Санкт-Петербургский Государственный Университет
Факультет прикладной математики – процессов управления

Работа по курсу
«Методы и средства научной визуализации»
на тему 2.4 «Самоорганизующаяся карта Кохонена»

Работу выполнял
студент 331 группы
Козырев Сергей Александрович

29 апреля 2020 г.

Содержание

1 Введение

При анализе и прогнозировании социально-экономических явлений исследователь довольно часто сталкивается с многомерностью их описания. Это происходит при решении задачи сегментирования рынка, построении типологии стран по достаточно большому числу показателей, прогнозирования конъюнктуры рынка отдельных товаров, изучении и прогнозировании экономической депрессии и многих других проблем.[1] Всё это можно отнести к задачам кластеризации.

Кластеризация (англ. cluster analysis) — задача группировки множества объектов на подмножества (кластеры) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию. Задача кластеризации относится к классу задач обучения без учителя.[2]

Задача кластеризации данных (также называемая таксономией, автоматической классификацией или группировкой объектов) является одной из наиболее важных и сложных задач анализа данных. Кластерный анализ представляет собой раздел статистического анализа данных, объединяющий методы разбиения (группировки) множества наблюдаемых объектов на сравнительно однородные группы, называемые кластерами. Однородность кластеров означает, что объекты, отнесенные к одному кластеру, должны быть близки относительно выбранной метрики. Объекты из разных кластеров должны существенно отличаться. Кластерный анализ является востребованной и успешно развивающейся дисциплиной современной теоретической информатики. Его методы имеют широкий спектр применений практически во всех областях человеческой деятельности, связанных с изучением объектов и процессов: медицине, биологии, химии, маркетингу, психологии, социологии, менеджменту, филологии, археологии и другим.[3]

Именно задачу кластеризации решает самоорганизующаяся карта Кохонена, которой посвящена данная работа.

1.1 История

Эту нейронную сеть предложил финский профессор Теуво Кохонен в 1980-х годах. Самоорганизующаяся карта Кохонена (SOM) выросла из ранних моделей нейронных сетей, особенно моделей ассоциативной памяти и адаптивного обучения (ср. Кохонен 1984). Новый стимул состоял в том, чтобы объяснить пространственную организацию функций мозга, особенно наблюдаемую в коре больших полушарий. Тем не менее, SOM не была первым шагом в этом направлении: следует упомянуть, по крайней мере, пространственно упорядоченные линейные детекторы фон дер Мальсбурга (1973) и модель нейронного поля Амари (1980).

Первой областью применения SOM было распознавание речи (см. рис. 1). В своей абстрактной форме SOM получила широкое распространение в области анализа и исследования данных (Kaski et al. 1998, Oja et al. 2003, Pöllä et al. 2007).[4]

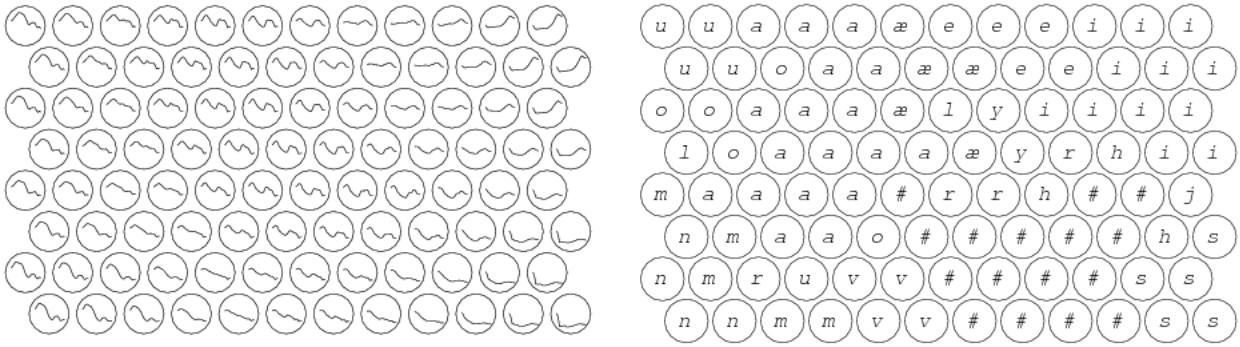

Рис. 1: Левое изображение: модели акустических спектров финских фонем, организованных на некотором расстоянии. Правое изображение: калибровка моделей по фонемным классам. Символ # обозначает взрывные согласные /k, p, t/.

1.2 Задачи, решаемые при помощи карт Кохонена

Самоорганизующиеся карты могут использоваться для решения таких задач, как моделирование, прогнозирование, поиск закономерностей в больших массивах данных, выявление наборов независимых признаков и сжатие информации.

Наиболее распространенное применение сетей Кохонена - решение задачи классификации без учителя, т.е. кластеризации.

Напомним, что при такой постановке задачи нам дан набор объектов, каждому из которых сопоставлена строка таблицы (вектор значений признаков). Требуется разбить исходное множество на классы, т.е. для каждого объекта найти класс, к которому он принадлежит.

В результате получения новой информации о классах возможна коррекция существующих правил классификации объектов.

Вот два из распространенных применений карт Кохонена: разведочный анализ данных и обнаружение новых явлений.

