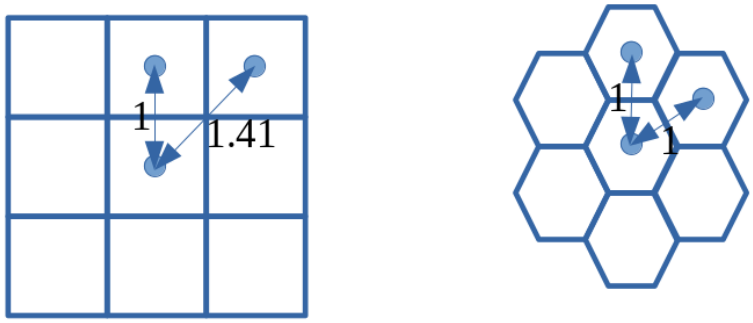
Описание Self-organizing map (SOM)

Нужно сразу оговориться, что и в русскоязычной и в англоязычной частях Интернета найти полное, но главное понятное описание SOM – задача не из легких. Такие статьи, как например эта, уверен была бы непонятная самому Тойво Кохонену. Терминология и формулы везде отличаются, поэтому в общем виде SOM можно описать следующим образом.
Самоорганизующаяся карта Кохонена – нейронная сеть с обучением без учителя, выполняющая задачу визуализации и кластеризации.
Кластеризация - процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы.
SOM состоит из компонентов, называемых узлами (nodes) или нейронами (neurons). Их количество задается аналитиком, исходя из собственного опыта или общепринятой практики для конкретной задачи. У каждого узла есть координаты на карте и соответствующий вектор веса W. Размерность вектора W равна размерности семпла входных данных. Например, входные данные представлены следующим образом:

№ семпла	Признак 1	Признак 2	...	Признак *M*-1	Признак *M*
Семпл 1	Значение признака 1 для семпла 1				
...					
Семпл *N*					Значение признака *M* для семпла *N*

Тогда вектор веса $W_{x,y} = (w_1,w_2,\ldots,w_M)$, где x,y – координаты нейронов (узлов) SOM.
Для визуализации карты выбирают форму отображения нейронов – прямоугольник или шестиугольник (рисунок 1) (шестиугольная форма для карты более корректная, так как в данном случае расстояния между соседними нейронами одинаковое, а при прямоугольной – разное).



а) б)

Рисунок 1 – Примеры расположения узлов на SOM.

SOM работает в двух режимах: training (обучение) and mapping (визуализация).

Создание SOM

Для создания SOM необходимо определиться с тем, сколько она будет иметь нейронов. Проще всего задавать это числом нейронов по одной стороне. Например при числе нейронов по одной стороне 10, общее число нейронов будет $H = 100$. (На самом деле, можно делать любой размер

SOM, включая одномерный). Стоит так же помнить, что в SOM отдельный нейрон – это и отдельный кластер. Если априори известно, что в обрабатываемых данных 1000 классов, то 100 нейронов не хватит для визуализации. И наоборот, если класса всего 2, то 100 нейронов будет многовато. Однако SOM и не применяют в задачах, где априори известно число классов. В этом есть в некотором роде замкнутый круг, однако лучше попытаться хоть что-то узнать о данных с помощью SOM, чем не знать ничего.
Второе, что нужно сделать – это провести начальную инициализацию векторов весов. Размерность семпла признаков известна заранее и составляет M. Поэтому в цикле по нейронам нужно сгенерировать H векторов по M случайных чисел.

Обучение SOM

Обучение SOM по своему принципу очень простое. В отличае от других нейронных сетей SOM обучается по принципу Конкурентного обучения (Competitive learning). Это метод при котором нейроны соперничают друг с другом, чтобы вектор их весов $W_{x,y}$ оказался как можно ближе к вектору признаков X предъявленного объекта.
Алгоритм обучения следующий:
1. Из всех семплов входных данных (общее количество которых N) выбираем случайным образом один X_n, где $n = 1,2,\ldots,N$.
2. В цикле по всем нейронам находим такой нейрон, «расстояние» докоторого наименьшее по формуле:

$$d^n_{W_{x,y}} = \sqrt{\sum_{m=1}^{M} (x^n_m - w^{x,y}_m)^2}, \quad (1)$$

где $d^n_{W_{x,y}}$ — Евклидово расстояние d от семпла X_n до вектора весов $W_{x,y}$ с координатами x,y.
Например:
$X_n = (3, 2, 7, 9.5)$;
$W_{x,y} = (5, 2.4, 10, 1)$;
тогда $d^n_{W_{x,y}} = \sqrt{(3-5)^2+(2-2.4)^2+(7-10)^2+(9.5-1)^2} = \sqrt{85.71} = 9.24$.
Такой нейрон иногда называют BMU (best matching unit). Если выяснилось, что в карте имеются несколько BMU, то нужно выбрать любой из них случайным образом.
3. Обновляем веса карты по формуле:

