SD 2018-2019
Tema1
Corecție, eliminarea excepțiilor și completarea datelor secvențiale

Roxana Pavelescu pavelescu.roxana13@gmail.com
Alexandru Andrei Rotaru rotarualexandruandrei94@gmail.com
deadline soft - 31 Martie 2019 23:55
deadline hard - 2 Aprilie 2019 23:55

University Politehnica of Bucharest

Tema 1

Schimbări

Față de varianta anterioară de enunț:

- am corectat formulele 6, 4 si 5
- am schimbat și corectat exemplul din secțiunea 1.4
- $k = 5$ peste tot în enunț
- am adăugat o precizare legată de modul de rulare
- am modificat explicațiile de la 1.1 pentru a fi în concordanță cu checker-ul

Introducere

În ziua de azi, majoritatea problemelor ajung să fie rezolvate cu tehnici de învățare automată. Pentru ca aceste tehnici să funcționeze, este nevoie de cantități semnificative de date. Aceste date provin de obicei dintr-o varietate de surse, iar majoritatea sunt nesigure. Deoarece în procesul de învățare calitatea datelor determină direct calitatea rezultatelor obținute, este foarte important ca datele să fie corecte și consistente. De aceea, o mare parte din timp, analiștii îl petrec încercând să corecteze și să completeze aceste date.

Posibile probleme prezente într-un set de date:

- valori de date ce nu au sens și nu ar trebui să existe (outlier/valori excepționale)
- lipsa datelor în anumite intervale de timp, datorate unor defecțiuni temporare (gaps)
- oscilația prea mare a valorilor (zgomot/noise)
- datele sunt neuniform distribuite în timp (într-o secundă avem 5 înregistrări, în următoarea secundă 22 înregistrări)

În figura 1 puteți vedea un exemplu de set de date generat de o piesă hardware cu defecte. Fâșiile albe reprezintă intervale fără date. Fâșiile verzi reprezintă intervale de date cu zgomot. Fâșia albastră reprezintă date perfecte, iar cea cu albastru deschis reprezintă date cu valori de excepție. Zona colorată mov reprezintă un interval de date neuniform distribuite în timp. Cu roșu avem reprezentate datele corecte, dacă nu am fi avut nicio problemă în procesul de colectare.