Разведочный анализ данных. Сеть Кохонена способна распознавать кластеры в данных, а также устанавливать близость классов. Таким образом, пользователь может улучшить свое понимание структуры данных, чтобы затем уточнить нейросетевую модель. Если в данных распознаны классы, то их можно обозначить, после чего сеть сможет решать задачи классификации. Сети Кохонена можно использовать и в тех задачах классификации, где классы уже заданы, - тогда преимущество будет в том, что сеть сможет выявить сходство между различными классами.

Обнаружение новых явлений. Сеть Кохонена распознает кластеры в обучающих данных и относит все данные к тем или иным кластерам. Если после этого сеть встретится с набором данных, непохожим ни на один из известных образцов, то она не сможет классифицировать такой набор и тем самым выявит его новизну.[10]

2 Нейронные сети Кохонена

Прежде чем мы приступим к самоорганизующимся картам, стоит разобрать более простую сеть - нейронную сеть Кохонена.

Нейронные сети Кохонена — более широкий класс нейронных сетей, основным элементом которых является слой Кохонена. Слой Кохонена состоит из адаптивных линейных сумматоров («линейных формальных нейронов»). Как правило, выходные сигналы слоя Кохонена обрабатываются по правилу «Победитель получает всё»: наибольший сигнал превращается в единичный, остальные обращаются в ноль. Структуру сети Кохонена вы можете видеть на Рис. 2

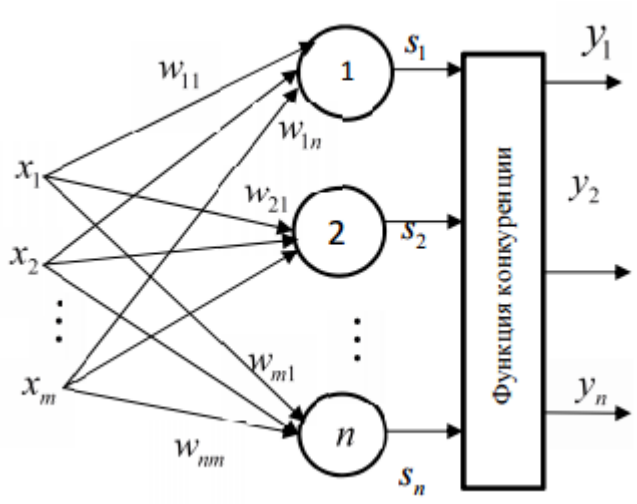

Рис. 2: Структура сети Кохонена

2.1 Слой Кохонена

Слой Кохонена состоит из некоторого количества n параллельно действующих линейных элементов. Все они имеют одинаковое число входов m и получают на свои входы один и тот же вектор входных сигналов $x = (x_1, ...x_m)$. На выходе j-го линейного элемента получаем сигнал

$$y_j = w_{j0} + \sum_{i=1}^{m} w_{ji} x_i,$$

где:

- w_{ji} — весовой коэффициент i-го входа j-го нейрона;
- i — номер входа;
- j — номер нейрона;
- w_{j0} — пороговый коэффициент.

После прохождения слоя линейных элементов сигналы посылаются на обработку по правилу «победитель забирает всё»: среди выходных сигналов выполняется поиск максимального y_j; его номер $j_{\max} = \arg\max_j\{y_j\}$. Окончательно, на выходе сигнал с номером $j_{\max}$ равен единице, остальные — нулю. Если максимум одновременно достигается для нескольких $j_{\max}$, то:

- либо принимают все соответствующие сигналы равными единице;
- либо равным единице принимают только первый сигнал в списке (по соглашению).

2.2 Геометрическая интерпретация

Большое распространение получили слои Кохонена, построенные следующим образом: каждому (j-му) нейрону сопоставляется точка $W_j = (w_{j1}, ...w_{jm})$ в m-мерном пространстве (пространстве сигналов). Для входного вектора $x = (x_1, ...x_m)$ вычисляются его евклидовы расстояния $\rho_j(x)$ до точек W_j и «ближайший получает всё» — тот нейрон, для которого это расстояние минимально, выдаёт единицу, остальные — нули. Следует заметить, что для сравнения расстояний достаточно вычислять линейную функцию сигнала:

$$\rho_j(x)^2 = \|x - W_j\|^2 = \|W_j\|^2 - 2\sum_{i=1}^{m} w_{ji}x_i + \|x\|^2$$

(здесь $\|y\|$ — Евклидова длина вектора: $\|y\|^2 = \sum_i y_i^2$). Последнее слагаемое $\|x\|^2$ одинаково для всех нейронов, поэтому для нахождения ближайшей точки оно не нужно. Задача сводится к поиску номера наибольшего из значений линейных функций:

$$j_{\max} = \arg\max_j \left\{ \sum_{i=1}^{m} w_{ji}x_i - \frac{1}{2}\|W_j\|^2 \right\}$$

Таким образом, координаты точки $W_j = (w_{j1}, ...w_{jm})$ совпадают с весами линейного нейрона слоя Кохонена (при этом значение порогового коэффициента $w_{j0} = -\|W_j\|^2/2$.

