Лабораторная работа №4

Генетическое программирование

Цель работы: Решение задачи символьной регрессии. Графическое отображение результатов оптимизации.

4.1 Теоретическая часть

В генетическом программировании (ГП) в качестве особи выступает программа, представленная в определенном формате, которая решает некоторую задачу. Часто это выполняется с использованием обучающих данных и индуктивного вывода. ГП очень близко к машинному обучению и поэтому в качестве фитнесс-функции как правило выступают функции ошибки. ГП работает с генетическим материалом переменной длины, что требует нестандартной формы представления генома и соответствующих генетических операторов.

Программы составляются из переменных, констант и функций, которые связаны некоторыми синтаксическими правилами. Поэтому определяется терминальное множество, содержащее константы и переменные, и функциональное множество, которое состоит, прежде всего, из операторов и необходимых элементарных функций (exp(x), sin(x) и т.п.). Следует отметить, что терминалы и функции играют различную роль. Терминалы обеспечивают входные значения в систему (программу), в то время как функции используются при обработке значений внутри системы. Термины «функции» и «терминалы» взяты из древовидного представления, и соответствуют узлам древовидных (или графоподобных) структур.

Терминальное множество

Терминальное множество включает в себя:

1) внешние входы в программу;

2) используемые в программе константы;

3) функции, которые не имеют аргументов.

Слово «терминал» используется, так как перечисленные выше объекты соответствуют терминальным (конечным, висячим) вершинам в древовидных структурах. Терминал дает некоторое (численное) значение, у него нет входных аргументов, и он имеет нулевую «арность».

Следует отметить тесную связь внешних входов с обучающими выборками, которые часто используются в ГП. При этом каждая переменная (признак, фактор и т.п.) обучающей выборки соответствует своему терминалу.

Функциональное множество

Функциональное множество состоит из операторов (взятых у языков программирования) и различных функций. Оно может быть достаточно широким и включать типовые конструкции различных языков программирования, такие как:

1) булевы функции И, ИЛИ, НЕ и т.п.;
2) арифметические функции сложения, вычитания, умножения, деления;
3) трансцендентные функции (тригонометрические, логарифмические);
4) функции присваивания значения переменным (a:=2);
5) условные операторы (if then, else: case или switch операторы ветвления);
6) операторы переходов (go to, jump, call- вызов функции);
7) операторы цикла (while do, repeat until, for do);
8) подпрограммы и функции.

С одной стороны, терминальное и функциональное множество должны быть достаточно большими для представления потенциального решения. Но другой стороны не следует сильно без необходимости расширять

функциональное множество, поскольку при этом резко возрастает пространство поиска решений. Набор функций существенно зависит от решаемой задачи.

Структуры для представления программ

Терминалы и функции должны быть объединены по определенным правилам в некоторые структуры, которые могут представлять программы и использоваться при их выполнении. Выбор структуры существенно влияет на порядок выполнения программы, распределение и использование локальной или глобальной памяти.

В настоящее время наиболее распространенными структурами для представления особей (потенциальных решений проблемы) являются:

1) древовидное представление;

2) линейная структура;

3) графоподобная структура.

Древовидное представление

В качестве примера рассмотрим арифметическую формулу, которую удобно представлять деревом. Рассмотрим арифметическую формулу $\frac{d}{e}-a*(b+c)$ (в обычном представлении). Отметим, что дерево (генотип) рис.4.1 также представляет эту формулу (фенотип) $\frac{d}{e}-a*(b+c)$.

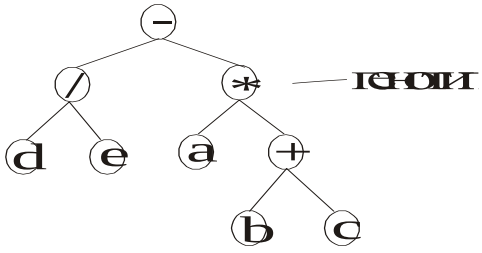


Рис.4.1 Древовидное представление формулы d/e-a*(b+c).

При этом листья дерева соответствуют терминалам, а внутренние узлы – функциям.

Древовидная форма представления генотипа оказывается для данного класса задач более эффективной, и позволяет работать с программами или выражениями различной длины. Важным аспектом является также использование памяти при выполнении программы. Древовидная структура позволяет использовать только локальную память в процессе выполнения. Локальность памяти встроена в саму древовидную структуру. Значения переменных доступны для функции только в дереве, корню которого соответствует функция. Например, значения переменных d, e являются локальными относительно узла ‘/’.

