Analiza matematyczna dla informatyków.

Mieczysław Cichoń, ver. 3.4/2023

Mieczysław Cichoń - WMI UAM

Odległości pomiędzy funkcjami. Przybliżenia.

Czas najwyższy sprecyzować co oznacza "przybliżmy funkcję" czy "błąd przybliżenia"...

Wszyscy intuicyjnie znają pojęcie odległości między dwoma punktami (obiektami). Naturalne wydaje się, że jest to długość odcinka łączącego te punkty.

A jednak czasami, nawet nieświadomie, inaczej mierzymy odległość ... Podamy tu jak w matematyce usystematyzowano to pojęcie i wskażemy kilka przykładów.

Wprowadzimy pojęcie metryki (odległości) na dowolnym zbiorze X.

Metryki.

Definicja. Jeżeli w niepustym zbiorze X ... dla każdych dwóch elementów x, y tego zbioru przyporządkowano liczbę nieujemną $d(x, y)$ taką, że

$1^0 \quad d(x, y) = 0 \iff x = y \quad ,$

$2^0 \quad d(x, y) = d(y, x) \quad ,$

$3^0 \quad d(x, y) \leq d(x, z) + d(z, y)$ dla każdego $z \in X$.

Wówczas d nazywamy odległością albo metryką, a parę (X, d) przestrzenią metryczną. Elementy x przestrzeni metrycznej X nazywamy punktami, a warunek 3^0 - nierównością trójkąta.

Przykłady.

Warto tu podkreślić, że definicja podaje jedynie aksjomaty jakie musi spełnić funkcja d, aby być metryką, natomiast konstrukcja takiej funkcji może być różna. Stąd w szczególności na zbiorze X może być kilka różnych metryk. Podamy tu kilka ciekawych i charakterystycznych przykładów.

Rozpocznijmy od pokazania, że metrykę można wprowadzić na dowolnym zbiorze X. Określmy funkcję d następująco:

$$d(x,y) = \begin{cases} 0 & x = y \ , \\ 1 & x \neq y \ . \end{cases}$$

Metrykę tę nazywamy **dyskretną**.

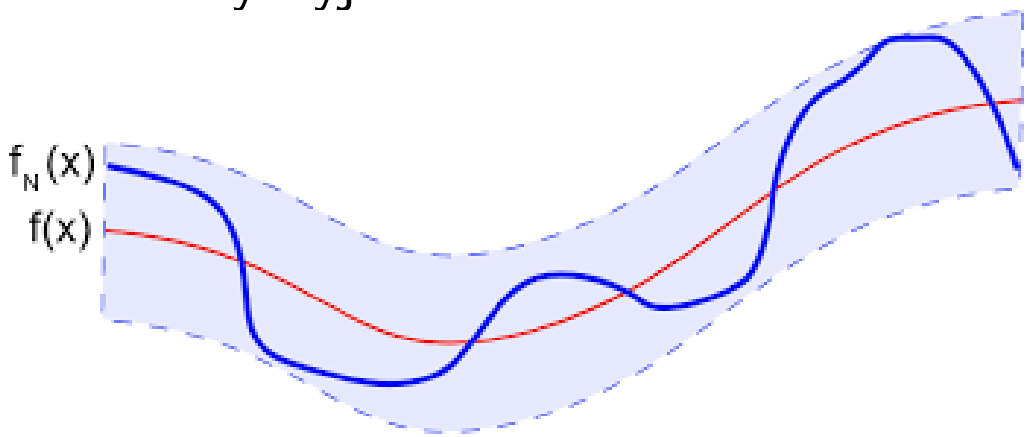
Metryka jednostajna.

A teraz metryka jednostajna dla funkcji ciągłych (ogólniej: ograniczonych): $x, y \in C(a, b)$

$$d(x, y) = \sup_{t \in [0,1]} |x(t) - y(t)| .$$

To ważny przykład w wielu dziedzinach matematyki i informatyki. Tę samą metrykę można rozpatrywać na obszerniejszych przestrzeniach, niż tylko funkcji ciągłych!

