math_2020_035

Міністерство освіти і науки України

Чернівецький національний університет

імені Юрія Федьковича

Факультет математики та інформатики
(повна назва інституту/факультету)

Кафедра математичного моделювання
(повна назва кафедри)

Використання методів машинного навчання у медицині

Дипломна робота

Рівень вищої освіти - другий (магістерський)

Виконала:
студентка 6 курсу, групи 607
спеціальності 124 – Системний аналіз
(назва спеціальності)
Мосійчук Оксана Орестівна
(прізвище, ім'я та по-батькові)
Керівник к.ф.-м.н., доцент Дорошенко І.В.
(науковий ступінь, вчене звання, прізвище та ініціали)

До захисту допущено:

Протокол засідання кафедри № 6

від „3" грудня 2019 р.

зав. кафедри _____ доц. Піддубна Л.А.

Чернівці – 2019

Анотація

Дана дипломна робота присвячена вивченню методів машинного навчання та застосуванню їх у медицині. Основна увага приділяється методу кластеризації. Кластеризація значно скоротила час з використанням алгоритмів кластеризації Маркова на графах. Вдалося перейти від класичної постановки проблеми (один донор відповідає одному реципієнту) до ймовірнісної проблеми, в якій декілька донорів та реципієнтів об’єднані в одну групу (кластер). Самостійною частиною роботи є розв’язання за допомогою мови R конкретної задачі класифікації.

Зміст

Вступ

Окрім важливості набору даних, загалом машинне навчання складається з двох критичних факторів, моделювання та оптимізація. Моделювання означає, як моделювати розподіл ймовірностей даної навчальної вибірки, а потім методи оптимізації використовуються для пошуку найкращих параметрів обраної моделі.

Компонент досвіду або тренування машинного навчання часто застосовується до даних, що генеруються випадковим чином. Завдання учня – обробити такі генеровані випадковим чином приклади до висновків, які стосуються середовища, з якого вибираються ці приклади. Цей опис машинного навчання підкреслює його тісний зв'язок зі статистикою. Дійсно, багато спільного між двома дисциплінами, як з точки зору цілей, так і прийомів. Однак є кілька суттєвих відмінностей: якщо лікар придумує гіпотезу про наявність кореляції між курінням та хворобою серця, то це роль статистики для оцінки вибірки пацієнтів та перевірки обґрунтованості цієї гіпотези (це загальне статистичне завдання тестування гіпотез). Навпаки, машинне навчання має на меті використовувати дані, зібрані з вибірки пацієнтів, щоб скласти опис причин серцевих захворювань. На відміну від традиційної статистики, в машинному навчанні алгоритмічні міркування відіграють головну роль. Машинне навчання – це виконання навчання комп'ютерами; отже, алгоритмічні питання є ключовими. Ще одна відмінність полягає в тому хоч статистику часто цікавить асимптотична поведінка (наприклад, конвергенція статистичних оцінок на основі вибірки, оскільки розміри вибірки зростають до нескінченності), то теорія машинного навчання фокусується на граничних зразках. А саме, враховуючи розмір доступних зразків, теорія машинного навчання має на меті визначити ступінь точності, яку може очікувати студент на основі таких зразків.

У той час як у статистиці прийнято працювати в рамках припущення певних передбачених моделей даних (наприклад, припущення про нормальність розподілу даних, що генерують дані, або лінійність

функціональних залежностей), в машинному навчанні акцент робиться на роботі в умовах "без розподілу", де учень припускає якомога менше про природу розподілу даних і дозволяє алгоритму навчання визначити, які моделі найкраще наближені.

Прогрес медицини за останні 20 років не такий швидкий, як у технології. Тому застосування Data Mining є надзвичайно важливим для медицини. Використовуючи підходи до аналізу даних, можна зрозуміти без зайвих втрат, як певний препарат діє на групи пацієнтів, наприклад, із серцевою недостатністю та захворюваннями нирок; передбачити поширення конкретного вірусу; щоб отримати відповіді на запитання щодо показників, які є ключовими при трансплантації нирок.

У цій роботі використовуються методи машинного навчання для аналізу даних донорських пацієнтів для Програми обміну нирок. Використання машинного навчання в програмі обміну нирками забезпечить групи сумісності між донорами та реципієнтами, в результаті чого більше людей отримають необхідну трансплантацію.

Основна проблема детермінованих систем – це саме проблема перерахунку, яка виникає, коли одного (декількох) донорів (реципієнтів) видаляють із системи. У детермінованій системі після виключення одного або декількох системних об'єктів потрібно перерахувати всі співвідношення, тобто перерахувати значення всіх змінних x_i. Крім того, у цьому випадку система повинна додатково видалити "зайвих" одержувачів (донорів). Ця проблема вирішується лише шляхом пошуку ще одного рішення задачі лінійного програмування. У завданнях кластеризації ця проблема виникає рідше, оскільки кожен кластер містить достатньо велику кількість об'єктів. Це припущення дозволяє уникнути перерахунку рішення проблеми після усунення одного або декількох донорів або одержувачів з розгляду.

Магістерська робота присвячена застосуванню нової методики вирішення проблеми пошуку найкращих зв’язків між донорами та реципієнтами – кластерного аналізу. Ця область була обрана (алгоритм без вчителя), оскільки

проблеми оптимізації, з якими вирішувались такі завдання, стосуються саме цієї галузі дослідження.

Розділ І. Задачі машинного навчання

1.1. Застосування машинного навчання

Сьогодні роль машинного навчання значно зросла. Хоча його все-таки використовують лише у великих дослідженнях, багато людей стикаються з машиним навчанням в їх повсякденному житті. Ці дослідження та застосування такі:

- Освіта: однією з найважливіших галузей застосування є освіта. Нещодавно були проведені деякі дослідження з метою виявлення та підвищення успіху. Незважаючи на проекти зроблені в галузі освіти в останні роки, бажаного успіху не досягнуто. Існує маса факторів, які впливають на цю невдачу. Однак не визначено, який фактор має більший вплив на цю невдачу. У цьому контексті за допомогою анкети, застосованої до учнів середньої школи, успіхи учнів на уроках передбачили моделі машинного навчання, які привели до успіху. Аналогічно є деякі дослідження з метою визначення кваліфікації студентів у вищих закладах. У 2007 році було проведено дослідження в університеті Памуккале, де студенти ідентифіковані як ризиковані студенти відповідно до провалу курсу математики. У дослідженні було з'ясовано бали вступних іспитів у університет 434 студентів; математика, природничі науки, турецькі тести та бали випускних іспитів зіграли головну роль у прогнозуванні успіху з математики. У дослідженні для навчання було використано 289 даних студентів та 145 студентів. Як результат, 86 відсотків учнів, які здали математику були правильно оцінені. [9]
- Обробка зображень: у цьому методі машине навчання спрямоване на обробку та вдосконалення записаних зображень. Деякі області застосування: системи безпеки, виявлення обличчя, медицина (для діагностики хворих тканин та органів), військова (для обробки

підводних і супутникових знімків), виявлення руху, виявлення об'єктів.

- Обчислювальна біологія: послідовність ДНК, знаходження пухлини, виявлення наркотиків.
- Мовні технології: автоматичний переклад письмових текстів, машини з питаннями-відповідями, автоматичне узагальнення тексту, розуміння мови та команди.
- Автомобільна, авіаційна та виробнича: виявлення несправностей до їх виникнення, виробництво автономних транспортних засобів.
- Роздрібна торгівля: індивідуальний аналіз полиць для осіб, рекомендація знарядь, матеріальні оцінки, тенденції купівлі-продажу.
- Фінанси: кредитний контроль та оцінка ризиків, алгоритмічна торгівля.
- Сільське господарство: прогнозування врожайності або недоліків шляхом аналізу супутникових знімків.
- Людські ресурси: вибір найбільш успішного кандидата серед багатьох кандидатів.
- Енергія: розрахунок навантажень для опалення та охолодження для будівельних конструкцій, аналіз споживання енергії, розумне управління мережею.
- Метеорологія: прогноз погоди за допомогою датчиків.
- Здоров'я: попередження та діагностика шляхом аналізу даних про пацієнта, визначення захворювань, аналіз охорони здоров’я.
- Кібербезпека: виявлення шкідливого мережевого трафіку, з’ясування адресного шахрайства. [22]

Поряд із розвитком технології в останні роки, велику роль відігравали машини у нашому житті. У кожній частині нашого життя існує багато даних, і ці дані зростають з кожним днем. Завдяки машинам дані використовуються

дуже ефективно. Хоча вважається, що машини використовуються лише у галузі техніки та інформатики, вони зустрічаються в кожній частині людського життя. Фірми, які вже визнали це активно інвестують в цю сферу і досягають успіху. В майбутньому машини будуть успішними в роботах, які не зможе виконати людина. Деякі з відомих напрямків бізнесу зникнуть і з’являться нові сфери бізнесу. У такому середовищі потужність інформаційних технологій та машин повинні суворо враховуватися. [21]

1.2. Задачі машинного навчання з учителем

1.2.1. Регресійний аналіз

Регресійний аналіз – це статистичний метод моделювання взаємозв'язку між залежною (цільовою) та незалежною (предикторною) змінною з однією або кількома незалежними змінними. Більш конкретно, регресійний аналіз допомагає нам зрозуміти, як змінюється значення залежної змінної, що відповідає незалежній змінній, коли інші незалежні змінні утримуються фіксованими. Він передбачає постійні/реальні значення, такі як температура, вік, зарплата, ціна тощо. Регресія – це контрольована методика навчання, яка допомагає знаходити кореляцію між змінними і дає змогу передбачити змінну безперервного виходу на основі однієї або декількох змінних предиктора. В основному використовується для прогнозування, моделювання часових рядів та визначення причинно-наслідкового зв’язку між змінними.

В регресії ми побудуємо графік між змінними, який найкраще відповідає заданим точкам даних, використовуючи цей графік, модель машинного навчання може робити прогнози щодо даних. Регресія показує лінію або криву, яка проходить через усі точки даних на графіку прогнозування цілі таким чином, щоб вертикальна відстань між точками даних та лінією регресії була мінімальною. Відстань між точками даних та рядком вказує на те, чи має модель сильний зв'язок чи ні.

Термінології, пов'язані з регресійним аналізом:

1. Залежна змінна: основним фактором регресійного аналізу, який ми хочемо передбачити є залежна змінна. Її також називають цільовою змінною.
2. Незалежна змінна: фактори, які впливають на залежні змінні або використовуються для прогнозування значень залежних змінних, називаються незалежною змінною, а також предикатом.
3. Викиди: Outlier – це спостереження, яке містить або дуже низьке, або дуже високе значення порівняно з іншими спостережуваними значеннями. Викид може завадити результату, тому цього слід уникати.
4. Мультіколінеарність: якщо незалежні змінні сильно корелюються між собою, ніж інші змінні, то така умова називається мультиколінеарністю. Вона не повинна бути присутня у наборі даних, оскільки це створює проблеми під час ранжування найбільш впливової змінної.
5. Недостатня обробка та перенавчання: якщо наш алгоритм добре працює з набором даних, але не добре відповідає тестовому набору даних, то така проблема називається перенавчанням. І якщо наш алгоритм не працює навіть з навчальним набором даних, то така проблема називається недостатньою обробкою. [11]

Види регресії:

- ✓ проста лінійна регресія
- ✓ логістична регресія
- ✓ поліноміальна регресія
- ✓ SVR
- ✓ регресія дерева рішень
- ✓ випадкова регресія лісу.

Кожному об'єкту відповідає його ознаковий опис $(f_1(x),..., f_n(x))$, де $f_j: X \to \mathbb{R}$ – числові ознаки, $j = 1,...,n$. Лінійною моделлю регресії називається лінійна комбінація ознак з коефіцієнтами $a \in \mathbb{R}^n$:

$$g(x,a)=\sum_{j=1}^{n}a_j f_j(x). \qquad (1.1)$$

Введемо матричні позначення $F=(f_j(x_i))_{l\times n}$ – матриця об'єкти-ознаки; $y=(y_i)_{l\times 1}$ – цільовий вектор; $a=(a_j)_{n\times 1}$ – вектор параметрів.

В матричних позначеннях функціонал *Q* матиме вигляд

$$Q(a)=\| Fa-y\|^2 \qquad (1.2)$$

Запишемо необхідну умову мінімуму в матричному вигляді

$$\frac{\partial Q}{\partial a}(a)=2F^T(Fa-y)=0, \qquad (1.3)$$

звідки випливає, що $F^TFa=F^Ty$. Ця система лінійних рівнянь відносно *a* називається нормальною системою для задачі найменших квадратів. Якщо матриця F^TF розміру $n\times n$ невироджена, то розв'язком нормальної системи буде вектор

$$a^*=(F^TF)^{-1}F^Ty=F^+y. \qquad (1.4)$$

Матриця $F^+=(F^TF)^{-1}F^T$ називається псевдооборотною для прямокутної матриці *F*. Підставляючи знайдене рішення в вихідний функціонал, отримаємо

$$Q(a^*)=\| P_Fy-y\|^2, \qquad (1.5)$$

де $P_F=FF^+F(F^TF)^{-1}F^T$ — проекційна матриця.

Рішення має просту геометричну інтерпретацію. Похідною P_Fy є проекція цільового вектора *y* на лінійну оболонку стовпців матриці *F* (P_Fy-y) є проекція цільового вектора *y* на ортогональне доповнення цієї лінійної оболонки. Значення функціоналу $Q(a^*)=\| P_Fy-y\|^2$ є квадрат довжиною перпендикуляра, опущеного з *y* на лінійну оболонку. Таким чином, МНК знаходить найкоротший шлях від *y* до лінійної оболонки стовпців *F*.

Відома велика кількість численних методів розв'язку нормальної системи. Найбільшою популярністю користуються методи, засновані на ортогональних розкладах матриці *F*. Ці методи ефективні, володіють хорошою

чисельною стійкістю і дозволяють будувати різноманітні модифікації і узагальнення. Метод логістичної регресії заснований на досить сильних імовірнісних припущеннях, які мають відразу кілька цікавих наслідків. По-перше, лінійний алгоритм класифікації виявляється оптимальним Байєсівським класифікатором. По-друге, однозначно визначається функція витрат. По-третє, виникає цікава додаткова можливість поряд з класифікацією об'єкта отримувати чисельні оцінки ймовірності його приналежності кожному з класів.[2]

Базові припущення.

Нехай класів два, Y = {-1, 1}, об'єкти описуються n числовими ознаками f_j: $X \to R$, $j = 1,...,n$. Будемо вважати $X = R^n$, ототожнюючи об'єкти з їх ознаковими описами: $x \equiv (f_1(x), ..., f_n(x))$.

Гіпотеза 1. [1] Множина прецедентів $X \times Y$ є імовірнісним простором. Вибірка прецедентів $X^l = (x_i, y_i)_{i=1}^l$ отримана випадково і незалежно відповідно ймовірнісному розподілу з щільністю $p(x, y) = P_y p_y(x) = P(y|x)p(x)$, де P_y – апріорні ймовірності, $p_y(x)$ — функція правдоподібності, $P(y|x)$— апостеріорні ймовірності класів $y \in Y$.

