а-те-ма-ти-чес-ки

Основи на индуктивното машинно самообучение
(машинно самообучение без учител)

Здравко Марков

July 14, 1998

Contents

Chapter 1

Машинно самообучение без учител

1.1 Увод

Както и при човека процесът на машинното самообучение може да протича и със или без помощта на учител. Ролята на учителя при традиционните индуктивни методи е да зададе подходящо множество от примери заедно с тяхната класификация (например принадлежност или не към научаваното понятие) и да насочва работата на системата ако е необходимо (например чрез отговори на въпроси, които тя задава). Самообучението без учител най-често се свързва с обработката на данни (примери), получавани от някакъв независим от самообучаващата се система източник, с цел извличане на полезна информация или знания за процеса, който ги генерира. В тази област най-често се решават следните две конкретни задачи:

- *Кластеризация.* Групиране на примерите по отношение на някакъв признак за сходство. В тази област се включва както класическата (известна от приложната математика) агломеративна кластеризация, при която обектите само се групират, както и концептуалната кластеризация (същинска подобласт на МС), при която се създават и интензионални описания на кластерите.

- *Извличане на знания от данни* (ИЗД) (Knowledge Discovery in Databases – KDD). Подобно на машинното самообучение тази област е интердисциплинарна и използва методи както от самото МС така и от области като статистика, разпознаване на образи, невронни мрежи, бази от данни и изкуствен интелект. ИЗД е свързано с решаване на множество различни задачи, общото между които е, че се търсят нови методи и алгоритми с цел улесняване на анализа на големи бази от данни и тяхното по-ефективно използване. В този смисъл ИЗД е предимно приложна област.

1.2 Агломеративна кластеризация

Основната задача, която се решава в тази област, е при зададено множество от примери и критерий за сходство (близост, разстояние) между отделните примери те да се групират в класове (кластери). Броят на кластерите се задава предварително или се получава автоматично в процеса на работа на алгоритъма. Критерият, който се използува най-често при групиране на примерите, е да се *максимизира сходството между примерите от един кластер и минимизира сходстовото между примерите от различни кластери.*

1.2.1 Оценки за сходство и разстояния

Класически пример на задача за кластеризация е групирането на множество от точки в n-мерното евклидово пространство. Един очевиден критерий за сходство тук е *евклидовото разстояние.* Ако X и Y са точки с координати съответно $x_1, ..., x_n$ и $y_1, ..., y_n$ то изразът

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{1}$$

дефинира *разстояние* между P и Q. В общия случай за да бъде една функция d разстояние (метрика) в множеството X тя трябва да притежава следните свойства:

1. $d(x,y) > 0, \forall x,y \in X, x \neq y$;
2. $d(x,x) = 0, \forall x \in X$;
3. Ако $d(x,y) = 0$, то $x = y$[1];
4. $d(x,y) = d(y,x), \forall x,y \in X$ (симетричност);
5. $d(x,z) \leq d(x,y) + d(y,z), \forall x,y,z \in X$ (неравенство на триъгълника)

Формула (1) може да се прилага само към примери, чиито атрибути приемат числени стойности. За символни (номинални или структурни) или смесени атрибути се прилагат други функции. Най-простото разстояние за такива атрибути е е т.нар. *разстояние на Хеминг*[2]. То се дефинира като *брой на атрибутите с различаващи се стойности.* За илюстрация на този вид разстояние да разгледаме примерите, показани на фигура 1.1. Това са 4 примера на клетки, описани с три атрибута – цвят, брой на ядра и брой на опашки.

Ето няколко примера на разстояние между клетките от фигура 1.1:

$h(E_1, E_2) = 2$ (различен брой ядра и брой опашки)

[1] Без това условие имаме *псевдо-метрика.*

[2] Оригиналната версия на разстоянието на Хеминг се отнася за двоични вектори

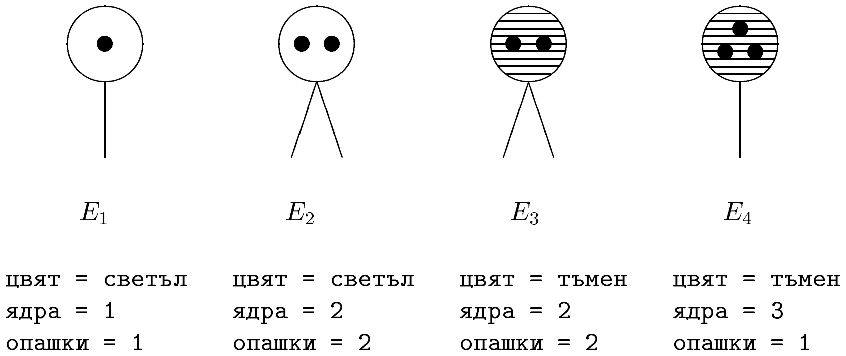

Figure 1.1: Примери на едноклетъчни организми,описани с три атрибута

$h(E_1, E_3) = 3$ (различен цвят, брой ядра и брой опашки)
$h(E_2, E_3) = 1$ (различен цвят)

Съществуват множество други критерии за сходство и разстояние върху примери, зададени със символни или смесени (символни и числови) атрибути. Тъй като те играят основна роля в алгоритмите за самообучение чрез запомняне тяхното подробно описание може да се намери в глава **??**. Тук ще разгледаме една мярка за сходство върху атомарни формули (вижте раздел **??**), тъй като тя позволява прилагането на алгоритмите за кластеризация или самообучение чрез запомняне с един много по-изразителен език - езика на *предикатното смятане от първи ред.*

Първо ще дефинираме функцията *размер на формула*, въведена от Рейнолдс в [8]. Нека A е атомарна формула (атом). Тогава $size(A)$ е броят на всички символи в A (броени толкова пъти, колкото пъти се срещат) минус броя на различните променливи. Например:

