Титульник

# Реферат

Реферат включает в себя три части:

− сведения об объеме текста, количестве и характере иллюстраций, количестве использованных источников;

− перечень ключевых слов, выражающих отдельные понятия, существенные для данной предметной области (например: СИСТЕМА АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ, ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ);

− основной текст, раскрывающий:

а) сущность работы: цель, объект и методы исследования;

б) конкретные сведения о том, что разработано и каким путем;

в) краткие выводы относительно источников экономической эффективности, возможности и области использования.

Оглавление

[Реферат 2](#_Toc421188730)

[Введение 5](#_Toc421188731)

[1. Краткий обзор 7](#_Toc421188732)

[2. Введение в генетические алгоритмы 9](#_Toc421188733)

[3. Проблема представления в генетических алгоритмах 13](#_Toc421188734)

[4. Обзор генетического программирования 14](#_Toc421188735)

[5. Подробное описание генетического программирования 19](#_Toc421188736)

[5.1. Адаптируемые структуры 19](#_Toc421188737)

[5.1.1. Замкнутость множеств 20](#_Toc421188738)

[5.1.2. Достаточность множеств 22](#_Toc421188739)

[5.1.3. Универсальность выбора функций и терминалов 22](#_Toc421188740)

[5.2. Начальные структуры 22](#_Toc421188741)

[5.3. Фитнес 26](#_Toc421188742)

[5.3.1. Исходный фитнес 28](#_Toc421188743)

[5.3.2. Стандартизованный фитнес 29](#_Toc421188744)

[5.3.3. Отрегулированный фитнес 30](#_Toc421188745)

[5.3.4. Нормализованный фитнес 31](#_Toc421188746)

[5.4. Главные операции изменения структур 31](#_Toc421188747)

[5.4.1. Репродукция 31](#_Toc421188748)

[5.4.2. Скрещивание 31](#_Toc421188749)

[5.5. Второстепенные операции изменения структур 32](#_Toc421188750)

[5.5.1. Мутация 32](#_Toc421188751)

[5.5.2. Перестановка 33](#_Toc421188752)

[5.5.3. Редактирование 33](#_Toc421188753)

[5.5.4. Инкапсуляция 33](#_Toc421188754)

[5.5.5. Прореживание 33](#_Toc421188755)

[6. Практический результат 34](#_Toc421188756)

[Заключение 37](#_Toc421188757)

[Список использованных источников 38](#_Toc421188758)

# Введение

Символьная регрессия – задача нахождения формулы, которая описывает некую зависимость. Более формально, её можно сформулировать как построение регрессионной модели в виде суперпозиции заданных функций.

Актуальность выбранной темы обосновывается тем, что символьная регрессия может использоваться для получения эмпирических зависимостей на основе экспериментальных данных. Она широко применяется для решения задач моделирования и прогнозирования.

Аналоги. Символьная регрессия отличается от традиционной линейной, квадратичной или полиномиальной регрессии, которые просто находят числовые коэффициенты для функции, вид которой заранее известен.

Проблема. Какая функциональная зависимость существует между двумя разными наборами данных? Другими словами, существует два набора данных. Есть предположение, что они зависят друг от друга. Необходимо найти вид функциональной зависимости.

Цель – нахождение функции, наиболее оптимально устанавливающей, согласно некоторым критериям (нормальность, гладкость), зависимость двух наборов данных друг от друга.

Задачи:

1. Изучить соответствующую литературу.

2. Разработать алгоритм нахождения функциональной зависимости.

3. Реализовать программу по данному алгоритму.

4. Протестировать программу с реальными данными.

5. Обобщить полученные результаты и сделать соответствующие выводы.

Объектом исследования выступает алгоритм установления функциональной зависимости.

# Краткий обзор

В природе биологические структуры, которые наиболее успешно борются со своей окружающей средой, выживают и размножаются с более высокой скоростью. Биологи понимают структуры, которые они наблюдают в природе, как следствие естественного отбора по Дарвину, действующего в среде в течение промежутка времени. Другими словами, в природе структура – это следствие пригодности. То есть пригодность порождает структуру, основываясь на естественном отборе, а также с помощью половой рекомбинации (генетического скрещивания) и мутации.

Символьная регрессия заключается в нахождении математического выражения, в символьной форме, которое обеспечивает хорошее, лучшее или совершенное соответствие между заданной конечной выборкой значений независимых переменных и связанной выборкой значений зависимых переменных. То есть, символьная регрессии заключается в нахождении модели, которая соответствует заданной выборке данных.

Когда переменные являются вещественными числами, символьная регрессия включает в себя одновременно и поиск вида функции, и поиск числовых коэффициентов модели. Этим символьная регрессия отличается от традиционной линейной, квадратичной или полиномиальной регрессии, которые просто находят числовые коэффициенты для функции, вид которой заранее известен.

