ЛЕКЦІЯ 3
РЕГРЕСІЯ ТА МАШИННЕ НАВЧАННЯ БЕЗ ВЧИТЕЛЯ

ПЛАН

1. Регресія та оцінка її якості
2. Навчання без вчителя
   2.1. Кластеризація
   2.2. Зменшення розмірності
   2.3. Асоціації

ЛІТЕРАТУРА

Сайт http://www.mmf.lnu.edu.ua/ar/1739

ВСТУП

Згадаємо, що ми вже вивчили із рис.1

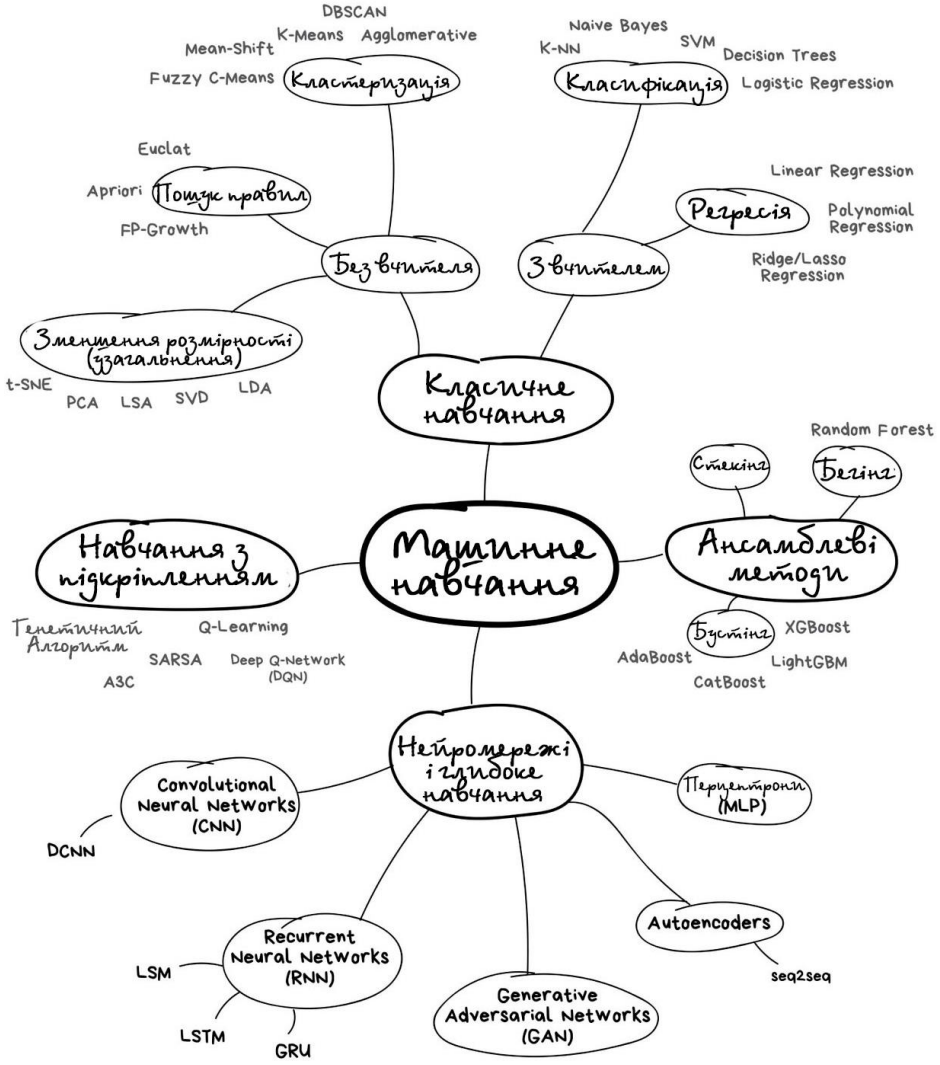
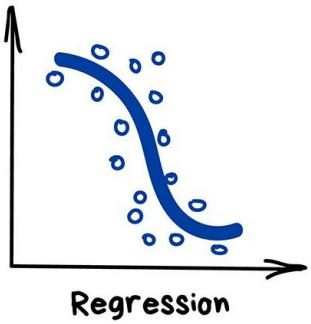


Рис.1.

Рис.2.

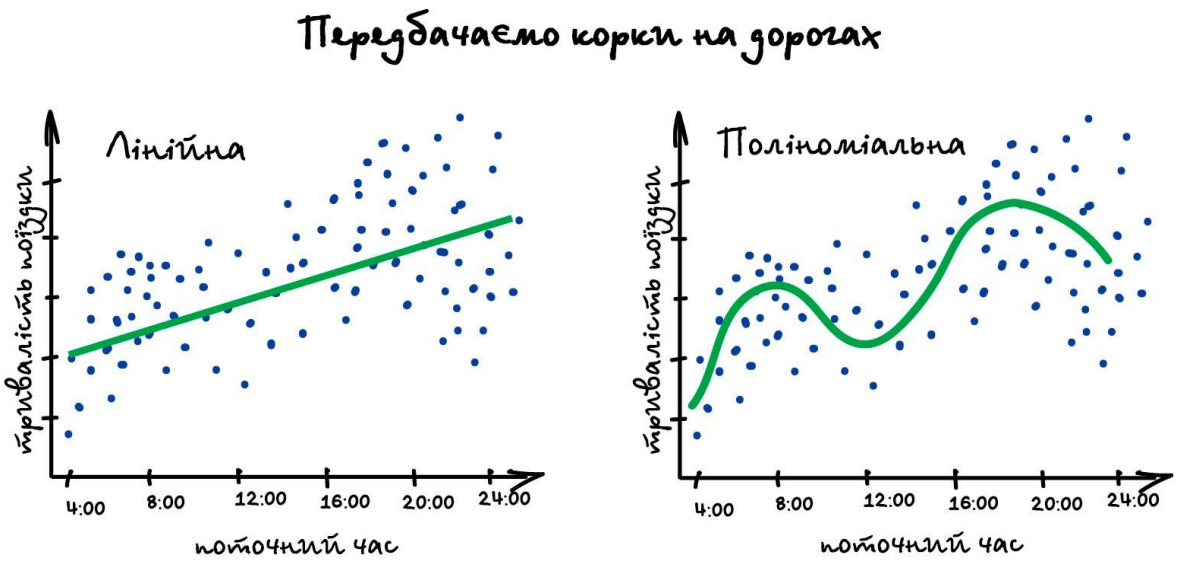
1. РЕГРЕСІЯ ТА ОЦІНКА ЇЇ ЯКОСТІ



«Намалюй лінію уздовж моїх точок. Саме так, це машинне навчання»
Сьогодні використовують для:
Прогноз вартості цінних паперів
Аналіз попиту, обсягу продажів
Медичні діагнози
Будь-які залежності числа від часу
Популярні алгоритми: Лінійна або Поліноміальна Регресія

Регресія - та ж класифікація, тільки замість категорії ми передбачаємо число. Вартість автомобіля з його пробігом, кількість корків щодо часу доби, обсяг попиту на товар від зростання компанії і.т.д. На регресію ідеально лягають будь-які задачі, де є залежність від часу.

Регресію дуже люблять фінансисти і аналітики, вона вбудована навіть в Excel. Всередині все працює, знову ж таки, банально: машина тупо намагається намалювати лінію, яка в середньому відображає залежність. Правда, на відміну від людини з фломастером і вайтбордом, робить вона це з точністю - обчислюючи середню відстань до кожної точки і намагаючись всім догодити.



Регресія

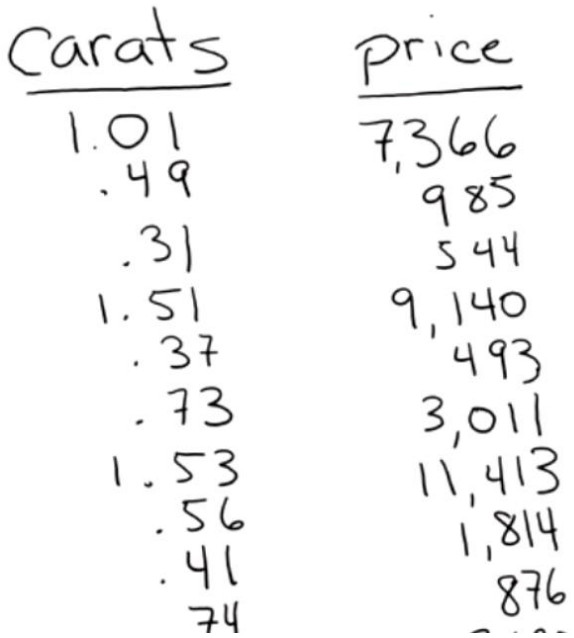
Коли регресія малює пряму лінію, її називають лінійною :))), коли криву - поліноміальною :)))))). Це два основні типи регресії, далі вже починаються рідкоземельні методи. Але так як в сім'ї не без виродка, є *Логістична Регресія*, яка насправді не регресія, а метод класифікації, від чого у всіх постійно плутанина. Не робіть так.

Схожість регресії і класифікації підтверджується ще й тим, що багато хто з класифікаторів, після невеликого тюнінгу, перетворюються в регресорів. Наприклад, ми можемо не просто дивитися до якого класу належить об'єкт, а запам'ятовувати, наскільки він близький - і ось, у нас регресія.

ПРИКЛАД

Припустимо, ми хочемо купити діамант. У нас є кільце, яке належало моєї бабусі. Його оправа містить діамант масою 1,35 карата, і я хочу знати, скільки він буде коштувати. Я беру з собою блокнот і ручку і йду в ювелірний магазин. Там я записую всі ціни на діаманти, які є у вітрині, а також їх масу в каратах. Починаючи з першого діаманта, який має масу 1,01 карата і стоїть 7366 дол. США.

Я йду далі і роблю те ж саме для інших діамантів в магазині.



Стовпці з даними діамантів

Зверніть увагу, що наш список містить два стовпці. Кожен стовпець має свій атрибут - масу в каратах і ціну - і кожен рядок є окремим точкою даних, що представляє один діамант.

Фактично ми створили невеликий набір даних у формі таблиці. Зверніть увагу, що він відповідає нашим критеріям якості.

