Доповідь

Кравець Ольга

Група ПМО-41

“Карта Кохонена”

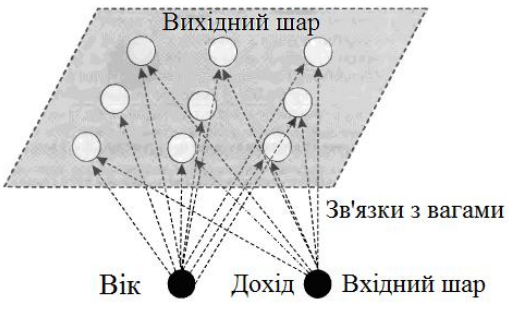
(Самоорганізаційна карта Кохонена)

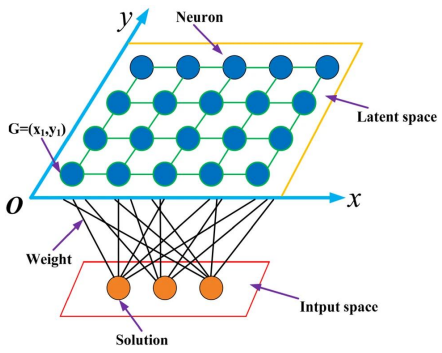
Поняття мережі Кохонена

Мережа Кохонена з’явилася у 1982 р., як спеціальний вид нейронних мереж для обробки зображень і звука. Основною метою мереж Кохонена є перетворення складних багатомірних даних в більш просту структуру малої розмірності. Але виявилось, що вони гарно пристосовані для кластерного аналізу, коли треба виявити приховані закономірності у великих масивах даних.

Мережа Кохонена складається з вузлів, які об’єднуються в кластери. Найбільш близькі вузли відповідають схожим об’єктам, а віддалені – не схожим. В основі побудови мережі лежить конкурентне навчання, коли вихідні вузли (нейрони) конкурують між собою за «перемогу». У ході змагань в процесі навчання нейрони вибірково налагоджуються для різних вхідних прикладів

Ілюстративний приклад для демонстрації 2-D SOM. У прикладі кожен нейрон має заздалегідь визначену позицію в латентному просторі $G = (x_1, y_1),\ x_1, y_1 \in \{1, 2, 3, 4, 5\}$





SOM складається з вхідного рівня та латентного рівня. Вхідний рівень є n-вимірним і використовує рішення як навчальні точки. Що стосується

латентного шару, то він являє собою двовимірний плоский масив, що складається з нейронів.

Вхідні нейрони утворюють вхідний шар мережі, якій містить по одному нейрону для кожного вхідного поля. Як і у звичайній мережі вхідні нейрони не беруть участі у процесі навчання. Їх задачею є передача значень вхідних полів початкової вибірки на нейрони вихідного шару. Кожен зв'язок між нейронами має певну вагу, яка в процесі ініціалізації встановлюється випадковим чином в інтервалі [0;1]. Процес навчання полягає у настроюванні ваг. На відміну від більшості нейронних мереж, мережа Кохонена не має прихованих шарів: дані з вхідного шару передаються безпосередньо на вихідний, нейрони якого впорядковані у одномірну або двохмірну решітку прямокутної або шестикутної форми. Значення кожної ознаки надходять через вхідні нейрони на нейрони вихідного шару.

Основні процедури процесу навчання мережі Кохонена

В процесі навчання і функціонування мережа Кохонена (KCN) виконує три процедури:

1. Конкуренція (competition). Вихідні нейрони конкурують між собою за те, щоб вектори їх ваг виявились як можна ближче до вектора ознак об'єкта. Вихідний нейрон, вектор ваг якого має найменшу відстань до вектора ознак об'єкта, оголошується переможцем.

2. Об'єднання (cooperation). Нейрон-переможець стає центром деякої групи сусідніх нейронів. У мережі Кохонена усі нейрони такого сусідства називають нагородженими правом підстроювання ваг. Тобто, не дивлячись на те, що нейрони у вихідному шарі не з'єднуються безпосередньо, вони мають схожі набори ваг завдяки сусідству з нейроном-переможцем.

3. Підстроювання ваг (adaptation). Нейрони, що є сусідами нейрона-переможця, беруть участь у підстроюванні ваг, тобто у навчанні.

