Генетические алгоритмы

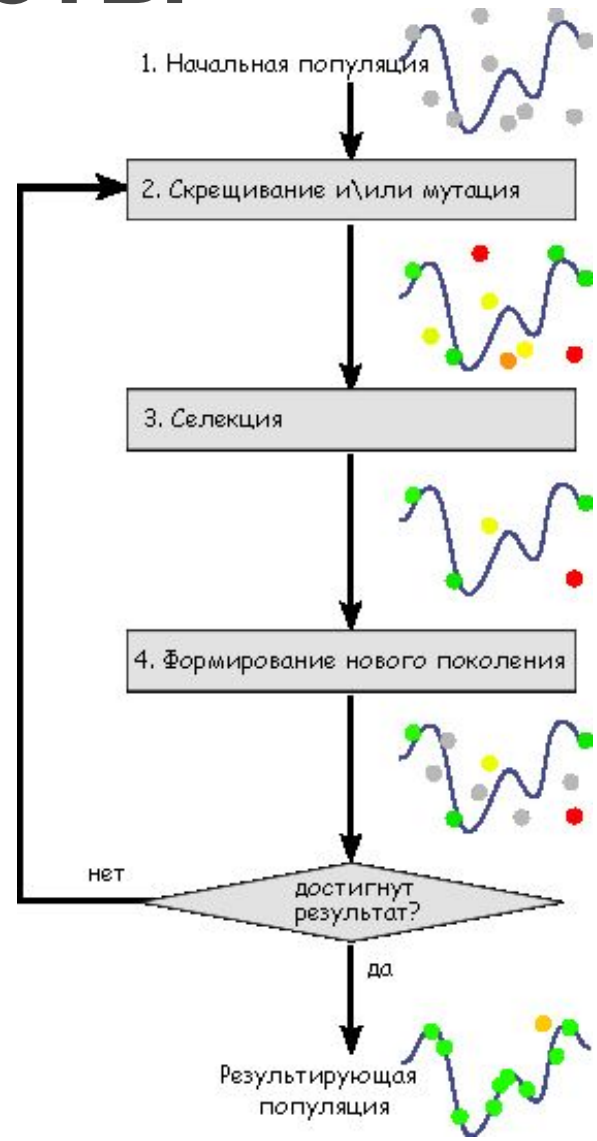
Генетический алгоритм

- эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию.
- Является разновидностью эволюционных вычислений.
- Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

Применение

- Оптимизация функций
- Оптимизация запросов в базах данных
- Разнообразные задачи на графах (задача коммивояжера, раскраска, нахождение паросочетаний)
- Настройка и обучение искусственной нейронной сети
- Задачи компоновки
- Составление расписаний
- Игровые стратегии
- Теория приближений
- Искусственная жизнь
- Биоинформатика (фолдинг белков)

Схема работы



- Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа») генов. Где каждый ген может быть битом, числом или неким другим объектом.
- Некоторым, обычно случайным, образом создаётся множество генотипов начальной популяции. Они оцениваются с использованием «функции приспособленности», в результате чего с каждым генотипом ассоциируется определённое значение («приспособленность»), которое определяет насколько хорошо фенотип им описываемый решает поставленную задачу.

- Из полученного множества решений («поколения») с учётом значения «приспособленности» выбираются решения (обычно лучшие особи имеют большую вероятность быть выбранными), к которым применяются «генетические операторы» (в большинстве случаев «скрещивание» — crossover и «мутация» — mutation), результатом чего является получение новых решений. Для них также вычисляется значение приспособленности, и затем производится отбор («селекция») лучших решений в следующее поколение.
- Этот набор действий повторяется итеративно, так моделируется «эволюционный процесс», продолжающийся несколько жизненных циклов (поколений), пока не будет выполнен критерий остановки алгоритма. Таким критерием может быть:
  - нахождение глобального, либо субоптимального решения;
  - исчерпание числа поколений, отпущенных на эволюцию;
  - исчерпание времени, отпущенного на эволюцию.