Дані є релевантними: маса безумовно пов'язана з ціною.

Вони точні: ми перевірили ціни, які записали.

Вони пов'язані: всі стовпці і рядки заповнені.

І, як ми побачимо далі, цих даних достатньо, щоб дати відповідь на наше питання.

Постановка точного питання

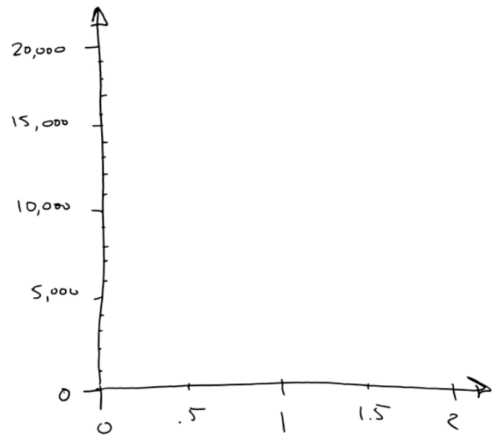
Тепер давайте точно сформулюємо наше запитання: "Скільки буде коштувати діамант масою 1,35 карата?"

У нашому списку немає діаманта масою 1,35 карата, тому необхідно використовувати наявні дані для отримання відповіді на це питання.

Побудова існуючих даних

Перше, що необхідно зробити, це намалювати горизонтальну числову лінію, яку називають віссю. На ній буде відображатися маса. Діапазон мас - від 0 до 2, тому ми намалюємо лінію, що охоплює цей діапазон, і помістимо на неї позначки для кожних 0,5 карата.

Потім ми намалюємо вертикальну вісь, на якій запишемо ціни, і з'єднаємо її з горизонтальною віссю маси. Одиницею виміру для цієї осі буде долар США. Тепер у нас є набір осей координат.

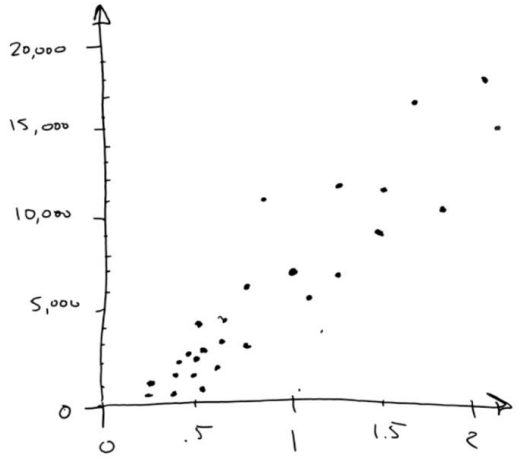


Осі маси і ціни

Тепер ми перетворимо ці дані в точкову діаграму. Це відмінний спосіб візуалізації числових наборів даних.

Беремо першу точку даних і проводимо вертикальну лінію від позначки 1,01 карата. Потім проводимо горизонтальну лінію від позначки 7366 дол. США. Там, де ці лінії перетинаються, малюємо точку. Ця точка і представляє наш перший діамант.

Тепер ми пройдемо по всім діамантів з нашого списку і зробимо те ж саме. В результаті ми отримаємо таку картину: безліч точок, кожна з яких відповідає одному діаманту.

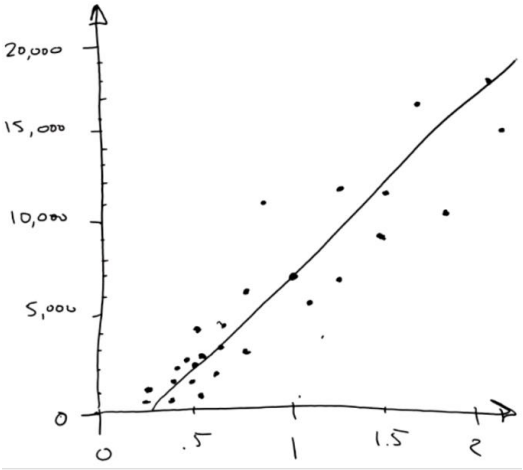


точкова діаграма

Побудова моделі на основі точок даних

Тепер, якщо поглянути на точки під невеликим кутом, то весь цей набір виглядає як товста і нечітка лінія. Можна скористатися маркером і провести через набір точок пряму лінію.

Намалювавши лінію, ми створили модель. Щоб краще зрозуміти, уявіть собі, що реальний світ зображений у спрощеній формі, як комікс. Комікс не відображає всієї дійсності: лінія не проходить через всі крапки даних. Але це корисне спрощення.



Лінія лінійної регресії

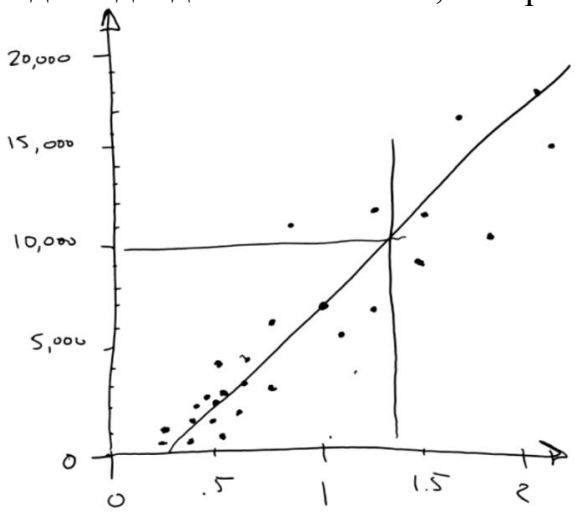
Той факт, що лінія не проходить через всі крапки, є нормальним. Фахівці з обробки даних пояснюють це так: існує модель (в нашому випадку - лінія), і кожна точка в моделі піддається впливу деякого шуму або відхилення, пов'язаного з нею. Є базовий функціональний зв'язок, а є мінливий реальний світ, який додає шум і невизначеність.

Так як ми намагаємося відповісти на питання **Скільки?** , Це називається регресією. І оскільки ми використовуємо пряму лінію, це - лінійна регресія.

Використання моделі для пошуку відповіді

Тепер у нас є модель і ми можемо поставити їй наше запитання: "Скільки буде коштувати діамант масою 1,35 карата?"

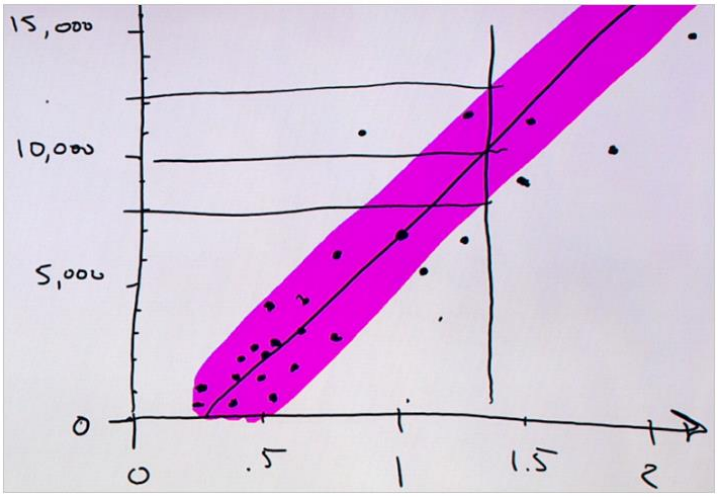
Для відповіді на питання ми проводимо вертикальну лінію від позначки 1,35 карата. Від точки її перетину з лінією моделі проводимо горизонтальну лінію до осі цін. Вона приводить нас до позначки 10 000. Ось так! Це і є відповідь: діамант масою 1,35 карата коштує близько 10 000 доларів США.



Пошук відповіді за допомогою моделі

Створення довірчого інтервалу

Було б логічним перевірити, наскільки точний цей прогноз. Корисно знати, чи буде діамант масою 1,35 карата коштувати рівно 10 000 доларів США, а може бути набагато більше або менше. Щоб це визначити, давайте намалюємо навколо лінії регресії так званий конверт, який буде включати в себе більшість точок. Цей конверт - наш довірчий інтервал. Ми можемо бути впевнені, що ціни потраплять в зазначений діапазон, адже це справедливо для більшості цін в минулому. Можна провести ще дві горизонтальні лінії від точок перетину лінії, проведеної від позначки 1,35 карата, з верхньою і нижньою межами даного конверта.



довірчі інтервали

Тепер ми можемо щось сказати про наш довірчий інтервал. Можна з упевненістю говорити, що ціна діаманта 1,35 карата складе близько 10 000 дол. США, при цьому вона може бути не нижче 8 000 і не вище 12 000 долл. США.

Тобто:

Ми поставили запитання, на яке можна відповісти за допомогою даних.

Ми створили модель, використовуючи лінійну регресію.

Ми зробили прогноз, доповнений довірчим інтервалом.

При цьому ми не користувалися математичними формулами або комп'ютерами.

Якби у нас були додаткові відомості, такі як:

огранювання діаманта;

відтінки кольорів (наскільки близький колір діаманта до білого);

наявність в камені сторонніх включень;

то ми б мали додаткові стовпці. В цьому випадку математика вже буде корисною. Коли є більше двох стовпців, важко намалювати точки на папері. Математика дозволить відмінно вписати отриману лінію або площину в ваші дані.

Крім того, якби замість невеликого переліку діамантів у нас було б дві тисячі або два мільйони каменів, то набагато швидше цю роботу можна було б зробити за допомогою комп'ютера.

Сьогодні ми поговоримо про те, як створити лінійну регресію, а також як зробити прогноз, використовуючи дані, більш детально.

Як будується модель

Лінійна регресійна модель має наступний вигляд:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u \qquad (2.1)$$

де y– залежна змінна;

$x_1, x_2, \ldots, x_n$– незалежні змінні;

u– випадкова похибка, розподіл якої в загальному випадку залежить від незалежних змінних, але математичне очікування якої рівне нулю.

Згідно з моделлю (2.1), математичне очікування залежної змінної є лінійною функцією незалежних змінних:

$$E(y) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u \qquad (2.2)$$

Вектор параметрів $\beta_0, \beta_1, \ldots, \beta_k$ є невідомим, і задача лінійної регресії полягає в оцінці цих параметрів на основі деяких експериментальних значень y і $x_1, x_2, \ldots, x_n$. Тобто, для деяких n експериментів є відомі значення $\{y_i, y_{i1}, \ldots, y_{ip}\}_{i=1}^{n}$ незалежних змінних і відповідне їм значення залежної змінної.

