



@unciiia

Пользователь

2 апреля 2015 в 17:25

Генетический алгоритм — наглядная реализация

из песочницы

Программирование*, Математика*

Года четыре назад, в универе услышал о таком методе оптимизации, как генетический алгоритм. О нем везде сообщалось ровно два факта: он клёвый и он не работает. Вернее, работает, но медленно, ненадежно, и нигде его не стоит использовать. Зато он красиво может продемонстрировать механизмы эволюции. В этой статье я покажу красивый способ вживую посмотреть на процессы эволюции на примере работы этого простого. Нужно лишь немного математики, программирования и все это приправить воображением.

Кратко об алгоритме

Итак, что же такое генетический алгоритм? Это, прежде всего, метод многомерной оптимизации, т.е. метод поиска минимума многомерной функции. Потенциально этот метод можно использовать для глобальной оптимизации, но с этим возникают сложности, опишу их позднее.

Сама суть метода заключается в том, что мы модулируем эволюционный процесс: у нас есть какая-то популяция (набор векторов), которая размножается, на которую воздействуют мутации и производится естественный отбор на основании минимизации целевой функции. Рассмотрим подробнее эти процессы.

Итак, прежде всего наша популяция должна **размножаться**. Основной принцип размножения — потомок похож на своих родителей. Т.е. мы задаем какой-то механизм наследования. И лучше будет, если он будет включать элемент случайности. Но скорость развития таких систем очень низкая — разнообразие генетическое падает, популяция вырождается. Т.е. значение функции перестает минимизироваться.

Для решения этой проблемы был введен механизм **мутации**, который заключается в случайном изменении каких-то особей. Этот механизм позволяет привнести что-то новое в генетическое разнообразие.

Следующий важный механизм — **селекция**. Как было сказано, селекция — отбор особей (можно из только родившихся, а можно из всех — показывает, что это не играет решающую роль), которые лучше минимизируют функцию. Обычно отбирают столько особей, сколько было до размножения, чтобы из эпохи в эпоху у нас было постоянное количество особей в популяции. Также принято отбирать «счастливиц» — как число особей, которые, возможно, плохо минимизируют функцию, но зато внесут разнообразия в последующие поколения.

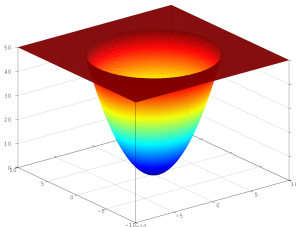
Этих трех механизмов чаще всего недостаточно, чтобы минимизировать функцию. Так популяция вырождается — рано или поздно локальный минимум забивает своим значением всю популяцию. Когда такое происходит, проводят процесс, называемый **встряской** (в природе аналогии глобальные катаклизмы), когда уничтожается почти вся популяция, и добавляются новые (случайные) особи.

Вот описание классического генетического алгоритма, он прост в реализации и есть место для фантазии и исследований.

Постановка задачи

Итак, когда я уже решил, что хочу попробовать реализовать этот легендарный (пусть и неудачливый) алгоритм, речь зашла о том, что же я буду минимизировать? Обычно берут какую-нибудь страшную многомерную функцию с синусами, косинусами и т.д. Но это не очень интересно и не наглядно. Пришла одна незатейливая идея — для отображения многомерного вектора отлично подходит изображение, где значение отвечает яркость. Таким образом, мы можем ввести простую функцию — расстояние до нашего целевого изображения, измеряемое в разности яркости пикселей. Для простоты и скорости я взял изображения с яркостью 0, либо 255.

С точки зрения математики такая оптимизация — сущий пустяк. График такой функции представляет собой огромную многомерную «яму» (ка трехмерный параболоид на рисунке), в которую неизбежно скатишься, если идти по градиенту. Единственный локальный минимум является глобальным.