Клас експонентних розподілів дуже широкий. До нього відносяться неперервні і дискретні розподіли: рівномірний, нормальний, гипергеометричний, пуассонівский, біноміальний, Г-розподіл, і інші.

Гіпотеза 2. [1] Функції правдоподібності класів $p_y(\mathrm{x})$ належать до експонентного сімейства щільностей, мають різні значення параметрів d і δ, але відрізняються значеннями параметра зсуву θ_y.

Основна теорема. [1] Нагадаємо, що оптимальний байєсівский класифікатор має вигляд $a(\mathrm{x}) = \arg\max_{y \in Y} \lambda_y P(\mathrm{y} \mid \mathrm{x})$, де λ_y — штраф за помилку на об’єктах класу y. В випадку двох класів:

$$a(\mathrm{x}) = \mathrm{sign}(\lambda_+ \mathrm{P}(+1 \mid \mathrm{x}) - \lambda_- \mathrm{P}(-1 \mid \mathrm{x})) = \mathrm{sign}(\frac{P(+1 \mid x)}{P(-1 \mid \mathrm{x})} - \frac{\lambda_-}{\lambda_+}).$$

Теорема 1. [1] Якщо справедливі гіпотези 1, 2, і серед ознак $f_1(x),..., f_n(x)$ є константа, то:

1) байєсівський класифікатор є лінійним: $a(x) = \text{sign}(\langle \omega, x \rangle - \omega_0)$, де $\omega_0 = \ln(\lambda_- / \lambda_+)$, а вектор ω не залежить від штрафів λ_-, λ_+;
2) апостеріорна ймовірність належності довільного об'єкта $x \in X$ класу $y \in \{-1,1\}$ може бути вирахувана за значенням дискримінантної функції: $P(y \mid x) = \sigma(\langle \omega, x \rangle y)$, де $\sigma(z) = \frac{1}{1+e^{-z}}$ — сигмоїдна функція.

Логістична регресія використовує сигмоїдну функцію або логістичну функцію, яка є складною функцією витрат.

Поліноміальна регресія – це тип регресії, який моделює нелінійний набір даних за допомогою лінійної моделі. Він схожий на множинні лінійні регресії, але підходить до нелінійної кривої між значенням *x* та відповідними умовними значеннями *y*. Припустимо, існує набір даних, який складається з точок даних, які присутні в нелінійному вигляді, тому в такому випадку лінійна регресія не найкраще підходить до цих точок даних. Для покриття таких точок даних нам потрібна поліноміальна регресія. Оригінальні ознаки перетворюються на поліноміальні особливості заданого ступеня, а потім моделюються за допомогою лінійної моделі. Що означає, що точки даних найкраще підходять за допомогою поліноміальної лінії. Рівняння для поліноміальної регресії також отримане з рівняння лінійної регресії, що означає лінійне рівняння регресії $Y = b_0 + b_1 x$, перетворюється в рівняння поліноміальної регресії $Y = b_0 + b_1 x + b_2 x^2 + b_3 x^3 + \cdots + b_n x^n$. Тут Y – прогнозований/цільовий вихід, $b_0, b_1, ... b_n$ – коефіцієнти регресії , *x* – наша незалежна/вхідна змінна. [13]

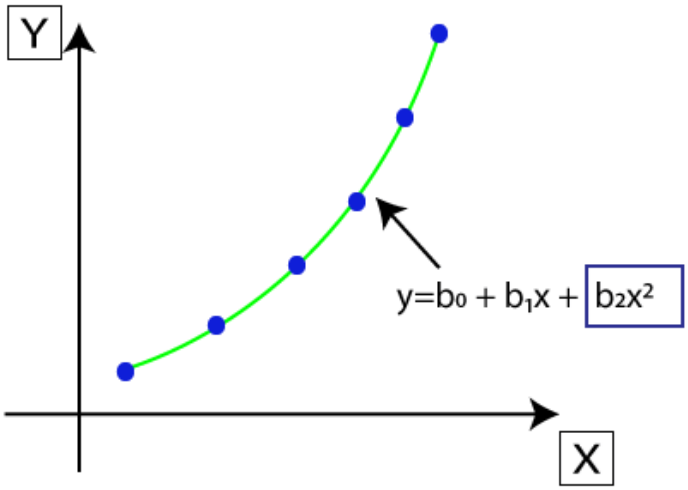


Рис.1

Support Vector Machine – це керований алгоритм навчання, який можна використовувати для регресії, а також проблем класифікації. Отже, якщо ми використовуємо його для проблем з регресією, то він називається SVR. Це алгоритм регресії, який працює для безперервних змінних. Нижче наведено кілька ключових слів, які використовуються у SVR: ядро – це функція, яка використовується для зіставлення даних з низькими розмірами у більш високі розмірні дані; гіперплощина у SVM – це лінія поділу між двома класами, але у SVR – це лінія, яка допомагає передбачити безперервні змінні та охоплює більшість точок даних; граничні лінії – це дві лінії, крім гіперплощини, що створює грань для точок даних; вектори підтримки – це точки даних, найближчі до гіперплощини та протилежного класу. У SVR ми завжди намагаємося визначити гіперплощину з максимальним запасом, щоб максимальна кількість точок даних була охоплена в цьому полі. Головною метою SVR є врахування максимальних точок даних у прикордонних лініях, а гіперплощина (найкраща лінія) повинна містити максимальну кількість точок даних.

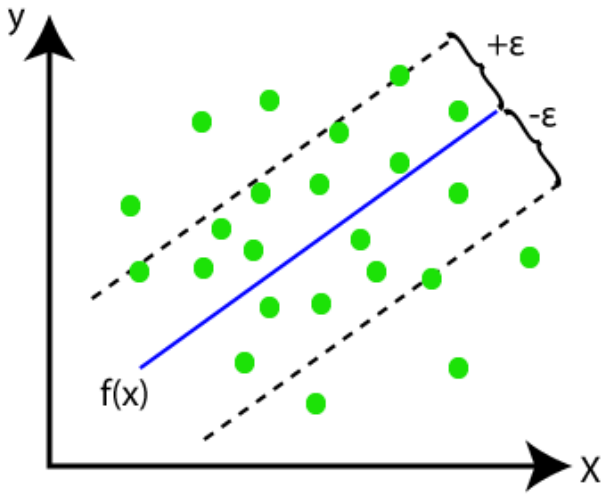


Рис. 2

Тут суцільна лінія називається гіперплощиною, а дві інші лінії відомі як граничні лінії.

Регресія дерева рішень. Дерево рішень – це керований алгоритм навчання, який можна використовувати для вирішення як класифікаційних, так і регресійних задач. Він може вирішити проблеми як для категоріальних, так і для числових даних. Регресія дерева рішення будує деревоподібну структуру, в якій кожен внутрішній вузол являє собою "тест" для атрибута, кожна гілка представляє результат тесту, а кожен вузол листя представляє остаточне рішення або результат.

Дерево рішення будується, починаючи з кореневого вузла/батьківського вузла (набору даних), який розбивається на лівий і правий дочірні вузли (підмножини набору даних). Ці дочірні вузли далі поділяються на їх дочірній вузол і самі стають батьківським вузлом цих вузлів. Розглянемо зображення нижче:

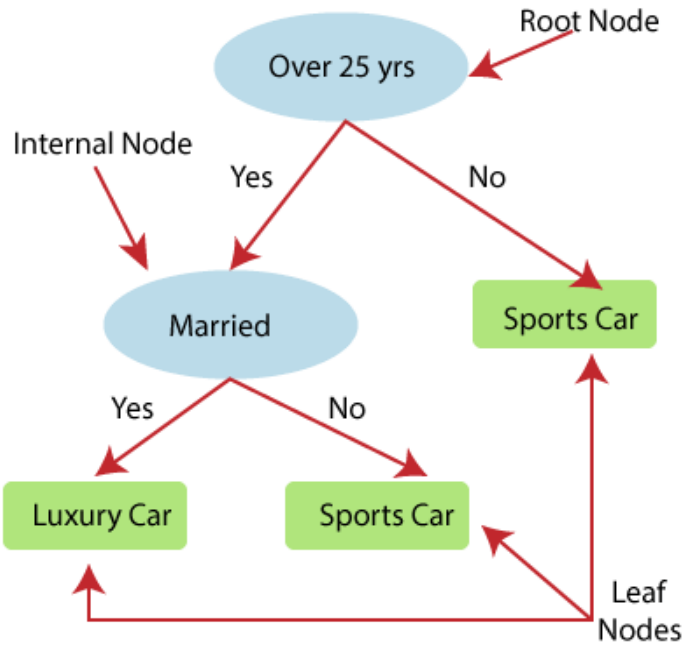


Рис.3. Дерево рішень

Тут модель намагається передбачити вибір людини між спортивними автомобілями або автомобілем класу люкс.

Випадковий ліс – це один із найпотужніших алгоритмів навчання, що контролюється, який здатний виконувати як регресію, так і класифікаційні завдання.

Регресія випадкових лісів – це метод ансамблевого навчання, який поєднує в собі декілька дерев рішень і прогнозує кінцевий результат, виходячи із середнього показника для кожного дерева. Комбіновані дерева рішень називаються базовими моделями і це можна представити більш формально як:

$$g(x) = f_0(x) + f_1(x) + f_2(x) + ...$$

Випадковий ліс використовує беггінг техніку, в якій агреговане дерево рішень працює паралельно і не взаємодіє між собою. За допомогою регресії випадкових лісів ми можемо запобігти надмірному розміщенню в моделі, створивши випадкові підмножини набору даних. [3]

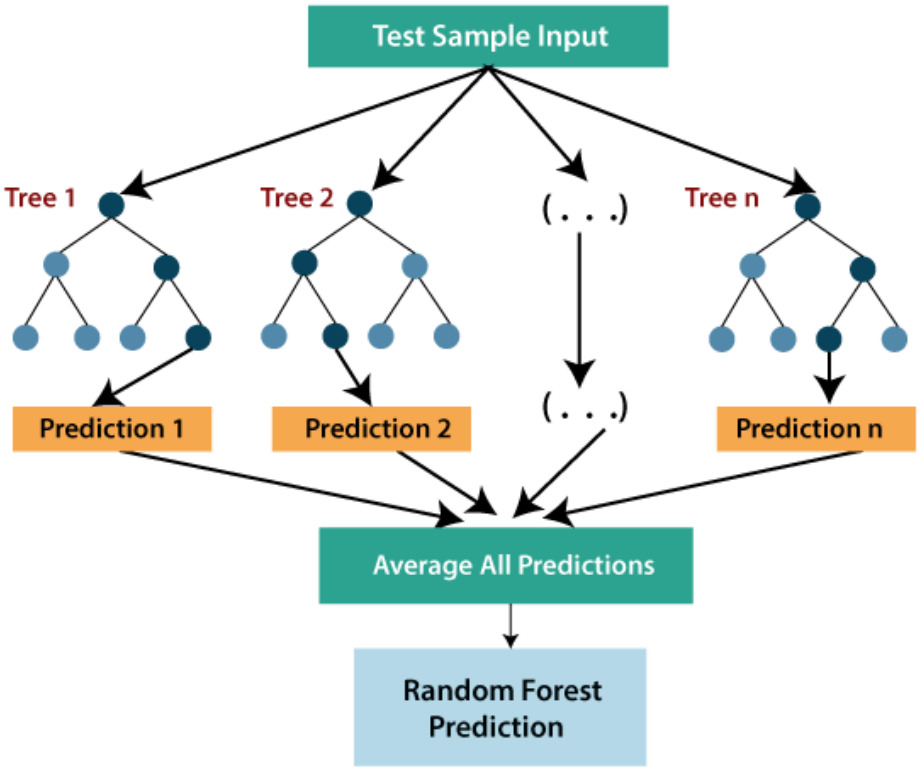


Рис. 4. Випадковий ліс

1.2.2. Класифікація

Задана множина об'єктів X, множина допустимих відповідей Y, і існує цільова функція $y^*: X \to Y$, значення якої $y_i = y^*(x_i)$ відомі тільки на кінцевій підмножині об'єктів $\{x_1,...,x_l\} \subset X$. Пари «об'єкт-відповідь» називають (x_i, y_i) прецедентами. Сукупність пар $X^l = (x_1, y_i)_{i=1}^l$ називається навчальною вибіркою.

Задача навчання за прецедентами полягає в тому, щоб по вибірці X^l встановити залежність y^*, тобто побудувати функцію вибору розв'язку $a: X \to Y$, яка наближала б цільову функцію $y^*(x)$, причому не тільки на об'єктах навчальної вибірки, але й на всій множині X.

Функція вибору розв'язку a повинна допускати ефективну комп'ютерну реалізацію; по цій причині називатимемо її алгоритмом.

Ознака f об'єкта x – це результат вимірювання деякої характеристики об'єкта. Формально ознакою називається відображення $f: X \to D_f$, де – D_f множина допустимих значень ознаки. Будь-який алгоритм теж можна $a: X \to Y$ розглядати як ознаку.

В залежності від природи множини ознаки діляться на декілька типів:

якщо $D_f = \{0,1\}$, то f – бінарна ознака;

якщо D_f – скінченна множина, то f – номінальна ознака;

якщо D_f – скінченна впорядкована множина, то f – порядкова ознака;

якщо $D_f = \mathbb{R}$, то f – кількісна ознака.

Якщо всі ознаки мають однаковий тип, $D_{f1} = ... = D_{f_n}$, то вихідні дані називаються однорідними, в протилежному випадку – різнорідними.

Нехай маємо набір ознак $f_1,..., f_n$. Вектор $(f_1(x),..., f_n(x))$ називають ознаковим описом об'єкта $x \in X$. В подальшому не будемо розрізняти об'єкти з і X їх ознакові описи, припускаючи $X = D_{f_1} \times ... \times D_{f_n}$. Сукупність ознакових описів всіх об'єктів вибірки X^l, записану у вигляді таблиці, називають матрицею об'єктів-ознак:

$$F = \left\| f_j(x_i) \right\|_{l \times n} = \begin{pmatrix} f_1(x_1) & ... & f_n(x_1) \\ ... & ... & ... \\ f_1(x_l) & ... & f_n(x_l) \end{pmatrix}. \qquad (1.6)$$

Матриця об'єктів-ознак є стандартним і найбільш поширеним способом представлення вихідних даних в прикладних задачах.

Відповіді і типи задач

В залежності від природи множини допустимих відповідей Y задачі навчання за прецедентами поділяються на наступні типи:

- якщо $Y = \{1,...,M\}$, то це задача класифікації на M неперетинних класах. В цьому випадку вся множина об'єктів X розбивається на класи $K_y = \{x \in X : y^*(x) = y\}$, і алгоритм $a(x)$ повинен давати відповідь на питання «якому класу належить x?». В деяких додатках класи називають образами і говорять про задачу розпізнавання образів;
- якщо $Y = \{0,1\}^M$, то це задача класифікації на M перетинних класах. В простішому випадку ця задача зводиться до рішення M незалежних задач класифікації з двома непересічними класами;

- якщо $Y = \mathbb{R}$, то це задача відновлення регресії.