Згідно з визначенням моделі для кожного експериментального випадку залежність між змінними визначається формулою:

$$y_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + u \qquad (2.3)$$

На основі цих даних потрібно оцінити значення параметрів $(\beta_0, \beta_1, \ldots, \beta_k)$, а також розподіл випадкової величини u. Зважаючи на характеристики досліджуваних змінних, можуть додаватися різні додаткові специфікації моделі і застосовуватися різні методи оцінки параметрів. Серед найпоширеніших специфікацій лінійних моделей є класична модель лінійної регресії та узагальнена модель лінійної регресії.

Згідно з класичною моделлю лінійної регресії додатково вводяться такі вимоги щодо специфікації моделі та відомих експериментальних даних:

$\forall i \neq j E(u_i u_j | x_i) = 0$ (відсутність кореляції залишків);

$\forall i E(u_j^2 | x_i) = \sigma^2$ (гомоскедастичність).

Часто додається також умова нормальності випадкових відхилень, яка дозволяє провести значно ширший аналіз оцінок параметрів та їх значимості, хоча і не є обов'язковою для можливості використання наприклад методу найменших квадратів $(u_j | x_i) \sim N(0, \sigma^2)$.

Передбачимо, що незалежна змінна набула значень $x_1, x_2 \dots, x_n$, внаслідок чого залежна змінна набула значень $y_1, y_2 \dots, y_n$. У припущенні лінійної залежності отримуємо n рівностей.

$$y_i = a_0 + a_1 x_i + \varepsilon_i, i = 1 \dots n, \qquad (2.4)$$

де ε_i– незалежні і розподілені так само, як ε.

Потрібно за значеннями пар (x_i, y_i) оцінити невідомі a_0, a_1. Як ми вже знаємо, кожне завдання оцінювання пов'язане з деяким критерієм якості. У теорії, що викладається нами, таким критерієм є критерій найменших квадратів:

$$\sum_{i=1}^{n} \varepsilon_i^2 - \min. \qquad (2.5)$$

Запишемо цю суму інакше, так, щоб була помітна залежність від a_0, a_1:

$$\sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} [\bar{y}(x) - y_i]^2 = \sum_{i=1}^{n} [y_i - a_0 - a_1 x_1]^2 \qquad (2.6)$$

Тепер остаточно приходимо до такої задачі: знайти такі значення невідомих параметрів a_0, a_1щоб функція 2.11 набула найменшого значення:

$$Q(a_0 a_1) = \sum_{i=1}^{n} [y_i - a_0 - a_1 x_1]^2 \qquad (2.7)$$

Метод розв'язання цієї задачі відомий з курсу вищої математики. Знаходимо часткові похідні функції Q і прирівнюємо їх до нуля, внаслідок чого приходимо до системи лінійних рівнянь:

$$\begin{cases} \frac{\partial Q}{\partial a_0} = -2 \sum_{i=1}^{n} [y_i - a_0 - a_1 x_i] = 0, \\ \frac{\partial Q}{\partial a_1} = -2 \sum_{i=1}^{n} [y_i - a_0 - a_1 x_i] x_i = 0. \end{cases} \qquad (2.8)$$

Після очевидних перетворень отримуємо систему:

$$\begin{cases} n a_0 + a_1 \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_{i,} \\ a_0 \sum_{i=1}^{n} x_i + a_i \sum_{i=1}^{n} {x_i}^2 = \sum_{i=1}^{n} x_i y_i \end{cases} \qquad (2.9)$$

Позначимо вибіркові середні:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i\,, \overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i\, y_i, \overline{x}^2 = \frac{1}{n}\sum_{i=1}^{n} {x_i}^2. \qquad (2.10)$$

У цих позначеннях після ділення кожного рівняння системи на n вона набуде вигляду:

$$\begin{cases} a_0 + a_1\overline{x} = \overline{y}, \\ a_x\overline{x} + a_1\overline{x}^2 = \overline{xv} \end{cases} \qquad (2.11)$$

а її рішення (шукані оцінки коефіцієнтів рівняння регресії) буде таким:

$$\hat{a}_0 = \frac{\overline{x^2}\cdot\overline{y} - \overline{xy}\cdot\overline{x}}{\overline{x^2} - (\overline{x})^2} \qquad (2.12)$$
$$\hat{a}_1 = \frac{\overline{xy} - \overline{y}\cdot\overline{x}}{\overline{x^2} - (\overline{x})^2}$$

Якщо ввести ще позначення $S_x^2 = \overline{x^2} - \overline{(x)}^2$ і перетворити вираз $\hat{a}_0$:

$$\hat{a}_0 = \frac{\overline{x^2}\cdot\overline{y} - \overline{xy}\cdot\overline{x} \pm \overline{y}\cdot(\overline{x})^2}{S_{..}^2} = \frac{S_x^2\cdot\overline{y} - \overline{x}(\overline{xy} - \overline{y}\cdot\overline{x})}{S_{..}^2} = \overline{y} - \hat{a}_1\cdot\overline{x}, \qquad (2.14)$$

то оцінка функції регресії набуде такого вигляду:

$$\tilde{\overline{y}}(x) = \hat{a}_0 + \hat{a}_1 x = \overline{y} - \hat{a}_1\cdot\overline{x} + \hat{a}\cdot x = \overline{y} + \hat{a}_1(x - \overline{x}) \qquad (2.15)$$

Таким чином, отримали рівняння регресії, що є моделлю для опису даних.

Оцінки якості регресії

Найбільш типовими заходами якості в задачах регресії є

Середня квадратична помилка (англ. Mean Squared Error, MSE)

MSE застосовується в ситуаціях, коли нам треба підкреслити великі помилки і вибрати модель, яка дає менше помилок прогнозу. Грубі помилки стають помітнішими за рахунок того, що помилку прогнозу ми зводимо в квадрсссат. І модель, яка дає нам менше значення середньоквадратичної помилки, можна сказати, що що у цій моделі менше грубих помилок.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(a(x_i) - y_i)^2$$

Середня абсолютна помилка (англ. Mean Absolute Error, MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|a(x_i) - y_i|$$

Середньоквадратичний функціонал сильніше штрафує за великі відхилення в порівнянні з средньоабсолютним, і тому більш чутливий до

викидів. При використанні будь-якого з цих двох функціоналів може бути корисно проаналізувати, які об'єкти вносять найбільший внесок у загальну помилку - не виключено, що на цих об'єктах була допущена помилка при обчисленні ознак або цільової величини.

Ефективне значення помилки підходить для порівняння двох моделей або для контролю якості під час навчання, але не дозволяє зробити висновків про те, на скільки добре дана модель вирішує завдання. Наприклад, MSE = 10 є дуже поганим показником, якщо цільова змінна приймає значення від 0 до 1, і дуже хорошим, якщо цільова змінна лежить в інтервалі (10000, 100000). У таких ситуаціях замість середньоквадратичної помилки корисно використовувати коефіцієнт детермінації - R^2

Коефіцієнт детермінації

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(a(x_i) - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Коефіцієнт детермінації вимірює частку дисперсії, внесеною моделлю, в загальній дисперсії цільової змінної. Фактично, дана міра якості - це нормована середньоквадратична помилка. Якщо вона близька до одиниці, то модель добре пояснює дані, якщо ж вона близька до нуля, то прогнози можна порівняти за якістю з сталою пророкуванням.

Середня абсолютна процентна помилка (англ. Mean Absolute Percentage Error, MAPE)

$$MAPE = 100\% \times \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - a(x_i)|}{|y_i|}$$

Це коефіцієнт, який не має розмірності, з дуже простою інтерпретацією. Його можна вимірювати в частках або відсотках. Якщо у вас вийшло, наприклад, що MAPE = 11.4%, то це говорить про те, що помилка склала 11,4% від фактичних значень. Основна проблема цієї помилки - нестабільність.

Корінь із середньої квадратичної помилки (англ. Root Mean Squared Error, RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y_i})^2}$$

Приблизно така ж проблема, як і в MAPE: так як кожне відхилення зводиться в квадрат, будь невелике відхилення може значно вплинути на показник помилки. Варто зазначити, що існує також помилка MSE, з якої RMSE якраз і виходить шляхом вилучення кореня.

Симетричная MAPE (англ. Symmetric MAPE, SMAPE)

$$SMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{2\times|y_i - a(x_i)|}{|y_i| + |a(x_i)|}$$

Середня абсолютна масштабована помилка (англ. Mean absolute scaled error, MASE)

$$MASE = \frac{\sum_{i=1}^{n}|Y_i - e_i|}{\frac{n}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

MASE є дуже хорошим варіантом для розрахунку точності, так як сама помилка не залежить від масштабів даних і є симетричною: тобто позитивні і негативні відхилення від факту розглядаються в рівній мірі. *Зверніть увагу, що в MASE ми маємо справу з двома сумами: та, що в чисельнику, відповідає тестової вибірці, та, що в знаменнику - навчальної.* Друга фактично являє собою середню абсолютну помилку прогнозу. Вона ж відповідає середньому абсолютному відхиленню ряду в перших різницях. Ця величина, по суті, показує, наскільки навчальна вибірка передбачувана. Вона може бути дорівнює нулю тільки в тому випадку, коли всі значення в навчальній вибірці рівні один одному, що відповідає відсутності будь-яких змін в ряді даних, ситуації на практиці майже неможливою. Крім того, якщо ряд має тенденцію до зростання або зниження, його перші різниці будуть коливатися близько деякого фіксованого рівня. В результаті цього з різних рядах з різною структурою, знаменники будуть більш-менш порівнянними. Все це, звичайно ж, є очевидними плюсами MASE, так як дозволяє складати різні значення за різними рядах і отримувати несмещённые оцінки.

Недолік MASE в тому, що її важко інтерпретувати. Наприклад, MASE = 1.21 ні про що, по суті, не говорить. Це просто означає, що помилка прогнозу виявилася в 1.21 рази вище середнього абсолютного відхилення ряду в перших різницях, і нічого більше.

