Лабораторная работа №4
Нечёткая кластеризация. Самоорганизующиеся карты Кохонена

Цель работы: изучить метод нечёткой кластеризации Fuzzy C-means и научиться применять его для решения задач с помощью MATLAB; освоить основные принципы решения задачи кластеризации и использованием нейронных сетей со слоем Кохонена и самоорганизующихся карт.

Продолжительность работы: 4 часа.

Теоретические сведения

Кластерный анализ. Нечёткая кластеризация

Кластерный анализ (кластеризация) – это объединение в группы (кластеры) на основе схожести признаков для объектов одной группы и отличий между группами. Большинство алгоритмов кластеризации не опираются на традиционные для статистических методов допущения; они могут использоваться в условиях почти полного отсутствия информации о законах распределения данных. Кластеризацию проводят для объектов с количественными (числовыми), качественными или смешанными признаками.

Рассмотрим множество объектов X и множество номеров (имён, меток) кластеров Y. Для множества объектов зададим функцию расстояния $\rho(x, x')$. Исходное множество объектов как правило делится на обучающую $X^m = \{x_1, \dots, x_m\}$ и тестовую $X^p = \{x_m, \dots, x_{m+p}\}$ выборку из X. Задача кластеризации заключается в разбиении множества объектов на подмножества (кластеры). Кластеры состоят из объектов, для которых функция ρ меньше всего. Каждому объекту $x_i \in X$ приписывается номер кластера y_i.

Под алгоритмом кластеризации подразумевают функцию f, осуществляющую отображение $f_a: X \to Y$ (объекту $x \in X$ ставится в соответствие номер кластера $y \in Y$). В задаче классификации множество Y известно заранее (обучение с учителем), но в задаче кластеризации оно неизвестно (обучение без учителя).

Для того чтобы определить наличие сходства или различия между объектами кластеризации, используется метрическое расстояние между ними. Если каждый объект описывается i свойствами (признаками), то он может быть представлен как точка в i-мерном пространстве, и сходство с другими объектами будет определяться как соответствующее расстояние.

Существует множество метрик, в качестве основных используются следующие.

1. Евклидово расстояние является наиболее распространённой функцией расстояния и представляет собой геометрическое расстояние между объектами в многомерном пространстве:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}. \qquad (1)$$

2. Квадрат евклидова расстояния используется для более чёткой сепарации отдалённых друг от друга объектов. Данное расстояние вычисляется в соответствии со следующим выражением:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2. \quad (2)$$

3. Метрика «манхэттенское расстояние» является средней разностью по координатам. Как правило, её использование идентично обычному расстоянию Евклида:

$$\rho(x, x') = \sum_{i}^{n} |x_i - x_i'|. \quad (3)$$

4. Метрика Чебышева может использоваться в случае, когда необходимо определить два различных объекта, если они разнятся по какой-либо одной координате:

$$\rho(x, x') = \max(|x_i - x_i'|). \quad (4)$$

5. Метрика степенного расстояния используется при необходимости увеличения или уменьшения веса и относится к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по формуле:

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p}. \quad (5)$$

где r и p - параметры, которые определяет пользователь. Параметр p связан с постепенным взвешиванием разностей по отдельным координатам, параметр r связан с прогрессивным взвешиванием больших расстояний между объектами.

Результаты кластеризации могут существенно отличаться при использовании разных мер. Основной используемой мерой является евклидово расстояние.

Существует множество методов кластеризации, которые можно разделить на чёткие и нечёткие. Чёткие методы кластеризации разбивают исходное множество объектов X на несколько непересекающихся подмножеств. При этом любой объект из X принадлежит только одному кластеру. Нечёткие методы кластеризации позволяют одному и тому же объекту принадлежать одновременно нескольким (или даже всем) кластерам, но с различной степенью. Нечёткая кластеризация во многих ситуациях более «естественна», чем чёткая, например, для объектов, расположенных на границах кластеров.

FCM-алгоритм кластеризации

Алгоритм нечёткой кластеризации называют **FCM-алгоритмом** (**Fuzzy Classifier Means**, **Fuzzy C-Means**). Целью **FCM**-алгоритма кластеризации является автоматическая классификация множества объектов, которые задаются векторами признаков в пространстве признаков. Другими словами, такой алгоритм определяет кластеры и соответственно классифицирует объекты. Кластеры представляются нечёткими множествами, и, кроме того, границы между кластерами также являются нечеткими.

FCM-алгоритм кластеризации предполагает, что объекты принадлежат всем кластерам с определенной ФП. Степень принадлежности определяется расстоянием от объекта до соответствующих кластерных центров. Данный алгоритм итерационно вычисляет центры кластеров и новые степени принадлежности объектов.

Для заданного числа K входных векторов $x_k\left(k = \overline{1,K}\right)$ и N выделяемых кластеров $c_j\left(j = \overline{1,N}\right)$ предполагается, что любой x_k принадлежит любому $c_j\left(j = \overline{1,N}\right)$ с принадлежностью $\mu_{jk} \in [0,1]$, где j – номер кластера, а k – номер входного вектора.

Принимаются во внимание следующие условия нормирования для μ_{jk}:

$$\begin{gathered}\sum_{j=1}^{N} \mu_{jk} = 1, \forall k = 1, \dots, K; \\ 0 < \sum_{j=1}^{N} \mu_{jk} \le K, \forall j = 1, \dots, N.\end{gathered} \tag{6}$$

Цель алгоритма – минимизация суммы всех взвешенных расстояний $\|x_k - c_j\|$.

$$\sum_{j=1}^{N}\sum_{k=1}^{K} (\mu_{jk})^q \|x_k - c_j\| \to min, \tag{7}$$

где q – фиксированный параметр, задаваемый перед итерациями.

