УДК 004.032.26

DOI 10.31651/2076-5886-2021-1-91-112

PACS 07.05.Mh

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент,
доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

ПІСКУН Олександр Варфоломійович,
кандидат технічних наук, доцент,
завідувач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

ДІХТЯРЕНКО Володимир Анатолійович,
Драбівський НВК «Загальноосвітня школа І-ІІІ ст. ім. С.В. Васильченка-гімназія» Драбівської районної ради

БІЛЕЦЬКА Надія Андріївна,
вчитель математики вищої кваліфікаційної категорії, педагогічне звання «вчитель-методист» Драбівського НВК «загальноосвітня школа І-ІІІ ст. ім. С.В. Васильченка-гімназія» Драбівської районної ради

ПОПЕРЕДНЄ ВИЗНАЧЕННЯ КІЛЬКОСТІ КЛАСТЕРІВ ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ТИПУ SOFM

У статті описано авторський алгоритм апріорного визначення кількості кластерів за допомогою самоорганізованих карт ознак Кохонена. У першій частині статті наведено поняття кластеризації та описано використовуваний алгоритм кластеризації k-середніх. У другій частині наведено теоретичні відомості та алгоритм роботи штучної нейронної мережі Кохонена; на прикладах продемонстровано роботу нейронної мережі та проаналізовано отримані результати. У третій частині наведено розроблений алгоритм початкового автоматичного визначення кластерів у системі для алгоритму k-means, отриманого за допомогою SOFM; проаналізовано отримані результати та сформовано подальші напрямки дослідження обраної теми.

Ключові слова: *машинне навчання, кластеризація, k-means, штучні нейронні мережі, SOFM.*

Вступ

Протягом останніх кількох років спостерігається стрімкий розвиток напрямку штучного інтелекту, що називається “машинне навчання”. У основі цього напрямку

лежать методи автономної роботи обчислювальних систем, які дозволяють розв'язувати різноманітні задачі: комп'ютерного бачення, обробки текстів, прийняття рішень тощо. У цьому сенсі важливою підзадачею є скорочення обсягу оброблюваних даних, яких на поточний момент можуть бути тисячі гігабайтів. Для розв'язування таких підзадач, тобто, для скорочення кількості інформативних даних, часто використовуються методи групування даних для визначення їх спільних характеристик: методи кластеризації.

Задача кластеризації полягає у поділі досліджуваної множини об'єктів на групи "схожих" об'єктів, які називаються кластерами. Відповідно, метод розв'язування задачі розбиття множини елементів на кластери називають кластерним аналізом.

Перші публікації з кластерного аналізу з'явились у кінці 30-х років минулого століття, але активний розвиток цих методів і їх широке використання почалося у кінці 60-х – на початку 70-х років. Надалі цей напрямок багатовимірного аналізу інтенсивно розвивався: з'явилися нові методи, модифікації вже відомих алгоритмів, істотно розширилася область застосування кластерного аналізу. Якщо спочатку методи багатовимірної класифікації використовувалися у психології, археології, біології, то зараз вони стали активно застосовуватися у соціології, економіці, статистиці, історичних дослідженнях. Особливо розширилося їх використання у зв'язку з появою і розвитком ЕОМ і, зокрема, персональних комп'ютерів. Це пов'язано, перш за все, з трудомісткістю обробки великих масивів інформації (обробка матриць великих розмірів).

Для наукових досліджень вивчення результатів кластеризації, а саме – з'ясування причин, за якими об'єкти об'єднуються в групи – здатне відкрити нові перспективні напрямки. Традиційним прикладом, який зазвичай наводять для цього випадку, є періодична таблиця елементів. У 1869 р. Дмитро Менделєєв розділив 60 відомих на той час елементів на кластери або періоди. Елементи, що потрапили у одну групу, мали схожі характеристики. Вивчення причин, за якими елементи розбивалися на явно виражені кластери, значною мірою визначило пріоритети наукових досліджень на роки вперед. Але лише через 50 років квантова фізика дала переконливі пояснення періодичної системи.

Мета кластерного аналізу полягає у пошуку структур, які існують у даних, що виражається в утворенні груп схожих між собою об'єктів (це і є кластерами). Водночас дія кластерного аналізу полягає й у структуризації досліджуваних об'єктів. Це означає, що методи кластеризації необхідні для виявлення структури у даних, яку нелегко знайти при візуальному обстеженні або за допомогою експертів.

Однак, методи кластерного аналізу потребують деякої апріорної (початкової) інформації, зокрема, перед проведенням кластерного аналізу для більшості методів вже потрібно знати кількість кластерів, на які необхідно розбити множину об'єктів. Таку інформацію не завжди можливо отримати внаслідок, наприклад, великої розмірності даних, тому дослідники у багатьох випадках обирають кількість кластерів інтуїтивно, що, зрозуміло, не завжди приводить до значущих результатів. Тому перед проведенням кластеризації видається можливим використати які-небудь інші методи, що дозволили б оцінити (принаймні, приблизно) кількість можливих кластерів.

Однією з груп таких методів є використання штучних нейронних мереж [3, 4, 5]. Серед багатьох видів штучних нейронних мереж є такі, що застосовують при автоматичній класифікації множини об'єктів, тобто, якраз їх розподілі по групах, причому, кількість таких груп заранні невідома.

Метою дослідження є визначення можливості використання конкретного виду штучних нейронних мереж – самоорганізованої карти ознак – для попереднього

визначення можливої кількості кластерів для конкретного методу кластеризації – методу k-means.

Завданнями, висунутими у дослідженні, є:

1) розгляд алгоритму кластеризації k-means та його реалізація;
2) розгляд самоорганізованої карти ознак та її реалізація;
3) дослідження можливостей використання самоорганізованої карти ознак для отримання апріорної інформації стосовно кількості кластерів для алгоритму k-means та вироблення рекомендацій по застосуванню самоорганізованих карт ознак при кластеризації даних.

Виклад основного матеріалу

1. Алгоритм кластеризації k-середніх

Слово cluster (“кластер”) перекладається з англійської як згусток, пучок, група. Споріднені поняття, що використовуються в літературі, – клас, таксон, згущення.

Задача кластеризації (англ. clustering) на відміну від інших відомих класів задач машинного навчання передбачає так зване “навчання без учителя” (англ. “unsupervised learning”): її суть у тому, щоб надати досліднику інструмент для автоматичного поділу наявних об’єктів на класи на підставі подібності тих чи інших характеристик (факторів) цих об’єктів. При цьому ні самі класи, ні їх кількість заздалегідь не відомі. Виникає питання: як комп’ютер виконає розбиття на класи, якщо самі ці класи не відомі? Кластери будуються таким чином, щоб характеристики об’єктів одного класу були дуже близькі і при цьому сильно відрізнялися від характеристик об’єктів інших кластерів. Очевидно, що розуміння “подібності” і “відмінності” сильно залежить від предметної області і навіть від конкретної задачі.

Кластеризація може забезпечити краще розуміння даних і спростити їх подальшу обробку. Наприклад, сегментування (поділ на групи) ринку за будь-якими ознаками споживчої поведінки населення дозволяє проводити адресні акції і різні маркетингові заходи, спрямовані на збільшення обсягу продажів.

Іншим прикладом можна навести задачу класифікації технічних рухомих засобів за деяким критерієм, наприклад – кількістю коліс. Всі об’єкти, подібні мотоциклу, потраплять при цьому в одну групу, а всі об’єкти, подібні автомобілю – у іншу. Ці групи потім аналізуються, і від групи подібних мотоциклу засобів відділяється група мопедів чи скутерів як засобів, що мають двигун внутрішнього згорання меншого об’єму. Група мопедів подібна групі мотоциклів, тому ці групи повинні розміститися близько одна від одної і далеко від групи технічних засобів, подібних автомобілю. Алгоритми кластеризації виконують такі операції зі зразками даних.

Значна перевага кластерного аналізу у тому, що він дозволяє здійснювати розбиття множини об’єктів не за одним параметром, а за цілим набором ознак. Наприклад, у випадку автомобілів та мотоциклів однієї ознаки “кількість коліс” – недостатньо, оскільки існують квадроцикли з чотирма колесами, як у автомобіля, але об’ємом двигуна, як у мотоцикла, і тому необхідна ще одна ознака, за якою проводити класифікацію. Крім того, кластерний аналіз, на відміну від більшості математико-статистичних методів, не накладає ніяких обмежень на вид розглядуваних об’єктів і дозволяє розглядати множину вихідних даних практично довільної природи. Кластерний аналіз дозволяє розглядати досить великий обсяг інформації і різко скорочувати, стискати великі масиви інформації, робити їх компактними і наочними за рахунок того, що усередині кластера об’єкти не розрізняються, тобто, розглядаються, як один об’єкт.

Загалом поділ множини об’єктів на кластери повинен відповідати наступним двом вимогам:

– об’єкти усередині одного кластера повинні бути у певному сенсі подібні (наприклад, велосипеди подібні один до одного, так само мотоцикли, автомобілі, тощо);
– кластери, подібні у певному сенсі, повинні розміщуватися близько один від іншого (наприклад, кластер велосипедів буде розміщений ближче до кластера мотоциклів, аніж кластера автомобілів).

На рис. 1 показано розташування на площині даних, які природним чином організовуються у три кластери: точка, що відповідає зразку, потрапляє у певний кластер, якщо вона розміщена близько до точок цього кластера у порівнянні з точками, які належать іншим кластерам. Мірою близькості (або подібності) двох точок зазвичай є квадрат Евклідової відстані між ними, який обчислюється за формулою

$$d = \sum_{i=1}^{n} \left(x_{pi} - x_{qi}\right)^2 ,$$

де d – квадрат Евклідової відстані між точкою p та точкою q;

x_{pi} – i-а координата точки (зразка) p;

x_{qi} – i-а координата точки (зразка) q;

n – значення розмірності.

