очень большом множестве.

- ***Решение задачи о рюкзаке***

Формально в задаче о рюкзаке имеются следующие компоненты:

- набор предметов, каждый из которых имеет ценность и вес;
- рюкзак (сумка, контейнер), в который можно поместить предметы ограниченного суммарного веса.

Наша цель – отобрать группу предметов максимальной суммарной ценности, так чтобы суммарный вес не превышал емкость контейнера.

Предмет	Вес	Ценность
Карта	9	150
Компас	13	35
Вода	153	200
Сэндвич	50	160
Глюкоза	15	60
Кружка	68	45
Банан	27	60
Яблоко	39	40
Сыр	23	30
Пиво	52	10
Крем от загара	11	70
Камера	32	30
Футболка	24	15
Брюки	48	10
Зонтик	73	40

DEAP - пакет для создания генетических алгоритмов

Пакет DEAP (сокращение от Distributed Evolutionary Algorithms in Python – распределенные эволюционные алгоритмы на Python) разработан специально для создания реализаций ГА на Python.

Модуль **creator** позволяет создавать новые объекты (классы) в программе.

Модуль **tools** содержит операторы для эволюционных алгоритмов (отбора, скрещивания и мутации).

Crossover	Mutation	Selection
cxOnePoint()	mutGaussian()	selTournament()
cxTwoPoint()	mutShuffleIndexes()	selRoulette()
cxUniform()	mutFlipBit()	selNSGA2()
cxPartialyMatched()	mutPolynomialBounded()	selNSGA3()
cxUniformPartialyMatched()	mutUniformInt()	selSPEA2()
cxOrdered()	mutESLogNormal()	selRandom()
cxBlend()		selBest()
cxESBlend()		selWorst()
cxESTwoPoint()		selTournamentDCD()
cxSimulatedBinary()		selDoubleTournament()
cxSimulatedBinaryBounded()		selStochasticUniversalSampling()
cxMessyOnePoint()		selLexicase()
		selEpsilonLexicase()
		selAutomaticEpsilonLexicase()

Преимущества и недостатки генетических алгоритмов

Генетические алгоритмы имеют свои преимущества и недостатки по сравнению с классическими алгоритмами оптимизации.

Преимущества генетических алгоритмов:

- способны находить глобальные оптимумы (с большей вероятностью, чем традиционные алгоритмы);
- можно применять в задачах, которые сложно формализовать математически, либо даже совсем не имеющие математического описания;
- устойчивы к шуму (например, когда данные не могут быть точно измерены или основаны на субъективном восприятии человека);
- можно реализовать на уровне параллельных вычислений для ускорения работы алгоритма;
- возможность непрерывного обучения (оптимизации) популяции при постоянном изменении окружающей среды (это изменение

сказывается на значениях функции принадлежности и популяции необходимо постоянно подстраиваться под новые обстоятельства).

Недостатки генетических алгоритмов:

- специфичная формализация исходной задачи на уровне генов и хромосом, а также операторов отбора, скрещивания и мутации;
- определение внешних параметров работы алгоритма (размер популяции, вероятности скрещивания и мутации, способ отбора и кроссинговера и т.п.);
- относительно большой объем вычислений (каждая особь в популяции, фактически, отдельный ход оптимизации задачи);
- некоторые особи в популяции могут быстро занять локальный оптимум и потянуть за собой всю популяцию (будут продублированы оператором отбора много раз, и популяция будет состоять, в основном, из их отпрысков), в результате другой, глобальный оптимум может быть не обнаружен;
- генетический алгоритм не гарантирует нахождение именно оптимального решения, нам приходится довольствоваться тем лучшим решением, которое было найдено (однако, это недостаток многих алгоритмов оптимизации).

Отличия от традиционных алгоритмов

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

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

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

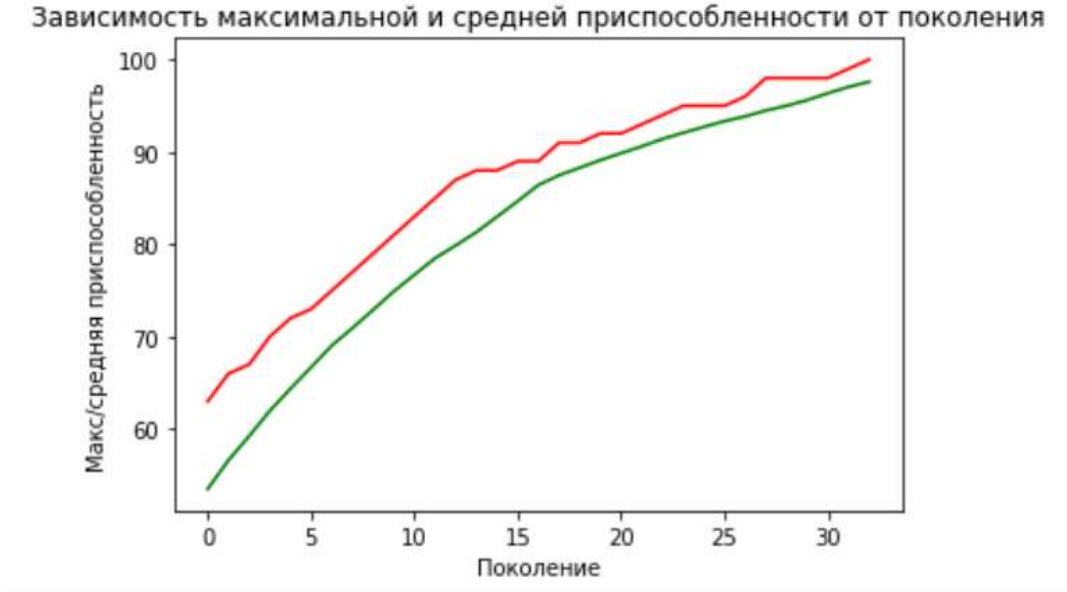
Многие традиционные алгоритмы по природе своей детерминированы, тогда как правила, применяемые генетическими алгоритмами для перехода от предыдущего поколения к следующему, вероятностные.

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.

```
Поколение 33: Макс приспособ. = 100, Средняя приспособ.= 97.59
Лучший индивидуум =  1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```


Зависимость максимальной и средней приспособленности от поколения
Макс/средняя приспособленность
Поколение
100
90
80
70
60
0
5
10
15
20
25
30


Как машины ведут себя при пожаре
Классическое программирование
«Я просчитал все варианты событий и ты сейчас должен связать верёвку из хлебного мякиша»
Машинное обучение
«По статистике люди гибнут в 6% пожаров, поэтому рекомендую вам умереть прямо сейчас»
Обучение с подкреплением
«Да просто беги от огня АААААААА!!!!