Поиск математического выражения в символьной форме может рассматриваться в качестве компьютерной программы, которая принимает значения независимых переменных в качестве входных параметров и вычисляет значения зависимых переменных на выходе.

В случае зашумленности данных проблему нахождения функциональной зависимости данных часто называют эмпирическим открытием. Если независимые переменные находятся в промежутке неотрицательных чисел, то символьную регрессию часто называют последовательной индукцией. Если есть несколько зависимых переменных, процесс называется символьной множественной регрессией.

# Введение в генетические алгоритмы

В природе эволюция происходит при наличии следующих условий:

* Организм способен воспроизводить себя;
* Существует популяция таких способных к размножению особей;
* Есть некоторое разнообразие организмов;
* Некоторые различия в способности выжить связаны с этим разнообразием.

В естественной среде разнообразие обеспечивается различием хромосом особей популяции. Это различие преобразуется в изменение структуры организма и его поведения в среде. А это в свою очередь влияет на способность выживания в среде и скорость размножения. Организмы, которые способны лучше выполнять задачи в своей среде, чаще выживают и чаще размножаются, в отличие от менее пригодных особей. Эта концепция естественного отбора и выживания сильнейших была описана Чарльзом Дарвином в книге «О происхождении видов путем естественного отбора» (1859). С течением времени это приводит к тому, что в популяции остаются только особи с такими структурами организма и поведением, которые позволяют им выживать и производить себе подобных. Таким образом, структура особей в популяции меняется из-за естественного отбора. Когда мы видим ощутимые различия в структуре, возникшие из-за разницы в пригодности особей, то говорим, что популяция эволюционировала.

Когда у нас есть популяция особей, то наличие различий, дифференцированно влияющих на способность выжить, почти неизбежно. Поэтому на практике достаточно только первого условия для начала эволюции.

Книга Джона Холланда «Adaptation in Natural and Artificial Systems» (1975) дала основу для наблюдения за всеми адаптивными системами, а затем показала, как эволюционный процесс может быть применен к искусственным системам. Любая проблема адаптации может быть сформулирована в генетических терминах. А после формулировки такая проблема может быть решена генетическим алгоритмом.

Генетический алгоритм симулирует эволюционные процессы Дарвина и природные генетические операции с хромосомами.

Генетический алгоритм является параллельным математическим алгоритмом, который преобразует набор отдельных математических объектов (как правило, символьных строк фиксированной длины), каждый из которых связан с соответствующим значением фитнес-функции (пригодности), в новую популяцию (следующее поколение) с помощью операций, основанных на концепции выживания сильнейших по Дарвину и природных генетических операций (в частности, половой рекомбинации).

Почему генетический алгоритм работает?

На первый взгляд кажется, что тестирование случайным образом созданных строк не даст ничего, кроме значения фитнеса для этих проверяемых точек.

С помощью полученных значений фитнеса можно узнать средний фитнес популяции. Это оценка средней пригодности пространства поиска. Она имеет статистическую дисперсию, т.к. это не среднее значение всех точек пространства поиска, а лишь расчет, основанный на тестируемых точках.

После получения среднего значения фитнеса мы иначе смотрим на проверенные строки популяции. Теперь можно увидеть, какие строки лучше, и насколько лучше остальных они решают заданную проблему.

Теперь необходимо решить, что делать дальше.

Одним из вариантов может быть продолжение случайного выбора точек пространства поиска и проверка их пригодности. Но случайный слепой поиск стратегии не адаптивен и не интеллектуален в том смысле, что мы не используем полученную информацию об окружающей среде, чтобы повлиять на направление поиска. Для любой проблемы с нетривиальным пространством поиска невозможно протестировать больше, чем очень малую часть от общего количества точек пространства поиска методом случайного слепого поиска. Пусть есть KL точек проблемной области, представленных в виде строк размером L над алфавитом размера K. Например, если возможно проверить миллиард точек за секунду, и если слепой случайный поиск продолжается с начала Вселенной (т.е. около 15 млрд. лет), то мы бы нашли только около 1027 точек пространства поиска. Пространство поиска 1027 – это приблизительно290 точек, соответствующих двоичной строке с относительно скромной длиной L = 90.

Другой вариант состоит в «жадном» использовании лучшего результата тестирования начальной популяции. «Жадная» стратегия предполагает применение этого лучшего результата без тестирования каких-либо других точек пространства. «Жадная» стратегия, в отличие от случайного слепого поиска, является адаптивной и интеллектуальной, потому что использует информацию, полученную на одном этапе поиска, чтобы влиять на направление поиска следующего шага. В целом можно ожидать, что такая стратегия будет в два раза лучше случайного слепого поиска.

Но «жадная» стратегия дает мнимую уверенность, что лучшая точка пространства поиска будет случайно выбрана в маленькую начальную популяцию. В любом интересном пространстве поиска значимого размера маловероятно, что лучшая точка поколения начальной популяции окажется глобальным оптимумом всего пространства поиска, это маловероятно также для всех ранних поколений. Наша цель состоит в максимизации пользы на протяжении большого промежутка времени, а «жадная» стратегия весьма преждевременна на данном этапе.

