Изкуствен интелект - зимен семестър, 2006/2007 учебна година

Лекция 11:
Машинно самообучение

Същност на машинното самообучение

Х. Саймън: "Самообучението означава настъпване на такива промени в системата, които са адаптивни в смисъл, че позволяват на тази система при всеки следващ опит да извършва дадена работа по-ефективно и по-ефектно, отколкото при предишните опити".

Р. Михалски: "Самообучението е конструиране или модифициране на различни представяния на предмета на дейност".

САМООБУЧЕНИЕ ЧРЕЗ НАИЗУСТЯВАНЕ

Представлява най-примитивен тип машинно самообучение (МС), аналог на ученето наизуст (зазубрянето) при човешкото обучение.

Идея. Запазват се резултати от работата на системата (условията на тежки, трудоемки задачи и получените решения) с цел използването им наготово в случай, че постъпи заявка за решаване на вече позната задача. В чистия си вид методът не предполага никаква допълнителна обработка на тези резултати. Оказва се, че при редица задачи такова запазване води до чувствително подобряване на работата на системата (повишаване на бързодействието, икономия на памет и др.).

САМООБУЧЕНИЕ ЧРЕЗ МАКРООПЕРАТОРИ

Идея. Същата, както при самообучението чрез наизустяване – вариант на този тип МС, при който се предполага допълнителна обработка (обобщаване) на съхраняваните резултати от работата на системата. Използва се най-често в системи, свързани с планиране на действията.

Пример. Системата STRIPS има обучаваща подсистема за натрупване на резултати от вече извършена от нея работа под формата на макрооператори. Всеки макрооператор представлява обобщен план за решаване на даден тип задачи и се състои от *предусловия* (обобщават съществена част от началното състояние при решаването на конкретна задача за планиране), *тяло* (обобщение на конструирания от системата план) и *следусловия* (обобщение на целта при решаването на конкретна задача за планиране).

САМООБУЧЕНИЕ ЧРЕЗ УТОЧНЯВАНЕ НА ПАРАМЕТРИТЕ

Идея. Много системи с изкуствен интелект използват оценяващи (или разделящи) функции, които представляват комбинации от стойности на подходящо подбрано множество от параметри (признаци, фактори), като всеки параметър участва в оценяващата функция със съответен коефициент (тегло). При проектирането и първоначалното програмиране на такива системи обикновено е трудно априори да се определят правилно теглата на всички фактори, които участват във формулировката на

оценяващата функция. Един възможен подход за правилното определяне на теглата на отделните фактори е свързан с включването на средства за модифициране (доуточняване) на тези тегла на основата на натрупания опит от работата на системата с текущия вариант на оценяваща функция.

Това е един от най-често използваните методи за МС в областта на разпознаването на образи и при невронните мрежи.

САМООБУЧЕНИЕ ЧРЕЗ СЪВЕТИ

Нека предположим, че съществува (външен) учител, който е заинтересован обучаващата се система да научи нещо. Един такъв учител би се разочаровал от работата с повечето обучаващи се програми. На много от тях може да се каже кога да започнат работа и дали дават верен или грешен отговор, но нищо повече.

Най-често учителят иска да каже на ученика не само че е на погрешен път, а също и защо. В началото на историята на ИИ J. McCarthy използва термина "получател на съвети", за да опише програма, която може да се обучава с помощта на услужлив учител. Прогресът на действителната реализация на мечтата на McCarthy обаче е бавен. Причината за това е, че е трудно да се измислят достатъчно широки канали за комуникация от учителя към получателя на съвети. В идеалния случай комуникацията трябва да е на естествен език (напр. английски). Това обаче поражда много проблеми. Повечето програми за обработка на естествен език работят с прости разкази, а не с общи правила. Проблемите с многозначността, отнасянето на местоименията и т.н. могат само да станат по-трудни, когато изреченията, които трябва да се разберат, са по-сложни.

Съществен проблем тук е обстоятелството, че човекът, даващ съвети на естествен език, често не познава вътрешната структура на системата, получаваща съветите. Подсистемата за разбиране на естествен език трябва да може в такава ситуация да съотнесе съветите към вътрешната структура на обучаващата се система. Полученият съвет трябва да се операционализира чрез формулиране на подходящи правила, извършване на промяна в използваната оценяваща функция и т.н.

Алтернативен вариант е да се изисква от учителя да разбира точно момента, в който съветът се търси, и приблизително какъв трябва да бъде неговият формат, така че да може да пригоди съвета си към този формат. Много често обаче такова изискване трудно може да се изпълни.

