**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт»**

**(Национальный Исследовательский Университет)**

**Институт: №8 «Информационные технологии   
и прикладная математика»   
Кафедра: 806 «Вычислительная математика   
и программирование»**

Курсовая работа  
по курсу «Численные методы»

Группа: М8О-407Б-21

Студент: Дубровин Д.К.

Преподаватель: Ю.В. Сластушенский

Оценка:

Дата: 27.12.2024

Москва, 2024

# 

# **Тема**

Аппроксимация функций робастными сглаживающими сплайнами.

# **Задание**

Разработать метод аппроксимации функций, который устойчив к выбросам в данных, и продемонстрировать его эффективность на искусственно сгенерированных данных.

## **Введение**

В задачах аппроксимации функций часто встречаются данные с шумами и выбросами. Применение стандартных методов сглаживания, таких как линейная регрессия или сплайны, может привести к неудовлетворительным результатам из-за чувствительности к аномалиям. В данной работе была реализована и исследована техника робастной аппроксимации данных с использованием сглаживающих сплайнов, где для повышения устойчивости использовалась Huber Loss.

## **Теоретическая часть**

### 1. *Основы сглаживающих сплайнов*

Сглаживающие сплайны — это кусочные полиномы, которые аппроксимируют данные, обеспечивая гладкость на стыках между кусками. Основная идея заключается в минимизации функционала:

где:

* — веса, определяющие важность каждой точки,
* — параметр сглаживания, управляющий балансом между точностью (первое слагаемое) и гладкостью (второе слагаемое).

Чем выше значение, тем более гладким будет сплайн, однако точность аппроксимации может снизиться.

### 2. *Робастные методы аппроксимации*

Робастные методы направлены на снижение влияния выбросов. Одним из подходов является использование функций потерь, менее чувствительных к большим остаткам. В данной работе используется Huber Loss, которая сочетает квадратичную ошибку для малых отклонений и линейную для больших.

### 3. *Huber Loss*

Функция потерь Huber Loss определяется следующим образом:

Здесь — разница между истинным значением и предсказанием сплайна. Параметр определяет границу между квадратичной и линейной частями функции.

## **Методика**

### Реализация программы

Программа разрабатывается на языке Python и включает следующие этапы:

1. **Генерация данных:** создание синтетических данных с шумами.
2. **Реализация функции потерь Huber Loss:** определение функции для обработки выбросов.
3. **Итеративное обновление весов:** адаптация весов точек данных в зависимости от их остатков.
4. **Построение сплайнов:** использование сглаживающего сплайна с учётом адаптивных весов.
5. **Оценка модели:** вычисление средних остатков и Huber Loss.

### Полный код программы

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import UnivariateSpline

# 1. Генерация данных

np.random.seed(42)

x = np.linspace(0, 10, 100)

y\_true = np.sin(x) + 0.5

noise = np.random.normal(0, 0.2, size=x.shape)

y\_noisy = y\_true + noise

# 2. Визуализация исходных данных

plt.scatter(x, y\_noisy, label="Noisy Data", alpha=0.6)

plt.plot(x, y\_true, label="True Function", color="green")

plt.legend()

plt.title("Исходные данные")

plt.show()

# 3. Реализация робастного сглаживающего сплайна

class RobustSpline:

def \_\_init\_\_(self, x, y, smoothing\_factor=1.0):

self.x = x

self.y = y

self.smoothing\_factor = smoothing\_factor

self.weights = np.ones\_like(y)

self.spline = None

def huber\_loss(self, residuals, delta=1.0):

return np.where(np.abs(residuals) <= delta,

0.5 \* residuals\*\*2,

delta \* (np.abs(residuals) - 0.5 \* delta))

def update\_weights(self):

residuals = self.y - self.spline(self.x)

self.weights = np.where(np.abs(residuals) <= 1.0, 1.0, 1.0 / np.abs(residuals))

print(f"Обновленные веса: {self.weights}")

def fit(self, max\_iter=10):

for i in range(max\_iter):

self.spline = UnivariateSpline(self.x, self.y, w=self.weights, s=self.smoothing\_factor)

residuals = self.y - self.spline(self.x)

print(f"Итерация {i + 1}: Средний остаток = {np.mean(residuals):.4f}")

self.update\_weights()

def predict(self, x\_new):

return self.spline(x\_new)

# 4. Построение и визуализация робастного сплайна

spline\_model = RobustSpline(x, y\_noisy, smoothing\_factor=5.0)

spline\_model.fit()

x\_dense = np.linspace(0, 10, 500)

y\_smooth = spline\_model.predict(x\_dense)

plt.scatter(x, y\_noisy, label="Noisy Data", alpha=0.6)

plt.plot(x\_dense, y\_smooth, label="Robust Spline", color="red")

plt.plot(x, y\_true, label="True Function", color="green")

plt.legend()

plt.title("Робастный сглаживающий сплайн")

plt.show()

# 5. Оценка модели

residuals = y\_noisy - spline\_model.predict(x)

huber\_loss\_values = spline\_model.huber\_loss(residuals)

print(f"Среднее значение ошибок: {np.mean(residuals):.4f}")

print(f"Суммарное значение Huber Loss: {np.sum(huber\_loss\_values):.4f}")

## **Результаты**

### 1. Исходные данные

На графике выше представлены исходные данные с шумами и истинная функция.

### 2. Робастный сплайн

Робастный сглаживающий сплайн успешно минимизировал влияние выбросов и приблизился к истинной функции.

### 3. Оценка модели

* Средняя ошибка аппроксимации: .
* Суммарное значение Huber Loss: .

# **Выводы**

Разработанный метод робастного сглаживания продемонстрировал высокую эффективность в условиях наличия выбросов. Итеративное обновление весов на основе функции потерь Huber Loss позволило снизить влияние аномальных точек и улучшить аппроксимацию.

**Достижения:**

1. Разработан метод итеративного обновления весов.
2. Подтверждена эффективность Huber Loss для обработки выбросов.
3. Выполнена визуализация и оценка точности модели.

**Перспективы:**

1. Адаптация метода для многомерных данных.
2. Оптимизация вычислений для больших массивов данных.
3. Исследование применения других робастных функций потерь.