Кодирование признаков, представленных целыми числами

- Бинарное кодирование не лишено недостатков. Основной недостаток заключается в том, что соседние числа отличаются в значениях нескольких битов, так например числа 7 и 8 в битовом представлении различаются в 4-х позициях, что затрудняет функционирование генетического алгоритма и увеличивает время, необходимое для его сходимости. Для того, чтобы избежать эту проблему лучше использовать кодирование, при котором соседние числа отличаются меньшим количеством позиций, в идеале значением одного бита. Таким кодом является код Грея

Код Грея

- 0 - 0000
- 1 - 0001
- 2 - 0011
- 3 - 0010
- 4 - 0110
- 5 - 0111
- 6 - 0101
- 7 - 0100
- 8 - 1100
- 9 - 1101
- 10 - 1111
- 11 - 1110

Кодирование признаков

- Самый простой способ – использовать битовое представление. Хотя такой вариант имеет те же недостатки, что и для целых чисел. Поэтому на практике обычно применяют следующую последовательность действий:
- 1.Разбивают весь интервал допустимых значений признака на участки с требуемой точностью.
- 2.Принимают значение гена как целочисленное число, определяющее номер интервала (используя код Грея).
- 3.В качестве значения параметра принимают число, являющиеся серединой этого интервала.

Пример

- Допустим, что значения признака лежат в интервале [0,1]. При кодировании использовалось разбиение участка на 256 интервалов. Для кодирования их номера нам потребуется таким образом 8 бит. Допустим значение гена: 00100101bG (заглавная буква G показывает, что используется кодирование по коду Грея).
- Для начала, используя код Грея, найдем соответствующий ему номер интервала: 25hG->36h->54d.
- Теперь посмотрим, какой интервал ему соответствует… После несложных подсчетов получаем интервал [0,20703125, 0,2109375].
- Значит значение нашего параметра будет (0,20703125+0,2109375)/2=0,208984375.

Определение фенотипа

- для того, чтобы определить фенотип объекта (то есть значения признаков, описывающих объект) нам необходимо только знать значения генов, соответствующим этим признакам, то есть генотип объекта. При этом совокупность генов, описывающих генотип объекта, представляет собой *хромосому*. В некоторых реализациях ее также называют особью. Таким образом, в реализации генетического алгоритма хромосома представляет собой битовую строку фиксированной длины. При этом каждому участку строки соответствует ген. Длина генов внутри хромосомы может быть одинаковой или различной. Чаще всего применяют гены одинаковой длины.

Обобщенный алгоритм

- 1. Задать целевую функцию (приспособленности) для особей популяции
- 2. Создать начальную популяцию
- 3. (Начало цикла)
  - A. Размножение (скрещивание)
  - B. Мутирование
  - C. Вычислить значение целевой функции для всех особей
  - D. Формирование нового поколения (селекция)
  - E. Если выполняются условия останова, то (конец цикла), иначе на шаг 3.

Создание начальной популяции

- Перед первым шагом нужно случайным образом создать начальную популяцию; даже если она окажется совершенно неконкурентоспособной, генетический алгоритм все равно достаточно быстро переведет ее в жизнеспособную популяцию. Таким образом, на первом шаге можно особенно не стараться сделать слишком уж приспособленных особей, достаточно, чтобы они соответствовали формату особей популяции, и на них можно было подсчитать функцию приспособленности (Fitness). Итогом первого шага является популяция H, состоящая из N особей.

Размножение (скрещивание)

- Размножение в генетических алгоритмах обычно половое — чтобы произвести потомка, нужны несколько родителей, обычно два.
- Размножение в разных алгоритмах определяется по-разному — оно, конечно, зависит от представления данных. Главное требование к размножению — чтобы потомок или потомки имели возможность унаследовать черты обоих родителей, «смешав» их каким-либо способом.
- Почему особи для размножения обычно выбираются из всей популяции H, а не из выживших на первом шаге элементов H_0 (хотя последний вариант тоже имеет право на существование)? Дело в том, что главный бич многих генетических алгоритмов — недостаток разнообразия (diversity) в особях. Достаточно быстро выделяется один-единственный генотип, который представляет собой локальный максимум, а затем все элементы популяции проигрывают ему отбор, и вся популяция «забивается» копиями этой особи. Есть разные способы борьбы с таким нежелательным эффектом; один из них — выбор для размножения не самых приспособленных, но вообще всех особей.