Tema 1

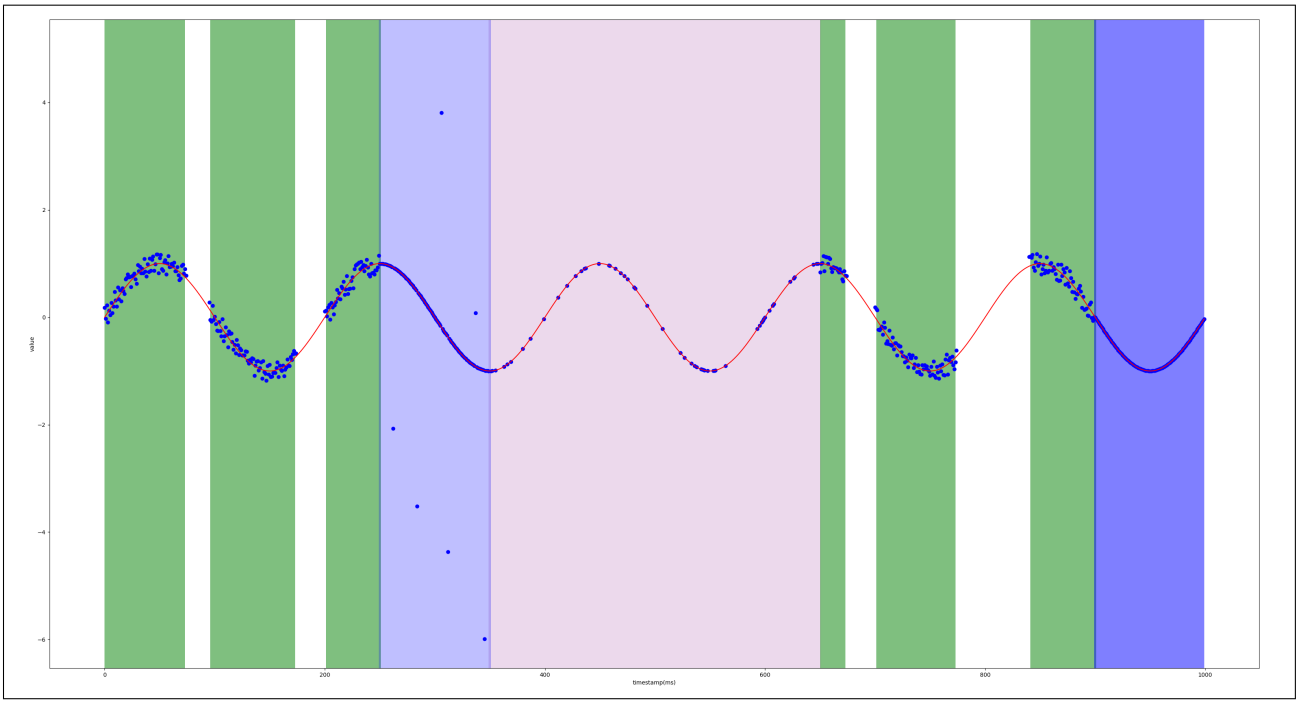

Figure 1: Valoriile funcției sinus cauzate de un senzor defect

Scopul temei Rolul vostru este de a implementa 4 metode de remediere a problemelor enunțate mai sus, folosind structurile de date despre care ați învățat la curs (**liste**). După implementarea acestei teme ar trebui să:

- puteți identifica problemele în care folosirea listelor este adecvată
- să puteți folosi operațiile de bază pe liste pentru a rezolva o problemă dată
- înțelegeți avantajele oferite de liste față de alte structuri de date

1 Sarcini de lucru

1.1 Eliminare de excepții folosind metode statistice

Eliminarea excepțiilor se va realiza prin parcurgerea listei de valori, folosind o fereastră de dimensiune $k = 5$ pentru care se va calcula la fiecare pas **media** și **deviația standard**.Pentru simplitate, vom considera k impar. Fereastra va fi construită în jurul unui nod și nodul respectiv va fi eliminat dacă diferă prea mult de celelalte nodurile din fereastră. Primele $int(k/2)$ si ultimele $int(k/2)$ valori din listă vor fi ignorate în procesul de eliminare și vor rămâne în listă.

$$\bar{x} = \frac{\sum_i^n x_i}{n}, \textit{media valorilor numerice din fereastră} \quad (1)$$

$$\sigma = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n}}, \textit{deviația standard a valorilor din fereastră} \quad (2)$$

Dacă valoarea din nodul curent nu este în intervalul $[\bar{x} - \sigma, \bar{x} + \sigma]$ nodul va fi eliminat din listă.

Exemplu Pentru o intrare de tipul:
10
0.000000
0.314159
0.628314
0.942464
− 1.743396
1.570732
1.884844
2.198938
2.513010
2.827057
Explicație: Se parcurg elementele de la 0.63 până la 2.19 inclusiv

0.00 0.31 **0.63** 0.94 -1.74 average=0.03 deviation=1.05 - ok
0.31 0.63 **0.94** -1.74 1.57 average=0.34 deviation=1.12 - ok
0.63 0.94 ***-1.74*** *1.57 1.88 average=0.66 deviation=1.43 - not ok*
0.94 -1.74 **1.57** 1.88 2.20 average=0.97 deviation=1.58 - ok
-1.74 1.57 **1.88** 2.20 2.51 average=1.28 deviation=1.73 - ok
1.57 1.88 **2.20** 2.51 2.83 average=2.20 deviation=0.49 - ok

Outputul va arăta:
9
0.000000
0.314159
0.628314
0.942464
1.570732
1.884844
2.198938
2.513010
2.827057

1.2 Eliminare de zgomot folosind filtre

Timestamp-ul datelor rămâne identic cu cel inițial. Se modifică doar valoarea. Cu alte cuvinte, veți modifca valoarea datelor, nu și timestamp-ul. Pentru fiecare valoare calculată cu unul din filtrele de mai jos, timestamp-ul va fi cel din centrul sublistei folosite în calculul respectiv.

1.2.1 Filtrare mediană

Valoarea **mediană** se obține sortând mulțimea și luând valoarea de poziția din mijloc. Filtrarea mediană funcționează considerând o listă de lungimea $k = 5$ în jurul fiecărui nod. Pentru fiecare fereastră se consideră valoarea mediană, și se creează o noua listă din aceste valori.