Крос-валідація

Хороший спосіб оцінки моделі передбачає застосування крос-валідації (скользящего контролю або перехресної перевірки).

В цьому випадку фіксується деяка множина розбиття вихідної вибірки на дві підвибірки: навчальну і контрольну. Для кожного розбиття виконується настройка алгоритму за навчальною підвибіркою, потім оцінюється його середня помилка на об'єктах контрольної підвибірки. Оцінкою змінного контролю називається середня по всім розбиттям величина помилки на контрольних підвибірках.

2. НАВЧАННЯ БЕЗ ВЧИТЕЛЯ

Навчання без вчителя (Unsupervised Learning) було винайдено пізніше, аж у 90-ті, і на практиці використовується рідше. Але бувають задачі, де у нас просто немає вибору.

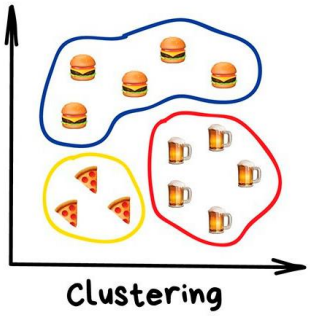
Розмічені дані - дорога рідкість. Але що робити якщо я хочу, наприклад, написати класифікатор автобусів - йти на вулицю, руками фотографувати мільйон Неопланів і підписувати де який? Так і все життя пройде, а у мене ще ігри в танчики не пройдено.

Коли немає міток, є надія на капіталізм, соціальне розшарування і мільйон китайців з сервісів типу Яндекс. Толока, які готові робити для вас що завгодно за п'ять центів. Так зазвичай і роблять на практиці. А ви думали де Яндекс бере більшість своїх датасетів?

Або можна спробувати навчання без учителя. Хоча, відверто кажучи, зі своєї практики я не пам'ятаю, щоб десь воно спрацювало добре.

Навчання без вчителя, все ж, частіше використовують як метод аналізу даних, а не як основний алгоритм. Спеціальний юзер з дипломом котрогось НУ вкидає туди купу сміття і спостерігає. Кластери є? Залежності з'явилися? Ні? Ну що ж, продовжуй, праця облагороджує. Ти ж хотів працювати в датасаєнсі.

2.1. Кластеризація



«Розділяє об'єкти за невідомою ознакою. Машина сама вирішує як краще»

Сьогодні використовують для:

Сегментація ринку (типів покупців, лояльності)

Об'єднання близьких точок на карті

Стиснення зображень

Аналіз і розмітки нових даних

Детектори аномальної поведінки

Популярні алгоритми: Метод K-середніх , Mean-Shift , DBSCAN

Кластеризація - це класифікація, але без заздалегідь відомих класів. Вона сама шукає схожі об'єкти та об'єднує їх в кластери. Кількість кластерів можна задати заздалегідь або довірити це машині. Схожість об'єктів машина визначає за тими ознаками, які ми їй розмітили - у кого багато схожих характеристик, тих давай в один клас.

Відмінний приклад кластеризації - маркери на картах в Інтернеті. Коли ви шукаєте всі крафтові бари у Львоі, машині доводиться групувати їх в

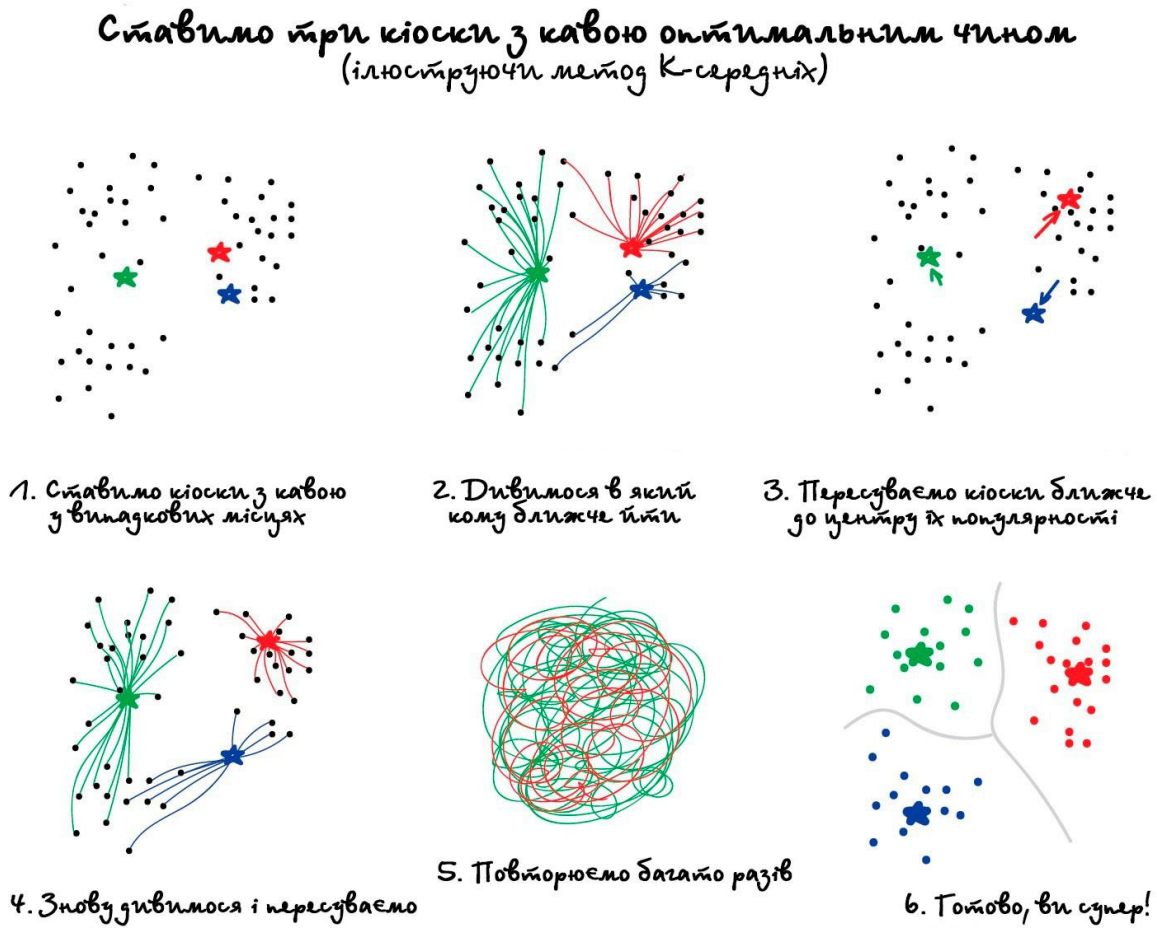
кружечки з циферками, інакше браузер зависне в потугах намалювати мільйон маркерів.

Більш складні приклади кластеризації можна згадати в додатках iPhoto або Google Photos, які знаходять особи людей на фотографіях і групують їх у альбоми. Додаток не знає як звуть ваших друзів, але може відрізнити їх за характерними рисами обличчя. Типова кластеризація.

Правда для початку їм доводиться знайти ці самі «характерні риси», а це вже тільки з учителем.

Стиснення зображень - ще одна популярна проблема. Зберігаючи картинку в PNG, ви можете встановити палітру, скажімо, в 32 кольори. Тоді кластеризація знайде всі «приблизно червоні» пікселі зображення, вирахує з них «середній червоний по лікарні» і замінить всі червоні на нього. Менше кольорів - менший файл.

Проблема тільки, як бути з кольорами типу Cyan - ось він ближче до зеленого чи синього? Тут нам допоможе популярний алгоритм кластеризації - Метод К-середніх (K-Means) . Ми випадковим чином кидаємо на палітру кольорів наші 32 точки, називаючи їх центроїдами. Всі інші точки відносимо до найближчого центроїду від них - виходять так би мовити сузір'я з найближчих кольорів. Потім рухаємо центроїд в центр свого сузір'я і повторюємо поки центроїди не перестануть рухатися. Кластери виявлені, стабільні і їх рівно 32 як і треба було.



Шукати центроїди зручно і просто, але в реальних задачах кластери можуть бути зовсім не круглої форми. Ось ви геолог, якому потрібно знайти на карті схожі за структурою гірські породи - ваші кластери не тільки будуть вкладені одна в одну, але ви ще й не знаєте скільки їх взагалі вийде.

Є більш досконалі способи кластеризації, але не всі знають, який коли слід застосовувати, і далеко не всі розуміють, як вони працюють.

Affinity propagation

https://habr.com/ru/post/321216/

Affinity propagation (AP, він же метод поширення близькості) отримує на вхід матрицю схожості між елементами датасета $\mathbf{S} = N \times N$ і повертає набір міток, присвоєних цим елементам. Без зайвих слів відразу викладу алгоритм на стіл:

Algorithm 1 Affinity propagation

function AFFINITY PROPAGATION($\mathbf{S}, maxiterations$)

$\mathbf{R}, \mathbf{A} \leftarrow \mathbf{0}, \mathbf{0}$

while $i < maxiterations$ **do** {

$$\forall i,k : r(i,k) \leftarrow s(i,k) - \max_{\forall j: j \neq k} (a(i,j) + s(i,j)) \quad (1)$$

$$\forall i,k, i \neq k : a(i,k) \leftarrow \min\left(0, r(k,k) + \sum_{\forall j: j \neq i, j \neq k} max\big(0, r(j,k)\big)\right) \quad (2)$$

$$\forall k : a(k,k) = \sum_{\forall j: j \neq k} \max(0, r(j,k)) \quad (3)$$

}

$$\forall i : c_i \leftarrow \arg\max_k \big(a(i,k) + r(i,k)\big) \quad (4)$$

return $(c_1, c_2 \ldots c_N)$

Ха, було б що викладати. Всього три рядки, якщо розглядати основний цикл; (4) - явно правило присвоєння міток. Але не все так просто. Більшості програмістів зовсім не очевидно, що ж ці три рядки роблять з матрицями $\mathbf{S}$, $\mathbf{R}$ и $\mathbf{A}$. Офіційна стаття пояснює, що $\mathbf{R}$ и $\mathbf{A}$ - матриці «відповідальності» і «доступності» відповідно, і що в циклі відбувається «обмін повідомленнями» між потенційними лідерами кластерів і іншими точками. Чесно сказати, це досить поверхневе пояснення, і толком незрозуміло ні призначення цих матриць ні як і чому змінюються їх значення.