Для достижения вышеуказанной цели необходимо решить следующую систему уравнений:

$$\begin{gathered}{}^{\delta}/_{\delta\mu_{jk}} \left(\sum_{j=1}^{N}\sum_{k=1}^{K} (\mu_{jk})^q \|x_k - c_j\| \right) = 0, j = \overline{1,N}, k = \overline{1,K}, \\ {}^{\delta}/_{\delta c_j} \left(\sum_{j=1}^{N}\sum_{k=1}^{K} (\mu_{jk})^q \|x_k - c_j\| \right) = 0, j = \overline{1,N}.\end{gathered} \tag{8}$$

Совместно с условиями нормирования μ_{jk} данная система уравнений имеет следующее решение:

$$c_j = \frac{\sum_{j=1}^{N}(\mu_{jk})^{q*x_k}}{\sum_{k=1}^{K}(\mu_{jk})^{q}}, j = \overline{1,N};$$

$$\mu_{jk} = \frac{1/\|x_k - c_j\|^{1/(q-1)}}{\sum_{j=1}^{N}\left(1/\|x_k - c_j\|^{1/(q-1)}\right)}, \qquad j = \overline{1,N}, k = \overline{1,K}, \tag{9}$$

где c_j - (взвешенный центр гравитации).

Алгоритм нечеткой кластеризации выполняется по шагам.

Шаг 1. Инициализация.

Выбираются следующие параметры:

– необходимое количество кластеров N, $N < 2 < K$;

– тип расстояний (например, расстояние по Евклиду);

– фиксированный параметр q (обычно 1,5);

– начальная (на нулевой итерации) матрица функций принадлежности $U^{(0)} = (\mu_{jk})^{(0)}$объектов $x_k(k = \overline{1,K})$с учетом заданных начальных центров кластеров $c_j(j = \overline{1,N})$.

Шаг 2. Регулирование позиций $c_j^{(t)}$ центров кластеров.

На *t*-м итерационном шаге при известной матрице $\mu_{jk}^{(t)}$ вычисляется $c_j^{(t)}$в соответствии с вышеприведенным решением системы уравнений.

Шаг 3. Корректировка значений принадлежности μ_{jk}.

Учитывая известные $c_j^{(t)}$, вычисляются $\mu_{jk}^{(t)}$, если $x_k \neq c_j$, в противном случае:

$$\mu_{jk}^{(t+1)} = \begin{cases} 1, \text{если } k = j, \\ 0, \text{если } k \neq j. \end{cases} \tag{10}$$

Шаг 4. Остановка алгоритма.

Алгоритм нечёткой кластеризации останавливается при выполнении следующего условия:

$$\left\|U^{(t+1)} - U^{(t)}\right\| \leq \varepsilon, \tag{11}$$

где $\| \|$ – матричная норма (например, евклидова норма); ε – заранее задаваемый уровень точности.

Решение задачи нечёткой кластеризации в MATLAB

Существуют два способа решения задач кластеризации в MATLAB: с использованием командной строки или графического интерфейса пользователя. Рассмотрим первый из указанных способов.

Для нахождения центров кластеров в MATLAB имеется встроенная функция **fcm**, описание которой представлено ниже.

Описание функции: **[center, U, obj_fcn] = fcm(data, cluster_n).**

Аргументами данной функции являются:

1) **data** – множество данных, подлежащих кластеризации, каждая строка описывает точку в многомерном пространстве характеристик;
2) **cluster_n** – количество кластеров (более одного).

Функцией возвращаются следующие параметры:

1) **center** – матрица центров кластеров, каждая строка которой содержит координаты центра отдельного кластера;
2) **U** – результирующая матрица функции принадлежности;
3) **obj_fcn** – значение целевой функции на каждой итерации.

Пример 1. Программа нечеткой кластеризации.

```
loadfcmdata.dat; %загрузка данных, подлежащих кластеризации, из файла
[center, U, obj_fcn] = fcm(fcmdata, 2);% определение центра кластеризации
(два кластера)
maxU = max(U);% определение максимальной степени принадлежности
отдельного элемента данных кластеру
index1 = find (U(1, :) == maxU);% распределение строк матрицы данных
между соответствующими кластерами
index2 = find(U(2, :) == maxU);
plot (fcmdata (index1, 1), fcmdata (index1, 2),' ko', 'markersize', 5,
'LineWidth' ,1); % построение данных, соответствующих первому кластеру
holdon
plot(fcmdata (index2, 1), fcmdata(index2, 2), 'kx', 'markersize', 5,
'LineWidth', 1);% построение данных, соответствующих второму кластеру
plot(center(1, 1), center(1, 2), 'ko', 'markersize', 15, 'LineWidth', 2)
%построение кластерных центров
plot (center (2, 1), center (2, 2), 'kx', 'markersize', 15, 'LineWidth',
2) %построение кластерных центров
```

На рис. 1 представлено множество данных, подлежащих кластеризации, и найденные центры кластеров для примера 1.

Функция **fcm** выполняется итерационно до тех пор, пока изменения целевой функции превышают некоторый заданный порог.

На каждом шаге в командном окне MATLAB выводятся порядковый номер итерации и соответствующее текущее значение целевой функции (табл. 1).

Таблица 1 – Изменение целевой функции.

Номер итерации	Значения целевой функции	Номер итерации	Значения целевой функции
1	8,94	7	3,81
2	7,31	8	3,80
3	6,90	9	3,79
4	5,41	10	3,79
5	4,08	11	3,79
6	3,83	12	3,78

Для оценки динамики изменения значений целевой функции используется команда построения графика **plot (obj_fcm)**. Результаты примера 1 показаны на рис. 2.

Функцию кластеризации можно вызвать с дополнительным набором параметров: **fcm(data,cluster_n,options)**. Дополнительные аргументы используются для управления процессом кластеризации:

- **options**(1) – показатель степени для матрицы *U* (по умолчанию: 2.0);
- **options**(2) – максимальное количество итераций (по умолчанию: 100);
- **options**(3) – минимально допустимое изменение значений целевой функции (по умолчанию: 1*e*–5);

— **options**(4) – отображение информации на каждом шаге (по умолчанию: 1).