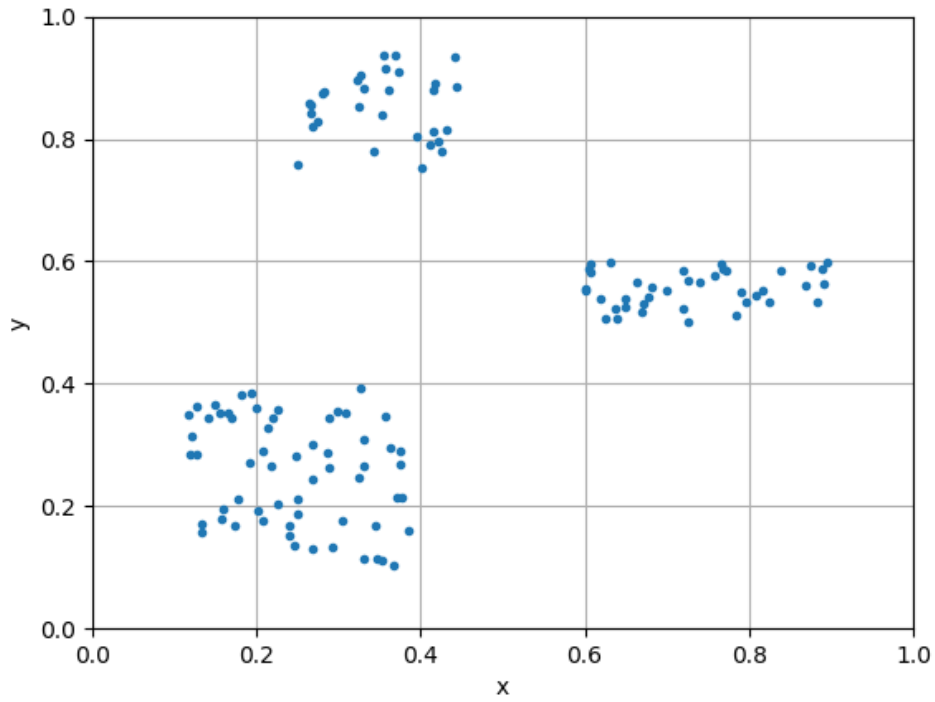


Рис. 1. Дані, що формують три кластери

Якщо для кластера j розглянути вектор $\vec{p}_j$, який визначається центроїдом (точкою, що відповідає усередненій характеристиці розташування усіх зразків у кластері), то для даних на рис. 1 рішення про те, якому з кластерів належить довільний вектор $\vec{x}$ (що зображується на площині у вигляді точки), визначається на основі результату обрахунків:

$$idx(\vec{x}) = min\, d(\vec{p}_j, \vec{x}) \text{ для усіх } j ,$$

де повертається індекс кластера з найменшим квадратом Евклідової відстані до вектора $\vec{x}$. Вектори $\vec{p}_j$ можуть розглядатися як прототипи кластерів, і ці прототипи можуть

служити для представлення ключових ознак кластера. Наприклад, якщо необхідно розділити на групи автомобілі і мотоцикли, цілком зрозуміло, що у якості ознаки можна вибрати кількість коліс. Тому значення елементів, що означають таку кількість у векторах-прототипах двох кластерів, повинні істотно відрізнятися.

Алгоритм кластеризації є статистичною процедурою виділення груп з наявного набору даних. Існує чимало алгоритмів кластеризації найрізноманітніших рівнів складності. Один з найпростіших підходів полягає в тому, щоб припустити існування певної кількості кластерів і довільним чином вибрати координати для кожного з прототипів. Потім кожен вектор з набору даних зв'язується з найближчим до нього прототипом, і новими прототипами стають центроїди усіх векторів, пов'язаних з вихідним зразком. На рис. 2 показаний випадковий вибір прототипів, які до кінця навчання повинні переміститися у центри кластерів, як показано на рис. 3.

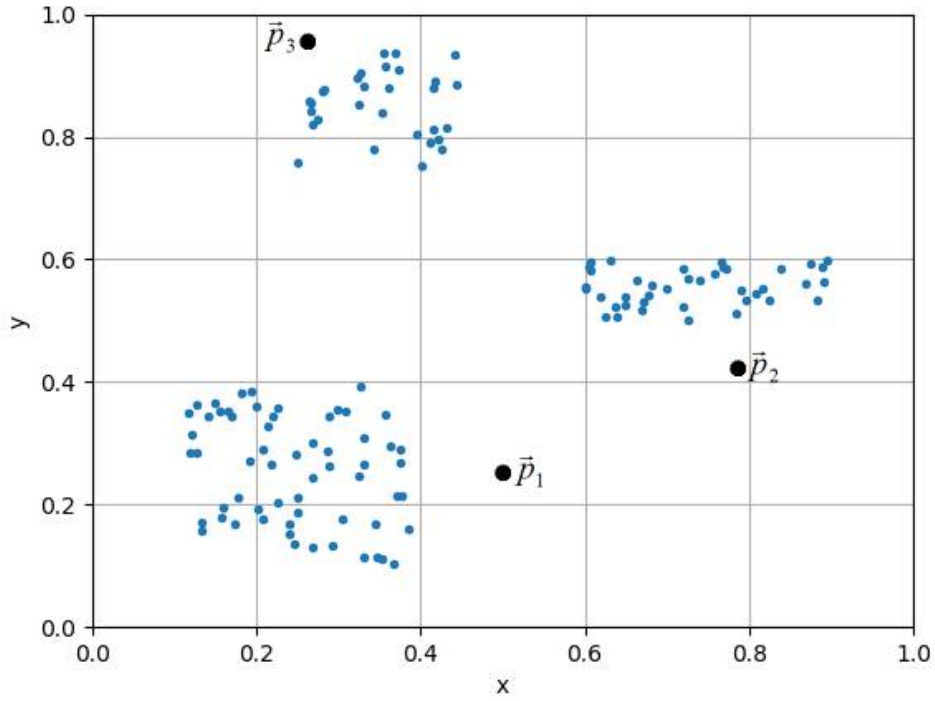


Рис. 2. Три випадкові вектори, що будуть зміщуватись, виступаючи у ролі прототипів для кластерів

Вище наведено ідею одного з найпростіших алгоритмів кластеризації – алгоритму k-means (k-середніх).

У основі алгоритму k-means лежить ітеративний процес стабілізації центроїдів кластерів. Основною характеристикою кластера є його центроїд і уся робота алгоритму спрямована на стабілізацію або – у кращому разі – повне припинення зміни центроїда кластера. Алгоритм k-means будує k кластерів, розміщених на якомога більших відстанях один від одного. Повний опис алгоритму можна знайти у роботі Дж. Хартігана [1]. Нижче подано псевдокод алгоритму.

Алгоритм 1 (алгоритм k-means)
Вхід: множина даних D, кількість кластерів k
Вихід: множина кластерів C, кожний з яких подається точкою центра кластера c_i, та масив *clust*, що ставить у відповідність кожну точку номеру кластера, до якого вона належить

(1) випадковим чином вибрати k точок з множини D
(2) використати вибрані точки у вигляді центрів кластерів C
(3) поки не досягнуто критерія зупинки
  (3.1) перепризначити точки з множини D до найближчого кластера
  (3.2) оновити *clust* таким чином, щоб кожна точка належала кластеру, центр якого знаходиться найближче до неї
  (3.3) оновити C, перерахувавши центри кластерів

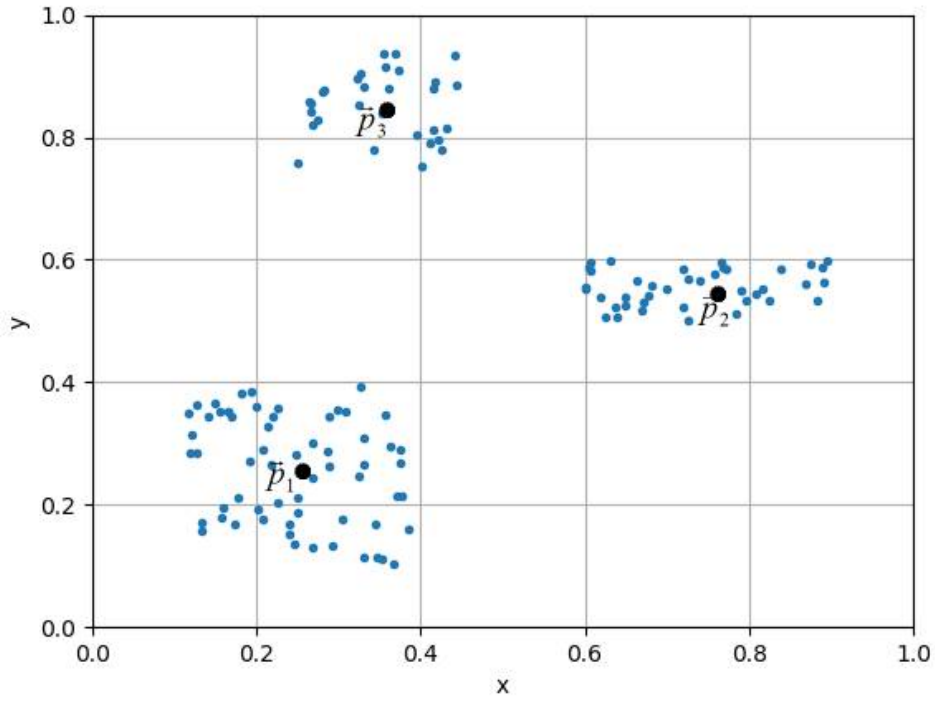


Рис. 3. Кожний з прототипів змістився у точку, що відповідає центроїду кластера

Кластеризація методом k-means починається з вибору k випадково розташованих центроїдів (зразків, що представляють центр кластера). Кожному елементу призначається найближчий центроїд. Після того, як призначення виконано, кожен центр ваги переміщується у точку, яка розраховується, як середнє по усім приписаним до нього елементів. Потім призначення виконується знову. Ця процедура повторюється до тих пір, поки не буде досягнута умова зупинки.

