Для построения сложного B-сплайна в Python необходимо понимать основные теоретические и математические концепции, связанные с B-сплайнами. Вот основные аспекты, которые нужно знать:

Теоретические основы B-сплайнов

Поговорим о самом простом B-сплайне:

1. **Определение B-сплайнов**:
    - B-сплайны (B-splines) — это кусочно-полиномиальные функции, которые используются для аппроксимации кривых и поверхностей. Они определяются набором контрольных точек, узловым вектором и степенью сплайна.
2. **Базисные функции B-сплайнов**:

- Базисные функции B-сплайнов $N_{i,p}(u)$ определяются рекурсивно. Они задают форму и гладкость сплайна.
- Рекурсивные формулы для базисных функций:

$$N_{i,0}(u) = \begin{cases} 1, & \text{если } T_i \leq u < T_{i+1} \\ 0, & \text{иначе} \end{cases}$$

$$N_{i,p}(u) = \frac{u - T_i}{T_{i+p} - T_i} N_{i,p-1}(u) + \frac{T_{i+p+1} - u}{T_{i+p+1} - T_{i+1}} N_{i+1,p-1}(u)$$

1. **Контрольные точки**:
    - Контрольные точки определяют форму сплайна. Они могут быть двумерными (для кривых) или трехмерными (для поверхностей).
2. **Узловой вектор**:
    - Узловой вектор *T* определяет сегменты сплайна. Он должен быть согласован с количеством контрольных точек и степенью сплайна.
3. **Степень сплайна**:
    - Степень сплайна *p* определяет гладкость кривой. Например, кубические B-сплайны имеют степень 3.

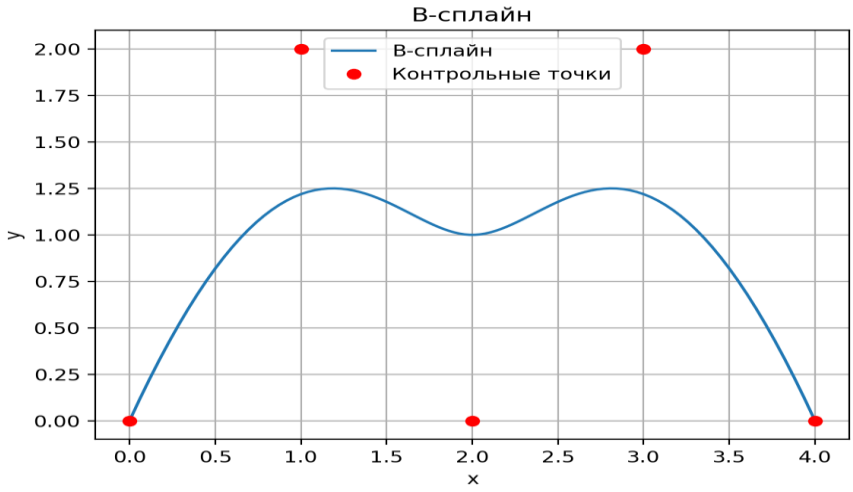
Практическая реализация в Python

1. **Библиотеки**:
    - Использование библиотек **numpy** для работы с массивами и матрицами.
    - Использование библиотеки **scipy** для работы с B-сплайнами (класс **BSpline**).
    - Использование библиотеки **matplotlib** для визуализации сплайнов.

2. **Пример кода и график**:

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

# Контрольные точки
control_points = np.array([[0, 0], [1, 2], [2, 0], [3, 2], [4, 0]])

# Степень сплайна
degree = 3

# Количество контрольных точек
n = len(control_points)

# Узловой вектор
knots = np.concatenate((np.zeros(degree), np.linspace(0, 1, n - degree + 1), np.ones(degree)))

# Создание B-сплайна для каждой координаты
spline_x = BSpline(knots, control_points[:, 0], degree)
spline_y = BSpline(knots, control_points[:, 1], degree)
```

