Mining the potential relationship between cancer cases and industrial pollution based on high influence ordered-pair patterns

Abstract This supplementary document presents the proofs and the detail of the function "searchRI".

1. The influence index does not meet the downward closure property.

Proof. We use an example to illustrate the problem. For example, there are three patterns $pc_1 = <\{a,b\},\{B,C\}>$, $pc_2 = <\{b\},\{B,C\}>$, $pc_3 = <\{a\},\{B\}>$ in Figure 1, and table 1 show the table instances of three patterns.

(1) $FIR(B,pc_1) = \sum_{B.t\in\pi_{c_j}(TI(pc_1))} SII(B.t)/FIS(B) = (SII(B.1)+SII(B.2))/FIS(B)$

$FIR(B,pc_2) = \sum_{B.t\in\pi_{c_j}(TI(pc_2))} SII(B.t)/FIS(B) = (SII(B.1)+SII(B.2))/FIS(B)$

In the pattern pc_1, B.1 is affected by a.1 and b.1; in the pattern pc_2, B.1 is affected by b.1. Based on the definition of superimposed influence, the superimposed influence of B.1 in the pattern pc_1 is bigger than that of B.1 in the pattern pc_2. The same true for the instance B.2, so $FIR(B,pc_1) > FIR(B,pc_2)$. The same can be obtained, $FIR(C,pc_1) > FIR(C,pc_2)$,

so $PII(pc_1) = min(FIR(B,pc_1), FIR(C,pc_1)) > min(FIR(B,pc_2), FIR(C,pc_2)) = PII(pc_2)$

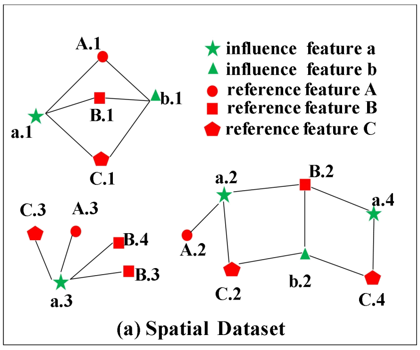

Figure 1 an example of a spatial dataset

Table 1 table instances of the two patterns in Figure 1

a	b	B	C	b	B	C	a	B
a.1	b.1	B.1	C.1	b.1	B.1	C.1	a.1	B.1
a.2	b.2	B.2	C.2	b.2	B.2	C.2	a.2	B.2
a.4	b.2	B.2	C.4	b.2	B.2	C.4	a.4	B.2
							a.3	B.4
							a.3	B.3

(2) $FIR(B,pc_1) = (SII(B.1)+SII(B.2))/FIS(B) = (0.595+0.522)/(0.595+0.523+ 0.413+ 0.555) = 0.536$

$PII(pc_3) = (SII(B.1)+SII(B.2)+SII(B.3)+SII(B.4))/FIS(B) = \left(0.414+ 0.308+ 0.413+ 0.555\right)/2.086 = 0.81$

$PII(pc_1) \le FIR(B,pc_1) < PII(pc_3)$

To sum up, $PII(pc_2) \le PII(pc_1) < PII(pc_3)$, so the influence index does not meet the downward closure property.

Lemma 1 (Conditional Monotonicity) If influence ordered-pair patterns have the same influence features, the influence index is anti-monotone as the size of patterns increase.

Proof. Given two influence ordered-pair patterns $pc = \langle IFS_{pc}, RFS_{pc} \rangle$, $pc' = \langle IFS_{pc}, RFS_{pc'} \rangle$ where $RFS_{pc'} \subseteq RFS_{pc}$.

For a reference feature $c_j \in (RFS_{pc} \cap RFS_{pc'})$, any instance of c_j participating in a row instance of the pattern pc also certainly participates in a row instance of the pattern pc', so $FIR(c_j, pc) \le FIR(c_j, pc')$,that is, the influence ratio of the feature is antimonotone. The influence index of the pattern is also antimonotonic

because:

$$PII(pc) = min_{c_j \in RFS_{pc}}(FIR(c_j, pc)) \le min_{c_j \in RFS_{pc}}(FIR(c_j, pc')) \le min_{c_j \in RFS_{pc'}}(FIR(c_j, pc')) = PII(pc') \ .$$

Lemma 2 *The limit influence index of a pattern is an upper bound of the influence index of the pattern.*

Proof. The maximum of the superimposed influence of $c_j.t$ is **max superimposed influence** of $c_j.t$, so $SII(c_j.t) \le MSII(c_j.t)$.

$$FIR(c_j, pc) = \sum_{c_j.t \in \pi_{c_j}(TI(pc))} SII(c_j.t) / FIS(c_j) \le \sum_{c_j.t \in \pi_{c_j}(TI(pc))} SII(c_j.t) / FIS(c_j) = LIR(c_j, pc)$$

$$PII(pc) = min_{c_j \in R}(FIR(c_j, pc)) \le min_{c_j \in R}(LIR(c_j, pc)) = LII(pc) \ .$$

Lemma 3 *The limit influence index is anti-monotone as the size of patterns increase.*

Proof. Given two influence ordered-pair patterns $C = \langle I, R \rangle$, $C' = \langle I', R' \rangle$ and a feature f_k where $C' \subseteq C$, $I' \cup R' \cup \{f_k\} = I \cup R$.