Інтуїтивне пояснення

У певному просторі, в деякій державі живуть точки. У точок багатий внутрішній світ, але існує деяке правило $s(i,k)$, за яким вони можуть сказати, наскільки *схожий* на них сусід. Більш того, у них вже є таблиця $\mathbf{S}$, де записані всі $s(i,k)$, і вона майже ніколи не змінюється.

Одним жити на світі нудно, тому точки хочуть зібратися в групи за інтересами. Гуртки формуються навколо лідера (exemplar), який представляє інтереси всіх членів товариства. Кожна точка хотіла б бачити лідером когось, хто максимально на неї схожий, але готова миритися з іншими кандидатами, якщо ті подобаються багатьом іншим. Слід додати, що точки досить-таки скромні: всі думають, що лідером має бути хтось інший. Можна переформулювати їх низьку самооцінку так: кожна точка вважає, що коли справа доходить до об'єднання в групи, то вона сама на себе не схожа $(s(k,k) < 0)$. Переконати точку стати лідером, можуть або колективні підбадьорення з боку схожих на неї товаришів, або, навпаки, якщо в суспільстві не буде зовсім нікого схожого на неї.

Точки заздалегідь не знають ні що це будуть за колективи, ні загальна їх кількість. Об'єднання йде зверху вниз: спочатку точки чіпляються за президентів груп, потім тільки розмірковують, хто ще підтримує того ж кандидата. На вибір точки в якості лідера впливають три параметра: *схожість* (про нього вже сказано), *відповідальність* і *доступність*. *Відповідальність* (responsibility, таблиця R з елементами $r(i,k))$ відповідає за те, наскільки *i* хоче бачити *k* своїм ватажком. Відповідальність покладається кожною точкою на кандидата в лідери групи. *Доступність* (availability, таблиця ***A***, з елементами $a(i,k))$ - є відповідь від потенційного ватажка *k*, наскільки добре *k* готова представляти інтереси *i*. Відповідальність і доступність точки обчислюють в тому числі і самі для себе. Тільки коли велика самовідповідальність (так, я хочу представляти свої інтереси) і самодоступность (так, я можу представляти свої інтереси), тобто $a(k,k) + r(k,k) > 0$, точка може перебороти вроджену невпевненість в собі. Рядові точки в кінцевому підсумку приєднуються до лідера, для якого у них найбільша сума $a(i,k) + r(i,k)$. Зазвичай на самому початку, $\mathbf{R} = \mathbf{0}$ и $\mathbf{A} = \mathbf{0}$.

Візьмемо простий приклад: точки *X, Y, Z, U, V, W*, весь внутрішній світ яких - любов до котиків, *K*. У *X* алергія на кішок, для нього будемо вважати *K = -2, Y* ставиться до них прохолодно, *K = -1*, а *Z* просто не до них, *K = 0*. У будинку *U* чотири кішки (*K = 4*), у *V* - п'ять (*K = 5*), а у W цілих сорок (*K = 40*). Визначимо несхожість як абсолютну величину різниці *K*. *X* несхожий на *Y* на один умовний бал, а на *U* - на цілих шість. Тоді схожість, $s(i,k)$ - це просто мінус несхожість. Виходить, що точки з нульовою схожістю якраз таки однакові; чим більш негативно $s(i,k)$, тим сильніше відрізняються точки. Трохи контрінтуітівно, ну да ладно. Розмір «самонепохожесті» в прикладі оцінимо в 2 бали $(s(k,k) = -2)$.

Отже, на першому етапі кожна точка *i* покладає відповідальність на всіх *k* (в тому числі і на себе) пропорційно подібністю між *i* та *k* і обернено пропорційно подібністю між *i* та самим схожим на нього вектором крім k ((1) с $a(i,j)=0$).

Таким чином:

- Найближча (сама схожа) точка задає розподіл відповідальності для всіх інших точок. Розташування точок далі перших двох поки що впливає тільки на $r(i,k)$ відведену їм і тільки їм.

- Відповідальність, покладена на найближчу точку також залежить від розташування другої найближчої точки.

- Якщо в радіусі досяжності *i* є кілька більш-менш схожих на нього кандидатів, на тих буде покладено приблизно однакова відповідальність.

- $s(k,k)$ виступає свого роду обмежувачем - якщо якась точка занадто сильно схожа на всі інші, їй нічого не залишається, крім як сподіватися тільки на себе.

Якщо $r(i,k)>0$, то *i* хотіла б, щоб *k* був її представником. $r(k,k)>0$ означає, що сама *k* хоче бути засновником колективу, $r(k,k)<0$ - що *k* хоче належати іншому колективу.

Повертаючись до прикладу: *X* покладає на *Y* *відповідальність* в розмірі -1 - (- 2) = 1 бал, на *Z* – -2 - (- 1) = - 1 бал, на *U* – -6 - (- 1) = - 5 , а на себе -2 - (- 1) = - 1. *X* взагалі-то не проти, щоб лідером групи котоненавистників був *Z*, якщо *Y* не захоче бути їм, але навряд чи буде спілкуватися з *U* і *V*, навіть якщо вони зберуть велику команду. *W* настільки несхожа на всі інші точки, що їй нічого не залишається, як сподіватися відповідальність в розмірі -2-(-35)=33 бали тільки на себе.

Потім точки починають думати, наскільки вони самі готові бути лідером (доступні, available, для лідерства). Доступність для самого себе (3) складається з усієї позитивної відповідальності, «голосів», відданих точці. Для *k* неважливо, скільки точок думають, що вона буде погано їх представляти. Для неї головне, що хоч хтось думає, що вона буде їх представляти добре. Доступність *k* для *i* залежить від того, наскільки сильно *k* готовий представляти сам себе і від кількості позитивних відгуків про нього (2) (хтось інший теж вважає, що він буде відмінним представником колективу). Ця величина обмежується зверху нулем, щоб знизити вплив точок, про яких дуже багато хто думає добре, щоб вони не об'єднали в одну групу ще більше точок і цикл не вийшов з під контролю. Чим менше $r(k,k)$, тим більше голосів потрібно зібрати *k*, щоб $a(i,k)$ дорівнювало 0 (тобто ця точка була не проти взяти під своє крило ще й *i*).

Починається новий етап виборів, але тепер уже $\mathbf{A}\neq\mathbf{0}$. Згадаймо, що $a(k,k)\geq 0$, а $a(i,k)\leq 0, i\neq k$. Це по-різному впливає на $r(i,k)$

1. $i = k$. Тоді $a(k,k) \geq 0$ не грає ролі, і в (1) усі $-max(\dots)$ в правій частині будуть не менше, ніж були на першому кроці. $a(i,j) < 0$ по суті віддаляє точку *i* від *j*. Самовідповідальність точки підвищується від того, що у самого кращого кандидата з її точки зору, погані відгуки.

2. $i \neq k$. Тут виступають обидва ефекти. Цей випадок розщеплюється на два:

   $a(i,j) + s(i,j), i \neq j$ — max. Те ж саме що і в разі 1, але підвищується відповідальність покладається на точку *k* ;

   $a(i,i) + s(i,i)$ — max. Тоді $-max(\dots)$ буде не більше, ніж на першому кроці, $r(i,k)$ зменшується. Якщо продовжувати аналогію, то це як якби точка, яка і так вже замислювалася, чи не стати її лідером, отримала додаткове схвалення.

Доступність залишає в змаганні точки які або готові самі постояти за себе (W, $a(i,i) = 0$, але це ніяк не впливає на (1), тому що навіть з урахуванням поправки *W* знаходиться дуже далеко від інших точок) або ті, за яких готові постояти інші (Y, в (3) у неї по доданку від *X* і від *Z*). *X* і *Z*, які не строго найкраще схожі на когось, але при цьому на когось таки схожі, вибувають зі змагання. Це впливає на розподіл відповідальності, що впливає на розподіл доступності і так далі. Зрештою, алгоритм зупиняється - $a(i,k)$ перестають змінюватися. *X*, *Y* і *Z* об'єднуються в компанію навколо *Y*; дружать *U* і *V* з *U* в якості лідера; *W* добре і з 40 кішками.

Перепишемо вирішальне правило, щоб отримати ще один погляд на вирішальне правило алгоритму. Скористаємося (1) і (3). Позначимо $\tilde{s}(i,k) = a(i,k) + s(i,k)$ — схожість з поправкою на те, що *k* говорить про своїх командирських здібностях i; $\tilde{d}(i,k) = -\tilde{s}(i,k)$ — несхожість *i* - *k* з поправкою.

$$
\begin{aligned}
a(k,k) + r(k,k) &> 0 \\
\sum_{\forall j: j \neq k} \max(0, r(j,k)) + r(k,k) &> 0 \\
\sum_{\forall j: j \neq k} \max(0, r(j,k)) + s(k,k) - \max_{\forall j: j \neq k} \tilde{s}(j,k) &> 0 \\
\underbrace{\sum_{\forall j: j \neq k} \max(0, r(j,k))}_{\text{голоса}} + \underbrace{\min_{\forall j: j \neq k} \tilde{d}(j,k)}_{\text{одиночество}} &> \underbrace{-s(k,k)}_{\text{калибратор}}
\end{aligned}
\tag{5}
$$

Приблизно те, що було сформульовано спочатку. Звідси видно, що чим менше впевнені в собі точки, тим більше «голосів» необхідно зібрати або тим

більше несхожим на інші потрібно бути, щоб стати лідером. Тобто чим менше $s(k,k)$, тим крупніше виходять групи.

Давайте поговоримо про нюанси застосування алгоритму.