Проблема только в том, что уже близко к минимуму количество путей, по которым можно спуститься, сильно сокращается, а всего у нас столько направлений, сколько измерений (т.е. количество пикселей). Очевидно, что решать эту задачу при помощи генетического алгоритма не стоит, но мы можем посмотреть интересные процессы, протекающие в нашей популяции.

Реализация

Были реализованы все механизмы, описанные в первом параграфе. Размножение проводилось простым скрещиванием случайных пикселей от «мамы» и от «папы». Мутации производились путем изменения значения случайного пикселя у случайной особи на противоположное. А встряска производилась, если минимум не меняется на протяжении пяти шагов. Тогда производится «экстремальная мутация» — замена происходит более интенсивно, чем обычно.

В качестве исходных картинок я брал нонограммы («японские сканворды»), но, по правде говоря, можно брать просто черные квадраты — нет абсолютно никакой разницы. Ниже показаны результаты для нескольких изображений. Здесь для всех, кроме «домика», количество мутаций 100 в среднем на каждую особь, особей в популяции было 100, при размножении популяция увеличивалась в 4 раза. Счастливиц было 30 в каждой эпохе. Для домика значения были выбраны меньшие (30 особей в популяции, мутаций по 50 на особь).



Экспериментально я установил, что использование «счастливчиков» в селекции понижает скорость стремления популяции к минимуму, но зато помогает выбираться из стагнации — без «счастливчиков» стагнация будет постоянна. Что можно увидеть из графиков: левый график — развитие популяции «фараона» со счастливыми, правый — без счастливых.

Таким образом, мы видим, что этот алгоритм позволяет решить поставленную задачу, пусть и за очень долгое время. Слишком большое количество встрясок, в случае больших изображений, может решить большее количество особей в популяции. Оптимальный подбор параметров для разных размерностей я оставляю за рамками данного поста.

Глобальная оптимизация

Как было сказано, локальная оптимизация — задача довольно тривиальная, даже для многомерных случаев. Гораздо интереснее посмотреть, как будет алгоритм справляться с глобальной оптимизацией. Но для этого нужно сначала построить функцию со множеством локальных минимумов, это в нашем случае не так сложно. Достаточно брать минимум из расстояний до нескольких изображений (домик, динозаврик, рыбка, корабли). Тогда первоначальный алгоритм будет «скатываться» в какую-то случайную ямку. И можно просто запускать его несколько раз.

Но есть более интересное решение данной проблемы: можно понять, что мы скатились в локальный минимум, сделать сильную встряску (или вообще инициировать особи заново), и в дальнейшем добавлять штрафы при приближении к известному минимуму. Как видно, картинки чередуются. Замечу, что мы не имеем права трогать исходную функцию. Но мы можем запоминать локальные минимумы и самостоятельно догонять их.

На этой картинке изображен результат, когда при достижении локального минимума (сильная стагнация), популяция просто вымирает.

Здесь популяция вымирает, и добавляется небольшой штраф (в размере обычного расстояния до известного минимума). Это сильно снижает вероятность повторов.

Более интересно, когда популяция не вымирает, а просто начинает подстраиваться под новые условия (след. рисунок). Это достигается при штрафе в виде $0.000001 * \text{sum}^4$. В таком случае, новые образы становятся немного зашумлены:

Этот шум устраняется путем ограничения штрафа в $\max(0.000001 * \text{sum}^4, 20)$. Но мы видим, что четвертого локального минимума (динозавров) достичь не удастся — скорее всего, потому, что он слишком близко расположен к какому-то другому.

Биологическая интерпретация

Какие же выводы мы можем сделать из, не побоюсь этого слова, моделирования? Прежде всего, мы видим, половое размножение — важнейший двигатель развития и приспособляемости. Но только его не достаточно. Роль случайных, маленьких изменений чрезвычайно важна. Именно они обеспечивают возникновение новых видов животных в процессе эволюции, а у нас обеспечивает разнообразие популяции.

