Лекция 4. Намаляване на обема на данни

В повечето от реални случаи данни, подлежащи на анализ, имат много голям обем, което значително усложнява работата на изследователя. Първо, обработката на такива данни заема много време, което прави самият анализ практически непригоден или дори неосъществим, и второ, това ограничава изследователя в избора на "инструменти" – алгоритмите за ИЗД, тъй като не всички те еднакво добре се справят с големи обеми данни. Методите за намаляване на обема на данни целят да бъде получено такова редуцирано представяне на данни, което е значително по-малко по своя обем, но позволява да бъдат получени от него същите (или почти същите) аналитични резултати, както и от пълния обем данни. Тези методи могат да бъдат групирани на:

1. *Методи за намаляване на размерността на данни*, които откриват и премахват нерелевантни, слабо релевантни или излишни атрибути.
2. *Методи за компресиране на данни*, които използват различни техники за кодиране, за да намалят размера на множество данни, подлежащо на по-нататъшното анализиране.
3. *Методи за моделиране и подбор на данни*, които заменят оригиналното представяне на данни с алтернативното, по-малко представяне, като, например, параметричните модели (при които вместо оригинални данни се пазят само определени параметри на модела) или непараметрични методи от типа на клъстеризация или създаване на извадки.
4. *Методи за обобщение чрез йерархии на понятия*, които заменят "сурови" първоначални стойности на атрибути със техни обобщени стойности на по-високо ниво на абстракцията.

4.1. Методи за намаляване на размерността на данни

Една база данни, подлежаща на анализ, може да съдържа стотици атрибути, много от които могат да бъдат нерелевантни или излишни за конкретната ИЗД задача. Например, ако задачата е да бъдат класифицирани потребители на една фирма за продажба на електроника от гледната точка, кои от тях най-вероятно ще поръчат новият CD плеер след като бъдат уведомени за неговото постъпване в продажба, такъв атрибут като *Телефонния_номер* на потребителя е съмнително да бъде релевантен, в отличието от атрибутите *Възраст* и *Музикалните_предпочитания*. Макар, че е възможно изборът на някои полезни за задачата атрибути да бъде направен от експерта в конкретна предметна област, решаването й е една доста сложна и много продължителна дейност, особено когато поведението на данни не е добре изучено. Изхвърлянето на някои релевантни атрибути, както и оставянето на нерелевантните може да доведе до сериозни проблеми при прилагане на избрания ИЗД алгоритъм и, следователно, до намаляване на качеството на извлечените закономерности. Освен това, допълнителният обем на данни, предизвикан от

използване на нерелевантни атрибути, може да доведе и до забавяне на ИЗД процеса.

4.1.1. Избор на подмножество от атрибути

Методите за намаляване на размерността на данни намаляват обем на използваните данни чрез премахване на нерелевантни атрибути. Обикновено, за целта се използват методи за избор на подмножеството от атрибути. Целта на *избора на подмножество от атрибути* е да бъде намерено такова минимално множество от атрибути, че полученото разпределение на класове на данни с използване само на тези атрибути е колкото възможно по-близо до разпределение на данни по класове, изчислено с използване на всички оригинални атрибути. Анализирането на намалената по този начин база данни дава и още едно допълнително предимство – получените след ИЗД процеса модели (закономерности) съдържат по-малко количество на атрибути, което ги прави по-разбираеми за потребителя.

Тази задача, известна в областта на машинно самообучение като *задача за избор на признаци* (feature selection), може да се разглежда като задача за търсене в пространство на всички възможни комбинации на атрибути (признаци) на "най-доброто" подмножество от тях. За решаването на тази задача е необходимо да бъдат избрани следните два компонента:

- *Оценъчна функция*, която ще се използва за оценяването, какво е това "най-доброто"
- *Процедура за търсене*, проверяваща всяка от възможностите

Ако изборът на атрибути е предназначен за последващото решаване на *класификационната задача*, то в качеството на оценъчната функция може да се използва истинската грешка на избрания класификатор или нейното приближение, изчислено върху някое независимо, достатъчно голямо множество от тестови примери. Подобен подход се нарича *подход на обвивката (wrapper approach)*. Подходът към избора на атрибути без използване на някой конкретен класификатор се нарича *филтърен*. Подходът на обвивката води до създаване на класификатори с по-голяма точност, тъй като той оптимизира оценъчната функция на алгоритъма при премахване на атрибути. Обаче, този метод е значително по-скъп от филтърния от изчислителната гледна точка.

При филтърния подход използването на истинската грешка на класификационния алгоритъм е невъзможно (тъй като класификаторът е неизвестен), по тази причина се използват други мерки, които непряко оценяват тази характеристика. Най-често това са определени статистически критерии от типа на проста или множествена детерминация или F-критерий (за тях по-подробно четете в лекция, посветена на регресията), оценяващи статистическата значимост на текущото подмножество от атрибути. С други думи, те оценяват доколко наличието на избрани атрибути може да обясни вариабелност (разсейване) в стойностите на целевия атрибут.

Ако броят на атрибутите е d, броят на всички възможни подмножества на атрибути е 2^d. Очевидно е, че при голямо d пълните процедури за търсене са практически неприложими от изчислителната гледна точка. По тази причина се използват някои евристични неоптимални стратегии за търсене. Това налага и използване на допълнителен *критерий за спиране*, определящ кога търсенето трябва да бъде прекратено. Могат да бъдат използвани различни критерии за спиране, като, например предварително зададен праг върху използваната евристична оценка за качеството на текущото подмножество, или изискването за монотоността на избраната оценъчна функция.