Розглянемо набір з m значень полів n-ого запису початкової вибірки, який є вхідним вектором

$$\mathbf{X}_n = (x_{n1}, x_{n2}, \ldots x_{nm})^T$$

і поточний вектор ваг j-го вихідного нейрону

$$\mathbf{W}_j = (w_{1j}, w_{2j}, \ldots w_{mj})^T$$

У навчанні по Кохонену нейрони, які є сусідами нейрону-переможця підстроюють свої ваги, використовуючи комбінацію вхідних векторів і поточних векторів ваг:

$w_{ij}^{\text{нове}} = w_{ij}^{\text{поточне}} + \eta\, x_{ni} - w_{ij}^{\text{поточне}})$, де $0<\eta\leq 1$ - коефіцієнт швидкості навчання.

Швидкість навчання мережі Кохонена має бути функцією від кількості ітерацій, яка зменшується. Тому процес навчання мережі можна розділити на дві фази: грубого та точного підстроювання. При ініціалізації мережі початкові ваги нейронів призначаються випадково, якщо відсутні апріорні знання про характер розподілу у початковій вибірці. Також, при ініціалізації задаються початкова швидкість навчання і радіус навчання R, тобто кількість нейронів, які будуть вважатись сусідами для нейрона-переможця і підстроювати свої ваги разом із ним. Радіус є максимальним на початку навчання і зменшується по мірі навчання.

Алгоритм навчання мережі Кохонена

Алгоритм Кохонена складається з таких кроків.

1. Ініціалізація. Для нейронів мережі встановлюються початкові ваги, а також задаються початкові швидкість навчання η (ета) і радіус навчання R.

2. Збудження. На вхідний шар подається вектор впливу X_n, що містить значення вхідних полів запису навчальної вибірки.

3. Конкуренція. Для кожного вихідного нейрону обчислюється відстань $D(W_j, X_n)$ між векторами ваг усіх нейронів вихідного шару і вектором вхідного впливу. Наприклад, для евклідової відстані отримаємо:

$$\mathbf{D(W_j, X_n)} = \sqrt{\sum_i (w_{ij} - x_{ni})^2}$$

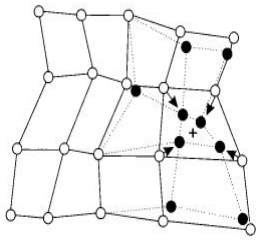
Переможцем стає нейрон j, для якого така відстань виявиться найменшою.

4. Об’єднання. Визначаються усі нейрони, розташовані в межах радіусу навчання відносно нейрона-переможця.

5. Підстроювання. Здійснюється підстроювання ваг нейронів в межах радіуса навчання відповідно до формули

$$w_{ij}^{\text{нове}} = w_{ij}^{\text{поточне}} + (x_{ni} - w_{ij}^{\text{поточне}}).$$

При цьому ваги нейронів - сусідів нейрона-переможця, підстроюються у бік його вектора ваг, як це наведено на рисунку, де координати вхідного вектору помічені знаком «+», а вигляд мережі після модифікації штриховими лініями.

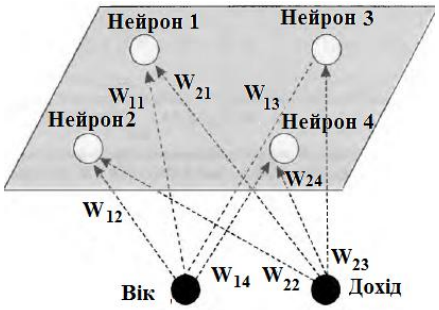


6. Корекція. Змінюється радіус і параметр швидкості навчання відповідно із заданим законом.

Приклад роботи мережі Кохонена

Розглянемо приклад роботи мережі Кохонена, що містить 2 х 2 нейрона у вихідному шарі, а множина даних представлена атрибутами Вік і Дохід з попередньо нормалізованими даними.

У зв’язку з малим розміром мережі встановимо радіус навчання R=0, тобто можливість підстроювати ваги буде надаватися лише нейрону-переможцю. Коефіцієнт швидкості навчання встановимо η=0,5.



