Danmarks Tekniske Universitet

Kursusnavn	Introduktion til programmering og databehandling
Kursusnummer	02631, 02632, 02692
Hjælpemidler	Alle hjælpemidler er tilladt
Varighed	2 timer
Vægtning	Opgaverne vægtes ens

Indhold

Aflevering

Du skal aflevere dine løsninger elektronisk:

1. Du kan uploade dine løsninger individuelt på CodeJudge (dtu.codejudge.net/prog-f17/assignment) under *Afleveringer/Exam*. Når du afleverer en løsning på CodeJudge bliver det test-eksempel som fremgår i opgavebeskrivelsen kørt på din løsning. Hvis din løsning består denne ene test, vil den fremgå som *Submitted*. Det betyder at din løsning består på dette ene test-eksempel. Du kan uploade til CodeJudge så mange gange som du ønsker under eksamen.

2. Du skal uploade alle dine løsninger på CampusNet. Hver assignment skal uploades som en separat .py fil med samme navn som funktionen i opgaven:

   (a) `salePrice.py`
   (b) `ISBNCheckDigit.py`
   (c) `partialCorrelation.py`
   (d) `zeroCrossing.py`
   (e) `pairwiseRank.py`

   Filerne skal afleveres separat (*ikke* som en zip-fil) og skal have præcis disse filnavne.

Efter eksamen vil dine løsninger blive automatisk evalueret på CodeJudge på en række forskellige tests for at undersøge om de generelt fungerer korrekt. Bedømmelsen af din aflevering foretages udelukkende på baggrund af hvor mange af de automatiserede test der bestås.

- Du skal sikre dig at din kode følger specifikationerne nøje.
- Hver løsning må ikke indeholde nogen yderligere kode udover den specificerede funktion.
- Husk at du kan tjekke om dine løsninger følger specifikationerne ved at uploade dem til CodeJudge.
- Bemærk at alle vektorer og matricer anvendt som input eller output skal være numpy arrays.

Assignment A Sale price

En butik har udsalg og giver et tilbud: "Køb to og få den billigste gratis." Du får til opgave at implementere tilbudet i butikkens computersystem. Når en kunde ønsker at købe et antal varer, er metoden til at beregne den totale pris følgende: Kunden betaler for den dyreste vare, får den næst-dyreste vare gratis, betaler for den tredje-dyreste vare, får den fjerde-dyreste vare gratis, og så videre.

Problemdefinition

Skriv en funktion med navnet `salePrice` der tager som input en vektor med priser på varer som en kunde har valgt, og beregner den samlede tilbuds-pris.

Løsningsskabelon

```
def salePrice(prices):
  #insert your code
  return total
```

Input

`prices`	Priser på de valgte varer (vektor af decimaltal).

Output

`total`	Samlet udsalgs-pris (decimaltal).

Eksempel

Forestil dig en kunde der har valgt 5 varer med de følgende priser:

$$9.95,\ 129.50,\ 9.95,\ 40.00,\ 17.75$$

Hvis vi sorterer priserne,

$$129.50,\ \cancel{40.00},\ 17.75,\ \cancel{9.95},\ 9.95$$

kan vi se at den samlede udsalgspris kan beregnes som:

$$129.50 + 17.75 + 9.95 = \underline{157.20}$$

A

Assignment B ISBN-10 check digit

Den sidste karakter i et 10-cifret International Standard Book Number (ISBN) er et kontrol-ciffer som kan beregnes udfra de første 9 cifre. Det beregnes ved at gange hvert ciffer med dets position i tallet (begyndende med position 1, talt fra venstre) og derefter tages summen af disse produkter modulo 11. Hvis vi betegner de 10 cifre som *abcdefghij* hvor j er kontrol-cifret, kan kontrol-cifret beregnes som:

$$j = (1 \cdot a + 2 \cdot b + 3 \cdot c + 4 \cdot d + 5 \cdot e + 6 \cdot f + 7 \cdot g + 8 \cdot h + 9 \cdot i) \bmod 11$$

Bemærk at kontrol-cifret er et tal mellem 0 og 10, hvor værdien 10 repræsenteres som karakteren `X`.

Problemdefinition

Skriv en funktion med navnet `ISBNCheckDigit` der tager som input en vektor med de første 9 cifre i et ISBN nummer og beregner kontrol-cifret som en streng indeholdende enten et numerisk ciffer `0`–`9` eller `X`.