To ważna metryka. Wydaje się bardzo dobrą w zastosowaniach informatycznych, ale posiada jedną istotną wadę: wymaga sprawdzenia różnic wartości funkcji we **wszystkich** punktach przedziału, a to jest trudne i nie zawsze wykonalne, a ponadto - bardzo restrykcyjne.



Inne metryki na rodzinach funkcji.

W praktyce informatycznej poza metryką jednostajną korzysta się na ogół z dwóch innych metryk (nawet częściej niż z tej). Nie będziemy skupiać się teraz na zbiorze funkcji (tak naprawdę klas funkcji), na których definiujemy te metryki, temat można poszerzyć korzystając z literatury.

Niech x, y będą funkcjami całkowalnymi. Utożsamiając funkcje równe prawie wszędzie (poza zbiorem miary zero) mamy na takiej rodzinie metrykę d_1:

$$d_1(x, y) = \int_a^b |x(t) - y(t)|\, dt,$$

d_1

... a jeżeli te funkcje są całkowalne z kwadratem (tj. funkcje x^2 i y^2 są całkowalne) - z takim samym utożsamieniem mamy metrykę "średniokwadratową" d_2:

$$d_2(x,y) = \left(\int_a^b |x(t) - y(t)|^2 \, dt \right)^{\frac{1}{2}}.$$

Może to zaskakujące, ale "wersja dyskretna" tej metryki jest na ogół podstawową do oceny odległości pomiędzy funkcjami (lub zbiorami ich wartości) w większości algorytmów **aproksymacyjnych**, a jest już absolutnie podstawowa (z drobną modyfikacją) w **szeregach Fouriera**.

Niestety, badanie **zależności** *pomiędzy różnymi typami zbieżności ciągów i szeregów funkcyjnych jest dość skomplikowane i wykracza poza zakres naszych zajęć...*

Metryka Hamminga.

Metryka Hamminga to miara odmienności dwóch ciągów o takiej samej długości, wyrażająca liczbę miejsc (pozycji), na których te dwa ciągi się różnią. Innymi słowy jest to najmniejsza liczba zmian (operacji zastępowania elementu innym), jakie pozwalają przeprowadzić jeden ciąg na drugi.

Jest stosowana w kodowaniu i w przypadku ciągów binarnych (można stosować też dla innych łańcuchów) a i b odległość Hamminga jest równa liczbie jedynek w słowie a *XOR* b, czyli jeśli $a = (a_1, a_2, ..., a_n)$, $b = (b_1, b_2, ..., b_n)$, to

$$d_{HM}(a, b) = \sum_{i=1}^{n} [a_i(1 - b_i) + b_i(1 - a_i)].$$

Dla zainteresowanych: pewną odmianą odległości Hamminga jest odległość Canberra - proszę poszukać informacji...

Przykład (H1).

Obliczyć odległość Hamminga łańcuchów znaków:

$$a = [1110001001] \quad , \quad b = [1100011001].$$

[Odp. $d_{HM}(a, b) = 0 + 0 + 1 + 0 + 0 + 1 + 0 + 0 + 0 + 0 = 2$]

Zadanie dodatkowe: obliczyć odległość wyrazów "listek" i "lastyk"...

Wykrywanie błędów.

Jeżeli zastosujemy sposób kodowania, w którym w sposób zamierzony zwiększamy odległość Hamminga pomiędzy możliwymi wyrazami kodu, to w ten sposób konstruujemy **kod "z wykrywaniem błędu"**.

Jeśli w trakcie odczytu pojawi się wyraz o mniejszej odległości Hamminga od pozostałych wyrazów niż ustalona, będzie to świadczyć o wystąpieniu jakiegoś przekłamania. Na tym polega ten sposób z przesyłaniem liczb parzystych, ponieważ odległość Hamminga pomiędzy kolejnymi (sąsiednimi) liczbami parzystymi wynosi 2.