Признавая, что мы нашли лучшую точку поколения, не рекомендуется исключать все остальные точки. Мы должны дать преимущество всем точкам, превосходящим среднее значение фитнеса.

Но если мы не будет проверять новые точки пространства, то вернемся к только что отклоненной «жадной» стратегии, т.е. будем наблюдать только за лучшими точками начальной выборки.

Оптимальная адаптивная (интеллектуальная) система должна обрабатывать имеющуюся на данный момент информацию об окружающей среде, чтобы найти оптимальное соотношение между стоимостью освоения новых точек в пространстве и стоимостью применения уже оцененных точек проблемной области. Этот выбор, по сути, должен отражать статистическую дисперсию, связанную с затратами.

Однако в генетическом алгоритме, как и в природе, особи, фактически находящиеся в популяции, имеют второстепенное значение для эволюционного процесса.

# Проблема представления в генетических алгоритмах

Представление является ключевым вопросом в генетическом алгоритме, потому что генетические алгоритмы работаю непосредственно с кодовым представлением проблемы. Обычный генетический алгоритм, работающий с символьными строками фиксированной длины, способен решить множество проблем. Тем не менее, использованием строк заданной длины оставляет много вопросов нерешенными.

Для большинства проблем наиболее естественным представлением для решения проблемы является иерархическая компьютерная программа, а не символьная строка заданной длины. Размер и вид иерархической компьютерной программы, которая позволит решить данную проблему, как правило, неизвестны заранее, поэтому программа должна иметь возможность изменения размера и вида. Представлять символьные строки фиксированной длины в виде иерархической компьютерной программы, способной динамически изменять размер и вид, довольно трудно и неестественно.

# Обзор генетического программирования

Генетическое программирование пробует решить проблему представления в генетических алгоритмах путем увеличения сложности адаптируемых структур. В частности, адаптируемые структуры в генетическом программировании являются общими иерархическими компьютерными программами, динамически изменяющими размер и вид.

Многие, казалось бы, разные проблемы в искусственном интеллекте, символьной обработке и машинном обучении можно рассматривать как требующие компьютерной программы, вычисляющей некоторый требуемый результат в зависимости от входных параметров.

Процесс решения этих проблем можно сформулировать как поиск наиболее подходящей индивидуальной компьютерной программы среди всех возможных компьютерных программ. Пространство поиска состоит из всех возможных компьютерных программ, составленных из функций и терминальных символов, соответствующих проблемной области. Генетическое программирование предоставляет способ поиска этих наиболее подходящих индивидуальных компьютерных программ.

В генетическом программировании популяции сотен или тысяч компьютерных программ генетически выведены. Эта селекция осуществляется с помощью принципа выживания сильнейших и воспроизводства наиболее приспособленных особей вместе с генетической рекомбинацией (скрещиванием) организмов путем применения операций, подходящих для компьютерных программ. Компьютерная программа, которая решает (или приблизительно решает) определенную проблему может возникнуть из комбинации естественного отбора Дарвина и генетических операций.

Генетическое программирование начинается с генерации случайным образом выбранной начальной популяции компьютерных программ из функций и терминалов, соответствующих проблемной области. Функции могут быть стандартными арифметическими операциями, операциями программирования, математическими функциями, логическими функциями или предметно-ориентированными фикциями. В зависимости от конкретной задачи, компьютерная программа может работать с логическими значениями, целыми, вещественными или комплексными числами, векторами, символами. Создание этой начальной популяции в действительности «слепой» случайный поиск в пространстве проблемной области

Каждая программа в популяции оценивается, насколько хорошо она выполняет свои задачи в проблемной среде. Эта оценка носит название фитнес-меры (меры пригодности). Ее вид зависит от проблемы.

Для многих задач пригодность естественно измерять ошибкой, погрешностью компьютерной программы. Чем ближе эта ошибка к нулю, тем лучше данная программа. Также может фитнес может быть комбинацией таких факторов, как корректность, экономность и бережливость.

Как правило, каждая компьютерная программа популяции отработает в нескольких фитнес случаях, тогда фитнес будет считаться в виде суммы или среднего арифметического значений фитнеса всех случаев. Эти фитнес случаи могут быть выборкой из различных значений независимой переменной или представлять собой выборку из различных начальных условий системы. Например, пригодность компьютерной программы может быть суммой абсолютной величины от разности вычисленного программой значения и корректного решения проблемы. Эта сумма может быть получена из выборки 50 различных входных значений программы. 50 фитнес случаев могут быть выбраны случайным образом или структурированы другим способом.

За исключением случаев, когда проблема мала и проста, она не может быть легко решена путем слепого случайного поиска, компьютерные программы нулевого поколения будут иметь очень плохой фитнес. Тем не менее, некоторые особи в популяции будут немного пригоднее остальных. Эти различия в эффективности следует использовать в дальнейшем.