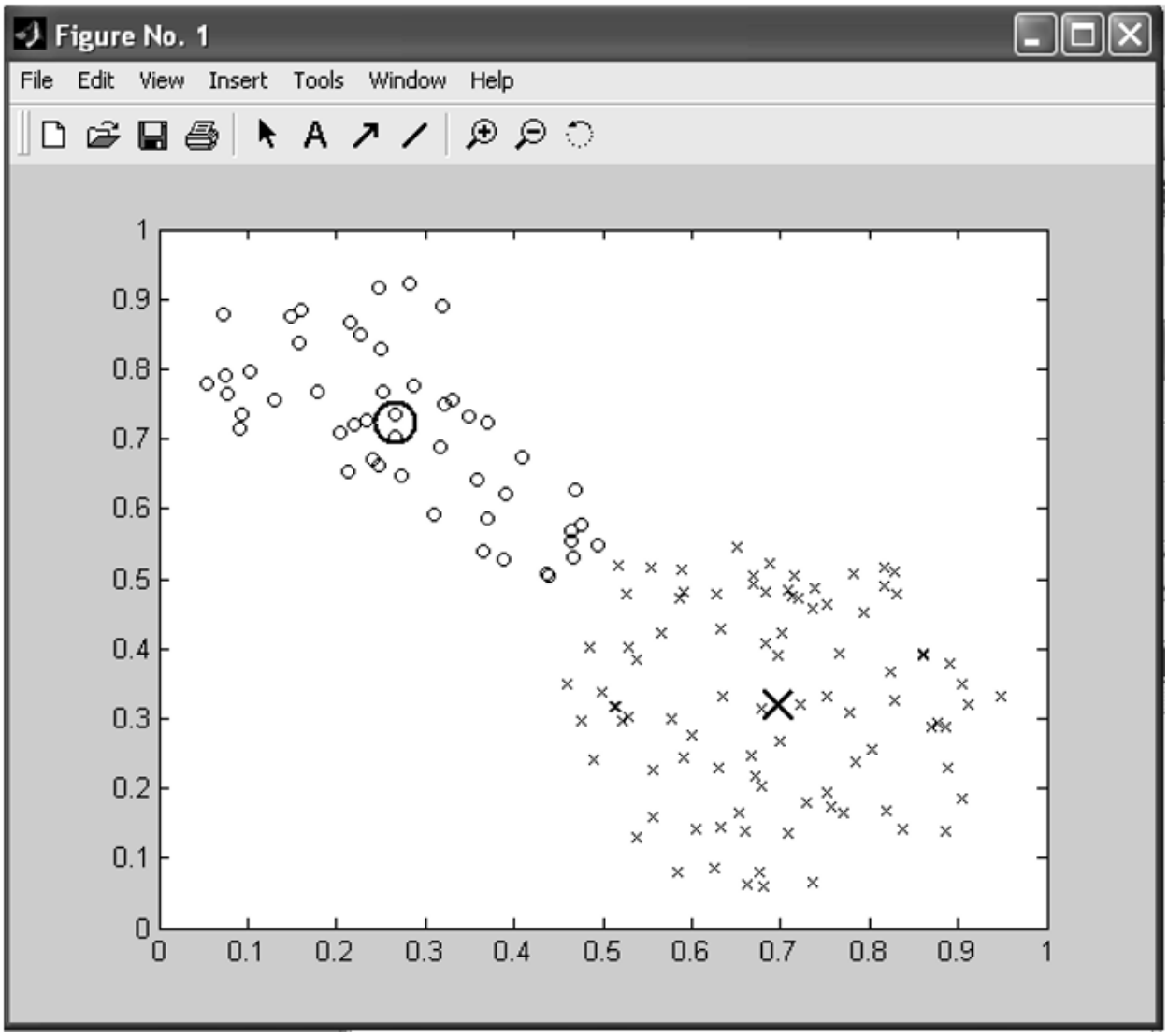


Рис. 1. Множество анализируемых данных и центры кластеров

Пример определения функции **fcm** с дополнительными параметрами: **[center,U,obj_fcn] = fcm(fcmdata, 2, [2,100,le-5, 1])**.

Второй способ решения задач кластеризации в MATLAB вызывается командой **findcluster**. Главное окно инструмента кластеризации показано на рис. 3

Кнопка <**LoadData**> используется для загрузки подлежащих кластеризации сходных данных следующего формата: каждая строка представляет собой точку в многомерном пространстве характеристик, количество строк соответствует количеству точек (элементов данных). Графическую интерпретацию исходных данных можно наблюдать в одноименном окне главного окна инструмента.