Если заданы точки $W_j = (w_{j1}, ...w_{jm})$, то m-мерное пространство разбивается на соответствующие многогранники Вороного-Дирихле V_j: многогранник V_j состоит из точек, которые ближе к W_j, чем к другим W_k ($k \neq j$). [5]

2.3 Обучение сети Кохонена

Сейчас мы рассмотрим, как при помощи сети Кохонена можно решить задачу классификации. Задача классификации отличается от задачи кластеризации тем, что нам заранее задано количество классов, на которые требуется разбить исходные данные.

Пусть у нас есть набор из M исходных образов, который нужно классифицировать, т.е. разбить на K классов. Каждый образ будем описывать некоторым вектором

$$x^m = [x_1^m, x_2^m, ..., x_N^m],$$

где x_i^m суть действительные числа.

Обучать мы будем весовые коэффициенты

$$W^k = [w_1^k, w_2^k, ..., w_N^k],$$

$k = 1, 2, ..., K$, где K есть количество классов на которые мы разбиваем исходные образы. Заметим, что каждый вектор весов имеет такую же размерность, как и входные вектора.

Также скажем, что количество шагов обучения заранее определено, пусть это будет T.

Итак, алгоритм обучения сети:

1. Нормировка исходных векторов.
   - Вычислим $Max_n = \max_m x_n^m$, $Min_n = \min_m x_n^m$.
   - Введем обозначения $a_n = \dfrac{1}{Max_n - Min_n}$, $b_n = \dfrac{-Min_n}{Max_n - Min_n}$.
   - Нормируем вектора $x_n^m = a_n x_n^m + b_n,\ n = 1, ..., N$.
2. Инициализируем веса случайным образом значениями от 0 до 1.
3. Инициализируем $t = 0$.
4. Для каждого X_m:
   ищем ближайший вектор W_k и для найденного вектора W_k корректируем компоненты:
   $$w_n^k = w_n^k + a(t)(x_n^m - w_n^k),$$
   где $a(t) > 0$ - шаг обучения, убывающая функция.
5. Если $t < T$, увеличиваем t на 1, выполняем шаг 4

Теперь наша сеть обучена, и мы можем приступать её к использованию.

Теперь любой вектор X нужно нормировать и найти ближайший вектор W_k, где номер k и будет номером класса. Поскольку мы инициализировали веса случайным образом, то собственно номера классов роли не играют, важно лишь группировка образов по классам.

Если к весам W_k применить процедуру обратную к нормировке, то коэффициенты весов для каждого класса укажут средние значения компонент для класса.[12]

Теперь же, когда мы изучили более простой вариант, перейдём к самоорганизующейся карте Кохонена - методу, который позволяет производить уже кластерный анализ.

3 Самоорганизующаяся карта Кохонена

Самоорганизующуюся карта - это разновидность сети Кохонена, которая позволяет не только производить кластеризацию объектов, но и выполнять многомерную визуализацию ее результатов.

Отличие самоорганизующейся карты от обычной сети Кохонена заключается в количестве выходных нейронов: в сети Кохонена оно должно соответствовать количеству кластеров, а в карте – количеству сегментов, из которого она должна состоять, т.е. размеру карты. Чем больше число сегментов в карте, тем детальнее она представляет распределение объектов в пространстве признаков.

Число входных нейронов карты, как и сети Кохонена, должно быть равно числу признаков, по которым производится кластеризация.

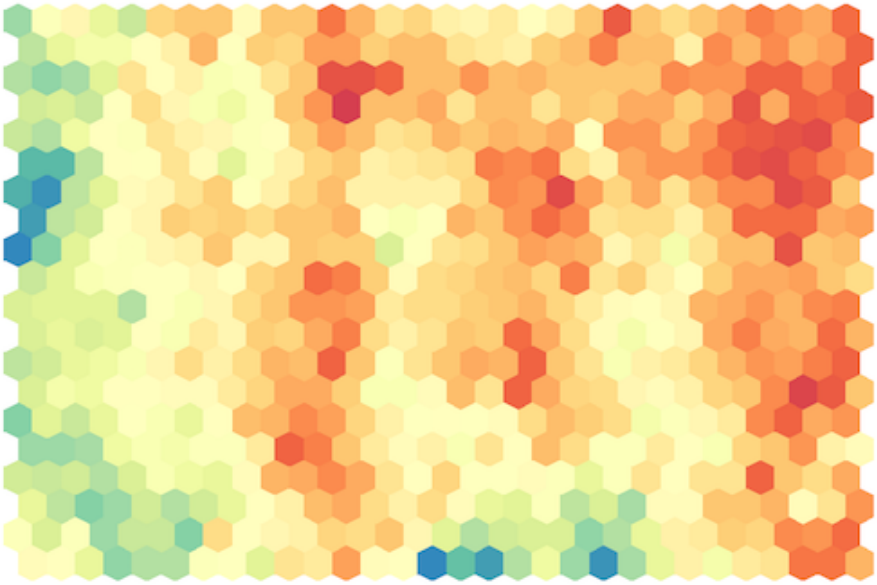

Рис. 3: Примерная иллюстрация получаемой карты. Важно понимать, что, как правило, каждой ячейке соответствует один нейрон.

