

Дерева рішень. Бектрекінг (пошук із поверненнями). Каркаси (з'єднувальні дерева).

Дерева рішень.

Дерева рішень є найдавнішим алгоритмом аналізу даних. Роботи в цьому напрямку розчали Ховленд (Hoveland) та Хант (Hunt) у 1950-х роках.

Дерева рішень – це спосіб представлення правил в ієрархічно послідовній структурі, де кожному об'єкту відповідає лише єдиний кінцевий вузол, що надає відповідь.

Дерево рішень використовують як математичну модель у задачах класифікації та прогнозування. До них відносяться проблеми медичного діагностування, оцінювання кредитного ризику, визначення тенденцій на фінансових ринках тощо.

Наведемо важливі терміни:

- Об'єкт – Приклад, шаблон, спостереження
- Атрибут – Властивість, ознака, незалежна змінна
- Мітка класу – Залежна змінна, цільова змінна, ознака, що визначає клас об'єкту.
- Вузол – Внутрішній вузол дерева, вузол перевірки.
- Лист – Кінцевий вузол дерева, вузол відповіді.
- Перевірка – умова у вузлі.

Етапи побудови дерев рішень:

При побудові дерев рішень особлива увага приділяється наступним питанням: вибір критерію атрибуту, за яким відбувається розбиття, зупинка навчання і відсікання гілок. Розглянемо всі ці питання.

Правило розбиття

Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову (перевірку), яка б розбивала множину, асоційовану з цим вузлом на підмножини. В якості такої перевірки повинен бути вибраний один з атрибутів. Загальне правило для вибору атрибута: обраний атрибут повинен розбити множину так, щоб одержані в результаті підмножини склалися з об'єктів, які належать до одного класу, або були максимально наближені до цього, тобто кількість об'єктів з інших класів ("домішків") в кожній з цих множин було якомога менше.

Зупинка навчання

Подальша побудова дерева зупиняється, якщо глибина дерева перевищує задане значення.

Для оцінки доцільності подальшого розбиття можна використати "ранню зупинку". Вона приваблива в плані економії часу навчання, але цей підхід надає менш точні класифікаційні моделі і тому є небажаним.

Правило відсікання

Під точністю (розпізнавання) дерева рішень розуміють відношення правильно класифікованих об'єктів при навчанні до загальної кількості об'єктів з навчальної множини, а під похибкою - кількість неправильно класифікованих. Припустимо, що нам відомо спосіб оцінки похибки дерева, гілок і листя. Тоді, можна використати просте правило:

- Побудувати дерево;

- Відсікти або замінити піддеревом ті гілки, які призводять до зростання помилки.

На відміну від процесу побудови, відсікання гілок відбувається знизу вгору, рухаючись з листя дерева, відзначаючи вузли як листя, або замінюючи їх на піддерева. В більшості практичних завдань відсікання надає добрі результати.

Правила:

Іноді усічені дерева можуть бути складними для сприйняття. В такому випадку, можна вдаватися до методики видобутку правил з дерева з наступним створенням наборів правил, що описують класи.

Для видобутку правил необхідно дослідити всі шляхи від кореня до кожного листа дерева. Кожен такий шлях надасть правило, де умовами будуть перевірки з вузлів, які зустрілися на шляху.

Для прикладу розглянемо задачу оцінювання ризику при кредитуванні фізичних осіб, тобто визначення кредитоспроможності потенційного клієнта.

Існує багато переваг дерева рішень серед яких:

- Швидкий процес навчання
- Генерація правил в тих областях, де експерту складно формалізувати свої знання
- Правила формуються на природній мові
- Інтуїтивно зрозуміла модель
- Висока точність прогнозу
- Дерева рішень є самоадаптованими моделями, де втручання людини є мінімальним
- Висока якість результату досягається за рахунок визначення значущих факторів для вибору відповіді.
- Отриманий результат є статистично обґрунтованим.

Завдання, які вирішуються можна об'єднати в наступні три класи:

- Опис даних. Дерева рішень дозволяють зберігати інформацію про дані в компактній формі з точним описом об'єктів.
- Класифікація. Дерева рішень добре справляються із завданнями класифікації - віднесення об'єктів до одного з заздалегідь відомих класів. Цільова змінна повинна мати дискретні значення.
- Регресія. Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, до цього класу відносяться задачі чисельного прогнозування (передбачення значень цільової змінної).

Бектрекінг (пошук із поверненнями).