```python
# Построение графика
t = np.linspace(0, 1, 100)
x = spline_x(t)
y = spline_y(t)

plt.plot(x, y, label='B-сплайн')
plt.plot(control_points[:, 0], control_points[:, 1], 'ro', label='Контрольные точки')
plt.xlabel('x')
plt.ylabel('y')
plt.title('B-сплайн')
plt.legend()
plt.grid(True)
plt.show()
```



Код проверен

Поговорим о том какие сплайны вообще бывают:

1. Линейные сплайны

Линейные сплайны — это кусочно-линейные функции, которые соединяют точки данных прямыми линиями. Они просты в реализации и используются для интерполяции данных, когда не требуется высокая степень гладкости.

2. Кубические сплайны

Кубические сплайны — это кусочно-кубические функции, которые обеспечивают гладкость до второго порядка (непрерывность первой и второй производных). Они широко используются для интерполяции данных и аппроксимации кривых.

3. Безье-сплайны (Bezier splines)

Безье-сплайны — это параметрические кривые, которые определяются набором контрольных точек. Они обеспечивают гладкость и простоту управления формой кривой. Безье-сплайны часто используются в компьютерной графике и анимации.

4. Гермитовы сплайны (Hermite splines)

Гермитовы сплайны — это кусочно-кубические функции, которые используют значения функции и её производной в узлах для определения кривой. Они обеспечивают гладкость и точность интерполяции.

5. Каталонские сплайны (Catmull-Rom splines)

Каталонские сплайны — это кубические сплайны, которые используют контрольные точки для определения кривой. Они обеспечивают гладкость и простоту управления формой кривой. Каталонские сплайны часто используются в компьютерной графике и анимации.

6. Сплайны с минимальной энергией (Minimum Energy Splines)

Эти сплайны минимизируют интеграл квадрата второй производной, что обеспечивает гладкость и минимальную кривизну кривой. Они используются в различных областях, включая компьютерную графику и анализ данных.

7. Сплайны с натяжением (Tension splines)

Сплайны с натяжением включают параметр натяжения, который позволяет контролировать степень кривизны кривой. Они используются для создания более реалистичных кривых в компьютерной графике и анимации.

Следующие сплайны из проектов Парфа:

8. Приближение Z-сплайнами

Z-сплайны — это тип сплайнов, который используется для аппроксимации данных с учетом гладкости и локального управления. Они могут быть рассмотрены как расширение B-сплайнов с дополнительными свойствами, такими как адаптивность и гибкость. Z-сплайны часто используются в компьютерной графике и проектировании.

9. Приближение T-сплайнами

T-сплайны — это расширение NURBS, которое позволяет более гибкое управление формой кривой или поверхности. T-сплайны используют T-сетки для определения контрольных точек, что позволяет избежать необходимости в регулярной сетке узлов. Это делает их особенно полезными для моделирования сложных форм в CAD-системах и компьютерной графике.

10. Приближение S-сплайнами

S-сплайны (Smoothed splines) — это тип сплайнов, который использует сглаживание для улучшения аппроксимации данных. S-сплайны могут быть рассмотрены как расширение B-сплайнов с дополнительными свойствами сглаживания. Они часто используются в анализе данных и статистике для создания гладких кривых и поверхностей.

11. Приближение PHT-сплайнами

PHT-сплайны (Progressive Hierarchical T-splines) — это расширение T-сплайнов, которое использует иерархическую структуру для улучшения аппроксимации данных. PHT-сплайны позволяют более гибкое управление формой кривой или поверхности, что делает их полезными для моделирования сложных форм в CAD-системах и компьютерной графике.

12. Приближение сплайнами со штрафами (в том числе P-сплайнами)

Сплайны со штрафами (Penalized splines) — это тип сплайнов, который использует штрафные функции для регуляризации. P-сплайны (P-splines) — это частный случай сплайнов со штрафами, которые используют B-сплайны с дополнительными штрафными функциями для улучшения гладкости и предотвращения переобучения. P-сплайны часто используются в анализе данных и статистике для аппроксимации данных.