3.1 Структура

Самоорганизующаяся карта отличается от типичной нейронной сети и по своей архитектуре, и по алгоритмическим свойствам. Она состоит всего лишь из одного слоя вместо нескольких. Важным отличием алгоритма SOM является то, что в нем все нейроны (узлы, центры классов...) упорядочены в некоторую структуру(обычно двумерную сетку, см. Рис. 4). При этом в ходе обучения модифицируется не только нейрон-победитель, но и его соседи, но в меньшей степени. За счет этого SOM можно считать одним из методов проецирования многомерного пространства в пространство с более низкой размерностью. При использовании этого алгоритма вектора, схожие в исходном пространстве, оказываются рядом и на полученной карте.[9]

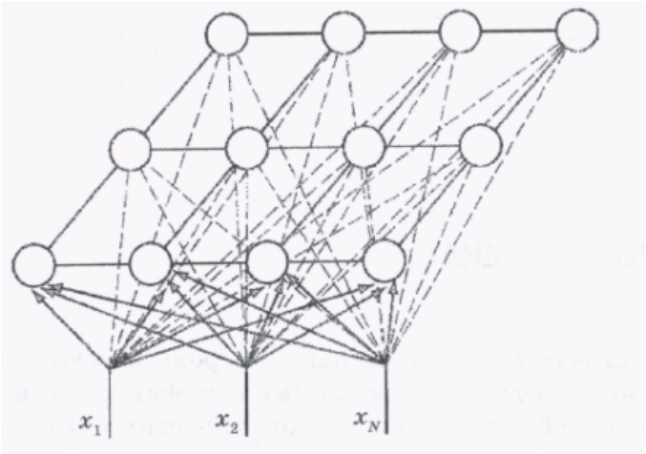

Рис. 4: Структура сети, нейроны составляют двумерную сетку.

3.2 Алгоритм обучения SOM

Теперь рассмотрим алгоритм обучения SOM. По существу, он не сильно отличается от алгоритма обучения нейронной сети Кохонена. По ходу обучения мы также будем находить

ближайший к вектору исходных данных X_m вектор весов W_k и изменять его, но изменения также будут затрагивать его соседей.

Пусть у нас есть набор из M исходных образов, который нужно кластеризовать. Каждый образ будем описывать некоторым вектором

$$X^m = [x_1^m, x_2^m, ..., x_N^m],$$

где x_i^m суть действительные числа.

Обучать мы будем весовые коэффициенты

$$W^k = [w_1^k, w_2^k, ..., w_N^k],$$

$k = 1, 2, ..., K$, где K есть количество нейронов.

Количество шагов равно λ.

Алгоритм:

1. инициализируем веса нейронов
2. берём случайный вектор X^t из множества входных данных, находим ближайший к нему нейрон (best matching unit, BMU), пусть u - его индекс.
3. изменяем веса нейронов по формуле:

$$W^v(s+1) = W^v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot (X^t - W^v(s)),$$

   где:

   - $W^v(s)$ - веса нейрона W_v на итерации s.
   - $\theta(u, v, s)$ - расстояние между нейронами W^u и W^v. Обычно используется один из двух вариантов: либо функция Гаусса, либо же расстояние до соседей равно 1, а до остальных нейронов оно приравнивается 0.
   - $\alpha(s) > 0$ - шаг обучения, убывающая функция.

4. повторяем с шага 2, пока $s < \lambda$.[8]

Отметим, что предложенный алгоритм не использует явно никакого критерия оптимизации. Хотя ясно, что, по крайней мере, будет уменьшаться среднее расстояние от каждой точки данных до ближайшего узла карты. Средний квадрат такого расстояния иногда используют для оценки точности карты.[14]

После завершения обучения мы разбиваем нейроны на отдельные кластеры. И теперь мы можем кластеризовать исходные данные, занося их кластеры, которым принадлежат ближайшие к ним нейроны. Пример разбиения нейронов на кластеры вы можете видеть на Рис. 10

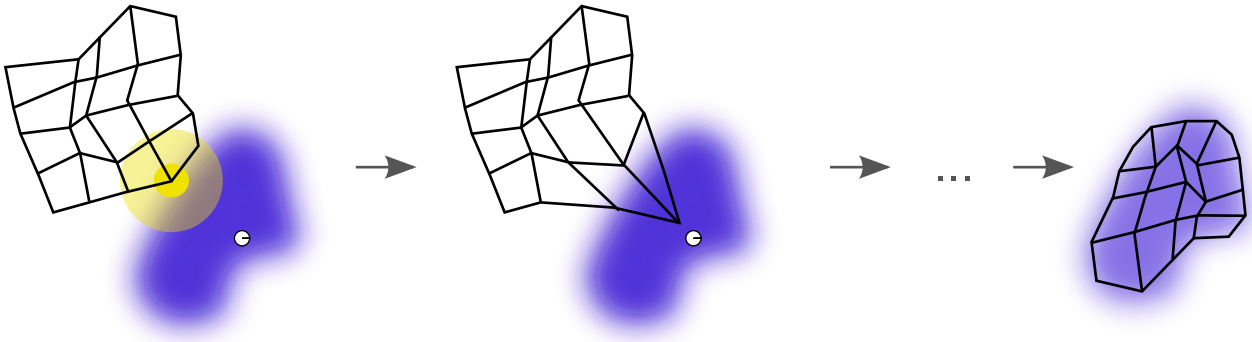

Рис. 5: Иллюстрация того, как веса нейронов в процессе обучения постепенно приближаются к исходным данным.