Задачі прогнозування є частинними випадками класифікації чи відновлення регресії, коли $x \in X$ – опис минулої поведінки об'єкта x, $y \in Y$ – опис деяких характеристик його майбутньої поведінки.

Моделлю алгоритмів називається параметричне сімейство відображень $A = \{g(x,\theta) \mid \theta \in \Theta\}$, де $g : X \times \Theta \to Y$ – деяка фіксована функція, Θ – множина допустимих значень параметра θ, яка називається множиною параметрів або простором пошуку.

Метод навчання – це відображення $\mu : (X \times Y)^l \to A$, яке довільній кінцевій вибірці $X^l = (x_i, y_i)_{i=1}^l$ ставить у відповідність алгоритм $a \in A$. Метод μ будує алгоритм по вибірці. Метод навчання повинен допускати ефективну програмну реалізацію.

Отже, в задачах навчання за прецедентами чітко розрізняють два етапи. На етапі навчання метод μ за вибіркою X^l будує алгоритм $a = \mu(X^l)$. На етапі застосування алгоритм a для нових об'єктів x видає відповіді $y = a(x)$. Етап навчання є найбільш складним. Як правило, він зводиться до пошуку параметрів моделі, що доставляють оптимальне значення заданому функціоналу якості. [3]

Класичний метод навчання, який називається мінімізацією емпіричного ризику полягає в тому, щоб знайти в заданій моделі A алгоритм a, який доставляє мінімальне значення функціоналу якості Q на заданій навчальній вибірці X^l:

$$\mu(X^l) = \arg\min_{a \in A} Q(a, X^l). \qquad (1.7)$$

Байєсівський підхід є класичним в теорії розпізнавання образів і лежить в основі багатьох методів. Він спирається на теорему про те, що якщо щільності розподілу класів відомі, то алгоритм класифікації, що має мінімальну ймовірність помилок, можна виписати в явному вигляді. Для оцінювання

щільності класів за вибіркою застосовуються різні підходи. Розглянемо три: параметричний, непараметричний і оцінювання суміші розподілів.

Ймовірнісна постановка задачі класифікації

Нехай X – множина об'єктів, Y – кінцева множина імен класів, множина $X \times Y$ є ймовірнісним простором зі щільністю розподілу $p(x, y) = P(y)p(x \mid y)$. Ймовірності появи об'єктів кожного з класів $P_y = P(y)$ називаються апріорними ймовірностями класів. Щільності розподілів $p_y(x) = p(x \mid y)$ називаються функціями правдоподібності класів. Ймовірнісна постановка задачі класифікації поділяється на дві незалежні підзадачі.[15]

Задача 1. Є проста вибірка $X^l = (x_i, y_i)_{i=1}^l$ з невідомого розподілу $p(x, y) = P_y p_y(x)$. Потрібно побудувати емпіричні оцінки апріорних ймовірностей $\hat{P}_y$ і функцій правдоподібності $\hat{p}_y(x)$ для кожного з класів $y \in Y$.

Задача 2. За відомими щільностями розподілу $p_y(x)$ і апріорними щільностями P_y всіх класів $y \in Y$ побудувати алгоритм $a(x)$, що мінімізує ймовірність помилкової класифікації.

Друга задача вирішується відносно легко. Перша задача має безліч рішень, оскільки багато розподілів $p(x, y)$ могли б породити одну і ту ж вибірку X^l. Доводиться залучати різні припущення про щільності, що і призводить до великої різноманітності байєсівских методів.

Непараметрична класифікація

Непараметричні методи класифікації засновані на локальному оцінюванні щільності розподілу класів $p_y(x)$ в околі класифікуючого об’єкта $x \in X$. Для класифікації об'єкта x застосовується основна формула.

Непараметричні оцінки щільності

Локальне оцінювання спирається на саме визначення щільності. Почнемо з найпростіших одновимірних оцінок. Вони вже можуть виявитися корисними

на практиці, зокрема, при побудові «наївних» байєсівських класифікаторів. Крім того, вони підкажуть ідеї узагальнення на багатовимірний випадок. [18]

Дискретний випадок. Нехай X – кінцева множина, причому $|X| \ll m$.

Оцінкою щільності служить гістограма значень x_i, які зустрілися в вибірці $X^m = (x_i)_{i=1}^m$:

$$\hat{p}(x) = \frac{1}{m}\sum_{i=1}^{m}[x_i = x]. \qquad (1.8)$$

Ця оцінка не може бути застосована, тим більше, в неперервному випадку, так як її значення майже завжди дорівнюватиме нулю.

Одновимірний неперервний випадок. Нехай $X = \mathbb{R}$. Згідно з визначенням щільності, $p(x) = \lim_{h\to 0}\frac{1}{2h}P[x-h, x+h]$, де $P[a,b]$ – ймовірнісна міра відрізка $[a,b]$. Відповідно, емпірична оцінка щільності визначається як частка точок вибірки, що лежать всередині відрізка $[x-h, x+h]$, де h – невід'ємний параметр,який називається шириною вікна:

$$\hat{p}_h(x) = \frac{1}{2mh}\sum_{i=1}^{m}[|x - x_i| < h]. \qquad (1.9)$$

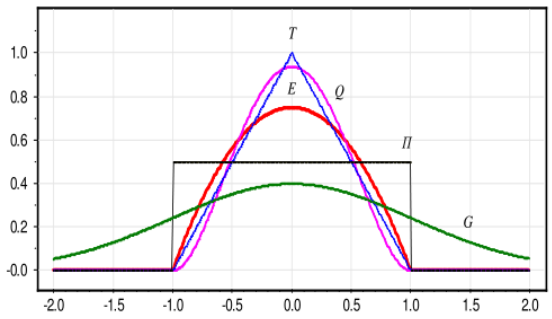
Функція $\hat{p}_h(x)$ є кусково-сталою, що призводить до появи широких зон невпевненості, в яких максимум досягається одночасно для декількох класів $y \in Y$. Проблема вирішується за допомогою локальної непараметричної оцінки Парзена-Розенблатта:

$$\hat{p}_h(x) = \frac{1}{mh}\sum_{i=1}^{m}K\left(\frac{x - x_i}{h}\right), \qquad (1.10)$$

де $K(z)$ – функція, яка називається, парна і нормована $\int K(z)dz = 1$.

Функція $\hat{p}_h(x)$ має ту ж ступінь гладкості, що і ядро $K(z)$, і, завдяки нормуванню, дійсно може інтерпретуватися як щільність ймовірності: $\int \hat{p}_h(x)dx = 1$ при будь-якому h.

Функція ядра K практично не впливає на якість відновлення щільності і на якість класифікації. У той же час, вона визначає ступінь гладкості функції $\hat{p}_h(x)$. Вид ядра може також впливати на ефективність обчислень. Гаусівське ядро G вимагає перегляду всієї вибірки для обчислення значення $\hat{p}_h(x)$ в довільній точці x. Ядра *E, Q, T, П* є фінітними (мають обмеженого носія, рис.5), і для них досить взяти тільки ті точки вибірки, які потрапляють в окіл точки x радіуса h.



Часто використовувані ядра:

E – Епанєчнікова;

Q – квартичне;

T – трикутне;

G – гаусівське;

П – прямокутне.

Рис. 5

Метод опорних векторів

Розглянемо задачу класифікації на два непересічних класи, в якій об'єкти описуються *n*-мірними векторами: $X = \mathbb{R}^n, Y = \{-1; +1\}$.

Будемо будувати лінійний пороговий класифікатор:

$$a(x) = sign(\sum_{j=1}^{n} w_j x^j - w_0) = sign(\langle w, x \rangle - w_0), \qquad (1.11)$$

де $x = (x^1, ..., x^n)$ – ознаковий опис об'єкта x; вектор $w = (w^1, ..., w^n) \in \mathbb{R}^n$ і скалярний поріг $w_0 \in \mathbb{R}$ є параметрами алгоритму. Рівняння $\langle w, x \rangle = w_0$ описує гіперплощину, що розділяє класи в просторі $\mathbb{R}^n$.

Припустимо, що вибірка $X^l = (x_i, y_i)_{i=1}^l$ лінійно роздільна і існують значення параметрів w, w_0, при яких функціонал числа помилок

$$Q(w, w_0) = \sum_{i=1}^{l} [y_i(\langle w, x_i \rangle - w_0) \le 0] \qquad (1.12)$$

приймає нульове значення. Але тоді роздільна гіперплощина не єдина. Можна вибрати інші її положення, що реалізують таке ж розбиття вибірки на два класи.

Ідея методу полягає в тому, щоб розумним чином розпорядитися цією свободою вибору.

*Оптимальна розділююча гіперплощина***.** Вимагатимемо, щоб роздільна гіперплощина максимально далеко відстояла від найближчих до неї точок обох класів. Спочатку даний принцип класифікації виник з евристичних міркувань: цілком природно вважати, що максимізація зазору між класами повинна сприяти більш надійній класифікації. Пізніше цей принцип отримав і теоретичне обґрунтування.

Нормування. Зауважимо, що параметри лінійного порогового класифікатора визначені з точністю до нормування: алгоритм *a(x)* не зміниться, якщо *w* і *w0* тимчасово помножити на одну і ту ж додатну константу. Зручно вибрати цю константу таким чином, щоб виконувалася умова

$$\min_{i=1,\dots,l} y_i(\langle w, x_i \rangle - w_0) = 1. \qquad (1.13)$$

Безліч точок $\{x : -1 \le \langle w, x \rangle - w_0 \le 1\}$ описує смугу, що розділяє класи (рис.6). Жоден з об'єктів навчальної вибірки не потрапляє всередину цієї смуги. Межами смуги служать дві паралельні гіперплощини з вектором нормалі *w*. Розділююча гіперплощина проходить рівно по середині між ними. Об'єкти, найближчі до роздільної гіперплощини, лежать на кордонах смуги, і саме на них досягається мінімум. У кожному з класів є хоча б один такий об'єкт, в іншому випадку роздільну смугу можна було б ще трохи розширити, порушуючи тим самим би принцип максимального зазору.

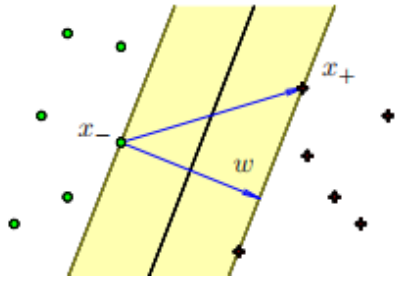


Рис. 6

Лінійно роздільна вибірка. Навчальні об'єкти x_- і x_+ знаходяться на кордоні роздільної смуги. Вектор нормалі *w* до розділюючої гіперплощини визначає ширину смуги.

Ширина роздільної смуги. Щоб розділююча гіперплощина якомога далі розташовувалася від точок вибірки, ширина смуги повинна бути

максимальною. Нехай x_- і x_+ – два навчальних об'єкта класів –1 і +1 відповідно, що лежать на кордоні смуги. Тоді ширина смуги є

$$\left\langle (x_+ - x_-), \frac{w}{\|w\|} \right\rangle = \frac{\langle w, x_+ \rangle - \langle w, x_- \rangle}{\|w\|} = \frac{(w_0 + 1) - (w_0 - 1)}{\|w\|} = \frac{2}{\|w\|}.$$

Ширина смуги максимальна, коли норма вектора *w* мінімальна. [1]

1.2.3. Приклади задач класифікації зображень та діагностики захворювань

Приклад 1. У завданнях медичної діагностики в ролі об'єктів виступають пацієнти. Ознаки характеризують результати обстежень, симптоми захворювання і методи лікування, які застосовувалися. Приклади бінарних ознак – стать, наявність головного болю, слабкості, нудоти, і т.д. Порядкова ознака – тяжкість стану (задовільний, середньої тяжкості, важкий, вкрай важкий). Кількісні ознаки – вік, пульс, артеріальний тиск, вміст гемоглобіну в крові, доза препарату, і т.д. Ознаковий опис пацієнта є, по суті справи, формалізованої історією хвороби. Накопичивши достатню кількість прецедентів, можна вирішувати різні завдання: класифікувати вид захворювання (диференціальна діагностика); визначати найбільш доцільний спосіб лікування; прогнозувати тривалість і результат захворювання; оцінювати ризик ускладнень; знаходити синдроми – найбільш характерні для даного захворювання сукупності симптомів. Цінність такого роду систем в тому, що вони здатні миттєво аналізувати і узагальнювати величезну кількість прецедентів – можливість, недоступна людині. [8]

Приклад 2. Задача розпізнавання (класифікації) іриса на 3 класи. Тут цільова змінна Y∈{setosa, versicolor, virginica}, $X_1, X_2, \ldots, X_n \in \mathbb{R}$.

Рис.7

Ознаки:

- довжина чашелистика (см)
- ширина чашелистика (см)
- довжина пелюстки (см)
- ширина пелюстки (см)

Приклад 3. Задача розпізнавання рукописних цифр. Цільова змінна $Y \in \{0,1,2,\ldots,9\}$, ознаки $X_1, X_2, \ldots, X_{784} \in [0,255]$ – пікселі зображення розміром 28×28. Приклад об'єктів:

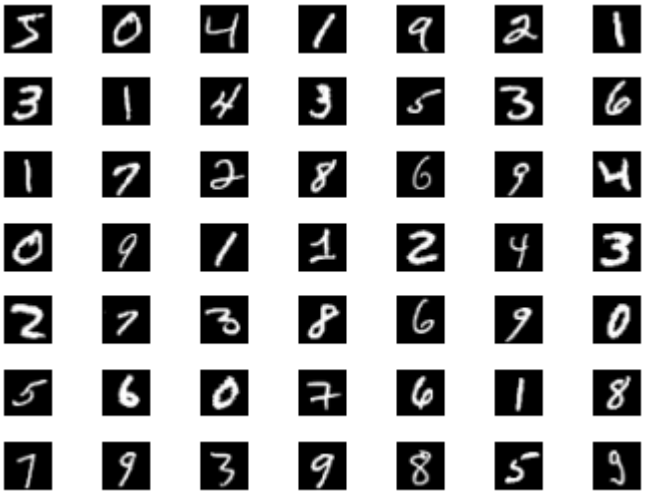


Рис.8

Дані: http://yann.lecun.com/exdb/mnist/

Приклад 4. По набору певних характеристик пацієнта (симптомів), таких як температура тіла, артеріальний тиск, вміст гемоглобіну в крові і т.п., потрібно визначити, яке у хворого захворювання (і хворий він взагалі). Об'єктами є пацієнти, їх признаковая описом – набір характеристик, а виходом – номер класу.

