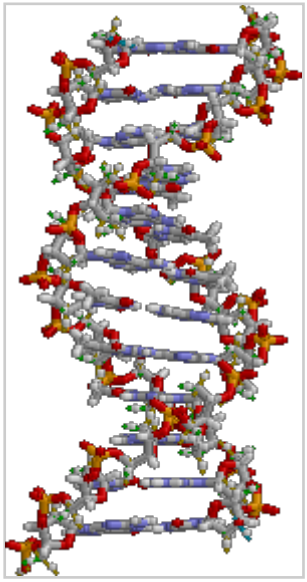
Генетический алгоритм

Материал из MachineLearning.

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



ДНК

Содержание

Постановка задачи

Для **функции приспособленности** $W(x)$ в пространстве поиска X требуется найти $x^* = \arg\max_{x \in X} W(x)$ (или $x^* = \arg\min_{x \in X} W(x)$).

Описание алгоритма

1. Случайным образом генерируется конечный набор пробных решений: $P^1 = \left\{p_1^1 \dots p_n^1\right\},\ p_i^1 \in X$ (первое поколение, n - размер популяции).
2. Оценка приспособленности текущего поколения: $F^k = \left\{f_1^k \dots f_n^k\right\},\ f_i^k = W(p_i^k)$
3. Выход, если выполняется критерий останова, иначе
4. Генерация нового поколения посредством операторов селекции S, скрещивания C и мутаций M: $P^{k+1} = M \cdot C \cdot S(P^k, F^k)$ и переход к пункту 2.

В процессе селекции ~~выживают~~ отбирают только несколько лучших пробных решений, остальные далее не используются. Скрещивание за место пары решений создаёт другую, элементы которой перемешаны каким-то особым образом. Мутация случайным образом меняет какую-нибудь компоненту пробного решения на иную.

Иные обозначения

Несмотря на внушительный возраст, в генетических алгоритма до сих пор используют различную терминологию, проистекающей как из генетики, так и из кибернетики.

Встречаются такие обозначения:

- Функция приспособленности (Fitness) $W(x)$ - целевая функция;
- Особь - пробное решение $p_i^k \in X$;
- Популяция - все поколения, вносящие вклад в последующее. Чаще всего поколение и популяция - синонимы;
- Ген - компонент вектора x пространства поиска X;
- Скрещивание - кроссинговер (crossingover).

Применение генетических алгоритмов

Генетические алгоритмы применяются для решения следующих задач:

1. Оптимизация функций

2. Оптимизация запросов в базах данных
3. Разнообразные задачи на графах (задача коммивояжёра, раскраска, нахождение паросочетаний)
4. Настройка и обучение искусственной нейронной сети
5. Задачи компоновки
6. Составление расписаний
7. Игровые стратегии
8. Теория приближений
9. Искусственная жизнь
10. Биоинформатика (свёртывание белков)

Подробное описание алгоритма

Кодирование пространства поиска

В ГА часто используют следующие типы кодирования компонент пространства поиска:

- Бинарный, если признак сам по себе является бинарным;
- Численный в двоичной системе. Расширенный вариант бинарного, где используется фиксированное число бит. Самый простой в реализации, но имеет существенный недостаток (см. ниже);
- Кодирование кодом Грея (http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B4_%D0%93%D1%80%D0%B5%D1%8F) . Избавляет от проблем предыдущего варианта, но добавляет накладные расходы на кодирование/декодирование;
- С небинарными операторами скрещивания и мутаций:
  - Числа с плавающей запятой. Используются в том случае, когда масштаб изменения признака заранее не известен;
  - Номинальные типы и более абстрактные сущности;
- Типы с автоподстройкой...

Начальная популяция

Начальная популяция генерируется обычно случайно. Единственный критерий - достаточное разнообразие особей, чтобы популяция не свалилась в ближайший экстремум.