3.3 Методы представления

После того, как самоорганизующейся карта обучена, нам необходимо как-то взглянуть на получившийся результат. Существует несколько стандартных способов это сделать, и сейчас мы их рассмотрим.

3.3.1 U-matrix

Унифицированная матрица расстояний(U-матрица, U-matrix, unified distance matrix) - это одно из представлений самоорганизующейся карты. Она визуализирует расстояния между нейронами. Расстояние между соседними нейронами вычисляется и представляется в виде различной окраски между соседними узлами. Темная окраска между нейронами соответствует большому расстоянию и, следовательно, разрыву между значениями во входном пространстве. Светлая окраска между нейронами означает, что векторы находятся близко друг к другу во входном пространстве. Светлые области можно рассматривать как кластеры, а темные - как разделители кластеров. Это может быть полезным представлением, когда вы пытаетесь найти кластеры во входных данных, не имея никакой априорной информации о них.[11] Пример U-matrix приведён на Рис. 6 Также существует трёхмерное расширение U-matrix - U-array (см. Рис. 7)

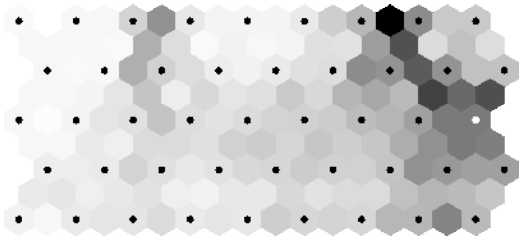

Рис. 6: Пример U-matrix. Чёрные точки обозначают нейроны, чем больше расстояние между соседними нейронами, тем темнее ячейки, находящиеся между ними.

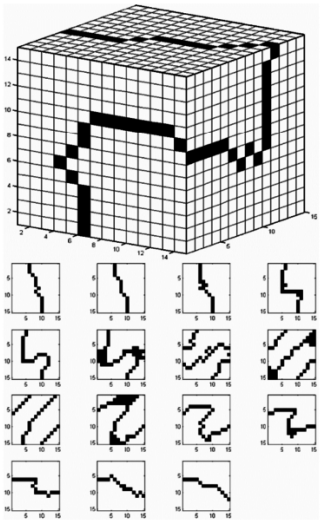

Рис. 7: U-array - трёхмерная U-matrix

3.3.2 Карта входов нейронов

Карта входов нейронов строится для каждого входа (x_i), где нейроны раскрашиваются в соответствии с весом w_i, соответствующим этому входу. Пример можно увидеть на рис. 10, где все карты, кроме первых трёх, являются картами входов.

3.3.3 Карта выходов нейронов

На карту выходов нейронов проецируется взаимное расположение исследуемых входных данных. Примером служит правая нижняя карта на Рис. 11

3.3.4 Специальные карты

Это карта кластеров, матрица расстояний(U-matrix), матрица плотности попадания и другие карты, которые характеризуют кластеры, полученные в результате обучения сети Кохонена. Примером служат первые две карты из рис. 10

Важно понимать, что между всеми рассмотренными картами существует взаимосвязь - все они являются разными раскрасками одних и тех же нейронов. Каждый пример из обучающей выборки имеет одно и то же расположение на всех картах.[10]

3.4 Вычисление расстояний в U-matrix

В данном пункте мы углубимся в вопрос вычисления расстояний для построения U-matrix. В этом нам поможет работа [16], в которой приведены формулы и соответствующие пояснения.

Унифицированная матрица для двумерных карт (U-matrix) имеет размерность $(2n-1)$x$(2m-1)$, где n-число нейронов по оси x, m – число нейронов по оси y. U-matrix включает в себя следующие расстояния между нейронами: $dx(x,y)$, $dy(x,y)$ и $dxy(x,y)$.

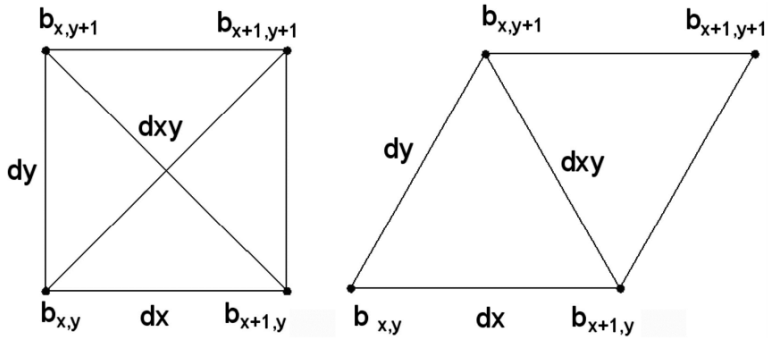

Рис. 8: Обозначения для прямоугольной и гексогональной решёток

Для прямоугольной и гексогональной матрицы $dx(x,y)$ и $dy(x,y)$ вычисляются одинаково:

$$dx(x,y) = \|b_{x,y} - b_{x+1,y}\| = \sqrt{\sum_i \left(w_{i_{x,y}} - w_{i_{x+1}y+1}\right)^2}$$
$$dy(x,y) = \|b_{x,y} - b_{x,y+1}\| = \sqrt{\sum_i \left(w_{i_{x,y}} - w_{i_{x,y+1}}\right)^2}$$

