УДК 681.3.00:007

О.С. Балабанов

ПРИСКОРЕННЯ АЛГОРИТМІВ ВІДТВОРЕННЯ БАЙЄСОВИХ МЕРЕЖ. АДАПТАЦІЯ ДО СТРУКТУР БЕЗ ЦИКЛІВ

Показано можливості вдосконалення алгоритмів відтворення структури моделі залежностей з даних. Пропонована модернізація, не погіршуючи роботу алгоритму в загальному випадку, дозволяє відтворювати структури залежностей типу ліс (дерево) чи полі-ліс на основі тестів першого рангу, що значно зменшує обчислювальну складність.

Вступ

Байєсова мережа – це модель, що описує систему ймовірнісних залежностей в наочній структурованій формі. Такі моделі звуть також каузальними мережами, системами структуральних рівнянь, моделями незалежності на орієнтованих графах і т. д. [1–5]. Байєсові мережі – зручний апарат для моделювання неповністю спостережуваних систем і розв'язання аналітичних задач за схемою: «відоме значення певних змінних – знайти значення інших змінних моделі». Байєсова мережа може бути або сконструйована експертом, або виведена автоматично (зокрема, на основі фактів умовної незалежності). Виведення можливе у формі синтезу (коли незалежності задані як знання) або у формі емпіричної індукції (на основі даних спостережень за об'єктом моделювання). Через те, що алгоритми емпіричної індукції зазвичай розраховані на випадок відсутності апріорних знань про модель (і часто невідомим є навіть темпоральний порядок змінних моделі), ці алгоритми мають перебірний характер і є дуже складними обчислювально [2–4]. Натомість для спеціальних простих підкласів структур моделей розроблено низку простих алгоритмів виведення. Проте зрозуміло, що спеціалізовані алгоритми нездатні відтворити адекватну модель в загальному випадку. Водночас, аналітик не може передбачити, чи належить адекватна модель до спеціального підкласу структур. У роботі показано, як можна надати універсальному алгоритму адаптивних властивостей. Пропонується модифікувати алгоритм так, що, зберігаючи здатність працювати зі складними моделями, алгоритм забезпечує простоту виведення у випадку, коли структура генеративної моделі є однозв'язною.

Відправні положення

Важливо розрізняти дві постановки задачі індуктивного виведення моделі:

- відтворювальна (реконструкційна);
- редукційна (апроксимаційна).

Коли генеративна модель вкладається в інструментальний клас методу виведення, це – відтворення моделі (реконструкція); коли ні – редукція (структурна апроксимація). Алгоритм редукції видає «покриття» істинної моделі сурогатною. Результати даної роботи спрямовані на реконструкцію моделі.

Нагадаємо елементарні поняття. Шлях у графі – це послідовність сусідніх ребер $X - Y - \cdots - Z$ будь-якої орієнтації, де всі проміжні вершини – різні. Оршлях (строго орієнтований шлях) – це шлях, на якому всі ребра орієнтовані узгоджено, тобто в одному напрямку. Орієнтований цикл (орцикл, «циклон») – це оршлях, на якому початкова вершина тотожна останній. Якщо в орграфі є дуга $X \to Y$, то вершина X зветься «батьком» вершини Y, а вершина Y – дитиною X. Коли орієнтація дуги $X \to Y$ невідома чи необов'язкова, позначаємо її як ребро $X — Y$. (Кажемо, що X та Y суміжні.) Між вершинами графа та змінними моделі встановлена взаємнооднозначна відповідність.

Ациклічний орієнтований граф (АОГ) – це орграф без «циклонів». АОГ-модель залежностей – це пара $(G, \mathcal{G})$, де G – АОГ, а $\mathcal{G}$ – сукупність локально

заданих параметрів. У байєсових мережах змінні дискретні (номінальні), а параметри – це локально визначені умовні розподіли ймовірностей. АОГ-моделі з неперервними змінними, лінійними залежностями та нормальними дистурбаціями звуться гауссовими мережами. Результати роботи дійсні також для гауссових мереж (як і для всіх форм параметризації моделі).