Exemplu $[1, 2, 3, 6, 19, 7, 8, 6, 4, 4, 4, 21, 4, 2, -1]$
$mediana([1, 2, 3, 6, 19]) \longrightarrow mediana([1, 2, \mathbf{3}, 6, 19]) \longrightarrow 3$
$mediana([2, 3, 6, 19, 7]) \longrightarrow mediana([2, 3, \mathbf{6}, 7, 19]) \longrightarrow 6$
$mediana([3, 6, 19, 7, 8]) \longrightarrow mediana([3, 6, \mathbf{7}, 8, 19]) \longrightarrow 7$
$mediana([6, 19, 7, 8, 6]) \longrightarrow mediana([6, 6, \mathbf{7}, 8, 19]) \longrightarrow 7$
$mediana([19, 7, 8, 6, 4]) \longrightarrow mediana([4, 6, \mathbf{7}, 8, 19]) \longrightarrow 7$
...
$mediana([4, 4, 21, 4, 2]) \longrightarrow mediana([2, 4, \mathbf{4}, 4, 21]) \longrightarrow 4$
$mediana([4, 21, 4, 2, -1]) \longrightarrow mediana([-1, 2, \mathbf{4}, 4, 21]) \longrightarrow 4$
Rezultat $\longrightarrow [3, 6, 7, 7, 7, 6, 4, 4, 4, 4, 4]$

1.2.2 Filtrare folosind media aritmetică

Se parcurge lista de valori considerând subliste de lungime $k = 5$. Pentru fiecare sublistă, se calculează media aritmetică și se adaugă rezultatul într-o altă listă.

Exemplu $[1, 2, 3, 6, 19, 7, 8]$
$media([1, 2, 3, 6, 19]) \longrightarrow 6.2$
$media([2, 3, 6, 19, 7]) \longrightarrow 7.4$
$media([3, 6, 19, 7, 8]) \longrightarrow 8.6$
Rezultat $\longrightarrow [6.2, 7.4, 8.6]$

1.3 Uniformizarea frecvenței în timp a datelor

Anumite intervale de timp pot sa conțină mai multe date decât altele. Ideal, ne-am dori ca frecvența datelor să fie constantă, însă, datorită modului de colectare sau senzorilor, rareori

se întâmplă acest lucru. Vom încerca să *mutăm* datele din zonele cu frecvență mare către zonele cu frecvența mai mică. Dacă diferența temporală între oricare două date consecutive se află în intervalul $\in [0.1, 1]$ secunde, atunci se va înlocui valoarea nodului curent cu valoarea medie dintre nodul anterior și nodul curent.

$$(t_i, x_i) = (\frac{t_{i-1} + t_i}{2}, \frac{x_{i-1} + x_i}{2}) \tag{3}$$

Exemplu Pentru o intrare de tipul:
3
1000 2.0
2000 3.0
2500 7.0
Output-ul va arăta:
3
1000 2.0
1500 2.5
2000 4.75

1.4 Completarea datelor

Dacă intervalele lipsă sunt mai mari de un prag fixat, atunci procesul de completare va fi mai complex decât o simplă medie între margini. Pentru fiecare interval vom considera cei mai apropiați vecini la stânga și la dreapta. Pentru simplitate, luăm k = 3 vecini pentru fiecare margine a intervalului curent. Folosind aceste valori, aproximăm valorile care ar fi trebuit să apară în acel interval, folosind o medie ponderată a vecinilor. Pentru a putea parametriza această formulă, introducem și un factor de scalare C, care ne permite sa generăm puncte oriunde în intervalul $\in [right[k].timestamp, left[k].timestamp]$ 6.

Cele mai apropiate elemente de marginile intervalului se află spre finalul listelor **right** și **left**.

$$C = \frac{timestamp - left[k].timestamp}{right[k].timestamp - left[k].timestamp}; \tag{4}$$

Deoarece am vrea ca influența punctelor să scadă pe măsură ce ne depărtăm de intervalul curent, introducem un coeficient care va scădea influenta valorilor pe masura ce ne departam de interval. Parametrul i semnifică poziția curentă a punctului, iar k reprezintă numărul de puncte luate în considerare.

$$w(i, k) = \frac{(\frac{i}{k-1})^2 * 0.9 + 0.1}{\sum_i^k ((\frac{i}{k-1})^2 * 0.9 + 0.1)} \tag{5}$$

$$f(left[], right[], timestamp) = (1 - C) * (\sum_{i}^{k} left[i] * w(i,k)) + C * \sum_{i}^{k} right[i] * w(i,k) \quad (6)$$

Vom folosi aceste formule pentru a ne asigura că fiecare interval de 1 secundă din set are cel puțin 5 valori (5Hz).