Най-разпространени евристични стратегии за избор на атрибути са следните:

- *Избор на най-добри атрибути при предположение за тяхна независимост*. При този подход се избират тези от атрибутите, които минават прост тест за статистическата значимост. Това е най-лесната, но и най-слабата стратегия, тъй като не се отчитат никакви съществуващи зависимости между атрибути.

- *Постъпков избор на най-добрия атрибут чрез добавяне (stepwise forward selection)*. При тази стратегия отначало се избира само един най-добър атрибут. След това, последователно, от останалите атрибути се избира следващия най-добър атрибут при условие, че предишния атрибут е вече избран. Тъй като при тази стратегия не се използва връщане назад за преразглеждане на вече взети решение (backtracking), очевидно е, че тя е субоптимална, обаче е по-добра от първата, тъй като частично оценява взаимодействието между атрибутите.

- *Постъпков избор на най-добрия атрибут чрез премахване (stepwise backward elimination)*. Тази стратегия започва с пълен набор на атрибути. На всяка последователна стъпка тя премахва най-лошия атрибут от останалото подмножество.

- *Комбиниране на постъпковото добавяне и премахване*. Двете предишни стратегии могат да се комбинират по такъв начин, че на всяка стъпка процедурата избира най-добрия атрибут и премахва най-лошия от останалите.

Описаните евристични стратегии за търсене се използват като за филтърните, така и за методите на обвивката. Към филтърните методи могат да бъдат отнесени и методи, които използват за избор на атрибути други оценъчни функции, например, ентропията - при класификационни дървета (ID3 и C4.5) или Евклидово разстояние – при RELIEFF (вижте подробно описание на тези алгоритми в лекции по машинно самообучение). В едно класификационно дърво всеки нетерминален възел представлява тест върху стойности на определен атрибут, а терминалните възли представляват стойностите на целевия атрибут (класове). При построяване на дървото изборът на всеки атрибут се прави постъпково и без възврат, като в качеството на оценъчната функция се използва мярка за информационна печалба или Gini индекс (и двата критерия оценяват "класовата" еднородност на подмножества от примери, получавани при разбиването на цялото множество от

примери по стойностите на избрания атрибут) - т.е. се прилага стратегия 2. Атрибутите, които не присъстват в дървото, се разглеждат като нерелевантни. Описаният метод за избор на атрибути чрез построяване на класификационното дърво показа добри резултати при последващото решаване на класификационни задачи с такива класификатори като невронни мрежи и метод на най-близкия съсед.

RELIEFF използва за избор на атрибути мярката, базирана върху изчисляване на разстояние между обекти (класифицирани примери). Той опитва да оцени атрибути от гледната точка, доколко всеки от тях, независимо един от друг, добре разделя съседни обекти от различни класове – т.е. прилага стратегия 1. Методът оценява всеки атрибут по скалата [-1, +1] (-1 – абсолютно нерелевантен; +1 – абсолютно релевантен), изчислена на базата на точността на използвания класификатор върху обучаващото множество от данни. Като нерелевантните се разглеждат всички атрибути с оценката под зададено прагово ниво. Методът показва много добри резултати с почти всички класификатори, обаче е трудно приложим за много големи (по размер) бази данни.

Макар, че всички методи за евристичния избор на атрибути се базират на много слаба теория, тяхното прилагане може в значителна степен да подобри последващата работа на различни класификатори.

4.2. Компресия на данни

Методите за компресията на данни прилагат кодиране или трансформация на данни, за да получат намалено или "компресирано" (сгъстено) представяне на оригиналните данни. Ако оригиналните данни могат да бъдат възстановени (реконструирани) от компресираните без никаква загуба на информацията, то такива техники за компресиране се наричат *съхраняващи (lossless)*. Ако само апроксимацията на оригинални данни може да бъде реконструирана, то такива техники се наричат *губещи (lossy)*. Съществуват няколко съхраняващи техники за компресия на символните низове, обаче те позволяват само доста ограничен кръг операции върху компресираните данни. Ние ще разгледаме накратко четири популярни и ефективни губещи метода за компресията на данни – дискретна вълнова трансформация, анализът на основни компоненти (principal component analysis – PCA), метод за локалното линейно вграждане и алгоритъм Fastmap.

4.2.1. Дискретни вълнови трансформации

Дискретната вълнова трансформация (discrete wavelet transformation – DWT) е една линейна техника от областта на обработка на сигналите, която се прилага към някой вектор от данни *D* и го трансформира в друг вектор *D'* със същата дължина, състоящ от вълнови коефициенти. Полезността на DWT се състои в това, че преобразуваните данни могат да бъдат отрязани. Компресираното приближение на оригиналните данни може да бъде запазено само чрез съхраняване на една малка част от най-голями вълнови коефициенти. Например, запазват се само

коефициенти, над някой предварително зададен праг - останалите коефициенти се установяват в нула. Полученото представяне на данни е много разпръснато, така че специални операции в такова пространство са много бързи и ефективни от изчислителната гледна точка. Апроксимацията на оригиналните данни може да се получи чрез прилагане на обратна DWT.