Випадковим чином виберемо початкові значення ваг нейронів:

w_{11}	w_{21}	w_{12}	w_{22}	w_{13}	w_{23}	w_{14}	w_{24}
0,9	0,8	0,9	0,2	0,1	0,8	0,1	0,2

Сформуємо набір записів вхідної вибірки:

№	x_{i1}	x_{1j}	Опис
1	x_{11}=0,8	x_{12}=0,8	Літня людина з високим доходом
2	x_{21}=0,8	x_{22}=0,1	Літня людина з низьким доходом
3	x_{31}=0,2	x_{32}=0,8	Молода людина з високим доходом
4	x_{41}=0,1	x_{42}=0,2	Молода людина з низьким доходом

Конкуренція. Обчислимо евклідову відстань між вхідним вектором X_1 і векторами ваг усіх чотирьох нейронів вихідного шару.

Нейрон 1: $D(\mathbf{W}_1, \mathbf{X}_1) = \sqrt{(w_{11} - x_{11})^2 + (w_{21} - x_{12})^2} = \sqrt{(0,9-0,8)^2 + (0,8-0,8)^2} = 0,1.$

Нейрон 2: $D(\mathbf{W}_2, \mathbf{X}_1) = \sqrt{(w_{12} - x_{11})^2 + (w_{22} - x_{12})^2} = \sqrt{(0,9-0,8)^2 + (0,2-0,8)^2} = 0,61.$

Нейрон 3: $D(\mathbf{W}_3, \mathbf{X}_1) = \sqrt{(w_{13} - x_{11})^2 + (w_{23} - x_{12})^2} = \sqrt{(0,1-0,8)^2 + (0,8-0,8)^2} = 0,7.$

Нейрон 4: $D(\mathbf{W}_4, \mathbf{X}_1) = \sqrt{(w_{14} - x_{11})^2 + (w_{24} - x_{12})^2} = \sqrt{(0,1-0,8)^2 + (0,2-0,8)^2} = 0,92.$

Переміг нейрон 1, який формує кластер для захоплення літніх людей з високим доходом

Об'єднання. Оскільки радіус навчання дорівнює нулю, тільки нейрон-переможець буде нагороджений можливістю підстроювання свого вектора ваг.

Підстроювання. Для першого нейрона отримуємо формулу:

для Віку: $w_{11}^{\text{нове}} = w_{11}^{\text{поточне}} + \eta(x_{11} - w_{11}^{\text{поточне}}) = 0,9+0,5x(0,8 -0,9)=0,85.$

для Доходу: $w_{21}^{\text{нове}} = w_{21}^{\text{поточне}} + \eta(x_{12} - w_{21}^{\text{поточне}}) = 0,8+0,5x(0,8 -0,8)=0,8.$

Дане налагоджування дозволить нейрону 1 у подальшому більш успішно захоплювати записи з інформацією про літніх людей з високим доходом.

Початкові значення ваг нейронів:

w_{11}	w_{21}	w_{12}	w_{22}	w_{13}	w_{23}	w_{14}	w_{24}
0,9	0,8	0,9	0,2	0,1	0,8	0,1	0,2

Набір записів вхідної вибірки:

№	x_{i1}	x_{1j}	Опис
1	x_{11}=0,8	x_{12}=0,8	Літня людина з високим доходом
2	x_{21}=0,8	x_{22}=0,1	Літня людина з низьким доходом
3	x_{31}=0,2	x_{32}=0,8	Молода людина з високим доходом
4	x_{41}=0,1	x_{42}=0,2	Молода людина з низьким доходом

Виконавши операції конкуренції та підстроювання для другого вхідного вектору X_2=(0,8;0,1) отримуємо:

$D(W_1, X_2)$	$D(W_2, X_2)$	$D(W_3, X_2)$	$D(W_4, X_2)$	w_{12} нове	w_{22} нове
0,71	0,14	0,99	0,71	0,85	0,15

Переміг нейрон 2. Він відкриває кластер для захоплення літніх людей з малим доходом.

Для третього і четвертого нейронів, відповідно, отримаємо такі нові значення ваг

w_{13} нове	w_{23} нове	w_{14} нове	w_{24} нове
0,15	0,85	0,1	0,15

які будуть відповідати кластерам для молодих людей з високим доходом і молодих людей з низьким доходом.

Таким чином 4 вихідні нейрони представляють 4 різних кластера.

Кількість вихідних нейронів мережі Кохонена має відповідати кількості кластерів, які треба побудувати.