Принципы репродукции и выживания наиболее приспособленных особей и генетические операции половой рекомбинации (скрещивание) используются для создания нового поколения индивидуальных компьютерных программ из текущей популяции.

Операция репродукции включает в себя селекцию, пропорциональную значениям фитнеса, компьютерных программ из текущей популяции и позволяет отобранным особям выжить путем копирования в новую популяцию.

Генетический процесс полового скрещивания двух родителей – компьютерных программ – используется для создания новых потомков от родителей, выбранных пропорционально фитнесу. Программы-родители обычно имеют различный размер и форму. Программы-потомки составляются из подвыражений (поддеревьев, подпрограмм) родителей. Эти потомки, как правило, различаются размером и видом от своих родителей.

Интуитивно, если две компьютерные программы несколько эффективны в решении проблемы, то их части, возможно, тоже немного пригодны. Скрещивая случайно выбранные части относительно пригодных программ, мы можем получить новую компьютерную программу, которая даже лучше решает проблему.

После выполнения операций репродукции и скрещивания с текущей популяцией, популяция потомков (новое поколение) помещается в старую популяцию (прошлое поколение).

Каждая особь новой популяции компьютерных программ затем проверяется на пригодность, и процесс повторяется в течение многих поколений.

На каждом этапе этого параллельного, локально управляемого, децентрализованного процесса состоянием процесса будет являться только текущая популяция особей. Движущая сила этого процесса состоит только в наблюдении за пригодностью особей текущей популяции.

Данный алгоритм будет производить популяцию компьютерных программ, которые через много поколений, как правило, демонстрируют увеличение средней пригодности. Кроме того, это популяции могут быстро и эффективно приспосабливаться к изменениям в окружающей среде.

Как правило, лучшая особь, которая появляется в любом по счету поколении, обозначается как результат генетического программирования.

Важной особенностью генетического программирования является иерархический характер производимых программ. Результаты генетического программирования по своей природе, сути иерархичны.

Динамическая изменчивость также является важным признаком компьютерных программ, созданных генетическим программированием. Было бы трудно и неестественно пытаться заранее уточнить или ограничить размер и форму возможного решения. Более того, такая предварительная спецификация сужает пространство поиска решений и может исключить нахождение решения вообще.

Еще одной важной особенностью генетического программирования выступает отсутствие или сравнительно малая роль предварительной обработки входных данных и постобработки выходных значений. Входные параметры, промежуточные результаты и выходные значения обычно выражаются непосредственно в терминах естественной терминологии предметной области. Элементы функционального множества также естественны для проблемной области.

И наконец, структуры, подвергающиеся адаптации, активны. Они не являются пассивными кодировками решения проблемы. Вместо этого с учетом компьютера, на котором происходит запуск, программы в генетическом программировании – это активные структуры, способные выполниться в их текущем виде.

Парадигма генетического программирования является независимой от проблемной области. Это обеспечивает единый, унифицированный подход к проблеме нахождения решения в виде компьютерной программы.

# Подробное описание генетического программирования

Адаптация (или обучение) предполагает изменение некоторой структуры так, чтобы она лучше выполняла свои задачи в окружающей среде.

## Адаптируемые структуры

В каждой адаптивной системе или системе обучения, хотя бы одна структура подвергается адаптации.

В обычном генетическом алгоритме и генетическом программировании в качестве адаптируемых структур выступает популяция особей из всего пространства поиска. Генетические метода отличаются от большинства других методов поиска тем, что они включают одновременный параллельный поиск с участием сотен или тысяч точек всего пространства поиска.

Отдельные адаптируемые структуры в генетическом программировании являются иерархически структурированными компьютерными программами. Размер, форма и содержание этих компьютерных программ может быть динамически изменено в ходе выполнения процесса.

Множество возможных структур в генетическом программировании – это множество всех возможных композиций функций, которые могут быть составлены рекурсивно из функционального множества *F = {f1, f2, ..., fN\_func}* и множества терминальных символов *T = {a1, a2, ..., aN\_term}.* Каждая конкретная функция *fi* из функционального множества *F* принимает указанное число *z(fi)* аргументов. То есть функция *fi* имеет арность равную *z(fi)*.

Функционально множество может состоять из следующих элементов:

* арифметические операции (+, -, \* , /);
* математические функции (sin, cos, exp, log);
* логические операции (конъюнкция, дизъюнкция, отрицание);
* условные операторы (If-Then-Else);
* операторы циклов (for, while, do);
* другие проблемно-ориентированные функции.

Терминальными символами обычно являются либо переменные «атомы» (представляющие входы, сенсоры, датчики или переменные состояния некоторой системы), либо постоянные «атомы» (такие как число 3 или логическая константа NIL). Иногда в качестве термов также берутся функции, не принимающие явных аргументов, реальная функциональность таких функций заключается в создании побочных эффектов для состояний системы.