Дія алгоритму така, що він прагне мінімізувати середньоквадратичне відхилення на точках кожного кластера:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \mu_i\right)^2 ,$$

де k – кількість кластерів;
S_i – отримані кластери, $i = 1,2,...,k$;
μ_i – центри мас векторів $x_j \in S_i$.

Під час роботи алгоритму на кожній ітерації переобчислюється центр мас для кожного кластера, отриманого на попередньому кроці, потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближче до вектора

згідно з обраною метрикою. Алгоритм завершується, коли на якійсь ітерації не відбувається зміни кластерів.

Основний тип задач, які розв'язує алгоритм k-means, – наявність припущень (гіпотез) відносно існування певної кількості кластерів у наборі даних, при цьому вони мають бути різні настільки, наскільки це можливо. Вибір числа k може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції.

Умовою зупинки алгоритму може бути одна з наступних:

1) центроїди кластерів більше не змінюються;
2) досягнуто порогового значення помилки;
3) досягнуто певної максимальної (порогової) кількості ітерацій.

Виконання першого критерію є найбажанішим, оскільки його досягнення означає повну стабілізацію структури кластерів. Другий критерій означає, що на черговій ітерації кожний з центроїдів від попереднього положення до наступного змістився не більше ніж на якесь наперед задане значення ε, що називається помилкою. Третій критерій виконується, якщо отримана нестійка структура, що постійно коливається – наприклад, у випадку, коли одна точка по черзі відноситься то до одного, то до іншого кластера.

Перевагами алгоритму k-means є наступні:

- алгоритм простий у використанні;
- висока швидкість роботи алгоритму;
- зрозумілість і прозорість алгоритму.

Однак, алгоритм має й суттєві недоліки:

- висока чутливість до викидів, тобто, окремо розташованих точок;
- повільно працює на великих базах даних;
- не справляється з задачею, коли об'єкт належить до різних кластерів однаково чи не належить жодному з кластерів.

Центроїди можна розглядати як підсумок для досліджуваного набору даних (прототип). Іноді буває зручно представляти кластер кількома прототипами, щоб отримати більш детальну характеристику даних. Щоб розпізнати кластери, які є частинами більшого кластера, необхідно знати положення усіх прототипів один відносно іншого. Однією з проблем застосування алгоритмів кластеризації є вибір оптимальної кількості кластерів. Якщо таку кількість вибрати занадто малою, то можуть бути втрачені деякі важливі характеристики даних, а якщо кластерів виявиться занадто багато, то ми не отримаємо ніякої ефективної підсумкової інформації про дані (може навіть статися, що кожен зразок утворить свій кластер).

Реалізація алгоритму

Алгоритм k-means під час роботи було реалізовано мовою Python.

У програмі реалізовано наступні функції.

Функція `makeInitSet(P)` призначена для створення початкової множини точок, що складається з кількох наборів (кластерів). На вхід функція приймає список `P`, кожен елемент якого є набором даних для формування чергового кластера. Структура одного елементу списку `P` є наступною:

```
(n_i (m_i1, m_i2, ...) (s_i1, s_i2, ...)),
```

де `n_i` – кількість точок;

`s_i1` – розтяг точок вздовж осі абсцис;

`s_i2` – розтяг точок вздовж осі ординат;

`m_i1` – зміщення точок вздовж осі абсцис;

`m_i2` – зміщення точок вздовж осі ординат.

Ідея формування множини точок є наступною. Береться випадкове число x в інтервалі від 0 до 1, на яким потім виконується наступне перетворення:

$$(x \cdot s) + m .$$

Шляхом множення x переноситься з відрізка $[0,1]$ на відрізок $[0,s]$, а за допомогою додавання x переноситься з відрізка $[0,s]$ на відрізок $[m,s+m]$.

Згенерувавши для кожної координати точки різні випадкові числа та задавши різні параметри s та m можна отримати розміщені у різних місцях декартової площини та витягнуті вздовж однієї з осей хмари точок. Повертає функція згенеровану множину точок у вигляді прямокутного масиву координат, де точки розміщені по рядкам.

Наприклад, для набору точок, поданого на рис. 4, було задано наступні параметри:

– для хмари “1”: (50,(0.25,0.65),(0.3,0.4)), тобто, 50 точок, зміщення по x на 0.25, зміщення по y на 0.65, розтяг по x на 0.3, розтяг по y на 0.4;
– для хмари “2”: (50,(0.6,0.5),(0.3,0.3));
– для хмари “3”: (50,(0.1,0.1),(0.5,0.3)).

Функція `makeClusters(X,k)` розбиває множину X на k кластерів, беручи в якості центрів кластерів k випадкових точок. Повертає функція масив центрів кластерів `C` та список (лінійний масив) номерів кластерів, до яких відносяться відповідні точки, `CL`. Для безпосередньо самого розбиття функція використовує функцію `classifyPoints(X,C)`.

Функція `classifyPoints(X,C)` розбиває множину X на кластери, призначаючи кожній точці номер кластера, що відповідає номеру точки з множини `C` (центру кластера), до якої найближче знаходиться дана точка. Повертає функція список (лінійний масив) номерів кластерів, до яких відносяться відповідні точки, `CL`.

Нарешті, функція `kMeans(X,CL,C)` виконує алгоритм k-means для множини точок `X`, заданої початкової множини центрів `C` та початкового розбиття точок на кластери `CL`. Виконується функція або ж кількість ітерацій, що задана в якості четвертого параметра, або ж, якщо параметр не заданий,– поки чергове зміщення центрів кластерів не стане меншим за помилку, задане у функції значення якої 0.001.

Реалізована функція у відповідності з алгоритмом 1, поданим вище.

Роботу реалізованого алгоритму можна прослідкувати на рис. 4-7.

На рис. 4 подано початковий набір точок, згенерованих, як було вказано вище.

На рис. 5 подано початкову конфігурацію центрів кластерів. Для цього було обрано три випадкові точки з поданого набору, які й послужили центрами кластерів. На рисунках центри кластерів виділено маркерами більшого розміру порівняно з і звичайними точками. Як можна бачити, початкові центри кластерів належать вхідному набору 1. Зазначимо, що при різних запусках центри кластерів та саме розбиття точок по кластерам будуть різними, а тому позначення, використані на рисунках – трикутники, круги, хрестики – є умовними, так само, як і кольори.

Після виконання однієї ітерації центри кластерів починають зміщуватись у напрямку реальних кластерів, що наявні в наборі: центр синіх трикутників починає зміщуватись у напрямку множини 3, центр зелених хрестиків – у напрямку множини 2, центр чорних точок – переміщується по множині 1.

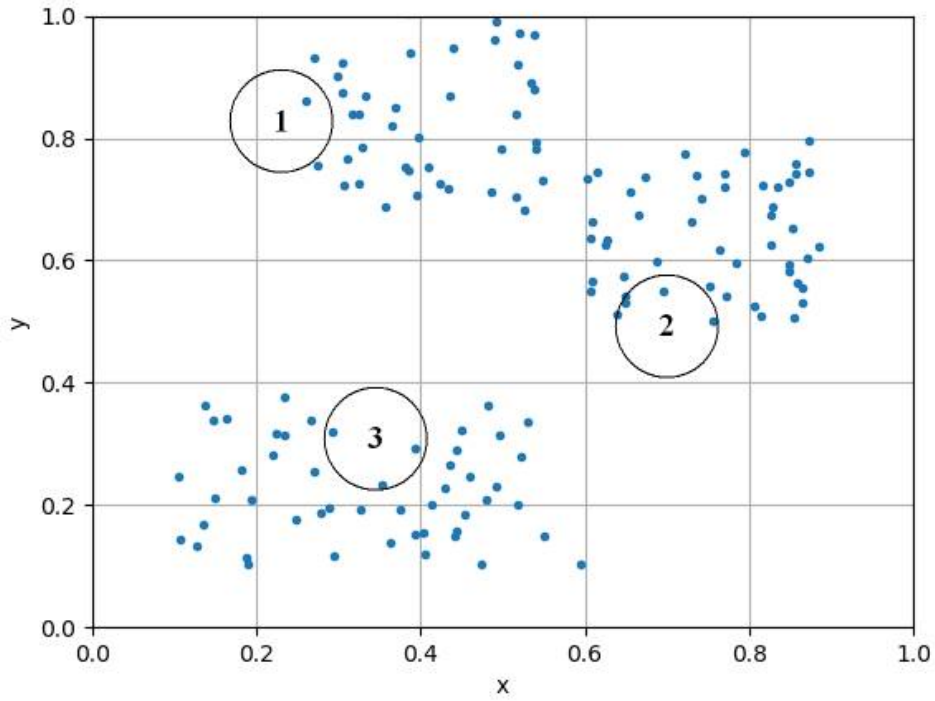


Рис. 4. Початковий набір точок. Точки розміщено так, щоб було чітко видно три кластери (1, 2, 3)