Можуть зустрічатися ознаки різних типів:

• бінарні (стать, наявність головного болю),

• номінальні (біль може бути тупий, ріжучий, колючий і т.п.),

•порядкові (стан хворого може бути задовільним, середньої тяжкості, тяжкий, вкрай важкий),

• кількісні (температура тіла, пульс, тиск).

Є дані про 114 осіб із захворюванням щитовидної залози.

У 61 – підвищений рівень вільного гормону Т4, у 53 – рівень гормону в нормі.

Ознаки:

- heart – частота серцевих скорочень (пульс),
- SDNN – стандартне відхилення тривалості RR-інтервалів.

Можна навчитися передбачати (допускаючи невеликі помилки) рівень вільного Т4 по heart і SDNN у нових пацієнтів. [7]

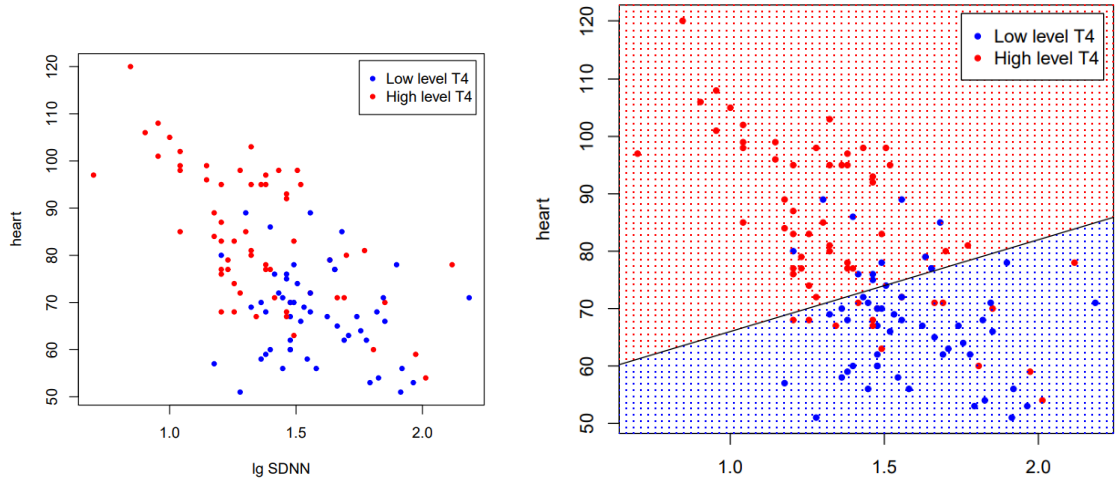


Рис.9

Розділ ІІ. Задачі машинного навчання без учителя

2.1. Кластеризація даних

Кластеризація – це задача, яка полягає у розбитті сукупності предметів на групки, які називаються кластерами. Усередині окремої групи мають знаходитися «подібні» предмети, а предмети різних груп повинні найбільш відрізнятися. У задачі кластеризації перелік груп чітко не поданий і визначається під час роботи алгоритму.

Кластер має такі математичні ознаки: центр, радіус, середньоквадратичне відхилення, розмір кластера. Центр кластера – це середнє геометричне місце точок в просторі змінних. Радіус кластера – максимальна відстань точок від центру кластера. Спірний об'єкт – це об'єкт, який у міру подібності може бути віднесений до кількох кластерів. Розмір кластера може бути визначений або по радіусу кластера, або по середньоквадратичному відхиленню об'єктів для цього кластера. Об'єкт відноситься до кластеру, якщо відстань від об'єкта до центру кластера менше радіуса кластера. Якщо ця умова виконується для двох і більше кластерів, об'єкт є спірним. Невизначеність такої задачі може усунути аналітик. Принцип кластерного аналізу спирається на такі гіпотези. Перша – ознаки об'єкта, які розглядаються, допускають розбиття сукупності об'єктів на кластери. Друга – правильність вибору масштабу або одиниць вимірювання ознак.

Застосування кластерного аналізу в узагальненому вигляді:

- Визначення вибірки.
- Підбір великої кількості змінних, за допомогою яких відбудеться оцінювання об'єктів у вибірці. У разі потреби провести нормалізацію.
- Знаходження міри схожості. Застосування методу кластерного аналізу для створення кластерів.
- Кінцевий результат.

Після аналізу є можливість скорегувати метрику і методу кластеризації для отримання оптимального результату. Для того, що визначити визначати

«подібність» об'єктів складаємо вектор характеристик для кожного об'єкта – як правило, це набір числових значень, наприклад, зростання-вага людини, проводимо нормалізацію (необхідно, щоб всі дані знаходилися в одному діапазоні). Наступним етапом є ступінь подібності. Основні метрики:

Евклідова відстань є найпоширенішою (геометрична відстань в багатовимірному просторі):

$$p(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2} \tag{2.1}$$

Квадрат евклідової відстані. Використовується для додання більшої ваги більш віддаленим один від одного об'єктів:

$$p(x, x') = \sum_{i}^{n} (x_i - x_i')^2 \tag{2.2}$$

Манхетенська відстань. Це відстань є усередненим значенням різниць по координатах:

$$p(x, x') = \sum_{i}^{n} |x_i - x_i'| \tag{2.3}$$

Степенева відстань. Застосовується в разі, коли необхідно збільшити або зменшити вагу, що відноситься до розмірності, для якої відповідні об'єкти сильно відрізняються::

$$p(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p} \tag{2.4}$$

де r і p – параметри, які задаються вручну. Параметр p відповідає за поступове зважування різниць за окремими координатами, параметр r відповідальний за прогресивне зважування великих відстаней між об'єктами. Характерно, що вибір метрики повністю залежить від дослідника, оскільки результати кластеризації можуть істотно відрізнятися при використанні різних методів. [6]

Алгоритми кластерного аналізу

Всі методи можна умовно поділити на 3 основні категорії, а саме:

- Ітераційні
- Ієрархічні
- Щільнісні

Ітераційні методи виявляють більш високу стійкість по відношенню до шумів і викидів, некоректного вибору метрики, включенню незначущих змінних в набір, який бере участь в кластеризації. Аналітик повинен наперед визначити кількість кластерів, кількість ітерацій або правило зупинки, а також деякі інші параметри кластеризації. Якщо немає припущень щодо числа кластерів, рекомендують використовувати ієрархічні алгоритми. Однак якщо обсяг вибірки не дозволяє це зробити, можливий шлях – проведення ряду експериментів з різною кількістю кластерів, наприклад, почати розбиття сукупності даних з двох груп і, поступово збільшуючи їх кількість, порівнювати результати. За рахунок такого "варіювання" результатів досягається досить велика гнучкість кластеризації.

Алгоритм k-середніх(k-means). Це ітеративний алгоритм, який намагається розділити набір даних на *K* заздалегідь визначениз непересетинних підгрупи (кластери), де кожна точка даних належить лише одній групі. Він намагається зробити точки даних між кластерами максимально схожими, зберігаючи кластери якомога більш (далеко). Він призначає точки даних кластеру таким чином, що сума квадратичної відстані між точками даних та центроїдом кластера (середнє арифметичне всіх точок даних, що належать до цього кластера), є мінімальним. Чим менше варіацій у нас в кластерах, тим гомогеннішими (подібними) точки даних є в межах одного кластеру.

Алгоритм працює так:

- Вказати кількість кластерів К.
- Ініціалізувати центроїди, спочатку перемішуючи набір даних, а потім випадковим чином вибираючи К точки для даних центроїдів

без заміни.

- Продовжувати ітерацію, поки не зміниться центроїд, тобто призначення точок даних кластерам не змінюється.
- Обчислити суму квадратної відстані між точками даних та всіма центроїдами.
- Призначити кожну точку даних найближчому кластеру (центроїд).
- Обчисліть центроїди для кластерів, взявши середнє значення всіх точок даних, що належать кожному кластеру.

Підхід kmeans до вирішення проблеми називається Очікування-Максимізація. О-крок – це присвоєння точок даних найближчому кластеру. М–крок обчислює центроїд кожного кластеру. Цільова функція:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \left\| x^i - \mu_k \right\|^2 \qquad (2.5)$$

де w_{ik} = 1 для точки x_i даних, якщо вона належить кластеру k; в іншому випадку w_{ik} = 0. Також μ_k є центроїдом кластера x_i. Це проблема мінімізації двох частин. Спочатку мінімізуємо J w_{ik} і обробляємо μ_k фіксовано. Потім ми мінімізуємо J μ_k і обробляємо w_{ik} фіксованим. Тут можна відзначити кілька речей: оскільки алгоритми кластеризації, включаючи kmeans, використовують вимірювання на основі відстані для визначення схожості між точками даних, рекомендується стандартизувати дані середнім рівнем нуля та стандартним відхиленням, оскільки майже завжди функції будь-якого набору даних мають різні одиниці вимірювання наприклад, вік проти доходу.

Враховуючи ітеративний характер kmeans та випадкову ініціалізацію центроїдів на початку алгоритму, різні ініціалізації можуть призвести до різних кластерів, оскільки алгоритм kmeans може застрягнути в локальному оптимумі та може не сходитися до глобального оптимуму . Тому рекомендується запускати алгоритм, використовуючи різні ініціалізації центроїдів і вибирати результати пробігу, які давали меншу суму відстані у квадраті.

Основна ідея – на кожній ітерації заново вираховується центр мас, для кожного кластера, потім вектори розбиваються на нові класи, відповідно до того який з отриманих центрів виявився ближчим за метрикою.

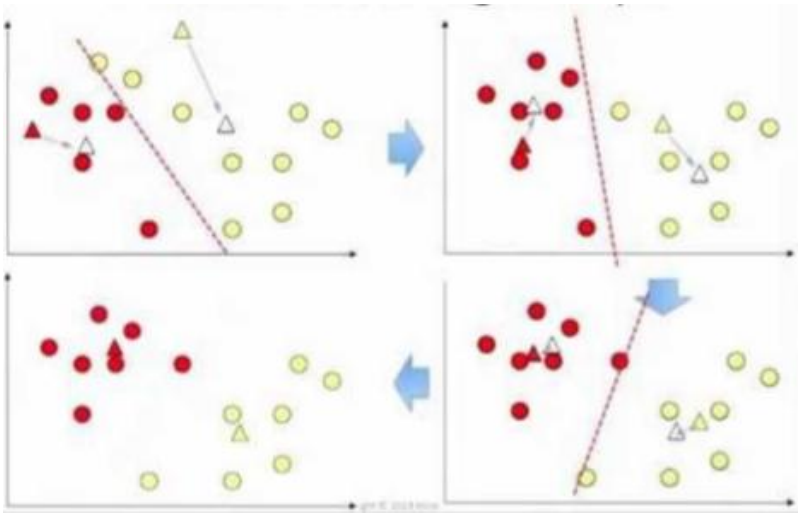


Рис.1 Алгоритм K-means

Переваги алгоритму k-середніх: простота використання; швидкість використання; зрозумілість і прозорість алгоритму.

Недоліки алгоритму k-середніх: алгоритм занадто чутливий до викидів, які можуть спотворювати середнє.

Алгоритм EM(Expectation–maximization). EM-алгоритм — алгоритм, що використовується для знаходження оцінок максимальної схожості параметрів ймовірних моделей, у випадку, коли модель залежить від деяких прихованих змінних. Кожна ітерація алгоритму складається з двох кроків. На E-кроці (expectation) вираховується очікуване значення функції правдоподібності, при цьому приховані змінні розглядаються як спостережувані. На M-кроці (maximization) вираховується оцінка максимальної схожості, таким чином збільшується очікувана схожість, вирахувана на E-кроці. Потім це значення використовується для E-кроку на наступній ітерації. Алгоритм виконується до збіжності.

За певних обставин зручно розглядати EM-алгоритм як два чергуються кроку максимізації. Розглянемо функцію:

$$F(q,\theta) = E_q[\log L(\theta; x, Z)] + H(q) = -D_{KL}(q||pz|x(\cdot\,|x;\theta)) + \log L(\theta; x)$$

де q – розподіл ймовірностей неспостережуваних змінних Z;

pZ|X(· |x;θ) – умовний розподіл неспостережуваних змінних при фіксованих спостережуваних x і параметрах розподілення ймовірностей неспостережуваних змінних θ;

H – ентропія і DKL – відстань Кульбака-Лейблера.

Тоді кроки EM-алгоритму можна показати як:

E(xpectation) крок: Вибираємо q, щоб максимізувати F:

$$q^{(t)} = arg\max_{q} F(q, \theta^{(t)}) \tag{2.6}$$

M(aximization) крок: Вибираємо θ, щоб максимізувати F:

$$\theta^{(t+1)} = arg\max_{\theta} F\left(q^{(t)}, \theta\right) \tag{2.7}$$

Алгоритм EM простий і легкий в реалізації, не чутливий до ізольованим об'єктам і швидко сходиться при вдалій ініціалізації. Однак він вимагає для ініціалізації вказівки кількості кластерів k, що має на увазі наявність апріорних знань про дані. Крім того, при невдалій ініціалізації збіжність алгоритму може виявитися повільною або може бути отриманий неякісний результат. [4]

Алгоритм PAM(partitioning around Medoids). Алгоритм PAM заснований на пошуку k репрезентативних об'єктів або медоїдів серед спостережень набору даних. Після знаходження набору k медоїдів кластери будуються шляхом присвоєння кожного спостереження найближчому медоїду. Далі, кожен обраний медоїдний m і кожна немедоїдна точка даних обмінюються і обчислюється цільова функція. Цільова функція відповідає сумі несхожості всіх об'єктів до їх найближчого медоїда. Крок SWAP намагається покращити якість кластеризації шляхом обміну виділеними об'єктами (медоїдами) та невибраними об'єктами. Якщо об'єктивну функцію можна зменшити, обмінявши вибраний об'єкт з невибраним об'єктом, тоді здійснюється SWAP. Це продовжується до тих пір, поки об'єктивну функцію більше не можна зменшити. Мета – знайти k репрезентативних об'єктів, які мінімізують суму відмінностей спостережень до їх найближчого репрезентативного об'єкта. Підсумовуючи, алгоритм PAM протікає у дві фази наступним чином:

Фаза збірки:

1. Вибрати k об'єктів, щоб стати медоїдами, або, якщо ці об’єкти надані, використовувати їх як медоїди;
2. Обчислити матрицю несхожості, якщо вона не була надана;
3. Призначити кожен предмет до його найближчого медоїда.

Фаза заміни:

1. для кожного пошуку кластера, якщо будь-який об’єкт кластера зменшує середній коефіцієнт несхожості; якщо це так, вибрати сутність, яка зменшує цей коефіцієнт найбільше як медоїд для цього кластера;
2. Якщо хоча б один медоїд змінився, перейти до (3), інакше закінчити алгоритм.

