Aproksymacja
Sprawozdanie z laboratorium 5

Jakub Grześ

12.04.2024

1 Treść zadań

1.1 Zadanie 1

Wykonaj aproksymację średniokwadratową punktową populacji Stanów Zjednoczonych w przedziale $[1900, 1980]$ wielomianami stopnia m dla $0 \leq m \leq 6$.

a) Dla każdego m dokonaj ekstrapolacji wielomianu do roku 1990. Porównaj otrzymaną wartość z prawdziwą wartością dla roku 1990, wynoszącą 248 709 873. Ile wynosi błąd względny ekstrapolacji dla roku 1990? Dla jakiego m błąd względny był najmniejszy?

b) Zbyt niski stopień wielomianu oznacza, że model nie jest w stanie uwzględnić zmienności danych (duże obciążenie). Zbyt wysoki stopień wielomianu oznacza z kolei, że model uwzględnia szum lub błędy danych (duża wariancja), co w szczególności obserwowaliśmy w przypadku interpolacji. Wielomian stopnia m posiada $k = m + 1$ parametrów. Stopień wielomianu, m, jest hiperparametrem modelu. Do wyboru optymalnego stopnia wielomianu można posłużyć się kryterium informacyjnym Akaikego (ang. Akaike information criterion):

$$AIC = 2k + n \ln\left(\frac{\sum_{i=1}^{n}(y_i - \hat{y}(x_i))^2}{n}\right),$$

gdzie y_i $(i = 1, \ldots, n)$ oznacza prawdziwą liczbę osób w roku x_i,natomiast $\hat{y}(x_i)$ liczbę osób przewidywaną przez model, tzn. wartość wielomianu $\hat{y}(x)$. Ponieważ rozmiar próbki jest niewielki (dane z dziewięciu lat, $n = 9$), $n/k < 40$, należy użyć wzoru ze składnikiem korygującym:

$$AICc = AIC + \frac{2k(k+1)}{n-k-1}.$$

Mniejsze wartości kryterium oznaczają lepszy model. Czy wyznaczony w ten sposób stopień m, odpowiadający najmniejszej wartości $AICc$, pokrywa się z wartością z poprzedniego podpunktu?

1.2 Zadanie 2

Wykonaj aproksymację średniokwadratową ciągłą funkcji $f(x) = \sqrt{x}$ w przedziale $[0, 2]$ wielomianem drugiego stopnia, używając wielomianów Czebyszewa. Aproksymacja ta jest tańszym obliczeniowo zamiennikiem aproksymacji jednostajnej.

2 Rozwiązanie zadań

2.1 Zadanie 1

2.1.1 Podpunkt a

Aproksymacja średniokwadratowa jest metodą numeryczną pozwalającą na znalezienie wielomianu, który w najlepszy sposób przybliża zestaw danych. Jeśli dysponujemy zestawem punktów $(x_i, f(x_i))$, gdzie $i = 0, \ldots, m$, to aproksymacja średniokwadratowa dąży do minimalizacji sumy kwadratów różnic między wartościami obserwowanymi $f(x_i)$, a wartościami generowanymi przez wielomian $p(x)$ stopnia n. Wielomian jest wybierany w taki sposób, aby suma

$$\sum_{i=0}^{m} (f(x_i) - p(x_i))^2 \tag{1}$$

była jak najmniejsza.

Można posłużyć się równaniem normalnym, które w postaci macierzowej prezentuje się następująco:

$$A^T A c = A^T y \tag{2}$$

$$A = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \ldots & x_m^n \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}, \quad y = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix}$$

Gdzie wartości x są latami, a $f(x)$ populacją dla roku x.

Ekstrapolację do roku 1990 przeprowadzono za pomocą funkcji *exstrapolate*, która buduje macierz A dla stopnia m. Następnie za pomocą funkcji *linalg.solve* z biblioteki *Numpy* rozwiązujące równanie normalne ze względu na c. Później tworzy dodatkowy wektor, gdzie za x przyjmowany jest rok, dla którego chcemy dokonać ekstrapolacji (tutaj 1990). Następnie stworzony wektor jest mnożony przez znalezione wcześniej współczynniki prowadząc do przewidywanej wartość $f(1990)$.

```
def extrapolate(m, points, target_year):
    years = points[:,0]
    population = points[:,1]
    n = len(years)
    A = np.zeros((n, m+1))
    for i in range(n):
        for j in range(m+1):
            A[i][j] = years[i]**j
    c = np.linalg.solve(A.T @ A, A.T @ population)
    extrapolated_points =
    np.array([target_year**i for i in range(m+1)])
    return c @ extrapolated_points
```

Następnie funkcję uruchomiono dla każdego $0 \leq m \leq 6$ i wybrano tę, dla której błąd względny wobec prawdziwej wartości podanej z góry był najmniejszy.