Визначення 1. Колізором (колайдером) в орграфі зветься фрагмент із двох дуг вигляду $X \to Y \leftarrow Z$. Ланцюг (безколізорний шлях) в орграфі – це шлях, який не містить жодного колізора.

Підкласи АОГ-моделей визначають «топологічними» обмеженнями на структуру графа моделі. Найбільш відомі монопотокові моделі, полі-ліси й ліси. Ці підкласи легко визначаються і вкладаються один в інший [6, 7]. Монопотоковий граф – це орграф, у якому кожен цикл суміжності має два чи більше колізорів [6]. Полі-ліс – це орграф, у якому немає циклів. Ліс – це орграф, у якому немає ані циклів, ані колізорів. У лісі кожна вершина має не більше одного батька. (Зазначимо, що в літературі часто вживаються терміни «дерево» та «полідерево». Але поняття ліс є більш повним, бо охоплює випадок, коли граф розпадається на окремі компоненти.)

Названа ієрархія моделей – не лише формальна. Ці підкласи різняться експресивними можливостями, статистичними властивостями і принципами індуктивного виведення.

Визначення 2 (d-сепарація). Шлях π в АОГ-моделі називають d-закритим за допомогою (кондиціонування) множини вершин $\mathbf{S}$, якщо і тільки якщо

1) існує вершина X, $X \in \mathbf{S}$, $X \in \pi$, і на шляху π є дуга $X \to$ чи $\leftarrow X$, **або**

2) на шляху π лежить хоча б один колізор $\to Y \leftarrow$, причому $Y \notin \mathbf{S}$ й немає жодного оршляху вигляду $Y \to \cdots \to Z$, де $Z \in \mathbf{S}$.

Якщо при кондиціонуванні множини $\mathbf{S}$ існує хоча б один d-відкритий шлях між X та Y, то говорять, що вершини X та Y є d-залежні (d-з'єднані). У противному разі (коли немає жодного відкритого шляху) вершини X та Y є d-незалежні, а множина $\mathbf{S}$ d-сепарує вершини X та Y. Будемо позначати таку d-сепарацію предикатом $\mathrm{Ds}(X;\mathbf{S};Y)$, а множину $\mathbf{S}$ називатимемо (графовим) сепаратором для пари (X,Y); неявно розуміється $X,Y \notin \mathbf{S}$. Коли множина $\mathbf{S}$ є пустою, факт d-сепарації записується коротше: $\mathrm{Ds}(X;;Y)$.

У застосуванні до структур, які не мають колізорів, пункт 2 критерію d-сепарації можна відкинути і дати спрощений критерій (forest d-separation): шлях є d-закритим за допомогою множини вершин $\mathbf{S}$, якщо і тільки якщо на цьому шляху лежить принаймні одна вершина, що входить до $\mathbf{S}$.

Факти d-сепарації зчеплені зі структурою АОГ-моделі за принципом:

$$(X \text{ — } Y) \Leftrightarrow \forall \mathbf{Z}: \neg \mathrm{Ds}(X;\mathbf{Z};Y).$$

Умовну незалежність X від Y при кондиціонуванні (фіксації) набору змінних $\mathbf{Z}$ будемо виражати предикатом $\mathrm{Pr}(X;\mathbf{Z};Y)$, де $X,Y \notin \mathbf{Z}$. Така незалежність означає, що $p(Y \mid X,\mathbf{Z}) = p(Y \mid \mathbf{Z})$ або, еквівалентно, $p(XY \mid \mathbf{Z}) = p(X \mid \mathbf{Z})p(Y \mid \mathbf{Z})$. При цьому множина змінних $\mathbf{Z}$ зветься емпіричним сепаратором для пари (X,Y). Для систем лінійних залежностей умовну незалежність виражають через частний коефіцієнт кореляції $\rho_{XY.\mathbf{Z}} = 0$.

Згідно відомої теореми від Verma&Pearl (див. [2, 4]), критерій d-сепарації визначає марківські властивості АОГ-моделі:

$$\mathrm{Ds}(X;\mathbf{Z};Y) \Rightarrow \mathrm{Pr}(X;\mathbf{Z};Y). \qquad (1)$$

Усі такі структурно-детерміновані факти умовної незалежності є дійсні за будь-якої параметризації моделі.