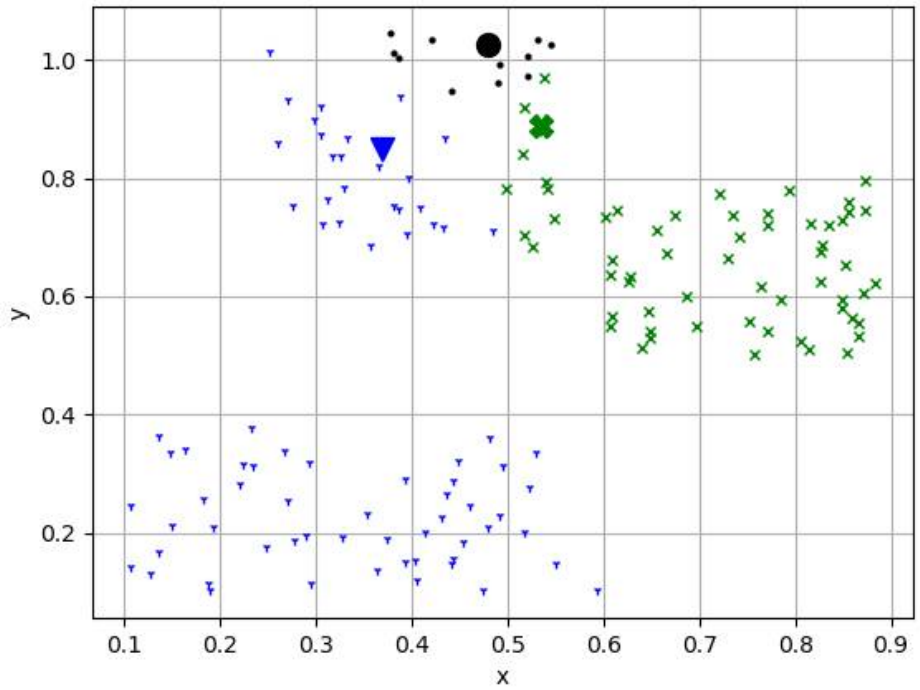


Рис. 5. Випадковим чином вибрані центри кластерів та проведено розподіл точок по кластерам

Нарешті, на рис. 7 подано остаточну – стабільну – конфігурацію, отриману в результаті виконання алгоритму k-means. Як видно, у даному випадку було знайдено правильні кластери, що тепер можна ідентифікувати за їх центрами.

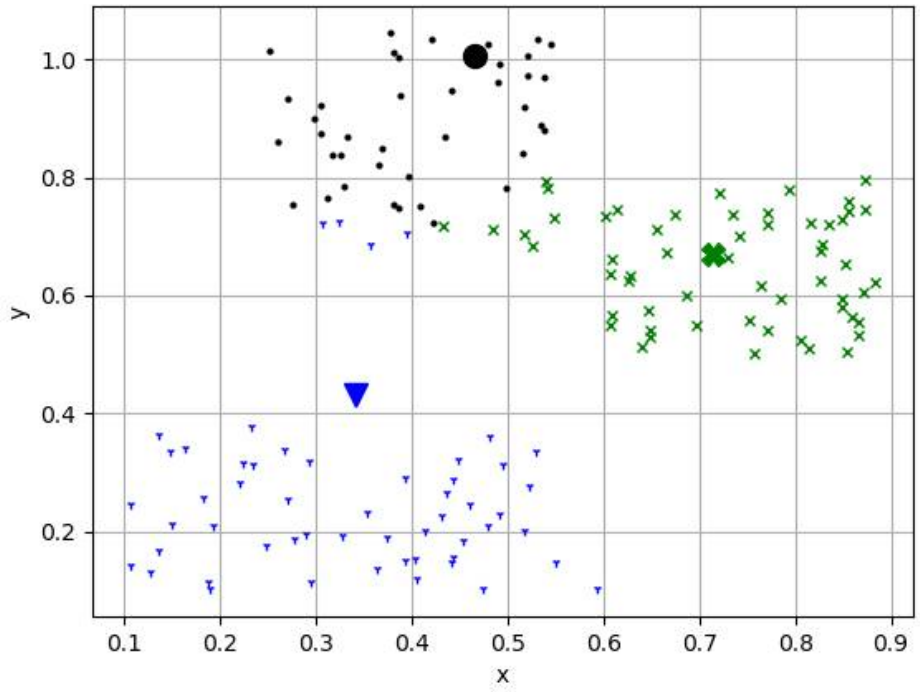


Рис. 6. Виконана одна ітерація алгоритму k-means. Центроїди починають наближатись до центрів кластерів

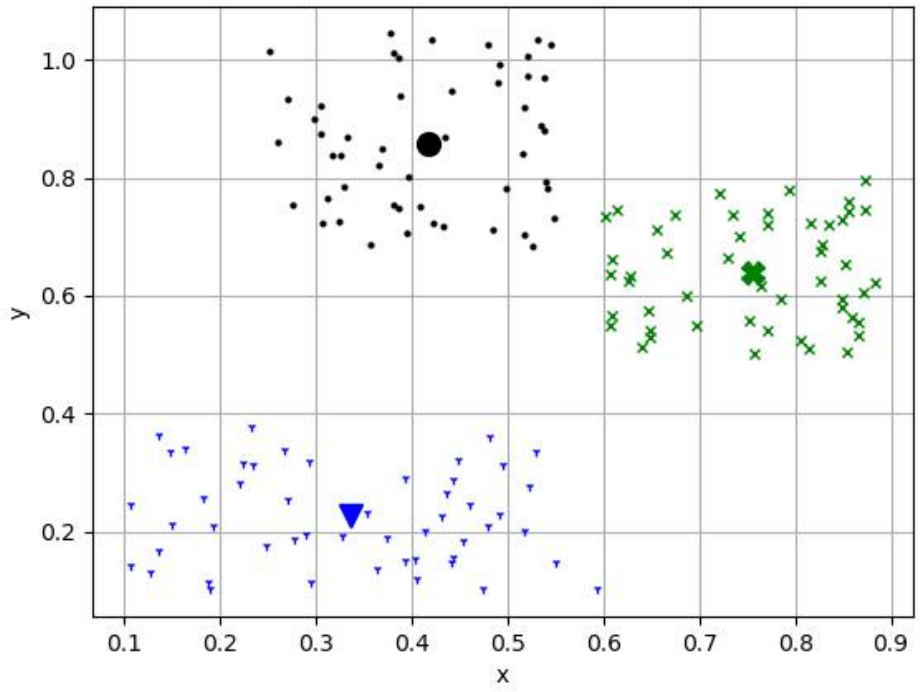


Рис. 7. Закінчено роботу алгоритму k-means. Центри кластерів стабілізувались

Дослідження роботи описаного й реалізованого алгоритму дозволило виявити наступні недоліки.

1. Іноді алгоритм може не знаходити правильні кластери навіть при чітко вираженій структурі набору точок, що залежить від початкового вибору центрів кластерів: при невдалому виборі кластери алгоритм може зупинитись при стабілізації центрів і неправильному визначенні кластерів. Одна з таких конфігурацій зображена на

рис. 8. Як видно з рисунка, виділено неправильні кластери, коли до одного попали множини 1 та 2 з рис. 4 (параметри множин однакові, хоча при різних запусках програми генеруються різні координати точок), а два інші кластери ділять між собою множину 3. Хоча така помилка виникає рідко, вона, все ж, існує. Однак, для того, щоб уникнути цієї помилки, можна виконувати процедуру кластеризації кілька разів, вибираючи, в результаті, кластери, що є однаковими після кількох різних запусків програми.

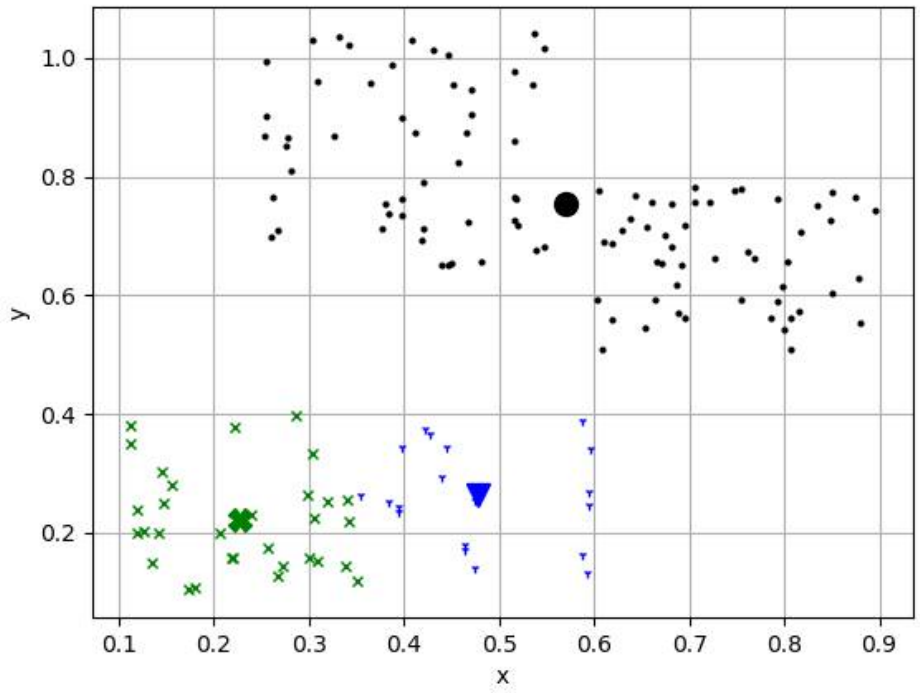


Рис. 8. Стабілізація кластерів при неправильному їх визначенні

2. При вказанні неправильної кількості кластерів алгоритм генерує неправильний результат. Наприклад, на рис. 9 подано результат виконання програми при використанні двох множин з різними параметрами та чотирьох кластерів.

Для того, щоб позбавитись недоліку 2, ми пропонуємо використати один зі способів попередньої оцінки можливої кількості кластерів, використавши результати роботи нейронної мережі з навчанням без учителя, що являє собою самоорганізовану карту ознак – мережу Кохонена.

Отже, у даному розділі розглянуто алгоритм пошуку кластерів k-means, описано результати роботи реалізації алгоритму мовою Python, та після аналізу результатів роботи програми визначено недоліки алгоритму.