$size(p(f(a), a)) = 4$ (четири символа - 0 променливи)
$size(p(f(X), X)) = 3$ (четири символа - 1 променлива)
$size(p(f(X), Y)) = 2$ (четири символа - 2 променливи)

При сравняваване размерите на атоми, които се поглъщат (както е в примера по-горе) се вижда, че по-общите атоми имат по-малък размер. Това ни дава право да дефинираме функцията *разлика между атоми*, които се поглъщат. Нека A и B са атоми, т.че $A \geq B$ (т.е. $\exists\theta, A\theta = B$), тогава

$$diff(A, B) = size(A) - size(B).$$

Лесно се вижда, че разликата между атоми които са варианти (получават се един от друг чрез преименуване на променривите) е 0. Също така, ако $A > B$ (т.е. $A \geq B$, но $B \not\geq A$), то $diff(A, B) > 0$.

A_1 = прародител(баща(баща(баща(иван))))
A_2 = прародител(баща(майка(баща(иван))))
A_3 = прародител(баща(майка(иван)))
A_4 = прародител(баща(баща(мария)))
A_5 = прародител(баща(баща(майка(иван))))
A_6 = прародител(баща(майка(майка(иван))))

Figure 1.2: Основни атоми, описващи релацията "прародител"

Функцията $diff(A, B)$ дава добра оценка на сходството между атомите, които се поглъщат. Тъй като обаче релацията $\geq$ е частична наредба, очевидно съществуват атоми, за които не може да се изчисли разликата – това са атомите, несравними относно $\geq$. Решение на този проблем може да се намери като се приложи критерии за сходство, който се използува традиционно в йерархични структури – близостта на два елемента се свързва с дължината на пътищата до техния общ родител. Както е известно множеството от класовете на еквивалентност относно релацията $\geq$ върху термове образува *полурешетка* (вижте раздел ??). Следователно всеки два класа от еквивалентни термове имат единствена горна граница – класът на термовете, еквивалентни на тяхното най-малко общо обобщение (lgg). Тъй като разликата между термове в един клас на еквивалентност е 0, то очевидно функцията

$$d(A, B) = diff(lgg(A, B), A) + diff(lgg(A, B), B) \qquad (2)$$

дава оценка на сходството между два произволни терма (атома)[3] Въпреки че тази функция не отговаря на формалната дефиниция на метрика, тя е добро евристично приближение, удобно за практически цели.

За илюстрация на въведената оценка на сходство между атоми да разгледаме фигура 1.2, където е показано множество от основни атоми, описващи чрез функционални символи релацията "прародител":

За да приложим формула (2) към тези атоми е необходимо изчисляването на lgg. Например:

$d(A_1, A_5) = 4$ ($lgg(A_1, A_5)$ = прародител(баща(баща(X))))
$d(A_1, A_3) = 5$ ($lgg(A_1, A_5)$ = прародител(баща(X)))
$d(A_3, A_4) = 4$ ($lgg(A_1, A_5)$ = прародител(баща(X)))

При съблюдаването на основния критерии за кластеризацията (максимизиране на вътрешнокластерното сходство и междукластерното различие) очевидно ще е необходимо да се оценява и сходство между множества от

[3] Тук ограничаваме разглеждането до термове с еднакъв предикатен символ и брой аргументи. Това гарантира, че е lgg-то на такива термове е от същия тип (lgg на терми с различни функтори е променлива). Следователно ако разглеждаме предикатния символ като функционален символ, всички въведени дефиниции важат и за атоми.

примери. За *разстояние между множества* често се използуват функции, които всъщност не отговарят на формалната дефиниция на разстояние. За практическите цели на оценка на сходството обаче те са достатъчно добри приближения. Такива са например например минималното, максималното или средното разстояние между елементите на множествата.

Съществува формален подход, които ни дава възможност да дефинираме коректно разстояние между множества при наличието на разстояние между обектите им. Това е т.нар. *хаусдорфово разстояние*, което се определя по следния начин: ако d е разстояние в множеството X, то функцията

$$d'(A,B) = max \left\{ \begin{array}{l} max_{x\in A} min_{y\in B} d(x,y) \\ max_{y\in B} min_{x\in A} d(x,y) \end{array} \right.$$

е разстояние в множеството от всички непразни подмножества на X.

1.2.2 Основен алгоритъм

Нека е дадено множество от примери $S = \{E_1, E_2, ..., E_n\}$. Най-простият алгоритъм за кластеризация строи двоично дърво с листа $E_1, E_2, ..., E_n$. Алгоритъмът е следния:

1. Инициализация: построява се множество от дървета $S' = \{T_1, T_2, ..., T_n\}$, където $T_i = \{E_i\}$ $(i = 1, ..., n)$

2. Изчислява се сходството между всички двойки $_i$ и T_j $(i, j = 1, ..., n)$, например чрез разстояние между множествата от листата им.

3. Нека (T_k, T_l) е двойката с най-добро сходство. Тогава в множеството от дървета S' тя се замества с дърво, получено чрез свързване на корените на T_k и T_l с общ корен.

4. Повтарят се точки 2 и 3 докато в S' остане само един елемент.

На фигура 1.3 е показано двоичното дърво, получено чрез прилагане на алгоритъма към примерите на клетки, разгледани в предишния раздел. За мярка на сходство между дървета е използувано минималното хемингово разстояние между листата им. Фигурата илюстрира и начинът, по който дървото се чертае без пресичане на дъгите си. За тази цел при всяко прилагане на стъпка 3 от алгоритъма новополученото дърво се чертае винаги от едната страна (например най-вляво) като неговите поддървета се преместват.