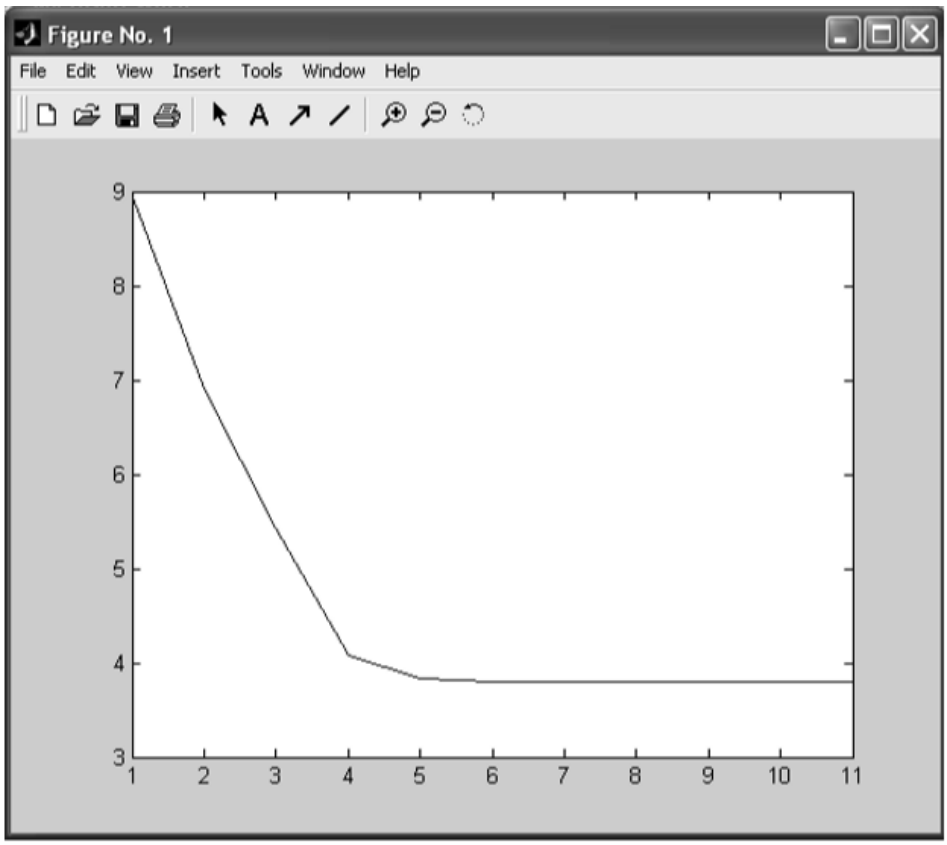


Рис. 2. График изменения значений целевой функции

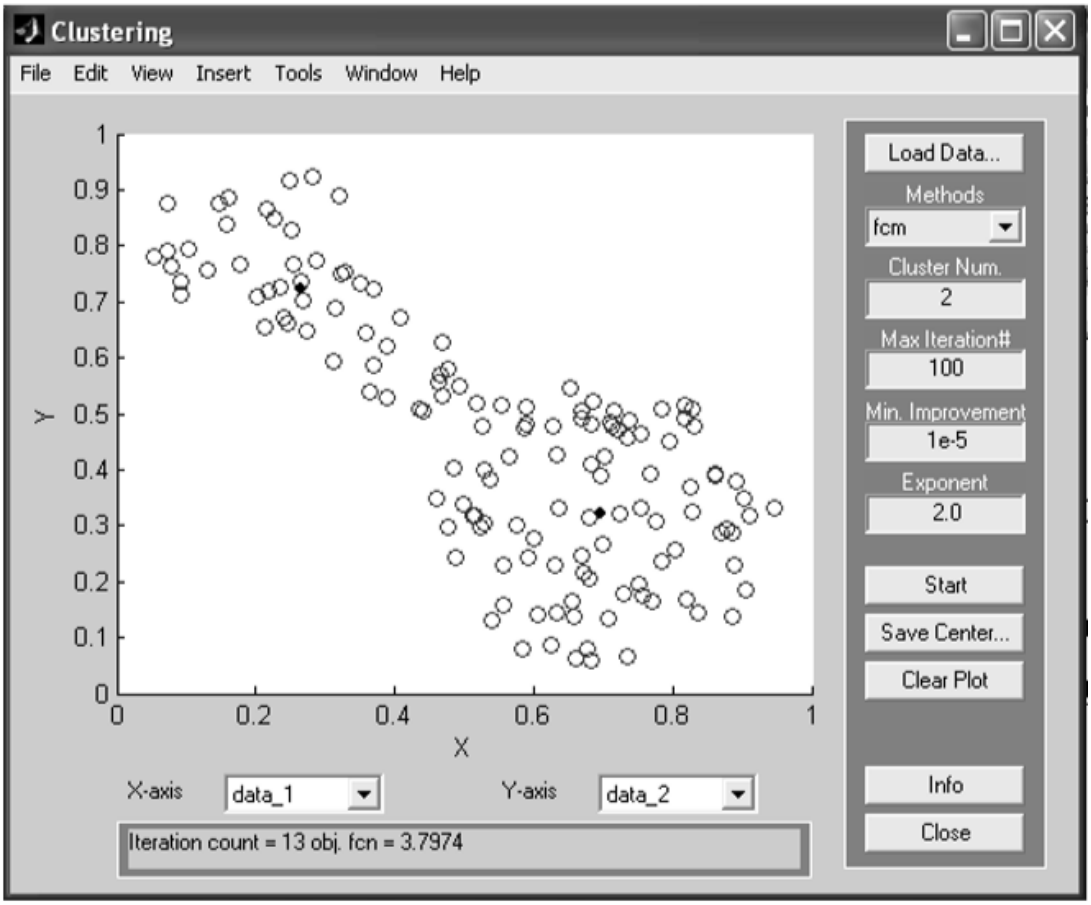


Рис. 3. Главное окно кластеризации в MATLAB

Выбор типа алгоритма кластеризации осуществляется с использованием ниспадающего меню **Methods** (пункт меню – **fcm**). Далее определяются параметры алгоритма кластеризации:

- количество кластеров (строка ввода – **Cluster Num**);
- максимальное количество итераций (строка ввода – **Max Iteration**);
- минимальное значение улучшения целевой функции (строка ввода – **Min.Improvement**);
- показатель степени при матрице ФП (строка ввода – **Exponent**).

После определения необходимых значений указанных параметров осуществляется запуск алгоритма кластеризации с помощью кнопки <**Start**>.

Количество произведенных итераций и значение целевой функции можно просмотреть в нижней части главного окна инструмента кластеризации.

Самоорганизующиеся карты Кохонена

Самоорганизующаяся карта Кохонена (Self-organizing map, SOM) представляет из себя вычислительный метод, предназначенный для задач кластеризации и визуализации, а также анализа данных из пространств высокой размерности, полученных экспериментально. Идея сети Кохонена принадлежит финскому ученому Тойво Кохонену (1982 год). Прародителями модели его самоорганизующейся сети были ранние нейросетевые модели.

Целью применения данного метода является поиск скрытых закономерностей в данных, основанный на снижении размерности исходного пространства в пространство меньшей размерности. При этом топология исходного пространства не меняется. В результате обучения данной модели получается решётка, состоящая из обученных нейронов, она же и называется картой исходного пространства.

Архитектура самоорганизующейся карты Кохонена следующая: имеется два слоя нейронов – входной (распределительный) и выходной (слой Кохонена). При этом нейроны второго слоя расположены в виде двумерной решётки (квадратная либо шестиугольная сетка) так, что каждый нейрон из первого слоя соединён с каждым нейроном второго слоя (рис. 4).