2. Самоорганізована карта ознак Кохонена

Самоорганізована карта ознак (мережа SOFM – Self-Organizing Feature Map) [2, 3, 5] має набір вхідних елементів, кількість яких відповідає розмірності навчальних точок, і набір вихідних елементів, які виступають у якості прототипів. Базова архітектура мережі SOFM показана на рис. 10. Наприклад, для даних, показаних на рис. 1, буде необхідною мережа з двома вхідними і принаймні трьома вихідними елементами, що представляють три кластери.

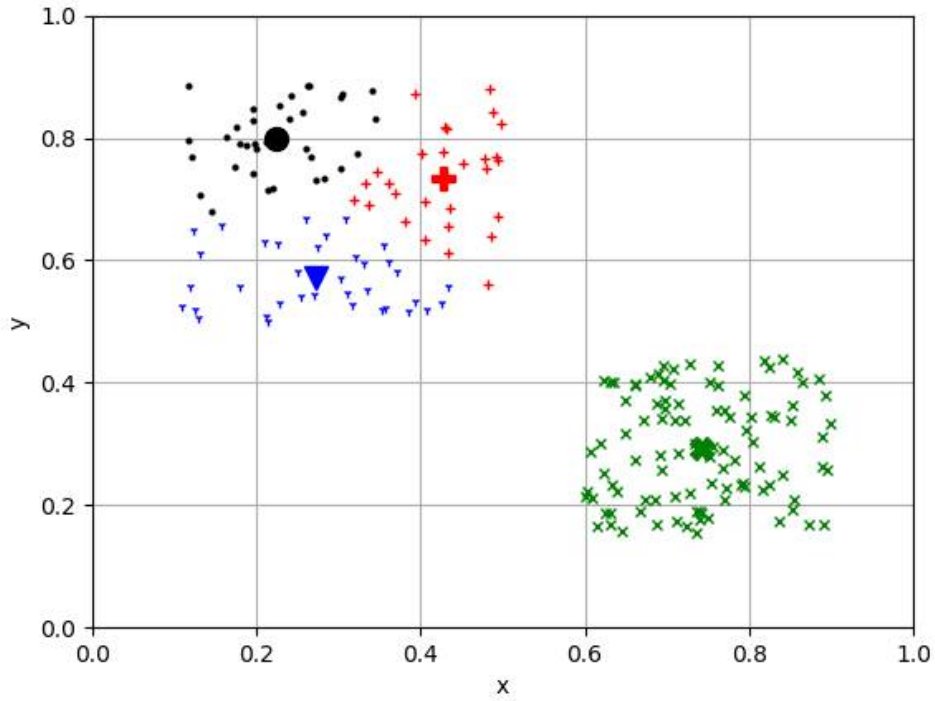


Рис. 9. Утворено 4 кластери при явних двох кластерах. Така конфігурація є результатом неправильно заданої кількості шуканих кластерів для алгоритму k-means

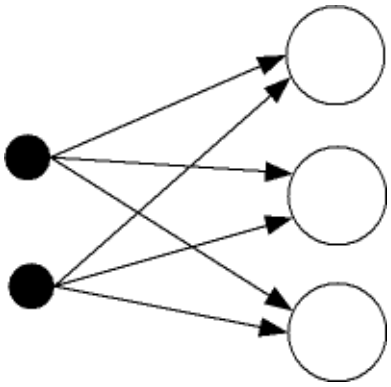


Рис. 10. Мережа має два вхідних і три кластерних елементи, причому кожний елемент вхідного шару пов'язаний з кожним елементом кластерного шару

Вхідні елементи призначені лише для того, щоб розподіляти дані вхідної точки між вихідними елементами мережі. Вихідні елементи називаються кластерними елементами.

Кожний зі зв'язків від вхідного елемента до вихідного має свою вагу, що виражається певним числом. Такі ваги позначаються w_{ij}, де i – номер вихідного елемента мережі, а j – номер вхідного елемента, або координата вхідної точки. Оскільки кількість вхідних елементів відповідає розмірності вхідних точок, а кожен вхідний елемент пов'язаний з усіма кластерними елементами, загальна кількість вагових значень, що впливають на кластерний елемент, теж виявляється рівною розмірності вхідних точок. Часто зручно інтерпретувати вагові значення кластерного елемента як значення координат, що описують позицію кластера у просторі вхідних

даних. На рис. 11 показано, як пов'язуються вагові значення з простором вхідних даних.

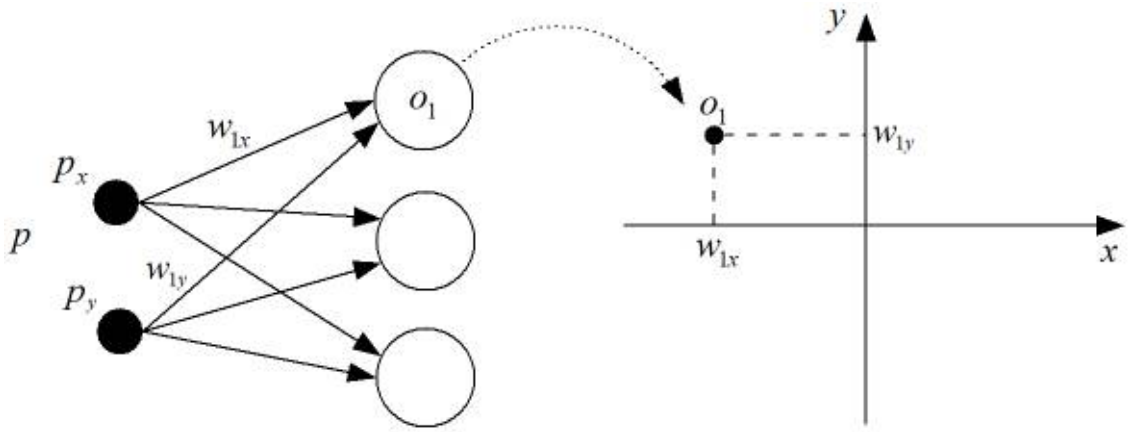


Рис. 11. На схемі показано, як в якості центроїда може виступати кластерний елемент. Вхідний шар використовує значення координат p_x, p_y точки p вхідного простору. У процесі навчання вагові значення коригуються й по закінченні навчання усі кластерні елементи займуть у просторі вхідних даних положення, що відповідає отриманим для них ваговим значенням

Кластерні елементи розміщуються у вигляді одно- або двовимірного масиву, як показано на рис. 12. Під час навчання усі елементи можуть розглядатися як претенденти на нагороду у вигляді навчальних точок. Коли на конкурс виставляється будь-яка навчальна точка, обчислюються відстані від неї до усіх кластерних елементів, і елемент, який знаходиться до даної навчальної точки найближче, оголошується елементом-переможцем.

Для елемента-переможця проводиться коригування вагових значень так, щоб цей кластерний елемент розмістився до навчальної точки ще ближче. Зазвичай коригування вагових значень виконується і для елементів, що знаходяться поруч з елементом-переможцем. Вагові значення елемента оновлюються, якщо він знаходиться усередині кола заданого радіуса з центром у елементі-переможці.

Під час навчання радіус зазвичай поступово зменшується. Норма навчання обмежує величину, на яку кластерний елемент може пересунутися у напрямку до навчального вектора, і, подібно до радіуса, норма навчання теж з часом поступово зменшується. У прикладах, поданих у роботі, розглядається розміщення кластерних елементів у вигляді квадратної сітки, але можуть використовуватися й інші форми, наприклад: лінійне розміщення елементів чи шестикутна сітка. Від топології залежить лише те, які елементи повинні оновлюватися для даного конкретного радіуса.

Зазвичай кількість кластерних елементів вибирають меншою, аніж кількість навчальних зразків, оскільки метою є отримання спрощеної характеристики даних. До кінця навчання кластерні елементи забезпечують «інформаційний звіт» по простору вхідних зразків. Кластерні елементи виступають у ролі карти ознак простору вхідних даних. Наприклад, у результаті кластеризації зображень символів різні області кластерних елементів будуть характеризувати різні символи чи їх комбінації.

