ПРОГРАММНАЯ ГЕНЕРАЦИЯ СЛУЧАЙНЫХ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ

Прохоров К.В.,
студент 2 курса ВГУ имени П.М. Машерова, г. Витебск, Республика Беларусь;
Научный руководитель – Ермоченко С.А., канд. физ.-мат. наук, доцент

В ряде программных продуктов образовательной направленности из области математики актуальной проблемой является случайная генерация некоторой математической формулы, например, для генерации условия задач на контрольную работу или экзамен [3]. Но на настоящий момент не существует достаточно гибкого инструмента, решающего такую задачу.

Целью данной работы является выработка концепции программного модуля для генерации случайных математических функций одной переменной как суперпозиции различных элементарных математических функций со случайными числовыми параметрами.

Одним из основных требований к генератору является представление функции в программном коде в таком виде, который обеспечивал в дальнейшем удобную обработку этой функции. В качестве одного из примеров подобной обработки рассматривается нахождение производной сгенерированной функции.

Материалы и методы. Материалом для работы послужила идея представления функции одной переменной как суперпозиции элементарных математических функций. Для реализации программного модуля решено было использовать в качестве методов исследования принципы объектно-ориентированного программирования (ООП) и язык программирования Java, а также общенаучные методы анализа и обобщения.

Результаты и их обсуждение. Для решения задачи внутреннего представления математических функций в программном коде воспользуемся шаблоном проектирования (ШП) Decorator [1], для чего описывается абстрактный класс ElementaryFunction, реализующий стандартный интерфейс Function<Double, Double>. В данном абстрактном классе, в соответствии с ШП Decorator, описывается поле – ссылка на объект класса, реализующего интерфейс Function<Double, Double>, которая является аргументом элементарной функции, а также абстрактный защищённый метод calc(), принимающий единственный параметр типа double, и возвращающий результат типа double. В стандартном методе интерфейса Function.apply() реализуется суперпозиция двух функций, см. листинг:

```
abstract public class ElementaryFunction implements Function<Double, Double> {
    private Function<Double, Double> function;
    public ElementaryFunction(Function<Double, Double> function) {
        this.function = function;
    }
    @Override
    public final Double apply(Double x) {
        return calc(function != null ? function.apply(x) : x);
    }
    protected abstract double calc(double x);
}
```

Аналогично через ШП «Composite» [1] реализуется сумма, разность, произведение и частное двух функций:

```
abstract public class CompositeFunction implements Function<Double, Double>
{
    private Function<Double, Double> function1;
    private Function<Double, Double> function2;
    public ElementaryFunction(Function<Double, Double> function1,
Function<Double, Double> function2) {
        this.function1 = function1;
        this.function2 = function2
    }
    @Override
    public final Double apply(Double x) {
        return calc(function1 != null ? function1.apply(x) : x,
function2 != null ? function2.apply(x) : x);
    }
    protected abstract double calc(double u, double v);
}
```

Сам генератор случайных функций реализуется через ШП Builder [1], который позволяет для каждой элементарной функции случайным образом генерировать коэффициенты (если они есть). При этом, так как в ШП Builder каждому классу иерархии элементарных функций соответствует отдельный подкласс-строитель, а, возможно, и несколько таких подклассов, то можно гибко реализовать различные способы генерации. Также генератор итоговой случайной функции может выбирать случайным образом набор строителей из подготовленного под конкретную задачу подмножества имеющихся строителей, а также комбинацию этих строителей, определяя уровень вложенности функций в суперпозициях и их последовательность.