$$W(t) = W(t-1) + \eta(t) \cdot h(t,\rho) \cdot (X - W(t-1)), \quad (2)$$

$W(t)$ – вектор новых значений веса (иногда встречается «прототип-вектор»);
X – входной вектор (вектор признаков (семпл) одного объекта);
$W(t-1)$ – вектор значений веса на предыдущем шаге;
$h(t, \rho)$ – функция близости узлов (нейронов) в топологии карты Кохонена;
$\eta(t)$ – скорость обучения.
Как видно, формула (2) состоит из двух слагаемых. Новые значения вектора весов определяются путем прибавления (или вычитания) к значениям вектора $W(t-1)$ на предыдущем шаге обучения некоторого числа (второго слагаемого). В свою очередь это второе слагаемое определяется как разница между входным вектором и вектором весов на предыдущем шаге, умноженная на функцию близости и скорость обучения. Отсюда становится понятно, что в процессе обучения значения вектора весов $W_{x,y}$ стараются принять значения очень близкие (похожие) на значения входного вектора. Если убрать из (2) функцию близости и скорость и раскрыть скобки, то можно увидеть что на второй итерации обучения значение $W(t)$ просто станет равно X.
Скорость обучения убывает по экспоненте:

$$\eta(t) = k_0 \cdot e^{-k \cdot t}, \quad (3)$$

t – номер итерации;
k – некоторый коэффициент, регулирующий скорость обучения;
k_0 – некоторое начальное значение с которого начнет убывать скорость.
Примеры представлены на рисунке 2.

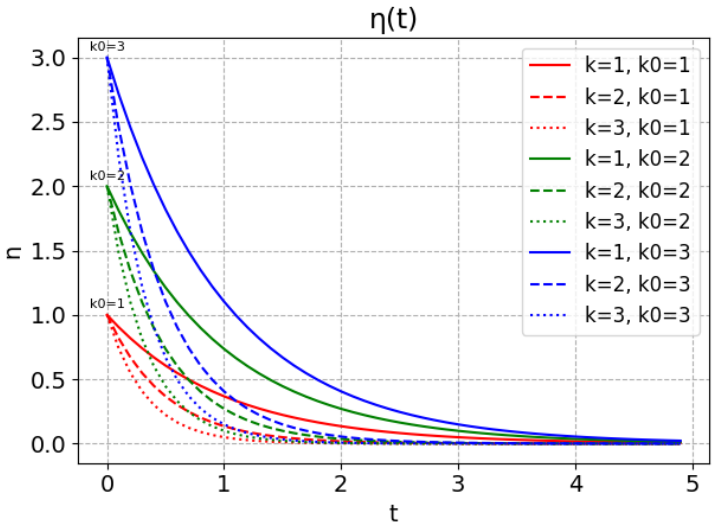


Рисунок 2 – Пример изменения скорости обучения от времени.

Как видно из рисунка 2 от значения k_0 зависит начальная величина скорости. То есть на первых итерациях при расчете второго слагаемого, его значение будет больше, и соответственно будет больше разница между $W(t-1)$ и $W(t)$. С каждым шагом эта разница будет уменьшаться, то есть процесс подстройки весов карты с каждой итерацией обучающего цикла будет все более точнее и в итоге приближаясь к нулю (но не становясь нулем, это видно из функции экспоненты).
Функция близости узлов изменяется по функции Гаусса:

$$h(t,\rho)=e^{\frac{-\rho^2}{2\cdot\sigma(t)}}, \quad (4)$$

$$\sigma(t)=a_0\cdot e^{-a\cdot t}, \quad (5)$$

a – некоторый коэффициент, регулирующий диаметр функции близости;
a_0 – некоторое начальное значение с которого начнет убывать диаметр;
ρ – функция расстояния между узлами на карте Кохонена.
Из формулы 4 видно что функция близости зависит от трех параметров a, a_0 и ρ. Два из них a и a_0 – это константы, задаваемые аналитиком самостоятельно (как и в случае для k_0 и k). Третий параметр ρ считается также как и расстояние d в формуле (1), только туда подставляются другие значения, а именно координаты узлов (поэтому шестиугольник более правильное представление карты, чем прямоугольник).
Например, оказалось, что узел с координатами 2,3 является BMU. Тогда расстояние до узла с координатами 3,3 равно

$$\rho=\sqrt{(2-3)^2+(3-3)^2}=\sqrt{1}=1,$$

а до узла с координатами 3,4

$$\rho=\sqrt{(2-3)^2+(3-4)^2}=\sqrt{2}=1.41,$$

как и было показано на рисунке 1 а).
Примеры представлены на рисунках 3,4.

