СЕТИ КОХОНЕНА

Горбаченко В. И.

Рассматриваются архитектура и алгоритмы обучения наиболее известных разновидностей сетей Кохонена: слои (сети) Кохонена, реализующие кластеризацию входных векторов, самоорганизующиеся карты Кохонена (*SOM — self-organizing maps*), реализующие визуализацию многомерных данных на плоскости, сети векторного квантования, обучаемые с учителем (*LVQ — learning vector quantization*), сети встречного распространения.

1. Сети Кохонена

1.1. Принципы построения сети Кохонена

Сети (слои) Кохонена (*Kohonen T.*) [1–3] относятся к *самоорганизующимся* нейронным сетям. Самоорганизующаяся сеть позволяет выявлять *кластеры* (группы) входных векторов, обладающих некоторыми общими свойствами.

Кластеризация — это разделение исследуемого множества объектов на группы "похожих" объектов, называемых *кластерами* [4–5]. Синонимами термина "кластер" (англ. *Cluster* — сгусток, пучок, группа) являются термины класс, таксон, сгущение. Задача кластеризации принципиально отличается от задачи классификации. Решением задачи *классификации* является отнесение каждого из объектов к одному из *заранее определенных* классов. В задаче кластеризации происходит отнесение объекта к одному из *заранее неопределенных* классов. Разбиение объектов по кластерам осуществляется *при одновременном формировании кластеров*.

Кластеризация позволяет сгруппировать сходные данные, что облегчает решение ряда задач Data Mining [4]:

- *Изучение данных, облегчение анализа*. Содержательный анализ полученных кластеров позволяет обнаружить закономерности. Например, можно выявить группы клиентов сети сотовой связи, для которых можно предложить новый тарифный план. Другие примеры — выявление групп покупателей торговой сети, сегментация рынка. Анализ содержания кластера позволяет применить к объектам различных кластеров разные методы анализа.
- *Прогнозирование*. Относя новый объект к одному из кластеров, можно прогнозировать поведение объекта, поскольку его поведение будет схожим с поведением объектов кластера.
- *Обнаружение аномалий*. Содержательный анализ кластеров помогает выявить аномалии. Обычно, это кластеры, в которые попадает мало объектов.

Важно отметить роль *содержательной интерпретации каждого кластера*. Каждому кластеру необходимо присвоить содержательное название, отражающее суть объектов кластера. Для этого необходимо выявить, признаки, объединяющие объекты в кластер. Это может потребовать статистического анализа свойств объекта кластера.

С помощью сетей Кохонена производится кластеризация объектов, описываемых количественными характеристиками.

Формально *задача кластеризации* описывается следующим образом [6]. Дано множество объектов $I = \{i_1, i_2, \ldots, i_n\}$, каждый из которых характеризуется вектором $\mathbf{x}_j$, $j = 1,2,\ldots,n$ атрибутов (параметров): $\mathbf{x}_j = \{x_{j1}, x_{j2}, \ldots, x_{jm}\}$. Требуется построить множество кластеров C и отображение F множества I на множество C, то есть $F : I \to C$. Задача кластеризации состоит в построении множества

$$C = \{c_1, c_2, \ldots, c_k, \ \ldots, c_g\},$$

где c_k — *кластер*, содержащий "похожие" объекты из множества I:

$$c_k = \left\{ i_j, i_p \middle| i_j \in I, i_p \in I \text{ и } d\left(i_j, i_p\right) < \sigma \right\}, \qquad (1)$$

σ — величина, определяющая меру близости для включения объектов в один кластер, $d\left(i_j, i_p\right)$ — мера близости между объектами, называемая *расстоянием*.

Если расстояние $d\left(i_j, i_p\right)$ меньше некоторого значения σ, то объекты считаются близкими и помещаются в один кластер. В противном случае считается, что объекты отличны друг от друга и их помещают в разные кластеры. Условие (1) известно как *гипотеза компактности*.

Кластеризация основана на использовании расстояния между векторами [3, 5]. Неотрицательное число $d\left(\mathbf{x}, \mathbf{y}\right)$ называется *расстоянием* (*метрикой*) между векторами $\mathbf{x}$ и $\mathbf{y}$, если выполняются следующие условия:

1. $d\left(\mathbf{x}, \mathbf{y}\right) \geq 0$ для всех $\mathbf{x}$ и $\mathbf{y}$.
2. $d\left(\mathbf{x}, \mathbf{y}\right) = 0$, тогда и только тогда, когда $\mathbf{x} = \mathbf{y}$.
3. $d\left(\mathbf{x}, \mathbf{y}\right) = d\left(\mathbf{y}, \mathbf{x}\right)$.
4. $d\left(\mathbf{x}, \mathbf{y}\right) \leq d\left(\mathbf{x}, \mathbf{k}\right) + d\left(\mathbf{k}, \mathbf{y}\right)$ — неравенство треугольника.