Инициализация начальной популяции

Данный этап в ГП является не таким простым, как в классическом ГА, где генерация случайных двоичных строк не представляет особых проблем. Это связано с различной сложностью особей и их структурой. Естественно методы инициализации начальной популяции различны для разных форм представления программ. Одним из важнейших параметров в ГП является максимально возможный размер (сложность) программы.

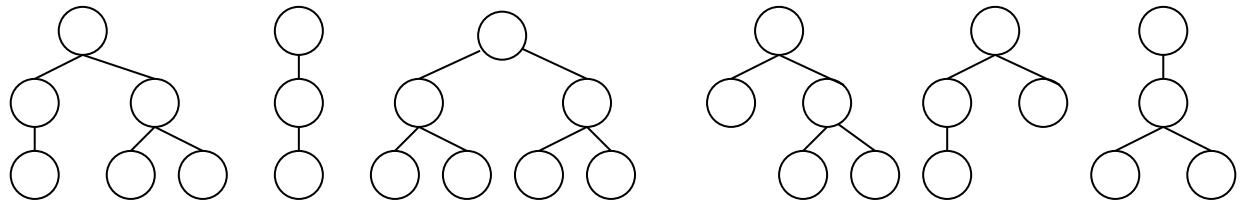
Инициализация древовидных структур

Для деревьев в качестве меры сложности используется максимальная глубина дерева или общее число узлов в дереве. Глубиной узла называется минимальное число узлов, которые необходимо пройти от корня дерева к этому узлу. Максимальной глубиной дерева Dm называется максимально возможная глубина в дереве для терминального символа (листа). Если арность каждого узла равна двум, то общее число узлов не превышает 2^{B_b}, которое также используется в качестве меры сложности.

Инициализация древовидных структур выполняется путем случайного выбора функциональных и терминальных символов при заданной максимальной глубине дерева. Применяются два основных метода: а) полная (full) и б) растущая (grow) инициализация, которые показаны на рис.4.2 .

В полном методе при генерации дерева, пока не достигнута максимальная глубина, допускается выбор только функциональных символов, а на последнем уровне (максимальной глубины) выбираются только терминальные символы.

В растущей инициализации генерируются нерегулярные деревья с различной глубиной листьев вследствие случайного на каждом шаге выбора функционального или терминального символа. Здесь при выборе терминального символа рост дерева прекращается по текущей ветви и поэтому дерево имеет нерегулярную структуру.



а) б)

Рис.4.2. Деревья, генерируемые при инициализации разными методами

При использовании первого метода начальная популяция содержит однородное множество структур, что способствует вырождению генетического материала (и преждевременной сходимости в локальных экстремумах). Поэтому на практике часто эти два метода используют одновременно следующим образом. Начальная популяция генерируется так, чтобы в нее входили деревья с разной максимальной длиной примерно поровну (для нашего примера D_m=1, D_m=2, D_m=3. D_m=4). Для каждой глубины

первая половина деревьев генерируется полным методом, а вторая – растущей инициализацией.

Кроссинговер в генетическом программировании

В начальной популяции особи (потенциальные решения), как правило, имеют низкие (плохие) значения фитнесс-функции. В процессе эволюции с помощью генетических операторов популяция развивается и значения фитнесс-функции улучшаются. В ГП используются те же, что и в ГА, генетические операторы кроссинговера, мутации и репродукции. Отметим, что в терминах машинного обучения они соответствуют операторам поиска решения. Далее мы рассмотрим выполнение генетических операторов на ранее определенных структурах.

Выполнение кроссиновера на древовидных структурах

Для древообразной формы представления используются следующие три основные оператора кроссинговера (ОК):

а) узловой ОК;

б) кроссинговер поддеревьев;

в) смешанный.

В узловом операторе кроссинговера выбираются два родителя (два дерева) и узлы в этих деревьях. Первый родитель называется доминантом, второй – рецессивом. Узлы в деревьях могут быть разного типа. Поэтому сначала необходимо убедиться, что выбранные узлы у родителей являются взаимозаменяемыми. Если узел во втором родителе не соответствует типу узла первого родителя, то случайным образом выбирается другой узел во втором родителе, который опять подлежит проверке на предмет совместимости. Далее производится обмен узлов.

В кроссинговере поддеревьев родители обмениваются не узлами, а определяемыми ими поддеревьями. Оператор выполняется следующим образом:

1. Выбираются родители (один – доминантный, другой – рецессивный). Далее необходимо убедиться, что выбранные узлы взаимозаменяемы, т.е. принадлежат одному типу. Иначе, как и в предыдущем случае в рецессивном дереве выбирается другой узел с последующей проверкой.
2. Затем производится обмен поддеревьев, которые определены этими узлами.
3. Вычисляется размер ожидаемых потомков. Если ожидаемый размер (сложность потомка) не превышает заданный порог, такой обмен ветвями запоминается.