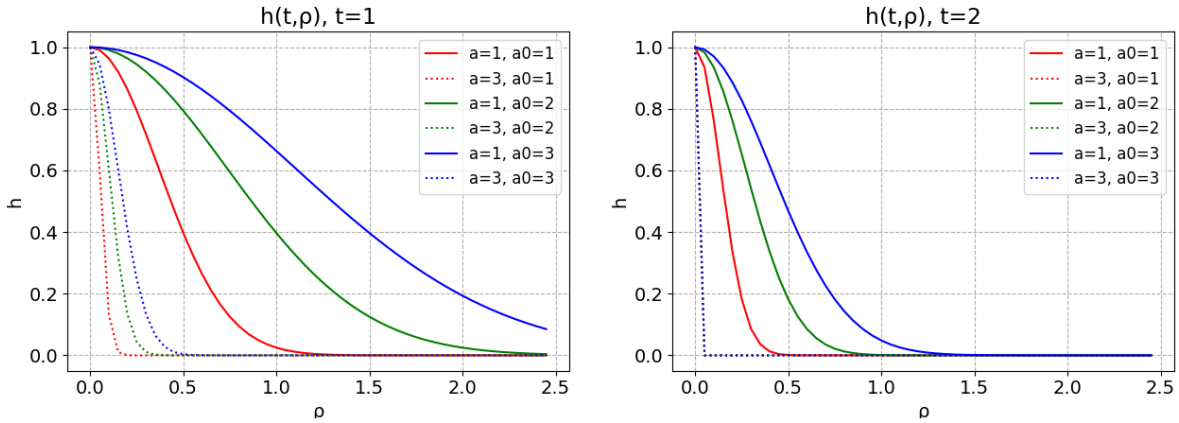


Рисунок 3 – Пример изменения значения функции близости от расстояния между узлами и временем.