Като пример за практическа програма, постигнала интересен компромис в това отношение, ще разгледаме накратко програмата (системата) TEIRESIAS, която играе ролята на интерфейс за получаване на съвети към системата MYCIN. Положителна черта на системата MYCIN е, че тя има сравнително прост модел. На всички въпроси се отговаря чрез обратен извод върху правила от ограничена област. Ето защо учителят може да забележи пропуските в знанията на MYCIN от почти същата гледна точка, от която ги вижда и самата система.

Системата TEIRESIAS работи върху примерни задачи, дадени й от учителя. Когато не е съгласен с решението на дадена задача, предложено от системата, учителят може да проследи веригата от изводи, довела до направеното заключение. Ако тази верига не е коректна, то в нея трябва или да има достигнато заключение, което не е трябвало да се достигне, или да липсва заключение, което е трябвало да се достигне. С други думи, могат да се разглеждат две възможни ситуации:

- някое правило е извело ***P*** от предпоставки, които са верни (удовлетворени), въпреки че ***P*** всъщност не следва от тях;

- някое заключение ***P*** е трябвало да се достигне, въпреки че всички правила, които биха могли да го изведат, не са били активирани поради някоя неудовлетворена предпоставка.

В първия случай програмата трябва да извърши специализация, т.е. правилото, което е извело ***P***, трябва да се промени така, че да не го извежда в дадената ситуация. Във втория случай програмата трябва да извърши обобщение или чрез добавяне на ново правило, което извежда ***P*** в текущата ситуация, или чрез отслабване на условието на някое съществуващо правило, което извежда ***P***. Въпреки че системата TEIRESIAS може да реши кой от тези подходи да се избере, тя не се опитва да намери сама правилното обобщение или специализация. Вместо това тя пита учителя, който трябва да го зададе.

Така през по-голямата част от работата си TEIRESIAS използва въпроси с кратки отговори или менюта за управление на диалога с учителя. Тя се отдалечава от този стил единствено когато възприема ново правило или нов конюнкт в условието на съществуващо правило. Тогава системата приема новата информация на естествен език, като разчита на твърди очаквания за това, което се въвежда, и на строги ограничения върху значението на думите. Дори и при тези ограничения обаче е възможно да се получи грешно значение. Ето защо, след като получи вътрешното представяне на отговора на естествен език, тя го превежда отново на естествен език и се консултира дали значението му е същото. Ако учителят отговори отрицателно, тогава TEIRESIAS опитва алтернативни преводи или поканва учителя да перефразира казаното.

По този начин TEIRESIAS извършва специализации и обобщения, ръководени от учител, основаващи се на две прости идеи:

- системата разчита на воден от меню диалог, който прави нейната вътрешна структура максимално ясна за учителя;
- когато е необходим вход на естествен език, системата се консултира с учителя, за да се убеди, че входният текст е преведен (разбран) правилно.

САМООБУЧЕНИЕ ЧРЕЗ ПРИМЕРИ

Идея. Това е тип МС (частен случай на т. нар. *индуктивно самообучение*), което обикновено се прилага при системи, предназначени за решаване на задачи, свързани с класификация на обекти.

Класификацията е процес, при който по даден обект (описанието на даден обект) се получава името на класа, към който принадлежи този обект. Класификацията е важен етап от решаването на много задачи от областта на ИИ. При създаването на системи за решаване на такива задачи преди всичко трябва да бъдат дефинирани класовете, с които ще се работи. Често е много трудно да се даде пълна и вярна дефиниция на отделните класове в разглежданата област. Поради това е полезно създаването на самообучаващи се програмни системи, които могат сами да изграждат дефинициите на класовете от предметната област. Задачата за изграждането на дефинициите на класове е известна като *изучаване на понятия* (concept learning). МС чрез примери е най-популярният метод за изучаване на понятия.

Характерно за този тип МС е използването на множество от т. нар. ***обучаващи примери***, които могат да бъдат от два типа: *положителни* и *отрицателни* обучаващи примери.

Ще разгледаме накратко два популярни подхода при МС чрез примери: подхода на т. нар. *покриване* и подхода на т. нар. *отделяне*.