Общата процедура по прилагане на дискретна вълнова трансформация използва така наречения *пирамидален алгоритъм*, който на всяка итерация разполовява данните, водейки до бързи изчисления. Методът се състои в следното:

1. Дължината L на входен вектор данни трябва да е цяло число степен на 2. Това условие се постига с добавяне на необходимото число нули.
2. Всяка трансформация включва прилагане на две функции. Първата прави определено изглаждане на данни, от типа на сумиране или претеглено осредняване. Втората изпълнява претегляна разлика, изтъквайки по-детайлни характеристики на данни.
3. Тези две функции се прилагат към двойки от входни данни, произвеждайки като резултат два множества с размер L/2 всеки. В общия случай те представляват една изгладена, низко честотна версия на входни данни и една високо честотна версия на тях.
4. Функциите се прилагат рекурсивно към данни, получени на предишната стъпка, докато не бъдат получени множества с размер 2.
5. Изборът на стойностите от тези множества данни, получени на описаните итерации, конструира вълнови коефициенти на трансформираните данни.

Вълновите трансформации дават много добри резултати на разпръснати и несиметрично разпределени непрекъснати данни. Те се използват в такива приложения, като компресията на пръстови отпечатъци, компютърното зрение, анализ на времеви серии и изчистването на данни.

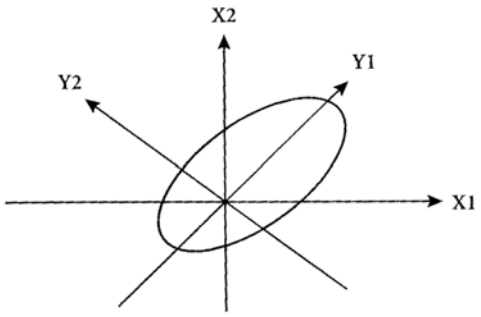
4.2.2. Анализ на основни компоненти

Да предположим, че оригиналната база данни съдържа N записа (вектора на данни) с размерност n (т.е. с n атрибута). Методът на анализа на основни компоненти (АОК), наричан още *трансформацията на Karhunen-Loeve или K-L метод*, търси оптимален начин да отобрази тези n-мерни вектори в k-мерни вектори, където $k \leq n$. Под оптималността се разбира, че методът минимизира средна квадратична грешка, където грешката е (Евклидово) разстояние между всяка n-мерна точка и нейния k-мерния образ. Основната идея на метода се състои в търсенето на k n-мерни ортогонални вектора, които се използват за оптималното представяне на оригинални данни. При $k < n$ методът води до намаляване на размерността и, следователно, до компресията на данни. Обаче, в отличие от методите за избор на атрибути, АОК "комбинира" най-важното от оригиналните атрибути чрез създаване (линейната комбинация) на едно алтернативно по-малко множество от

атрибути. Образът на оригиналните данни се получава чрез проектирането на тези данни върху намаленото множество от нови атрибути.

В основа на метода лежи следната процедура:

1. Входните данни се нормализират, така че стойностите на всеки атрибут попадат в един и същ диапазон. По този начин атрибутите с по-голям диапазон на стойностите няма да доминират атрибутите с по-малък диапазон.
2. Изчисляват се *k* ортонормални n-мерни вектора, които служат като базис за нормализираните входни данни. Това са единични вектори, всеки от които е перпендикулярен на всички други базисни вектори. Именно тези базисни вектори се наричат *основни компоненти*. Входните данни са линейната комбинация от основните компоненти.
3. Основните компоненти се сортират в намаляващ ред по своята "важност" или сила, която се измерва със степента на разсейването на оригиналните данни при проекцията върху съответния основен компонент. Т.е. първият основен компонент е тази нова координатна ос в n-мерното пространство, при проектирането върху която се наблюдава най-голямата дисперсията (разсейването) на оригинални данни. Втората ос показва следващото по големина разсейване и т.н. Например, на фиг. 4-1 са показани първите два основни компонента Y1 и Y2 за данни, първоначално представени чрез атрибути (оси) X1 и X2. Тази трансформация помага да бъдат намерени групи или шаблони в данни.



Фиг. 4.1. Анализ на основните компоненти – Y1 и Y2 са първите основни компоненти за зададени данни

4. Тъй като компонентите са сортирани в намаляващ ред съгласно своята "значимост", размерът на данните може да бъде намален чрез елиминиране на по-незначителни от тях, т.е. тези с по малко разсейване (вариабилност на данни върху тях). Чрез използване на основните компоненти е възможно да бъде реконструирана една добра апроксимация на оригиналните данни.

Методът за анализа на основните компоненти често се използва за разпознаване на образи. От изчислителната гледна точка той не е скъп, тъй като води своето начало от матричната алгебра. На практика, основните компоненти са собствените вектори на матрицата на ковариацията на оригиналните атрибути и операциите по тяхното изчисляване се свеждат до традиционните алгоритми за умножаване на матрици и намирането на обратната матрица. Обаче, методът е доста бавен за данни с голяма размерност. Например, той е трудно приложим за задачите за търсене или филтриране на информация, където документите се описват чрез речници, съдържащи десетки хиляди думи (атрибута).