Løsningsskabelon

```
def ISBNCheckDigit(isbn):
  #insert your code
  return checkDigit
```

Input	
`isbn`	Første 9 cifre i et ISBN nummer (vektor af heltal 0–9).

Output	
`checkDigit`	Kontrol-ciffer (tekst-streng, `0`–`9` eller `X`).

Eksempel

Eksempel 1: Forestil dig følgende første 9 cifre i et ISBN nummer: [1, 4, 9, 1, 9, 3, 9, 3, 6]. Kontrol-cifret kan beregnes som:

$$\begin{aligned} j &= (1 \cdot 1 + 2 \cdot 4 + 3 \cdot 9 + 4 \cdot 1 + 5 \cdot 9 + 6 \cdot 3 + 7 \cdot 9 + 8 \cdot 3 + 9 \cdot 6) \bmod 11 \\ &= (1 + 8 + 27 + 4 + 45 + 18 + 63 + 24 + 54) \bmod 11 \\ &= 244 \bmod 11 = 2 \end{aligned}$$

Således skal strengen `2` returneres.

Eksempel 2: Forestil dig følgende første 9 cifre i et ISBN nummer: [1, 5, 8, 4, 8, 8, 3, 8, 8]. Kontrol-cifret kan beregnes som:

$$\begin{aligned} j &= (1 \cdot 1 + 2 \cdot 5 + 3 \cdot 8 + 4 \cdot 4 + 5 \cdot 8 + 6 \cdot 8 + 7 \cdot 3 + 8 \cdot 8 + 9 \cdot 8) \bmod 11 \\ &= (1 + 10 + 24 + 16 + 40 + 48 + 21 + 64 + 72) \bmod 11 \\ &= 296 \bmod 11 = 10 \end{aligned}$$

Således skal strengen `X` returneres.

Assignment C Partial correlation

Indenfor statistik er *partiel korrelation* (partial correlation) et mål for graden af association mellem stokastiske variable hvor effekten af et sæt andre variable er fjernet. Givet en $N \times K$ matrix X der indeholder N observationer af K stokastiske variable, kan en $K \times K$ matrix P indeholdende partielle korrelationer mellem hvert par af variable beregnes som følger:

$$C = \left(\frac{1}{N} X^\top X\right)^{-1}, \qquad p_{i,j} = \begin{cases} \dfrac{-c_{i,j}}{\sqrt{c_{i,i} \cdot c_{j,j}}} & i \neq j \\ 1 & i = j \end{cases}$$

hvor $c_{i,j}$ betegner element i, j i matricen C og $p_{i,j}$ betegner element i, j i matricen P.

Problemdefinition

Skriv en funktion med navnet `partialCorrelation` der tager som input en matrix af observationer X og beregner matricen af partielle korrelationer P.

Løsningsskabelon

```
def partialCorrelation(X):
  #insert your code
  return P
```

Input

`X`	Observationer ($N \times K$ matrix).

Output

`P`	Partielle korrelationer ($K \times K$ matrix).

Eksempel

Forestil dig følgende input matrix X hvorfra matricen C kan beregnes:

$$X = \begin{bmatrix} -0.1 & 0.4 & -1.0 \\ 3.4 & 1.8 & -1.0 \\ -1.9 & -1.5 & 0.7 \\ 0.8 & 1.6 & 4.4 \\ 0.7 & 1.0 & 3.0 \end{bmatrix}, \qquad C \approx \begin{bmatrix} 3.422 & -4.742 & 0.934 \\ -4.742 & 7.219 & -1.437 \\ 0.934 & -1.437 & 0.448 \end{bmatrix},$$

Matricen af partielle korrelationer P kan derefter beregnes som:

$$P \approx \begin{bmatrix} 1 & 0.954 & -0.754 \\ 0.954 & 1 & 0.799 \\ -0.754 & 0.799 & 1 \end{bmatrix},$$

hvilket er det endelige resultat (her vist med tre decimalers præcision). Bemærk at matricen er symmetrisk og at diagonalen indeholder et-taller.

C

Assignment D Zero crossing

I digital audio-signalanalyse kan mange forskellige features anvendes til at karakterisere et signal. Et eksempel på en feature er *zero crossing*, dvs. antallet af gange signalet skifter fortegn.