Методи відтворення АОГ-моделей з даних базуються на припущенні каузальної необманливості (faithfulness) розподілу ймовірностей щодо генеративної АОГ-моделі [2, 4, 8]. Це припущення виражається як імплікація, обернена відносно (1). Але в скінченній відбірці даних це припущення виконується тільки у локальному сенсі, і то не завжди. Втім, для

практики можна послабити припущення необманливості. Відомі методи constraint-based (або «сепараційного») підходу до виводу структури моделі, зокрема, алгоритм PC [2, 3], спираються на таке правило ідентифікації ребра:

$$(X \text{ — } Y) \Leftrightarrow \forall \mathbf{Z} : (X, Y \notin \mathbf{Z}) : \neg \Pr(X; \mathbf{Z}; Y).$$

Кардинальність умови $\mathbf{Z}$ визначає ранг тесту умовної незалежності $\Pr(X; \mathbf{Z}; Y)$. Із зростанням рангу збільшується перебірна складність виведення і відбувається «подрібнення» відбірки даних. Необхідно уникати складних сепараторів та екстенсивного їх пошуку.

Алгоритми сепараційного підходу можуть бути застосовані в двох царинах (або режимах):

- у режимі синтезу, коли модель виводиться з бази знань про незалежності;
- у режимі виведення з відбірки даних.

Основна частина алгоритму має два етапи. На першому етапі ідентифікують всі ребра моделі, на другому – визначають орієнтації (спрямування) ребер. При виведенні моделі з даних додається третій етап – обчислення параметрів.

Псевдокод основної частини алгоритму PC (за [2]) записано на рис. 1.

Зауважимо, що для відтворення АОГ-моделей відомі алгоритми з поліноміальною кількістю тестів. Але для повної характеристики обчислювальної складності необхідно врахувати, що ті алгоритми використовують тести складного формату.

Між тим відомо, що ліси та полі-ліси відтворюються з даних алгоритмами квадратичної або суб-кубічної складності [9, 10]. Але ці алгоритми спеціалізовані й при застосуванні до загального випадку АОГ-моделей опиняються в режимі редукції. Для монопотокових моделей теж існують спеціалізовані алгоритми суб-кубічної складності [7].

У наступному розділі показано, як можна за допомогою простих засобів надати алгоритмам, розрахованим на відтворення АОГ-моделей, адаптивних властивостей, які забезпечать економічне відтворення структур залежностей без циклів. Пропонуються такі вдосконалення, що алгоритм, залишаючись універсальним, зможе відтворювати ліси та полі-ліси залежностей на основі тестів першого рангу, подібно спеціалізованим алгоритмам.

```
S1   Form the complete undirected graph U on the set of variables V
S2   k = 0
     Repeat
       For each pair of variables X and Y that are adjacent in (the
       current) U such that ADJ(U;X)\{Y} or ADJ(U;Y)\{X}  has at
       least k elements, check through the subsets of ADJ(U;X)\{Y}
       and the subsets of ADJ(U;Y)\{X} that have exactly k
       variables. If a subset S is found conditional on which X and
       Y are independent, remove the edge between X and Y in U, and
       record S as Sepset(X;Y)
       k = k + 1
     until for each ordered pair of adjacent variables X and Y,
     ADJ(U;X)\{Y} has less than k elements
 S3  Execute the orientation rules
```

Рис. 1. Опис алгоритму PC

Можна бачити, що алгоритм шукає можливий сепаратор для пари, перебираючи всі підмножини суміжних вершин, причому цей перебір він виконує також коли є ребро. Це призводить до виконання тестів високого рангу (більшість з яких – непродуктивні).