Кластеризацію можна представити у вигляді завдання дискретної максимізації з обмеженнями. Нехай на безлічі елементів задана функція подібності $s(i,j)$. Наше завдання знайти такий вектор міток $\mathbf{c} = \{c_1, c_2 ... c_N\}, c_i \in \{1 \dots N\}$, який максимізує функцію

$$S(c) = \sum_{i=1}^{N} s(i, c_i) + \sum_{k=1}^{N} \delta(c_k)$$

де $\delta(c_k)$ - член обмежувач, рівний $-\infty$, якщо існує точка *i*, яка вибрала точку *k* своїм лідером $(c_i = k)$, але сама *k* лідером себе не вважає $(c_k \neq k)$. Погана новина: знаходження ідеального **c** - це **NP**-складна задача, відома як задача про розміщення об'єктів. Проте, для її вирішення існує кілька наближених алгоритмів. У методах, що нас цікавлять, s_i, c_i и δ_i представляються вершинами двудольного графа, після чого між ними відбувається обмін інформацією, що дозволяє з ймовірнісної точки зору оцінити, яка мітка краще підійде для кожного елемента. Взагалі поширення близькості - це окремий випадок (скоріше, звуження) циклічного поширення переконань (loopy belief propagation, LBP), але замість суми ймовірностей (підтип Sum-Product) в деяких місцях ми беремо тільки максимальну (підтип Max-Product) з них. По-перше, LBP-Sum-Product на порядок складніше, по-друге, там легше мати справу з обчислювальними проблемами, по-третє теоретики стверджують, що для завдання кластеризації це має більший сенс.

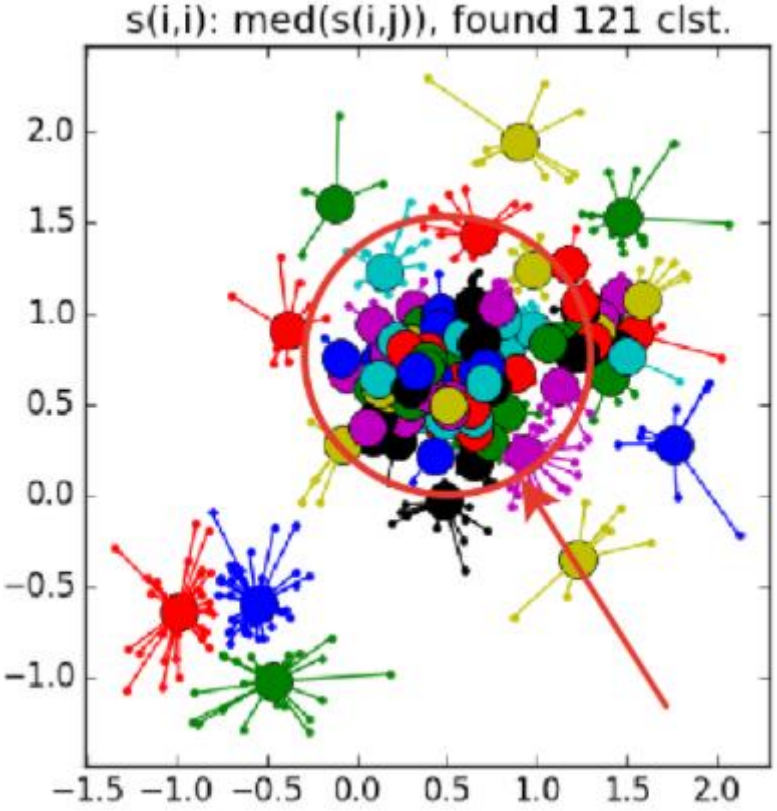
Affinity propagation детермінований. Він має складність $O(N^2T)$, де *N* - розмір набору даних, а *T* - кількість ітерацій, і займає $O(N^2)$ пам'яті. Існують модифікації алгоритму для розріджених даних, але все одно AP сильно смутніє зі збільшенням розміру датасета. Це досить серйозний недолік. Зате поширення близькості не залежить від розмірності елементів даних. Поки що не існує розпаралеленого варіанти AP. Тривіальне ж розпаралелювання у вигляді безлічі запусків алгоритму не підходить в силу детермінованості.

Affinity propagation, як багато інших алгоритмів, можна перервати достроково, якщо $\mathbf{R}$ и $\mathbf{A}$ перестають оновлюватися. Для цього майже у всіх реалізаціях використовується два значення: максимальна кількість ітерацій і період, з яким перевіряється величина оновлень.

Affinity propagation схильний до обчислювальних осциляцій у випадках, коли є кілька хороших розбиттів на кластери. Щоб уникнути проблем, по-перше, на самому початку до матриці подібності додається трохи шуму (дуже-дуже небагато, щоб не вплинути на детерменірованность, в sklearn-імплементації близько 10^{-16}), а по-друге, при оновленні $\mathbf{R}$ и $\mathbf{A}$

використовується не просте присвоювання, а *присвоювання з експоненціальним згладжуванням*. Друга оптимізація притому в цілому добре впливає на якість результату, але через неї збільшується кількість необхідних ітерацій T . Автори радять використовувати параметр згладжування $0.5 \leq \gamma < 1.0$ зі значенням за замовчуванням в 0.5. Якщо алгоритм не сходиться або сходиться частково, слід збільшити γ до 0.9 або до 0.95 з відповідним збільшенням кількості ітерацій.

Ось так виглядає відмова - купа дрібних кластерів, оточена кільцем кластерів середнього розміру:



Як вже було сказано, замість наперед заданої кількості кластерів використовується параметр «самоподібності» $s(k,k);$; чим менше $s(k,k);$, тим крупніше кластери. Існують евристики для автоматичного підстроювання значення цього параметра: використовуйте медіану за всіма $s(i,k)$ для більшого числа кластерів; 25 перцентиль або навіть мінімум по $s(i,k)$ - для меншого (все одно доведеться підганяти, ха-ха). При занадто маленькому або занадто велике значення «самоподібності» алгоритм і зовсім не видасть якихось корисних результатів.

У якості $s(i,k)$ само собою напрошується використовувати негативне евклідова відстань між *i* та *j* , але вас ніхто не обмежує у виборі. Навіть в разі датасета з векторів дійсних чисел можна перепробувати багато цікавого. Взагалі ж, автори стверджують, що на функцію схожості не накладено жодних особливих обмежень; навіть не обов'язково, щоб виконувалося

правило симетрії або правило трикутника. Але варто зауважити, що чим більше хитромудра у вас $s(i,j)$, тим менше ймовірність, що алгоритм зійдеться до чогось цікавого.

Розміри кластерів, отриманих в ході поширення близькості, варіюються в досить невеликих межах, і якщо в датасета є сильно різні за розміром скупчення, AP може або пропустити маленькі, або порахувати великі за кілька. Зазвичай друга ситуація менш неприємна - вона може бути виправлена. Тому часто AP потребує постобробці - додаткової кластеризації лідерів груп. Підходить будь-який інший метод, тут може допомогти додаткова інформація про завдання. Слід пам'ятати, що чесним самотньо стоячим точкам виділяються свої кластери; такі викиди потрібно фільтрувати перед постобробкою.

Використовуйте Affinity propagation, коли

У вас не дуже великий $(N < 10^5 - 10^6)$ або в міру великий, але розріджений $(N < 10^6 - 10^7)$ датасет

Заздалегідь відома функція близькості

Ви очікуєте побачити безліч кластерів різної форми і кількістю елементів, що трохи варіюється.

Ви готові повозитися з пост обробкою.

Складність елементів датасета значення не має

Властивості функції близькості значення не мають

Кількість кластерів значення не має

Застосування Стверджується, що різні вчені успішно застосовували Affinity propagation для

сегментування зображень

Виділення груп в геномі

Розбиття міст на групи по доступності

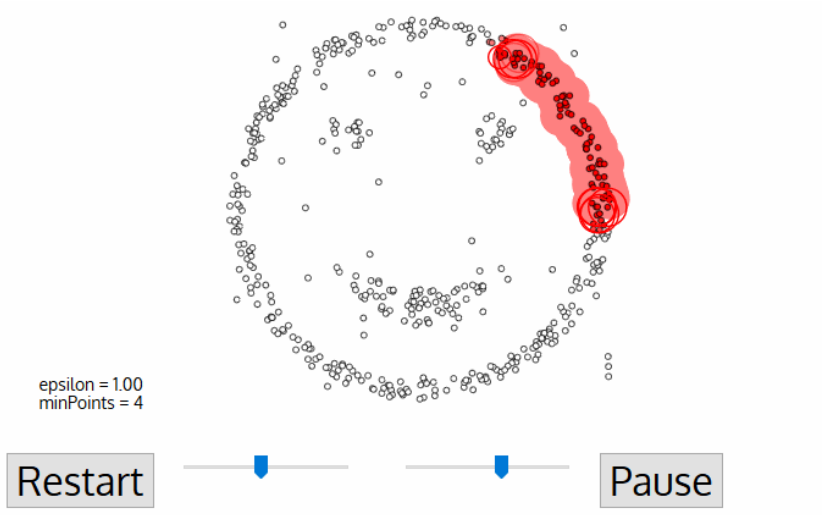
Кластеризації зразків граніту

Групування фільмів на Netflix

У всіх цих випадках був отриманий результат краще, ніж за допомогою k-means. Не те щоб це якийсь суперський результат сам по собі.

Алгоритм DBSCAN

Хитрим завданням - хитрі методи. DBSCAN , наприклад. Він сам знаходить скупчення точок і будує навколо кластери. Його легко зрозуміти, якщо уявити, що точки - це люди на площі. Знаходимо трьох людей, які знаходяться близько одна до одної, і пропонуємо їм взятися за руки. Потім вони починають брати за руку тих, кого можуть досягти. Так по ланцюжку, поки ніхто більше не зможе взяти когось за руку - це і буде перший кластер. Повторюємо, поки не поділимо всіх. Ті, кому взагалі нікого брати за руку - це аномалії, викидаємо. В динаміці виглядає досить красиво:



DBSCAN (Density-based spatial clustering of applications with noise, щільнісний алгоритм просторової кластеризації з присутністю шуму), як випливає з назви, оперує щільністю даних. На вхід він просить матрицю близькості і два загадкових параметра - радіус ϵ-окрестності і кількість сусідів. Так відразу й не зрозумієш, як їх вибрати, причому тут щільність, і чому саме DBSCAN добре розправляється з зашумленими даними. Без цього складно визначити межі його застосовності.

Інтуїтивне пояснення

У гігантському залі натовп людей справляє чийсь день народження. Хтось тиняється один, але більшість - з товаришами. Деякі компанії просто товпляться юрбою, деякі - водять хороводи або танцюють ламбаду.

Ми хочемо розбити людей в залі на групи.