Очевидно при втората итерация от прилагането на алгоритъма в точка 3 е направен случаен избор на двойка дървета. Това се дължи на факта, че разстоянията между всички двойки измежду множествата $\{E_1\}$, $\{E_2, E_3\}$ и $\{E_4\}$ са равни (в случая на 2). Използувайки този факт можем да конструираме по-общо кластерно дърво, в което поддърветата $\{E_1\}$, $\{E_2, E_3\}$ и $\{E_4\}$ да са директни наследници на корена. Разширението на алгоритъма,

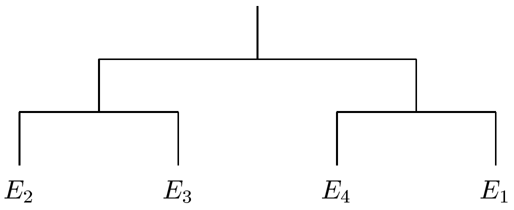

Figure 1.3: Двоично дърво, получено при кластеризацията на примерите на едноклетъчни организми

разгледано в следващия раздел дава директна възможност за построяване на подобни кластерни дървета.

Независимо от факта, че агломеративната кластеризация само групира примери и за разлика от индуктивните методи за МС не строй интензионални описани получените дървета носят ценна информация за областта на примерите. Кластерните дървета всъщност описват екстензионално съществуващи понятия в областта на примерите, като по-този начин лесно могат да се използват за индуктивно обобщение. Например групирането на две дървета означава, че между тях има сходство, т.е. някои атрибути имат еднакви стойности. Следователно само тези атрибути са достатъчни за описание на кластера (обобщение чрез пропускане на атрибут). Например кластерът, състоящ се от примерите E_2 и E_3 може да се опише като клетки, които имат по две ядра и по две опашки (вижте следващия раздел).

Предварителната обработка на примери е най-важното приложение на агломеративната кластеризация. Освен групирането на примерите в класове, което след това могат да се използват като вход за някои индуктивен алгоритъм, тя може да се използува за откриване на *типични примери* (например централните примери за даден кластер, т.е. такива, с минимално средно разстояние до останалите) или *шум* (примери, силно отдалечени от кластера).

1.2.3 Построяване на граф на обобщенията

Съществуват подходи, които заедно с агломеративната кластеризация извършват и индуктивната стъпка, т.е. построяването на интезионални описания на възлите на кластерното дърво. Тук ще разгледаме един такъв подход, който е директно разширение на основния алгоритъм за кластеризация, описан в предишния раздел.

Основата идея е да се използува естествената структура за обобщение на примерите, чрез която се изчисляват разстоянията между атоми. В този случай корена на дървото, получено чрез свързване на корените на T_k и T_l от стъпка 3 на алгоритъма може да се опише с lgg-то на листата на T_k и T_l (отбележете, че това lgg не винаги съвпада с lgg-то на корените на T_k и

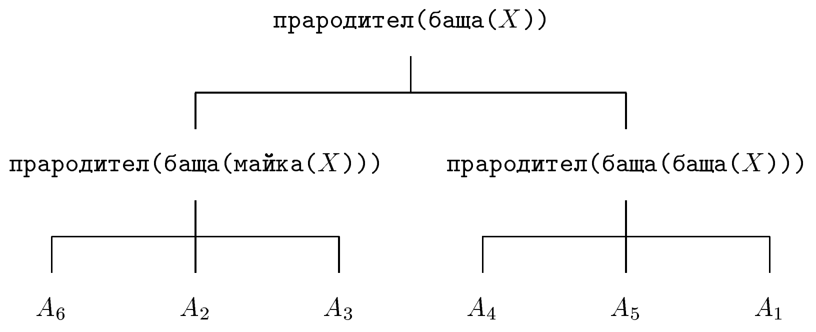

Figure 1.4: Дърво на обобщението с листа атомите, описващи релацията "прародител"

T_l). След това към този корен могат да се "прикачат" и всички дървета (и листа) от текущите кластери, чиито корени се поглъщат от него.

Описаната модификация на алгоритъма дава две съществени предимства:

- Алгоритъмът строи по-обща структура – *ацикличен граф*. Това позволява групирането на повече от два примера в един кластер, а също така допуска пресичане на кластерите (общи примери в различни кластери). Естествено по-този начин се отразява по-пълно и по-точно проблемната област на примерите.

- Заедно с групирането на примерите алгоритъмът строи и интензионални представяния на кластерите, които са естествени обобщения (индуктивни хипотези) за примерите.

По този начин алгоритъма за генериране на двоично кластерно дърво се превръща в алгоритъм за построяване на граф на обобщенията.

На фигура 1.4 е показан графът на обобщенията (в случая дърво) на атомите от фигура 1.2, получен при прилагането на описаното обобщение на алгоритъма за кластеризация. За разстояние между дървета е използувано средното разстояние между листата им, изчислено по формула (2).

Вижда се, че двата основни клона на дървото дават и двете дефиниции на релацията "прародител", описана екстензионално с примерите от листата. Разбира се, при по-сложно дърво или граф не винаги е ясно нивото, на което се получават "най-добрите" обобщение. В тези случаи е необходимо да се използуват други оценки върху множеството от дървета, с които работи алгоритъма, за да може обобщението да се преустанови на подходящо ниво. Възможни кандидати за такива оценки са *ентропията* (раздел ?.?) или *грешката* (раздел ?.? - ID3). При тяхното използуване може (а в случая на оценка на грешката, трябва) да се използува и класификацията на примерите (ако е известна). Тъй тако графът на обобщенията всъщност е една йерархична структура от индуктивни хипотези този вид оценки могат да

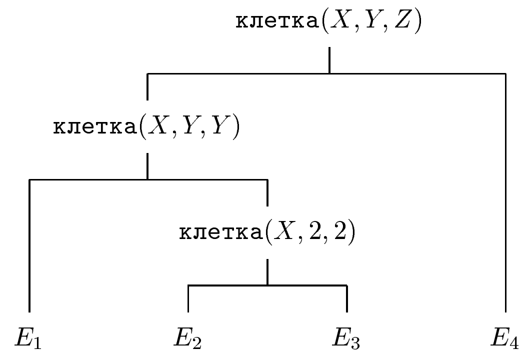

Figure 1.5: Дърво на обобщението с листа примерите на едноклетъчни организми, представени релационно

се прилагат и на стъпка 2 от алгоритъма за избор на най-добра двойка дървета които да се обединят в нова хипотеза вместо да се избират две дървета с най-малко разстояние между листата си.