13. Приближение сплайнами Коханек-Бартельса (Kochanek-Bartels TCB-сплайнами)

Сплайны Коханек-Бартельса (Kochanek-Bartels TCB-splines) — это тип сплайнов, который использует параметры натяжения, скручивания и угла для управления формой кривой. TCB-сплайны позволяют создавать более реалистичные и гибкие кривые, что делает их полезными в компьютерной графике и анимации.

14. Приближение бета-сплайнами

Бета-сплайны (Beta-splines) — это тип сплайнов, который использует бета-функции для представления кривых и поверхностей. Бета-сплайны обеспечивают гладкость и гибкость, что делает их полезными для моделирования сложных форм в компьютерной графике и проектировании.

15. Интегрирование с помощью сплайнов

Интегрирование с помощью сплайнов — это метод, который использует сплайны для аппроксимации интегралов. Сплайны позволяют создавать гладкие кривые и поверхности, что делает их полезными для численного интегрирования в различных областях, таких как анализ данных и машинное обучение.

И это я еще не все перечислил.)

Аппроксимация данных — это процесс нахождения функции, которая наилучшим образом описывает набор данных. В отличие от интерполяции, где функция проходит точно через все точки данных, аппроксимация позволяет функции проходить близко к точкам данных, но не обязательно через них. Это полезно, когда данные содержат шум или ошибки, и цель состоит в нахождении общей тенденции или закономерности.

Основные цели аппроксимации данных:

1. **Сглаживание данных**: Уменьшение влияния шума и выбросов.
2. **Нахождение тренда**: Определение основной закономерности или тенденции в данных.
3. **Прогнозирование**: Использование аппроксимирующей функции для прогнозирования будущих значений.
4. **Упрощение модели**: Создание более простой и интерпретируемой модели данных.

Парфенов в прошлый раз говорил о сплайнах с ограничениями, для которых есть инкрементация. Они обычно относятся к сплайнам, которые подчиняются определенным условиям или ограничениям, таким как гладкость, монотонность или ограничения на производные. Инкрементация, если я правильно понимаю, может означать постепенное изменение параметров сплайна для удовлетворения этих ограничений.

Примеры сплайнов с ограничениями

1. **Сплайны с ограничениями на производные**:
    - **Монотонные сплайны**: Сплайны, которые сохраняют монотонность (возрастание или убывание) на всем интервале.
    - **Сплайны с ограничениями на вторую производную**: Сплайны, которые ограничивают кривизну кривой.
2. **Сплайны с ограничениями на гладкость**:
    - **Сплайны с минимальной энергией**: Сплайны, которые минимизируют интеграл квадрата второй производной, что обеспечивает гладкость кривой.
3. **Сплайны с ограничениями на контрольные точки**:
    - **Сплайны с фиксированными контрольными точками**: Сплайны, которые должны проходить через определенные точки.

Регрессивные сплайны в питоне:

Вроде неплохо написано, код не запускал.

https://www.kirenz.com/blog/posts/2021-12-06-regression-splines-in-python/

P-сплайны или B-сплайны: основа и узлы

Методичка по P-сплайнам

Введение

P-сплайны — это метод сглаживания, который использует B-сплайны для представления кривой и штрафные методы для регуляризации. Основная идея заключается в том, чтобы минимизировать сумму квадратов отклонений от данных и штрафного члена, который наказывает за излишнюю кривизну кривой.

Основные шаги

1. **Определение B-сплайнов**:
    - B-сплайны — это базисные функции, которые используются для представления кривой. B-сплайны определяются степенью и узловым вектором.
2. **Определение штрафного члена**:
    - Штрафной член используется для регуляризации кривой. Он наказывает за излишнюю кривизну кривой, что помогает избежать переобучения.
3. **Минимизация целевой функции**:
    - Целевая функция включает в себя сумму квадратов отклонений от данных и штрафного члена. Минимизация этой функции позволяет найти оптимальные коэффициенты B-сплайнов.