Этот тип ОК является основным. При этом под размером (под)дерева понимается, как и ранее, либо его высота, либо число его вершин .

При смешанном операторе кроссинговера для некоторых узлов выполняется узловой ОК, а для других - кроссиновер поддеревьев.

Мутация в генетическом программировании

После выполнения кроссиговера с заданной малой вероятностью выполняется мутация для выбранной одной особи-программы.

Выполнение мутации на древовидных структурах

Для деревьев используются следующие операторы мутации (ОМ):

а) узловая;

б) усекающая;

в) растущая.

Узловая мутация выполняется следующим образом:

- выбрать случайным образом узел, подлежащий мутации, определить его тип;

- случайно выбрать из соответствующего множества вариантов узлов отличный от рассматриваемого узел;
- поменять исходный узел на выбранный.

Усекающая мутация производится так:

- определяется или выбирается узел;
- случайным образом выбирается терминальный символ из заданного множества;
- обрезается ветвь узла мутации;
- вместо обрезанной ветви помещается выбранный терминальный символ.

Растущая мутация выполняется следующим образом:

- определяется узел мутации;
- если узел нетерминальный то необходимо отсечь ветви исходящие из него, иначе выбрать другой узел.
- вычислить размер (сложность) остатка дерева;
- вместо отсеченного дерева вырастить случайным образом новое дерево так, чтобы размер нового построенного дерева не превышал заданный порог.

Это очень мощный оператор, который обладает большими возможностями.

Фитнесс-функция в генетическом программировании

В отличие от генетических алгоритмов, где часто при поиске экстремумов в качестве фитнесс-функции используется исходная целевая функция, в ГП фитнесс-функция обычно определяет меру близости между реальными y_i и требуемыми d_i выходными значениями (например, при использовании ГП в символьной регрессии). Поэтому в качестве фитнесс-функции часто используется абсолютная или квадратичная ошибка.

В этом случае фитнесс-функция использует обучающее множество данных, на котором выполняется обучение системы. С помощью фитнесс-функции в процессе обучения реализуется обратная связь, которая показывает насколько хорошо данная особь-программа реализует необходимую функцию на обучающем множестве. Для этого можно использовать различные виды фитнес-функций, некоторые из которых мы рассмотрим ниже.

Рассмотрим следующий пример [5] с обучающей выборкой, представленной табл.6.7. Каждая строка таблицы определяет один элемент (x,y) обучающей выборки. Необходимо в процессе эволюции построить программу (или формулу в случае символьной регрессии), которая для каждого входного значения x вычисляет необходимое (в соответствии с табл.6.5) значение y (фактически нам необходимо реализовать функцию $f(x) = x^2 + x$).

Таблица 4.1.

№	Вход x	Выход d
1	1	2
2	2	6
3	4	20
4	7	56
5	9	90

Рассмотрим в качестве фитнесс-функции ошибку в метрике абсолютных значений $f_a = \sum_{i=1}^{n} |y_i - d_i|$, где суммирование выполняется по обучающей выборке. Эта фитнесс-функция соответствует первому определению «непрерывной», поскольку чем ближе значения y_i к d_i, тем меньше значение фитнесс-функции. Приведенная фитнесс-функция является также стандартизованной, так как в случае идеального решения дает нулевое значение.

Часто в качестве фитнесс-функции также используют квадратичную ошибку $f_s = \sum_{i=1}^{n} (y_i - d_i)^2$. Таблица 6.8 показывает различие для этих двух фитнесс-функций в том случае, если на некотором (промежуточном) этапе в качестве особи оценивается (плохо обученная) программа, реализующая функцию $f(x) = x^2$.

Мы рассмотрели использование в качестве фитнесс-функции ошибки в двух метриках, которые характерны для применения ГП в качестве символьной регрессии.

Таблица 4.2

№	Вход x	Выход d	Выход y	Ошибка f_a	Ошибка f_s
1	1	2	1	1	2
2	2	6	4	2	4
3	4	20	16	4	16
4	7	56	49	7	49
5	9	90	81	9	81
	Общая ошибка			23	151

Естественно разработано множество типов фитнесс-функций вид которых существенно зависит от исследуемой проблемы. В некоторых случаях, кроме близости решений учитываются и другие критерии, например, длина или время выполнения программы. В этом случае говорят о многокритериальных фитнесс-функциях.

Общий алгоритм генетического программирования

Таким образом, для решения задачи с помощью ГП необходимо выполнить описанные выше предварительные этапы:

1) Определить терминальное множество;
2) Определить функциональное множество;
3) Определить фитнесс-функцию;

4) Определить значения параметров, такие как мощность популяции, максимальный размер особи, вероятности кроссинговера и мутации, способ отбора родителей, критерий окончания эволюции (например, максимальное число поколений) и т.п.