4.2.3. Локално линейно вграждане

Локалното линейно вграждане (Locally linear Embedding – LLE) е една техника за намаляване на размерността, която се базира на идеята за анализиране на препокриващи се локални области с цел намиране на локалната структура в данни. LLE може да се представи по следния начин:

Алгоритъм LLE

1. Намери най-близките съседи за всяка точка данни
2. Изрази всяка точка $\mathbf{x_i}$ като линейна комбинация от други точки, т.е. $\mathbf{x}_i = \sum_j w_{ij}\mathbf{x}_j$, където $\sum_j w_{ij} = 1$ и $w_{ij} = 0$, когато $\mathbf{x}_j$ не е най-близкия съсед на $\mathbf{x}_i$.
3. Намери координати на всяка точка в пространството с по-малка размерност p чрез използване на теглата, намерени на стъпка 2.

На стъпка 2 матрицата на теглата **W**, чиито елементи са w_{ij}, се намира чрез минимизация на средно-квадратичната грешка, представена от следното уравнение: $Error(\mathbf{W}) = \sum_i (\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j)^2$ - това е една добре известна задача – задача на най-малките квадрати, решението на което дава линейна алгебра.

Истинското намаление на размерността става на Стъпка 3. Знаейки тегловната матрица и броя на размерностите p, задавани от потребителя, алгоритмът конструира „ запазващо локални области вграждане" на данни в по-низко размерно пространство. Ако $\mathbf{y}_i$ е векторът от по-низко размерното пространство, съответстващ на $\mathbf{x}_i$, а **Y** е новата матрица на данни, в която i-я ред е $\mathbf{y}_i$, то това може да бъде направено чрез намирането на **Y**, което минимизира следващото уравнение: $Error(\mathbf{Y}) = \sum_i (\mathbf{y}_i - \sum_j w_{ij}\mathbf{y}_j)^2$

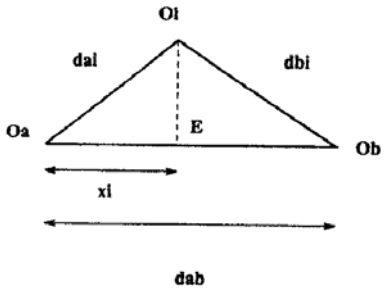
4.2.4. Алгоритъм FastMap

Многоразмерното скалиране (Multidemensional scaling – MDS) е една от техниките, която често се използва за намаляване на размерността. Съществуват различни вариации на тази техника, но всички те използват една и съща стратегия: търси се проекцията на данни в едно по-низко размерно пространство, която запазва в най-голяма степен разстояния между двойки точки в съответствие с избраната функция за измерване на разстояния. По тази причина MDS работи с матрицата на различията (разстояния) и по този начин може да се използва дори за данни, които в оригиналния си вид няма представяне във вида на вектори (например символни низове). Ще разгледаме един от най-ефективни методи от тази област - FastMap, който има две основни разлики от традиционните MDS алгоритми – той е значително по-бърз и има инкрементален характер.

Алгоритъм FastMap (Faloutsos & Lin 1995) е разработен за целите на бързото търсене в мултимедийни бази данни, както и за визуализацията на многомерни данни. Алгоритмът решава задача за проектиране на N обекта, за които е известна N x N матрицата на взаимните разстояния (или просто мярка за разстояние между два обекта $D(*,*)$), в N точки от k-мерно пространство по начина, запазващ, до голяма степен, съответствията в разстояния между обектите. Решаваната до сега задача за проектиране на N n-мерни обекта в N k-мерни обекта ($k \leq n$) е частен случай на тази по-обща задача.

При условие, че вече изчислени взаимни разстояния между обекти в оригинално n-мерно пространство (оригиналната версия на алгоритъма използва Евклидово разстояние), намирането на проекциите на обекти в редуцираното k-мерно пространство се осъществява рекурсивно за k стъпки. На всяка стъпка се осъществяват следните действия:

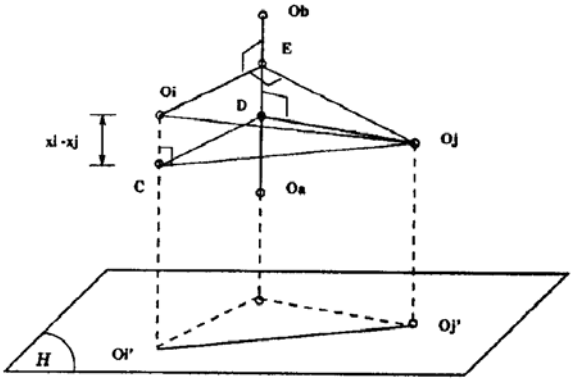
1. Избират се два *опорни (pivot) обекта* O_a и O_b. Линията, която минава през тях, се разглежда като първата ос на търсеното k-мерно пространство.
2. За всеки обект O_i неговата координата x_i по тази ос се изчислява по следната формула:
   $$x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2d_{a,b}} \quad (4.1)$$
   където $d_{a,b}$ е разстояние между O_a и O_b, $d_{a,i}$ и $d_{b,i}$ са разстояния между O_i и O_a и O_b, съответно, които са вече изчислени (предварително или на предишната стъпка от рекурсията).



3. За да бъдат изчислени координати на обекти по следващата (в примера - втора) ос необходимо е предварително да бъдат преизчислени взаимните разстояния между всички обекти в намаленото (в случая *n-1*-мерно) пространство. Тези разстояния се изчисляват на базата на разстояния в текущото пространство (в примера n-мерното) и координати на обектите върху предишната (в случая – първата) ос по формула:

   $$(d'_{i,j})^2 = (d_{i,j})^2 - (x_i - x_j)^2,\ i, j = 1,..., N \quad (4.2)$$

където $d'_{i,j}$ е разстояние между обекти O_i и O_j в редуцираното пространство, $d_{i,j}$ е разстояние между обекти O_i и O_j в оригиналното (от предишната стъпка) пространство, а x_i и x_j са координати на съответните обекти върху първата (предишната) ос.