Z kolei Liczby Mersenne'a są w zapisie binarnym ciągami samych jedynek. Jeżeli zatem odbierzemy liczbę, która w swoim zapisie binarnym nie stanowi jednolitego ciagu jedynek (lecz np. kombinację zer i jedynek), będzie to świadczyć o wystąpieniu przekłamania. [por. Hamming]

Metryka Levenshteina.

Metryka Levenshteina jest to metryka w przestrzeni ciągów znaków, zdefiniowana następująco - działaniem prostym na napisie nazwiemy: wstawienie nowego znaku do napisu, usunięcie znaku z napisu, zamianę znaku w napisie na inny znak.

Odległością pomiędzy dwoma napisami jest najmniejsza liczba działań prostych, przeprowadzających jeden napis na drugi.

Miara ta znajduje zastosowanie w przetwarzaniu informacji, danych w postaci ciągów symboli: w maszynowym rozpoznawaniu mowy, analizie DNA, rozpoznawaniu plagiatów, korekcie pisowni (np. wyszukiwanie w spisie telefonicznym błędnie podanego nazwiska), itp. **Ma dalsze uogólnienia...**

Przykłady (L1)-(L2).

Obliczyć odległość Leveshteina łańcuchów znaków:
"kitten" oraz "sitting".

Rozwiązanie. Potrzebne 3 operacje proste:

kitten $\rightarrow$ sitten (zamiana "s" na "k")

sitten $\rightarrow$ sittin (zamiana "i" na "e")

sittin $\rightarrow$ sitting (dodajemy znak "g" na końcu łańcucha).

Czyli $d_L(kitten, sitting) = 3$.

Przykład (L2). Obliczyć (*samodzielnie*) odległość Leveshteina łańcuchów znaków:

$$a = [ABC] \qquad , \qquad b = [AC].$$

Przykład.

A tu zadanie z konkursu dla uczniów szkół średnich "Koala" 2020/21 - to właśnie **metryka Levenshteina** pomiędzy danymi układami monet z działaniem prostym opisanym w zadaniu (m.in. przesuwanie par monet o różnych nominałach itd.)...

9. Przekładaniec

Ułóżcie trzy złotówki i dwie dwuzłotówki na przemian w rzędzie, dokładnie tak jak na obrazku (a). Ruch polega na przesunięciu dwóch stykających się monet o różnych nominałach na inne miejsce w rzędzie, przy czym obowiązują takie zasady:

- Obie przesuwane monety muszą nadal się stykać i pozostać w tej samej kolejności.
- Pozostałe monety pozostają na swoich miejscach i nie można ich przesuwać, np. zsuwać po powstaniu luk.
- Cały rząd może zmienić położenie, w trakcie przesuwania mogą powstawać luki w układzie, np. takie jak na rysunku (c), gdzie pierwsza luka może pomieścić dwie monety, a druga trzy.

Wykonując jak najmniej ruchów, doprowadźcie do układu, w którym złotówki zajmują trzy pierwsze miejsca w rzędzie, a dwuzłotówki – dwa kolejne, dokładnie tak jak na obrazku (b). Ile ruchów wykonaliście?

Nazwijmy rozpiętością układu odległość między pierwszą a ostatnią monetą w układzie, mierzoną średnicą monety (np. rozpiętość układu początkowego to 4, a rozpiętość układu z rysunku (c) to 9). Rozpiętością serii ruchów nazwijmy największą z rozpiętości wszystkich układów powstających po drodze (czyli jeśli np. rozpiętości układów powstających w kolejnych krokach to: 4, 5, 6, 7, 9, 6, 7, 4, to rozpiętość całej serii wynosi 9). Spośród możliwych rozwiązań o minimalnej liczbie ruchów wybierzcie to, dla którego rozpiętość jest najmniejsza. Ile wynosi ta rozpiętość?

Jako odpowiedź podajcie dwie liczby: liczbę ruchów i rozpiętość.

