Лекция 13. Самообучение чрез запомняне – оценяване и избор на атрибути и екземпляри

13.1. "Проклятието на размерността"

Изразът „проклятието на размерността" беше за пръв използван в 1961 г. от американски математик Ричард Белман за обяснение на факта, че много от алгоритмите, които работят добре с данни с малка размерност, драстично влошават своето поведение при данни с голяма размерност. В областта на машинно самообучение този израз натоварен с още по-голям смисъл. Първо, способността на индуктивните алгоритми да правят коректни обобщения извън обучаващите данни експоненциално се влошава при нарастване на размерността на данни (броя на използвани атрибути), тъй като обучаващото множество с фиксиран брой примери започва да покрива все по-малка област от пространството на възможни примери. Например, дори при един сравнително умерен брой на двоични атрибути (100), използвани за описание на примери, и един огромен брой на обучаващите примери (1 милиард), това обучаващо множество заема само 10^{-18} част от пространството на възможни примери (с размер 2^{100}).

Освен това, размерността оказва огромно влияние на алгоритмите, базирани на сходство (алгоритмите на k най-близки съседи и т.н.). Да разгледаме, например, алгоритъма на най-близкия съсед, използващ в качеството на метрика разстоянието по Хеминг, и да предположим, че истинският клас на примера, описан със 100 двоични атрибута, се определя от логическото И на първите два атрибута x_1 и x_2. Очевидно е, че при наличието на останалите 98 нерелевантни атрибута $x_3 \ldots x_{100}$, техния шум напълно ще заглуши релевантния сигнал от първите два атрибута и алгоритмът на най-близкия съсед ще прави, на практика, абсолютно случайни предсказвания.

Още по-обезпокоително е, че алгоритмът на най-близкия съсед ще има проблеми с класифицирането дори, ако всички 100 атрибута са напълно релевантни! Причината за това е, че при голям брой размерности всички примери изглеждат еднакво сходни. Да предположим, например, че примерите са разположени равномерно във възли на една пространствена решетка и да разгледаме някакъв тестов пример $\mathbf{x}_t$. Ако решетката е d-мерна, то $\mathbf{x}_t$ ще има $2d$ най-близко разположени примера (съседа), като всеки от тях ще се намира на еднакво разстояние от $\mathbf{x}_t$. При нарастване на размерността, все повече и повече примери стават най-близките съседи на $\mathbf{x}_t$, така че изборът на най-близкия съсед (а следователно и класа на тестовия пример) става на практика случаен.

Описаният проблем е само един пример на по-общия проблем – с нарастване на размерността най-близките съседи стават разположени изобщо не е наблизо! Да разгледаме, като пример, k най-близки съседи в едно множество от N точки,

разположени равномерно във вътрешността на n-мерния хиперкуб със страна 1. Ще дефинираме k-околност на една точка като най-малкия хиперкуб, който съдържа k най-близки съседи на тази точка. Нека l е средната дължина на страна на такъв хиперкуб. Тогава обемът на хиперкуба (който съдържа k точки) е равен на l^n, а обемът на цялото пространство (което съдържа N точки) е 1. От тук следва, че $l^n = k/N$. Взимайки съответния корен и от двете страни получаваме, че $l=(k/N)^{1/n}$.

Нека, за да бъдем по-конкретни, да предположим, че $k = 10$, а $N = 1000000$. При две размерности ($n = 2$) средният размерът на k-околността е $l = 0.003$ – една много малка част от размера на цялото пространство (квадрат със страна равна на 1). При 3 размерности l ще е равна само на 3% от дължината на страна на единичния куб. Обаче при 17 размерности l става равна на половината от дължината на единичния хиперкуб, а при 200 размерности – дължината на страната на хиперкуба, съдържащ k-най-близки съседа на една точка, става равна на 94% от страната на хиперкуба, съдържащ всички точки в пространството! С други думи се оказва, че при голяма размерност почти всички точки са разположени близо до повърхността на хиперкуба. Така, например, при 200 размерности около 98% от всички точки са разположени в една обвивка, близка до повърхността на единичния хиперкуб, с дебелина от 1% от страната на този хиперкуб.

И така, построяването на някой класификатор, работещ в дву или тримерното пространство е лесна задача – ние можем да намерим една разумна граница, разделяща примери от различни класове, само чрез визуалната проверка на данните (сред изследователите дори битува една шега, че ако хората би били способни да видят в пространства с много размерности, от машинното самообучение изобщо би нямало никаква нужда). Обаче при голяма размерност е трудно да се разбере, какво става, а това затруднява построяване на добри класификатори. Наивно е да се мисли, че използване на повече атрибути никога няма да влоши класификация, тъй като те предоставят повече информация за класа. На практика, това предимство може да бъде превърнато в недостатък от проклятието на размерността.

За щастие, положението не е толкова безизходно – съществува един ефект, който частично противодейства на проклятието – той може да бъде наречен "благословия на нееднородността" (Domingos 2011). В повечето от приложения обучаващите примерите не са разпределени еднородно в пространството от примери, а са концентрирани във или около групи от примери в пространства с по-малка размерност. По тази причина за построявана на ефективни класификатори се използват определени алгоритми, които явно намаляват размерността на пространството на примери. В днешната лекция ще разгледаме някои от тях.