Адаптационные структуры в генетическом программировании отличаются от структур, подвергающихся адаптации, в обычном генетическом алгоритме, который оперирует строками. В обычном генетическом алгоритме структуры представляют одномерные линейные строки фиксированной длины. В вариации Стивена Смита (1980б 1983) обычного генетического алгоритма адаптационные структуры являются одномерными линейными строками переменной длины.

В генетическом программировании терминальное и функциональное множества должны быть выбраны так, чтобы они удовлетворяли требованиями замкнутости и достаточности.

### Замкнутость множеств

Свойство замкнутости требует, чтобы каждая функция из функционального множества могла принять в качестве аргумента любое значение и тип данных, которые могут быть возвращены любой функцией из функционального множества, а также любой элемент терминального множества. То есть каждая функция из функционального множества должна быть четко определена и замкнута для любой комбинации аргументов, с которыми она может встретиться.

В обычных программах арифметические операции с численными переменными иногда не определены (например, деление на нуль). Многие обычные математические функции иногда не определены (например, логарифм нуля). Кроме того, возвращаемые некоторыми математическими функциями значения могут входить в список неприемлемых типов данных для проблемной области (например, квадратный корень или логарифм отрицательного числа). Также логическое значение, обычно возвращаемое условным оператором, как правило, не принимается в качестве аргумента арифметическими функциями.

Может показаться, что соблюдение свойства замкнутости для обычной компьютерной программы невозможно или приведет к очень сложной и ограниченной синтаксической структуре. На самом деле это не так. Замкнутость может быть достигнута простым способом для подавляющего большинства задач просто путем тщательной обработки небольшого количества ситуаций.

Если арифметическая операция деления получит в качестве второго аргумента число 0, то свойство замкнутости будет нарушено. Один простой подход гарантирует замкнутость – определение защищенной функции деления. Защищенная функция деления принимает два аргумента и возвращает 1 при попытке деления на 0 (включая деление 0 на 0), а в других случаях возвращает нормальное частное.

Свойство замкнутости желаемо, но не абсолютно необходимо. Если свойство замкнутости не превалирует, то существуют альтернативные способы: исключение особей с нежелательным результатом или система штрафов для таких структур. Вопрос обработки таких ситуаций не уникален для генетических методов, а широко обсуждается в связи с другими алгоритмами. Удовлетворительного решения этой проблемы пока не существует, поэтому мы будет соблюдать свойство замкнутости.

### Достаточность множеств

Свойство достаточности требует, чтобы терминальное и функциональное множества могли выразить решение проблемы. Пользователь генетического программирования должен убедиться, что композиция функция и термов приведет к решению проблемы.

Этап определения переменных, у которых достаточно возможностей решить определенную проблему, является общим практически для каждой проблемы в науке.

В зависимости от проблемы этот шаг идентификации может быть очевидным, а может потребовать глубокого понимания предметной области.

### Универсальность выбора функций и терминалов

Этапы определения примитивных функций и терминальных символов в генетическом программировании эквивалентны аналогичным необходимым этапам в других парадигмах машинного обучения. Эти два шага часто явно не определяются, обсуждаются или признаются исследователями других парадигм. Причиной этого упущения может быть то, что исследователь считает выбор примитивных функций и терминалов присущим формулировке задачи. Такой взгляд особенно понятен, если ученый фокусируется лишь на одном конкретном типе проблемы специфической области.

## Начальные структуры

Начальные структуры в генетическом программировании состоят из особей исходной популяции, каждая из которых представляет решение проблемы в виде польской записи.

Создание каждого выражения начальной популяции выполняется в виде дерева со случайно выбранным корнем и упорядоченными ветвями, представляющего данную польскую нотацию.

Начинаем со случайного выбора одной функции из функционального множества *F*, которая станет корнем дерева. Мы ограничиваем выбор функциональным множеством, поскольку нам необходимо создать иерархическую структуру, а не вырожденную структуру, состоящую из одного терминального символа.

Рисунок 1 демонстрирует начало создания случайного дерева программы. Функция + (с двумя аргументами) была выбрана случайно из функционального множества *F* в качества корня дерева.



Рисунок 1-Создание корня дерева

Когда узел дерева помечается функцией *f* из *F*, то *z(f)* линий, где *z(f)* – количество аргументов функции *f*, выходит из этого узла. Затем для каждой такой линии случайно выбирается элемент из объединенного множества , функций и терминалов, для конечной точки – другого узла – этой линий.

Если в качестве узла была выбрана функция, то дальнейшее создание дерева продолжается рекурсивно так, как было описано выше. Например, на рисунке 2, функция \* была выбрана из множества в качестве внутреннего узла (номер 2) для конечной точки левой линии корневого узла (функция +, номер 1). Функция \* принимает два аргумента, поэтому из 2-ой вершины выходят две линии.