Najlepszy wynik uzyskano dla wielomianu stopnia: 2, wyniósł on: 2.41%

Okazało się jednak, że dla wielomianów stopnia wyższego niż 2 wynik zdaje się zbyt wysoki. Dla $m = 3$ wyniósł on ponad 352000%. Stwierdzono więc wysokie prawdopodobieństwo tego, że algorytm jest niestabilny numerycznie i powtórzono eksperyment dla funkcji wykonującej to samo zadanie, ale korzystającej z funkcji bibliotecznych.

```
def extrapolate_lib(m, points, target_year):
    years = points[:, 0]
    population = points[:, 1]
    # Zwraca wektor wspolczynnikow minimalizujacych blad kwadratowy
    coefficients = np.polyfit(years, population, m)
    # Oblicza wartosc f(target_year) dla zadanych wspolczynnikow
    extrapolated_value = np.polyval(coefficients, target_year)
    return extrapolated_value
```

Najlepszy wynik uzyskano dla wielomianu stopnia: 4, wyniósł on: 2.25%

Dla wielomianu stopnia 2 i 1 uzyskano identyczne wyniki, więc stwierdzono poprawność logiczną obu algorytmów, ale używający funkcji bibliotecznych za bardziej stabilny. To jego wynik przyjęto więc jako poprawny.

2.1.2 Podpunkt b

Przeprowadzono eksperyment znajdujący wartość m dającej najlepsze rezultaty na podstawie współczynnika Akaikego. Stworzono funkcję obliczającą jego skorygowaną wartość (z powodu niewielkiego zestawu danych) dla zadanego stopnia wielomianu i zestawu punktów.

```
def aic_c(m, points):
    years = points[:,0]
    population = points[:,1]
    n = len(years)
    k = m + 1
    aic = n * np.log(np.sum((extrapolate(m, points, years)
    - population)**2) / n) + 2 * k
    aic_c = aic + 2 * k * (k + 1) / (n - k - 1)
    return aic_c
```

Obliczono jego wartość dla wszystkich m i wybrano ten, dla którego wartość współczynnika była najniższa.

Najniższą wartość współczynnika AICc uzyskano dla wielomianu stopnia: 2

Wartość uzyskana w podpunkcie b jest różna od tej z podpunktu a. Może to wynikać z tego, że w podpunkcie a skupiono się wyłącznie na błędzie względnym dla konkretnej wartości. Współczynnik Akaikego ocenia model bardziej ogólnie, każąc go za nadmierną złożoność. Pomaga on znaleźć optymalny stopień wielomianu przez ustalenie kompromisu między jego zdolnością do uchwycenia zmienności danych a wrażliwością na błędy wynikające z jego wysokiego stopnia.

2.2 Zadanie 2

Aproksymacja średniokwadratowa ciągła funkcji $f(x)$ polega na znalezieniu takiego wielomianu $p_*(x)$ stopnia n, dla którego suma kwadratów różnic pomiędzy wartościami funkcji $f(x)$ a wartościami wielomianu $p_*(x)$ jest jak najmniejsza w danym przedziale. Matematycznie możemy to zapisać jako:

$$p_*(x) = \arg\min_{p \in P_n} \int_a^b (f(x) - p(x))^2 \, dx, \tag{3}$$

gdzie P_n oznacza przestrzeń wielomianów stopnia co najwyżej n.

W przypadku użycia wielomianów ortogonalnych, takich jak wielomiany Czebyszewa, problem aproksymacji przyjmuje postać:

$$p_*(x) = \sum_{k=0}^{n} c_k \varphi_k(x), \tag{4}$$

gdzie c_k to współczynniki obliczone na podstawie iloczynów skalarnych z funkcją $f(x)$ oraz $\varphi_k(x)$ to wielomiany ortogonalne Czebyszewa stopnia k.