Важнейшую роль в эволюции Земли играли природные катаклизмы и массовые вымирания (вымирания динозавров, насекомых и т.д. — крупнейшее было около десяти — см. диаграмму ниже). Это было подтверждено и нашим моделированием. А отбор «счастливчиков» показал, что слабые организмы на сегодня способны в будущем стать основой для последующих поколений.

Как говорится, все как в жизни. Этот метод «сделай эволюцию сам» наглядно показывает интересные механизмы и их роль в развитии. Конечно, существует много более стоящих эволюционных моделей (основанных, конечно, на диффузах), учитывающих больше факторов, более приближенных к жизни. Конечно, существуют более эффективные методы оптимизации.

P.S.

Писал программу на Matlab (вернее, даже на Octave), потому что тут все — голые матрицы, и есть инструменты для работы с картинками. Исходный код прилагается.

[Исходный код](#)

генетический алгоритм, многомерная оптимизация, эволюция, matlab

↑ +28 ↓

👁 51,4k ★ 233



@unciiia

карма
рейтинг
11,0
0,0

ПОХОЖИЕ ПУБЛИКАЦИИ

13 февраля 2012 в 22:53

Генетические алгоритмы. От теории к практике

+32

35k

239

36

20 сентября 2011 в 10:57

Генетический алгоритм. Просто о сложном

+36

127k

287

40

7 января 2011 в 21:31

Генетические алгоритмы в MATLAB

+42

32,3k

78

19

САМОЕ ЧИТАЕМОЕ

Разра

Сейчас

Сутки

Неделя

Месяц

Как стать веб-разработчиком в 2017 году — план действий

+32

19k

354

56

Опасайтесь прозрачных пикселей

+102

13,2k

88

39

Трагедия стопроцентного покрытия кода

+20

2k

11

2

Твоя идея — ерунда

+55

16,5k

53

50

Борьба с перехватом HTTPS-трафика. Опыт Яндекс.Браузера

+11

1,9k

12

6

Комментарии (12)

rma4ok

2 апреля 2015 в 18:48

#

Эта статья №4 в Google по запросу «голимые матрицы»

НЛО прилетело и опубликовало эту надпись здесь

OlegUV

2 апреля 2015 в 20:37

#

Хотелось бы больше деталей, а то получается «Берём песок, старый аккумулятор и немного цветмета — и вот у нас готова хрустальная люстра». Что есть ос она записывается на бумаге символами, какой вообще аппарат аналитических выкладок, как происходит мутация — запись формулами и т.д. Всего этого не : а почитал бы с большим удовольствием.

uncīia

2 апреля 2015 в 21:27 (комментарий был изменён)

#

h

↑

Особь — это просто вектор, содержащий нули или единицы. Мы показываем его в виде изображения. Как раз гифки показывают лучшую особь в каждой э «Мутации производились путем изменения значения случайного пикселя у случайной особи на противоположное». Как говорится, берем и меняем. Что касается аналитических выкладок — не очень понял, о чем вы.

OlegUV

3 апреля 2015 в 07:15

#

h

↑

Вот, уже понятнее! Но всё равно, вопросов масса, просто идём по тексту дальше, и по ходу:

> Здесь для всех, кроме «домика», количество мутаций было 100 в среднем на каждую особь, особей в популяции было 100, при размножении популя

увеличивалась в 4 раза.

Как особи размножаются?

У одной пары родителей одна дочерняя особь или может быть несколько?

Один родитель участвует в нескольких парах или нескольких?

При каких условиях особь выпадает из процесса, т.е. умирает?

и т.д.

В тексте полно мест, вызывающих массу вопросов у людей не знакомых с генетическими алгоритмами...

Обязательно продолжайте писать про генетику — это очень интересно.



uncilia 3 апреля 2015 в 08:21 # 1

Во все процессы добавлен элемент случайности. Так, при размножении выбирается $4 \cdot N$ раз пара особей (случайно), и у них получается детеныш. Получается он путем случайного скрещивания каких-то элементов векторов — в таком дискретном пространстве это есть аналог среднего арифметического. Опять же мера скрещивания (в какой мере детеныш будет похож на одного из родителей) определяется случайно. Все распределения равномерны.