Рисунок 2-Выбор внутреннего узла

Если в качестве любого узла выбирается терминальный символ, то этот узел становится листом, и процесс создания поддерева для этой вершины прекращается. Например, на рисунке 3 терминальный символ A был выбран в качестве вершины для левой линий функции \*. Аналогичным образом, терминалы B и C стали вершинами двух правых линий функций \* и + соответственно. Этот процесс продолжается рекурсивно слева направо, пока все дерево не создано, как показано на рисунке 3.



Рисунок 3-Созданное дерево программы

Данный процесс генерации может быть реализован различными способами, приводящими к получению случайных начальных деревьев разного размера и вида. Два основных метода называются «полным» методом и «растущим» методом. Длина дерева определяется, как длина самого длинного пути от корня к листу.

Полный метод создания начальной популяции состоит в генерации дерева, у которого длина каждого пути от листа к корню равна указанной максимальной глубине. Такой вид дерева можно получить путем ограничения выбора функциональным множеством для вершин, у которых глубина меньше заданной. А затем ограничить выбор только терминальным множеством для вершин с максимальной глубиной.

Растущий метод генерации начальной популяции состоит в создании деревьев различной формы. Длина пути от листа до корня не больше заданной максимальной глубины. Это достигается путем случайного выбора внутреннего узла из множества , при этом длина пути от корня к узлу меньше максимальной глубины. А вершины дерева, чья длина равна максимальной, становятся листами и выбираются из терминального множества.

Лучший результат для широкого диапазона проблемы дает объединенный метод (ramped half-and-half), при котором начальная популяция создается чередованием полного и растущего методов. В генетическом программировании мы, как правило, не знаем заранее (или не хотим указывать) размер и форму решения. Объединенный метод генерации деревьев создает широкое разнообразие деревьев различного размера и вида.

Объединенный метод генерации – это смешанный метод, включающий и растущий, и полный методы. Данный способ состоит из создания равного количества деревьев с глубиной, которая находится в интервале от 2 до максимальной заданной глубины. Например, если максимальная глубина равна 6, то 20% деревьев будут иметь глубину 2, 20% деревьев глубину 3, и так далее до глубины 6. Затем для каждого значения глубины 50% деревьев создаются полным методом, а остальные 50% растущим методом.

Заметим, что у всех деревьев, созданных полным методом с заданной глубиной, длина пути от корня к листу одинаковая, равная максимальной глубине, и поэтому эти деревья имеют одинаковую форму. В отличие от этого, у всех деревьев, полученных растущим методом с данным значением глубины, ни один путь от корня дерева до листа не превышает максимальной глубины. Поэтому эти деревья значительно отличаются друг от друга по виду даже при одинаковой максимальной глубине.

Таким образом, объединенный метод создает деревья с большим разнообразием размеров и видов.

Повторяющиеся особи в начальной популяции непродуктивны: они тратят вычислительные ресурсы и приводят к нежелательному сокращению генетического разнообразия популяции. Поэтому желательно, но необязательно, предотвратить появление дубликатов в начальной случайно популяции. В генетическом программировании вероятность появления повторяющихся особей в начальной популяции особенно высока, когда деревья малы. Таким образом, каждое созданное выражение проверяется на уникальность перед добавлением в популяцию. Если новое выражение – дубликат, то процесс повторяется до создания уникальной особи. Иногда (например, для маленьких деревьев) мы должны подставить большее по размеру дерево во время процесса генерации, если исчерпано множество возможных деревьев данного размера.

Разнообразие популяции – это доля особей, у которых нет точной копии во всей популяции. Если выполняется проверка на дублирование при создании особей, то разнообразие начальной популяции равно 100%. В последующих поколениях появление одинаковых особей при использовании репродукции является неотъемлемой частью генетического процесса.

В противоположность этому в обычном генетическом алгоритме, работающем со строками символом фиксированной длины, каждый из символов строки начальной популяции обычно создается с помощью применения бинарного генератора случайных чисел. Например, бинарные строки длиной 453, используемые Джефферсоном и другими (1991), создаются с помощью двоичного генератора случайных чисел, имеющего пространство поиска размером 2453 (т.е. около 10137). Было бы необычно получить дубликаты среди всего лишь 65,536 индивидуальных строк популяции, когда пространство поиска размером 10137. Поэтому в обычных генетических алгоритмах, как правило, не проводится проверка одинаковых особей.

## Фитнес

Фитнес (пригодность) является движущей силой естественного отбора по Дарвину, а также обычных генетических алгоритмов и генетического программирования.

В природе пригодность – это вероятность того, что особь доживет до репродукционного возраста и воспроизведется. Данный показатель может учитываться при расчете числа потомков. В искусственном мире математических алгоритмов мы оцениваем пригодность каким-либо способом, а затем используем фитнес для контроля применения операций, изменяющих структуры в нашей искусственной популяции.

Фитнес может быть вычислен при помощи различных методов, явных и неявных.