Поняття карти Кохонена

Карти Кохонена або Самоорганізаційні Карти Кохонена (Self-organizing map – SOM) – будуються на основі нейронних мереж Кохонена і призначені для візуалізації багатовимірних об’єктів на двохвимірній карті, де відстані між об’єктами відповідають відстаням між їх векторами у багатовимірному просторі, а самі значення ознак відображаються різними кольорами і відтінками.

Таким чином, можна провести аналогію SOM зі звичайною географічною картою, яка лініями відображає кордони країн (кластерів), а різними кольорами - рельєф поверхні (значення параметрів).

Якщо розмірність простору ознак набора даних дорівнює 2 (наприклад Вік і Дохід), то і вектори ваг нейронів будуть двохкомпонентними і відображення результатів кластерізації на двохвимірній карті не складе проблем. Але вже при 3-х ознаках, такі проблеми виникають.

Здається, відобразити багатовимірний простір на площину можна за допомогою проекцій, але при цьому неминуче порушується топологічна подібність: значно відділені один від іншого об’єкти у початковому просторі ознак, можуть розташуватись поряд один з іншим на карті, як це видно з наведеного рисунку. Інший рисунок показує, що при класифікації двох груп об’єктів, що відрізняються шириною і висотою, при вилученні з розгляду довжини стануть практично нерозрізненими, виключить можливість коректної кластеризації (задача розбиття заданої вибірки об'єктів на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися).

Методика побудови карти Кохонена

Карта Кохонена є засобом візуалізації і є окремим випадком мережі Кохонена, у якій кількість нейронів набагато перевищує кількість кластерів. SOM складається з комірок прямокутної або, частіше, шестикутної форми, де відстані між центрами суміжних комірок є однаковими.

Кожній комірці відповідає нейрон мережі Кохонена. Навчання нейронів карти здійснюється так само, як і навчання нейронів мережі Кохонена. Об'єкти, вектори ознак яких близькі між собою, потрапляють в одну комірку карти або в комірки, розташовані поруч.

Хоча відстань між об’єктами надає уяву про ступень їх подібності або відмінності, але важливою є інформація і про те, у чому саме проявляється ця подібність або відмінність, за якими ознаками об’єкти розрізняються більше, а за якими менше і т. ін. відповідь на ці питання і надає спеціальна розмальовка, яка виконує функцію третього виміру.

Але на одній карті можна використати розмальовку лише за однією ознакою. Отож, для візуалізації кількох ознак треба будувати окремі карти.

Наприклад, для об’єктів з двома ознаками (Вік і Дохід), для отримання інформації щодо значень цих ознак, треба будувати дві карти.

Отримавши усю сукупність карт для кожної з ознак, можна наочно порівняти значення кожної ознаки даного кластера зі значеннями кожної з ознак інших кластерів. У кожну комірку в загальному випадку потрапляє кілька об'єктів. Тому обчислюється або середнє значення параметра об'єктів кожної комірки або мінімальне чи максимальне значення. Якщо в комірку не потрапив жоден об'єкт (комірці відповідає мертвий нейрон), то за значення комірки береться вага нейрона, відповідного даному параметру.

Види карт Кохонена

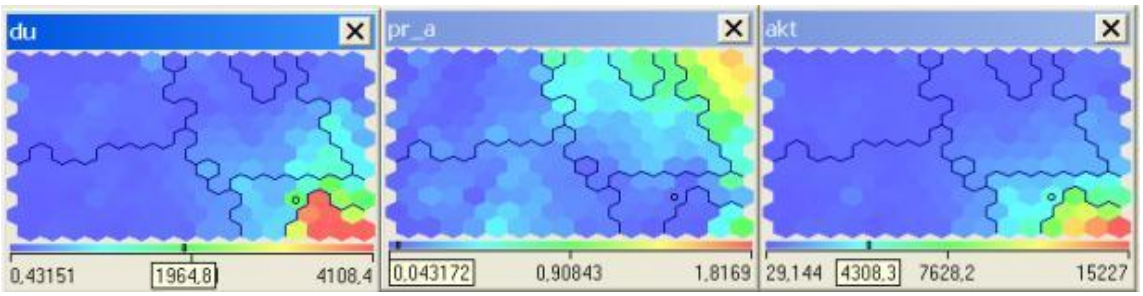
В результаті роботи алгоритму самоорганізації отримуємо такі карти:

Карта входів нейронів:

ваги нейронів підлаштовуються під значення вхідних змінних і відображають їх внутрішню структуру. Для кожного входу формується своя карта, розфарбована у відповідності зі значенням конкретної ваги нейрона. Давайте розглянемо фрагмент карти, що складається з карт трьох входів:

- при аналізі даних використовують кілька карт входів;

- на одній з карт виділяють область певного кольору - це означає, що відповідні вхідні приклади мають приблизно однакове значення відповідного входу. Кольоровий розподіл нейронів з цієї області аналізується на інших картах для визначення схожих або відмінних характеристик.



Виділимо на першій карті об'єкти, розташовані в правому нижньому кутку, які мають найбільші значення. Можна помітити, що ці ж об'єкти мають найбільші значення показника, зображеного на третій карті. Також можна зробити висновок, що існує взаємозв'язок між цими показниками. Можна також визначити, наприклад, таку характеристику: кластер у правому верхньому кутку, характеризується низькими значеннями показників du (депозити) та akt (активи) і високими значеннями показників pr_a (прибутковість активів). Ця інформація дозволяє охарактеризувати кластер, що знаходиться у правому верхньому кутку, як банки з невеликими активами, невеликими залученими депозитними коштами, але з найбільш прибутковими активами, тобто це група невеликих, але найбільш прибуткових банків.

Карта виходів нейронів:

відображає взаємне розташування досліджуваних вхідних даних.

Нейрони з однаковими значеннями виходів утворюють кластери - замкнуті області на карті.

Спеціальні карти:

карта кластерів, матриця відстаней, матриця щільності попадання і інші карти, які характеризують кластери, отримані в результаті навчання мережі Кохонена.

Переваги і недоліки карт Кохонена

Двома **найбільш поширеними застосуваннями** карт Кохонена є розвідувальний аналіз даних і виявлення нових явищ.

Розвідувальний аналіз даних. Мережа Кохонена здатна розпізнавати кластери в даних, а також встановлювати близькість класів. Таким чином, користувач може поліпшити своє розуміння структури даних, щоб потім уточнити нейромережеву модель. Якщо в даних розпізнані класи, то їх можна позначити, після чого мережа зможе вирішувати завдання класифікації.

Мережі Кохонена можна використовувати і в тих задачах класифікації, де класи вже задані, - тоді перевага буде в тому, що мережа зможе виявити подібність між різними класами.

Виявлення нових явищ. Мережа Кохонена розпізнає кластери в навчальних даних і відносить всі дані до тих чи інших кластерів. Якщо після цього мережа зустрінеться з набором даних, несхожим на жодний з відомих зразків, то вона не зможе класифікувати такий набір і тим самим виявить його новизну. Принциповою відмінністю карт Кохонена від інших моделей нейронних мереж є інший підхід до навчання, а саме - некероване або неконтрольоване навчання. Цей тип навчання дозволяє даним навчальної вибірки містити значення тільки вхідних змінних. Мережа Кохонена вчиться розуміти саму структуру даних і вирішує завдання кластеризації.

Але у карт Кохонена є ряд **недоліків** і обмежень, що примушують у деяких випадках обережно ставитись до самого процесу побудови карт, і до інтерпретації його результатів.

Необхідність перевірки гіпотез: однією з найбільш серйозних претензій є те, що самі по собі карти задачу кластеризації не вирішують, а лише дозволяють висунути гіпотези про наявність кластерної структури і залежностей в наборі даних. Висунуті гіпотези потрібно підтверджувати іншими методами, тому що вони можуть виявитися помилковими.

Евристичний характер методу: недоліком є і евристичний характер методу. Якщо навчати карту кілька разів, завжди будуть отримані несхожі результати, оскільки встановлювані випадковим чином початкові ваги нейронів будуть різні і навчання кожен раз буде йти трохи інакше.

Проблема "мертвих" нейроні: крім того, може виникнути така проблема, як «мертві» нейрони - нейрони, ваги яких були проініціалізовані таким чином, що їх вектори опинилися в тій частині простору ознак, де відсутні або майже відсутні вектори об'єктів. При цьому обробку вхідних даних буде здійснювати надто мала кількість нейронів, яких може не вистачити для якісної кластеризації

Загальні проблеми кластерізації (кластерний аналіз — задача розбиття заданої вибірки об'єктів на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися).