Оценка приспособленности

Оценка приспособленности часто проводится в две стадии. Первая - собственно оценка: $F^k = \left\{ f_1^k \dots f_n^k \right\}, \;\; f_i^k = W(p_i^k)$. Вторая - дополнительные преобразования. Например, ею может быть нормировка к виду $F^{k'} = \left\{ f_1^{k'} \dots f_n^{k'} \right\}, \;\; f_i^{k'} = (f_i^k - f_0)/(f_1 - f_0)$, где f_1 и f_0, соответственно, лучший и худший показатели в текущей популяции.

Оператор отбора (селекции)

На этом этапе отбирается оптимальная популяция для дальнейшего

размножения. Обычно берут определённое число лучших по приспособленности. Имеет смысл также отбрасывать "клонов", т.е. особей с одинаковым набором генов.

Оператор скрещивания

Чаще всего скрещивание производят над двумя лучшими особями. Результатом является также обычно две особи с компонентами, взятыми от их родителей. Цель этого оператора - распространение хороших генов по популяции и стягивание плотности популяции к тем областям, где она и так велика в том предположении, что "нас много там, где хорошо".

- В одноточечном варианте, результатом скрещивания родителей $p_i^k = (a_1, a_2, \ldots a_n), \;\; p_j^k = \{b_1, b_2, \ldots b_n\} \in P^k$ в k-ой популяции станут два

элемента популяции $k+1$, такие что $p_i^{k+1} = \{a_1, \ldots, a_c, b_{c+1}, \ldots, b_n\}, \;\; p_j^{k+1} = \{b_1, \ldots, b_c, a_{c+1}, \ldots a_n\} \in P^{k+1}$, где точка c выбирается случайно. В двухточечном варианте, соответственно, точек пересечения будет две, и они также выбираются случайно. Легко расширить эту конструкцию и до n точек. Нужно заметить, что в случае нечётного n, происходит n+1-точечный кроссинговер с n+1-ой точкой между последней и первой компонентами.

- Скрещиванием с маской является результат в виде двух потомков с компонентами, принадлежность которых определяется по битовой маске. Т.е. результатом скрещивания родителей $p_i^k = (a_1, a_2, \ldots, a_n), \;\; p_j^k = \{b_1, b_2, \ldots, b_n\} \in P^k$ в k-ой популяции станут два

элемента популяции $k+1$, такие что $p_i^{k+1} = (c_1, \ldots, c_n), \;\; p_j^{k+1} = \{d_1, \ldots, d_n\} \in P^{k+1}$, где $c_i = a_i$ при $m_i = 0$ и $c_i = b_i$ при $m_i \neq 0$, и противоположные условия для второго отпрыска. Маска выбирается случайно. Для простоты, ею может быть третья особь.

- В непрерывном пространстве можно ввести такую аналогию для скрещивания:

$P^{k+1}(x) = \int_X dy P^k(x) P^k(y) \exp\left[-\frac{(x-y)^T M_c (x-y)}{\rho(x,y)}\right]$, где $P^k(x)$ - плотность генофонда к-ой популяции, $\rho(x,y)$ - расстояние между двумя особями с генами x и y, M_c - матрица силы скрещивания.

Оператор мутаций

Оператор мутаций просто меняет произвольное число элементов в особи на другие произвольные. Фактически он является неким диссипативным элементом, с одной стороны вытягивающим из локальных экстремумов, с другой - приносящим новую информацию в популяцию.

- Инвертирует бит в случае бинарного признака.
- Изменяет на некоторую величину числовой признак. Причём, скорее на ближайший.
- Заменит на другой номинальный признак.
- В непрерывном пространстве можно ввести следующую аналогию:

$P^{k+1}(x) = \int_X dy P^k(y) \exp\left[-(x-y)^T M_m (x-y)\right]$, где $P^k(x)$ - плотность генофонда к-ой популяции, M_c - матрица силы мутаций.

Критерии останова

- нахождение глобального, либо субоптимального решения;
- выходом на «плато»;
- исчерпание числа поколений, отпущенных на эволюцию;
- исчерпание времени, отпущенного на эволюцию;
- исчерпание заданного числа обращений к целевой функции.