Классический оператор скрещивания

- 1.из популяции выбираются две особи, которые будут родителями;
- 2.определяется (обычно случайным образом) точка разрыва;
- 3.потомок определяется как конкатенация части первого и второго родителя.

Пример

- Хромосома_1: 0000000000
- Хромосома_2: 1111111111
- Допустим разрыв происходит после 3-го бита хромосомы, тогда
- Хромосома_1: 0000000000 >> 000 1111111 Результирующая_хромосома_1
- Хромосома_2: 1111111111 >> 111 0000000 Результирующая_хромосома_2
- Затем с вероятностью 0,5 определяется одна из результирующих хромосом в качестве потомка.

Мутации

- К мутациям относится все то же самое, что и к размножению: есть некоторая доля мутантов m, являющаяся параметром генетического алгоритма, и на шаге мутаций нужно выбрать mN особей, а затем изменить их в соответствии с заранее определенными операциями мутации.

- При использовании данного оператора каждый бит в хромосоме с определенной вероятностью инвертируется.
- Кроме того, используется еще и так называемый оператор инверсии, который заключается в том, что хромосома делится на две части, и затем они меняются местами.

Отбор

- На этапе отбора нужно из всей популяции выбрать определенную ее долю, которая останется «в живых» на этом этапе эволюции. Есть разные способы проводить отбор. Вероятность выживания особи h должна зависеть от значения функции приспособленности Fitness(h). Сама доля выживших s обычно является параметром генетического алгоритма, и ее просто задают заранее. По итогам отбора из N особей популяции H должны остаться sN особей, которые войдут в итоговую популяцию H'. Остальные особи погибают.

Вариации операторов

- кроссовер может быть не одноточечный (как было описано выше), а многоточечный, когда формируется несколько точек разрыва (чаще всего две). Кроме того, в некоторых реализациях алгоритма оператор мутации представляет собой инверсию только одного случайно выбранного бита хромосомы.

Алгоритм

1. Инициировать начальный момент времени t=0. Случайным образом сформировать начальную популяцию, состоящую из k особей. $B_0 = \{A_1, A_2, \ldots, A_k\}$
2. Вычислить *приспособленность* каждой особи $F_{Ai} = fit(A_i)$, i=1…k и популяции в целом $F_t = fit(B_t)$ (также иногда называемую термином фиттнес). Значение этой функции определяет насколько хорошо подходит особь, описанная данной хромосомой, для решения задачи.
3. Выбрать особь A_c из популяции. $A_c = Get(B_t)$
4. С определенной вероятностью (вероятностью кроссовера P_c) выбрать вторую особь из популяции $A_{c1} = Get(B_t)$ и произвести оператор кроссовера $A_c = Crossing(A_c, A_{c1})$.
5. С определенной вероятностью (вероятностью мутации P_m) выполнить оператор мутации. $A_c = mutation(A_c)$.
6. С определенной вероятностью (вероятностью инверсии P_i) выполнить оператор инверсии $A_c = inversion(A_c)$.
7. Поместить полученную хромосому в новую популяцию $insert(B_{t+1}, A_c)$.
8. Выполнить операции, начиная с пункта 3, k раз.
9. Увеличить номер текущей эпохи t=t+1.
10. Если выполнилось условие останова, то завершить работу, иначе переход на шаг 2.