Основний результат

Ліси залежностей можна характеризувати такою аксіомою:

$$(X \text{ — } Y) \& \neg \mathrm{Ds}(Y;;Z) \Rightarrow \neg \mathrm{Ds}(X;;Z) \& \\ \& [\mathrm{Ds}(X;Y;Z) \text{ або } \mathrm{Ds}(Y;X;Z)]. \quad (2)$$

(Нагадаємо, що сепарація – симетрична). Зазначимо, що аксіома (2) покриває заборону циклонів, а аксіома монопотокових моделей з [6] – ні.

Інакше, ліси залежностей можна описати наступною аксіомою: якщо є дуга $X \to Y$, то не існує такої Z ($Z \neq X$), що є дуга $Z \to Y$. Але до цієї аксіоми треба додати заборону циклонів. Крім того, останній варіант характеризації незручний тим, що явно не виражає марковських властивостей. Відсутність колізорів означає, що в кожній зв'язаній компоненті моделі є один-єдиний корінь. (Але корінь не ідентифікується відношеннями незалежності.)

Для характеризації полі-лісів залежностей необхідна складніша аксіома:

$$(X \text{ — } Y) \& \neg \mathrm{Ds}(Y;;Z_1) \& \neg \mathrm{Ds}(Z_1;;Z_2) \& \ldots$$
$$\ldots \& \neg \mathrm{Ds}(Z_{k-1};;Z_k) \Rightarrow \text{ або } \mathrm{Ds}(X;;Z_k),$$
$$\text{або } \mathrm{Ds}(X;Y;Z_k), \text{ або } \mathrm{Ds}(Y;X;Z_k). \quad (3)$$

Неформальний коментарій до аксіом. Залежність у деревах завжди транзитивна, при цьому для нереберної залежності існує одноелементний сепаратор. У полі-деревах припускається ще інакший паттерн залежностей – не транзитивна залежність через колізор. Звісно, при переході до емпіричного виміру залежність може виявитися слабкою або навіть колапсувати в дискретних моделях (порушення (1)).

З метою обґрунтування «сепараційних» методів реконструкції було введено наступне поняття [8, 11].

Визначення 3. Локально-мінімальним сепаратором в АОГ для пари вершин (X,Y) зветься такий сепаратор $\mathbf{Z}$, що після вилучення з $\mathbf{Z}$ будь-якого його члена він перестає бути сепаратором для (X,Y). Формально: $\mathrm{Ds}(X;\mathbf{Z};Y) \&$ $\& \forall W \in \mathbf{Z} : \neg \mathrm{Ds}(X;\mathbf{Z} \setminus \{W\};Y)$.

Очевидно, у лісі та полі-лісі кожний локально-мінімальний сепаратор є або одноелементним, або пустим.

Для загального випадку АОГ-моделі було доведено [11] таке правило.

Твердження 1 (правило «відсторонення» кандидатів у сепаратор (‘placing aside’)). Якщо в АОГ-моделі вірне $\mathrm{Ds}(Z;X;Y) \& \neg \mathrm{Ds}(Z;;Y)$, то вершина Z не є членом ніякого локально-мінімального сепаратора для пари вершин (X,Y).

Для дерев залежностей правило відсторонення сприймається як мало не очевидне, і довести його простіше. А емпірична форма правила відсторонення у випадку лісів є коректна навіть без припущень необманливості.

З правила «відсторонення» та твердження 1 з [11] (про «стрижень» сепаратора) випливає наступне правило:

Твердження 2 (правило єдиної перемички; ‘single spanning link’):

$$\neg \mathrm{Ds}(X;;Y) \& \forall Z \in \mathbf{U} \setminus \{X,Y\} : [\mathrm{Ds}(Z;;X) \vee$$
$$\vee \mathrm{Ds}(Z;;Y) \vee \mathrm{Ds}(Z;X;Y) \vee \mathrm{Ds}(Z;Y;X)] \Rightarrow$$
$$\Rightarrow X \text{ — } Y .$$

Тут $\mathbf{U}$ – множина всіх вершин моделі; символ “$\vee$” означає диз'юнкцію.

