Задача кластеризації

це якщо я хочу класифікувати набір даних, але не знаю за якими ознаками, просто знаю на скільки груп(кластерів) маю поділити дані, але не знаю які в цих групах будуть ознаки.

Самоорганізаційні мережі вирішують задачі кластеризації (мережі, які навчаються “без вчителя”): **прошарок Кохонена**(**newc**) та **карта Кохонена** (**newsom**).

Різниця між картою та прошарком:

- *кардинально відрізняються* тим, що в прошарку немає топології, а в карті є топологія;

Топологія означає, що нейрони початково не лежать ізольовано (розкидано), а зв’язані між собою, тобто мають зв’язки, так звані сітки.

- в прошарку є зміщення, в карті - немає, відповідно, по різному боряться з так званими “мертвими” нейронами;
- мережі з невпорядкованими нейронами - прошарки Кохонена та мережі з упорядкованими нейронами - карти Кохонена. Останні відображають структуру даних таким чином, що близьким кластерам даних на карті відповідають близько розташовані нейрони.

Прошарок Кохонена (newc)

Самоорганізаційна нейронна мережа з єдиним прошарком, завдання якої полягає в тому, щоб правильно згрупувати (кластеризувати) вектори входу, що надходять на неї.

Архітектура

Архітектура схожа на радіально базисну нейронну мережу, тільки інша активаційна функція, так звана конкуруюча. Є лише один прошарок - **конкуруючий прошарок**. (*кількість нейронів у прошарку співпадає з кількістю кластерів, на які хочу поділити свій набір даних*, тобто, наприклад, хочу 8 кластерів, має бути 8 нейронів на конкуруючому прошарку).

Що відбувається на цьому прошарку: допустимо поступив вхідний вектор, обчилюється від’ємна евклідова відстань між цим вхідним вектором та вектором ваг кожного з нейронів конкуруючого прихованого прошарку (тут від’ємна евклідова відстань в класичному сенсі, тобто перед різницею стоїть мінус). Тобто, знайшла від’ємну евклідову відстань вхідного вектора з першим нейроном, з другим нейроном і тд. Оскільки евклідова відстань від’ємна - це означає, що там де класична сама відстань була найменша, якщо я перед тим поставлю мінус, то серед тих всіх чисел, які я зараз отримала як від’ємні евклідові відстані від вхідного вектора до різних нейронів найбільше число - це і буде то число, яке характеризує найближчий нейрон до мого вхідного вектора.

Тобто, допустимо, якийсь нейрон був найближчим до мого вхідного вектора, тоді класична евклідова відстань є найменше по модулю число для

того нейрона і якщо я ставлю перед цим мінус і перед всіма іншими відстанями поставила мінус, то в результаті порівняння, таке від'ємне число, яке по модулю насправді найменше, в реальності буде найбільшим.

Принцип роботи конкуруючої функції:

вибирає якраз серед отих відстаней, скорегованих на зміщення, ту яка найбільша. Тобто, на неї поступило кілька чисел (стільки, скільки в мене є нейронів), відстаней від вхідного вектора до кожного з нейронів і додане зміщення (b). Де той набір чисел виявився найбільшим, того нейрона і оголошують переможцем і того нейрона оголошуть відповідним моєму вхідному вектору.

Нейрон-переможець

Нейрон-переможець в майбутньому корегує свої ваги, тотбто його ваги змінюють. Всіх решта нейронів, які програли, ваги не змінюватимуться.

РИС. Арх. Мер. Прошарок:

Тобто, поступає вхідний вектор ваг на конкоруючий прошарок, обчислюється від'ємна евклідова відстань до кожного з нейронів, додається зміщення, якщо воно є і далі конкуруюча функція з тих всіх чисел обирає найбільше і каже, що цей нейрон переміг, бо насправді найбільше число буде там, де був найближчий до вхідного вектора нейрон. Цього нейрона вона оголошує переможцем, цього нейрона ваги змінюються за правилом Кохонена

Правило навчання прошароку Кохонена

Правило навчання прошароку Кохонена (або *правило Кохонена*) полягає у тому, щоб налаштувати належним чином елементи матриці ваг. Припустимо, що нейрон ***i*** переміг при подачі входу $p(q)$ на кроці самонавчання q, тоді рядок ***i*** матриці ваг коригується відповідно до правила Кохонена наступним чином:

$$_i\mathbf{IW}_{11}(q) = {}_i\mathbf{IW}_{11}(q-1) + \alpha\left(\mathbf{p}(q) - {}_i\mathbf{IW}_{11}(q-1)\right).$$