13.2. Определяне на релевантни атрибути

Един от основните недостатъци на методите за самообучение чрез запомняне (както и на всички останали емпирични методи за самообучение от примери) е тяхната чувствителност към наличието на нерелевантни атрибути в описанието на

примерите. Приносът на такива атрибути при изчисляване на глобалното разстояние между примери представлява шум за задачата на класификацията, който може да "заглуши" релевантните компоненти. Изследователите в областта на машинното самообучение са разработили множество различни методи за преодоляване на този недостатък, които могат да се разделят на две групи. Първата са т. нар. *избиращи признаци* (feature selection) алгоритми, които избират за релевантни само *определено подмножество* от всички атрибути на примера (а останалите изобщо не се отчитат). Втората група - *претеглящи признаци* (feature weighting) алгоритми - използва *всички* атрибути, обаче преценява по различен начин приноса на всеки от тях за общата класификация на примера чрез различни *тегла* на различните атрибути. Тъй като първата група методи може да се разглежда като определено подмножество от алгоритмите на втората група, използващи само двоични тегла (0 и 1), в настоящия раздел са представени само някои по-известни алгоритми за претегляне на признаци.

Повечето от разгледаните досега методи за самообучение чрез запомняне могат да се представят като алгоритми k-NN, използващи за изчисляване на разстояние между примери функцията:

$$d(X,Y) = \left(\sum_{i=1}^{n} w_i * \delta^L(x_i, y_i) \right)^{\frac{1}{L}}$$

където $\delta(x_i,y_i)$ е избраното атрибутно разстояние, а параметърът $L \geq 1$. Тук $w_i = 1$ за всички атрибути $(i = 1,\ldots, n)$, което означава, че всички те се приемат за еднакво важни за класификацията.

В основа на всички методи за претегляне на признаци е идеята за определяне на относителната важност на всеки атрибут чрез специфичен тегловен коефициент $w_i \geq 0$[1], като се предполага, че w_i за нерелевантни признаци са близки (или равни) на нула. Тъй като повечето от алгоритмите използват нормираното разстояние, най-често $0 \leq w_i \leq 1$ и $\sum_{i=1}^{n} w_i = 1$.

Съществуващите методи за определяне на теглата на атрибутите могат да бъдат разбити на две групи — *итеративни* (online) и *статистически* (batch). Итеративните методи последователно обработват всеки обучаващ пример само веднъж, докато статистическите методи неколкократно преглеждат всички обучаващи примери.

13.2.1. Итеративни методи за оценяване на атрибути

Итеративните методи за оценяване на атрибути се базират на обратната връзка от използвания вариант на алгоритъма k-NN. Целта на методите е да се оптимизират

[1] Тегловните коефициенти не могат да бъдат отрицателни, тъй като това би довело до нарушаване на основния принцип на алгоритмите за самообучение чрез запомняне: $d(X, Y) = 0 \Leftrightarrow X = Y$.

теглата на признаците по такъв начин, че, от една страна, да бъде увеличено сходството между тестовия пример и съседните екземпляри от същия клас, а, от друга страна, да бъде намалено сходството на примера със съседните екземпляри от други класове.

Един итеративен метод е приложен от Кост и Салзберг в системата EACH. При правилната класификация на примера, ако *i*-тият признак съвпада, неговото тегло w_i се увеличава с определена стъпка Δ_i. Ако признакът не съвпада, теглото му се намалява със същата стойност. При неправилната класификация теглата на *несъвпадащите* признаци се увеличават, а на съвпадащите се намаляват.

Доказано е, че методът не работи добре при несиметрично разпределение на класовете. Този проблем е решен от Кира и Рендел в алгоритъма Relief (описанието на алгоритъма е приведено в Таблица 13-1).

Основната идея на алгоритъма е "претегляне" на атрибути в зависимост от това, доколко добре техните стойности разделят съседните примери. За тази цел за всеки пример се търсят два негови най-близки съседа — един от същия клас (той се нарича *най-близък положителен съсед),* а другият - от друг клас *(най-близък отрицателен съсед)*. Като входни параметри на алгоритъма се използват атрибутното разстояние $\delta(x_i, y_i)$, което се смята за нормирано, и общото разстояние $d(X,Y)$, свързано с атрибутното чрез формулата[2]:

$$d(X,Y) = \left(\sum_{i=1}^{n} \delta^L(x_i, y_i) \right)^{\frac{1}{L}}$$

Посочените параметри се използват за намиране на най-близките към примера съседи.

Коефициентите w_i оценяват качеството на атрибутите. В основата на използваната формула е идеята, че един "добър" атрибут трябва да има една и съща стойност за всички примери от един и същ клас (изваждането на $\delta(e_i, r_i)$) и да приема различни стойности за примери от различни класове (добавянето на $\delta(e_i,n_i)$). Нормализацията (разделянето на m) гарантира, че всички тегла ще бъдат в интервала [-1,1]. $w_i = -1$ означава, че і-тият атрибут е абсолютно нерелевантен, докато $w_i = 1$ показва, че атрибутът е много важен за успешна класификация.

Алгоритъм Relief

Вход: Обучаващи данни $D = \{X_1, \ldots, X_n\}$

Метрика $d(X,Y) = \left(\sum_{i=1}^{p} \delta^L(x_i, y_i) \right)^{\frac{1}{L}}$

Параметър m – брой на итерации

- *Инициализация:*

  За всички атрибути направи: $w_i \leftarrow 0$

[2] В оригиналния вариант на алгоритъма е използвано абсолютното разстояние (L = 1).