Эвристики

Генетические алгоритмы богаты возможностями встраивания различных эвристик. До сих пор не существует (и не будет!) точных критериев оптимального размера популяции, способов мутаций и скрещивания, выбора начальной популяции и т.п.

Плоидность

Каждая особь состоит не из одного, а нескольких пробных решений. Каждый кратный элемент пробного решения имеет активный(доминантный) или неактивный(рецессивный) статус, и, тем самым, проявляет или не проявляет (или же проявляет с определённой интенсивностью) себя при вычислении целевой функции. Кратность пробных решений в особи называется плоидностью (2 - диплоидный набор, 3 - триплоидный, n - n-плоидный набор).

Преимущества:

- Вытягивает популяцию из локального экстремума, т.к:
  - Поддерживает генетическое разнообразие, не позволяет вырождаться;
  - Позволяет естественным путём воссоздать аналог инцеста.

Недостатки:

- Усложнение алгоритма;
- Требует большего числа итераций до схождения в экстремум.

Отдельная статья: плоидность

Мета ГА

....

Простейший пример ГА

Подбор ключа 2048 бит

```
#include <stdio.h>
#include <stdlib.h>

#include <iostream>
#include <algorithm>
#include <vector>

using namespace std;

#define KEY_LEN 2048 // Длина ключа
bool key[KEY_LEN]; //Неизвестный ключ

#define POP_SIZE 20 // Размер популяции
#define ELITE 5 // Количество привелигированных особей
struct gen_struct { // Особь, кортеж из генов и значения пригодности
 bool gen[KEY_LEN];
 int fitness;
 gen_struct(){ //Инициализация случайными значениями
  for(int i=0;i<KEY_LEN;++i)
   gen[i]=(random()%2==0);
 }
 bool operator<(const gen_struct & g)const{
  return (g.fitness>fitness);
 }
 gen_struct(const gen_struct& g){
  for(int i=0;i<KEY_LEN;++i) gen[i]=g.gen[i];
  fitness=g.fitness;
 }
 void Mutate(){ //Оператор мутаций
  gen[random()%KEY_LEN]=random()%2;
 }
 void Crossingover(gen_struct & g){ //Оператор скрещивания
  for(int i=0;i<KEY_LEN;++i){
   if(random()%2){
    gen[i]=g.gen[i];
   }
  }
 }
};

int Fitness(bool gen[]){ //Целевая функция
 int ret=0;
 for(int i=0;i<KEY_LEN;i++){
  ret+=(key[i]!=gen[i]);
 }
 return ret;
}


void GA(gen_struct &ret){
 vector<gen_struct> pop;
 pop.resize(POP_SIZE);
 while(1){
  for(int i=0;i<POP_SIZE;++i)
   if((pop[i].fitness=Fitness(pop[i].gen))==0){
      ret=pop[i];
      return;
     }
  sort(pop.begin(), pop.end());
  for(int i=ELITE;i<POP_SIZE;++i){// 5 лучших остаются без изменений
```

```
    pop[i].Crossingover(pop[random()%(ELITE)]);
    pop[i].Mutate();
   }
  }
}


int main(){
  for(int i=0;i<KEY_LEN;i++){
    key[i]=(random()%2==0);
  }
  gen_struct ret;
  GA(ret);
  cout << "key=";
  for(int i=0;i<KEY_LEN; i++){
    cout << ret.gen[i];
  }
  cout << endl;
}
```

Эффективность генетических алгоритмов

Холланд недвусмысленно пишет[1] (http://qai.narod.ru/Papers/holland_2000.html) , что при прочих равных условиях ГА будет работать хуже, чем специальный алгоритм, рассчитанный на конкретную задачу (тип признаков, целевую функцию). Например, полный перебор конечного небольшого пространства или любой эффективных алгоритм спуска будет всегда эффективнее чем ГА. Тем не менее, в ситуации, когда о задаче ничего a priori не известно, можно полагаться на результат работы простейшего генетического алгоритма как некого приближения.