А вот $dxy(x,y)$ отличается. Например, для прямоугольной:

$$dxy(x,y) = \frac{1}{2}\left(\frac{\|b_{x,y} - b_{x+1,y+1}\|}{\sqrt{2}} + \frac{\|b_{x,y+1} - b_{x+1,y}\|}{\sqrt{2}}\right) =$$
$$= \frac{1}{2\sqrt{2}}[\sqrt{\sum_i \left(w_{i_{xy}} - w_{i_{x+1},y+1}\right)^2} + \sqrt{\sum_i \left(w_{i_{xy+1}} - w_{i_{x+1},y}\right)^2}]$$

и для гексогональной:

$$dxy(x,y) = \|b_{x,y+1} - b_{x+1,y}\| = \sqrt{\sum_i \left(w_{i_{xy+1}} - w_{i_{x+1},y}\right)^2}$$

3.4.1 Вычисление расстояний в U-array

U-array является расширением U-matrix для трехмерных карт. Этот массив будет иметь размерность $(2n-1)$x$(2m-1)$x$(2p-1)$, где p - число нейронов по оси z. U-array включает в себя дополнительные дистанции: $dz(x,y,z)$, $dxz(x,y,z)$, $dyz(x,y,z)$ и $dxyz(x,y,z)$. Расстояние $dz(x,y,z)$ рассчитывается аналогично расстояниям $dx(x,y)$ и $dy(x,y)$. Расстояния $dxz(x,y,z)$ и $dyz(x,y,z)$ рассчитываются таким же способом, как и расстояние $dxy(x,y)$.

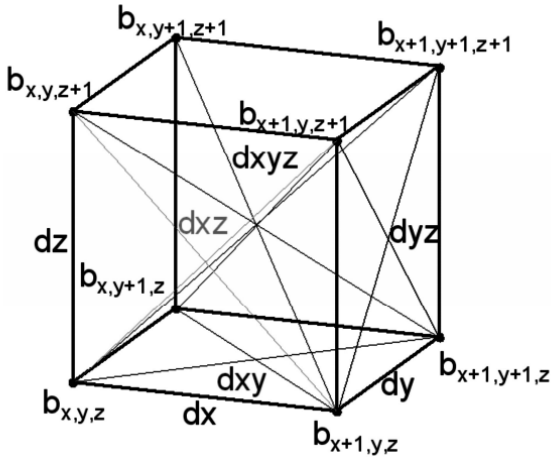

Рис. 9: Обозначение расстояний U-array для прямоугольной решётки

Расстояние $dxyz(x,y,z)$ на прямоугольной решётке можно рассчитывать по следующей формуле:

$$dxyz(x,y,z) = \frac{1}{4}\left(\frac{\|b_{x,y,z} - b_{x+1,y+1,z+1}\|}{\sqrt{3}} + \right.$$
$$+ \frac{\|b_{x+1,y,z} - b_{x,y+1,z+1}\|}{\sqrt{3}} + \frac{\|b_{x+1,y+1,z} - b_{x,y,z+1}\|}{\sqrt{3}} +$$
$$\left. + \frac{\|b_{x,y+1,z} - b_{x+1,y,z+1}\|}{\sqrt{3}} \right) =$$
$$= \frac{1}{4\sqrt{3}}\left[\sqrt{\sum_i \left(w_{i_{x,y,z}} - w_{i_{x+1,y+1,z+1}}\right)^2} + \right.$$
$$+ \sqrt{\sum_i \left(w_{i_{x+1,y,z}} - w_{i_{x,y+1,z+1}}\right)^2} +$$
$$+ \sqrt{\sum_i \left(w_{i_{x+1,y+1,z}} - w_{i_{x,y,z+1}}\right)^2} +$$
$$\left. + \sqrt{\sum_i \left(w_{i_{x,y+1,z}} - w_{i_{x+1,y,z+1}}\right)^2}\right]$$

Для вычисления расстояний на гексогональной карте требуется уже несколько больших групп формул, поэтому для тех, кто хочет разобраться в этом вопросе, снова привожу ссылку на статью [16] и предлагаю ознакомиться самостоятельно.

4 Примеры использования

Рассмотрим несколько примеров использования карт.

1. Пример из области политики. Здесь сеть обучена на наборе данных о Конгрессе США.