В сетях Кохонена обычно применяется *евклидово расстояние*

$$d_E\left(\mathbf{x}, \mathbf{y}\right) = \sqrt{\sum_{i=1}^{m} \left(x_i - y_i\right)^2} = \left\| \mathbf{x} - \mathbf{y} \right\|.$$

Евклидово расстояние между векторами $\mathbf{x}$ и $\mathbf{y}$ представляет собой евклидову норму разности векторов, или длину отрезка, соединяющего точки $\mathbf{x}$ и $\mathbf{y}$.

Евклидово расстояние является частным случаем *расстояния Минковского* (*H. Minkowski*)

$$d_p\left(\mathbf{x}, \mathbf{y}\right) = \left(\sum_{i=1}^{m} \left| x_i - y_i \right|^p \right)^{1/p} = \left\| \mathbf{x} - \mathbf{y} \right\|_p,$$

где $\left\| \mathbf{z} \right\|_p = \left(\sum_{i=1}^{m} \left| z_i \right|^p \right)^{1/p}$ — p-норма вектора $\mathbf{z}$.

Тогда 2-норма — это евклидова норма.

Другой частный случай — 1-норма, которая называется *манхэттенским расстоянием* (расстоянием городских кварталов)

$$d_1(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m} |x_i - y_i|.$$

Манхеттенское расстояние — это расстояние, которое мы проходим, двигаясь параллельно осям координат, как в Манхеттнене или других городах с прямоугольной продольно-поперечной планировкой улиц. Известны и другие виды расстояний [5].

Сеть (*слой*) *Кохонена* (рис. 1) — это однослойная сеть, построенная из нейронов типа WTA (*Winner Takes All* — победитель получает все) — см. раздел 2.6.

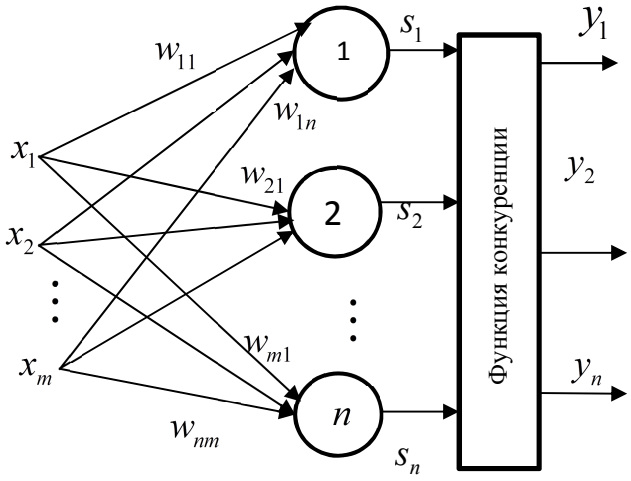

Рис. 1. Структура сети Кохонена

Каждый нейрон сети соединен со всеми компонентами m-мерного входного вектора $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$. Входной вектор — это описание одного из объектов, подлежащих кластеризации. Количество нейронов совпадает с количеством кластеров, которое должна выделить сеть. В качестве нейронов сети Кохонена применяются линейные взвешенные сумматоры

$$s_j = b_j + \sum_{i=1}^{m} w_{ij} x_i ,$$

где j — номер нейрона, i — номер входа, s_j — выход адаптивного сумматора, w_{ij} — вес i-го входа j-го нейрона, b_j — порог.

Каждый j-ый нейрон описывается вектором весов $\mathbf{w}_j = \left(w_{1j}, w_{2j}, \ldots, w_{mj}\right)$, где m – число компонентов входных векторов. С выходов адаптивных сумматоров сигнал поступает на функцию конкуренции, работающую по правилу "победитель получает всё". Функция конкуренции находит выход адаптивный сумматор с максимальным значением выхода. Пусть k — номер такого сумматора. Тогда на выходе сети формируется выходной сигнал $y_k = 1$, остальные выходные сигналы равны нулю. Если максимум достигается одновременно на выходах нескольких сумматоров, то выходной сигнал, равный единице, соответствует одному из них, например, первому.

Обучение сети Кохонена представляет собой подбор значений весов, минимизирующих ошибки от замены близких в смысле используемой метрики входных векторов вектором весов. Такой подход называется векторным квантованием [7] и применяется в задачах сжатия аудио- и видеосигналов. Идея векторного квантования состоит в компактном представлении многомерных входных векторов с помощью ограниченного набора *опорных векторов* меньшей размерности, образующих *кодовую таблицу*. В случае сети Кохонена входные векторы кодируются номерами нейронов-победителей (номерами кластеров). Таким образом, все векторы из некоторой области входного пространства заменяются одним и тем же опорным вектором, являющимся их ближайшим соседом. При использовании евклидова расстояния входное пространство разбивается на *многогранники* (мозаику) *Вороного* (Вороной Г. Ф.). Пример [3] многогранников Вороного для двумерного пространства приведен на рис. 2.