Exemplu Să presupunem că avem în listă avem la un momentan următoarele valori și avem pragul de 1 secundă:
6
1801 1.71
2100 2.8
2196 2.53
4753 0.84
4900 1.62
5255 1.64

Atunci completarea valorilor în intervalul listă va arăta:
18
1801 1.71
2100 2.80
2196 2.53
2396 2.42
2596 2.31
2796 2.19
2996 2.08
3196 1.96
3396 1.85
3596 1.73
3796 1.62
3996 1.51
4196 1.39
4396 1.28
4596 1.16
4753 0.84
4900 1.62
5255 1.64

Explicație: diferența (4753-2196) >1000 - adică mai mare decât 1 secunde, așadar, pe intervalul acesta, avem nevoie să generam valori. Vom genera valori începând cu 2196 + 200 și

până când ajungem la 4596 sau depășim această valoare. Pentru calculul valorilor, aplicăm efectiv formulele de mai sus, unde k = 3, left = [(1801, 1.71), (2100, 2.8) (2196, 2.53)], right = [(4753, 0.84), (4900, 1.62), (5255, 1.64)], iar în calculul C-ului, valoarea timestamp este dată de timestamp-ul pentru care dorim să completăm cu date. Pentru timestamp-ul 2396 valorile intermediare sunt:
C(2396, left, right) = 0.08
w(0, 3) = 0.07
w(1, 3) = 0.23
w(2, 3) = 0.70
f(left, right, 2396) = 2.42

1.5 Bonus: Statistici 20 puncte

Se consideră intervalul $\mathbb{R}$ împărțit în intervale de lungime δ. Pentru fiecare interval se va număra de câte ori avem un punct în mulțimea de date care se află în acel interval. La final, se vor afișa pe cate o linie intervalele sortate după marginea inferioară și numărul de elemente ce s-au regăsit în acel interval.

Exemplu Pentru $\delta = 250$ $l = [0, 1, 4, 12, 15, 251, 252, 1008, 1128]$
Rezultat
$[0, 250]$ 5
$[250, 500]$ 2
$[1000, 1250]$ 2

2 Mod de rulare

Pentru rezolvarea temei veți avea de urmărit o serie de comenzi pe care le veți primi ca argumente din linie de comandă. Citirea și scrierea se vor face la **stdin** respectiv **stdout**.

Argumentele pe care programul vostru va trebui să le implementeze sunt:

- e1 - eliminare de excepții folosind statistici
- e2 - eliminare de zgomot prin filtru median
- e3 - eliminare de zgomot folosind filtrul bazat pe media aritmetică
- u - uniformizarea frecvenței
- c - completarea datelor lipsă

- st< *delta* > - Se vor calcula statisticile conform descrierii din secțiunea 1.5. Intervalul de timp δ va trebui extras din argument.

```
./tema1 --st1000
```

Programul vostru trebuie să poată primi mai multe argumente la un moment dat, iar acestea vor fi executate în ordinea în care apar în linia de comandă. Comenzile de prelucrare vor avea forma:

```
./tema1 --command1 --command2 --command3
```

3 Formatul datelor

Modul de reprezentare a datelor:

$$D = \{x | x = (timestamp, value), timestamp \in [0, 300000], value \in \mathbb{R}\} \tag{7}$$

timestamp - valoarea este primită ca număr întreg, reprezentând timpul în milisecunde.

Formatul datelor de intrare/ieșire:
numarPerechi
timestamp valoareReala
timestamp valoareReala
...
timestamp valoareReala

Exemplu:
1
92 38.10
Specificație: precizia datelor trebuie să fie de 2 zecimale.

4 Precizări și restricții

- Temele trebuie să fie încărcate pe vmchecker. NU se acceptă teme trimise pe e-mail sau altfel decât prin intermediul vmchecker-ului.
- O rezolvare constă într-o arhivă de tip zip, care conține toate fișierele sursă, alături de un Makefile, ce va fi folosit pentru compilare, și un fișier README, în care se vor preciza detaliile de implementare. Toate fișierele se vor afla în rădăcina arhivei
- Punctajul alocat pentru Readme este **5p** iar pentru coding style este tot **5p**.

- Makefile-ul trebuie să aibă obligatoriu regulile pentru **build** și **clean**. Regula build trebuie să aibă ca efect compilarea surselor și crearea binarului.
- **Nu** se pot folosi **vectori**. Implementarea se va face în mod exclusiv cu **liste** (liste de liste etc.)
- Deadline-ul soft al temei este 31 martie. Se vor mai accepta trimiteri de teme până pe 2 aprilie, cu depunctare 10 puncte pe zi.