(a)  (b) 

```
int LevenshteinDistance(char s[1..m], char t[1..n])

declare int d[0..m, 0..n] // d - tablica (m+1) na (n+1)
 for i from 0 to m
       d[i, 0] := i
 for j from 1 to n
       d[0, j] := j

 for i from 1 to m
     for j from 1 to n
       if s[i] = t[j] then cost := 0
                      else cost := 1
       d[i, j] := minimum(d[i-1, j] + 1,   // usuwanie
                          d[i, j-1] + 1,  // wstawianie
                          d[i-1, j-1] + cost) // zamiana

   return d[m, n]
```

“Metryki” podobieństwa w informatyce. Dywergencje.

W analizie danych przy formowaniu skupień wykorzystywane są miary rozbieżności (dywergencje) lub odległości pomiędzy obiektami.

Na przykład, gdybyśmy mieli pogrupować dania barowe, moglibyśmy wziąć pod uwagę liczbę kalorii, cenę, subiektywne oceny smaku itd.

Z punktu widzenia algorytmu łączenia jest jednak obojętne, czy odległości które mu “zadajemy” są odległościami w sensie metryki, czy też jakimiś innymi pochodnymi miarami odległości, które dla badacza mają większe znaczenie; zatem tylko od badacza zależy wybór odpowiedniej miary. Takie “miary podobieństwa” lub “miary rozbieżnosci” posiadają jedynie część własności metryki (np. nie muszą spełniać nierówności trójkata). Jak się jednak okaże: metryki będą miały jednak swoje zalety...

Przykład: metryki w nauczaniu maszynowym.

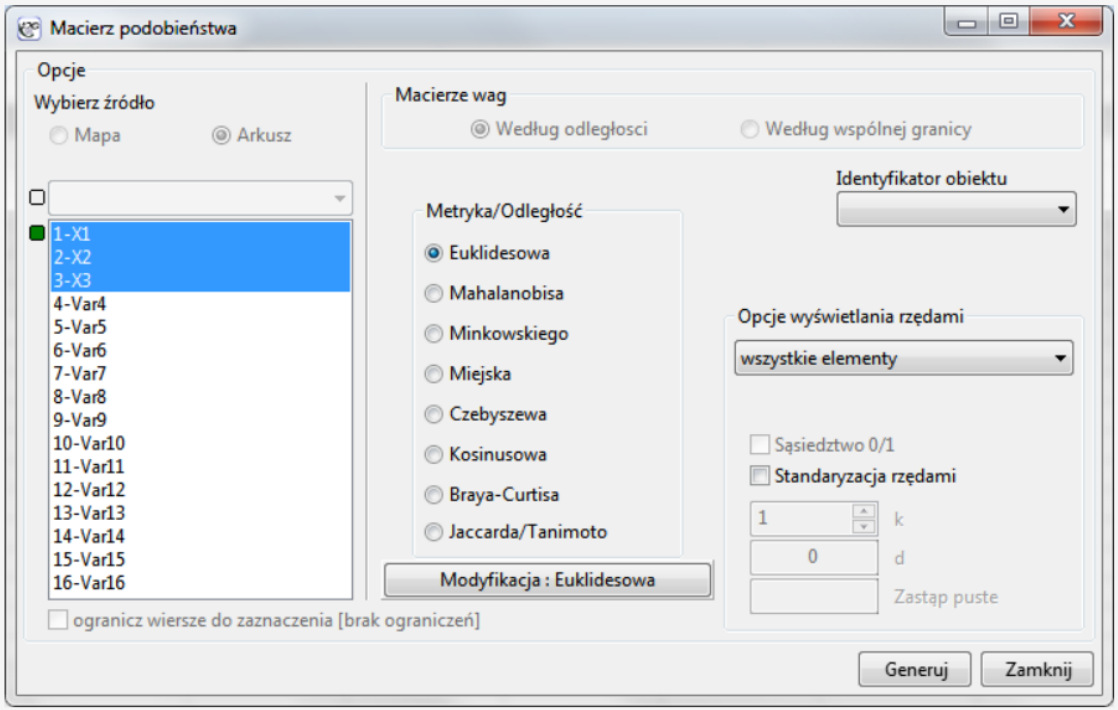
Źródło: prosty materiał - ale po angielsku. Będzie jeszcze WIELE innych!