Накрая на този раздел ще илюстрираме и как предложеното ръзширение на алгоритъма за кластеризация може да се прилага към примери, описани на езика атрибут-стойност. Всъщност това би трябвало да е ясно на читателя, тъй като този език е от съждителен тип и е подмножество на езика на предикатното смятане от първи ред.

Нека представим примерите на клетки от фигура 1.1 в релационен вид чрез предиката *клетка(¡цвят¿, ¡ядра¿, ¡опашки¿)*. В примерите на мястото на всеки атрибут стои неговата стойност. По този начин получаваме следните атоми:

E_1 = клетка(светъл, 1, 1)

E_2 = клетка(светъл, 2, 2)

E_3 = клетка(тъмен, 2, 2)

E_4 = клетка(тъмен, 3, 1)

Прилагайки алгоритъма за построяване на граф на обобщенията към тези атоми получаваме дървото, показано на фигура 1.5. Сравняването на това дърво и дървото, получено на основата на езика атрибут стойност (фигура 1.3), показва наличието на един нов кластер, обединяващ клетките с еднакъв брой ядра и опашки (клетка(X, Y, Y)). Това обобщение на примерите (индуктивна хипотеза), което очевидно отразява закономерност в изследваната проблемна област, показва по-голямата изразителност на езика на предикатното смятане от първи ред.

1.3 Концептуална кластеризация

В предишния раздел показахме, че агломеративната кластеризация може да се използува както за предварителна обработка на примерите, така и за директно генериране на индуктивни хипотези (когато оценката за сходство позволява това). Тук ще разгледаме, подходи за кластеризация, които са разработени специално за целите на индуктивното самообучение, т.е. тяхната цел е преди всичко създаването на интензионални описания на кластерите.

1.3.1 CLUSTER/2

CLUSTER/2 е един евристичен алгоритъм за концептуална кластеризация, създаден от Михалски ([7]). Тук ще споменем само основните му идеи, които позволяват да се оценят неговите предимства и недостатъци. Целта на алгоритъма е да се формират k категории (понятия, хипотези) в множество от некласифицирани обекти (примери), при предварително задано k. Най-общо CLUSTER/2 може да се опише по следния начин:

1. От първоначалното множество от обекти S се избират (по случаен начин или чрез някаква фунция за избор) k на брой обекта $s_1, s_2, ..., s_k$, наречени зародиши.

2. За всеки зародиш s_i $(i = 1, ..., k)$, използуван като положителен пример и останалите зародиши, използувани като отрицателни примери, се построява най-общата коректна хипотеза G_i $(i = 1, ..., k)$. За тази цел се използува някакъв стандартен алгоритъм за ИМС, например от фамилията покриващи алгоритми.

3. Класифицират се всички обекти от S чрез хипотезите G_i, в резултат на което се получават k категории – $S_1, S_2, ..., S_k$. След това всяка хипотеза G_i се специализира максимално при условие, че покрива всички примери от S_i. (Целта на тази стъпка е намаляване на евентуалното припокриване на категориите.)

4. Ако все още има припокриващи се категории, за всяка от тях се избира нов зародиш, който се явява център на категорията (в термините на някаква метрика или мярка за сходство), и се повтарят стъпки 2 и 3.

5. Алгоритъмът завършва кагато се удовлетвори някакъв предварително дефиниран критерии за оценка на хипотезите (например сложност).

6. Ако след няколко последователни стъпки няма подобрение на качеството на кластеризацията (не намалява припокриването на категориите S_i), то тогава се прави нов избор на зародиши по различен критерии (например гранични за категорията обекти) и алгоритъмът се повтаря.

Основната цел на този алгоритъм е да намери *необходими и достатъчни* условия за принадлежност към дадена категория (понятие). Такива условия може би са удобни за решаването на някои практически задачи, обаче са в противоречие с резултатите от психологическите изследвания на начините, по които човек категоризира обектите. Например в *теорията за фамилното сходство* ([9]) известният съвременен философ Витгенщайн твърди, че човек групира обектите не по наличието и отсъствието на дадени признаци (в термините МС, стойности на атрибути), а по-скоро чрез система от сходства между отделните представители на категориите. В подкрепа на това твърдение често се цитира примера за понятието ”игра” (вижте раздел ???). Може ли да се намери общо описание на всички игри (напр. детска игра, футбол, тото, шах и т.н.)? Очевидно това е трудно, а може би невъзможно. Много лесно е обаче да посочим приликите между всеки два представителя на това понятие.

Друга важна особеност на категоризацията при човека е изпозуването на *базови категории*. В отличие от строгите йерархични представяния (например таксономични дървета), масово използувани в класическите системи с ИИ, хората най-често използуват категории, които не са нито най-общите, нито най-специфичните от даден клас. Например, понятието ”стол” се използува много по-често от неговото обобщение ”мебел” или специализация – ”кухненски стол”.

Важно е да се отбележи също и още един когнитивен аспект на категоризацията – човек обикновен градира пренадлежността към дадено понятие. Системите за МС, които разгледахме до тук обаче класифицират обектите твърдо, без да отчитат степен (вероятност) на тяхната категоризация.

В следващия раздел ще разгледаме един алгоритъм за концептуална кластеризация, който притежава гореизброените качества, въпреки че не е създаден с претенциите за когнитивен модел.