Як було зазначено вище, алгоритм PAM працює з матрицею несхожості, і для обчислення цієї матриці алгоритм може використовувати дві метрики: Евклідову і Манхетенську.[5]

Алгоритм BIRCH. Завдяки узагальненому вигляду кластерів, швидкість кластеризації збільшується, алгоритм BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) при цьому володіє великим масштабуванням. У цьому алгоритмі реалізований двоетапний процес кластеризації. В ході першого етапу формується попередній набір кластерів. На другому етапі до виявлених кластерів застосовуються інші алгоритми кластеризації – придатні для роботи в оперативній пам'яті. Аналогія, що описує цей алгоритм. Якщо кожен елемент даних варто уявити собі як намистину, що лежить на поверхні столу, то кластери намистин можна "замінити" тенісними кульками і перейти до більш детального вивчення кластерів тенісних кульок. Число намистин може виявитися досить велике, проте діаметр тенісних кульок можна підібрати таким чином, щоб на другому етапі можна було, застосувавши традиційні алгоритми кластеризації, визначити дійсну складну форму кластерів.

Алгоритм WaveCluster. WaveCluster є алгоритмом кластеризації заснованим на основі хвильових перетворень. На початку роботи алгоритму

дані узагальнюються шляхом накладення на простір даних багатовимірної решітки. На подальших кроках алгоритму аналізуються не окремі точки, а узагальнені характеристики точок, які потрапили в одну клітинку решітки. В результаті такого узагальнення необхідна інформація вміщується в оперативній пам'яті. На наступних кроках для визначення кластерів алгоритм застосовує хвильове перетворення до узагальнених даних.

Головні особливості WaveCluster:

• складність реалізації;

• алгоритм може виявляти кластери довільних форм;

• алгоритм не чутливий до шумів;

• алгоритм може бути застосований тільки до даних низької розмірності.

Алгоритм CLARA. (Clustering Large Applications, (Kaufman and Rousseeuw 1990)) – це розширення до методів k-medoids (PAM) для обробки даних, що містять велику кількість об'єктів (більше декількох тисяч спостережень) з метою скорочення часу обчислення та зберігання оперативної пам’яті. Це досягається за допомогою підходу вибірки. Замість пошуку медоїдів (в кластерному аналізі – об'єкт, що належить набору даних або кластеру, відмінність (наприклад, за координатами) якого з іншими об'єктами в наборі даних або кластері мінімальна) для всього набору даних CLARA розглядає невелику вибірку даних із фіксованим розміром та застосовує алгоритм PAM для створення оптимального набору медоїдів для вибірки. Якість отриманих медоїдів вимірюється середньою різницею між кожним об'єктом у всій наборі даних та медоїдом його кластеру, визначеним як функція витрат.

CLARA повторює процеси вибірки та кластеризації заздалегідь задану кількість разів, щоб мінімізувати зміщення вибірки. Кінцеві результати кластеризації відповідають набору медоїдів з мінімальними витратами.

Алгоритм такий:

1. Створити випадковим чином з початкового набору даних кілька підмножин із фіксованим розміром.

2. Обчислити алгоритм PAM для кожної підмножини та вибрати відповідні k репрезентативні об'єкти (медоїди). Призначити кожне спостереження за всіма наборами даних найближчим медоїдом.
3. Обчислити середню (або суму) відмінностей спостережень до їх найближчого медоїда.
4. Зберегти підмножину даних, для якої середнє значення (або сума) мінімальна. Подальший аналіз проводиться на останньому розділі.

Кожен набір підданих змушений містити медоїди, отримані з найкращого набору підданих до цього часу. До цього набору додаються випадково проведені спостереження, поки не буде досягнуто вибірки.

Щільнісні методи. Ця група розглядає кластери, як території простору даних з високою щільністю об'єктів, які розділені територіями з низькою щільністю. Їх часто застосовуються для фільтрації шуму та знаходження кластерів довільної форми. [6]

Алгоритм DBSCAN (Просторова кластеризація із шумом на основі щільності) – популярний метод машинного навчання, який використовується в алгоритмах побудови моделі. Методи навчання без нагляду (учителя) – це коли немає чіткої мети чи результату, яку ми прагнемо знайти. Натомість ми групуємо дані разом на основі подібності спостережень. DBSCAN – це метод кластеризації. що використовується в машинному навчанні для відділення кластерів високої щільності від кластерів низької щільності. Зважаючи на те, що DBSCAN – алгоритм кластеризації на основі щільності, він робить велику роботу в пошуку областей у даних, які мають високу щільність спостережень, порівняно з областями даних, які не дуже щільні за спостереженнями. DBSCAN також може сортувати дані в кластери різної форми, що є ще однією сильною перевагою. DBSCAN працює так:

- Розділяє набір даних на n вимірів;
- Для кожної точки в наборі даних DBSCAN формує *n* розмірну форму навколо цієї точки даних, а потім підраховує, скільки точок даних потрапляє під цю форму;

- DBSCAN вважає цю форму кластером. DBSCAN ітеративно розширює кластер, проходячи кожну окрему точку в кластері та підраховуючи кількість інших точок даних поблизу.

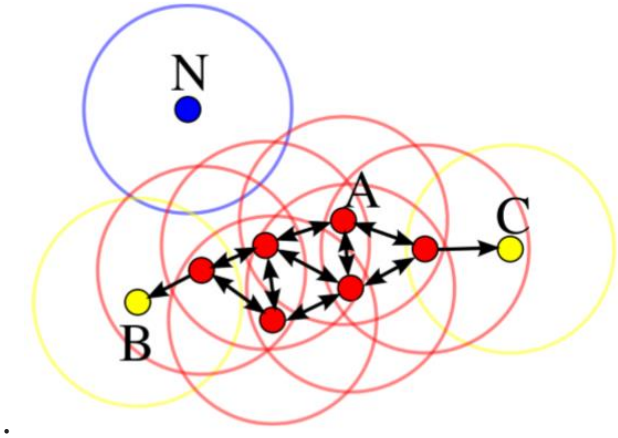


Рис.2. Алгоритм DBSCAN

Для пошуку кластерів алгоритм DBSCAN перевіряє ε-околицю кожного об'єкта (Рис.2). Якщо ε-околиця об'єкта p містить більше точок ніж MinPts, то створюється новий кластер з кореневим об'єктом p. Потім DBSCAN ітеративно збирає об'єкти безпосередньо щільно-досяжні з кореневих об'єктів, які можуть привести до об'єднання кількох щільно-досяжних кластерів. Процес завершується, коли ні до одного кластеру не може бути додано жодного нового об'єкта. Хоча, на відміну від методів розбиття, DBSCAN не вимагає заздалегідь вказувати число одержуваних кластерів, виникне потреба у вказівках значень параметрів ε і MinPts, які безпосередньо впливають на результат кластеризації. Оптимальні значення цих параметрів складно визначити, особливо для багатовимірних просторів даних. [4]

2.2. Зменшення розмірності даних

У проблемах класифікації машинного навчання часто є занадто багато факторів, на основі яких робиться остаточна класифікація. Ці фактори в основному є змінними, які називаються ознаками. Чим більша кількість функцій, тим складніше візуалізувати навчальний набір, а потім працювати над ним. Іноді більшість цих особливостей є співвіднесеними, а значить, і зайвими. Тут вступають у дію алгоритми зменшення розмірності. Зменшення

розмірності – це процес зменшення кількості розглянутих випадкових змінних шляхом отримання набору основних змінних. Його можна розділити на вибір функції та вилучення функцій.

Обирання ознак — це процес пошуку підмножини первісних змінних (ознак або властивостей) для використання в побудові моделі. Є три стратегії: фільтрування (наприклад, отримання інформації), обгортання (наприклад, пошук, який керується точністю), вкладення або вбудування (ознаки обираються для додавання або видалення при створенні моделі ґрунтуючись на помилках прогнозування).

Конструювання ознак перетворює дані з багатовимірного простору в простір невеликої кількості вимірів. Таке перетворення може бути лінійним, як в методі головних компонент, проте також існує багато методів нелінійного зниження розмірності. Для багатовимірних даних можна використовувати тензорнее представлення для скорочення розмірності через навчання полілінійного підпростору.

Метод головних компонент (МГК) – один з основних способів зменшення розмірності даних з втратою мінімальної кількості інформації, розроблений Карлом Пірсоном (Karl Pearson) в 1901р. Застосовується в багатьох областях, таких як розпізнавання образів, комп'ютерний зір, стиснення даних і т.п.

Основним завданням методу головних компонент є заміна вихідних даних на якісь агреговані значення в новому просторі, вирішуючи при цьому два завдання – першt з яких полягає в об'єднанні найбільш важливих (з точки зору мінімізації середньоквадратичної помилки) значень в меншу кількість параметрів, але найбільш інформативних (зменшення розмірності простору даних), а друга – зменшити шум в даних.

Для вирішення цієї проблеми МГК шукає простір, який найкращим чином відображає дисперсію даних. Напрямок з найбільшою прогнозованою дисперсією називається першою головною компонентою. Ортогональний напрямок, який захоплює другу за величиною прогнозовану дисперсію,

називається другою головною компонентою і так далі. Зауважимо, той напрямок, який максимізує дисперсію, мінімізує середньоквадратичну помилку.

Знаходження головних компонент зводиться до обчислення власних векторів і власних значень коваріаційної матриці вихідних даних. Іноді метод головних компонент називають перетворенням Кархунена-Лоева (Karhunen-Loeve) або перетворенням Хотеллінга (Hotelling transform).

Розклад невід'ємних матриць (РНМ). РНМ розкладає невід'ємну матрицю на добуток двох невід'ємних матриць, що було перспективним інструментом в таких областях, де існують лише невід'ємні сигнали, такі як астрономія. РНМ добре відома завдяки правилу мультиплікативного оновлення, який постійно розроблявся: включення невизначеностей, розгляд відсутніх даних та паралельність обчислень, послідовність побудови, що веде до стабільності та лінійності РНМ, як і інші оновлення.[19]

За допомогою стабільної компонентної бази під час побудови та лінійності процеса моделювання, послідовний РНМ здатний зберігати потік при прямому відтворенні навколозоряних структур в астрономії, як один із способів виявлення екзопланет, особливо при безпосередньому зображені навколозоряних дисків. У порівнянні з МГК, РНМ не видаляє середнє матриць, що призводить до нефізичних невід'ємних потоків, тому РНМ здатний зберігати більше інформації, ніж МГК.

Ядровий метод головних компонент. Метод головних компонент можна використати нелінійним шляхом за допомогою ядрового трюку. Отримана методика здатна побудувати нелінійні відображення, які максимізують дисперсію даних. Отримана методика називається ядровим методом головних компонент.

Лінійний розділювальний аналіз (ЛРА) – це узагальнення лінійного дискримінанта Фішера, який використовується для статистики, розпізнавання образів та машинного навчання, щоб знайти лінійну комбінацію ознак, які характеризують або відокремлюють два або більше класів об'єктів або подій. Автокодувальники можуть використовуватися для навчання нелінійним

функціям зменшення розмірності та кодування разом із оберненою функцією, яка дозволяє перейти від кодуванного до оригінального зображення. [24]

2.3. Марковський алгоритм кластеризації

Марковський алгоритм кластеризації (MCL, Markov Clustering Algorithm) – алкоритм кластерного аналізу заснований на потоці (випадковому блуканні) в графі. Спочатку розроблений для виділення кластерів в простому графі, однак може бути застосований до будь-яких об'єктів для яких задана матриця подібності/ відмінності.

Терміни та визначення:

Граф складається з двох типів об'єктів:

1) вершин (вузлів) – V;

2) ребер (пар вершин з'єднаних між собою) – E. Формально це можна записати як G: = (V, E).

Крім цього, кожен граф можна представити у вигляді матриці суміжності (M) розміром V * V. Де M_{ij} дорівнює відстань між вузлом *i* і вузлом *j*.

Випадковий обхід графа – такий обхід вершин граф, при якому вибір наступної вершини залежить тільки від поточного стану на графі, а ймовірність перейти до будь-якої вершини розраховується виходячи з матриці суміжності. Слід зазначити що при такому обході одна вершина може бути переглянута необмежену кількість разів.[20]

Випадкове блукання в графі являє собою ланцюг Маркова.

Строго визначити що таке кластер в графі не можливо однак можна зробити наступні зауваження:

- ✓ довжина шляху між вузлами одного кластера мала в порівнянні з довжиною шляху між точками, що належать різним кластерам
- ✓ при випадковому обході графа, перш ніж покинути кластер будуть відвідані багато з його вершин.

Метод MCL – відстань між вузлами графа відносяться до одного кластеру, менше ніж відстань між вузлами, що належать до різних кластерів. тобто ймовірність переходу (потік) між вузлами всередині одного кластера

набагато більше, ніж між вузлами, що відносяться до різних кластерів. Таким чином, якщо посилювати потік там де він сильний і послаблювати його там де він слабкий, то згідно парадигмі кластеризації графа границі між різними кластерами зникатимуть. Таким чином буде виявлена кластерна структура графа.

Загальний опис методу. Моделювання потоку в графі здійснюється шляхом перетворення його в марковский граф (Рис.3). [23]

Рис.3. Перетворення графа в марковский граф

На першому кроці граф перетворюється в матрицю відстаней між вузлами (суміжну матрицю). У випадку якщо граф є не зваженим можна вважати, що вага всіх ребер дорівнює одиниці. На другому кроці відбувається перетворення суміжної матриці в матрицю ймовірностей переходів між вузлами (стохастичну матрицю). Для цього як правило нормують значення в кожному окремому стовпці матриці відстаней, однак може бути застосований будь-який інший алгоритм. Після того як стохастичну матриця отримана, до неї почергово застосовують дві функції (поширення і накачування) до тих пір поки матриця змінюється.

1) поширення (N) – являє собою зведення матриці в ступінь N. Дана операція підсилює потік з вершини на потенційних учасників кластера. 2) накачування (K) – являє собою застосування добутку Адамара матриці самої на себе. (добуток Адамара – бінарна операція над двома матрицями однаковою розмірності, результатом якої є матриця тієї ж розмірності, в якій кожен

елемент з індексами i, j – це добуток елементів з індексами i, j вихідних матриць). Дана операція зменшує вірогідність переходів між кластерами швидше, ніж ймовірності переходів всередині одного кластера. Так само на цьому кроці відбувається нормування кожного стовпця матриці.

В даному алгоритмі необхідно підбирати степінь поширення і накачуванні (так як це безпосередньо впливає на розбиття). Збільшення ступеня поширення – збільшує середній розмір кластера, в той час як збільшення ступеня накачування –призводить до збільшення кількості кластерів.

В ході виконання алгоритму виділяються вузли які є "центрами" кластерів. Можна помітити в ході роботи алгоритму граф стає все менш і менш зв'язним, поки не буде розділений на кластери.[25]

Розділ ІІІ. Методи машинного навчання у медицині

3.1. Постановка задачі

Трансплантація органів займає сильну лідерську позицію серед молодих медичних технологій, швидко увійшовши у життя трохи більше 50 років тому. Медичний потенціал трансплантології як науки є надзвичайним, оскільки з використанням трансплантації органів стало можливим поліпшити життя при різних пороках нирок.

