Московский государственный университет имени М.В. Ломоносова
физический факультет
NeuroFuzzy Lab.

Системы «интеллектуального» программирования

Зубюк Андрей Владимирович
zubjuk@physics.msu.ru
http://NeuroFuzzy.Phys.MSU.ru

МФК «Искусственный интеллект в естествознании»

«Обычное» программирование — императивная парадигма

Любой алгоритм может быть представлен в виде последовательности команд из некоторого конечного набора, реализуемых на вычислительной машине, имеющей память конечного объёма.

Современные процессоры умеют выполнять элементарные команды: перемещение данных в памяти, арифметические операции, переходы (условные или безусловные) к другим частям программы.

Но мы привыкли иметь дело с **языками программирования** высокого уровня **(Basic, C/C++, Pascal, Fortran)** с конструкциями if, for, while и др., функциями (процедурами), классами и т. п.

В любом случае **императивная** программа — это последовательность операций, **заданная программистом** и нацеленная на **решение определённой задачи**:

- составление правильной последовательности операций — работа программиста,
- программист может допустить ошибку, тогда заданная им последовательность операций не приведёт к решению задачи.

Типичные «грабли» новичка

Программист-новичок не в состоянии понять, чего от него «хочет» вычислительная машина. Не в состоянии чётко сформулировать алгоритм решения задачи, выразить его на языке программирования.

Типичные «грабли» новичка

- Программист составляет последовательность действий, которая, как ему кажется, решит поставленную задачу.
- Программа не приводит к желаемому результату.
- Программист-новичок не в состоянии объективно оценить свою программу и понять, как заданная последовательность действий «воспринимается» вычислительной машиной. Это приводит к недоумению и отторжению: «я всё запрограммировал правильно, а оно не работает!».

Системы «интеллектуального» программирования

Основная идея — программист задаёт не алгоритм решения задачи, а условия задачи. Решение задачи — «ответственность» системы, которая выполняет программу, а не программиста.

- Реляционные базы данных. Наиболее распространённый **язык программирования — SQL**.
- Логическое программирование. Наиболее распространённый **язык программирования — Prolog** (programming in logic).
- Функциональное программирование. Наиболее распространённый в настоящее время «чисто» функциональный **язык программирования — Haskell**. Но функциональная парадигма широко используется в системах, написанных на других языках, в т. ч. в системах символьных вычислений (Wolfram Mathematica), современных системах ИИ (TensorFlow, PyTorch и др.).

Идея реляционных баз данных (Edgar Codd, 1970)

Вся имеющаяся информация представляется в форме **отношений (relation)** между объектами. Система (управления базами данных — СУБД) реализует операции над отношениями (обработку запросов).

Пример отношения — сравнение чисел. $x \leqslant y$ — это множество точек (x, y) на плоскости, для которых координата x не меньше, чем y.

Т. е. отношение — это множество пар объектов. В общем случае — объектов (унарные отношения), пар (бинарные отношения), троек (триарные), четвёрок и т. д.

Примеры отношений и запросов

«Поставщик-деталь»

Пост.	Дет.
П-1	Д-2
П-2	Д-1
П-3	Д-1

«Деталь-изделие»

Дет.	Изд.
Д-1	И-2
Д-2	И-2
Д-2	И-1

«Поставщик-деталь-изделие»
(объединение отношений)

Пост.	Дет.	Изд.
П-1	Д-2	И-1
П-1	Д-2	И-2
П-2	Д-1	И-2
П-3	Д-1	И-2

Запрос: найти всех поставщиков, которые могут поставлять делали для изделия И-1.
Результат может быть получен с использованием триарного отношения «поставщик-деталь-изделие». Результат — унарное отношение.

Пост.
П-1

Наиболее распространённый **язык программирования, позволяющий заполнять реляционные базы данных и формировать запросы — SQL**.