- metryka Minkowskiego d_p,
- metryka taksówkowa (Manhattan distance) d_1,
- metryka supremalna (Chebyshev distance) d_∞,
- metryka euklidesowa d_2,
- metryka Hamminga: *Metryka porównująca dwa łańcuchy danych. Jest to liczba bitów łańcuchów o takiej samej długości różnych na tej samej pozycji. Obliczając tę odległość mędzy łańcuchami wykonamy operację XOR* $(a \oplus b)$ *i zliczając liczbę jedynek w otrzymanym wyniku.*
- odległość cosinusowa (oparta o kąt pomiędzy danymi punktami i środkiem układu współrzędnych).

patrz też tutaj ponadto ten materiał i wreszcie kolejna część tutaj...

Dostosowywanie metryk do zagadnienia.

Wiele specjalizowanych programów ma z góry wprowadzone różne metryki ale i daje możliwości ich dodatkowego dostosowywania - jeśli nie, to warto to zrobić samodzielnie!



Niepodobieństwo/podobieństwo obiektów wyrażamy za pomocą odległości będących najczęściej metryką. Nie każda miara odległości jest jednak metryką. Aby odległość mogła być nazwana metryką musi spełniać 4 warunki:

- odległość pomiędzy obiektami nie może być ujemna
- odległość między dwoma obiektami wynosi 0 wtedy i tylko wtedy gdy są one identyczne
- odległość musi być symetryczna, tzn. odległość z obiektu x_1 do x_2 musi być taka sama jak z x_2 do x_1
- odległość musi spełniać nierówność trójkąta

/ tu: "PQStat"

Python.

Oczywiście - klasyczne metryki są zaimplementowane w wielu programach i nie trzeba tworzyć wszystkiego samodzielnie (niekiedy można jednak na ich podstawie tworzyć nowe). Ale trzeba wiedzieć KTÓRA z metryk jest przydatna w badanym zagadnieniu i tu rola programisty. W Pythonie potrzebujemy biblioteki *scipy*:

```
from scipy.spatial.distance import cityblock, euclidean, minkowski
import numpy as np

# Definiowanie dwóch punktów w przestrzeni R^3
p1 = np.array([1, 2, 3])
p2 = np.array([4, 5, 6])

# Obliczanie odległości między punktami w różnych metrykach
d_cityblock = cityblock(p1, p2)    # metryka miejska (taksówkowa)
d_euclidean = euclidean(p1, p2)    # metryka euklidesowa
d_minkowski = minkowski(p1, p2, 3) # metryka Minkowskiego

# Wyświetlanie wyników
print("Odległość w metryce miejskiej:", d_cityblock)
print("Odległość w metryce euklidesowej:", d_euclidean)
print("Odległość w metryce Minkowskiego:", d_minkowski)
```

Ciągi funkcyjne.

Być może główną operacją opartą o metryki w informatyce będzie jednak przybliżanie danej danej funkcji f ciągami funkcji (f_k) o pewnych "lepszych" własnościach np. łatwiej lub dokładniej wyliczalnych *(np. funkcję ciągłą wielomianem, funkcją schodkową lub łamaną).*

Musimy wyjaśnić

- jak to zrobić (w jakim sensie jest zbieżność),
- jak kontrolować błąd popełniany przy tej operacji (odległość granicy f od przybliżenia f_k).

Ciągi w przestrzeniach metrycznych.

Na początek ważna definicja. Proszę zawsze wyobrażać sobie w tym miejscu ciągi *funkcji* przybliżających zadaną już funkcję...

Definicja. Mówimy, że ciąg (x_n) punktów $x_n \in X$ w przestrzeni metrycznej (X, d) jest zbieżny do punktu $x \in X$ gdy

$$d(x_n, x) \longrightarrow 0 \qquad \text{dla} \quad n \to \infty \ ,$$

gdzie zbieżność jest w sensie ciągu liczbowego tj.

$$\underset{\varepsilon>0}{\forall} \quad \underset{N\in\mathbb{N}}{\exists} \quad \underset{n\geq N}{\forall} \ d(x_n, x) < \varepsilon \ .$$

Warto zauważyć, że ciąg (x_n) jest zbieżny do $x \in X \iff$ dla każdej kuli $K(x, r)$ (dla $r > 0$) prawie wszystkie wyrazy ciągu (x_n) należą do kuli $K(x, r)$.