- *Обучение:*

За *i* от 1 до *m* направи:

1. Избери случайно пример $E = <\{e_1,\ldots,e_n\}, c_E> \in D$.
2. Намери негов най-близък положителен съсед $R = <\{r_1,\ldots,r_n\}, c_E> \in D$ и най-близък отрицателен съсед $N = <\{n_1,\ldots,n_n\}, c_N> \in D$, $c_N \neq c_E$.
3. За всички атрибути направи:

$$w_i \leftarrow w_i - \frac{\delta(e_i, r_i)}{m} + \frac{\delta(e_i, n_i)}{m}$$

Таблица 13-1. Алгоритъм Relief

Описаният алгоритъм може да работи както с непрекъснати, така и с номинални атрибути, обаче *само за два класа*. Освен това той силно зависи от шума и не може да обработва непълни данни. Разширението на алгоритъма - ReliefF, премахващо посочените недостатъци, е предложено от Игор Кононенко (Robnik-Sikonja, Kononenko 2003).

Първо, вместо един положителен съсед се намират k такива съседи и второ, вместо един най-добър отрицателен съсед алгоритмът намира по k такива съседи от всеки клас *c*, несъвпадащ с класа c_E на проверявания пример. Приносът на най-близки положителни съседи за изчисляване на теглата w_i се получава чрез осредняването, а приносът на най-близки отрицателни съседи за всеки клас се претегля чрез априорната вероятност на класа *P(c)* (оценена на базата на всички обучаващи примери), делена върху сумата на априорните вероятности на всички „отрицателни класове ($1 - P(C_E)$). Псевдокодът на алгоритъма ReliefF е представен в Таблица 13-2.

Алгоритъм ReliefF

Вход: Обучаващи данни $D = \{X_1, \ldots, X_n\}$

Метрика $d(X,Y) = \left(\sum_{i=1}^{p} \delta^L(x_i, y_i) \right)^{\frac{1}{L}}$

Параметър *m* – брой на итерации

Параметър k – брой на най-близки примери

- *Инициализация:*

За всички атрибути направи: $w_i \leftarrow 0$

- *Обучение:*

За *i* от 1 до *m* направи:

1. Избери случайно пример $E = <\{e_1,\ldots,e_p\}, c_E> \in D$.
2. Намери *k* негови най-близки положителни съседи
   $R^j = <\{r_1^j,\ldots,r_p^j\}, c_E> \in D, \quad j = 1,\ldots,k$
3. За всеки клас $c \neq c_E$ направи
4. Намери k най-близки съседи от клас *c*:
   $N(c)^j = <\{n_1^j,\ldots,n_p^j\}, c> \in D, \; j = 1,\ldots,k$

5. За всички атрибути *i* направи:

$$w_i \leftarrow w_i - \sum_{j=1}^{k} \frac{\delta(e_i, r_i^j)}{k \cdot m} + \frac{1}{m \cdot k} \sum_{c \neq c_E} \frac{P(c)}{1 - P(c_E)} \sum_{j=1}^{k} \delta(e_i, n_i^j(c))$$

Таблица 13-2. Алгоритъм ReliefF

Целта е алгоритмът да намери теглата, при които атрибутите да разделят по най-добрия начин всяка двойка от класове независимо от това, кои два класа се намират най-близко един до друг.

За да може алгоритмът да работи с непълни данни, функцията $\delta(x_i, y_i)$ е разширена за липсващите стойности на i-тия атрибут чрез изчисляване на вероятностите, че атрибутът в двата примера приема различни стойности[3]:

- ако само един пример (например *Y)* има неизвестна стойност на i-тия атрибут, то $\delta(x_i, y_i = ?) = 1 - P(x_i|c_Y)$;
- ако стойностите на атрибута са неизвестни и в двата примера, то

$$\delta(x_i = ?, y_i = ?) = 1 - \sum_{v \in V_i} P(v \mid c_X) P(v \mid c_Y),$$

където V_i е множество от известни стойности на атрибута A_i.

Условните вероятности се апроксимират чрез относителните честоти на срещане на съответните стойности в обучаващото множество, например $P(x_i|c_Y)$ се изчислява като отношението на броя на примерите от класа на *Y*, в които стойността на i-тия атрибут е равна на x_i, към общия брой на примерите от класа на *Y* в обучаващата извадка.

Резултатите на ReliefF зависят от избор на параметрите *k* и *m*. Експериментите показали, че най-добрите резултати (от гледната точка на класификационната точност) се получават при $k = 10$.

Изборът на броя на итерации *m* зависи от броя на обучаващите примери, тъй като позволява да бъде установен баланс между точността на изчислявани статистически приближения и сложността на изчисления. При сравнително неголеми бази данни авторите на алгоритъма предлагат да бъдат използвани всички обучаващи примери (т.е. $m = n$). В общия случай броят на итерации зависи от конкретна база данни и трябва да бъде установен експериментално. Във всички случаи е желателно да бъдат избирани примери, които покриват цялото пространство от примери.

При класификацията оригиналният Relief използва още един входен параметър τ - т. нар. *параметър на релевантността.* Ако $w_i > \tau$, то $w_i \leftarrow 1$. В противен случай

[3] При използване на непрекъснати атрибути те трябва предварително да бъдат дискретизирани.

$w_i \leftarrow 0$. По този начин Relief работи като *избиращ признаци* алгоритъм. За да използваме получената оценка за важността на атрибути във вид на тегловни коефициенти, участващи в метриката за сходство (т. е. да превърнем Relief в един *претеглящ признаци* алгоритъм), се налага те да бъдат преобразувани по такъв начин, че да са в интервала [0,1]. Това може да се направи по следния начин:

- **Привеждане на w_i в интервал [0,1]:**
  1. За всички атрибути : ако $w_i \leq 0$ то $w_i \leftarrow 0$,

  2. **Иначе** За всички атрибути направи:

$$w_i \leftarrow \frac{w_i}{\sum_{i=1}^{n} w_i}$$

13.2.2. Статистически методи за оценяване на атрибути

Съществуващите статистически методи за избор на релевантните атрибути могат да бъдат разбити на две основни групи. Към първата група спадат методите, базиращи се на изчисляване на условните вероятности, дискретизация на непрекъснати и бинаризация на номинални атрибути. Втората група методи за оценяване на атрибути използва предложената от Шенън мярка за взаимната информация между стойностите на атрибута и класа на обучаващите примери.