Подход на т. нар. ***покриване***: целта при построяване на описанията е те да *покриват примерите* (положителните примери) за дадения клас. По-точно, за всеки клас се задава (пълно) множество от положителни обучаващи примери (описания на обекти, принадлежащи на класа) и множество от отрицателни обучаващи примери (описания на обекти, които не принадлежат на класа, но приличат на обекти от положителните примери). Извършва се *обобщаване* на положителните примери (по такъв начин, че полученото описание на класа да включва като частни случаи описанията на всички положителни примери) и *ограничаване* на отрицателните примери (по такъв начин, че описанието на класа да изключва описанията на отрицателните примери).

Пример: програмата ARCHES на П. Уинстън за класификация на обекти от света на кубовете.

Подход на т. нар. ***отделяне***: целта е *примерите за даден клас да се отделят* от тези за другите класове. Типичен представител на такава *разделяща стратегия* е подходът на *класификационните дървета*.

Построяване на класификационни дървета

Класификационно дърво (КД): Всеки *възел* в едно КД представя даден атрибут, срещащ се в примерите. *Дъгите*, излизащи от даден възел, представят различните възможни стойности на този атрибут. Всеки *лист* на дървото определя класификацията на даден пример, като стойностите на съответните атрибути са зададени по пътя от корена към този лист.

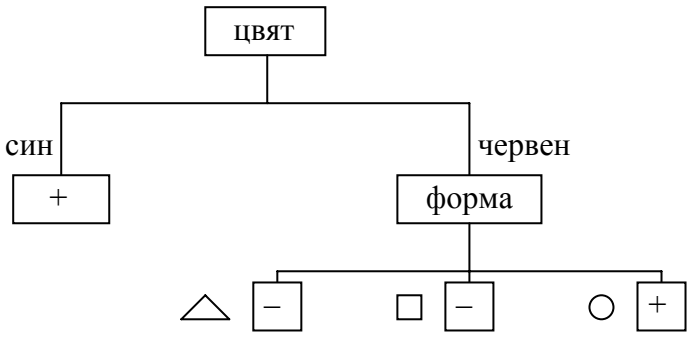
В случай на едно понятие листата се маркират с "+" за положителните примери и с "-" – за отрицателните примери.

Примерна предметна област: свят на цветните фигури. Нека разгледаме множеството $E = E^+ \cup E^-$ от положителни и отрицателни примери:

E^+ = {[червен, кръг], [син, триъгълник], [син, квадрат]}

E^- = {[червен, квадрат], [червен, триъгълник]}

Класификационното дърво, което разделя горното множество E на положителни и отрицателни примери, изглежда например както е показано на фиг. 1.



Фиг. 1. Класификационно дърво за примери от един клас

Базов алгоритъм за построяване на класификационно дърво

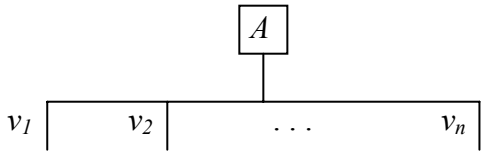
Вход: множество от обучаващи примери E.

Изход: класификационно дърво.

Ще разгледаме вариант на алгоритъма за самообучение на едно понятие (примерите са положителни или отрицателни).

1. Ако всички примери в E са положителни, генерира се лист от тип "+". Ако всички примери в E са отрицателни, генерира се лист от тип "–". В противен случай се

избира атрибут A със стойности v_1, v_2, … , v_n и се генерира възел с името на атрибута и n изходни дъги, както е показано на следващата фигура:



2. Множеството от обучаващи примери E се разделя на подмножества E_1, E_2, ... , E_n – такива, че всяко множество E_i съдържа примерите от E, които имат стойност v_i на атрибута A. Тъй като възелът от разглеждания тип предизвиква разделяне на множеството от примери, той се нарича *разделящ възел*.
3. Алгоритъмът се прилага рекурсивно за всяко множество E_i (i = 1, 2, …, n), като коренът на всяко от получените дървета се свързва със съответното разклонение v_i, излизащо от възела A.

Разгледаният базов алгоритъм се нарича TDIDT (Top-Down Induction of Decision Trees).