Однак трансплантація має деякі особливості, які суттєво обмежують її терапевтичний життєзабезпечувальний потенціал, і це виявляє її парадоксальну природу – при максимальній ефективності цього виду допомоги неможливо досягти її максимальної доступності для всіх, хто її потребує. На відміну від інших медичних спеціальностей, його "вродженою" особливістю є не висока вартість обладнання та лікарських засобів і навіть не міждисциплінарний характер цієї галузі медичної діяльності. Головною особливістю трансплантації є її залежність від найбільш незвичайного, з традиційної точки зору, ресурсу, необхідного для надання цієї допомоги, – ресурсу «здорових» органів донорів.

Сьогодні найпоширеніший варіант сучасної трансплантації нирок – від людини до людини. Трансплантація нирки показана людям, які страждають хронічною нирковою недостатністю. До цього вони «прив’язуються» до апарату штучної нирки і змушені регулярно проходити гемодіаліз.

Трансплантація нирок порівняно з іншими видами лікування (гемодіаліз, перитонеальний діаліз) дає найкращі результати, збільшуючи тривалість життя в середньому в 2 рази та покращуючи її якість. Особливо цей вид лікування важливий для дітей, оскільки завдяки гемодіалізу на розвиток дитини суттєво впливає розвиток дитини.

У наш час з розвитком Data Science стало можливим провести експериментальне дослідження без смерті донорів та реципієнтів. Завдяки розробці алгоритмів машинного навчання лікарі з трансплантації можуть відповісти на питання сумісності донорів та реципієнтів, аналізуючи наявні дані. Ці дані представляють накопичений досвід та спостереження в галузі

трансплантології. Так, наприклад, стало можливим встановити, як той чи інший фактор впливає на тривалість життя донора та реципієнта.

Використання алгоритмів машинного навчання дає змогу передбачити тривалість життя донора та реципієнта після трансплантації. Однак головним досягненням трансплантології є вдосконалення Програми обміну нирок за допомогою кластеризації даних – алгоритму автоматизованого машинного навчання. Використовуючи цей алгоритм, у цій роботі донори та одержувачі будуть об'єднані окремо у групи. Потім, використовуючи матрицю сумісності, донори та реципієнти встановлюватимуть ймовірнісні відносини. За допомогою цього імовірнісного взаємозв'язку були створені групи сумісності донорів та реципієнтів. Крім того, кожна група буде сортована за ймовірністю. Тобто на першому місці буде пара реципієнт-донор, яка має найбільшу ймовірність сумісності та «виживання» після трансплантації. Таке сортування дозволить негайно визначити наступного одержувача для донора у разі смерті першого одержувача. У цій роботі поріг ймовірності не встановлений. Оскільки лікарі з трансплантації самі повинні встановити прийнятний ймовірнісний рівень трансплантації. Ця робота зосереджена на забезпеченні високого рівня точності кластеризації, що дозволить точно вибрати групи донорів-одержувачів. [14]

Робота має на меті вирішити проблему, цілі якої є

1. Математична модель для стохастичних програм обміну нирок за алгоритмом машинного навчання без учителя - класифікація.

2. Отримати найбільш сумісні групи донорів та реципієнтів на основі реальних даних про трансплантацію нирки за допомогою алгоритму Маркова для кластеризації на графах.

3. Сортувати за ймовірністю одержувачів для конкретного донора. Це сортування забезпечить донора для наступного одержувача у смерті першого одержувача.

Основна ідея як типового прикладу KEP наведена на рисунку 3.1.

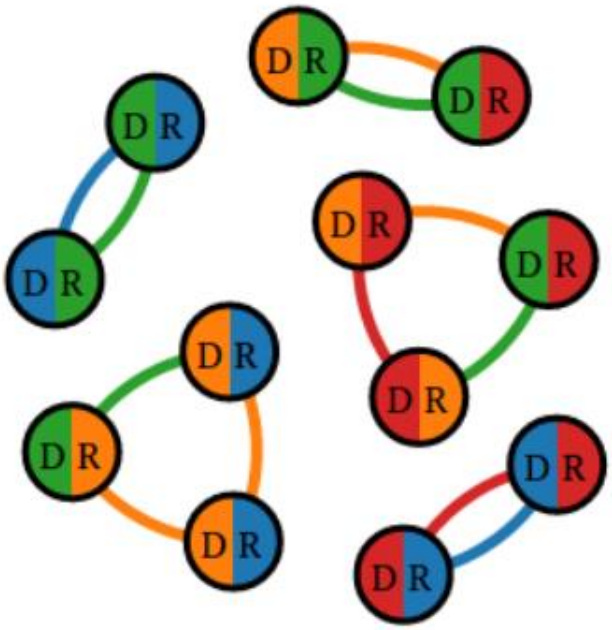


Рис.1. KEP

Після використання машинного навчання (кластеризації) ці групи можуть бути кінцевими за кількістю. Кількість донорів та одержувачів у кластері також є обмеженою та залежить від вхідних даних. Дані, що належать до одного кластеру, за деякими показниками схожі. У той же час дані різних кластерів відрізняються за деякими показниками. [23]

3.2. Історія трансплантології

Емеріх Уллман, угорський хірург, першим провів експерименти з трансплантації нирок у тварин; з 1902 по 1914 рік Алексіс Каррель також проводив експерименти з трансплантації нирок, збереження та методики застосування. Як результат, саме він розробив основні принципи збереження донорського органу. Алексіс Каррел був удостоєний Нобелівської премії в 1912 році за роботу з пересадки органів.

Першу спробу пересадити орган з тварини людині зробив Матьє Джабуле. Він пересадив нирку свині пацієнту з нефротичним синдромом, але операція, на жаль, призвела до смерті пацієнта. У перші роки XX століття були здійснені інші спроби пересадити органи від тварин, переважно свиней та мавп, людям, але всі вони були безуспішними.

Український хірург Юрій Юрійович Вороний 3 квітня 1933 року в Харкові вперше у світі здійснив спробу трансплантації нирки від людини до людини. Під час операції він пересадив нирку з трупа 60-річного чоловіка молодій дівчині 26 років.

Нирку трансплантували в стегно як тимчасовий захід, але операція не врятувала життя пацієнта. Справа в тому, що на момент операції вже минуло 6 годин з моменту смерті донора, а Вороной не мав даних про нежиттєздатність нирки після тривалої ішемії. Це призвело до природно невдалого результату операції, пацієнт помер.

Вороной не розглядав ідею взяти орган у живої людини, вважаючи, що "неможливо заподіяти відому інвалідність здоровій людині, вирізавши орган, необхідний для трансплантації для проблемного порятунку пацієнта".

До Вороного ніхто не намагався пересадити цілий орган з трупа за допомогою судинного шва, український хірург став першовідкривачем у трансплантації трупних органів. Перші документальні докази трансплантації трупної нирки Вороной опублікував в 1934 році в італійському журналі «Vinerva Chirurgisa», де було зазначено, що нирка була включена в кровообіг і почала функціонувати самостійно.

У 1943 році голландському вченому У. Колфу вдалося реалізувати ідею про можливість виведення з крові токсинів, що накопичуються при нирковій недостатності шляхом гемодіалізу. У. Колф назвав свій пристрій «штучною ниркою». 17 березня 1943 р. Цей пристрій вперше було використано для лікування хворого на уремію [10].

Винахід В. Кольфа зіграв величезну роль у розвитку проблеми трансплантації нирки, оскільки хронічний гемодіаліз дозволив продовжити життя пацієнтів з нирковою недостатністю та уремією, що дозволило багатьом з них чекати на трансплантацію донорського органу .

У 1950 році в Чикаго лікарі пересадили 44-річній жінці нирку від трупа тієї ж групи крові. Операція пройшла успішно, оскільки трансплантація «працювала» протягом 53 днів.

Подібні операції в Європі проводили багато вчених, але вони ускладнювалися або неправильними умовами зберігання органу, або ускладненнями під час операції.

До 1950 року сучасний апарат штучної нирки був готовий до використання у США, а методи переливання крові та терапії антибіотиками, які були широко поширені в цей період, зробили складніші хірургічні операції більш безпечними. 31 березня 1951 року в Бостоні (США) хірург Скол вперше переніс 37-річному пацієнту з хронічним гломерулонефритом на гемодіаліз на алогенну нирку, видалену у хворого на рак нижньої третини сечоводу. Обидві операції проводилися одночасно в двох операційних, тривалість ішемії трансплантації нирок становила 70 хвилин. На жаль, ця операція закінчилася відторгненням трансплантата, і реципієнт помер через 5 тижнів від прогресування ниркової недостатності, незважаючи на гемодіаліз. Тим не менш, шлях до клінічної трансплантації нирки був відкритим, і в найближчі роки до США.

П'ятнадцять алогенних трансплантацій нирок було проведено пацієнтам на хронічному гемодіалізі. Цей початковий досвід дав важливі докази того, що для досягнення прийнятної якості життя у реципієнта необхідно видалити власну нирку, уражену гломерулонефритом. В іншому випадку ризик виникнення важких ускладнень реноваскулярної гіпертензії залишається високим, а артерії пересадженої нирки в цих умовах швидко переживають склероз.

Першою успішною трансплантацією нирки стала так звана «сімейна трансплантація», яку виконував Джозеф Мюррей та інші хірурги під наглядом лікаря загальної практики Джона Мерріла. 26 жовтня 1954 року чоловіка Річарда Херріка госпіталізували з недостатністю нирок, але у нього був брат-близнюк Рональд. Після кількох підготовчих операцій 23 грудня того ж року було проведено пересадку нирки. В результаті Річард прожив ще 9 років, але помер від рецидиву захворювання, а Рональд помер лише в 2010 році. Трансплантація нирок між братами-близнюками вважалася успішною.

У 1959 році була проведена перша трансплантація нирки від посмертного неспорідненого донора, але потім загальне опромінення організму було використано для придушення імунітету. Пізніше вчені розпочали роботу над створенням препарату, який дозволив би пересадити незалежних донорів. До 1980 року був створений перший подібний препарат, який розпочав нову еру в трансплантації.

Висновки

Основна увага у магістерській роботі приділяється методу кластеризації. Кластеризація значно скоротила час з використанням алгоритмів кластеризації Маркова на графах. Вдалося перейти від класичної постановки проблеми (один донор відповідає одному реципієнту) до ймовірнісної проблеми, в якій декілька донорів та реципієнтів об’єднані в одну групу (кластер).

Таким чином, з точки зору кластерного аналізу, в роботі ми змогли перейти від жорсткого кластеризації до м'якого кластеризації. Цей перехід нам вдалося здійснити, спростивши вимоги до розв’язку $x = (x_1, ..., x_n)$.

Таким чином, було виявлено перехід від детермінованого до ймовірнісного випадку вирішення проблеми найкращого поєднання між донорами та реципієнтами, що розширює набір завдань, де можна використовувати цей підхід. Друга перевага цього методу полягає в тому, що цей метод зменшує обчислювальну складність алгоритму і дозволяє уникнути перерахунку.

Список літератури

1. Воронцов К.В. Математические методы обучения по прецендентам (теория обучения машин). Електронный курс лекций. – М:МФТИ. – 2011. –141 с.
2. Вьюгин В.В. Математические основы машинного обучения и прогнозирования. – М: МЦНМО, 2013 – 304 с.
3. Гихман И.И., Скороход А.В., Ядренко М.И. Теория вероятности и математическая статистика.–К: Вища шк., 1979. – 320 с.
4. Дюран Б. Кластерный анализ. / Дюран Б., Оделл П. – К.: Статистика, 1999. – 128 с.
5. Жамбю М. Иерархический кластер-анализ и соответствия. / Жамбю М. – К.: Финансы и статистика, 1988. – 345 с.
6. Мандель И. Д. Кластерный анализ. / Мандель И. Д. – К.: Финансы и статистика, 1988. – 10 с.
7. Мерков А. Б. Распознавание образов. Введение в методы статистического обучения. 2011. – 256 с.
8. Мерков А. Б. Распознавание образов. Построение и обучение вероятностных моделей. 2014. – 238 с.
9. Alpaydin E., Introduction to machine learning, 2nd ed., The MIT Press, 2010.
10. Baytinger V.F. Istoriya khirurgii v litsakh [Surgery History People]. Tomsk: Krasnoe znamya Publ., 2007. – 248 p.
11. Bishop C. M. Pattern Recognition and Machine Learning. – Springer, 2006. – 738 p.
12. Claude Sammut, Geoffrey I. Webb (Eds.) Encyclopedia of Machine Learning, 2011.
13. Hastic Tr., Tibshirani R., Friedman J. The elements of statistical learning.- Springer series of statistics: Springer,2003. – 552 p.