Методи, базиращи се на условните вероятности

Два са по-известни метода, използващи условните вероятности за определяне теглата на атрибути - първият се нарича *метод за изчисляване на междукласовата важност на признаци* (CCF-метод: Cross-Category Feature Importance). Методът изчислява теглата чрез осредняване по класове:

$$w_i = \sum_{c \in C} P(c \mid f_i)^2$$

Мотивацията за създаването на метода CCP е да се назначават по-големи тегла на признаци, които се срещат по-рядко сред класовете[4]. Обаче алгоритмът е нечувствителен към разпределянето на признаците по класове (т.е. теглото на признака не зависи от класа), което изглежда нереалистично за практически приложения. По тази причина е бил създаден т. нар. *метод за изчисляване на категориалната важност на признаци* или PCF-метод (Per Category Feature Importance). Той назначава по-големи тегла на атрибути, силно корелиращи с дадения клас:

$$w_i(c) = P(c \mid f_i)$$

[4] Става дума за *двоичен* признак.

В метода теглото на i-тия признак на примера от клас *c* е условната вероятност, че примерът принадлежи на *c* при зададена стойност на признака. Доказано е, че методът има тенденция да класифицира повечето от примерите към най-често срещания клас.

Методи, използващи мярка за взаимна информация

Мярката за взаимната информация между стойностите на атрибута и класа на обучаващите примери — MI-мярката (от Mutual Information) - определя степента, с която се намалява неопределеността в стойността на една променлива при наличието на някакви знания за стойността на друга променлива. При използването на тази мярка теглата на атрибутите се изчисляват по следната формула:

$$w_i = \sum_{v \in V_i} \sum_{c \in C} P(c_X = c \wedge x_i = v) * \log \frac{P(c_X = c \wedge x_i = v)}{P(c_X = c) * P(x_i = v)},$$

където $P(c_X = c)$ е вероятността, че класът на произволен пример X от обучаващото множество е c, а $P(x_i = v)$ е вероятността, че стойността на неговия i-ти атрибут е v. Тук V_i е множеството от всички възможни стойности на i-тия атрибут.

Използването на мярката за взаимна информация предполага, че описващите примери атрибути са независими. Това води до деградацията на класификационното поведение на използващите мярката алгоритми при решаването на задачи с множество взаимодействуващи признаци.

Проведените сравнителни експерименти показаха, че повечето от описаните методи за оценяване на атрибути могат да отсейват нерелевантни признаци освен в случаите, когато примерите съдържат голямо количество взаимодействуващи признаци. Статистическите методи до голяма степен зависят от качеството на предварителната обработка на данни (избрания метод за нормализация и дискретизация на атрибути). Итеративните методи превъзхождат по точност статистическите методи в проблемните области с малък брой взаимодействуващи признаци. Скоростта на обучение на итеративните методи е по-голяма от тази на статистическите.

13.3. Оценяване на екземпляри

В началото на предишната лекция бе споменато, че едно от основните свойства на естествените понятия е тяхната *структурност*, т. е. различни екземпляри на едно и също понятие го представят в различна степен. Например един стол с четири крака “по-добре” представя понятието “стол” от стола с пет или три крака (в случая първият стол е по-типичен представител на понятието). В контекста на разгледаните алгоритми за самообучение това означава, че при еднакво сходство на новия обект с два или повече екземпляра от различни понятия предимство се дава на екземпляра с най-голяма степен на представителност за своята категория.

Описаният ефект може да бъде лесно постигнат, ако с всеки екземпляр *X* свържем тегловен коефициент Wx > 0, отразяващ степента на представителността му в съответната категория, а разстоянието между два произволни екземпляра да дефинираме като

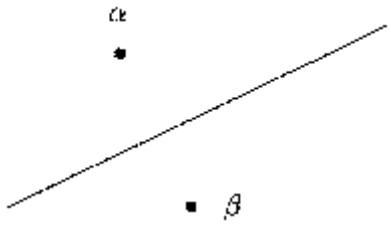
$$\Delta(X,Y) = W_X * W_Y * d(X,Y)$$

където *d(X, Y) е* вече познатата мярка за измерване на разстояние между два обекта, например:

$$d(X,Y) = \left(\sum_{i=1}^{n} w_i * \delta^L(x_i, y_i) \right)^{\frac{1}{L}}$$

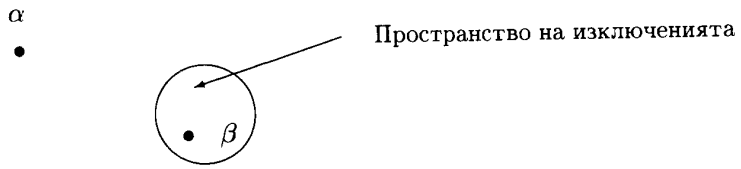
Ако *Y* е нов пример (т. е. с неизвестна класификация), то $W_Y = 1$. Колкото по-представителен е един пример, толкова е *по-малък* неговият тегловен коефициент.