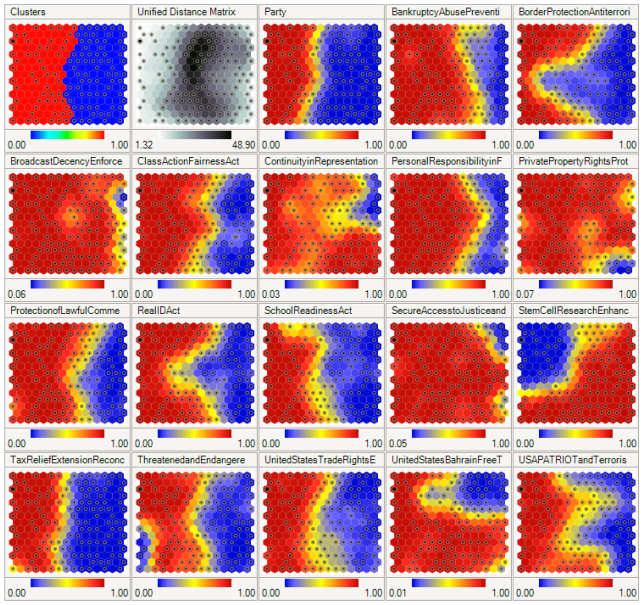

Рис. 10: Самоорганизующаяся карта, показывающая схему голосования в Конгрессе США. Исходные данные представляли собой таблицу со строкой для каждого члена Конгресса и столбцами для определенных голосов, содержащими голос каждого члена "да "/ "нет "/ "воздержался". Алгоритм SOM расположил эти элементы в двумерной сетке, расположив подобные элементы ближе друг к другу. Первый график показывает группировку, когда данные разделены на два кластера. Второй график показывает среднее расстояние до соседей: большие расстояния темнее. Третий график предсказывает членство в Республиканской (красной) или Демократической (синей) партии. Остальные графики являются картами по одному из входов(карта входов нейронов): красный цвет означает предсказанный голос "да"по этому законопроекту, синий - голос "нет".[7]

2. Здесь сеть обучена на наборе данных "Ирисы Фишера"[13].

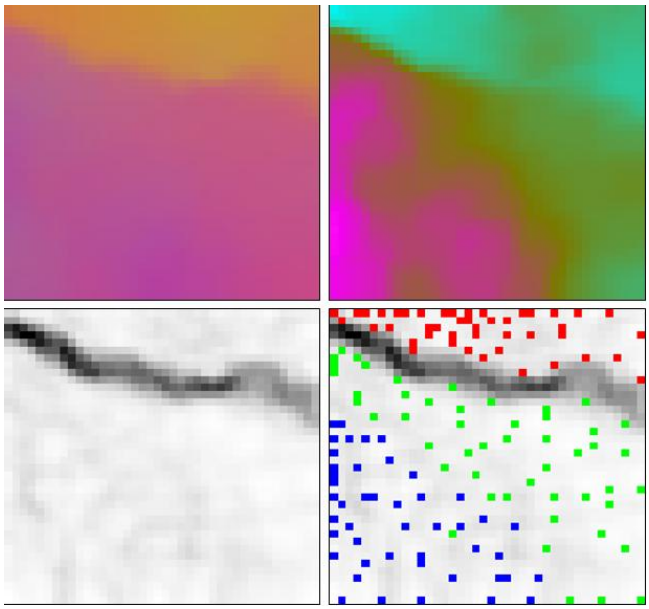

Рис. 11: Самоорганизующаяся карта, натренированная на наборе данных "Ирисы Фишера"[13]. Вверху слева: цветное изображение, образованное первыми тремя измерениями четырехмерных весовых векторов нейронов SOM. Вверху справа: псевдоцветное изображение величины векторов веса SOM. Внизу слева: U-Матрица (Евклидово расстояние между весовыми векторами соседних ячеек). Внизу справа: карта выходов нейронов, наложение точек данных (красный: I. setosa, зеленый: I. versicolor и синий: I. virginica) на U-матрицу, основанную на минимальном евклидовом расстоянии между векторами данных и весовыми векторами SOM.

3. В данной статье автор показывает, как в Python сгенерировать набор данных, а потом обучить на нём сеть Кохонена.

   В ней он сначала генерирует множество трёхмерных векторов(цвета в пространстве RGB) и обучает на нём сеть, созданную в TensorFlow. Результаты приведены ниже.[15]