Але як виділити групи настільки різної форми, та ще й не забути про одинаків? Спробуємо оцінити щільність натовпу навколо кожної людини. Напевно, якщо щільність натовпу між двома людьми вище певного порогу, то вони належать одній компанії. Справді, буде дивно, якщо люди, які ведуть «паровозик», будуть відноситися до різних груп, навіть якщо щільність ланцюжка між ними змінюється в деяких межах.

Будемо говорити, що поруч з деякою людиною зібрався натовп, якщо близько до нього стоять кілька інших осіб. Ага, відразу видно, що потрібно задати два параметра. Що значить «*близько*»? Візьмемо якусь інтуїтивно зрозумілу відстань. Скажімо, якщо люди можуть доторкнутися до голів один одного, то вони знаходяться близько. Близько метра. Тепер, скільки саме «*кілька інших осіб*»? Припустимо, три людини.

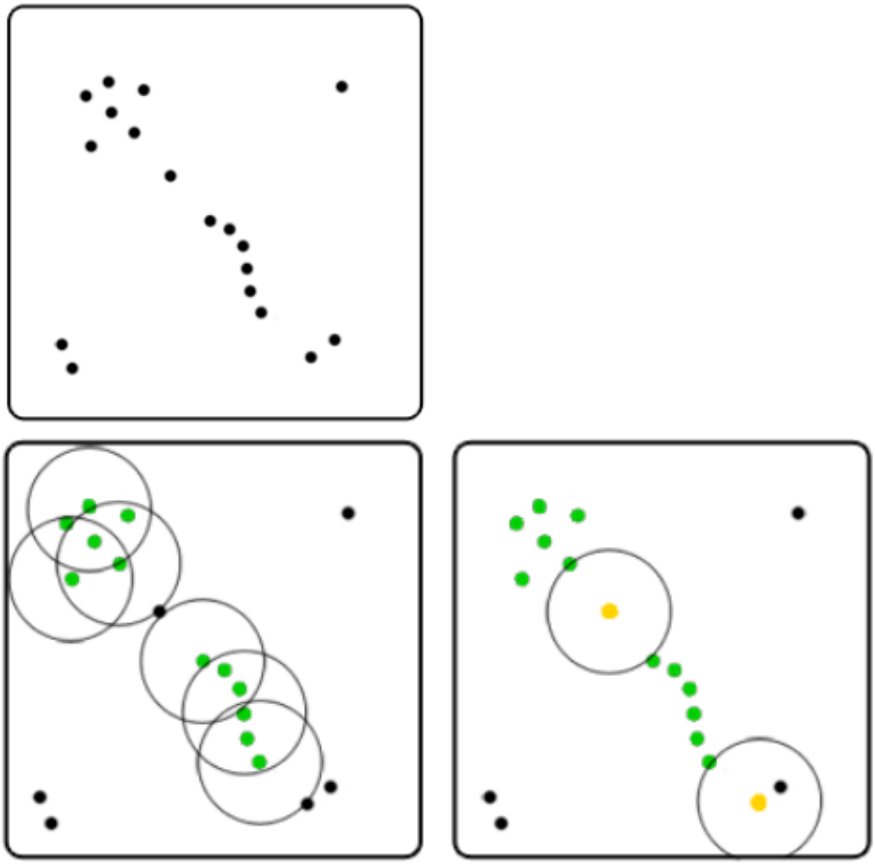
Нехай кожен підрахує, скільки людей стоять в радіусі метра від нього. Всі, у кого є хоча б три сусіда, беруть в руки зелені прапорці. Тепер вони корінні елементи, саме вони формують групи.

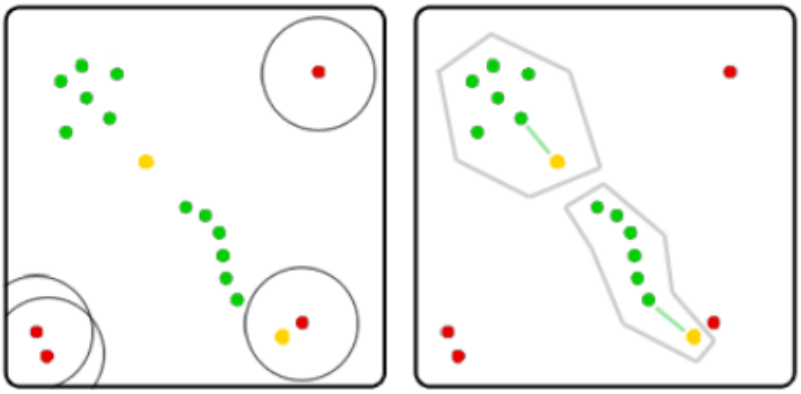
Звернемося до людей, у яких менше трьох сусідів. Виберемо тих, у яких принаймні один сусідів тримає зелений прапор, і вручимо їм жовті прапори. Скажімо, що вони знаходяться на кордоні груп.

Залишилися одинаки, у яких мало того що немає трьох сусідів, так ще й жоден з них не тримає зелений прапор. Роздамо їм червоні прапори. Будемо вважати, що вони не належать жодній групі.

Таким чином, якщо від однієї людини до іншої можна створити ланцюжок «зелених» людей, то ці дві людини належать одній групі. Очевидно, що всі подібні збіговиська розділені або порожнім простором або людьми з жовтими прапорами. Можна їх пронумерувати: кожен в групі №1 може досягти по ланцюжку рук кожного іншого в групі №1, але нікого в №2, №3 і так далі. Те ж для інших груп.

Якщо поруч з людиною з жовтим прапорцем є тільки один «зелений» сусід, то він буде належати тій групі, до якої належить його сусід. Якщо таких сусідів кілька, і у них різні групи, то доведеться вибирати. Тут можна скористатися різними методами - подивитися, хто з сусідів найближчий, наприклад. Доведеться якось обходити крайові випадки, але нічого страшного.





Як правило, немає сенсу позначати всіх корінних елементів натовпу відразу. Раз від кожного корінного елемента групи можна провести ланцюжок до кожного іншого, то все одно з якого починати обхід - рано чи пізно знайдеш всіх. Тут краще підходить ітеративний варіант:

1. Підходимо до випадкової людини з натовпу.
2. Якщо поруч з ним менше трьох осіб, переносимо його в список можливих одинаків і вибираємо кого-небудь іншого.
3. Інакше:
   - Виключаємо його зі списку людей, яких треба обійти.
   - Вручаємо цій людині зелений прапорець і створюємо нову групу, в якій він поки що єдиний мешканець.
   - Обходимо всіх його сусідів. Якщо його сусід вже в списку потенційних одинаків або поруч з ним мало інших людей, то перед нами край натовпу. Для простоти можна відразу помітити його жовтим прапором, приєднати до групи і продовжити обхід. Якщо сусід теж виявляється «зеленим», то він не починає нову групу, а приєднується до вже створеної; крім того ми додаємо в список обходу сусідів сусіда. Повторюємо цей пункт, поки список обходу не опиниться порожній.
4. Повторюємо кроки 1-3, поки так чи інакше не обійдемо всіх людей.
5. Розбираємося зі списком одинаків. Якщо на кроці 3 ми вже розкидали всіх крайових, то в ньому залишилися тільки викиди-одинаки - можна відразу закінчити. Якщо немає, то потрібно якось розподілити людей, що залишилися в списку.

Формальний підхід

Введемо кілька визначень. Нехай задана деяка симетрична функція відстані $\rho(x, y)$ і константи ϵ и m. Тоді:

1. Назвемо область $E(x)$, для якої $\forall y : \rho(x, y) \leq \epsilon$, ϵ-околом (окрестностью) об'єкта *x* .
2. Кореневим об'єктом або ядерним об'єктом ступеня *m* називається об'єкт, ϵ-окіл якого містить не менше *m* об'єктів: $|E(x)| \geq m$.

3. Об'єкт *p* безпосередньо щільно-досяжний з об'єкта *q*, якщо $p \in E(q)$ и q - кореневий об'єкт.
4. Об'єкт *p* щільно-досяжний з об'єкта *q*, якщо $\exists p_1, p_2 \dots p_n, p_1 = q, p_n = p,$, такі що $\forall i \in 1 \dots n-1 : p_{i+1}$ безпосередньо щільно-досяжний з p_i .

Виберемо який-небудь кореневий об'єкт *p* з датасета, помітимо його і помістимо всіх його безпосередньо щільно-досяжних сусідів в список обходу. Тепер для кожної *q* зі списку: помітимо цю точку, і, якщо вона теж коренева, додамо всіх її сусідів в список обходу. Тривіально доводиться, що кластери помічених точок, сформовані в ході цього алгоритму максимальні (тобто їх не можна розширити ще однією точкою, щоб задовольнялися умови) і зв'язні в сенсі щільно-досяжності. Звідси випливає, що якщо ми обійшли не всі точки, можна перезапустити обхід з якого-небудь іншого кореневого об'єкта, і новий кластер не проковтне попередній.

Приклад більш коректної реалізації DBSCAN на Python можна знайти в пакеті sklearn.

Використовуйте DBSCAN, коли:

У вас в міру великий $(N \approx 10^6)$ датасет. Навіть $N \approx 10^7 - 10^8$ якщо під рукою оптимізована і распаралленная реалізація.

Заздалегідь відома функція близькості, симетрична, бажано, не дуже складна. KD-Tree оптимізація часто працює тільки з евклідовою відстанню.

Ви очікуєте побачити згустки даних екзотичної форми: вкладені і аномальні кластери, складки малої розмірності.

Щільність кордонів між згустками менше щільності найменш щільного кластера. Краще якщо кластери зовсім відокремлені один від одного.

Складність елементів датасета значення не має. Однак їх повинно бути досить, щоб не виникало сильних розривів в щільності (див. Попередній пункт).

Кількість елементів в кластері може варіюватися як завгодно.

Кількість викидів значення не має (в розумних межах), якщо вони розсіяні по великому об’єму.

Кількість кластерів значення не має.