Да анализираме как се отразява въвеждането на тегла на екземплярите върху пространството на признаците. Да разгледаме отначало пространство, съдържащо само две точки с класификациите α и *β*. При липса на тегла тези две точки определят една хиперравнина, разделяща n-мерното пространство на признаците на две области *a* и *β* (вижте двумерната илюстрация по- долу).



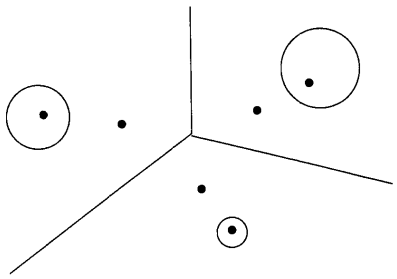
При използване на алгоритъма NN всяка нова точка ще бъде класифицирана като *a* или *β* в зависимост от това, дали е разположена над или под от тази хиперравнина.

Претеглянето на екземпляри геометрично може да се разглежда като създаване на определена сферична обвивка около екземпляра с по-голямо тегло (вижте следващата рисунка). Обвивката определя нещо като "пространство на изключенията", което се свива с нарастването на теглото на екземпляра. Само точки, попадащи *вътре* в това пространство, ще бъдат класифицирани чрез класа на пораждащия пространството екземпляр.



Когато теглата са равни, се получава специалният случай на хиперравнина, показан на първата рисунка. В по-общия случай, когато пространството съдържа много екземпляри, тези с най-малките тегла ще разделят пространството на няколко

области чрез определени хиперравнини. Ако теглата на тези "най-представителни" екземпляри не са равни, то границите на областите се формират от хиперокръжностите с много голям радиус. На практика всеки екземпляр играе ролята на "правило" за своята област, присвояващо неговата класификация на всеки пример, попаднал в областта. Екземплярите с по-голямо тегло образуват около себе си пространствата на изключенията. Пример на така разделено пространство е показан на следващата рисунка.



13.3.1. Определяне на типичността на екземпляри

Един от подходите за определяне на степента на представителност на екземпляра чрез неговата *типичност* е предложен от Джиампинг Занг. Типичността на екземпляра се определя въз основа на неговото фамилно сходство, дефинирано като функция от две стойности — средното сходство на екземпляра с други екземпляри от същото понятие *(вътрешнокласовото сходство)* и средното сходство на екземпляра с екземпляри от други класове *(междукласовото сходство)*. Ако означим с $s(X_i,X_j)$ сходството между екземплярите X_i и X_j, то вътрешнокласовото $S_{\cup}(X_i)$ и междукласовото сходство $S_{\cap}(X_i)$ на екземпляра X_i се определят по следните формули:

$$S_{\cup}(X_i) = \frac{1}{N} * \sum_{j=1}^{N} s(X_i, X_j), \qquad S_{\cap}(X_i) = \frac{1}{M} * \sum_{j=1}^{M} s(X_i, Y_j),$$

където всички X_j са от същия клас както и X_i, а N е техният брой; Y_i означава екземпляр от клас, несъвпадащ с класа на X_i, а M е общият брой на такива екземпляри. Сходството между екземпляри се определя като величина, обратна на разстоянието между тях[5] (вижте предишната лекция).

Интуитивно е ясно, че колко по-голямо е сходството на екземпляра с други екземпляри от същия клас и колко по-малко е неговото сходство с представители на други класове, толкова по-голямо е фамилното сходство на екземпляра и толкова по-типичен представител на класа е той. С други думи, типичните екземпляри имат високо вътрешнокласовото и малко междукласовото сходство.

[5] В своята работа Занг използва съотношение *s(X, Y) = 1 – d(X, Y)*.

Занг предлага да се измерва типичността на екземпляра *typ(X)* като отношение на неговото вътрешнокласово към междукласовото сходство:

$$typ(X) = \frac{S_{\cup}(X)}{S_{\cap}(X)}.$$

Екземплярите с типичност, по-голяма от единица, могат да се считат за най-представителни в класа; тези с типичност под единица са възможни изключения (нетипични представители) или просто "шум", а тези с типичност, близка до единица, се разглеждат като близки до границата на своя клас. Връзката между тегловния коефициент Wx, използван при измерване на разстояние Δ до някой нов обект, подлежащ на класифициране, и типичността на екземпляра се определя по формулата:

$$W_X = \frac{1}{typ(X)}.$$

13.3.2. Представителност и класификационно поведение

Един итеративен метод за изчисляване на представителността на екземпляр *чрез оценяване на неговото поведение при класификация на други примери от обучаващата извадка* е предложен от Кост и Салзберг. Алгоритмът се инициализира чрез случайно избрани примери от обучаващата извадка (т. нар. *ядра)*. Броят на ядрата се задава от потребителя и служи като входния параметър на алгоритъма.

Теглото $W_X = \frac{N}{M}$ се определя като частното от деление на общия брой на използването на екземпляра *N* (броят на случаите, когато той се е оказал най-близък до новия пример) към броя на неговото коректно използване *M* (т. е. когато той класифицира новия пример правилно). По този начин по-представителните (т.е. по-точните при класифициране) екземпляри ще имат тегло Wx ≈ *1* а по-непредставителните Wx > 1. Тези "некоректни" екземпляри могат да представляват или "шум", или изключения - малки области в пространството на признаците, за които не важат общите правила. Предложеният метод е насочен не към филтриране на зашумени примери, а към идентификация на екземпляри-изключения. В Таблица 13-3 е приведен алгоритъмът PEBLS (Parallel Exemplar-Based Learning System), който е една модификация на оригиналния алгоритъм, предложена от същите автори. Става дума за начина за началното инициализиране на новите екземпляри. В оригиналния алгоритъм всеки новозапомнен екземпляр се инициализира с теглото $\frac{1}{1}$. Да предположим, че системата вече е била обучена върху *n - 1* екземпляра. В резултат на обучението тя ги подрежда в определена йерархия на представителност, отразяваща структурата на проблемната област. Ако след класифициране следващият n-ти екземпляр е запомнен с тегло, равно на 1, то той автоматично ще стане един от най-представителните класификатори в областта. Предложената модификация избягва този недостатък, като инициализира

теглото на новозапомнения екземпляр с теглото на екземпляра, който го класифицира.