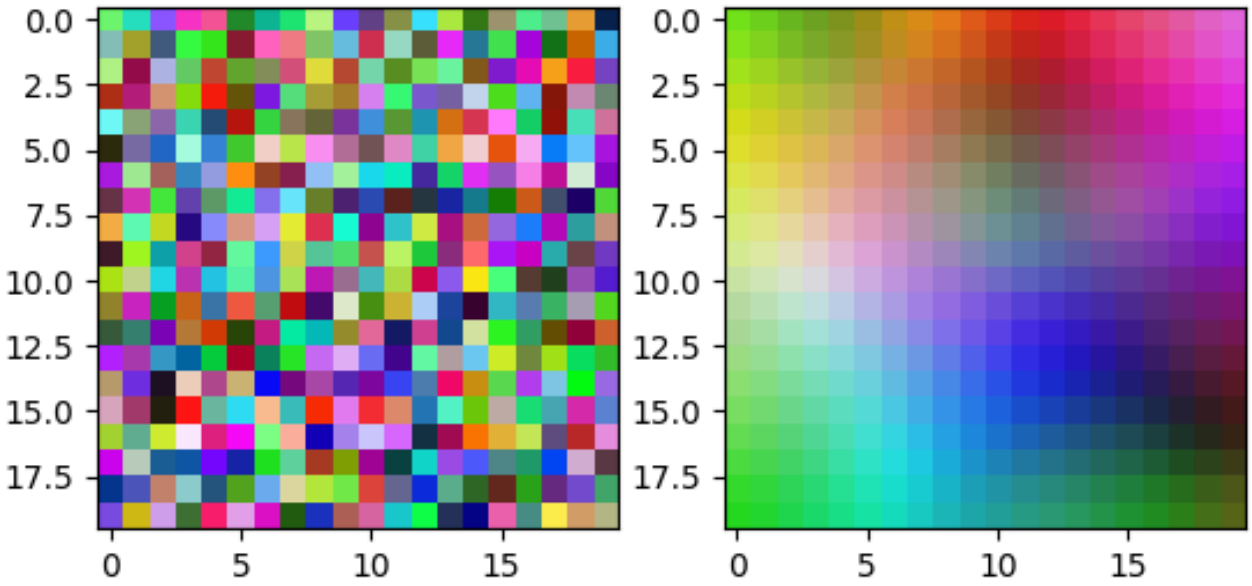

Рис. 12: Карта весов в начале обучения (слева) и в конце обучения (справа) для 20 нейронов.

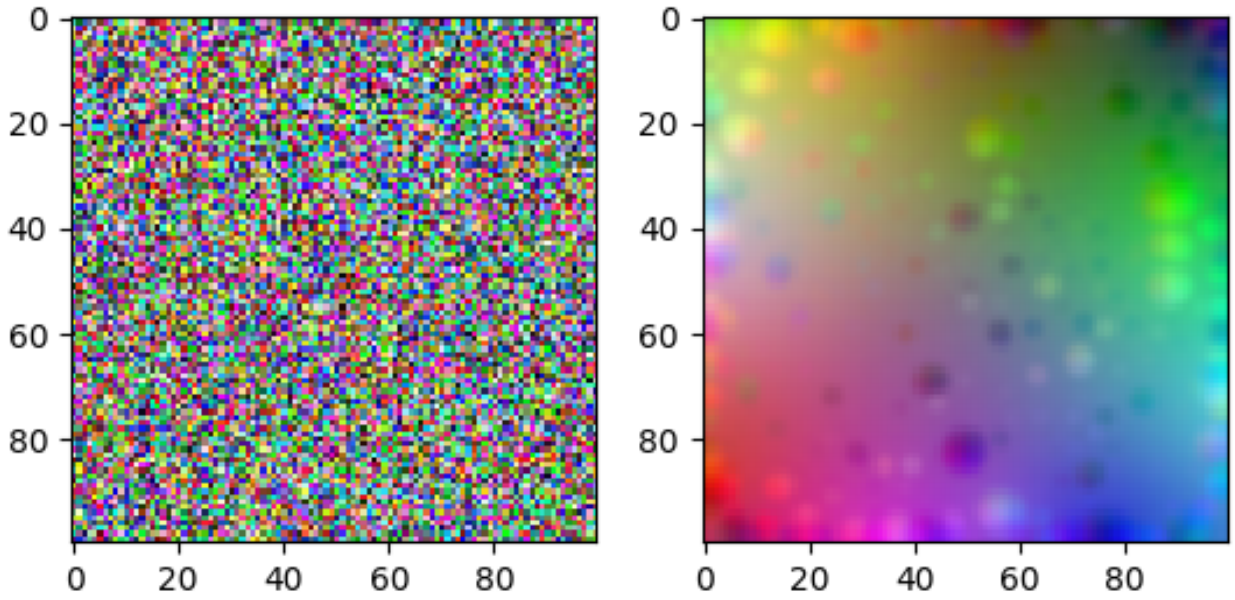

Рис. 13: Карта весов для сети 100x100 нейронов, после 350тыс. итераций обучения.

5 Вопросы

1. слой Кохонена
2. геометрическая интерпретация

3. структура самоорганизующихся карт
4. алгоритм обучения SOM
5. методы представления

6 Источники

[1] https://ami.nstu.ru/~vms/lecture/data_mining/kurs_klaster.htm

[2] https://neerc.ifmo.ru/wiki/index.php?title=Кластеризация

[3] https://docplayer.ru/57111584-Soderzhanie-vvedenie-zadacha-klasterizacii-dannyh-html

[4] Teuvo Kohonen and Timo Honkela (2007) Kohonen network. Scholarpedia, 2(1):1568.

[5] https://ru.wikipedia.org/wiki/Нейронная_сеть_Кохонена

[6] https://ru.wikipedia.org/wiki/Самоорганизующаяся_карта_Кохонена

[7] https://en.wikipedia.org/wiki/Self-organizing_map

[8] https://en.wikipedia.org/wiki/Self-organizing_map#Algorithm

[9] https://basegroup.ru/community/articles/som

[10] https://www.intuit.ru/studies/courses/6/6/lecture/180?page=3

[11] https://users.ics.aalto.fi/jhollmen/dippa/node24.html

[12] http://ai.lector.ru/?go=lection04

[13] https://ru.wikipedia.org/wiki/Ирисы_Фишера

[14] http://pca.narod.ru/ZinovyevBook.pdf

[15] https://habr.com/ru/post/334810/

[16] http://www.dspa.ru/articles/year2015/jour15_2/art15_2_4.pdf