Описаните три точки се повтарят рекурсивно до като не бъдат изчислени координати на всички обекти върху k-та координатна ос.

Опорните обекти O_a и O_b се избират по такъв начин, че да максимизират разстояние $D(O_a, O_b)$. За да намали броя на необходими за тази цел изчисления, които са $O(N^2)$, авторите използват следния евристичен алгоритъм със сложността $O(N)$:

Алгоритъм избери_обекти(O, D())

Вход:

Множество O от N обекта
Функция за разстояние $D()$

Начало

1. Избира се произволен обект, които се приема за втория опорен обект O_b.
2. Като първият опорен обект O_a се избира обектът, който се намира на най-голямо разстояние от O_b съгласно избраната мярка за разстояние.
3. В качеството на нов втори опорен обект O_b се избира обект, който се намира на най-голямо разстояние от O_a.
4. Стъпките 2 и 3 се повтарят зададено количество итерации и получените накрая обекти се използват като опорни.

Край

В оригиналния алгоритъм брой на итерации за избор на опорните обекти са 5.

И така, алгоритъм FastMap може да се опише по следния начин:

Алгоритъм FastMap(k, D(), O)

Вход:

Множество *O* от *N* обекта
Функция за разстояние *D()*
Желания брой на размерности *k* на целевото пространство

Начало

Глобални променливи:

X[] – *N x k* масив – в края на работата на алгоритъма i-ят ред на масива ще съдържа k-мерния образ на i-я обекта

PA[] - *2 x k* масив – съхранява идентификатори на опорните обекти – по една двойка за всяко рекурсивно извикване.

int j = 0 – указател на текущата колонка в X[]

1) ***АКО*** $k \leq 0$ ***Край***
   ИНАЧЕ j = j + 1
2) /* Избор на опорни обекти */
   Нека O_a и O_b са резултат от работата на алгоритъма *избери_обекти(O, D())* по избор на опорни обекти, описан по-горе;
3) /* Записване на идентификатори на опорните обекти */
   PA[1, j] = a; **PA**[2, j] = b;
4) ***АКО*** D(O_a,O_b) *= 0*
   X[i, j] = 0 за всички i и ***Край*** /* всички разстояния между обекти са 0 */
5) /* проектиране на обекти върху линията (O_a,O_b) */
   За всеки обект Oi
   Изчисли x_i съгласно ф-ла (4.1) и обнови глобалния масив:
   X[i, j] = x_i
6) /* проектиране на обекти върху хипер-равнина, перпендикулярна на линията (O_a,O_b); функцията за разстояние *D'()* между двете проекции се задава с ф-ла (4.2)
   FastMap(k – 1, D'(), O)

Край

При всяко рекурсивно извикване алгоритмът определя координати на N обекта върху новата ос. Така, че i-ят обект се отобразява в точка P_i = (**X**[i, 1], **X**[i, 2], …, **X**[i, k]).

Алгоритъм FastMap е много удобен за визуализация и клъстерен анализ на многомерни данни. В повечето случаи оригиналните данни се проектират в 2-мерното или 3-мерното пространство, които позволяват лесна и нагледна визуализация.

4.3. Методи за моделиране и подбор на данни

Методите за моделиране и подбор на данни заменят оригиналното представяне на данни с алтернативното, по-малко представяне. Те могат да бъдат разбити на две големи групи – *параметрични методи*, заменящи оригинални данни с техните модели, при които се пазят само параметри на модела, и непараметрични методи от типа на *клъстеризация* или *подбор на данни*. Тъй като повечето от параметрични методи, както и методите за клъстеризация, ще бъдат разгледани в отделни лекции, в настоящия раздел само ще се запознаем с техники за намаляване на обема на данни чрез техния подбор.

Подбор на данни (sampling) може да използва като техниката за намаляване на обема на данни, тъй като позволява едно голямо множество от данни да бъде заменено със значително по-малка случайна извадка (или подмножество) от данни. Да предположим, че една голяма база данни D съдържа N записи (обекта). Най-често използваните методи за намаляване на данни чрез подбор са:

- *Прост случаен подбор на извадка с размер n без връщане*. Този тип извадка се създава чрез случаен избор на n от N обекта от D $(n < N)$, като вероятността за избор на един произволен обект е $1/N$, т.е. всички обекти в базата са равновероятни. Обектът, веднъж попаднал в извадката, не се връща в базата данни.
- *Прост случаен подбор на извадка с размер n с връщане*. Този тип извадка се създава по начин, сходен с описаният по-горе, само с това изключение, че след всеки случаен избор на нов обект и поместването му в извадката, копията на обекта се връща обратно в база данни D. По този начин получената извадка може да съдържа няколко копия на един и същ обект.
- *Клъстерен (групов) случаен подбор*. Този метод представлява случаен подбор на клъстери (групи) от обекти, които се включват в извадката. Ако обектите в D са групирани в M взаимно непресичащи се "клъстери", то е възможно да бъде създадена случайна извадка от m клъстера $(m < M)$. Например, записите в една база от данни обикновено се търсят и се показват по страници, така че всяка страница може да се разглежда като клъстер. Намаленото по обем представяне на такава база може да се направи чрез прилагането на техниката за създаване на случайна извадка без връщане, прилагана към страниците, създавайки клъстерната извадка. В клъстерната извадка могат да бъдат включени както всички обекти от случайно избрани клъстери, така и само някои случайно избрани обекти от всеки клъстер.
- *Стратифициран (райониран) случаен подбор*. Ако D е разделена на взаимно непресичащи се части, наречени *страта (райони)*, то една стратифицирана