Єдиного універсального алгоритму кластерізації не існує і вона буде ефективною лише за умов чіткого розуміння переваг, недоліків і обмежень вибраного алгоритму

Причини відсутності універсального алгоритму:

1. **Невизначеність у виборі критерію якості кластеризації** - обумовлюється складністю оцінки взаємного розташування об'єктів, які описуються десятками і сотнями властивостей. Наприклад, k-means (Кластериза́ція ме́тодом k-сере́дніх — популярний метод кластеризації, — впорядкування множини об'єктів у порівняно однорідні групи) хороші працює з даними, що утворюють компактні згустки, оскільки заснований на обчисленні відстаней до центрів кластера, але не може розділити вкладені множини об'єктів.

2. **Труднощі вибору міри близькості** - різни типи даних вимагають комбінації метрик, що погіршує роботу алгоритму, оскільки ефективні алгоритм для змішаних наборів даних мало.

3. **Різні необхідні машинні ресурси** - що обумовлює вибір різних алгоритмів для задач різного масштабу. У масштабованих алгоритмах відмовляються від локальної функції оптимізації, як парне порівняння об'єктів у k-means, яке потребує великих обчислювальних витрат. Мінімізація обчислень вимагає глобальної функції оптимізації, яка суттєво відрізняється для різних задач.

4. **Вибір кількості кластерів** - рідко зустрічаються задачі з точно заданою кількістю кластерів. Якщо алгоритм не підтримує автоматичне визначення оптимальної кількості кластерів, то використовують емпірічні правила, які пропонують використовувати від 4 до 9 кластерів. Величезна

кількість алгоритмів кластеризації виключає її об'єктивний характер. Будь-яка кластеризація суб'єктивна, тому що виконується на основі кінцевого підмножини властивостей об'єктів. Вибір цієї підмножини є суб'єктивним, як і вибір критерію якості та міри близькості.

Для коректного застосування кластеризації необхідно дотримуватися наступних правил.

- Правило 1. Перед кластеризацією чітко окреслити цілі її проведення: спрощення подальшого аналізу, стиснення даних і т.п. Кластеризація сама по собі не має особливої цінності.

- Правило 2. Переконатися, що обраний алгоритм коректно працює з наявними даними. Якщо алгоритм не вміє працювати зі змішаними наборами даних треба зробити набір однорідним, відмовившись від категоріальних або числових ознак.

- Правило 3. Обов'язково здійснити змістовну інтерпретацію кожного отриманого кластера: намагатися зрозуміти, чому об'єкти були згруповані у певний кластер, що їх об'єднує. Для цього можна використовувати візуальний аналіз, графіки, кластерограми, статистичні характеристики кластерів, карти. Корисно кожному кластеру дати осмислену назву з декількох слів.

Перейдемо до програмної реалізації карти Кохонена

Реалізація самоорганізованої карти в Python за допомогою NumPy.

Оскільки у стандартній бібліотеці машинного навчання Scikit-Learn немає вбудованої процедури для SOM, ми виконаємо швидку реалізацію вручну за допомогою NumPy. Модель неконтрольованого машинного навчання досить проста і проста у реалізації.

Ми спершу реалізуємо SOM як двовимірну сітку mxn, отже, потрібен 3D-масив NumPy. Третій вимір потрібен для збереження вагових коефіцієнтів у кожній комірці.

Запуск самоорганізуючої карти на практичному прикладі

Одним із часто цитованих прикладів навчання SOM є випадкові кольори. Ми можемо навчити сітку SOM і легко візуалізувати, як різні подібні кольори розташовуються в сусідніх клітинках.

Клітини, розташовані далеко одна від одної, мають різні кольори.

Давайте запустимо функцію train_SOM() на навчальній матриці даних, заповненій випадковими кольорами RGB.