Ciąg (szereg $\Rightarrow$ gdy położyć S_n w miejsce f_n) liczbowy (f_n) może na rozważanym podzbiorze $E \subset D$ (w szczególności na całej dziedzinie D, lub na swoim obszarze zbieżności) być

- zbieżny punktowo, to znaczy zbieżny jest ciąg $(f_n(x))$ w każdym punkcie $x \in E$,
- zbieżny punktowo bezwzględnie, to znaczy być zbieżnym bezwzględnie w każdym punkcie $x \in E$: $(|f_n(x)|)$ jest zbieżny,
- zbieżny jednostajnie, to znaczy $a_n = \sup_{x\in E} f_n$ jeat zbieżny, czyli spełniać warunek

$$\forall_{\epsilon>0}\exists_{n_0}\forall_{n\geq n_0}\forall_{x\in E}\ |f(x) - f_n(x)| < \epsilon,$$

- zbieżny jednostajnie bezwzględnie, kiedy ciąg $b_n = \sup_{x\in E} |f_n|$ jest zbieżny.

Proste motywacje.

A gdzie to napotkamy w informatyce? Jest wiele takich sytuacji, choć nie zawsze padnie wówczas słowo *ciąg*. Klasyczne przypadki omówimy na wykładzie i będa to:

1) ciągi funkcji aproksymujacych daną funkcję f (mając f szukamy g_n dla której odległość of f (wyjaśnimy jak to rozumieć) jest np. mniejsza niż $\frac{1}{n}$ - i tak dla każdego n - powstaje ciąg,

2) wielomiany będące sumami częściowymi szeregów Taylora i przybliżające wartości funkcji rozwijanej w taki szereg (tu mamy i ciąg i szereg funkcyjny), lub wielomiany aproksymacyjne i ich zastosowania w metodzie najmniejszych kwadratów, która jest stosowana w analizie numerycznej i statystyce. Metoda ta polega na znalezieniu wielomianu $P_n(x)$, który najlepiej "pasuje" do zadanej funkcji f na danym przedziale $[a, b]$ poprzez minimalizację średniego kwadratu różnicy pomiędzy $f(x)$ a $P_n(x)$ na tym przedziale.

3) pojawi sie podobna idea przybliżania funkcji ciągiem wielomianów trygonometrycznych i szeregiem (Fouriera), np. ciąg funkcji trygonometrycznych: $f_n(x) = sin(nx)$. Ciąg ten składa się z funkcji o różnych okresach (częstotliwościach fal). W informatyce taki ciąg funkcji może być wykorzystany w dziedzinie przetwarzania sygnałów do analizy i przetwarzania fal dźwiękowych lub obrazów.

4) ciąg funkcji gamma: $f_n(x) = \Gamma(n+x)$. Ciąg ten składa się z funkcji gamma dla różnych wartości argumentów. Funkcja gamma jest zdefiniowana dla wszystkich liczb zespolonych, z wyjątkiem ujemnych liczb całkowitych i zer. W praktyce jednak często korzysta się z wartości funkcji gamma dla dodatnich liczb całkowitych i połówek. Ciąg funkcji gamma ma szerokie zastosowanie w matematyce i fizyce, gdzie jest wykorzystywany do rozwiązywania różnych problemów, takich jak równania różniczkowe, całki i wiele innych. W informatyce ciąg funkcji gamma jest stosowany w wielu algorytmach numerycznych, takich jak metoda Gaussa-Laguerre'a do całkowania numerycznego, czy algorytm Brenta do znajdowania pierwiastków funkcji.

Twierdzenie Weierstrassa o aproksymacji.