Этап отбора

- Наиболее часто используется метод отбора, называемый *рулеткой*. При использовании такого метода вероятность выбора хромосомы определяется ее приспособленностью, то есть $P_{Get(Ai)} \sim Fit(A_i)/Fit(B_t)$. Использование этого метода приводит к тому, что вероятность передачи признаков более приспособленными особями потомкам возрастает.
- Другой часто используемый метод – турнирный отбор. Он заключается в том, что случайно выбирается несколько особей из популяции (обычно 2) и победителем выбирается особь с наибольшей приспособленностью.
- Кроме того, в некоторых реализациях алгоритма применяется так называемая стратегия элитизма, которая заключается в том, что особи с наибольшей приспособленностью гарантировано переходят в новую популяцию. Использование элитизма обычно позволяет ускорить сходимость генетического алгоритма. Недостаток использования стратегии элитизма в том, что повышается вероятность попадания алгоритма в локальный минимум.

Определение критериев останова

- Обычно в качестве них применяются или ограничение на максимальное число эпох функционирования алгоритма, или определение его сходимости, обычно путем сравнивания приспособленности популяции на нескольких эпохах и остановки при стабилизации этого параметра.

Пример на C++

```
# include <iostream>
# include <algorithm>
# include <numeric>
int main()
{
using namespace std;
srand((unsigned)time(NULL));
const int N = 1000;
int a[N]; //заполняем нулями
fill(a, a+N, 0);
for (;;)
{ //мутация в случайную сторону каждого элемента:
for (int i = 0; i < N; ++i)
if (rand()%2 == 1) a[i] += 1; else a[i] -= 1;
//теперь выбираем лучших, отсортировав по возрастанию...
sort(a, a+N);
//... и тогда лучшие окажутся во второй половине массива. //скопируем лучших в первую половину, куда они оставили потомство, а первые умерли:
copy(a+N/2, a+N, a /*куда*/);
//теперь посмотрим на среднее состояние популяции. Как видим, оно всё лучше и лучше.
cout << accumulate(a, a+N, 0) / N << endl;
}
}
```

Непрерывные генетические алгоритмы

Проблемы двоичного кодирования

- двоичное представление хромосом влечет за собой определенные трудности при поиске в непрерывных пространствах большой размерности, и когда требуется высокая точность найденного решения.
- точность представления определяется количеством разрядов, используемых для кодирования одной хромосомы. Поэтому при увеличении N пространство поиска расширяется и становится огромным
- при увеличении длины битовой строки необходимо увеличивать и численность популяции

Вещественное кодирование

- хромосома есть вектор вещественных чисел
- Длина хромосомы будет совпадать с длиной вектора-решения оптимизационной задачи, иначе говоря, *каждый ген будет отвечать за одну переменную*. Генотип объекта становится идентичным его фенотипу.

Преимущества

- Использование непрерывных генов делает возможным поиск в больших пространствах (даже в неизвестных), что трудно делать в случае двоичных генов, когда увеличение пространства поиска сокращает точность решения при неизменной длине хромосомы.
- Одной из важных черт непрерывных ГА является их способность к локальной настройке решений.
- Использование RGA для представления решений удобно, поскольку близко к постановке большинства прикладных задач. Кроме того, отсутствие операций кодирования/декодирования, которые необходимы в BGA, повышает скорость работы алгоритма.

Особенности

- В качестве операторов отбора особей в родительскую пару здесь подходят любые известные из BGA: рулетка, турнирный, случайный.
- Операторы скрещивания и мутации не годятся: в классических реализациях они работают с битовыми строками.

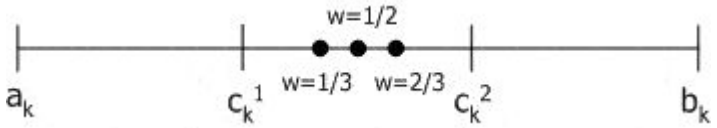
Оператор кроссовера

- Большинство real-coded алгоритмов генерируют новые векторы в окрестности родительских пар.
- Пусть $C_1=(c_1^1,c_2^1,\ldots,c_n^1)$ и $C_2=(c_1^2,c_2^2,\ldots,c_n^2)$ – две хромосомы, выбранные оператором селекции для скрещивания. Предполагается, что $c_k^1<=c_k^2$ и $f(C_1)>=f(C_2)$.