Основният проблем на алгоритъма PEBLS е оптималният избор на броя на ядрата, който изисква дълга и тежка от изчислителната гледна точка процедура за кръс-валидация на обучаващите данни.

Алгоритъм PEBLS

Вход: s - брой на ядрата.

- **Инициализация:**
  1. Избери случайно s обучаващи примера X_i, $i = 1, \ldots, s$
  2. $W_{X_i} \leftarrow \frac{1}{1}$, $i = 1, \ldots, s$
  3. Запомни тези примери като екземпляри на *Описание* на научаваните понятия: *Описание* := $\{X_1, \ldots, X_s\}$.

- **Обучение:**
  1. Избери случайно пример Y от останалите обучаващи примери.
  2. Намери най-близкия до Y съсед X от *Описанието* с тегло $W_X = \frac{N}{M}$, използвайки формулата $\Delta(X,Y) = W_X d(X,Y)$
  3. Запомни Y като нов екземпляр на *Описанието* и му присвои тегло $W_Y \leftarrow W_X$.
  4. *Ако* Y е класифициран правилно, то $W_X \leftarrow \frac{N+1}{M+1}$ иначе $W_X \leftarrow \frac{N+1}{M}$.
  5. Повтори стъпки 1-4 до пълното изчерпване на обучаващото множество.

Таблица 13-3. Алгоритъм PEBLS

13.4. Научаване на компактно описание на понятия

Всички описани досега алгоритми за самообучение чрез запомняне имат една обща характеристика — екстенсионалното описание на научаваното понятие включва в себе си *всички* примери от обучаващото множество. В настоящия раздел ще бъдат разгледани няколко алгоритъма избиращи за описание на понятие само определени екземпляри. Първият от тях използва за тази цел класификационното поведение на екземпляри. Вторият използва като критерий за избор въведеното в предишния раздел понятие „типичност" на екземпляра. Третата група методи научава компактното описание на понятия чрез конструиране и запомняне на нови *„изкуствени"* екземпляри, получавани от обучаващите примери чрез определени процедури за обобщение. Основната цел на всички тези подходи е да създадат методи устойчиви към шума и позволяващи научаване на компактни описания на понятия без (практическата) загуба на класификационната точност.

13.4.1. Алгоритъм IB2

Дейвид Аха е разработил и изследвал поведението на цялата фамилия от итеративни алгоритми за самообучение чрез запомняне. Тези алгоритми, наречени IBL (от Instance-Based Learning) се описват чрез следните три компоненти:

1. Ф*ункция на сходство:* изчислява сходството между обучаващия пример и описанието на понятията, представени чрез свои екземпляри.
2. *Класификационна функция:* получава резултатите от функцията на сходство и записи за класификационното поведение на екземпляри. Като резултат се извежда класификацията на проверявания пример.
3. *Функция за обновяване на описание:* обработва записите за класификационното поведение на екземпляри, резултатите от функцията на сходство и класификационната функция и решава, кои обучаващи примери трябва да бъдат включени в описанието на научаваните понятия.

Всичките алгоритми са итеративни варианти на алгоритъма за най-близкия съсед и използват една и съща функция на сходство:

$$s(X,Y) = -\sqrt{\sum_{i=1}^{n} \delta(x_i, y_i)}$$

където $\delta(x_i, y_i) = (x_i - y_i)^2$ за непрекъснати атрибути и $\delta(x_i, y_i) = (x_i \neq y_i)$ за номинални атрибути. Липсващите стойности на атрибути се подразбират максимално различни от наличните, а ако и двете стойности липсват, то $\delta(x_i = ?, y_i = ?) = 1$.

В настоящия раздел ще бъде разгледан само един от тези алгоритми – IB2, който създава компактното описание на понятия (вижте Таблица 13-4). Описанието на по-сложни алгоритми от тази фамилия - IB3 и IB4, можете да намерете в (Aha. et al. 1991).

Както се вижда от алгоритъма IB2, описанието на научавани понятия се конструира от обучаващите примери, които *не могат* да бъдат класифицирани правилно. Анализът показва, че повечето от тези примери се намират близо до границите на научаваните понятия. Алгоритмът създава значително по-компактни описания в сравнение с други алгоритми (например с IB1 - итеративен вариант на алгоритъма NN), запазвайки същата класификационна точност в проблемните области с ниско ниво на шум. Обаче точността на IB2 се намалява в значително зашумени проблемни области или в такива, в които научаваните понятия не са добре дефинирани (например медицинската диагностика), т. е. имат много различни изключения.

Алгоритъм IB2

- **Инициализация:**
  Описанието $\leftarrow \varnothing$ (текущо описание на понятия).
- **Обучение:**

1. Избери случайно пример *Y* от обучаващото множество.
2. Намери в *Описанието* екземпляр *X с* най-голямото сходство с *Y* (най-близкия съсед на *Y*)
3. *Ако* $c_X = c_Y$, *то* забрави *Y* и премини на стъпка 5. *Иначе*
4. Добави *Y* в *Описанието*.
5. Повтори стъпките 1-4 до пълно изчерпване на обучаващото множество.

