

3. Изход

Изход: Представяне на знания

Таблицы – Представяне на данни/ решения под формата на таблици.

Линейни модели – Модели, които използват линейни уравнения за предсказване на стойности.

Дървета – Структури, които представят решения чрез **разклонения**, които се базират на различни атрибути.

Правила – Логически изрази, които дефинират как да се вземат решения въз основа на данни.

Правила за класификация – Правила, които категоризират данни в определени класове.

Асоциативни правила – Правила, които откриват взаимовръзки или зависимости между различни елементи в данни.

Правила с изключения – Правила, които включват изключения, когато обикновеното правило не важи.

По-изразителни правила – Правила, които могат да се използват за по-сложни задачи и които имат по-голяма гъвкавост.

Представяне на инстанции – Представяне, при което данните са представени чрез примери или инстанции.

Клъстери – Групи подобни елементи, създадени чрез клъстеризация.

Изход: Представяне на структурни шаблони

- Множество различни начини за представяне на шаблони, вкл. **дървета на решения, правила, представяне на инстанции ...**
- Това се нарича също така "представяне на знания".

- Представянето на данни определя метода на извеждане на заключения.
- Разбирането на изхода е ключово за разбирането на основните методи на обучение.
- Разл. типове изходи се използват за разл. проблеми на обучение

Таблицы на решения

- Най-простият начин за представяне на изхода е да използваме формата, който се използва за представяне на входните данни.
- **Таблица на решения** - Основният проблем е изборът на правилните атрибути.

Линейни модели

- Просто представяне- традиционно се използва **регресия**.
- Входните данни (атрибутни стойности) и изходът са числови.
- Изходът е сумата на претеглените входни атрибути.
- Трикът е да се намерят добри стойности за претеглянията –разл. начини- пр. минимизиране на квадратичната грешка.

Линейни модели за класификация

- **Бинарна класификация:** Линия разделя двата класа.
- **Решаваща граница** – определя мястото, където решението се променя от една стойност на класа към друга.
- Прогнозата се прави, като се заменят наблюдаваните стойности на атрибутите в уравнението.
- Границата става високодименсионален **хиперплан** когато има много атрибути.

Дървета на решения

- Подход "**разделяй и владей**" произвежда дърво.
- Възлите на дървото включват тест за определен атрибут.
- Обикновено стойността на атрибута се сравнява с константа.
- Други възможности:

- Сравняване на стойности на два атрибута.
- Използване на функция от един или повече атрибути.
- **Листата** на дървото задават **класификация**, набор от класификации или вероятностно разпределение за инстанции.
- Неизвестната инстанция се пренасочва надолу по дървото.

Номинални и числови атрибути в дървета

- **Номинални:** Броят на децата е равен на броя стойности на атрибута. Атрибутът няма да се тества повече от веднъж. Др.възможности: разделяне 2 събсета
- **Числови:** Тества се дали стойността е по-голяма или по-малка от константа. Атрибутът може да се тества няколко пъти.
- Др. възможности: Триразделяне (или многопосочно разделяне).

Пропуснати стойности - Дали липсата на стойност има значение?

- Да: „липсваща“ е отделна стойност.
- Не: „липсваща“ трябва да се третира по специален начин.

Решение А: Присвояване на инстанцията на най-популярния клон.

Решение В: Разделяне на инстанцията на парчета. Парчетата получават тежест според пропорцията от обучаващите инстанции, които минават през всеки клон. Класификациите от листата се комбинират с тежестите, които са достигнали до тях.

Дървета за числово прогнозиране /регресионно дърво/

- **Регресия:** Процес на изчисляване на израз, който прогнозира числова стойност.
- **Дърво за регресия:** „Дърво на решения“, където всяко листо прогнозира числова стойност. Прогнозираната стойност е средната стойност на обучаващите инстанции, които достигат до листото.

- **Моделно дърво:** „Дърво за регресия“ с линейни регресионни модели в листата-Линейни участъци, които представят непрекъснатата функция. В листото имаме линейна регресия.