Розгорнувши аксіому (2) за допомогою критерію d-сепарації, легко бачити, що кожне ребро дерева ідентифікується правилом єдиної перемички. Дійсно, нехай є ребро $X \text{ — } Y$. Тоді кожна вершина, d-залежна від X та Y, відсторонюється. При цьому множина d-залежних вершин розпадається на дві підмножини: «ліві» та «праві». «Ліві» вершини зв'язані з Y через X, а «праві» вершини зв'язані з X через Y. Відтак, для всіх «лівих» вершин Z буде $\mathrm{Ds}(Z;X;Y)$, а для всіх «правих» вершин W буде $\mathrm{Ds}(X;Y;W)$. Тож, на основі тестів першого рангу ідентифікується таке: а) існування ребра $X \text{ — } Y$ (згідно правила єдиної перемички); б) відсутність ребер між вершиною Y та «лівими» вершинами Z; в) відсутність ребер між X та «правими» вершинами W.

Отже, дерево буде цілком ідентифіковане тестами першого рангу.

Для полі-лісів правило єдиної перемички діє аналогічно, тільки для деяких пар вершин замість відсторонення спрацьовує терм $\mathrm{Ds}(Z;;X)$ або терм $\mathrm{Ds}(Z;;Y)$. Отже, маємо такий результат.

Твердження 3. Якщо в лісі (або полі-лісі) існує ребро $X \text{ — } Y$, то щодо пари (X,Y) для всіх вершин $Z \in \mathbf{U} \setminus \{X,Y\}$ виконується умова правила єдиної

перемички, тобто

$$\mathrm{Ds}(Z;;X)\vee \mathrm{Ds}(Z;;Y)\vee$$

$$\vee \mathrm{Ds}(Z;X;Y)\vee \mathrm{Ds}(Z;Y;X).$$

Результат формульовано у графових термінах (що відповідає режиму синтезу моделі), але він чинний також і для режиму виведення з даних. Треба тільки замінити графові предикати $\mathrm{Ds}(*)$ на ізоморфні емпіричні предикати $\mathrm{Pr}(*)$.

Для коректного відтворення лісів та полі-лісів залежностей з даних за допомогою пропонованого алгоритму достатньо самого простого припущення – полегшеної форми реберної необманливості [8]. Пропонована модифікація гарантує виграш, допоки ліс не деградує, тобто не перетвориться на ланцюг чи не розпадеться на ізольовані ребра.

Спрогнозуємо роботу алгоритму PC у випадку дерева залежностей. Ясно, що алгоритм PC не може «знати», що невідома структура належить до підкласу дерев, а відтак, буде працювати як звичайно у загальному випадку. Алгоритм PC легко виявить відсутність ребер на основі тестів першого рангу. Але далі алгоритм PC буде продовжувати шукати сепаратор для всіх тих пар вершин, які поєднані ребрами. І ця безперспективна робота буде продовжуватися допоки алгоритм не випробує всі підмножини вершин, суміжних до кожної вершини, дотичної до ребра, що розглядається. При цьому алгоритм може дійти до тестів високого рангу. Нагадаємо, що при збільшенні рангу тесту ускладнюється обчислення (з даних) інформації, потрібної для виконання тесту.

Якщо вмонтувати в алгоритм PC (після циклу $k=1$) правило «відсторонення» та правило єдиної перемички, то після виконання тестів першого рангу ці два правила спрацюють, і процес виведення лісу (полі-лісу) залежностей на тому й завершиться. Потрібна модифікація алгоритму зводиться до того, аби ті пари вершин, для яких спрацювало правило єдиної перемички, надалі не розглядалися (для них вже не треба шукати сепаратори).

Ілюстративні приклади виведення моделі

Аби наглядно показати, наскільки можна поліпшити алгоритм PC, розглянемо приклади. Нехай маємо дерево, де кожна вершина, за виключенням кінцевих («листя»), має r суміжних вершин. Для наочності візьмемо дерево, показане на рис. 2. Ступінь розгалуження r тут дорівнює чотирьом. Кількість вершин у цьому дереві дорівнює 29, кількість ребер – 28. Кількість крайніх ребер (у яких одна з вершин є «листям») дорівнює 20. Кількість «не крайніх» ребер (у обох кінців яких є по три сусідніх ребра) дорівнює 8.