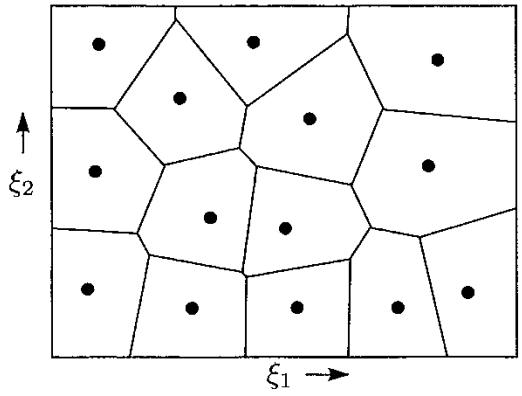

Рис. 2. Пример многогранников Вороного [8]

В многомерном пространстве мозаика Вороного образуется гиперплоскостями.

1.2. Обучение сети Кохонена

В сетях Кохонена используется обучение без учителя. Для обучения сети применяются *механизмы конкуренции*. При подаче на вход сети вектора $\mathbf{x}$ побеждает тот нейрон, вектор весов которого в наименьшей степени отличаются от входного вектора. Для нейрона-победителя выполняется соотношение

$$d\left(\mathbf{x},\mathbf{w}_j\right)=\min_{1\le i\le n} d\left(\mathbf{x},\mathbf{w}_i\right),$$

где n — количество нейронов, j — номер нейрона-победителя, $d\left(\mathbf{x},\mathbf{w}\right)$ — расстояние (в смысле выбранной метрики) между векторами $\mathbf{x}$ и $\mathbf{w}$.

Чаще всего в качестве меры расстояния используется е*вклидова мера*

$$d\left(\mathbf{x},\mathbf{w}_i\right)=\left\|\mathbf{x}-\mathbf{w}_i\right\|=\sqrt{\sum_{j=1}^{m}\left(x_j-w_{ji}\right)^2}\ .$$

Используются и другие меры расстояния (метрики).

Конкурирующая функция активации анализирует значения сумматоров и формирует выходы нейронов, равные 0 для всех нейронов, кроме одного "нейрона-победителя", имеющего на выходе максимальное значение. Таким образом, вектор выхода имеет единственный элемент, равный 1, который соответствует нейрону-победителю, а остальные равны 0. *Номер активного*

нейрона определяет ту группу (кластер), к которой наиболее близок входной вектор.

В сети Кохонена входные значения желательно (хотя и не обязательно) *нормировать*. Для этого следует воспользоваться одной из следующих формул:

$$x_{ні} = \frac{x_i}{\sqrt{\sum_{i=1}^{m} x_i^2}}, \quad x_{ні} = \frac{x_i}{|x_i|},$$

где $x_{ні}$ — нормированный компонент входного вектора.

Нормирование входных данных положительным образом сказывается на скорости обучения сети.

Перед процессом обучения производится *инициализация сети*, то есть первоначальное задание векторов весов. В простейшем случае задаются случайные значения весов. *Процесс обучения сети Кохонена* состоит из циклического повторения ряда шагов:

1. Подача исходных данных на входы. Обычно это случайная выборка одного из входных векторов.

2. Нахождение выхода каждого нейрона.

3. Определение "выигравшего" нейрона (веса которого в наименьшей степени отличаются от соответствующих компонентов входного вектора), или нейрона-победителя.

4. Корректировка весов "выигравшего" нейрона по *правилу Кохонена*

$$\mathbf{w}_i^{(k+1)} = \mathbf{w}_i^{(k)} + \eta_i^{(k)}\left[\mathbf{x} - \mathbf{w}_i^{(k)}\right], \qquad (2)$$

где $\mathbf{x}$ — входной вектор, k — номер цикла обучения, $\eta_i^{(k)}$ — коэффициент скорости обучения i-го нейрона в k-ом цикле обучения.

5. Переход на шаг 1, если обучение не завершено. Часто, например, в Neural Network Toolbox MATLAB, задается число циклов обучения. Можно проверять достижение малой величины функционала ошибки

$$E = \frac{1}{Q}\sum_{i=1}^{Q}\left\|\mathbf{x}_i - \mathbf{w}_{\mathbf{x}_i}\right\|^2, \qquad (3)$$

где $\mathbf{w}_{\mathbf{x}_i}$ — вектор весов нейрона-победителя при предъявлении входного вектора $\mathbf{x}_i$, Q — размер обучающей выборки.