Далее просто ведется сортировка и отбирается 70% из «топа». Остальные 30% забиваются случайно — я показал, что такие «счастливчики» очень везучи, если мы реализуем вставку.

Если что-то еще не понятно, спрашивайте.

bya 3 апреля 2015 в 05:52 (комментарий был изменён) #



Автор не написал самое главное. Но это не его вина. Практически все про это не пишут.

Самое сложное и нетривиальное (в отличие от самого алгоритма) это правильно выбрать генетический код.

1. Генетический код не обязан представлять решение задачи. Из него достаточно просто (относительно объема вычислений) должно строиться решение и желательно, но не обязательно, однозначно.

2. Кроссовер на генетическом коде должен давать решение близкое, в смысле требуемого экстремума, к решениям построенным для его родителей.

В качестве примера избитая задача коммивояжера. Те, кто выбирают в качестве кода порядок обхода городов глубоко не правы, поскольку такой генетически удовлетворяет 2 пункту ни в коем разе.



uncilia 3 апреля 2015 в 08:30 (комментарий был изменён) # 1

У меня как раз получается задача ставится очень просто: устроить генетический код к некоторому значению. Я сделал так, что вот есть лучший экземпляр, можно как-то стремиться, а уж по каким требованиям он является лучшим — это не моя забота. Задача очень формализована.

Кроссовер тут является аналогом среднего арифметического. Гарантировать, что решение ближе, чем решения родителей, естественно, нельзя. Но все-таки видим, что популяция сходится, и движение по графику всегда идет вниз.

nemilya 3 апреля 2015 в 09:14 #

Я вот просто размышляю. Если мы принимаем случайные процессы в развитии, но в любой момент времени мы должны видеть весь спектр разнообразных вариантов.

Если мы ждем появления буквы А — то ведь все другие варианты тоже должны присутствовать.



uncilia 3 апреля 2015 в 09:56 # 1

Тут случайности контролируемые. Т.е. на каждом шаге у нас все ближе и ближе к искомой картинке. Тут нет полного перебора всех вариантов.

iroln 3 апреля 2015 в 14:11 (комментарий был изменён) #

Раз уж реализовали на языке Matlab, не пробовали забавы ради сравнить свою реализацию с реализацией ГА из Global Optimization Toolbox? Вот документация:

www.mathworks.com/help/gads/ga.html

www.mathworks.com/help/gads/genetic-algorithm.html

Я когда-то возился с ГА, ради интереса пробовал использовать его для оптимизации параметров стегаграфического контейнера, но практического смысла не узрел. Надёжнее использовать какой-нибудь pattern search метод, вроде такого:

www.mathworks.com/help/gads/particle-swarm.html

www.norg.uminho.pt/aivaz/pswarm/

optimizer 11 апреля 2015 в 08:54 #

я интересовался этой темой в конце 90-х начале 2000-х, и тогда не попадался на глаза прием со «счастливчиками», наоборот, протаскивали «элит» (по блату сказать). Прям прогресс в демократии :)

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.

Какой тар быстрее, и есть ли альтернатива Judy

284 2 6

Реверс-инжиниринг «Казаков», часть последняя: второе дыхание

841 10 2

«Дальше не придумали, импровизируй» или Agile в информационной безопасности




337 5 1

Использование модулей C++ в Visual Studio 2017

522 8 2

Трагедия стопроцентного покрытия кода

1,9k 11 1

Аккаунт	Разделы	Информация	Услуги	Приложения
Войти	Публикации	О сайте	Реклама	<div> </div>
Регистрация	Хабы	Правила	Тарифы	
	Компании	Помощь	Контент	
	Пользователи	Соглашение	Семинары	
	Песочница	Помощь стартапам		
<div> © 2006 – 2017 «TM»</div>		Служба поддержки	Мобильная версия	