Правила за класификация -Популярен алтернативен метод на дървета на решения.

- **Антецедент (предусловие, лява страна):** серия от тестове (по същия начин, както тестовите на възлите в дърво на решения). Тестовите обикновено се комбинират с логическо "И" (но може да бъдат и общи логически изрази).
- **Следствие (заключение, дясна страна):** класове, набор от класове или вероятностно разпределение, присвоени от правилото.
- Индивидуалните правила често се комбинират логически с "ИЛИ".
- Възникват конфликти, ако различни заключения са приложими-политики за разрешаване конфликтите – избираме по важност, по обхват

От дървета към правила

- Лесно: преобразуване на дърво в набор от правила.
- Едно правило за всяко листо:
 - **Антецедент** съдържа условие за всеки възел по пътя от корена до листото.
 - **Следствието** е класът, присвоен от листото.
- Това води до правила, които са еднозначни.
- Не е важно в какъв ред се изпълняват правилата.
- Но: резултатните правила са ненужно сложни.
- **Окастрияне (pruning)** за премахване на излишни тестове/правила.

От правила към дървета

- По-трудно: преобразуване на набор от правила в дърво.

- Дървото не може лесно да изрази дизюнкцията (или) между правилата. Пример: правила, които тестват различни атрибути.
- Трябва да се преодолее симетрията. Дърво съдържа идентични поддървета (**проблем с репликираните поддървета**).

Дърво за проста дизюнкция - дърво, което представя логическа операция "или" между две или повече условия.

- **Исключващо или (exclusive-or) проблем** - логическа операция, при която резултатът е истина, само ако едно от двете условия е вярно, но не и двете едновременно. Ако имаме две стойности, А и В, тогава $A \text{ XOR } B$ ще бъде вярно, ако А е вярно и В е невярно, или ако В е вярно и А е невярно, но ако и двете са верни или и двете са неверни, резултатът ще бъде невярно.
- **Дърво с репликирано поддърво**- в дървото има повтарящи се (репликирани) части - ако имате поддърво, което се използва на различни места в основното дърво, то ще бъде "**репликирано**" – ще се среща повече от веднъж в различни части на дървото. Това може да създаде проблеми, тъй като може да доведе до излишно повторение на същите изчисления или условия.

„Знания“ в правилата

- Дали правилата са **независими** парчета знание? (Изглежда лесно да добавим /премахваме правило към съществуваща база от правила.)
- Проблем: пренебрегва как се изпълняват правилата.
- Два начина за изпълнение на набор от правила:
 - Подреден набор от правила (**списък с решения, decision list**)-Редът е важен за интерпретацията
 - Не подреден набор - може да се припокриват и да водят до различни заключения за една и съща инстанция.

Интерпретиране на правилата

- Какво ако две или повече правила са в конфликт?

- Да не се даде заключение изобщо?
- Да се вземе правилото, което е най-популярно/с най-голямо покритие/ в обучаващите данни?
- Какво ако не се прилага правило за даден тестови пример?
 - Да не се даде заключение изобщо?
 - Да се вземе класът, който е най-чест в обучаващите данни?

Специален случай: Булев клас

- Предположение: ако инстанцията не принадлежи на клас "да", тя принадлежи на клас "не".
- Трик: **учим само правила за клас "да"** и използваме правило по подразбиране за "не".
- Редът на правилата не е важен. Няма конфликти!
- Правилото може да бъде написано в дизюнктивна нормална форма.

Асоциативни правила- прогнозира всякакви атрибути и комбинации от атрибути. **Те не са предназначени да се използват заедно като набор.**

- Проблем: огромен брой възможни асоциации.
- Изходът трябва да е ограничен до показване само на най-предсказуемите асоциации- с висока **подкрепа** и **увереност**.
- Лява част предсказва дясна част

Подкрепа и увереност на правило

- **Подкрепа/Support/:** брой инстанции, които са правилно предсказани.
- **Увереност/Confidence/:** брой правилни предсказания, като пропорция от всички инстанции, на които правилото се прилага.