Twierdzenie. Dla każdej funkcji ciągłej określonej na przedziale domkniętym i przyjmującej wartości rzeczywiste istnieje **ciąg wielomianów** zbieżny jednostajnie do tej funkcji na tym przedziale.

Co ważne (dla informatyków), ten ciąg może składać się z tzw. wielomianów Bersteina (por. krzywe Béziera!):

$$f_n(x) = \sum_{k=0}^{n} \binom{n}{k} f\left(\frac{k}{n}\right) x^k (1-x)^{n-k}$$

Wielomiany bazowe Bernsteina służą do przedstawiania szeroko stosowanych w grafice komputerowej: krzywych Béziera, płatów Béziera i wywodzących się z nich innych rodzajów krzywych i powierzchni.

Pamiętajmy też o zajęciach z interpolacji (wkrótce)!!

Zastosowanie: błąd oszacowania.

I tu warto zwrócić uwagę, że *błąd oszacowania* funkcji poprzez wielomiany (np. Bernsteina) to właśnie realizacja pomysłu zastosowania zbieżności ciągów funkcyjnych (lub jak kto woli: odległości pomiędzy funkcjami). Mamy np. w Pythonie:

```
import numpy as np
from scipy.special import binom

def bernstein_error(f, n, x):
    """
    Oblicza błąd przybliżenia funkcji f(x) wielomianem Bernsteina
    stopnia n w punkcie x.
    """
    # Obliczenie normy n-tej pochodnej funkcji f(x)
    norm = np.max(np.abs(np.gradient(f(np.linspace(0, 1, 1001)), 1 / 1000,
     edge_order=2, axis=0)))
    # Obliczenie normy bazy wielomianów Bernsteina stopnia n
    bernstein_basis = lambda k, n, t: binom(n, k) * t**k * (1 - t)**(n - k)
    norm_basis = np.max(np.abs(bernstein_basis(np.arange(0, n + 1), n, x)))
    # Obliczenie błędu
    error = norm / (2**n * np.math.factorial(n)) * norm_basis
    return error
```

Uogólnijmy teraz znaną dla ciągów liczbowych definicję:

Definicja. Ciąg (x_n) punktów przestrzeni metrycznej (X, d) spełnia warunek Cauchy'ego (jest ciągiem Cauchy'ego), gdy dla dowolnego $\varepsilon > 0$ istnieje taka liczba $N \in \mathbb{N}$ taka, że dla dowolnych $m, n > N$ mamy $d(x_n, x_m) < \varepsilon$.

A teraz kilka związanych z tym własności:

Twierdzenie.

(a) Ciąg zbieżny jest ciągiem Cauchy'ego.

(b) Ciąg Cauchy'ego jest ograniczony.

(c) Ciąg Cauchy'ego zawierający podciąg zbieżny do punktu x jest też zbieżny do x.

Nie zachodzi natomiast twierdzenie analogiczne do twierdzenia Bolzano-Weierstrassa dla ciągów liczbowych !

Przestrzeń zupełna.

Mamy więc nową definicję:

Definicja. Przestrzeń metryczna (X, d) nazywa się zupełną, jeżeli każdy ciąg Cauchy'ego w tej przestrzeni jest ciągiem zbieżnym (do pewnego $x \in X$).

Tak więc tw. Bolzano-Weierstrassa stwierdza, że $(\mathbb{R}, d_2)$ jest przestrzenią zupełną. My możemy podać ogólniejsze:

Twierdzenie. *Przestrzeń* $(\mathbb{R}^k, d_2)$ *jest zupełna.*

Ciąg kolejnych przybliżeń.

Definicja. Odwzorowanie $T : X \to X$ przestrzeni metrycznej $< X, d >$ w siebie nazywamy kontrakcją lub odwzorowaniem zwężającym, gdy istnieje taka liczba L, spełniająca nierówności $0 < L < 1$, że dla każdych $x, y \in X$ zachodzi nierówność

$$d(T(x), T(y)) \leq Ld(x, y) \ .$$

Przestrzeń metryczną nazywamy zupełną, gdy każdy ciąg spełniający warunek Cauchy'ego jest zbieżny.