```
Функция P_Spline(данные, степень, узловой_вектор, штрафной_параметр):
    Определить B-сплайны на основе степени и узлового вектора
    Определить матрицу B, где каждая строка соответствует значению B-сплайна в точке данных
    Определить матрицу D, которая представляет штрафной член (например, вторую производную)
    Определить целевую функцию: F = (Y - B*θ)^T * (Y - B*θ) + λ * θ^T * D^T * D * θ
    Минимизировать целевую функцию F по отношению к θ
    Вернуть оптимальные коэффициенты θ

Функция B_Spline(степень, узловой_вектор, точка):
    Определить базисные функции B-сплайна для заданной точки
    Вернуть значения базисных функций

Функция Минимизация(целевая_функция):
    Использовать метод минимизации (например, метод наименьших квадратов) для нахождения оптимал
    Вернуть оптимальные коэффициенты
```

Простой пример P-сплайна

1. **Генерация данных**:
    - Создадим простые данные с шумом.
2. **Построение B-сплайнов**:
    - Используем библиотеку **scipy** для построения B-сплайнов.
3. **Регуляризация и минимизация**:
    - Введем простую регуляризацию и минимизируем целевую функцию.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import BSpline
from scipy.optimize import minimize

# Генерация данных
np.random.seed(0)
x = np.linspace(0, 10, 100)
y = np.sin(x) + 0.5 * np.random.randn(100)

# Определение степени и узлов для B-сплайнов
degree = 3
knots = np.linspace(0, 10, 10)

# Построение матрицы базисных функций B-сплайнов
def b_spline_basis(degree, knots, x):
    basis = np.zeros((len(x), len(knots) - degree - 1))
    for i in range(len(x)):
        basis[i, :] = BSpline(knots, np.eye(len(knots) - degree - 1), degree)(x[i])
    return basis

B = b_spline_basis(degree, knots, x)
```

```
# Определение целевой функции с регуляризацией
lambda_param = 0.1
D = np.diff(np.eye(B.shape[1]), n=2)

def objective(theta):
    residuals = y - B @ theta
    penalty = lambda_param * theta.T @ D.T @ D @ theta
    return np.sum(residuals**2) + penalty

# Минимизация целевой функции
theta_init = np.zeros(B.shape[1])
result = minimize(objective, theta_init)
theta_opt = result.x
```

```
# Визуализация
x_new = np.linspace(0, 10, 500)
B_new = b_spline_basis(degree, knots, x_new)
y_new = B_new @ theta_opt

plt.plot(x, y, 'ro', label='Noisy Data')
plt.plot(x_new, y_new, 'b-', label='P-Spline')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title('P-Spline Smoothing')
plt.show()
```

Объяснение кода:

1. **Генерация данных**:
    - Создаем массив **x** от 0 до 10 с 100 точками.
    - Генерируем значения **y** как синусоиду с добавленным шумом.
2. **Построение B-сплайнов**:
    - Определяем степень B-сплайнов (**degree**) и узлы (**knots**).
    - Функция **b_spline_basis** строит матрицу базисных функций B-сплайнов для заданных узлов и степени.
3. **Регуляризация и минимизация**:
    - Определяем матрицу **D** для регуляризации.
    - Функция **objective** вычисляет целевую функцию, включающую сумму квадратов отклонений и штраф за изменение коэффициентов.
    - Используем **minimize** из библиотеки **scipy.optimize** для минимизации целевой функции.
4. **Визуализация**:
    - Строим сглаженную кривую P-сплайна и визуализируем результаты.

Код от нейронки, пытался запускать миллиард раз, но аи не фиксит ошибки…

Сложный пример работы сплайнов с данными в AIC (необязательно)

2.1 Кусочная регрессия

При анализе взаимосвязи между ответом, *y*, и объясняющей переменной, *x*, может быть очевидно, что в разных диапазонах *x* возникают разные линейные зависимости. В таких случаях одна линейная модель может не давать адекватного описания, и нелинейная модель также может оказаться неподходящей. Кусочная регрессия - это форма регрессии, которая касается полиномиальных моделей (от линейных до полиномов более высокого

порядка), встроенных в данные для различных диапазонов x. Точки останова - это значения x, при которых изменяется наклон функции.Значение точки разрыва может быть известно или неизвестно до начала анализа, но обычно оно неизвестно и его необходимо оценить. Функция регрессии в точке разрыва может быть разрывной, но модель можно построить таким образом, чтобы функция была непрерывной во всех точках, включая точки разрыва. Однако, прежде чем рассматривать сглаживающие P-сплайны, необходимо понять, что такое сплайн. Иными словами, сплайн *k*-го порядка — это кусочно-полиномиальная функция степени k, которая является непрерывной и имеет непрерывные производные порядка 1, ..., $k - 1$ в узловых точках. Рассмотрим простой случай, когда кусочно-полином является линейным (кусочно-линейная регрессия):

Когда имеется только одна точка останова, при $x = c$, модель может быть записана следующим образом:

$y = a1 + b1x$; для $x < c$

$y = a2 + b2x$; для $x > c$

Для того, чтобы функция регрессии была непрерывной в точке останова, два уравнения для *y* должны быть равны в узле. Результатом является модель кусочной регрессии, непрерывная при $x = c$:

$y = a1 + b1x$; для $x < c$

$y = a1 + c(b1 - b2) + b2x$; для $x > c$

Для соответствия модели данным можно использовать методы нелинейной регрессии методом наименьших квадратов (Ryan et al., 2007)). (См. раздел “Приложения”).

2.2 B-сплайны

Теперь мы можем рассмотреть подгонку полинома более высокого порядка. B-сплайн состоит из частей полинома, которые соединяются в специальных точках (breakpoints), называемых “узлами”. В данном узле B-сплайны отличны от нуля. Мы называем B-сплайном степени *q*, который обладает следующими свойствами:

• Состоит из $q + 1$ частей полинома, каждая степени *q*;

• Эти детали соединяются в q внутренних узлах

• Сплайн строго положительный в области, охватываемой $q + 2$ узлами, но нулевой в других местах;

• Он перекрывает на $2q$ участках полинома соседний B-сплайн.