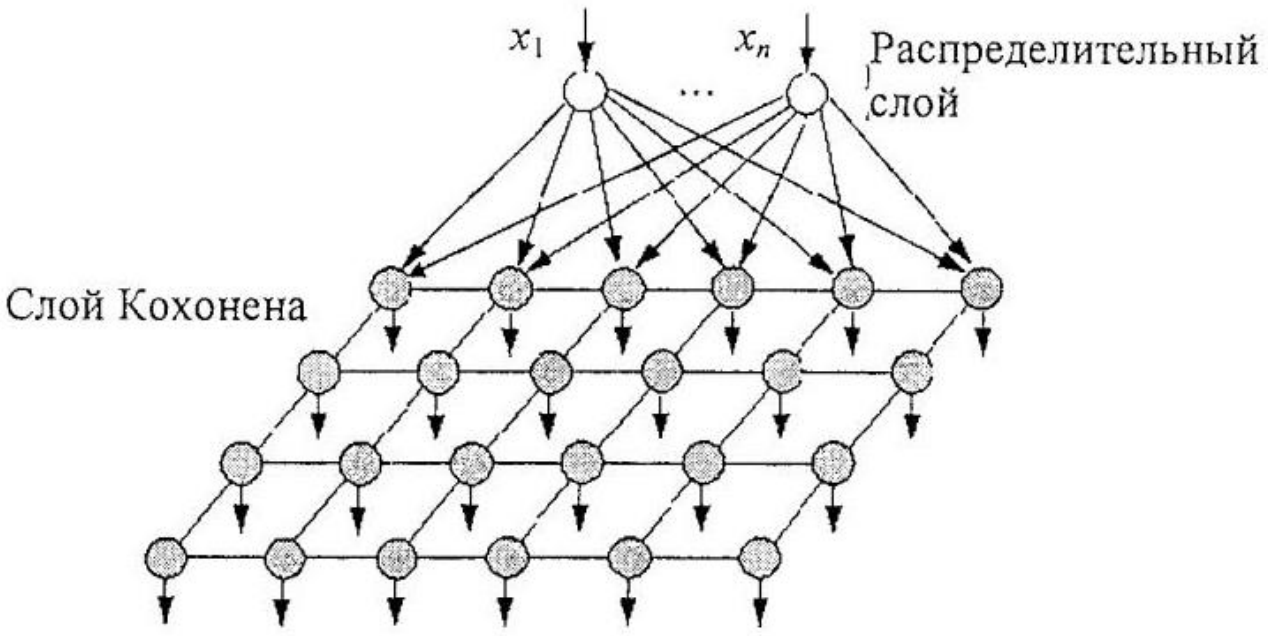
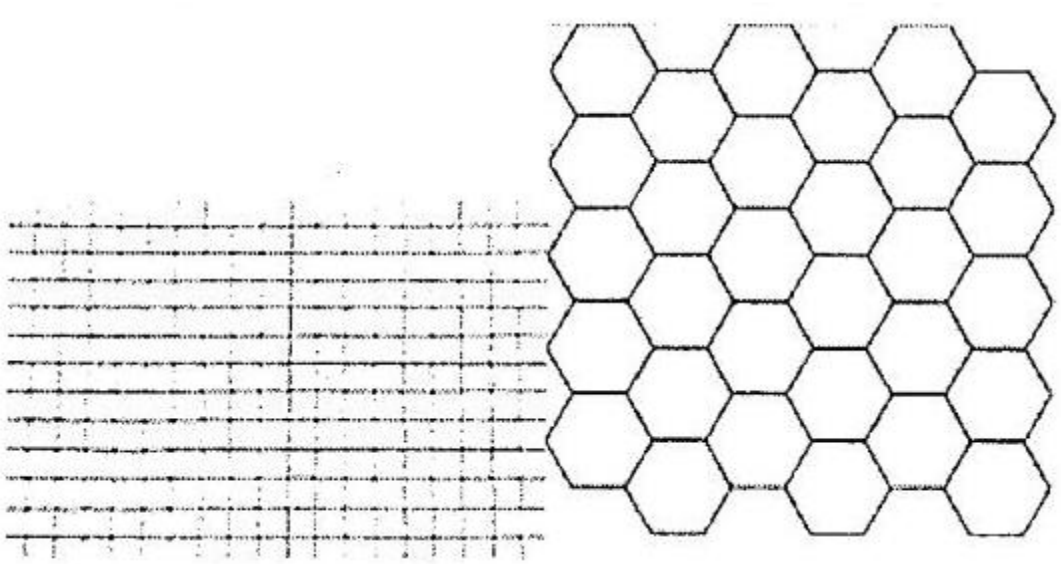


Рис. 4. Общая структура карты Кохонена

Количество нейронов входного слоя равно размерности исходного пространства. Часто нейроны слоя Кохонена называют кластерными элементами. Более точно их количество определяет количество кластеров, на которые карта может разбить данные, подающиеся на распределительный слой. Чем больше кластерных элементов, тем более гранулярная кластеризация. Как уже было сказано, топология слоя Кохонена может быть представлена в виде либо четырёхугольной, либо шестиугольной сетки. Второй вариант более привлекателен в силу того, что расстояния для каждого нейрона до каждого соседнего с ним одинаковые. Более наглядно это видно на рисунках (рис. 5 (а),(б)).



(а) (б)

Рис. 5. Четырёхугольная (а) и шестиугольная (б) решётки Кохонена

Каждый нейрон выходного слоя определяется вектором весов (размерность вектора есть размерность входного пространства) и упорядоченной парой (x, y), определяющей позицию нейрона на карте Кохонена.

Алгоритм обучения нейросети

Сеть Кохонена обучается методом последовательных приближений. В процессе обучения таких сетей на входы подаются данные, но сеть при этом подстраивается не под эталонное значение выхода, а под закономерности во входных данных. Начинается обучение с выбранного случайным образом выходного расположения центров.

В процессе последовательной подачи на вход сети обучающих примеров определяется наиболее схожий нейрон (тот, у которого скалярное произведение весов и поданного на вход вектора минимально). Этот нейрон объявляется победителем и является центром при подстройке весов у соседних нейронов. Такое правило обучения предполагает «соревновательное» обучение с учетом расстояния нейронов от «нейрона-победителя».