1.3.2 COBWEB

COBWEB ([3]) е инкрементален алгоритъм за самообучение без учител, който генерира таксономично дърво от категории, без да е предварително зададен техния брой. Обектите се представят по обичайният начин за системите за ИМС – чрез набор от двойки ”атрибут-стойност”. Категориите обаче за разлика от хипотезите в ИМС, се представят *вероятностно* – чрез условната вероятност

$$p(f_i = v_{ij} | c_k)$$

атрибутът f_i да има стойност v_{ij}, при условие че обектът принадлежи на категорията c_k.

При постъпване на нов обект алгоритъмът или го включва директно в съществуваща категория или първо модифицира йерархията за да включи обекта след това. Критерият, използуван за тази цел, се основава на т.нар.

полезност на категоризацията, оценка предложена в [5] за целите на моделиране на базовите категории и потвърдена с психологически експерименти. Основната идея на тази оценка е, че добрата категоризация трябва да максимизира *едновременно*:

- вероятността обектите в една категория имат еднакви стойности на атрибутите си;
- вероятността обекти от различни категории да имат различни стойности на атрибутите се.

Тези два критерия се реализират чрез следната формула:

$$\sum_k \sum_i \sum_j p(f_i = v_{ij}) p(f_i = v_{ij}|c_k) p(c_k|f_i = v_{ij}), \tag{3}$$

където сумите са по всички категории, всички атрибути и всички стойности.

Членът $p(f_i = v_{ij}|c_k)$ дава оценка на вероятността един обект да има стойност v_{ij} на атрибута си f_j, при условие че принадлежи на категорията c_k. Колкото по-висока е тази вероятност, толкова по вероятно е два обекта в една и съща категория да имат общи стойности на атрибутите си. Членът $p(c_k|f_i = v_{ij})$ оценява вероятността един обект да принадлежи на категориата c_k, при условие че има стойност v_{ij} на атрибута си f_i. Колкото по-голяма е тази вероятност, толкова по-малко вероятно е обекти от различни категории да имат общи стойности на атрибутите си. Вероятността $p(f_i = v_{ij})$ се използува за теловен коефициент, който осигурява по-често срещаните двойки атрибут-стойност да имат по-голямо влияние върхо общата оценка.

Формулата на Бейс ни дава следната връзка между използуваните по-горе условни вироятности:

$$p(a_i = v_{ij}) p(c_k|a_i = v_{ij}) = p(c_k) p(a_i = v_{ij}|c_k).$$

Прилагайки тази връзка към формула (3) получаваме

$$\sum_k p(c_k) \sum_i \sum_j p(f_i = v_{ij}|c_k)^2 \tag{4}$$

Показано е, че изразът $\sum_i \sum_j p(f_i = v_{ij}|c_k)^2$ дава очакваният брой на двойки атрибут-стойност, които успешно могат да бъдат познати за произволен обект от класа c_k. Това "познаване" става с вероятност равна на вероятността за появата на съответната двойка, т.е. $f_i = v_{ij}$ е вярно с вероятност $p(f_i = v_{ij}|c_k)$. *Полезността на категоризацията* се дефинира като *нарастването* на броят на вярно познатите двойки $f_i = v_{ij}$ при наличието на k известни категории, в сравнение с броят на вярно познатите двойки без да е известна категоризацията на обектите (т.е. те принадлежат на една обща категория). Последният брой може да се запише като $\sum_i \sum_j p(f_i = v_{ij})^2$. За да оценим нарастването този член трябва да се извади от

(4). За може оценката да не зависи от броя на категориите, получената разлика трябва да се раздели на броя на категориите k. По този начин получаваме окончателната формула за *полезността на категоризацията*:

$$\frac{\sum_k p(c_k) \sum_i \sum_j p(f_i = v_{ij}|c_k)^2 - \sum_i \sum_j p(f_i = v_{ij})^2}{k} \quad (5)$$

Алгоритъмът COBWEB реализира търсене по метода на *най-бързото спускане* в пространстовото от възможни таксономии чрез прилагане на *полезността на категоризацията* като оценъчна функция за избор на следваща таксономия. При постъпване на нов обект, текущата таксономия се модифицира чрез прилагане на някой от следните четири оператора:

1. Добавяне на новия обект към съществуваща категория.
2. Създаване на нова категория (наследник на корена) и добавяне на обекта към нея.
3. Сливане на двете категории, които при добавянето на обекта към тях дават най-висока категориална полезност.
4. Разделяне на категорията, която при добавянето на обекта към нея дава най-висока категориална полезност.

Сливането на две категории C_1 и C_2 става като се създава нова категория C, която има за наследници C_1 и C_2, и предшественик – общият предшественик на C_1 и C_2. Разделянето е обратната операция, при която C се замества с C_1 и C_2, които от своя страна се свързват директно с предшественика на C.

COBWEB работи рекурсивно като започва от корена на категориалното дърво и обработва последоватлно по-долните нива по следния алгоритъм:
cobweb($Node$,$Instance$)

- Ако $Node$ е листо то:
  - Създават се два наследника на $Node$ - L_1 и L_2;
  - Приравняват се вероятностите[4] на L_1 с тези на $Node$;
  - Приравняват се вероятностите на L_2 с тези на $Instance$;
  - Добавя се $Instance$ към $Node$, изчислявайки съответно вероятностите на $Node$.
- В противен случай за всеки наследник C_i на $Node$ се изчислява стойността S_i на категориалната полезност на кластеризацията, получена от добавянето на $Instance$ към C_i, като:
  - S_1 = най-голямата стойност ($Instance$ е включен в C_1);

[4]Става въпрос за условните вероятности $p(f_i = v_{ij}|c_k)$, чрез които се представят категориите C_k

  - S_2 = втората по големина стойност след S_1 ($Instance$ е включен в C_2);
  - S_3 = стойността, получена при включването на $Instance$ в нова категория, наследник на $Node$;
  - S_4 = стойността, получена при сливането на C_1 и C_2;
  - S_5 = стойността, получена при разделянето на C_1.