извадка от *D* се получава чрез прилагането на техниката за случайната извадка без връщане към всеки район *(стратум*). Това позволява да осигури една действително представителната извадка, особено в случаи, когато разпределението на данни е силно деформирано. Например, стратифицираната извадка може да бъде получена от данни за клиентите на една фирма, където за всяка възрастова група се създава по един стратум. По този начин в стратифицираната извадка ще бъдат представени всички възрастови групи, дори тази, с най-малкия брой клиенти.

Едно от важните предимства за използването на извадки като методи за намаляване на обема на данни, е че изчислителната сложност за създаването на една извадка е пропорционална на нейния размер *n*, а не на размера *N* на цялата база данни. Следователно, сложността на такива методи е сублинейна по отношение на размера на данни. Всички други техники за намаляване на обема на данни изискват най-малко едно пълно минаване през цялата база *D*. При фиксиран размер на извадката, изчислителната сложност на методите за създаване на извадките нараства само линейно при увеличаване на размерността (броя на атрибутите) на базата данни.

Използването на извадки е най-разпространеният метод за анализ на данни. Чрез използване на централната пределна теорема (от теорията на вероятностите) е възможно да бъде определен такъв минимален размер на извадката, който ще бъде достатъчен за оценяване на дадената функция със зададен интервал на грешката при оценяването. Този размер на извадката може да бъде съществено по-малък от размера на цялата база. Освен това създаването на извадки е един естествен избор за последователното уточняване на редуцираното множество от данни. Подобното множество може да бъде доуточнявано просто чрез увеличаване на размера на извадките.

4.4. Методи за обобщение чрез понятийни йерархии

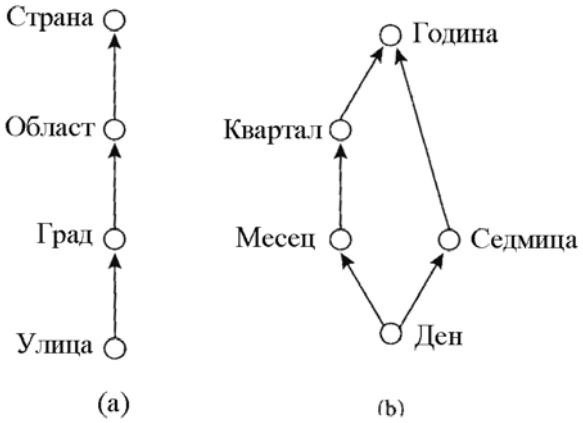
4.4.1. Понятийни йерархии

Понятийната йерархия дефинира последователността от отображения на едно множество от понятия от по-долното ниво на абстракцията върху по-общи понятия от по-високи нива. Да разгледаме, например, една възможна понятийна йерархия за размерността *Местоположение* Значенията на тази размерност могат да бъдат, например, имената на градове – София, Варна, Белград, Берлин, Отава, Вашингтон. Обаче всеки град може да бъде отобразен, например, в страната, към която той принадлежи, а страната – към континент. В този пример използваме за размерност *Местоположение* понятийната йерархия *Град* < *Страна* < *Континент*, която позволява да разглеждаме една и съща база данни на различни нива на абстракцията. Обърнете внимание, че при всяко преминаване на по-горното ниво от йерархията се намалява брой на възможни стойности на съответния атрибут, което води до ускоряване на работата на различни ИЗД алгоритми (например класификационни дървета). Освен това, при този процес могат да възникват

множество еднакви (дублиращи се) записи, които се намират и се отстраняват в процеса на подготовката на данни. По този начин се намалява и обемът на самата база данни.

Много от понятийните йерархии са заложени неявно в самата схема на база данни. Например, същата размерност *Местоположение* може да бъде представена в една база данни като таблица, описана с такива атрибути, като *Улица, Град, Област, Страна*. Тези атрибути са свързани по-между си с пълна наредба, образуваща следната понятийна йерархия: *Улица* < *Град* < *Област* < *Страна*. (виж фиг. 4-1(а)) Атрибутите на една понятийна йерархия могат да бъдат организирани и в частична наредба, образувайки *решетка (lattice)*. Например, една понятийна йерархия на размерност *Време* може да се представи като частична наредба на такива атрибути като *Ден, Седмица, Месец, Квартал, Година*: *Ден <{Месец < Квартал; Седмица}* < *Година* (виж фиг. 4-1 (в))[1].

Понятийната йерархия, образуваща пълна или частична наредба между атрибутите в схемата на база данни се нарича *схемната йерархия*. Схемната йерархия може да се използва от изследователя за конструиране на плоска таблица от данни, избирайки за съответната размерност необходимото ниво на абстракцията, позволяващо намиране на интересни закономерности.