Обучение при этом заключается не в минимизации ошибки, а в подстройке весов (внутренних параметров нейронной сети) для наибольшего совпадения с входными данными.

Основной итерационный алгоритм Кохонена последовательно проходит ряд эпох, на каждой из которых обрабатывается один пример из обучающей выборки. Входные сигналы последовательно предъявляются сети, при этом желаемые выходные сигналы не определяются. После предъявления достаточного числа входных векторов синаптические веса сети становятся способны определить кластеры. Веса организуются так, что топологически близкие узлы чувствительны к похожим входным сигналам.

В результате работы алгоритма центр кластера устанавливается в определенной

позиции, удовлетворительным образом кластеризующей примеры, для которых данный нейрон является «победителем». В результате обучения сети необходимо определить меру соседства нейронов, т.е. окрестность «нейрона-победителя». Окрестность представляет собой несколько нейронов, которые окружают «нейрон-победитель».

Сначала к окрестности принадлежит большое число нейронов, далее ее размер постепенно уменьшается. Сеть формирует топологическую структуру, в которой похожие примеры образуют группы примеров, близко находящиеся на топологической карте.

Кохонен существенно упростил решение задачи, выделяя из всех нейронов слоя лишь один *c*-й нейрон, для которого взвешенная сумма входных сигналов минимальна:

$$c = \underset{i}{\operatorname{argmax}}(x^T w_j). \tag{12}$$

Отметим, что весьма полезной операцией предварительной обработки входных векторов является их нормализация:

$$\bar{x}_i = \frac{x_i}{\|x\|}, i = 1, \dots, N, \tag{13}$$

превращающая векторы входных сигналов в единичные с тем же направлением.

$$\|x\| = \left(\sum_{i=1}^{N} x_i^2 \right)^{1/2}. \tag{14}$$

В этом случае вследствие того, что сумма весов каждого нейрона одного слоя $\sum_i w_{ij}$ для всех нейронов этого слоя одинакова и $x = 1$, условие (12) эквивалентно условию:

$$c = \underset{i}{\operatorname{argmax}}(x - w_j). \tag{15}$$

Таким образом, будет активирован только тот нейрон, вектор весов которого *w* наиболее близок к входному вектору *x*. А так как перед началом обучения неизвестно, какой именно нейрон будет активироваться при предъявлении сети конкретного входного вектора, сеть обучается без учителя, т.е. самообучается.

Вводя потенциальную функцию f_{ij} – функцию расстояния (соседства) между *i*–м и *j*–м нейронами с местоположениями r_i и r_j соответственно, монотонно убывающую с увеличением расстояния между этими нейронами, Кохонен предложил следующий алгоритм коррекции весов:

$$w_{ij}(k+1) = w_{ij}(k) + \alpha(k) f_{ij}(k) \Big(x(k) - w_{ij}(k) \Big), \tag{16}$$

где $\alpha(k) \in [0, 1]$ – изменяющийся во времени коэффициент усиления (обычно выбирают $\alpha = 1$ на первой итерации, постепенно уменьшая в процессе обучения до нуля); $f_{ij}(k)$ – монотонно убывающая функция

$$f_{ij}(k) = f\big(\|r_i - r_j\|, k\big) = f(d, k) = f(d, \delta), \tag{17}$$

где r_i и r_j - векторы, определяющие положение нейронов *i* и *j* в решётке.

При принятой метрике $d = \|r_i - r_j\|$ функция $f_{ij}(k)$ с ростом времени k стремится к нулю. На практике вместо параметра времени k используют параметр расстояния δ, задающий величину области соседства и уменьшающийся с течением времен до нуля.

Выбор функции $f_{ij}(k)$ также влияет на величины весов всех нейронов в слое. Очевидно, что для «нейрона-победителя» c:

$$f_c(\|r_i - r_j\|) = f_c(0) = 1. \quad (18)$$

На рис. 6 показан пример изменения двумерных весов карты $w_j = (w_{j1}, w_{j2})^T$, образующей цепь. При появлении входного образа x наиболее сильно изменяется весовой вектор «нейрона-победителя» 5, менее сильно – веся расположенных рядом с ним нейронов 3, 4, 6, 7. Нейроны 1, 2, 8, 9 лежат вне области соседства, их весовые коэффициенты не изменяются.

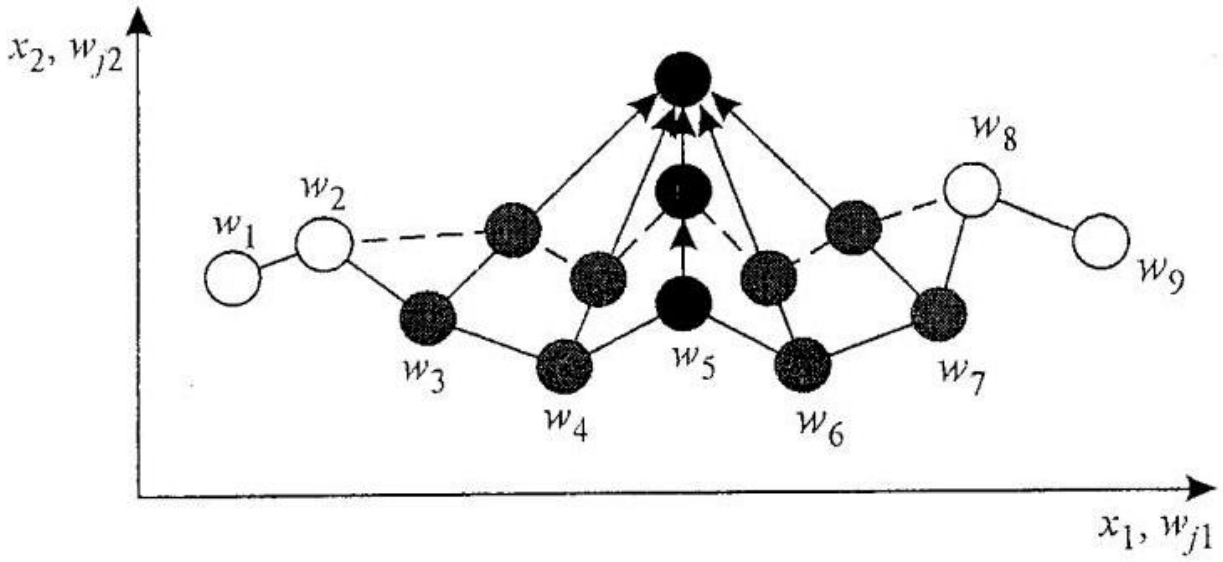


Рис. 6. Изменение весов карты Кохонена

Таким образом, алгоритм обучения сети Кохонена может быть описан способом, представленным далее.