Де Бур (1978) разработал алгоритм для вычисления B-сплайна любых степеней из B-сплайна низкой степени. В этом блоге мы используем только равноудаленные узлы, в то время как алгоритм Де Бура работает для произвольного расположения узлов.

Пусть $Bj(x; q)$ обозначает значение *j*-го B-сплайна степени *q* в точке *x* для заданной равноотстоящей сетки узлов. Приближённая кривая $y\hat{}(x)$ для данных (xi , yi) представляет собой линейную комбинацию:

$$\hat{y}(x) = \sum_{j=1}^{n} \hat{\alpha} j B_j(x; q)$$

где *α^ = (^α1, ..., a^n)* - вектор неизвестных коэффициентов регрессии, оцениваемых с использованием обычных методов наименьших квадратов;

$$\hat{\alpha} = (B'B)^{-1}B'y$$

Штрафные санкции

Рассмотрим регрессию m точек данных (*xi , yi*) на множестве из n B-сплайнов *Bj* .Целевая функция наименьших квадратов для минимизации равна

$$S = \sum_{i=1}^{n} \left\{ y_i - \sum_{j=1}^{n} \alpha_j B_j(x;q) \right\}^2$$

Когда количество узлов увеличивается, расчетная кривая (по сравнению с истинной кривой) колеблется, что означает чрезмерную подгонку данных. Чтобы сделать результат менее гибким, O'Салливан (1988) ввел штраф для второй производной и сформировал следующую целевую функцию:

$$S = \sum_{i=1}^{n} \left\{ y_i - \sum_{j=1}^{n} \alpha_j B_j(x;q) \right\}^2 + \lambda \int_{x_{min}}^{x_{max}} \left\{ \sum_{j=1}^{n} \alpha_j B_j(x;q) \right\}^2 dx$$

Поскольку во второй производной нет ничего особенного; фактически, также могут использоваться более высокие или более низкие порядки (Эйлерс и Маркс 1996) ввели базовый штраф за конечные разности коэффициентов соседних B-сплайнов:

$$S = \sum_{i=1}^{n} \left\{ y_i - \sum_{j=1}^{n} \alpha_j B_j(x;q) \right\}^2 + \lambda \sum_{j=k+1}^{n} \left\{ \delta^k \alpha_j \right\}^2$$

Этот подход уменьшает размерность задачи до n, количества B-сплайнов, вместо m, количества наблюдений. Параметр λ сохраняется для непрерывного контроля плавности подгонки.

Свойства P-сплайнов

P-сплайны не показывают граничных эффектов, как это делают многие типы сглаживателей ядра. Более того, сплайны с штрафами могут точно соответствовать полиномиальным данным. Пусть заданы (*xi , yi*). Если *yi* является многочленом от *x* степени *k*, то P-сплайны порядка *k* или выше со штрафом порядка *k + 1* или выше будут точно соответствовать данным, независимо от значения λ.

P-сплайны могут сохранять моменты данных.Это особенно полезно в контексте сглаживания плотности: среднее значение и дисперсия расчетной плотности будут равны среднему значению и дисперсии данных при любом объеме сглаживания (Эйлерс и Маркс 1996). Кроме того, предел подгонки P-сплайнов с сильным параметром сглаживания четко определен и обеспечивает связь с полиномиальными моделями.

Оптимальное сглаживание

Как нам определить оптимальное значение для λ? Один из подходов к линейным данным заключается в том, чтобы взять дисперсию остатков от $y\hat{}i$, которые вычисляются при λ = 0. Альтернативный выбор дисперсии может быть основан на обобщенной перекрестной проверке (Wahba 1990). Мы вычисляем:

$$GCV(\lambda) = \sum_{i=1}^{m} \frac{(y_i - \hat{y}_i)^2}{(m - \sum_{i=1}^{m} h_{ii})^2}$$

где *hii* - элементы матрицы hat . Наилучшим λ является значение, минимизирующее GCV(λ) де Бура (2001).

(Полный вывод матрицы hat опущен по соображениям простоты. Однако полезно знать, что трасса *H* будет приближаться к *k* по мере увеличения λ).

