Numpy - zadania

Wektory 1D

1. Wylosuj wektor x 30 liczb z rozkładu jednostajnego z przedziału $[-4, 4]$. Następnie wyznacz:

   A) wszystkie wartości ze zbioru $[-2, -1]$ i $[1, 2]$
   B) liczbę wszystkich wartości nieujemnych
   C) średnią arytmetyczną wartości bezwzględnych (modułów) elementów
   D) najmniejszą i największą wartość
   E) wartość najbliższą i najdalszą od 0 (zachowując jej znak)
   F) wartość najbliższą i najdalszą od 2 (zachowując jej znak)
   G) część ułamkową wszystkich elementów
      - -1.23 -> -0.23
      - 2.23 -> 0.23
      - 3.99 -> 0.99
      - -4.99 -> -0.99

   H) interpolację liniową tego wektora na przedział [-1, 1]: minimum $x_{\min}$ przechodzi na $y_{\min} = -1$, maksimum $x_{\max}$ na $y_{\max} = 1$, a pozostałe elementy są liniowo przeskalowane według poniższego wzoru:
   $$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \cdot (y_{\max} - y_{\min}) + y_{\min}$$
   I) średnią wartość kwadratów liczb większych od 3 lub mniejszych od -2
   J) utwórz wektor napisów y o długości takiej samej, jaką ma x, dla którego y_i przyjmuje wartość `'nieujemna'`, jeśli x_i jest nieujemne oraz `'ujemna'` w przeciwnym wypadku
   K) jego średnią (bez używania funkcji `mean()`)
   L) minimum i maksimum, ale nie używając funkcji `min()` i `max()` (rozwiązanie nie musi być optymalne)
   M) utwórz wektor liczbowy y o długości takiej samej, jaką ma x, dla którego y_i przyjmuje wartość $k + 0.5$ wtedy i tylko wtedy, gdy x_i należy do $[k, k+1)$, gdzie k to liczba całkowita (prosty histogram)

2. Dla dwóch wektorów równej długości x i y oblicz ich korelację ze wzoru:

   A)
   $$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$
   B)
   $$r = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^{n}(x_i^2 - n\bar{x}^2)}\sqrt{\sum_{i=1}^{n}(y_i^2 - n\bar{y}^2)}}$$
   C)
   $$r = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}\sqrt{n\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}}$$

   Sprawdź wyniki dla następujących par wektorów:

   - `x=[1,2,3,4,5,6,7,8,9,10]` i `y=[10,9,8,7,6,5,4,3,2,1]`
   - x jest losową próbką ze standardowego rozkładu normalnego, a $y = 5x + 2$
   - zarówno x jak i y jest losową próbką ze standardowego rozkładu normalnego

3. Napisz funkcję, która standaryzuje wartości w podanym wektorze numerycznym, tj. przeskalowuje elementy w taki sposób, że ich średnia jest równa 0, a odchylenie standardowe 1.

4. Oblicz iloczyn skalarny dwóch wektorów (suma iloczynów ich współrzędnych):

$$\mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i$$

Ile różnych sposobów na mnożenie macierzowe znajdziesz w `Numpy`?

5. Oblicz trzy miary błędu pomiędzy wektorami:

   A) Oblicz *root-mean-squared-error* pomiędzy dwoma wektorami:

   $$\mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2}$$

   B) Oblicz *mean-absolute-error* pomiędzy dwoma wektorami:

   $$\mathrm{MAE}(\mathbf{x}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - x_i|$$

   C) Oblicz *median-absolute-error* pomiędzy dwoma wektorami:

   $$\mathrm{MedAE}(\mathbf{x}, \mathbf{y}) = median(|y_1 - x_1|, \ldots, |y_n - x_n|)$$

   Czy umiesz pokazać przykład, gdzie będzie widać, za co karzą poszczególne miary błędu? Innymi słowy dane będą te same, ale miary błędów różne i będziemy umieli uzasadnić czemu jedna miara jest większa od drugiej?

6. Oblicz znormalizowany dystans Hamminga pomiędzy dwoma wektorami liczb całkowitych, x i y o równej długości n. Najpierw policz na ilu współrzędnych wektory x i y się różnią. Następnie podziel tę liczbę przez długość n. Przykładowo, jeśli `x=[1,2,1,3]` i `y=[1,3,1,2]` to miara powinna wynieść 0.5.

7. Mamy dane dwa wektory: $\mathbf{y}$ (tylko liczby całkowite 0 lub 1, np. etykiety klas) oraz $\hat{\mathbf{y}}$ (liczby rzeczywiste pomiędzy 0 a 1, np. prawdopodobieństwa przynależności do klasy 1). Oblicz miarę błędu o nazwie *cross-entropy loss*:

$$-\frac{1}{n}(\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i))$$

Gdzie log to logarytm o podstawie 10.

8. Dla danego wektora liczbowego zawierającego braki danych (`NaN`) uzupełnij je średnią wartością z pozostałych, poprawnych wartości.