Тест

При прохождении теста укажите e-mail и фамилию, имя, отчество, которые вы указали при регистрации на курс. По этим данным будут суммироваться результаты всех ваших тестов в семестре. После ответа на тест вам на почту должно прийти письмо с вашими ответами.

https://docs.google.com/forms/d/e/1FAIpQLSeDqF9yaqMnwOJlLLL2Bj0KqCibjAZjuJbKWgnxW5NiBlOP8g/viewform

Логическое программирование

Идея логического программирования во многом схожа с идеей реляционных баз данных:

1. Сначала программист задаёт все имеющиеся в его распоряжении данные — формирует базу знаний.
2. Затем программист формирует запросы, а система логического программирования ищет ответы на запросы.

Основным понятием логического программирования является **предикат** — функция, принимающая значения «истина» и «ложь».

Исходные данные задаются в форме предикатов: фактов и правил. Запросы могут иметь вид:

- Истинен ли предикат (имеет ли он значение «истина»)?
- В каких случаях предикат истинен?

Наиболее распространённый **язык логического программирования — Prolog**.

Примеры фактов, правил и запросов

Определим следующие предикаты:
«ребёнок-родитель», «братья», «племянник-дядя».

Факты:
«Иван является ребёнком Павла», «Сергей является ребёнком Павла»,
«Пётр и Павел — братья».

Правила:
«X является племянником Y, если и только если X является ребёнком Z, и Z и Y — братья».

Запросы:

- Является ли Иван племянником Петра? Да (предикат «племянник-дядя» истинен).
- Кто является племянником Петра? Иван, Сергей (предикат «племянник-дядя» истинен для Ивана и Петра, Сергея и Петра).
- Кто является дядей Сергея? Пётр (предикат «племянник-дядя» истинен для Сергея и Петра).

Автоматическое доказательство теорем

Системы логического программирования могут использоваться (и во многом задумывались) для автоматизации доказательства математических теорем.

В этом случае

- факты — это аксиомы математической теории или уже доказанные теоремы,
- правила — это математические определения, задающие связи между математическими понятиями,
- доказательство теорем — это проверка истинности предикатов.

Функциональное программирование

Идея функционального программирования — **всё есть функция**.

Функция понимается в математическом смысле — как правило, которое каждому значению из области определения ставит в соответствие одно единственное определённое значение из области значений.

Запись $f : X \to Y$ означает, что функция f определена на множестве X (область определения) и принимает значения на множестве Y (область значений).

Функция или процедура

В императивном программировании функцией часто называют процедуру, подпрограмму, т.е. обособленную поименованную часть целой программы, которая может быть вызвана из других частей программы.

Процедуры, не являющиеся функциями в математическом смысле:

```
void print_integer(int x) {
  printf("An␣integer␣to␣print␣is␣...
   %d\n", x);
}
```

```
int s = 0;
int cumsum(int x) {
    s = s + x;
    return s;
}
```

Процедуры, играющие роль «истинных» функций: sin(), cos() и др.

Возможности «истинно» функциональных систем

NeURO FUzzY

В «истинно» функциональных системах **функция** — такой же объект, как обычная переменная в императивном программировании. С функциями можно совершать операции, т. е. передавать их в качестве аргументов другим функциям.

Примеры:

- Производная — это функция diff(), аргументом и значением которой является функция: $\mathrm{diff}(\sin(x)) = \cos(x)$, $\mathrm{diff}(x^2) = 2x$ и т. д..
- Поиск минимума — это функция $\arg\min()$, аргументом которой является функция, значением — элемент области определения, при котором достигается минимум: $\arg\min\limits_x \left(x^2 - 2x + 7\right) = 2$.

Функциональная парадигма играет ключевую роль в современных системах машинного обучения, в т. ч. обучения искусственных нейронных сетей, основная задача которых — **автоматическая минимизация** сложных функций, обычно методами градиентного спуска, в основе которых лежит **автоматическое вычисление производных**.