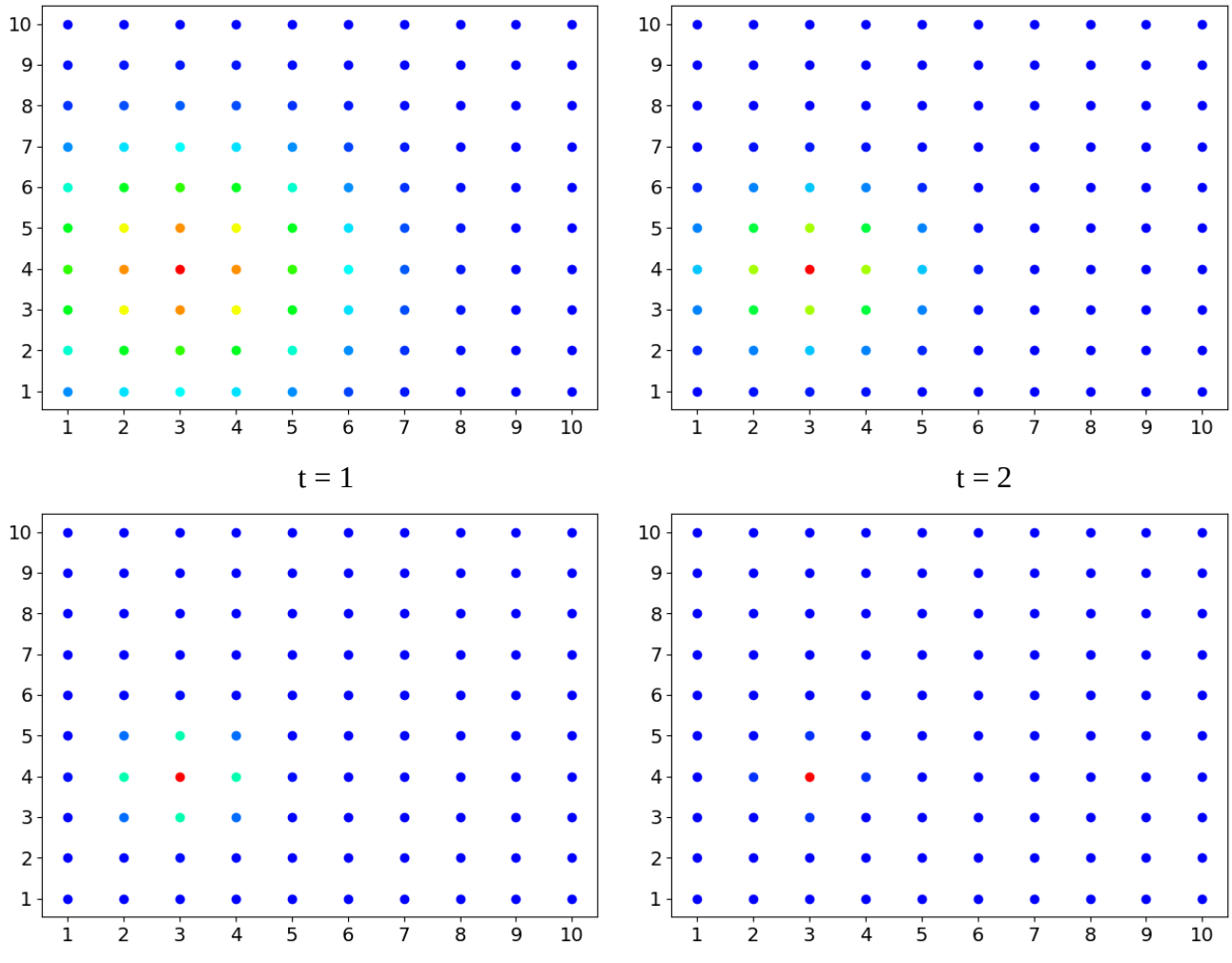


Рисунок 4 – Пример влияния BMU на соседнии нейроны при $a = 0.5$, $a_0 = 3$.

Из рисунка 4 видно, что с каждой итерацией t радиус и сила влияния BMU уменьшаются.

Таким образом одна итерация обучения состоит из 3-х простых шагов. Обучение необходимо продолжать, пока SOM не настроится должным образом. Это можно определить по разным критериям, например вычислив значение критерия среднего растояния до ближайших нейронов

$$E=\frac{1}{N}\cdot\sum_{n=1}^{N} d_n, \qquad (6)$$

где d_n – это минимальное расстояние d (рассчитанное по формуле (1)) между одним из нейронов (BMU) и n-ым семплом входных данных.

Другим вариантом остановки может просто являться достижение заданного количества итераций T.

Пример обучения SOM представлен на рисунках 5, 6. Это “Hello, world!” в мире SOM. Массив входных данных размером 8 семплов по 3 признака. Каждый семпл – отдельный цвет в палитре RGB. Каждый признак – это порция цвета в каждом из каналов от 0 до 255:

Красный (Red): (255,0,0);
Зеленый (Green): (0,128,0);
Голубой (Blue): (0,0,255);
Темно-зеленый (Dark Green): (0,100,0);
Темно-синий (Dark Blue): (0,0,139);
Желтый (Yellow): (255,255,0);
Оранжевый (Orange): (255,165,0);
Фиолетовый (Purple): (128,0,128).

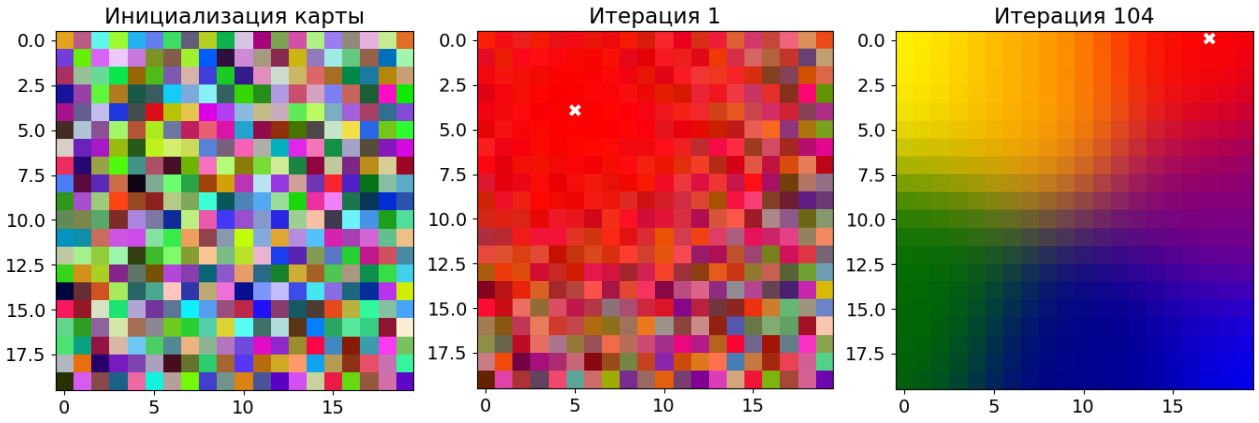


Рисунок 5 – Пример обучения SOM.