Фиг. 4-1. а) Йерархия на размерност Местоположение; b) Решетка на размерност Време

4.4.2. Създаване на понятийни йерархии за непрекъснати атрибути

Понятийната йерархия за някой *непрекъснат атрибут* определя неговата дискретизация на различни нива. "Ръчно" определяне на понятийните йерархии за

[1] Тъй като *Седмица* пресича границата на два последователни месеца, тя е, обикновено, не се третира като по-низката абстракция на *Месец*. Вместо това, тя често се разглежда като една по-низка абстракция на *Година*, тъй като една година съдържа приблизително 52 седмици.

непрекъснати атрибути е трудоемкият процес поради широкото разнообразие на възможни диапазони на данни и често обновяване на атрибутните стойности. Освен това, подобни "ръчни" йерархия могат да бъдат много субективни.

Понятийните йерархии за непрекъснати атрибути могат да се създават автоматично на базата на анализа на разпределение на данни. В предишната лекция, посветена на методите за предварителната подготовка на данни, ние сме разгледани някои методи за изглаждане и дискретизация на непрекъснати атрибути. Повечето от тях могат да бъдат използвани итеративно за създаване на понятийни йерархии за непрекъснати атрибути. Например, една такава дискретизационна йерархия може да се получи чрез разбиване на числовият диапазон на атрибута на зададеното множество от интервали с еднаква ширина, изглаждането им със средната или медианната стойност и рекурсивното повтаряне на тази процедура с цел пораждане на понятийната йерархия.

Дискретизационната йерархия може лесно да се получи чрез прилагането на агломеративната клъстерна дискретизация и дискретизация, базирана на ентропия, подробно описани в предишната лекция. За тази цел е необходимо само да се пазят всички междинни дискретизационни интервали, получавани в процеса на рекурсивната работа на тези методи.

Макар, че използването на подобни методи за автоматично пораждане на понятийни (дискретизационни) йерархии за непрекъснати атрибути е много полезно за последващата работа на различни ИЗД алгоритми, те не са достатъчно "естествени" за човека-потребител. Например, разбиването на диапазона на годишната заплата на интервали от типа на (50 000$, 60 000] е често много по-желателно, от колко на интервали от типа на (51 263.98, 60 872.34], които са получени в резултат от работата на сложни дискретизационни алгоритми. За подобно "естествено" сегментиране на числови данни в относително еднообразни интервали, може да се използва, така наречено *правило 3-4-5*.

Правилото 3-4-5 разделя данни от зададения диапазон на 3, 4 или 5 относително равни по ширина интервали, извършвайки разделяне рекурсивно ниво по ниво, като се базира върху диапазона на най-значимата цифра. Правилото има следния вид:

- Ако интервалът покрива 3, 6, 7 или 9 различни стойности на най-значимата цифра, то интервалът трябва да се разбие на 3 под-интервала (3 под-интервала с еднаква ширина за 3, 6, 9 и 3 интервала с групиране 2-3-2 за 7).
- Ако интервалът покрива 2, 4 или 8 различни стойности на най-значимата цифра, то интервалът трябва да се разбие на 4 под-интервала с еднаква ширина.
- Ако интервалът покрива 1, 5 или 10 различни стойности на най-значимата цифра, то интервалът трябва да се разбие на 5 под-интервала с еднаква ширина.

Това правило се прилага рекурсивно към всеки интервал, създавайки понятийната йерархия за дадения непрекъснат атрибут. Тъй като в изследваното множество от

данни е възможно наличието на много големи положителни или отрицателни стойности, той най-горното ниво йерархията, създадено само на базата на тези максимални и минимални стойности, може да доведе до получаване на изкривени резултати. Така заплатите на няколко човека могат на няколко порядъка да превишават заплатите на останалите хора от базата данни. Разделянето, базирано на такива максимални стойности, може да доведе до силно изкривена йерархия. По тази причина най-горното ниво на разделянето трябва да се прави на базата на диапазона на стойностите, представляващи мнозинство от обекти (например, от 5-ти до 95-ти процентил) от базата. Много големи или малки стойности, намиращи се извън това най-горното разбиване, ще формират отделни интервали, които се обработват по-отделно, но по същия начин. Ще разгледаме приложението на правилото 3-4-5 на следващия пример.