14. Malaver Perez E. R., Rueda Guerrero W. J., Niño Rocha H. C. and Becerra Ospina V. E., "Modelos de optimización del programa de intercambio de riñones con múltiples etapas," Escuela Colombiana de ingeniería Julioa Garavito, Bogota, 2018.
15. Ng A., Jordan M., and Weiss Y., “On spectral clustering: analysis and an algorithm,” Advances in Neural Information Processing Systems, vol. 14, 2002.
16. Schmidhuber J., “Deep Learning in Neural Networks: An Overview” http://arxiv.org/abs/1404.7828, 2014
17. Song, Hyun Ah, and Soo-Young Lee. “Hierarchical Representation Using NMF.” Neural Information Processing. Springer Berlin Heidelberg, 2013.
18. Xu R., Wunsch D.I.I., “Survey of clustering algorithms,” IEEE Trans.Neural Networks, vol. 16, no. 3, pp. 645–678, 2005.
19. Journal of Educational Technology & Online Learning, “A Research on Machine Learning Methods and Its Applications” Özer ÇELİK (Corresponding author), Serthan Salih ALTUNAYDIN
20. CS 595D Presentation By Kathy Macropol “Clustering on Graphs: The Markov Cluster Algorithm (MCL)”
21. https://www.doc.ic.ac.uk/~jce317/history-machine-learning.html
22. https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#4f54d0f315e7
23. http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%
24. https://www.researchgate.net/publication/321686595_Vvedenie_v_masinnoe_obucenie
25. http://www.uic.unn.ru/~zny/ml/Course/ml_pres.pdf

Додатки

```
library(readr)

# зчитуємо дані з LIVING_DONOR_DATA
LIVING_DONOR_DATA <- read_delim("LIVING_DONOR_DATA.DAT", "\t", escape_double =
FALSE, col_names = TRUE,trim_ws = TRUE)

# Подивимося на типи даних. Як ми бачимо є багато пропущених даних, які названі
як"."
str(LIVING_DONOR_DATA)

# Маємо 179 стовпців і 161066 рядочків
dim((LIVING_DONOR_DATA))

## [1] 161066    179

# Зараз подивимося дані

view(LIVING_DONOR_DATA)


# Встановимо імена з STAR файла Documentation
colnames(LIVING_DONOR_DATA) <- c("Second", "First", "ABO","AGE_DON",
"ANESTHETIC_COMP",
"ARRHYTHMIA",
"ARRHYTHMIA_POSTOP",
"BILIARY_COMP",
"BILIARY_COMP_GRADE",
"BIOPSY_LI",
"BP_POSTOP_DIAST",
"BP_POSTOP_SYST",
"BP_PREOP_DIAST",
"BP_PREOP_SYST",
"CANCER_FREE",
"CITIZEN_COUNTRY",
"CITIZENSHIP",
"CMV_IGG",
"CMV_IGM",
"CMV_NUCLEIC",
"CMV_TOTAL",
"COD",
"COD_OSTXT",
"CONVERT_OPEN_KI",
"CONVERT_OPEN_LU",
"DEATH_DT",
"DIABETES",
"DON_DATE",
"DON_ORG",
"DON_ORG2",
"DONOR_ID",
"DUR_ABSTINENCE",
"EBV_IGG",
"EBV_IGM",
"EBV_TOTAL",
"EDUCATION",
"ETHCAT_DON",
```

```
"FFP_UNITS",
"FUNC_STAT",
"GENDER",
"HBV_CORE",
"HBV_DNA",
"HBV_SUR_ANTIGEN",
"HCV_ANTIBODY",
"HCV_RIBA",
"HCV_RNA",
"HEALTH_INS",
"HIST_CANCER",
"HIST_CANCER_OSTXT",
"HIST_CIG",
"HIST_HYPER",
"HOME_STATE",
"HYPER_DIET",
"HYPER_DIUR",
"HYPER_MEDS",
"HYPERTENSION",
"INIT_DISCHARGE_DT",
"INTRAOP_COMP",
"INTRAOP_COMP_OSTXT",
"INTRAOP_COMP_REASON",
"KI_CREAT_POSTOP",
"KI_CREAT_PREOP",
"KI_PROC_TY",
"KIDNEY_RECOV",
"LI_PROC_TY",
"LIV_DON_TY",
"LIV_DON_TY_OSTXT",
"LIVER_RECOV",
"LU_COMP",
"LU_COMP_OSTXT",
"LU_COMP_REASON",
"LU_PROC_TY",
"LUNG_RECOV",
"MACRO_FAT",
"MARITAL_STAT",
"MICRO_FAT",
"NON_AUTO_BLOOD",
"ORG_RECOVERY_DT",
"OTH_COMP_KI",
"OTH_COMP_KI_INTER",
"OTH_COMP_KI_INTER_OSTXT",
"OTH_COMP_LI",
"OTH_COMP_LI_INTER",
"OTH_COMP_LI_INTER_OSTXT",
"OTH_INTER_PROC_KI",
"OTH_INTER_PROC_KI_DT",
"OTH_INTER_PROC_KI_OSTXT",
"OTH_INTER_PROC_LI",
"OTH_INTER_PROC_LI_DT",
"OTH_INTER_PROC_LI_OSTXT",
"PACK_YRS",
"PHYSICAL_CAPACITY",
"PLATELETS_UNITS",
```

```
"POSTOP_ALBUM",
"POSTOP_ALK_PHOS",
"POSTOP_BILI",
"POSTOP_CREAT_LI",
"POSTOP_INR",
"POSTOP_SGOT_AST",
"POSTOP_SGPT_ALT",
"POSTOP_TEST_DT",
"POSTOP_URINE_PROTEIN",
"POSTOP_URINE_RATIO",
"PRBC_UNITS",
"PREDON_HGT",
"PREDON_WGT",
"PREOP_ALBUM",
"PREOP_ALK_PHOS",
"PREOP_BILI",
"PREOP_CREAT_LI",
"PREOP_FEF_AFTER",
"PREOP_FEF_BEFORE",
"PREOP_FEV1_AFTER",
"PREOP_FEV1_BEFORE",
"PREOP_FVC_AFTER",
"PREOP_FVC_BEFORE",
"PREOP_INR",
"PREOP_LUNG_CAP",
"PREOP_PAO2",
"PREOP_SGOT_AST",
"PREOP_SGPT_ALT",
"PREOP_TLC_AFTER",
"PREOP_TLC_BEFORE",
"PREOP_URINE_PROTEIN",
"PREOP_URINE_RATIO",
"PX_STAT",
"READMISSION_KI",
"READMISSION_KI_DT",
"READMISSION_KI_OSTXT",
"READMISSION_KI_REASON",
"READMISSION_LI",
"READMISSION_LI_DT",
"READMISSION_LI_OSTXT",
"READMISSION_LI_REASON",
"READMISSION_LU",
"READMISSION_LU_DT",
"READMISSION_LU_OSTXT",
"READMISSION_LU_REASON",
"RECOV_FACILITY_CODE",
"REGION",
"REOP_BILIARY",
"REOP_BILIARY_DT",
"REOP_BLEED_KI",
"REOP_BLEED_KI_DT",
"REOP_BLEED_LI",
"REOP_BLEED_LI_DT",
"REOP_BOWEL_KI",
"REOP_BOWEL_KI_DT",
"REOP_BOWEL_LI",
```

```
"REOP_BOWEL_LI_DT",
"REOP_HERNIA_KI",
"REOP_HERNIA_KI_DT",
"REOP_HERNIA_LI",
"REOP_HERNIA_LI_DT",
"REOP_LI_FAIL",
"REOP_LI_FAIL_DT",
"REOP_OTH_KI",
"REOP_OTH_KI_DT",
"REOP_OTH_KI_OSTXT",
"REOP_OTH_LI",
"REOP_OTH_LI_DT",
"REOP_OTH_LI_OSTXT",
"REOP_VASC_KI",
"REOP_VASC_KI_DT",
"REOP_VASC_LI",
"REOP_VASC_LI_DT",
"REOPERATION_KI",
"REOPERATION_LI",
"SACRIFICE_LOBE",
"THORAC_TUBES",
"TOBACCO_USE",
"VASC_COMP_KI",
"VASC_COMP_KI_INTER",
"VASC_COMP_KI_INTER_OSTXT",
"VASC_COMP_LI",
"VASC_COMP_LI_INTER",
"VASC_COMP_LI_INTER_OSTXT",
"VIRUSES_TESTED",
"WGT_KG")

# Замініть всі пропущені значення на NA
LIVING_DONOR_DATA[ LIVING_DONOR_DATA == "." ] <- NA

# створити фрейм даних із значущими змінними
df <- data.frame(LIVING_DONOR_DATA$AGE_DON, LIVING_DONOR_DATA$ARRHYTHMIA,
                 LIVING_DONOR_DATA$CANCER_FREE,
LIVING_DONOR_DATA$BP_POSTOP_DIAST,
                 LIVING_DONOR_DATA$BP_POSTOP_SYST,
LIVING_DONOR_DATA$HIST_CANCER,
                 LIVING_DONOR_DATA$HIST_CIG, LIVING_DONOR_DATA$HIST_HYPER,
                 LIVING_DONOR_DATA$KI_CREAT_POSTOP,
LIVING_DONOR_DATA$KI_CREAT_PREOP,
                 LIVING_DONOR_DATA$CONVERT_OPEN_KI)
names(df) <- c("age", "arythmia", "cancer", "postoppresd", "postoppress",
"histcan", "histcig", "histhiper", "creatpost", "creatpre", "dead")


# визначити тільки не NA об'єкти
df1 = na.omit(df)
df1$postoppress = as.character(df1$postoppress)
df1 = df1[df1$postoppress != '999',]
df1$postoppress = as.factor(df1$postoppress)

# змінна типів змінних
```

```r
df1$cancer = as.numeric(as.character(df1$cancer))
df1$creatpost = as.numeric(as.character(df1$creatpost))
df1$creatpre = as.numeric(as.character(df1$creatpre))
df1$postoppresd = as.numeric(as.character(df1$postoppresd))
df1$postoppress = as.numeric(as.character(df1$postoppress))

str(df1)
```

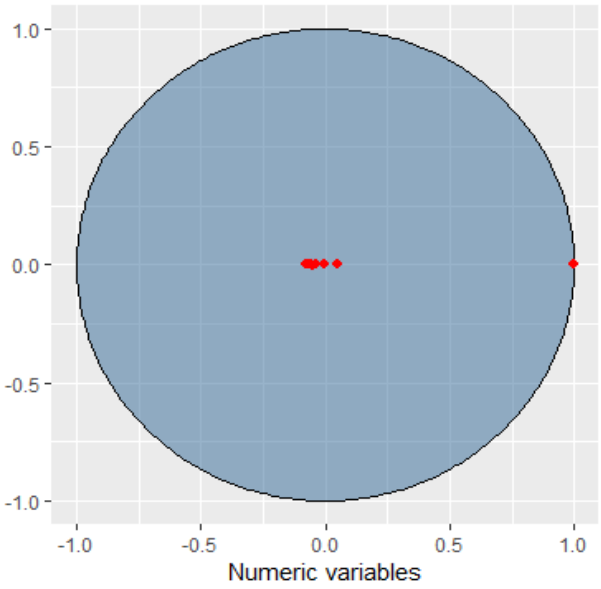
```
## 'data.frame':    58579 obs. of  11 variables:
##  $ age        : num  19 46 25 28 49 37 57 43 56 40 ...
##  $ arythmia   : num  5 7 5 3 6 5 2 9 7 5 ...
##  $ cancer     : num  0.7 0.66 0.91 0.96 0.97 0.76 0.66 0.7 1 1 ...
##  $ postoppresd: num  1 1 1 1 1 4 1 1 1 1 ...
##  $ postoppress: num  4 7 7 2 10 1 9 10 11 2 ...
##  $ histcan    : Factor w/ 3 levels "N","U","Y": 1 1 1 1 1 1 1 1 1 2 ...
##  $ histcig    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ histhiper  : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ creatpost  : num  110 116 119 129 123 123 116 115 119 132 ...
##  $ creatpre   : num  70 69 68 76 78 58 78 57 80 87 ...
##  $ dead       : Factor w/ 3 levels "N","U","Y": 3 3 3 3 3 1 3 3 3 1 ...
##  - attr(*, "na.action")= 'omit' Named int  1 2 3 4 5 6 7 8 9 10 ...
##   ..- attr(*, "names")= chr  "1" "2" "3" "4" ...
```

```r
# видалити всі невідомі спостереження (UNKNOWN histcan, dead)
df1 = df1[df1$dead != unique(df1$dead)[3],]
df1 = df1[df1$histcan != unique(df1$histcan)[2],]

# Відокремити дані на три типи:
  # 1 - numeric (age, cancer, creatpost, creatpre)
  # 2 - binary (histcan, histcig, histhiper, dead)
  # 3 - order (arythmia, postoppresd, postoppress)

df_number = df1[, c(1,3,9,10)]
df_binary = df1[, c(6:8,11)]
df_order  = df1[, c(2,4,5)]

# Нормалізація числових змінних
for (j in 1:dim(df_number)[2]){
  df_number[, j] = (df_number[, j] - min(df_number[, j]))/
    (max(df_number[, j]) - min(df_number[, j]))
}

# Візьмемо випадкові індекси з усієї вибірки для кластеризації

IND = sample(1:dim(df1)[1], 20)

# використовуємо дані лише з цими індексами
df_number = df_number[IND,]
df_binary = df_binary[IND,]
df_order  = df_order[IND,]


# знаходимо матрицю схожості для числових даних
S_number = as.matrix(dist(df_number))
S_number = 1- S_number
diag(S_number) = 0
```

```r
for (i in 1:dim(S_number)[1]){
  S_number[i,] = S_number[i,]/sum(S_number[i,])
}

# знаходимо матрицю схожості для двійкових даних
N = dim(S_number)[1]
S_binary = matrix(NA, nrow = N, ncol = N)

for (i in 1:N){
  for (j in i:N){
    S_binary[i,j] = mean(df_binary[i,] == df_binary[j,])
    S_binary[j,i] = S_binary[i,j]
  }
  # print(i)
}
diag(S_binary) = 0

for (i in 1:N){
  S_binary[i,] = S_binary[i,]/sum(S_binary[i,])
}

# знаходимо матрицю схожості для впорядкованих даних

SIM_ord = function(x,y,alpha = 0.5){
  k = length(x)
  res = sum(alpha^abs(x-y))/k
  return(res)
}


S_order = matrix(NA, nrow = N, ncol = N)
for (i in 1:N){
  for (j in i:N){
    S_order[i,j] = SIM_ord(df_order[i,], df_order[j,], alpha = 0.02)
    S_order[j,i] = S_order[i,j]
  }
  print(i)
}
```

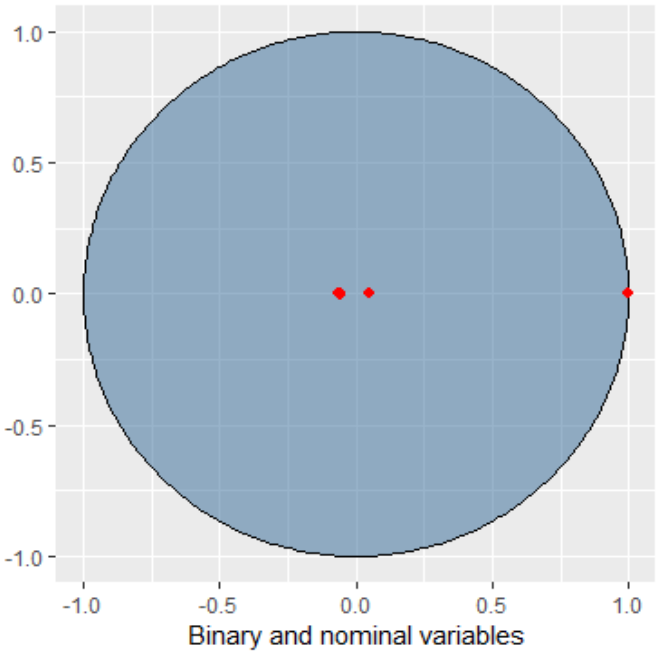
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
```