Наведений нижче код ініціалізує навчальну матрицю даних і сітку SOM із випадковими кольорами RGB. Він також відображає дані навчання та випадково ініціалізовану сітку SOM. Зауважте, що навчальна матриця має розмір 3000x3, однак для візуалізації ми змінили її на матрицю 50x60x3:

`find_BMU()` returns the grid cell coordinates of the best matching unit when given the `SOM` grid and a training example `x`. It computes the square of the Euclidean distance between each cell weight and `x`, and returns `(g,h)`, i.e., the cell coordinates with the minimum distance.

find_BMU() повертає координати комірки сітки найкращої відповідної одиниці, якщо задано сітку SOM і навчальний приклад x. Він обчислює квадрат евклідової відстані між вагою кожної клітинки та x і повертає (g,h), тобто координати клітинки з мінімальною відстанню.

```
# Обчислює квадрат евклідової відстані між вагою кожної клітинки та x
# Повертає (g,h) - координати клітинки з мінімальною відстанню
```

The `update_weights()` function requires an SOM grid, a training example `x`, the parameters `learn_rate` and `radius_sq`, the coordinates of the best matching unit, and a `step` parameter. Theoretically, all cells of the SOM are updated on the next training example. However, we showed earlier that the change is negligible for cells that are far away from the BMU. Hence, we can make the code more efficient by changing only the cells in a small vicinity of the BMU.
The `step` parameter specifies the maximum number of cells on the left, right, above, and below to change when updating the weights.

```
BMU (Best Matching Unit)  - Найкраща відповідна одиниця
```

Для функції update_weights() потрібна сітка SOM, навчальний приклад x, параметри learn_rate і radius_sq, координати найкращої відповідної одиниці та параметр кроку. Теоретично всі комірки SOM оновлюються на наступному навчальному прикладі. Однак раніше ми показали, що зміна незначна для клітин, які знаходяться далеко від Найкращої відповідної одиниці. Таким чином, ми можемо зробити код більш

ефективним, змінивши лише клітинки в невеликій близькості від Найкращої відповідної одиниці. Параметр step указує максимальну кількість комірок ліворуч, праворуч, зверху та знизу, яку потрібно змінити під час оновлення вагових коефіцієнтів.

```
# Оновіть ваги комірок SOM, якщо надано один навчальний приклад
# і параметри моделі разом із координатами найкращої відповідної одиниці як
кортеж
```

Finally, the `train_SOM()` function implements the main training procedure of an SOM. It requires an initialized or partially trained `SOM` grid and `train_data` as parameters. The advantage is to be able to train the SOM from a previous trained stage. Additionally `learn_rate` and `radius_sq` parameters are required along with their corresponding decay rates `lr_decay` and `radius_decay`. The `epochs` parameter is set to 10 by default but can be changed if needed.

Нарешті, функція train_SOM() реалізує основну процедуру навчання SOM. Для цього потрібна ініціалізована або частково навчена сітка SOM і train_data як параметри. Перевагою є можливість тренувати SOM з попереднього етапу навчання. Крім того, потрібні параметри learn_rate та radius_sq разом з їхніми відповідними темпами затухання lr_decay та radius_decay. Параметр епох за замовчуванням має значення 10, але його можна змінити за потреби.

Давайте тепер навчимо SOM і перевіримо його кожні 5 епох, щоб отримати короткий огляд його прогресу:

Наведений вище приклад дуже цікавий, оскільки він показує, як сітка автоматично впорядковує кольори RGB так, щоб різні відтінки одного кольору були поруч у сітці SOM. Розташування має місце ще в першій епосі, але воно не ідеальне. Ми бачимо, що SOM збігається приблизно в 10 епохах, а в наступних епохах відбувається менше змін.

Вплив швидкості навчання та радіуса

Щоб побачити, як швидкість навчання змінюється для різних швидкостей навчання та радіусів, ми можемо запустити SOM для 10 епох, починаючи з тієї самої початкової сітки. Наведений нижче код тренує SOM для трьох різних значень швидкості навчання та трьох різних радіусів.

SOM рендериться (візуалізується) після 5 епох для кожної симуляції.

Наведений приклад показує, що для значень радіуса, близьких до нуля (перший стовпець), SOM змінює лише окремі клітинки, але не сусідні клітинки. Отже, належна карта не створюється незалежно від швидкості навчання. Подібний випадок також зустрічається для менших темпів навчання (перший рядок, другий стовпець). Як і в будь-якому іншому алгоритмі машинного навчання, для ідеального навчання потрібен хороший баланс параметрів.

Висновок: хоча SOM більше не дуже популярні в спільноті машинного навчання, вони залишаються хорошою моделлю для зведення та візуалізації даних.