Кількість тестів першого рангу, які

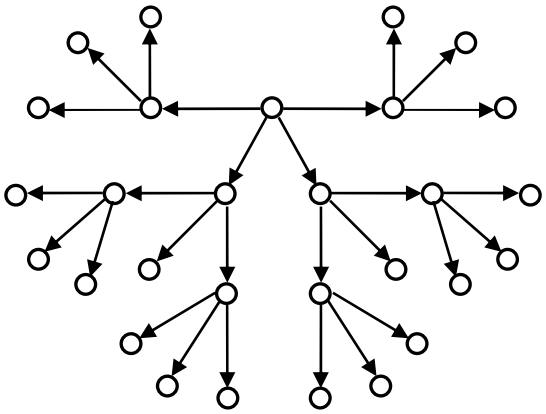


Рис. 2. Приклад дерева залежностей

виконає алгоритм PC при виведенні з реальної відбірки даних, точно спрогнозувати не можна. В цій моделі кількість тестів першого рангу буде не менше 6*9 = 54, якщо всі ланцюги довжиною два ребра продукують статистично значущі залежності. (Інакше відпадає потреба у тестах першого рангу). А максимальна кількість 29*28*27/2 = = 10962 тестів досягається, коли статистично міцними є всі ланцюги дерева (довжиною до 6 ребер включно).

Для ідентифікації кожного «не крайнього» ребра алгоритму PC знадобиться виконати два тести третього рангу і шість тестів другого рангу. Отже, при виведенні цієї структури алгоритм PC загалом виконає 6*8+3*20 = 108 тестів другого рангу та 2*8+20 = 36 тестів третього рангу. Кількість тестів другого та

третього рангу може зменшитися тільки при невиконанні реберної необманливості (але в такому разі алгоритм пропустить відповідні ребра).

Якщо озброїти алгоритм правилом єдиної перемички, то для виведення цієї структури буде достатньо виконати тести тільки нульового та першого рангу. Причому кількість таких тестів залишиться такою самою, що у РС.

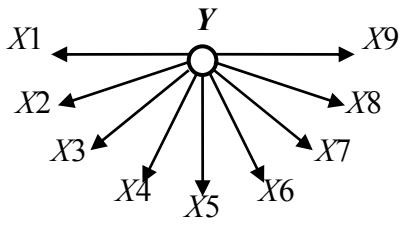
Легко перенести ілюстрацію на випадок полі-дерева. Взявши за основу структуру з рис. 2, можна змінити орієнтацію будь-яких ребер. Результатом буде полі-дерево. При цьому показники складності виведення моделі зміняться несуттєво, а саме, тільки зменшиться кількість тестів першого рангу.

Отже, зазначена модифікація алгоритму виведення надає радикального прискорення у випадку, коли генеративна модель має структуру дерева. Фактор прискорення буде зростати пропорційно ступені r розгалуження дерева.

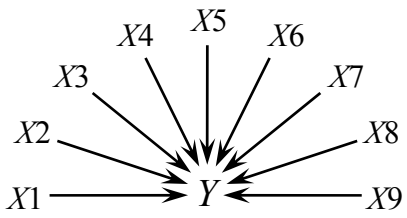
Недоліки алгоритму РС ще контрастніше проявляються у випадку 1-рівневого дерева (рис. 3, а). Така структура відома під назвою наївний байєсівський класифікатор. При реконструкції цієї моделі з 10-ма вершинами (одна коренева Y та 9 її дітей) алгоритм РС буде виконувати складні тести аж до 8-го рангу включно. Справді, оскільки вершина Y має 9 суміжних вершин, при верифікації кожного ребра $Y — Xi$ алгоритм буде випробувати (як можливі сепаратори) всі підмножини всіх вершин, за виключенням самих Y та Xi . У підсумку алгоритм РС виконає 9 тестів 8-го рангу, 72 тесту 7-го рангу і т. д. Сукупна кількість тестів від 2-го рангу до 8-го дорівнюватиме 2223. Це може виглядати парадоксом, але для цієї простої моделі алгоритм РС потребує таку саму кількість тестів вище першого рангу, як і для повно зв'язаної (насиченої) моделі з 10-ти вершин.