Таблица 13-4. Алгоритъм IB2

Като всеки итеративен алгоритъм IB2 има още един недостатък - създаваните от него описания зависят от реда, в който постъпват обучаващите примери. Действително, един обучаващ пример, класифициран правилно (и следователно невключен в описанието) при някое междинно състояние на търсеното описание на понятие, може да не бъде класифициран правилно при друго състояние на описанието. За да бъде избегнат този недостатък, всички примери, класифицирани правилно при конструирането на описанието, трябва *отново* да бъдат проверени. Този процес трябва да бъде повтарян, докато нито един обучаващ пример не може да бъде добавен към описанието. Естествено, този подход води до увеличаване на изчислителната сложност на алгоритъма, обаче съществено повишава неговата класификационна точност.

13.4.2. Алгоритъм TIBL

TIBL (Typical Instance-Based Learning) е алгоритъм за създаване на компактно описание на понятия чрез оценяване на типичността на техните екземпляри. Типичността на екземпляра се определя като отношение на неговото вътрешнокласово към междукласовото сходство (вижте раздел 13.3.1), сходството между два примера се определя като величина, обратна на разстоянието между тях: *s(X, Y)* = 1 - *d(X, Y),* а за изчисляването на самото разстояние се използва нормализираното Евклидово разстояние:

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

$$\delta(x_i, y_i) = \begin{cases} \dfrac{|x_i - y_i|}{\max_i - \min_i}, & \text{когато i - ят атрибут е непрекъснат} \\ 1, & \text{ако } x_i \neq y_i \text{ и атрибутът е номинален} \\ 0, & \text{ако } x_i = y_i \text{ и атрибутът е номинален} \\ 0.5, & \text{ако една или двете стойности } \textit{липсват} \end{cases}$$

За класификация на нов пример се използва вариантът на алгоритъма за най-близкия съсед, при който разстоянието между запомнения екземпляр X и примера Y се изчислява като $\Delta(X,Y) = W_X * d(X,Y)$.

Алгоритмът TIBL (вижте Таблица 13-5) се състои от три основни стъпки:

- На първата се изчислява типичността на всички примери от обучаващото множество.
- На втората стъпка се проверява качеството на наученото (текущо) описание на понятия чрез използването му за класификация на останалите обучаващи примери. Ако всички примери са класифицирани правилно, работата на алгоритъма приключва.
- В противен случай се прави опит да се подобри текущото описание (третата стъпка). Това става, като в множеството от *правилно* класифицирани чрез описанието примери се търси пример с максимална типичност, който правилно класифицира най-типичен *неправилно* класифициран пример.

Втората и третата стъпка се повтарят до научаване на описанието, което правилно класифицира всички примери от обучаващото множество. Независимо от итеративния характер на процедурата по научаване на описанието на понятията, TIBL е неитеративен алгоритъм, тъй като обучаващите примери се обработват по няколко пъти. Създаваното от алгоритъма описание обикновено включва: а) няколко екземпляра от всяко понятие с висока степен на типичност *(typ(X)* > 1), отразяващи основни, централни тенденции на понятията; б) няколко "нетипични" екземпляра *(typ(X)* < 1), отразяващи „изключенията" и в) съвсем малко гранични екземпляри *(typ(X)* ≈ 1), определящи границите между понятията.

Емпиричната проверка на алгоритъма показа, че той създава значително по-компактни (в сравнение с обучаващата извадка) описания без практическа загуба на класификационната точност, а за някои проблемни области алгоритмът дори значително превъзхожда точността на обикновения алгоритъм за най-близкия съсед. Както и други IBL-алгоритми TIBL е много чувствителен към броя на нерелевантни атрибути, използвани за описание на примери.

Алгоритъм TIBL

- **Инициализация:**

1. *Изчисляване на типичността:* За всеки пример X от обучаващото множество изчисли *typ(X).*
2. *Инициализация на множества:*
   Описание - описание на понятия,
   Решени - правилно класифицирани обучаващи примери и
   Нерешени - неправилно класифицирани обучаващи примери.
   Описание ← ∅, *Решени* ← ∅, *Нерешени* ← цялото обучаващо множество.

- **Обучение:**

Класификация на примера чрез наученото Описание:

1. От множеството на *Нерешени* примери избери пример *Y* с максимална стойност на типичност *typ(Y)*.
2. Намери в *Описанието* екземпляр *X* с най-малкото разстояние *Δ(X, Y)* (най-близкия съсед на *Y*).
3. ***Ако*** $c_Y = c_X$, то
   Нерешени ← *Нерешени* – *{Y}*
   Решени ← *Решени* + *{Y}*
   премини на стъпка 10,
   Иначе

Класификация на примера чрез правилно Решени примери:

4. Намери в *Решени* пример *X* с максималната стойност на типичност *typ(X)*.
5. $W_X \leftarrow \frac{1}{typ(X)}$
6. Ако при добавяне към *Описанието, X* класифицира правилно *Y,* то добави *X* в *Описанието,*

   Иначе
7. Повтори стъпките 4-6 до пълното изчерпване на множеството *Решени.*
8. Ако *Y* не се класифицира правилно, добави *Y* към *Описанието* и $W_Y \leftarrow \frac{1}{typ(Y)}$

Обновяване на множествата от Решени и Нерешени примери:

9. С помощта на наученото *Описание* класифицирай всички останали примери от обучаващото множество и в зависимост от резултата формирай множествата *Решени* и *Нерешени.*
10. Повтори стъпките 1-9 до пълното изчерпване на множеството от *Нерешени* примери.