Как видно из рисунка 5 на первой итерации алгоритм случайно выбрал красный цвет из входных данных, нашел на карте наиболее похожий на красный цвет узел и обновил значения весов ближайших узлов. В итоге за 104 итерации обучения на карте расположилось 8 цветов, как это и было во входных данных.

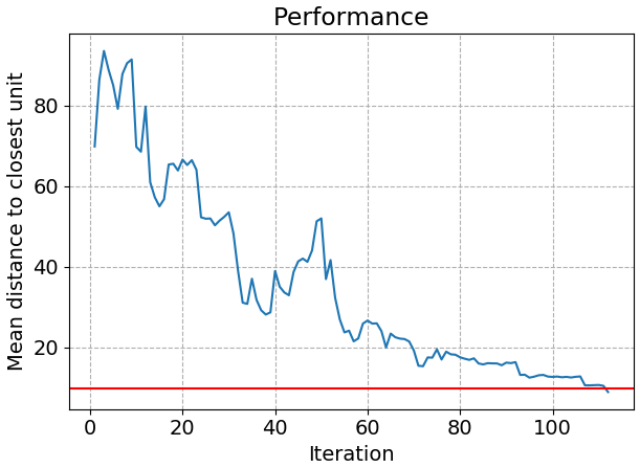


Рисунок 6 – Кривая обучения SOM.

Из рисунка 6 видно, как изменялась точность карты, рассчитанная по формуле (6).
Нужно сразу оговориться, что этот пример можно применять только в учебных целях. Из-за того, что во многих источниках (учебниках, статьях) по SOM приводится именно этот пример, то он сбивает с правильного понимания процесса визуализации карты.
Что мы имеем после настройки карты? *H* векторов весов по *M* чисел. Каждый вектор весов соответствует нейрону с координатой *x*, *y*. Поэтому задача визуализации настроенной карты сводится к задаче придания цвета отдельным нейронам на плоскости. Как мы знаем для цветовой модели RGB цвет складывается из 3-х составляющих. По этой причине, чтобы придать цвет нейрону его вектор весов должен иметь 3 элемента (числа), а это означает, что входные данные имеют всего 3 признака. Для данных, имеющих 4 признака можно применить цветовую модель CMYK.

Работа SOM

Итак, реализовав вышеописанный алгоритм мы получим настроенную SOM. В чем же заключается работа SOM? Всё очень просто – в SOM подается семпл и для него находится BMU. Результат работы – это координаты *x*, *y* BMU, вот и всё.
Рассмотрим другой пример.

Продукты	**Белки**	**Жиры**	**Углеводы**
Яблоки	0.4	0.1	11.8
Авокадо	1.9	19.5	1.9
Бананы	1.2	0.3	23.2
Говяжий стейк	20.9	7.9	0
Биг Мак	13	11	19
Бразильские орехи	15.5	68.3	2.9
Хлеб	10.5	3.2	37
Масло	1	81	0
Сыр	25	34.4	0.1
Ватрушка	6.4	22.7	28.2

Печенье	5.7	29.3	58.7
Кукурузные хлопья	7	0.9	84
Яйца	12.5	10.8	0
Жареный цыпленок	17	20	7
Жаркое	3	13	36
Горячий шоколад	3.8	10.2	19.4
Пепперони	20.9	38.3	5.1
Пицца	12.5	11	30
Пирог со свининой	10.1	24.2	27.3
Картофель	1.7	0.3	16.1
Рис	6.9	2.8	74
Жаркое из курицы	26.1	5.8	0.3
Сахар	0	0	95.1
Стейк из тунца	25.6	0.5	0
Вода	0	0	0

Данные представим в формате csv. Создадим SOM 10x10 и обучим её.