- Пример: 4 хладни дни с нормална влажност. Подкрепа = 4, увереност = 100%. Обикновено: **минималната** подкрепа и увереност са предварително зададени

Интерпретиране на асоциативни правила: Интерпретацията не е

Rules with exceptions - да се позволи на правилата да имат изключения. По-сложен пример: изключения на изключения на изключения...

Предимства на използването на изключения

- Правилата могат да бъдат обновявани **инкрементално**-обновяване правилото на база изключение.
- Лесно е да се вкарат нови данни; Лесно е да се интегрират знания от домейна.
- Хората често мислят в контекста на изключения.
- Всяко заключение може да се разглежда само в контекста на правилата и изключенията, които водят до него.
- Локалността е важна за разбирането на големи набори от правила.
- “Нормалните” набори от правила не предлагат това предимство.

Повече за изключения - “Default...except if...then...” е логически еквивалентно на “if...then...else” (else е “default”). Но: изключенията предлагат психологическо предимство. Предположение: дефинициите и тестовете, които се правят в началото, обикновено важат по-широко, отколкото изключенията по-нататък.

Правила включващи връзки, релации - До момента всички правила включваха сравняване на стойността на атрибут с константа (температура < 45). Тези правила се наричат “пропозиционални” /от пропозиционалната логика/. Какво се случва, ако проблемът включва взаимоотношения между примери (родословно дърво)? Не може да се изрази с пропозиционални правила.

A propositional solution - Използването на взаимоотношения между атрибути дава възможност за правила като: “Атрибут 1 е по-голям от Атрибут 2”. Това описание по-добре обобщава нови данни. Стандартни отношения: $=$, $<$, $>$. Но: търсенето на взаимоотношения между атрибути може да бъде скъпо. Простото решение: добавяне на допълнителни атрибути (например, бинарен атрибут “ширината ли е по-малка от височината?”).

Rules with variables - Използване на променливи и множествени взаимоотношения: “Върхът на стълба от блокове стои: Цялата стълба стои.” Рекурсивно дефиниране!

Inductive logic programming (ILP) -Рекурсивното дефиниране може да се види като логическа програма. Техники за учене на логически програми произхождат от областта на “индуктивното логическо програмиране” (ILP). Но: рекурсивните дефиниции са **трудни** за учене. Малко практични проблеми изискват рекурсия. Много ILP техники са ограничени до не-рекурсивни дефиниции, за да улеснят ученето.

Instance-based representation/Представяне основано на примери/ - Най-простата форма на учене: учене чрез повторение/зубрене/. Тренираните инстанции се търсят за инстанция, която най-много прилича на новата. Самите инстанции представляват знанията. „учене, базирано на инстанции“. Функцията за сходство определя какво е “научено”. Instance-based learning е лениво, мързеливо учене. Методи: nearest-neighbor, k-nearest-neighbor и др. За да намерим най-близкия съсед мерим разстояние.

The distance function - Най - простият случай: един числов атрибут. Разстоянието е разликата между стойностите на атрибутите (или функция на тях). Няколко числови атрибута: обикновено се използва Евклидово разстояние и атрибутите се нормализират. **Номинални атрибути:** разстоянието е 1, ако стойностите са различни, и 0, ако са еднакви. Дали всички атрибути са еднакво важни? Може да се наложи претегляне на атрибутите.

Learning prototypes - Само тези инстанции, които участват в решение, трябва да се съхраняват. Шумните инстанции трябва да бъдат филтрирани. Идеята: да се използват само прототипни/най-близки/ примери.

Rectangular generalizations - Правилото на най-близкия съсед се използва извън правоъгълниците. Правоъгълниците са правила! (Но те могат да бъдат по-консервативни от “нормалните” правила). Нестедни правоъгълници са правила с изключения.

Representing clusters: Проста 2D репрезентация; Диаграма на Вен; Вероятностно разпределение; Дендрограма. Мерим разстояния и разделяме.