Щоб застосувати цей метод, розв'язок задачі повинен мати вигляд скінченої послідовності (x_1, \dots, x_n) . Головна ідея методу полягає в тому, що розв'язок будують поступово, почи-

наючи з порожньої послідовності \emptyset (довжиною 0). Загалом, якщо є частковий (неповний) розв'язок (x_1, \dots, x_i) , де $i < n$, то намагаємося знайти таке допустиме значення x_{i+1} , що можна продовжувати $(x_1, \dots, x_i, x_{i+1})$ до одержання повного розв'язку. Якщо таке допустиме, але ще не використане значення x_{i+1} існує, то долучаємо цю нову компоненту до часткового розв'язку та продовжуємо процес для послідовності $(x_1, \dots, x_i, x_{i+1})$. Якщо такого значення x_{i+1} немає, то повертаємося до попередньої послідовності (x_1, \dots, x_{i-1}) і продовжуємо процес, шукаючи нове, іще не використане значення x_i' . Тому цей процес називають *бектрекінг* (англ. *backtracking* – пошук із поверненнями).

Метод використання:

Опишемо загальний метод, який дає змогу значно зменшити обсяг обчислювань в алгоритмах типу повного перебору. Щоб застосувати цей метод, розв'язок задачі повинен мати вигляд скінченної послідовності (x_1, \dots, x_n) .

Ідея методу:

Головна ідея методу полягає в тому, що розв'язок будують поступово, починаючи з порожньої послідовності \emptyset (довжиною 0). Загалом, якщо є частковий (неповний) розв'язок (x_1, \dots, x_i) , де $i < n$, то намагаємося знайти таке допустиме значення x_{i+1} , що можна продовжувати $(x_1, \dots, x_i, x_{i+1})$ до отримання повного розв'язку.

Якщо таке припустиме, але ще не використане значення x_{i+1} існує, то долучаємо цю нову компоненту до часткового розв'язку і продовжуємо процес для послідовності $(x_1, \dots, x_i, x_{i+1})$.

Якщо такого значення x_{i+1} немає, то повертаємося до попередньої послідовності (x_1, \dots, x_{i-1}) і продовжуємо процес, шукаючи нове, ще не використане значення x_i' . Тому цей процес називають *бектрекінг* (англ. *backtracking* – пошук із поверненнями).

Алгоритм пошуку:

Алгоритм пошуку з поверненням перераховує множину неповних кандидатів що, в принципі, можуть бути доповнені кількома шляхами для отримання всіх можливих розв'язків даної задачі. Доповнення будується покроково, послідовністю кроків розширення кандидата.

Зрозуміло, що неповні кандидати є вузлами деревоподібної структури, потенційне дерево пошуку. Кожний неповний кандидат є батьком кандидатів відмінних від нього на один крок розширення; листями дерева є кандидати які не можуть бути розширені далі.

Алгоритм пошуку з вертанням обходить це дерево пошуку рекурсивно, від кореня донизу, пошуку в глибину. В кожному вузлі C , алгоритм перевіряє чи може C бути доповнене до вірного розв'язку. Якщо ні, ціле піддерево з корнем в C пропускається (обрізається).

Інакше, алгоритм перевіряє чи є C вірним розв'язком, і якщо так повідомляє про це користувача; і рекурсивно обходить усі піддерева. Обидва тести і дочірні вузли кожного вузла визначаються за допомогою поданих користувачем процедур.

Внаслідок цього, актуальне дерево пошуку яке обходить алгоритм становить лише частину потенційного дерева. Загальна ціна алгоритму це кількість вузлів актуального дерева помножена на вартість отримання і обробки кожного вузла. Цей факт має бути врахованим коли обирається потенційне дерево і реалізується тест на обрізання.

Роботу цього алгоритму можна інтерпретувати як процес обходу деякого дерева. Кожна його вершина відповідає деякій послідовності (x_1, \dots, x_i) , причому вершини, які відповідають послідовностям вигляду (x_1, \dots, x_i, y) , – сини цієї вершини. Корінь дерева

відповідає порожній послідовності.

Правило руки:

Обхід усіх гілок можна здійснити, наприклад, за правилом правої або лівої руки. Це правило визначає гілку, по якій потрібно йти на черговому кроці пошуку. Можна використовувати певне правило для даної схеми пошуку.

Нехай на деякому кроці роботи алгоритму ми знаходимося в деякій вершини дерева і необхідно прийняти рішення про те по якій гілці йти далі. Врахуємо, що з кожної вершини довільну кількість гілок йде вниз і тільки одна наверх. Можливі наступні ситуації:

Усі гілки йдуть вниз вже пройдені. Фізично це може визначатися позначкою установлюваної на гілки в тому випадку якщо по ній здійснюється повернення. Тоді необхідно йти по гілці йде вгору і помітити її як пройдену.