Натомість модифікований (за вказаними пропозиціями) алгоритм завершить виведення цієї моделі відразу після виконання тестів першого рангу і правила єдиної перемички.

Для моделі типу полі-дерево, показаному на рис. 3, б, характеристики будуть аналогічні. Алгоритм РС завершить реконструкцію цього полі-дерева тестами 8-го рангу. А модифікованому алгоритму буде достатньо тестів 1-го рангу. Єдина відмінність випадку полі-дерева від випадку дерева полягає у зменшенні кількості тестів 1-го рангу (бо відсутність ребер $Xi — Xj$ ідентифікується тестами нульового рангу).



а



б

Рис. 3. Прості структури, незручні для алгоритму РС:
а – однорівневе дерево залежностей;
б – однорівневе полі-дерево

Підсумок

Показано, що за рахунок простої корекції можна вдосконалити алгоритми constraint-based («сепараційного») підходу до виведення структури моделі. Внаслідок пропонованого підсилення алгоритм збереже здатність відтворювати моделі в загальному класі структур (без орієнтованих циклів), і водночас зможе виводити модель значно швидше, коли структура генеративної моделі не має ніяких циклів. Тобто алгоритм автоматично адаптується до моделей у підкласі лісів та полі-лісів залежностей.

Продемонстровані можливості адаптації індуктивного алгоритму до простих моделей не обмежуються випадком деревовидних структур залежностей. Подібне скорочення пошуку сепараторів можна забезпечити у всіх випадках, коли структура моделі містить принагідний фрагмент («розрізається» на частини простими тестами). Більш того, можна озброїти алгоритм багатьма іншими правилами, які значно розширять адаптивні можливості індуктивного виведення [8].

1. *Pearl J.* Probabilistic reasoning in intelligent systems: Networks of Plausible Inference. – San Mateo: Morgan Kaufmann, 1988. –552 p.
2. *Spirtes P., Glymour C., Scheines R.* Causation, prediction and search. 2nd Ed., New York: MIT Press, 2001. – 496 p.
3. *Spirtes P. and C. Glymour.* An algorithm for fast recovery of sparse causal graphs // Social Science Computer Review. – 1991. – Vol. 9. – P. 62–72.
4. *Neapolitan R.E.* Learning bayesian networks. – Upper Saddle River: Prentice Hall, NJ, 2004. – 693 p.
5. *Андон Ф.И., Балабанов А.С.* Структурные статистические модели: инструмент познания и моделирования // Системні дослідження та інформаційні технології. – 2007. – № 1. – С. 79 – 98.
6. *Балабанов О.С.* Системи ймовірнісних залежностей: графові та статистичні властивості // Математичні машини та системи. – 2009. – № 3. – С. 80–97.
7. *Балабанов А.С.* Реконструкция модели вероятностных зависимостей по статистическим данным. Инструментарий и алгоритм // Проблемы управления и информатики. – 2009. – № 6. – С. 90–103.
8. *Балабанов А.С.* Формирование минимальных d-сепараторов в системе зависимостей // Кибернетика и системный анализ. – 2009. – № 5. – С. 38–50.
9. *Chow C.K., Liu C.N.* Approximating discrete probability distributions with dependence trees // IEEE trans. on Information Theory. – 1968. – Vol. 14, N 3. – P. 462–467.
10. *Балабанов О.С.* Індуктивне відтворення деревовидних структур систем залежностей // Проблемы программирования. – 2001. – № 1–2. – С. 95–108.
11. *Балабанов О.С.* Правила підбору сепараторів в байєсівських мережах // Проблеми програмування. – 2007. – № 4. – С. 33–43.

Отримано 22.09.2010

***Про автора*:**

Балабанов Олександр Степанович,
старший науковий співробітник,
кандидат технічних наук.

***Місце роботи автора*:**

Інститут програмних систем
НАН України,
03187, Київ-187,
Проспект Академіка Глушкова, 40.
Тел.: (044) 526 3420.
e-mail: bas@isofts.kiev.ua