Таблица 13-5. Алгоритъм TIBL

13.4.3. Научаване на прототипи – алгоритъм SNMC

В първия раздел на предишната лекция беше спомената идеята на Елеонор Рош и Каролин Мервис, че в нашия мозък понятията се представят чрез определени свои образи - прототипи, представящи ги по най-добрия начин. Досега бяха разгледани алгоритми, които избират за прототипи *реално* съществуващи представители (алгоритмите IB2, TIBL). В настоящия раздел ще разгледаме накратко алгоритми, избиращи прототипи като *идеализирани* представители на понятието, които се конструират изкуствено. Ще разгледаме само методи, които представят прототипи

чрез изкуствени екземпляри, описващи *средните* атрибутни стойности за научаваното понятие. Те са наричат *класификатори по минималното разстояние* (Minimum-distance classifiers).

Един от първите класификатори по минималното разстояние е предложен от Дуда и Харт. Това е вариант на алгоритъма на най-близкия съсед, в който всяко понятие (клас) е представен чрез своя един единствен "среден" екземпляр-прототип. Ако понятието c се описва с n атрибута, то прототипът на това понятие $Proto(c) = < \bar{x}_1(c),..., \bar{x}_n(c) >$, където $\bar{x}_i(c)$ означава средното аритметично на i-тия атрибут за съответното понятие (клас). Основните недостатъци на подхода са: а) невъзможността за прилагането му към понятия, описвани с номинални атрибути; б) невъзможността за използване на тегла на атрибутите и в) създаването на само един прототип за всяко понятие.

Ще разгледаме алгоритъм SNMC (Symbolic Nearest Mean Classifier) (Datta & Kibler 1997), който избягва посочените недостатъци.

Първите два от тях се избягват чрез използване за атрибутното разстояние на вече добре познатата ни метрика MVDM[6]:

$$\delta_{MVDM}(v_1, v_2) = \frac{1}{2}\sum_{c \in C} | P(c \mid v_1) - P(c \mid v_2) |,$$

където v_1 и v_2 са две номинални стойности на един и същ атрибут.

Използвайки тази метрика, Дата и Киблер обобщават понятието "средно аритметично" за случая на номинални атрибути. Те дефинират средното μ на множество V от номинални стойности като стойност, минимизираща тяхното отклонение (дисперсия):

$$\mu = \arg\min_{v_1 \in V} \sum_{v_2 \in V, v_2 \neq v_1} \delta^2_{MVDM}(v_1, v_2)$$

Първият вариант на алгоритъма (SNM) научава само по един прототип за всеки клас и класифицира тестови примери чрез намирането на най-близък прототип, използвайки разстоянието:

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} \delta^2_{\mathrm{MVDM}}(x_i, y_i)}.$$

В много проблемни области научаването на един прототип за всеки клас е недостатъчно за коректното им описание. За научаването на повече от един прототип за клас, обучаващите примери за всеки клас трябва да бъдат разбити на групи *(клъстери)* от "най-сходни" примери и всеки клъстер да бъде представен чрез собствен прототип. SNMC използва алгоритъм за кластеризация k-MEANS, който има като параметри метриката за изчисляване на разстояние (в нашия случай това е MVDM) и броят на клъстерите k. Алгоритмът премества всеки пример в най-

[6] В оригиналния алгоритъм метриката е нормализирана по броя на класове |C|.

близкия до него клъстер, докато не бъдат преместени всички примери от същия клас. За определяне броя на клъстерите за всеки клас се използва класификационната точност на алгоритъма, изчислена върху цялото обучаващо множество. Основната идея е да се увеличава броят на клъстерите за дадения клас само когато това води до увеличаване на точността на прототипите, представящи обучаващите примери (вижте Таблица 13-6).

Алгоритъм SNMC
Вход: Обучаващите примери
Метрика MVDM

- **Инициализация:**
  Дискретизирай всички непрекъснати атрибути
  $k_c \leftarrow 1$ за всеки клас c
  Научи прототипи за всеки клас чрез *K-Means(k_c)* и запомни ги като *Описание*
  Изчисли точността на *Описание* върху всички обучаващи данни
- **Обучение:**
  Повтори
  За всеки клас c
  Повтори
  Научи разбиването на клас c на повече клъстери:
  Научи $k_c + 1$ прототипи за клас c чрез *K-Means(k_c+1)*
  Научи *Ново_Описание* чрез замяна в *Описание* на старите k_c прототипи на клас c с новите $k_c + 1$ прототипи на същия класс
  Изчисли точността на *Ново_Описание* върху всички обучаващи данни
  Ако точност(Ново_Описание) > точност(Описание) то :
  Описание $\leftarrow$ *Ново_Описание*
  $k_c \leftarrow k_c + 1$
  Ако точността не се подобрила
  Започни научаването на прототипи за следващия клас
  Докато не бъдат научени прототипи за всички класове.

Таблица 13-6. Алгоритъм SNMC

И така, алгоритъмът SNMC разделя обучаващите примери по класове, намира за всеки клас оптимален брой клъстери, максимизиращ класификационната точност върху обучаващото множество и създава по един прототип за всеки клъстер. Експерименталната проверка на алгоритъма върху 20 бази от данни показа, че той като цяло дава по-добрата точност от такива алгоритми като C4.5 (вижте лекцията за класификационни дървета) и PEBLS.