```
## [1] 17
## [1] 18
## [1] 19
## [1] 20

diag(S_order) = 0

for (i in 1:N){
  S_order[i,] = S_order[i,]/sum(S_order[i,])
}

# знаходимо загальну матрицю ймовірностей
w = c(0.1, 0.4, 0.5)

P = w[1]*S_number + w[2]*S_binary + w[3]*S_order

# обчислюємо кількість кластерів
EIG_r1 = eigen(S_number)$values
EIG_r2 = eigen(S_binary)$values
EIG_r3 = eigen(S_order)$values
EIG_r4 = eigen(P)$values

library(ggplot2)
library(ggforce)

p1 = ggplot() +
  geom_circle(aes(x0 = 0, y0 = 0, r = 1, fill = 1), alpha = 0.5)+
  geom_point(aes(x = Re(EIG_r1), y = Im(EIG_r4)), col = 2, size = 1.8)+
  coord_fixed()+
  xlab('Numeric variables')+ ylab(' ')+
  theme(legend.position = "none")
p1
```

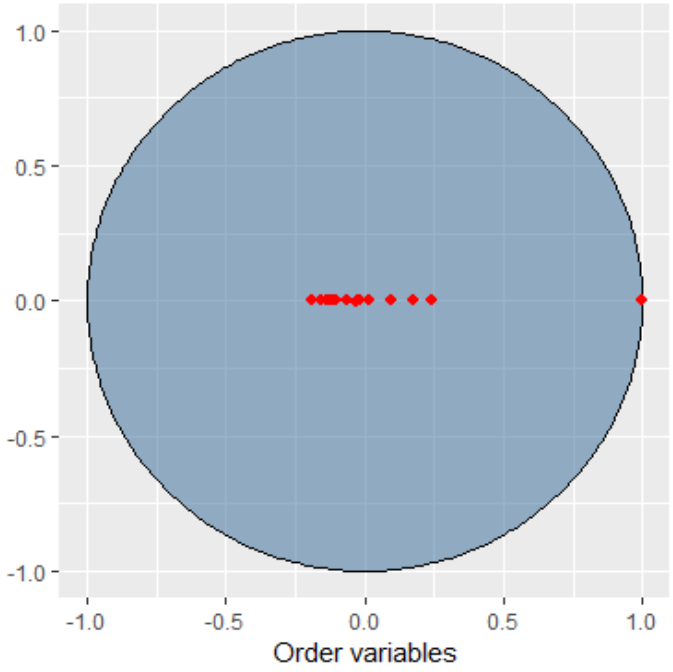



```
p2 = ggplot() +
  geom_circle(aes(x0 = 0, y0 = 0, r = 1, fill = 1), alpha = 0.5)+
  geom_point(aes(x = Re(EIG_r2), y = Im(EIG_r4)), col = 2, size = 1.8)+
  coord_fixed()+
  xlab('Binary and nominal variables')+ ylab(' ')+
```

```
  theme(legend.position = "none")
p2
```

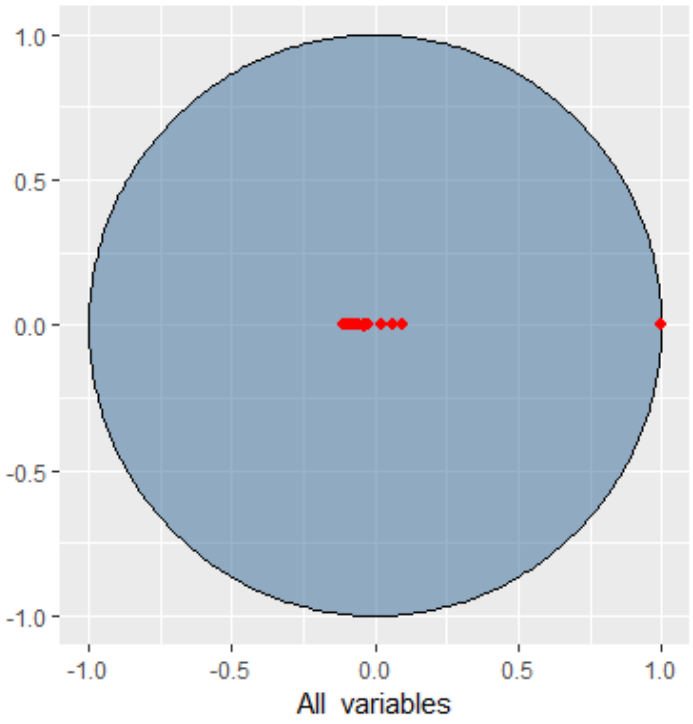



```
p3 = ggplot() +
  geom_circle(aes(x0 = 0, y0 = 0, r = 1, fill = 1), alpha = 0.5)+
  geom_point(aes(x = Re(EIG_r3), y = Im(EIG_r4)), col = 2, size = 1.8)+
  coord_fixed()+
  xlab('Order variables')+ ylab(' ')+
  theme(legend.position = "none")
p3
```




```
p4 = ggplot() +
  geom_circle(aes(x0 = 0, y0 = 0, r = 1, fill = 1), alpha = 0.5)+
  geom_point(aes(x = Re(EIG_r4), y = Im(EIG_r4)), col = 2, size = 1.8)+
  coord_fixed()+
```

```
  xlab('All  variables')+ ylab(' ')+
  theme(legend.position = "none")
p4
```




```
# видалення дуже малих ймовірностей
P1 = P
for (i in 1:N){
  P1[i, P1[i,] < 1.2/N ] = 0
  P1[i,] = P1[i,]/sum(P1[i,])
}

P = P1

# Марковський алгоритм
r = 2; s = 2 # параметри алгоритму

eps = 10^(-14)
P_old = P
P_new = P-1
k = 1

while (max(abs(P_old - P_new))>eps){
+   P_old = P_new
+   P_new = P_old%^%r
+   P_new = P_new^s
+   for (i in 1:N){
+     P_new[i,] = P_new[i,]/sum(P_new[i,])
+   }
+   k = k+1
+   print(c(k, max(abs(P_old - P_new))))
+ }
SS = eigen(P_new)
SS
```

```
eigen() decomposition
$values
 [1] -19.00000000   1.00000000   0.96106104   0.83285050
 [5]   0.65584668   0.52619947   0.46495899   0.35817874
 [9]  -0.32870871  -0.29976746  -0.29093663  -0.26244254
[13]  -0.24561361  -0.23554574  -0.17155309   0.06213914
[17]   0.03725789  -0.02441309  -0.02203702   0.00318553

$vectors
            [,1]        [,2]        [,3]        [,4]
 [1,] -0.2236068  0.06255061 -0.31345572 -0.04896052
 [2,] -0.2236068  0.06255061 -0.35544767 -0.07449733
 [3,] -0.2236068  0.06255061  0.07712873  0.10051718
 [4,] -0.2236068  0.06255061  0.22280313 -0.33086025
 [5,] -0.2236068  0.06255061  0.16547353 -0.02112591
 [6,] -0.2236068  0.06255061  0.22284909 -0.33113513
 [7,] -0.2236068  0.06255061  0.23633888 -0.41549733
 [8,] -0.2236068 -0.94749148 -0.02849922 -0.02971947
 [9,] -0.2236068  0.06255061 -0.07937504  0.25231829
[10,] -0.2236068  0.06255061  0.13997045  0.12579045
[11,] -0.2236068  0.06255061  0.14438993  0.17057954
[12,] -0.2236068  0.06255061  0.16637177  0.35739787
[13,] -0.2236068  0.06255061 -0.33026308 -0.05970422
[14,] -0.2236068  0.06255061 -0.35539107 -0.07445912
[15,] -0.2236068  0.06255061 -0.33080559 -0.06000243
[16,] -0.2236068  0.06255061  0.16587730  0.35256247
[17,] -0.2236068  0.06255061 -0.23465594 -0.01405380
[18,] -0.2236068  0.06255061  0.22274304 -0.33049674
[19,] -0.2236068  0.06255061  0.13969339  0.12598287
[20,] -0.2236068 -0.17841943  0.12314433  0.30039598
             [,5]        [,6]         [,7]       [,8]
 [1,]  0.10601845 -0.05254286  0.250927494 -0.0529982
 [2,]  0.06359760 -0.05254286 -0.134019727 -0.0529982
 [3,]  0.19713045 -0.05254286  0.065673030 -0.0529982
 [4,]  0.05269712 -0.05254286  0.047391640 -0.0529982
 [5,]  0.15829190 -0.05254286  0.021316285 -0.0529982
 [6,]  0.05258745 -0.05254286  0.047424521 -0.0529982
 [7,]  0.01583440 -0.05254286  0.060382750 -0.0529982
 [8,]  0.09743926 -0.05254286  0.041884145 -0.0529982
 [9,] -0.65857604  0.97341959 -0.899857776 -0.0529982
[10,]  0.21453661 -0.05254286  0.004795746 -0.0529982
[11,]  0.09190216 -0.05254286  0.024784180 -0.0529982
[12,] -0.32786804 -0.05254286  0.075077298 -0.0529982
[13,]  0.08657593 -0.05254286  0.051015370 -0.0529982
[14,]  0.06367236 -0.05254286 -0.133183718 -0.0529982
[15,]  0.08617804 -0.05254286  0.048856296 -0.0529982
[16,] -0.31415018 -0.05254286  0.072784443 -0.0529982
[17,]  0.13416095 -0.05254286  0.214305878 -0.0529982
[18,]  0.05284575 -0.05254286  0.047344729 -0.0529982
[19,]  0.21463004 -0.05254286  0.005174946 -0.0529982
[20,] -0.35397017 -0.05254286  0.110332205  0.9729505
              [,9]         [,10]         [,11]
 [1,]  0.073024974  0.0207592321  0.0019219974
 [2,]  0.191646709  0.0322129809  0.0024092438
 [3,] -0.599483966 -0.0525153979 -0.0016107926
 [4,] -0.081253065  0.1925667645  0.6976135115
```

```
 [5,]  0.239159771 -0.5808174639 -0.3575926912
 [6,] -0.077112062  0.2057963748  0.1694329054
 [7,]  0.123752989 -0.3357995604 -0.2152670115
 [8,]  0.004974726  0.0007688172  0.0006297787
 [9,]  0.006395896  0.0420586608  0.0304404616
[10,]  0.036050836  0.0332129579  0.0095088396
[11,]  0.245494550  0.5269060527  0.3327481222
[12,] -0.069183276 -0.1644563621 -0.1042187344
[13,] -0.305687765 -0.0494370914 -0.0021689091
[14,]  0.195287320  0.0328734771  0.0024475000
[15,] -0.286630405 -0.0469500133 -0.0020523004
[16,] -0.091509201 -0.2185497394 -0.1395148648
[17,]  0.462970136  0.0574313826  0.0034760214
[18,] -0.091666558  0.3150861463 -0.4094020420
[19,]  0.044849942  0.0306927059  0.0074356812
[20,] -0.014471590 -0.0408406408 -0.0254237123
             [,12]         [,13]        [,14]       [,15]
 [1,] -0.097109439 -6.464481e-02 -0.333871835 -0.35022135
 [2,] -0.092155088 -4.586721e-02 -0.230870908 -0.14232170
 [3,]  0.005466502 -3.701120e-02 -0.257483255 -0.41112081
 [4,]  0.082681321 -8.702518e-03 -0.085900069  0.11419404
 [5,] -0.385260094  2.890508e-02  0.246947838 -0.19747128
 [6,]  0.415186018 -1.174890e-02 -0.103097380  0.11907994
 [7,] -0.264194618  2.291400e-02  0.205705246 -0.25174402
 [8,]  0.009094696  3.152991e-03  0.023836357 -0.01344216
 [9,]  0.099980805  2.835915e-02  0.176155110  0.05956555
[10,] -0.059014690 -6.745467e-03 -0.093837716  0.33473842
[11,]  0.623403964  4.632584e-02  0.426034191 -0.39639947
[12,] -0.187886446 -1.079930e-02 -0.106667157  0.11148323
[13,]  0.155153983 -6.356686e-01  0.465003696  0.17212681
[14,] -0.094462771 -3.916928e-02 -0.238295017 -0.14711751
[15,]  0.163566554  7.626747e-01  0.315165266  0.13993186
[16,] -0.261064670 -1.633333e-02 -0.160663597  0.18444994
[17,] -0.084778671  2.178135e-03 -0.045497551  0.19093040
[18,]  0.082336875 -7.916680e-03 -0.079942379  0.11156265
[19,] -0.061859190 -5.921238e-03 -0.087696837  0.34532635
[20,] -0.037603513 -5.397336e-05 -0.005573094  0.01070090
             [,16]         [,17]         [,18]
 [1,] -0.635819742  0.1420197874  4.276637e-03
 [2,]  0.031649168 -0.0135094439  7.099649e-01
 [3,]  0.329247580 -0.0619159344  4.771433e-04
 [4,] -0.080041593 -0.2208005724 -2.126168e-04
 [5,] -0.148688764 -0.3090397745 -2.155828e-04
 [6,] -0.080098544 -0.2213881766 -2.125862e-04
 [7,]  0.266735746  0.8027182290 -2.281651e-04
 [8,] -0.014982658 -0.0073023798 -2.166000e-04
 [9,]  0.292232569 -0.0610417412 -2.163297e-03
[10,] -0.138628274  0.1765586985  7.926214e-05
[11,] -0.042523359  0.0336879011 -2.157032e-04
[12,]  0.008491436 -0.0470127107 -2.161845e-04
[13,] -0.046973501 -0.0036776965 -6.157772e-03
[14,]  0.029679535 -0.0131691259 -7.041921e-01
[15,] -0.058970151 -0.0003048502  4.342219e-04
[16,] -0.005491933 -0.0149725872 -2.178980e-04
[17,]  0.496037443 -0.1372194699  4.267685e-04
[18,] -0.080473147 -0.2217319833 -2.126197e-04
```

```
[19,] -0.118859549  0.1773259124 -1.203099e-03
[20,] -0.016573911 -0.0062543902 -2.165909e-04
              [,19]        [,20]
 [1,] -0.0184685606 -0.024905415
 [2,] -0.0019766685  0.003995782
 [3,] -0.0025908948  0.006778234
 [4,] -0.0028750964  0.025548665
 [5,] -0.0009952037  0.019507898
 [6,] -0.0028949520  0.025662862
 [7,]  0.0075636111 -0.064866036
 [8,] -0.0002189039  0.004245285
 [9,]  0.0078756503  0.353714344
[10,]  0.7068643206 -0.187477160
[11,] -0.0008018551  0.335660830
[12,] -0.0005785519 -0.743085319
[13,]  0.0009133941  0.004857393
[14,] -0.0001263601  0.004065613
[15,]  0.0003134550  0.004046189
[16,]  0.0007120456  0.366755316
[17,]  0.0169551315  0.031033527
[18,] -0.0028737093  0.025581343
[19,] -0.7067927457 -0.179023802
[20,] -0.0002278341 -0.007863786

sum(SS$values==1)
[1] 0
library(MCL)
mcl(P, addLoops=F, ESM = T)

$n.iterations
[1] 20

$Cluster
 [1] 4 1 5 2 2 2 2 0 4 5 5 4 1 1 1 5 4 2 5 5

$Equilibrium.state.matrix
   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
1  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
2  0 1 0 0 0 0 0 0 0  0  0  0  1  1  1  0  0  0  0  0
3  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
4  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
5  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
6  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
7  0 0 0 1 1 1 1 0 0  0  0  0  0  0  0  0  0  1  0  0
8  0 0 0 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  0  0
9  1 0 0 0 0 0 0 0 1  0  0  1  0  0  0  0  1  0  0  0
10 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
11 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
12 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
13 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
14 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
15 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
16 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
17 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
18 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
```

```
19 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0
20 0 0 1 0 0 0 0 0 0  1  1  0  0  0  0  1  0  0  1  1
```