6. **Алгоритми за обучение на дървета и правила**

Decision trees; От ID3 до C4.5 (pruning, numeric attributes, ...); Classification rules; От PRISM до RIPPER и PART (pruning, numeric data, …); Association Rules; Faster rule mining with frequent-pattern trees

Decision Trees (Дървета на решения) -модели за машинно обучение, които се използват за **класификация и регресия**. Работят чрез разклоняване на данните въз основа на стойности на различни атрибути, като крайните възли представляват категориални или числови стойности. Те представляват йерархична структура, където всеки вътрешен възел е тест върху някакъв атрибут, а листата представят крайните класове (класификация) / стойности (регресия).

-ID3 (Iterative Dichotomiser 3) е алгоритъм за изграждане на дървета за решения, който използва информационна печалба (Information Gain), за да избира най-добрите атрибути за разделяне. **C4.5** е подобрена версия на ID3 и включва няколко важни подобрения:

✅ **Pruning (орязване)** – намалява размера на дървото, като премахва клонове, които не носят допълнителна информация, за да се избегне overfitting (прекомерно напасване). **Стабилност при шумни данни**: изисква механизъм за орязване (pruning)

✅ **Обработка на числови атрибути** – C4.5 може да обработва както категорийни, така и числови данни, като намира гранични стойности за разделяне.

✅ **Работа с липсващи стойности** – вместо да игнорира липсващите стойности, алгоритъмът използва вероятностен подход за тяхното компенсиране.

C4.5 става един от най-популярните алгоритми за изграждане на дървета на решения. По-късно той има комерсиален наследник: C5.0.

Числови атрибути

Стандартен метод: двойни разделения (binary splits); Пример: temp < 45; За разлика от номиналните атрибути, всеки числов атрибут има много възможни точки на разделяне

Решение:

1.Оценява се информационната печалба (или друга мярка) за всяка възможна точка на разделяне на атрибута

2.Избира се "най-добрата" точка на разделяне

3.Информационната печалба за най-добрата точка е информационната печалба за атрибута

Недостатък: по-висока изчислителна сложност

Двоично срещу многопосочно разделяне

- Разделянето по **номинален атрибут** (многопосочно) **изчерпва цялата информация** в този атрибут
- Номинален атрибут се **тества най-много веднъж** по всяка пътека в дървото
- **Не е така за двоично разделяне на числови атрибути!**
- Числов атрибут може да се **тества няколко пъти** по една и съща пътека в дървото
- **Недостатък:** дървото става трудно за четене
- **Решения:**
    - Предварително **дискретизиране** на числовите атрибути
    - Използване на **многопосочни** вместо **двоични** разделения

Двоично срещу многопосочно разделяне: Разделянето на атрибутите в дървото може да се извърши по два начина:

-Двоично разделяне (Binary Split) – разделя стойностите на две групи (например Температура < 30 и Температура ≥ 30).

-Многопосочно разделяне (Multi-way Split) – разделя на повече от две групи наведнъж (например: Температура < 25, 25 ≤ Температура < 35, Температура ≥ 35).

- Разделянето по **номинален атрибут** (многопосочно) **изчерпва цялата информация** в този атрибут
- Номинален атрибут се **тества най-много веднъж** по всяка пътека в дървото
- **Не е така за двоично разделяне на числови атрибути!**
- Числов атрибут може да се **тества няколко пъти** по една и съща пътека в дървото
    - **Недостатък:** дървото става трудно за четене **Решения:** Предварително **дискретизиране** (групиране на стойности в интервали) на числовите атрибути; Използване на **многопосочни** вместо **двоични** разделения

Изчисляване на многопосочни разделения

- **Прост и ефективен начин** за генериране на многопосочни разделения: **алчни (greedy) алгоритми. Но:** Динамичното програмиране може да намери **оптималното** многопосочно разделение за **O(n²)** време- Целта е да намерим най-доброто разделение на данните на множество подгрупи (интервали), така че тези подгрупи да са възможно най-чисти, т.е. да имат минимално "нечистота" или impurity.
- **Формули:**
    - imp(k,i,j) е **нечистотата** (impurity) на **най-доброто разделение** на стойностите от x_i до x_j в k под-интервала
    - imp(k,1,i)=min0<j<i(imp(k–1,1,j)+imp(1,j+1,i))

- imp(k,1,N) ни дава **най-доброто** k-пътно разделение

  imp(k,1,i)=min(imp(k-1,1,j)+imp(1,j+1,i))

- **На практика: Алчният алгоритъм** работи **почти толкова добре**, колкото динамичното/ оптимално програмиране.

Обработка на липсващи стойности: C4.5 използва **метод на фракционни инстанции**: Разделя екземплярите с липсващи стойности на части. Част от екземпляра, преминаваща по даден клон, получава тежест, пропорционална на популярността на този клон.Тежестите се сумират до 1. **Info gain** (приход от информация) работи с фракционни екземпляри. Използват се суми от тежести вместо броячи. По време на класификацията, екземплярите се разделят на части по същия начин. Обединяване на разпределението на вероятностите, използвайки тежестите

Gain (или information gain) - мярка за **намаляване на несигурността или хаоса** в данни след определено разделение. Показва колко информация е "спечелена" след като разбием набора от данни на подгрупи по даден атрибут. Основна концепция в изграждането на дървета за решения, която ни помага да изберем най-добрия атрибут за разделение, като измерва колко информация се спечелва след разделянето. Това е основата на повечето алгоритми за вземане на решения, като ID3, C4.5 и други.

Ентропия- мярка **за хаоса или несигурността** в набора от данни. Ако целият набор от данни съдържа само един клас (например само "Да" или само "Не"), ентропията е нула, защото няма несигурност.