После этого можно разрабатывать непосредственно сам эволюционный алгоритм, реализующий ГП для конкретной задачи.

Например, решение задачи на основе ГП можно представить следующей последовательностью действий.

1) установка параметров эволюции;
2) инициализация начальной популяции;
3) t:=0;
4) оценка особей, входящих в популяцию;
5) t:=t+1;
6) отбор родителей;
7) создание потомков выбранных пар родителей – выполнение оператора кроссинговера;
8) мутация новых особей;
9) расширение популяции новыми порожденными особями;
10) сокращение расширенной популяции до исходного размера;
11) если критерий останова алгоритма выполнен, то выбор лучшей особи в конечной популяции – результат работы алгоритма. Иначе переход на шаг 4.

Символьная регрессия

Этот раздел является одним из важнейших приложений ГП. Данный термин подчеркивает то, что здесь объектом поиска является символьное описание модели, в отличие от множества коэффициентов в стандартных методах. Этот подход существенно отличается от других методов регрессии и использования нейросетей прямого распространения, где структура (и сложность) модели предполагается известной и фактически необходимо найти только ее коэффициенты. В случае символьной регрессии вид и

сложность функции заранее неизвестны и могут изменяться в процессе поиска.

Символьная регрессия (СР) ищет аналитические выражения, точно описывающие изучаемые явления. Главное преимущество этого подхода заключается в том, что он может возвращать интерпретируемую модель, которая может быть информативной для пользователей, сохраняя при этом высокую точность. Текущим стандартом для тестирования этих алгоритмов является тест функции (benchmarks), которые позволяют оценивать методы на сотнях наборов данных, представляющих собой смесь реальных и моделируемых процессов, охватывающих несколько предметных областей. Отметим, что часто возможности оценить интерпретируемость ограничены размерностью выражений на реальных данных и точностью модели на синтетических данных. На практике размерность модели — лишь один из многих факторов, используемых экспертами для определения того, насколько интеллектуальной является модель.

Целью параметризованной функции символьной регрессии $f(x,\theta)$[1] является нахождение выражения, наилучшим образом соответствующего заданным данным. Такие выражения могут описывать нелинейные взаимодействия с точностью, аналогичной точности непрозрачных моделей (например, глубоких нейронных сетей или древовидных ансамблей), позволяя людям понимать их поведение в мельчайших деталях, как это делают прозрачные модели (например, линейные модели или деревья решений). Поэтому, СР используется для обнаружения новых явлений на основе собранных данных, расширяя наши знания в физике, химии , медицине и других областях.

Показано, что задача СР является NP-трудной; в ряде различных областей предложены алгоритмы для ее решения, от эволюционных вычислений до байесовской оптимизации и глубокого обучения. Компьютерные эксперименты показывают, что современные алгоритмы СР, основанные на генетическом программировании (ГП), как правило, являются наилучшими

для решения реальных табличных задач. Более того, методы СР работают так же хорошо или даже лучше, чем методы машинного обучения, общепризнанные, как передовые, для подобных задач.

Задача регрессии может быть определена на основе множества значений входных независимых переменных *x* и зависимой выходной переменной *y*. Целью поиска является аппроксимация *y* с помощью переменных *x* и коэффициентов w следующим образом $y = f(x, w) + \varepsilon$, где ε представляет шум (ошибку).

В стандартних методах регресии вид функции f предполагается известным, например, в линейной регресии - $f(x, w) = w_0 + w_1 x_1 + ... + w_n x_n$. Здесь коэффициенты w_i обычно находятся методом наименьших квадратов. В нелинейных методах, например с использованием нейронных сетей прямого распространения, функция имеет вид $f(x, w) = w_0 \bullet g(w_h x)$. Здесь коэффициенты w_0 и w_h представляют синаптические веса нейронной сети выходного и скрытых слоев соответственно.

Как уже отмечалось, символьная регрессия на основе ГП не использует некоторую заранее предопределенную форму функции $f(x, w)$. Здесь функция $f(x, w)$ представляется древовидной структурой и строится эволюционным методом с использованием определенного функционального и терминального множеств. В качестве фитнесс-функции обычно используется квадратичная ошибка, которая оценивает качество решения и обеспечивает обратную связь при поиске решения. Для определенности обозначим функции множества, зависящие от одной переменной через $h_1,\ldots, h_k$ и функции от двух переменных как $g_1,\ldots, g_l$. В этой нотации функция $f(x, w)$ представляется в виде суперпозиции функций h_i , g_j, и например, может иметь следующий вид: $f(x, w) = h_1(g_2(g_1(x_3, w_1), h_2(x_1)))$.