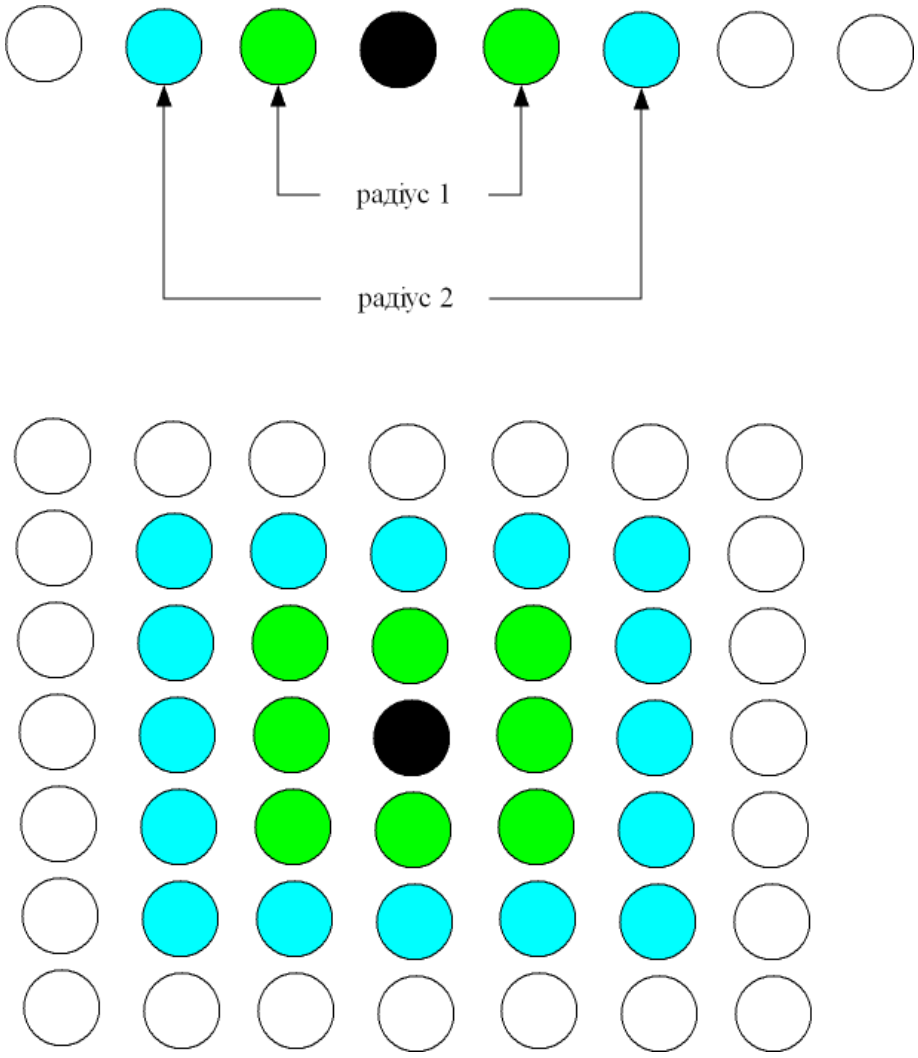


Рис. 12. Угорі кластерні елементи розташовані у один ряд, внизу – у вигляді квадратної сітки. Від розташування залежить, які саме елементи попадуть усередину круга заданого радіуса з центром у елементі-переможці. У випадку радіуса 1 оновлюється чорний та зелені елементи, для радіуса 2 – чорний, зелені та блакитні елементи

Навчання мережі SOFM

У наступному алгоритмі η позначає норму навчання, а n – крок у часі.

Алгоритм 2 (алгоритм навчання мережі SOFM)

Вхід: множина точок D, кількість k кластерних елементів у сітці

Вихід: змінені ваги кластерних елементів, що трактуються як координати розміщення кластерних елементів

(1) ініціалізувати вагові значення випадковими числами

(2) виконувати, поки не виконається критерій зупинки

(2.1) для кожного вхідного вектора

(2.1.1) для кожного кластерного елемента обчислити відстань до навчального вектора:

$$d_j = \sum_i \left(w_{ij} - x_i\right)^2$$

(2.1.2) знайти елемент j, для якого відстань є найменшою

(2.1.3) для елементів з кола заданого радіуса оновити вагові вектори за формулою

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)(x_i - w_{ij}(n))$$

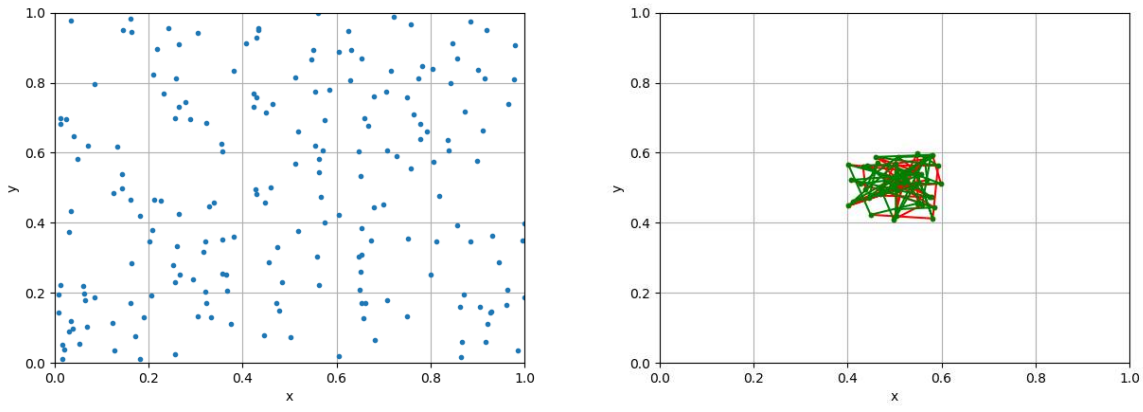
(2.1.4) перевірити, чи потрібне оновлення норми навчання або радіусу

(2.1.5) перевірити критерій зупинки

Навчальні точки вибираються з вхідної множини точок випадковим чином. Умова зупинки виконується тоді, коли величини зміни вагів для усіх кластерних елементів стають дуже малими: за цієї умови навчальні точки повинні потрапляти в одні й ті ж зони карти при переході від однієї епохи (ітерації) до наступної.

Норма навчання з часом змінюється. Вона може, наприклад, спочатку мати значення 0.9, а потім зменшуватися лінійно до деякого фіксованого мінімального значення (наприклад, 0.001), після чого залишатися незмінною. Радіус зазвичай вибирається досить великим, щоб спочатку оновлювалися усі елементи. Радіус теж згодом зменшується, і в кінці, як правило, повинні оновлюватися лише кілька сусідніх з елементом-переможцем елементів, або взагалі лише сам цей елемент. Норма навчання теж може залежати від того, наскільки близько розміщується оновлюваний елемент до елемента-переможця.

Карта ознак проходить два етапи навчання. На першому етапі елементи упорядковуються так, щоб відображати простір вхідних елементів, а на другому етапі відбувається уточнення їх позицій. Як правило, процес представляється візуально шляхом використання двовимірних даних і побудови відповідної поверхні. Наприклад, вхідні вектори вибираються випадковим чином на основі рівномірного розподілу у деякому квадраті, і починається навчання карти. У певні моменти у ході навчання будуються зображення карти шляхом використання відповідності, показаної на рис 11. Елементи з'єднуються лініями, щоб показати їх відносне розташування. Спочатку карта має вигляд сильно "зіжмаканої", але поступово у ході навчання вона розгортається і розправляється. Кінцевим результатом навчання є карта, що покриває увесь вхідний простір і є досить регулярною (тобто, елементи виявляються розподіленими майже рівномірно). Для прикладу було розглянуто карту з топологією квадрату з 49 елементів (7×7), і для 200 точок даних (рис. 13a), взятих з одиничного квадрату, було проведене її навчання, яке починалося з випадкового набору вагових значень, що задають розміщення кластерних елементів у центрі вхідного простору, як показано на рис. 13b.

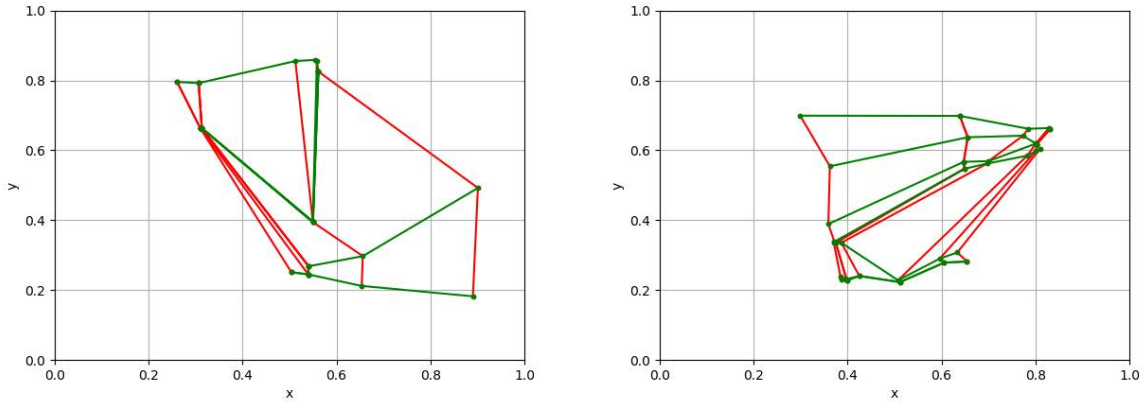


a) b)

Рис. 13. a) початковий набір точок з квадрату $[0,1]\times[0,1]$; b) вагові вектори ініціалізуються випадковими значеннями з відрізку $[0.4,0.6]$

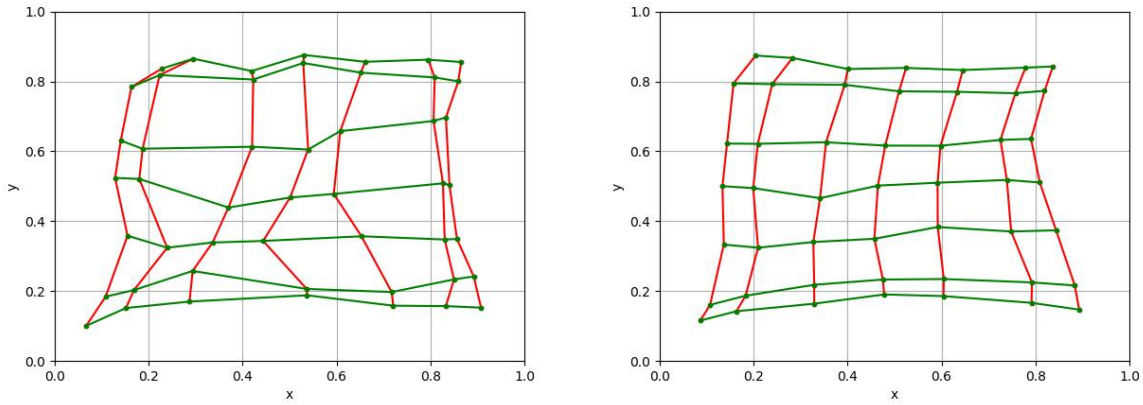
На рис. 14 і 15 ілюструється процес розгортання карти з часом. Як і для інших типів мереж, у даному випадку результат навчання залежить від навчальних даних і вибору параметрів навчання.

