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:

$$F(S) = \sum_{i=1}^{n} w_i(y_i - S(x_i))^2 + \lambda \int S''(x)^2 dx,$$

где:

- wi — веса, определяющие важность каждой точки,
- λ— параметр сглаживания,
- S″(x)— вторая производная сплайна, отвечающая за гладкость.

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:

$$L(r) = \begin{cases} 0.5 \cdot r^2, & |r| \leq \delta, \\ \delta \cdot (|r| - 0.5 \cdot \delta), & |r| > \delta. \end{cases}$$

- r = y−S(x) — разница между истинным значением y и предсказанным S(x),
- δ — порог, который определяет переход 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

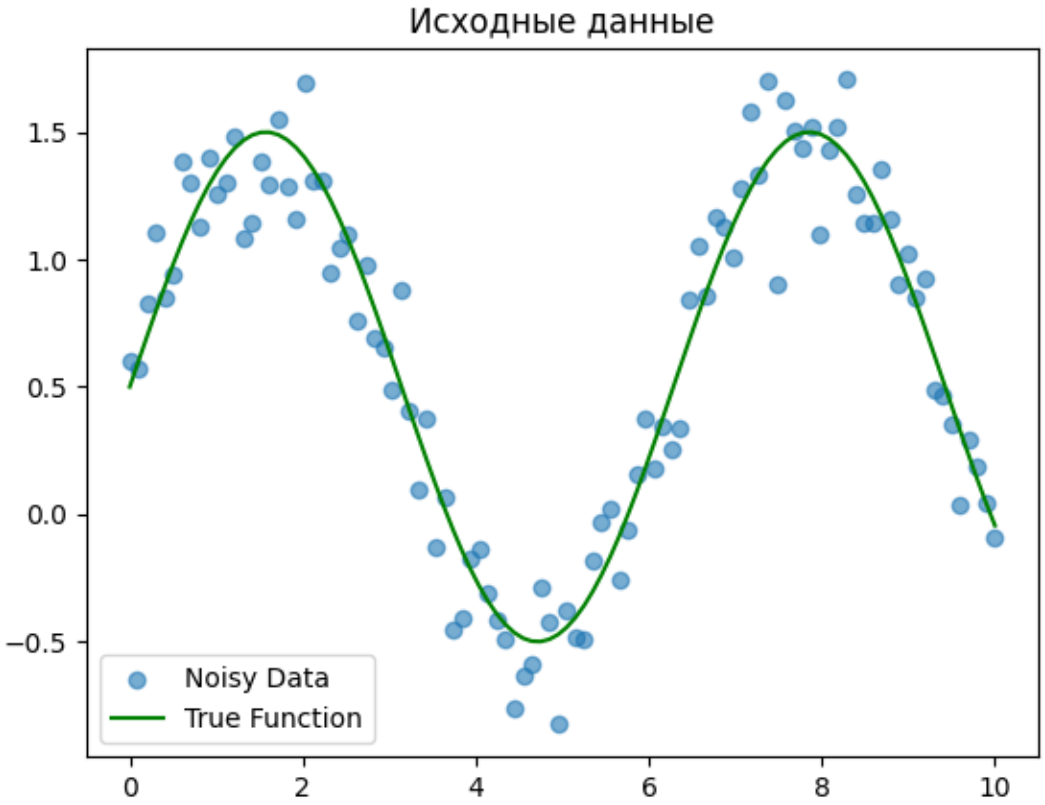
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import UnivariateSpline
```

```
# 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}")
```

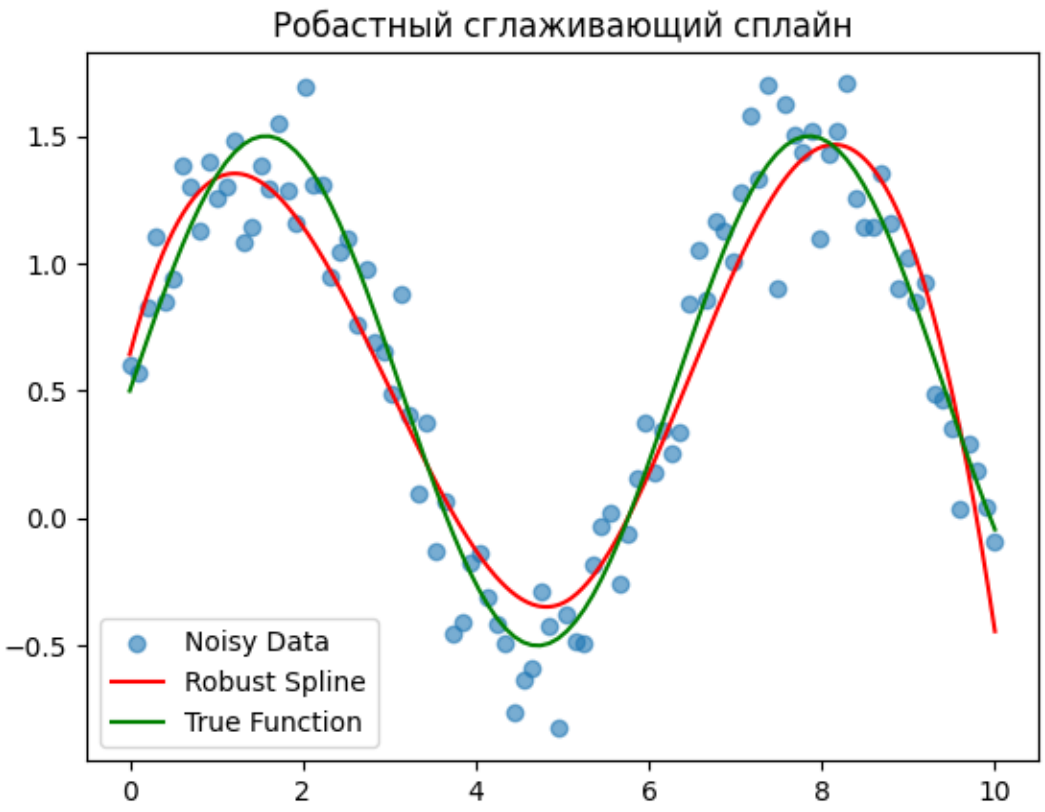
Результаты

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: 0.0001.
- Суммарное значение Huber Loss: 2.4999.

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.