Наиболее распространенным подходом вычисления пригодности является создание определенной фитнес меры для каждой особи популяции. Данный подход используется подавляющим большинством обычных генетических алгоритмов. Каждой особи популяции присваивается скалярное значение пригодности при помощи некоторой четко и явно определенной процедуры оценки.

Фитнес также может быть вычислен путем совместной эволюции (коэволюции), при которой пригодность игровой стратегии определяется применением это стратегии против всей популяции (или отобранного числа) противоположной стратегии.

Тот факт, что особи существуют и выживают в популяции, а также успешно воспроизводятся, может свидетельствовать об их пригодности (как это происходит в природе). Такое неявное определение фитнеса часто используется в научных исследованиях (Ray 1990; Holland 1990, 1992). Однако, на данный момент, мы сосредоточимся на более общей ситуации, когда фитнес вычисляется явно.

Существует 4 меры фитнеса:

* исходный (raw) фитнес;
* стандартизованный (standardized) фитнес;
* отрегулированный (adjusted) фитнес;
* нормированный (normalized) фитнес.

### Исходный фитнес

Исходный фитнес – это измерение фитнеса, сформулированное в естественной терминологии проблемы. Например, исходным фитнесом для символьной регрессии является погрешность полученного результата вычисления функции относительно желаемого. Чем меньше погрешность, тем лучше.

Наиболее общим определением исходного фитнеса является ошибка. То есть исходный фитнес отдельного выражения – это сумма расстояний для всех фитнес случаев между полученным результатов для конкретного фитнес случая и желаемым результатом для этого же фитнес случая. Выражение может быть логическим, целочисленным, вещественным, комплексным, вектором или символьным значением.

Если выражение целочисленное или вещественное, то сумма дистанций вычисляется в виде суммы абсолютных значений разности полученного и необходимого результатов. Когда исходным фитнесом является ошибка, исходный фитнес r(i, t) отдельного выражения i в популяции размера M любого поколения шага t равна

*,*

где *S(i, j)* – вычисленное значение выражения i для фитнес случая j (всего N случаев), C(j) – необходимое значение для фитнес случая j.

Если выражение логическое или символьное, то сумма расстояний будет эквивалентна числу несоответствий. Если выражение комплексное или векторное, то сумма расстояний получается отдельно для каждого компонента структуры.

Если выражение вещественное или целочисленное, то квадратный корень суммы квадратов расстояний может быть использован в качестве альтернативы для измерения фитнеса.

Т.к. исходный фитнес основывается на естественной терминологии проблемы, чем лучше значение, тем оно меньше (когда в качестве фитнеса берется ошибка) или больше (когда используется для фитнеса достигнутая выгода).

### Стандартизованный фитнес

Стандартизованный фитнес *s(i, t)* пересчитывает исходный фитнес так, чтобы наименьшее численное значение всегда было лучшим результатом. Например, для проблемы оптимального управления, заключающейся в минимизации расходов, меньшее значение исходного фитнеса лучше. Также, если проблема заключается в минимизации ошибки, то меньшее значение исходного фитнеса лучше.

Для таких проблем, когда меньшее значение исходного фитнеса является лучшим, стандартизованный фитнес равняется исходному фитнесу. То есть *s(i, t) = r(i, t)*.

Удобно и желательно сделать лучшим значением стандартизованного фитнеса 0.

Если для определенных проблем большее значение исходного фитнеса лучше, то стандартизованный фитнес вычисляется из исходного фитнеса. Стандартизованный фитнес равняется разности максимально возможного значения исходного фитнеса и данного фитнеса. То есть *s(i, t) = rmax – r(i, t)*.

Если верхняя граница неизвестна, а большее значение исходного фитнеса является лучшим, то отрегулированный и нормализованный фитнесы могут быть вычислены непосредственно из исходного фитнеса.

### Отрегулированный фитнес

Кроме того для решения проблемы используется оптимальная регулировка фитнеса. Отрегулированный фитнес *a(i, t)* вычисляется из стандартизованного фитнеса следующим образом:

,

где *s(i, t)* – стандартизованный фитнес для особи *i* поколения *t*.

Отрегулированный фитнес находится в промежутке от 0 до 1. Чем больше его значение, тем лучше эта особь в популяции.

Необязательно использовать отрегулированный фитнес в генетическом программировании, но это хорошо помогает. Отрегулированный фитнес хорошо преувеличивает важность малых различий в значениях стандартизованной функции. Таким образом, по мере улучшения популяции большой акцент делается на небольшие различия, что позволяет увидеть разницу между хорошей и очень хорошей особью. Это преувеличение особенно велико, если стандартизованный фитнес достигает нуля, когда лучшее решение проблемы найдено. Например, если стандартизованный фитнес находится в промежутке от 0 (лучший) до 64 (худший), отрегулированный фитнес двух плохих особей, оцениваемых 64 и 63, будет 0.0154 и 0.0159 соответственно. А для особей с оценкой 4 и 3 отрегулированный фитнес равен 0.20 и 0.25 соответственно. Данный эффект слабеет (но все равно значителен), когда лучшее значение стандартизованного фитнеса не может быть определено.