Таким образом, нейрон, чей вектор весов был ближе к входному вектору, обновляется, чтобы быть еще ближе. В результате этот нейрон, скорее всего, выиграет конкуренцию при подаче на вход близкого вектора и проиграет при подаче существенно отличающегося вектора. После многократной подачи обучающих векторов будет иметься нейрон, который выдает 1, когда вектор принадлежит кластеру, и 0, когда вектор не принадлежит кластеру. Таким образом, сеть учится классифицировать входные векторы.

При обучении сети Кохонена возникает проблема так называемых *"мертвых" нейронов*. Одно из ограничений всякого конкурирующего слоя состоит в том, что некоторые нейроны оказываются незадействованными. Это проявляется в том, что нейроны, имеющие начальные весовые векторы, значительно удаленные от векторов входа, никогда не выигрывают конкуренции, независимо от того, как долго продолжается обучение. В результате оказывается, что такие векторы не используются при обучении и соответствующие нейроны никогда не оказываются победителями. Такие "нейроны-неудачники" называют *"мертвыми" нейронами,* поскольку они не выполняют никакой полезной функции. Таким образом, входные данные будут интерпретироваться меньшим числом нейронов. Поэтому надо дать шанс победить всем нейронам. Для этого алгоритм обучения модифицируют таким образом, чтобы "мертвые" нейроны участвовали в обучении.

Например [1], алгоритм обучения модифицируют таким образом, чтобы нейрон-победитель терял активность. Одним из приемов учета активности нейронов является подсчет *потенциала* p_i каждого нейрона в процессе обучения. Первоначально нейронам присваивается потенциал $p_i(0) = \frac{1}{n}$, где

n — число нейронов (кластеров). В k-ом цикле обучения потенциал определяется по правилам:

$$p_i(k) = \begin{cases} p_i(k-1) + \dfrac{1}{n}, & i \neq j, \\ p_i(k-1) - p_{\min}, & i = j, \end{cases}$$

где j — номер нейрона-победителя.

Если значение потенциала $p_i(k)$ падает ниже уровня $p_{\min}$, то нейрон исключается из рассмотрения — "отдыхает". При $p_{\min} = 0$ нейроны не исключаются из борьбы. При $p_{\min} = 1$ нейроны побеждают по очереди, так как в каждый цикл обучения только один из них готов к борьбе. На практике хороший результат получается при $p_{\min} \approx 0.75$.

В *Neural Network Toolbox* для борьбы с "мертвыми" нейронами используется изменение смещения нейронов [9]. Соответствующее правило настройки, учитывающее нечувствительность мертвых нейронов, реализовано в виде функции **`learncon`** и заключается в следующем. В начале процедуры настройки всем нейронам конкурирующего слоя присваивается одинаковый параметр активности $c_0 = \dfrac{1}{N}$, где N — количество нейронов конкурирующего слоя, равное числу кластеров. В процессе настройки функция **`learncon`** корректирует этот параметр таким образом, чтобы его значения для активных нейронов становились больше, а для неактивных нейронов меньше. Соответствующая формула для вектора параметров активности выглядит следующим образом:

$$\mathbf{c}^{(k+1)} = (1 - r_l)\mathbf{c}^{(k)} + r_l \mathbf{s}^{(k)},$$

где r_l — параметр скорости настройки; k — номер цикла обучения; $\mathbf{s}^{(k)}$ — вектор выходов адаптивных сумматоров в k-ом цикле обучения.

Компоненты вектора смещения вычисляется по формуле

$$b_i^{(k+1)} = e^{\left(1 - \ln c_i^{(k+1)}\right)} - b_i^{(k)}.$$

Смещение для нейрона-победителя уменьшится, а смещения для остальных нейронов немного увеличатся. Параметр скорости настройки r_l по умолчанию равен 0.001. Увеличение смещений для неактивных нейронов позволяет расширить диапазон покрытия входных значений, и неактивный нейрон начинает формировать кластер. В конечном счете, он может начать притягивать новые входные векторы. Это дает два преимущества. Если нейрон не выигрывает конкуренции потому, что его вектор весов существенно отличается от векторов, поступающих на вход сети, то его смещение по мере обучения становится достаточно большим, и он становится конкурентоспособным. Когда это происходит, его вектор весов начинает приближаться к некоторой группе векторов входа. Как только нейрон начинает побеждать, его смещение начинает уменьшаться. Таким образом, задача активизации "мертвых" нейронов оказывается решенной. Второе преимущество, связанное с настройкой смещений, состоит в том, что они позволяют выровнять значения параметра активности и обеспечить притяжение приблизительно одинакового количества векторов входа. Таким образом, если один из кластеров притягивает большее число векторов входа, чем другой, то более заполненная область притянет дополнительное количество нейронов и будет поделена на меньшие по размерам кластеры.