Отримана в результаті навчання мережі SOFM карта подана на рис. 15b. При навчанні використовувалось зменшення радіусу на 1 після кожних 400 ітерацій, а зменшення норми навчання – на 0.00035 на кожній ітерації. Всього було проведено 3000 ітерацій, при цьому мінімальне значення радіуса було встановлене в 1, а мінімальне значення коефіцієнта навчання – 0.001.



a) b)

Рис. 14. a) карта вихідних елементів після 500 ітерацій навчання мережі SOFM; b) карта вихідних елементів після 1000 ітерацій навчання мережі SOFM. Елементи, розміщені горизонтально на карті, з'єднані зеленими відрізками, розміщені вертикально на карті, з'єднані червоними відрізками



a) b)

Рис. 15. a) карта вихідних елементів після 2000 ітерацій навчання мережі SOFM; b) карта вихідних елементів після 3000 ітерацій навчання мережі SOFM

Як видно з рис.15b, отримана карта Є не абсолютно рівною. Більше того, в результаті навчання мережі можна отримати навіть "зім'яті" ділянки. Скручування можуть бути результатом неправильного вибору початкових вагових значень, які на початку навчання задають місце розташування кожного кластерного елемента у просторі вхідних даних. Наприклад, якщо на початку навчання усі елементи мають такі ваги, що вони виявляються зміщеними у одному напрямку у просторі вхідних даних, то процес упорядкування може ускладнитися.

У випадку, якщо, наприклад, карта буде лінійним масивом, під час навчання цей лінійний масив скручується так, щоб заповнити простір вхідних даних. Ті ж властивості просторового заповнення можна застосувати й у разі інших форм. Наприклад, якщо набір навчальних даних вибирається рівномірно з трикутника, то можна розмістити елементи квадратної карти так, щоб ці елементи заповнили трикутник.

Реалізація мережі SOFM

Подані у даному розділі рисунки 13-15 створені за допомогою розробленої під час написання програми навчання мережі SOFM.

Для генерації початкових даних використовується функція `makeInitSetSOFM(P,d)`, що приймає в якості аргументів параметри, схожі на параметри функції `makeInitSet(P)`, описаної у попередньому розділі.

Значення вагів зв'язків генеруються за допомогою функції `makeWSOFM2d(nX,nY,limits)`, що приймає в якості параметрів кількість вхідних елементів `nX`, кількість вихідних елементів `nY`, та список коефіцієнтів масштабування значень параметрів, які, зазвичай, повинні знаходитись у квадраті $[0.4,0.6]\times[0.4,0.6]$.

Виконання однієї ітерації навчання проводиться у функції `makeOneIteration2d(X,W,R,etha)`, що приймає в якості параметрів множину точок `X`, набір вагів вихідних елементів мережі `W`, радіус `R` для визначення оновлюваних елементів, та норму навчання `etha`. У функції реалізовані кроки 2.1.1-2.14 алгоритму 2.

Нарешті, додаткова функція `classifySOFMPoints(X,W)` призначена для розподілу точок вхідної множини X по кластерам, заданим ваговими коефіцієнтами (чи, іншими словами, вихідними елементами) `W`. Повертає функція прямокутний масив кількості точок, розміщених у кожному з кластерів. На відміну від алгоритму кластеризації k-means, у функції не визначається належність кожної точки до певного кластеру, оскільки завданням цієї функції є лише визначення кількості точок, що належать до кожного кластеру для подальшого прийняття рішення про кількість наявних кластерів у вхідних даних. Наприклад, для набору точок, зображеного на рис. 13а, результат розподілу точок по кластерам є наступним:

```
[[ 6.  1.  7.  5.  2.  6. 11.]
 [ 2.  5.  1.  3.  4.  2.  4.]
 [ 5.  3.  4.  3.  4.  7.  4.]
 [ 4.  3.  2.  4.  2.  2.  4.]
 [ 5.  3.  5.  3.  3.  6.  5.]
 [ 6.  3.  3.  3.  2.  1.  2.]
 [ 8.  3.  7.  2.  6.  3. 11.]]
```

З наведеного прикладу видно, що до першого (верхній лівий) кластера віднесено 6 точок, сьомого (верхній правий) – 11 точок, останнього (нижній правий) – 11 точок. Приблизно рівномірне розміщення точок у кластерах пояснюється їх приблизно рівномірним розподілом по одиничному квадрату (див. рис. 13а). Однак, ***головним припущенням даної роботи*** є те, що при вираженій кластеризації точок їх розподіл по кластерним елементам карти SOFM буде таким, що дозволить оцінити кількість наявних кластерів.

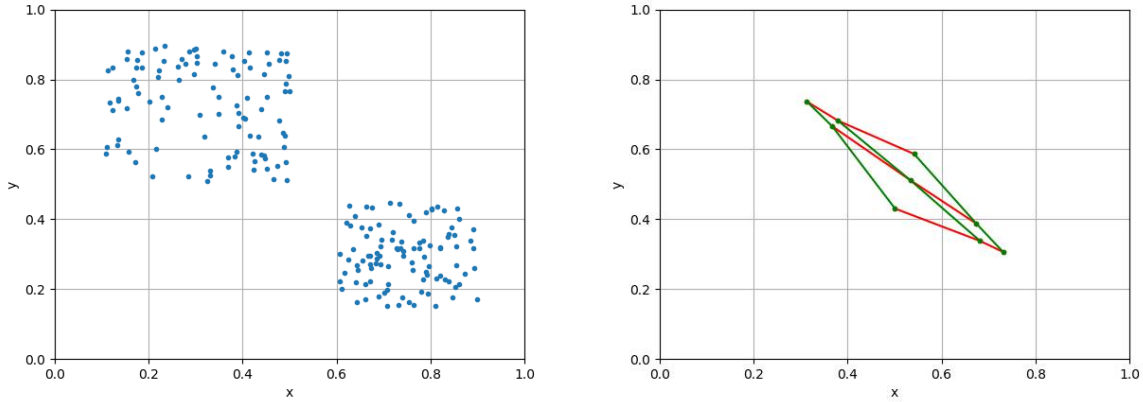
Отже, у даній частині роботи розглянуто алгоритм карти SOFM, описано результати роботи реалізації алгоритму мовою Python, та сформульовано головну гіпотезу роботи.

3. Алгоритм автоматичного визначення кількості кластерів

Для перевірки гіпотези, сформульованої у кінці розділу 2, було об'єднано програми, розроблені у двох попередніх розділах.

У отриманій програмі спочатку генерується набір даних подібно до наведеного на рис. 1, тобто, таких, що мають яскраво виражені кластери. Потім на згенерованих даних проводиться навчання мережі SOFM та аналізується отримана інформація.

У експерименті 1 було згенеровано множину точок, що складається з двох кластерів (рис. 16a). В якості вихідних елементів мережі SOFM було обрано сітку розміром 3×3, що в результаті може давати до 9 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 16b.



a) b)

Рис. 16. a) вхідна множина точок з двома вираженими кластерами; b) карта вихідних елементів розміром 3×3 після 3000 ітерацій навчання мережі SOFM

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[ 0. 17. 66.]
 [15.  5. 17.]
 [54. 19.  7.]]
```

У даному випадку видно дві особливості результату експерименту.

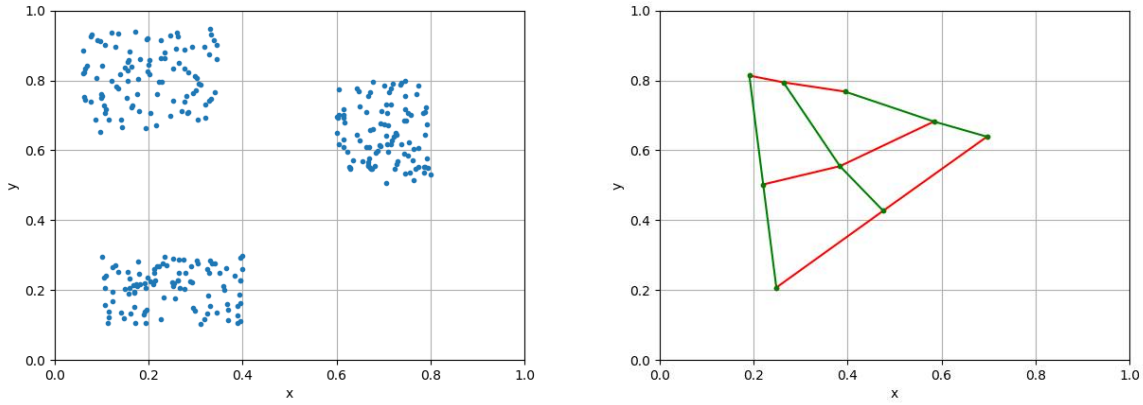
1. Отримано сітку, у якій змінилось топографічне розташування елементів: елементи, що знаходяться у сітці по рядках, розташувались переважно вертикально і навпаки.

2. Отримано 2 кластерні елементи з суттєво більшою кількістю точок – 66 та 54 – порівняно з іншими кластерами. Кількість таких кластерів рівна 2.

У експерименті 2 було згенеровано множину точок, що складається з трьох кластерів (рис. 17a). В якості вихідних елементів мережі SOFM було обрано сітку розміром 3×3, що в результаті знову може давати до 9 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 17b.

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[82.  2. 98.]
 [18.  0.  0.]
 [ 5. 34. 61.]]
```



a) b)

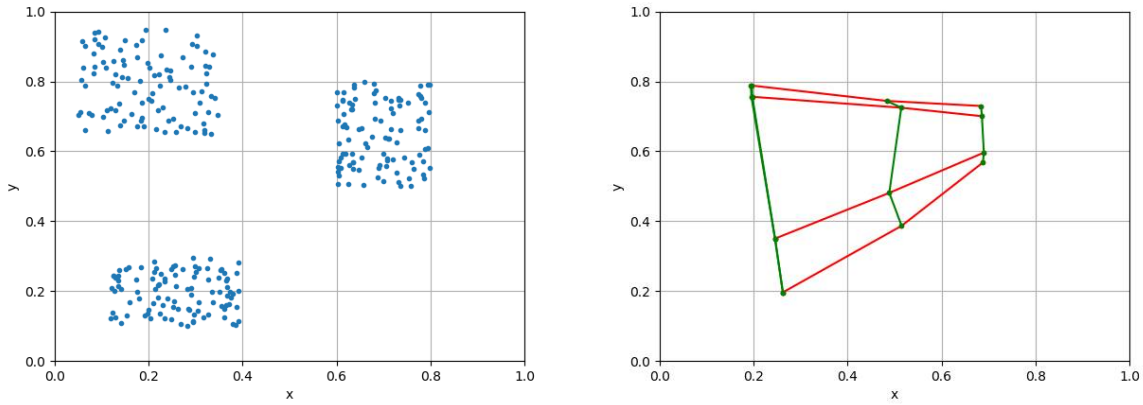
Рис. 17. a) вхідна множина точок з трьома вираженими кластерами; b) карта вихідних елементів розміром 3×3 після 3000 ітерацій навчання мережі SOFM

Результатами експерименту є наступні.