Серед гілок провідних вниз є не пройдені. Знайдемо серед них саму ліву і підемо по ній.

Сформульоване правило ніяк не враховує події відбуваються в вершинах дерева. Тим часом вершина від вершини може відрізнятись і не тільки становищем в дереві. Наприклад, в розглянутій вище задачі при переході вниз наростає сума, а при переході вгору та сума зменшується.

Таким чином, існує клас задач, для яких до дерева комбінацій даних може бути прив'язана деяка величина змінюється закономірним чином. Звичайно, це не обов'язково збільшення. Спробуємо описати поведінку цієї величини в загальному вигляді. Назвемо її характеристикою дерева.

Зміна властивостей:

Характеристика змінюється всередині деякого числового інтервалу.

Ця зміна має властивість монотонності при русі по дереву вниз.

Існує критичне значення (ліва або права межа інтервалу), таке, що якщо характеристика досягає цього критичного значення, то подальший пошук рішення втрачає сенс.

З урахуванням такої характеристики описане вище правило обходу дерева трохи змінюється і тепер виглядає ось так:

Усі гілки йдуть вниз вже пройдені. Тоді необхідно йти по гілці йде вгору і позначити її як пройдену.

Серед гілок провідних вниз є не пройдені, але характеристика досягла своєї критичної величини. Тоді необхідно йти по гілці йде вгору і позначити її як пройдену.

Серед гілок провідних вниз є не пройдені і характеристика не досягла критичного значення. Знайдемо серед них саму ліву і підемо по ній.

Узагальнення методу Бектрекінга:

Обхід дерева комбінацій відповідно до описаного вище правилом і є метод бектрекінга (BackTracking - зворотний хід). Це правило досить загальне і під нього підходить досить багато завдань, але все-таки це не сама загальне формулювання. Думаю, існує багато можливостей розширити правило.

Каркаси (з'єднувальні дерева).

Каркас (з'єднувальне дерево) – називають підграф просто зв'язного графа, який являє собою дерево та містить усі вершини цього графа. Взагалі, кістякове дерево складається з

деякої підмножини ребер графа, таких, що рухаючись цими ребрами можна з будь-якої вершини графа потрапити до будь-якої іншої.

Будь-яке каркасне дерево у графі з n вершинами містить рівно $n - 1$ ребер. Кількість кістякових дерев у повному графі з n вершинами подається відомою формулою Келі: $n^{(n-2)}$

Для того щоб отримати каркас графа G , який має n вершин і m ребер можна використати процедуру вилучення ребер, які належать простим циклам. Потрібно вилучити ребра $Y(G) = m - (n - 1) = m - n + 1$, де $Y(G)$ — цикломатичне число графа G . Цикломатичне число — це числова характеристика зв'язності графа. $Y(G) \geq 0$ оскільки, якщо граф G має n вершин та k компонентів, то кількість m його ребер задовольняє нерівність $n - k \leq m \leq 1/2 (n - k)(n - k + 1)$, а цикломатичне число дерева дорівнює 0.

Побудова каркасу

Алгоритм, в основі якого лежить вилучення ребер із простих циклів є неефективний для комп'ютерної реалізації, оскільки для його виконання потрібно ідентифікувати прості цикли, а це складна задача.

З точки зору комп'ютерної реалізації, ефективним алгоритмом побудови каркасів є алгоритм послідовного добору ребер у каркас. Цю дію можна виконувати як і за допомогою пошуку углиб так і за пошуком ушир. Воно складається з усіх пар ребер (u, v) , таких, що алгоритм, переглядаючи вершину u виявляє в її списку суміжності нову, невиявлену раніше вершину v . Кістякове дерево, побудоване обходом графа починаючи з вершини s за алгоритмом Дейкстри, має властивість, що найкоротший шлях у графі з вершини s до будь-якої іншої вершини — це шлях з s до цієї вершини в побудованому дереві.

Теорема : Нехай T каркас графа G , побудований пошуком ушир, починаючи з вершини a . Тоді шлях з a до довільної вершини v в T — найкоротший шлях з a до v в графі G .

Зауваження: Використовуючи для побудови найкоротшого шляху від вершини a алгоритмом Дейкстри (припускаючи, що довжина кожного ребра дорівнює 1), одержуємо те саме дерево, що й за алгоритмом вшир. Але складність першого алгоритму становить $O(n^2)$ або $O(m \log n)$ в залежності від подання графа, а складність другого алгоритму (вшир), $O(m + n)$ де n — кількість вершин, m — кількість ребер у графа G . Алгоритм Дейкстри є більш універсальний, він придатний для всіх додатніх довжин графу, а не тільки 1.