Dla wielomianów ortogonalnych $\varphi_0, \ldots, \varphi_n$ mamy własność ortogonalności wyrażoną jako:

$$\langle \varphi_i, \varphi_j \rangle = 0 \quad \text{dla} \quad i \neq j. \tag{5}$$

Co prowadzi do prostego układu równań dla obliczenia współczynników c_k:

$$c_k = \frac{\langle f, \varphi_k \rangle}{\langle \varphi_k, \varphi_k \rangle}. \tag{6}$$

W przypadku wielomianów Czebyszewa w przedziale $[-1, 1]$, iloczyn skalarny $\langle f, g \rangle$ jest zdefiniowany jako:

$$\langle f, g \rangle = \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\, dx. \tag{7}$$

Aby zastosować wielomiany Czebyszewa do aproksymacji funkcji $f(x) = \sqrt{x}$ w przedziale $[0, 2]$, skalujemy zmienne do przedziały $[-1, 1]$. Wtedy odpowiednie wielomiany Czebyszewa $T_k(x)$ mają postać:

$$T_k(x) = \begin{cases} 1 & \text{dla } k = 0, \\ x & \text{dla } k = 1, \\ 2x^2 - 1 & \text{dla } k = 2. \end{cases} \tag{8}$$

Te kroki realizują poniższe funkcje.

```
def T_k(k, x): # Zakladamy uzycie wielomianu 2 stopnia
    if k == 0: return 1
    if k == 1: return x
    if k == 2: return 2 * x**2 - 1
    else: return None
def weight(x):
    return 1 / np.sqrt(1 - x**2)
def f(x):
    return np.sqrt(x + 1)
def calculate_coefficients():
    c_k_values = []
    for k in range(3):
        # Iloczyn skalarny <f, Tk>, licznik
        nominator = spi.quad(lambda x: weight(x) * f(x)
        * T_k(k, x), -1, 1)[0]
        # Iloczyn skalarny <Tk, Tk>, mianownik
        denominator = spi.quad(lambda x: weight(x) * T_k(k, x)
        * T_k(k, x), -1, 1)[0]
        # Wspolczynnik c_k
        c_k = nominator / denominator
        c_k_values.append(c_k)
    return c_k_values
```

```
def calculate_value(x):
    c0, c1, c2 = calculate_coefficients()
    return c0 * T_k(0, x) + c1 * T_k(1, x) + c2 * T_k(2, x)
```

Obliczono wartości dla 100 równoodległych punktów na dziele [0, 2] przez *calcualte_value* i funkcję biblioteczną. Wyniki przedstawiono na wykresie.

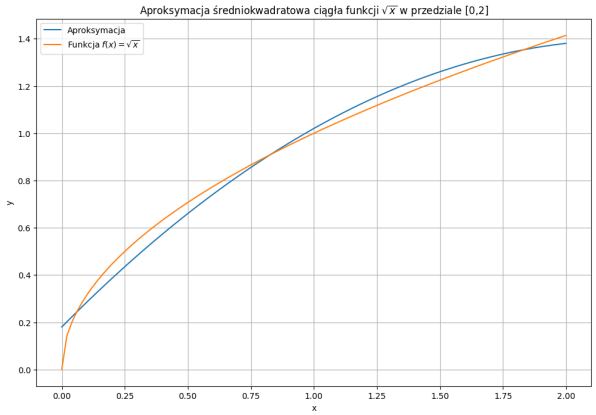

Figure 1: Aproksymacja ciągłą dla $f(x) = \sqrt{x}$

Mimo użycia wielomianu niskiego stopnia funkcja została dobrze przybliżona. Operacja była tania obliczeniowo i w niektórych przypadkach dostatecznie precyzyjna.

3 Wnioski

Aproksymacja jest użytecznym narzędziem pozwalającym m.in. na ekstrapolację na podstawie posiadanych danych. Decydując się na stopień wielomianu aproksymującego, można skorzystać z różnych kryteriów, niekoniecznie dających zgodne wyniki, które należy dobrać odpowiednio do danego zadania, w zależności od naszych priorytetów. Podczas obliczania macierzy dla aproksymacji dyskretnej, należy uważać na podnoszenie danych do wysokich potęg, mogące prowadzić do błędów. Aproksymacja może być wykonana zarówno dla dyskretnego zbioru punktów, jak i dla funkcji ciągłych, co może być użyteczne, gdy interesują nas jedynie jej przybliżone wartości np. gdy wysoka precyzja jest kosztowna obliczeniowo.

Bibliografia

[1] Marcin Kuta, *Approximation*

[2] Marian Bubak, Katarzyna Rycerz, *Wykład 5. Aproksymacja*