(До старих ваг додається різниця між вхідним вектором і старими вагами помножена на певний коефіцієнт, який відповідає кроку навчання)

"Мертві" нейрони

Це ті, які знаходяться так далеко, що скільки б не навчалося, вони всеодно не будуть задіяні. Щоб їх використати використовують зсуви (зміщення), які дозволяють нейрону стати конкурентним з нейронами-переможцями. Що роблять зміщення? Після того, як обчислили евклідові відстані від вхідного вектора до кожного нейрона система ніби бачить, порівнює що якийсь нейрон (нейрони) у поганій ситуації, вони від вхідного вектора знаходяться дуже далеко, це означає, що вони взагалі в кінцевому результаті можуть виявитися дуже далеко від всіх. Система розуміє, що від першого, допустимо, вхдіного вектора, цей "мертвий" прям дуже далеко. Тоді система додає до тих нейронів, які далеко знаходяться від вхідного вектора, зміщення. Це зміщення насправді не пересуває самого нейрона в просторі, бо пересуватися нейрон може лише тоді, коли його ваги змінюютьтся. Система ніби уявно його підсуває в інше місце, тобто нейрон обманює вхідний вектор за

рахунок зміщень і каже, що він не аж так далеко, а десь ближче і кожен так нейрон, який далеко від конкретного вхідного вектора так починає за допогою системи обманювати вхідні вектори і відповідно конкуруюча функція вже працює з зовсім іншими числами і такі нейрони мають шанс виграти конкуренцію і оголоситись переможцями.

Learncon якраз виконує цю функцію.

Збільшення зсувів для неактивних нейронів дозволяє розширити діапазон покриття вхідних значень і неактивний нейрон починає формувати кластер. Зрештою, він може почати притягувати нові вхідні вектори.

Це дає дві ***переваги***:

Якщо нейрон не виграє конкуренції, тому що його вектор ваг істотно відрізняється від векторів, що надходять на вхід мережі, його зміщення в міру навчання стає досить великим і він стає конкурентоспроможним. Коли це відбувається, його вектор ваги починає наближатися до певної групи векторів входу. Як тільки нейрон починає перемагати, його зміщення починає зменшуватися. Таким чином, завдання активізації "мертвих" нейронів виявляється вирішеним.

Друга перевага, пов'язана з налаштуванням зсувів, полягає в тому, що вони дозволяють вирівняти значення параметра активності та забезпечити "притягування" приблизно однакової кількості векторів входу. Таким чином, якщо один із кластерів притягує більшу кількість векторів входу, ніж інший, то біль заповнена область притягне додаткову кількість нейронів і будеподілена на менші по розміру кластери.

Навчання

Є нейрони на графіку і по одному вектору входить, нейрон який є найближчим і є переможцем, який зміщується до вектора ближче, коли всі вектори зайдуть і нейрон буде рівно віддалений від векторів свого околу і буде кінець.

У процесі навчання кожен нейрон у прошарку, ваговий вектор якого близький до групи векторів входу стає визначальним для цієї групи векторів. В кінцевому рахунку, якщо є достатня кількість нейронів, кожна група векторів входу матиме нейрон, який виводить 1, коли представлений вектор цієї групи, і 0 в іншому випадку, або, іншими словами, формується кластер.

Карта Кохонена (newsom)

Мережа, що самоорганізовується, у вигляді карти Кохонена призначена для вирішення завдань кластеризації вхідних векторів.

Відрізняють від прошарку тим, що мають топологія, що означає, що нейрони початково не лежать ізольовано (розкидано), а вони лежать всі зв’язані між собою, тобто вони всі зі зв’язками, є так звані сітки і ті сітки можуть бути різних видів і кажуть тоді, що карта Кохонена має різну топологію. Ми

розглядаємо прямокутну сітку(**gridtop**), гексагональну сітку(**hextop**) і рандомну сітку(**randtop**).

Прямокутна сітка **gridtop** - це класична, така як паркан.

Гексагональна сітка **hextop** - це коли з трикутничків складається сітка. (Зауважимо, що М-функція hextop використовується ***за замовчуванням*** при створенні карт Кохонена при застосуванні функції newsom)

Рандомна сітка **randtop** - це коли самі вузли сітки розкидані як-небудь, але зв'язки між ними є.