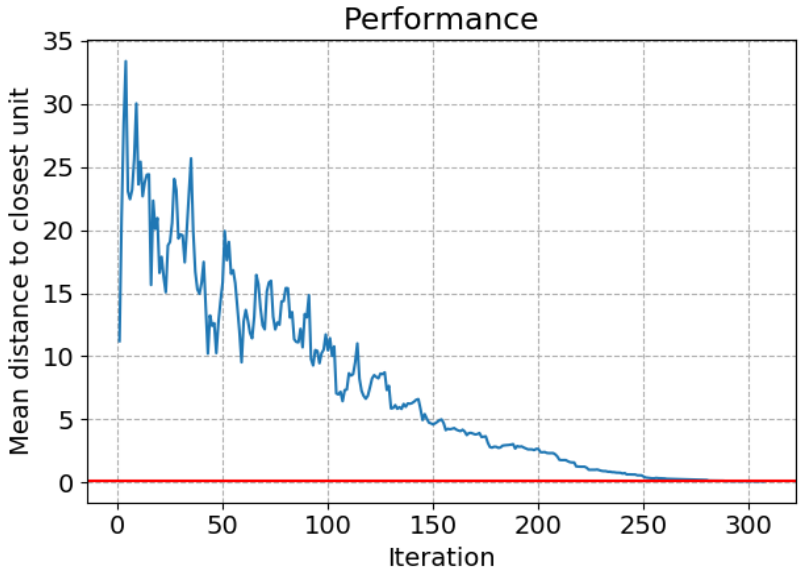


Рисунок 7 – Кривая обучения SOM для визуализации продуктов по БЖУ.

Как видно из рисунка 7 обучение SOM прошло успешно и требуемая точность (красная слиния) достигнута за 306 итерации.

Далее в обученную SOM можно подать один из продуктов и посмотреть какой из нейронов сигнализирует нам о том, что на вход SOM поданы данные по БЖУ этого продукта

То есть после работы SOM мы получим расстояния до каждого вектора весов от входных данных. Данное расстояние можно окрасить разными палитрами, например:

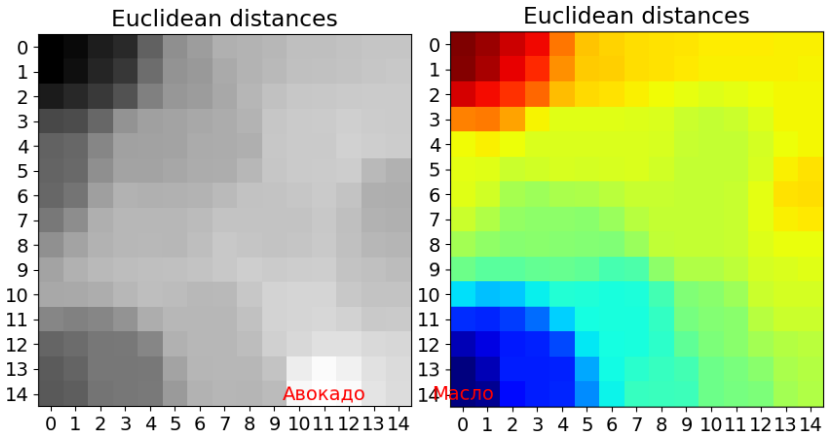


Рисунок 8 – Примеры работы SOM, раскаршенные в оттенки серого или в градиентную палитру.

Анализ данных с SOM

Однако работа SOM в режиме нахождения BMU не является приоритетной. Их предназначение это анализ данных путем визуализации карт. Это очень гибкий процесс, требующий от аналитика некоторого опыта работы с данными. Визуализировать результат можно разными способами. Один из наиболее распространенных – это раскраска карты, порожденная отдельными компонентами (признаками). Полученные раскраски в совокупности образуют атлас.

Например, в вышеописанном примере у каждого продукта было 3 признака. Соответственно атлас состоять из 3-х карт:

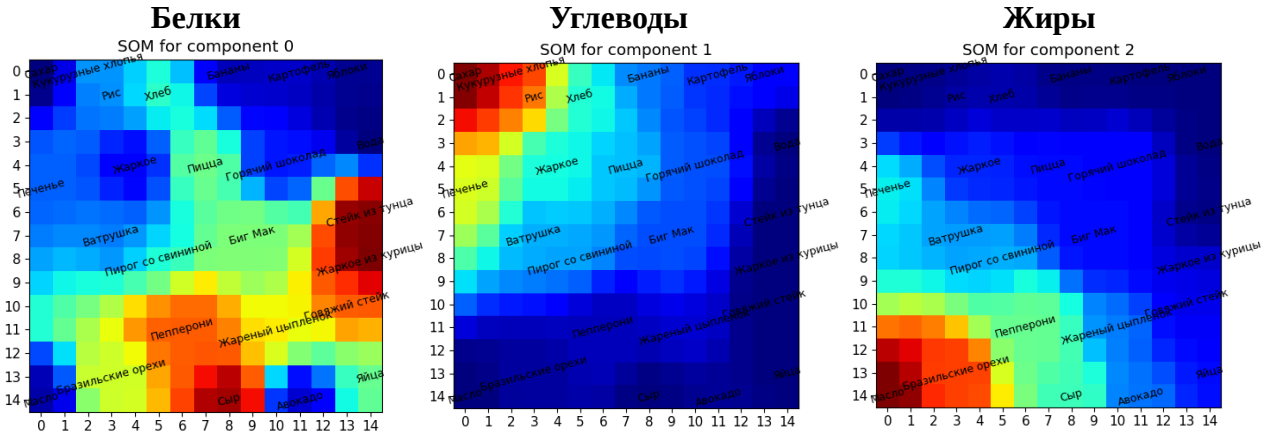


Рисунок 8 – Атлас карт для признаков.