9. Dla danego wektora liczbowego x o długości n i wartości $k \leq \frac{n-1}{2}$ wyznacz średnią k-winsorowską, to znaczy średnią arytmatyczną z podwektora x, w którym k najmniejszych i k największych elementów zostaje zastąpionych przez odpowiednio $(k + 1)$-szą wartość najmniejszą i największą.

10. Zaimplementuj funkcję `lead(x, n)` oraz `lag(x, n)`. Pierwsza pozbywa się pierwszych n wartości, a następnie dodaje na koniec n wartości brakujących. Przykładowo `lead([1, 2, 3, 4, 5], 2) == [3, 4, 5, NaN, NaN]`. Druga działa analogicznie: `lag([1, 2, 3, 4, 5], 2) == [NaN, NaN, 1, 2, 3]`.

11. Zaimplementuj `cumall()` i `cumany()`, czyli skumulowane wersje `all()` i `any()`. Przykładowo:

    - `cumall([True, True, True, False, True, False]) == [True, True, True, False, False, False]`
    - `cumany([False, False, True, False, True]) == [False, False, True, True, True]`

12. Napisz funkcję `factorial_string()`, która zwraca przybliżoną wartość silni według wzoru Stirlinga:

$$n! \simeq (\frac{n}{e})^n \sqrt{2\pi n}$$

Znajdź funkcję, która oblicza silnię w sposób dokładny. Zaproponuj testy wzoru przybliżonego: czy wraz ze wzrostem n wzór przybliża coraz lepiej czy coraz gorzej? Czy obliczenia przebiegają wolniej czy szybciej?

13. Korzystając ze wzoru Leibniza

$$\sum_{i=0}^{n} \frac{(-1)^i}{2i+1} \simeq \frac{\pi}{4}$$

oblicz przybliżoną wartość liczby π dla $1.000, 10.000, 100.000$ początkowych wyrazów i porównaj uzyskane wyniki ze stałą `pi` w `Numpy`. Zwróć uwagę na to, że może być wiele różnych implementacji podanego wzoru. Wymyśl najwięcej jak umiesz i porównaj ich prędkość działania.

14. Podana wyżej metoda nie jest jedyną na przybliżanie liczby π. Skorzystamy teraz z metody Monte Carlo, której algorytm wygląda następująco:

   A) wylosuj n punktów w dwuwymiarowej przestrzeni $[-1, 1] \times [-1, 1]$
   B) sprawdź ile punktów jest oddalonych od punktu $(0, 0)$ o mniej niż 1
   C) podziel tę liczbę przez n i przemnóż przez 4

   Do losowania punktów użyj funkcji `np.random.uniform()`.

15. Wypisz w postaci wektora liczb całkowitych dziesięć pierwszych liczb rozwinięcia dziesiętnego liczby π (korzystając ze stałej w `Numpy`). Wynik powinien być następujący: `[3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]`.

16. Napisz funkcję `moda()`, która zwraca modę, tzn. najczęściej pojawiającą się wartość w wektorze. Jeśli moda nie jest unikalna, zwróć dowolną.

17. Napisz samodzielnie funkcję `regresja()`, która policzy współczynniki α i β w modelu prostej regresji liniowej, tj. $y = \alpha + \beta x$. Funkcja na wejściu przyjmuje dwa wektory liczbowe, x i y.

$$\beta = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\alpha = \bar{y} - \beta\bar{x}$$

Oto kod na wygenerowanie i narysowanie chmury punktów i dowolnej prostej:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

beta = 2 # założone idealne współczynniki
alfa = 3

# chmura punktow, do ktorej się dopasowujemy
n = 150
x = np.random.uniform(-5,5, n)
y = beta*x+alfa + np.random.normal(0, 2, n)


# ... estymujecie ze wzoru

beta_estimated = 5 # zamiast liczb trzeba zaimplementowac wzory
```

```
alfa_estimated = 3
print(f"beta_estimated = {beta_estimated}, alfa_estimated = {alfa_estimated}")