Далее рассмотрим детально пример использования символьной регрессии из для аппроксимации данных, представленных в табл.4.3. Необходимо найти функцию f(x), которая аппроксимирует с заданной точностью эти

«экспериментальные» данные. Для решения задачи определим в соответствии с вышесказанным:

Терминальное множество: переменная x и константы в диапазоне [-5,5];

Функциональное множество: арифметические функции +, - , * , % (защищенное деление).

Koza ввел следующую удобную и «прозрачную» форму для перечисления параметров, которая представлена в табл.4.4.

Далее приведем некоторые полученные экспериментальные данные результатов эволюции для различных запусков программ из. В начальной популяции после инициализации лучшая особь представлена деревом рис.4.4, которая реализует (не минимальным образом !) функцию $f_0(x) = \frac{x}{3}$.последующих рисунках рис.4.5 –4.8 представлены лучшие особи последующих поколений. Здесь в первом поколении лучшая особь реализует $f_1(x) = \frac{x}{6-3x}$ В последующих рисунках рис.4.6 –4.8 представлены лучшие особи последующих поколений. Здесь в первом поколении лучшая особь реализует $f_1(x) = \frac{x}{6-3x}$ и соответствующее дерево рис.4.5 сильно избыточно.Аналогично во втором поколении лучшая особь реализует функцию $f_2(x) = \frac{x}{x(x-4)-1+\frac{4}{x}-\frac{\frac{9(x+1)}{5x}+x}{6-3x}}$ и дерево тоже сильно избыточно. Наконец в третьем поколении получена лучшая особь $f_3(x) = \frac{x^2}{2}$, которая дает оптимальное решение в простейшей форме.

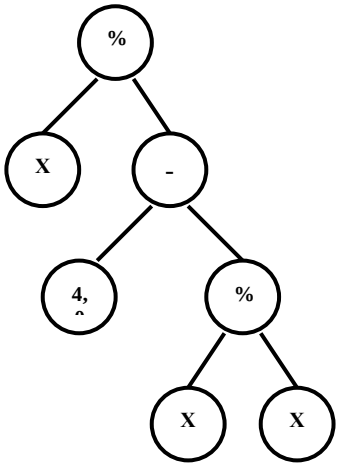


Рис.4.4. Лучшая особь в поколении 0.

Таблица 4.3

№	Вход x	Выход y
1	0.000	0.000
2	0.100	0.005
3	0.200	0.020
4	0.300	0.045
5	0.400	0.080
6	0.500	0.125
7	0.600	0.180
8	0.700	0.245
9	0.800	0.320
10	0.900	0.405