Problemdefinition

Skriv en funktion med navnet `zeroCrossing` der tager som input et signal (som en vektor) og returnerer antallet af gange signalet skifter fortegn.

Løsningsskabelon

```
def zeroCrossing(signal):
  #insert your code
  return count
```

Input

`signal` Audio-signal (vektor med decimaltal).

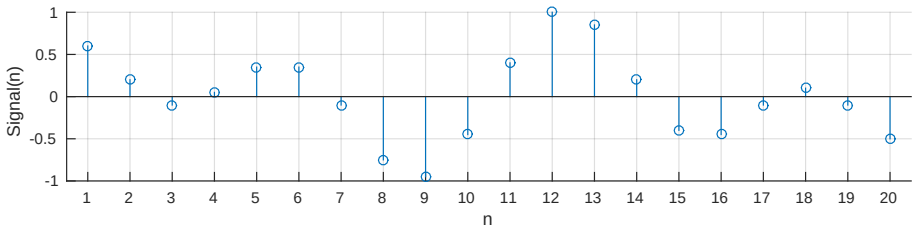
Output

`count` Antallet af fortegnsskift (heltal).

Eksempel

Forestil dig følgende signal som input til funktionen (negative tal vist med rød farve):

[0.6, 0.2, −0.1, 0.1, 0.3, 0.3, −0.1, −0.8, −0.9, −0.5, 0.4, 1.0, 0.8, 0.2, −0.4, −0.5, −0.1, 0.1, −0.1, −0.5]



Da signalet skifter fortegn 7 gange, skal funktionen returnere tallet 7.

D

Assignment E Ranking from pairwise comparison

Et antal mennesker får fremvist to forskellige produkter (udvalgt fra K produkter) og bliver spurgt hvilket de foretrækker. Dette resulterer i et datasæt som kan gemmes i en $K \times K$ matrix P med parvise præferencer, hvor hvert element $p_{i,j}$ angiver hvor mange gange produkt i er blevet foretrukket frem for produkt j.

Lad $n_{i,j}$ betegne det totale antal sammenligninger mellem produkt i og j (samlet i matricen N), og lad w_i betegne det totale antal sammenligninger "vundet" af produkt i (samlet i vektoren w). Vi kan beregne disse udfra matricen P således:

$$n_{i,j} = p_{i,j} + p_{j,i}, \qquad w_i = \sum_{j=1}^{K} p_{i,j}.$$

Vi ønsker nu at rangordne produkterne ifølge præferencerne. Vi anvender følgende algoritme:

Initialisér $r_i = 1$ for $i \in 1, \ldots, K$
Gentag *100 gange*
- Beregn $\rho_i = \dfrac{w_i}{\sum_{j=1}^{K} \frac{n_{i,j}}{r_i + r_j}}$ for $i \in 1, \ldots, K$
- Beregn $r_i = \dfrac{\rho_i}{\sum_{j=1}^{K} \rho_j}$ for $i \in 1, \ldots, K$

Efter at have kørt algoritmen indeholder r_i en "score" for hvert produkt

Problemdefinition

Skriv en funktion med navnet `pairwiseRank` der tager som input en matrix P indeholdende parvise præferencer og beregner rangordens-scorer vha. ovenstående algoritme.

Løsningsskabelon

```
def pairwiseRank(P):
  #insert your code
  return score
```

Input	
`P`	Matrix med parvise præferencer.

Output	
`score`	Rangordens-scorer (vektor med K elementer).

Eksempel

Forestil dig følgende matrix med parvise præferencer, hvorfra det totale antal sammenligninger og det totale antal "vundne" sammenligninger kan beregnes:

$$P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 5 & 0 & 1 & 6 \\ 8 & 8 & 0 & 5 \\ 7 & 3 & 1 & 0 \end{bmatrix}, \qquad N = \begin{bmatrix} 0 & 6 & 8 & 8 \\ 6 & 0 & 9 & 9 \\ 8 & 9 & 0 & 6 \\ 8 & 9 & 6 & 0 \end{bmatrix}, \qquad w = [2,\ 12,\ 21,\ 11].$$

Efter at have kørt algoritmen får vi tallene $r \approx [0.0150,\ 0.1258,\ 0.7746,\ 0.0845]$ hvilket er det endelige resultat.

E