- Ако най-голямата стойност е S_1, изпълнява се cobweb(C_1, $Instance$).

- Ако най-голямата стойност е S_3, то вероятностите на новата категория се установяват равни на тези на $Instance$.

- Ако най-голямата стойност е S_4, изпълнява се cobweb(C_3, $Instance$), където C_3 е резултатът от сливането на C_1 и C_2.

- Ако най-голямата стойност е S_5, то C_1 се разделя и се изпълнява cobweb($Node$, $Instance$).

Илюстрация на работана на алгоритъма е дадена на фигура 1.6, където е показана кластеризацията на примерите на едноклетъчни организми, използувани в раздел ??. Всяка категория C_i е показана като таблица със съответните условни вероятности на стойностите на атрибутите цвят, брой на ядрата и брой на опашките. Дадена е също и вероятността $P(C_i)$ на всяка категория C_i, която е равна на броя на обектите в нея разделен на общия брой обекти, обработени до момента. Категориите C_2, C_4, C_5 и C_6 съдържат по един обект – съответно E_1, E_4, E_2 и E_3.

Въпреки използувания алгоритъм за търсене по метода на най-бързото спускане, които както е известно често води до попадане в локален екстремум, експериментите с COBWEB показват, че той доста успешно успява да се справи с подобни ситуации. Причината за това е използуването на опрераторите за сливане и разделяне на категории, които в известен смисъл дават възможност за връщане назад в процеса на търсене и по-този начин компенсират "лошия" избор на категоризация на предишни стъпки от алгоритъма.

Едно естествено развитие на описания алгоритъм COBWEB е разширяването му с възможност за работа с числови атрибути с реални стойности. При обработката на числови атрибути в МС обикновено се използуват два подхода – заменянето им с номинални (например чрез подходяща дискретизация) и директното им използуване за описание на понятията.

Разширението на COBWEB използува втория подход. За тази цел при представянето на категориите вместо вероятности за появата на всяка двойка атрибут-стойност се използува функцията на разпределението на стойностите на атрибута. Използува се *нормално разпределение*, като за всеки атрибут в дадената категория се съхраняват неговата *средна стойност* (метематическо очакване) и *средноквадратично отклонение*.

C_1	$P(C_1) = 1.0$	
атр.	ст.	P
опашки	1	0.5
	2	0.5
цвят	светъл	0.5
	тъмен	0.5
ядра	1	0.25
	2	0.50
	3	0.25

C_2	$P(C_2) = 0.25$	
атр.	ст.	P
опашки	1	1.0
	2	0.0
цвят	светъл	1.0
	тъмен	0.0
ядра	1.0	1.0
	2	0.0
	3	0.0

C_3	$P(C_3) = 0.5$	
атр.	ст.	P
опашки	1	0.0
	2	1.0
цвят	светъл	0.5
	тъмен	0.5
ядра	1.0	0.0
	2	1.0
	3	0.0

C_4	$P(C_1) = 0.25$	
атр.	ст.	P
опашки	1	1.0
	2	0.0
цвят	светъл	0.0
	тъмен	1.0
ядра	1.0	0.0
	2	0.0
	3	1.0

C_5	$P(C_5) = 0.25$	
атр.	ст.	P
опашки	1	0.0
	2	1.0
цвят	светъл	1.0
	тъмен	0.0
ядра	1.0	0.0
	2	1.0
	3	0.0

C_6	$P(C_6) = 0.25$	
атр.	ст.	P
опашки	1	0.0
	2	1.0
цвят	светъл	0.0
	тъмен	1.0
ядра	1.0	0.0
	2	1.0
	3	0.0

Figure 1.6: Кластеризация на примерите на едноклетъчни организми чрез алгоритъма COBWEB

Оценъчната функция при търсенето в пространството от категоризации (полезността на категоризацията) също се модифицира за да може да оценява новото представяне на категориите. За тази цел е необходимо вътрешните суми от формула (5), т.е. $\sum_j p(f_i = v_{ij}|c_k)^2$ и $\sum_j p(f_i = v_{ij})^2$ да се заменят с интеграли, отчитайки и функцията на разпределението на стойностите на атрибутите. Приемайки, че разределението на стойностите е нормално, получваме следната непрекъсната форма на тези суми:

$$\int \frac{1}{\sigma_i^2 2\pi} e^{\frac{(x-a_i)^2}{\sigma_i^2}} dx = \frac{1}{\sigma_i} \frac{1}{2\sqrt{\pi}},$$

където a_i и σ_i са съответно математическото очакване (средната стойност) и средноквадратичното отклонение на стойностите на атрибута f_i. След като се пропусне константата $\frac{1}{2\sqrt{\pi}}$ (тъй като стойностите на формулите се изпозуват само за сравнение) получаваме окончателния вид на оценъчната функция за числови атрибути с реални стойности:

$$\frac{\sum_k p(c_k) \sum_i \frac{1}{\sigma_{ik}} - \sum_i \frac{1}{\sigma_{ip}}}{k}, \tag{6}$$

където сумата по k е по категориите в съответното ниво, сумите по i са по всички атрибути, σ_{ik} е средноквадратичното отклонение на стойностите на атрибута f_i в категорията c_k и σ_{ip} е средноквадратичното отклонение на стойностите на атрибута f_i в категорията - предшественик на текущото ниво (за първото ниво след корена това са всички примери, постъпили до момента).