У DBSCAN є історія успішних застосувань. Наприклад, можна відзначити його використання в задачах:

Виявлення соціальних гуртків

Сегментування зображень

Моделювання поведінки користувачів веб-сайтів

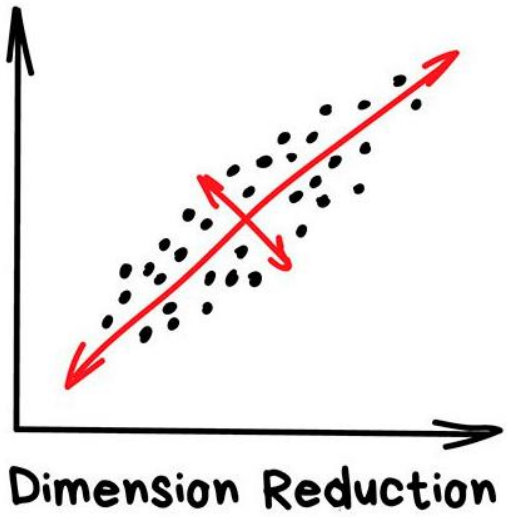
Попередня обробка в завданні передбачення погоди

Тим, хто цікавиться кластеризацією, рекомендую статтю The 5 Clustering Algorithms Data Scientists Need to Know

Як і класифікація, кластеризація також може використовуватися **як детектор аномалій.** Поведінка користувача після реєстрації різко відрізняється від нормального? Заблокувати його і створити тікет саппорту, щоб перевірили бот це чи ні. При цьому нам навіть не треба знати, що є «нормальна поведінка» - ми просто вивантажуємо всі дії користувачів в модель, і нехай машина сама розбирається хто тут нормальний.

Працює такий підхід, в порівнянні з класифікацією, не дуже. Але за спробу не б'ють, раптом вийде.

2.2. Зменшення Розмірності (Узагальнення)



«Збирає конкретні ознаки в абстракції вищого рівня»

Сьогодні використовують для:

Рекомендаційні Системи (★)

Красиві візуалізації

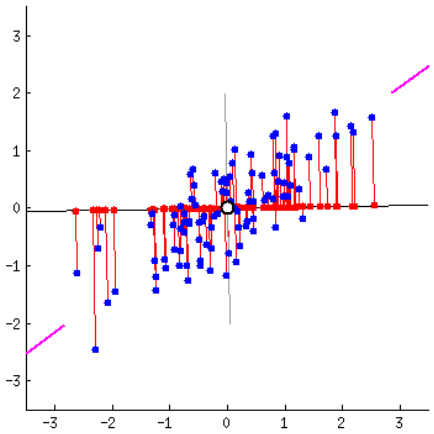
Визначення тематики та пошуку схожих документів

Аналіз фейковий зображень

Ризик-менеджмент

Популярні алгоритми: Метод головних компонент (PCA), Сингулярне розкладання (SVD), Латентне розміщення Діріхле (LDA), Латентно-семантичний аналіз (LSA, pLSA, GLSA), t-SNE (для візуалізації)

Спочатку це були методи хардкорних Data Scientist'ів, яким вивантажували дві фури цифр і говорили знайти там що-небудь цікаве. Коли просто будувати графіки в екселі вже не допомагало, вони придумали напружити машини шукати закономірності замість них. Так у них з'явилися методи, які назвали Dimension Reduction або Feature Learning.

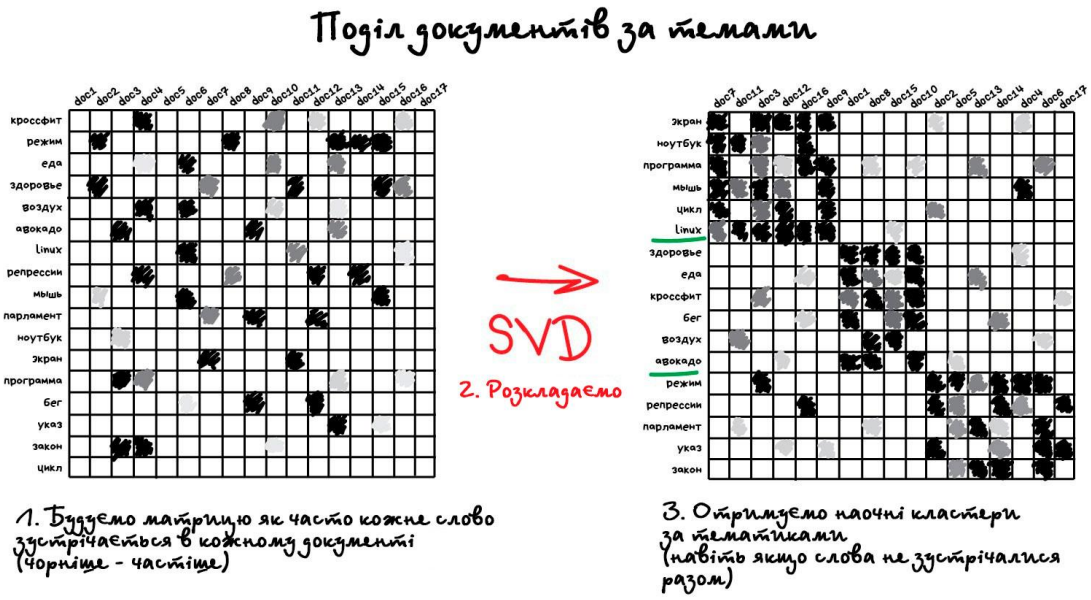


Для нас практична користь їх методів в тому, що ми можемо об'єднати декілька ознак в одну і отримати абстракцію. Наприклад, собаки з трикутними вухами, довгими носами і великими хвостами об'єднуються в корисну абстракцію «вівчарки». Очевидно, що ми втрачаємо інформацію про конкретних вівчарок, але нова абстракція по-любому корисніша цих зайвих деталей. Плюс, навчання на меншій кількості розмірностей йде сильно швидше.

Інструмент на диво добре підійшов для визначення тематик текстів (Topic Modelling). Ми змогли абстрагуватися від конкретних слів до рівня смислів навіть без залучення вчителя зі списком категорій. Алгоритм назвали Латентно-семантичний аналіз (LSA), і його ідея була в тому, що частота появи слова в тексті залежить від його тематики: в наукових статтях більше технічних термінів, в новинах про політику - імен політиків. Так, ми могли б просто взяти всі слова з статей і кластеризувати, як ми робили з кіосками вище, але тоді ми б втратили всі корисні зв'язки між словами, наприклад, що батарейка і акумулятор означають одне і те ж в різних документах.

Точність такої системи - повне дно, навіть не намагайтеся.

Потрібно якось об'єднати слова і документи в одну ознаку, щоб не втрачати ці приховані (латентні) зв'язку. Звідси і з'явилася назва методу. Виявилося, що Сингулярне розкладання (SVD) легко справляється з цим завданням, виявляючи для нас корисні тематичні кластери зі слів, які зустрічаються разом.



Для розуміння рекомендую статтю [Як зменшити кількість вимірювань і отримати з цього користь](), а практичне застосування добре описано в статті [Алгоритм LSA для пошуку схожих документів]() .

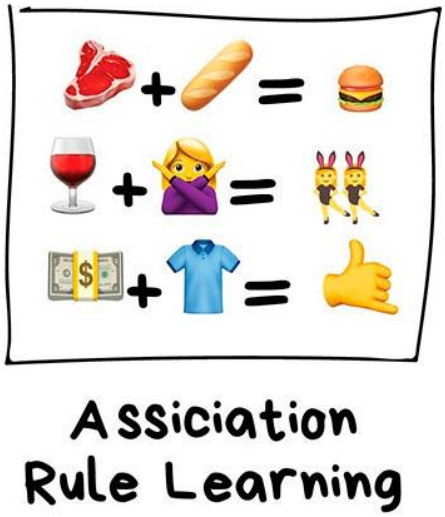
Інше мега-популярне застосування методу зменшення розмірності знайшли в рекомендаційних системах і колаборативній фільтрації. Виявилося, що якщо абстрагувати ними оцінки користувачів фільмів, виходить непогана система рекомендацій кіно, музики, ігор і чого завгодно взагалі.

Отримана абстракція буде важко зрозуміла мозком, але коли дослідники почали пильно розглядати нові ознаки, вони виявили, що якісь з них явно корелюють з віком користувача (діти частіше грали в майнкрафт і дивилися мультфільми), інші з певними жанрами кіно, а треті взагалі з синдромом пошуку глибокого сенсу життя.

Машина, яка не знала нічого, крім оцінок користувачів, змогла дістатися до таких високих матерій, навіть не розуміючи їх. Достойно. Далі можна проводити соцопитування і писати дипломні роботи про те, чому бородаті дядьки люблять дегенеративні мультики.

На цю тему є непогана лекція - [Як працюють рекомендаційні системи]()

2.3. Пошук правил (асоціація)



«Шукає закономірності в потоці замовлень»

Сьогодні використовують для:

Прогноз акцій і розпродажів

Аналіз товарів, які купують разом

Розташування товарів на полицях

Аналіз патернів поведінки на веб-сайтах

Популярні алгоритми: Apriori, Euclat, FP-growth

Сюди входять всі методи аналізу продуктових кошиків, стратегій маркетингу і інших послідовностей.

Припустимо, покупець бере в дальньому кутку магазину пиво і йде на касу. Чи варто ставити на його шляху горішки? Чи часто люди беруть їх разом? Горішки з пивом, напевно так, але які ще товари купують разом? Коли ви власник мережі гіпермаркетів, відповідь для вас не завжди очевидна, але одне тактичне поліпшення в розташуванні товарів може принести гарний прибуток.

Це ж стосується інтернет-магазинів, де завдання ще цікавіше - за яким товаром покупець повернеться наступного разу?

З незрозумілих причин, пошук правил - найбільш погано продумана категорія серед всіх методів навчання. Класичні способи полягають в тупому переборі пар всіх куплених товарів за допомогою дерев або множин. Самі алгоритми працюють наполовину - можуть шукати закономірності, але не вміють узагальнювати або відтворювати їх на нових прикладах.

У реальності кожен великий ритейлер пиляє свій велосипед і жодних особливих проривів в цій області не зустрічається. Максимальний рівень технологій тут - створити систему рекомендацій, як в пункті вище. Хоча може я просто далекий від цієї області?


Юзер купив диван
він обожнює дивани!!!
Рекомендувати йому ще 148 диванів
amazon
ebay
AliExpress