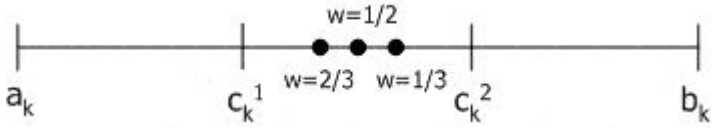
Популярные операторы

- Плоский кроссовер (flat crossover): создается потомок $H=(h_1,\ldots,h_k,\ldots,h_n)$, h_k, $k=1,\ldots,$ n – случайное число из интервала $[c_k^1,c_k^2]$.
- Простейший кроссовер (simple crossover): случайным образом выбирается число k из интервала $\{1,2,\ldots,n-1\}$ и генерируются два потомка $H1=(c_1^1,c_2^1,\ldots,c_k^1,c_{k+1}^2,\ldots,c_n^2)$ и $H_2=(c_1^2,c_2^2,\ldots,c_k^2,c_{k+1}^1,\ldots,c_n^2)$.

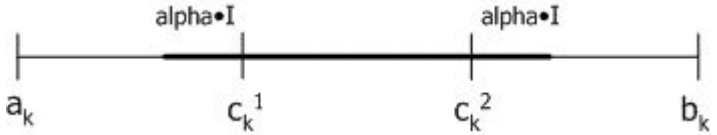
- Арифметический кроссовер (arithmetical crossover): создаются два потомка $H_1=(h_1^1,\ldots,h_n^1)$, $H_2=(h_1^2,\ldots,h_n^2)$, где $h_k^1=w*c_k^1+(1-w)*c_k^2$, $h_k^2=w*c_k^2+(1-w)*c_k^1$, $k=1,\ldots,n$, w либо константа (равномерный арифметический кроссовер) из интервала [0;1], либо изменяется с увеличением эпох (неравномерный арифметический кроссовер).



- Геометрический кроссовер (geometrical crossover): создаются два потомка $H_1=(h_1^1,\ldots,h_n^1)$, $H_2=(h_1^2,\ldots,h_n^2)$, где $h_k^1=(c_k^1)^{w}*(c_k^2)^{1-w}$, $(c_k^2)^{w}*(c_k^1)^{1-w}$, w – случайное число из интервала [0;1].



- Смешанный кроссовер (blend, BLX-alpha crossover): генерируется один потомок $H=(h_1,\ldots,h_k,\ldots,h_n)$, где h_k – случайное число из интервала $[c_{min}-I*alpha, c_{max}+I*alpha]$, $c_{min}=\min(c_k^1,c_k^2)$, $c_{max}=\max(c_k^1,c_k^2)$, $I=c_{max}-c_{min}$. BLX-0.0 кроссовер превращается в плоский



- Линейный кроссовер (linear crossover): создаются три потомка $H_q=(h_1^q,\ldots,h_k^q,\ldots,h_n^q)$, $q=1,2,3$, где $h_k^1=0.5*c_k^1+0.5*c_k^2$, $h_k^2=1.5*c_k^1-0.5*c_k^2$, $h_k^3=-0.5*c_k^1+1.5*c_k^2$. На этапе селекции в этом кроссовере отбираются два наиболее сильных потомка.

- Дискретный кроссовер (discrete crossover): каждый ген h_k выбирается случайно по равномерному закону из конечного множества $\{c_k^1, c_k^2\}$.