Формула (6) може да се разглежда като вариант на *полезността на категоризацията*. Тази форма обаче създава един проблем, когато в категориалното дърво има категории, състоящи се само от един обект. Стойността на σ_{ik} за такива категории е 0, което води до невъзможност за изчисляване на члена $\frac{1}{\sigma_{ik}}$ (безкрайност). Едно практическо решение на този проблем е изборът на минимална стойност на средноквадратичното отклонение, която се използва за категории от един пример. Тъй като тази стойност има силно влияние върху процеса на добавяне на нови категории (категории с един пример, които в зависимост от този параметър могат силно да увеличат общата оценка на категоризацията), тя се използва като параметър на алгоритъма, чрез който може да се управлява дълбочината на получаваните дървета.

Описаните по-горе разширения на COBWEB са реализирани в алгоритъма CLASSIT ([4]).

1.4 Извличане на знания от данни

Извличането на знания от данни (ИЗД) е нова и бързоразвиваща се интердисциплинарна област, силно свързана с МС. Както се отбелязва в уводната статия на първия том на списанието Data Mining and Knowledge Discovery ([1]) тази област би трябвало да възникне и да се развива като подобласт на *статистиката*. ИЗД и статистиката имат сходни цели – *самообучение от данни или превръщане на данните в информация или знания.* Както обичайно се дефинира[5] статистиката обхваща не само ИЗД но и машинното самообучение. Това важи най-вече за съвременната статистика, която изживява ренесансов период вследствие на широкото навлизани на компютрите в практиката на статистиците. Ето някои области на статистиката ([6]), които явно показват близостта на целите и задачите на статистиката, ИЗД и машинното самообучение:

- *Постановка на експерименти, вземане на извадки.* Това е класическа област, в която се изследват проблемите за избор на подходящи примери и случаи.
- *Експлораторен анализ на данни.* В тази област се изследват възможностите за генериране на хипотези.
- *Статистическа графика.* Тук се разработват методи за визуализация на данни.
- *Статистическо моделиране.* Това е област, в която се прилагат методите *регресия* (самообучение на функция, която изобразява всяка данна в реална стойност) и *класификация* (самообучение на функция, която изобразява всяка данна в една от няколко предварително зададени класове – в термините на МС, индуктивно самообучение).
- *Статистически извод.* Това е класическа област, в която се оценяват хипотези за данните и се предсказват резултати от екперименти.

Статистическият подход към изброените по-горе цели и задачи има някои специфични особености. Първо, това е *активният подход* към данните. Статистиците разработват методи за експериментиране (т.е. събиране на данни) – проблем, които е извън обсега на методите на МС и ИЗД, занимаващи се с *ретроспективен* анализ на данни. Второ, в МС и ИЗД често се отделя повече внимание на *разбираемостта* на хипотезите, отколкото на тяхната точност и предсказваща способност. Важно за МС и ИЗД е възможността за обработка на *големи количества* данни, въпрос свързан с ефективното им представяне и достъп до тях.

В този смисъл ИЗД се занимава с формализирането и реализацията на някои аспекти от статистическата обработка на данни с цел тяхното практическо използуване от широк кръг потребители без необходимост от предварителни познания върху използуваните методи. Заедно със статистическите методи се използуват и методи от разпознаване на образи,

[5] Най-често статистиката се дефинира като математическа дисциплина, която се занимава със събиране, структуриране и интерпретация на числови данни и по-специално с анализ на общи характеристики на дадена съвкупност от данни чрез изследване на отделни нейни извадки.

невронни мрежи, машинно самообучение, бази данни и др. Тъй като обикновено прилагането на тези методи е най-често върху големи количества данни, използува се термина ”дълбане в данните” (data mining). След масовото навлизане на компютърната техника в статистиката през 60-те години този термин се използува с негативен отенък от статистиците, като синоним на безцелно ровене в данните без изпозуване на строги математически подходи. С развитието на областта ИЗД, терминът ”дълбане в данните” получи и позитивен смисъл. Понастоящем всеки алгоритъм, които извлича или организира в структура накакви шаблони, намерени в данните, или установява съотвествие между модели и данни се възприема като алгоритъм от тази област. За да подчертаем позитивния смисъл на тази област по-нататък в изложението вместо ”дълбане в данните” ще използуваме термина *търсене в данните* (ТД).

Търсенето в данните е само една стъпка от целия процес на извличане на знания от данните, който е обект на изследване на областта ИЗД. За разлика от МС и споменатите по-горе статистически методи, ИЗД е сложен *интерактивен* и *итеративен* процес. Най-общо той включва следните стъпки ([2]):

1. Анализ на приложната област и наличните априорни знания с цел ясно дефиниране на *целта* на целия процес от гледна точка на потребителя.
2. Избор на *релевантни на задачата данни* – извличане на необходимите данни от наличните бази данни или правене на съответните извадки, върху които ще се прилагат методите на ТД.
3. Предварителна обработка на данните с цел отстраняване на *шума* (или събиране на информация за неговите характеристики, например ниво, атрибути, които засяга и др.) и избор на стратегия за работа с *липсващите атрибути*.
4. Редукция на *размерността* на данните. Избор на атрибути, за представяне на данните или трансформация на данните с цел намаляване на броя на обработваните атрибути. Последното може да се направи чрез избор на по-подходящ език за представяне (вижте например примера от раздел ???).
5. Избор на *метод за ТД*, който най-добре съответствува на поставената цел.
6. Избор на модел на данните и конкретен *алгоритъм за ТД*. Тази стъпка включва избор на модел (представяне на хипотезите), подходящ за избрания алоритъм или за целите на крайния потребител (например той може да се интересува повече от разбираемостта на модела, отколкото от неговата предсказваща способност) и настройка на параметрите на алгоритъма.
7. Търсене в данните (ТД). Търсене на *шаблони* или *закономерности* в данните чрез прилагане на статистически обработки и алгоритми

на МС. Резултати от тази стъпка са хипотези във вид на правила, класификационни дървета или кластерни дървета, разделящи функции и др.