Пример. Нека обучаващото множество E се състои от следните примери (положителните са маркирани с "+", а отрицателните – с "–"):

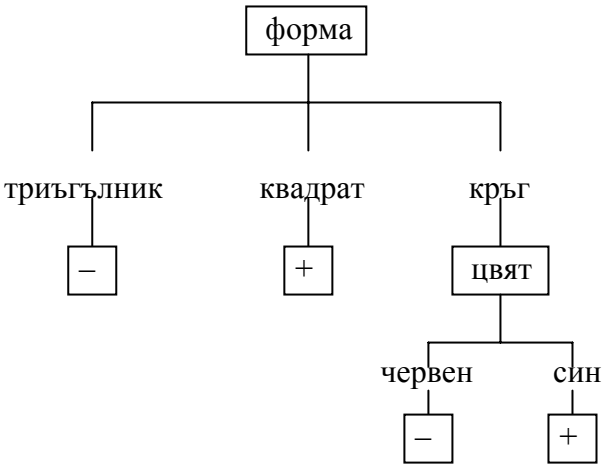
1. {цвят = червен, форма = кръг} –
2. {цвят = червен, форма = квадрат} +
3. {цвят = червен, форма = триъгълник} –
4. {цвят = син, форма = триъгълник} –
5. {цвят = син, форма = квадрат} +
6. {цвят = син, форма = кръг} +

Тъй като обучаващите примери не са нито само положителни, нито само отрицателни, трябва да се избере атрибут и да се построи съответен разделящ възел. Нека да изберем атрибута **форма**. Възможните му стойности (стойностите, които се срещат в обучаващото множество) са **триъгълник**, **квадрат** и **кръг**. Ето защо от разделящия възел **форма** излизат три дъги, които съответстват на тези стойности.

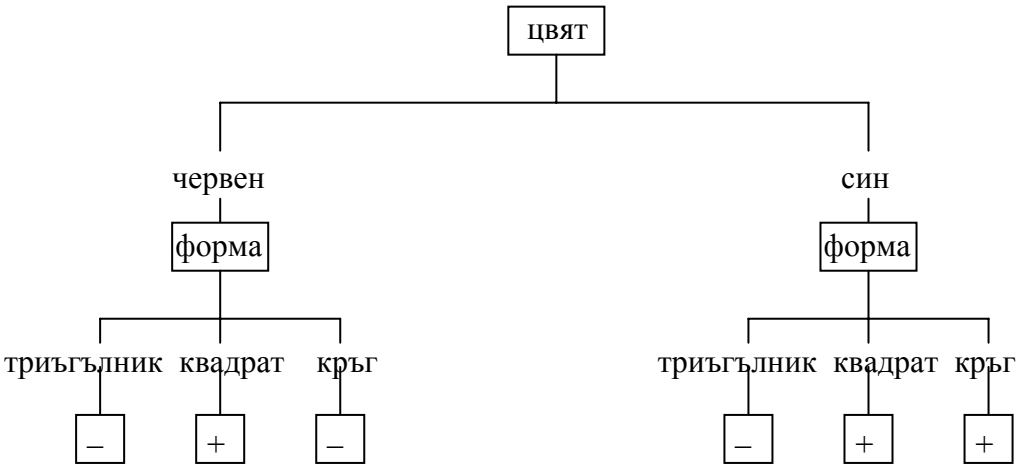
По-нататък множеството E се разделя на три подмножества E_1, E_2 и E_3, съответни на стойностите **триъгълник**, **квадрат** и **кръг**: $E_1 = \{3,4\}$, $E_2 = \{2,5\}$ и $E_3 = \{1,6\}$.

Сега прилагаме алгоритъма поотделно за всяко от тези множества. Тъй като E_1 и E_2 съдържат примери от един и същ клас (съответно само отрицателни и само положителни), алгоритъмът директно генерира лист от тип "–" за E_1 и лист от тип "+" за E_2. Множеството E_3 обаче не е еднородно и следователно за него трябва да се построи разделящ възел. Единственият останал атрибут е **цвят** с възможни стойности **червен** и **син**. Разбиването на E_3 на две множества относно тези стойности на атрибута вече дава две еднородни множества {1} и {6}, съдържащи съответно само отрицателни и само положителни примери. Това от своя страна води до генерирането на два листа, с което цялото класификационно дърво е построено (фиг. 2).

Естествено, видът на класификационното дърво зависи от избора на атрибут в т. 1 на алгоритъма TDIDT. Това може да се види, като се сравнят фигури 2 и 3.



Фиг. 2. Класификационно дърво с първи атрибут **`форма`**



Фиг. 3. Класификационно дърво с първи атрибут **`цвят`**

При конструирането на класификационни дървета целта е да се построи *“компактно” дърво с минимален брой разделящи възли* (т.е. целта е да се минимизира необходимият брой тестове). Така за предпочитане са плитките и силно разклонени дървета.

Постигането на тази цел зависи от избора на “добри” атрибути. Критерий за качеството на даден атрибут е способността му да разделя множеството от примери на групи от еднородни примери. Формално това най-често се прави с използване на т. нар. *информационен критерий*.