Набор данных

Наиболее важным видом сырой нефти, используемой в Европе, является нефть марки Brent. Нефть марки Brent - это основная торговая категория легкой сырой нефти, которая служит основным ориентиром для закупок нефти по всему миру. Он используется для определения цен на две трети мировых поставок сырой нефти, торгуемых на международном рынке. В этой статье мы анализируем ежемесячные спотовые цены на нефть марки Brent с января 2010 года по февраль 2018 года. Набор данных содержит 98 наблюдений: месячные цены за баррель в долларах США. Следующие данные являются иллюстративными наблюдениями за 2010 год.

Europe_Brent_Spot_Price_FOB<-read.table(file.choose(),header=T,sep=",")
> Europe_Brent_Spot_Price_FOB[1:12,1:2]
Date spotprice
1 10-Jan 76.17
2 10-Feb 73.75
3 10-Mar 78.83
4 10-Apr 84.82
5 10-May 75.95
6 10-Jun 74.76
7 10-Jul 75.58
8 10-Aug 77.04
9 10-Sep 77.84
10 10-Oct 82.67
11 10-Nov 85.28
12 10-Dec 91.45

dat=Europe_Brent_Spot_Price_FOB[["spotprice"]]

plot(c(1:98),dat,xlab="Months(Jan2010−Feb2018)", ylab="BrentSpotPrices ")

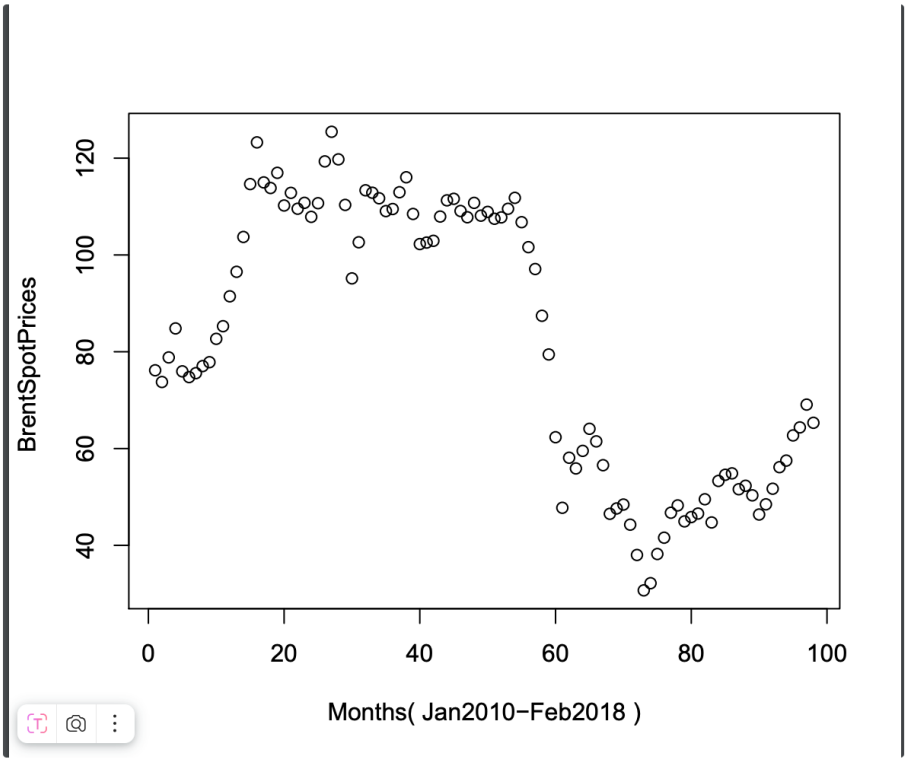


Figure 1: Time series of Brent Crude Spot Prices from January 2010 to February 2018.

В этом примере мы проиллюстрируем кусочно-линейную регрессию с единственной точкой останова. Глядя на диаграмму рассеяния данных, мы замечаем тенденцию к увеличению от нуля до двадцати, за которой следует тенденция к уменьшению, которая возвращается к отметке 70. Поэтому мы будем использовать метод AIC, чтобы определить, какая точка останова является “лучшим” выбором.

```
data=Europe_Brent_Spot_Price_FOB
> data$time=c(1:98)
> subdatax<-data$time
> subdatay<-data$spotprice
> piecewise<-lm(subdatay~subdatax)
> x=vector(length=69)
> library(segmented)
> for(i in 4:73){ # use loop to check AIC at each time
+ segment.mod<- segmented(piecewise,seg.Z=~subdatax,psi=i)
+ x[i-3]=AIC(segment.mod)
+ }
> which(x==min(x))+3
[1] 24
```