Необходимо заметить, что для других методов выбора, отличающихся от пропорциональных фитнесу, отрегулированная функция неуместна и не используется.

### Нормализованный фитнес

Если метод селекции основывается на пропорциональности фитнесу, то понятие нормализованного фитнеса также необходимо.

Нормализованный фитнес n(i, t) вычисляется из значения отрегулированного фитнеса следующим образом:

У нормализованного фитнеса есть три характеристики:

* Он находится в промежутке от 0 до 1.
* Чем больше его значение, тем лучше особь популяции.
* Сумма значений нормализованного фитнеса равна 1.

### Главные операции изменения структур

This section describes the two primary operations used to modify the structures undergoing adaptation in genetic programming:

* Darwinian reproduction
* crossover (sexual recombination).

### Репродукция

The reproduction operation for genetic programming is the basic engine of Darwinian natural selection and survival of the fittest. The reproduction operation is asexual in that it operates on only one parental S-expression and produces only one offspring S-expression on each occasion when it is performed.

### Скрещивание

The crossover (sexual recombination) operation for genetic programming creates variation in the population by producing new offspring that consist of parts taken from each parent. The crossover operation starts with two parental S-expressions and produces two offspring S-expressions. That is, it is a sexual operation.

## Второстепенные операции изменения структур

In addition to the two primary genetic operations of reproduction and crossover in genetic programming, there are five optional secondary operations worth mentioning:

· mutation

· permutation

· editing

· encapsulation, and

· decimation.

### Мутация

The mutation operation introduces random changes in structures in the population.

In conventional genetic algorithms operating on strings, the mutation operation can be beneficial in reintroducing diversity in a population that may be tending to converge prematurely. In the conventional genetic algorithm, it is common for a particular symbol (i.e., an allele) appearing at a particular position on a chromosome string to disappear at an early stage of a run because that particular allele is associated with inferior performance, given the alleles prevailing at other positions of the chromosome string at that stage of the run. Then, because of the

nonlinearities of the problem, the now-extinct allele may be precisely what is needed to achieve optimal performance at a later stage of the run, since a different and better combination of alleles is now prevailing at the other positions of the chromosome string. The situation just described is not conjectural but is, in fact, very typical. Genetic methods are normally applied to problems with highly nonlinear search spaces, and this situation is the essence of what is involved in nonlinear search spaces.

### Перестановка

The *permutation* operation is a generalization of the inversion operation for the conventional genetic algorithm operating on strings.

### Редактирование

The *editing* operation provides a means to edit and simplify S-expressions as genetic programming is running.

### Инкапсуляция

The encapsulation operation is a means for automatically identifying a potentially useful subtree and giving it a name so that it can be referenced and used later.

### Прореживание

For some complex problems, the distribution of fitness values over the initial random population may be skewed so that a very large percentage of the individuals have very poor fitness (e.g., a raw fitness of 0). This skewing may occur in problems where individuals in the population are assigned some penalty value of fitness because they would otherwise consume an infinite amount of time (as in time optimal control problems or problems involving iterative loops). In such problems, enormous amounts of computer time may be expended and wasted in early generations on very poor individuals. Moreover, when a highly skewed distribution of fitness values occurs, the few individuals with marginally better fitness values immediately begin to dominate the population and the variety of the population quickly begins to drop. In genetic programming, the crossover operation is usually capable of quickly reintroducing variety into the population. However, because the selection of parents to participate in crossover is based on fitness, the crossover operation concentrates on the few individuals in the population with the marginally better fitness values.

# Практический результат

Тестирование работы программы сначала производилось с тестовыми данными.

Вид исходной функции:

Множество независимых переменных состоит из переменной *x,* принимающей следующие значения: {-1, -0.3, -0.2, 0, 0.1, 0.5, 0.6, 1, 1.5, 2}.

Ожидаемые зависимые от функции значения: {4.301267, 0.2033, 0.019941, 0, 0.153999, 1.9349, 2.6784, 6.301, 6.7908977, -3.524658}.

Второстепенные независимые значения: {1.2, 1.7, 1.9, 0.2, -0.5, -0.8}.

Второстепенные зависимые значения: {7.644479, 3.4426275, -1.3505534155622465, 0.4199413380264879, 0.934942963676, 2.8428921923105914}.

В результате работы программы были получены функции, суммарная квадратичная ошибка которых была меньше 1. Из этих функций была выбрана функция с минимальной ошибкой = 0.457872389693 и функция с минимальной суммарной квадратичной ошибкой по второстепенным данным (ошибка по основным значениям = 0.999405090213)

Ошибка = 0.457872389693:



Ошибка = 0.999405090213:



# Заключение

заключение

# Список использованных источников