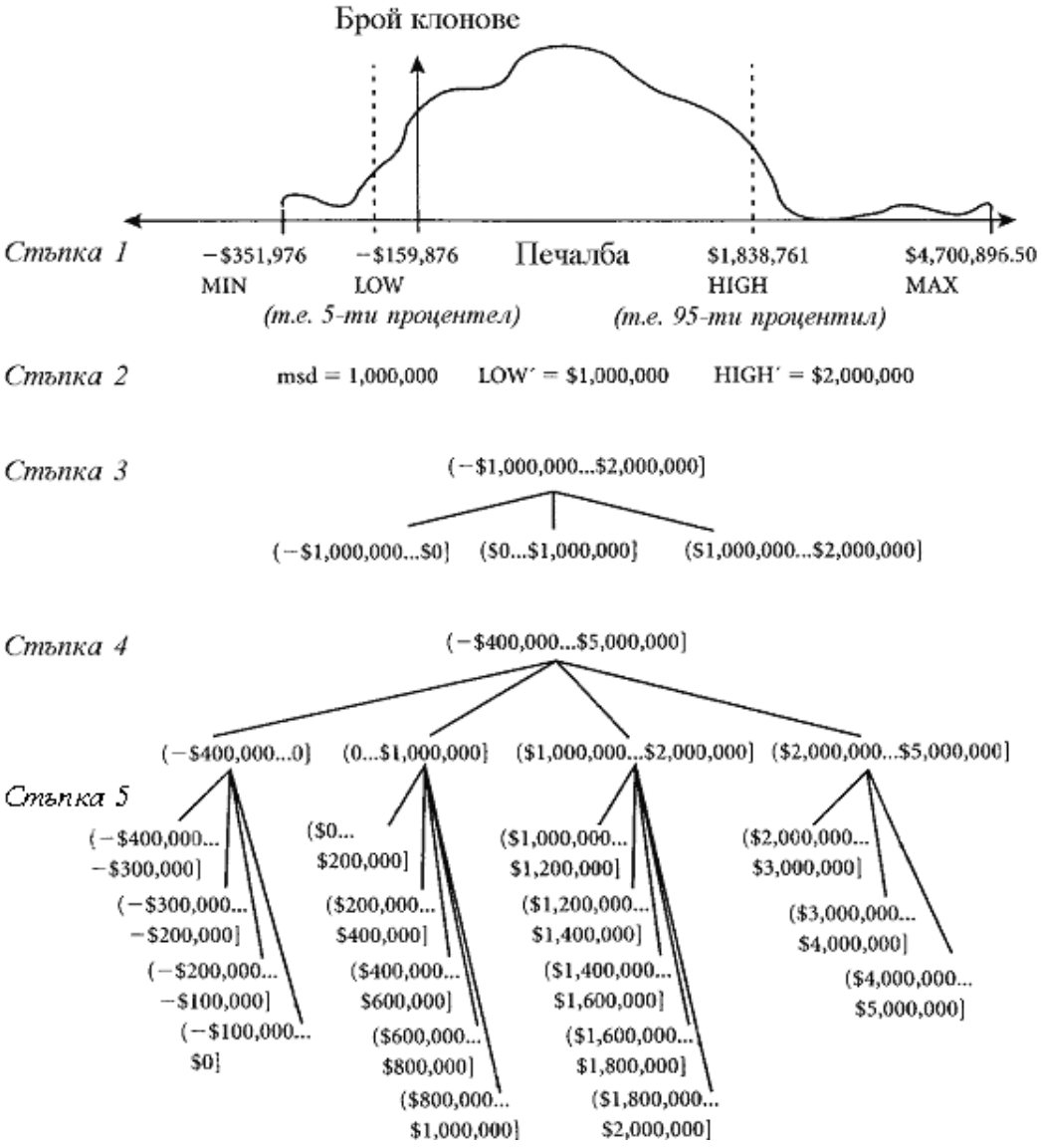
Пример 4.1. Да предположим, че печалбата на различни клонове на една фирма за 2003 г. варира от –351976.00$ до 4700896.50$. Потребителят иска да бъде автоматично създадена понятийната йерархия за атрибута *Печалба* на база на наличните данни.

Да предположим, че данни между 5-ти и 95-ти процентил се променят в диапазона между – 159 876$ и 1 838 761$. Резултатите от прилагане на правилото 3-4-5 са показани на фиг.4-2.

.

1. Базирайки се на описаната по-горе информация, минималната и максималната стойности са MIN = -351976.00$ и MAX = 4700896.50. Долната (5-ти процентил) и горната (95-ти процентил) стойностите ще се използват за най-горното ниво на сегментация, т.е. LOW = -159 876$; HIGH = 1 836 761$.Анализирайки стойности LOW и HIGH, виждаме, че най-значимата цифра (msd) е тази в разреда за милиони долари (т.е. msd = 1000000$). След закръгляване на LOW надолу към милиони долари, получаваме LOW’ = - 1000000$; закръглявайки HIGH нагоре към милиони долари получаваме HIGH’ = + 2000000$.
2. Тъй като този интервал съдържа три различни стойности на най-значимата цифра (2000000 – (- 1000000))/ 1000000 = 3, то съгласно правилото 3-4-5 този интервал се разбива 3 еднакви по ширина под-интервала: (-1000000$ … 0$], (0$ … 1000000$] и (1000000$ … 2000000$]. Това разбиване представя горното ниво на понятийната йерархията.
3. Сега да проанализираме екстремалните стойности MIN и MAX, за да видим, доколко добре те “пасват” на горното ниво на йерархията. Тъй като първия интервал (- 1000000$ … 0$] покрива стойността MIN, т.е. LOW’ < MIN, ние може да уточним лявата граница на този интервал, за да го направим по-малък. Най-значимата цифра на MIN е цифрата в позиция за стотици хиляди. Закръглявайки MIN надолу до стотици хиляди, получаваме: MIN’ = -400000$. Следователно, първия интервал става (-400 000$ … 0$].

   Тъй като последният интервал (1000000$ … 2000000$] не покрива стойността MAX, т.е. MAX > HIGH’, ние трябва да създадем новия интервал, покриващ тази стойност. Закръглявайки MAX нагоре в най-значимата му цифра, получаваме новият интервал (2000000$ … 5000000$].
4. Рекурсивно всеки интервал може по-нататък да се разбие на по-долните нива съгласно правило 3-4-5 (виж фиг. 4.2).



Фиг. 4.2. Автоматично създаване на понятийната йерархия за Печалба чрез правилото 3-4-5