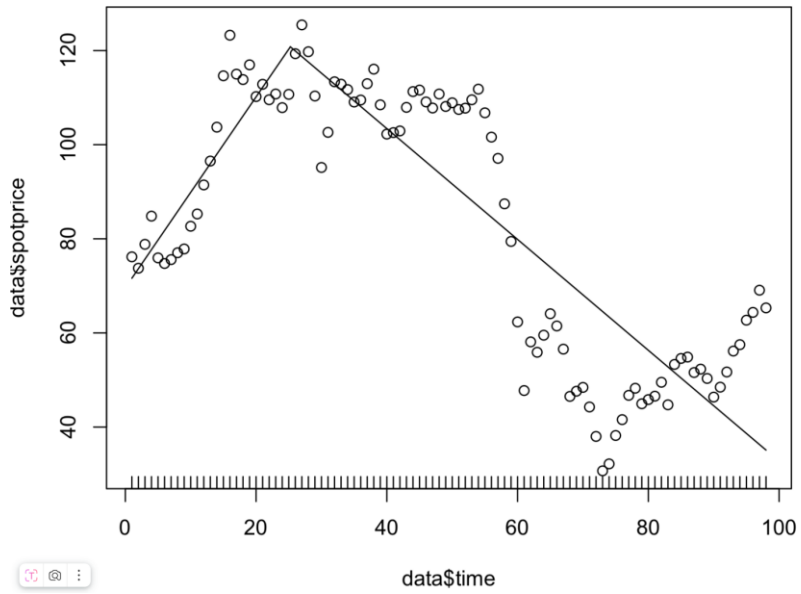
slope(segment.mod)$subdatax
Est. St.Err. t value CI(95%).l CI(95%).u
slope1 2.0282 0.404610 5.0128 1.2248 2.8316
slope2 -1.1776 0.081031 -14.5320 -1.3385 -1.0167

```
summary(segment.mod)$adj.r.squared
[1] 0.7306383
```

It appears that 24 is our “best” choice based on AIC for a breakpoint. With this choice, the fitted model explains up to 73.06 percent of variations in the data.

plot(data$time,data$spotprice)
> par(new=TRUE)
> plot(segment.mod, add=T,main=”piecewise regression”,xlab=”time”,ylab=”price”)

(это не совсем нужно для понимания)



Применение к обобщенному линейному моделированию

Мы сглаживаем с помощью B-сплайнов третьей степени и штрафа второго порядка. Выбранные узлы делят область *x* (0-98) на 20 интервалов равной ширины (таблица 1) и на 30 интервалов в таблице 2. При изменении λ на приблизительно геометрической сетке мы получаем результаты, представленные в таблицах ниже, где дисперсия вычисляется по GCV(λ) при оптимальном значении λ.

В R есть встроенная функция spline.des(), которая вычисляет (производные) B-сплайнов. Нам нужно только построить последовательность узлов. Предположим, что *XL* находится слева от *x*-домен, xr справа, и что в этом домене есть интервалы ndx. Для вычисления B для данного вектора *x* на основе B-сплайнов степени bdeg мы можем использовать следующую функцию (Эйлерс и Маркс 1996):

библиотека (pspline)
> библиотека (сплайны)
> библиотека (mgcv)

bspline<-функция(x, xl, xr, ndx, bdeg)
+ {
+ dx<-(xr-xl)/ndx
+ узлы<-seq(xl-(bdeg*dx),xr+(bdeg*dx),by=dx)
+ B<-сплайн.des(узлы, x, bdeg+1, 0*x)$дизайн
+ B
+ }
> x=c(1:98)
> y = данные в долларах США по минимальной цене

```
> xl = min ( x )
> xr = 60
> ndx = 20
> bdeg = 3 # Порядок следования B−s p l i n e s
> pord=2 # 2nd−o r de r p e n a l t y

#Когда лямбда =0.1

лямбда =0.1

B<-bspline(x, XL,xr,ndx,bdeg )

D<- диагностика (ncol(B))

D1<-разница(D)

D2<-разница(D1)

a<-solve(t(B) %*% B + лямбда * t (D2) %*% D2, t(B) %*% y )

yhat<-B %*% a

s<-сумма(( y– )-2)

Q<-решить ( t (B) %*% B + лямбда *t (D2) %*% D2) # инверсия матрицы

  t<-сумма (diag(Q %*% ( t(B) %*% B ) ) )

  gcv <− s/ ( (nrow (B) − t )^2 )

gcv

[ 1 ] 21.95668

#Когда лямбда =0,125

лямбда =0.125

gcv

[ 1 ]17.41674
```

Table 1: Values of several diagnostics for several values of *lambda* with n=20 knots.