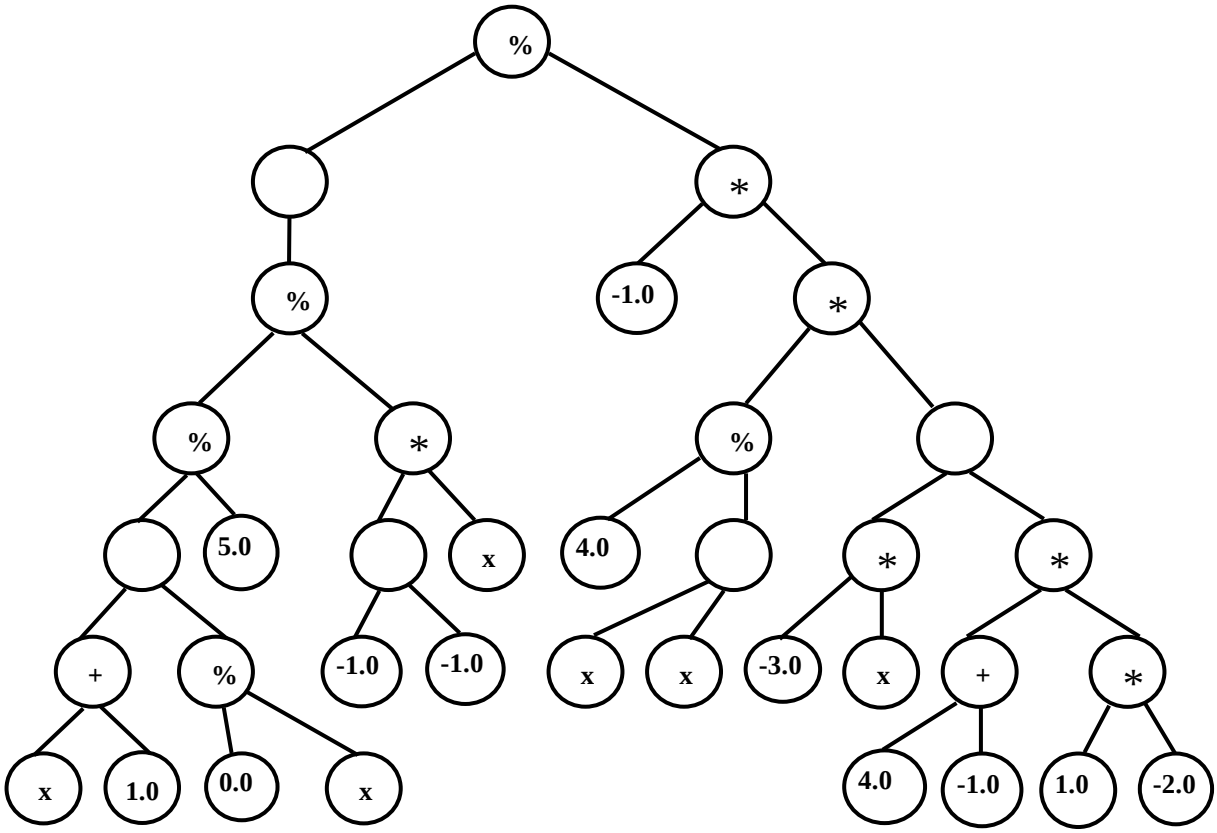


Рис.4.5 Лучшая особь поколения 1

Таблица 4.4

Параметры	Значения
Цель:	Эволюция функции, аппроксимирующей данные
Терминальное множество	Переменная x, Целые от –5 до +5
Функциональное множество	ADD, SUB, MUL, DIV
Мощность популяции: Вероятн. кроссинговера: Вероятность мутации: Отбор родителей: Максимальное число поколений: Максимальная глубина после кроссинговера: Максимальная глубина мутации: Метод инициализации:	600 0.90 0.05 турнирный, с мощностью тура 4 100 200 4 Растущая