Pruning (рязане на дървото): Основен проблем: При пълното разрастване на дървото моделът може да стане прекалено сложен и да запомни шума в данните, вместо да извлича истинските закономерности (overfitting).За да се избегне - два метода:

1. **Postpruning** (следобработващо рязане)
    - Първо изграждаме пълното дърво.
    - След това премахваме ненадеждните разклонения.
    - По-често използвано, защото намалява риска от спиране твърде рано.
2. **Prepruning** (предварително рязане)
    - Спираме разклоненията, ако няма статистически значима връзка между атрибута и класа.
    - Използва се χ^2 -квадрат тест за значимост.
    - ID3 също използва тази техника.

Разбиране на Early Stopping и Свързаните Концепции

Early Stopping (Ранно спиране) - Early stopping е техника, при която Pre-pruning може да прекъсне процеса на растеж на дървото твърде рано. Това е проблем, особено в случаи като XOR (Parity problem), където няма индивидуален атрибут, който да е значимо свързан с класа. Истинската структура се проявява едва когато дървото е напълно развито, след което може да бъде подрязано (pruned).

• Но: Проблеми от типа XOR са редки в практиката И:
Предпринудителното спиране е по-бързо от постпринудителното

Постпринудително отсичане: Първо, изградете пълното дърво. След това, отсечете го. Напълно израслото дърво показва всички взаимодействия между атрибутите. Проблем: Някои поддървета може да са резултат от случайни ефекти. **Заместване на поддърво** и **Повдигане на поддърво** са две основни операции, които се използват в процеса на постпринудително отсичане (postpruning) на

решаващо дърво. Те имат за цел да премахнат ненадеждните или незначителни части от дървото, които могат да водят до пренасочване или прекомерно приспособяване към тренировъчните данни. **Две операции за отсичане:**

-Заместване на поддърво: Тази операция замества дадено поддърво с листен възел. Листният възел съдържа класовата стойност, която е най-често срещана в поддървото.

-Повдигане на поддърво: Повдигането на поддърво е операция, при която поддървото се "премества" на по-високо ниво в дървото, като става част от родителския възел. Тази операция може да се извърши, когато установим, че разделението в поддървото не е значимо, или когато поддървото само добавя шум в дървото.

• Възможни стратегии: Оценка на грешката; Тест за значимост; Принцип на MDL (Минимална описание на дължината)

Два основни вида подрязване (Pruning):Замяна на поддърво; Издигане на поддърво

Възможни стратегии за подрязване: Error estimation (Оценка на грешката); Significance testing (Статистическа значимост); MDL principle (Минимална дължина на описанието)

Издигане на поддърво и Оценка на Грешките

- Подрязваме само ако това не увеличава очакваната грешка.
- Грешката върху тренировъчните данни не е добър показател (би довело до минимално подрязване).
- Възможно решение: използване на hold-out set (задържан набор от данни), което води до reduced-error pruning.

Методът на C4.5 за Подрязване

- Използва доверителен интервал от тренировъчните данни.
- Прилага Bernoulli-процес за определяне на граници за подрязване.
- Изчислява очакваната грешка за всеки възел въз основа на грешките в неговите листа.

Сложност на Индукция на Дървото

Приемаме: m атрибута; n тренировъчни примера; Дълбочина на дървото: O(log n)

Операции и тяхната сложност:

- Изграждане на дърво – O(m * n log n)
- Subtree replacement – O(n)
- Subtree raising – O(n * (log n)²)

От Дървета към Правила-Един прост начин за конвертиране на дърво в правила е едно правило на листо.

Класификационни правила (Classification Rules)- метод за представяне на модели за машинно обучение, като вместо дърво, резултатът е набор от "ако-тогава" правила. "**Ако температурата е над 20°C и няма вятър, тогава човек ще спортува.**"

PRISM – алгоритъм за генериране на точни правила чрез максимално увеличаване на информационната печалба.

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) – по-ефективен алгоритъм, който орязва излишните условия и работи добре с големи и шумни данни.

PART – съчетава дървета за решения и правила за по-ефективна класификация.

C4.5rules използва жадно подрязване на условията, за да намали грешката.

Асоциативни правила (Association Rules) - се използват за намиране на зависимости между елементи в големи набори от данни. Те често се прилагат в анализ на пазарни кошници (Market Basket Analysis). Честите шаблони (frequent-pattern trees, FP-trees) позволяват по-бързо извличане на асоциативни правила чрез компактно представяне на честите комбинации. "**Ако клиент купи хляб, то с вероятност 70% ще купи и масло.**"

Процедура за Избор на Правила

1. Разглеждаме всички правила за даден клас.
2. Избираме „добро“ подмножество от тях (ръководено от MDL).
3. Подреждаме тези подмножества, за да избегнем конфликти.
4. Премахваме ненужни правила, ако това намалява грешката.

Недостатък: C4.5rules е бавен за големи и шумни данни;
Наследникът му C5.0 rules е по-бърз и малко по-точен.

Cost-Complexity Pruning (Подрязване - база сложност на разходите)

- Методът на C4.5 понякога не подрязва достатъчно.
- Размерът на дървото продължава да расте, дори ако представянето върху независими данни не се подобрява.
- CART (Classification and Regression Trees) използва Cost-Complexity Pruning, което прилага кръстосана проверка или hold-out set за избор на подходящ размер на крайното дърво.

TDIDT (Top-Down Induction of Decision Trees) -Най-широко изследваният метод за машинно обучение в сферата на data mining. Различните критерии за избор на атрибут обикновено не водят до големи разлики в резултатите. Различните методи за подрязване основно променят размера на крайното дърво.