Шаг 1. Инициализация.

Весовым коэффициентам всех нейронов присваиваются малые случайные значения и осуществляется их нормализация. Выбирается соответствующая потенциальная функция $f_{ij}(d)$ и назначается начальное значение коэффициента усиления α_0.

Шаг 2. Выбор обучающего сигнала.

Из всего множества векторов, обучающих входных сигналов в соответствии с функцией распределения $P(x)$, выбирается один вектор x, который представляет «сенсорный сигнал», предъявляемый сети.

Шаг 3. Анализ отклика (выбор нейрона).

По формуле (12) определяется активированный нейрон.

Шаг 4. Процесс обучения.

В соответствии с алгоритмом (16) изменяются весовые коэффициенты активированного нейрона и соседних с ним до тех пор, пока не будет получено требуемое значение критерия качества обучения или не будет предъявлено заданное число обучающих входных векторов. Окончательное значение весовых коэффициентов совпадает с нормализованными векторами входов.

Поскольку сеть Кохонена осуществляет проецирование n–мерного пространства образов на m–мерную сеть, анализ сходимости алгоритма обучения представляет собой довольно сложную задачу.

Если бы с каждым нейроном слоя ассоциировался один входной вектор, то вес

любого нейрона слоя Кохонена мог бы быть получен с помощью одного вычисления, так как вес «нейрона-победителя» корректировался бы с $\alpha = 1$. Однако обычное обучающее множество включает много схожих между собой входных векторов, и сеть Кохонена должна быть обучена активировать один и тот же нейрон для каждого из них. Это достигается усреднением входных векторов путём уменьшения величины, а не при предъявлении каждого следующего входного сигнала. Таким образом, веса, ассоциированные с нейроном, усредняются и принимают значение вблизи центра входных сигналов, для которых данный нейрон является «победителем».

Особенности и проблемы карт Кохонена

Уникальность метода самоорганизующихся карт состоит в преобразовании *n*-мерного пространства в двумерное. Объекты (векторы признаков которых близки) попадают в одну ячейку или в ячейки, расположенные на карте вблизи. Следовательно, двумерная карта Кохонена отражает на плоскости близость многомерных векторов признаков. Обычно требуется анализировать, по каким конкретно параметрам проявляется сходство объектов. Для этого применяется раскраска карт Кохонена. Необходимо создать столько карт, сколько параметров (компонентов входных векторов) анализируется. Каждая карта соответствует одному параметру объекта. Ячейки карты раскрашиваются в разные цвета (или оттенки серого цвета) в зависимости от значения весов нейронов, соответствующих каждой ячейке. Выделяются диапазоны значений весов. Каждому диапазону ставится в соответствие цвет (или оттенок серого), и ячейки карты «раскрашиваются» соответствующими цветами.

В результате работы алгоритма могут быть получены следующие карты, перечисленные далее.

Карта входов нейронов.

Веса нейронов подстраиваются под значения входных переменных и отображают их внутреннюю структуру. Для каждого входа рисуется своя карта, раскрашенная в соответствии со значением конкретного веса нейрона.

При анализе данных используют несколько карт входов.

На одной из карт выделяют область определенного цвета - это означает, что соответствующие входные примеры имеют приблизительно одинаковое значение соответствующего входа. Цветовое распределение нейронов из этой области анализируется на других картах для определения схожих или отличительных характеристик. Пример рассмотренных карт входов будет приведен ниже.

Карта выходов нейронов.

На карту выходов нейронов проецируется взаимное расположение исследуемых входных данных. Нейроны с одинаковыми значениями выходов образуют кластеры - замкнутые области на карте, которые включают нейроны с одинаковыми значениями выходов.

Специальные карты.

Это карта кластеров, матрица расстояний, матрица плотности попадания и другие карты, которые характеризуют кластеры, полученные в результате обучения сети Кохонена.

Координаты каждой карты определяют положение одного нейрона. Так, координаты [15:30] определяют нейрон, который находится на пересечении 15-го столбца с 30-м рядом в матрице нейронов. Рассмотрим, что же представляют собой эти карты.

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.

Полученную карту можно использовать как средство визуализации при анализе данных. В результате обучения карта Кохонена классифицирует входные примеры на кластеры (группы схожих примеров) и визуально отображает многомерные входные данные на плоскости нейронов.

При реализации сети Кохонена возникают проблемы, описанные далее.

Выбор коэффициента обучения α.

Этот выбор влияет как на скорость обучения, так и на устойчивость получаемого решения. Очевидно, что процесс обучения ускоряется (скорость сходимости алгоритма обучения увеличивается) при выборе α близким к единице. Однако в этом случае предъявление сети различных входных векторов, относящихся к одному классу, приведет к изменениям соответствующего вектора весовых коэффициентов. Напротив, при $\alpha \to 0$ скорость обучения будет медленной, однако вектор весовых коэффициентов, достигнув центра класса, при подаче на вход сети различных сигналов, относящихся к одному классу, будет оставаться вблизи этого центра. Поэтому одним из путей ускорения процесса обучения при одновременном обеспечении получения устойчивого решения является выбор переменного α с $\alpha \to 1$ на начальных этапах обучения и $\alpha \to 0$ - на заключительных. К сожалению, такой подход не применим в тех случаях, когда сеть должна непрерывно подстраиваться к предъявляемым ей новым входным сигналам.