Рассмотрим теперь, как построенная иерархия позволит обрабатывать сгенерированные функции на примере вычисления производных. Для нахождения производной функции, представленной в виде композиции объектов различных классов, построенной в соответствии с ШП Decorator и Builder, используется идея ШП Template method [1], в реализации которого используются возможности обобщений (generics) и интроспекции (reflection) в языке программирования Java [2]. В этом случае создадим иерархию классов, вычисляющих производную каждой элементарной функции. В вершине иерархии таких вычислителей опишем параметризованный интерфейс DerivativeCalculator, который будет использовать обобщённый тип T – наследник интерфейса Function<Double, Double>. В интерфейсе опишем метод для вычисления производной, возвращающий класс, реализующий интерфейс Function<Double, Double>:

```
public interface DerivativeCalculator <T extends Function<Double, Double> {
    Function<Double, Double> calc(T function);
}
```

Для каждого подкласса классов ElementaryFunction и CompositeFunction будет создавать свой подкласс, реализующий интерфейс DerivativeCalculator, в котором вместо обобщённого типа T будет подставляться исходный подкласс-наследник ElementaryFunction или CompositeFunction.

Заключение. Таким образом, в работе рассмотрен подход для генерации случайной функции, являющейся суперпозицией элементарных функций. Показана идея дальнейшей работы с полученной иерархией объектов для решения задачи нахождения

производных функций. Перспективой реализации проекта является построение метрик оценки сложности функции.

1. Гамма Э. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – Санкт-Петербург: Питер, 2020. – 368 с.

2. Гослинг Дж. Язык программирования Java SE 8. Подробное описание, 5-е издание / Дж. Гослинг, Б. Джой, Г. Стил, Г. Брача, А. Бакли. – Москва: Вильямс, 2015. – 672 с.

3. Ермоченко, С. А. Концепция визуализации данных в учебном приложении для дисциплины «Теоретическая механика» / Наука – образованию, производству, экономике: материалы XXIV (71) Регион. науч.-практ. конф. преподавателей, научн. сотрудников и аспирантов, Витебск, 14 февраля 2019 г.: в 2 т. – Витебск: ВГУ имени П. М. Машерова, 2019. – Т. 1. – С. 9-11. URL: https://rep.vsu.by/handle/123456789/17787.

ПРИМЕНЕНИЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ В ВООРУЖЕННЫХ СИЛАХ

Романовский Н.Д.,
студент 3 курса Белорусского государственного университета информатики и радиоэлектроники, г. Минск, Республика Беларусь
Научный руководитель – Богатырев А.А., канд. воен. наук

Современные вооруженные силы постоянно ищут новые способы улучшения своей эффективности и эффективности своего оборудования. Одним из перспективных направлений в этом направлении является применение искусственных нейронных сетей (ИНС).

Искусственные нейронные сети – это математическая модель, которая моделирует работу нервной системы живых организмов и может использоваться для решения различных задач в области искусственного интеллекта. ИНС состоит из множества связанных между собой элементов, которые называются искусственными нейронами. Искусственные нейроны имеют входы для входных данных, выходы для выходных данных и веса, которые определяют важность каждого входа для выхода. Информация передается между нейронами через связи.

Целью данного проекта является изучить применение ИНС в современных вооруженных силах, а также потенциальное их развитие.

Материал и методы. В качестве материалов использована литература, в которой описываются нейронные сети [1], а также глубоков обучение ИНС [2]. Методами в данном проекте является изучение литературы, связанной с ним.

Результаты и их обсуждение. ИНС являются мощным инструментом в области обработки информации и принятия решений, они нашли применение во многих аспектах военной деятельности.

Одним из основных применений ИНС в военной сфере является обработка информации с различных датчиков, таких как радары, оптические и инфракрасные приборы, беспилотные летательные аппараты и другие, с целью определения обстановки на поле боя и выявления потенциальных угроз. ИНС также могут использоваться для распознавания образов, таких как транспортные средства, люди, здания и другие объекты на земле, в воздухе или на воде, что помогает обнаруживать и идентифицировать цели.

Кроме того, ИНС могут применяться в автоматизации процессов принятия решений, например, при выборе наиболее эффективного маршрута или определении оптимальных тактик действий в конкретной ситуации. Они также могут использоваться для прогнозирования поведения противника на основе анализа его действий и ранее собранных данных.

ИНС также широко используются для управления беспилотными техническими средствами (беспилотные летательные аппараты (БПЛА), беспилотные машины, подводные аппараты и т.д.). Использование ИНС позволяет беспилотным системам выполнять