(1) For an influence feature $c_j \in (I \cap I')$, any instance of c_j that participates in a row instance of the pattern C also certainly participates in a row instance of the pattern C', so $LIR(c_j, C) \le LIR(c_j, C')$, that is, the limit influence ratio is antimonotone.

(2) 1) if f_k is a reference feature, $\langle I', R' \cup \{f_k\} \rangle = \langle I, R \rangle = C$

From lemma 1, it can be known that $LIR(f_i, \langle I', R' \cup \{f_k\} \rangle) \le LIR(f_i, \langle I', R' \rangle)$

$$\begin{aligned}
LII(C) &= LII(\langle I', R' \cup \{f_k\} \rangle) = \min_{f_i \in R' \cup \{f_k\}}(LIR(f_i, \langle I', R' \cup \{f_k\} \rangle)) \\
&= min(\underset{f_i \in R'}{LIR}(f_i, \langle I', R' \cup \{f_k\} \rangle), LIR(f_k, \langle I', R' \cup \{f_k\} \rangle) \\
&\le \min_{f_i \in R'}(LIR(f_i, \langle I', R' \cup \{f_k\} \rangle)) \\
&\le \min_{f_i \in R'}(LIR(f_i, \langle I', R' \rangle)) = LII(C')
\end{aligned}$$

2) if f_k is an influence feature, $\langle I' \cup \{f_k\}, R' \rangle = \langle I, R \rangle = C$

and it can be seen from 1 that $LIR(c_j, C) \le LIR(c_j, C')$.

$$LII(C) = \min_{f_i \in R}(LIR(f_i, C)) = \min_{f_i \in R'}(LIR(f_i, C)) \le \min_{f_i \in R'}(LIR(f_i, C')) = LII(C')$$

so limit influence index is antimonotone.

Lemma 4 *The participating instances of* f_i *in an influence ordered-pair pattern* pc *must be included in* $CPIS(f_i, pc)$ *,* i.e., $PIS(f_i, pc) \subseteq CPIS(f_i, pc)$.

Proof. $\forall f_i.j \in PIS(f_i, pc)$, there must be a row instance containing $f_i.j$ .According to the join method, if $f_i.j$ participate in the row instance of pc ,then $f_i.j$ must participate in row instance of pc_1 and row instance of pc_2 at the same time, i.e., $f_i.j \in \{PIS(f_i, pc_1) \cap PIS(f_i, pc_2)\}$, so $PIS(f_i, pc) \subseteq CPIS(f_i, pc)$.

Lemma 5 *For an influence ordered-pair pattern* pc *and the corresponding feature* f_i *,* $CFIR(f_i, pc) = \sum_{f_i.j \in CFIR(f_i, pc)} SII(f_i.j) / FIS(f_i)$ *is an upper bound of the influence ratio of* f_i *in* pc *.*

Proof. From Theorem 4, it can be known that $PIS(f_i, pc) \subseteq CPIS(f_i, pc)$

$$\therefore CFIR(f_i, pc) = \sum_{f_i.j \in CFIR(f_i, pc)} SII(f_i.j) / FIS(f_i) \ge \sum_{f_i.j \in PIS(f_i, pc)} SII(f_i.j) / FIS(f_i) = FIR(f_i, pc)$$

Algorithm 3: $RI =$ searchRI($f_i.j$, pc)

---

1) $k = pc.$length()
2) RI .resize(k) //The capacity of RI is set to k
3) $OssArr = \varnothing$
4) **for** $f_p \in \{pc - \{f_i\}\}$ **do:**
5) get $Oss(f_i.j, f_p, pc)$ and $OssArr[f_p] = Oss(f_i.j, f_p, pc)$
6) **end for**
7) $featurePos = 0$
8) f_p = $pc[featurePos]$
7) **for** $instancePos = 0$; $instancePos < Oss.$size(); $instancePos{+}{+}$
8) $RI[0] = particiInstanceArr[instancePos]$ //Select this instance, and search for the next feature
9) gen_RI_recursion(RI , $OssArr$, k , $featurePos + 1$,0, $k - 1$)
10) if(verifyRowInstance((RI)) **return** RI
11) gen_RI_recursion(RI , $OssArr$, k , $featurePos$, $instancePos$ + 1, k) //Do not select this instance, and search the next instance of the feature
12) if(verifyRowInstance((RI)) **return** RI
13) **return** $\varnothing$

Algorithm 4: gen_RI_recursion(RI , $OssArr$, k , $featurePos$, $instancePos$, $remainder$)

1) **if** $remainder == 0$: **return** // The remaining position is 0, exit the recursion
2) $f_p = pc[featurePos]$
3) $Oss = OssArr[f_p]$
4) **if** $instancePos + 1 > Oss$ .size() **return** // $instancePos$ starts from 0; exceed the size of the search space Oss , then exit the recursion
5) $RI[featurePos] = Oss[instancePos]$ //Select the instance, search the next feature, and $instancePos$ is set to 0, and subtract 1 from $remainder$.
6) gen_RI_recursion(RI , $OssArr$, k , $featurePos + 1$, 0 , $remainder$ -1);
7) gen_RI_recursion(RI , $OssArr$, k , $featurePos$, $instancePos + 1$, $remainder$);//Do not select this instance, and search the next instance for the feature