Рандомизация весов.

Рандомизация весов слоя Кохонена может породить серьезные проблемы при обучении, так как в результате этой операции весовые векторы распределяются равномерно по поверхности гиперсферы. Как правило, входные векторы распределены неравномерно и группируются на относительно малой части поверхности гиперсферы. Поэтому большинство весовых векторов окажутся настолько удаленными от любого входного вектора, что не будут активированы и станут бесполезными. Более того, оставшихся активированных нейронов может оказаться слишком мало, чтобы разбить близко расположенные входные векторы на кластеры.

Выбор начальных значений векторов весовых коэффициентов и нейронов.

Если начальные значения выбраны неудачно, т. е. расположенными далеко от предъявляемых входных векторов, то нейрон не окажется победителем ни при каких входных сигналах, а, следовательно, не обучится.

Выбор параметра расстояния δ.

Если сначала параметр δ выбран малым или очень быстро уменьшается, то далеко расположенные друг от друга нейроны не могут влиять друг на друга. Хотя две части в такой карте настраиваются правильно, общая карта будет иметь топологический дефект (рис. 7).

Количество нейронов в слое.

Число нейронов в слое Кохонена должно соответствовать числу классов входных сигналов. Это может быть недопустимо в тех задачах, когда число классов заранее известно.

Классы входных сигналов.

Слой Кохонена может формировать только классы, представляющие собой выпуклые области входного пространства.

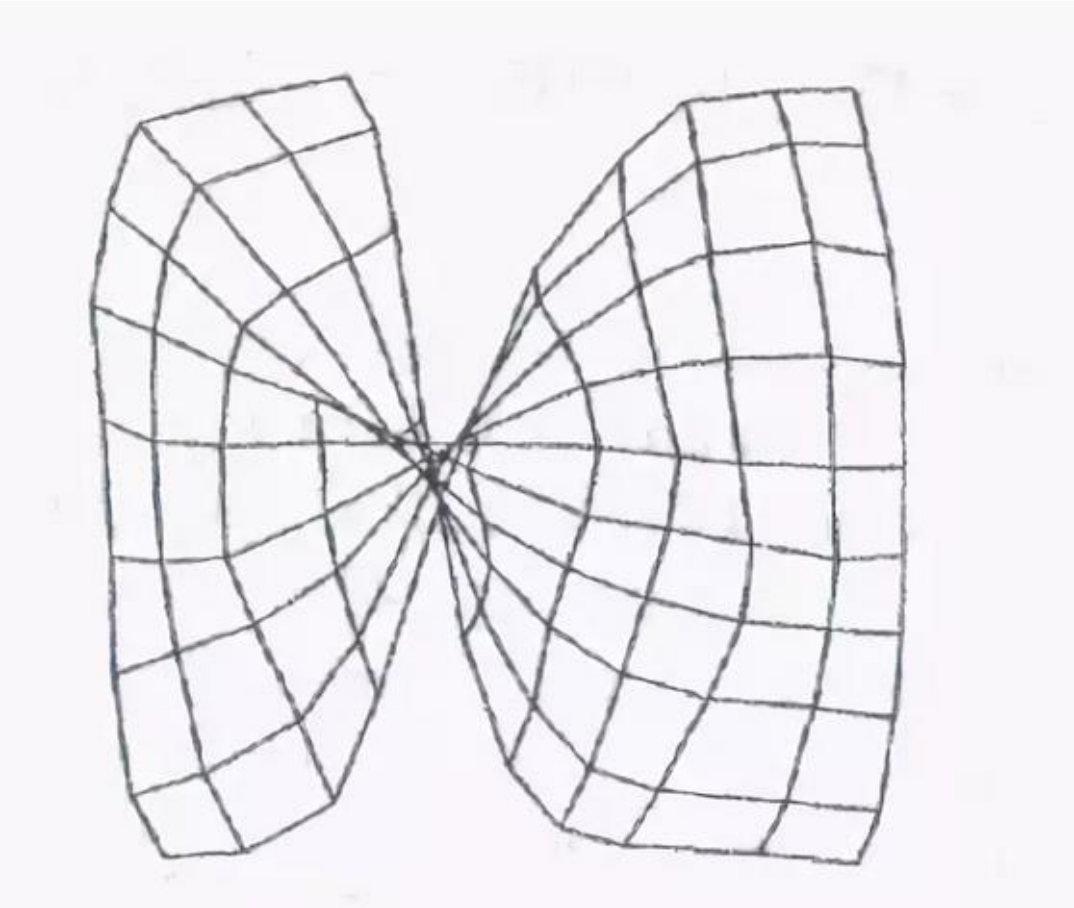


Рис. 7. Топологический дефект карты Кохонена

Задание

1. Ознакомиться с нечёткой кластеризацией методом C-средних.
2. Построить двумерный график распределения объектов по классам в зависимости от двух признаков (координат) $x1$, $x2$ и трёхмерный график – для трёх признаков (координат) – $x1$, $x2$, $x3$. Данные сгенерировать самостоятельно, вызывать их из файла .dat или из переменной рабочего пространства.
3. Провести кластеризацию сгенерированных данных с использованием метода C-средних. Построить графики (объекты одного кластера отмечать одинаковыми маркерами или цветами, построить также центры кластеров).
4. Исследовать зависимость качества кластеризации от количества итераций алгоритма.
5. Ознакомиться со средствами построения карт Кохонена с помощью MATLAB (функция newc).
6. Построить нейронную сеть со слоем Кохонена, которая разделяет входные данные на кластеры и ищет их центры. Для обучения использовать случайные трёхмерные векторы. Построить график исходных данных и выявленных центров.

Контрольные вопросы

1. Что такое кластерный анализ?
2. Какие метрики определения расстояния между объектами вы знаете?
3. Опишите основные шаги алгоритма Fuzzy C-means.

4. Назовите достоинства и недостатки алгоритма Fuzzy C-means.
5. Для каких целей используются самоорганизующиеся карты Кохонена?
6. В чём заключается уникальность метода самоорганизующихся карт?
7. В чём заключается проблема рандомизации весов?