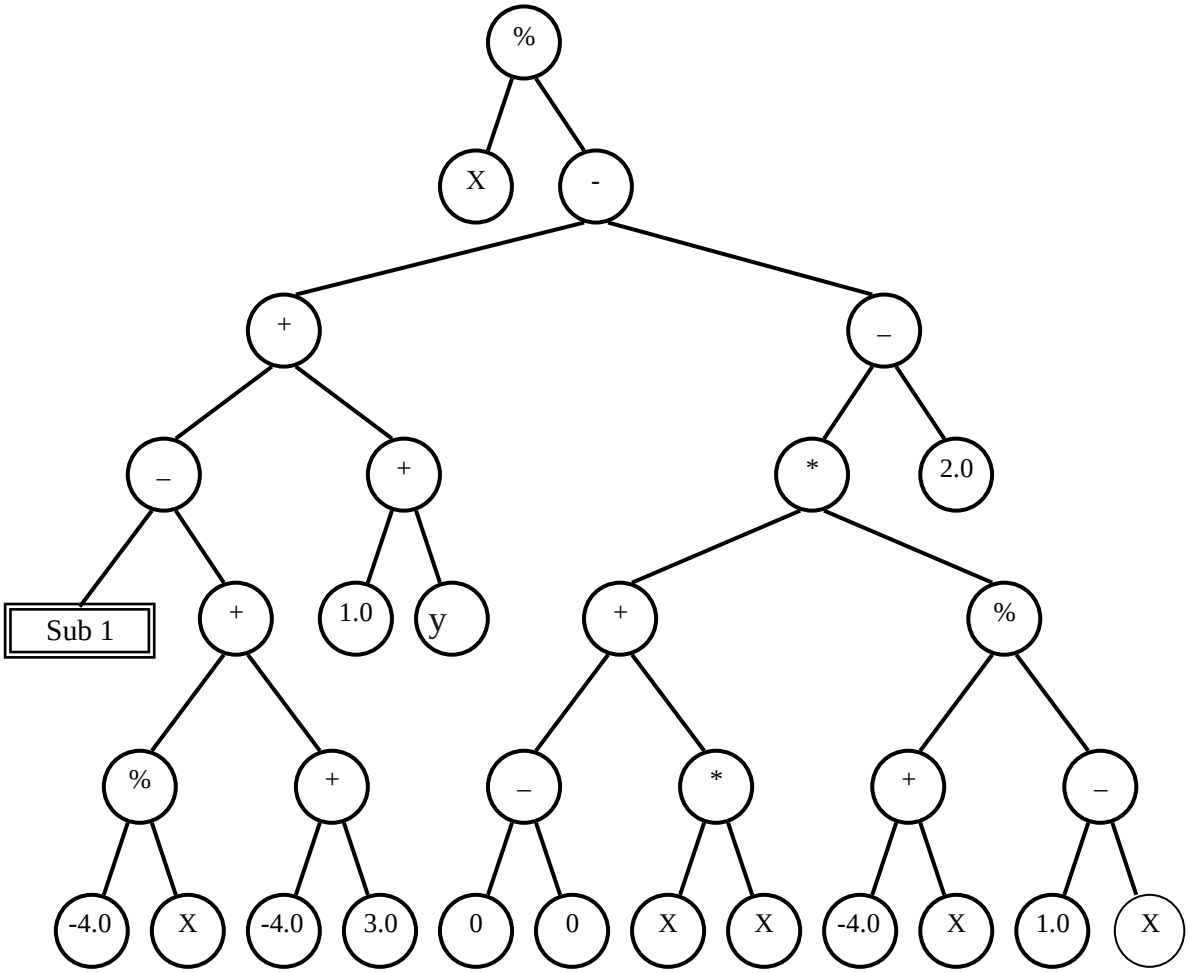


Рис.4.6. Лучшая особь поколения 2.

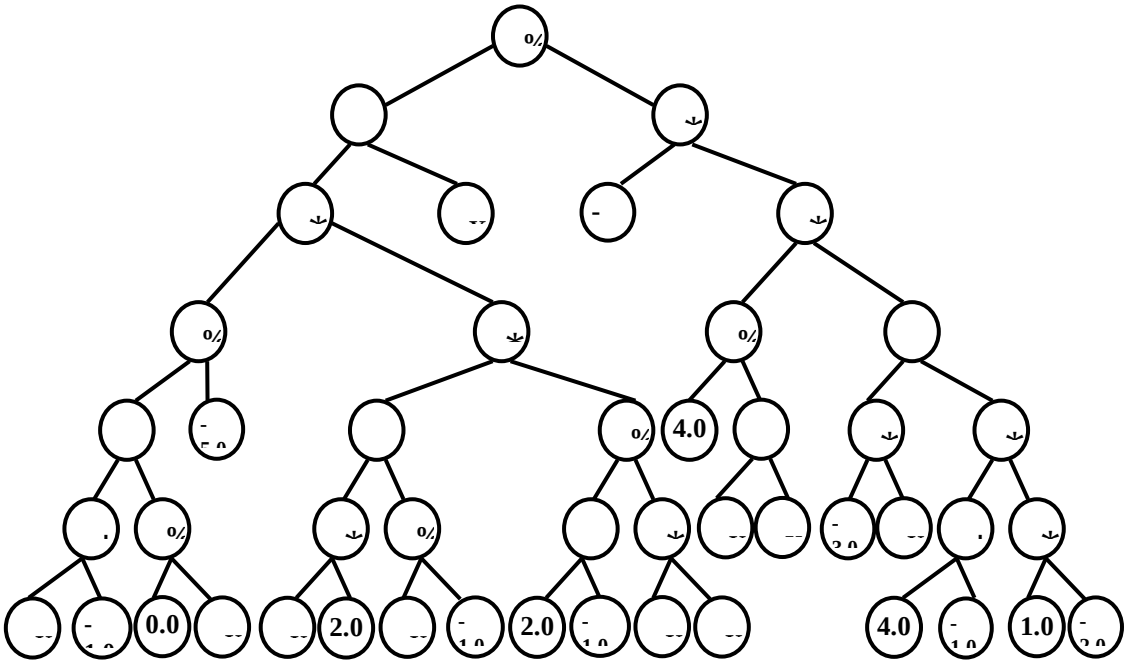


Рис.4.7. Лучшая особь поколения 3

В табл.6.11 для сравнения представлены значения функций лучших особей первых поколений (0-3).Отметим, что была выполнена еще одна итерация (4-е поколение). На рис.6.19 представлена лучшая особь четвертого поколения, которая также реализует функцию $f_4(x)=\frac{x^2}{2}$, но избыточность соответствующего дерева выросла. Конечно, для данного примера можно было использовать и классические методы регрессии, но он носит чисто иллюстративный характер.

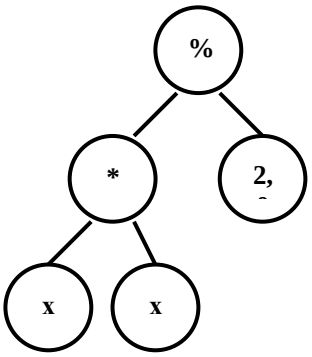


Рис.4.8. Лучшая особь поколения 4.

4.1 Практическая часть

Порядок выполнения лабораторной работы

1.При домашней подготовке:

1. изучить теоретический материал;
2. ознакомиться с типами структур для представления программы;
3. рассмотреть способы инициализации, начальной популяции;
4. рассмотреть способы выполнения операторов кроссинговера и мутации;
5. выполнить индивидуальное задание на любом языке высокого уровня с необходимыми комментариями и выводами.