λ	0.001	0.01	0.1	0.125	0.3	0.5	0.8	1
GVC	21.973	21.957	22.497	23.209	38.233	68.675	335.519	436.364

Table 2: Values of several diagnostics for several values of *lambda* with n=30 knots.

λ	0.001	0.01	0.1	0.125	0.3	0.5	0.8	1
GVC	18.156	18.104	17.485	17.416	26.026	49.874	212.170	411.38

При 20 узлах был найден минимум GVC (21,957) для λ = 0,01. При увеличении числа узлов до 30 был найден минимум GVC (17,416) для λ = 0.125. Действительно, в алгоритме, предложенном Эйлерсом и Марксом, штраф увеличивается по количеству узлов. Больший параметр сглаживания (λ) применяется к штрафу второго порядка.

```
номинал ( mfrow ( 1 , 2 ) )
```

подгонка<−smooth.spline(x , y , a l l . узлы = FALSE, nknots = 20 , лямбда = 0.01 , точность = 2)

график (x , y , col = ”черный ”)

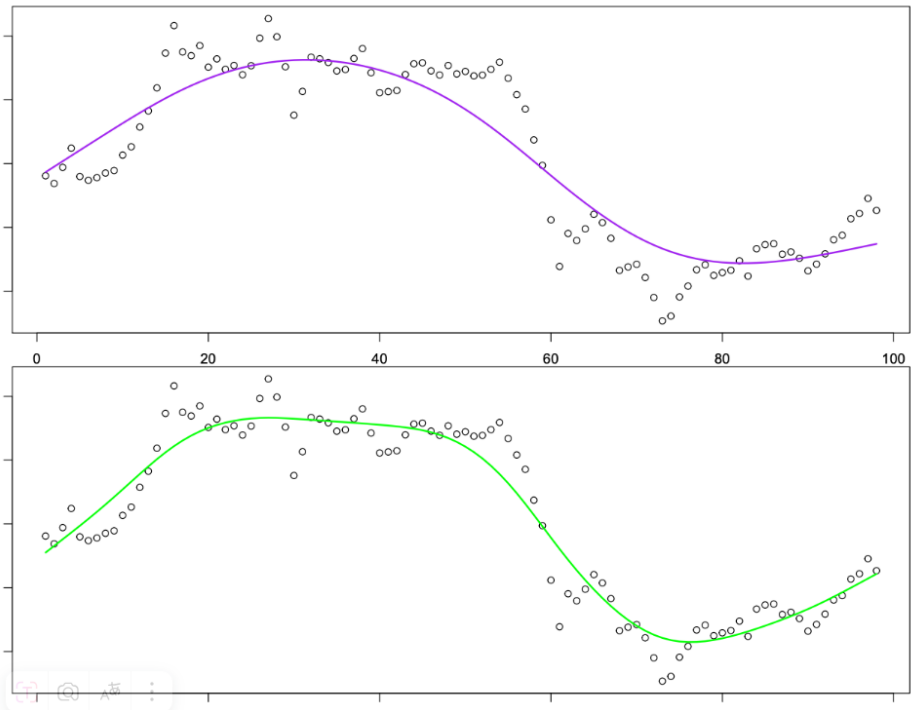
линии (fit , lwd=2, col=”фиолетовый“)

подгонка<−гладкая . сплайн(x , y , все.узлы = FALSE, nknots = 30 , лямбда = 0,125, штраф = 2))

график (x , y , col = ”черный ”)

параметры (fit , lwd= 2, col= ”зеленый”)

Рисунок 3: На верхнем графике показаны установленные P-сплайны с 20 узлами, в то время как на нижнем графике показаны установленные P-сплайны с 30 узлами.



Заключение

P-сплайны могут быть очень полезны в (обобщенных) моделях. Для каждого измерения вводятся базис B-сплайна и штрафные санкции. N узлов в каждом базовом и d измерениях приведут к последовательным (взвешенным) уравнениям регрессии (Эйлерс и Маркс (1996)). Интересно отметить, что объем работы по исследованию нескольких значений λ в значительной степени не зависит от количества точек данных при использовании GCV. В целом, алгоритм эффективен и прост в реализации.Необходимые вычисления, включая перекрестную проверку, сопоставимы по размеру с вычислениями для задачи регрессии среднего размера.