1. Отримано сітку, звужену вправо, де на вихідній множині знаходиться лише один кластер.

2. Отримано 3 кластерні елементи з суттєво більшою кількістю точок – 98, 82, 61 – порівняно з іншими кластерами. Кількість таких кластерів рівна 3.

Нарешті, у експерименті 3 було згенеровано множину точок, що складається з трьох кластерів (рис. 18a), але в якості вихідних елементів мережі SOFM було обрано сітку розміром 4×4, що в результаті може давати до 16 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 18b.



a) b)

Рис. 18. a) вхідна множина точок з трьома вираженими кластерами; b) карта вихідних елементів розміром 4×4 після 3000 ітерацій навчання мережі SOFM

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[34.  0. 48. 45.]
 [17.  0.  0.  7.]
 [17.  0. 27. 16.]
 [32.  2. 20. 35.]]
```

Результатами експерименту є наступні.

1. Отримано сітку, звужену вправо, де на вихідній множині знаходиться лише один кластер.

2. Отримано 3 групи кластерних елементів, розділені між собою кластер ними елементами з кількістю точок, менше 10. Першою групою є перший стовпчик сітки: 34, 17, 17, 32. Другою групою є два верхні праві елементи сітки: 48, 45. Третьою групою є 4 елементи у нижньому правому куті: 27, 16, 20, 35.

Таким чином, результати експериментів показують можливість застосування мережі SOFM для оцінки кількості кластерних елементів. Однак, сама оцінка вважається на поточний момент складнішою, аніж просто визначення кількості вихідних елементів мережі SOFM з високими значеннями кількості точок вхідної множини, що їм відповідають.

Таким чином, на основі проведеного дослідження можна сформулювати наступний алгоритм визначення кластерів та проведення кластеризації множини об'єктів.

Алгоритм 3 (кластеризація множини точок з апріорною оцінкою кількості кластерів)
Вхід: множина точок D
Вихід: множина кластерів C, кожний з яких подається точкою центра кластера c_i, та масив $clust$, що ставить у відповідність кожну точку номеру кластера, до якого вона належить

(1) навчити на множині D мережу SOFM
(2) підрахувати кількість елементів множини D, що відповідають кожному вихідному елементу мережі SOFM
(3) провести аналіз розподілу кількості елементів, що відповідають вихідним елементам мережі SOFM та визначити апріорну кількість кластерів k
(4) провести кластеризацію множини D за алгоритмом k-means з використанням кількості кластерів k

У даному випадку недолік алгоритму k-means, що стосується неправильної початкової кількості кластерів, буде дещо згладженим за рахунок попередньої оцінки та отримання більш адекватної кількості кластерів.

Однак, зауважимо, що саме крок (3) алгоритму 3 потребує додаткових досліджень.

Ще одним напрямком подальшого дослідження є збільшення розмірності простору ознак та оцінка результатів роботи алгоритму 3 на множині точок більшої розмірності.

Висновки

Отже, у результаті виконання дослідження:

1. Розглянуто алгоритм пошуку кластерів k-means, описано результати роботи реалізації алгоритму мовою Python, та визначено недоліки алгоритму.

2. Розглянуто та реалізовано мовою Python алгоритм навчання мережі SOFM.

3. Сформульовано та перевірено гіпотезу про можливість застосування мережі SOFM для попередньої оцінки кількості кластерів у заданому наборі точок. Сформовано алгоритм та визначено подальші напрямки дослідження.

Список використаної літератури:

1. Hartigan, J.A. (1975) Clustering Algorithms. John Wiley & Sons, New York, 351 p.
2. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, 78, 1464-1480. http://dx.doi.org/10.1109/5.58325
3. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.
4. Рашид Т. Создаем нейронную сеть.: Пер. с англ. – СПб.: ООО "Альфа-книга", 2017. – 272 с.
5. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.

References:

1. Hartigan, J.A. (1975) Clustering Algorithms. John Wiley & Sons, New York, 351 p.
2. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, 78, 1464-1480. http://dx.doi.org/10.1109/5.58325
3. Osovsky S. Neural Networks for information processing. – M.: Finance and Statistics, 2002. – 344 c. [in Russian].
4. Rashid T. Making neural networks. – SPb.: "Alpha-book" LTd, 2017. – 272 c. [in Russian].
5. Haikin S. Neural networks, 2nd Ed. – M.: Williams Publishing, 2006. – 1104 c. [in Russian].

SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Chair of Department of Applied Mathematics and Informatics, Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIKHTIARENKO Volodymyr,
Drabivskyi NVK "Secondary School of I-III st. named after S.V. Vasylchenko-gymnasium" of Drabiv District Council
BILETSKA Nadiya,
Teacher of Mathematics of the High Qualification Category, Pedagogical Title "Teacher-Methodologist" of Drabivskyi NVK "Secondary School of I-III st. named after S.V. Vasylchenko-gymnasium" of Drabiv District Council

PRELIMINARY DETERMINATION OF THE NUMBER OF CLUSTERS USING SOFM ARTIFICIAL NEURAL NETWORKS

Summary. Introduction. *The task of clustering is to divide the set of objects under study into groups of "similar" objects, called clusters. The purpose of cluster analysis is to find the structures that exist in the data, which is expressed in the formation of groups of similar objects. At the same time, the effect of cluster analysis is to structure the objects under study. This means that clustering techniques are needed to identify a structure in the data that is not easy to find by visual inspection or with the help of experts. However, cluster analysis methods require some a priori information. In particular, before performing a cluster analysis, most methods already need to know the number of clusters. Such information is not always possible due to, for example, the large dimension of the data, so researchers in many cases choose the number of clusters intuitively, which, of course, does not always lead to significant results. Therefore, before clustering, it seems possible to use any other methods that would allow to estimate at least approximately the number of possible clusters.*

Purpose. *The aim of the study is to determine the possibility of using a specific type of artificial neural networks - a self-organized map of features - to pre-determine the number of clusters for a particular method of clustering - the method of k-means. The tasks set in the study are: reviewing of the k-means clustering algorithm and its implementation; reviewing of a self-organized map of signs and its implementation; to analyze the possibilities of using a self-organized feature map to obtain a priori information on the number of clusters for the k-means algorithm and develop recommendations for the use of self-organized feature maps in data clustering.*

Results. *One of the simplest clustering algorithms is k-means, which is based on an iterative process of stabilizing cluster centroids. The main characteristic of the cluster is its centroid and all the work of the algorithm is aimed at stabilization or - at best - complete freezing of changes in the centroid of the cluster. The k-means algorithm builds k clusters placed at as large distances from each*

other as possible. The main type of problem solved by the k-means algorithm is the presence of hypotheses about the existence of a certain number of clusters in the data set, and they should be as different as possible. The choice of the number k can be based on the results of previous research, theoretical considerations or intuition. When specifying the wrong number of clusters, the algorithm generates the wrong result, to avoid which we propose to use one of the methods of preliminary estimation of the possible number of clusters, using the results of neural network with learning without a teacher, which is a self-organizing map - Kohonen network.

The self-organized map of features has a set of input elements, the number of which corresponds to the dimension of the training points, and a set of output elements that act as prototypes. Cluster elements are placed in the form of a one- or two-dimensional array. During the training, all elements can be considered as candidates for the award in the form of training points. For the winning element, the weight values are adjusted so that this cluster element is located even closer to the training point. The weight values of an element are updated if it is inside a circle of a given radius centered in the winning element. During training, the radius usually gradually decreases. In the first stage, the elements are arranged so as to reflect the space of the input elements, and in the second stage, their positions are refined. Typically, the process is represented visually by using two-dimensional data and constructing an appropriate surface.

During our work we made a series of experiments to combine the two above algorithms, using the results of the SOFM network as a priori data for the k-means algorithm. The results of the experiments show the possibility of using the SOFM network to estimate the number of cluster elements. However, the estimate itself is currently considered more complex than simply determining the number of SOFM network outputs with high values of the number of input set points corresponding to them.

***Conclusion.** As a result of the work we have hext conclusions. 1. The algorithm of k-means is reviewed, the results of work of algorithm implementation in Python language are described, and shortcomings of algorithm are defined. 2. The algorithm of SOFM network learning is reviewed and implemented in Python language. 3. The hypothesis about the possibility of using the SOFM network for preliminary estimation of the number of clusters in a given set of points is formulated and tested. An algorithm is formed and further directions of research are determined.*

***Keywords:** machine learning, clustering, k-means, neural networks, SOFM.*

Одержано редакцією 08.02.2021 р.
Прийнято до публікації 10.06.2021 р.