x_plot = np.arange(-5, 5, 0.1)
y_plot = beta_estimated*x_plot+alfa_estimated

plt.plot(x, y, 'o')
plt.plot(x_plot, y_plot, 'r')
plt.show()
```

18. Zaimplementuj funkcję `crossover(x,y)`, która realizuje krzyżowanie: przyjmuje dwa wektory równej długości n - x, y i zwraca wektor również długości n, w którym losowa połowa wartości jest przepisywana z wektora x a druga połowa z y (pozycje, na których przepisujemy z x, a na których z y, mają zostać wylosowane, jedyny ważny warunek: jednych i drugich ma być tyle samo).
19. Zaimplementuj funkcję `mutate(x, p)`, która realizuje mutację: modyfikuje wektor x w taki sposób, że każdą wartość zmienia z prawdopodobieństwem p poprzez dodanie do niej wartości losowej z rozkładu normalnego o średniej 0 i odchyleniu standardowym o wartości jednej setnej wartości bezwględnej danej wartości.
20. Napisz funkcję `losuj()` która przyjmuje:
    A) liczbę całkowitą n
    B) wektor numeryczny x o długości k o unikatowych wartościach
    C) wektor prawdopodobieństw p o długości k

    Zadaniem jest wylosować n losowych wartości z x zgodnie z prawdopodobieństwami p (ze zwracaniem), ale bazując tylko na generatorze liczb z rozkładu jednostajnego. Przykładowo, jeśli dostaniemy `x=[11,22,33]`, `p=[0.2, 0.3, 0.5]` i `n=1000` to średnio powinniśmy dostać 200 razy liczbę 11, 300 razy liczbę 22 i 500 razy liczbę 33.
    Algorytm: wygeneruj losową wartość u z $U[0,1]$. Znajdź takie m, że wartość u znajduje się pomiędzy sumą $m-1$ pierwszych prawdopodobieństw z wektora p a sumą m pierwszych prawdopodobieństw z wektora p. Zwróć m-tą wartość z x.
21. Napisz funkcję `interpoluj_liniowo()` która przyjmuje:
    A) rosnąco posortowany wektor x o długości n
    B) dowolny wektor y o długości n
    C) wektor z o długości k i elementach z przedziału tego samego, co wartości w x. Niech f będzie kawałkami liniową interpolacją punktów $(x_1, y_1), \ldots, (x_n, y_n)$. Zwróć wektor w o długości k taki że $w_i = f(z_i)$.

Oto kod na narysowanie sytuacji z zadania wraz z przykładowymi wartościami:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

x = np.array([-5,-3,2,7])
y = np.array([4,5,-8,8])
z = np.array([-4,-2,0,1,5])

plt.plot(x, y, 'r')
plt.plot(z,np.repeat(0,z.size),'o')
plt.show()
```

Oto kod na interpolację dla jednego punktu, pod warunkiem, że wiemy, do którego przedziału ten wpada:

```
z_1 = 0

x_0 = -3 # ponieważ wiemy do którego przedziału wpada wartość z_1
```

```
x_1 = 2  # przepisujemy sobie na bok potrzebne wartości
y_0 = 5  # czyli lewy i prawy kraniec przedziału w x i y
y_1 = -8

y_prim = y_0 + (y_1-y_0)/(x_1-x_0) * (z_1 - x_0) # znany wzór, gdy mamy gotowe liczby
y_prim
```

Macierze 2D:

1. Dla dowolnej macierzy odwróć:
   - kolejność jej wierszy
   - kolejność jej kolumn
2. Mając daną macierz $n \times k$ z elementami rzeczywistymi zastosuj funkcję softmax do każdego wiersza, to znaczy
$$x_{i,j} \to \frac{\exp(x_{i,j})}{\sum_{l=1}^{k} \exp(x_{i,l})}$$
   Następnie dokonaj kodowania one-hot dla każdego wiersza, czyli znajdź kolumnę z wartością najbliżej 1 (największą wartością). Zwróć wektor o długości n.
3. Mamy dany wektor T o długości n liczb ze zbioru $\{0, \ldots, k-1\}$. Napisz funkcję, która dokona kodowania one-hot-encode dla każdej wartości z T. Innymi słowy zwróć macierz zer i jedynek R o n wierszach i k kolumnach, taką że jedynka znajduje się w i-tym wierszu na j-tej kolumnie wtedy i tylko wtedy gdy i-ty element wektora T równa się j.
4. Prawą macierzą stochastyczną nazywamy macierz kwadratową, której elementami są nieujemne liczby rzeczywiste i w której każdy wiersz sumuje się do jedynki. Przykładem prawej macierzy stochastycznej jest macierz:
$$R = \begin{pmatrix} 0.5 & 0.3 & 0.2 \\ 0.2 & 0.8 & 0 \\ 0.3 & 0.3 & 0.4 \end{pmatrix}$$
   Napisz funkcję `do_stochastycznej()`, która przyjmuje kwadratową macierz (niekoniecznie prawą stochastyczną) i wykonuje następujące czynności:
   A) Sprawdza, czy wszystkie elementy są nieujemne
   B) Sprawdza, czy w każdym wierszu jest co najmniej jeden element większy od zera
   C) Tworzy nową macierz na bazie wejściowej macierzy, w której wiersze są "unormowane", czyli sumują się do jedynki. Dla przykładu, dla następującego wiersza: $5, 3, 2$ w macierzy zwracanej ten wiersz powinien wyglądać tak: $0.5, 0.3, 0.2$. A robi się to tak: trzeba zsumować wszystkie elementy z wiersza a później każdy element podzielić przez tę sumę.
5. Niech macierz X przechowuje n punktów (n wierszy) o d współrzędnych (d kolumn). Znajdź boks okalający te punkty. Innymi słowy, zwróć macierz B o 2 wierszach i d kolumnach taką że pierwszy wiersz to minimalne wartości w każdym wymiarze, a drugi wiersz to maksymalne wartości w każdym wymiarze.
6. Niech X przechowuje n punktów (n wierszy) o d współrzędnych (d kolumn). Niech Y przechowuje m punktów (m wierszy) o d współrzędnych (d kolumn). Zwróć wektor R, który ma długość m, taki że jego i-ta współrzędna oznacza indeks punktu X, który ma najmniejszy dystans do i-tego punktu w Y.