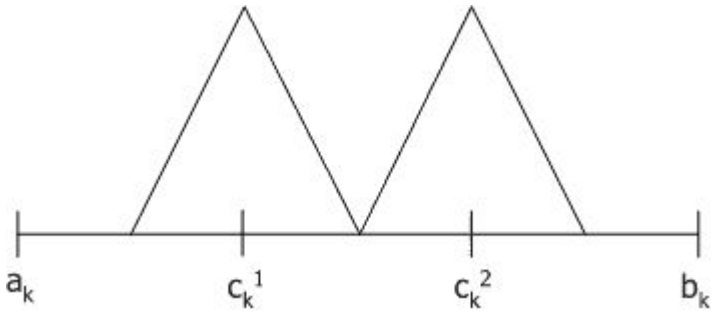
- Расширенный линейчатый кроссовер (extended line crossover): ген $h_k = c_k^1 + w*(c_k^2 - c_k^1)$, w – случайное число из интервала [-0.25;1.25].

- Эвристический кроссовер (Wright's heuristic crossover). Пусть C_1 – один из двух родителей с лучшей приспособленностью. Тогда $h_k = w*(c_k^1 - c_k^2) + c_k^1$, w – случайное число из интервала [0;1].

- Нечеткий кроссовер (fuzzy recombination, FR-d crossover): создаются два потомка $H_1=(h_1^1,\ldots,h_n^1)$, $H_2=(h_1^1,\ldots,h_n^2)$. Вероятность того, что в i-том гене появится число v_i , задается распределением $p(v_i)$ $\{F(c_k^1),F(c_k^2)\}$, где $F(c_k^1),F(c_k^2)$ – распределения вероятностей треугольной формы (треугольные нечеткие функции принадлежности) со следующими свойствами ($c_k^1<=c_k^2$ и $I=|c_k^1-c_k^2|$):

Распределение вероятностей	Минимум	Центр	Максимум
$F(c_k^1)$	c_k^1-d*I	c_k^1	c_k^1+d*I
$F(c_k^2)$	c_k^2-d*I	c_k^2	c_k^2+d*I

- Параметр d определяет степень перекрытия треугольных функций принадлежности, по умолчанию d=0.5



- BLX-кроссовер с параметром alpha=0.5 – превосходит по эффективности большинство простых кроссоверов.

Оператор мутации

- наибольшее распространение получили: случайная и неравномерная мутация (random and non-uniform mutation).
- При случайной мутации ген, подлежащий изменению, принимает случайное значение из интервала своего изменения. В неравномерной мутации значение гена после оператора мутации рассчитывается по формуле:

$$c_k^* = \begin{cases} c_k + \Delta(t, b_k - c_k), & w = 0 \\ c_k - \Delta(t, b_k - c_k), & w = 1 \end{cases},$$

$$\Delta(t, y) = y\left[1 - r^{\left(1 - \frac{t}{e\_max}\right)^b}\right].$$

SBX-кроссовер

- SBX (англ.: Simulated Binary Crossover) – кроссовер, имитирующий двоичный. Был разработан в 1995 году исследовательской группой под руководством K. Deb'a. Как следует из его названия, этот кроссовер моделирует принципы работы двоичного оператора скрещивания.
- Для генерации потомков используется следующий алгоритм, использующий выражение для P(β). Создаются два потомка $H_k=(h_1^k, \ldots, h_j^k, \ldots, h_n^k)$, k=1,2, где , – число, пол$h_j^1 = 0.5 \cdot [(1-\beta_k)c_j^1 + (1+\beta_k)c_j^2]$ формуле:

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{n+1}}, & u(0,1) \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{n+1}}, & u(0,1) > 0.5. \end{cases}$$

- В формуле $u(0,1)$ – случайное число, распределенное по равномерному закону, n [2,5] – параметр кроссовера.
- Эксперименты автора SBX кроссовера показали, что он во многих случаях эффективнее BLX, хотя, очевидно, что не существует ни одного кроссовера, эффективного во всех случаях. Исследования показывают, что использование нескольких различных операторов кроссовера позволяет уменьшить вероятность преждевременной сходимости, т.е. улучшить эффективность алгоритма оптимизации в целом. Для этого могут использоваться специальные стратегии, изменяющие вероятность применения отдельного эволюционного оператора в зависимости от его «успешности», или использование гибридных кроссоверов