1. Во время занятия:
    1. продемонстрировать результаты выполнения работы;
    2. получить допуск к защите лабораторной работы.
    3. защитить отчет по лабораторной работе.

Задание

1. Разработать эволюционный алгоритм, реализующий ГП в режиме символьной регрессии для нахождения заданной по варианту функции (таб. 4.1).
2. Структура для представления программы – древовидное представление.
3. Терминальное множество: переменные x_1, x_2, x_3,.. и константы в соответствии с заданием по варианту.
4. Функциональное множество: +, -, *, /, abs(), sin(), cos(), exp(), возведение в степень, корень квадратный,…
5. Выберите фитнесс-функцию – меру близости между реальными значениями выхода и требуемыми (квадратичную, абсолютную или другую ошибку).
6. Сгенерировать для заданной по варианту функции «псевдо-экспериментальные данные» в области исследования, которые выступают в качестве обучающего множества.
7. Программно реализовать символьную регрессию на «псевдо-экспериментальных данных».
8. Представить графически найденное решение (в виде дерева и формулы) на каждой итерации.
9. Сравнить найденное решение(полученную формулу) с представленным в условии задачи.
10. Вывести графики исходной и полученной путем символьной регрессии функций (разным цветом) и сравнить их.
11. Вывести таблицу значений параметров (типа табл.4.4), использованных в символьной регрессии.

Таблица 4.5. Тест функции символьной регрессии

№	Вид функции	Область исследования	Контрольный вопрос
1	$x^4 + x^3 + x^2 + x$	20 случайных точек $x \subseteq [-1,1]$	1
2	$x^4 - x^3 + x^2 - x$	20 случайных точек $x \subseteq [-1,1]$	2
3	$x^4 + 2x^3 + 3x^2 + 4x$	20 случайных точек $x \subseteq [-1,1]$	3
4	$x^6 - 2x^4 + x^2$	20 случайных точек $x \subseteq [-1,1]$	4
5	$\sin(x^2)\cos(x) - 1$	20 случайных точек $x \subseteq [-1,1]$	5
6	$\sin(x) + \sin(x + x^2)$	20 случайных точек $x \subseteq [-1,1]$	6
7	$\log(x + 1) + \log(x^2 + 1)$	20 случайных точек $x \subseteq [0,2]$	7
8	$\sqrt{x}$	20 случайных точек $x \subseteq [0,4]$	8
9	$\sin(x) + \sin(y^2)$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	9
10	$2\sin(x)\cos(y)$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	10
11`	$0.4x_1x_2 - 1.5x_1 + 2.5x_2 + 1$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	11`
12	$\frac{0.4x_1x_2 - 1.5x_1 + 2.5x_2 + 1}{0.2(x_1^2 + x_2^2) + 1}$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	12
13	$\frac{0.4x_1x_2 - 1.5x_1 + 2.5x_2 + 1 + 5.5\sin(x_1 + x_2)}{0.2(x_1^2 + x_2^2) + 1}$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	13
14	$\frac{30xz}{(x-10)y^2}$	100 случайных точек $x \subseteq [-1,1] \times [-1,1] \times [-1,1]$	14
15	$\frac{e^{-(x_1-1)^2}}{1.2 + (x_2 - 2.5)^2}$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	15
16	$6.87 + 11\sqrt{7.23x_0x_3x_4}$	100 случайных точек $x \subseteq [-1,1] \times [-1,1]$	16

Содержание отчета.

1. Титульный лист.
2. Индивидуальное задание по варианту.
3. Краткие теоретические сведения.
4. Программа и результаты выполнения индивидуального задания с комментариями и выводами.
5. Письменный ответ на контрольный вопрос по варианту (номер контрольного вопроса совпадает с номером варианта).

Контрольные вопросы

1. Чем отличаются терминальное и функциональное множества?
2. Какие структуры используются для представления программ в ГП?
3. Опишите древовидное представление.
4. Опишите линейное представление программы.
5. Опишите представление программы в виде графа.
6. Какие знаете методы инициализации древовидных структур?
7. Как производится инициализация линейных структур?
8. Какие виды кроссинговера вы знаете для древовидных структур?
9. Как выполняется кроссинговер на линейных структурах?
10. Какие виды кроссинговера вы знаете для графоподобных структур?
11. Какие виды мутации вы знаете для древовидных структур?
12. Как производится мутация на линейных структурах?
13. Как можно определить фитнесс-функцию в ГП?
14. Что такое интроны?
15. Приведите общий алгоритм ГП.
16. Какие фитнесс-функции используются в символьной регрессии.