8. Интерпретация на получените от предната стъпка резултати, оценка на тяхната полезност и евентуално връщане към стъпки 1 – 7. В тази стъпка могат да се използуват средства за *визуализация* на получените хипотези (шаблони, закономерности), открити в данните, както и на данните при наличието на намерените хипотези.

9. Директно *използуване* на намерените знания (например документиране) или включването им в друга система (например система за подпомагане на вземането на решения или експертна система), която ги използува. В тази стъпка се включва и проверката за *съвместимост* на получените знания със съществуващите или извлечените преди това с помощта на системи за ИЗД.

От изброените по-горе стъпки, търсенето в данните е най-силно свързано с разглежданите в тази книга методи. Разбира се тук могат да се прилагат всички възможни алгоритми от статистиката или МС. Използуваемоста на отделните алгоритми за МС в случая се определя най-вече от възможността им за работа с *големи количества данни* и натрупания опит от практическото им използуване. Ето някои от тези методи и алгоритми за търсене в данните, използувани в ИЗД:

- *Класификация.* В тази група попадат класическите индуктивни методи за генериране на правила и класификационни дървета, както и алгоритми за самообучение чрез запомняне (например алгоритъмът на най-близкия съсед). Използуват се предимно прости езици за представяне на примерите (напр. атрибут-стойност), тъй като те дават възможност за ефективна обработка на големи количества данни.

- *Регресия.* Задачата тук е сходна с тази при класификацията, като обаче атрибута, който определя принадлежността на примерите към даден клас, приема реални стойности. По този начин целта е намирането на подходящи функции, описващи данните с най-добра точност. Използуват се както класически методи (например линейни функции), така и нови методи за търсене на линейни и нелинейни комбинации от набор базови функции (например полиноми, сигмоидни функции и др.). Последните включват *невронни мрежи* и *генетични алгоритми* – области, които предлагат мощни алгоритми за търсене в сложни и с голяма размерност пространства.

- *Клатеризация.* Тук се прилагат предимно класическите статистически методи за кластеризация. Често цитирано приложение на тези методи в контекста на ИЗД е откриването на хомогенни групи от клиенти в големи бази от данни за търговски операции.

- *Резюмиране.* В тази област се търсят компактни описания на подмножества в данните. Най-простото такова описание е средната стойност и средноквадратичното отклонение за всеки атрибут.

- *Моделирне на зависимости между атрибути.* Такива модели се търсят на две нива – структурно ниво (зависимост или независимосто на атрибути, често изобравявано графично) или функционално ниво (намиране на количествена форма на зависимостта, например коефициент на корелация или конкретна функция, свързваща зависимите параметри).

- *Откриване на промени в данните.* Тук се изследват методи за намиране на значими промени в данните при сравнение с предишни или с нормативни стойности.

Разбира се областта ИЗД е отворена за нови подходи. В този смисъл всяко практическо приложение или решаване на конктретна задача с много голяма размерност с помощта на който и да алгоритъм от МС може да се разглежда като принос в областта ИЗД.

1.5 Въпроси и задачи

1.5.1 Покажете, че средното, минималното и максимално разстояние между елементите на множествата не отговарят на формалните критерии за разсдтояние (метрика).

1.5.2 Покажете, че функцията на сходство между атоми (2) не е метрика.

1.5.3 Докажете, че хаусдорфовото разстояние е метрика.

1.5.4 Как се променят кластерните дървета от фигури 1.3 и 1.4 при промяна на типа на оценката за сходство между дърветата (минимално, максимално, средно или хаусдорфово разстояние).

1.5.5 Построеете кластерно дърво (чрез езика "атрибут-стойност") и граф на обобщенията (чрез езика на предикатното смятане от първи ред) от примерите на животни (задача 11 от глава **??**).

1.5.6 Модифицирайте алгоритъма за построяване на граф на обобщенията, като на стъпка 2 използувате ентропията като оценка на новото дърво (в случая е необходимо използуването на класификация на примерите). Оценете работата на новия алгоритъм с кластеризация на животните (задача 11 от глава **??**).

1.5.7 Модифицирайте алгоритъма за построяване на граф на обобщенията, като промените условието за край. Използувайте оценка на грешката при класификация (раздел **??**).

1.5.8 Приложете алгоритъма за построяване на граф на ообщенията към положителните примерите за предиката `append` (раздел **??**).

1.5.9 Важен ли е редът на примерите, обработвани от алгоритъма COBWEB? Ще се промени ли дървото от фигура 1.6, ако променим редът на примерите.

1.5.10 Сравнете дърветата от фигури 1.3, 1.5 и 1.6, които описват различни кластеризации на едно и също множество от обекти. Обяснете разликите.

1.5.11 Използувайки алгоритъма COBWEB построете категориалното дърво на множеството от примери на животни, описани в задача 11 от глава **??**. Сравнете го с дърветата, получени от алгоритмите за агломеративна кластеризация и за построяване на граф на обобщенията. Кое от тези дървета описва най-добре истинската таксономия на животните?

Bibliography

[1] U. Fayyad. Editorial. *Data Mining and Knowledge Discovery*, 1:5–10, 1997.

[2] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery. *AI Magazine*, fall:37–54, 1996.

[3] D. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[4] J. H. Gennari, P. Langley, and D. Fisher. Models of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.

[5] M. Gluck and J. Corter. Information, uncertainty and the utility of categories. In *Proceedings of the 7-th Annual Conference of the Cognitive Science Society*, pages 283–287, Irvine, CA, 1985.

[6] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery*, 1:11–28, 1997.

[7] R. Michalski and R. Stepp. Learning from observation: conceptual clustering. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: Artificial Intelligence Approach*, volume 1, pages 331–363. Tioga, 1983.

[8] J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, 5:135–152, 1970.

[9] L. Wittgenstein. *Philosophical Investigations.* Macmillan, New York, 1953.