Twierdzenie (Banacha o punkcie stałym).

Niech X będzie przestrzenią metryczną zupełną. Jeżeli odwzorowanie $T : X \to X$ jest kontrakcją ze stałą $L < 1$, to istnieje dokładnie jeden punkt stały x_0 tego odwzorowania. Ponadto punkt ten można otrzymać jako granicę ciągu (x_n) określonego w następujący sposób: x_1 jest dowolnym punktem z przestrzeni X, a $x_{n+1} = T(x_n)$ dla $n = 1, 2, \ldots$.

Prawdziwa jest również następująca nierówność

$$d(x_n, x_0) \leq \frac{L^{n-1}}{1-L} \cdot d(x_2, x_1).$$

Ciąg (x_n) nazywamy ciągiem kolejnych przybliżeń lub ciągiem kolejnych iteracji odwzorowania T. Nierówność ta przedstawia błąd popełniony kiedy rozwiązanie równania $T(x) = x$ zastąpimy rozwiązaniem przybliżonym x_n.

Zadanie.

O tej i innych **metodach iteracyjnych** - na metodach numerycznych.

Metoda iteracji prostych bazuje na twierdzeniu Banacha o punkcie stałym i ma kilka zalet...

Ćwiczenie: obliczyć tą metodą przybliżone rozwiązanie równania w przedziale $[0, 2]$:

$$x - \frac{x^2 + 2}{3x^2 + 1} = 0.$$

Rozwiązanie. Sprawdzamy zachodzenie warunku kontrakcji dla operatora T dla przestrzeni metrycznej $(\mathbb{R}, |\cdot|)$ (czyli z wartością bezwzględną):

$$\begin{aligned}
|\frac{x^2+2}{3x^2+1} - \frac{y^2+2}{3y^2+1}| &= |\frac{(3y^2+1)(x^2+2) - (3x^2+1)(y^2+2)}{(3x^2+1)(3y^2+1)}| = \\
&= |\frac{5y^2-5x^2}{(3x^2+1)(3y^2+1)}| = |\frac{5(y-x)(y+x)}{(3x^2+1)(3y^2+1)}| \leq \\
&\leq |y-x| \cdot |\frac{5(y+x)}{(3x^2+1)(3y^2+1)}| \leq \frac{5\cdot 4}{2} \cdot |y-x|
\end{aligned}$$

(pamiętajmy, że $x, y \in [0,2]$). Czyli T jest kontrakcją ze stałą $L = \frac{1}{2}$.

Teraz wybieramy punkt początkowy dla iteracji w zadanym przedziale, np. $x_0 = 1 \in [0, 2]$. Mamy

$$x_1 = T(x_0) = T(1) = \frac{3}{80} = 0,0375$$

i liczymy dalej

$$x_2 = T(x_1) = T\left(\frac{3}{80}\right) = \frac{1}{20} \cdot \frac{(\frac{3}{80})^2 + 2}{3(\frac{3}{80})^2 + 1} \approx 0,070379.$$

Kolejny krok - samodzielnie...

I ostatnia część zadania: skoro mamy dwa pierwsze elementy, to proszę oszacować błąd ze wzoru $d(x_n, x_0) \leq \frac{1}{2^{n-2}} \cdot d(x_2, x_1)$.

[Odp. - oczekiwane jest tylko przybliżenie!, wynik dokładny jest ... skomplikowany i nie jest tak łatwo go uzyskać, a poza tym i tak byłby wartością przybliżoną $\approx 0,097681$]

Uwaga.: dokładnie na tym twierdzeniu opiera się metoda iteracji prostych, która bedzie powszechnie stosowana - ze względu na jej prostotę i kontrolę błędów, a poznamy (na analizie matematycznej) jeszcze sposoby pozwalajace łatwiej sprawdzić, czy odwzorowanie jest kontrakcją. Zwróćmy uwagę, że kontrolujemy błąd w n-tym kroku.

Więcej o metrykach i ich zastosowaniach - na **innym** przedmiocie (będzie wśród takich do wyboru)...