Оскільки є сітки і є зв'язки між нейронами, то в нас є таке поняття як відстань між нейронами. Вона важлива через те, що оці зв'язки, які між ними є, захочуть під час навчання не лише нейрона-переможця змінювати, а й тягнути в сторону вхідного вектора інших нейронів, які причеплені зв'язками до мого нейрона-переможця. Щоб зрозуміти скількох інших нейронів тягнути нам це допомагають околи, які утворюються маючи радіус, який дорівнює певній відстані, яка береться з метрики, які можуть бути різні, тому в нас не лише є різні топології, а в нас і по-різному можуть утворюватися околи.

Фунції для розрахування відстаней

У ППП NNT використовується 4 функції розрахунку відстаней між вузлами сітки:

- ***dist*** - класична евклідова відстань, яка дає класичні околи, до яких ми звикли кружечками

$$d = \mathrm{sqrt}(\mathrm{sum}((\mathbf{pos}_i - \mathbf{pos}_j).^2)),$$

де posi, posj - вектори координат нейронів з номерами i і j.

- ***boxdist*** - дає околи квадратиками
- ***linkdist*** - зв'язкова відстань, тобто рахується кількість паличок між моїми вузлами решітки
- ***mandist*** - манхетенська відстань (приблизно ромбіками)

Карти не використовуються зміщення (баєси).

Радіуси і околи існую для мертвих нейронів. В картах будуть змінюватись ваги не лише нейрона-переможця, але і його сусідів в певному околі (радіусі).

Підтягування відбувається обернено пропорційно до відстані від нейрона до вхідного вектор.

Починають навчання таким чином, що радіус беруть максимально великий, так, щоб взагалі охопити всю мою решітку. В процесі навчання радіус ми зменшуємо. В кінцевому результаті радіус доходить до 0 - це означає, що лише один в кінцевому результаті нейрон буде підтягуватися до одного навчального прикладу. Іноді не доходить до 0, якщо забагато вузлів і тоді будуть кілька підтягуватися. Таким чином ми боремося з проблемою "мертвих" нейронів.

Те, що є зв’язки дає нам те, що сусіди теж підтягуються в певному радіусі (ніби ці зв’язки тягнуть їх). Тягнуть не завжди - коли ми дозволяємо, точніше наш радіус. Тмм це відрізняється від прошарку, в якому зв’язків немає і ніби той канат не тягне того сусіда, інші нейрони не тягнуться, якщо є нейрон-переможець, а тут воно все ніби зв’язане мотузками і воно тягне сусідів, якщо ми дозволяємо і дозволяє радіус, звісно.

По суті **навчання** складається з двох етапів: відбувається перебудова (градки) і дрібне підлаштування, коли вже ідеально кластеризується так як вже більш подібно до прошарку.

В картах вже не завжди співпадає кількість кластерів і кількість вузлів решітки, буває таке, що розмірність решітки більша в реальності за кількість кластерів.

Архітектура

Фактично така сама як в прошарку Кохонена. Теж є в прихованому конкуруючому прошарку певна кількість нейронів, тільки те, що тут вони поєднані зв’язками.

Не використовуються зміщення (баєси).

Конкуруюча функція активації повертає 1 для елемента виходу a^1, відповідного нейрону, що переміг; усі інші елементи вектора a^1 дорівнюють 0. Однак у мережі Кохонена виконується перерозподіл нейронів, що є сусідніми з нейроном, що переміг. При цьому можна вибирати різні топології розміщення нейронів та різні метрики для обчислення відстаней між нейронами.

Карта Кохонена для визначення нейрона-переможця використовує ту ж процедуру, яка застосовується і у прошарку Кохонена. Однак на карті Кохонена одночасно змінюються вагові коефіцієнти сусідніх нейронів відповідно до наступного співвідношення, яке називається правилом Кохонена:

$$_i\mathbf{w}(q) = (1-\alpha)\,_i\mathbf{w}(q-1) + \alpha\mathbf{p}(q).$$

Одновимірна карта Кохонена

Коли все відбувається на прямій. Тут ми можемо зустрітися з такою ситуацією, що крайніх точок буде більше елементів в кластері, ніж в інших, бо ті, що в середині мають вибір чи їм піти на право, чи на ліво, а крайні вибору не мають, їм всеодно тільки або вправо, або вліву, залежно з якого боку розташовані і тому так виходить, що крайні мають зазвичай трошки більше елементів в кластерах. (приклад б з коду)

Двовимірна карта Кохонена

Багато точок, кластеризується решітка і в процесі навчання по різному жмякається, тягнеться і тд (вона дуже еластична).