Существуют некоторый класс функций (Hyperplane-defined functions, Holland), с которым за приемлемое время[2] (http://qai.narod.ru/Papers/holland_2000.html) справляются только генетические алгоритмы.

Тестовые функции

В процессе разработки эффективной реализации генетического алгоритма появляется необходимость простой тестовой функции, которая

1. учитывает все сильные стороны ГА (многомерная, многоэкстремальная и т.п.);
2. имеет аналитическое точное решение.
3. проста в реализации
4. её проблематично "взломать"

Самая банальная, сферическая функция, $W(x)=\sum_{i=1}^{N}(x_i-x_i^*)^2$ моментально решается любым алгоритмом спуска. Многоэкстремальная $W(x)=\sum_{i=1}^{N}\left[1-\cos\left(x_i-x_i^*\right)+0.001\left(x_i-x_i^*\right)^2\right]$. Подобных функций можно придумать(и было придумано) большое количество, но малое число из них

удовлетворяет нужным требованиям. Холландером были предложены функции HDR[3] (http://qai.narod.ru/Papers/holland_2000.html) , которые изначально строятся с целью обеспечения всех выдвинутых условий.

Мнения

Конвергенция с генетикой и синтетической теории эволюции

Прообраз генетического алгоритма пришёл из биологии и, несомненно, продолжает «подсматривать» оттуда ответы на многие вопросы, возникающие при проектировании эффективных реализаций генетических алгоритмов. Более того, идеи по оптимизации ГА находят подтверждение и у биологов. Например, такие явления как га- ди- и квадру- плоидные наборы хромосом, инцест, удвоение гена, управление скоростью мутаций, триггеры и т.п. Тем не менее, математика и кибернетика позволяет выйти за пределы химической реальности клетки и представить n-плоидный набор хромосом, объектные сущности вместо генов и т.п.

Нейронные сети и эволюционное моделирование

ИНС и ГА часто пытаются применять в тандеме, т.к. и те и другие имеют корни из биологии. Тем не менее (см. выше об эффективности), во-первых алгоритм обратного распространения ошибки настройки ИНС работает значительно (на порядок) быстрее в простых случаях. А во-вторых, настройка нейронной сети в биологии происходит методом, который, скорее всего, значительно разнится с генетическим подходом.

Аналогия с другими алгоритмами

В случае отключённого кроссинговера ГА начинает вести себя по образу случайного поиска. Также у ГА есть аналогия со случайным поиском с адаптацией. Во многих стохастических алгоритмах можно найти аналогию с ГА.

Преимущества ГА

- Большое число свободных параметров, позволяющим эффективно встраивать эвристики;
- Эффективное распараллеливание;
- Работает заведомо не хуже абсолютно случайного поиска;
- Связь с биологией, дающая некоторую надежду на исключительную эффективность ГА в природе.

Недостатки ГА

- Большое количество свободных параметров, которое превращает "работу с ГА" в "игру с ГА";
- Недоказанность сходимости;
- В простых целевых функциях (гладкие, один экстремум и т.п.) генетика

всегда проигрывает по скорости простым алгоритмам поиска.

Читайте также

- Теорема схемы

...

Ссылки

Генетические алгоритмы и не только (http://qai.narod.ru/)
Генетические алгоритмы на basegroup.ru (http://basegroup.ru/library/optimization)
GAUL: Genetic Algorithm Utility Library — нетривиальная обобщенная свободная реализация GA (http://gaul.sourceforge.net/documentation.html)
Демон Дарвина... (http://neuroschool.narod.ru/books/evogrbn.html) -- Отлично рассмотрен с точки зрения математики накопительный эффект от рецессивности (см. выше "плоидность");
Г. К. Вороновский и др., *Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности* (http://masters.donntu.edu.ua/2005/fvti/trubarov/library/kpi_ga.pdf) -- пример положительного эффекта диплоидности на качество схождения ГА в глобальном экстремуме.

Источник — «http://www.machinelearning.ru/wiki/index.php?title=%D0%93%D0%B5%D0%BD%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC»

Категории: Эволюционные алгоритмы | Дискретная оптимизация