Из атласа видно, что SOM очевидно правильно расположил продукы на плоскости по каждой компоненте. Это не кажеться удивительным, потому что эта информация для нас привычна, но что важно – продукты (объекты) с похожими характеристиками (по всем компонентам) расположились рядом, образуя кластеры. Для данной выбрки из 225 нейронов карты только 24 задействованы и каждому из этих 24 нейронов (которые являются BMU для каждого входного семпла) соответствует один семпл из входных данных.

Но могут быть и другие случаи, всё очень индивидуально. Например можно построить карту, отвеающую на вопросы:

Сколько объектов связано с каждым узлом?

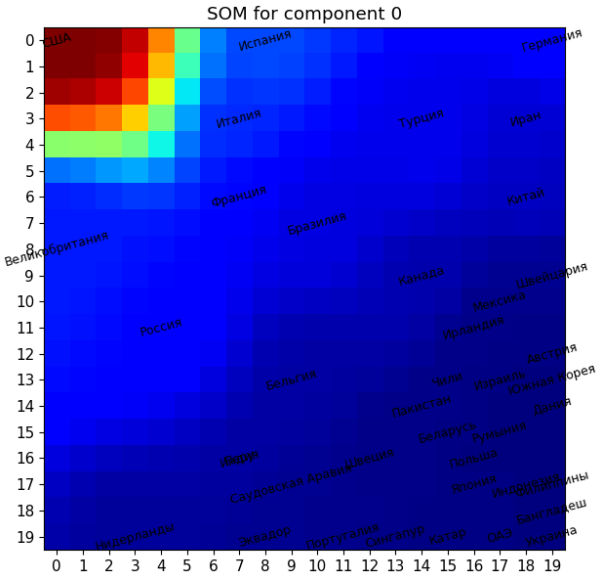
Каково среднее расстояние объектов узла до его прототип векторов?

И т.д. Подробно про примеры визуализации есть в литературе.

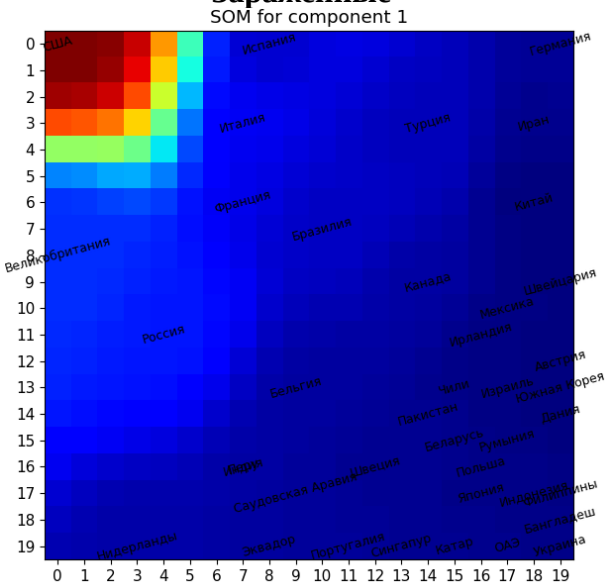
Пример о насущном

Выгрузим в csv официальную статистику по COVID-19 по странам на 6 мая 2020. Для простоты эксперимента возьмем топ-40 по числу подтвержденных случаев.
Создадим SOM размером 20 на 20 и обучим его.
Далее отобразим атлас по компонентами