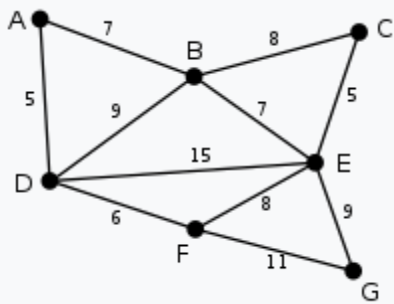
Алгоритм Крускала

Візьмемо зв'язаний зв'язний граф $G = (V, E)$, де V — множина вершин, E — множина ребер, для кожного з яких задано вагу. Мінімальним каркасним (або кістяковим) деревом називають ациклічну множину ребер, що поєднують усі вершини графа і чия загальна вага мінімальна.

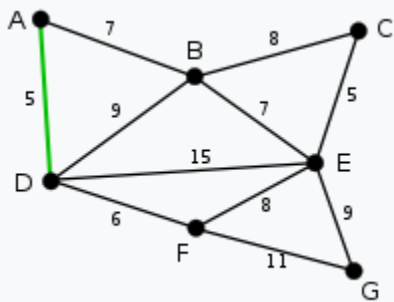
Алгоритм Крускала слід почати з побудови виродженого лісу, який містить V дерев, кожне з яких складається з однієї вершини. Далі потрібно виконати операції об'єднання двох дерев, для цього використовують найкращі можливі ребра, поки не утвориться єдине дерево.

Це дерево і буде мінімальним кістяковим деревом.

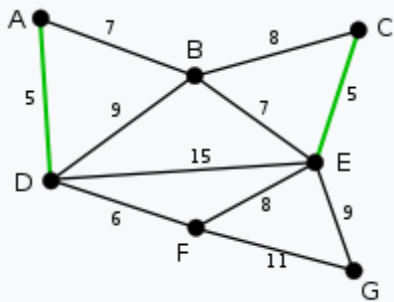
Алгоритм Крускала є класичним алгоритмом розв'язання задачі пошуку мінімального кістякового дерева.



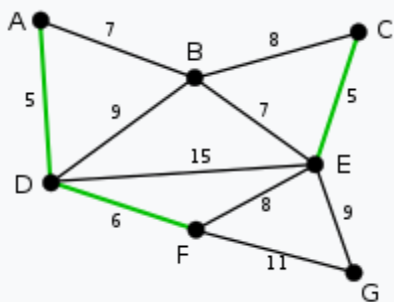
Початковий граф. Цифри над ребрами позначають їх вагу. Жодне з ребер не додане до кістякового дерева.



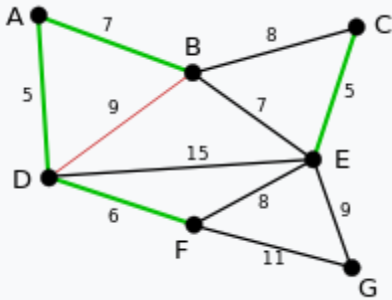
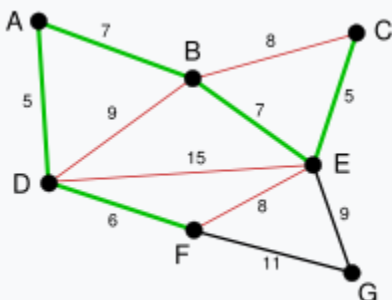
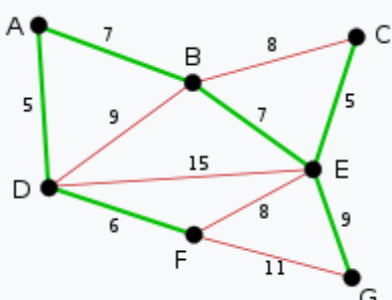
AD і **CE** мають найменшу вагу 5, і **AD** вибирається з них довільно та додається до кістякового дерева.



На цьому кроці **CE** є найлегшим ребром з вагою 5, тому воно також додається до дерева.



Аналогічним чином обирається найлегше з недоданих ребер графа **DF** з вагою 6 і додається до кістякового дерева.

	<p>Наступними найлегшими ребрами є AB і BE, обидва вагою 7. AB обирається довільно і додається до кістякового дерева. BD фарбується у червоний колір, оскільки воно є частиною циклу ABD.</p>
	<p>Наступним додається ребро BE з вагою 7. Червоним забарвлюємо ребра BC (цикл BCE), DE (цикл DEBA) і FE (цикл FEBAD).</p>
	<p>Додаємо ребро EG вагою 9 і отримуємо мінімальне кістякове дерево.</p>