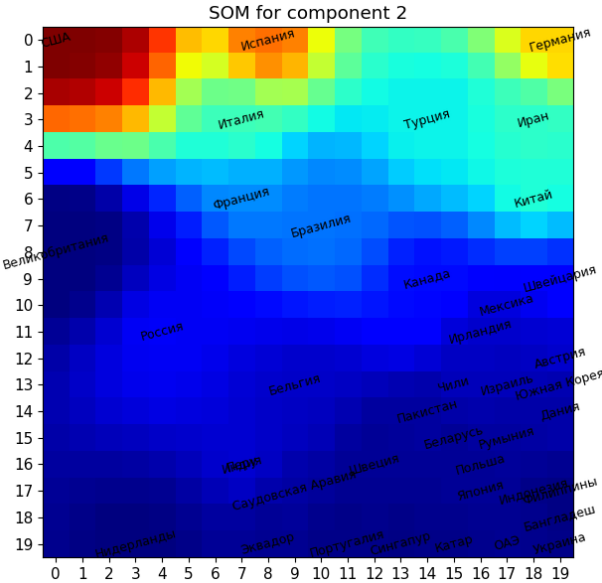
Всего случаев



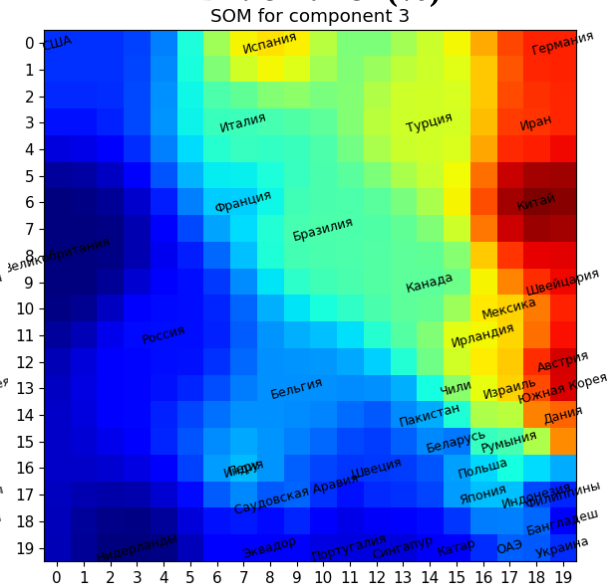
Зараженные



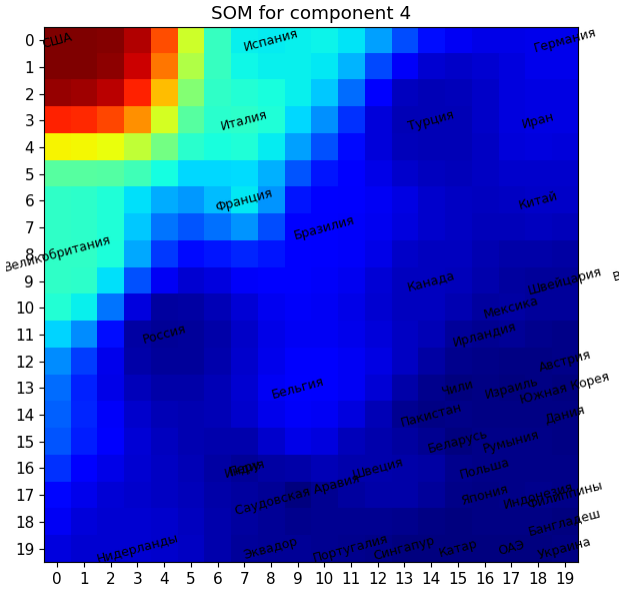
Вылечились



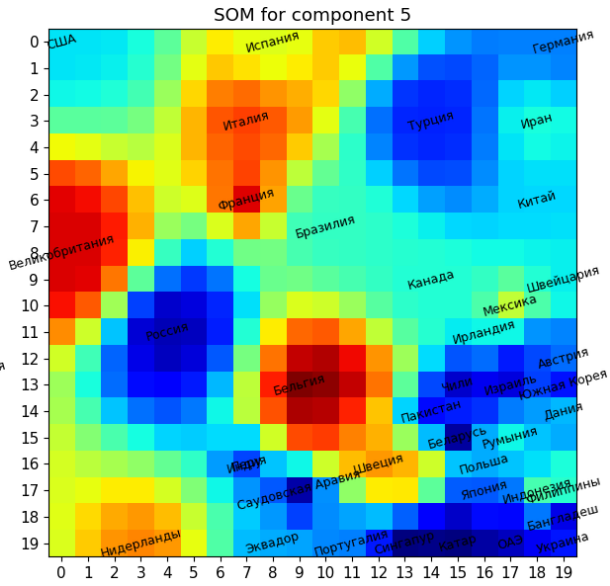
Вылечились (%)



Умерло



Умерло (%)



Выводы из атласа делайте сами.

Литература

1. https://basegroup.ru/community/articles/som
2. https://www.mql5.com/ru/articles/283
3. https://ranalytics.github.io/data-mining/105-Cohonen-Maps.html